UNIVERSIDAD AUTONOMA METROPOLITANA
UNIDAD IZTAPALAPA
CIENCIAS BASICAS E INGENIERIA

“RECONOCIMIENTO DE DIGITOS AISLADOS EN ESPAÑOL”

Realizado por:

Alejandra Sotuyo Espinosa
Ma. Guadalupe Vargas Contreras

Asesoras:

M en C. Alma E. Martínez
M en C. Fabiola Martínez

Noviembre de 2003.

Introducción

Imitar las habilidades naturales del ser humano es sin duda uno de los retos más complejos con los que se enfrentan los científicos e ingenieros que tratan de construir aparatos y sistemas avanzados. En este sentido, resulta muy complicado conseguir máquinas que caminen con naturalidad, que posean articulaciones semejantes a nuestras manos, que puedan reconocer imágenes, capaces de entender el habla, etc. Como máximo nivel de dificultad, relacionado con las metas expuestas, se encuentra el reto de imitar correctamente las capacidades del cerebro humano. Para alcanzar estas metas es necesario unificar conocimientos de muy diversas disciplinas de la ciencia y de la técnica.

El reconocimiento de la voz constituye una parte importante del tratamiento del habla. Las técnicas de reconocimiento más desarrolladas son aquellas comúnmente usadas para el idioma inglés las cuales incluyen el Análisis de Predicción Lineal (LPC) y el Alineamiento Temporal (DTW), estos algoritmos han tenido éxito habiendo sido sometidos a pruebas bajo diversos ambientes.

Debe tenerse en cuenta la importancia de desarrollar estas técnicas para el idioma español pues las características lingüísticas difieren en forma marcada cuando se requiera llevar hacia un reconocimiento más completo. El enfoque que se ha dado en un principio ha sido el reconocer palabras aisladas, es decir que las palabras se pronuncian entre pausas pequeñas de tal forma que el procesamiento se realiza teniendo como unidades lingüísticas las palabras de un vocabulario específico.

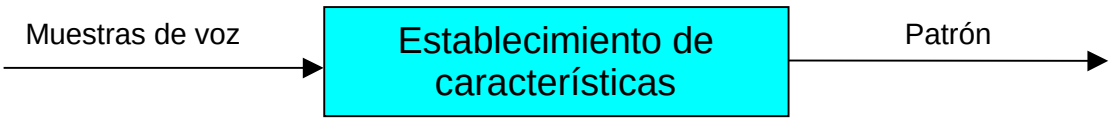
El presente trabajo se enfoca en particular al estudio del funcionamiento de un algoritmo desarrollado para el reconocimiento del habla de los dígitos 0-9 en inglés (Isolated Word Speech Recognition of the English Digits) por Jay Land de la Universidad de Florida en 1999[1].

Basado en la técnica de Alineamiento Temporal Dinámico (Dynamic Time Warping), que es una de las estrategias fundamentales para el reconocimiento del habla y que esta basada en la utilización de la técnica de comparación de patrones.

Naturaleza de la voz

Traspaso de la señal al dominio de la frecuencia.

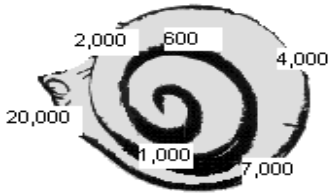
Cuando se expliquen los métodos tradicionales de reconocimiento del habla, se recurrirá a esquemas en los que la primera etapa es la siguiente **(Fig. 1)**:



(Fig. 1)Establecimiento de características espectrales

El establecimiento de características espectrales se refiere, en esta primera etapa, al traspaso de la señal de voz desde el dominio del tiempo al dominio de la frecuencia.
La frecuencia es un término que está íntimamente relacionado con la conversión de una señal analógica a digital. Una señal analógica puede tomar infinitos valores en un momento dado, pero si queremos transformarla en una señal digital, esta solo podrá tomar un número determinado y finito de valores en ese mismo momento. Por lo tanto la Frecuencia de Muestreo es el número de veces por segundo que se muestrea la señal analógica para pasarla a un formato digital.

El oído humano descompone las señales auditivas que le llegan en sus frecuencias fundamentales. Para ello, situado en el oído interno, se encuentra el caracol, que filtra las frecuencias de forma natural. Las ondas sonoras se introducen en esta estructura helicoidal rebotando en sus paredes y llegando, según sea la longitud de onda de cada frecuencia, más o menos al interior del caracol **(Fig. 2).**

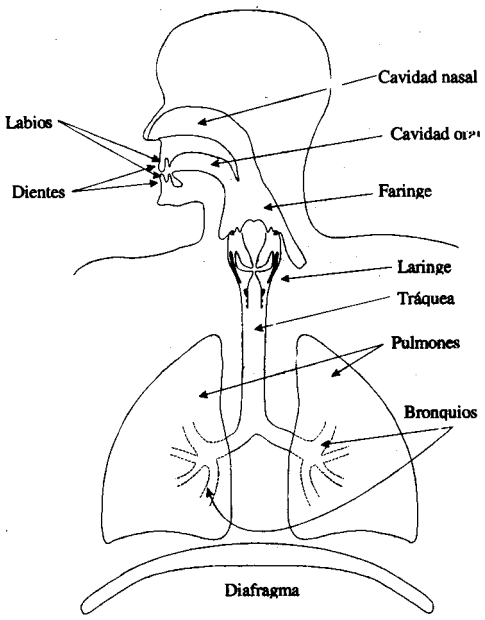


(Fig. 2)

Directamente relacionado con la forma en la que nuestro oído recibe la información, nos encontramos la manera en la que nuestro aparato articulatorio **(Fig. 3)** crea el habla. El mecanismo de producción del habla, brevemente resumido es el siguiente:

1. El diafragma empuja los pulmones, haciendo que se expulse el aire.
2. El aire circula por la tráquea y laringe, pasando por las cuerdas vocales y haciendo que vibren con un tono fundamental.
3. El tono fundamental producido por las cuerdas vocales pasa, a través de la laringe, a la caja de resonancia que forman las cavidades nasales y orales.

4. Algunas frecuencias entran en resonancia en las cavidades nasales y orales, saliendo hacia el exterior como la información más importante del habla.



(Fig. 3)

Lo expuesto muestra que la información más importante del habla se encuentra en sus frecuencias, sin embargo, la señal de voz se toma en tiempo utilizando convertidores analógico / digitales en un proceso denominado muestreo.

Se utilizan una gran variedad de técnicas para llevar a cabo el reconocimiento de voz. Hay diversos tipos de reconocimiento, y a su vez hay diversos tipos de análisis y comprensión.

Típicamente el reconocimiento comienza con el muestreo digital del lenguaje. El siguiente paso es el proceso de la señal acústica. Las técnicas más frecuentes incluyen el análisis espectral; por ejemplo, análisis LPC (Linear Predictive Coding, Codificación Lineal Predictiva), MFCC (Mel Frequency Cepstral Coefficients) y otras.

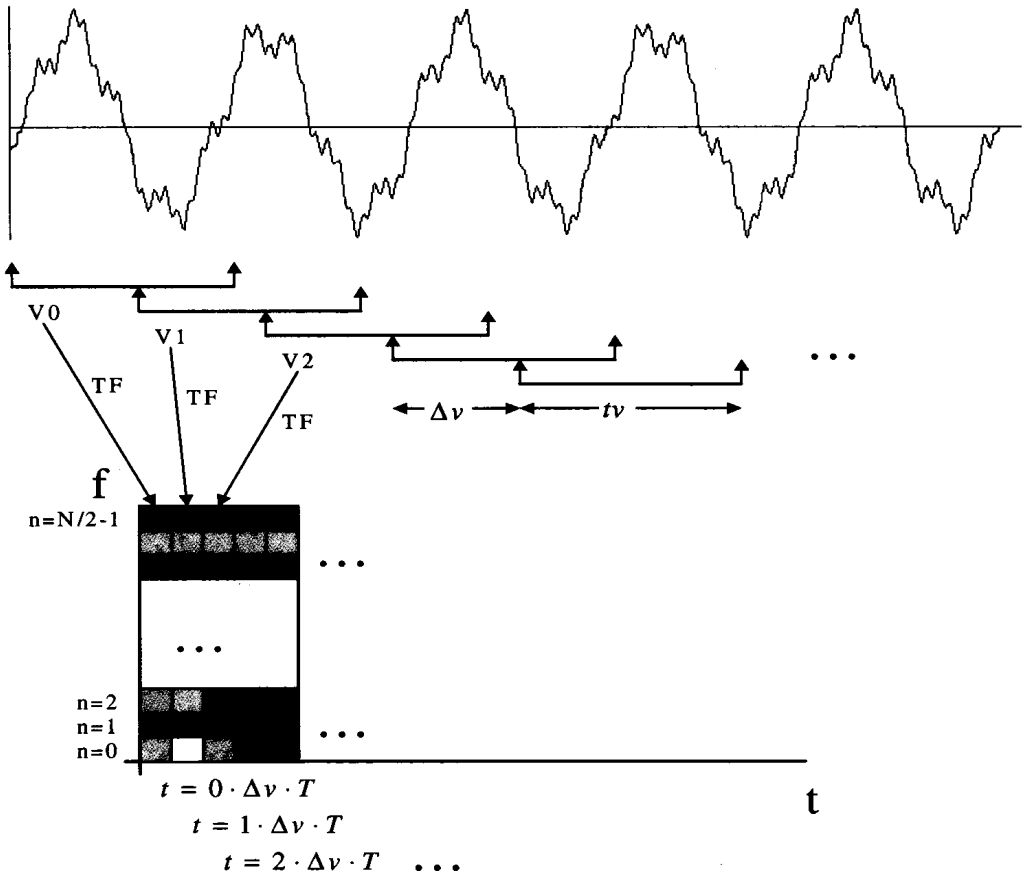
El siguiente paso es el reconocimiento de fonemas, grupos de fonemas y palabras. Este paso puede ejecutarse de varias formas, mediante DTW (Dynamic Time Warping), HMM (Hidden Markov Modelling), NNs (Neural Networks), sistemas expertos y combinación de técnicas.

Actualmente los sistemas basados en HMMs son los más frecuentemente utilizados y la aproximación más exitosa.

La mayoría de los sistemas se ayudan de algún tipo de conocimiento sobre el lenguaje para ayudar al proceso de reconocimiento.

En la **(Fig. 4)** se representa en primer lugar la señal de voz en el dominio del tiempo. Las muestras de la señal se dividen en secciones de igual tamaño (ventanas) y sobre cada conjunto de muestras correspondientes a una ventana se aplica un método de traspaso a frecuencias. Si el tamaño de la ventana es de N muestras, entonces conseguimos N/2 valores de frecuencias. Por lo tanto obtendremos tantos conjuntos de N/2 frecuencias (columnas en el gráfico) como ventanas en el dominio del tiempo. Si tomamos 11.025 muestras / segundo conseguimos un ancho de banda aproximado de 5.500 Hz., lo que es suficiente para la mayor parte de los espectrogramas de voz.

Si optamos por ventanas temporales de N=256 muestras entonces obtenemos 128 frecuencias (N/2), que distribuidas a lo largo del ancho de banda (5.500 Hz.) nos proporciona una resolución espectral de aproximadamente 43 Hz. (5.550/128).



(Fig. 4)

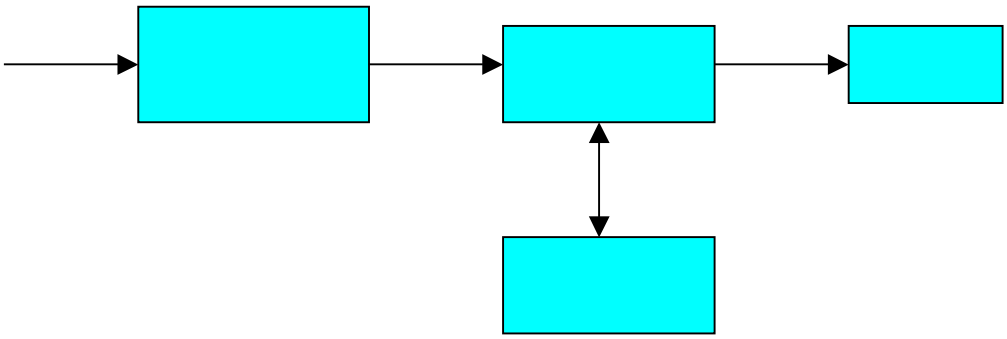
Según se diseñe el tamaño de la ventana (relacionado con la frecuencia de muestreo) se conseguirán espectros de voz en banda ancha o banda estrecha.

