UNIVERSIDAD NACIONAL DE INGENIERIA

FACULTAD DE ELECTROTECNIA Y COMPUTACION

DISEÑO E IMPLEMENTACIÓN DE UN SISTEMA DE RECONOCIMIENTO DE PATRONES DE VOZ BASADO EN LOS MODELOS OCULTOS DE MARKOV UTILIZANDO LA PLATAFORMA DE PROGRAMACIÓN MATLAB

Autores:

- Br. Sabrina Jamileth Mendoza Treminio
- Br. Claudia Auxiliadora Méndez Torres

Tutor:
Ing. Fernando Flores

Managua, Agosto 2013

DEDICATORIA

Gracias ***Dios,*** por darme vida y salud para culminar mis estudios.

Dedico esta tesis a mis padres, ***Alejandro*** y ***Azucena*** que me han apoyado en cada etapa de mi vida, a mi hermana **Carmen** y mis hermanos **Alejandro** y **Miguel** a quienes siempre he tratado de dar el mejor ejemplo. A mi novio, **Edvin**, para que sigamos juntos y logremos todo las metas que nos hemos propuesto.

Y mis ***amigos***, por el cariño y el apoyo que me han brindado.

Sabrina Jamileth Mendoza Treminio

Gracias a Dios por las oportunidades presentadas en mi vida, entre ellas el estudio y culminación de esta carrera, una nueva etapa en el camino profesional. Gracias a mi familia por el apoyo incondicional y las palabras de aliento que nunca faltaron cuando alguna vez dije: “¡Ya no más!”.

Dedico esta tesis a todos quienes estuvieron a mi lado para darme apoyo, amor, cariño, comprensión y atención durante todo el trayecto y quienes se alegran conmigo por un triunfo alcanzado. A Sabrina porque sin ella esto no sería posible. A mi familia, amigos y mis pilares fundamentales… Papá, mamá y hermano. A mi querido Pilin (Q.E.P.D.), quien seguro desde el cielo celebra orgulloso uno de sus frutos.

Claudia Auxiliadora Méndez Torres.

RESUMEN

Esta tesis gira en torno a dos puntos principales: una investigación teórica acerca de la señal de voz y la implementación de un algoritmo de reconocimiento de palabras aisladas utilizando Modelos Ocultos de Markov. Para ello se realiza un análisis de las características de la voz, como señal de objeto de estudio. Se analiza el proceso de su creación, digitalización, codificación y parametrización para permitir al sistema identificar y comparar patrones característicos de cada palabra independientemente de quién sea su locutor.

CONTENIDO

LISTA DE FIGURAS

LISTA DE TABLAS

I. INTRODUCCIÓN

Con el rápido desarrollo de las tecnologías de comunicaciones, la interacción oral entre hombre y máquina ha tomado mucha importancia, debido a que es aplicable a una gran variedad de situaciones como: sistemas convertidores texto-voz, tareas multi-lingüísticas, telefonistas automáticas, sistemas de ayuda a discapacitados, entre otros.

En la actualidad, hay un gran número de exitosos productos comerciales con reconocimiento de habla integrado. Sin embargo, se desconocen los algoritmos implementados en estas aplicaciones, por lo que no pueden ser manipulados ni utilizados en otros proyectos. Para esto se presenta un sistema de reconocimiento de voz, realizado en MATLAB, basado en los modelos ocultos de Markov que permite reconocer palabras aisladas y es independiente del locutor. Este sistema está conformado por las siguientes etapas: adquisición, muestreo y cuantificación de la señal de voz, pre procesamiento, segmentación, parametrización, que permite extraer las características importantes de la palabra con menor cantidad de datos y HMM para el entrenamiento y reconocimiento de la palabra. Se creó una interfaz de usuario que permite grabar o cargar un archivo .WAV con la palabra que se desea reconocer. Además en la interfaz de usuario es posible agregar y entrenar nuevas palabras.

El sistema se verificó a través de una serie de experimentos, diseñados para comprobar que los parámetros seleccionados (técnica de parametrización, cantidad de secuencia del observación, cantidad de estados y tamaño del codebook de los HMM) a través de la investigación teórica fueran en efecto los que generan la mejor tasa de reconocimiento. Además se comprobó el funcionamiento del sistema al aumentar su base de datos (20 palabras) y agregar ruido.

Este proyecto tiene la finalidad de fortalecer la línea de investigación en procesamiento digital de voz, además de adquirir conocimientos en un área que no está incluida en el pensum de la carrera, complementando así la formación que tendremos como Ingenieros Electrónicos.

El documento se desarrolla en ocho capítulos:
Los primeros cinco capítulos comprenden la parte teórica, donde se explica la producción y percepción de la señal de voz, la adquisición de la señal, el pre-procesamiento, la segmentación, la parametrización, la cuantificación vectorial y los modelos ocultos de Markov para el reconocimiento de voz.

El sexto capítulo está concentrado el diseño del reconocedor de voz utilizando la plataforma de programación MATLAB.

Los últimos dos capítulos muestran los experimentos y resultados obtenidos para la verificación de la precisión del sistema de reconocimiento de voz bajo distintas condiciones.

II. OBJETIVOS

OBJETIVO GENERAL

- Implementar un sistema de reconocimiento de patrones de voz independiente del locutor, para palabras aisladas basado en los Modelos Ocultos de Markov utilizando la plataforma de programación MATLAB.

OBJETIVO ESPECIFICO

- Implementar algoritmo para la extracción de parámetro de voz basado en un modelo del oído interno humano.

- Implementar algoritmo de reconocimiento de palabras aisladas utilizando Modelos Ocultos de Markov.

- Diseñar una interfaz gráfica para sistema de reconocimiento de patrones de voz.

- Verificar la efectividad del sistema para distintos locutores.

III. JUSTIFICACIÓN

MOTIVACIÓN

El habla es el medio más espontaneó y natural de comunicación entre los hombres, sin embargo, hasta el presente se puede afirmar que en su comunicación con las máquinas el hombre ha hecho uso exclusivo del lenguaje escrito. Resulta natural, por tanto, extender la capacidad de comunicación hombre-máquina al mensaje oral. Además de la naturalidad y espontaneidad aludidas, la comunicación verbal hombre-máquina presenta importantes ventajas en gran cantidad de aplicaciones, como el diálogo interactivo o la entrada de grandes cantidades de datos en la máquina. Una de estas ventajas es que en la comunicación verbal las manos y la vista del usuario quedan liberadas, lo que permite dedicarse a una tarea simultánea a la comunicación. Ello ofrece posibilidades muy interesantes en el gobierno de sistemas de gran complejidad en los que la atención visual sea muy importante.

En lo que a su aplicación práctica se refiere, el reconocimiento del habla empezó como un reto científico de emular el comportamiento humano con máquinas, siendo objeto de interés para un público y unas aplicaciones bastante específicas y limitadas [1]. La situación actual no podría ser más contraria, esta tecnología ha sido denominada tercera revolución industrial.

Los avances tecnológicos previos y los originados por el nacimiento de la Sociedad de las Tecnologías de la Información y las Comunicaciones (TICs), han provocado una revolución en la demanda de interfaces usuario máquina lo más amigables y transparentes posible para el usuario [2]. El dialogo hombre-máquina aparece en este escenario como mecanismo óptimo y natural desde varios puntos de vista:

- Desde un punto de vista científico-tecnológico, la inteligencia artificial y, en particular, el reto de emular la comunicación oral humana sigue siendo un estimulo atractivo [3].

- Desde un punto de vista económico, hay que señalar tres factores:
  1. La tecnología se ha convertido en un bien de consumo.

  2. La demanda de reconocimiento del habla crece de manera incesante en los sistemas de telecomunicación, en los sistemas de control y en los sistemas de entrada de datos y de acceso a bases de datos [2].

  3. Las personas de edad avanzada en las poblaciones del primer mundo y los inmigrantes de países con menor penetración de las TICs, son dos sectores del mercado con un alto potencial como consumidores. Estos dos sectores demandan interfaces universales, intuitivas y amigables.

El análisis anterior enmarca la motivación de este trabajo científico basado en el reconocimiento de voz. El reconocedor debe brindar las máximas prestaciones posibles en las condiciones más adversas imaginables. El robustecimiento de un reconocedor de voz se puede definir como la aportación de mecanismos que lo hagan menos vulnerable a los desajustes entre condiciones de entrenamiento y evaluación [4]. Además los sistemas de reconocimiento diseñados específicamente para un locutor ya no tienen sentido, ya que una multitud de usuarios con rasgos fonéticos y dialectales muy variados pueden acceder a las aplicaciones de estos sistemas.

Esta tesis aplica todos los aspectos anteriormente comentados para diseñar un reconocedor de voz, que nos permita alta flexibilidad con una estructura sencilla y, a la vez, robustez para que la variabilidad del locutor se pueda compensar.

IV. MARCO TEÓRICO

La sección teórica pretende dar una base esencial acerca del análisis de la señal de voz involucrado en las tareas de reconocimiento, con el fin de entender los principios básicos en los cuales se centran los procedimientos e implementaciones de esta tesis.

Esta sección está dividida en cinco capítulos. En el primero se describen las características de las señales de voz; el segundo trata de la adquisición, pre-procesamiento y segmentación de la señal, donde eliminan componentes no deseados, se resaltan y mejoran sustancialmente las características antes de la parametrización. En el tercer capítulo se presentan las técnicas de extracción de parámetros. En el capítulo cuatro se define la cuantificación vectorial; y el quinto se refiere a los modelos ocultos de Markov para reconocimiento de voz.

CAPITULO 1: CARACTERÍSTICAS DE LAS SEÑALES DE VOZ

1.1 LA SEÑAL DE VOZ

Una breve introducción a cómo la señal de voz es producida y percibida por el cuerpo humano es un buen punto de partida para entrar en el campo del reconocimiento del habla.

El proceso de producción y percepción del habla humana, entre el hablante y el oyente, se muestra en la figura 1.

Figura 1. Comunicación de voz humana **[1]**

El primer elemento está asociado con la formulación de la señal de voz en la mente del hablante. Esta formulación es utilizada por el mecanismo vocal humano para producir la forma de onda, la cual es trasmitida al oyente a través del aire. Durante la transferencia la señal acústica puede ser afectada por fuentes externas, por ejemplo ruido, resultando en una señal de onda más compleja. Cuando la señal alcanza el sistema de oído del receptor, este percibe la señal y su mente empieza el procesamiento del contenido de la forma de onda, de modo que el receptor entienda la información que el hablante le está transmitiendo. El sistema de reconocimiento se encarga de simular como el receptor procesa las señales de voz producida por el locutor.

1.2 PRODUCCIÓN DEL HABLA

Para diseñar un sistema de reconocimiento de voz que sea independiente del locutor, es fundamental conocer y determinar los mecanismos que han producido el mensaje hablado. Es por ello que se van a presentar algunos conceptos fundamentales y básicos del mecanismo de producción del habla, tanto en el órgano físico como la producción propia del mensaje.

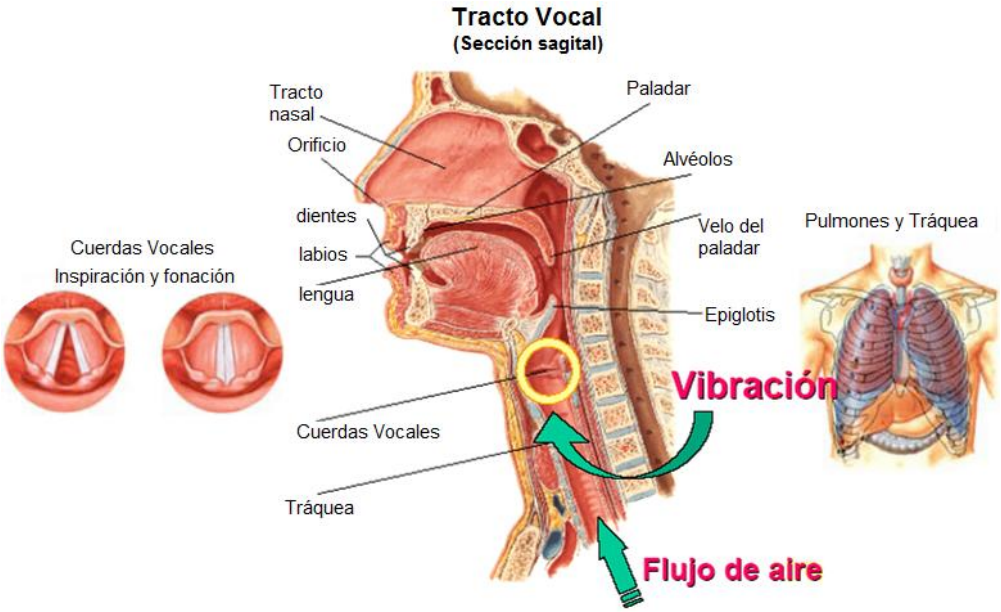

Figura 2. Sección del tracto vocal **[4]**

El conjunto de órganos que intervienen en la fonación[1] puede dividirse en tres grupos bien delimitados [5]:

1. Cavidades **infraglóticas** (sistema sub-Glotal) u **órgano respiratorio**, compuesta por pulmones, bronquios y tráquea.
2. Cavidad **laríngea** u **órgano fonador**, compuesta por cavidades glóticas (laringe, cuerdas vocales) y resonadores (nasal, bucal y faríngeo).
3. Cavidades **supraglóticas**, compuesta por la faringe, la cavidad bucal y la cavidad nasal.

El aparato fonador es controlado por el sistema nervioso central, específicamente por el área Broca, situada en el hemisferio izquierdo de la corteza cerebral.

Para producir sonidos, el aire expulsado de los pulmones (ver figura 2) debe generar una vibración en la laringe. La laringe está formada por un conjunto de cartílagos y una serie de ligamentos y membranas que sostienen a las cuerdas vocales [2]. La tensión, elasticidad, altura, anchura, longitud y grosor de las cuerdas vocales pueden variar, lo que da lugar a diferentes efectos sonoros.

Los sonidos se pueden presentar en tres estados:

- **Silencio**: Ningún sonido es producido.

- **No sonoros**: Las cuerdas vocales no vibran, resultando en una forma de onda aleatoria no periódica.

- **Sonoros**. Son aquellos sonidos que hacen vibrar las cuerdas vocales. Esta vibración es cuasiperiódica[2] y su espectro es muy rico en armónicos, que son múltiplos de la frecuencia de vibración de las cuerdas. A esta frecuencia de

[1] Trabajo muscular realizado para emitir sonidos inteligibles.
[2] La forma de onda cuasiperiódica significa que la señal de voz puede analizarse como una señal periódica en un tiempo corto (5-100ms)

vibración de las cuerdas se le llama frecuencia fundamental. La frecuencia fundamental depende de la presión ejercida al pasar el aire por las cuerdas y de la tensión de estas. En un hombre la frecuencia fundamental se encuentra en el rango 50-250 Hz, mientras en la mujer el rango es más amplio, encontrándose entre 100 y 500 Hz [6].

1.3 MODELO FUENTE – FILTRO DE LA PRODUCCIÓN DEL HABLA

El tracto vocal es modelado como una concatenación de tubos acústicos de distinto diámetro, con o sin pérdidas. Lo cual resulta en un modelo lineal inestacionario, ya que las secciones de los tubos van cambiando de acuerdo al fonema que se está emitiendo [7]. Se puede decir entonces que, el tracto vocal actúa como una cavidad resonante formando regiones donde el sonido producido es filtrado. La figura 3 muestra lo anterior.

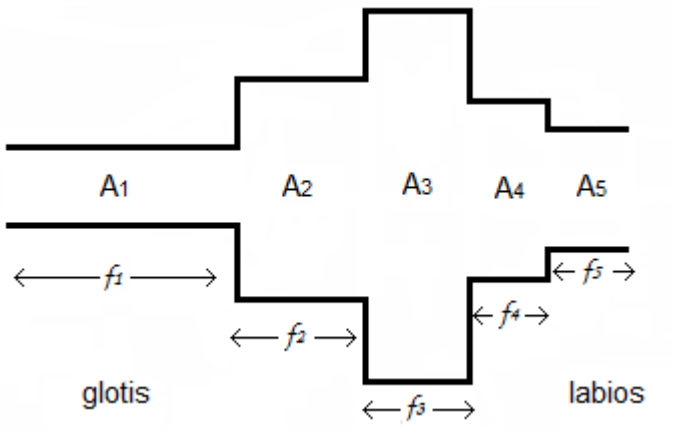

Figura 3. Modelado acústico del tracto vocal. **[6]**

Este sistema de resonancia se puede ver como un filtro que da forma al espectro de la fuente de sonido para producir la señal de voz y es modelado a través del modelo fuente – filtro. [1]

El modelo fuente-filtro consiste en una señal de excitación que modela la fuente de sonido (e[n]) pasándola a través de un filtro de polos únicamente (h[n]); para producir señal de voz (s[n]) [8].

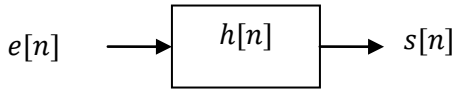

Figura 4. Modelo básico fuente – filtro de la producción del habla **[6]**

Para los sonidos sonoros, la señal de excitación es un tren de impulsos convolucionados con el pulso glotal[3] (ver Figura 5), mientras que para sonidos sordos, la señal de excitación es el ruido aleatorio. Ambos tiene un factor de ganancia G de forma que se pueda controlar la intensidad de la excitación.

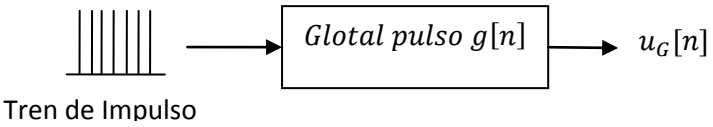

Figure 5. Modelo de excitación glotal para sonido sonoro. **[6]**

Para un modelo completo de fuente-filtro, como se muestra en la Figura 6, el pulso glotal, el tracto vocal y la radiación deben ser modelados individualmente como filtros lineales [6]. La función de transferencia,$V(z)$, representa las resonancias del tracto vocal, y la función de transferencia,$R(z)$, a los modelos de la presión de aire en los labios.

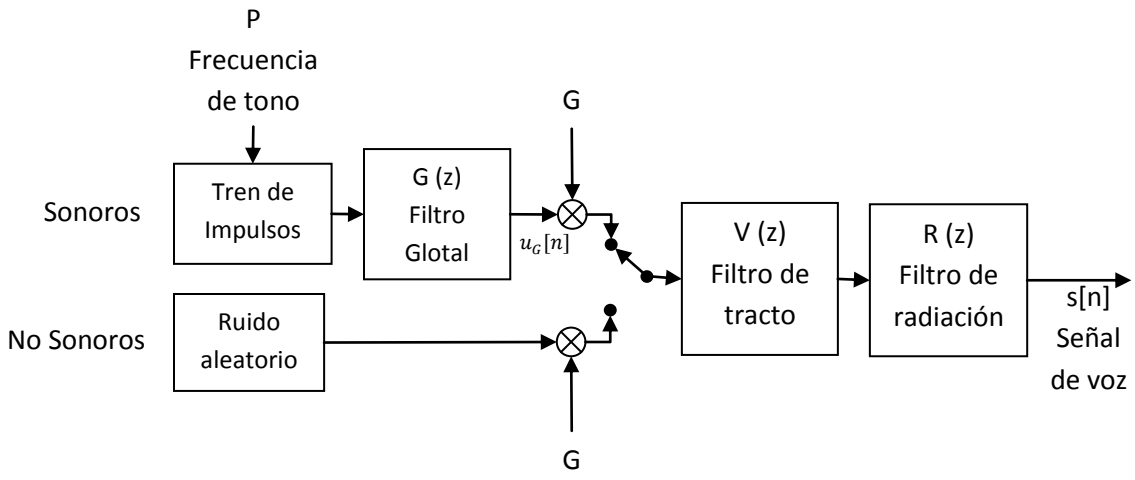

Figura 6. Modelo general en tiempo discreto de la producción del habla **[9]**

[3] Término utilizado en el estudio de la lingüística para describir las variaciones en la calidad de la voz por la manipulación de los pliegues de las cuerdas vocales.

