UNIVERSIDAD NACIONAL DE INGENIERIA

FACULTAD DE INGENIERIA INDUSTRIAL Y DE SISTEMAS

ESCUELA PROFESIONAL DE INGENIERIA DE SISTEMAS

RECONOCIMIENTO DE VOZ

TESIS

PARA OPTAR EL TITULO PROFESIONAL
DE INGENIERO DE SISTEMAS

GLEN DARIO RODRIGUEZ RAFAEL

LIMA, 1997

Dedicado a mis padres

INDICE

SUMARIO

El problema que se ha planteado es el desarrollo de una método o de una mezcla de métodos que permitan realizar el reconocimiento de voz dependiente del hablante y en palabra aislada, además de evaluar su grado de certeza, sus ventajas y sus defectos.

Se ha adoptado los métodos de evaluación matricial difusa y análisis del espectro de frecuencias. La implementación específica de esta solución ha pasado por sucesivos ensayos de prueba y error, hasta obtener una solución aceptable.

Las conclusiones a las que se han llegado son: los métodos adoptados son adecuados para situaciones controladas de trabajo, pero no para situaciones extremas ni de oficina. La búsqueda de mejores métodos pasa por la indexación de los datos del sonido y el perfeccionamiento de los métodos de comparación.

DESCRIPTORES TEMATICOS

1. Tecnologías de sonido y audio digital.
2. Reconocimiento de voz.
3. Enfoques dependientes del hablante
4. Enfoques de palabra aislada.
5. Evaluación matricial
6. Procesamiento digital de señales (Digital signal processing)
7. Espectros de frecuencias
8. Análisis de ondas
9. Sonidos vocálicos y no vocálicos.

INTRODUCCION

La motivación fundamental de la presente tesis es la tendencia claramente perceptible de que los usuarios finales sistemas informáticos exigirán, activa o pasivamente, mejores interfases hombre-computadora, especialmente las interfases para la entrada de datos. Dichas interfases han pasado a través de un lento proceso de evolución por las fases de los indicadores de luces (propio de las primitivas computadoras que se programaban por hardware), el comando modo texto, las GUI (interfases gráficas de usuario) y algunos tímidos intentos por implementar interfases orientadas a objetos. Cada una de estas etapas ha presentado paradigmas y formas peculiares, propias del estado de la tecnología en el momento de ser creadas y de las características del usuario a quien trataban de satisfacer. Pero lo que la mayoría de especialistas cree es que habrá una interfase que se establecerá en un futuro no muy lejano, y que dará el empuje final a las tecnologías de información en su afán por llegar a todas las personas y todas las tareas, y esa es la interfase verbal.

Existen en el mercado algunas implementaciones de esta interfase, pero

han sido de lejos muy limitadas en su alcance, debido a que las tecnologías necesarias están en su infancia, y sobre todo, arrastran consigo los paradigmas de las interfase GUI. Los productos actuales que tratan de implantar interfases verbales solo repiten los esquemas de las GUIs, de tal manera que la voz se convierte en meramente un "shortcut", un sinónimo de una opción del menú de las aplicaciones con interfase gráfica. La interfase verbal brinda la oportunidad de cambiar totalmente la relación hombre-computadora, de tal manera que la computadora se adapte a la naturaleza del hombre y no a la inversa, como hasta la actualidad ocurre. Para ello se requiere pensar en los nuevos paradigmas que la interfase verbal traerá tanto para los usuarios neófitos como para los profesionales de la informática.

Para la realización plena de este nueva interfase se requiere la madurez de dos tecnologías: el procesamiento del lenguaje natural y el reconocimiento de la voz humana. La presente tesis se centra en esta última tecnología y en su rol dentro de las interfases verbales.

Una de los propósitos de la presente tesis, además del estudio de la tecnología de reconocimiento de voz, ha sido el entregar un trabajo con un alto contenido experimental, caso muy raro en la especialidad de Ingeniería de Sistemas de la UNI. El autor cree que la Facultad forma a sus alumnos con dos sesgos peligrosos: el consumismo y el formalismo. Somos consumistas de tecnología ajena, no somos creadores. Los pocos profesionales que escapan a esta

categoría generalmente caen en el segundo sesgo, el formalismo, que consiste en realizar investigación y desarrollo de metodologías, algoritmos y otras estructuras formales que no se comparan con una realidad para ser confirmadas o rechazadas; casos como la proposición de metodologías de análisis y diseño de sistemas, de la elaboración de modelos de simulación y otros, son investigaciones con poco o nulo componente experimental. En el caso de las simulaciones, el contenido experimental es descuidado por la elaboración de una estructura formal (modelo) más impresionante o ingeniosa, cuando el objetivo del modelo es en realidad tomar datos de la realidad, crear el modelo que explique su comportamiento, volver a la realidad a ver que tanto se ajusta el modelo a la realidad, y finalmente usar el modelo para mejorar la realidad.

El formalismo no es malo de por sí, lo que es malo es que se constituya en un sesgo que ha cortado las habilidades y destrezas de experimentación en los estudiantes y egresados FIIS. Esto nos ha cerrado muchas vías de desarrollo, en temas de investigación y de producción vinculados al desarrollo de conocimientos vía la experimentación. También ha reducido la efectividad y la trascendencia de nuestros esfuerzos en las pocas investigaciones y desarrollos que se han realizado hasta la fecha. El autor eleva sus votos para que la FIIS retome la orientación experimental que siempre ha estado ligada a la Ciencia y a la Ingeniería.

1. CONCEPTOS GENERALES

1.1- Audio Digital

La naturaleza del sonido en el mundo físico es ondulatoria, y como todo fenómeno del mundo macroscópico material, es analógico. El sonido es una onda mecánica longitudinal que se propaga a través de medios elásticos materiales (como el aire y el agua).

Las ondas son perturbaciones en un medio que se propagan a través del mismo a una velocidad constante y característica del medio. La onda no es un objeto material, pero sí es una entidad física real, ya que se mueve, transporta energía e interacciona con los objetos materiales. Una onda longitudinal es una onda en la que los puntos del medio se mueven hacia atrás y hacia adelante, en dirección paralela a la de la propagación de la onda.

Al desplazarse a través del aire, el sonido se comporta como una onda esférica que se extiende en esferas concéntricas desde un punto central (el origen

del sonido). Cuando la onda se propaga a través del aire, las moléculas del aire se desplazan hacia atrás y hacia adelante alrededor de su posición de equilibrio. Lo que varia en una posición en el espacio son la densidad y la presión. Lo que detecta nuestro oído es dicha variación de presión. El oído humano puede detectar como sonido a las ondas longitudinales dentro del intervalo entre 20 y 20000 Hz.

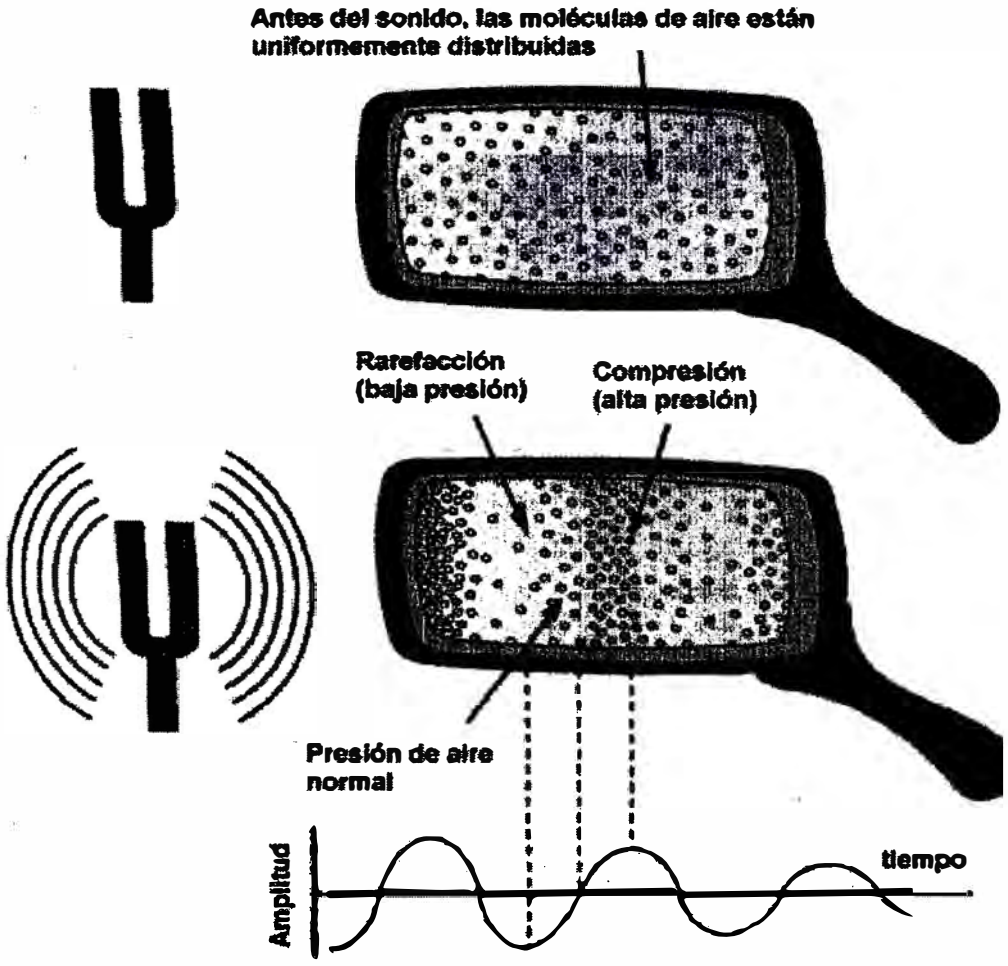


Figura 1: Relación entre los fenómenos sonoros y la onda sonora

La intensidad que capta el oído humano esta en función de la diferencia que existe entre la presión máxima de la onda y la presión del aire en estado no perturbado; sin embargo esa función es no lineal, pues también intervienen

elementos subjetivos o de percepción (procesos fisiológicos y psicológicos).

Para que el sonido pueda ser grabado, almacenado, reproducido o procesado por una computadora, es necesario que se pueda traducir en información digital, mediante un proceso conocido en inglés como "analog to digital conversion" (ADC, conversión de analógico a digital). Una vez que el sonido se ha convertido en bytes, la CPU puede hacer infinidad de cosas con él: puede añadirse reverberación, ecos, mezclar dos sonidos diferentes, eliminar extractos de sonido, etc. Por un proceso de "digital to analog conversion" (DAC) se convierte la serie de bytes en una señal eléctrica analógica que pueda después pasar por unos parlantes y convertirse en sonido.

1.2- Captura - Sampling

Las tarjetas de sonido tienen un circuito de PCM (pulse code modulation = modulación en código de pulsos). Cuando el sonido llega a un micrófono, es convertido en una señal eléctrica de similar comportamiento.

El ratio de muestreo (sample rate o sampling rate) se define como el número de muestras de sonido que se toman por unidad de tiempo. A mayor ratio de muestreo, mejor calidad del sonido, por que se capta en forma más fiel los cambios en el sonido. De acuerdo al Teorema de Nyquist (ver Apéndice A) sólo se puede reproducir con confiabilidad una onda si el ratio de muestreo es por lo

menos el doble de la frecuencia del componente de mayor frecuencia. Como el oído humano puede escuchar sonidos de hasta 20000 Hz, entonces la mínima velocidad de muestreo podría ser hasta de unos 40000 Hz.

El otro componente de la calidad del sonido digital es el tamaño del muestreo. Consiste en el tamaño de cada dato individual del muestreo (en otras palabras, el tamaño de la variable que almacena una muestra). Los más comunes tamaños son 8 y 16 bits por sample. A mayor tamaño de muestreo, mayor el rango de intensidades de sonido que se puede representar. Por ejemplo, con 8 bits podemos representar 256 niveles de intensidad (unos 50 decibeles), mientras que con 16 bits podemos representar 65536 niveles (unos 90 decibeles).

El último componente en la calidad del sonido es el número de canales. Según este parámetro, se puede dividir el sonido en monoaural o en stereo. Para efectos de la voz humana, un solo canal basta (mono).

1.3- Pros y contras del audio digital

Las ventajas en el uso del audio digital son enormes. La capacidad de procesar de mil formas el sonido, ya sea aislado o mezclando un sonido con otro ha permitido la creación de una nueva industria de edición y efectos de sonido. En el campo del reconocimiento de voz, el audio digital es la forma más extendida de captura para los programas comerciales debido a la masiva difusión

de tarjetas de sonido.

Sin embargo, esta tecnología tiene sus inconvenientes. El principal de ellos es el espacio de memoria o de disco que ocupa el audio digital. Este puede variar entre 8000 Kb por segundo (la peor calidad, con muestreo de 8000 samples por segundo, soportado por formatos de audio bastante antiguos y en desuso) hasta 176400 Kb por segundo (calidad de equipo CD stereo). Esto significa que a mayor calidad, mayores recursos se gastan. Incluso en la calidad más pobre, el consumo de disco es grande en comparación de los formatos MIDI (música sintetizada) y los de algunas tarjetas propietarias que efectúan procesos previos a la voz (de compresión o de reducción a parámetros).

1.4- Reconocimiento de voz o Reconocimiento del habla

Reconocimiento de habla es la tecnología que da a la computadora la habilidad de comprender ordenes verbales, o en términos más generales, de usar la voz como un medio adicional de interfase. Dentro del campo del reconocimiento de habla hay diversos enfoques y "dominios de problema". Algunos son relativamente sencillos, mientras que otros son más complicados. Podemos agruparlos de las siguientes maneras:

1.4.1- Según los hablantes que reconoce

1.4.1.1- Dependientes del hablante (speaker-dependant).- es necesario que el usuario entrene previamente al sistema para que este pueda usar el sistema.

1.4.1.2- Independientes del hablante (speaker-independant).- no es necesario un entrenamiento previo. En algunos casos no se necesita indicarle nada al sistema, en otros debe indicarse a que grupo pertenece el usuario (hombre, mujer, joven varón, joven mujer, niño) a fin de facilitar el proceso de reconocimiento.

1.4.2- Según la continuidad del habla

1.4.2.1- Palabra aislada (isolated-word).- el sistema reconoce una palabra a la vez, o es necesario hacer una pausa muy significativa entre palabra y palabra.

1.4.2.2- Habla continua (continuous-speech).- el sistema puede reconocer las palabras pronunciadas en forma contínua, a la velocidad de una conversación lenta por lo menos.

En la presente tesis, desarrollaremos un modelo dependiente del hablante y de palabra aislada.

1.5- Aplicaciones del reconocimiento de voz

Las principales aplicaciones del reconocimiento de voz se dan en las actividades en las cuales las interfases tradicionales (línea de comandos y GUIs) no bastan o no se pueden usar. Por ejemplo, actividades donde las dos manos ya están ocupadas por el teclado y el mouse pero aún necesita más entrada de datos, o en las cuales las manos o la vista son usadas en otros dispositivos como bisturies o microscopios. Sin embargo, la mayor cantidad de esfuerzos han sido desplegados para satisfacer las necesidades de las siguientes áreas:

1.5.1- Automatización de Oficinas

En las oficinas modernas existen muchas tareas que se realizan con la asistencia de las computadoras personales, desde la redacción de documentos hasta el correo electrónico. Y muchas de estas tareas son actualmente realizadas por el propio interesado, no por su secretaria o su asistente, al mismo tiempo que firma documentos o contesta el teléfono. El oficinista sólo tiene dos manos, por lo que sus tareas más rutinarias a veces se ven interrumpidas o retrasadas. Esta situación ha motivado que muchas personas o empresas se encuentren a la búsqueda de herramientas de "ofimática" que permitan automatizar algunas tareas cotidianas. Una herramienta que no cae en esta categoría pero que sin duda ayudaría mucho es el reconocimiento de voz, aplicada como un sustituto del mouse y del teclado.

Los desarrollos más vanguardistas del día de hoy están relacionados con el desarrollo de motores de reconocimiento de voz vía Internet. La idea

consiste en usar la gran potencia de los procesadores de los web servers o de otras computadoras con arquitectura RISC para procesar pedidos de búsqueda o consultas específicas grabadas en la máquina con el web browser. Así la computadora personal no necesita un sofisticado software para el que requeriría mucho espacio, y las consultas como "Clima en Lima" o "Buscar: computadoras portátiles" se realizarían verbalmente en lugar de llenar campos o formularios . Los problemas a solucionar son la carga de tráfico y una tecnología de reconocimiento más veloz, por que evidentemente los cuellos de botella son las líneas de comunicación y el procesador del servidor. El MIT (Instituto de Tecnología de Massachusetts) y un popular buscador de Internet (Yahoo) son las primeras entidades que desarrollan software para incursionar en este campo desde 1996. Microsoft esta usando otro enfoque, que consiste en que el CPU de la PC con web browser realice el reconocimiento de voz, como un "agente" del sistema operativo, y traslade el resultado al web server vía TCP/IP.

1.5.2- Procesos Productivos Industriales

Varios procesos industriales, tales como el ensamblaje de autos y la fabricación de componentes electrónicos, requieren del uso de herramientas tan complejas que no son suficientes las dos manos para operarlas eficientemente, o que requieren tener la vista concentrada en el objeto de fabricación. Para estos procesos, la sustitución de botones de mando por comandos de voz representa una mejora en la productividad y en la seguridad

del trabajador. Otros procesos no son tan complejos tecnológicamente, pero la reducción de personal en muchas empresas obliga al trabajador a realizar más tareas simultáneamente, ocupando ambas manos.

1.6- Aplicaciones Comerciales disponibles actualmente.

Muchas de los principales corporaciones de hardware y software han tenido equipos o laboratorios dedicados al desarrollo de tecnologías de reconocimiento de voz. Algunos de ellos no proporcionaron mayores frutos, mientras que otros lograron el desarrollo de sistemas comerciales de mayor o menor éxito. La llegada de las microcomputadoras de mediana potencia de cómputo (a partir de los modelos de procesador 80386) permitieron que las tecnologías de multimedia se llevaran de las workstation a las PCs,. comenzando con la venta masiva de tarjetas de sonido de la marca pionera Adlib, y llegando a mayores niveles de prestaciones y ventas con Creative Voice y su Sound Blaster. Muchas pequeñas compañías en ese momento comenzaron a hacer sus pininos en las tecnologías de reconocimiento de voz para PCs, destacándose entre ellos Dragon Systems, quien hasta el día de hoy se mantiene en los primeros lugares de la industria. Mientras los primeros intentos de reconocer palabras fueron implementados mitad por software, mitad por hardware (tarjetas de sonido con componentes adicionales para reconocimiento de voz), las actuales soluciones comerciales vienen implementadas mayoritariamente por software, pudiendo emplearse una tarjeta de sonido común y corriente.

Entre los software mencionados destacan:

1. Dragon Dictate, producto de Dragon Systems. Este software permite tanto programar macros, o sea, sustitución de comandos de mouse y teclado por comandos de voz, así como dictados simples. Se incluye en muchos juegos y aplicaciones educativas, debido a que su motor de reconocimiento de voz es relativamente pequeño y rápido, además tiene librerías para implementarlo parcialmente en DOS, a diferencia de la mayoría de competidores.

2. IBM VoiceType, producto de IBM. Las últimas versiones son motores muy completos de sustitución de programación de macros de voz y de apuntador de dictados con muy buena calidad. Sus versiones iniciales necesitaban una tarjeta de sonido especial, que hacía procesamiento digital de señales usando el procesador MWave de IBM. Sin embargo, las últimas versiones se adaptan a cualquier tarjeta de sonido soportada por el sistema operativo (Windows u OS/2), y necesita hacer uso intensivo del CPU, por lo que sólo es funcional en PCs Pentium o superiores.

