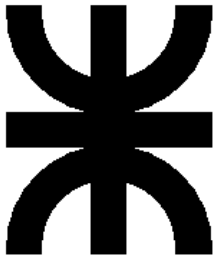


UNIVERSIDAD TECNOLÓGICA NACIONAL
FACULTAD REGIONAL SAN NICOLAS

INGENIERÍA EN ELECTRÓNICA

PROBLEMA DE INGENIERÍA

TÉCNICAS DIGITALES III

RECONOCIMIENTO DE VOZ PARA APLICACIÓN EN DOMÓTICA

Integrantes**:**

- Bellesi Facundo
- Ortiz Flavio

Docentes**:**

- Profesor: Poblete Felipe
- Auxiliar: González Mariano

AÑO 2008

INDICE

OBJETIVOS DEL TRABAJO ***3***

MATERIAS INTEGRADAS ***3***

POSIBLES APLICACIONES ***3***

BIBLIOGRAFÍA ***3***

DESARROLLO ***4***

INTRODUCCIÓN....4

TEORÍA DE FUNCIONAMIENTO....5

FUNDAMENTOS DE LA VOZ....5
RECONOCIMIENTO AUTOMÁTICO DEL HABLA:....6
PRINCIPALES ÁREAS DE TRABAJO EN RECONOCIMIENTO DEL HABLA....6
PROCESAMIENTO DE LA SEÑAL DE VOZ....7
OBTENCIÓN DE INFORMACIÓN MEDIANTE MICRÓFONO....8
PRE-PROCESAMIENTO....8
PARAMETRIZACIÓN....10
RECONOCIMIENTO DEL HABLA. DECISIÓN.....12

DESARROLLO PRÁCTICO....15

MICRÓFONO....16
ADQUISICIÓN. MATLAB....16
PROCESAMIENTO DE LA SEÑAL....18
PARAMETRIZACIÓN....21
RECONOCIMIENTO DEL HABLA. DECISIÓN....23
INTERFAZ GRÁFICA....24

CONCLUSIONES....32

ANEXOS ***33***

LISTADOS DE PROGRAMAS....33

GUI.m:....33
ingresoaudio.m:....44
vozgrabar.m:....45
Reconocedor.m:....46
silencio.m:....47
melcepst.m:....47
función disteu.m:....51
función disteusq.m:....51

OBJETIVOS DEL TRABAJO

El presente problema de ingeniería consiste en el reconocimiento de una serie de palabras aisladas y de un número reducido de locutores con el fin de que las palabras emitidas solo por los locutores habilitados sean interpretadas como órdenes aplicadas al campo de la domótica (encendido de luces, acceso a determinados lugares, activación de alarmas, etc.) Se adquirirá la voz a través del micrófono de la PC y en la misma se hará un entorno gráfico.

MATERIAS INTEGRADAS

Técnicas digitales 3: Tema: Procesamiento digital de señales
Bibliografía: Capítulo 8: Procesamiento digital de señales.
Capítulo 8b: Procesamiento digital de señales. Aplicación.
Capítulo 8c: Procesamiento digital de señales. Filtros.

Análisis de señales y sistemas: Tema: Energía de una señal. Transformada de Fourier.

Comunicaciones I: Tema: Autocorrelación y correlación cruzada de señales.

Técnicas digitales 2: Tema: Muestreo.

POSIBLES APLICACIONES

El reconocimiento de voz es utilizado comercialmente en controles de accesos, en aplicaciones de domótica, en el comando de sillas de rueda, en el control de robots, etc. En este caso, y como se comentó con anterioridad será utilizado para identificar ordenes que comandarán acciones aplicadas al campo de la domótica como el encendido de las luces de la casa, o artefactos eléctricos o incluso activación de alarmas.

BIBLIOGRAFÍA

- Paper: Algoritmos y Métodos para el Reconocimiento de Voz en Español Mediante Sílabas. José Luis Oropeza Rodríguez. Centro de Investigación en Computación-IPN México D. F.
- Paper: IMPLEMENTACIÓN DE UN RECONOCEDOR DE PALABRAS AISLADAS DEPENDIENTE DEL LOCUTOR. César San Martín S.1 Roberto Carrillo A.1
- Paper: Reconocimiento de Voz para Niños con Discapacidad en el Habla. Universidad de las Américas Puebla. Escuela de Ingeniería. Departamento de Ingeniería Electrónica. Tesis profesional presentada por Jorge Andrés Franco Galván.
- Reconocimiento de voz .Informe de Proyecto de Graduación para optar por el grado de Bachiller en Ingeniería Electrónica. Roberto Calvo Arias.
- Sistema de reconocimiento de voz en MATLab. Trabajo de graduación de Genoveva Velásquez Ramírez.
- VOICEBOX: Librería de MATLAB para el reconocimiento del habla. Home page: http://www.ee.ic.ac.uk/hp/staff/dmb/voicebox/voicebox.html.
- Miyara, Federico. La Voz Humana. UNR Editora, Rosario. Argentina.
- Manual de interfaz gráfica de usuario en MATLab. Universidad Técnica Particular de Loja. Escuela de Electrónica y Telecomunicaciones. Diego Orlando Barragán Guerrero.
- http://www.secyt.frba.utn.edu.ar/gia/IA1_IntroReconocimientoVoz.pdf
-Diseño e implementación de un prototipo de reconocimiento de voz basado en modelos ocultos de Markov para comandar el movimiento de una silla de ruedas en un ambiente controlado. *www.gnewbook.org/action/file/download?file_guid=70523*

DESARROLLO

INTRODUCCIÓN

La domótica es un campo de la electrónica que crece día a día y donde nuevas aplicaciones aparecen permanentemente. Esto se debe a la búsqueda de simplificar algunas tareas hogareñas apelando a la tecnología. La implementación de las computadoras en este campo amplía considerablemente el espectro de posibilidades, gracias a su gran capacidad de procesamiento, permite la creación de una interfaz amigable con el usuario y además puede trabajar en conjunto con infinidad de dispositivos para alcanzar el objetivo.
La forma de interacción principal elegido entre el sistema a realizar y el usuario es a través de la voz del mismo. La voz, además de permitir la comunicación contiene mucha información de lo que se esta diciendo y de la persona que lo está diciendo. Entonces lo que se hará es un procesamiento digital de las señales de voz. El procesamiento digital de señales de voz tiene una gran variedad de aplicaciones, existe una base para el tratamiento digital de señales, que puede ser implementada para lograr obtener lo que nos interese según la aplicación. El Sistema de Reconocimiento de Voz es una de las aplicaciones del procesamiento digital de señales de voz. El sistema consiste en obtener una señal de voz que permita reconocer qué palabra se esta hablando y quien la esta realizando (dentro de un número limitado de locutores).
Para la resolución del problema de identificación de órdenes con fines domóticos se desarrollará un sistema que se describe a continuación. Utilización de un micrófono multimedia para capturar la voz y adquisición a través de la placa de sonido. Las señales de voz se almacenarán en archivos .wav para luego ser levantados para su análisis. El software de análisis elegido es MATlab y en este mismo también se hará el entorno gráfico. Una vez reconocida la palabra y locutor, o sea, una vez que se ha caracterizado la orden, la misma se ejecutará a través de archivos de sonido en la PC. Es decir si por ejemplo se detectó una orden para prender las luces de la casa, el software ejecutará un archivo de sonido con la orden reconocida. En ningún caso existirá una comunicación con un hardware externo a fin de que la orden se efectivice físicamente.
Como para el reconocimiento se hará una comparación de la señal adquirida con diferentes patrones de las palabras elegidas como órdenes de los diferentes locutores que estarán previamente guardados, el entorno gráfico deberá separar entre la parte de entrenamiento o de generación de esa base de patrones, y la parte de comparación y reconocimiento final.

Para llevar a cabo tal proyecto, el mismo deberá ser separado en diferentes etapas que se describirán en los correspondientes capítulos. Las mismas son las siguientes:

Adquisición: La señal captada por el micrófono y tomada por la placa de sonido será almacenada como archivo .wav. La señal de voz es de 16 bits de resolución, de un canal, una frecuencia de muestreo de 44100Hz y los archivos de voz tendrán una duración de 3 segundos a partir de que se presione un botón diseñado para tal fin en el entorno gráfico. Con esto estandarizamos la grabación de archivos.

Endpoint Detection: Una vez realizada la adquisición, es necesario realizar un filtrado de la señal adquirida, y detectar el comienzo y fin de la señal con el objetivo de eliminar información redundante de entrada al sistema (eliminar silencios al comienzo y al final de la señal de voz). Esto se realizará a través de un cálculo de energía y detección de umbrales de actividad de señal.

Caracterización: A través de la obtención de los coeficientes Cepstrum en la escala de Mel se extraerán las características principales de las señales par definir cada patrón.

Comparación: Las archivos de voz ingresados en la etapa de reconocimiento se compararán con los patrones ingresados en la etapa de entrenamiento. Un algoritmo de medida de distancias será utilizado en esta etapa.
Toma de decisión: A partir de la medida de distancia se elige un umbral de decisión a partir de la cual se ejecutará el archivo .wav con la orden reconocida.

TEORÍA DE FUNCIONAMIENTO

FUNDAMENTOS DE LA VOZ

La señal de voz es una onda de presión acústica que se genera voluntariamente a partir de movimientos de la estructura anatómica del sistema fonador humano. La producción de la voz comienza en el cerebro con la conceptualización de la idea que se desea transmitir, la cual se asocia a una estructura lingüística, seleccionando las palabras adecuadas y ordenándolas de acuerdo con unas reglas gramaticales. A continuación el cerebro produce los comandos nerviosos que mueven los órganos vocales para producir los sonidos. La unidad mínima de una cadena hablada es el fonema, el cual posee un carácter distintivo en la estructura de la lengua. La combinación de los fonemas da origen a las silabas, las cuales conforman las palabras, y estas a su vez, las oraciones. El tracto vocal empieza a la salida de la laringe y termina a la entrada de los labios. El tracto nasal empieza en el paladar y termina en los orificios nasales. Los parámetros principales del sistema articulatorio son: las cuerdas vocales, el paladar, la lengua, los dientes, los labios y las mandíbulas. Los distintos sonidos se producen al pasar el aire emitido por los pulmones, a través de todo el sistema de producción, en una determinada posición del aparato articulatorio. Desde el punto de vista de la ingeniería es lógico pensar que este sistema físico puede representarse como un filtro, cuya función de transferencia depende del sonido articulado y, por tanto, de la posición de los distintos órganos involucrados en la generación de la voz. La entrada del filtro se puede modelar mediante una señal de excitación, que corresponde al paso del aire generado por los pulmones a través de la traquea y las cuerdas vocales, y también será dependiente del sonido generado.

Las cuerdas vocales son dos membranas dentro de la laringe orientadas de adelante hacia atrás. Por delante se unen en el cartílago tiroides, por detrás, cada una esta sujeta a uno de los dos cartílagos aritenoides, los cuales pueden separarse voluntariamente por medio de músculos. La abertura entre ambas cuerdas se denomina glotis. Cuando las cuerdas vocales se encuentran separadas la glotis adopta una forma triangular. El aire pasa libremente y casi no se produce sonido; este es el caso de la respiración. Cuando la glotis comienza a cerrarse, el aire que la atraviesa proveniente de los pulmones experimenta una turbulencia, produciendo un ruido conocido como aspiración. Ahora, al cerrarse más, las cuerdas vocales comienzan a vibrar de modo audible, produciéndose un sonido tonal, es decir periódico. La frecuencia de este sonido depende de varios factores, entre otros del tamaño y la masa de las cuerdas vocales, de la tensión de las cuerdas vocales, de la tensión que se les aplique y de la velocidad del flujo del aire proveniente de los pulmones. A mayor tamaño, menor frecuencia de vibración, A mayor tensión la frecuencia aumenta, siendo los sonidos más agudos. También aumenta la frecuencia al crecer la velocidad del flujo de aire. Finalmente, es posible obturar la glotis completamente, en cual caso no se produce sonido. Sobre la glotis se encuentra la epiglotis, un cartílago de la faringe que permite tapar la glotis durante la deglución para evitar que el alimento ingerido se introduzca en el tracto respiratorio. La porción que incluye las cavidades faringea, oral y nasal junto con los elementos articulatorios se denomina cavidad supraglótica en tanto que los espacios por debajo de la laringe, es decir la traquea, los bronquios y los pulmones, se denominan cavidades infraglóticas.
Varios de los elementos de la cavidad supraglótica se controlan a voluntad, permitiendo modificar dentro de márgenes muy amplios los sonidos producidos por las cuerdas vocales o agregar partes distintivas a estos, y hasta producir sonidos propios. Esto se efectúa con dos mecanismos

principales: el filtrado y la articulación. El filtrado actúa modificando el espectro del sonido. Tiene lugar en las cuatro cavidades supraglóticas principales: la faringe, la cavidad nasal, la cavidad oral y la cavidad labial. Las mismas constituyen resonadores acústicos que enfatizan determinadas bandas frecuenciales del espectro generado por las cuerdas vocales, conduciendo al concepto de formantes, es decir que se refuerza la amplitud de grupos de armónicos situados alrededor de una determinada frecuencia. En resumen, en el habla los formantes se determinan por el proceso de filtrado que se produce en el tracto vocal por la configuración de los articuladores.

RECONOCIMIENTO AUTOMÁTICO DEL HABLA:

Las principales características que diferencian a los sistemas basados en reconocimiento del Habla frente a otras alternativas son: la naturalidad que supone utilizar el habla en las operaciones de comando y control, y la robustez y precisión en la comunicación para diferentes usuarios y diferentes entornos. Los resultados que esta tecnología proporcione, deben contrastar con los derivados de otras alternativas como: teclados, mouse, paneles y otros. Dado el amplio campo de aplicación de la tecnología del habla, se ha propuesto una clasificación, en tres grupos diferentes:

- **Aplicaciones locales**

  En esta área se pretende realizar interfaces hombre-máquina, que sustituyan la utilización de teclado y mouse, para dotar al usuario de la movilidad, que estos le restan, también facilitar el uso de máquinas a los usuarios discapacitados.

- **Respuesta vocal interactiva**

  En estas aplicaciones se involucra la difusión o captura de información, por parte de un gran número de usuarios, en particular se utiliza la red telefónica como vehículo de acceso a la información. Como por ejemplo para sustituir interfaces de detección de tonos DTMF, como lo son las consultas de cuentas bancarias, mensajería vocal, transmisión de información general, movimientos de cuentas y otros.

- **Automatización de sistemas telefónicos**

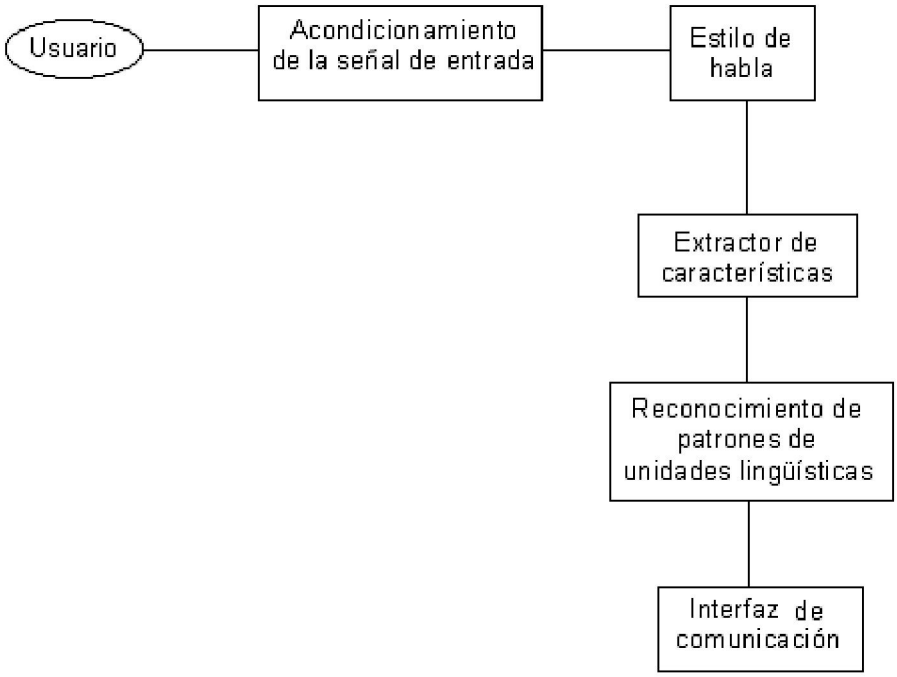
  En estas hallamos la marcación por voz, manejo de agendas, directorio público, entre otras.

PRINCIPALES ÁREAS DE TRABAJO EN RECONOCIMIENTO DEL HABLA

Un diagrama simplificado para un modelo de reconocimiento automático del habla, mostrado en la figura 1, sugiere algunos puntos importantes de considerar a la hora de llevar a cabo una implementación. Entre las principales áreas de trabajo que intervienen en el diseño y especificación de sistemas de Reconocimiento del Habla actuales, se encuentran las siguientes:

- Procesamiento de la señal de voz.
- Técnicas de reconocimiento de patrones.
- Diferentes estilos de habla.
- Dependencia del locutor.