Al combinar de $G(z), V(z)\ y\ R(z)$, como el único filtro de polos, $H(z)$, se obtiene, un nuevo diagrama simple que se muestra en la Figura 7.

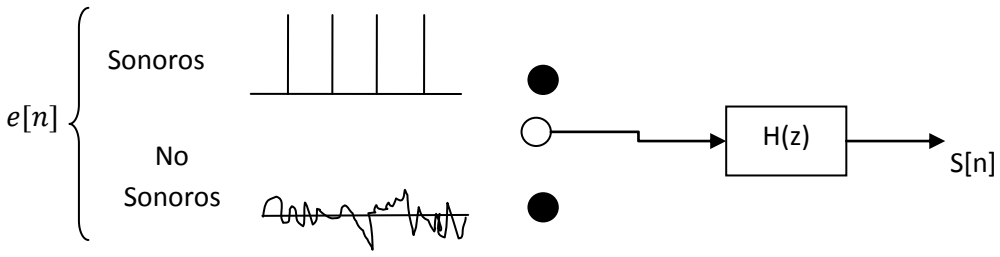

Figure 7. Modelo fuente-filtro para la producción del habla

El tracto vocal, por poseer una característica de cavidad resonante, tiene un número muy elevado de resonancias, de las cuales se consideran solo las tres o cuatro primeras (formantes). Estas cubren un rango de frecuencias entre 100 y 3500 Hz, ya que las resonancias de alta frecuencia son atenuadas por la característica del tracto que actúa como un filtro pasa-bajo con una caída de aproximadamente -12dB por octava. [10]

1.4 PERCEPCIÓN DEL HABLA

La capacidad de comprender el lenguaje oral se deriva del funcionamiento de un conjunto muy complejo de procesos perceptivos, cognitivos y lingüísticos que permiten al oyente recuperar el significado de un enunciado cuando lo oye.

El mecanismo físico de la percepción del habla, al igual que la audición, se realiza por medio de dos órganos fundamentales, el sistema auditivo periférico y el sistema nervioso central auditivo.

El Sistema auditivo periférico es lo que vulgarmente se llama oído. En la figura 8 pueden observarse las 4 partes en las que se divide el sistema auditivo: oído externo, oído medio, oído interno y el sistema nervioso central auditivo.

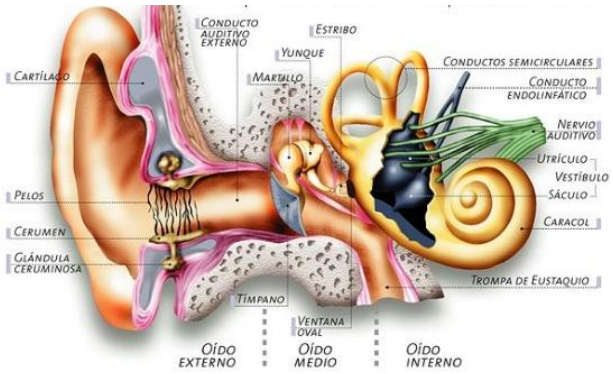

Figura 8. Estructura del sistema periférico auditivo. **[4]**

Los modos de funcionamiento son los siguientes [9]:

- **Oído externo**: funciona por vibración del aire. Consiste de la parte externa visible y el canal auditivo externo. Mide aproximadamente 2.5 cm. Su función es canalizar la energía acústica.

- **Oído medio**: funciona por movimiento mecánico de los huesecillos. Transforma la energía acústica en energía mecánica, transmitiéndola hasta el oído interno.
- **Oído interno**: El funcionamiento se divide en tres, primero es mecánico, por el movimiento del estribo, luego hidrodinámico por el movimiento de los líquidos interiores a la cóclea y finalmente electroquímico. Aquí se realiza la definitiva transformación de la energía mecánica en impulsos eléctricos.

- **Sistema nervioso central auditivo**: El funcionamiento es electroquímico, el movimiento de las células ciliadas provocan una reacción química que a su vez genera un impulso eléctrico.

Cuando el sonido llega al oído, las ondas sonoras son recogidas por el pabellón auricular (o aurícula).Una vez recogido el sonido, las vibraciones provocadas por la variación de presión del aire cruzan el canal auditivo externo y llegan a la membrana

del tímpano. El conducto auditivo actúa como una etapa de potencia natural que amplifica automáticamente los sonidos más bajos que proceden del exterior. En el oído medio, se produce la transformación de la energía acústica en energía mecánica.

La presión de las ondas sonoras hace que el tímpano vibre empujando a los ósculos que, a su vez, transmiten el movimiento del tímpano al oído interno. Cada osiculo empuja a su adyacente y finalmente a través de la ventana oval [2]. Esta presión ejercida sobre la ventana oval, penetra en el interior de la cóclea, la cual se comunica directamente con el nervio auditivo, conduciendo una representación del sonido al cerebro. La cóclea es un tubo en forma de espiral (de 3.5 cm aproximadamente) y está dividida longitudinalmente por la membrana basilar en dos cámaras que contienen líquido linfático [9].

La cóclea puede ser aproximada como un banco de filtros. Los filtros correspondientes al extremo más próximo a la ventana oval y al tímpano responden a las altas frecuencias, ya que la membrana es rígida y ligera. Por el contrario, en el extremo más distante, la membrana basilar es pesada y suave, por lo que los filtros correspondientes responden a las bajas frecuencias. [8]

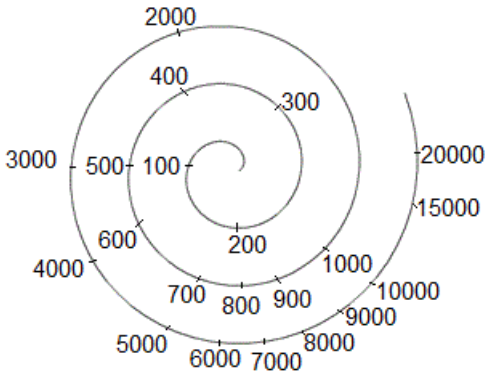

Figura 9. Distribución de frecuencia en la cóclea **[9]**.

Estos filtros se producen a lo largo de la membrana basilar y tienen como función aumentar la resolución de frecuencia de la cóclea y así incrementar la habilidad de discriminar entre distintos sonidos [11]. Este banco de filtros no sigue una

configuración lineal, y el ancho de banda y morfología de cada filtro depende de su frecuencia central.

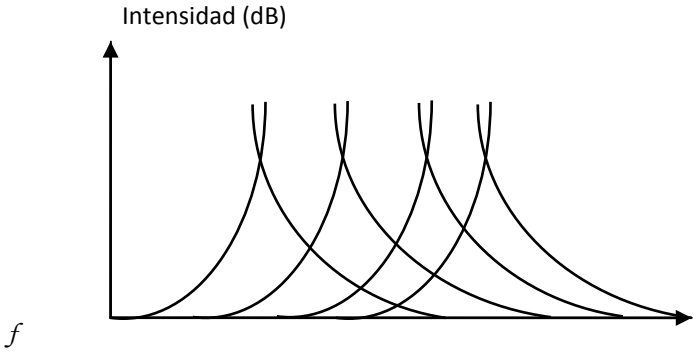

Figura 10. Esquema de las bandas criticas del sistema auditivo humano.

El ancho de banda de cada filtro auditivo se denomina banda crítica [9]. Las bandas críticas, esquematizadas en la figura 10, son rangos de frecuencia dentro de los cuales un sonido bloquea o enmascara la percepción de otro sonido. Las bandas críticas conceptualmente están ligadas a lo que sucede en la membrana basilar, ya que una onda que estimula la membrana basilar perturba la membrana dentro de una pequeña área más allá del punto de primer contacto, excitando a los nervios de toda el área vecina. Por lo tanto, las frecuencias cercanas a la frecuencia original tienen mucho efecto sobre la sensación de intensidad del sonido. Es importante destacar que el concepto de banda crítica es una construcción teórica y no algo físicamente comprobado.

Una inquietud que surge de inmediato es preguntarse cuantas bandas críticas existen en el sistema auditivo y cuál es la frecuencia central de cada una. Con objeto de aproximarse a la sensibilidad del oído humano, que no tiene una respuesta lineal, existen dos escalas que sobresalen Bark y Mel.

ESCALA BARK

Consta de una escala de medición de las bandas críticas que se detalla en la tabla 1. La escala tiene un rango del 1 al 24 y corresponde a las primeras veinticuatro bandas críticas del sistema auditivo [11].

Banda Critica	Frec. Central	Ancho de banda	Frec. Min.	Frec. Máx.
(Bark)	(Hertz)	(Hertz)	(Hertz)	(Hertz)
1	50	-	-	100
2	150	100	100	200
3	250	100	200	300
4	350	100	300	400
5	450	110	400	510
6	570	120	510	630
7	700	140	630	770
8	840	150	770	920
9	1000	160	920	1080
10	1170	190	1080	1270
11	1370	210	1270	1480
12	1600	240	1480	1720
13	1850	280	1720	2000
14	2150	320	2000	2320
15	2500	380	2320	2700
16	2900	450	2700	3150
17	3400	550	3150	3700
18	4000	700	3700	4400
19	4800	900	4400	5300
20	5800	1100	5300	6400
21	7000	1300	6400	7700
22	8500	1800	7700	9500
23	10500	2500	9500	12000
24	13500	3500	12000	15500

Tabla 1. Escala de Bark para estimación de las bandas criticas del sistema auditivo **[9]**.

Las bandas críticas del oído son continuas, y un tono de cualquier frecuencia audible siempre encuentra una banda crítica que incluya dicha frecuencia [9]. La frecuencia Bark, *Zc,* puede ser expresada en términos de la frecuencia (en KHz)

(1.1)

como:

$$Z_c = 13tan^{-1}(0.76f) + 3.5tan^{-1}\left(\frac{f}{75}\right)^2$$

Donde Z_c es la frecuencia en escala Bark y f es la frecuencia en Hz.

ESCALA MEL

La escala Mel es una escala perceptual de frecuencias que se utiliza para representar de manera más fidedigna la percepción de la frecuencia de un sonido en el oído humano, y fue propuesta por Stevens, Volkman y Newmann en 1937. El nombre Mel se deriva de melodía, para dejar en evidencia que se trata de una escala basada en comparaciones entre frecuencias.

La escala de Mel es lineal para frecuencias inferiores a 1kHz, y logarítmica para frecuencias superiores, posee el mismo número de muestras tanto para frecuencia menores como superiores a 1kHz (figura 11) [5].

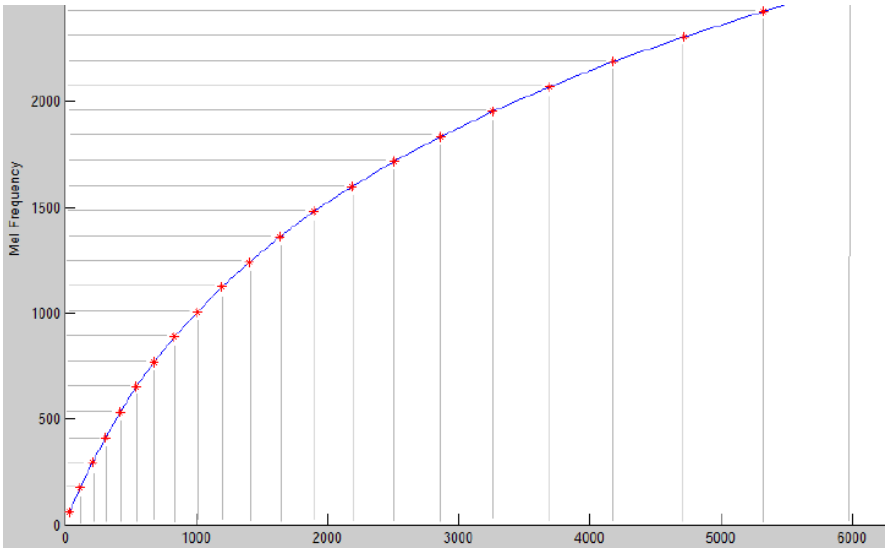

Figura 11. Representación de escala Mel.

La escala Mel ha sido ampliamente utilizada en modernos sistemas de reconocimiento de habla y puede ser aproximada en función de la frecuencia lineal como:

$$\dot{f} = 2594\, log_{10}\left(1 + \frac{f}{700}\right) \qquad (1.2)$$

Donde f representa la frecuencia en escala lineal y $\acute{f}$ la frecuencia en escala Mel. Una Mel se define como la milésima parte de la afinación de un tono de 1 Khz. [8]

CAPITULO 2. ADQUISICIÓN, PRE-PROCESAMIENTO Y SEGMENTACIÓN DE LA SEÑAL DE VOZ.

2.1 ADQUISICIÓN

El sistema de reconocimiento de voz, diseñado en este proyecto de fin de carrera, tiene dos modalidades para adquirir la señal voz, la primera a través de archivos con extensión .WAV y la segunda a través de una grabación.

2.1.1 ARCHIVO

Los archivos .WAV que se utilizan para entrenar y comprobar el funcionamiento del sistema, fueron extraídos de la base de datos MAMI (Database of isolated spoken words through a mobile). La cual contiene grabaciones de 23 personas (14 varones y 9 mujeres, con edades entre 24 y 42 años de edad) que repiten 5 veces cada una de las 47 palabras [12]. Todos los usuarios utilizaron para grabar un teléfono HTC Touch (TM) con plataforma Windows Mobile 6.0. Para iniciar o detener la grabación el usuario necesita presionar la pantalla. La frecuencia de muestreo para estas grabaciones fue de 11025 Hz y 16 bits por muestra.

2.1.2 GRABACIÓN

La grabación consiste en captar mediante un micrófono la onda acústica producida por el locutor y digitalizarla para poder tratarla en la computadora. Es importante destacar que tanto el micrófono usado como el lugar en el que se realiza la grabación pueden afectar las tasas de reconocimiento. Especialmente si no son iguales a los utilizados en el proceso de entrenamiento (etapa de aprendizaje del sistema) y de reconocimiento (etapa de funcionamiento del sistema).

2.1.2.1 MUESTREO

La herramienta de adquisición de datos de MATLAB (MATLAB Data Adquisition Tool Box versión 2.7) está orientada a utilizar el hardware de adquisición de datos para medir y analizar fenómenos físicos.

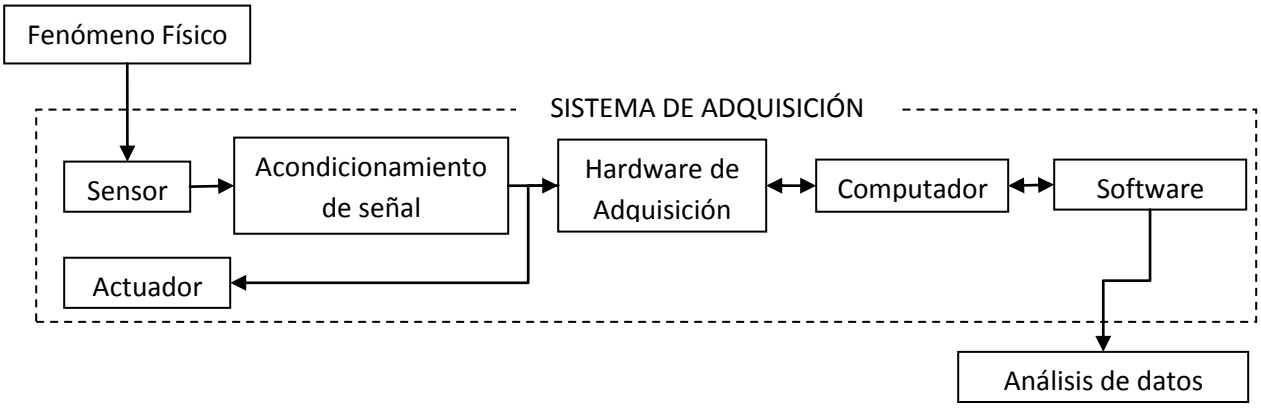

Figura 12. Sistema de adquisición de datos. [13]

El sistema de adquisición de datos es una colección de objetos de software y hardware que permite conectar la computadora con el mundo físico [14]. Según el hardware conectado, sea interno o expandido, se tienen dentro del DaqToolbox cuatro subsistemas:

- Entrada análoga. (Analog Imput Subsystem)
- Salida análoga
- Entrada y salida digital
- Timer y contadores

El subsistema de entrada análoga tiene como función muestrear y cuantificar la señal análoga empleando uno o más canales, la señal análoga es continua en tiempo y amplitud [15], el muestreo toma pequeñas muestras de la señal en tiempos discretos, mientras que la cuantificación divide los valores de voltaje en amplitudes discretas.

2.1.2.2 MUESTREO CON EL DAQTOOLBOX 2.5 ANALOG INPUT

El sistema toma muestras del sensor, micrófono, a intervalos constantes. En la mayoría de los conversores Análogo/Digital, que se encuentran dentro de las tarjetas de sonido, el muestreo se realiza mediante un circuito de muestreo y retención (Sampling and Hold) S/H [2]. Un S/H consiste de un buffer para la señal

seguido de un selector electrónico conectado a un capacitor. La operación sigue los siguientes pasos:

En un instante de muestreo dado, el selector conecta el buffer y el capacitor a una entrada. El capacitor se carga al voltaje de la entrada. La carga se mantiene hasta que el conversor A/D digitaliza la señal. El proceso completo se repite para el siguiente instante de muestreo.

Muy pocas tarjetas de sonido poseen conversores para cada canal y la mayoría multiplexan las entradas a un solo conversor (Figura 20).

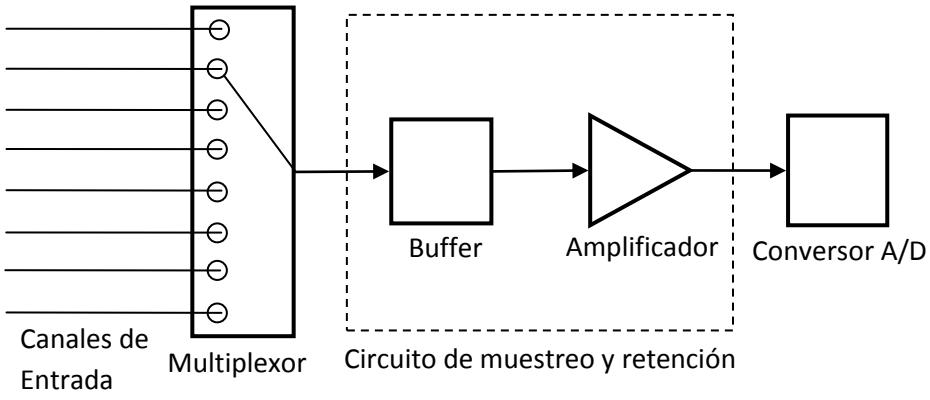

Figura 13. Sistema multiplexor de conversor análogo/digital

Las características del objeto Analog Input empleado se muestran en la sección V (Implementación de sistema de reconocimiento de voz)

2.1.5 CUANTIFICACIÓN CON EL DAQTOOLBOX 2.5 ANALOG INPUT

Cuando se realiza la toma, la señal análoga muestreada debe ser convertida a un valor binario que el computador pueda leer.