3. Voice Assist, de Creative Labs. Esta herramienta de reconocimiento de voz es bastante sencilla, es de palabras aislada y dependiente del hablante y orientada a reemplazar teclas de función o comandos de menú por

comandos orales. Sin embargo es sumamente rápida incluso con una 386, usa pocos recursos de computador y funciona aceptablemente con poco entrenamiento, siempre y cuando se mantenga un ritmo similar en el momento del habla. Caso contrario su capacidad de reconocimiento decae notoriamente.

2. PATRONES DE LA VOZ HUMANA

En esta sección usaremos algunas muestras de voz para proponer un esquema general que explique:

a) Las características distintivas de cada voz.

b) Las características comunes de todas las voces estudiadas en cada sonido.

c) Las diferencias entre la voz humana y el sonido ambiental.

2.1- El tracto vocal.

En esta sección discutiremos las características anatómicas del tracto vocal humano, de la generación del sonido y del habla. Este conocimiento es la base para establecer ciertas premisas para la presente investigación.

2.1.1- Funciones del tracto vocal

Obsérvese la estructura anatómica del tracto vocal en la siguiente figura:

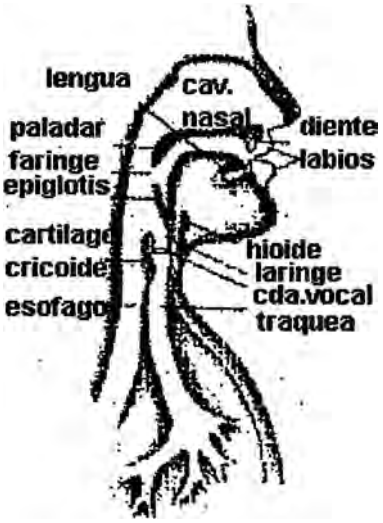


Figura 2: Organos vocales del hombre

En la figura 2 se muestra un corte transversal de la cabeza humana, en donde se pueden apreciar la principales partes del aparato fonador. El tracto vocal propiamente dicho es un tubo de unos 17 cm de longitud. Este termina en un extremo con las cuerdas vocales (o la abertura entre ellas, la glotis) y el otro lado por los labios. La sección de área está determinada por el emplazamiento de los labios, lengua, mandíbula y el velo del paladar, y puede variar desde 0 (completamente cerrada) hasta aproximadamente 20 cm^2.

Una cavidad auxiliar, el tracto nasal, puede ser acoplada al tracto vocal por medio de la acción de la puerta atrapadora del velo. El tracto nasal comienza en el velo y termina en las fosas nasales. En el hombre, la cavidad es aproximadamente de 12 cm de largo y tiene un volumen de unos 60 cm^3. Durante los sonidos no nasales el vello sella la cavidad nasal y ningún sonido es producido en las fosas nasales. Así como las fosas nasales, la caja toráxica,

los senos paranasales y la zona nasal de la faringe actúan como resonadores adicionales al tracto vocal en algunos sonidos.

El sonido se puede producir de 3 formas distintas. Los sonidos vocálicos son producidos por la elevación de la presión del aire en los pulmones, forzando un flujo de aire a través de las cuerdas vocales (la glotis) y causando la vibración de las cuerdas. La interrupción del flujo produce pulsos cuasiperiódicos de espectro ancho que excitan el tracto vocal, tal como lo haría un instrumento de viento. Los ligamentos vibrantes de las cuerdas vocales son de aproximadamente 18 mm de longitud y la abertura glotal es de unos 5 mm^2.

Los sonidos fricativos de voz son generados por la formación de una constricción en algún punto del tracto y forzando al aire a través de la constricción para producir turbulencias. Una fuente de presión sonora es creada entonces. Los mayores representantes de estos sonidos en el habla son los sonidos de las consonantes fe, ye y equis.

Los sonidos explosivos son el resultado de cerrar completamente la cavidad, reconstruyendo la presión nuevamente hacia el frente detrás de la cerradura y liberándola abruptamente. El ruido fricativo o aspiración típicamente sigue a la liberación transitoria del aire. Esto se aprecia en las sílabas que empiezan con pe, be, te y de. En general, ambos sonidos, los

fricativos y los explosivos son similares desde el punto de vista de sus características como ondas, por lo que se agrupan en sonidos no vocálicos.

2.1.2- Modelo del tracto vocal

Antes de establecer un modelo del tracto vocal, pasaremos a revisar rápidamente cómo es que el tracto vocal puede generar muchos diferentes sonidos con tan pocos elementos móviles.

La producción del habla se puede dividir en dos fases distintas: la producción de un sonido audible y el control ejercido sobre este sonido genérico para producir otro sonido específico. Todos los sonidos tienen su origen en la espiración. El aire que es expulsado de los pulmones asciende por la tráquea, pasa a través de la laringe, entra en la faringe y sale por la nariz y por la boca. Las cuerdas vocales permanecen relajadas mientras se respira normalmente, pero cuando se desea emitir un sonido vocálico ellas se cierran y taponan la laringe. Cuando se expira el aire, aumenta la presión bajo las cuerdas, hasta que la presión se hace suficientemente grande para que una parte del aire pase con fuerza entre las cuerdas, reduciendo la presión, con lo que nuevamente se cierran y se repite el proceso.

De este modo se producen una serie periódica de vibraciones sonoras con una frecuencia que depende de la tensión y la masa de las cuerdas vocales. En

general la frecuencia es menor en los varones que en las mujeres por que las cuerdas vocales del varón son tienen más masa; en el varón promedio la frecuencia de esta señal es de unos 125 Hz. El sonido producido hasta este momento se puede considerar como la superposición de muchas ondas sinusoidales, cuyas frecuencias forman una sucesión armónica del tipo f_1=125 Hz, f_2=250 Hz, f_3=375 Hz, etc. cada uno de los cuáles tiene una amplitud más o menos igual.

Luego esta señal pasa por las cavidades resonadoras de la boca, nariz y faringe, que se configuran en distinta forma según el sonido deseado. Esta forma es la causa que una misma señal de origen resone en forma distinta y cree distintos sonidos finales. Desde este punto de vista, estas cavidades actúan como un ecualizador sumamente complejo. Cada configuración para los sonidos vocálicos hace resonar especialmente 2, 3 ó 4 frecuencias diferentes. Esto puede apreciarse mejor en la figura 3.

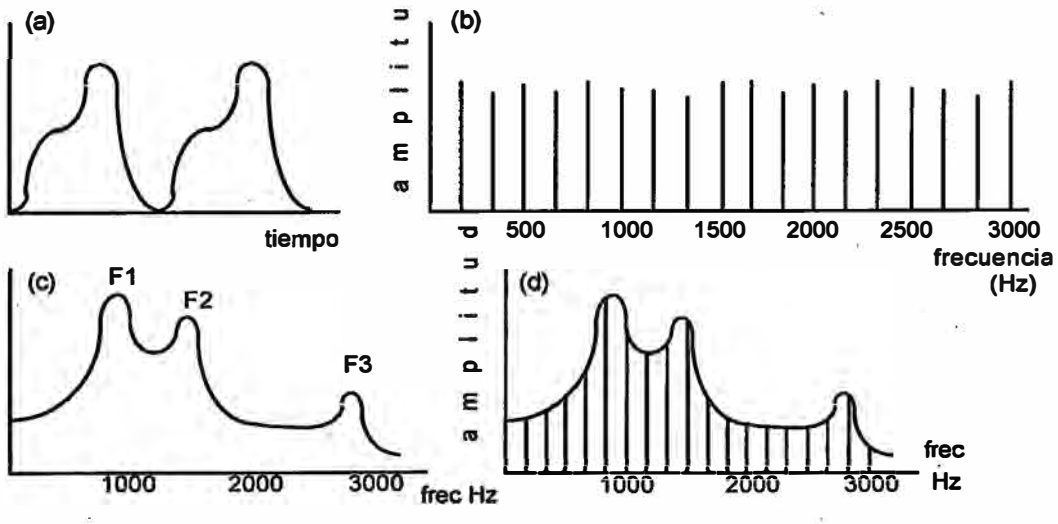


Figura 3: Fases en la producción de un sonido vocálico

En (a), se aprecia la vibración proveniente de la laringe como una onda periódica. En (b) se ve el espectro de la onda de (a) como una serie armónica de amplitud más o menos similar. En (c) se aprecia la curva de respuesta de la cavidad bucal como una función de resonancia. En (d) se aprecia el espectro del sonido después de resonar en la cavidad bucal, lo que forma el sonido vocálico.

Los sonidos no vocálicos son creados por un mecanismo distinto, por el estrechamiento del tracto vocal por otro elemento diferente a las cuerdas vocales, como puede ser la lengua, los labios, los dientes o la nariz, o por su cierre completo en los labios. Se puede decir que se originan en una fuente de ruido, por que la señal inicial que los origina es mucho no está compuesta por una serie armónica (se puede considerar de espectro contínuo) y es menos homogénea en sus amplitudes. Esta señal también es modificada por los resonadores que conforman el tracto vocal superior.

Gracias a estos conceptos, el análisis de las señales habladas se puede llevar a cabo mediante un modelo, similar a los que se estudian en la teoría de control de señales, que describe el proceso del habla clasificando las señales en dos tipos:

- Las señales sonoras que se caracterizan por tener alta energía (mayor

amplitud) y contenido frecuencial en el rango de los 100 Hz a 4000 Hz las cuales se generan por intermedio de las cuerdas vocales y además presentan cierta periodicidad.

- Las señales no sonoras que se caracterizan por tener baja energía (menor amplitud) y componente frecuencial uniforme presentando aleatoreidad en forma de ruido blanco.

Es así que la señal dividida en dos partes en el modelo se presenta con dos fuentes de excitación las que alimentan al sistema acústico que conforma el tracto vocal. La figura 4 muestra el modelo.

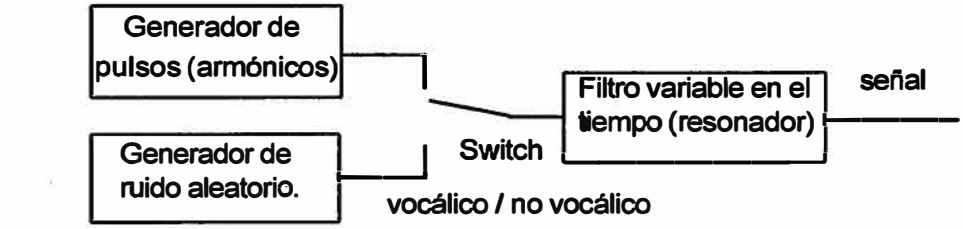


Figura 4: Modelo de excitación y filtro.

El tracto vocal modelado se manifiesta como un filtro variable en el tiempo cuyos parámetros varían en el tiempo en función de la acción consciente que se realiza al pronunciar una palabra. El filtro variable en el tiempo tiene dos posibles señales de entrada que dependerán del tipo de señal, sonora o no sonora, con características bastante diferentes.

Para señales sonoras la excitación será un tren de impulsos de frecuencia controlada, mientras que para las señales no sonoras la excitación será ruido aleatorio. La combinación de estas señales modelan el funcionamiento de la glotis. El espectro de frecuencias de la señal vocal puede obtenerse a partir del producto del espectro de la excitación por la respuesta en frecuencia del filtro. El tracto vocal manifiesta un número muy grande de resonancia, sin embargo se considera sólo las tres o cuatro primeras que toman el nombre de formantes y cubren un rango de frecuencias entre 100 y 3500 Hz. Esto debido a que las resonancias de alta frecuencias son atenuadas por la característica frecuencial del tracto que tiende a actuar como un filtro pasabajo con una caída de aproximadamente -12 dB por octava. Este modelo es una simplificación del proceso del habla. Los sonidos no vocálicos no se filtran por el tracto con la misma extensión en que lo hacen las señales sonoras por lo que el modelo no es muy preciso para este tipo de señales. Además el modelo supone que las dos señales puede separarse sin considerar ninguna interacción entre ellas, lo cual no es cierto ya que la vibración de cuerdas vocales es afectada por las ondas de presión dentro del tracto. Sin embargo estas consideraciones pueden ser ignoradas resultando el modelo lo suficientemente adecuado.

2.2- Patrones y características de los fonemas

El sonido en general se puede representar como ondas, y toda onda se puede expresar de manera única como la superposición de ondas sinusoidales de longitudes de onda y amplitudes definidas (Teorema de Fourier). Las diferentes

componentes sinusoidales se pueden representar en gráficos con la frecuencia en el eje de las X y la amplitud en el eje de las Y (a estos gráficos se les llama espectros de la onda inicial). Una onda periódica (o en el caso de los sonidos, para un extracto determinado que se tome como periódico) tiene un espectro discreto, formado por ondas de longitud de onda λ/n , donde lambda es la longitud de onda del período y n es un numero entero mayor que 0.

En el caso de instrumentos musicales que dan notas perfectas, la onda es periódica y estacionaria; es entonces cuando se han efectuado estudios en diferentes instrumentos y se ha llegado a la conclusión que aunque la frecuencia fundamental, que define la nota, es igual en todos los instrumentos, los espectros son diferentes, con diferente importancia relativa de cada una de las demás componentes. Esa diferencia en las componentes armónicas contribuyen a la calidad del tono del instrumento (algo así como el timbre de voz de cada instrumento), que lo diferencia de los demás. Es por eso que un Fa del piano es diferente al Fa del violín; más delante se discutirá este punto en más profundidad, pero es necesario su conocimiento para comprender el estudio de los fonemas.

Los fonemas son unidades mínimas de las palabras que son diferenciables auditivamente. El estudio de la fonética tiene una larga trayectoría como disciplina, debido a la necesidad de conocer los sonidos de una lengua para estudiar sus orígenes, para poder enseñarla a quien no la conoce, o para ayudar a quien tiene problemas de lenguaje. Los fonetistas han encontrado que el

castellano tiene unos 24 fonemas en España, y unos 22 ó 23 en Latinoamérica. Estos son:

- En España: /p/, /b/, /t/, /d/, /k/, /g/, /f/, /θ/, /S/, /y/, /x/, /ĉ/, /m/, /n/, /ñ/, /l/, /l̲/, /r/, /rr/, /a/, /e/, /i/, /o/, /u/
- En Latinoamérica: /p/, /b/, /t/, /d/, /k/, /g/, /f/, /S/, /y/, /x/, /ĉ/, /m/, /n/, /ñ/, /l/, /l̲/ (partes de Perú, Bolivia y Paraguay), /r/, /rr/, /a/, /e/, /i/, /o/, /u/

Lo que ha interesado en este trabajo es el hecho que el habla de la lengua castellana se puede dividir en fonemas, y que algunos de estos fonemas se pueden representar convenientemente como si fuesen las notas musicales, pues en pequeños lapsos de tiempo son prácticamente un tono sostenido. Estos fonemas son los vocálicos: /a/, /e/, /i/, /o/, /u/.

En cambio, los fonemas consonanticos, ya sean fricativos u oclusivos, o alguna variante propia del castellano, no se pueden considerar como una onda con comportamiento periódico, sino como un ruido con cierto patrón .Esto ya se vio en la sección 2.2.2 .

2.3- La frecuencia y la amplitud de las ondas.

Como hemos descrito, las notas (y los fonemas puros también, en tanto se asemejan a la nota de un instrumento), tienen una frecuencia fundamental y varias

armónicas, cada una con diferentes amplitud. La frecuencia fundamental es la que define el tono (la propiedad del sonido que distingue lo grave de lo agudo) y las demás frecuencias caracterizan el timbre de cada voz.

El papel de la frecuencia fundamental no es importante en la distinción de los fonemas; la mayor prueba de ello es que podemos entender perfectamente tanto el habla de hombres como de mujeres a pesar de la diferencia enorme en la frecuencia fundamental. Sin embargo, el papel de la amplitud si es importante, debido a una propiedad dentro de las palabras conocida como entonación. La entonación viene a ser una variación en la intensidad del sonido en el tiempo que permite distinguir a una palabra de otra.

Sin embargo, la entonación no es una característica que por sí sola permite identificar una palabra, por que existen muchas palabras que comparten una misma entonación, y casi toda entonación puede atribuirse a más de una palabra. Para afinar el criterio de identificación hay que agregar más propiedades a la comparación.

Esa otra propiedad puede ser la variación de la frecuencia en el tiempo. Es bastante cierto que el tono promedio de un sonido no nos indica nada, pero el aumento o disminución de tono sí tiene relevancia en la identificación de una palabra. La excepción a esta regla es el canto: cuando se canta, las palabras no sufren la misma variación de frecuencia que en el habla normal.

2.4- Rasgos comunes y diferencias entre las diferentes voces.

Las explicaciones del punto 2.1 sobre los instrumentos musicales también son validas para la voz. Por más que uno trate de lograr la misma "nota" de otra voz, las componentes armónicas son distintas y diferencian una voz de otra. Y eso en el supuesto negado que todas las voces hablaran en el mismo tono (frecuencia fundamental).

Sin embargo, para que nos podamos entender entre hablantes de un mismo idioma, las voces que conforman una palabra deben respetar ciertas convenciones. Si no fuese así, no podríamos comunicarnos. Por ello es que las frecuencias fundamentales y las armónicas, si bien no son las mismas en las diferentes personas, guardan cierta coherencia que pasamos a describir en los siguientes párrafos.

La primera fase de la producción del habla es la producción por medio de las cuerdas vocales, de una onda periódica que tiene un espectro de armónicos de amplitud casi igual. La segunda fase es la amplificación de algunos de estos armónicos por la resonancia dentro de la cavidad del habla (involucrando a la lengua, el paladar y los labios). La idea es que de un conjunto de componentes armónicas más o menos de la misma amplitud, se amplifiquen algunas y otras se disminuyan, usando al tracto vocal como un equalizador, operado naturalmente por el cerebro.

Los diferentes sonidos vocálicos se distinguen por sus frecuencias formantes, pero no de una manera decisiva. Pero entre hombres y mujeres las formantes son más diferentes para cada vocal: para la A larga (como en father) la frecuencia fundamental en los hombres es de 124 Hz. en promedio y para las mujeres es 212; para la O de afternoon es de 137 Hz para hombres y 232 Hz para mujeres. esto nos lleva a la conclusión que la frecuencia fundamental no juega un papel importante para discernir los fonemas vocálicos. Sin embargo, la razón de las frecuencias formantes para una determinada vocal es aproximadamente la misma. Por ejemplo, las razones F_2/F_1 y F_3/F_1 para la A larga son de 1.49 y 3.34 en hombres y de 1.44 y 3.31 para mujeres. De esto se desprende que las razones juegan un papel determinante a la hora de distinguir un fonema de otro. Un hombre puede hablar con voz de falsete, con lo que cambia la frecuencia fundamental de sus cuerdas pero no las frecuencias formantes de su cavidad del habla.

2.5- Distinción entre la voz humana y el ruido ambiental.

El primer proceso que se debe realizar con la voz digitalizada, tanto en la fase de entrenamiento como en la fase de reconocimiento, es aislar la palabra del "silencio" circundante. Se remarca que el "silencio" para propósitos de esre programa de aplicación no es el silencio absoluto, sino un nivel relativamente bajo de ruido ambiental. Se debe remarcar que incluso en un ambiente

aparentemente silencioso pueden existir sonido en frecuencias no audibles por el oído humano, peor que por el teorema de Nyquist (véase Anexo A) quedan grabados o digitalizados como ruidos de frecuencias fantasmas. A primera vista, parecía que era una labor relativamente sencilla distinguir el silencio de la palabra hablada, pero el caso es que muchos fonemas (especialmente los de ciertas consonantes) pueden llegar a confundirse con el ruido ambiental, si se analizan fuera del contexto de todo el sonido digitalizado.