- Tarea de reconocimiento.
- Bases de datos para entrenamiento y reconocimiento.



PROCESAMIENTO DE LA SEÑAL DE VOZ

La primera operación que debe realizar un reconocedor es procesar la señal de voz de entrada al sistema, con objeto de extraer la información acústica relevante para la tarea que debemos realizar. Los rasgos o características que se deben extraer de la señal de voz, son el resultado de un largo proceso de investigación sobre diferentes procedimientos de parametrización de la voz. Planteándose como solución actual más extendida una parametrización de la envolvente espectral que incluye consideraciones preceptúales a partir del funcionamiento del oído.

Para reducir el número de parámetros posibles, la parametrización se combina con la utilización de técnicas discriminativas, seleccionándose el subconjunto con los parámetros más eficientes o distintivos.

La señal de entrada puede venir acompañada por efectos perturbadores, los cuales se desea sean eliminados, para ello se ha generado tres líneas principales de trabajo:

- **Detección robusta de voz:** apareciendo innumerables procedimientos de discriminación entre voz o ruido (silencio) para diferentes tipos de ruido.
- **Reducción de ruido:** distinguiéndose procedimientos que actúan directamente sobre la señal de voz y procedimientos que buscan compensar el efecto del ruido sobre la parametrización de la voz.
- **Cancelación de ecos:** incorporando técnicas de filtrado adaptativo que permitan al usuario comenzar a hablar mientras, desde el terminal remoto, se le está comunicando un mensaje que puede provocar un eco en la voz que entra al reconocedor.

OBTENCIÓN DE INFORMACIÓN MEDIANTE MICRÓFONO

Micrófono: El micrófono es un transductor electroacústico. Su función es la de transformar (traducir) la presión acústica ejercida sobre su capsula por las ondas sonoras en energía eléctrica. El audio es un fenómeno analógico. Para grabar una señal de voz se hace la conversión de la señal analógica del micrófono en una señal digital por medio del conversor A/D en la tarjeta de sonido. Cuando un micrófono esta operando las ondas de sonido hacen que vibre el elemento magnético del micrófono causando una corriente eléctrica hacia la tarjeta de sonido, donde el conversor A/D básicamente graba los voltajes eléctricos en intervalos específicos. Hay dos factores importantes durante este proceso. Primero esta la taza de muestreo o que tan seguido los valores de voltaje son grabados. Segundo, son los bits por segundo, o que tan exactamente los valores son grabados. Un tercero podría ser el número de canales (mono o estereo), pero para las aplicaciones de reconocimiento de voz un canal mono es suficiente. La mayoría de aplicaciones vienen con valores pre-determinados, para desarrollo del código se debería de cambiar los parámetros para ver lo que mejor funciona en el algoritmo. Dado a que el habla es relativamente de bajas frecuencias (entre 100Hz-8kHz), una frecuencia de muestreo de 16000 muestras/seg provee una mayor exactitud en la adquisición de la información, la frecuencia de nyquist.
La cuantificación mas comúnmente usada, es de 8 bits, mínimo requerido para una calidad baja, puede mejorarse su S/R con una técnica no lineal de cuantificación, se obtienen excelentes resultados aumentando la cuantificación a 16 bits.
La siguiente etapa será aquella que se encargue de amplificar las señales a niveles que sean manejables. A partir de la señal analógica obtenida se hace necesario convertir la señal a formato digital para poder procesarla en la computadora lo que se realiza mediante dos procesos: muestreo y cuantificación. La señal vocal tiene componentes de frecuencia que pueden llegar a los 10 kHz., sin embargo la mayor parte de los sonidos vocales tiene energía espectral significativa hasta los 5 Khz. solamente los sonidos fricativos poseen componentes que pueden llegar a los 10 kHz.
En nuestro caso se optó por una frecuenta de muestreo de 44100Hz y una resolución de 16 bits mono canal.

PRE-PROCESAMIENTO

Se hace necesario para el análisis realizar un pre-procesamiento de la señal vocal. Esto se realiza a través de técnicas que permitan extraer la información acústica directamente a partir de la señal vocal emitida.

Filtro

En primera instancia se realiza un filtrado de la señal para reducir los efectos del ruido que acompaña la señal de voz ingresada. Al diseñar el mismo se deberá considerar el hecho que la elección de las frecuencias de corte y paso de este, condicionarán al sistema en la discriminación de voces espectralmente parecidas.

Segmentación (determinación de los puntos de inicio y fin de la señal para eliminar el ruido):

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 señal que llevan información de voz de aquellas que no llevan voz. Este procedimiento evita gastar memoria y

tiempo de cálculo en las tramas que no contienen información evitando así obtener resultados erróneos en el análisis de las señales de voz.
Se han planteado diferentes maneras de clasificar eventos en una señal de voz. Quizás la más simple y más empleada, esta relacionada con la generación de la voz e incluye tres estados: silencio cuando no hay voz; sonoro cuando se presenta vibración de las cuerdas vocales; y sordo cuando las cuerdas vocales no vibran.
En general, un sistema de clasificación de voz puede tener inconvenientes en distinguir un fonema sordo de baja amplitud y corta duración, del silencio; o un fonema sonoro de baja amplitud de un fonema sordo o incluso del silencio. El error a su vez aumenta a medida que la relación señal a ruido disminuye.
El problema de encontrar los puntos de inicio y fin de palabra es fundamental en procesamiento de voz. Por ejemplo, en reconocimiento automático de palabras aisladas, es necesario encontrar las regiones de la señal que corresponden a cada palabra a ser analizada. De la correcta segmentación de la señal depende en gran medida la exactitud del proceso de reconocimiento. De hecho, las fallas en la segmentación de la señal constituyen una de las principales fuentes de error en los sistemas de reconocimiento de voz, ya que algunos sonidos que pueden captarse, correspondientes a ruido de fondo, podrían eventualmente confundirse con voz; por ejemplo, el espectro de la respiración tiene semejanzas con el de un fonema fricativo. La detección de los límites de palabra también se realiza con el fin de evitar cálculos innecesarios, al procesar únicamente las partes de la señal que corresponden a voz.

Ventaneo

En la etapa siguiente, la señal filtrada y sin silencios se hace necesario tomar la señal en tramas de N muestras, donde N es un valor que se escoge tomando en cuenta que la señal de voz es estacionaria a “trozos”, condición necesaria para realizar el análisis de Fourier en tiempo corto. El intervalo de tiempo en el que la señal se considera estacionaria depende de la velocidad de cambios del tracto vocal y las cuerdas vocales y comúnmente se establece un valor entre 20 y 40 ms. Estas tramas se pasarán por una ventana que en general es una ventana de Hamming. Las ventanas rectangulares y las no rectangulares tienen sus ventajas y desventajas en cuanto al ancho del lóbulo principal y a la atenuación de los lóbulos secundarios. En el caso de la ventana de Hamming se cumple con el principio de atenuar bruscamente los lóbulos secundarios aunque el principal es más ancho que para una ventana rectangular. Esta es la ventana se define como:

$$wnT = 0.54\text{-}0.46\cos 2nN \; 0 \leq n \leq N$$

Como la ventana de Hamming atenúa mucho la señal que queda en bordes de dicha ventana se hace un solapamiento cada 10 o 20 ms para aplicarla así no se pierde información útil.

En el reconocimiento del habla, la señal de voz pre-procesada se ingresa a un nuevo procesamiento para producir una representación de la voz en forma de secuencia de vectores o agrupaciones de valores que se denominan parámetros, que deben representar la información contenida en la envolvente del espectro.
Hay que tener en cuenta que el número de parámetros debe ser reducido, para no saturar la base de datos, ya que mientras más parámetros tenga la representación menos fiables son los resultados y mas costosa la implementación. Existen distintos métodos de análisis para la extracción de características, y se concentran en diferentes aspectos representativos. En este caso analizaremos los dos de mayor importancia para el análisis de la voz:
- Análisis de predicción lineal (LPC)
- Análisis cepstral

PARAMETRIZACIÓN

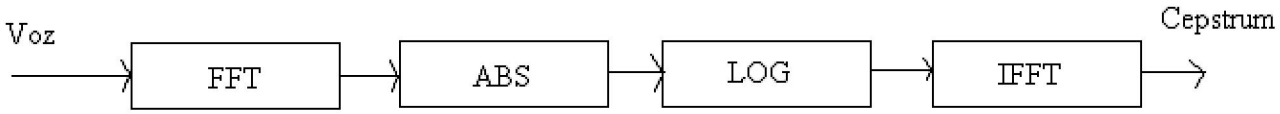
Coeficientes de predicción lineal (LPC):

Se trata de una de las técnicas más potentes de análisis de voz, y uno de los métodos más útiles para codificar voz con buena calidad. Su función es representar la envolvente espectral de una señal digital de voz en una forma comprimida, utilizando la información de un modelo lineal, con lo cual se proporcionan unas aproximaciones a los parámetros de la voz muy precisas.
Se fundamenta en establecer un modelo de filtro de tipo todo polo, para la fuente de sonido. La principal motivación del modelo todo polo viene dada porque permite describir la función de transferencia de un tubo, que sin perdidas esta formado por diferentes secciones. El modelo recibe este nombre porque pretende extrapolar el valor de la siguiente muestra de voz s(n) como la suma ponderada de muestras pasadas s(n-1), s(n-2),…, s(n-K):

$$s(n) \approx -\sum_{k=1}^{p} \alpha_k s(n-k)$$

Cepstrum

Como se sabe los sonidos de la voz se pueden representar mediante un espectrograma, que indica las componentes frecuenciales de la señal de voz. Es así entonces como el espectro nos proporciona información acerca de los parámetros del modelo de producción de voz, tanto de la excitación como del filtro que representa el tracto vocal.
Desde el principio de la década de los 70 los sistemas homo mórficos han tenido una gran importancia en los sistemas de reconocimiento de voz. Estos sistemas homo mórficos son una clase de sistemas no lineales que obedecen a un principio de superposición. De estos los sistemas lineales son un caso especial.



Modelo de obtención de los coeficientes cesptrales

En el sistema de reconocimiento de voz en MATLAB existe una función para obtener los coeficientes cepstrales utilizando *la FFT* . La función utilizada es la *rceps*, que nos proporciona el cepstrum real de la función ingresada, por medio del algoritmo mostrado en la figura de arriba. La razón principal para utilizar los coeficientes cepstrales es que tienen la ventaja adicional que uno puede derivar de ellos una serie de parámetros que son invariantes sin importar las distorsiones que puedan ser introducidas por el micrófono o por cualquier sistema de transmisión.

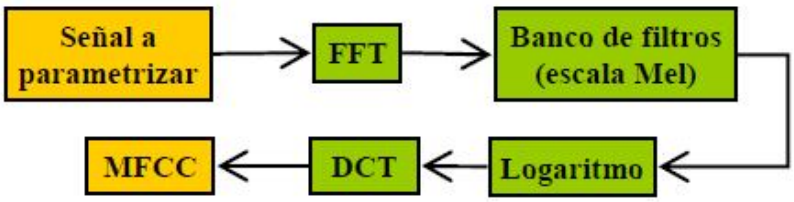
Coeficientes cepstrales de frecuencia Mel: MFCC.

Una familia de coeficientes directamente relacionada con los LPC son los llamados mel-cepstrum o MFCC, los cuales son de gran utilidad en la extracción de los parámetros de la señal de voz, ya que están basadaos en la variación conocida de los anchos de banda de las frecuencias críticas del oído. Los filtros que se lo aplican a la señal en la técnica MFCC están espaciados linealmente para frecuencias menores a 1000 Hz y logarítmicamente para frecuencias mayores de 1000 Hz, con el fin

de capturar las características fonéticamente importantes del habla. A esta escala se le denomina "Escala de MEL" y su fórmula matemática es la siguiente:

$$Mel(f) = 2595 \cdot \log\left(1 + \frac{f}{700}\right)$$

Los pasos necesarios para el cálculo de los MFCC se muestran aquí:



El primer bloque calcula la transformada de Fourier de tiempo corto a cada una de las tramas obtenidas de la etapa de pre-procesamiento mediante la ecuación:

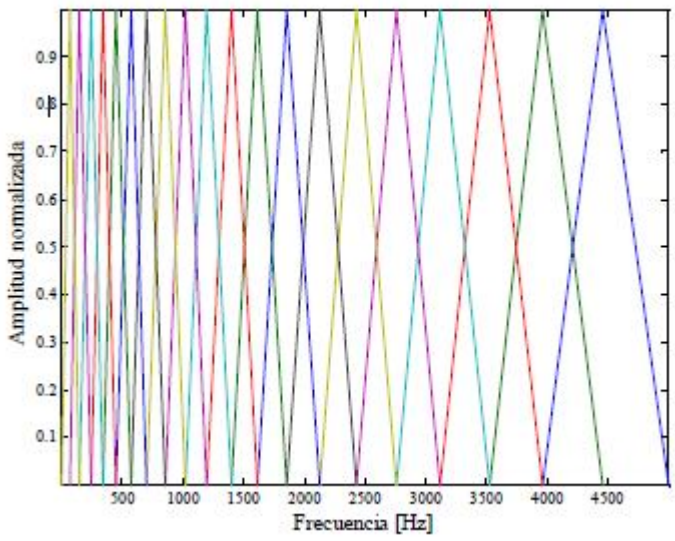
$$X(n,\omega_k) = \sum_{m=-\infty}^{\infty} x(m) \cdot w(n-m) \cdot e^{-j\omega_k m}$$

Donde: $\omega_k = \frac{2\pi}{N} \cdot k$

En el segundo bloque el cuadrado de la magnitud $X(n,\omega_k)$ es ponderado por una serie de filtros distribuidos sobre la escala de MEL para luego calcular la llamada "log-energía" del filtro l-ésimo mediante la ecuación:

$$E_{Mel}(n,l) = \frac{1}{A_l} \cdot \sum_{k=L_l}^{U_l} \left| V_L(\omega_K) \cdot X(n,\omega_K) \right|^2$$

El banco de filtros linealmente espaciado en la escala de Mel tiene la forma que se muestra abajo y los filtros que lo conforman pueden ser triangulares o tener otras formas, tales como Hamming, Hanning o Kaiser, pero el triangular es el más utilizado.



RECONOCIMIENTO DEL HABLA. DECISIÓN.

Las técnicas e parametrización explicadas en el punto anterior tienen como finalidad generar una serie de coeficientes que representan las características de la señal de voz, que pueden ser usadas en la fase de reconocimiento del habla y que no se obtienen mediante un análisis temporal o frecuencial. El tamaño de la matriz obtenida del proceso de parametrización depende directamente de la longitud (variable) de la señal de voz, la cual tiene 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, para que el tamaño de las matrices usadas para el reconocimiento del habla sea el mismo.

La estandarización de la matriz de coeficientes cepstrales constituye el primer paso a realizar en el proceso de reconocimiento del habla y se denomina Cuantización Vectorial. El paso siguiente corresponde al cálculo de la diferencia entre la señal de voz del hablante y las señales que se encuentran en la base de datos de entrenamiento del sistema; dicha diferenciase obtiene mediante el cálculo de la Distancia Euclidiana en varias dimensiones. Cada uno de los procedimientos se explica en las secciones siguientes.

CUANTIFICACION VECTORIAL

Una parte importante en cualquier tipo de procesamiento de voz es la optimización de los algoritmos en cuanto a velocidad y almacenamiento, entonces, la cuantificación de vectores trae consigo la idea de clasificar un conjunto de vectores, luego de lo cual se buscarán los mejores representantes para reducir el tamaño de la información a manejar. La forma de medir la fidelidad de un cuantificador es determinar el error que este produce al reemplazar los datos de entrada que recibe por los vectores representantes o codewords, dicho parámetro es llamado error por distorsión. La finalidad de un cuantificador es obtener un conjunto de vectores representativos llamado codebook, que presente el menor error por distorsión, por ejemplo para cuantificar los vectores de observación.

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.

Componentes de un cuantificador vectorial

Para construir un cuantificador vectorial se necesita:

1. Un gran número de vectores de observación, V1, V2,....., Vn, que conforman el grupo de entrenamiento. El grupo de entrenamiento se usa para crear el grupo de vectores del codebook "óptimo" que representa la variabilidad espectral observada en el grupo de entrenamiento.
2. Una medición de distancia entre cada par de vectores espectrales de observación para agrupar el conjunto de vectores de entrenamiento como axial también para asociar o clasificar vectores arbitrarios a cada entrada del codebook.
3. Un procedimiento de clasificación para ubicar y calcular los centroides. Sobre la base del particionamiento que clasifica el grupo de n vectores en *M* clústeres o sectores primero elegimos el número *M,* codewords del codebook, para luego proceder a la clasificación.

Vectores de observación

Al final de los distintos pasos para el tratamiento de la señal de voz, se obtiene un vector que contiene la información vocal que representa a la ventana temporal correspondiente, de alguna manera una colección de características que describen de la mejor manera posible la voz humana. Estos vectores son conocidos en la literatura del reconocimiento de voz como vectores de observación. Cabe aclarar que existen varias formas de representación de estas características como LPC (Linear Prediction Code) o Auditory System, pero la que en la actualidad da los mejores resultados es el análisis Cepstral, en particular los coeficientes MFCC (Mel Frecuency Cepstral Coeficients). También suele incorporarse al vector de observación la información de la primera y segunda derivadas del Cepstrum con respecto al tiempo para agregar información de las características dinámicas del sistema y el logaritmo de la energía total de la ventana.