Durante la cuantificación, el conversor A/D emplea un número finito de valores convenientemente espaciados para representar la señal análoga [15]. El número de diferentes valores es determinado por el número de bits empleados para la conversión. Por defecto el sistema de adquisición realiza conversión a 16 bit [13].

En el Anexo se muestra en “grabar” el código de programa de adquisición, muestreo y cuantificación.

2.2 PRE-PROCESAMIENTO

La etapa de pre-procesamiento de la señal de voz corresponde a los pasos necesarios para remover componentes no deseados de la señal que nos permitan analizarla con la técnica de parametrización elegida para tal fin.

La figura 12 muestra el diagrama de bloques del pre-procesamiento aplicado a la señal de voz.

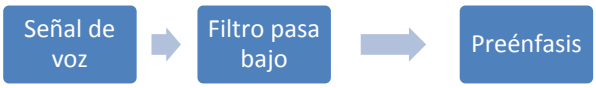

Figura **14**. Pre-procesamiento de la señal de voz

Las siguientes fases: pre-procesamiento (filtro pasa-bajo y preénfasis) y segmentación (análisis de energía, detección de punto inicial y final y aplicación de ventana), se describen a continuación.

2.2.1 FILTRO PASA-BAJOS

El ancho de banda útil de la voz es de 8KHz, pero la información relevante está contenida hasta los 4 KHz. aproximadamente. Se utiliza un filtro pasa-bajos, con el fin de eliminar la información en frecuencias superiores a los 4 Khz., con lo cual se ahorra tiempo de procesamiento, y se evita la producción de aliasing[4]. De esta manera, se impide que las réplicas del espectro de la señal, que se producen como consecuencia del muestreo, se solapen ocasionando distorsión.

[4] Efecto que causa que señales continuas distintas se tornen indistinguibles cuando se muestrean digitalmente.

2.2.2 PREÉNFASIS

Se realiza la etapa de preénfasis, con el fin de hacer la señal menos susceptible a truncamientos, aplanarla espectralmente y compensar la caida de 6 dB que experimenta la señal al pasar a traves del tracto vocal. [7]

Se utiliza un filtro digital de primer orden cuya funcion de transferencia se muestra en la ecuacion:

$$H(z) = 1 - a \cdot z^{-1} \qquad (2.1)$$

Donde $0.9 \leq a \leq 0.95$. Este valor se escoge cercano a la unidad a fin de que la estructura de los formantes mayores sean acentuadas. Se abordaran mas detalles sobre su impletacion en sección cinco (Implementación de sistema de reconocimiento de voz).

2.3 SEGMENTACIÓN

La etapa de segmentación de la señal de voz corresponde a los pasos necesarios para cortar la señal en segmentos de análisis. La señal de voz es asumida como estacionaria en estos segmentos. Realizamos un análisis de energía para determinar si la señal adquirida contiene tramas de sonidos, en caso de tenerlos calculamos el punto inicial y final de la palabra.

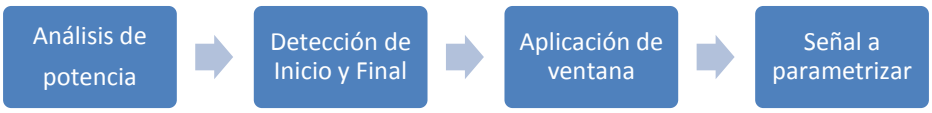

Figura 15. Segmentación de la señal de voz

2.3.1 ANÁLISIS DE ENERGÍA

La variación de energía en la señal de voz se debe a la variación de la presión subglotal y de la forma del tracto vocal [16]. La energía media es útil para distinguir sonidos sordos y sonoros en la señal de voz, debido a que los valores de esta

(2.2)

característica aumentan en los sonidos sonoros respecto a los sordos. Para calcular la energía media se utiliza la fórmula que se muestra a continuación.

$$\boldsymbol{E} = \frac{1}{N}\sum_{\boldsymbol{n=0}}^{\boldsymbol{N-1}} |\boldsymbol{x}[\boldsymbol{n}]|^{\boldsymbol{2}}$$

Donde: *x* representa a la señal de voz y N el número de muestras que forman la señal. Se abordaran mas detalles sobre su impletacion en sección cinco (Implementación de sistema de reconocimiento de voz).

2.3.2 DETECCIÓN DE INICIO Y FINAL

En el capítulo 1.3 del presente documento, se mencionó que un fragmento de voz se considera vocalizado si durante su generación el flujo de aire es alterado por las vibraciones de las cuerdas vocales del locutor. Dicho proceso aporta a la señal resultante características que permiten diferenciarla de las secciones no vocalizadas de voz. En particular, las componentes vocalizadas presentan un comportamiento periódico y transporta una mayor cantidad de energía que los sonidos no vocalizados.

Determinar con exactitud el inicio y el final de los segmentos vocalizados ayuda a disminuir la cantidad de información y por lo tanto el tiempo de procesamiento. Utilizar un solo criterio de decisión para determinar si un segmento de voz es vocalizado o no vocalizado no es suficiente, por ello este trabajo utiliza tres criterios de decisión basados en el análisis de magnitud, cruce por cero y tono. [16]

2.3.2.1 DETECCIÓN DE VOZ POR SUMA DE MAGNITUD

La magnitud es un parámetro cuyo comportamiento es muy similar al de la energía. La principal diferencia entre ambos parámetros radica en que la magnitud no disminuye tanto como la energía en los tramos sordos [15]. Computacionalmente hablando, el cálculo de la magnitud es más sencillo de realizar que el de la energía.

El cálculo de la magnitud se describe por la ecuación:

$$\boldsymbol{M[m] = \sum_{n=0}^{N-1} |x[n]|} \qquad (2.3)$$

Donde: *x* representa a la señal de voz, *w* representa a la ventana de análisis, *N* es el tamaño de la ventana.

2.3.2.2 DETECCIÓN DE VOZ POR CRUCE POR CERO

La tasa de cruces por cero (TCC) mide, tal y como indica su nombre, las veces que la señal de voz pasa por el nivel cero durante el segmento bajo análisis. Esta medida nos proporciona una idea general de la distribución en frecuencia de la señal. Una TCC elevada indica que el segmento de voz tiene un contenido espectral en alta frecuencia importante, mientras que una tasa baja implica que casi toda la señal está en baja frecuencia. Un segmento sonoro posee un espectro centrado en baja frecuencia y uno sordo tiene una componente en alta frecuencia superior. [16]

$$TCC[m] = \frac{1}{N}\sum_{n}\frac{1}{2}|sgn(x[n]) - sgn(x[n-1])|w(m-n) \qquad (2.4)$$

Donde: *x* representa a la señal de voz, *w* representa a la ventana de análisis, *N* es el tamaño de la ventana y sgn() es la función signo la cual viene definida por:

$$sgn(x) = \begin{cases} +1 & x \geq 0 \\ -1 & x < 0 \end{cases} \qquad (2.5)$$

2.3.2.3 DETECCIÓN DE TONO (PITCH) DEL SEGMENTO DE AUDIO.

La autocorrelación es una medida del parecido de una señal consigo misma y se emplea para determinar el tono (*pitch*) de la voz [17]. El *pitch*, en tratamiento de voz,

es la frecuencia fundamental a la cual vibran las cuerdas vocales, esta vibración produce una repetición en el patrón temporal de la señal y esta repetición se ve reflejada en forma de un máximo local en la función de autocorrelación [16]. En ausencia de vibración de las cuerdas vocales, esto es cuando el segmento es sordo, la autocorrelación no indica nada.

El cálculo de la autocorrelación en el dominio temporal se realiza por medio de la siguiente expresión:

$$R_m[k] = \sum_{n=0}^{N-1} \{w[m-n]x[n]\}\{w[m-(n+k)]x[n+k]\} \qquad (2.6)$$

$$k = 0,1,2,\dots,p$$

Donde: *x* representa a la señal de voz, *w* representa al perfil de la ventana de análisis, *N* es el tamaño de la ventana y p es el número de elementos a calcular de la autocorrelación. Se abordaran mas detalles sobre su impletacion en sección cinco (Implementación de sistema de reconocimiento de voz).

2.3.3 APLICACIÓN DE VENTANA

El enventanado es un proceso donde se obtienen tramas o segmentos consecutivos de señal, se utiliza ya que la señal de voz varía lentamente en periodos entre 5 y 100ms.

En general, la aplicación de ventanas conduce al alisamiento y, por tanto, distorsión del espectro original. Para mantener la distorsión espectral dentro de un mínimo, se requiere el cumplimiento de las siguientes propiedades por parte de las ventanas:

- Lóbulo principal reducido, lo que requiere una ventana amplia en el tiempo.
- Pequeños o insignificantes lóbulos laterales que requiere una ventana de tiempo alisada sin cortes abruptos o flancos [18].

Estos dos requerimientos no pueden alcanzarse simultáneamente, por lo cual en la determinación de la ventana más adecuada debe realizarse un compromiso.

La ventana Hamming (figura 17) cumple con el requerimiento de que la atenuación de los lóbulos secundarios sea brusca lo que permite obtener una mayor resolución espectral [1]. La ventana de Hamming se define matemáticamente por la ecuación:

$$w(n) = 0.54 - 0.46\cos\left(\frac{2\pi n}{N-1}\right) \qquad (2.7)$$

Donde, n es un número natural que determina el tamaño de la ventana.

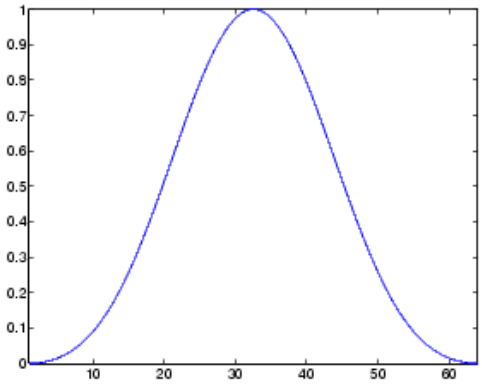

Figura 16. Ventana de Hamming [18]

CAPITULO 3: TÉCNICAS DE PARAMETRIZACIÓN DE LA SEÑAL DE VOZ.

Este paso es de gran importancia, consiste en extraer la información más relevante de los bloques de la señal. Una variedad de opciones se pueden aplicar para esta tarea, sin embargo los dos métodos más utilizados son el predictor lineal y mel-cepstrum [7]. Las razones por las que estos métodos han sido ampliamente utilizados se muestran a continuación [7] [10]:

- Proveen un buen modelo para la señal de voz.
- Los procedimientos que utilizan ambos métodos lidera a una razonable separación fuente-tracto vocal. Lo cual significa que proveen una muy buena representación de las características del tracto vocal.

- Tienen un modelo manejable analíticamente.
- La experiencia ha determinado que estos métodos funcionan muy bien en aplicaciones de reconocimiento.

3.1 PREDICTOR LINEAL

El método de predicción lineal toma como base un modelo del tracto vocal representado como un filtro lineal variable en el tiempo [17]. Según este modelo se distinguen dos elementos separados en la producción de voz: la excitación y el tracto vocal. La onda de voz es resultado de la convolución entre la excitación y el filtro (tracto vocal). [15]

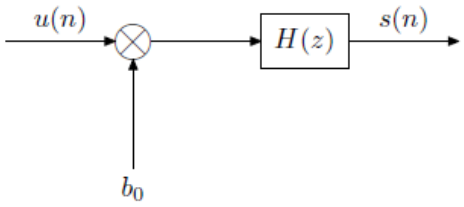

Figura 17. Modelo del tracto vocal.

Este método caracteriza la forma del espectro de un segmento de voz con un número reducido de parámetros que permiten una codificación eficiente. La codificación Lineal Predictiva, como también se llama a este método, predice una señal en el dominio del tiempo con base en una combinación de muestras previas, linealmente distribuidas como se muestra en la ecuación [19].

$$s(n) = b_o u(n) + a_1 s(n-1) + a_2 s(n-2) + \cdots + a_p s(n-p) \qquad (3.1)$$

O equivalente:

$$s(n) = b_o u(n) + \sum_{i=1}^{p} a_i s(n-i) \qquad (3.2)$$

Donde $u(n)$ es la señal de excitación normalizada, b_o es la ganancia de la señal de excitación y los coeficientes $a_1, a_2, \ldots, a_p$ son un conjunto de constantes reales

conocidas como coeficientes predictores, que necesitan ser calculados y p es el orden del predictor. Una de las ventajas de este método es que además de ser muy preciso, es muy adecuado para la implementación computacional, pues es sencillo y de rápida ejecución.

El error entre el valor real de la función y la función aproximada está dado por la ecuación:

$$e_n = s_n - \grave{s_n} = s_n - \left[b_o u(n) + \sum_{i=1}^{p} a_i s(n-i) \right] \tag{3.3}$$

El error total cuadrático es el que se muestra en la ecuación.

$$E = \sum_n e_n^2 = \sum_n \left(s_n \grave{s_n} \right)^2 \tag{3.4}$$

El problema de la predicción lineal radica en encontrar los coeficientes predictores a_k que minimicen el error e_n. La condición para la minimización del error total cuadrático se obtiene estableciendo la derivada parcial del error total cuadrático E con respecto a cada de los coeficientes predictores a_k a cero [19]. El resultado se muestra en la ecuación:

$$\sum_{k=1}^{p} a_k R_{(i-k)} = R_i \tag{3.5}$$

Donde R_i es la función de autocorrelación de la señal S_n y se define por:

(3.5)

$$R(x) = \frac{\sum_{n=0}^{N-1-|x|} s(n)s(n+|x|)}{\sum_{n=0}^{N-1} s^2(n)}$$

La ecuación anterior, es una ecuación matricial, existen dos métodos fundamentales de solución que son los de autocorrelación y covarianza. Esta tesis se enfoca en el método de autocorrelación.

En el método de autocorrelación se utilizan, únicamente los puntos contenidos dentro de la ventana definida para la señal [19]. Se definen los límites de n suponiendo que el segmento de voz es cero fuera del intervalo $0 \leq n \leq N + p - 1$. De la solución por medio de mínimos cuadrados, resultan las siguientes ecuaciones expresadas en forma de matriz. La matriz de autocorrelación es mostrada a continuación:

$$\begin{pmatrix} r(0) & r(1) & \cdots & r(p-1) \\ r(1) & r(2) & \cdots & r(p-2) \\ \vdots & & \ddots & \vdots \\ r(p-1) & r(p-2) & \cdots & r(0) \end{pmatrix} \begin{bmatrix} a_1 \\ a_2 \\ \vdots \\ a_p \end{bmatrix} = \begin{bmatrix} r(1) \\ r(2) \\ \vdots \\ r(p) \end{bmatrix}$$

Esta matriz tiene Toeplitz[5] lo que permite la aplicación de la recursión de Levinson-Durbin para resolver el sistema [15].

La recursión de Levinson-Durbin es un algoritmo que permite encontrar un filtro de respuesta infinita al impulso[6] a partir de una secuencia determinística de autocorrelación dada. El filtro producido por la recursión de Levinson-Durbin es de fase mínima[7] [19].

El algoritmo de Levinson-Durbin encuentra de forma recursiva los coeficientes para el filtro de primer orden, que se denota por a_1^1, donde el superíndice denota el orden del filtro, y el subíndice, el número del coeficiente para ese orden. Enseguida, el algoritmo encuentra los coeficientes para el filtro de segundo orden a_1^2 y a_2^2, y asi sucesivamente hasta encontrar los coeficientes del filtro de orden p, $a_1^p, a_2^p, \ldots, a_p^p$.

El análisis de predicción lineal se basa en la consideración del filtro de tipo “solo polo”, de la forma:

$$H(z) = \frac{1}{A(z)} \qquad (3.6)$$

(3.7)

[5]Matriz simétrica y los elementos de sus diagonales son iguales.
[6] Filtro digital que si la entrada es una señal impulso, la salida tendrá un número infinito de términos no nulos
[7]Cuando todos los ceros y polos están en el interior de la circunferencia unidad.

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

Donde $H(z)$ es la función de transferencia de un filtro que modela el tracto vocal.

Los coeficientes de predicción lineal (LPCs) constituyen uno de los métodos más utilizados en la extracción de parámetros de voz a partir de la información del tracto vocal. Las referencias consultadas utilizan los LPCs tanto en la caracterización de segmentos de voz como en algoritmos de compresión.

3.2 MEL-CEPSTRUM

En lugar de utilizar los predictores lineales, otro método es muy comúnmente utilizado en el reconocimiento del habla, es conocido como mel-cepstrum [13]. Este método consiste en dos partes: el cálculo del cepstrum y un método llamado escala de Mel.

3.2.1 CEPSTRUM

El método cepstrum es una manera de calcular el filtro que modela al tracto vocal $H(z)$ utilizando el procesamiento homomórfico[8] [20]. En este caso las dos señales no están combinadas linealmente (una convolución no puede ser descrita como una simple combinación lineal).

Como se mostró en la figura 13, la señal de voz,$s(n)$, puede verse como el resultado de la convolución entre $u(n)$ y $h(n)$:

$$s(n) = b_o \cdot u(n) * h(n) \tag{3.8}$$

En el dominio de la frecuencia:

$$S(z) = b_o \cdot U(z) * H(z) \tag{3.9}$$

[8]Trata de la transformación a dominio lineal de las señales combinadas en una forma no lineal.

Como la excitación,$U(z)$, y el tracto vocal, $H(z)$, están combinado por una multiplicación es difícil separarlos. Si se aplica la función log a cada lado de la igualdad la ecuación se vuelve suma:

$$\log S(z) = \log\big(b_o \cdot U(z)H(z)\big) = \log\big(b_o \cdot U(z)) + \log\,(H(z)\big) \tag{3.10}$$

La propiedad aditiva del logaritmo del espectro también es válida cuando se aplica la transformada inversa. El resultado de esta operación se llama *cepstrum,* es decir el cepstrum es la transformada inversa de Fourier de espectro logarítmico de la señal [3].

Para evitar tomar logaritmo de números complejos, se aplica valor absoluto a la señal $S(z)$, esto se define "cepstrum real". Los pasos para crear un cepstrum real se muestra en la figura 14:

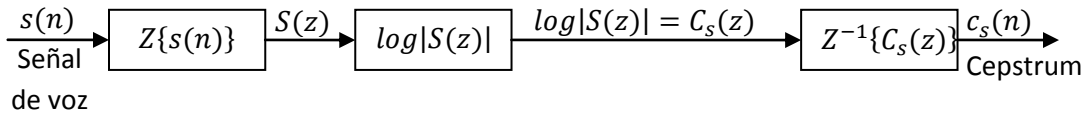

Figura 18. Pasos para obtener Cepstrum real

El cepstrum real es una secuencia entera de índice N, debido a que $C_s(z) = log|S(z)|$, es real y entero. Esta propiedad permite aplicar la transformada inversa del coseno a $C_s(Z)$ con el fin de obtener $c_s(n)$.

Las bajas componentes cepstrales corresponden a variación lentas del espectro, por lo que contiene información de la envolvente del espectro que se relaciona precisamente con el filtro que modela el tracto vocal [7]. Las altas componentes del cepstrum están más relacionadas a la fuente de excitación.

Para propósito de reconocimiento de voz es importante la información del tracto vocal más que la fuente de excitación por lo que los primeros coeficientes cepstrales contienen información importante para la extracción y características de parámetros de voz. Las referencias consultadas utilizan los 10 o 14 primeros coeficientes cepstrales sobre ventanas de 20 a 50 ms para reconocimiento de voz.