El ruido ambiental puede ser considerado como un ruido blanco o aleatorio, pero recuérdese que las consonantes también son producto de un ruido aleatorio, con ciertas modificaciones (resonancias). La mayor diferencia entre el ruido ambiental y los sonidos no vocálicos es la intensidad, es decir, la amplitud. Aún así, los inicios de un sonido no vocálico, débiles por el hecho de su reciente generación, son suficientemente débiles como para ser confundidos con ruido ambiental.

Para distinguirlos se ha usado un algoritmo que examina tanto un segmento del sonido como su contexto. Los pasos que se siguen son los siguientes:

1. Ir al inicio del sonido digitalizado.
2. Tomar el subsiguiente segmento de sonido (40 muestras ó 2 ms aproximadamente).

3. Si en promedio, su intensidad cae en un rango R_1 (silencio perfecto ± 1.8) es "silencio" previo a la palabra y se vuelve a (2)
4. Caso contrario, tomamos los siguientes 2 ms (aproximadamente). Si el promedio de estos 2 ms cae en el rango R_2 , entonces el segmento original es "silencio" previo a la palabra y se regresa a (2).
5. En el otro caso, este es el comienzo de la palabra.
6. Por un proceso similar, pero del final para el comienzo, se haya el fin de la palabra
7. Vamos al inicio de la palabra
8. Se toma el subsiguiente segmento de sonido.
9. Se realiza el proceso de condensar información hallando los máximos y mínimos de la onda.
10. Si esta fuera del rango R_3 , entonces sigue siendo palabra y se va a (8).
11. Caso contrario, no es palabra y se va a (8)

Los rangos R1, R2 y R3 se definen en la proximidad del nivel de silencio más o menos unos 4 niveles de intensidad.

La ejecución de este proceso es vital, porque centra nuestros análisis posteriores solamente en los segmentos de sonido que son nuestra área de interés directa. Podemos apreciarlo mejor mediante el siguiente diagrama de flujo:

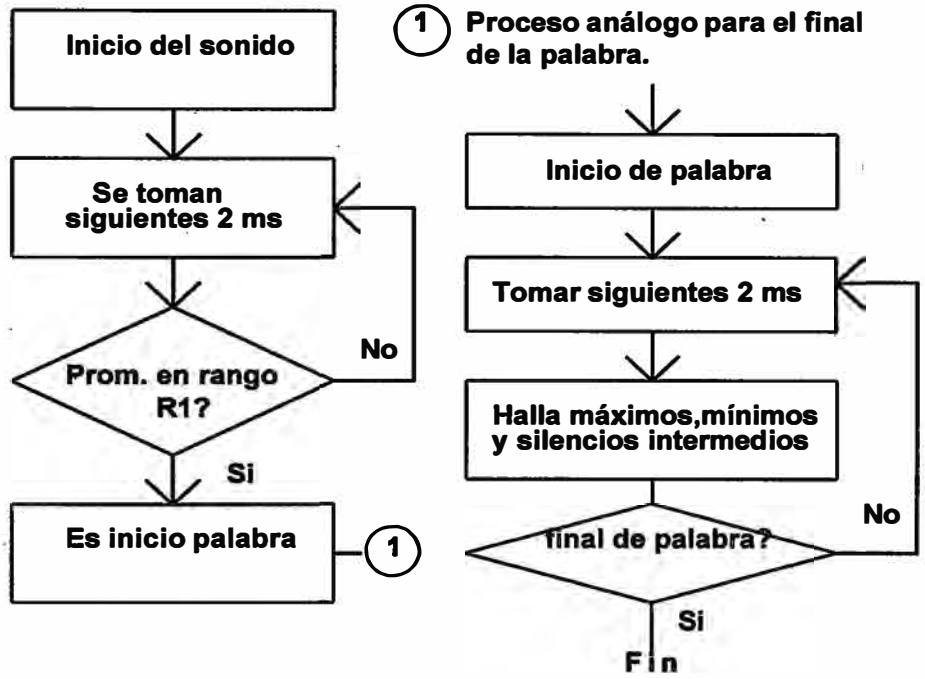


Diagrama I: Delimitación de la palabra y procesos afines

3. ESTRUCTURA-MODELO Y ALGORITMO DE RECONOCIMIENTO

3.1- Parsing de los sonidos. Modelos que los representan.

Al inicio de esta investigación ,se pretendía encontrar un modelo que contuviese en si mismo suficiente información para describir el sonido hablado y compararlo con otros sonidos. Este vendría a ser el primer modelo, que se procede a describir en los puntos 3.1.1 y 3.1.2. Sin embargo, después de trabajar con este modelo y hacer una importante cantidad de pruebas, se llegó a la conclusión que no era completamente funcional, adoleciendo de una serie de inconvenientes en la precisión del reconocimiento, a pesar de lo cuál no era descartable. Por lo tanto, se empezó a desarrollar un segundo modelo, que consistiría de la adición al primer modelo de un segundo sub modelo, lo que se describe a partir del punto 3.1.3 en adelante.

3.1.1- Información relevante de la voz. Modelo matricial.

Si llegásemos a capturar una palabra promedio (entre 0.4 y 0.6 segundos de duración) son una frecuencia de muestreo de 22050 Hz, y usáramos toda la información de un formato PCM, es decir sin compresión, obtendríamos entre 8820 y 13230 datos, de tal forma que entre un dos datos consecutivos sólo

habría una diferencia del orden de centésimas de milésimas de segundo (1 / 22050 = 0,000045 s). No es razonable usar toda la carga de datos que las tarjetas de sonido nos brindan, por que en cierto modo hay mucha redundancia entre los datos (por ejemplo, para una serie de 5 ó 10 datos consecutivos, la secuencia puede llegar a ser lineal o escalonada). Además, desde el punto de vista computacional, trabajar con toda esa cantidad de datos consumiría demasiado tiempo de CPU, demasiado espacio en memoria y en disco, en resumen, sería muy ineficiente. La solución es usar resumir esos datos, usando un subconjunto de ellos.

Sin embargo, no se puede seleccionar caprichosamente dicho subconjunto. Se podría tomar una muestra cada cuatro (la muestra No.1, la 5, la 9, etc.) por poner un ejemplo, lo cual reduciría los datos a una cuarta parte, unos 3000 datos para la palabra promedio, pero eso sería lo mismo que cambiar la frecuencia de muestreo a la cuarta parte, o sea 5000 Hz aproximadamente. Según el teorema de Nyquist (ver anexo A) se generarían sonidos fantasma para las frecuencias mayores de 2500 Hz, las cuáles son muy comunes en la voz femenina y también se presentan en algunos sonidos de la voz masculina. El criterio para escoger los datos que definirían el sonido debe ser mejor que el ejemplo anterior, de tal manera que no perjudique la precisión de la data y que permita que se reduzca más la cantidad de dichos datos. Los principales factores que se deben preservar son:

- La frecuencia
- La amplitud
- La posición en el tiempo

Además, se debe lograr la menor pérdida de información, el mayor ahorro de espacio y un algoritmo rápido.

El método que se ha escogido es la reducción a máximos y mínimos y su expresión en forma matricial. Consiste en describir a la onda en función de sus puntos de máxima y mínima amplitud relativa (máximos y mínimos locales). Demostraremos que su utilización tiene las bondades necesarias. La discusión de la estructura matricial y su razón de ser se detallan mejor en la sección 3.2.

Los máximos y mínimos respetan la amplitud, ya que esta no es sino la diferencia entre máximos y mínimos, así como la frecuencia, por lo menos en forma discreta. La inmensa mayoría de sonidos del habla humana están encima de los 300 Hz.; como las ondas son de mediana o alta frecuencia, la onda sinusoidal se puede aproximar bastante bien a un diente de sierra (resultado que se puede obtener por el teorema de Fourier). El cálculo de máximos y mínimos reduce el número de muestras a menos de 10% del sonido (500 a 1000 muestras).

El algoritmo consiste en un barrido de los datos y en la identificación del cambio de pendiente de la curva, como se muestra a continuación:

1. Se va al comienzo de la palabra (como se menciona en el algoritmo de distinción entre voz humana y ruido ambienta).
2. Al signo de la pendiente y a la última pendiente valida se les asigna el valor de 0.
3. Se pasa al siguiente sample del sonido.
4. Si el valor de este sample es menor al del anterior, se asigna al signo de la pendiente el valor de -1.
5. Si el valor de este sample es mayor al del anterior, se asigna al signo de la pendiente el valor de 1.
6. Si ambos son iguales se asigna al signo de la pendiente el valor de 0.
7. Si el actual signo de la pendiente es igual al anterior y no son cero, se asigna su valor a la última pendiente valida y se regresa al paso 3.
8. Si el actual signo de la pendiente es igual al anterior y son cero, se regresa al paso 3.
9. Si el actual signo de la pendiente es diferente al anterior y ninguno es cero, se asigna el valor del actual signo de la pendiente a la última pendiente valida, se graba el actual punto en la relación de puntos máximos y mínimos, y se regresa al paso 3.
10. Si el actual signo de la pendiente es cero se regresa al paso 3 (posible inicio de un asíntota)

11. Si el actual signo de la pendiente es diferente de cero ,y al anterior es cero, y el actual signo de la pendiente es igual a la última pendiente valida, se regresa al paso 3 (fin de la asíntota).
12. Si el actual signo de la pendiente es diferente de cero ,y al anterior es cero, y el actual signo de la pendiente es diferente a la última pendiente valida, se graba el actual punto en la relación de puntos máximos y mínimos, y se regresa al paso 3.

3.1.2- Los silencios intermedios

Una vez simplificada la data, se deben efectuar procesos que la hagan más significativa aún, para proceder después a almacenar un conjunto de voces como el "concepto" de la palabra.

El primer proceso que vamos a realizar es la identificación de los silencios intermedios y su eliminación del conjunto de datos. Los silencios son elementos muy valiosos en las palabras con sonidos fricativos como "izquierda". La presencia de silencios, además de aportar información adicional en sí misma, nos permite reducir la cantidad de datos a almacenar.

El algoritmo de identificación de los momentos de silencio es similar al que separa el inicio y fin de las palabras, excepto en el hecho que se debe ser más flexible con la búsqueda de sonidos. Es decir, que debe tener unos límites más pequeños para considerar a un fragmento del sonido como silencio, y

unos límites más grandes para considerarlo como voz humana. Cuando se realizaban las pruebas iniciales para fijar estos límites, se advirtió que en algunas palabras como "izquierda" existe un gran vacío entre una sílaba y la siguiente, por lo menos grande para el sistema (hasta 0.3 s). El enfoque inicial era estricto, es decir que si pasaba cierta cantidad de tiempo (2000 muestras ó 0.1 s aproximadamente) sin que la señal sonora retomara las características exigidas para el inicio de la señal, se consideraba terminada la palabra. El caso de la palabra "izquierda" llevó a reconsiderar este criterio, que finalmente busca inicio y fin de la palabra desde ambos extremos de la señal grabada, y dentro de estos límites, establece un criterio menos exigente para considerar que una parte de la señal es vocalización humana.

El algoritmo es el siguiente:

1. Se va al comienzo de la palabra (como se menciona en el algoritmo de distinción entre voz humana y ruido ambienta).
2. Tomar el subsiguiente segmento de sonido (2 ms aproximadamente).
3. Si en promedio, su intensidad cae en un rango R_1 (silencio perfecto ± 1.5) es "silencio en medio de la palabra".
4. Caso contrario, se registra como parte de la voz, y también se toma en cuenta para el cálculo de máximos y mínimos.

Por conveniencia en la programación, este proceso se hace

simultáneamente a la ubicación de los máximos y mínimos. Si el algoritmo encuentra un fragmento de "silencio en medio de la palabra", se obvia el algoritmo que busca máximos y mínimos hasta que se reinicia el sonido.

En el siguiente diagrama se puede observar los procesos generales desarrollados hasta esta sección:

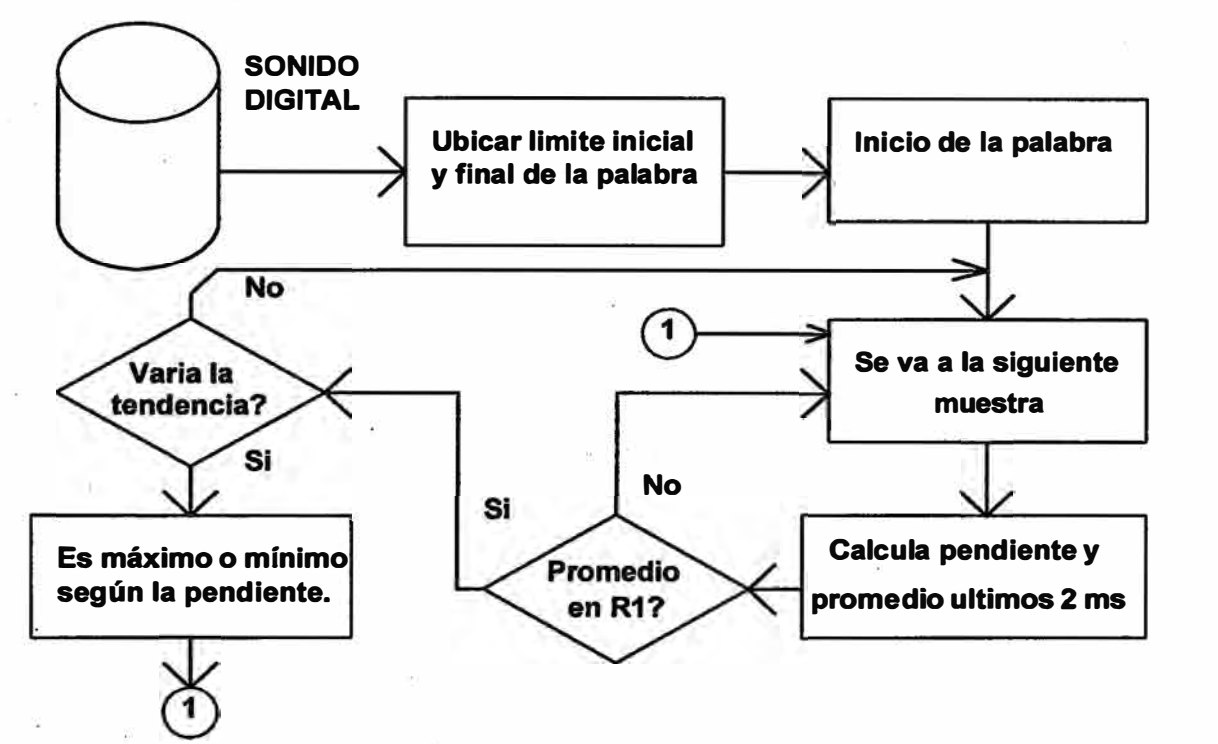


Diagrama II: Procesos relacionados al modelo matricial con máximos y mínimos

3.1.3- Análisis de ondas y el espectro para las vocales

Al inicio del desarrollo de esta investigación, se tenía la idea que se podría crear un motor de reconocimiento de voz sin necesidad de usar el análisis de ondas más allá de sus conceptos fundamentales. Mientras se desarrollaban los prototipos iniciales del modelo y del sistema, empezaban a surgir

inconvenientes en el modelo y los algoritmos usados que los hacían algo menos precisos de lo pensado inicialmente. La acumulación en etapas sucesivas de estas imprecisiones o inexactitudes se tradujo en que la palabra correcta , es decir la que correspondía al sonido pronunciado, no siempre era la primera o tenia alternativas muy cercanas. El número de casos llegaba a un 50% de aciertos categóricos, un 10% de aciertos marginales y un 40% de fallos para un diccionario de 20 palabras.

Habían dos alternativas: se descartaba el modelo matricial, o se complementaba con otras herramientas. El modelo matricial, pese a su alta tasa de fallos, tenía la virtud que la palabra correcta siempre estaba entre las 3 ó 4 primeras y que erraban la palabra realmente pronunciada por un pequeño margen. Esto influyó en que se escogiera la segunda alternativa, la búsqueda de otros criterios de comparación que ayudaran al modelo matricial. Y entre las herramientas más usadas en la tecnología de voz se encuentra el análisis de ondas, pero de naturaleza digital, no analógica.

En los diferentes sistemas actualmente usados ha habido un enorme aporte de una rama de la ingeniería conocida como Procesamiento Digital de Señales (DSP - Digital Signal Processing). Los ingenieros electrónicos han sido quienes más han desarrollado las técnicas de DSP orientadas a sus campos de acción, debido a que la mayor parte de sus fundamentos descansan el Análisis de Ondas o de Fourier, tan usado en telecomunicaciones. Sin embargo, el

DSP puede usarse un muchas otras aplicaciones, tal como el análisis del sonido, que es otro tipo de onda. Lo que distingue al DSP del análisis de señales normal es que estudia el análisis de ondas de las que sólo se conoce un conjunto de muestras discretas, es decir, sólo se conoce el valor digital de ciertos puntos de la onda.

Las diferentes operaciones que se deben realizar se hacen sobre un conjunto de samples o muestras, generalmente a una frecuencia de muestreo predeterminada, y luego estas muestras nos revelan las características de la onda en forma muy cercana a lo que haría el análisis analógico de la onda original. Toda aplicación de los métodos del DSP tiene tres partes:

- Obtener el espectro de frecuencias de una señal.
- Operar sobre este espectro (alterándolo o sólo obteniendo parámetros de él).
- Reconstruir la señal del nuevo espectro (sólo si se alteró en el paso anterior).

Si se efectúan operaciones que alteran el espectro, se pueden hacer modificaciones a la señal que serían impensables por otro medio: filtros, suavización, ecualización, etc. Si se obtienen ciertos valores o parámetros del espectro, es posible obtener información sobre la conformación de las ondas imposible de obtener de otra fuente. En el caso del presente estudio, la utilidad principal del DSP es identificar pequeños fragmentos del sonido con algunos sonidos específicos.

Lo primero que se debe tener en cuenta es que la voz como tal no es una onda adecuada para usar el análisis de ondas. Para que una señal representada por una función f(t) en el tiempo, de periodo T, pueda ser analizada en una serie de senos y cosenos (o lo que es lo mismo, en una serie de senos tal como se demuestra en el Anexo D) se debe cumplir las siguientes condiciones:

- f(t) tiene un número finito de máximos y mínimos dentro de T.
- f(t) tiene un número finito de discontinuidades dentro de T
- La integral en el tiempo del valor absoluto f(t) en T debe ser una cantidad finita.

Para el caso de una señal física, como un fragmento de voz, se cumplen las tres condiciones. La única salvedad es que hay que simular que la señal es periódica, por el expediente de considerar un fragmento limitado de la señal, y repetirlo periódicamente hacia ambos lados. Establecidas estas convenciones, podemos considerar a un fragmento de la señal como objeto de análisis, a fin de obtener su espectro de frecuencias.

Se ha determinado, mediante muchas experiencias previas, que la máxima longitud en que una señal se comporta más o menos como una onda periódica es de unos 20 ms, ó en unas 440 muestras. Pero el análisis discreto de ondas se basa en un algoritmo conocido como Transformada Rápida de Fourier ó

Método de Cooley-Tukey que necesita un número de muestras o samples potencia de 2. Para nuestro caso, se han escogido $2^9 = 512$ muestras de sonido.

El algoritmo de Cooley-Tukey hace uso intensivo de los números complejos, y en líneas generales se trata de realizar una serie de operaciones sobre las 512 entradas y obtener otras 512 salidas, de las cuáles son relevantes las primeras 256, por que la otra mitad son idénticas en orden invertido. Los detalles del algoritmo se detallan en el Anexo D.

El fragmento de sonido que se debe escoger tiene que representar a un mismo fonema, en el caso contrario el espectro obtenido se vuelve dependiente del momento preciso en que se pasa de un fonema a otro, lo cuál no puede determinarse a priori. Es por ello que se necesita de algún método para asegurar que el fragmento a analizar pertenece a un mismo fonema. Este proceso es difícil en el caso general, es decir, si se desea saber cuando termina un sonido fricativo y cuando comienza una vocal, por ejemplo en el siguiente gráfico se muestra la señal producida al vocalizar la palabra "doce".