Clasificación de Vectores

El objetivo de un modulo clasificador es agrupar una cantidad de vectores característicos, ***N***, en una cantidad *M (MDN)*, discreta, de sectores o celdas de clasificación logrando que las características en cada sector sean similares. Existen muchos criterios para lograr dicho objetivo y a continuación veremos algunos de los más comunes.
Imaginemos que la media multidimensional de un determinado sector *i*, es μi (con $1<i<M$), y a continuación ingresa al clasificador un vector de observación *o*, se puede clasificar dicho vector calculando la "distancia" a la que se halla de cada una de las *M* medias y asignándolo al sector mas "cercano". Este método de clasificacion se denomina k-Means debido a que se agrupan los vectores en torno a *k* valores medios, quedando formados *k* sectores (en nuestro caso $k=M$). Existe el problema de inicialización de los valores de μi, y su reestimación a medida que progresa el algoritmo.

Algoritmos de Clasificación:

Podemos decir, en general, que los *N* vectores originales de tamaño *D* quedarán representados por *M* vectores, cada uno de los cuales es llamado "palabra de código" o codeword (Cw), el grupo

entero de dichos vectores, forma un "libro de códigos" o codebook, quedan entonces delimitadas *M* regiones o sectores, llamados regiones de Voronoi. Los principales algoritmos de clasificación de vectores son descritos a continuación:

Algoritmo K-Means:

1. Inicialización: Arbitrariamente elegimos *M* vectores o palabras de código, 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.

La forma de cada sector o celda o partición es muy dependiente de la medida de distorsión espectral y las estadísticas de los vectores en el grupo de entrenamiento. Este método es el más simple y por tanto existen numerosas modificaciones y mejoras, algunos de sus puntos débiles son:

1. Los resultados dependen en forma muy acentuada de los valores iniciales elegidos como palabras de código.
2. También hay gran dependencia del número de sectores *M* axial como de la implementación de la "distancia" usada.
3. Puede suceder que algunos de los sectores resulten vacíos.

Algoritmo LBG:

Se analizará con algún detalle debido a su buen desempeño, para eso comenzaremos por el algoritmo fundamental LBG. El algoritmo LBG, lleva su nombre debido a sus autores Y. Linde, A. Buzo y R. M. Garrí, en él se elige 1 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 numero de codewords deseados. Este algoritmo de gran popularidad (que utiliza el algoritmo k-Means) produce codebooks que logran un mínimo local en la función de error por distorsión. Para generar un codebook de *M* sectores o palabras de código:
En primer lugar designando un codeword inicial para luego utilizando una técnica de división llegar a obtener un codebook inicial, luego iterando la misma técnica de división en los codewords hasta que llegamos a obtener el numero de codewords igual a *M* que va a ser el tamaño del codebook deseado.

El procesamiento se denomina división binaria:

1. Designar 1 vector del codebook o codeword inicial, este resulta ser el centroide del grupo de los vectores de entrenamiento.
2. Calcular la media del grupo de entrenamiento: Calcular el error o distancia media entre el codeword inicial y los vectores de entrenamiento.
3. Duplicar el tamaño del codebook mediante la división de cada codeword.

4. Usar el algoritmo K-Means para tomar el mejor grupo de centroides para la separación del codebook.
5. Iterar pasos 3 y 4 hasta llegar a un codebook de tamaño *M*.

Una de las causas que motivo el uso de un VQ fue la suposición que, en el límite, el codebook debería idealmente tener 36 vectores, uno por cada fonema, suposición que es incorrecta.

Utilización del cuantificador y del codebook

Una vez construido el codebook, el procedimiento para cuantificar vectores es básicamente realizar una búsqueda completa a través del codebook para encontrar el mejor representante. Si anotamos los vectores del codebook, de tamaño *M,* como *Cw, 1≤ w ≤ M,* y tomamos al vector de observación a ser cuantificado como V, luego el vector representante o codeword, *Vd.*,* es:

> Un procedimiento de cuantificación para señal de voz elige el vector más cercano del codebook al vector de observación y utiliza ese vector denominado codeword, como la representación resultante para etapas posteriores. Se refiere como al vector "vecino" más cercano, toma como entrada, vectores de señal de voz y da como respuesta, a su salida, el vector que mejor representa esa entrada.

Distancia Euclidiana

La dsitancia Euclidiana se emplea, para el reconocimiento del habla, como método para calcular la diferencia existente entre el codebook obtenido de la palabra dicha por el hablante y el resto de codebooks almacenados en la base de datos de entrenamiento del sistema. El resultado final de dicha comparación es un valor numérico que representa la distancia entre dos matrices de iguales dimensiones. El codebook perteneciente a la base de datos de entrenamiento del sistema cuya distancia al codebook generado para representar la palabra dicha por el hablante sea menor que el nivel de comparación establecido por el programador de la aplicación, identifica la palabra con la que ya existe mayor semejanza. La fórmula matemática empleada para el cálculo de la distancia euclidiana en dos o mñas dimensiones se muestra en la siguiente ecuación:

$$d(v,w)=\sqrt{(v_1-w_1)^2+(v_2-w_2)^2+(v_3-w_3)^2+\ldots+(v_n-w_n)^2}$$

Donde $v=[v_1\ v_2\ v_3....v_n]$ y $w=[w_1\ w_2\ w_3....w_n]$ son matrices de longitud n

Se acostumbra usar dicha métrica para el cálculo de la distancia entre dos puntos, pero su aplicaión va más allá, permitiendo calcular la distancia entre dos matrices.

DESARROLLO PRÁCTICO

En lo que concierne a esta parte del informe se describirá como se fueron realizando y llevando a la práctica los diferentes ítems mencionados durante el marco teórico.
A continuación comenzamos con el desarrollo de cada parte:

MICRÓFONO

Se utilizó un micrófono integrado a un par de auriculares de la marca Genius: modelo HS-02B. El mismo cuenta con las siguientes características.
Conector del micrófono: plug estéreo 3.5mm
Frecuencia de respuesta del micrófono: 80Hz ~ 16KHz
Impedancia del micrófono: 2.2KOhm / DC 4,5V
Sensibilidad del micrófono: Requiere 58

HS-02B y HS-02C son dos auriculares ajustables de diadema con almohadillas de esponja que puede utilizar durante períodos prolongados con gran comodidad. Cuentan con un micrófono flexible y control de volumen en línea que hace que su voz suene mucho más clara. El HS-02B y el HS-02C son ideales para mantener conversaciones en MSN, Ski pe y demás programas de mensajería instantánea por Internet.

Su condición de unidireccional facilita el procesamiento ya que solo toma la voz del locutor cuando esta habla a una distancia corta, permitiéndole hablar a mayor distancia sin que el sistema tome lo que dice si lo necesita. Por otra parte está condición minimiza considerablemente la influencia de ruidos externos ajenos a las órdenes emitidas por el locutor.

Al margen del modelo y marca del micrófono, es recomendable que sea unidireccional para minimizar los efectos de ruidos externos.

ADQUISICIÓN. MATLAB

Como se mencionó más arriba se eligió una frecuencia de muestreo de 44100Hz con una resolución de 16 bits y monocanal.
La forma más fácil en MATLab de adquirir audio por palca de sonido es a través de la función:

wavrecord(n,Fs)

Esta función graba n muestras de la señal de audio, muestreadas a una frecuencia de Fs. En una primera instancia esta fue la forma elegida para adquirir la señal pero más adelante se requirió que se vaya mostrando en pantalla el audio a medida que se iba adquiriendo. De esta manera hubo que buscar un modo alternativo de adquirir ya que la función *wavrecord* no dejaba disponible la señal para el ploteo en pantalla hasta que no se haya terminado con la adquisición. Entonces se recurrió al código que se muestra más abajo y que se explicará brevemente. Este código permite hacer una adquisición de datos ya sea por la placa de sonido o por algún DSP conectado a la computadora de acuerdo a la configuración de instrucciones que se haga.

handles.AI = analoginput('winsound'); Esta línea indica que se creará un objeto de entrada analógica denominado handles.AI que manejará la placa de sonido.

chan = addchannel(handles.AI,1); Se selecciona un solo canal.

set(handles.AI,'SampleRate',44100) Esta instrucción indica las muestras a tomar por segundo.

*set(handles.AI,'SamplesPerTrigger',duration*ActualRate)* con esta instrucción se indica la cantidad total de muestras a tomar.

En nuestra aplicación se colocaron dos botones que realizan una adquisición de audio. Con uno se ingresan los patrones y se realiza utilizando las instrucciones anteriores. El otro se utiliza para iniciar el reconocimiento de órdenes y la adquisición es activada por un umbral de audio en el micrófono. En este caso se deberán especificar las siguientes instrucciones para que la adquisición se inicie por la presencia de audio en el micrófono.

set(AI,'TriggerType','software');	Disparo por presencia de audio.
set(AI,'TriggerCondition','rising');	La condición de disparo es superado un umbral
set(AI,'TriggerConditionValue',0.13);	El umbral es 0.13 voltios
set(AI,'TriggerChannel',AI.Channel(1));	Se indica el canal de disparo

En cualquiera de los dos botones se comienza con la adquisición con la instrucción
start(handles.AI)
En el caso que se especifico el *trigger* por software los datos para matlab solo interesarán a partir de que se supere el umbral y no desde que se ejecute *start.*

A continuación se indicará cual es el trozo de código que permite adquirir una determinada cantidad de muestras, plotearlas, continuar con la adquisición, plotear nuevamente y así hasta terminar con la adquisición. El ejecutarse tan rápidamente pareciera que se gráfica en tiempo real la señal. La función fundamental para esto es *drawnow.*

```
while handles.AI.SamplesAcquired < preview
end
while handles.AI.SamplesAcquired < duration*ActualRate
   data = peekdata(handles.AI,preview);
   for i=1:882
      if abs(data(i))<8e-3
      data(i) =0;
      end
   end
   set(handles.h,'ydata',data)
   drawnow
end
data = getdata(handles.AI);
handles.grafica=1:1:88200;
handles.h = plot(handles.grafica,data);
title('Orden')
xlabel('Muestras')
ylabel('Nivel de voz')
```

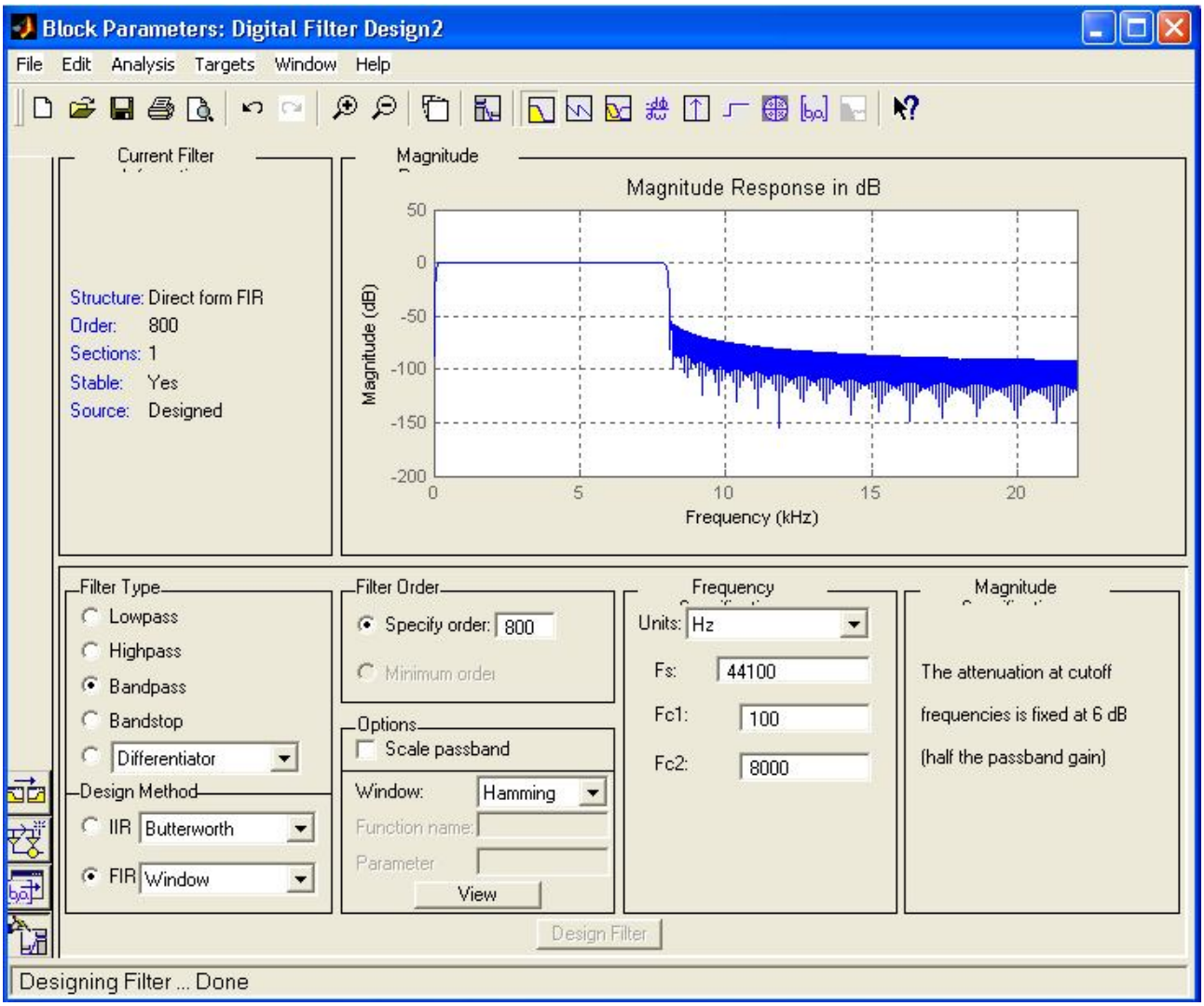
Después de terminar con la adquisición y gráfica de la señal de audio se elimina el objeto de entrada analógica que inicialmente se había creado con las siguientes funciones de MATLab.

```
delete(handles.AI)
clear handles.AI
```

PROCESAMIENTO DE LA SEÑAL

Filtrado:

El diseño del filtro se realizó a través de la herramienta Digital Filter Design2 de MATLab. En la misma se puede hacer una selección de los parámetros característicos de un filtro, observar su respuesta en frecuencia, obtener los coeficientes característicos del mismo y conocer su estabilidad.



Como se puede observar en el bloque se especificó que se quería diseñar un filtro FIR pasa-banda, tipo ventana, donde esa ventana era la de Hamming, con frecuencias de corte en 100Hz y 8000Hz a una frecuencia de muestreo a 44100Hz. Se obtuvo un filtro de orden 800, estable con una respuesta en frecuencia como la que se muestra en la imagen. Hay que aclarar que muchas de las características finales del filtro se obtuvieron a partir de múltiples pruebas donde se evaluó la respuesta del mismo, su selectividad, estabilidad, atenuación en bandas de paso, atenuación en bandas de corte y rizado en la banda de paso; lo que nos llevó a la decisión final.
Sobre el editor de MATLab el filtro diseñado se implemento a través de los siguientes comandos:

```
B = FIR1(800,[100 8000]/22050);
patron=filter(B,1,ua4);
```

Donde FIR1 es la función que devuelve los coeficientes del filtro, donde se le ingresaron como parámetros el orden del mismo y las frecuencias de corte. El tipo de ventana no se especificó ya que esta función trae por defecto la de Hamming que era la elegida. La función filter hace pasar la señal de voz (ua4 en este caso) a través de los coeficientes que contiene B (que se obtuvieron con FIR1).