3.2.2 COEFICIENTES CEPSTRALES DE FRECUENCIA MEL

Una representación derivada de los coeficientes cepstrum son los Coeficientes Cepstrales de frecuencia Mel (MFCC) cuya diferencia fundamental con los anteriores es que las bandas de frecuencia son posicionadas de acuerdo a una escala logarítmica conocida como escala MEL (ver sección 1.4) que aproxima la frecuencia del sistema auditivo humano de forma más eficiente que como lo hacen las bandas obtenidas directamente de una transformada de Fourier (FFT) o la transformada discreta del coseno (DCT). Los pasos para calcular estos coeficientes se explicaran con mayor detalle en la sección cinco.

CAPITULO 4: CUANTIFICACIÓN VECTORIAL

Las técnicas de parametrización explicadas en la sección anterior tienen como finalidad generar una serie de coeficientes que representan las características de la señal de voz, para ser usados en la fase de reconocimiento de habla. El tamaño de la matriz obtenida del proceso de parametrización depende directamente de la longitud de la señal de voz, la cual tiene relación con la palabra en sí y el hablante. Por esta razón, se hace necesaria la estandarización de la matriz que contiene los coeficientes cepstrales calculados, de forma que el tamaño de las matrices usadas para el reconocimiento del habla sea el mismo.

La estandarización de la matriz de coeficientes cepstrales se denomina cuantificación vectorial. La esencia de la cuantificación vectorial, aplicada al caso particular del reconocimiento del habla, es la de obtener a partir de una matriz cualquiera de coeficientes cepstrales, una matriz de tamaño fijo que se parezca lo más posible a la original [21]. El espacio generado por los coeficientes cepstrales es dividido en un conjunto de regiones convexas mutuamente excluyentes y para cada una se calcula el centroide, que en dos dimensiones se representa como un punto que se encuentra a la menor distancia de todos los puntos (coeficientes cepstrales) que pertenecen a esa región. El conjunto de centroides obtenidos de la partición del espacio generado por los coeficientes cepstrales se denomina codebook. Por lo

tanto, para caracterizar cada palabra se calcula su codebook correspondiente [22]. En la figura 15 se representan dos etapas del cálculo de los centroides.

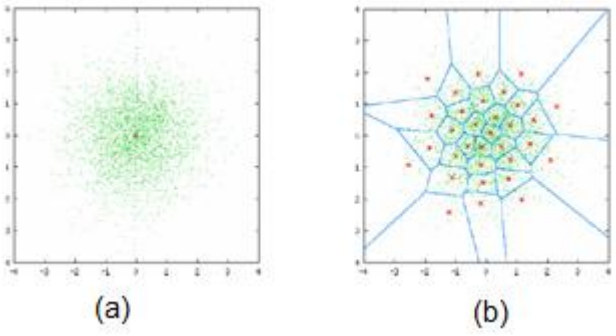

Figura 19. División del espacio generado por los coeficientes cepstrales de la vocal "a": (a) Primer cuadrante generado. (b) Codebook obtenido [22]

En la figura 15a se observa la ubicación del primer centroide calculado, mientras que en la figura 15b se aprecia el codebook correspondiente a todos los centroides generados mediantes la Cuantificación vectorial.

Ventajas:

- Reduce el almacenamiento de la información de análisis.
- Se reduce el cálculo para determinar distancias entre vectores espectrales.
- La representación del VQ se limita a una tabla que contiene las distancias entre pares de vectores del codebook.
- Representación discreta de las señales de voz. Asociando una característica fonética con cada vector del codebook, el proceso de elección del vector que mejor lo representa es equivalente a asignar una característica fonética a cada segmento de voz.

Desventajas:

- Distorsión en la representación del vector. Hay un número finito de vectores en el codebook, el proceso de "elección" del mejor representante es equivalente a cuantificar el vector y conduce a un cierto nivel de error de

cuantificación. De cualquier modo con cualquier codebook finito siempre habrá un nivel de ruido o error.

- El almacenamiento requerido para los vectores del codebook no es pequeño. Cuanto más grande sea el codebook menor es el error. Para un codebook de 1000 o más entradas, el almacenamiento no es irrelevante. Hay que realizar un balance entre error de cuantificación, procesamiento y almacenamiento del codebook.

Los principales algoritmos para la creación del codebook son Kmeans y LGB [22]. En este proyecto de fin de carrera utilizamos LGB, sin embargo ambos métodos se describen a continuación.

4.1 ALGORITMO K-MEANS

1. Inicialización: Arbitrariamente elegimos *M* vectores o codewords, como el grupo inicial del codebook.
2. Búsqueda del más cercano: Por cada vector de observación, se busca el codeword en el codebook que es el más cercano (en términos de distancia), y asigna a ese vector a la celda correspondiente.
3. Actualización del centroide: actualiza el codeword en cada celda o sector usando el centroide de los vectores de entrenamiento asignados a un sector.
4. Iteración: Repite los pasos 2 y 3 hasta que la distancia media caiga debajo de un umbral prefijado.

4.2 ALGORITMO LBG

El algoritmo LBG, lleva su nombre debido a sus autores Y. Linde, A.Buzo y R. M. Garrí, en el se elige un codeword inicial de entre los vectores de datos a clasificar, luego se utiliza el algoritmo de división binaria para duplicar el número de codewords, los vectores de observación se agrupan en torno a los codewords que les presentan menor distancia, se recalculan los codewords como la media multidimensional de cada sector y se agrupan nuevamente los datos, el proceso se

detiene cuando el codebook no presenta variación significativa y al llegar al número de codewords deseados [22].

Este algoritmo es de gran popularidad porque produce codebooks que logran un mínimo local en la función de error por distorsión [22].

Se abordaran mas detalles sobre su impletacion en sección cinco (Implementación de sistema de reconocimiento de voz).

CAPITULO 5: MODELOS OCULTOS DE MARKOV (HIDDEN MARKOV MODELS)

Un modelo oculto de Markov se define como una máquina de estados finitos en que el siguiente estado depende únicamente del estado actual y asociado a cada transición entre estados se produce un vector de observaciones. Los modelos de Markov llevan asociados dos procesos: uno oculto (no observable directamente) correspondiente a las transiciones entre estados, y otro observable (y directamente relacionado con el primero), cuyas realizaciones son vectores de observaciones que se producen desde cada estado y forman la plantilla a reconocer [23].

5.1 ELEMENTOS DE MODELOS DE MARKOV

Los elementos de un HMM son [7]:

1. N, el número de estados del modelo. Aunque los estados estén ocultos, para muchas aplicaciones prácticas hay un significado físico adjunto a los estados del modelo. Se denota cada estado individual como $S = \{S_1, S_2, \dots S_N\}$ y el estado en el tiempo t como q_t donde $q_t = S$.

2. M, el número de símbolos de distintas observaciones por estado (por ejemplo el tamaño del alfabeto). El símbolo de observación corresponde a la salida física del sistema que se modelando. Se denotan los símbolos individuales como $V = \{V_1, V_2, \dots V_M\}$.

3. Matriz de probabilidad de transición de estados $A = \{a_{ij}\}$ donde:

$$a_{ij} = P[q_{t+1} = S_j | q_t = S_i], \qquad 1 \leq i, j \leq N$$

4. Matriz de probabilidad de generación de vectores o distribuciones $B = \{b_j(k)\}$. Cada uno de sus elementos se define como la probabilidad de que

un determinado vector de características en un determinado tiempo haya sido generado por la distribución correspondiente, donde:

$$b_j(k) = P\left[v_k ent | q_t = S_j\right], \qquad 1 \leq j \leq N$$

$$1 \leq k \leq$$

5. La distribución de estado inicial $\pi = \{\pi_i\}$representa las condiciones iniciales de los modelos de Markov donde:

$$\pi_i = P[q_1 = S_i], \qquad 1 \leq i \leq N$$

De tal forma, que cualquier modelo de Markov se puede definir como:

$$\lambda = (\mathrm{A}, \mathrm{B}, \pi)$$

5.2 TIPOS DE MODELOS OCULTOS DE MARCOV

Se pueden utilizar distintos tipos de estructuras para los modelos ocultos de Markov. Las estructuras están definidas por la matriz de transición A. La estructura más sencilla es la estructura ergódica o completamente conectada, en este estado todo puede ser alcanzado desde cualquier otro estado del modelo. Como se muestra en la figura 16a, para un modelo de cuatro estados, tiene la propiedad $0 < a_{ij} < 1$ (el cero y el uno tienen que ser excluidos para que la propiedad de ergódica pueda cumplirse). La matriz de transición A, para una modelo ergódica, se puede describir como:

$$A = \begin{bmatrix} a_{11} & a_{12} & a_{13} & a_{14} \\ a_{21} & a_{22} & a_{23} & a_{24} \\ a_{31} & a_{32} & a_{33} & a_{34} \\ a_{41} & a_{42} & a_{43} & a_{44} \end{bmatrix}$$

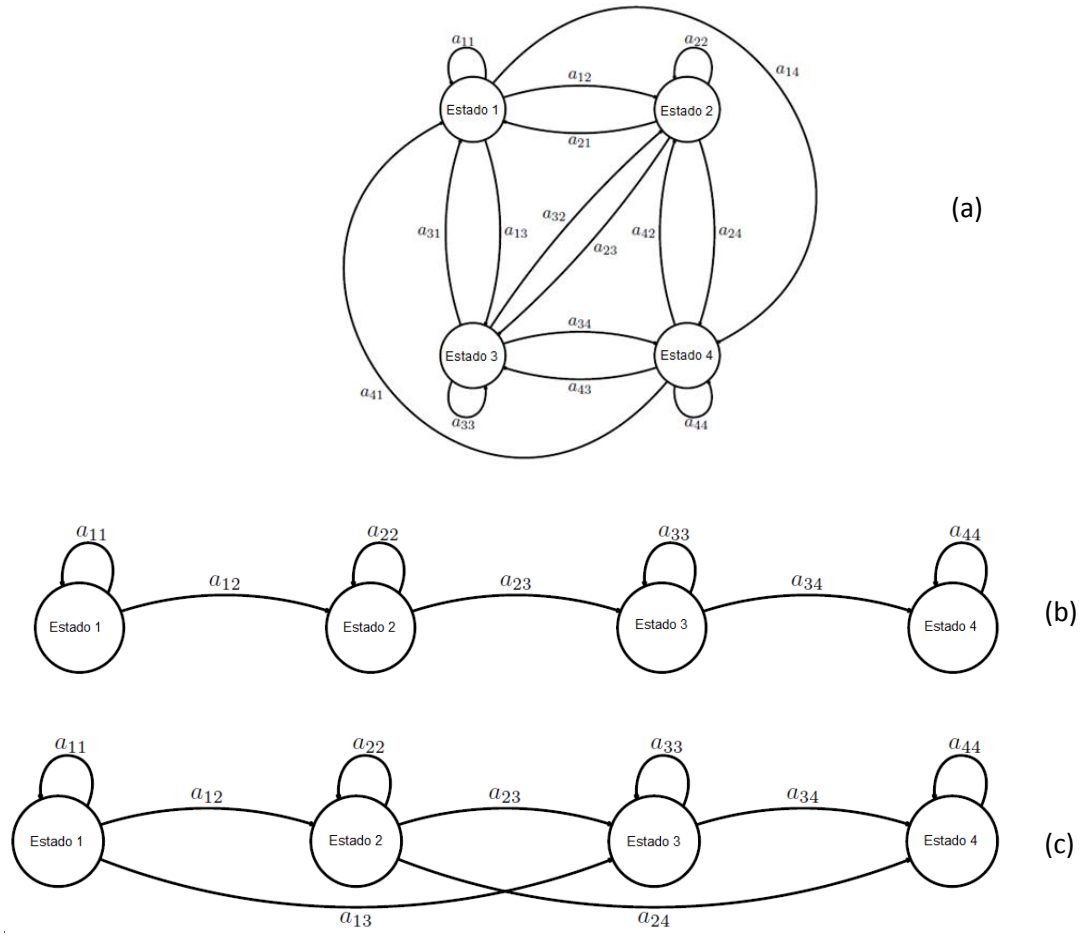

Figura 20. Distintos tipos de modelos HMM [23]

En los sistemas de reconocimiento de voz, es mejor usar un modelo que organiza los estados de forma sucesiva, debido a que esta es una propiedad de la voz. El modelo que cumple este requerimiento, es el modelo izquierda-derecha o modelo Bakis, figura 16b,c. La propiedad para el modelo izquierdad-derecha es:

$$a_{ij} = 0, \qquad j < i$$

Esto significa que no se puede hacer ningún salto al estado previo. La longitud de las transiciones usualmente estan restringidas a una logitud máxima, tipicamente dos o tres:

$$\mathrm{a_{ij}} = 0, \qquad \mathrm{j} < i + \Delta$$

Para este modelo, los coeficientes de transicion de estados para el último estado, tienen las siguientes propiedades:

$$a_{NN} = 1$$

$$a_{Nj} = 0, \quad j < N$$

En la figura 16b y la figura 16c se presentan dos modelos izquíerda-derecha. En la fígura 16b, $\Delta = 1$ y la matriz de transición A es:

$$A = \begin{bmatrix} a_{11} & a_{12} & 0 & 0 \\ 0 & a_{22} & a_{23} & 0 \\ 0 & 0 & a_{33} & a_{34} \\ 0 & 0 & 0 & a_{44} \end{bmatrix}$$

Y en la fígura 16c , $\Delta = 2$ y la matriz de transición A es:

$$A = \begin{bmatrix} a_{11} & a_{12} & a_{13} & 0 \\ 0 & a_{22} & a_{23} & a_{24} \\ 0 & 0 & a_{33} & a_{34} \\ 0 & 0 & 0 & a_{44} \end{bmatrix}$$

La selección de la estructura del modelo, como izquíerda-derecha, no requiere modificación en los algoritmos de entrenamiento. Esto es debido a que cualquier probabilidad de estado que sea cero permanecerá en cero en los algoritmo de entrenamiento.

5.3 PROBLEMAS DE LOS HMM

Desde el punto de vista del modelado de segmentos o símbolos de voz, los HMM son muy versátiles, pudiéndose realizar el modelo de fonemas, palabras, y hasta frases enteras. En el caso de nuestra tesis utilizamos modelo de palabras.

Los modelos de Markov se caracterizan por tres problemas que hay que resolver para que se obtengan modelos útiles en aplicaciones reales [23] [24]:

- Problema de evaluación: Dada una secuencia de observaciones y un modelo, calcular la probabilidad de que la secuencia observada haya sido producida por dicho modelo.
- Problema de estimación: Dada una secuencia de observaciones y un modelo, encontrar una secuencia de estados que sea óptima en algún sentido.
- Problema de entrenamiento: Dada una secuencia de observaciones de entrenamiento, encontrar los parámetros del modelo de forma óptima.

5.4 SOLUCIÓN DE LOS TRES PROBLEMAS BÁSICOS DE HMM.

5.4.1 SOLUCIÓN AL PROBLEMA UNO.

Se desea calcular la probabilidad de la secuencia de observaciones $O = O_1, O_2, \dots, O_T$, dado el modelo λ es decir $P(O|\lambda)$. La forma más directa de resolver este problema es enumerar cada posible secuencia de estado de longitud T (el número de observaciones). Considerar una secuencia de estado fijo:

$$O = q_1 q_2 \dots q_T$$

Esto significa que para calcular $P(O|\lambda)$, de acuerdo con la ecuación 5.1, es necesario realizar $2T \cdot N^T$ calculaciones, debido a que para cada t=1, 2, 3…T, existen N posibles estados que pueden ser alcanzados (es decir N^T posibles secuencias de estados), y para cada secuencia de estados acerca de 2T calculaciones son requeridas para cada termino en la suma de la ecuación 5.1. Los cálculos requeridos son computacionalmente inviable aún para valores pequeños de N y T.

Para reducir la cantidad de cálculos necesarios se utiliza el algoritmo de avance y retroceso [24] [7].

5.4.1.1 ALGORITMO AVANCE- RETROCESO

5.4.1.1.1 ALGORITMO DE AVANCE

Considerar una variable delantera $\alpha_t(i)$ definida como:

$$\alpha_t(i) = P\big((o_1 o_2 \cdots o_t, q_t = i)\big|\lambda\big)$$

Donde t representa el tiempo e i es el estado. Esto da que $\alpha_t(i)$ será la probabilidad de observación secuencia parcial, $o_1 o_2 \cdots o_t$ (hasta el momento t) al estar en estado i en el tiempo t. La variable delantera se puede calcular inductivamente, ver la figura 30.

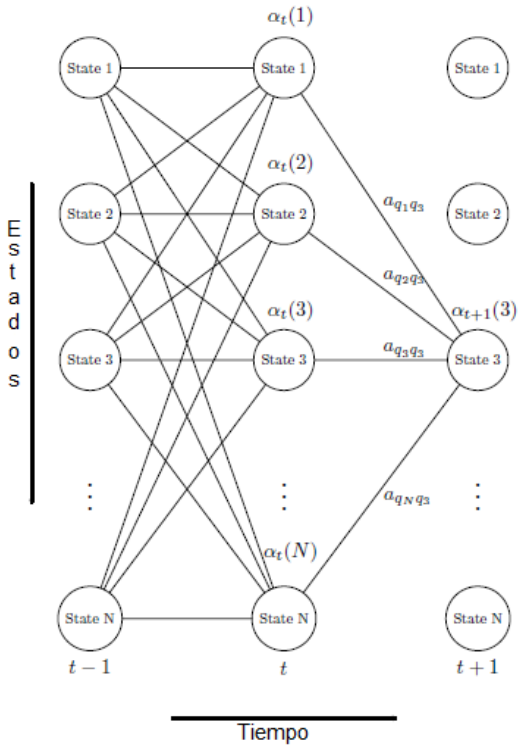

Figura 21. Procedimiento de avance

$\alpha_t + 1(i)$ se obtendrá por la suma de la variable delantera para todos los estados N en el tiempo t multiplicado con su variable correspondiente de estado, a_{ij} y por la probabilidad de emisión $b_j(o_t + 1)$. Esto se puede hacer con el siguiente procedimiento:

1. Inicialización

Establecer $t = 1$;

$$\alpha_1(i) = \pi_i b_i(o_1)$$

2. Inducción

$$a_t + 1(j) = b_j(o_{t+1}) \sum_{i=1}^{N} \alpha_t(i) a_{ij}, 1 \leq j \leq N$$

3. Tiempo de actualización

Establecer $t = t + 1$;

Volver al paso 2 si $t < T$;

De lo contrario, terminar el algoritmo (ir paso 4).

4. Terminación

$$P(O|\lambda) = \sum_{i=1}^{N} \alpha_T(i)$$

Si se utiliza el algoritmo delantero hay una necesidad de multiplicar $N(N+1)(T-1)+N$ y sumar $N(N-1)(T-1)$. De nuevo para $N = 5$ (estados), $T = 100$ (observaciones), esto produce la multiplicacion $5(5+1)(100-1)+5 = 2915$ y las sumas $5(5-1)(100-1) = 1980$. Esta es una gran mejora en comparacion con el metodo directo (10^{72} multiplicaciones y 10^{69} sumas).

5.4.1.1.2 Algoritmo de retroceso

La recursión se describe en el algoritmo delantero, también se puede hacer en el tiempo de inversión. Definiendo la variable de retroceso $\beta_t(i)$como:

$$\beta_t(i) = P(o_{t+1} o_{t+2} \cdots o_T | q_t = i, \lambda)$$

Es decir, la probabilidad secuencial de observacion parcial de $t+1$ hasta el final, deterinado estado i en el tiempo t y el modelo λ. Notar que la definición de la variable delantera es una probabilidad conjunta mientras que la probabilidad de

retroceso es un probabilidad condicional. De manera similar (según el algoritmo delantero), el de retroceso puede ser calculada inductivamente, ver figura 31.