Obsérvese lo difícil que es distinguir el fin de los fonemas **d** y **s** del inicio de las vocales **o** y **e,** en el siguiente gr'afico:

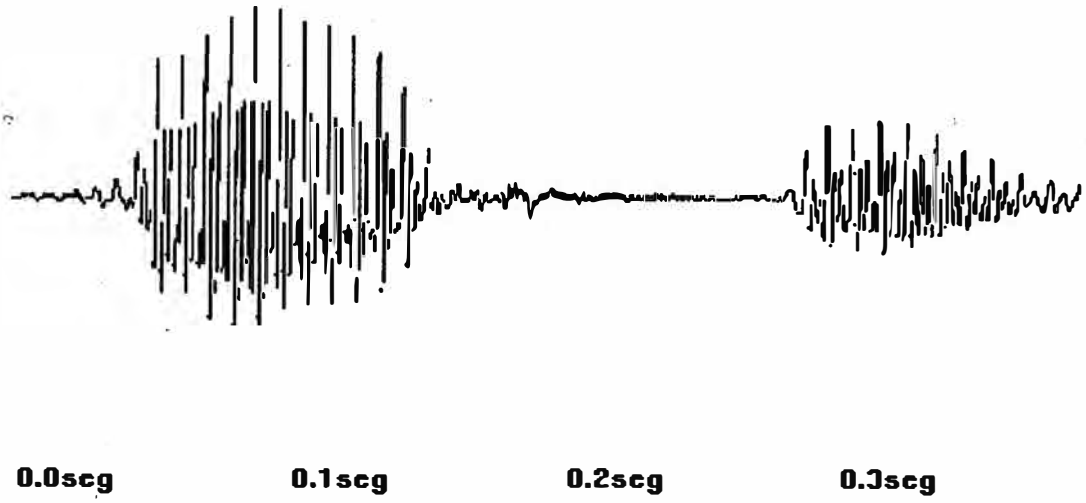


Figura 5: Señal del sonido producido al vocalizar "doce"

Lo que sucede es que en nuestro idioma existe un breve período de transición entre los fonemas vocálicos y no vocálicos consecutivos, o entre dos sonidos vocálicos consecutivos. Este período de transición se produce por varias razones, algunas propias del idioma castellano (que no es gutural) y otras propias del aparato fonador del ser humano, que permite que por un brevísimo espacio de tiempo un sonido siga resonando en las cavidades internas aún después que se dejó de emitir. Además, el sonido no vocálico en general es de una gran complejidad y su aparación en la palabra es muy fugaz en el tiempo. Los estudiosos del DSP simplifican el estudio de las consonantes considerándolas como "ruido blanco".

Estos fenómenos son las razones por la que el presente trabajo no ha

podido resolver satisfactoriamente el reconocimiento de la totalidad de la palabra por medio del análisis de onda, ya que sólo nos brindaría información confiable de las vocales mismas, información muy distorsionada de las mezclas vocálicas-no vocálicas, e información inútil de los sonidos no vocálicos.

Hay casos en que el sonido no vocálico tiene una intensidad propia, pudiendo llevar a confusión sobre su naturaleza. Por ello el algoritmo que ha de discernir los fragmentos seguros de sonido vocálico del resto de la palabra debe ser conservador, es decir, evitar que se consideren vocálicos aquellos sonidos que no lo son, a expensas de cometer a veces el error inverso, el de descartar sonidos que si son vocálicos. El esquema general del algoritmo a usarse se detalla a continuación:

1- Empezamos por el segundo máximo-mínimo
2- Identificamos si es un máximo si es mayor que el máximo-mínimo anterior. Si este es el caso continuamos, si no se va a (7)
3- Comparamos el actual máximo con los últimos 12 máximos. Si es mayor que ellos, o la amplitud que alcanza desde el nivel neutral es mayor que el 95% de la amplitud que alcanza el máximo de los 12 últimos alcanza desde el nivel neutral, marcamos el flag de máximos como +1, caso contrario como -1.
4- Actualizamos los últimos 12 máximos.

5- Si el anterior flag de máximos fue +1 y es igual que el actual, se suma 1 al contador "maxasc". Caso contrario, se repone a 0.

6- Se va a (10).

7- Comparamos el actual mínimos con los últimos 12 mínimos. Si es menor que ellos, o el valor absoluto de la amplitud que alcanza desde el nivel neutral es mayor que el 95% del valor absoluto de la amplitud que alcanza el máximo de los 12 últimos alcanza desde el nivel neutral, marcamos el flag de mínimos como +1, caso contrario como -1

8- Actualizamos los últimos 12 mínimos.

9- Si el anterior flag de mínimos fue +1 y es igual que el actual, se suma 1 al contador "minasc". Caso contrario, se repone a 0.

10- Si en los últimos 12 máximos el máximo de ellos es uno de los primeros 6, si la situación análoga se presenta en los últimos 12 mínimos, y los contadores maxasc y minasc son mayores que 30, podremos considerar que este fragmento es vocálico, y se usará el punto del primer máximo de la lista como inicio de su análisis. Además, saltamos 20 máximos-mínimos en la búsqueda.

11- Pasamos al siguiente máximo-mínimo, o terminamos si llegamos al último.

12- Regresamos a (2).

En lineas generales, este algoritmo nos permite calcular los picos relativos en la pronunciación, que posean una duración mínima a fin de filtrar cualquier

posible sonido no vocálico, y asumirlos como fragmentos netamente vocálicos.

3.1.4- Características de los sonidos vocálicos básicos.

Las diferentes vocales constituyen en el castellano los sonidos vocálicos básicos. Algo que parece tan sencillo como distinguir una **A** de una **E** puede llegar a ser sumamente complicado para un programa de computación, debido a que el proceso que el cerebro usa para distinguir los sonidos más simples todavía es una incógnita. La mejor prueba de esto son las onomatopeyas ¿Cómo explicar que el ser humano asigne una "pronunciación" onomatopéyica a ruidos como el graznido de un ave o el golpe de una mano en una puerta? Hay poca discusión que los ovejas no hablan, pero tampoco se discute que sus balidos parecen un "cuac". La identificación de un sonido con un conjunto de fonemas no necesita que haya contenidos semánticos involucrados, sino que radica en características del sonido mismo.

Como se ha mencionado anteriormente, esas características parecen ser las amplitudes de las frecuencias formantes del sonido. Estas se pueden hallar con la transformada rápida de Fourier. Para poder enfocar el análisis del sonido vocálico, debemos convenir en usar sólo algunas de sus características para realizar la comparación, y esta convención debe tener algún sustento experimental.

Según se ha mencionado anteriormente, en la sección 2.4, las diferencias entre las voces proceden tanto de la frecuencia fundamental como del espectro que cada voz produce. La diferencia en tono (frecuencia fundamental) permite agrupar las voces en forma general como graves (masculinas) y agudas (femeninas), pero dentro de un mismo tono podemos distinguir gran cantidad de voces por el timbre, propiedad relacionada al espectro de frecuencias. Esta afirmación se sustenta en la experimentación realizada con diferentes instrumentos musicales tocando una misma nota. En el gráfico siguiente, se pueden apreciar los espectros de la misma nota tocada por un piano y un violín:

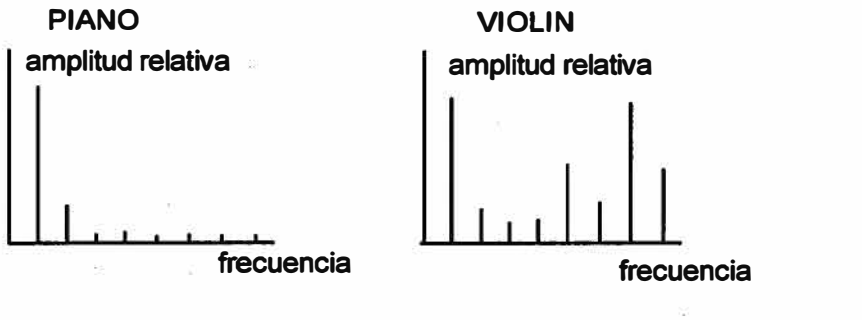


Figura 6: Espectros de un piano y un violín tocando una misma nota

La diferencia en timbre también ha sido relacionada con el espectro para el caso de las voces humanas. Así como las diferencias de timbre son diferencias de espectro para una misma nota (frecuencia fundamental), la diferencia entre fonemas está más asociada al espectro que al tono.

Si bien es cierto que en la conversación coloquial existe mucho contenido

semántico en la entonación de las frases, también es cierto que el cerebro humano asigna significado aislado a las palabras y hasta a los fonemas, aunque sólo sea el significado de su identificación o diferenciación respecto a otras palabras o sonidos. Al nivel de palabras, la entonación juega un papel importante todavía, pero a nivel silábico o fonético esto no es aplicable.

El estudio del espectro de las vocales en castellano no se ha podido encontrar en ningún libro o referencia bibliográfica. Por ello, en esta tesis se ha procedido a analizarlo de primera mano, mediante recolección de muestras por cada vocal. A continuación pasamos a detallar las características del espectro de cada vocal, incidiendo en las particularidades que las diferencian unas de otras. En los gráficos que se incluyen, la escala de la amplitud es relativa, por que sólo es importante tener en cuenta las proporciones entre las amplitudes, y la escala de la frecuencia viene dada por la cantidad de muestras que se usan para la transformada rápida de Fourier (en este caso 512 muestras = período = 0.0232 segundos, por lo tanto la unidad de frecuencia en el gráfico es su inversa = 43.066 Hz). Además , cuando se hace referencia a una frecuencia formante en determinada posición, en la práctica (y en el sistema desarrollado también) no se trata de la posición exacta o permanente, pues en cada pronunciación puede diferir hacia una frecuencia superior o inferior.

Vocal A .- las principales características de la A son la presencia de las formantes aproximadamente en las frecuencias 3, 6, 9, 11 ó 12 y 14. Las

relaciones entre las amplitudes de estas formantes son las siguientes:

$F_3/F_6 = 0.58$ $F_6/F_9 = 1.66$

$F_9/F_{11\ ó\ 12} = 1.24$ $F_{11\ ó\ 12}/F_{14} = 0.7$

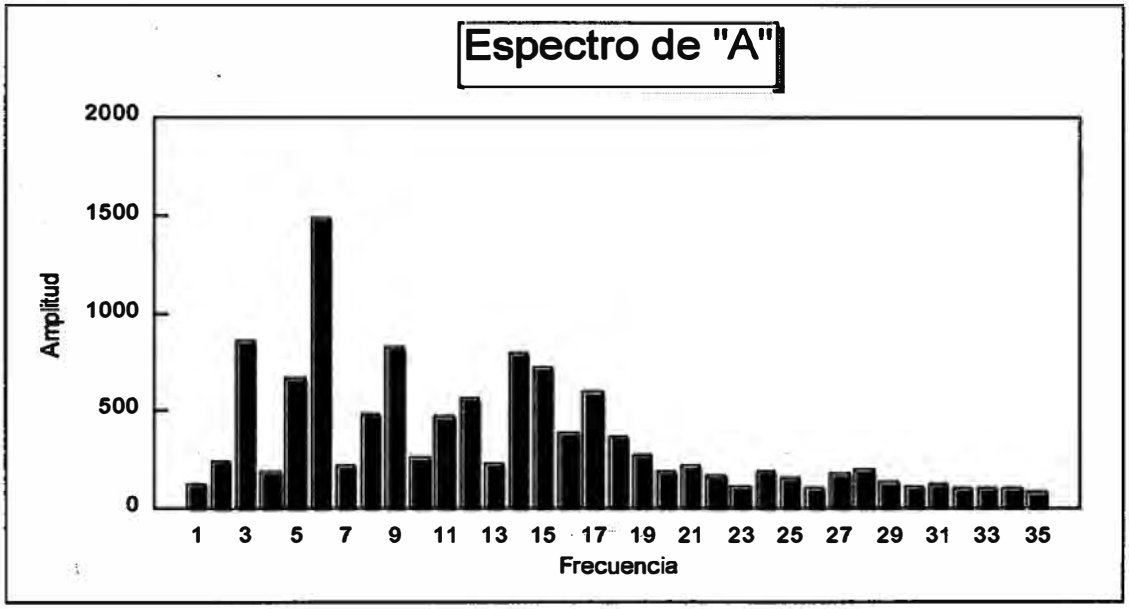


Figura 7: Espectro para la vocal "A"

Además, una característica particular de la "A", que permite diferenciarla bastante bien del resto de vocales es que tiene una pequeña mini-formante cerca a la frecuencia 28 ó 29, que es apreciable en su caso pero que es totalmente despreciable en las demás vocales.

Vocal E .- las principales características de la E son la presencia de las formantes aproximadamente en las frecuencias 3, 6 y 9. Las relaciones entre las amplitudes de estas formantes son las siguientes:

$F_3/F_6 = 0.57$ $F_6/F_9 = 0.67$

$F_3/F_9 = 0.4$

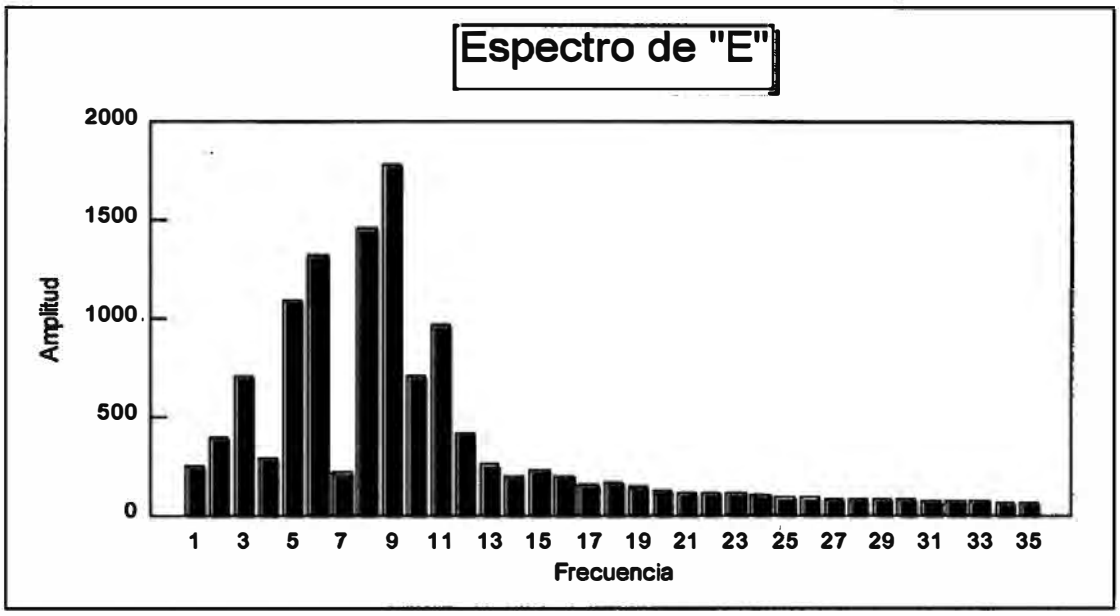


Figura 8: Espectro para la vocal "E"

La "E" tiene mucho parecido con la "O", a nivel de espectro, así que se ha desarrollado un criterio adicional para ayudar a que el sistema diferencia ambas vocales. Se toman el formante en la frecuencia 3 (en una de las frecuencias contiguas) y una posible formante en la frecuencia 11 ó 12. Si la mayor formante del sonido es muy cercana en amplitud a la formante de la frecuencia 11 ó 12 (90% o más) y si además la amplitud de la formante de la frecuencia 3 es relativamente pequeña (30% ó menos de la mayor formante), se trata de una "O"; si sólo una de las dos condiciones se cumple, persiste la duda, pero si ninguna se cumple se trata de una "E".

Vocal I .- las principales características de la I son la presencia de las formantes aproximadamente en las frecuencias 3, 6 y 9. Las relaciones entre las amplitudes de estas formantes son las siguientes:

$F_3/F_6 = 0.41$ $\qquad$ $F_6/F_9 = 2.24$

$F_3/F_9 = 0.85$

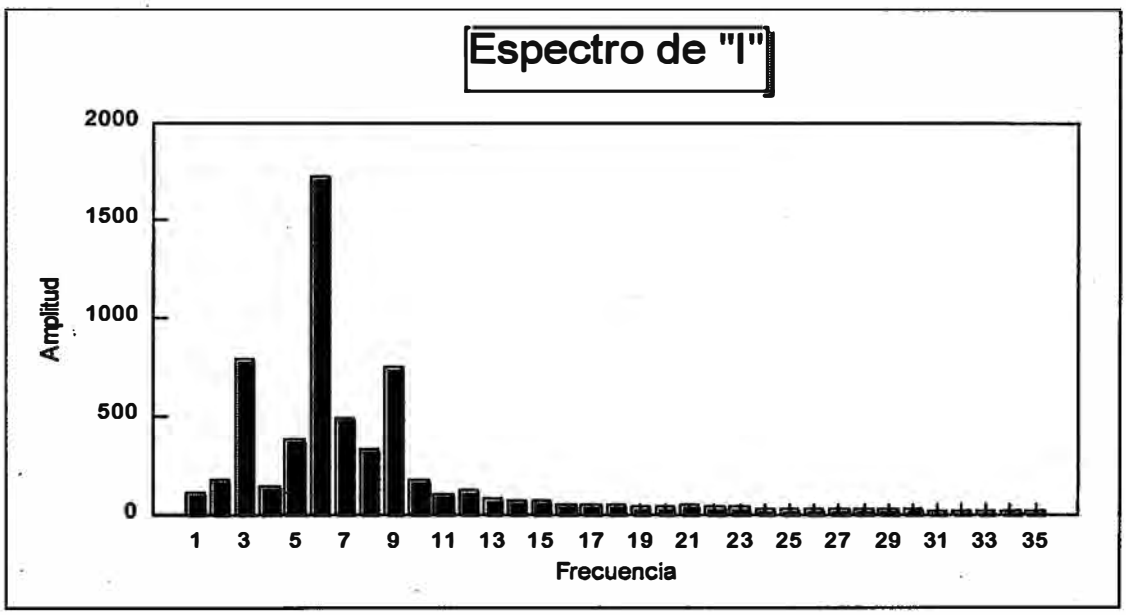


Figura 9: Espectro para la vocal "I"

La vocal "I" se parece mucho a la "U", pero no se ha podido encontrar un criterio de refinamiento del reconocimiento. Tomando en cuenta esa característica, se está dando menos peso a las diferencia entre la "I" y la "U" en el momento de comparar las palabras.

Vocal O .- las principales características de la O son la presencia de las

formantes aproximadamente en las frecuencias 5 ó 6, 8 ó 9 y 11. Las relaciones entre las amplitudes de estas formantes son las siguientes:

$F_{5\acute{o}6}/F_{8\acute{o}9} = 0.98$ $F_{8\acute{o}9}/F_{11} = 0.4$ $F_{5\acute{o}6}/F_{11} = 0.4$

Se comentó anteriormente el parecido entre los espectros de "E" y "O", para lo cuál se desarrolló un criterio de diferenciación, detallado en la sección correspondiente a "E". Obsérvese su espectro en la figura 10.