Detección y eliminación de los períodos de silencio de la señal:

Se comienza por hacer un cálculo de la energía de la señal de audio. Para esto se calcula el valor absoluto de la señal y se obtiene el pico máximo. Se divide la señal de audio por el valor antes obtenido para independizar la forma de onda respecto de la intensidad de la señal.

```
len = length(s); % longitud del  vector
d=max(abs(s));
s=s/d;
```

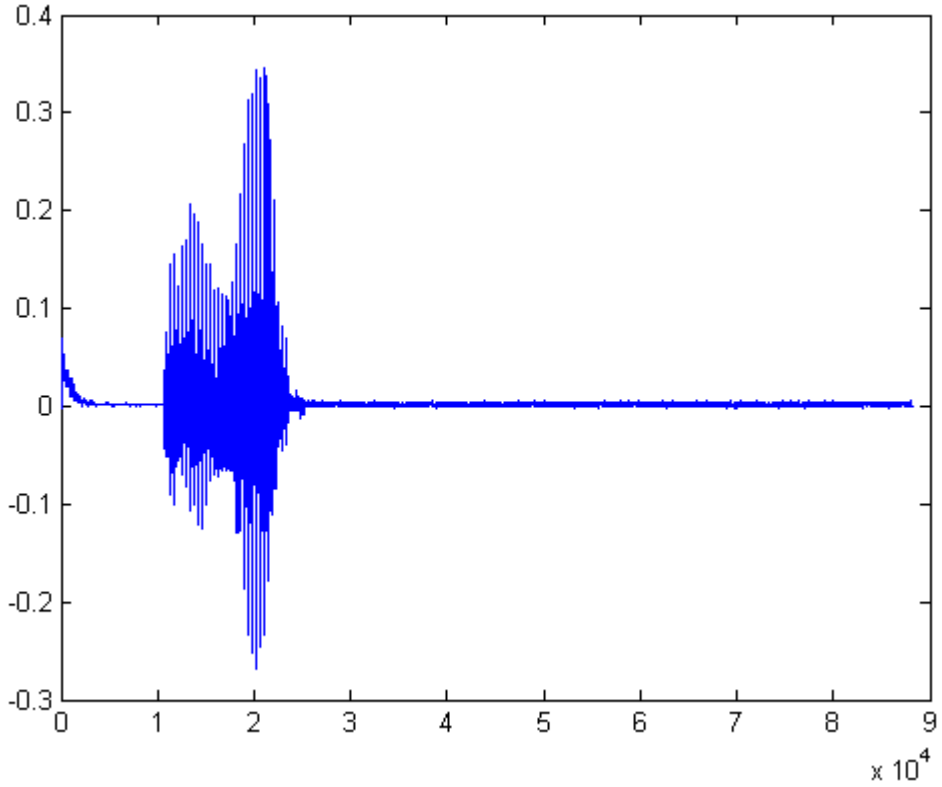
La señal normalizada se eleva al cuadrado y se divide por el número de muestras de la señal con lo que se obtiene la energía promedio de la señal.

```
avg_e = sum(s.*s)/len
```

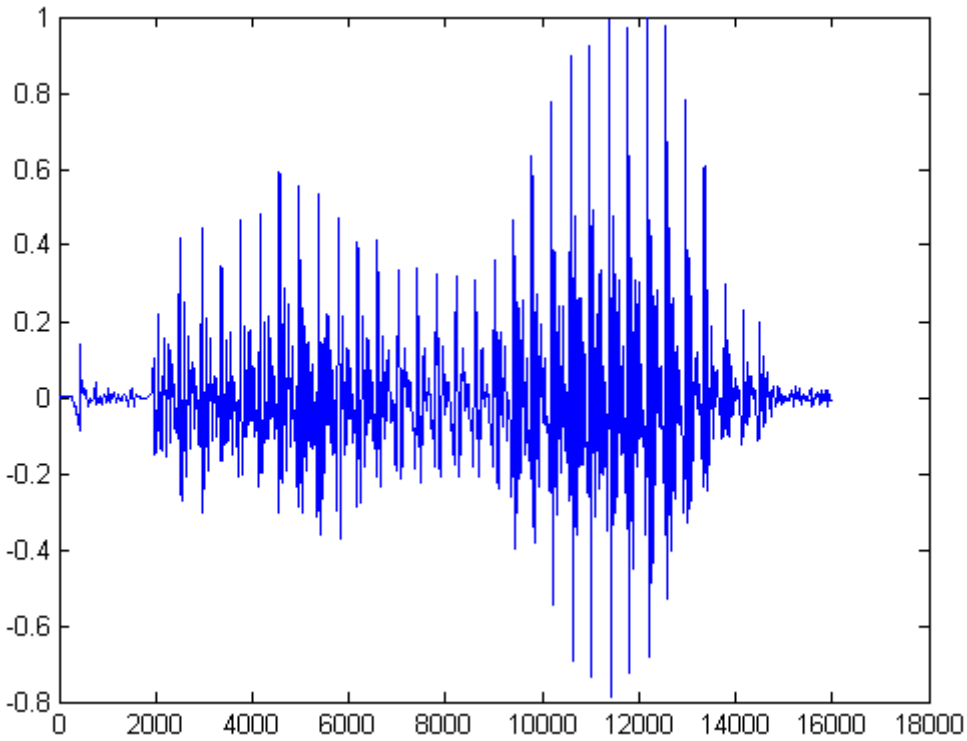
Luego lo que se hará es dividir la señal normalizada en ventanas de un número determinado de muestras, calcular la energía de ese trozo de señal y si esa energía es mayor a un porcentaje (tomado como umbral de decisión) de la energía promedio de la señal completa entonces dicha ventana se conserva. En caso contrario la ventana se desecha ya que se interpreta que al no llegar su energía al umbral establecido entonces esa ventana corresponde con un intervalo de silencio de la señal de audio.

```
THRES = 0.02;
y = [0];
for i = 1:400:len-400
seg = s(i:i+399)
e = sum(seg.*seg)/400;
if( e> THRES*avg_e)
y=[y;seg(1:end)];
end;
end;
```

THRES es el umbral de decisión y corresponde al 2% de la energía promedio de la señal entera. Este valor se definió después de varias pruebas observando cuales eran los resultados de eliminación de silencios. Las ventanas se eligieron de 400 muestras que a la frecuencia de muestreo (44100Hz) corresponde a aproximadamente 10ms por lo que no existe la posibilidad de que en el intento de eliminar silencio se elimine algún fonema de audio (cuya duración ronda entre los 10 y 20ms).En la siguiente imagen se mostrará la evolución temporal de la palabra “hola”. El archivo generado es de 2 segundos, o sea 88200 muestras. En el se pueden observar los períodos de silencio acompañados con ruido de fondo. Estos períodos no aportan información al sistema por lo que se busca eliminarlos.



Esta imagen corresponde al mismo archivo después de haber sido pasado por el algoritmo de eliminación de silencio.



PARAMETRIZACIÓN

Para la obtención de coeficientes que caractericen a los patrones guardados y/o a las órdenes de reconocimiento se utilizo la función melcepst.m. Esta función permite obtener los coeficientes cepstrales en escala de MEL cuya teoría se mencionó más arriba. La elección de los coficientes cepstrales en escala de MEL se definió debido a que fue la que mejores resultados prácticos nos dió.

La función melcepst.m pertenece a la librería Voicebox que se puede descargar de http://www.ee.ic.ac.uk/hp/staff/dmb/voicebox/voicebox.html.

Esta tiene como argumentos los siguientes parámetros:

melcepst(s,fs,w,nc,p,n,inc,fl,fh)

s= es la señal de audio con los intervalos de silencio eliminados.

fs= es la frecuencia de muestreo que en nuestro caso es de 44100Hz. Si se omite este parámetro, el algoritmo toma por defecto 11025Hz.

w= es el tipo de filtro que conforma el banco e filtros. Puede ser de Hamming, Hanning, rectangular, triangular, entre otros. Para el caso utilizamos una ventana de Hamming aunque se recomienda utilizar un ventaneo triangular para mejores resultados. La decisión de no utilizar un ventaneo triangular se debió a que la versión de MATLab sobre la que estábamos trabajando no permitía el uso de esta opción. A partir de la versión 7.0 de MATLab ya se puede utilizar.

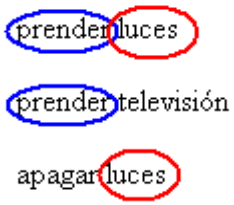
nc= el número de coeficientes cepstrales que caracterizarán a la parametrización. Por defecto son 12 pero en nuestro caso se utilizaron 14 coeficientes ya que fueron los que mejores resultados arrojaron en la práctica.

p= indica el número de filtros que tendrá el banco de filtros característico de este tipo de parametrización. Para nuestra aplicación se definió un total de 30 filtros para el banco.

n= longitud de las ventanas de muestras. La función **melcepst** además de la parametrización hace el ventaneo por lo que es un parámetro a especificar. En los otros métodos de parametrización como Cepstrum o LPC tendríamos que haber hecho el ventaneo de la señal antes de la parametrización. Vale aclarar que la función impone que sea una potencia de 2 entonces se eligió n=1024 que corresponde a una ventana de aproximadamente 23ms de duración, valor que se encuentra dentro del rango (20 a 40ms) donde la señal se considera estacionaria.

inc= indica el solapamiento entre ventanas. El porque del solapamiento se explicó en la parte teórica del ventaneo. El valor por defecto es n/2 que según nuestra elección de n sería 512 muestras pero se eligió inc=410 por mejores resultados.

Al ejecutar la función melcepst con los argumentos elegidos, esta nos devuelve una matriz con coeficientes que caracterizan a la señal de audio ingresada.
Otra consideración práctica que se realizó fue la de obtener dos matrices por cada patrón u orden de reconocimiento ingresada dividiendo la señal de audio de entrada en 2 partes. Esto se decidió así ya que como se verá más adelante. Todas las órdenes a identificar estaban compuestas por 2 palabras. En algunos de los casos se da que la primera palabra de una orden coincide con la primera de otra orden y con la segunda palabra de una tercera orden. Ejemplo:



Entonces estas coincidencias entorpecen la identificación. La división de la señal de audio favorece a la identificación pero como la duración de las diferentes palabras no es la misma (ej: televisión dura más que luces) se realizó un solapamiento en la división de la señal que eliminaría el inconveniente. Este solapamiento es tomar como primera parte de la orden desde la muestra 1 a la última menos 22000, y como segunda parte de la orden, se toma de la última menos 26000 a la última. Es decir hay un solapamiento de 400 muestras que soluciona lo antes mencionado. Traducido a código queda como sigue:

```
c01=melcepst(q(1:end-22000),fs,'M',14,30,1024,410,0,0.1815);
c02=melcepst(q(end-26000:end),fs,'M',14,30,1024,410,0,0.1815);
```

Entonces por cada orden del usuario se obtienen 2 matrices con coeficientes característicos de la misma.

c01 =

4.2436	-1.4401	1.3571	-1.7057	0.0533	-1.2835	-0.4034	-0.1248	-0.0416	-0.5090	-0.4414	-0.4740	-0.0248	-0.3350
6.6861	-3.0368	0.3904	-1.9813	-1.1730	-1.6137	-1.3042	-0.4093	0.0959	0.5109	-0.7858	-0.3643	0.1569	-0.1697
7.1107	-3.3831	0.6769	-0.6681	-1.8361	-1.5254	-0.6683	-0.7584	0.1979	0.7171	-0.9211	-0.2614	0.3143	-0.2417
6.7876	-3.2025	1.3406	0.0783	-1.6721	-1.9053	-0.4852	-1.3030	0.6542	1.0498	-0.3964	-0.5841	-0.1190	0.0741
5.6227	-2.3375	0.8181	-0.8049	-1.7284	-1.2176	-0.9907	-1.0233	0.3724	0.3333	-0.6590	0.0848	0.4649	-0.2088
7.8258	-4.2386	1.8657	-1.4081	-1.4314	-1.6621	-0.4457	-1.0381	0.7214	0.1085	-0.5310	0.2524	0.5899	-0.5657
7.6238	-4.0680	1.6383	-0.7441	-2.1066	-0.9586	-0.9343	-0.5337	0.6163	0.0090	-0.4293	0.1216	0.3403	-0.1725
7.4744	-4.2794	1.6340	-0.5958	-2.2985	-0.6115	-1.0734	-0.4795	0.6657	0.0276	-0.3335	0.0408	0.5739	-0.1848
6.5264	-3.0935	0.9506	-0.4741	-1.9200	-1.1641	-0.7538	-0.6970	0.2940	0.3636	-0.8151	0.1907	0.4664	-0.4920
6.2534	-3.0503	1.0085	-1.1144	-0.8670	-1.7694	-0.6779	-0.3016	-0.3029	0.6049	-0.7247	0.0372	0.5317	-0.3140
6.4373	-3.5453	2.0264	-2.1289	-0.0113	-2.5951	-0.9592	-0.2828	-0.7049	0.7244	-0.6969	0.0659	0.3825	-0.1764
6.8524	-3.6752	2.7376	-2.6707	0.6849	-2.5564	-1.0401	-0.3482	-0.7235	0.3779	-0.5802	0.1186	0.5243	-0.1318
6.0834	-2.4294	2.6209	-2.4788	1.0541	-2.1924	-1.0287	-0.3101	-0.6354	0.1413	-0.8095	-0.0665	0.5899	-0.0605
4.0749	0.0796	1.3353	-1.5994	0.7210	-1.5883	-1.2143	-0.3475	-0.3543	0.1761	-0.8799	-0.4092	0.7094	0.1908
3.6089	-1.1579	1.7155	-1.5101	0.4005	-1.3024	-1.6109	-0.3361	-0.3860	-0.1720	-0.5480	-0.7111	0.3673	0.4777
4.5600	-2.8444	1.9937	-2.2387	0.0813	-1.6622	-1.1457	-0.1783	-0.0468	-0.0515	-0.3966	-0.1788	0.1849	0.0591
5.4657	-3.7751	1.8650	-2.4429	-0.4903	-1.6307	-1.2804	0.0465	0.1679	0.0878	-0.2799	-0.1451	0.2344	-0.1391
5.4879	-3.4465	1.0714	-2.0360	-1.1800	-1.1558	-1.4114	0.1816	0.4847	0.0045	-0.1885	-0.5400	0.2494	-0.2476
4.9954	-2.8899	0.2259	-1.5633	-1.5498	-1.1178	-1.1885	0.1909	0.4576	0.0560	-0.3446	-0.7240	0.1824	-0.1099
6.1975	-3.4595	0.5273	-1.4204	-1.4609	-1.5361	-0.6266	-0.2082	0.5173	0.0977	-0.6899	-0.4774	0.2378	-0.0860
7.1195	-4.5249	1.3510	-1.5929	-1.5531	-1.4462	-0.5356	-0.9265	0.8638	0.0711	-0.5255	-0.3358	0.1826	-0.0803
7.8878	-4.7574	1.9157	-1.3207	-1.6565	-1.6086	-0.4577	-1.2978	0.5833	0.3852	-0.7412	-0.4436	0.3720	0.1696
7.3864	-3.0924	2.4407	-0.6289	-0.9745	-1.3012	0.0745	-1.0868	0.2584	0.4888	-1.2185	-1.1113	0.1830	-0.2577
5.1394	-2.8388	1.5425	0.0735	-1.4402	-0.2283	-0.6104	-1.4692	0.5590	0.8323	-0.9156	-0.4700	0.2335	-0.0613
6.5352	-3.2498	2.4499	-0.9832	-0.8575	-0.8547	-0.8073	-1.2113	0.5501	-0.0451	-1.0083	-0.3140	0.4385	-0.1788
7.0294	-3.5526	2.6551	-1.7095	-0.3685	-1.6116	-1.0137	-0.6402	-0.0082	0.2122	-0.5244	-0.4185	0.7174	-0.2254
5.5731	-1.9552	2.0634	-1.1263	-0.3478	-1.5223	-1.8340	-0.0925	-0.3310	0.3032	-0.4214	-0.4646	0.7599	-0.2716
4.1252	-0.2385	1.3238	-1.0679	0.3746	-1.7009	-0.9966	-0.1490	0.0099	0.1090	-0.5689	-0.4313	0.5601	-0.1215
5.2328	-2.3942	2.2614	-1.1492	-0.8246	-1.0176	-1.1856	-0.3284	-0.6324	-0.2374	0.1200	0.0456	0.2989	-0.2387
5.4133	-3.1292	2.2821	-1.4571	-1.4334	-1.0542	-1.4902	0.1202	-0.1167	-0.2972	-0.0366	-0.2279	0.4627	-0.3058
5.5794	-3.8100	2.8021	-1.7389	-1.4645	-1.4283	-1.1218	-0.1510	-0.0646	-0.4563	-0.0542	-0.2363	0.4813	-0.5018
5.7239	-3.7817	3.0123	-1.7039	-1.3018	-1.3183	-1.0892	-0.1623	-0.2270	-0.5340	-0.5324	-0.1641	0.2624	-0.3709
4.5187	-2.8283	2.7881	-0.9705	-1.0666	-0.8698	-1.1265	-0.8343	0.1272	-1.2137	-0.4036	-0.4443	0.3257	0.3020
4.6223	-2.2796	2.9561	-1.6557	-1.0188	-0.7331	-1.5126	-0.7259	0.4607	-1.3760	-0.6562	-0.0777	0.4117	-0.0701
4.6896	-2.4071	3.4861	-0.8653	-1.2377	-0.0591	-2.1788	-0.5860	0.0410	-1.0099	-0.7688	-0.1459	0.5112	-0.3017
5.1441	-2.6219	2.8565	-0.7109	-0.9651	0.1296	-1.7744	-0.4790	-0.1107	-1.3112	-0.4384	-0.1197	0.4805	-0.3381
5.2918	-2.6562	3.3210	-0.6906	-0.8035	-0.0607	-1.8384	-0.1721	0.1716	-1.1670	-0.3359	-0.1875	0.4286	-0.1927
6.0954	-3.1255	3.4252	-1.1468	-1.1293	-0.0346	-1.9270	0.4447	0.1649	-1.1747	-0.3424	-0.2795	0.3725	-0.0388
6.1937	-4.1071	3.8665	-1.2672	-1.1050	-0.7366	-1.6843	0.1662	-0.1040	-1.3048	-0.2114	-0.5255	0.4620	0.2118

Este es un ejemplo de la matriz de los coeficientes característicos de la primera parte de la orden “prender televisión”.