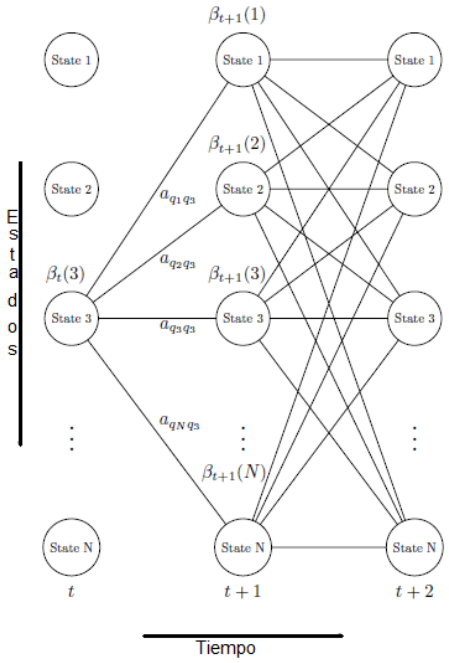

Figura 22. Procedimiento de retroceso

El algoritmo de retroceso incluye los siguientes pasos:

1. Inicialización

Establecer $t = T - 1$;

$$\beta_T(i) = 1, 1 \leq i \leq N$$

2. Inducción

$$\beta_t(i) = \sum_{j=1}^{N} \beta_{t+1}(i) a_{ij} b_j(o_{t+1}), 1 \leq i \leq N$$

3. Tiempo de actualización

Establecer $t = t - 1$;

Volver al paso 2 si $t > 0$;

De lo contrario terminar el algoritmo.

Tenga en cuenta que la inicialización paso 1 define arbitrariamente $\beta_T(i)$ será 1 para todo i.

5.4.2 SOLUCIÓN AL PROBLEMA DOS.

Distinto a la solución presentada para el problema uno, existen muchas formas de solucionar el problema dos [23]. La dificultad reside en la definición de "optima" secuencia de estado, ya que existen muchos posibles criterios de optimización.

Una de las técnicas utilizadas se conoce como *algoritmo de Viterbi* el cual es similar al *algoritmo de avance y retroceso.*

5.4.2.1 EL ALGORITMO VITERBI

Este algoritmo es similar al algoritmo de adelanto. La principal diferencia esta en que el agoritmo de adelanto sumas sobre los estados anteriores, mientras que el algoritmo Viterbi utiliza la maximizacion. El objetivo del algoritmo de Viterbi es encontrar la mejor secuencia de estados, $\boldsymbol{q} = (q_1, q_2, \cdots, q_T)$, para la secuencia de observación dada $O = (o_1, o_2, \cdots, o_T)$ y un modelo λ. Cnsidere las siguientes cantidades:

$$\delta_t(i) = \max_{q_1, q_2, \cdots, q_{t-1}} P\left(q_1 q_2 \cdots q_{t-1}, q_t = i, o_1 o_2 \cdots o_t | \lambda\right) \qquad (4.65)$$

Eso es la probabilidad de observar $o_1 o_2 \cdots o_t$ usando el mejor camino que termina en el estado i en el momento t, dado el modelo λ. Por el uso de la induccion $\delta_{t+1}(i)$ se puede encontrar como:

$$\delta_{t+1}(i) = b_j(o_{t+1}) \max_{1 \le i \le N} \left[\delta_t(i) a_{ij}\right] \qquad (4.66)$$

Para recuperar efectivamente la secuenca de estados, es necesario no perder de vista el argumento que maximiza (4.66), por cada t y j. Esto se hace por salvar el argumento de una matriz $\psi_t(j)$. Aquí sigue el algoritmo de Viterbi completo:

1. Inicialización

Establecer $t = 2$;

$$\delta_1(i) = \pi_i b_i(o_1), \qquad 1 \leq i \leq N \tag{4.67}$$

$$\psi_1(i) = 0, 1 \leq i \leq N \tag{4.68}$$

2. Inducción

$$\delta_t(j) = b_j(o_t) \max_{1 \leq i \leq N} \delta_{t-1}(i) a_{ij}, \qquad 1 \leq j \leq N \tag{4.69}$$

$$\psi_t(j) = arg \max_{1 \leq i \leq N} \left[\delta_{t-1}(i) a_{ij}\right], \quad 1 \leq j \leq N \tag{4.70}$$

3. Tiempo de actualización

Establecer $t = t + 1$;

Regresar al paso 2 si $t \leq T$;

De lo contrario, terminar el algoritmo (ir al paso 4).

4. Terminación

$$P^* = \max_{1 \leq i \leq N} [\delta_T(i)] \tag{4.71}$$

$$q_T^* = arg \max_{1 \leq i \leq N} [\delta_T(i)] \tag{4.72}$$

5. Camino de retroceso (secuencia de estado)

a) Inicialización

Establecer $t = T - 1$

b) Retroceso

$$q_t^* = \psi_{t+1}(q_{t+1}^*) \qquad (4.73)$$

c) Tiempo de actualización

Establecer $t = t - 1$;

Regresar al paso (b) si $t \geq 1$;

De lo contrario, terminar el algoritmo.

5.4.3 SOLUCIÓN AL PROBLEMA TRES.

El problema 3 trata de optimizar los parámetros del modelo. La secuencia de observación usada para ajustar los parámetros se llama secuencia de entrenamiento, dado que se usa para entrenar el HMM. Aunque no existe un camino analítico para resolver este problema, podemos escoger $\lambda = (\mathrm{A}, \mathrm{B}, \pi)$ tal que $P(O|\lambda)$ se maximice localmente usando un procedimiento iterativo llamado Baum-Welch [23] [24].

V. IMPLEMENTACIÓN DE SISTEMA DE RECONOCIMIENTO DE VOZ

CAPITULO 6: DISEÑO DEL SISTEMA DE RECONOCIMIENTO DE VOZ

El diseño elaborado en este proyecto, como el título lo indica, está basado en utilizar modelos ocultos de Markov para el reconocimiento de palabras aisladas. En este capítulo se describirán las estructuras del algoritmo de reconocimiento, incluyendo las librerías y funciones utilizadas en la programación. El diseño presenta las siguientes características: vocabulario expandible, independiente del locutor, palabras aisladas, ejecución sobre plataforma de software MATLAB 10b.

6.1 ESTRUCTURA DEL ALGORITMO RECONOCIMIENTO

La implementación del algoritmo se divide en tres tareas específicas: entrenamiento del codebook, entrenamiento del Modelo Oculto del Markov (HMM) de cada palabra y reconocimiento (Figura 17).

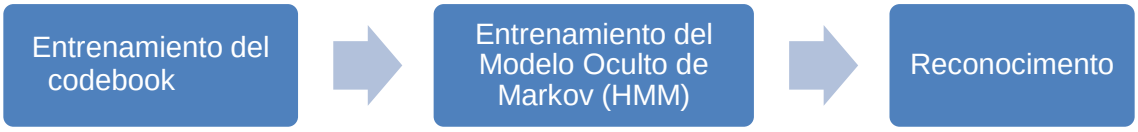

Figura 23. Estructura del prototipo de reconocimiento de palabras aisladas.

6.1.1 ENTRENAMIENTO DEL CODEBOOK

Para el entrenamiento del codebook se realizaron los siguientes pasos: adquisición y muestreo de la señal de voz, cuantificación, pre-procesamiento, segmentación extracción de parámetros característicos y cálculo del codebook (Figura 18). Es necesario realizar repeticiones de cada palabra y almacenar los parámetros característicos de cada una, en una matriz para el posterior cálculo de los centroides [1].

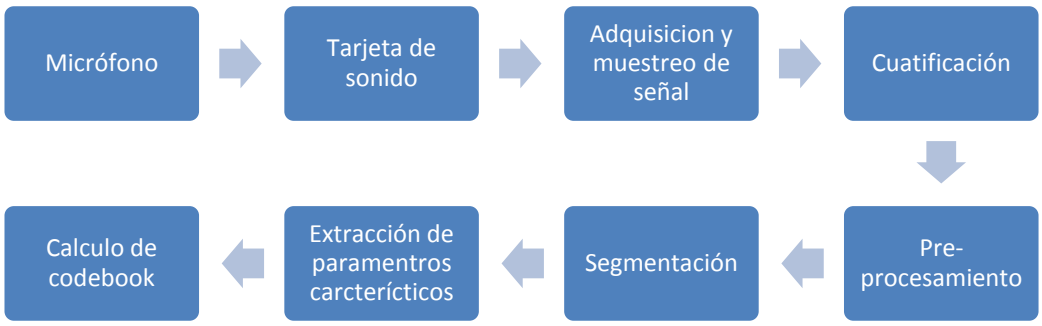

Figura 24. Entrenamiento del codebook.

El primer codebook se generó con la palabra “chicago” repetida 14 veces por cinco usuarios distintos (70 repeticiones), los datos se almacenaron en un archivo .mat.

6.1.1.1 ADQUISICIÓN

Como se menciono en el capítulo 2 de este documento, la adquisición de la señal se puede realizar en dos modalidades, la primera a través de un archivo .wav y la segunda a través de una grabación.

ARCHIVO

Los archivos .wav que se utilizan para entrenar y comprobar el funcionamiento del sistema fueron extraídos de la base de datos MAMI (Database of isolated spoken words through a mobile) Todos los usuarios utilizaron para grabar un teléfono HTC Touch (TM) con plataforma Windows Mobile 6.0 [12]. Para iniciar o detener la grabación el usuario necesita presionar la pantalla. La frecuencia de muestreo para estas grabaciones fue de 11025 Hz y 16 bits por muestra.

GRABACIÓN

Las características del objeto Analog Input empleado para la grabación fueron:

Bits por muestra: 16

Buffer en modo: Automático

Canal empleado: 1

Retardo por canal (Skew): Cero

Modo del retardo por canal: Cero

Origen del Reloj: Interno

Tipo de entrada: Entrada acoplada en AC.

Modo de almacenamiento: En memoria.

Nombre interno del dispositivo: Winsound.

Tasa de muestreo empleada: 8000 muestras por segundo.

6.1.1.2 PRE-PROCESAMIENTO

Consta de las siguientes fases: filtro pasa bajo y filtro de preénfasis.

FILTRO PASA-BAJOS

Se utiliza un filtro pasa-bajos con frecuencia de corte en 4khz y ganancia de 0 dBs, con el fin de eliminar la información en frecuencias superiores a los 4 Khz., con lo cual se ahorra tiempo de procesamiento.

```
[numd,demd] = butter(6,0.975,'low');
senyal=filter(numd,demd,senyal);
```

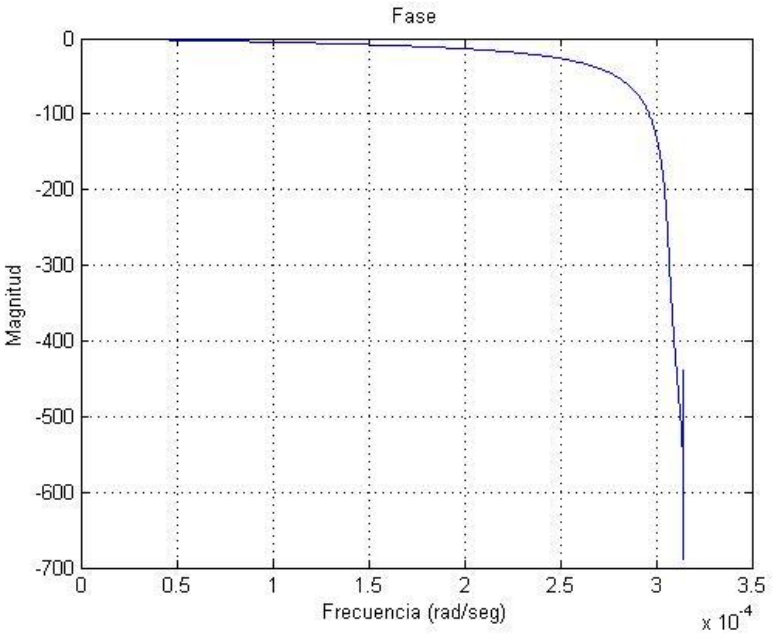

Figura 25. Filtro pasa-bajo

PREÉNFASIS

Antes de realizar el procesamiento digital de la señal, se realizó un filtrado pasa-alto de pre-énfasis con el objetivo de aumentar la energía relativa de las componentes de alta frecuencia en el espectro de la voz, y adicionalmente suavizar el espectro. El filtro de pre-énfasis empleado es de primer orden con frecuencia de corte en 2500 Hz y ganancia de 6 dBs, este tiene la forma:

$$H(z) = 1 - a(z) \qquad (6.2)$$

Luego de hacer pruebas con la constate del filtro se observó que el mejor funcionamiento se obtuvo en el valor a = 0.95. Como se mencionó en la sección 2.4 de este documento, el filtro de pre-énfasis compensa la caída de 6dB que sufre la voz cuando pasa a través del tracto vocal.

```
filter([1 -.95], 1, senyal);
```

En la figura 21 se muestra una trama de la palabra “chicago” antes y después de ser pre-enfatizada.

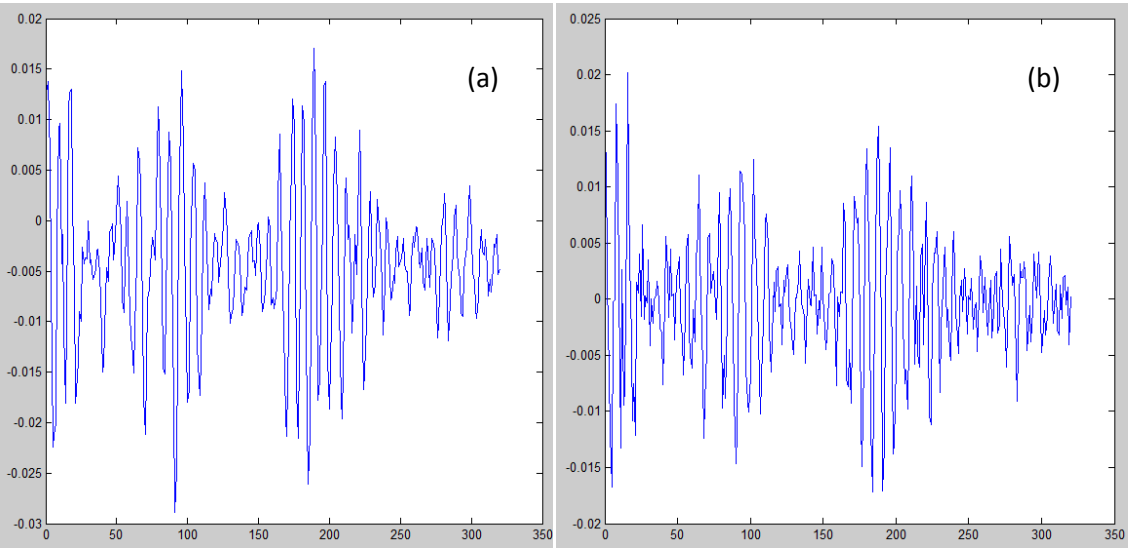

Figura 26. Trama de palabra “chicago”, (a) Antes de preénfasis, (b) Después de preénfasis.

6.1.1.3 SEGMENTACIÓN

Consta de las siguientes fases: análisis de energía, detección de inicio y fin de la palabra y aplicación de ventana.

ANÁLISIS DE ENERGÍA

La energía transportada por la señal de voz depende de las condiciones durante la adquisición de datos. Factores como el cambio de locutor, el tono al hablar o la presencia de ruido pueden modificarla. A través de la realización de pruebas se determinó que si la energía promedio supera el umbral de 0.000001 la señal contiene sonido, en caso contrario el programa generará un mensaje que informará al usuario que la señal grabada no tiene tramas de sonido.

```
[bool_energia] =energia_media(seyal, 8000);
ifbool_energia<=0.000001,
msgbox('La senyal no contiene Tramas de sonido','Advertencia');
```

DETECCIÓN DE PUNTO INICIAL Y FINAL

En el reconocimiento de señales de voz, se hace necesario determinar con adecuada precisión los puntos de inicio y final de cada palabra, es decir, se debe diferenciar las partes de la señal que llevan información de voz de aquellas que no. Este procedimiento evita gastar memoria y tiempo de cálculo en las tramas que no contienen información.

Para nuestro diseño utilizamos tres criterios para determinar si un segmento de voz es vocalizado o no, estos criterios son análisis de magnitud, cruce por cero y tono (ver figura 22).

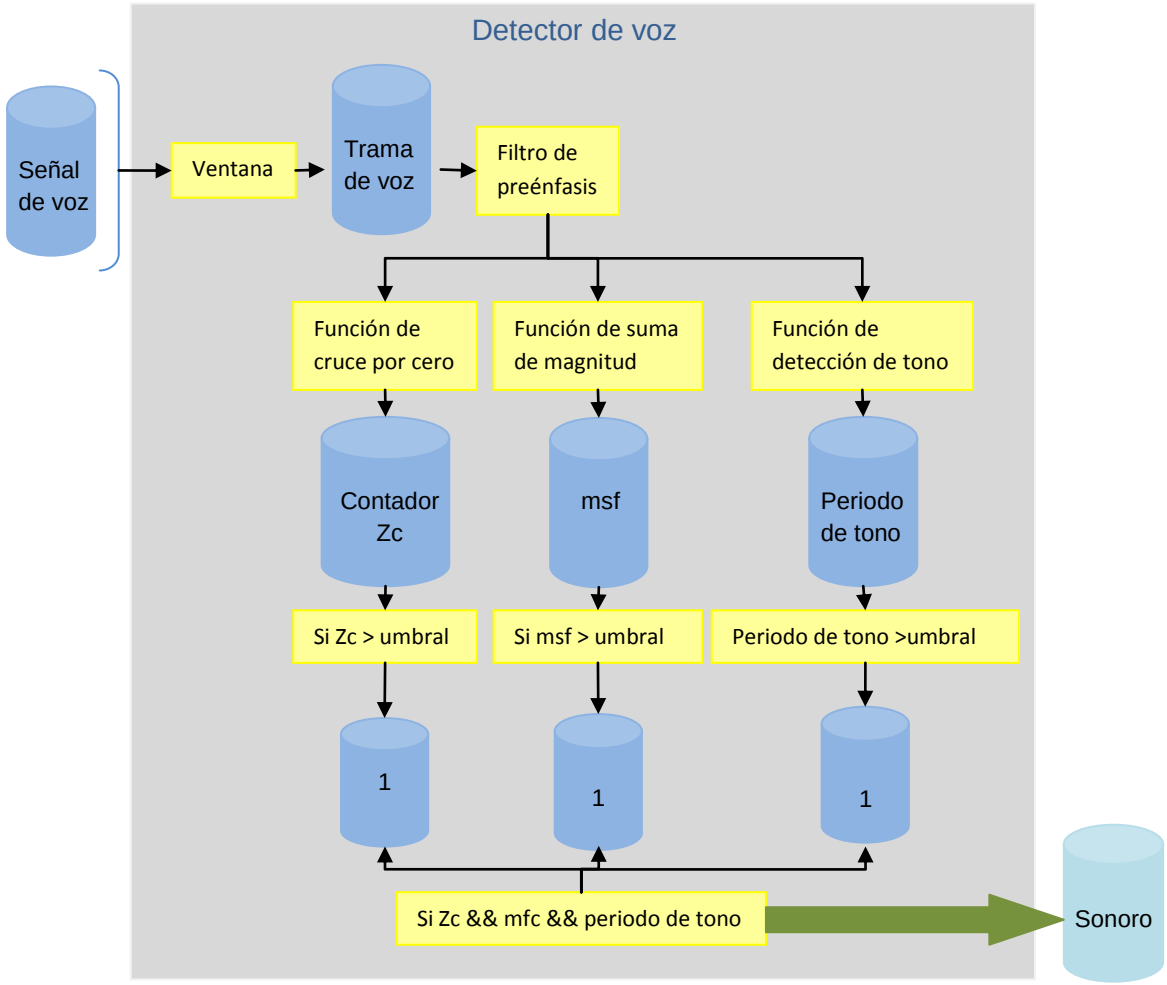

Figura 27. Algoritmo de detección de punto final

DETECCIÓN DE VOZ POR SUMA DE MAGNITUD.