Otro concepto importante es la relación inversa existente entre la resolución frecuencial y la resolución temporal. Fijada una frecuencia de muestreo, si aumentamos mucho el tamaño de la ventana temporal (N) conseguiremos una mayor resolución espectral (N/2), pero a costa de emplear una ventana temporal que puede ser demasiado grande como para captar cambios bruscos en la señal de voz.

Técnicas de comparación de patrones.

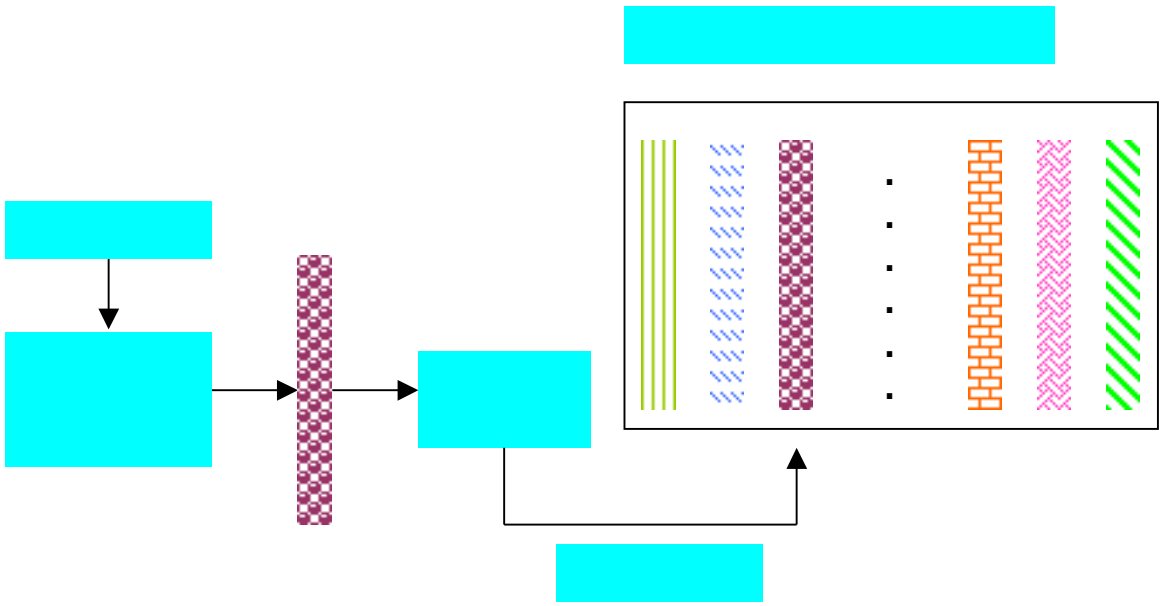
Esta estrategia ha sido muy utilizada en los reconocedores del habla tradicionales. Su principal ventaja inmediata reside en que no es necesario descubrir características espectrales de la voz a nivel fonético, lo que evita desarrollar etapas complejas de detección de formantes, de rasgos distintivos de los sonidos, tono de voz, etc.

El mayor inconveniente de este método estriba en la dificultad para crear una base de datos de patrones del habla que resulte completa, correcta y significativa. Estas bases de datos resultan complejas de obtener, entre otros motivos debido a sus grandes tamaños y a que la información espectral que contienen no tiene un significado fonético directo comprensible por parte de los diseñadores.



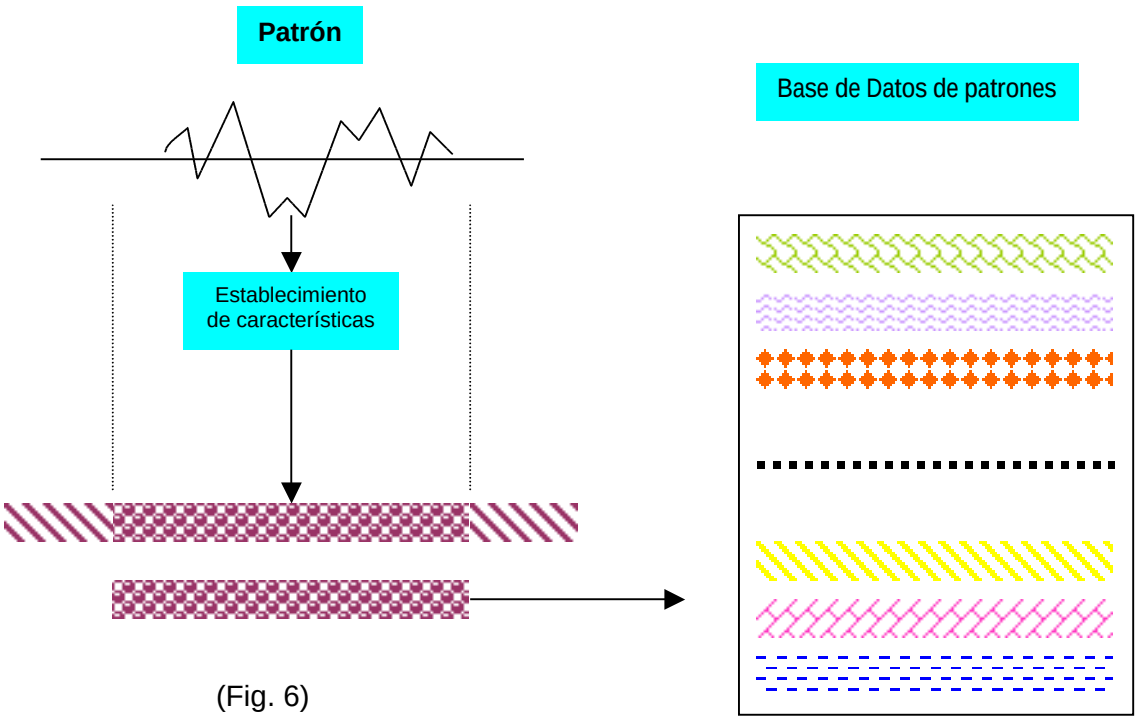
El diagrama siguiente muestra brevemente las etapas involucradas en el reconocimiento del habla empleando técnicas de comparación de patrones:

Los comparadores de patrones basan su funcionamiento en el establecimiento de una distancia matemática entre vectores, de tal manera que se puede calcular lo cercano que se encuentra cada patrón proveniente de las muestras de voz de entrada con todos los patrones existentes en la base de datos. Como se muestra en la **(Fig. 5)**:



La base de datos de patrones se obtiene seleccionando grupos significativos del idioma que se pretenden reconocer. Estos grupos pueden estar constituidos por unidades tales como sonidos básicos (correspondientes a fonemas y alófonos), difonemas, demialófonos, palabras, etc. Una vez seleccionado el grupo básico, se realizan grabaciones de estos sonidos (con un contexto asociado) y se obtienen sus características espectrales (habitualmente parámetros LPC o valores obtenidos aplicando la Transformada rápida de Fourier).

Antes de comparar con la base de datos cada patrón proveniente de la señal de voz que se pretende reconocer, resulta adecuado realizar un proceso de normalización y otro de alineación en el tiempo con el fin de asegurar en la medida de lo posible la coincidencia en el tiempo de los patrones. La alineación en el tiempo trata de aislar cada patrón de entrada de su contexto **(Fig. 6)**.



(Fig. 6)

El proceso de normalización es necesario para ajustar los tamaños temporales de los grupos de estudio. Se tiene como referencia que la duración de una misma palabra puede variar según sean los hablantes, contextos, estados de ánimo, etc. Esta situación nos llevaría a obtener resultados erróneos en la fase de comparación de patrones. Debemos conseguir que los patrones de entrada tengan la misma longitud que los almacenados en la base de datos.

Las técnicas de reconocimiento por comparación de patrones, aunque forman parte de la materia básica en el área del tratamiento de la señal, han sido mayoritariamente sustituidas por los modelos de reconocimiento automático paramétrico, entre los que se encuentran los algoritmos genéticos, las cadenas de Markov y las redes neuronales.

Proceso Informático de la Señal de Voz

Existe una fase previa que consiste en obtener muestras de voz empleando las técnicas de muestreo, cuantización y codificación. El proceso de obtención de muestras de voz, parten de una señal analógica producida por el hablante y el proceso de muestreo realizado por un convertidor analógico/digital:

Mientras se realiza el proceso de muestreo, los valores numéricos, de cada muestra (que indican amplitudes de la señal) se almacenan en la memoria de un computador o de un hardware específico de tratamiento de señal para que puedan ser procesados. En esta fase se suelen emplear dispositivos específicos de entrada / salida para realizar el traspaso del convertidor analógico/digital a la memoria.

La grabación digital captura el sonido almacenando los valores de amplitud de una onda a intervalos regulares de tiempo. La amplitud (altura) de una onda de sonido determina su volumen; la frecuencia (medida en Hertzios o Hz.) determina su escala (lo grave o aguda que suena).

Las ondas de sonido son continuas (analógicas) en la naturaleza, pero una computadora sólo puede trabajar con información digital. Así que la computadora almacena la amplitud de una señal grabada en instantes determinados. Luego recrea el sonido convirtiendo las muestras digitales de sonido de vuelta a una señal analógica mediante un (Convertidor Digital a Analógico). La frecuencia de muestreo indica cuántas muestras del sonido se toman en un segundo. Así una frecuencia de 22 Khz. indica 22.000 muestras por segundo. El ser humano puede oír entre 20 Hz. y 20 Khz.

Las muestras en memoria pueden ser procesadas directamente por el computador para implementar aplicaciones en tiempo real, o bien pueden ser almacenadas en un archivo para su uso posterior. En caso de que sean almacenadas en un archivo resulta conveniente que se utilice algún estándar de almacenamiento de archivos de voz (por ejemplo W A V), con el fin de unificar formatos y universalizar su utilización desde diferentes programas. A continuación se describe el formato de los archivos WAV.

Formato de los ficheros de sonido WAV

Uno de los formatos de fichero más utilizados para almacenar sonidos es el formato WAV. Se trata de almacenar las muestras una tras otra (a continuación de la cabecera del fichero, que entre otras cosas indica la frecuencia de muestreo), sin ningún tipo de compresión de datos, con cuantificación uniforme. La sencillez de este formato lo hace ideal para el tratamiento digital del sonido. El formato de los ficheros WAV es el siguiente:

Bytes	Contenido Usual	Propósito/Descripción
00 - 03	"RIFF"	Bloque de identificación (sin comillas).
04 - 07	???	Entero largo. Tamaño del fichero en bytes, incluyendo cabecera.
08 - 11	"WAVE"	Otro identificador.
12 - 15	"fmt "	Otro identificador
16 - 19	16, 0, 0, 0	Tamaño de la cabecera hasta este punto.
20 - 21	1, 0	Etiqueta de formato. (Algo así como la versión del tipo de formato utilizado).
22 - 23	1, 0	Número de canales (2 si es estéreo).
24 - 27	???	Frecuencia de muestreo (muestras / segundo).
28 - 31	???	Número medio de bytes / segundo.
32 - 33	1, 0	Alineamiento de bloque.
34 - 35	8, 0	Número de Bits por muestra (normalmente 8, 16 ó 32).
36 - 39	"data"	Marcador que indica el comienzo de los datos de las muestras.
40 - 43	???	Número de bytes muestreados.
resto	???	Muestras (cuantificación uniforme)

Los datos numéricos que ocupan más de un byte se representan de la siguiente forma: Primero están los bytes menos significativos, y a continuación los más significativos (conocido como "formato Intel").