RECONOCIMIENTO DEL HABLA. DECISIÓN

Distancia:

Una característica fundamental de los sistemas de reconocimiento es la forma en que las matrices características son combinadas y comparadas con los patrones de referencia. Para poder realizar estas operaciones es necesario definir una medida de distancia entre los vectores característicos. En el algoritmo de reconocimiento en MATLAB se utiliza una distancia Euclidea, definida del siguiente modo: por ejemplo si *fi* y *fi'*, con *i=0, 1, 2,…, D* son las componentes de dos vectores característicos *f* y *f'*, puede definirse la siguiente métrica.

$$d = \sqrt{\sum_{i=1}^{D} \left| f_i - f_i' \right|^2}$$

En el código esta fórmula se implemento con la función *disteu* que a su vez utiliza la función disteusq de la librería Voicebox que se puede descargar de http://www.ee.ic.ac.uk/hp/staff/dmb/voicebox/voicebox.html.
La función disteusq devuelve un vector distancia que la función disteu transforma en un único valor. Como las matrices ingresadas son dos; la que contiene las características de la orden a reconocer y la que contiene las características de un determinado patrón, esta función deberá ejecutarse tantas veces como patrones haya. Ya que la comparación es uno a uno.
De esta manera se obtiene un vector que contiene la distancia de la matriz característica de la orden respecto a cada patrón guardado. A continuación se mostrará un ejemplo de lo que se obtiene después de ingresar una orden determinada:

```
dt =

  Columns 1 through 14

    7.9201   11.3892    6.2783    6.9242    8.9079   10.0768   14.5455   12.4339   16.2177   15.6169   17.3571   17.2723   10.5772   11.1169

  Columns 15 through 28

   13.4350   13.8517   12.0490   12.0799    6.8557    9.2736    6.4133   12.9787    8.8801    8.6026   11.5952   10.2881   23.5066   11.7278
```

El vector dt contiene todas las distancias y por simple observación podemos ver que la menor de ellas es la que se encuentra en el tercer elemento del vector.
Paso siguiente es mediante código obtener el menor elemento que indicaría la menor distancia y si este es menor a un set point definido por el usuario se determina que esa es la frase identificada.
Se puede saber cual de todas es y que usuario la dijo por el orden del elemento, que indica la menor distancia. Esto es:

Elemento	Usuario	Frase
1	1	Prender luces
2	1	Apagar luces
3	1	Prender televisión
4	1	Apagar televisión
5	1	Encender alarma
6	1	Apagar alarma
7	1	Contraseña
8	2	Prender luces
9	2	Apagar luces

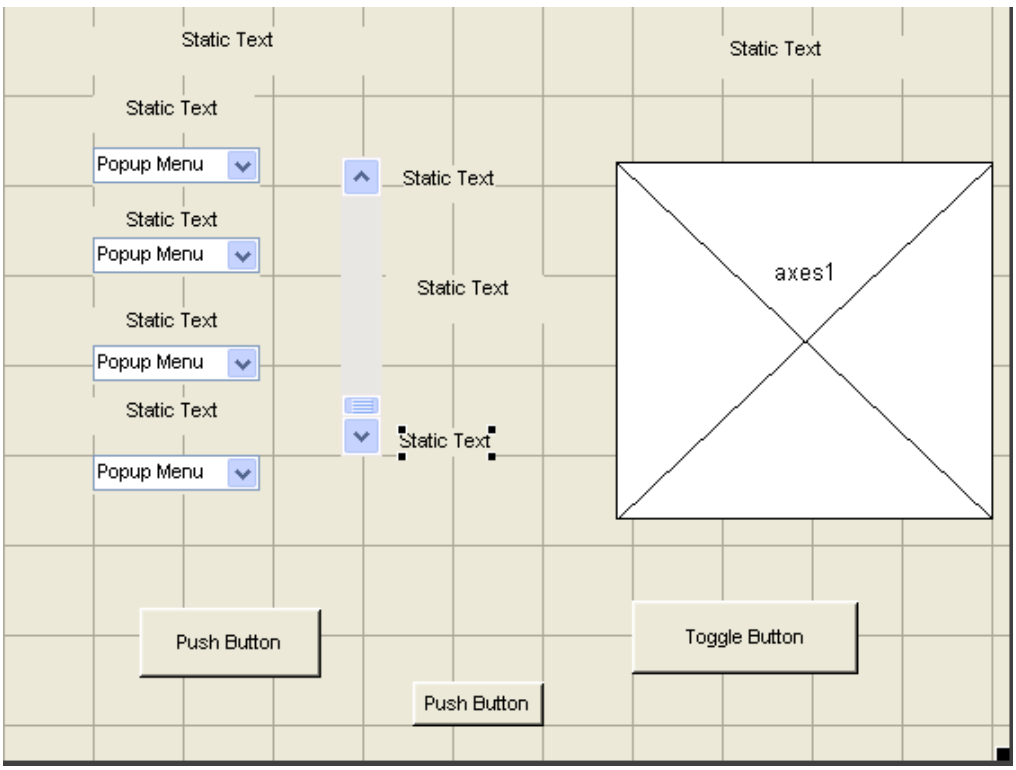
15	3	Prender luces
16	3	Apagar luces
22	4	Prender luces
28	4	Contraseña

Para el caso en el que se identifica “Encender alarma” o “Apagar alarma” se escuchará un audio que pedirá una confirmación de la orden ya que estas dos órdenes se definen como de mayor jerarquía y requieren de una doble confirmación.
Nuevamente se obtendrá un vector de distancia pero con los patrones de los 4 usuarios de las órdenes: “Confirmar orden” y “Omitir orden”. Se obtiene el menor de los elementos y si es menor que el set point se lo define como el patrón reconocido. Se puede dar el caso que el elemento menor del vector que contiene las distancias sea mayor que el set point. En ese caso se reproducirá un audio que indicará que el patrón es desconocido.

INTERFAZ GRÁFICA

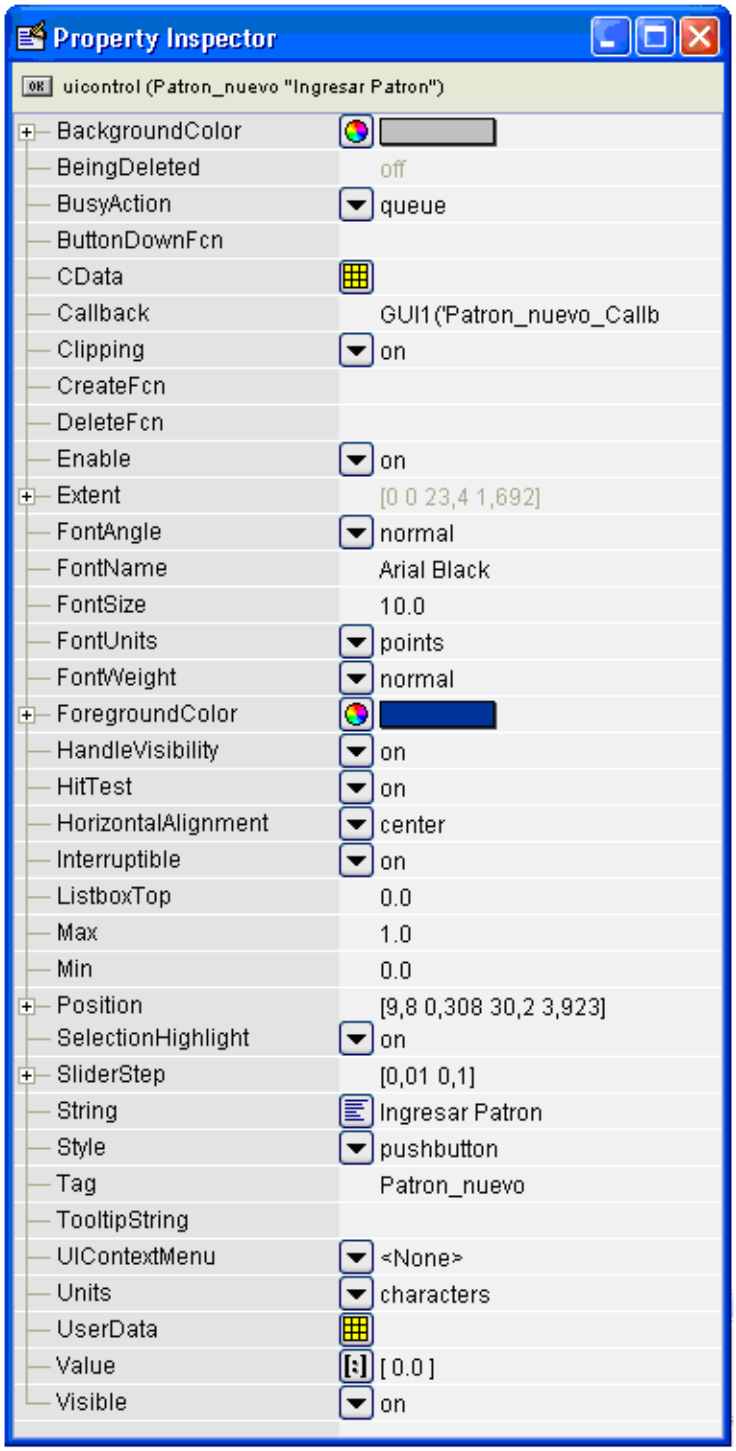
Para realizar la interfaz gráfica se recurrió al uso de la GUIDE de MATLab. Dicho entorno nos permite diseñar primero gráficamente la interfaz de usuario para luego volcar dicha interfaz a código. En ella se pueden ir colocando sobre un marco diferentes tipos de botones, textos fijos, textos dinámicos modificables por el usuario, checkbox, menús desplegables, imágenes, gráficos, entre otros.

A continuación se mostrará el marco inicial de sobre el que comenzamos a trabajar y en el que vamos colocando diferentes botones según nuestras necesidades.

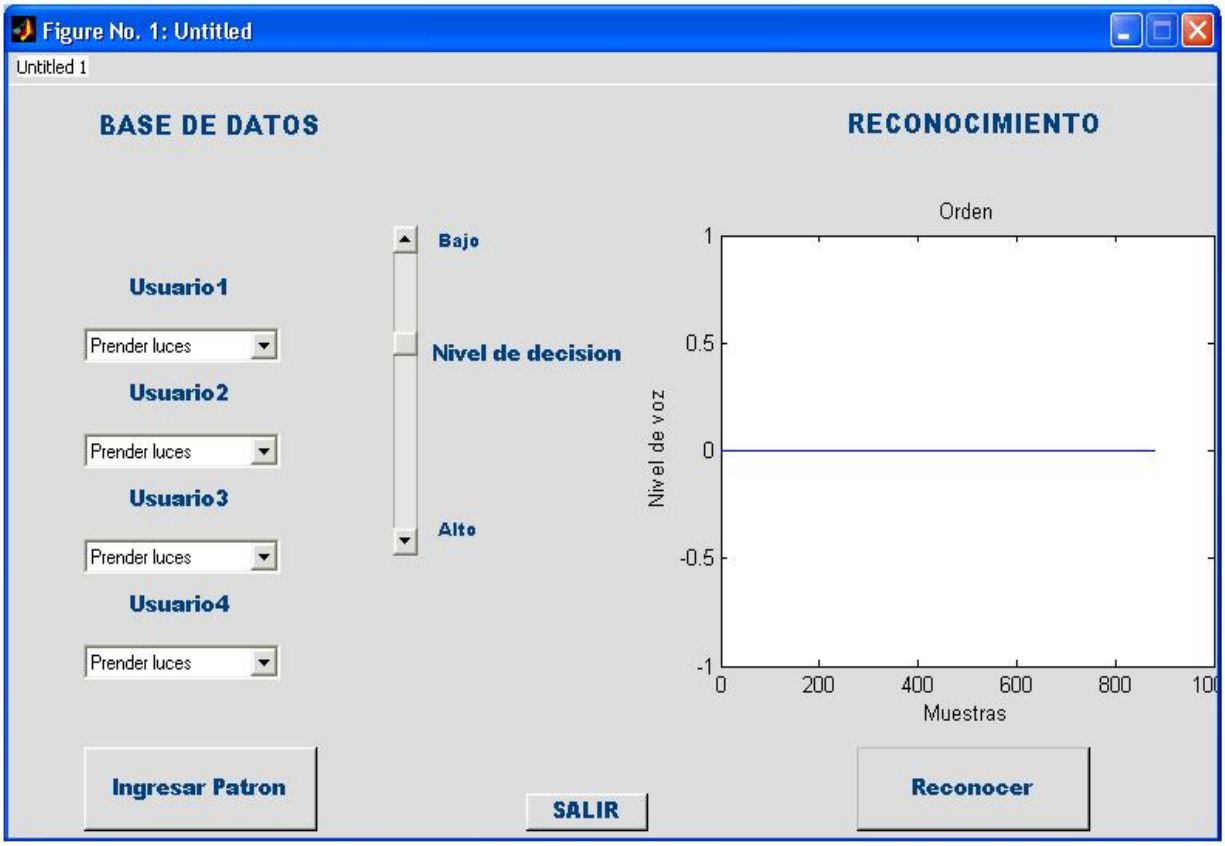


Los Static Text se utilizarán para indicar títulos y funcionalidades. Los Popup Menu serán menús desplegables que contendrán las órdenes que guardaremos y luego trataremos de identificar. El Slider o barra deslizante nos servirá para que el usuario elija el nivel de precisión del algoritmo de

decisión. Axes1 contendrá la gráfica del audio ingresado. El Push Button de la izquierda se utilizará para ingresar nuevos patrones mientras que el Push Button del centro se utilizará como salida. Por último el Toggle Button de la derecha se utilizará para iniciar o salir del modo audio donde se ingresar los comandos a identificar.
Paso siguiente se cambiará el texto, color, tamaño, forma, posición, tipo y color de letra etc. de cada uno de los objetos que introdujimos en el entorno según necesidades y gustos personales. Para esto debemos mencionar que haciendo doble clic sobre cualquiera de los objetos accedemos a una tabla con todas las propiedades del mismo para su modificación. La tabla que se despliega es la que se muestra abajo.



El entorno finalmente después de todas las modificaciones a los objetos queda como se muestra abajo:



Por el simple hecho de haber creado un marco en el cual ir colocándole botones, y a este haberle definido un color un tamaño, una posición, etc. MATLab va a generar automáticamente en el editor una estructura de código que justifica todas nuestras decisiones gráficas o visuales. Por lo que en el editor con lo primero que nos vamos a encontrar es con el siguiente código.

```
function varargout = GUI1(varargin)
% Last Modified by GUIDE v2.5 28-Sep-2011 12:54:20
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
                   'gui_Singleton',  gui_Singleton, ...
                   'gui_OpeningFcn', @GUI1_OpeningFcn, ...
                   'gui_OutputFcn',  @GUI1_OutputFcn, ...
                   'gui_LayoutFcn',  [] , ...
                   'gui_Callback',   []);
if nargin & isstr(varargin{1})
    gui_State.gui_Callback = str2func(varargin{1});
end

if nargout
    [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
else
    gui_mainfcn(gui_State, varargin{:});
end
```

```
% End initialization code - DO NOT EDIT

% --- Executes just before GUI1 is made visible.
function GUI1_OpeningFcn(hObject, eventdata, handles, varargin)
handles.output = hObject;

% Update handles structure
guidata(hObject, handles);
% UIWAIT makes GUI1 wait for user response (see UIRESUME)
% uiwait(handles.figure1);


% --- Outputs from this function are returned to the command line.
function varargout = GUI1_OutputFcn(hObject, eventdata, handles)

varargout{1} = handles.output;
```

MATLab creo automáticamente todo este código solo porque nosotros creamos visualmente lo siguiente:

Cuando agregamos el Push Button que utilizaremos para salir MATLab agrega este código:

```
% --- Executes on button press in Salir.
function Salir_Callback(hObject, eventdata, handles, varargin)
guidata(hObject, handles);
```

Entre las dos líneas de código que aparecen automáticamente el usuario deberá agregar todas las instrucciones que efectivicen la salida del programa cuando se presiona dicho botón.