Tomamos la señal de audio y la frecuencia de muestreo, y calculamos la frecuencia de muestreo normalizada para obtener los coeficiente del filtro butterworth. Una vez que tenemos los coeficientes aplicamos el filtro a la señal de voz y calculamos la suma de la magnitud de la salida del filtro. Asignamos el valor de esta suma a toda esta trama de la señal. Si el valor de cada trama es mayor que 67 % de la media de la señal más el 47% del valor mínimo, se asigna el valor 1 a la trama [16].

```
m_s_f=sum(abs(y1));
```

DETECCIÓN DE VOZ POR CRUCE POR CERO.

Utilizamos la técnica de cruce por cero para clasificar los segmentos en sonoros o no sonoros. Ya que las altas frecuencias implican un alto cruce por cero, existe una relación entre la distribución de energía con la frecuencia y la tasa de cruce por cero. Esta observación nos lleva a la conclusión que si la tasa de cruce por cero es alta la señal no es sonara y viceversa.

El algoritmo utilizado para calcular la tasa de cruce por cero es el siguiente: inicializamos en cero un contador de cruce por cero. Después comparamos el signo entre cada muestra y la siguiente. Si el signo cambió entre las dos muestras entonces incrementamos el contador de cruce por cero. Asignamos el valor de este contador a toda la trama de la señal. Si el valor de cada trama es mayor que 150 % de la media de la señal menos el 50% del valor mínimo, se asigna el valor 1 a la trama [16].

DETECCIÓN DE TONO DEL SEGMENTO DE AUDIO.

En esta función utilizamos como entrada la señal de voz y la frecuencia de muestreo inicial y calculamos el periodo mínimo basado en muestras de 2ms y el periodo máximo basado en muestras de 20ms.

Para la detección de tono utilizamos una modificación del algoritmo de recorte de centro, llamado técnica no lineal de recorte infinito de pico. En este algoritmo seleccionamos el umbral de corte a $\pm$ 68% de la muestra de valor máximo. Si una muestra supera el umbral se cambia el valor a 1, si una muestra no supera el umbral se cambia el valor a -1, en caso contrario el valor de la muestra se hace 0.

Después de aplicar la técnica no lineal de recorte infinito de pico a la señal de entrada, calculamos la autocorrelación de la señal modificada. Aplicamos un algoritmo de recolección de pico a la función de autocorrelación de cada segmento. El primer paso de este algoritmo es escoger el máximo pico de la salida de la autocorrelación.

Después de que hemos encontrado la magnitud más alta reducimos el área del segmento de audio entre el período mínimo (2 ms) y el período máximo (20 ms) desde el punto de magnitud más alto. Entonces buscamos el valor de magnitud máximo dentro del segmento reducido y el índice magnitud más alto, luego al punto más alto del segmento reducido le agregamos el periodo de tono mínimo para obtener el periodo de tono final del segmento de audio.

VENTANEO

Como se mencionó en la sección 2.4 de este documento se utilizará la ventana Hamming. Dicha ventana presenta una atenuación de −43dB aproximadamente, lo que causa que la longitud efectiva del intervalo de análisis se reduzca aproximadamente en un 40% debido a la atenuación de la señal en los bordes de la ventana. Por esta razón se emplearon ventanas deslizantes con traslape. La señal pre-acentuada se dividió en tramas de 40ms, cada trama se traslapo con la trama vecina 10ms. En la figura 23 una sección de la señal de voz antes y después de la aplicación de la ventana.

La ecuación de la ventana utilizada para el análisis de señales de voz, y se define como:

$$W_n = \begin{cases} 0.54 - 0.46\cos\left(\frac{2\pi n}{N-1}\right), para\ 0 \le n \le N-1 \\ 0, en\ caso\ contrario \end{cases} \qquad (6.3)$$

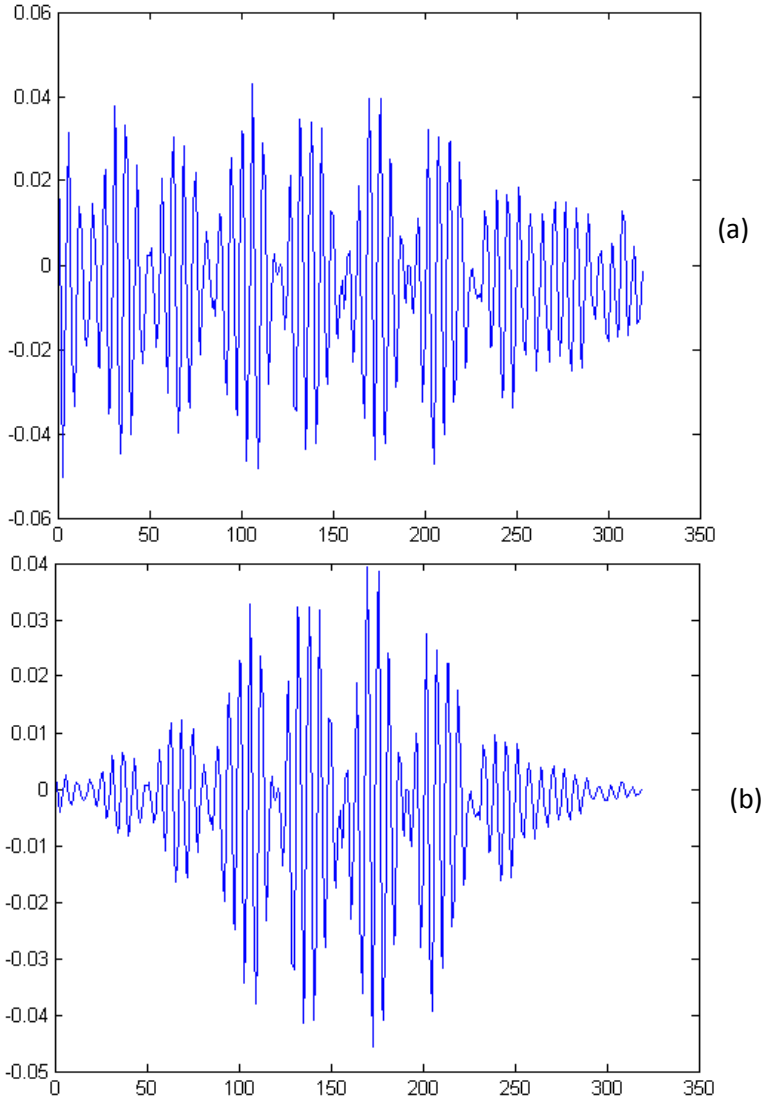

Figura 28. Trama de palabra "chicago", (a) Antes de ventana, (b) Después de ventana hamming

6.1.1.4 EXTRACCIÓN DE PARÁMETROS CARACTERÍSTICOS COEFICIENTES CEPTRALES DE FRECUENCIA MEL (MFCC)

El MFCC es una representación de la señal de voz que se define como el cepstrum real de una ventana de corta duración de la señal derivada de la FFT de esa señal que, se somete primero a un registro basado en la transformada del eje de frecuencia (Mel-frecuencia de la escala), y entonces se correlaciona mediante una modificación de la Transformada Discreta de coseno (DCT) [8]. La Figura 24 ilustra

el proceso completo para extraer los vectores MFFC de la señal de voz. Es necesario destacar que el proceso de extracción MFCC se aplica sobre cada trama de señal de voz de forma independiente.

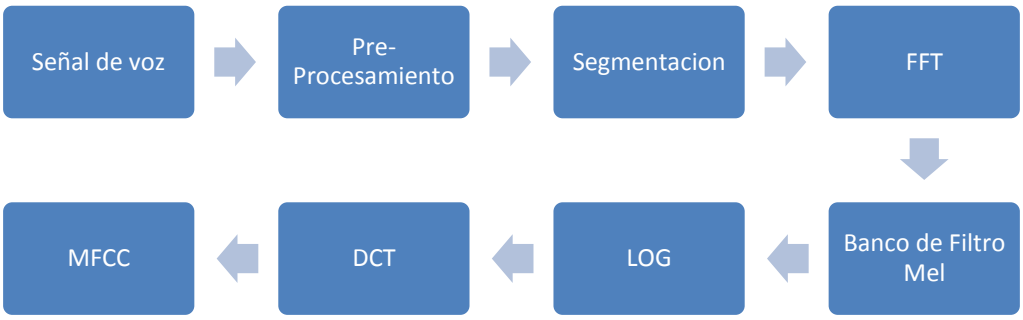

Figura 29. Proceso de extracción de MFCC

El primer paso del proceso de extracción de MFCC es calcular la Transformada Rápida de Fourier (FFT) de cada trama y obtener su magnitud. Esto se lleva a cabo utilizando la función *FFT ()* de la toolbox Signal Processing de MATLAB el número de puntos elegidos para el cálculo es 512. La salida de esta función esta en forma compleja por lo tanto hay que calcular el módulo de la transformada con la función abs ().

El próximo paso fue adaptar la resolución de frecuencia a una escala de frecuencia perceptual que satisface las propiedades de los oídos humanos, la escala seleccionada fue la escala Mel. Esta consiste en un conjunto de filtros paso-banda cuyo ancho de banda y distancias son aproximadamente iguales a las de las bandas críticas y cuya gama de las frecuencias centrales cubre las frecuencias más importantes para la percepción del habla [1].

En el banco de filtros Mel, utilizado en este diseño, la frecuencia más baja es 133.33 Hz y la segunda frecuencia esta espaciada 66.66 Hz este espacio continua siendo el mismo para el resto de frecuencias hasta los 1000 Hz a partir de entonces el

espacio aumenta logarítmicamente. La altura del triángulo también disminuye proporcionalmente a la separación entre frecuencias (figura 25).

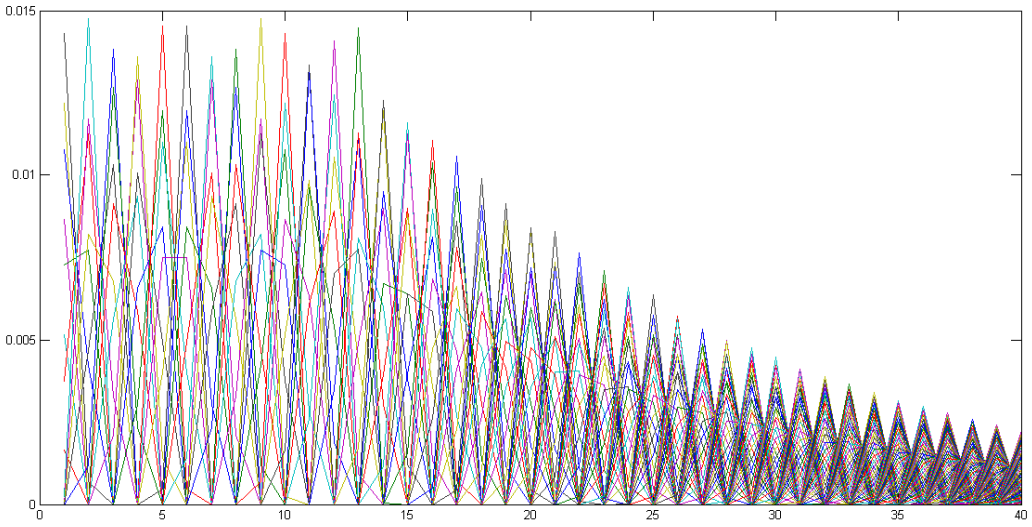

Figura 30. Banco de filtro Mel.

La entrada al banco de filtros Mel es el espectro de potencia de cada trama, Xframe [k], de tal forma que para cada trama es generado un vector de logaritmo espectral, eFrame [m]. Tal vector contiene las energías de frecuencia central de cada filtro. Así, que el banco de filtros muestrea la trama del espectro de la señal en las frecuencias centrales que conforma la escala de frecuencias Mel.

Como último paso se calcula la DCT(Transformada Discreta del Coseno) que no es más que la parte real de la transformada discreta de Fourier, el cálculo se realizó utilizando la función de MATLAB dct(). Una vez calculada esta transformada obtenemos a la salida una matriz de vectores MxN donde M son los 13 primeros coeficientes y N los cuadros de tiempo.

6.1.1.5 MODULO DE ENTRENAMIENTO DEL CODEBOOK

CALCULO DEL CODEBOOK

Para realizar reconocimiento de palabras aisladas se entrenó el Codebook con las palabras del vocabulario a reconocer. Con el método de parametrización MFCC se logró reducir la información de la señal de voz a 13 coeficientes. De esta forma

disminuyó la complejidad computacional del proceso de entrenamiento y reconocimiento. A pesar de esta reducción es necesario tener una menor cantidad de datos en el sistema para esto se empleó la cuantificación vectorial.

Este proceso consiste en promediar vectores de características semejantes, para formar un Codebook de 64 vectores. El vector resultante de cada uno de los promedios es el centroide (codeword o centro de masas) de la agrupación (o clúster) de vectores usados para calcularlo, y será el que mejor los represente en su media.

Se emplearon 64 centroides ya que experimentando con Codebooks más grandes (128 centroides) se pudo obtener un mejor modelado como se puede observar en el experimento 2, prueba 3 del capítulo 7, pero se aumentó significativamente el número de símbolos en las secuencias de entrenamiento de los HMM. Esto es debido a que la estimación de cada centroide es equivalente a la estimación de una media, y la variabilidad en la estimación de una media es inversamente proporcional al número de vectores usado. La dimensión del Codebook necesaria para conseguir una cuantificación adecuada es un asunto esencial en su diseño y define la potencia de cálculo requerida.

Para calcular los centroides utilizamos el algoritmo LBG, los pasos necesarios se muestran a continuación:

1) Elegir un codeword inicial entre los vectores de datos a clasificar.

2) Utilizar el algoritmo de división binaria para duplicar el número de codewords.

3) Clasificar cada vector de entrenamiento {xk} en uno de los clusters Ci, eligiendo el centroide más cercano.

4) Actualización del codebook: calcular el centroide de cada clúster como la media multidimensional de los vectores de entrenamiento contenidos en ese clúster.

5) Terminación: si la variación en la distorsión media global D entre esta iteración y la anterior es inferior a un umbral, PARAR. En caso contrario, ir al Paso 3.

En definitiva, este proceso de minimizar la distorsión media se divide en dos pasos básicos:

1) Reasignar los vectores: Asumiendo que se ha encontrado el centroide zi del clúster Ci, entonces la minimización consiste en asignar el conjunto de los vectores de entrenamiento a su clúster más cercano de acuerdo a la medida de distancia utilizada.

2) Recolocar los centroides: Por otro lado, dadas las particiones, para minimizar se encuentra el nuevo centroide de cada clúster de modo que se minimice la distorsión media dentro del clúster, que como ya hemos visto es el vector media de los vectores del clúster.

Iterando sobre estos dos pasos puede obtenerse un valor de la distorsión media global D más pequeño que el de la iteración anterior.

6.1.2 ENTRENAMIENTO DEL HMM DE CADA PALABRA

Para modelar el mecanismo de producción del habla se emplearon HMM. Su capacidad de discriminación se debe a que están constituidos por un conjunto de estados, comparables a los estados que atraviesa el tracto vocal cuando hablamos. Cada uno de los estados produce un conjunto de posibles salidas, asimilables a los sonidos que configuran la señal de voz.

El entrenamiento del HMM de cada palabra se realizó siguiendo los siguientes pasos: adquisición y muestreo de la señal de voz, cuantificación, pre-procesamiento, extracción de parámetros característicos, cuantificación vectorial y entrenamiento del modelo de cada palabra (Figura 26).

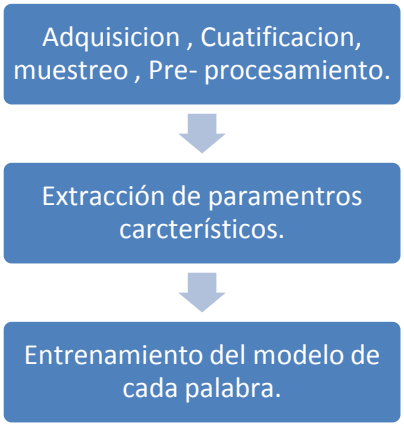

Figura 31. Proceso de entrenamiento de cada palabra del HMM.

En el proceso de cuantificación vectorial se asignó a cada vector de parámetros característicos su correspondiente representante dentro del codebook de centroides. Para el entrenamiento del HMM de cada palabra se realiza la cuantificación de sus repeticiones, entonces los datos de entrada para el entrenamiento son las repeticiones cuantificadas y la longitud de cada repetición. En el Anexo A se muestra el código del programa de la cuantificación vectorial.

Para dar solución al problema del aprendizaje o entrenamiento se buscó maximizar la probabilidad que una secuencia de observaciones haya sido generada por un HMM optimizando sus parámetros.

Se realizó una estimación de los parámetros iniciales distinta para cada uno de los HMM, dependiendo de la cantidad de estados, de la duración de la palabra y de cómo fue reentrenado el modelo al comprobar su funcionamiento con el reconocedor. Los parámetros A y B son probabilidades almacenadas en forma de matrices. Las primeras estimaciones se hicieron con matrices equiprobables y aparecieron probabilidades muy pequeñas, posteriormente se emplearon matrices con valores arbitrarios que se fueron refinando a medida que se volvían a entrenar los modelos.

Por la topología empleada, el entrenamiento debe realizarse con múltiples observaciones $O = O_1, O_2, O_3, \dots, O_k$. Se implementó el algoritmo Forward-Backward (adelante-atrás) para múltiples observaciones, también el algoritmo Baum-Welch para re estimar los parámetros del HMM.

Antes de describir el proceso de reestimación, necesitamos conocer:

- El número esperado de transiciones desde el estado i en O.
- El número esperado de transiciones desde el estado i al estado j en O.

PROCESO DE REESTIMACIÓN

El algoritmo del procedimiento reiterativo es el siguiente:

Se selecciona un modelo inicial elegido aleatoriamente.

Se calculan las transiciones y los símbolos de emisión más probables para el modelo inicial. Se crea un nuevo modelo en el que se mejora la probabilidad de las transiciones y símbolos determinados en el primer paso, de tal manera que el modelo nuevo tendrá una probabilidad mayor que el modelo.

Este algoritmo se repite varias veces hasta que no existe mejora entre el modelo anterior y el actual.

Con los cálculos anteriores se realizó el entrenamiento siguiendo los pasos:

1.- Estimar un conjunto inicial de parámetros {a,b}.
2.- Calcular las re-estimaciones de a y b de acuerdo a las fórmulas.
3.- Sustituir los antiguos valores de a y b por los valores re-estimados.
4.- Aplicar las pruebas de optimización.

6.1.3 RECONOCIMIENTO DE PALABRAS AISLADAS

La implementación de la etapa de reconocimiento se llevó a cabo realizando los pasos mostrados en la figura 27.