Bits

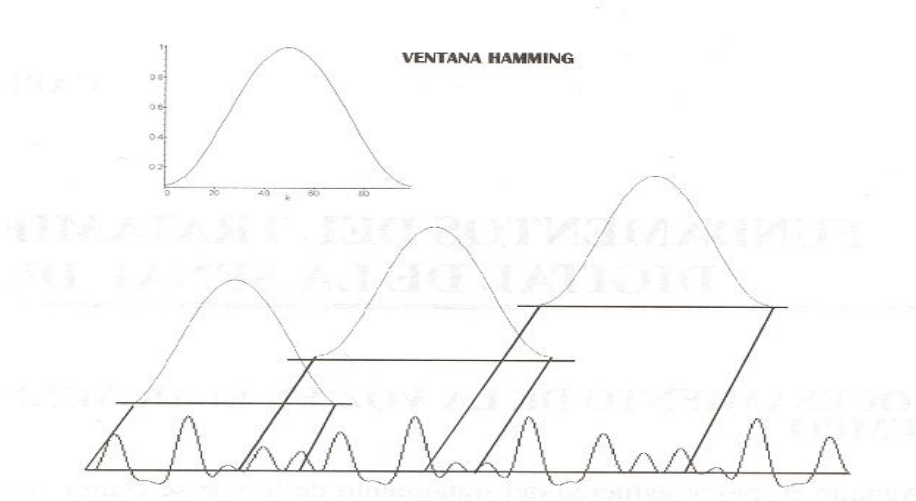
El número de bits también influye en la calidad de la grabación ya que indica el número de pasos medibles del sonido.

Las tarjetas de sonido toman las muestras a 16 bits. Esos bits definen la posición del altavoz. Para emitir sonidos, los altavoces se mueven dando golpes. Estos golpes hacen que el aire que nos rodea vibre, nuestros oídos captan esas vibraciones y las transforman en impulsos nerviosos que van a nuestro cerebro. Por lo tanto debemos indicarle al altavoz dónde debe "golpear". Para ello simplemente se envía una posición (en este caso un número). Entonces cuantas más posiciones se puedan representar, mejor será el sonido. Y cuantos más bits tengamos, más posiciones se podrán representar.

8 bits	256 posiciones
16 bits	65536 posiciones

Procesamiento de la voz en el dominio del tiempo

Existen métodos para el procesamiento de la voz en el dominio del tiempo que pueden ser muy útiles. Debido a la naturaleza cambiante de la voz, resulta más conveniente aplicar el análisis a porciones de voz, para observar la evolución de los distintos parámetros calculados; por ello es conveniente procesar porciones o ventanas de la señal. En este caso en particular se explicara el método de Hamming **(Fig. 7)**.



(Fig. 7)

El mecanismo de ventaneo es de la siguiente manera. A cada porción de la señal de voz (del tamaño deseado) se le asigna una ventana, de tal forma que las muestras queden ponderadas con los valores de la función de Hamming.

En este caso, las muestras que se encuentran en los extremos de la ventana tienen un peso mucho menor que las que se hallan en el medio, lo cual es muy adecuado para evitar que características de los extremos del bloque varíen la interpretación de lo que ocurre en la parte más significativa (central) de las muestras seleccionadas.

Como se puede observar en la Figura la colocación de las ventanas puede realizarse de tal forma que existan solapamientos. Aunque esto repercutirá negativamente en los

tiempos de respuesta de los algoritmos utilizados, proporcionará una mejor calidad en los resultados obtenidos.

La función de Hamming es la siguiente:

Ventana Hamming:

w (t) = 0.54 - 0.46*coseno(27Πt/L) en 0<=t<=L-1
w (t) = 0 en el resto de la señal

Esta ventana es la más utilizada, sin embargo, los valores en sus extremos quedan muy reducidos, por lo que se suele solapar este tipo de ventanas para eliminar el efecto mencionado.

Para cualquier ventana, su duración determina la cantidad de cambios que sé podrán obtener; con una duración temporal larga se omiten los cambios locales producidos en la señal, mientras que con una duración demasiado corta se reflejan demasiado los cambios puntuales y se reduce la resolución espectral.

Relacionando el tamaño de las ventanas y la frecuencia de muestreo de la señal, obtenemos un filtrado de dadas las características del aparato fonador habitualmente se sitúa entre los 45 Hz y los 300Hz.

Una vez establecido el mecanismo de ventanas, las características temporales más comunes que se utilizan en la señal de voz, son las siguientes:

- Energía y magnitud
- Cruces de cero y máximos

Energía y magnitud, estas son útiles para distinguir segmentos sordos y sonoros en la señal de voz, dado que, los valores de ambas características aumentan en los sonidos sonoros respectos a los sordos.

Los cruces por cero indican el número de veces que una señal continua toma el valor de cero. Para las señales discretas, un cruce por cero ocurre cuando dos muestras consecutivas difieren de signo, o bien una muestra toma el valor nulo.

Habitualmente, las señales con mayor frecuencia presentan un mayor valor en esta característica, el ruido también genera un gran número de pasos por cero, por lo que una utilización práctica consiste en analizar las señales grabadas desde esta óptica para comprobar su calidad.

Desde un punto de vista acústico, con los cruces por cero se puede intentar detectar las fricaciones del habla.

El problema que presentan los cruces por cero es la sensibilidad que se da a las componentes continuas de la señal. Se puede encontrar un estimador alternativo contabilizando los máximos o mínimos que existan en la señal de voz.

Procesamiento de la señal de voz

Los datos de voz para el presente trabajo, fueron procesados en 23.2 mseg (256 puntos) frames, los cuales fueron seccionados por 11.6 mseg (128 puntos) entre el procesamiento de frames. Los coeficientes cepstrales fueron encontrados para los formatos de frecuencia lineal y frecuencia mel. Para los coeficientes cepstrales de frecuencia lineal LFCC (Linear Frequency Cepstral Coefficients), se uso la siguiente ecuación:

$$LFCC_i = \sum_{k=0}^{K-1} Y_k \cos\left(\frac{\pi i k}{K}\right) \qquad i = 1,2,\ldots P$$

Para los coeficientes cepstrales de frecuencia mel, 10, los filtros del triángulo de 200Hz fueron espaciados uniformemente entre 0 y 1 khz, seguido por 10 filtros logaritmicamente aumentando el ancho de banda espaciados logarítmico a partir de 1 Khz. a la frecuencia de Nyquist. Las salidas registradas de energía Xk de los filtros de frecuencia mel se muestran en la **(Fig.8)** fueron utilizadas junto con la ecuación siguiente para calcular los coeficientes cepstrales de la frecuencia mel.

$$MFCC_i = \sum_{k=1}^{20} X_k \cos\left(\frac{\pi i (k - 0.5)}{20}\right) \qquad i = 1,2,\ldots P$$

Donde, 20 es el número de filtros de
la frecuencia mel y P es el orden

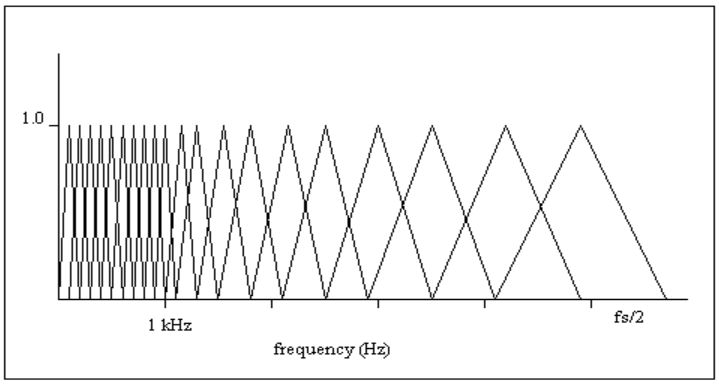


Fig. (8) Mel Frequency Triangle Filtres

La distancia métrica fue formada usando la distancia euclidiana para los coeficientes cepstrales sobre todos los frames después que fue aplicado el DTW para alinear los frames óptimamente.

Todas las trayectorias fueron dadas en un costo de transición de 1. La distancia métrica entre el frame i de la palabra prueba T y el frame j de la palabra referencia R fue calculada como sigue.

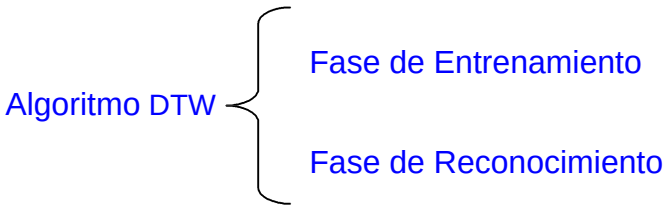
$$D_{i,j} = \left(\frac{1}{P}\right)\sqrt{\sum_{k=1}^{P}\left(CCT_{i,k} - CCR_{j,k}\right)^2}$$

Algoritmo DTW (Dynamic Time Warping)

El problema que se presenta cuando se pronuncia una palabra es que esta no siempre se realiza a la misma velocidad, lo que produce importantes distorsiones temporales. Estas distorsiones afectan no sola a la palabra considerada sino también a sus componentes acústicos. Las variaciones temporales no son generalmente proporcionales a la velocidad de locución y podrán variar de locutor a locutor. Por las características que posee la señal de voz se debe hacer uso de un algoritmo que permita la comparación de dos objetos, independientemente de sus duraciones a través del tiempo. Dicho método debe tolerar la variabilidad de las distorsiones temporales con que en el habla corriente se pronuncian los distintos fonemas y debe imponer tan solo restricciones físicas de continuidad, monotocidad, etc.

Un algoritmo que permite realizar la alineación entre palabras diferentes es el DTW (Dynamic Time Warping). Esta técnica se encarga de realizar la comparación de patrones acústicos, tomando en cuenta la variación en la escala del tiempo de dos palabras a comparar.

El alineamiento temporal dinámico se divide en dos fases;



Fase de Entrenamiento

En la primera de ellas, denominada fase de entrenamiento, el hablante pronuncia cada una de las palabras que conforman el vocabulario (n palabras), de tal forma que entrena al sistema para el reconocimiento de los futuros patrones o formas a reconocer. Luego se realiza la elección del mejor candidato de cada uno de las palabras por medio de las técnicas desarrolladas para el diccionario de datos. Finalmente, queda representada cada palabra por su correspondiente prototipo, a estas últimas cadenas de vectores se les denomina plantillas y son la representación del espectro de una palabra y su cambio a través del tiempo.