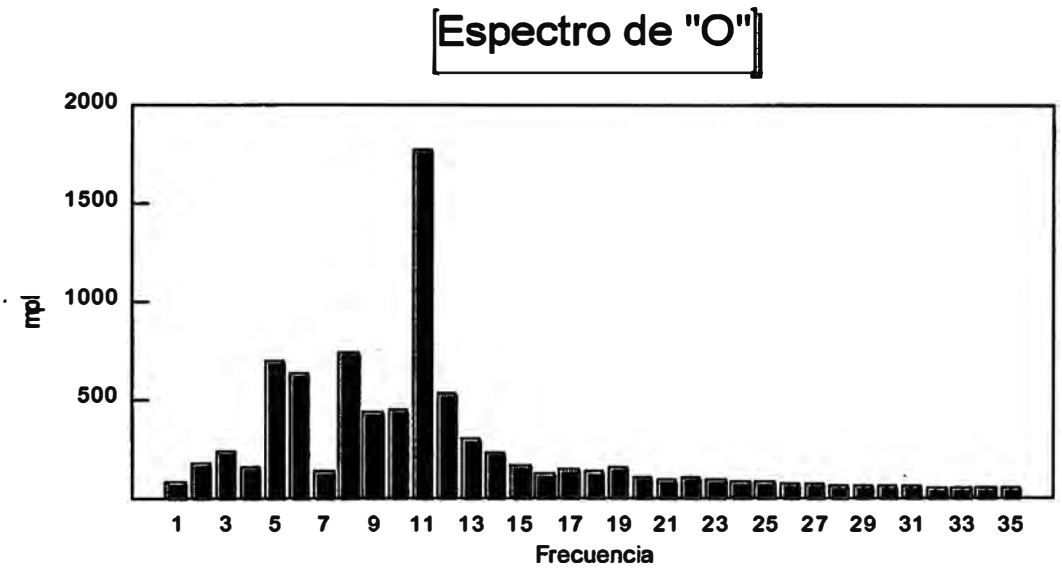


Figura 10: Espectro para la vocal "O"

Vocal U .- las principales características de la U son la presencia de las formantes aproximadamente en las frecuencias 3, 6 y 9. Las relaciones entre las amplitudes de estas formantes son las siguientes:

$F_3/F_6 = 0.3$ $F_6/F_9 = 2.16$ $F_3/F_9 = 0.62$

Obsérvese su espectro en la figura 11.

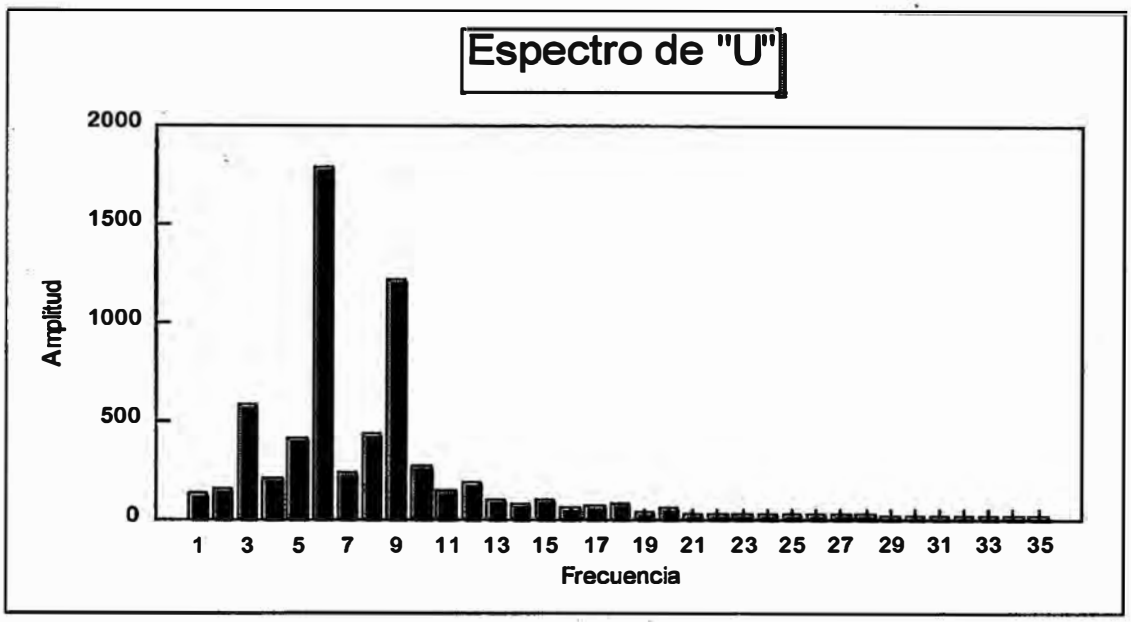


Figura 11: Espectro para la vocal "U"

3.2- El Modelo de una palabra

Para crear un modelo de una palabra se usan varios ejemplos de dicha palabra, es decir se graban varias veces la misma palabra pronunciada por el usuario, y se trata de guardar la información ,tanto la común a todos los ejemplos como las variaciones individuales.

Los factores que se deben considerar en la formulación del modelo son similares a los que se deben usar en la separación de la información relevante:

- Condensar la información

- No omitir ni alterar los datos importantes
- Algoritmo rápido

Por ello, las alternativas más practicas son usar modelos matriciales o modelos reticulares. Los modelos matriciales son fáciles y rápidos de implementar, pero no condensan bien la información y dificultan que el presente trabajo se pueda extender hacia el reconocimiento independiente del hablante (speaker independence) o hacia el reconocimiento de diccionarios amplios (para dictado electrónico). Los modelos reticulares pueden condensar e interrelacionar mejor la información, pero con mucho proceso de cómputo y con el riesgo de alterar algunos datos importantes, por lo que pueden haber algunos resultados inesperados.

Se ha escogido la alternativa de modelo matricial. En general tiene muchas ventajas, pero para subsanar las deficiencias que tiene, especialmente la necesidad de interrelacionar la información de los segmentos de audio, se implementará un mecanismo correctivo, en este caso un Análisis digital de ondas, tal como se mostró en una sección anterior y como se ampliará posteriormente. La matriz principal nos permitirá comparar en forma directa las similitudes y diferencias entre una palabra pronunciada y los modelos archivados; pero el análisis de fragmentos de sonido en sus ondas componentes nos permitirá reconocer ciertos fonemas en forma muy segura e independiente de su contexto (la palabra), permitiendo que el presente estudio sea extensible en el futuro a

reconocimiento de diccionarios grandes o ilimitados. En este caso, los sonidos que se van a identificar son los sonidos vocálicos básicos (a,e,i,o,u) sin importar que se presenten aislados o relacionados con otros sonidos, lo importante es que el sonido vocálico tenga la duración suficiente para seleccionar un fragmento de unos 20 ó 25 ms en que no sufra influencias de otros sonidos.

Se puede esquematizar el modelo empleado como un modelo de capas o niveles, que se adicionan o condicionan mutuamente:

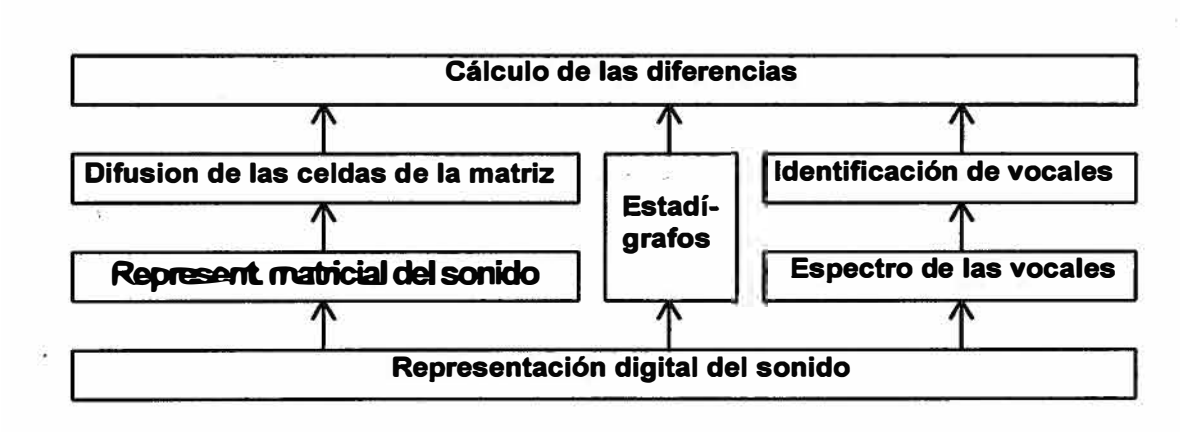


Diagrama III: Estructura jerárquica de los modelos y métodos usados

El condicionamiento que se ejerce entre la comparación de longitudes de onda y de amplitudes es la más importante que se ha considerado. Más adelante se reseña este hecho y la fórmula que se usa, pero en líneas generales, la idea subyacente es que si dos factores son importantes en una evaluación, es preferible escoger una alternativa en que ambos factores tengan una buena coincidencia a otra alternativa en la que un factor es casi perfectamente coincidente y el otro es poco coincidente.

Las estructuras de datos que van a guardar la información de una palabra se dividen en dos: las estructuras relacionadas al modelo matricial, y las relacionadas al espectro de ondas. Esta estructura de datos es la siguiente:

Estructura ConceptoPalabra

float longprompalabra	(longitud promedio de la palabra)
float desvstanpalabra	(desviación estándar de la longitud)
char numrepetpalabra	(número de repeticiones en el entrenamiento)
float amplmaxprompal	(promedio de la amplitud máxima)
unsigned int amplmaxbase	(amplitud máxima usada como escala)
unsigned int longbasepalab	(longitud usada como escala)
float modelopalabra[200][64]	(matriz de la amplitud)
float modlongonda[200][17]	(matriz de la longitud de onda)
float vocales_m[5]	(matriz de la proporción de cada vocal)

La estructura de los datos de trabajo, es decir para pasar del sonido digitalizado a un arreglo de máximos y mínimos es:

Matriz matriz_sonido[5000] de tipo data_sound;

Estructura data_sound

unsigned long sample	(posición del máximo o mínimo)
unsigned char amplitud	(amplitud del máximo o mínimo)

unsigned int long_onda (longitud de onda del máximo o mínimo)

3.3- Algoritmo para el entrenamiento inicial

Para entrenar al sistema en el reconocimiento de una palabra, lo primero que se debe hacer es escoger al usuario, es decir la persona a quien se va a asignar esa palabra, con su significado o acción asociada y su modelo de palabra. Una vez escogido un usuario, o definido si antes no existía, se define una nueva palabra a su diccionario, y se empieza a entrenar al sistema, repitiendo n veces dicha palabra.

Lo que el sistema procesa cada vez que se graba un sonido de entrenamiento es similar a lo que procesaría en el caso que reconozca una palabra. Este hecho debe ser invariable, sin importar que método de comparación, de evaluación, estructura de modelo, etc. se este utilizando, por que el proceso de evaluación debe comparar la salida de un proceso para la palabra a reconocer contra la salida de un proceso de entrenamiento. Si ambos procesos son distintos en esencia no habría comparación posible (por ejemplo, el sonido a reconocer se convierta en una red neural y el modelo de palabra sea el resultado de aplicar transformadas de Fourier).

El algoritmo se divide en 2 fases bien diferenciadas:

- La captura, obtención de los máximos y mínimos y transformación a una matriz.

- La obtención de los estadígrafos fundamentales.

3.3.1- La captura, obtención de los máximos y mínimos y transformación a una matriz.

El algoritmo usado es el siguiente:

1- Se inicia la captura del sonido

2- Se dejan pasar 2.5 segundos

3- Se finaliza la captura del sonido y se graba en un archivo.

4- Se obtiene los máximos y mínimos (según el algoritmo de la sección 3.1.1)

5- Cada dato considerado como máximo o mínimo incremente en 1 a una celda de la matriz de amplitudes del modelo, acorde a su ubicación temporal y a su valor.

6- Se calcula la diferencia en el tiempo entre un máximo o mínimo y su anterior. A esta diferencia se la considera, para propósitos de trabajo, una media longitud de onda.

7- Se graba esta diferencia incrementando en uno una celda de la matriz de longitudes de onda, acorde a su ubicación temporal y a su valor.

8- Se regresa a (5) hasta llegar al último máximo-mínimo.

9- Se regresa a (1) hasta llegar a la última ocurrencia del entrenamiento de la

palabra.

Por cada uno de los entrenamientos se consigue una matriz, pero en realidad es conveniente realizar 4 entrenamientos como mínimo por cada palabra. Estos diferentes entrenamientos por una misma palabra deben ser unidos entre sí, de tal forma que la información que proporcionan contribuya al modelo de la palabra en cuestión, cada uno por igual. El algoritmo que se usa para este propósito es sencillo, y se realiza como sigue:

1- Se homogeniza la duración del sonido a la escala común.
2- Se suma la matriz de amplitud de este entrenamiento a la matriz de amplitud del modelo de palabra.
3- Se suma la matriz de longitud de onda de este entrenamiento a la matriz de longitud de onda del modelo de palabra.
4- Se obtiene el siguiente sonido, si lo hay, y se procesa según el algoritmo anterior a este.
5- Se va a (1).

3.3.2- La obtención de los estadígrafos fundamentales.

Una vez creadas las matrices de amplitudes y de longitudes de onda del modelo de palabra, se procede a calcular los estadígrafos más importantes, que son media aritmética y la desviación estándar del tiempo de duración de la palabra, así como la escala temporal y de amplitud en que se ha grabado las

matrices. Lo ideal sería que las escalas sean el promedio, o la mayor de entre las ocurrencias, pero por eficiencia computacional, se ha utilizado el valor del primer entrenamiento.

El objetivo de los promedios y desviaciones estándar es tener un filtro para palabras muy largas o muy cortas con respecto a la palabra pronunciada. Por ejemplo, si se pronuncia la palabra "codifica" y en nuestro diccionario existen palabras tan cotas como "uno" ó "más", estas últimas obviamente deben ser descartadas sólo con la evaluación de las longitudes de las palabras. Mientras que el objetivo de las escalas es poder comparar palabras en base a una misma escala temporal, y agrupar los sonidos obtenidos en el entrenamiento inicial de un sonido efectuando las operaciones respectivas en la misma escala temporal. Supóngase que se entrena la palabra "divide" con 4 repeticiones, y se obtiene sonidos de longitudes 0.7, 0.75, 0.72 y 0.81 segundos; estos diferentes sonidos al agruparse se deben homogenizar en el tiempo, mediante el uso de una escala para cada uno de ellos.

3.3.3- La obtención de las vocales de la palabra entrenada.

Las vocales de la palabra entrenada se obtiene de la siguiente forma:

1- Se ubica el primer segmento vocálico según el algoritmo de la sección 3.1.3.
2- Se aplica en este segmento la Transforma Rápida de Fourier , según el

algoritmo que figura en el Anexo D.

3- Se obtiene el módulo o amplitud de cada una de las primeras 50 senoidales componentes (que son las relevantes en el estudio).
4- Dichos módulos se comparan con los criterios de evaluación mencionados en el punto 3.1.4.
5- Se asigna a este segmento vocálico la vocal que más coincide (menor valor de diferencia).
6- Se actualiza la relación de vocales, añadiendo la última hallada.
7- Se ubica el siguiente segmento vocálico, y si existe se va a (2).

Como se puede apreciar, este proceso es para un solo entrenamiento. Para poder usar las vocales de los diferentes entrenamientos, consolidándolas en una sola información del modelo de palabra, se debe efectuar un proceso inteligente de ponderación. Este proceso de ponderación de los diferentes conjuntos de vocales se ha implementado de una manera simple, pero que es suficientemente adecuada para el reconocimiento de palabras de un diccionario pequeño o mediano (máximo de unas 200 palabras). El método consiste en ponderar las vocales por la frecuencia en que se repiten en cada entrenamiento, representar esa frecuencia como porcentajes, y luego ponderar dichos porcentajes.

La principal desventaja de este método es que no toma en cuenta las ubicaciones de las vocales. Una mejora importante al actual sistema sería obtener como resultado del entrenamiento un vector con la lista de las vocales

encontradas, en el orden en que se pronuncian en la palabra. No se ha podido realizar esto principalmente por una razón: el porcentaje de errores en la identificación de las vocales es lo suficientemente alto cómo para confundir al sistema en la etapa de entrenamiento.

En efecto, si se entrenase la palabra "derecho" se podrían tener como resultados las series de vocales (e,e,o), (e,o,o), (e,?,o), (e,e,e), donde el signo "?" significa que no se puede identificar claramente que vocal es e incluso no se puede estar seguro que este segmento de sonido sea una vocal. En este caso, no se puede determinar que vocal es la segunda, por que tenemos que para cada caso la segunda vocal es: e, o, ? **or** o, e. En el tercer entrenamiento puede que la segunda vocal no sea distinguible, o puede ser que la segunda vocal en realidad no es vocal, y por lo tanto su lugar lo ocupe la o; el hecho es que no es sencillo determinar cuál es la segunda vocal. Pero si se puede determinar que la primera vocal es E y que se puede presumir con mucha certeza que la tercera es O.

Esta es la razón por la que no es sencillo determinar la secuencia de las vocales con una alta precisión. Si se hicieran mayores análisis de ondas en segmentos consecutivos a lo largo de las supuestas vocales, la precisión aumentaría lo suficiente para que la decisión se hiciese más sencilla y confiable, pero ello depende de procesadores más poderosos que los actuales, por que el análisis DSP gasta mucho tiempo de CPU.

3.4- Algoritmo de reconocimiento.

El algoritmo que se usa para el reconocimiento de una palabra consiste en 4 fases bien diferenciadas:

- La captura, obtención de los máximos y mínimos y transformación a una matriz.
- El realineamiento temporal y de amplitud (escalar la duración y la amplitud).
- La comparación de la matriz de un modelo de palabra con la matriz del sonido a comparar.
- La comparación de las vocales halladas por sus espectros con las vocales del modelo de palabra.

3.4.1- La captura, obtención de los máximos y mínimos y transformación a una matriz.

Respecto a la primera fase, no hay mucho que decir, por que es similar a la primera fase del entrenamiento de una nueva palabra. La captura se efectúa usando los servicios MCI (Multimedia Control Interface) de Windows, por un tiempo predeterminado. Luego se localiza el inicio y el fin de la palabra, y los máximos y mínimos.

En cuanto a la transformación a una matriz, debemos tener en cuenta que no tiene otro objeto que poder comparar una palabra entrenada con la palabra en cuestión, pero con una gran diferencia, que se explica en la siguiente

sección. En esencia, no hay diferencias sustanciales con los procedimientos de las palabras entrenadas.

El funcionamiento de los dispositivos MCI, la forma de usarlos bajo los sistemas operativos de Microsoft y los APIs existentes se explican con más detalle en el anexo E.

3.4.2- El realineamiento temporal y de amplitud. Escala de la duración y la amplitud.

La diferencia fundamental de esta parte del algoritmo es que no se debe homogenizar una sola vez, sino que se procede a pasar la palabra en cuestión a la misma escala temporal de cada una de las palabras entrenadas. Por tanto, este re-scaling se efectúa n veces con diferentes escalas, tanto en duración como en amplitud.

Para evitar que palabras, a todas luces diferentes por sus longitudes, pudiesen ser re-escaladas y por efectos del azar tuvieran baja diferencia, se procedió a introducir unos límites al re-scaling. Este problema se presentó en el momento de las pruebas, por lo que se probaron diferentes límites, de tal manera que una palabra tuviera opción de re-escalarse con su equivalente, pero no con las palabras con las que tenía problemas. Estos límites fueron de 30% hacia arriba y hacia abajo

3.4.3- La comparación de la matriz de un modelo de palabra con la matriz del sonido a comparar.