Esta es la estructura de para cuando presionamos el botón de ingresar un nuevo patrón. Recordemos que debemos agregar nuevas instrucciones para efectivizar las acciones. En este caso deberemos llamar a la función *ingresoaudio()* que se muestra más abajo en los códigos fuentes.

```
% --- Executes on button press in Patron_nuevo.
function Patron_nuevo_Callback(hObject, eventdata, handles)
guidata(hObject, handles);
```

Para los cuatro Popup Menu se crean las siguientes estructuras:

```
% --- Executes during object creation, after setting all properties.
function popupmenu1_CreateFcn(hObject, eventdata, handles)
if ispc
    set(hObject,'BackgroundColor','white');
else
    set(hObject,'BackgroundColor',get(0,'defaultUicontrolBackgroundColor'));
end
% --- Executes on selection change in popupmenu1.
function popupmenu1_Callback(hObject, eventdata, handles)
guidata(hObject, handles);

% --- Executes during object creation, after setting all properties.
function popupmenu3_CreateFcn(hObject, eventdata, handles)
if ispc
    set(hObject,'BackgroundColor','white');
else
    set(hObject,'BackgroundColor',get(0,'defaultUicontrolBackgroundColor'));
end

% --- Executes on selection change in popupmenu3.
function popupmenu3_Callback(hObject, eventdata, handles)
guidata(hObject, handles);

% --- Executes during object creation, after setting all properties.
function popupmenu4_CreateFcn(hObject, eventdata, handles)
if ispc
    set(hObject,'BackgroundColor','white');
else
    set(hObject,'BackgroundColor',get(0,'defaultUicontrolBackgroundColor'));
end

% --- Executes on selection change in popupmenu4.
function popupmenu4_Callback(hObject, eventdata, handles)
guidata(hObject, handles);

% --- Executes during object creation, after setting all properties.
function popupmenu2_CreateFcn(hObject, eventdata, handles)
if ispc
    set(hObject,'BackgroundColor','white');
```

```
else
    set(hObject,'BackgroundColor',get(0,'defaultUicontrolBackgroundColor'));
end

% --- Executes on selection change in popupmenu2.
function popupmenu2_Callback(hObject, eventdata, handles)
guidata(hObject, handles);
```

Para el Slider o barra de desplazamiento obtenemos de MATLab lo siguiente:

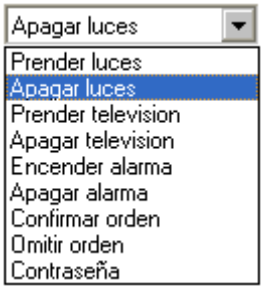
```
% --- Executes during object creation, after setting all properties.
function slider2_CreateFcn(hObject, eventdata, handles)
handles.sliderboton=load ('slider2.mat');
val=(handles.sliderboton.sld2-2)/5;
set(hObject,'Value',val);
usewhitebg = 1;
if usewhitebg
    set(hObject,'BackgroundColor',[.9 .9 .9]);
else
    set(hObject,'BackgroundColor',get(0,'defaultUicontrolBackgroundColor'));
end


% --- Executes on slider movement.
function slider2_Callback(hObject, eventdata, handles)
guidata(hObject, handles);
```

Y para el botón de Reconocimiento se obtiene la siguiente estructura. Para que cuando se presione dicho botón el programa realice las acciones que queremos, deberemos colocar el código donde se encuentra (*).

```
% --- Executes on button press in togglebutton4.
function togglebutton4_Callback(hObject, eventdata, handles)
(*)
guidata(hObject, handles);
```

Cuando presionamos el botón salir se despliega el cartel que se muestra abajo. Si optamos por la opción Si se cierran todas las ventanas del programa mientras que si presionamos No podemos continuar con la ejecución del mismo.

Este es el menú pop up que se desplegará al presionar sobre uno de ellos. Como se ve aparecen las 6 órdenes a identificar, más las ordenes de confirmación u omisión para cuando se desea Encender o Apagar la alarma y la contraseña para abandonar el modo Reconocer.



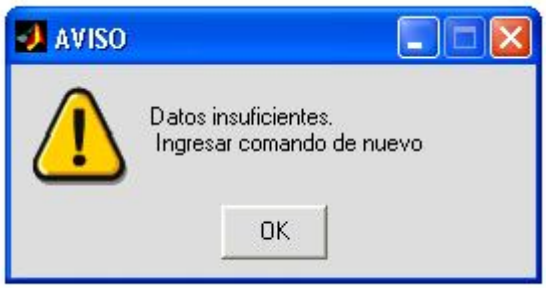
En el caso de presionar el botón de Patrón nuevo y después de haber ingresado el mismo, o sea, transcurridos los dos segundos a partir de que se lo pulso, se observará el siguiente cartel.

Si después de haber ingresado un patrón nuevo elegimos el usuario y la frase donde queremos que se guarde en los menús pop up se mostrará lo siguiente:

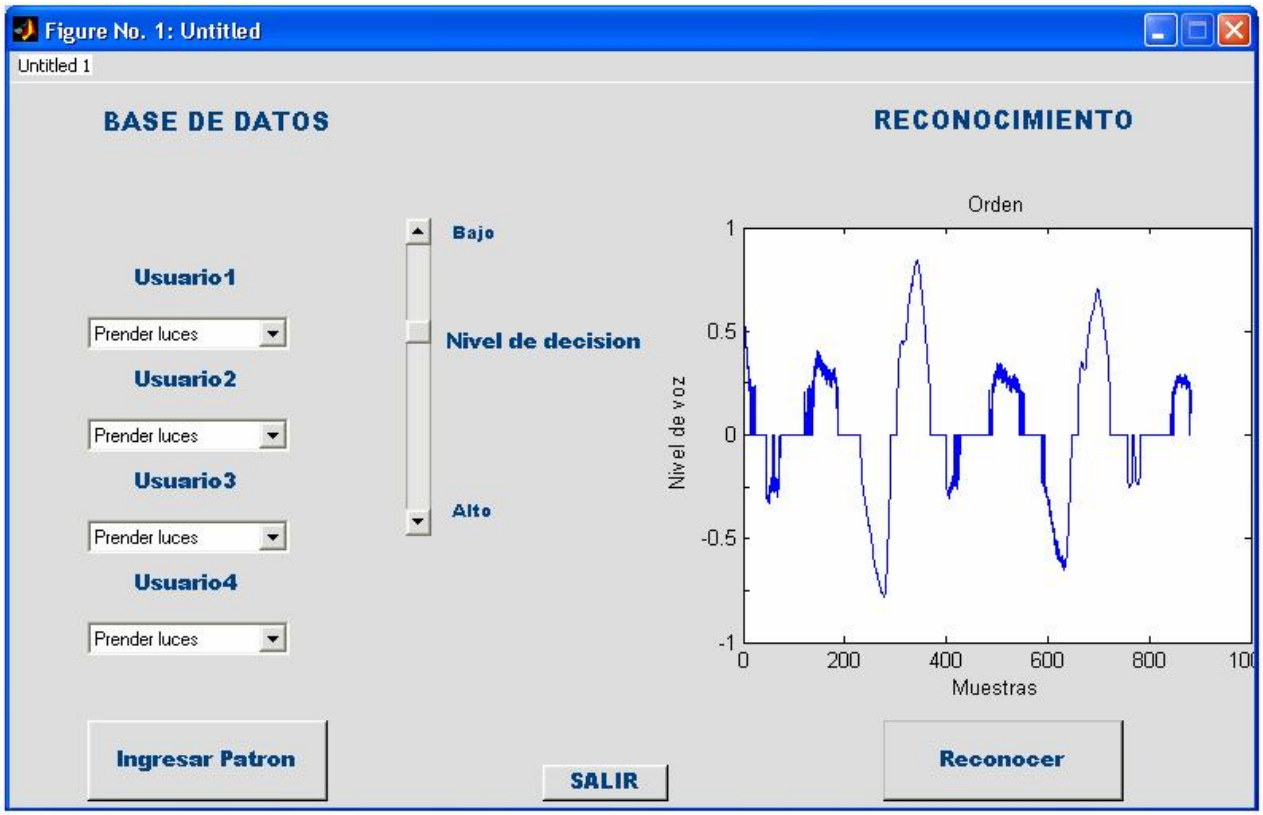
En caso de que se presione sobre los menús pop up y no se haya ingresado con anterioridad un patrón de audio nuevo se mostrará el siguiente mensaje de error:

Si recordamos las explicaciones dadas en la parte de Parametrización, se obtenían dos matrices de observación que contenían los coeficientes característicos de la frase. Como la primera se obtenía a partir de la señal de audio sin silencio e iba desde la muestra 1 a la última menos 26000, para que no haya un error en el manejo de las matrices y los vectores, la señal de audio sin silencio debería

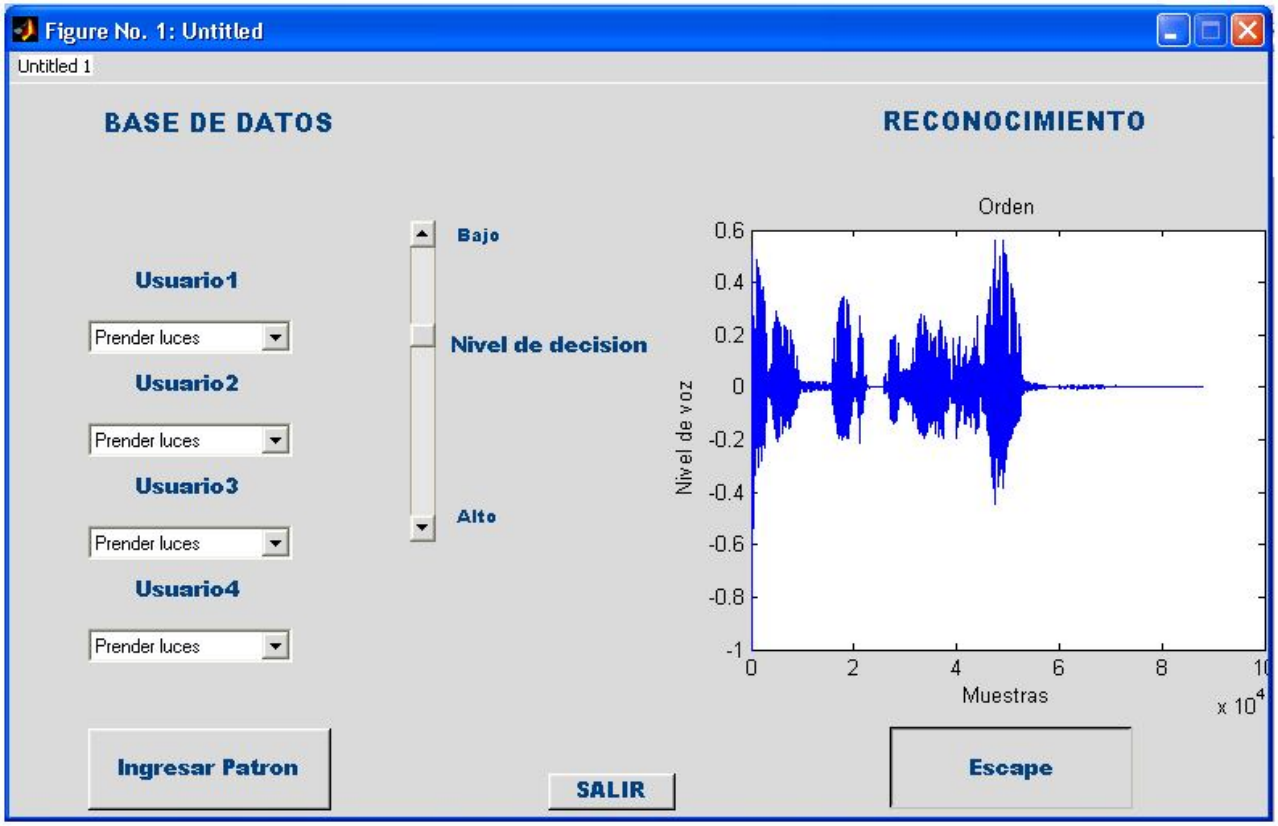
tener más de 26000 muestras. Esto es porque sino se podría indicar que el análisis se hiciera hasta la muestra (end-2600) cuyo resultado sería negativo si el número de muestras fuera menor a 26000 y produciría error. De esta manera se tomo como condición de compromiso que el archivo de audio que se genera cuando se ingresa un nuevo patrón o cuando se ingresa una frase a reconocer tenga como mínimo 28000 muestras. Si esta condición no se cumple de despliega el siguiente cartel que obliga a ingresar una nueva orden o patrón.



La siguiente imagen nos muestra la gráfica del audio de una orden a medida que dicha orden es reproducida por el locutor.



Para cuando el locutor termino la orden dentro de los dos segundos que le da el sistema a partir que inició a reproducirla, el sistema muestra la gráfica de la orden completa en el tiempo.



Es necesario aclarar que cuando comienza la adquisición y hasta finalizada la gráfica total de la señal (un par de segundos) el botón que en un principio se denominaba Reconocer cambia su leyenda a Escape. Esto se debe a que además de poder abandonar el modo audio de reconocimiento mediante la reproducción de la contraseña es posible abandonarlo presionando sobre el Toggle Button cuando en este figura la leyenda Escape. Esto sucede por un corto período cada vez que se detecta actividad en el micrófono, para luego figurar nuevamente la leyenda Reconocer. Para poder abandonar el modo audio por este método no es necesario decir la contraseña o alguna de las frases estipuladas, basta con que el sistema reconozca presencia de voz en el micrófono y el botón cambia su funcionalidad, y al presionarlo se abandona dicho modo.

CONCLUSIONES

En el presente problema hemos podido investigar sobre los diferentes métodos de reconocimiento del habla y experimentado con ellos. Si bien descubrimos que no existe un método que identifique o reconozca en el 100% de los casos o en cualquier situación, limitando el alcance de la aplicación y bajo condiciones controladas hemos obtenido resultados más que satisfactorios.
Por otro lado pudimos aprender a desarrollar entornos gráficos con MATLab lo que facilita en gran medida la interacción con diferentes programas.
Finalmente el problema nos deja la puerta abierta a futuras revisiones y mejoras de cara a proyectos un poco más ambiciosos.

ANEXOS

LISTADOS DE PROGRAMAS

El primer programa que se mostrará es la GUI del sistema. A partir de la selección de los diferentes botones de la aplicación se irán ejecuntando sus correspondientes funciones CallBack. Dentro de las mismas habrá otras funciones que se mostrarán luego.