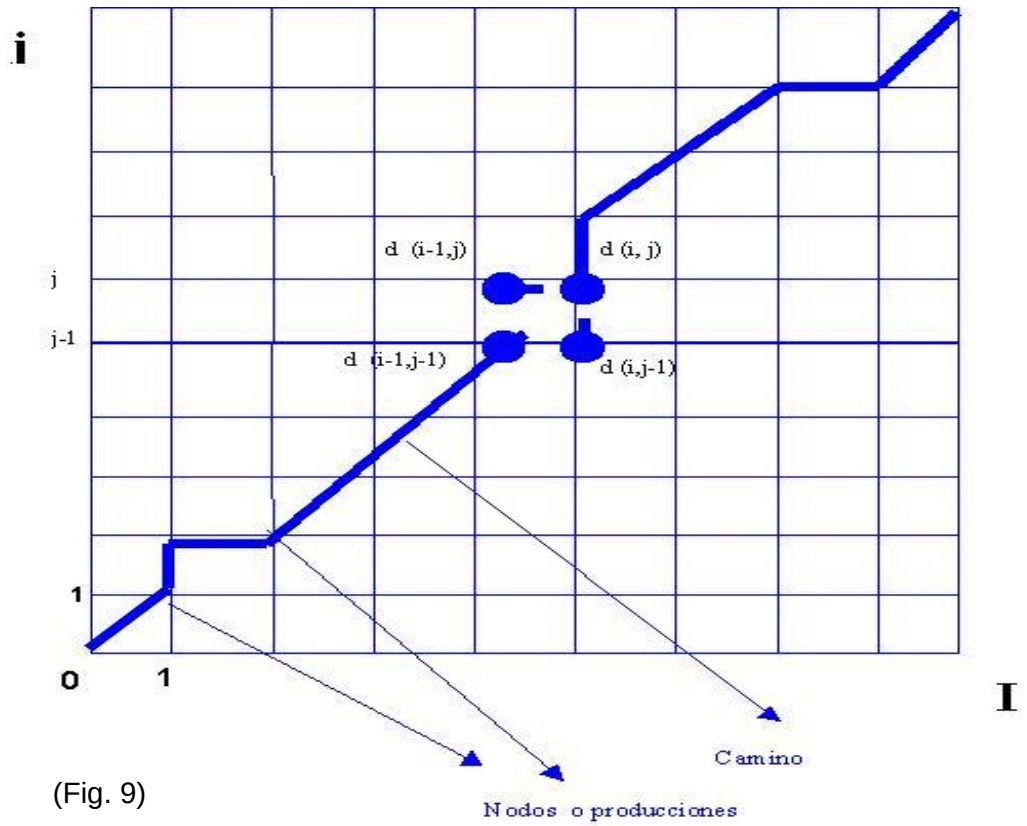
Fase de Reconocimiento

En la segunda fase (fase de reconocimiento), a las palabras pronunciadas por el locutor se les aplica todo el procedimiento de la fase de entrenamiento hasta la obtención de vectores delta cepstrales (ver ANEXO 2), los cuales se comparan con los prototipos previamente almacenados por medio de una medida de distancia. En este sistema de reconocimiento de voz, la medida para calcular la distorsión es la distancia euclidea.

El paso posterior al calculo de los coeficientes delta cepstrales es la realización del DTW, su punto de partida es el apareamiento de cada uno de los prototipos del diccionario de datos con la señal que se desea evaluar, de tal manera que se comparan los vectores delta cepstrales extraídos de la señal de voz de cada palabra que entra al sistema (patrón prueba) con los prototipos del diccionario (patrón referencia).

Las cadenas de la palabra (incógnita) y la palabra conocida (prototipo) se colocan sobre los ejes I y J respectivamente; de manera que los primeros componentes de cada vector quedan en el primer punto de cada eje **(Fig. 9)**

Vectores prototipo i(1),.....i(I)
Vectores incógnita j(1),.....j(I)



(Fig. 9)

Cada nodo en el plano es un número real positivo. El problema consiste en encontrar el camino que recorra el plano con la distancia mas corta, dicho camino debe empezar en el punto origen (0,0) y debe terminar en el nodo final (I, J). Las distancias o costos son asignados a los caminos tomando como base los puntos anteriores. Por ejemplo en la figura para llegar al punto d (i, j) se puede tomar d (i-1, j), d (i-1, j-1) ó d (i, j-1).

Si definimos los valores d (i, j) correspondientes a las distancias entre los vectores i-ésimo de la palabra desconocida, compuesta por I vectores y j-ésimo de la palabra patrón compuesta por J vectores, estos forman una matriz como la **(Fig. 9)**. El alineamiento de las dos secuencias de vectores se corresponde con un camino en la matriz de distancias (indicado con la línea gruesa) que parte del elemento d (1,1) y finaliza d (I, J). La distancia total entre los vectores alineados corresponde a la suma de los elementos de la matriz de distancias obtenidos en el camino, de forma que la búsqueda del alineamiento óptimo es equivalente al camino con menor distancia total.

Si definimos g (i, j) como la distancia acumulada del camino óptimo que parte del elemento d (1,1) y termina en d (i, j), se puede escribir la siguiente formula recursiva para la obtención de la distancia correspondiente a dicho camino.

$$g(1,1) = d(1,1)$$
$$g(i-1, j) + d(i, j)$$

$$g(i, j) = \min. \; g(i-1, j-1) + 2d(i,j)$$
$$g(i, j-1) + d(i, j)$$

$$D = \frac{g(i, j)}{(I+J)} \quad [@]$$

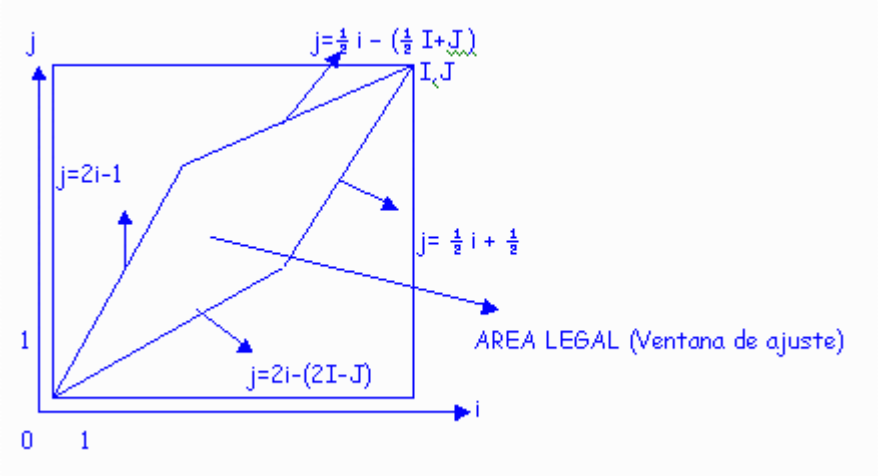
El valor de D corresponde a la distancia media para la alineación óptima de las dos palabras (incógnita y patrón), y se utiliza como criterio de decisión, seleccionando el patrón para el que dicha distancia es mínima.

Cada uno de los nodos o producciones debe ser un número real positivo. El problema como ya se planteó, consiste en encontrar el camino con la distancia más corta. Para encontrar tal distancia existen dos tipos de restricciones; locales y globales

Restricciones Globales y Locales

Para la señal de un locutor o hablante, hay un número de restricciones sobre la búsqueda de la ruta, las cuales pueden ser aplicadas para ayudar a minimizar la complejidad de la búsqueda. La primera restricción es que la búsqueda deberá ser monotonica. Esto puede ser forzado por la aplicación de las restricciones global y local siguientes.

Las restricciones globales son simples restricciones generales donde se valida la ruta buscada general. Es decir definen las restricciones del camino a nivel del plano en general **(Fig. 10)**.



(Fig. 10): plano que describe la restricción global para el DTW.

De la expresión [@] se deduce que para calcular g (I, J) es necesario calcular previamente los g (i, j), correspondientes a todos los puntos (i, j) del plano I, J, asegurándose de que el camino óptimo pasa o no por cada uno de ellos. Para limitar el número de puntos a considerar, se suele descartar aquellos cuya pertenencia al camino óptimo es físicamente improbable. Ello se consigue aplicando una ventana de ajuste que restrinja la máxima diferencia temporal entre los objetos comparados.

Medida de Disimilitud.- Para la obtención del camino óptimo se necesita disponer de una distancia que defina la disimilitud entre dos palabras que a la vez provenga de la función de alineamiento. La forma genérica de dicha función de distancia es:

$$D(i(k), j(k)) = \frac{\sum_{k=1}^{Kd} (i(k), j(k))\, \tilde{W}(k)}{N(\tilde{W})}$$

Donde D(i(k),j(k)) viene a ser una función, cuyo cálculo proporciona la distancia total a lo largo del camino óptimo, d(i(k),j(k)) es la distancia local entre las ventanas i(k) de la palabra referencia y j(k) de la palabra a reconocer, W(k) es una función de ponderación para k, y N(W) es un factor de normalización que depende de W.

El camino corresponderá a aquel que minimice la función de distancia total representada por:

$$\hat{D} = \underset{(K,i(k),j(k))}{\text{Min}} \{ \sum_{n=1}^{N} D[i(k),j(k)] \}$$

La distancia que da los mejores resultados viene a ser aquella que se expresa como:

$$d_{cep} = \sum_{n=1}^{p} (W(n)(c_t(n) - c_r(n)))^2$$

Se conoce como la distancia cepstral y está en función de los coeficientes cepstrales ct y cr de la palabra a reconocer y de la palabra referencia deducidos a partir de:

$$c(1) = -a$$
$$c(n) = -a_n - \sum_{k=1}^{n-1} (1 - \frac{k}{n}) a_k c(n-k), \quad 1 < n \leq p$$

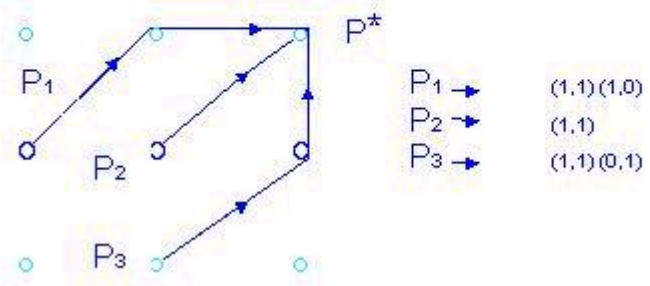
Como se podrá observar viene a ser una distancia euclídea y su uso está muy difundido en procesamiento de voz debido a su simplicidad y porque resulta ser una aproximación de la distancia entre dos espectros logarítmicos representados por sus coeficientes cepstrales. Es independiente de como se efectúa el algoritmo DTW.

En la **(Fig. 10)** se muestra una ventana de ajuste que indica el área por donde debe pasar el camino óptimo que contiene la disimilitud entre los dos objetos I, J (en este caso son los patrones de prueba e incógnita).

Las restricciones locales determinan la ruta válida buscada sobre una base local. Para determinar el valor del costo mínimo a un punto del plano es necesario calcular El valor del punto anterior entre puntos tales, **i**k-1 **i**k y jk-1 **j**k.

Esta transición de un nodo a otro hace fundamental el uso de las restricciones locales. La manera como se pueden aplicar las restricciones locales en el plano I, J **(Fig. 11)**. Donde P1, P2 y P3 son coordenadas que hacen parte del camino total P, que a su vez representa la disimilitud entre los objetos.
Definen las restricciones del camino a nivel del plano a nivel local.

P → (P1,q1)(P2,q2)(P3,q3).....(Pn,qn).



(Fig. 11)Plano que describe la restricción local para el DTW

En la figura se puede apreciar que para llegar al nodo P*, se puede hacer mediante los puntos p1, p2 y p3 (producciones). El punto p1 indica que primero se debe desplazar una unidad en X y otra en Y (1,1) (En este ejemplo se describió formalmente X y Y, para efectos de ilustrar que es un desplazamiento horizontal y otro vertical), y luego 1 en X sin ser necesario otro desplazamiento en Y (1,0). Para los puntos p2 y p3 se realiza de la misma manera.

Para la implementación del reconocedor de voz las restricciones locales aplicadas son las indicadas en la tabla 1, en donde se determina el tipo de restricción a evaluar cada vez que se realiza una transición entre nodos. La columna tipo es equivalente a la propiedad, la columna restricción es la representación gráfica de como se puede realizar una transición entre nodos y la columna producción es la encargada de detallar como se realiza la secuencia de un nodo a otro en términos de desplazamiento entre coordenadas.

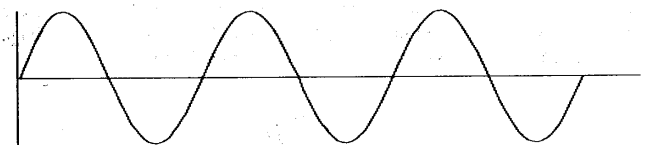
Las restricciones locales que se aplican en el plano de apareamiento entre plantillas son fundamentales ya que determinan la manera como debe evolucionar el análisis sucesivo de los diferentes nodos. Por su parte, las restricciones globales garantizan la no-realización de comparaciones improbables e innecesarias entre las palabras a reconocer.

Después de haber definido los tipos de restricciones que se deben aplicar en el alineamiento entre plantillas, se debe analizar la manera como se realizará el algoritmo que calculará la distorsión final entre patrones.