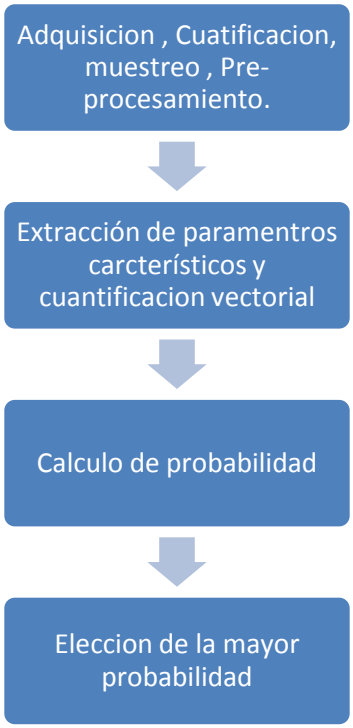

Figura 32. Proceso de reconocimiento de palabras aisladas.

La adquisición y muestreo de la señal de voz, cuantificación, filtro pasa bajos, pre-procesamiento y extracción de parámetros característicos se realizan utilizando las mismas funciones descritas en las secciones anteriores.

En el caso de la cuantificación vectorial, el vector de entrada (vector de características de palabra a reconocer) se compará con cada vector prototipo del codebook usando medida de distancia. Luego el vector de entrada se sustituye por el índice del vector prototipo que más se le parece.

CALCULO DE PROBABILIDAD

Se calcula la probabilidad de que la secuencia haya sido generada por alguno de los modelos de las palabras entrenadas. Para ello se utilizan el algoritmo Forward-Backward (adelante-atrás) y todos los parámetros del HMM calculados en la sección anterior.

6.2 INTERFAZ DEL PROGRAMA

MATLAB dispone de la posibilidad de crear una interfaz gráfica amigable que permita a un usuario ejecutar un programa de una manera sencilla. A continuación se explica con detalle la interfaz gráfica del sistema de reconocimiento del habla utilizando modelos ocultos de Markov.

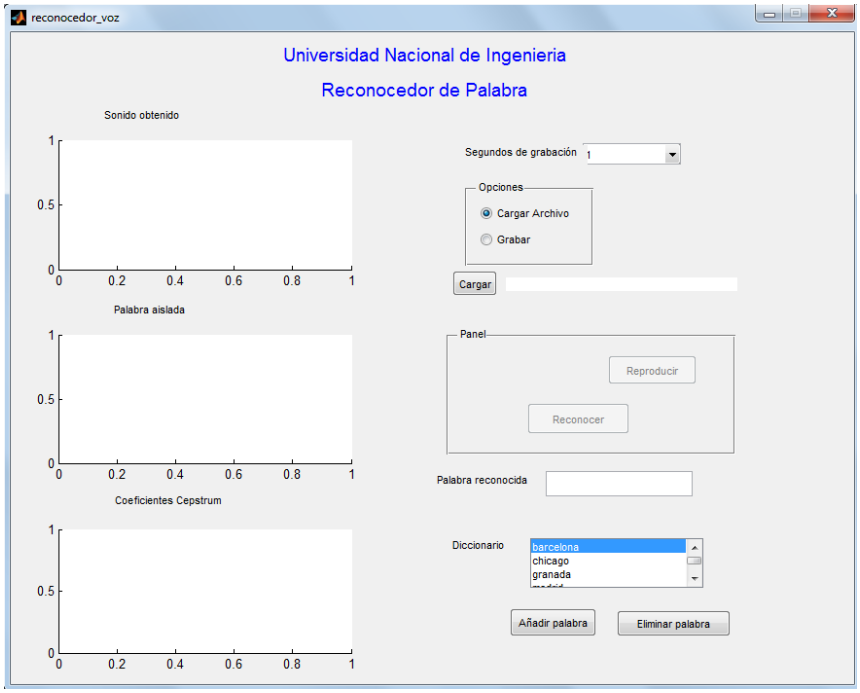

Figura 33. Ventana de reconocimiento para interfaz de usuario

Al ejecutar el programa nos aparece la ventana que se muestra en la figura 28, como se puede observar es posible elegir dos formas de introducir la señal de voz, una cargando un archivo .WAV y otra grabando. En el caso de grabar la señal de voz es posible elegir los segundos de grabación dependiendo de la palabra a grabar.

En el caso que no se detecte sonido la aplicación nos advertirá de que no se ha detectado sonido. En la figura también se puede observar que existen tres ejes, en el eje superior nos mostrará toda la señal, en el eje de en medio se mostrará la señal recortada es decir la palabra aislada donde ha detectado los bordes. El eje inferior mostrara la gráfica cepstrales.

Una vez cargada o grabada la palabra se activará la posibilidad de escucharla y de reconocerla si elegimos reconocerla el programa reconocerá la palabra aislada entre las palabras que tiene en el diccionario y se podrá ver en la caja de texto con la etiqueta “palabra reconocida”. Las palabras que contiene el diccionario podemos verlas en una lista que hay en la parte inferior izquierda de la ventana. También está disponible la posibilidad de eliminar una palabra del diccionario, para ello seleccionamos la palabra que queremos eliminar y hacemos clic sobre “Eliminar palabra”.

Otra opción que nos ofrece el programa es la de entrenar otra palabra y añadirla al diccionario, para llevar a cabo esto se debe elegir la opción “Añadir palabra” de la ventana principal, una vez elegida esta opción aparecerá la ventana que se puede observar en la figura 29.

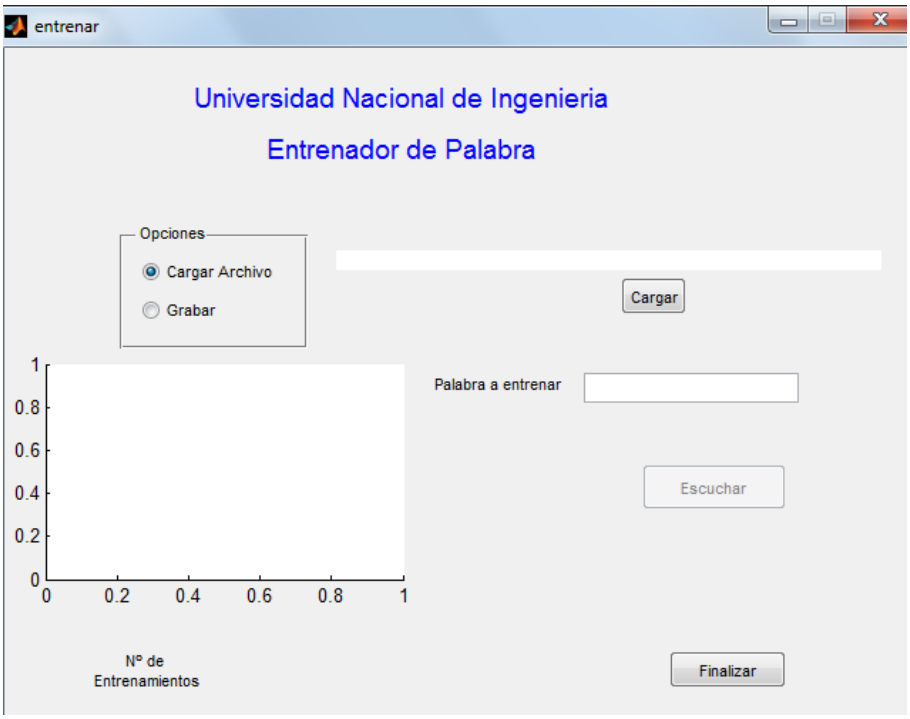

Figura 34. Ventana de entrenamiento para interfaz de usuario

El primer paso es escribir la palabra que deseamos añadir, luego tenemos dos opciones para añadir dicha palabra, podemos cargar un archivo de sonido .WAV o grabar. En caso de seleccionar grabar la duración de la grabación es de un segundo.

Una vez que la palabra ha sido cargada o la grabación ha concluido, se activan los botones “Siguiente” y “Escuchar”. Si hacemos clic en la primera opción el programa nos permite cargar o grabar otra repetición de esa palabra. Si hacemos clic en escuchar, el programa reproduce la palabra grabada y nos pregunta si deseemos agregarla al diccionario, en la parte inferior izquierda se muestra el número de veces que se ha repetido dicha palabra. Una vez finalizado el proceso, hacemos clic en finalizar, el programa preguntará si deseamos entrenar datos y luego vuelve a la ventana principal.

VI. EXPERIMENTOS Y RESULTADOS

Capítulo 7: MÉTODO DE VERIFICACIÓN

La precisión del reconocimiento es la medida más importante y directa del rendimiento de un sistema de reconocimiento de habla. Para garantizar que se ha seleccionado el mejor método de parametrización y los valores iniciales que garanticen el óptimo funcionamiento de HMM, se realizaron los experimentos descrito a continuación.

Para llevar a cabo dichos experimentos se utilizó la base de datos MAMI (Database of isolated spoken words through a mobile).

Las palabras de la base de datos están divididas en 6 categorías:

1. Naturaleza: rio, playa, parque, nieve, montana, lago, isla, cascada.

2. Ciudad: Zaragoza, Sevilla, Paris, Madrid, Londres, Granada, Barcelona, Chicago.

3. Personas: Zapatero, Raúl, Nuria, Pablo, Bill, Carlos, Clinton, Alierta.

4. Eventos: navidad, fiesta, cumpleaños, espectáculo, boda, bautizo, barbacoa.

5. Familia: tía, primo, padre, madre, hermano, bebe, abuelo, amigo.

Experimento #1: Comparación entre los métodos de parametrización estudiados.

Los métodos de extracción de parámetros presentados en este documento en el capítulo 3 fueron: el predictor lineal y MFCC.

Con el fin de determinar cuál método ofrece mayor precisión, se seleccionaron 5 palabras de la base de datos, se tomaron 70 muestras de cada palabra para entrenamiento y 27 para prueba del sistema. Las palabras seleccionadas fueron: Barcelona, Chicago, Granada, Madrid, Londres.

Experimento # 2: Verificación de parámetros iniciales para HMM

Como se explicó en el capítulo 5, no hay un método analítico para determinar los parámetros iniciales de los Modelos Ocultos de Markov de cada palabra, por lo tanto se realizaron pruebas variando los siguientes parámetros:

- Número de secuencias de observación para el entrenamiento de los HMM de cada palabra.
- Cantidad de estados del modelo.
- Número de centroides.

Se tomó un total de 350 muestras, 70 muestras de cada palabra y a partir de estas pronunciaciones se calculó el error en el reconocimiento de cada palabra. Las modificaciones se realizaron en el algoritmo de entrenamiento de los HMM y el algoritmo de cálculo del codebook.

Prueba 1: Se realizó modificando la cantidad de secuencias de observación para el entrenamiento del HMM de cada palabra, con matrices *aij* y *bjk* aleatorias, modelos de 6 estados y un codebook de 64 centroides generado con la base de datos MAMI.

Prueba 2: Se realizó cambiando el número de estados de los HMM de cada palabra en su entrenamiento, con matrices *aij* y *bjk* aleatorias, 70 secuencias de observación y un codebook de 64 centroides generado con la base de datos MAMI.

Prueba 3: Esta prueba se realizó modificando el tamaño del codebook generado con la base de datos MAMI, con matrices *aij* y *bjk* aleatorias, 70 secuencias de observación y modelos de 6 estados.

Experimento # 3: Extender el vocabulario.

El tamaño del vocabulario de un sistema de reconocimiento de voz afecta la precisión del mismo. A menor vocabulario mayor precisión y viceversa. Con el fin de verificar el correcto funcionamiento de nuestro sistema, se diseñó el siguiente experimento. El entrenamiento se realizará con 20 palabras, 70 repeticiones para entrenamiento y 27 para prueba. Las muestras para el experimento se tomaron de la base de datos MAMI. Para la extracción de característica se utilizó la técnica MFCC, el codebook de 64 centroides, 6 estados para el entrenamiento de los HMM de cada palabra y matrices *aij* y *bjk* aleatorias.

El vocabulario constó de las siguientes palabras: Zaragoza, Sevilla, Paris, Madrid, Londres, Granada, Barcelona, Chicago, rio, playa, parque, nieve, montana, lago, isla, cascada, abuelo, amigo, madre, bautizo.

Experimento # 5: Agregar ruido.

Los experimentos anteriores se realizaron utilizando micrófono y con la menor cantidad de ruido posible. Sin embargo la relación señal a ruido es de suma importancia para la precisión del reconocimiento de voz. Una baja relación señal a ruido (SNR) disminuye el desempeño del reconocedor.

En el capítulo 2 de este documento se describen los filtros que se agregaron al programa de reconocimiento de voz para aumentar el SNR. Es importante comprobar el funcionamiento del reconocedor en ambientes reales, para ello haremos uso de una base datos MAMI y agregaremos ruido blanco gaussiano a cada palabras.

El entrenamiento se realizará con 5 palabras, 70 repeticiones para entrenamiento (sin ruido) y 10 para prueba (con ruido). Para la extracción de característica se utilizara la técnica MFCC, el codebook será 64 centroides, 6 estados el entrenamiento de los HMM de cada palabra y matrices *aij* y *bjk* aleatorias.

CAPITULO 8: RESULTADOS

Los resultados que se presentan fueron obtenidos con la ejecución de los experimentos que se documentaron en la sección anterior.

Experimento #1: Comparación entre los métodos de parametrización estudiados.

Se realizaron pruebas comparativas entre los dos métodos de parametrización estudiados: MFCC y LPC usando como criterio de evaluación el porcentaje de acierto en el reconocimiento de las palabras. El resultado obtenido para las muestras de voz correspondientes se muestra en la tabla2.

	Barcelona	Chicago	Granada	Madrid	Londres	
	Error%	Error %	Error %	Error %	Error %	ETP
MFCC	11.11%	7.41%	7.41%	3.70%	3.70%	6.67%
LPC	11.11%	7.41%	48.15%	7.41%	11.11%	17.04%

Tabla 2. Error en el reconocimiento de palabras aisladas modificando técnica de extracción de parámetros LPC y MFCC

La celda sombreada en la tabla 2 representa el porcentaje de error que tuvo el algoritmo LPC para la palabra “Granada”. Según este resultado, de 27 muestras de voz correspondientes a dicha palabra, este algoritmo identifico 14 de ellas correctamente, lo que corresponde a un 51.85 % de aciertos para esta palabra.

De la tabla 2 se puede observar que el método con el menor porcentaje de error total es el MFCC, por lo que se utilizó esta técnica de parametrización para la nuestro sistema de reconocimiento de habla. Este resultado también fue encontrado en la revisión bibliográfica sobre el uso de la técnica MFCC para la parametrización de señales en aplicaciones para el reconocimiento de habla. [20]

Experimento # 2: Verificación de parámetros iniciales para HMM,

Prueba 1: El objetivo de esta prueba fue determinar la cantidad de secuencias de observación, para el entrenamiento del HMM de cada palabra, que permita tener un menor porcentaje de error. Los resultados se muestran a continuación.

	5 secuencias de observación	20 secuencias de observación	50 secuencias de observación	70 secuencias de observación
Palabra	***Error %***	***Error %***	***Error %***	***Error %***
Barcelona	29.63%	40.74%	25.93%	11.11%
Chicago	66.67%	62.96%	44.44%	7.41%
Granada	92.59%	62.96%	55.56%	7.41%
Madrid	96.30%	22.22%	33.33%	3.70%
Londres	77.78%	88.89%	33.33%	3.70%
Error total promedio (ETP)	72.59%	55.56%	38.52%	6.67%

Tabla 3. Error en el reconocimiento de palabras aisladas modificando el número de secuencia de observación de los HMM de entrenamiento de cada palabra

Observando los resultados de la tabla 3, se concluye que el error promedio total más bajo fue el obtenido con 70 secuencias de observación en el entrenamiento de los HMM. Con un número elevado de secuencias de observación para el entrenamiento del modelo de cada palabra se obtiene una mayor variabilidad, capacitando al modelo a responder satisfactoriamente a cualquier modificación en el estilo de habla. Sin embargo, con más de 70 secuencias de observación no se obtiene una reducción del error promedio considerable y se aumenta el tiempo de entrenamiento.

Prueba 2: El objetivo de esta prueba era calcular el número de estados de los HMM para el entrenamiento de cada palabra que permita obtener el menor porcentaje de error. Los resultados se muestran en la tabla 4.

	5 estados	6 estados	7 estados	8 estados	9 estados	10 estados
Palabra	***Error %***	***Error %***	***Error %***	***Error %***	***Error %***	***Error %***
Barcelona	25.93%	11.11%	29.63%	37.04%	40.74%	25.93%
Chicago	14.81%	7.41%	25.93%	25.93%	29.63%	29.63%
Granada	14.81%	7.41%	29.63%	40.74%	85.19%	44.44%
Madrid	18.52%	3.70%	18.52%	44.44%	44.44%	37.04%
Londres	22.22%	3.70%	33.33%	25.93%	14.81%	29.63%
ETP	19.26%	6.67%	27.41%	34.81%	42.96%	33.33%

Tabla 4. Error en el reconocimiento de palabras aisladas modificando el número de estados de los HMM de entrenamiento de cada palabra.

Como se puede observar en la tabla 4, los mejores resultados en cuanto a disminución total del error se obtuvieron entrenando los HMM de cada palabra con 6 estados.

Prueba 3: El objetivo de esta prueba fue calcular el tamaño del codebook que permitiera disminuir el porcentaje de error. Los resultados obtenidos se muestran en la siguiente tabla.

	32 centroides	**64 centroide**	**128 centroides**
Palabra	***Error %***	***Error %***	***Error %***
Barcelona	44.44%	11.11%	7.41%
Chicago	14.81%	7.41%	7.41%
Granada	44.44%	7.41%	7.41%
Madrid	37.04%	3.70%	3.70%
Londres	51.85%	3.70%	3.70%
EPT	38.52%	6.67%	5.93%

Tabla 5. Error en el reconocimiento de palabras aisladas modificando el número de centroides en la generación del codebook.

Observando los resultados de la tabla, el mejor funcionamiento se obtiene utilizando un codebook con 128 centroides característicos.

Aun cuando este tamaño de codebook permite obtener un menor porcentaje de error existe otra variable que debe ser tomada en cuenta "el tiempo de ejecución". Para calcular el tiempo de ejecución de cada instrucción se utilizó una herramienta de MATLAB llamada Profiler. La medida del tiempo de ejecución se tomó desde el final del proceso de adquisición hasta la elección de la máxima probabilidad. La tabla siguiente muestra el tiempo de ejecución para el caso de 68 y 128 centroides.

	64 centroide	128 centroides
Palabra	**Tiempo de ejecución (s)**	**Tiempo de ejecución (s)**
Barcelona	1.079	1.903
Chicago	1.047	2.032
Granada	1.219	2.084
Madrid	1.172	2.045
Londres	1.056	2.045
Tiempo promedio	1.1146	2.0218

Tabla 6.Tiempo de ejecución del programa de reconocimiento generando el codebook con 64 centroides y 128 centroides.

Como se observa en la tabla 6 una disminución tan pequeña en el error promedio no justifica una elevación del tiempo de ejecución del 81.39 % así que el codebook que mejor cumplió con el compromiso entre funcionamiento y velocidad de ejecución fue el implementado con 64 centroides característicos.

A través de los resultados de los experimentos anteriores, se comprueba que los métodos y valores seleccionados, basados en investigación teórica, para el diseño de nuestro sistema de reconocimiento de voz son los que permiten tener la mayor precisión. En resumen el sistema presenta las siguientes características:

- 13 coeficientes característicos.
- Codebook de 64 centroides.
- Coeficientes MFCC.
- 70 secuencias de observación para el entrenamiento de los HMM de cada palabra.
- 6 estados el entrenamiento de los HMM de cada palabra.

La eficiencia del algoritmo de palabras aisladas utilizando HMM es de **93.33%**. Se obtuvo una eficiencia muy alta teniendo en cuenta que es necesario que el diseño cumpla un compromiso entre velocidad de procesamiento y eficiencia en el reconocimiento.