GUI.m:

```
function varargout = GUI1(varargin)
% Last Modified by GUIDE v2.5 28-Sep-2011 12:54:20
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
                   'gui_Singleton',  gui_Singleton, ...
                   'gui_OpeningFcn', @GUI1_OpeningFcn, ...
                   'gui_OutputFcn',  @GUI1_OutputFcn, ...
                   'gui_LayoutFcn',  [] , ...
                   'gui_Callback',   []);
if nargin & isstr(varargin{1})
   gui_State.gui_Callback = str2func(varargin{1});
end

if nargout
   [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
else
   gui_mainfcn(gui_State, varargin{:});
end
% End initialization code - DO NOT EDIT

% --- Executes just before GUI1 is made visible.
function GUI1_OpeningFcn(hObject, eventdata, handles, varargin)
% Asignación del objeto axes para la imagen adjunta.
%axes(handles.imagen)
%[x,map]=imread('voip-icomuni.jpg','jpg');
%image(x),colormap(map),axis off,hold on
axes(handles.grafica)
handles.grafica=1:1:882;
handles.h = plot(handles.grafica,zeros(882,1));
title('Orden')
xlabel('Muestras')
ylabel('Nivel de voz')
%Carga_Patron;
handles.estruc_U11=load ('prenluzU1.mat');
handles.estruc_U12=load ('apagluzU1.mat');
handles.estruc_U13=load ('prenvisionU1.mat');
handles.estruc_U14=load ('apagvisionU1.mat');
handles.estruc_U15=load ('encealarmU1.mat');
handles.estruc_U16=load ('apagalarmU1.mat');
handles.estruc_U17=load ('confencU1.mat');
```

```
handles.estruc_U18=load ('confapaU1.mat');
handles.estruc_U19=load ('contraseñaU1.mat');
handles.estruc_U21=load ('prenluzU2.mat');
handles.estruc_U22=load ('apagluzU2.mat');
handles.estruc_U23=load ('prenvisionU2.mat');
handles.estruc_U24=load ('apagvisionU2.mat');
handles.estruc_U25=load ('encealarmU2.mat');
handles.estruc_U26=load ('apagalarmU2.mat');
handles.estruc_U27=load ('confencU2.mat');
handles.estruc_U28=load ('confapaU2.mat');
handles.estruc_U29=load ('contraseñaU2.mat');
handles.estruc_U31=load ('prenluzU3.mat');
handles.estruc_U32=load ('apagluzU3.mat');
handles.estruc_U33=load ('prenvisionU3.mat');
handles.estruc_U34=load ('apagvisionU3.mat');
handles.estruc_U35=load ('encealarmU3.mat');
handles.estruc_U36=load ('apagalarmU3.mat');
handles.estruc_U37=load ('confencU3.mat');
handles.estruc_U38=load ('confapaU3.mat');
handles.estruc_U39=load ('contraseñaU3.mat');
handles.estruc_U41=load ('prenluzU4.mat');
handles.estruc_U42=load ('apagluzU4.mat');
handles.estruc_U43=load ('prenvisionU4.mat');
handles.estruc_U44=load ('apagvisionU4.mat');
handles.estruc_U45=load ('encealarmU4.mat');
handles.estruc_U46=load ('apagalarmU4.mat');
handles.estruc_U47=load ('confencU4.mat');
handles.estruc_U48=load ('confapaU4.mat');
handles.estruc_U49=load ('contraseñaU4.mat');
handles.ctrl1=0;
handles.ctrl2=0;
handles.ctrl3=0;
handles.ctrl4=0;
handles.indice=0;
handles.sliderboton=load ('slider2.mat');
handles.output = hObject;

% Update handles structure
guidata(hObject, handles);
% UIWAIT makes GUI1 wait for user response (see UIRESUME)
% uiwait(handles.figure1);


% --- Outputs from this function are returned to the command line.
function varargout = GUI1_OutputFcn(hObject, eventdata, handles)

varargout{1} = handles.output;

% --- Executes on button press in Salir.
function Salir_Callback(hObject, eventdata, handles, varargin)
```

```
ans=questdlg('¿Desea salir del programa?','SALIR','Si','No','No');
if strcmp(ans,'No')
  guidata(hObject, handles);
   return;
end
clear,clc,close all

% --- Executes on button press in Patron_nuevo.
function Patron_nuevo_Callback(hObject, eventdata, handles)
set(handles.Patron_nuevo,'enable','inactive');
%set(handles.slider2,'enable','inactive');
set(handles.Salir,'enable','inactive');
set(handles.popupmenu1,'enable','inactive');
set(handles.popupmenu2,'enable','inactive');
set(handles.popupmenu3,'enable','inactive');
set(handles.popupmenu4,'enable','inactive');
[data2]=ingresoaudio;
wavwrite(data2,44100,16,'patron.wav');
ua4=wavread('patron.wav');
set(handles.Patron_nuevo,'enable','on');
%set(handles.slider2,'enable','on');
set(handles.Salir,'enable','on');
set(handles.popupmenu1,'enable','on');
set(handles.popupmenu2,'enable','on');
set(handles.popupmenu3,'enable','on');
set(handles.popupmenu4,'enable','on');
B = FIR1(800,[100 8000]/22050);
patron=filter(B,1,ua4);
Aa=0;
[cp1,cp2,Aa]=reconocedor(patron);
if Aa(1)>28000
  handles.patron_grabado1 = cp1;
  handles.patron_grabado2 = cp2;
  handles.patron_senial = ua4;
  handles.indice = sum(patron)/88200;
  msgbox('Patron ingresado');
  guidata(hObject, handles);
else
   warndlg({'Datos insuficientes.',' Ingresar comando de nuevo'} ,'AVISO');
end
guidata(hObject, handles);

% --- Executes during object creation, after setting all properties.
function popupmenu1_CreateFcn(hObject, eventdata, handles)
if ispc
   set(hObject,'BackgroundColor','white');
else
   set(hObject,'BackgroundColor',get(0,'defaultUicontrolBackgroundColor'));
end
% --- Executes on selection change in popupmenu1.
```

```
function popupmenu1_Callback(hObject, eventdata, handles)
P1=get(handles.popupmenu1,'Value');
if (handles.ctrl1~=handles.indice)
   handles.ctrl1=handles.indice;
   ctrl1=handles.indice;
   c1 = handles.patron_grabado1;
   c2 = handles.patron_grabado2;
   U = handles.patron_senial;
   switch P1
      case 1,
          save('prenluzU1.mat','c1','c2','U');
          handles.estruc_U11=load ('prenluzU1.mat');
      case 2,
          save('apagluzU1.mat','c1','c2','U');
          handles.estruc_U12=load ('apagluzU1.mat');
       case 3,
          save('prenvisionU1.mat','c1','c2','U');
          handles.estruc_U13=load ('prenvisionU1.mat');
      case 4,
          save('apagvisionU1.mat','c1','c2','U');
          handles.estruc_U14=load ('apagvisionU1.mat');
      case 5,
          save('encealarmU1.mat','c1','c2','U');
          handles.estruc_U15=load ('encealarmU1.mat');
      case 6,
          save('apagalarmU1.mat','c1','c2','U');
          handles.estruc_U16=load ('apagalarmU1.mat');
      case 7,
          save('confencU1.mat','c1','c2','U');
          handles.estruc_U17=load ('confencU1.mat');
      case 8,
          save('confapaU1.mat','c1','c2','U');
          handles.estruc_U18=load ('confapaU1.mat');
      case 9,
          save('contraseñaU1.mat','c1','c2','U');
          handles.estruc_U19=load ('contraseñaU1.mat');
    end
 msgbox('Patron Guardado');
else
  warndlg('Primero ingrese el Patron' ,'AVISO');
end
guidata(hObject, handles);
% --- Executes during object creation, after setting all properties.
function popupmenu3_CreateFcn(hObject, eventdata, handles)
if ispc
   set(hObject,'BackgroundColor','white');
else
   set(hObject,'BackgroundColor',get(0,'defaultUicontrolBackgroundColor'));
end
```

```
% --- Executes on selection change in popupmenu3.
function popupmenu3_Callback(hObject, eventdata, handles)
P3=get(handles.popupmenu3,'Value');
if handles.ctrl3~=handles.indice
   handles.ctrl3=handles.indice;
   c1 = handles.patron_grabado1;
   c2 = handles.patron_grabado2;
   U = handles.patron_senial;
   switch P3
      case 1,
         save('prenluzU3.mat','c1','c2','U');
         handles.estruc_U31=load ('prenluzU3.mat');
       case 2,
         save('apagluzU3.mat','c1','c2','U');
         handles.estruc_U32=load ('apagluzU3.mat');
      case 3,
         save('prenvisionU3.mat','c1','c2','U');
         handles.estruc_U33=load ('prenvisionU3.mat');
      case 4,
         save('apagvisionU3.mat','c1','c2','U');
         handles.estruc_U34=load ('apagvisionU3.mat');
      case 5,
         save('encealarmU3.mat','c1','c2','U');
         handles.estruc_U35=load ('encealarmU3.mat');
      case 6,
         save('apagalarmU3.mat','c1','c2','U');
         handles.estruc_U36=load ('apagalarmU3.mat');
      case 7,
         save('confencU3.mat','c1','c2','U');
         handles.estruc_U37=load ('confencU3.mat');
      case 8,
         save('confapaU3.mat','c1','c2','U');
         handles.estruc_U38=load ('confapaU3.mat');
      case 9,
         save('contraseñaU3.mat','c1','c2','U');
         handles.estruc_U39=load ('contraseñaU3.mat');
    end
 msgbox('Patron Guardado');
 else
 warndlg('Primero ingrese el Patron' ,'AVISO');
end
guidata(hObject, handles);

% --- Executes during object creation, after setting all properties.
function popupmenu4_CreateFcn(hObject, eventdata, handles)
if ispc
   set(hObject,'BackgroundColor','white');
else
   set(hObject,'BackgroundColor',get(0,'defaultUicontrolBackgroundColor'));
```

```
end


% --- Executes on selection change in popupmenu4.
function popupmenu4_Callback(hObject, eventdata, handles)
P4=get(handles.popupmenu4,'Value');
if handles.ctrl4~=handles.indice
   handles.ctrl4=handles.indice;
    c1 = handles.patron_grabado1;
    c2 = handles.patron_grabado2;
    U = handles.patron_senial;
    switch P4
        case 1,
            save('prenluzU4.mat','c1','c2','U');
            handles.estruc_U41=load ('prenluzU4.mat');
         case 2,
            save('apagluzU4.mat','c1','c2','U');
            handles.estruc_U42=load ('apagluzU4.mat');
        case 3,
            save('prenvisionU4.mat','c1','c2','U');
            handles.estruc_U43=load ('prenvisionU4.mat');
        case 4,
            save('apagvisionU4.mat','c1','c2','U');
            handles.estruc_U44=load ('apagvisionU4.mat');
        case 5,
            save('encealarmU4.mat','c1','c2','U');
            handles.estruc_U45=load ('encealarmU4.mat');
        case 6,
            save('apagalarmU4.mat','c1','c2','U');
            handles.estruc_U46=load ('apagalarmU4.mat');
        case 7,
            save('confencU4.mat','c1','c2','U');
            handles.estruc_U47=load ('confencU4.mat');
        case 8,
            save('confapaU4.mat','c1','c2','U');
            handles.estruc_U48=load ('confapaU4.mat');
        case 9,
            save('contraseñaU4.mat','c1','c2','U');
            handles.estruc_U49=load ('contraseñaU4.mat');
     end
 msgbox('Patron Guardado');
 else
 warndlg('Primero ingrese el Patron' ,'AVISO');
end
guidata(hObject, handles);

% --- Executes during object creation, after setting all properties.
function popupmenu2_CreateFcn(hObject, eventdata, handles)
if ispc
    set(hObject,'BackgroundColor','white');
```

```
else
    set(hObject,'BackgroundColor',get(0,'defaultUicontrolBackgroundColor'));
end


% --- Executes on selection change in popupmenu2.
function popupmenu2_Callback(hObject, eventdata, handles)

P2=get(handles.popupmenu2,'Value');
if handles.ctrl2~=handles.indice
    handles.ctrl2=handles.indice;
    c1 = handles.patron_grabado1;
    c2 = handles.patron_grabado2;
    U = handles.patron_senial;
    switch P2
        case 1,
            save('prenluzU2.mat','c1','c2','U');
            handles.estruc_U21=load ('prenluzU2.mat');
         case 2,
            save('apagluzU2.mat','c1','c2','U');
            handles.estruc_U22=load ('apagluzU2.mat');
        case 3,
            save('prenvisionU2.mat','c1','c2','U');
            handles.estruc_U23=load ('prenvisionU2.mat');
         case 4,
            save('apagvisionU2.mat','c1','c2','U');
            handles.estruc_U24=load ('apagvisionU2.mat');
        case 5,
            save('encealarmU2.mat','c1','c2','U');
            handles.estruc_U25=load ('encealarmU2.mat');
        case 6,
            save('apagalarmU2.mat','c1','c2','U');
            handles.estruc_U26=load ('apagalarmU2.mat');
        case 7,
            save('confencU2.mat','c1','c2','U');
            handles.estruc_U27=load ('confencU2.mat');
        case 8,
            save('confapaU2.mat','c1','c2','U');
            handles.estruc_U28=load ('confapaU2.mat');
        case 9,
            save('contraseñaU2.mat','c1','c2','U');
            handles.estruc_U29=load ('contraseñaU2.mat');
     end
 msgbox('Patron Guardado');
else
 warndlg('Primero ingrese el Patron' ,'AVISO');
end
guidata(hObject, handles);
```

```
% --- Executes during object creation, after setting all properties.
function slider2_CreateFcn(hObject, eventdata, handles)
% hObject    handle to slider2 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called
% Hint: slider controls usually have a light gray background, change
%       'usewhitebg' to 0 to use default.  See ISPC and COMPUTER.
handles.sliderboton=load ('slider2.mat');
val=(handles.sliderboton.sld2-2)/5;
set(hObject,'Value',val);
usewhitebg = 1;
if usewhitebg
   set(hObject,'BackgroundColor',[.9 .9 .9]);
else
   set(hObject,'BackgroundColor',get(0,'defaultUicontrolBackgroundColor'));
end


% --- Executes on slider movement.
function slider2_Callback(hObject, eventdata, handles)
% hObject    handle to slider2 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'Value') returns position of slider
%        get(hObject,'Min') and get(hObject,'Max') to determine range of slider
handles.slider2=get(hObject,'Value'); %Carga en handles.slider2 el valor delSlider
handles.slider2=handles.slider2.*5+2;
sld2=handles.slider2
save('slider2.mat','sld2');
handles.sliderboton=load ('slider2.mat');
guidata(hObject, handles);

% --- Executes on button press in togglebutton4.
function togglebutton4_Callback(hObject, eventdata, handles)
% hObject    handle to togglebutton4 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hint: get(hObject,'Value') returns toggle state of togglebutton4
handles.toogle=get(hObject,'Value');
toogle=handles.toogle;
set(handles.Patron_nuevo,'enable','inactive');
%set(handles.slider2,'enable','inactive');
set(handles.Salir,'enable','inactive');
set(handles.popupmenu1,'enable','inactive');
set(handles.popupmenu2,'enable','inactive');
set(handles.popupmenu3,'enable','inactive');
set(handles.popupmenu4,'enable','inactive');
set(handles.togglebutton4,'string','Reconocer');
```

```
Comp(1)=handles.estruc_U11;
Comp(2)=handles.estruc_U12;
Comp(3)=handles.estruc_U13;
Comp(4)=handles.estruc_U14;
Comp(5)=handles.estruc_U15;
Comp(6)=handles.estruc_U16;
Comp(7)=handles.estruc_U21;
Comp(8)=handles.estruc_U22;
Comp(9)=handles.estruc_U23;
Comp(10)=handles.estruc_U24;
Comp(11)=handles.estruc_U25;
Comp(12)=handles.estruc_U26;
Comp(13)=handles.estruc_U31;
Comp(14)=handles.estruc_U32;
Comp(15)=handles.estruc_U33;
Comp(16)=handles.estruc_U34;
Comp(17)=handles.estruc_U35;
Comp(18)=handles.estruc_U36;
Comp(19)=handles.estruc_U41;
Comp(20)=handles.estruc_U42;
Comp(21)=handles.estruc_U43;
Comp(22)=handles.estruc_U44;
Comp(23)=handles.estruc_U45;
Comp(24)=handles.estruc_U46;
Comp(25)=handles.estruc_U19;
Comp(26)=handles.estruc_U29;
Comp(27)=handles.estruc_U39;
Comp(28)=handles.estruc_U49;
Comp(29)=handles.estruc_U17;
Comp(30)=handles.estruc_U27;
Comp(31)=handles.estruc_U37;
Comp(32)=handles.estruc_U47;
Comp(33)=handles.estruc_U18;
Comp(34)=handles.estruc_U28;
Comp(35)=handles.estruc_U38;
Comp(36)=handles.estruc_U48;
%
for ff=1:1:1000
  handles.grafica=1:1:882;
  handles.h = plot(handles.grafica,zeros(882,1));
  h=handles.h;
  graf=handles.grafica;
  data=vozgrabar(graf,h,hObject,handles);
  [m,n]=size(data);
  if [m,n]~=[88200,1]
     return
  elseif isempty(data)==1
     return
  else
   wavwrite(data,44100,16,'test.wav');
```

```
ua2=wavread('test.wav');
B = FIR1(800,[100 8000]/22050);
test=filter(B,1,ua2);
[c01,c02,Ab]=reconocedor(test);
handles.toogle=get(hObject,'Value');
toogle=handles.toogle;
if Ab(1)>28000 && toogle==1
 for i=1:28
   d1(i)=disteu(Comp(i).c1,c01);
   d2(i)=disteu(Comp(i).c2,c02);
   dt(i)=(d1(i)+d2(i))/2;
 end
 [DD,Ind]=min(dt);
 dt
 AA=handles.sliderboton.sld2;
 if DD<AA
   if Ind==1 || Ind==7 || Ind==13 || Ind==19
     soundsc(Comp(Ind).U,44100);
   elseif (Ind==2 || Ind==8 || Ind==14 || Ind==20)
     soundsc(Comp(Ind).U,44100);
   elseif (Ind==3 || Ind==9 || Ind==15 || Ind==21)
     soundsc(Comp(Ind).U,44100);
   elseif (Ind==4 || Ind==10 || Ind==16 || Ind==22)
     soundsc(Comp(Ind).U,44100);
   elseif (Ind==5 || Ind==11 || Ind==17 || Ind==23)
     conf=wavread('confirmacion_encender.wav');
     soundsc(conf,44100);
     pause(2.2);
     DD
     x1=(1:1:15000)/44100;
     w=sin(2*pi*800.*x1);
     sound(w,44100);
     confencend = ingresoaudio;
     wavwrite(confencend,44100,16,'confirmar.wav');
     ua6=wavread('confirmar.wav');
     B = FIR1(800,[100 8000]/22050);
     confirmar=filter(B,1,ua6);
     [co1,co2,Ab2]=reconocedor(confirmar);
     for j=1:8
        d3(j)=disteu(Comp(j+28).c1,co1);
        d4(j)=disteu(Comp(j+28).c2,co2);
        dt2(j)=(d3(j)+d4(j))/2;
     end
     [DD2,Ind2]=min(dt2)
     dt2
     if DD2>AA
        uw=wavread('Patron_desco.wav');
        soundsc(uw,44100);
     elseif (Ind2==1 || Ind2==2 || Ind2==3 || Ind2==4)
        soundsc(Comp(Ind).U,44100);
```

```
    else
        soundsc(Comp(Ind2+28).U,44100);
    end
    clear DD2 d3 d4 dt2 Ind2 Ab2 x1 w ua6 conf confirmar confencend;
  elseif (Ind==6 || Ind==12 || Ind==18 || Ind==24)
    conf=wavread('confirmacion_apagar.wav');
    soundsc(conf,44100);
    pause(2.2);
    x1=(1:1:15000)/44100;
    w=sin(2*pi*800.*x1);
    sound(w,44100);
    confencend = ingresoaudio;
    wavwrite(confencend,44100,16,'confirmar.wav');
    ua6=wavread('confirmar.wav');
    B = FIR1(800,[100 8000]/22050);
    confirmar=filter(B,1,ua6);
    [co1,co2,Ab2]=reconocedor(confirmar);
    for j=1:8
        d3(j)=disteu(Comp(j+28).c1,co1);
        d4(j)=disteu(Comp(j+28).c2,co2);
        dt2(j)=(d3(j)+d4(j))/2;
    end
    [DD2,Ind2]=min(dt2);
    if DD2>AA
        uw=wavread('Patron_desco.wav');
        soundsc(uw,44100);
    elseif (Ind2==1 || Ind2==2 || Ind2==3 || Ind2==4)
        soundsc(Comp(Ind).U,44100);
    else
        soundsc(Comp(Ind+28).U,44100);
    end
    clear DD2 d3 d4 dt2 Ind2 Ab2 x1 w ua6 conf confirmar confencend;
  else
    soundsc(Comp(Ind).U,44100);
    set(handles.Patron_nuevo,'enable','on');
    %set(handles.slider2,'enable','on');
    set(handles.Salir,'enable','on');
    set(handles.popupmenu1,'enable','on');
    set(handles.popupmenu2,'enable','on');
    set(handles.popupmenu3,'enable','on');
    set(handles.popupmenu4,'enable','on');
    set(handles.togglebutton4,'string','Reconocer');
    guidata(hObject, handles);
    return
  end
elseif DD<AA+1.5 && DD>AA  && (Ind==25 || Ind==26 || Ind==27 || Ind==28)
  soundsc(Comp(Ind).U,44100);
  set(handles.togglebutton4,'enable','on');
  handles.toogle=get(hObject,'Value');
  set(handles.Patron_nuevo,'enable','on');
```

```
    %set(handles.slider2,'enable','on');
    set(handles.Salir,'enable','on');
    set(handles.popupmenu1,'enable','on');
    set(handles.popupmenu2,'enable','on');
    set(handles.popupmenu3,'enable','on');
    set(handles.popupmenu4,'enable','on');
    set(handles.togglebutton4,'string','Reconocer');
    guidata(hObject, handles);
    return
   else
    uw=wavread('Patron_desco.wav');
    soundsc(uw,44100);
   end
  elseif toogle==1
   ux=wavread('Datos_insuf.wav');
   soundsc(ux,44100);
  else
     set(handles.togglebutton4,'string','Reconocer');
     return
  end
 set(handles.togglebutton4,'string','Reconocer');
 pause(0.25)
 set(handles.togglebutton4,'string','Reconocer');
 clear data ua2 test Ab c01 c02 DD d1 d2 dt Ind;
 end
end
guidata(hObject, handles);
```