La matriz que resume la palabra entrenada y la que representa la palabra a probar se comparan elemento por elemento, pero con un artificio: no se comparan los valores exactos de los elementos, sino los valores difusos de ellos. Un valor difuso en una matriz es un valor que depende de la misma celda principalmente, pero que está influido por los valores de las celdas contiguas. Si se tiene la siguiente celda en la siguiente matriz:

5	3	4
6	**8**	1
9	2	0

El valor de la celda central es 8, pero su valor difuso es una función de 8 y de los valores de las celdas contiguas. Para la presente tesis se ha usado un fórmula aditiva con distintos pesos:

Valor difuso = (8 * Valor celda + (Suma valores contiguos))/16

Esta difusión de los valores de las celdas nos permite una compartación más flexible de las matrices. Es una técnica sumamente usada en el procesamiento digital de imágenes, para modificar los colores, el brillo, el contraste, etc. pero con aplicaciones en todo problema que involucre información que no tiene una posición fija o predeterminada en una matriz sino

un rango más o menos amplio de posibles ubicaciones. Por ejemplo, supóngase que en la celda (18,13) se ubica un valor sumamente alto de puntos máximos y mínimos, porque es el mayor máximo de la palabra (25 ocurrencias), pero en la palabra entrenada correspondiente, este valor se ha ubicado más frecuentemente en la celda (18,12) con 23 ocurrencias, y la celda (18,13) contiene cero ocurrencias; una comparación entre estos valores nos llevaría a una gran diferencia ,es decir a poca similitud entre las palabras. En cambio, con el uso de la difusión de los valores, la suma de las diferencias es menor porque ese valor de 25 ocurrencias influye en otras 8 celdas, de las cuáles 5 son comunes con las 8 que son influidas por el valor de 23 ocurrencias.

En el caso que la celda en cuestión no estuviese rodeada totalmente por celdas sino que se hallase en un borde de la matriz, se procedería de la siguiente manera:

12	10	8
5	**9**	6

Caso 1 (borde)

Valor difuso = (10 * Valor celda + (Suma valores contiguos))/15

10	4
8	6

Caso 2 (esquina)

Valor difuso = (6 * Valor celda + (Suma valores contiguos))/9

Se le asigna un mayor peso relativo a la celda original debido a que existen menos celdas contiguas que influyen en ella.

La comparación de las matrices de longitud de onda y de amplitud se efectúa sumando las diferencias absolutas entre los valores difusos de cada celda de la matriz modelo con su correspondiente celda de la matriz a evaluar. Una vez sumadas las discrepancias, se obtienen dos valores, que se llaman P1 y P2 (parecido 1 y parecido 2), en las que a mayor valor, menor parecido. Como ambos son valores que han sido obtenidos de análisis semejantes en el sonido, se condicionan uno al otro, debido a que sus valores no son totalmente independientes. Se necesita una función que sea aditiva (a mayor valor de cualquiera P1 ó P2, mayor valor del resultado final), pero que permita que entre dos juegos de valores P1,P2 que sumen igual, se prefiera el que sea más equilibrado y se rechaze el que tiene uno de los valores muy grande (muy diferente). La fórmula sería así:

$$\text{Parecido} = (\text{factor1} * P1) + (\text{factor2} * P2) + (\text{factor1} + \text{factor2}) * \sqrt{P1 * P2}\ /4$$

Se ha considerado un factor de evaluación auxiliar, que es importante sobre todo en las palabras

3.4.4- <u>La comparación de las vocales halladas por sus espectros con las vocales del modelo de palabra.</u>

Las vocales se han comparado por el porcentaje en que cada una de ellas aparecía en la palabra. Este método obviamente no es el mejor, cómo ya se afirmó anteriormente, debido a que no toma en cuenta la posición de las variables halladas, pero se ha realizado de esta manera por que la identificación de las vocales no es perfecta, y existen hasta 9 posibles problemas de posición de las vocales:

1. El algoritmo omitió una vocal en la palabra entrenada.
2. El algoritmo omitió una vocal en la palabra bajo prueba.
3. El algoritmo omitió distintas vocales en ambas palabras.
4. El algoritmo identificó mal una vocal en la palabra entrenada.
5. El algoritmo identificó mal una vocal en la palabra bajo prueba.
6. El algoritmo identificó mal diferentes vocales en ambas palabras.
7. El algoritmo añadió una vocal inexistente en la palabra entrenada.
8. El algoritmo añadió una vocal inexistente en la palabra bajo prueba.
9. El algoritmo añadió vocales inexistentes en ambas palabras.

Mediante los experimentos que se condujeron se pude comprobar que por lo menos uno de estos problemas se presentan en poco más del 40% de los casos. Puede parecer una cifra excesiva de errores, pero se debe de tomar en consideración que se trata de un error entre 3 ó 4 vocales, lo que en el peor de los casos significa una diferencia de 33%. Pero como este método se pondera junto a la comparación de las matrices, las desviaciones de uno son cubiertas

por los aciertos del otro.

Mediante un examen de frecuencias de las vocales en las palabras, realizado por cripto analistas en sus estudios para descifrar códigos ,se ha podido determinar que en el inglés la frecuencia de las vocales es en el siguiente orden: E, O, A, I, U. En castellano el orden es: E, A, O, I, U. La proporción de veces que se usa la E contra las que se usa la I es mayor que 2 a 1 en ambos idiomas.

La identificación en dos distintos segmentos de una misma vocal nos permite tener una probabilidad de 88.8% que por lo menos una vez se capturará la vocal correcta. Si el segmento vocálico es muy extenso (vocal importante en la palabra), puede tomarse tres muestras, lo que nos garantiza por lo menos un acierto el 96.3% y por lo menos dos aciertos el 74 % de las veces. Estas cifras proporcionan suficiente confianza como para intentar el uso de este método.

Todo lo tratado en este capítulo se puede resumir en los diagramas que se pueden apreciar en la siguiente página.

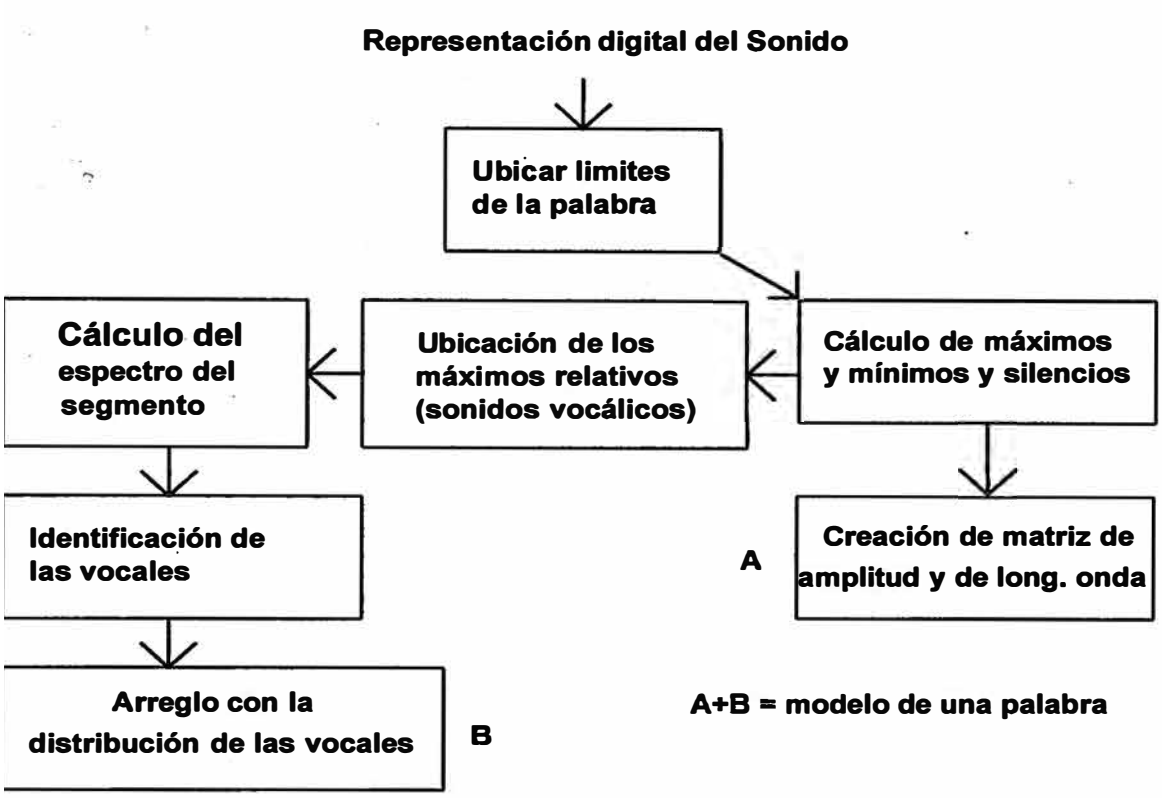


Diagrama IV: Partes del método antes de la comparación.

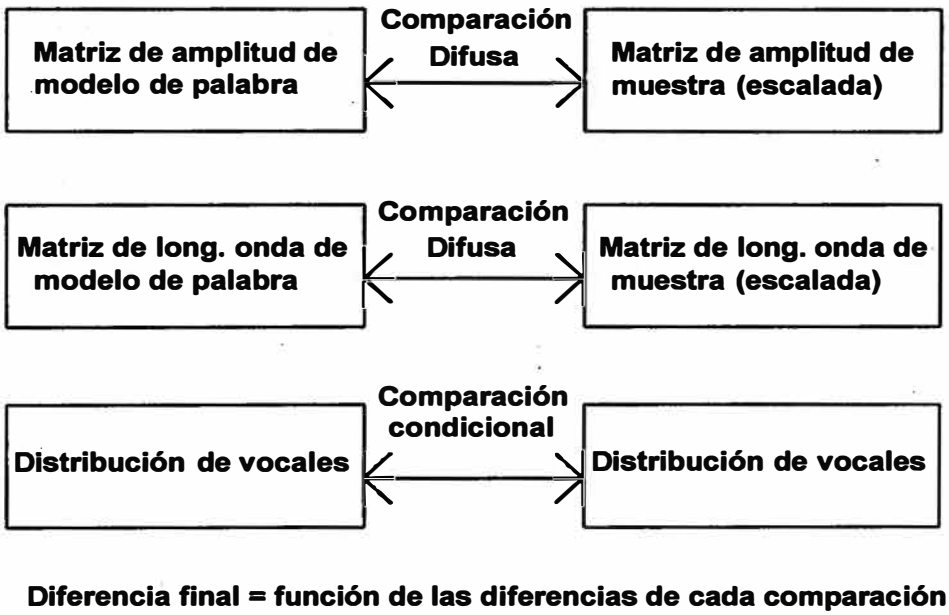


Diagrama V: La comparación entre un modelo de palabra y una muestra.

4. EVALUACION DE LOS RESULTADOS

4.1- Evaluación de la conveniencia del modelo usado.

Primero es conveniente definir que criterios se van a emplear para evaluar la conveniencia del modelo. Para esto debemos tomar en cuenta tanto a los demás modelos que se emplean en la actualidad como a las posibilidades y limitaciones de cada uno de ellos. Es por ello que se propone usar los siguientes factores:

- La flexibilidad del modelo, definida como la posibilidad de extender el uso del modelo a diferentes variaciones del reconocimiento de voz, es decir, que permite extender el reconocimiento al modo independiente del hablante o al del habla continua.
- La cantidad de recursos computacionales (memoria y tiempo de CPU), como medida de lo práctico que resulta el modelo para ser implementado en una computadora personal.
- La simplicidad del modelo, que permite una fácil implementación y la adición de elementos adicionales sin replantear el modelo.
- La convergencia inherente al modelo, es decir la posibilidad que se refine la

búsqueda si se dispone de mayor velocidad de proceso o más tiempo para procesar. Esto es, si el modelo permite que a mayor número de iteraciones o pasos, mayor exactitud, y si es así, con que convergencia.

- Función O(N) de la búsqueda, es decir si el aumento del tiempo de búsqueda de la palabra al aumentar el diccionario es lineal, cuadrática, exponencial, logarítmica, etc.

Como se puede observar, hemos separado claramente lo que es la evaluación del modelo en si, como una construcción abstracta, de su implementación específica en un momento dado. Esta implementación se va a evaluar en el punto 4.4, porque depende de lo bien que se haya aproximado los distintos parámetros (silencios, relaciones en los espectros, pesos de los diferentes criterios), donde esta aproximación es un proceso experimental de prueba y error que no invalida el modelo mientras que no se haya realizado el número suficiente de pruebas y que todas arrojen resultados insatisfactorios.

Se debe insistir en este detalle, por que la conveniencia de un modelo en la representación del conocimiento no se invalida por lo mal que se lleve a cabo esta representación la primera vez, sino por que una serie de diferentes intentos demuestren que el problema está en el modelo. Por ejemplo, el hecho de que el modelo de Orientación a Objetos hasta el momento no ha logrado ser implementado satisfactoriamente y no haya proporcionado las todas ventajas que prometía no significa que debe ser descartado, sino que debe esperase mayor

número de intentos hasta que consiga una implementación definitiva y exitosa, o en caso contrario se pueda llegar a la conclusión que el modelo se debe rechazar. Además, el modelo Orientado a Objetos ha tenido varios éxitos parciales, es decir implementaciones que han cumplido con algunas de las promesas que el modelo realizó.

En el caso del modelo propuesto para el reconocimiento de voz, en sí es la yuxtaposición de dos modelos más simples, uno el matricial y otro el del análisis de ondas, que descansan ambos en un principio de reducción de la data relevante (al que se ha bautizado como método de máximos y mínimos). La evaluación se realizó tomando en cuenta las bondades y debilidades del modelo en sí, como de sus sub-modelos conformantes y comparándolas un poco con los modelos de los cuáles se tiene algún conocimiento:

4.1.1- Respecto a la flexibilidad del modelo:

El sub modelo de la matriz de frecuencias y longitudes de onda puede extenderse al reconocimiento de voz para cualquier hablante, porque en sí la labor del reconocimiento no radica en el modelo sino en los algoritmos, pero en cambio sería muy difícil extender el sub modelo al reconocimiento continuo, por que uno de los principales supuestos en que se basa es que se debe identificar con bastante exactitud el inicio y el fin de las palabras, lo cuál es bastante difícil en el castellano.

Como bien expresan los lingüistas norteamericanos Castillo y Bond, una de las principales características que distinguen al castellano del inglés es la continuidad de la transición entre palabra y palabra dentro de un "grupo respiratorio", o sea dentro de la misma exhalación de aire. La parada gutural entre palabra y palabra es muy rara en castellano, pero muy frecuente en las lenguas germánicas. En nuestro idioma, la transición entre vocales de palabras contiguas es suave. "El hado", "helado" y " el lado" suenan muy parecido. "Para hacerlo" suena [paraaserlo] en lectura y [paraserlo] en conversación rápida.

Todo esto significa que el reconocimiento por palabra no puede efectuarse tal como se realiza ahora, sino que debe convertirse en un reconocimiento de sílaba por sílaba, y el algoritmo de reconocimiento de sílabas debe pasar las que reconoce a un motor que trate de armar las palabras según las sílabas que le van llegando. Esto requiere un algoritmo muy especial, como el de las cadenas ocultas de Markov, que trabaja con las probabilidades que para una serie de sílabas corresponda cierta palabra o palabras.

En cambio, el submodelo de análisis de ondas si puede ser adaptado tanto a múltiples usuarios como al reconocimiento continuo. Para ser adoptado al reconocimiento independiente del hablante, se debe implementar un reconocimiento de vocales que reconozca mejor los

picos y sus formantes. El reconocimiento por fonemas y sílabas puede ser efectuado sin mayor dificultad usando el análisis de ondas, pero lo principal es tener un algoritmo bastante complejo que permita reconocer las diferentes variante que la entonación de las frases imprime en el habla continua.

Respecto al tamaño del diccionario de palabras, cabe recordar que todos los sistemas de reconocimiento de voz existentes, así como los propuestos, usan diccionarios de palabras, por que no se puede transcribir una palabra hablada sólo por el reconocimiento de sus sonidos más elementales, ya que no se podría saber cuando se ha interpretado esas vocales correctamente. Por ejemplo, las sílabas sa-co-pu formarían una hipotética palabra "sacopu", pero el sistema necesitaría saber que esa palabra no existe (no pertenece al dominio de su problema) por lo que no puede considerarla como tal, sino como un error del usuario o error de interpretación.

4.1.2- <u>Respecto a los recursos computacionales consumidos:</u>

Una de las principales cualidades que debe poseer un modelo de representación y esquematización de datos para el reconocimiento de voz es la eficiencia. Es cierto que la eficiencia se encuentra en la implementación misma, pero también es cierto que dados dos modelos diferentes de representar datos, uno de ellos siempre se podrá

implementar más eficientemente que otro, si los sistemas se desarrollan con una calidad similar.

Por ejemplo, para poder implementar una hoja de cálculo se pueden usar varios modelos para representar una hoja y sus celdas. Un modelo puede ser de una sola gran matriz que representa a toda la hoja (la hoja de cálculo ocupa el mismo gran espacio de memoria este vacía o llena), mientras que otro puede ser de un arreglo o una cadena de registros y punteros donde están solamente las celdas con algún dato o formato (es decir, una hoja de cálculo nueva no ocupa memoria, y esta memoria se va usando conforme la cantidad de datos). Cada modelo tiene ventajas y desventajas con respecto al uso de la memoria y al tiempo de proceso necesario para efectuar sus funciones básicas.

En el caso del sub modelo matricial, este es medianamente eficiente en la utilización de la memoria, por que sólo llega a usar registros de 16200 variables float, unos 64800 bytes, por cada palabra, necesitando sólo dos de estas registros para que uso del algoritmo. De las celdas tipo float, se calcula que un 40% llega a tener datos dentro de los límites de la palabra, mientras que más allá de estos límites, no hay datos. En resumen, sólo tienen datos entre 20 y 30% de la matriz, según la palabra. Lo ideal sería usar un modelo de matrices pequeñas encadenadas por punteros, pero el problema de esta solución es la

complejidad de su manejo.

- La velocidad de acceso a una matriz es sumamente rápida, debido a su propia naturaleza. En esto el sub modelo matricial que se ha usado es el mejor existente para cantidades pequeñas o medianas de datos.

El sub modelo de análisis de onda nos permite un uso ahorrativo de la memoria, debido a que sólo requiere de la presencia de 512 variables float. Sin embargo, el análisis de la transformada de Fourier implica necesariamente el algoritmo de la transformada rápida de Fourier, la cual a pesar de ser el mejor método disponible a la fecha, consume mucho tiempo de CPU. Esto nos ha llevado a restringir su aplicación a unos pocos segmentos de voz en cada palabra, sacrificando la precisión del reconocimiento.

Como se observa, se ha tratado de mantener un balance entre los sub modelos, de modo que uno es rápido en proceso pero no tan bueno en memoria, mientras que el otro ahorra memoria pero exige un poco más de proceso.

Otros métodos, como el de redes neurales, son bastante buenos tanto en consumo de memoria como de CPU, pero tienen desventajas, que se mencionaran más adelante.

4.1.3- Simplicidad del modelo:

Al estar conformado por otros dos sub modelos menores y complementarios, este modelo podría parecer complejo y difícil de modificar, sin embargo el esquema que se ha usado es tal que cada sub modelo influye en la evaluación final de manera independiente, por lo que la modificación en uno de ellos no requiere la del otro.

Cada uno de los sub modelos es simple en esencia. Muchas de las complicaciones que se pueden observar en su implementación se deben a problemas propios de la arquitectura (Windows), del hardware de la tarjeta de sonido y de la interfase gráfica usada.

En cambio, el modelo de redes neuronales es demasiado dependiente de la configuración de la red que se ha diseñado, del número de capas, de entradas y de salidas. Una modificación mediana en el diseño del modelo traería como consecuencia su replanteamiento desde el inicio.

4.1.4- Convergencia de la solución:

En este modelo, lamentablemente, la capacidad de convergencia a la solución es bastante limitada y muy lenta.

El sub modelo matricial no puede mejorar su precisión, siendo esto su

principal defecto, lo poco que se puede hacer para mejorar el resultado es aumentar el número de entrenamientos por cada palabra, y aún así con poca mejora.