Experimento # 3: Extender el vocabulario

El objetivo de este experimento fue verificar el funcionamiento de nuestro sistema de reconocimiento de voz para un vocabulario de mayor tamaño. El vocabulario constará de las siguientes palabras: Zaragoza, Sevilla, Paris, Madrid, Londres, Granada, Barcelona, Chicago, rio, playa, parque, nieve, montana, lago, isla, cascada, abuelo, amigo, madre, bautizo. Las condiciones para este experimento se describieron en la sección anterior y los resultados se muestran a continuación.

Palabra	Error %
Abuelo	14.81%
Amigo	14.81%
Barcelona	14.81%
Bautizo	14.81%
Cascada	14.81%

Chicago	14.81%
Granada	11.11%
Isla	14.81%
Lago	14.81%
Londres	18.52%
Madre	14.81%
Madrid	18.52%
Montana	7.41%
Nieve	11.11%
Paris	14.81%
Parque	14.81%
Playa	14.81%
Rio	11.11%
Sevilla	18.52%
Zaragoza	14.81%
ETP	14.44%

Tabla 7. Error de reconocimiento de palabra al incrementar el vocabulario

En el experimento anterior se utilizó un vocabulario de 5 palabras y la eficiencia del sistema fue de **93.33%**, al cuadruplicar el número de palabras en el vocabulario la eficiencia del sistema disminuyó a 85.56%. Esto demuestra que el sistema es flexible y puede tener un vocabulario mayor sin afectar de manera significativa la eficiencia del mismo.

Experimento # 4: Agregar ruido.

El objetivo de este experimento es medir la precisión que posee el sistema de reconocimiento de voz cuando a se le agrega ruido a las palabras de prueba. Para agregar ruido utilizamos la función “awgn” de MATLAB con una relación señal a ruido de 10. Los resultados se muestran en la tabla 8.

Palabra	Error %
Barcelona	22.22%
Chicago	18.52%
Granada	14.81%
Madrid	18.52%
Londres	7.41%
ETP	16.30%

Tabla 8. Error de reconocimiento de palabra al agregar ruido

La efectividad del sistema bajo condiciones de ruido es 83.7%. Lo cual representa una disminución de 9.63% con respecto al valor calculado anteriormente.

VI. CONCLUSIONES

El presente proyecto de final de carrera constó de cuatro fases para su elaboración, una primera fase para la búsqueda de información, una segunda fase para el diseño e implementación de un sistema de reconocimiento del habla, una tercera fase para la prueba del sistema y una cuarta fase para la elaboración de la presente memoria. La fase de diseño e implementación se dividió en cuatro etapas: pre-procesamiento, parametrización, cuantificación vectorial y entrenamiento de modelos HMM.

La etapa de pre-procesamiento es de gran importancia ya que en esta se resaltan y mejoran sustancialmente las características de la señal a parametrizar. Dentro de esta fase, la segmentación y aplicación de la ventana son de vital importancia para poder asumir que la señal de voz es estacionaria en esas ventanas y aplicar técnicas que permitan el análisis de la misma. La ventana utilizada fue la Hamming, ya que ofrece el equilibrio entre la resolución en tiempo y frecuencia de la señal de voz requerido para la aplicación.

Para la etapa de parametrización o extracción de parámetros, se consideró desde un principio el uso de técnicas basadas en el cálculo de los coeficientes cepstrales, debido a su gran uso en aplicaciones que involucran el procesamiento de señales de voz y por la gran variedad de combinaciones que entre ellos se pueden obtener. Como resultado del estudio comparativo realizado entre LPC y MFCC, se escogió la técnica del MFCC porque su porcentaje acierto es del 93.33%. Además, la técnica MFCC permite agudizar las características de la señal e imitar el oído humano, basándose en sus anchos de banda críticos. Los parámetros MFCC funcionan de manera excelente con un número reducido de coeficientes, en nuestro diseño se utilizaron 13 coeficiente que permitieron obtener un modelamiento eficiente de las características del tracto vocal.

La cuantificación vectorial fue la técnica utilizada comprimir la cantidad de datos a procesar por medio del uso de un codebook con un número relativamente pequeño de centroides. Cada centroide adquiere una característica fonética que relaciona

directamente a los vectores de observación, clasificando estos según la información espectral que representen. Sin embargo en la generación del codebook se debe llegar a un compromiso entre error por distorsión, almacenamiento y costo computacional. Según los resultados obtenidos el codebook generado con 64 centroides es el que mejor cumple este compromiso.

La eficiencia del algoritmo de palabras aisladas utilizando Modelos Ocultos de Markov es de 93.33%. Se observó que el entrenamiento de los HMM hecho a partir de múltiples observaciones (varias repeticiones de la misma palabra) es capaz de producir buenos resultados con una iteración en el algoritmo de avance y retroceso. El modelo en proceso de entrenamiento se va acercando al modelo definitivo a medida que se reestiman parámetros con cada nueva repetición de la palabra. Inicialmente se probó con un número pequeño de observaciones que se incrementaron para aumentar el número de veces que se hacía el proceso de re-estimación y así encontrar los parámetros del modelo. El modelo definitivo se alcanza después de probar el reconocedor en funcionamiento, si el error de reconocimiento se acerca a cero no se incrementa el número de observaciones y se archivan los parámetros. Se emplearon 70 repeticiones.

En cuanto al lenguaje de programación utilizado para la aplicación de reconocimiento de voz, la elección fue MATLAB, ya que este posee una amplia gama de módulos para procesamiento de señales, en comparación con otros lenguajes de programación como visual basic, C++, entre otros.

Se utilizó la base de datos MAMI para verificar la efectividad del sistema para distintos usuarios. El sistema se entrenó con 5 palabras, cada uno de los 14 usuarios pronuncio la palabra 5 veces, es decir, se utilizaron 70 repeticiones por palabra. Para verificar, se utilizaron las mismas cinco palabras pronunciadas tres veces por 9 usuarios diferentes.
El desarrollo de la presente tesis ha dejado abiertas varias líneas de desarrollo para el futuro que se exponen en el siguiente apartado.

VII. RECOMENDACIONES

Las siguientes recomendaciones son para la continuación de este trabajo y persiguen obtener un proyecto más integral:

Implementar el prototipo de reconocimiento de palabras aisladas sobre un sistema autónomo (DSP, DSPIC o FPGAA) que aumente la eficiencia en cuanto a tiempo de procesamiento.

Variar la cantidad de estados por palabra para los modelos ocultos de Markov.

Utilizar otro método de reconocimiento diferente al utilizado en este trabajo, como los modelos ocultos de Markov continuos, alineamiento temporal dinámico o un híbrido entre modelos ocultos de Markov y redes neuronales.

Para optimizar el funcionamiento del prototipo de reconocimiento y su aplicación en el comando de un sistema móvil se recomienda implementar un sistema de reconocimiento de palabras clave dentro de un discurso continuo.

Un sistema de ayuda a discapacitados implementado en un computador, puede ser más que reconocimiento de voz por lo que se recomienda complementar el prototipo con otras funciones como reconocimiento de imágenes y seguimiento de trayectorias.

VIII. BIBLIOGRAFÍA

[1] J. &. H. W. Holmes, Speech Systhesis and Recognition, London: Tailor & Francis, 2001.

[2] R. Cole, «Survey of the states of the art in human lenguage technology,» Cambridge University Press, 1997.

[3] C. Uribe, «Motor computacional de reconocimiento de voz: Principios basicos para su construccion,» Universidad Tecnologica de Pereira, Colombia, 2007.

[4] R. C. Arias, «Reconocimiento de voz,» *Instituto Costarricense de Electricidad,* 2002.

[5] B. Moore, Frequency Selectivity in Hearing, London: Academic Press, 1986.

[6] G. Fant, Acoustic Theory of Speech Production, Gravenhage: Mounton & Co., 1996.

[7] L. &. J. B. Rabiner, Fundamentals of Speech Recognition, N.J.: Prentice Hall, 1993.

[8] X. A. A. &. H. H. Huang, Spoken Language Processing - A Guide to Theory Algorithm, and System Development, N.J.: Prentice Hall PTR, 2001.

[9] S. Gelfand, Hearing - An Introduction to Psychological and Physiological, New York: Marcel Dekker, 2004.

[10] S. Iosu, «Speech Recognition,» 24 Junio 1999. [En línea]. Available: www.euskalnet.net/iosus/speech/recog2.html. [Último acceso: 28 Septiembre 2012].

[11] B. Moore, Cochlear Hearing Loss, London: Whurr Publishers Ltd, 1998.

[12] X. A. a. N.Oliver, «MAMI: Multimodal annotations on a mobile phone,» *Proc. Mobile HCI,* 2008.

[13] F. Z. Marcos, Tratamiento Digital de Voz e Imagen, Alfaomega, 2001.

[14] G. V. Ramírez, «SISTEMA DE RECONOCIMIENTO DE VOZ EN MATLAB,» Universidad de San Carlos, Guatemala, 2008.

[15] D. Childers, Speech Processing and Synthesis toolboxes, N.J.: Wiley & Sons, inc,, 2000.

[16] J. E. Hingant, «Impletación y evaluación de métodos de detección del tono de la voz,»

Universidad de Alicante, Alicante, 2009.

[17] M. D. G. Proakis John G., Digital Signal Processing principle, algorithms, and application, Third Edition, New Jersey: Prentice Hall, 1996.

[18] J. H. J. J. P. John Deller R., «Discrete-Time Processing of Speech Signals,» *IEEE Press,* nº ISBN 0-7803-5386-2.

[19] J. Makhoul, Spectral Analysis of Speech by Linear Prediction, Cambridge: Bolt Beranek and Newman Inc, 1972.

[20] C.-F. C. C.-S. C. a. K.-P. P. Wei Han, «An Efficient MFCC Extraction Method in Speech Recognition,» *Department of Electronic Engineering, The Chinese University of Hong Kong ,* 2006.

[21] J. Frederick, «Statistical Methods for Speech Recognition,» *MIT press,* nº ISBN 0-262-10066-5, 1998.

[22] M. F. &. D. Rodríguez, «Estudio comparativo de diferentes distancias en sistemas basados en VQ para identificación automática de locutores,» Eupmt, Pamplona , 2005.

[23] L. R. B. Juang, «An Introduction to Hidden Markov Models,» *ASSP Magazine, IEEE,* vol. 3, nº 1, pp. 4-16, 1986.

[24] L.R.Rabiner., «A Tutorial on Hidden Markov Models and Selected Applications in Speech Recognition,» *Proc. of IEEE,* vol. 77, nº 2, pp. 257-286, 1989.

[25] H. H. Monson, Statistical Digital Signal Processing and Modeling, Toronto: John Wiley & Sons Inc., 1996.

[26] Z. X. R. R. P. Mammone Richard J., «Robust Speaker Recognition,» *IEEE SP Magazine,* vol. 13, nº ISSN 1053-5888, pp. 58-71, 1996.

[27] B. R. a. B. S. Davis K.H., «Automatic Recognition of Spoken Digits,» *J. Acoust. Soc. Am.,* vol. 6, nº 24, pp. 637-642, 1952.

[28] F. J. a. F. C.D, «Results Obtained From a Vowel Recognition Computer Program,» *J. Acoust. Soc. Am.,* vol. 11, nº 31, pp. 1480-1489, 1959.

[29] E. J. Baum L.E., «An inequality with applications to statistical estimation for probabilistic functions of Markov process and to model for ecology,» *Bullitin of American Mathematical*

Society, vol. 73, pp. 360-363, 1967.

[30] Y. Meng, «Speech Recognition on DSP: Algorithm Optimization and Performance Analysis,» 2004, pp. 4-6.

[31] I. V. Espinoza, «Aplicacion en reconocimento de voz utilizando HTK,» Pontificia Universidad Javeriana, Bogota, Colombia, 2005.

[32] J. P. Carmona, «Desarrollo de un sistema de reconocimiento de voz utilizando mixturas gaussianas,» Instituto Politecnico Nacional, DF, Mexico, 2006.

XI. Anexos

A. Código de programación de la aplicación encargada del reconocimiento de habla.

1. Función Grabar

```
function =grabar(segundos),
numero_buffer=1;
ai=analoginput('winsound');
addchannel(ai, 1);

set(ai, 'SampleRate',8000);
set(ai, 'SamplesPerTrigger',6400);
set(ai, 'TriggerRepeat', 1);
set(ai, 'TriggerType', 'Immediate');
set(ai, 'TimerPeriod', 0.1);
Fs=8000;
toffset=40e-3;
offset=Fs*40*10^(-3);

valor=1;
while valor<=segundos
%if flag==1,

 start(ai);

 numero_de_muestras = ai.SamplesAvailable;
 while numero_de_muestras<6400,
    numero_de_muestras = ai.SamplesAvailable;
 end
 stop(ai);

 [d,t] =getdata(ai);
 longitud=length(d);
 y=d;
 if valor==1,
   senyal((valor):((valor+longitud-1)))= y;
 else
    senyal(((valor-1)*longitud):(((((valor-1)*longitud)+longitud-
1))))=y;
 end
 valor=valor+1;
end
plot(senyal);
```

2. Función Cargar

```
[FileName,PathName,FilterIndex] = uigetfile({'*.wav'},'Open Audio
File');
a=strcat(PathName, FileName);
set(handles.direc,'string',a);

[senyal,fs] = wavread(a);
senyal=senyal';
wavplay(senyal,fs)
l=length(senyal);

t=0:1/l:1;
l=length(t);
t=t(1:l-1);
plot(t,senyal);
```

3. Función de Detección de punto final.

```
x=x';
frame_length = round(fs .* fsize);
N= frame_length - 1;

for b=1 : frame_length : (length(x)- frame_length),
    y1=x(b:b+N);
    y = filter([1 -.9378], 1, y1);      %Filtro de preenfasis

    msf(b:(b + N)) = func_vd_msf (y);
    zc(b:(b + N)) = func_vd_zc (y);
    pitch_plot(b:(b + N)) = pitch_detection (y,fs);

end

thresh_msf = (( (sum(msf)./length(msf)) - min(msf)) .* (0.67) ) +
min(msf);.
voiced_msf =  msf > thresh_msf;
thresh_zc = (( ( sum(zc)./length(zc) ) - min(zc) ) .* (1.5) ) +
min(zc);
voiced_zc = zc < thresh_zc;

thresh_pitch = (( (sum(pitch_plot)./length(pitch_plot)) -
min(pitch_plot)) .*(0.5) ) + min(pitch_plot);
voiced_pitch = pitch_plot > thresh_pitch;

for b=1:(length(x) - frame_length),
if voiced_msf(b) .* voiced_pitch(b) .* voiced_zc(b) == 1,
voiced(b) = 1;
else
voiced(b) = 0;
end
end
```

4. Función MFCC

```
mfc2(x,desplazamiento,long_ventana,fs)
%x=señal de entrada
%long_ventana=longitud de la ventana Hamming
%desplazamiento=desplazamiento de cada cuadro, para que exista
solapamiento  long_ventana>desplazamiento
%fs=frecuencia de muestreo
%coef=matriz de coeficientes cepstrales
coef=[];
fftSize=512;

desp=desplazamiento*fs;
window=long_ventana*fs;


frame=length(x);
tamany = round(frame/desp);
for (i=1:tamany)

    if i*desp+window>=frame,
      y(i*desp+1:frame)=0;
    else
    y1=x((i*desp)+1:(i*desp)+window-1);
    y3= y1 .* hamming(length(y1));
    y = filter([1 -.95], 1, y3);
    y2=abs(fft(y,fftSize));

      banco_filtros=frecuencias(fs,fftSize,frame);
      cepstrum=log10(banco_filtros*y2);
      coef(i,:)=dct(cepstrum);
      coef=coef(:,1:13)';
```

5. Función HMMRecognition

```
L = length(ytrain);
A = rand(S,S); A = A./repmat(sum(A),S,1);
B = rand(K,S); B = B./repmat(sum(B),K,1);
pi1 = rand(S,1); pi1 = pi1/sum(pi1);

loglike = zeros(maxiter,1);
loglike_old = -inf;
loglike_dif = inf;

k = 0;
while (k < maxiter) & (loglike_dif >= tol)
  k = k + 1;
  pi_c = 0; A_c = 0; B_c = 0;
  loglike_c = zeros(L,1);

  for (l = 1:L)
    y = ytrain{l};
    T = length(y);

    alpha = zeros(S,T);
    beta  = zeros(S,T);
    scale = zeros(T,1);

    alpha(:,1) = pi1.*B(y(1),:)';
    scale(1)   = sum(alpha(:,1));
    alpha(:,1) = alpha(:,1)/scale(1);
    for (t = 2:T)
      alpha(:,t) = A*alpha(:,t-1).*B(y(t),:)';
      scale(t)   = sum(alpha(:,t));
      alpha(:,t) = alpha(:,t)/scale(t);
    end

    beta(:,T) = 1/scale(T);
    for (t = (T-1):-1:1)
      beta(:,t) = A'*(beta(:,t+1).*B(y(t+1),:)')/scale(t);
    end

    loglike_c(l) = sum(log10(scale));

    gamma_c = (alpha.*beta).*repmat(scale',S,1);
    A_c  = A_c  + (alpha(:,1:T-1)*(B(y(2:T),:)'.*beta(:,2:T))').*A';
    B_c  = B_c  + (gamma_c*(repmat([1:K],T,1)==repmat(y(:),1,K)))';
    pi_c = pi_c + gamma_c(:,1);
  end

  loglike(k)  = sum(loglike_c);
  loglike_dif = loglike(k) - loglike_old;
  loglike_old = loglike(k);

  pi1 = pi_c/sum(pi_c);
```

```
  A    = A_c./repmat(sum(A_c),S,1);
  B    = B_c./repmat(sum(B_c),K,1);

  fprintf(1,'... Log likelihood: %6.3f\n',loglike(k));
end

loglike = loglike(2:k);

for (j = 1:R)
    logp(j,i) = hmmlogp(yk,A_m{j},B_m{j},pi_m{j});
end
logp
[loglike,guess] = max(logp);
```

6. Función HMM Codebook

```
R = length(data);
Y = [];
for (i = 1:R)
  L = length(data{i});
  for (j = 1:L)
    file_s = data{i}{j};
    fs=frec{i}{j};
    s=file_s;

    vsize=(1/fs)*320;
    dsize=(1/fs)*80;

    y = mfc2(s,dsize,vsize,fs);
    y=y(:,1:13)';
    Y = [Y y];

  end
end

[Yc,c,errlog] = kmeans(Y',Kmax,5000);
cb = Yc(unique(c),:)';
[N,K] = size(cb);
[N,T] = size(Y);
dist  = sqrt(errlog(end))/T;
```

7. Función HMM Entrenamiento

```
[dim,K] = size(cb);
A_m  = cell(0,0);
B_m  = cell(0,0);
pi_m = cell(0,0);
loglike_m = cell(0,0);
R = length(data);

 for (i = 1:R)
  L = length(data{i});
  ytrain = cell(L,1);
  for (j = 1:L)
    file_s = data{i}{j};
    fs=frec{i}{j};

    long=length(data{i}{j});

    s=file_s;

    vsize=(1/fs)*320;
    dsize=(1/fs)*80;

    y = mfc2(s,dsize,vsize,fs);
    y=y(:,1:13)';
    [dim,T] = size(y);

    yk = zeros(T,1);
    for (t = 1:T)
      [dist,k] = min(sum((cb-repmat(y(:,t),1,K)).^2));
      yk(t) = k;
    end;
    ytrain(j) = {yk};
  end
    [A,B,pi1,loglike] = hmmfb(ytrain,S,K,maxiter,tol);
    A_m=[A_m A]
    B_m=[B_m B];
    pi_m =[pi_m pi1];
    loglike_m = [loglike_m loglike]
 end
end
```