ANEXO 1

La señal sonora

La definición de sonido es: “Sensación producida en el órgano del oído por el movimiento vibratorio de los cuerpos, transmitido por un medio elástico, como el aire”. El origen del sonido es la vibración de los cuerpos, se dice que la vibración es periódica cuando se repite a intervalos determinados. Al golpear un diapasón se produciría una vibración periódica y con forma sinusoidal. En la **(Fig. 1.1)** se muestra su representación.



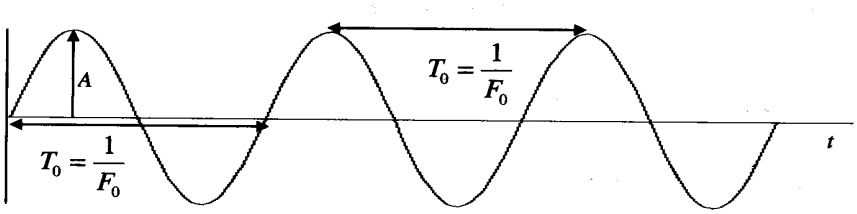
(Fig. 1.1)

Existen otros muchos fenómenos en la naturaleza cuyo movimiento tiene la misma forma sinusoidal (por ejemplo el movimiento ondulatorio del agua, el movimiento de un péndulo, etc.).

A continuación se define el modelo matemático que describe perfectamente el movimiento vibratorio que produce el diapasón.

Movimiento Armónico Simple

Cualquier sonido genérico se puede formar a partir de una suma de movimientos armónicos simples, en la siguiente **(Fig. 1.2)** figura se muestra los parámetros que describen la onda generada por el diapasón.

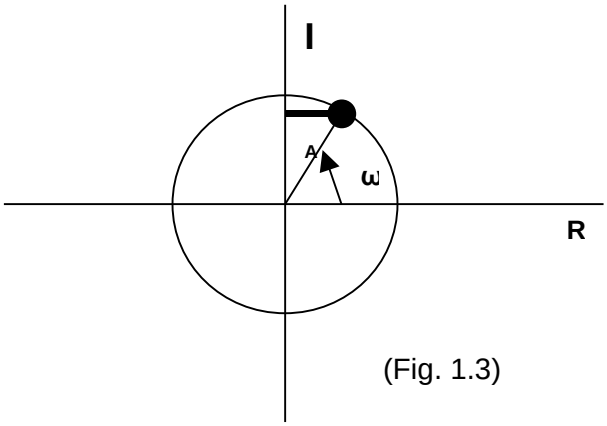


(Fig. 1.2)

El parámetro A es la amplitud de la onda, es el valor máximo que puede alcanzar. El parámetro To es el periodo, se expresa en segundos; To es igual a la inversa de la frecuencia: Fo . La frecuencia se expresa en ciclos por segundo o hercios (Hz).

La forma de onda se puede describir perfectamente con el siguiente modelo: partimos de un punto que se mueve de forma uniforme sobre una circunferencia a una velocidad angular de **ω**, en radianes por segundo.

La proyección de dicho punto sobre el eje R, representa el valor de la onda a lo largo del tiempo **(Fig. 1.3)**.

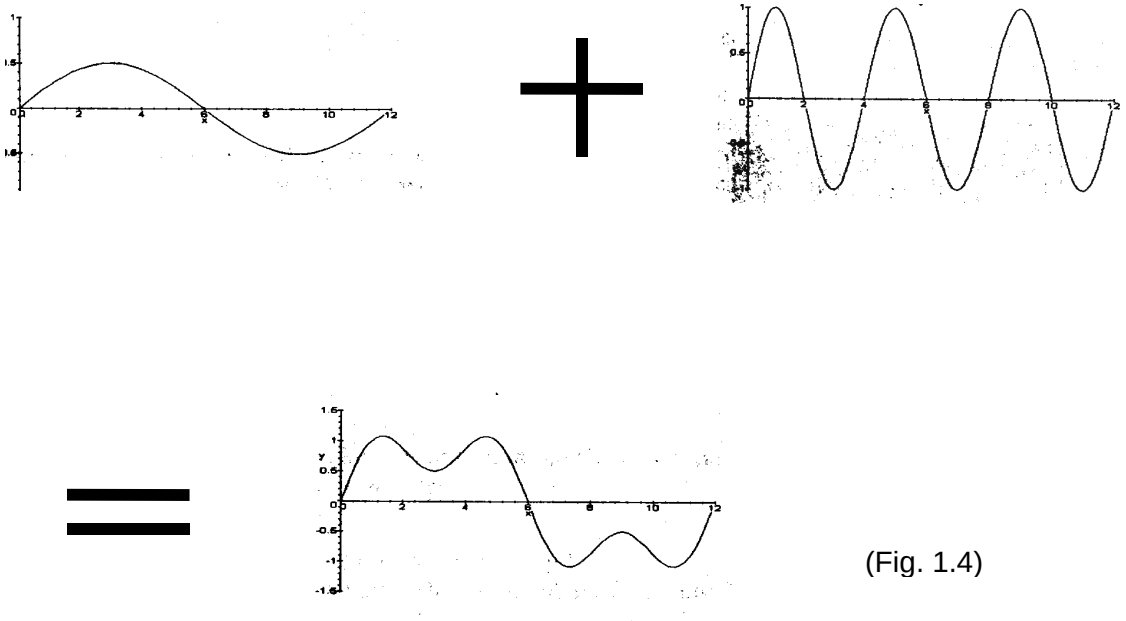


(Fig. 1.3)

La relación entre la velocidad angular y la frecuencia es: ω=2Πf. Esta proyección sobre el eje I se calcula mediante la función coseno: m (t)= A cos (ωt) = A cos (2Πft). Se podría ampliar la expresión teniendo en cuenta la fase inicial:
m (t)= A cos (2Πft + φo).

Sonido complejo

Cualquier sonido complejo se puede formar como una suma de armónicos simples de distinta frecuencia **(Fig. 1.4)**.



(Fig. 1.4)

Para facilitar el análisis de un sonido genérico debemos descomponerlo en el conjunto de armónicos simples que lo forman, ya que sus intensidades y frecuencias nos informan de las características del sonido original.

Una buena parte de los sonidos producidos en el habla son cuasiperiodicos, especialmente las vocales. Por ello, para reconocerlas se descomponen en sus ondas simples que lo componen, y según sean las frecuencias de las mismas se puede distinguir entre distintas vocales.

El sonido aperiódico o ruido es una onda que no presenta ninguna periodicidad. Al descomponerla en sus armónicos básicos resulta que la energía esta muy dispersa en el conjunto de armónicos posibles y es poco estable a lo largo del tiempo.

Como se deben de obtener las muestras

Sería razonable pensar que a mayor frecuencia de muestreo obtendríamos mejor calidad en la onda capturada, concepto que resulta falso. Si se aumenta la frecuencia de muestreo se consigue capturar frecuencias más altas de la onda original, pero las frecuencias bajas se siguen capturando con la misma calidad; este aumento de frecuencia conlleva un coste: convertidores analógicos/digitales más caros, mayor necesidad de memoria, mayor tiempo de cómputo, por lo tanto no se deben capturar muestras a más frecuencia de lo estrictamente necesario.

Criterio de Nyquist: el muestreo de una onda se debe realizar al menos al doble de su frecuencia máxima.

Si hacemos un muestreo por debajo, perdemos información de las frecuencias altas. Si hacemos un muestreo por encima del doble, aumentamos el coste informático sin ampliar el conjunto de frecuencias capturadas del sonido original. Por lo que debemos utilizar la frecuencia de muestreo oportuna a cada caso [Bernal y Bobadilla].

La calidad de la onda capturada depende de varios elementos: calidad del micrófono utilizado, calidad del convertidor analógico/digital y número de bits utilizados para el almacenamiento de cada muestra, como ya se había mencionado anteriormente.

El ancho de banda o rango de frecuencias que puede percibir el oído humano varía habitualmente desde 20 Hz hasta 20.000 Hz. Para una captura perfecta se debe utilizar una frecuencia de muestreo superior a 40.000 Hz. El estándar de un archivo tipo W A V determina que las frecuencias posibles son 11.025 Hz, 22.050 Hz y 44.100 Hz, con lo que sí hacemos un muestreo a 44.100 Hz recogerá la totalidad de los sonidos audibles.

La frecuencia máxima que se genera en el habla, en general, está por debajo de los 8.000 Hz. Por lo tanto, para capturar un mensaje hablado será suficiente utilizar la frecuencia de muestreo estándar de 22.050 Hz.

Producción de la voz humana (fonética articulatoria)

El análisis de la lengua se realiza a tres niveles:

- Nivel fonológico. Se estudian las unidades lingüísticas mínimas: fonemas. El conjunto de los fonemas se establecen por oposición, es decir, si se cambia un sonido de una palabra y la palabra cambia de significado, al sonido se l considera fonema. En las palabras coco, loco y toco hemos cambiado un fonema y su significado es distinto.

- Nivel morfosintáctico. Se estudian las palabras estableciendo su género, número y tiempo y las relaciones entre ellas.

- Nivel semántico. Se estudia el significado de las frases y su coherencia.

Dentro de la fonética podemos distinguir la articulatoria y la acústica. La primera estudia el movimiento de los órganos fonadores para la formación y emisión del sonido. La fonética acústica se ocupa de las características de la onda sonora y su percepción.

ANEXO 2

Técnicas de análisis

Las técnicas que se pueden usar son muy variadas. Todas tienden a destacar características independientes del hablante y a reducir las dependientes, pero manteniendo toda la información necesaria para el reconocimiento. Entre las técnicas más usuales tenemos LPC (Linear Prediction Coefficients), coeficientes cepstrales y bancos de filtros.

Cualquiera de estas técnicas da como resultado representaciones de la información sonora más compactas y más eficientes en los subsiguientes procesos de ajuste de patrones. La información que den debe ser la misma que poseía la señal sonora original.

Bancos de filtros

El uso de bancos de filtros digitales, implementados inicialmente como filtros analógicos, ha sido históricamente la primera aproximación al procesamiento del habla. Un banco de filtros paso banda puede entenderse como un modelo sencillo de las etapas iniciales del sistema auditivo humano.

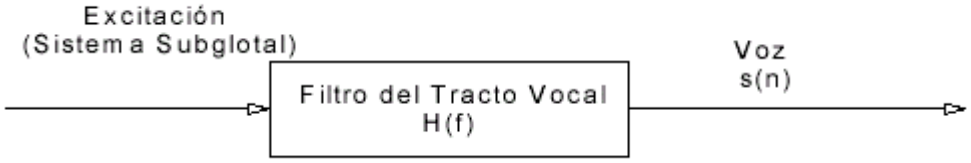
La señal inicial se descompone en un conjunto discreto de muestras espectrales, que contienen una información similar a la que se presenta en los niveles superiores del sistema auditivo de los seres humanos.

Su característica es que no tiene una respuesta lineal en frecuencia existen diferentes escalas. Pero la de nuestro interés es la escala Mel

Coeficientes Cepstrales

Desde la introducción en los primeros años de la década de los 70, las técnicas homomórficas de procesado de señal han tenido gran importancia dentro del campo del reconocimiento de voz.

Los sistemas homomórficos son una clase de sistemas no lineales que obedecen a un principio de superposición. Pueden servir para separar la acción del tracto vocal (Filtro lineal variable en el tiempo) de la señal de excitación. El procesado homomórfico del habla queda resumido en la **(Fig. 2.1)**.