El sub modelo de análisis de ondas si permite lograr una mayor precisión, permitiendo tomar más segmentos del mismo fonema para identificarlo mejor, pero su convergencia es decreciente. Si hablamos en términos estadísticos, la precisión se mide por la diferencia de 1 - Probabilidad de error tipo I - Probabilidad de error tipo II. Para una población determinada, mientras mayor es la muestra mayor es la precisión. Como se trata de un caso similar al contraste de la media de una población normal grande con varianza desconocida, tenemos que el estadígrafo de contraste es:

$$Z = (Media\ muestral - Media\ hipótesis) / (S/\sqrt{n})$$

Donde n es el número de elementos de la muestra y el estadígrafo Z tiene distribución normal. De la fórmula se puede deducir que la disminución de la imprecisión con respecto al aumento de la muestra varia en forma proporcional a la raíz cuadrada de las muestras de cada fonema, o en pocas palabras, para disminuir la imprecisión a la mitad, debe aumentarse las muestras de análisis de ondas en 4 veces.

4.1.5- <u>Respecto a la función O(N) de búsqueda:</u>

La función de O(N) de la búsqueda es la función del rendimiento de la eficiencia, es decir la expresión del tiempo esperado que el algoritmo debe tardar en función del número de palabras en su diccionario para dar una respuesta, siempre y cuando el algoritmo de implemente de la mejor manera que soporte el modelo.

Para este estudio, el modelo que se ha desarrollado no incluye ninguna facilidad para indexar las palabras del diccionario. Nótese que para el reconocimiento de voz, el indexar las palabras implica dos capacidades: el poder predeterminar, según el índice, que palabras tienen mayor o menor probabilidad de ser la palabra en estudio, y el realizarlo sin necesidad de pasar por todos los registros (palabras para el caso). Debido al hecho de no haberse realizado este indexamiento, no por que el modelo por sí no lo permita sino por que el autor no llegó a plantearse un esquema viable para este indexamiento, la función O(N) es proporcional a N, es decir la búsqueda en el diccionario es lineal.

Aunque la linealidad de la búsqueda no es mala de por sí (se considerara que O(N) es lenta para grados mayores que 1, como N^2), debe recordarse que para usar un vocabulario general mínimo se necesitan unas 600 palabras, y para que sea apropiado a los efectos de distado computarizado unas 3000 palabras. La eficiencia de la búsqueda no es aceptable.

El mejor esquema para una búsqueda rápida es el modelo de Cadenas ocultas de Markov, que permite esquematizar todo desde los fonemas y sílabas, para buscar palabras usando como índice a estos fonemas y sílabas. Por ejemplo, en el proceso de reconocimiento de la palabra "mesa", los índices serían "me" y "sa", con lo que la búsqueda inicial se restringe a pocas palabras con mayor probabilidad. Si alguna de ellas tiene un gran parecido, se escoje sin mayores dudas.

4.2- Evaluación de la conveniencia del algoritmo de entrenamiento.

La conveniencia del algoritmo de entrenamiento se evaluará por la cantidad de entrenamientos necesarios para obtener una representación aceptable de la palabra y por la velocidad con que el algoritmo funciona, pero desde el punto de vista del algoritmo, es decir, de la mayor o menor conveniencia comparada con otros algoritmos:

4.2.1- Cantidad de entrenamientos

Para obtener un rango aceptable para la duración del sonido, debemos usar los fundamentos de la estadística. Suponiendo que la distribución de la longitud de una palabra cumple una distribución normal, tenemos que la desviación estándar para dos muestras por entrenamiento sería $\sqrt{2}$ (*Valor máximo – Valor mínimo*)/2, lo que generalmente resulta ser alto pesar que la diferencia entre las dos muestras no siempre es alta, y

para un entrenamiento de n muestras la desviación estándar sería como máximo $\sqrt{n/(n-1)}\,(Máximo - mínimo)/2$, lo cual es menor, por que el factor inicial tiende a 1 y la diferencia entre las muestras extremas no crece demasiado en la mayoría de casos (esta aseveración está basada en los experimentos realizados con 2, 4 y 8 muestras por entrenamiento).

Se puede observar que siempre resulta recomendable un mayor número de muestras, pero esto implica tener un entrenamiento más lento. Por ello escogimos usar 4 muestras por entrenamiento, lo suficiente para no sacrificar la precisión del entrenamiento ni la velocidad del mismo.

Pero este factor, el promedio de duración de la palabra, no es el único que limita la reducción de las muestras. En la pronunciación de una palabra existe lo que se conoce como desalineación temporal, que se explica mejor con un ejemplo: al pronunciar una vez la palabra "casa", la primera vez la sílaba "ca" duró un 45% del total de tiempo y "sa" 55%, pero en otro intento ambas sílabas duran 50%, o "ca" puede descender a "40". Para alinear temporalmente las sílabas existen algoritmos de realineación temporal, que por razones de velocidad de proceso y falta de documentación no se han empleado. Para disminuir los efectos de esta desalineación, se necesitan como mínimo unas 3 muestras.

Comparémoslo con otros enfoques. La mayor parte de modelos y algoritmos conocidos necesitan por lo menos 3 muestras por entrenamiento para funcionar aceptablemente. La excepción es el algoritmo usado por Voice Assist, que se basa en la intensidad por tramos de tiempo, el cual puede funcionar aceptablemente incluso con un sólo entrenamiento, si las palabras no son muy parecidas, caso contrario también se recomienda unos 3 entrenamientos. Se puede decir, sin temor a equivocaciones, que el número de entrenamientos necesarios por palabra está dentro del estandar de los productos comerciales del reconocimiento de voz.

4.2.2- Velocidad

La principal desventaja del algoritmo en cuando a la velocidad es que el procesamiento de la palabra hablada se hace totalmente off-line, es decir se espera a que se haya terminado de realizar la grabación para empezar a realizar el proceso, el cual ni siquiera puede decidir cuando terminar de grabar, por que se usa un tiempo predeterminado y después se realiza el aislamiento de la palabra.

Casi todos los motores de reconocimiento de voz actuales usan una programación de multi threading (múltiples hilos de programación) por medio de la cuál hay un subprograma que en multitarea evalúa lo que otro subprograma va grabando y procesan a la vez que graban. Otro

programas controlan la grabación en bajo nivel, usurpando funciones del API del sistema operativo. Ninguna de estas opciones se ha podido considerar debido a falta de documentación disponible.

Sin embargo, una vez iniciado, el proceso de entrenamiento es aceptablemente rápido, debido a que el proceso inicial de hallar máximos y mínimos, tal vez el más laborioso en tiempo de CPU, ahorra bastante esfuerzo a los procesos que siguen, tanto a la comparación matricial como al análisis de onda. En pocas palabras, el algoritmo hace el trabajo pesado una sola vez.

4.3- Evaluación de la conveniencia del algoritmo de reconocimiento.

La conveniencia del algoritmo de reconocimiento se evaluará por la velocidad y la precisión del reconocimiento, pero desde el punto de vista del algoritmo, es decir, de la mayor o menor conveniencia comparada con otros algoritmos:

4.3.1- Velocidad:

La velocidad del reconocimiento tiene en general las mismas desventajas y ventajas que en el entrenamiento, con la diferencia que existe un problema adicional referente a la exploración o búsqueda en el diccionario, algo que ya se mencionó en el punto 4.1.5.

Al evaluar el parecido de cada modelo con la palabra buscada, el algoritmo procede secuencialmente, palabra por palabra, para al final escoger el valor con la diferencia más baja. Este proceso funciona bien para cantidades pequeñas de palabras, pero conforme el diccionario crece la búsqueda lineal se vuelve muy ineficiente. Para resolver este problema, se intentaron dos enfoques: el descarte rápido y la indexación.

El descarte rápido consiste en tomar una o dos características singulares de una palabra del diccionario, compararla con la palabra en estudio y descartarla si es que no coincidían dentro de ciertos límites. Fundamentalmente de intentó realizar este descarte rápido con la longitud, pero sólo es seguro este método si la diferencia de longitud entre ambas palabras es grande; lamentablemente la enorme mayoría de palabras duran entre 0.35 segundos y 0.55 segundos, con lo que este método ahorra tiempo en pocas ocasiones. También se probó con las columnas iniciales o finales de la matriz, pero el ruido del inicio de la grabación y los cortes impredecibles de los sonidos no vocálicos (especialmente las "s" finales) no lo hacen viable.

El segundo método que se analizó fue la indexación (véase punto 4.1.5). Esto requería hacer una muy importante adición al modelo inicial, pero la indexación del sonido necesita definir ciertos datos como índices, datos que tengan importancia como variables de decisión en el parecido

entre sonidos y que no sean de cálculo complicado. Después de varios intentos, no se encontró ningún conjunto de datos significativo y se abandonó este método.

4.3.2- Precisión:

La precisión que nos permite el sub modelo matricial depende mucho del tipo de palabras que se han de comparar, según el siguiente cuadro donde se muestra la precisión de una comparación entre una palabra y un modelo de palabra:

Long.Palabras	Corta	Intermedia	Larga
Corta	Mala	Regular	Buena
Intermedia	Regular	Regular	Buena
Larga	Buena	Buena	Muy buena

Donde palabras cortas son de duración menor a 0.35 seg, intermedias entre 0.35 y 0.55 segundos y largas mayores de 0.55 segundos. Una precisión muy buena es superior al 90%, una buena está entre 75% y 90%, una regular entre 55% y 75%, mientras que una precisión mala está por debajo del 55%. A simple vista, puede parecer que la precisión del submodelo es buena en promedio, sin embargo en los ejemplos que se han usado, así como en la mayoría de comandos de voz que usan las aplicaciones, el diccionario usado consta en su mayoría de palabras cortas e intermedias, por lo que se puede considerar que el sub modelo

matricial tiene una precisión solamente regular.

Esta es la causa principal que motivó la inclusión del sub modelo del análisis de ondas. La combinación de ambos modelos permite aumentar la precisión, debido a que en el análisis de onda funciona con similar precisión en palabras cortas que en las largas.

La descomposición en las formantes en el algoritmo implementado tiene una precisión entre un 70% y un 80% (según las pruebas realizadas se tenía 77.5% de precisión, como se detallará en la sección siguiente), que se compensa en el caso de las palabras del diccionario, puesto que han sido sometidas a varias repeticiones a efectos de entrenamiento. Según las pruebas realizadas, el algoritmo está tomando entre 2 y 3 puntos por vocal, por lo tanto en una palabra de dos sílabas tomará unos 5 puntos, y usando 4 repeticiones serían 20 vocales identificadas, de las cuáles 15 serían correctas y 5 falsas (asumiendo una precisión de 75% por cada intento). Los errores que se cometan serían pequeños. Suponiendo que la palabra fuese "salir", el algoritmo identificaría 8 ocurrencias de a, 7 de i, 2 de e, 2 de u y 1 de o. Se notará que los errores son pequeños respecto a los valores mayores.

Sin embargo, el caso es diferente para el reconocimiento propiamente dicho, por que se trata de una sola muestra. En el ejemplo anterior, si se

intenta reconocer la palabra "salir", el análisis de ondas en esta única ocurrencia podría arrojarnos 2 ocurrencias de a, 1 de e, 1 de i, 1 de u, extrapolando el análisis a un caso pesimista de 2 errores (60% de precisión) en el cual uno de los puntos dentro de la "a" se ha confundido por una "e", y uno de la "i" se ha confundido por una "u". Se ha usado por ello un algoritmo que disminuye la diferencia en la comparación entre vocales que se confunden para el sistema, según el siguiente cuadro que indica la probabilidad relativa de tomar una vocal de la muestra por otra en el modelo:

modelo\muestra	A	E	I	O	U
A	***	Media	Muy baja	Media	Muy baja
E	Media	***	Baja	Alta	Baja
I	Baja	Muy baja	***	Muy baja	Alta
O	Media	Media	Baja	***	Baja
U	Baja	Muy baja	Alta	Muy baja	***

Se disminuye la diferencia que hay en las secciones con probabilidad "alta", por ejemplo entre la "i" y la "o". Si la proporción en la muestra es i = 0.12, u = 0.56 y en el modelo es de i = 0.25, u= 0.40 la diferencia sería (0.25 - 0.12) + (0.56 - 0.4) = 0.29, pero el algoritmo disminuirá esta cantidad si la proporción de "i" y de "u" sumadas se parecen entre el modelo y la muestra.

Con esta regla, la precisión del submodelo de análisis de ondas al

identificar una palabra con dos sílabas llega a ser de poco más de un 80%, con la salvedad que puede identificar a más de una palabra como buen candidato para la muestra pronunciada. En combinación con el modelo matricial, que proporciona su propio esquema de comparación, llegan a una precisión como sistema que se explica en la siguiente sección.

4.4- Evaluación de la precisión y rapidez del sistema creado.

Esta evaluación es la más objetiva de todas, referente a la implementación real del modelo y del algoritmo, pues implica un programa desarrollado, una medición en tiempos en una PC determinada, así como la precisión del reconocimiento para un vocabulario determinado en un número predefinido de pruebas.

Se ha probado este sistema en una Pentium Pro, de 200 Mhz, con 32 Mb de memoria RAM y sistema operativo Windows 95. El vocabulario que se ha empleado consiste de las siguiente palabras:

- Los número del Uno al Nueve
- Izquierda
- Derecha
- Arriba
- Abajo
- Más
- Menos
- Por

- Entre
- Igual
- Salir
- Fuera?

En total, 20 palabras de las cuáles la mayoría tiene un máximo de dos sílabas (duración de hasta 0.5 segundos).

4.4.1- Velocidad: el promedio por palabra en el diccionario fue de 0.9 segundos. En total se demoró 18 segundos por todo el vocabulario. La diferencia de tiempo al reconocer diferentes palabras existe (por lo menos según los pasos que seguiría el programa), pero con reloj en mano no ha podido ser detectada, por lo que si existiera sería despreciable. Además, debe considerarse que el sistema aún puede optimizarse en performance, sin variar el algoritmo, debido a que se han duplicado algunos procesos por restricciones en el manejo de la memoria por el compilador C++

4.4.2- Precisión: las pruebas que se efectuaron consistieron en 3 ensayos por cada palabra, en series de 20 ensayos (1 por palabra) por día a lo largo de 3 días. En el siguiente cuadro resumen pueden apreciarse los resultados, donde un check (✓) indica un reconocimiento exitoso y un aspa (✗) indica error en el reconocimiento. Las marcas se presentan en el orden en que se realizaron las pruebas.

Palabra	Resultados	Palabra	Resultados
Uno	✗ ✓ ✓	Derecha	✓ ✓ ✓
Dos	✓ ✓ ✓	Arriba	✓ ✗ ✓
Tres	✓ ✗ ✓	Abajo	✓ ✓ ✓
Cuatro	✓ ✗ ✓	Más	✓ ✓ ✗
Cinco	✗ ✓ ✓	Menos	✓ ✓ ✗
Seis	✓ ✓ ✗	Por	✓ ✗ ✓
Siete	✗ ✓ ✓	Entre	✓ ✓ ✗
Ocho	✓ ✗ ✓	Igual	✗ ✓ ✓
Nueve	✓ ✓ ✗	Salir	✗ ✓ ✓
Izquierda	✓ ✓ ✓	Fuera?	✓ ✓ ✗

En resumen, la tasa de precisión es de 44 aciertos en 60 pruebas, o sea 73.3%. De los 16 errores, tenemos que en 7 de ellos la palabra correcta estaba entre las tres primeras elecciones del sistema, mientras que en las 9 restantes la palabra correcta estaba por debajo de ese nivel.

5. PROGRAMA DE APLICACION

Como complemento al sistema de reconocimiento de voz que se ha desarrollado en este trabajo, y además como elemento introductorio a las nuevas propuestas sobre interfases, se ha desarrollado un aplicativo que usa la interfase oral para realizar ciertas operaciones elementales sobre un ícono dentro de una ventana. Se pasa a describir los principios de las interfases orientadas al habla, la aplicación ejemplo y una evaluación de ella.

5.1- Descripción del programa y de los principios de la interfase verbal a implementarse.

En términos generales, hay 3 tipos de interfase por voz, según el paradigma en que se basa su desarrollo:

1. Imitación de la GUI, que consiste en la emulación de las opciones de movimiento y click del mouse o de los "shortcuts" (atajos o teclas de función) mediante la voz. Tiene la ventaja de que su implementación puede realizarse en los softwares comercializados actualmente (por

ejemplo en las suite para oficina), por que sólo requiere que el programa de reconocimiento de voz use emulación de teclado y mouse, o comunicación entre aplicaciones vía DDE (dynamic data interchange ó intercambio dinámico de datos) bajo Windows. En el aspecto de la tecnología de reconocimiento, son fáciles de implementar y rápidas en reconocer, pero no son buenos sustitutos del teclado para usuarios expertos.

2. Linea de comandos hablada, que consiste en implementar un lenguaje de comandos similar a la línea de comandos DOS, con un comando principal y un conjunto de parámetros. Así como en el DOS, existen comandos sin parámetros o con valores de parámetros por defecto. No pueden ser sistemas muy complicados, por que ello requeriría muchos comandos que el usuario encontraría difíciles de memorizar, y el sistema sería algo lento. Sin embargo, tienen la ventaja de que son fáciles de implementar y que su uso se adapta a los sectores industriales, donde se manejan por voz máquinas que ejecutan una serie de tareas bastante definidas, parametrizables y de una variedad bastante pequeña.
3. Lenguaje Natural: es el estado del arte. Requiere sistemas de reconocimiento continuo de palabras, las que pasan reconocidas como frases enteras a un motor de interpretación de lenguaje natural, el cuál elimina primero las palabras sin semántica o significado, tales como "por favor" o los artículos, luego trata de que las palabras restantes se encuadren en alguna de las gramáticas que tiene definido, y finalmente a

esa gramática de asigna los significados correspondientes, por medio de lo que son arboles sintácticos.

El ejemplo que se ha desarrollado es del segundo tipo, pero no se trata de una imitación de las líneas de comando de DOS o de UNIX. La diferencia de los paradigmas de las interfases con voz consiste en que las "líneas de comando" deben sonar parecido a las ordenes que damos en la vida real. Así, el comando "busca persona con código 123" en una línea de comandos tradicional sería "select personas, seek 123", pero en interfase hablada sería "busca persona 123".

Cabe indicar que estas interfases no son las únicas propuestas novedosas en lo que a interfases se refiere, pero si las más importantes en lo que se refiere a interfases verbales. Existe otra propuesta muy importante, que aunque no implica uso de la voz, no debe dejarse de mencionar. Esta es la interfase orientada al objeto, que ha sido implementada por algunos proveedores de software y hardware, tales como IBM en el SOM (System Object Model) en el que se basa el escritorio de OS/2, y el MacOS 7 de Apple. Estas interfases se conciben como una evolución y a la vez revolución de las interfases GUI, Por que si bien visualmente tiene elementos similares a las GUI, el comportamiento de estos elementos es totalmente diferente, pues cada uno de ellos es un objeto con propiedades, eventos y métodos propios; por ejemplo, los íconos de impresora son objetos que en verdad imprimen, si se escoge el ícono de un documento y se lo arrastra hacia ellos. La integración de browsers del WWW dentro de los

escritorios de Windows 95 y otros sistemas han permitido que Java sea ahora la novedad dentro de esta interfase. Se observará sin embargo que el reconocimiento de habla no ha recibido atención dentro de estos enfoques, cuya operación sigue dependiendo del teclado y el ratón.