ingresoaudio.m: Está función permite ingresar los nuevos patrones.

```
% FUNCIÓN INGRESO AUDIO
function [data]=ingresoaudio

handles.AI = analoginput('winsound');
chan = addchannel(handles.AI,1);
duration = 2; % Ten second acquisition
set(handles.AI,'SampleRate',44100)
ActualRate = get(handles.AI,'SampleRate');
set(handles.AI,'SamplesPerTrigger',duration*ActualRate)
preview = duration*ActualRate/100;
set(gcf,'doublebuffer','on')
handles.grafica=1:1:882;
handles.h = plot(handles.grafica,zeros(882,1));
title('Orden')
xlabel('Muestras')
ylabel('Nivel de voz')
data =zeros(88200,1);
start(handles.AI)
while handles.AI.SamplesAcquired < preview
end
```

```
while handles.AI.SamplesAcquired < duration*ActualRate
   data = peekdata(handles.AI,preview);
   for i=1:882
      if abs(data(i))<8e-3
      data(i) =0;
      end
   end
   set(handles.h,'ydata',data)
   drawnow
end
data = getdata(handles.AI);
handles.grafica=1:1:88200;
handles.h = plot(handles.grafica,data);
title('Orden')
xlabel('Muestras')
ylabel('Nivel de voz')
delete(handles.AI)
clear handles.AI
```

vozgrabar.m: Esta función permite el ingreso de las frases a ser reconocidas. La particularidad de la misma es que hay un umbral de voz que inicia la adquisición del audio.

```
% FUNCIÓN VOZGRABAR
function [data]=vozgrabar(graf,h,hObject,handles)
AI = analoginput('winsound');
chan = addchannel(AI,1);
duration = 2;
set(AI,'SampleRate',44100);
ActualRate = get(AI,'SampleRate');
set(AI,'SamplesPerTrigger',duration*ActualRate);
set(AI,'TriggerType','software');
set(AI,'TriggerCondition','rising');
set(AI,'TriggerConditionValue',0.13);
set(AI,'TriggerChannel',AI.Channel(1));
set(AI,'TimeOut',600);
preview = duration*ActualRate/100;
set(gcf,'doublebuffer','on');
data =[];
%set(handles.togglebutton4,'string','Reconocer');
start(AI)
%guidata(hObject, handles);
handles.toogle=get(hObject,'Value');
if handles.toogle==0
   set(handles.Patron_nuevo,'enable','on');
   %set(handles.slider2,'enable','on');
   set(handles.Salir,'enable','on');
   set(handles.popupmenu1,'enable','on');
   set(handles.popupmenu2,'enable','on');
   set(handles.popupmenu3,'enable','on');
   set(handles.popupmenu4,'enable','on');
```

```
   %set(handles.togglebutton4,'string','Escape');

   stop(AI)
   delete(AI)
   clear AI
   pause(0.25)
   return
else
  while AI.SamplesAcquired < preview
  end
  while AI.SamplesAcquired < duration*ActualRate
     guidata(hObject, handles);
     handles.toogle=get(hObject,'Value');
     if handles.toogle==0
     stop(AI)
     delete(AI)
     clear AI
     pause(0.25)
     return
  else
      data = peekdata(AI,preview);
      for i=1:1:882
        if abs(data(i))<8e-3
           data(i) =0;
       end
       end
    set(h,'ydata',data);
    drawnow
    set(handles.togglebutton4,'string','Escape');
 end
 end
  data = getdata(AI);
  graf=1:1:88200;
  h = plot(graf,data);
  title('Orden')
  xlabel('Muestras')
  ylabel('Nivel de voz')
end
delete(AI)
clear AI
pause(0.25)
%
```

Reconocedor.m: Llama a diferentes funciones para realizar el análisis del audio ingresado ya sea como patrón o como frase a reconocer.

```
% FUNCIÓN RECONOCEDOR
function [c01,c02,A]=reconocedor(y)
fs=44100;
q=silencio(y);
```

```
A=size(q);
if A(1)>28000
c01=melcepst(q(1:end-22000),fs,'M',14,30,1024,410,0,0.1815);
c02=melcepst(q(end-26000:end),fs,'M',14,30,1024,410,0,0.1815);
else
c01=zeros(31,14);
c02=zeros(61,14);
end
```

silencio.m: Elimina los períodos de silencio de la señal ingresa como patrón o a identificar.

```
% silencio
%Corta el silencio en la señal completa
function y = silencio(s)
len = length(s);% length del vector
d=max(abs(s));
s=s/d;
avg_e = sum(s.*s)/len; %promedio señal entera
THRES = 0.02;
y = [0];
for i = 1:400:len-400 % cada 10ms
seg = s(i:i+399);% segmentos
e = sum(seg.*seg)/400; % promedio de cada segmento
if( e> THRES*avg_e) % si el promedio energético es mayor que la señal
%completa por el valor umbral
y=[y;seg(1:end)];% almacena en y sino es eliminado como espacio en blanco
end;
end;
```

melcepst.m: Esta función permite obtener los coeficientes cepstrales en escala de MEL. Pertenece a la librería Voicebox que se puede descargar de http://www.ee.ic.ac.uk/hp/staff/dmb/voicebox/voicebox.html.

```
function c=melcepst(s,fs,w,nc,p,n,inc,fl,fh)
%MELCEPST Calculate the mel cepstrum of a signal C=(S,FS,W,NC,P,N,INC,FL,FH)
%
%
% Simple use: c=melcepst(s,fs)        % calculate mel cepstrum with 12 coefs, 256 sample frames
%                                     c=melcepst(s,fs,'e0dD') % include log energy, 0th cepstral coef,
delta and delta-delta coefs
%
% Inputs:
%     s   speech signal
%     fs  sample rate in Hz (default 11025)
%     nc  number of cepstral coefficients excluding 0'th coefficient (default 12)
%     n   length of frame in samples (default power of 2 < (0.03*fs))
%     p   number of filters in filterbank (default: floor(3*log(fs)) = approx 2.1 per ocatave)
%     inc frame increment (default n/2)
%     fl  low end of the lowest filter as a fraction of fs (default = 0)
%     fh  high end of highest filter as a fraction of fs (default = 0.5)
```

```
%
%               w   any sensible combination of the following:
%
%                                   'R'  rectangular window in time domain
%                                   'N'      Hanning window in time domain
%                                   'M'      Hamming window in time domain (default)
%
%                           't'  triangular shaped filters in mel domain (default)
%                           'n'  hanning shaped filters in mel domain
%                           'm'  hamming shaped filters in mel domain
%
%                                   'p'      filters act in the power domain
%                                   'a'      filters act in the absolute magnitude domain (default)
%
%                                '0'  include 0'th order cepstral coefficient
%                                   'e'  include log energy
%                                   'd'      include delta coefficients (dc/dt)
%                                   'D'      include delta-delta coefficients (d^2c/dt^2)
%
%                           'z'  highest and lowest filters taper down to zero (default)
%                           'y'  lowest filter remains at 1 down to 0 frequency and
%                                        highest filter remains at 1 up to nyquist freqency
%
%                           If 'ty' or 'ny' is specified, the total power in the fft is preserved.
%
% Outputs:      c     mel cepstrum output: one frame per row. Log energy, if requested, is the
%               first element of each row followed by the delta and then the delta-delta
%               coefficients.
%

% BUGS: (1) should have power limit as 1e-16 rather than 1e-6 (or possibly a better way of
choosing this)
%         and put into VOICEBOX
%       (2) get rdct to change the data length (properly) instead of doing it explicitly (wrongly)

%     Copyright (C) Mike Brookes 1997
%     Version: $Id: melcepst.m,v 1.7 2009/10/19 10:20:32 dmb Exp $
%
%   VOICEBOX is a MATLAB toolbox for speech processing.
%   Home page: http://www.ee.ic.ac.uk/hp/staff/dmb/voicebox/voicebox.html
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%   This program is free software; you can redistribute it and/or modify
%   it under the terms of the GNU General Public License as published by
%   the Free Software Foundation; either version 2 of the License, or
%   (at your option) any later version.
%
%   This program is distributed in the hope that it will be useful,
%   but WITHOUT ANY WARRANTY; without even the implied warranty of
```

```
%   MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
%   GNU General Public License for more details.
%
%   You can obtain a copy of the GNU General Public License from
%   http://www.gnu.org/copyleft/gpl.html or by writing to
%   Free Software Foundation, Inc.,675 Mass Ave, Cambridge, MA 02139, USA.
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

if nargin<2 fs=11025; end
if nargin<3 w='M'; end
if nargin<4 nc=12; end
if nargin<5 p=floor(3*log(fs)); end
if nargin<6 n=pow2(floor(log2(0.03*fs))); end
if nargin<9
  fh=0.5;
  if nargin<8
    fl=0;
    if nargin<7
      inc=floor(n/2);
    end
  end
end

if length(w)==0
  w='M';
end
if any(w=='R')
  z=enframe(s,n,inc);
elseif any (w=='N')
  z=enframe(s,hanning(n),inc);
else
  z=enframe(s,hamming(n),inc);
end
f=rfft(z.');
[m,a,b]=melbankm2(p,n,fs,fl,fh,w);
pw=f(a:b,:).*conj(f(a:b,:));
pth=max(pw(:))*1E-20;
if any(w=='p')
  y=log(max(m*pw,pth));
else
  ath=sqrt(pth);
  y=log(max(m*abs(f(a:b,:)),ath));
end
c=rdct(y).';
nf=size(c,1);
nc=nc+1;
if p>nc
  c(:,nc+1:end)=[];
elseif p<nc
```

```
  c=[c zeros(nf,nc-p)];
end
if ~any(w=='0')
  c(:,1)=[];
  nc=nc-1;
end
if any(w=='e')
  c=[log(sum(pw)).' c];
  nc=nc+1;
end

% calculate derivative

if any(w=='D')
 vf=(4:-1:-4)/60;
 af=(1:-1:-1)/2;
 ww=ones(5,1);
 cx=[c(ww,:); c; c(nf*ww,:)];
 vx=reshape(filter(vf,1,cx(:)),nf+10,nc);
 vx(1:8,:)=[];
 ax=reshape(filter(af,1,vx(:)),nf+2,nc);
 ax(1:2,:)=[];
 vx([1 nf+2],:)=[];
 if any(w=='d')
   c=[c vx ax];
 else
   c=[c ax];
 end
elseif any(w=='d')
 vf=(4:-1:-4)/60;
 ww=ones(4,1);
 cx=[c(ww,:); c; c(nf*ww,:)];
 vx=reshape(filter(vf,1,cx(:)),nf+8,nc);
 vx(1:8,:)=[];
 c=[c vx];
end

if nargout<1
  [nf,nc]=size(c);
  t=((0:nf-1)*inc+(n-1)/2)/fs;
  ci=(1:nc)-any(w=='0')-any(w=='e');
  imh = imagesc(t,ci,c.');
  axis('xy');
  xlabel('Time (s)');
  ylabel('Mel-cepstrum coefficient');
        map = (0:63)'/63;
        colormap([map map map]);
        colorbar;
end
```

función disteu.m: Determina el valor distancia entre la matriz característica de la orden a identificar y la matriz característica de un patrón determinado. Este valor se obtiene del vector distancia que le pasa la función disteusq.

```
function dist=disteu(f00,f01)
d=disteusq(f00,f01,'x');
dist=sum(min(d,[],2))/size(d,1);
```

función disteusq.m: Determina el vector distancia entre la matriz característica de la orden a identificar y la matriz característica de un patrón determinado. Esta función pertenece a la librería Voicebox que se puede descargar de http://www.ee.ic.ac.uk/hp/staff/dmb/voicebox/voicebox.html.

```
function d=disteusq(x,y,mode,w)
%DISTEUSQ calculate euclidean, squared euclidean or mahanalobis distance D=(X,Y,MODE,W)
%
% Inputs: X,Y       Vector sets to be compared. Each row contains a data vector.
%                   X and Y must have the same number of columns.
%
%       MODE        Character string selecting the following options:
%                     'x'  Calculate the full distance matrix from every row of X to every row of Y
%                     'd'  Calculate only the distance between corresponding rows of X and Y
%                          The default is 'd' if X and Y have the same number of rows otherwise 'x'.
%                     's'  take the square-root of the result to give the euclidean distance.
%
%       W           Optional weighting matrix: the distance calculated is (x-y)*W*(x-y)'
%                   If W is a vector, then the matrix diag(W) is used.
%
% Output: D         If MODE='d' then D is a column vector with the same number of rows as the
shorter of X and Y.
%                   If MODE='x' then D is a matrix with the same number of rows as X and the same
number of columns as Y'.
%

%     Copyright (C) Mike Brookes 1998
%     Version: $Id: disteusq.m,v 1.6 2010/04/19 16:56:22 dmb Exp $
%
%   VOICEBOX is a MATLAB toolbox for speech processing.
%   Home page: http://www.ee.ic.ac.uk/hp/staff/dmb/voicebox/voicebox.html
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%   This program is free software; you can redistribute it and/or modify
%   it under the terms of the GNU General Public License as published by
%   the Free Software Foundation; either version 2 of the License, or
%   (at your option) any later version.
%
%   This program is distributed in the hope that it will be useful,
%   but WITHOUT ANY WARRANTY; without even the implied warranty of
```

```
%   MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
%   GNU General Public License for more details.
%
%   You can obtain a copy of the GNU General Public License from
%   http://www.gnu.org/copyleft/gpl.html or by writing to
%   Free Software Foundation, Inc.,675 Mass Ave, Cambridge, MA 02139, USA.
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

[nx,p]=size(x); ny=size(y,1);
if nargin<3 | isempty(mode) mode='0'; end
if any(mode=='d') | (mode~='x' & nx==ny)

   % Do pairwise distance calculation

   nx=min(nx,ny);
   z=double(x(1:nx,:))-double(y(1:nx,:));
   if nargin<4
      d=sum(z.*conj(z),2);
   elseif min(size(w))==1
      wv=w(:).';
      d=sum(z.*wv(ones(size(z,1),1),:).*conj(z),2);
   else
      d=sum(z*w.*conj(z),2);
   end
else

   % Calculate full distance matrix

   if p>1

      % x and y are matrices

      if nargin<4
         z=permute(double(x(:,:,ones(1,ny))),[1 3 2])-permute(double(y(:,:,ones(1,nx))),[3 1 2]);
         d=sum(z.*conj(z),3);
      else
         nxy=nx*ny;
         z=reshape(permute(double(x(:,:,ones(1,ny))),[1 3 2])-permute(double(y(:,:,ones(1,nx))),[3 1
2]),nxy,p);
         if min(size(w))==1
            wv=w(:).';
            d=reshape(sum(z.*wv(ones(nxy,1),:).*conj(z),2),nx,ny);
         else
            d=reshape(sum(z*w.*conj(z),2),nx,ny);
         end
      end
   else

      % x and y are vectors
```

```
        z=double(x(:,ones(1,ny)))-double(y(:,ones(1,nx)))';
        if nargin<4
            d=z.*conj(z);
        else
            d=w*z.*conj(z);
        end
    end
end
if any(mode=='s')
    d=sqrt(d);
end
```