(Fig. 2.1).Técnicas homomorficas

La señal de voz s(n) se descompone en una parte de excitación e(n) y en un filtro Lineal H (eiθ). Así en el dominio frecuencial tenemos:

$$S(e^{i\theta}) = H(e^{i\theta})E(e^{i\theta})$$

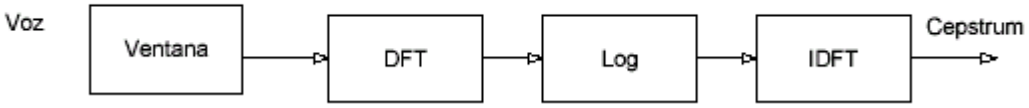
Para la mayoría de las aplicaciones de voz sólo necesitamos la amplitud espectral.
En el dominio logarítmico, las dos componentes anteriores pueden separarse empleando técnicas convencionales de procesado de la señal.

$$\log(|S(e^{i\theta})|) = \log(|H(e^{i\theta})|) + \log(|E(e^{i\theta})|)$$

c(n) son los coeficientes cepstrales derivados de la transformada de Fourier.

$$c(n) = \frac{1}{N_s}\sum_{k=0}^{N_s-1}\log|S_{med}(k)|\, e^{j\frac{2\pi}{N_s}kn} \qquad 0 \le n \le N_s - 1$$

N es el número de puntos con los que se hizo la DFT. Lo normal es usar sólo los primeros términos. Es posible a la hora de calcular un coeficiente cepstral emplear bandas definidas según escalas de Mel **(Fig. 2.2).**



(Fig. 2.2).Análisis Cepstral partiendo de la transformada discreta de Fourier

Coeficientes LFCC

Un objeto del tipo LFCC representa coeficientes cepstral de la escala linear de la frecuencia en función de tiempo. Los coeficientes se representan en marcos con intervalo de muestreo constante.

Coeficientes MFCC o de Mel

A este tipo de parámetros se les conoce como coeficientes cepstrales con frecuencia en escalas de Mel o MFCC (Mel Frequency Cepstral Coefficients). La idea que da origen a estos coeficientes es la obtención de vectores de coeficiente cepstrum en los cuales el espaciamiento en frecuencia no es lineal, sino que se distribuye en una escala perceptual tipo Mel:

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

Para realizar el cálculo de los parámetros Mel-cepstrum, se construye un banco de filtros triangulares equiespaciados en la escala de Mel que es aplicado a la señal de entrada en frecuencia tal y como se puede observar en la **(Fig. 8)**.

Normalmente los filtros se extienden sobre todo el rango de frecuencias. Sin embargo, a veces resultada deseable eliminar frecuencias no deseables o evitar colocar filtros en regiones en las que no hay energía de señal útil. Para ello se utilizan unas determinadas frecuencias inferior y superior, y el número especificado se distribuye uniformemente a lo largo de la banda de paso resultante. Una vez realizado el filtrado, se toma el logaritmo natural de las salidas del banco de filtros y se calcula los parámetros Mel-cepstrum como:

$$c_i = \sqrt{\frac{2}{N}} \sum_{j=1}^{N} m_j \cos\left(\frac{\pi \bullet i}{N}(j-0.5)\right)$$

donde N es el número de filtros del banco.

En la **(Fig. 2.3)**, muestra el esquema de parametrización par la obtención de MFCC.

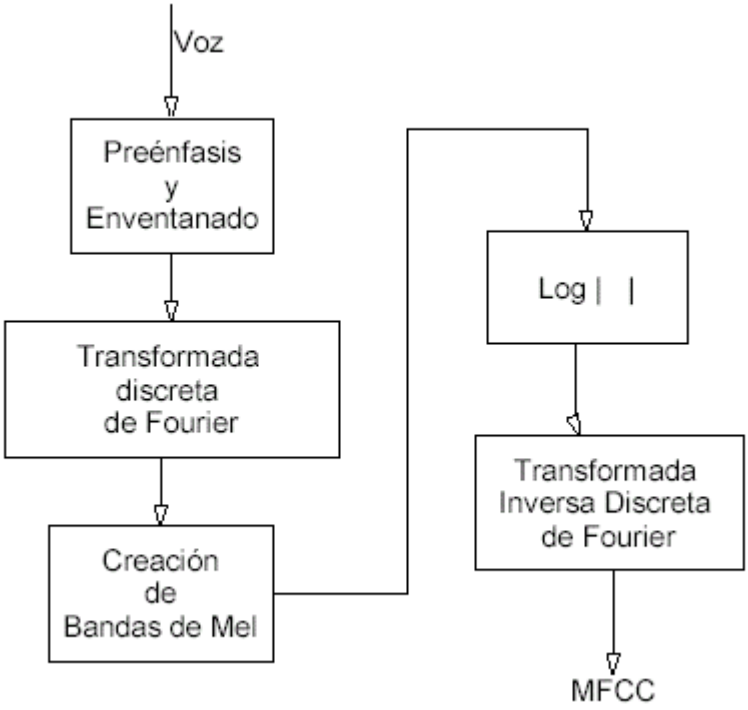


Fig. (2.3)

APÉNDICE A

Código Fuente (Para Matlab)[1]

```
% nr2.m
% Se usa el hablante a como referencia y el hablante a,b,c y d  como
test.
idref='a';
P=8;
refgen52(idref,P);
tic
idtest='a';
ACC=numrec52(idref,idtest,P);
toc
tic
idtest='b';
ACC=numrec52(idref,idtest,P);
toc
tic
idtest='c';
ACC=numrec52(idref,idtest,P);
toc
tic
idtest='d';
ACC=numrec52(idref,idtest,P);
toc
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
idref='b';
refgen52(idref,P);

tic
idtest='a';
ACC=numrec52(idref,idtest,P);
toc

tic
idtest='b';
ACC=numrec52(idref,idtest,P);
toc

tic
idtest='c';
ACC=numrec52(idref,idtest,P);
toc

tic
idtest='d';
ACC=numrec52(idref,idtest,P);
toc

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
idref='c';
refgen62(idref,P);
```

```
tic
idtest='a';
ACC=numrec52(idref,idtest,P);
toc

tic
idtest='b';
ACC=numrec52(idref,idtest,P);
toc

tic
idtest='c';
ACC=numrec52(idref,idtest,P);
toc

tic
idtest='d';
ACC=numrec52(idref,idtest,P);
toc

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
idref='d';
refgen52(idref,P);

tic
idtest='a';
ACC=numrec52(idref,idtest,P);
toc

tic
idtest='b';
ACC=numrec52(idref,idtest,P);
toc

tic
idtest='c';
ACC=numrec52(idref,idtest,P);
toc

tic
idtest='d';
ACC=numrec52(idref,idtest,P);
toc

%tic y toc mide el tiempo que transcurre en correrse el programa
idref='a';
refgen62(idref,P);

tic
idtest,P='a';
ACC=numrec62(idref,idtest,P);
toc

tic
idtest,P='b';
ACC=numrec62(idref,idtest,P);
toc
```

```
tic
idtest,P='c';
ACC=numrec62(idref,idtest,P);
toc

tic
idtest,P='d';
ACC=numrec62(idref,idtest,P);
toc

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
idref='b';
refgen62(idref,P);

tic
idtest,P='a';
ACC=numrec62(idref,idtest,P);
toc

tic
idtest,P='b';
ACC=numrec62(idref,idtest,P);
toc

tic
idtest,P='c';
ACC=numrec62(idref,idtest,P);
toc

tic
idtest,P='d';
ACC=numrec62(idref,idtest,P);
toc
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
idref='c';
refgen62(idref,P);

tic
idtest,P='a';
ACC=numrec62(idref,idtest,P);
toc

tic
idtest,P='b';
ACC=numrec62(idref,idtest,P);
toc

tic
idtest,P='c';
ACC=numrec62(idref,idtest,P);
toc

tic
idtest,P='d';
ACC=numrec62(idref,idtest,P);
toc
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
idref='d';
refgen62(idref,P);

tic
idtest='a';
ACC=numrec62(idref,idtest,P);
toc

tic
idtest='b';
ACC=numrec62(idref,idtest,P);
toc

tic
idtest='c';
ACC=numrec62(idref,idtest,P);
toc

tic
idtest='d';
ACC=numrec62(idref,idtest,P);
toc


%tic y toc mide el tiempo que transcurre en correrse el programa
```

```
function refgen52(idref,P)
% refgen5.m
% Este archivo genera el vocabulario de  referencia de coeficientes
% cepstral.  Los coeficientes para cada palabra se guarda en una matriz
% de P x  m.  P es la orden y  de m es el número de ventanas a procesar.
% El orden de P es determinado, el tamaño de la ventana es N, y el
% tamaño de paso de la ventana es M.
% P=8;  es del número de los coeficientes del LPC

M=128; % window step size
N=256; % window size
% Read digit wav files
id=idref; % identity of speaker used (jl,kl,pk,or vk)
[uno,dos,tres,fs]=readin2(id);
ccref_1=lfcc(uno,M,N,P);
ccref_2=lfcc(dos,M,N,P);
ccref_3=lfcc(tres,M,N,P);
save('vocab.mat','ccref_1','ccref_2','ccref_3');
save('ccref_1');
save('ccref_2');
save('ccref_3');


% readin2.m
% lectura de archivos wav de digitos
function [uno,dos,tres,fs]=readin2(id)
f = fullfile('C:','proyecto','muestras',id,'uno.wav');
[uno,fs,b]=wavread(f);
f = fullfile('C:','proyecto','muestras',id,'dos.wav');
[dos,fs,b]=wavread(f);
f = fullfile('C:','proyecto','muestras',id,'tres.wav');
[tres,fs,b]=wavread(f);

%[fileVector,fs,b]=wavread(file);
% wavread lee el archivo f
% nine vector que genera la muestra
% fs radio de la muestra
% b numero de bits de la muestra

%Hablante normalizado a valor de 1.0

uno=uno/max(abs(uno));
dos=dos/max(abs(dos));
tres=tres/max(abs(tres));
```

```
function Accur=numrec52(idref,idtest,P)
% numrec52.m
% Este archivo genera el vocabulario de referencia de coeficientes
% cepstral. Los coeficientes para cada de la palabras se guarda en matriz
% de P x  m. P es la orden y m es el número de ventanas a procesar.
% El orden de P es determinado, el tamaño de la ventana es N, y el tamaño
% de paso de la ventana es M.

load vocab;
%P=8; % number of LPC coeffs
M=128; % window step size
N=256; % window size
% Lee dígitos de test del archivo  wav
id=idtest; % identity of speaker used (jl,kl,pk,or vk)
[uno_t,dos_t,tres_t,fs]=readin2(id);
cct_1=lfcc(uno_t,M,N,P);
cct_2=lfcc(dos_t,M,N,P);
cct_3=lfcc(tres_t,M,N,P);
%Ocupa como referencia el archivo ccref_1
ref=ccref_1;
test=cct_1;
Match(1,1)=dtw1(test,ref);
test=cct_2;
Match(2,1)=dtw1(test,ref);
test=cct_3;
Match(3,1)=dtw1(test,ref);

%Ocupa como referencia el archivo ccref_2
ref=ccref_2;
test=cct_1;
Match(1,2)=dtw1(test,ref);
test=cct_2;
Match(2,2)=dtw1(test,ref);
test=cct_3;
Match(3,2)=dtw1(test,ref);

%Ocupa como referencia el archivo ccref_3
ref=ccref_3;
test=cct_1;
Match(1,3)=dtw1(test,ref);
test=cct_2;
Match(2,3)=dtw1(test,ref);
test=cct_3;
Match(3,3)=dtw1(test,ref);
idref
idtest
accum=0;
%for i=1:11
for i=1:3
    [val,ind]=min(Match(:,i));
    if ind==i
        accum=accum+1;
    end;
end;
Accur=100*accum/3
```

```
% dtw1.m
% Dynamic Time Warping Algorithm
function Dmin_Avg=dtw1(test,ref)
eps=3;
I=size(test,2);     % Regresa el tamaño de test con dimension 2
J=size(ref,2);      % Regresa el tamaño de test con dimension 2
Dmin=1e6*ones(I,J); % ones(i,j): Matriz de unos de ixj
C_t=zeros(I,J);     % Matriz de ceros de ixj
delmax=2.0;
delmin=0.5;
for i=1:1+eps
    Dmin(i,1)=L2norm(i,1,test,ref); %RMS Error
    C_t(i,1)=0;
end;
for j=1:1+eps
    Dmin(1,j)=L2norm(1,j,test,ref); %RMS Error
    C_t(1,j)=0;
end;
Dmin_old=Dmin;
C_told=C_t;
for i=2:I
    jmax1=delmax*(i-1)+eps+1; % Global Constraints
    jmax2=delmin*(i-1)+J-delmin*(I-eps-1);
    jmin1=1+delmin*(i-1-eps);
    jmin2=1+J-1-eps+delmax*(i-I);
    jmax=min(J,floor(min(jmax1,jmax2)));
    jmin=max(1,ceil(max(jmin1,jmin2)));
for j=jmin:jmax
    C_node=L2norm(i,j,test,ref); %RMS Error
    Prev_Min=1e6*ones(3,1);
for k=1:3
idiff=1;
jdiff=1;
if k==1
    jdiff=0;
end;
if k==3
    idiff=0;
end;
if i-idiff>=1 & j-jdiff>=1
    Prev_Min(k)=Dmin(i-idiff,j-jdiff);
end;
end;
[Dmin(i,j),k_ind]=min([Prev_Min;Dmin_old(i,j)]);
Dmin(i,j)=Dmin(i,j)+C_node;
idiff=1;
jdiff=1;
if k_ind==1 jdiff=0; end;
if k_ind==3 idiff=0; end;
if i-idiff>0 & j-jdiff>0
if i~=1 & j~=1 & i-idiff~=I & j-jdiff~=J
C_t(i,j)=1+C_t(i-idiff,j-jdiff);
else
C_t(i,j)=C_t(i-idiff,j-jdiff);
end;
end;
if k_ind==4 C_t(i,j)=C_told(i,j); end;
```

```
T_Cost=0;
if C_t(i,j)>0 T_Cost=1; end;
Dmin(i,j)=Dmin(i,j)+T_Cost;
Dmin_old=Dmin;
C_told=C_t;
end;
end;
Dmin_F=1e4;
for i=I-eps:I
if Dmin(i,J)<Dmin_F
Dmin_F=Dmin(i,J);
C_t_F=C_t(i,J);
end;
end;
for j=J-eps:J-1
if Dmin(I,j)<Dmin_F
Dmin_F=Dmin(I,j);
C_t_F=C_t(I,j);
end;
end;
Dmin_Avg=Dmin_F/C_t_F;
```

```
% nr2.m
idref='a';
refgen62(idref,P);

tic
idtest,P='a';
ACC=numrec62(idref,idtest,P);
toc

tic
idtest,P='b';
ACC=numrec62(idref,idtest,P);
toc

tic
idtest,P='c';
ACC=numrec62(idref,idtest,P);
toc

tic
idtest,P='d';
ACC=numrec62(idref,idtest,P);
toc

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
idref='b';
refgen62(idref,P);

tic
idtest,P='a';
ACC=numrec62(idref,idtest,P);
toc

tic
idtest,P='b';
ACC=numrec62(idref,idtest,P);
toc

tic
idtest,P='c';
ACC=numrec62(idref,idtest,P);
toc

tic
idtest,P='d';
ACC=numrec62(idref,idtest,P);
toc

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
idref='c';
refgen62(idref,P);

tic
idtest,P='a';
ACC=numrec62(idref,idtest,P);
```

```
toc

tic
idtest,P='b';
ACC=numrec62(idref,idtest,P);
toc

tic
idtest,P='c';
ACC=numrec62(idref,idtest,P);
toc

tic
idtest,P='d';
ACC=numrec62(idref,idtest,P);
toc

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
idref='d';
refgen62(idref,P);

tic
idtest,P='a';
ACC=numrec62(idref,idtest,P);
toc

tic
idtest,P='b';
ACC=numrec62(idref,idtest,P);
toc

tic
idtest,P='c';
ACC=numrec62(idref,idtest,P);
toc

tic
idtest,P='d';
ACC=numrec62(idref,idtest,P);
toc


%tic y toc mide el tiempo que transcurre en correrse el programa
```

```
function refgen62(idref,P)
% refgen62.m
% Este archivo genera el vocabulario  de referencia de coeficientes
% cepstral. Los coeficientes para cada palabra se guardan  en una matriz
% de P x de m.  P es el orden y de M es el número de ventanas a procesar.
% El orden P es determinado, el tamaño de la ventana es N, y el tamaño de
% paso de la ventana es M.
%P=8;  es del número de los coeficientes del LPC

M=128; % window step size
N=256; % window size
% Read digit wav files
id=idref; % identity of speaker used (jl,kl,pk,or vk)
[uno,dos,tres]=readin2(id);
ccref_1=mfcc(uno,M,N,P);
ccref_2=mfcc(dos,M,N,P);
ccref_3=mfcc(tres,M,N,P);
save('vocab.mat','ccref_1','ccref_2','ccref_3');
save('ccref_1');
save('ccref_2');
save('ccref_3');


function Accur=numrec52(idref,idtest,P)
% numrec5.m
% Este archivo genera el vocabulario de referencia de coeficientes
% cepstral.  Los coeficientes para cada palabra se guardan en matriz de P
% x de m.  P es el orden y  m es el número de ventanas a procesar.
% El orden es  determinado por P, el tamaño de la ventana es N, y el
% tamaño de paso de la ventana es M.
load vocab;
%P=8; % number of LPC coeffs
M=128; % window step size
N=256; % window size
% Read test digit wav files
id=idtest; % identity of speaker used (jl,kl,pk,or vk)
[uno_t,dos_t,tres_t,fs]=readin2(id);
cct_1=lfcc(uno_t,M,N,P);
cct_2=lfcc(dos_t,M,N,P);
cct_3=lfcc(tres_t,M,N,P);
%Ocupa como reerencia el archivo ccref_1
ref=ccref_1;
test=cct_1;
Match(1,1)=dtw1(test,ref);
test=cct_2;
Match(2,1)=dtw1(test,ref);
test=cct_3;
Match(3,1)=dtw1(test,ref);

%Ocupa como referencia el archivo ccref_2
ref=ccref_2;
test=cct_1;
Match(1,2)=dtw1(test,ref);
test=cct_2;
Match(2,2)=dtw1(test,ref);
test=cct_3;
Match(3,2)=dtw1(test,ref);
```

```
%Ocupa como referencia el archivo ccref_3
ref=ccref_3;
test=cct_1;
Match(1,3)=dtw1(test,ref);
test=cct_2;
Match(2,3)=dtw1(test,ref);
test=cct_3;
Match(3,3)=dtw1(test,ref);

idref
idtest
%
accum=0;
%for i=1:11
for i=1:3
    [val,ind]=min(Match(:,i));
    if ind==i
        accum=accum+1;
    end;
end;

Accur=100*accum/3


function Accur=numrec6(idref,idtest,P)
% numrec6.m
% Este archivo genera el vocabulario de referencia de coeficientes
% cepstral. Los coeficientes para cada  palabra se guardan en una matriz
% de P x de m.  P es el orden y m es el número de ventanas a procesar.
% El orden de P es determinado, el tamaño de la ventana es N, y el tamaño
% de paso de la ventana es M.

load vocab;
%P=8; % number of LPC coeffs
M=128; % window step size
N=256; % window size
% Read test digit wav files
id=idtest; % identity of speaker used (jl,kl,pk,or vk)
[uno_t,dos_t,tres_t,fs]=readin2(id);
cct_1=mfcc(uno_t,M,N,P);
cct_2=mfcc(dos_t,M,N,P);
cct_3=mfcc(tres_t,M,N,P);

ref=ccref_1;
test=cct_1;
Match(1,1)=dtw1(test,ref);
test=cct_2;
Match(2,1)=dtw1(test,ref);
test=cct_2;
Match(3,1)=dtw1(test,ref);
%
ref=ccref_2;
test=cct_1;
Match(1,2)=dtw1(test,ref);
test=cct_2;
Match(2,2)=dtw1(test,ref);
```

```
test=cct_3;
Match(3,2)=dtw1(test,ref);
%
ref=ccref_3;
test=cct_1;
Match(1,3)=dtw1(test,ref);
test=cct_2;
Match(2,3)=dtw1(test,ref);
test=cct_3;
Match(3,3)=dtw1(test,ref);

idref
idtest
%
accum=0;
for i=1:3
    [val,ind]=min(Match(:,i));
    if ind==i
        accum=accum+1;
    end;
end;
Accur=100*accum/3

function RMS = L2norm(i, j, test, ref);

P=10;
for k = 1:10
    diff_sq(k,1) = (ref(k,j)-test(k,i))^2;
end;
summation = sum(diff_sq);
RMS= (1/P)*(summation)^0.5;

% lfcc.m
% Calcula los coeficientes cepstral para la secuencia, usando una
% longitud N de la ventana, el tamaño del paso de la ventana es M, y
% orden P.

function ccep=lfcc(y,M,N,P);
NYQ=N/2;
seqlen=size(y,1);
m=0;
startpt=1;
endpt=N;
m=1;
while endpt<=seqlen
    winseq=hamming(N).*y(startpt:endpt);
    logspec=log(abs(fft(winseq)));
    for i=1:P
        coefwin=cos(pi*i*linspace(0,NYQ-1,NYQ)/NYQ)';
        ccep(i,m)=sum(coefwin.*logspec(1:NYQ));
    end;
    m=m+1;
    startpt=1+(m-1)*M;
    endpt=startpt+N-1;
end;
```

```
% mfcc.m
% Calcula los coeficientes cepstral para la secuencia y; usando la
% longitud N de la ventana, con un paso de la ventana de tamaño M
% y un orden P.

function ccep=mfcc(y,M,N,P);
NYQ=N/2;
% Triangle Filter Defs
for i=1:10
    fstart(i)=(2*i)-1;
end;
fcent=fstart+2;
fstop=fstart+4;
fstart(11)=23;
fstart(12)=27;
fstart(13)=31;
fstart(14)=35;
fstart(15)=40;
fstart(16)=46;
fstart(17)=55;
fstart(18)=61;
fstart(19)=70;
fstart(20)=81;
fcent(11)=27;
fcent(12)=31;
fcent(13)=35;
fcent(14)=40;
fcent(15)=46;
fcent(16)=55;
fcent(17)=61;
fcent(18)=70;
fcent(19)=81;
fcent(20)=93;
fstop(11)=31;
fstop(12)=35;
fstop(13)=40;
fstop(14)=46;
fstop(15)=55;
fstop(16)=61;
fstop(17)=70;
fstop(18)=81;
fstop(19)=93;
fstop(20)=108;
seqlen=size(y,1);
m=0;
startpt=1;
endpt=N;
m=1;
while endpt<=seqlen
    winseq=hamming(N).*y(startpt:endpt);
    magspec=abs(fft(winseq));
    for i=1:20 % Calc triangle filter outputs
        for j=fstart(i):fcent(i)
            filtmag(j)=(j-fstart(i))/(fcent(i)-fstart(i));
        end;
        for j=fcent(i)+1:fstop(i)
```

```
            filtmag(j)=1-(j-fcent(i))/(fstop(i)-fcent(i));
        end;

Y(i)=sum(magspec(fstart(i):fstop(i)).*filtmag(fstart(i):fstop(i))');
    end;
    Y=log(Y.^2);
    for i=1:P
        coefwin=cos((pi/20)*i*(linspace(1,20,20)-0.5))';
        ccep(i,m)=sum(coefwin.*Y');
    end;
    m=m+1;
    startpt=1+(m-1)*M;
    endpt=startpt+N-1;
end;
```

% mfcc.m
% Calcula los coeficientes cepstral para la secuencia y; usando una
% longitud N, con un paso de la ventana de tamaño M y P de la ventana.

```
function ccep=mfcc(y,M,N,P);
NYQ=N/2;
% Triangle Filter Defs
for i=1:10
    fstart(i)=(2*i)-1;
end;
fcent=fstart+2;
fstop=fstart+4;
fstart(11)=23;
fstart(12)=27;
fstart(13)=31;
fstart(14)=35;
fstart(15)=40;
fstart(16)=46;
fstart(17)=55;
fstart(18)=61;
fstart(19)=70;
fstart(20)=81;
fcent(11)=27;
fcent(12)=31;
fcent(13)=35;
fcent(14)=40;
fcent(15)=46;
fcent(16)=55;
fcent(17)=61;
fcent(18)=70;
fcent(19)=81;
fcent(20)=93;
fstop(11)=31;
fstop(12)=35;
fstop(13)=40;
fstop(14)=46;
fstop(15)=55;
fstop(16)=61;
fstop(17)=70;
fstop(18)=81;
fstop(19)=93;
fstop(20)=108;
```

```
seqlen=size(y,1);
m=0;
startpt=1;
endpt=N;
m=1;
while endpt<=seqlen
    winseq=hamming(N).*y(startpt:endpt);
    magspec=abs(fft(winseq));
    for i=1:20 % Calc triangle filter outputs
        for j=fstart(i):fcent(i)
            filtmag(j)=(j-fstart(i))/(fcent(i)-fstart(i));
        end;
        for j=fcent(i)+1:fstop(i)
            filtmag(j)=1-(j-fcent(i))/(fstop(i)-fcent(i));
        end;

Y(i)=sum(magspec(fstart(i):fstop(i)).*filtmag(fstart(i):fstop(i))');
    end;
    Y=log(Y.^2);
    for i=1:P
        coefwin=cos((pi/20)*i*(linspace(1,20,20)-0.5))';
        ccep(i,m)=sum(coefwin.*Y');
    end;
    m=m+1;
    startpt=1+(m-1)*M;
    endpt=startpt+N-1;
end;
```

APÉNDICE B

MANUAL DE USUARIO

Cálculo con la frecuencia lineal de los coeficientes cepstral.

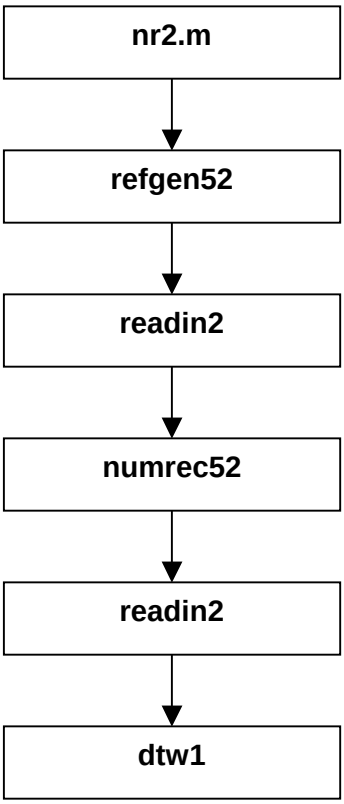


Diagrama de flujo de nr2.m

Programa principal o main ***nr2.m*** en el que se siguen los siguientes pasos:

1. Se especifica el directorio donde este el archivo que se va a utilizar como referencia, por ejemplo: ***Idref = 'a'*** donde se le asigna a idref el directorio ***a***.
2. Después se llama a la función ***refgen52 (idref, P)*** que recibe como parámetros el directorio donde se encuentran los archivos de referencia ***idref*** y ***P*** (número de coeficientes LPC).
3. Después se comienza a tomar el tiempo que transcurre entre los pasos los pasos 4 y 5 con ***tic.***
4. En el paso siguiente ***idtest = 'a'*** se asigna a idtest el directorio donde están los archivos de test.

5. Después se manda a llamar a la función ***numrec52 (idref, idtest, P)*** y se le asigna a la variable ACC .
6. Como último paso se deja de medir el tiempo transcurrido con ***toc.***

En ***refgen52(idref ,P)*** esta función genera los archivos de referencia de coeficientes cepstral. Esta recibe como parámetros ***idref*** el directorio donde están los archivos de referencia y ***P*** que es el orden o número de coeficientes de LPC. Para cada palabra es salvada en una matriz de P x M; m es el número de ventanas procesadas, N es el tamaño de la ventana y el paso del tamaño de la ventana es M. En esta fusión se siguen los siguientes pasos:

1. ***M*** es asignada con el tamaño de los pasos de la ventana.
2. ***N*** es el tamaño de la ventana.
3. ***id*** se le asigna el directorio donde esta el archivo wav
4. Se llama a la función ***readin2(id)***
5. Se llama a la función ***lfcc***
6. Se salva los valores devueltos por la función lfcc con ***save (file_ccref)***

En ***lfcc*** se calcula la frecuencia lineal de los coeficientes cepstral para la secuencia, con una ventana de longitud N, paso del tamaño de la ventana M y orden P.

En ***readin2(id)*** . ***Id*** es el directorio donde esta el archivos que se van a comparar (tests). En esta función se siguen los siguientes pasos:

1. *Leer los archivos wav* con la siguiente función ***[Vector, fs, b]=wavread(file)***. Donde se lee el archivo ***file*** donde file vector es el vector que genera la muestra, ***fs*** el radio de la muestra y ***b*** el número de bits de la muestra.
2. Se normalizan a 1.0 los valores almacenados en el vector

En ***numrec52(idref,idtest,P)*** esta función genera los archivos de referencia de coeficientes cepstral. Esta recibe como parámetros ***idref*** el directorio donde están los archivos de referencia y ***P*** que es el orden o número de coeficientes de LPC. Para cada palabra es salvada en una matriz de P x M; m es el número de ventanas procesadas, N es el tamaño de la ventana y el paso del tamaño de la ventana es M. En esta fusión se siguen los siguientes pasos:

1. ***M*** es asignada con el tamaño de los pasos de la ventana.
2. ***N*** es el tamaño de la ventana.
3. ***id*** se le asigna el directorio donde esta el archivo wav
4. Se llama a la función ***readin2(id)***
5. Se llama a la función ***lfcc***
6. Después ocupa como referencia uno de los archivos y lo compara con uno de los de test en la función dtw1; y lo almacena en un vector bidimensional M realizándolo sucesivamente con todos los archivos de test.
7. Después por último busca en la matriz M los valores mínimos de cada columna y si el número de columna es el mismo que el archivo de referencia se incrementa un contador, es decir se busca las coincidencias entre el archivo de referencia y el test sacando un porcentaje de estas coincidencias.

En ***dtw1(test,ref)*** se realiza el algoritmo Dynamic Time Warping

Cálculo con la frecuencia mel de los coeficientes cepstral.

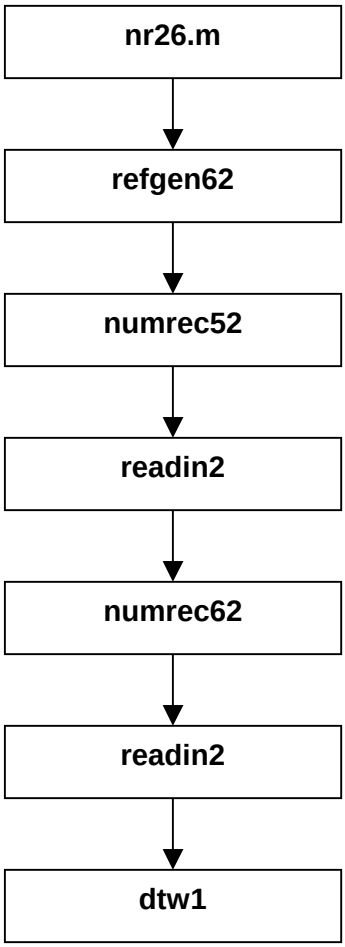


Diagrama de flujo de nr26.m

Programa principal o main ***nr26.m*** en el que se siguen las siguientes pasos:

7. Se especifica el directorio donde este el archivo que se va a utilizar como referencia; por ejemplo: ***Idref = 'a'*** donde se le asigna a idref el directorio ***a***.

8. Después se llama a la función ***refgen62 (Idref , P)*** que recibe como parámetros el directorio donde se encuentran los archivos de referencia ***idref*** y ***P***(número de coeficientes LPC).

9. Después se comienza a tomar el tiempo que transcurre entre los pasos los pasos 4 y 5 con ***tic.***

10. En el paso siguiente ***idtest = 'a'*** se asigna a idtest el directorio donde están los archivos de test.

11. Después se manda a llamar a la función ***numrec62 (idref, idtest, P)*** y se le asigna a la variable ACC.
12. Como último paso se deja de medir el tiempo transcurrido con ***toc.***

En ***refgen62 (idref ,P)*** esta función genera los archivos de referencia de coeficientes cepstral. Esta recibe como parámetros ***idref*** el directorio donde están los archivos de referencia y ***P*** que es el orden o número de coeficientes de LPC. Para cada palabra es salvada en una matriz de P x M; m es el número de ventanas procesadas, N es el tamaño de la ventana y el paso del tamaño de la ventana es M. En esta fusión se siguen los siguientes pasos:

7. ***M*** es asignada con el tamaño de los pasos de la ventana.
8. ***N*** es el tamaño de la ventana.
9. ***id*** se le asigna el directorio donde esta el archivo wav
10. Se llama a la función ***readin2(id)***
11. Se llama a la función ***mfcc***
12. Se salva los valores devueltos por la función lfcc con ***save (file_ccref)***

En ***mfcc*** se calcula la frecuencia mel de los coeficientes cepstral para la secuencia y con una ventana de longitud N, paso del tamaño de la ventana M y orden P .

En ***readin2(id)*** donde ***id*** es el directorio donde esta el archivos que se van a comparar (tests). En esta función se siguen los siguientes pasos:

3. *Leer los archivos wav* con la siguiente función ***[Vector, fs ,b]=wavread(file)***. Donde se lee el archivo ***file*** donde file vector es el vector que genera la muestra, ***fs*** el radio de la muestra y ***b*** el número de bits de la muestra.
4. Se normalizan a 1.0 los valores almacenados en el vector

En ***numrec62(idref,idtest,P)*** esta función genera los archivos de referencia de coeficientes cepstral. Esta recibe como parámetros ***idref*** el directorio donde están los archivos de referencia y ***P*** que es el orden o número de coeficientes de LPC. Para cada palabra es salvada en una matriz de P x M; m es el número de ventanas procesadas, N es el tamaño de la ventana y el paso del tamaño de la ventana es M. En esta fusión se siguen los siguientes pasos:

8. ***M*** es asignada con el tamaño de los pasos de la ventana.
9. ***N*** es el tamaño de la ventana.
10. ***id*** se le asigna el directorio donde esta el archivo wav
11. Se llama a la función ***readin2(id)***
12. Se llama a la función ***mfcc***
13. Después ocupa como referencia uno de los archivos y lo compara con uno de los de test en la función dtw1; y lo almacena en un vector bidimensional M realizándolo sucesivamente con todos los archivos de test.
14. Después por último busca en la matriz M los valores mínimos de cada columna y si el número de columna es el mismo que el archivo de referencia se incrementa un contador, es decir se busca las coincidencias entre el archivo de referencia y el test y se saca un porcentaje de estas coincidencias.

En ***dtw1(test,ref)*** realiza el algoritmo Dynamic Time Warping.

La función ***L2Norm***, calcula el error cuadrático.

BIBLIOGRAFIA

LIBROS

[Bernal y Bobadilla] Bernal Bermúdez Jesús, Gómez Vilda Pedro y Bobadilla Sancho Jesús, "Reconocimiento de voz y fonética acústica", Editorial Alfa omega y RA-MA, España (2000)

PAGINAS DE INTERNET

[1] http://www.gerc.eng.ufl.edu/STUDENTS/Jland/proj1/proj1.html (en esta dirección se encuentra el código fuente en Matlab)
[2] http://alek.pucp.edu.pe/~dflores/tesis/dtw.html
[3] http://galia.fc.uaslp.mx/~ortega/CAPITULO%207.htm
[4] http://ceres.ugr.es/~rubio/vitae/papers/preproceso.pdf
[5] http://www.euskalnet.net/iosus/speech
[6] http://www.infor.uva.es/~cesargf/proyectos/memorias/reconocimiento/tema3.pdf
[7] http://mail.udlap.mx/~ingrid/ingrid/RV/Proc_de_Voz.html
[8] http://www.net-research.net/swen/contraintelige/newpage21.htm

LINKS DE INTERES

http://www.geocities.com/CapeCanaveral/Runway/3015/inicial.html
http://www.cnel.ufl.edu/~kkale/6825Project.html (desarrollan el algoritmo en java)