El pequeño aplicativo que se ha diseñado consta de una ventana y las operaciones que soporta consisten en mover un icono en una ventana por ordenes orales. La comunicación entre el software de reconocimiento de voz y la aplicación ejemplo se realiza usando un canal DDE. Las ordenes que va a aceptar son:

(número) Arriba

(número) Abajo

(número) Izquierda

(número) Derecha

Centro

Fuera? + comando

Las cuatro primeras son ordenes de movimiento, que pueden o no tener un parámetro. En este caso el parámetro es pre-fijo (nótese cómo el nuevo paradigma nos permite acomodar la estructura de los comandos como más convenga) y consiste en el número de pasos que el ícono se va a mover en la dirección, y que en nuestro ejemplo van del número 1 al 9. El penúltimo comando es un comando

de movimiento a una ubicación predeterminada, en este caso el centro.

El último es una instrucción en vez de un comando, una instrucción condicional que ejecutará una acción si la figura esta fuera de los límites de la caja dibujada en la ventana. Por ejemplo, la línea completa "Fuera? ocho derecha" significa "si el ícono esta fuera de la caja muévelo ocho pasos a la derecha".

El diagrama de los estados posibles para la aplicación ejemplo es el siguiente:

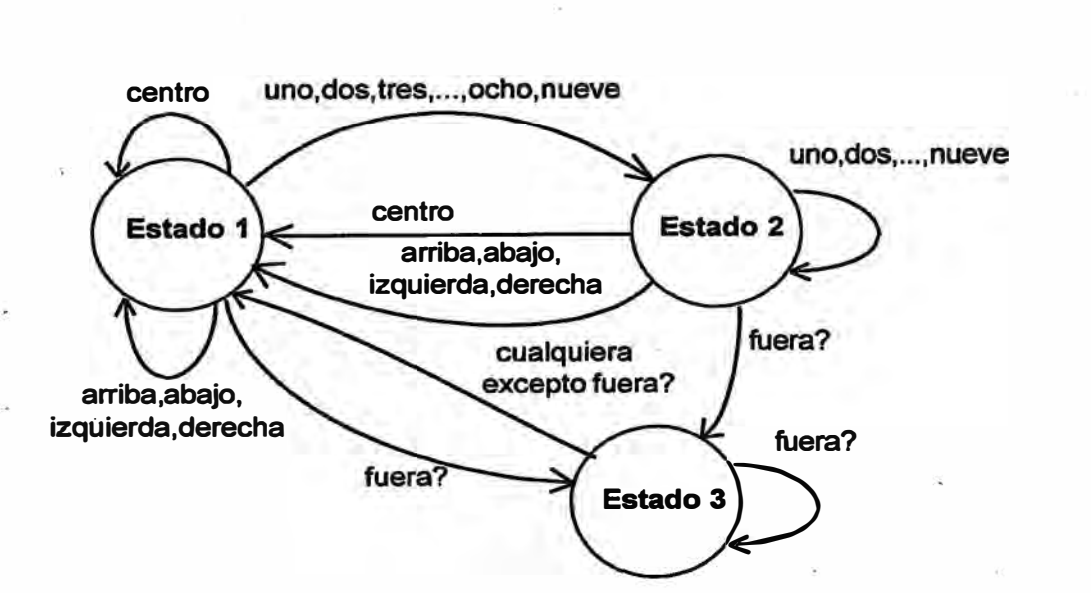


Figura 12: Diagrama de estados del aplicativo de demostración.

Como se puede observar, la base de un aplicativo con interfase orientada a la voz aprovecha mejor las ventajas de la voz si no es jerárquico como un menú, sino orientado al evento y al estado. En este caso, existen ciertos estados, como el estado 1, que son estados en que la aplicación está esperando una orden, mientras

que hay estados en que el sistema espera que se completen los comandos ya emitidos, o realizan evaluaciones, o se pueden encontrar realizando tareas repetitivas en espera de un comando de interrupción.

La principal diferencia con el enfoque GUI, sobretodo el basado en menus e iconos de comando, consiste en que el orden en que mientras que el enfoque GUI es orientado al tipo de tarea, a la tarea y a sus parámetros, en ese orden, en la interfase orientada a la voz se representan los comandos debe adaptarse a la naturaleza del lenguaje humano, que no tiene una estructura tan rígida. Por ejemplo, si se tiene una aplicación diseñada para el personal de las compañías de courier, como DHL ó UPS, que necesita manejar los paquetes con sus etiquetas de código de barra usando unos equipos de captura automática de datos con interfase GUI nativa, estos operarios necesitan usar por lo menos una mano para operar el paquete, y otra mano para usar el lector de código de barra. Cualquier operación en el equipo de captura de datos o terminal portátil (borrado, anotación, consulta de paquetes, etc.) es difícil de realizar con ambas manos ocupadas. Las compañías diseñadoras de estos terminales de captura de datos han creado una solución hardware, que consiste en un diminuto módulo de lectura que se usa como anillo en un dedo, dejando una mano casi libre. Esta solución es costosa y mala ergonómicamente; en este caso las interfases de voz se justifican.

La tarea del empleado de courier puede ser más fácil si el equipo de captura de datos reconociera comandos de voz para las labores más comunes

como anular un ingreso o consultar en pantalla. La captura de datos de este tipo consiste en obtener grandes cantidades información similar entre sí, por lo que la variedad de los comandos de voz es pequeña. Evidentemente, esta solución tendrá que esperar mejores métodos de reconocimiento de voz y la disminución del precio de la circuitería de sonido compacta (las tarjetas de sonido para portátiles).

5.2- Desempeño del aplicativo. Pros y contras de la interfase verbal.

La interfase verbal que se usó se comportó manera aceptablemente. La rapidez del uso no es comparable a las GUI, pero su facilidad de uso es similar y su intituitividad es mayor, porque requirió menos tiempo de entrenamiento en usuarios novatos que su similar GUI (la cuál usaba menus jerárquicos), debido a que no hacia falta explicar lo que es un click, un arrastre de ratón o una tecla de función. El promedio de tiempo para que el usuario novato empezara a usar la aplicación con interfase oral fue de 12 minutos, contra 23 minutos de la GUI.

Las ventajas que se han observado son:

- Intuitividad, ya que los usuarios novatos comprenden fácilmente las ordenes verbales que hay que brindarle a la aplicación.
- Facilidad de uso, siempre y cuando la aplicación no tenga demasiados comandos o comandos con demasiados parámetros.
- Es una alternativa de entrenamiento para los usuarios novatos, que pueden perder el temor o el recelo a usar computadoras si sus primeros

contactos son por ordenes orales.

Las desventajas son:

- La lentitud, si se compara la velocidad con que las tareas se pueden realizar con ratón y teclado por un usuario con experiencia. Los usuarios novatos también se demoran más con la interfase oral, pero la diferencia es mucho menor que con los usuarios experimentados.
- El sistema presenta inexactitudes en el reconocimiento, por lo que el usuario a veces debe repetir el comando sin haber cometido ningún error. Esto causa cierta frustración o molestia en algunos usuarios novatos.
- La necesidad de feedback, es decir de información que le indique al usuario si esta interactuando bien con el aplicativo. Este hecho es algo que no se había previsto en el momento del diseño del aplicativo, pero que salió como inquietud de los usuario experimentados y de algunos usuarios novatos. Los sistemas tradicionales proporcionan un feedback al usuario debido a su naturaleza gráfica (los menues se despliegan, los textos se llenan, los botones se marcan y parecen ser oprimidos), pero no se penso en el feedback del sistema con interfase oral.

El último punto, que al autor le ha parecido muy sorprendente, no se ha encontrado detallado en los diferentes libros que se han consultado sobre el

desarrollo de aplicaciones, sino que parece ser un elemento que por implícito se ha obviado. El principal objetivo de la aplicación era independizar al usuario de teclados o ratones, pero al parecer el usuario aún no está listo para independizarse del monitor, y este elemento visual cumple una labor más importante de lo que habíamos pensado.

Como se puede apreciar, las desventajas de las interfases orales son tan importantes o más que sus ventajas. La realidad muestra que las interfases orales sólo son recomendables en las siguientes situaciones:

- Cuando el usuario es novato, ya sea como interfase permanente o como entrenamiento para posteriores interfases
- Cuando el usuario esta impedido cabalmente del uso de ambas manos, ya sea por impedimento o por la naturaleza de su labor, y esta labor necesita sólo un pequeño conjunto de comandos orales.
- Cuando el proceso que el usuario debe realizar es por medio del teléfono y no disponga de modem (por ejemplo home banking), y donde por alguna razón no se pueda usar los tonos o los pulsos telefónicos, ya sean razones técnicas o por la complejidad de los comandos. Este último caso es aplicable a las transacciones bancarias por teléfono en las que la navegación de los sistemas tradicionales de consulta telefónica es muy lenta, o en las que se puede usar la voz como medio adicional de identificación. Esta última aplicación es la identificación por voz, que es materia de otro tipo de

investigaciones.

- Las aplicaciones de trabajo de campo, que necesitan ser efectuadas con computadoras portátiles o terminales de captura de datos, en condiciones en las que el uso del teclado o del mouse se vuelve incomoda o ineficiente, tales como la captura de códigos de paquetes, mercaderías, etc.

Cada aplicación requiere un enfoque distinto, pero en general sólo serían variaciones de los comandos de voz, por que el problema con el que tratan son situaciones sui generis, en las que el uso del teclado o el mouse es imposible o inconveniente.

6. CONCLUSIONES

Las conclusiones que se han extraído de la presente tesis son:

1. El reconocimiento de voz es una herramienta valiosa para implementar interfases en casos particulares en los que el empleo de los métodos tradicionales tiene impedimentos. Aún no es un sustituto adecuado para la generalidad de los casos
2. El método propuesto se inicio como un modelo matricial, el cuál demostró interesantes potencialidades pero también seria limitaciones, por lo que se buscó un segundo modelo para complementar al primero
3. El segundo modelo introducido fue el modelo de análisis de ondas usando la Transformada discreta rápida de Fourier, que permite la identificación de los segmentos vocálicos dentro de las palabras con una precisión aceptable.
4. Ambos modelos se pueden utilizar en forma conjunta debido a que ninguno de ellos depende o está condicionado por el otro. Al ser independientes, pueden ser usados como un modelo único aditivo, en el que cada sub modelo aporta su discriminación al modelo conjunto.

5. Los algoritmos que sirven para implantar el modelo se han optimizado en la medida de lo posible, pero aún quedan mejoras por realizar, la más importante de las cúales sería implementar un inétodo de indexación de los sonidos.
6. La performance del sistema final en cuanto a velocidad es aceptable para diccionarios de palabras pequeños, pero inaceptable para diccionarios mayores.
7. La precisión es de 73.3%, menor que la de los sistemas comerciales disponibles pero con capacidad de mejorar si se realiza más cantidad de cómputo por palabra.
8. Es posible extender el modelo a fin de atacar el problema del reconocimiento independiente del hablante. La solución descansaría más en el segundo modelo, el de análisis de formantes de ondas.
9. No es posible extender el modelo al reconocimiento de habla continua. El castellano, a diferencia del inglés, tiene particularidades que hacen más difícil crear sistemas de reconocimiento de voz continuos, pues la separación entre palabras es muy poco notable en el castellano.
10. La interface orientada a la voz aún no es tan práctica como las GUI, pero proporciona una mayor flexibilidad, permitiendo su uso de forma más intuitiva y pudiendo constituirse en una herramienta para introducir a usuarios conservadores en el uso de las computadoras personales.

RECOMENDACIONES

Las principales recomendaciones que se han de brindar a quienes deseen proseguir con estas investigaciones en reconocimiento de voz, ya sea prosiguiendo con el enfoque usado en esta tesis u obtando por uno distinto, serían las siguientes:

- Para quien desee continuar esta línea, se recomienda que se perfeccionen algunos detalles del algoritmo para hacerlo más eficiente, así como se estudie la posibilidad que los dos modelos usados, matricial y de análisis de ondas, no sean independientes, sino que tengan alguna especie de correlación que mejore el reconocimiento.
- La indexación de los patrones o "modelos" de las palabras es imprescindible si se desea atacar el problema del reconocimiento de grandes diccionarios de palabras o del reconocimiento de habla contínua. La búsqueda de las palabras con mayores probabilidades de parecido podría responder a la estructura de un árbol de n ramas, en el que se agrupen los índices por semejanza o parecido.
- Tenga en cuenta que gran parte del material sobre reconocimiento de voz está centrado en la lengua inglesa, no por la falta de estos programas para castellano, si no por que el interés en el mundo académico está centrado en las dificultades generales del reconocimiento de voz, no en las dificultades específicas de un idioma en particular. Por ello se ha de desarrollar

adaptaciones a los diferentes métodos.

- La evaluación de los resultados debe hacerse bajo dos consideraciones, una tomando el método desarrollado por sí solo y otra comparándolo con los sistemas comercialmente disponibles.
- Si la implementación final desea lanzarse al mercado comercial, ya sea como versión final o como beta, se han de desarrollar rutinas a bajo nivel para la grabación del sonido y se ha de realizar programación multihilos a fin de procesar la data mientras se graba, a fin de ganar tiempo. Todos los programas comerciales disponibles hoy tienen esas capacidades.

ANEXOS

ANEXO A

Conceptos Complementarios respecto al Audio

Teorema de Fourier: cualquier onda repetitiva o periódica puede ser descompuesta en la suma de una serie de ondas sinusoidales, una de las cuales es de la misma frecuencia que la onda periódica y se llama fundamental, y las demás son de frecuencias múltiplos de ella, y se llaman las armónicas. En forma general, para una función periódica f(x), con longitud de onda λ se tiene que:

$$f(x) = a_0 + \sum_{i=1}^{\infty} a_i sen(ix) + \sum_{i=1}^{\infty} b_i \cos(ix)$$

Análisis de Fourier: consiste en hallar la serie de Fourier en que se descompone una onda periódica, o en que se puede descomponer una función no periódica dentro de un rango finito a la que se le trata como si fuese periódica.

Espectro: se llama espectro a una representación de las amplitudes y frecuencias de una determinada onda, en un gráfico de frecuencia versus amplitud. Es un

modo conveniente de especificar una onda. Los espectros de ondas complejas pueden discretos, si la onda es periódicas (en este caso el espectro es un conjunto de puntos discretos, a espacios regulares) o continuos si la onda es aperiódica (el espectro es un conjunto de puntos continuos en el dominio de las frecuencias).

PCM (Pulse-Code Modulation): es el proceso más sencillo de conversión de sonido entre su forma analógica y su forma digital. Cada muestra o sample analógica es asignada a un valor de 8 ó 16 bits que es proporcional (lineal) a su amplitud. El audio de un CD de música está archivado como LPCM de 16 bits. Es un método muy sencillo de implementar en la circuiteria de una tarjeta de sonido.

Teorema de Nyquist (teorema del muestreo): solamente se puede reproducir o grabar fidedignamente una onda si el ratio de muestreo es por lo menos el doble que la frecuencia del componente de la onda de más alta frecuencia. Como la mayor frecuencia que puede escuchar el ser humano es de poco más de 20 kHz (en niños y personas con excelente audición), el ratio de muestreo de un CD de audio, de 44.1 kHz es suficientemente adecuado.

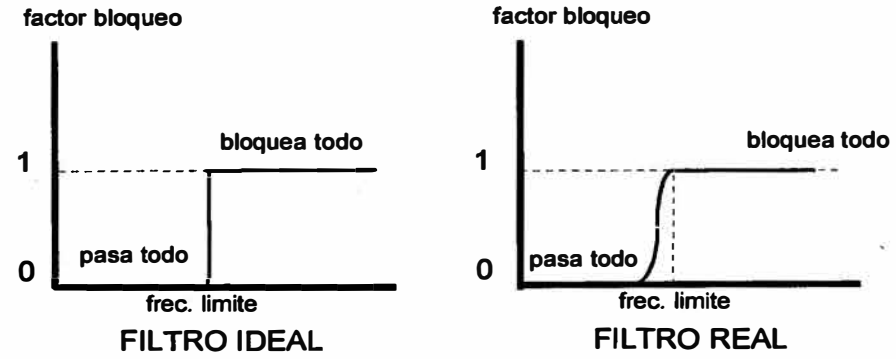
Ancho de banda: rango de frecuencias dentro del cuál un dispositivo puede producir sonidos o puede grabarlos, según sea el caso. Un mayor ancho de banda garantiza una mejor reproducción o grabación del sonido. Por ejemplo, el ancho de banda de una estación FM es 3 veces la de una AM, aproximadamente.

Ruido: sonidos aleatorios que se presentan repentinamente y enmascaran al sonido deseado. En este caso sonidos no deseados se superponen al sonido deseado.

Distorsión: cambio no deseado de las frecuencias de los componentes de una señal (por señal se entiende en esta caso tanto a la sonora como a sus análogas eléctricas o mecánicas) al pasar por un dispositivo de audio. Si fuese una tarjeta de sonido, el análogo es la señal eléctrica, en el caso de un tocadiscos también hay un análogo mecánico en el movimiento de la aguja y el brazo al leer el disco. Nótese que no hay superposición de sonidos, sino alteración del sonido deseado.

Filtro de entrada o de grabación (input filter): es un filtro digital que deja pasar libremente las señales bajo cierta frecuencia, mientras que bloquea las señales encima de dicha frecuencia. En las tarjetas de sonido esta frecuencia límite es determinada usando el teorema de Nyquist. Por ejemplo, si el ratio de muestreo es de 44.1 kHz, la mayor frecuencia que se podría grabar es de 22.05 kHz, y la frecuencia límite del filtro se establece un poco más abajo, en unos 20 kHz. Como técnicamente es imposible crear un filtro perfecto, que deje pasar totalmente las señales menores a 22.05 kHz y bloquee totalmente las mayores, los ingenieros se ven obligados a seleccionar una frecuencia límite menor a la teórica. La falta de un filtro o su funcionamiento defectuoso crean un tipo de distorsión llamado aliasing.

Aliasing: tipo de distorsión del sonido donde aparecen en la grabación sonidos falsos de baja frecuencia, sonidos que nunca existieron en el lugar y momento de la grabación. El aliasing ocurre cuando el ratio de muestreo es muy bajo para capturar fielmente los componentes de más alta frecuencia de los sonidos a grabar. Supóngase que se usa un ratio de muestreo de 11.025 kHz , según el teorema de Nyquist podríamos capturar adecuadamente componentes del sonido con frecuencias de hasta 5.5 kHz ; si se graba la voz de un niño (que tiene componentes de mayor frecuencia que 5.5 kHz, digamos 6 kHz) esos componentes no se graban y en su lugar aparecen sonidos falsos con una frecuencia igual a la diferencia entre la frecuencia original y el ratio de muestreo, es decir 6 kHz - 5.5 kHz = 0.5 kHz



Compresión de audio: la compresión de sonido usa técnicas especiales, diferentes a la de la compresión de programas o de gráficos.

Modulación LPCM (Linear Code Pulse Modulation): véase PCM

Modulación ADPCM (Adaptative Diferential PCM): es una variante sofisticada del PCM, que permite la compresión del audio, usando menos bits por muestra. Comprime el audio con una mínima pérdida de calidad. El truco detrás del ADPCM es capturar la diferencia entre un sample y el siguiente, en vez de capturar el valor de cada sample. Se puede reconstruir la forma de la onda tomando su valor inicial y estableciendo cada cambio. Por ejemplo, con sólo 4 bits (16 valores diferentes) por muestra se puede reproducir aceptablemente sonido de 16 bits en modo PCM; el valor de 4 bits no es la diferencia absoluta, sino la diferencia en un factor de escala, por el que se debe multiplicar el actual sample para obtener el siguiente (+10%, +30%, +50 %, +90%,... y sus respectivos negativos). Con ADPCM de 8 bits por sample se puede usar 256 factores distintos, más que suficiente para comprimir sonido PCM de 16 bits con muy buena calidad.

A-law y Ω-law: son algoritmos estándar de la CCITT para la compresión del habla. A-law es usado principalmente en Europa, mientras que Ω-law es más popular en Estados Unidos y Japón. Son algoritmos similares, que varían en algunos detalles de implementación; son primos cercanos del PCM. Mientras que con PCM las muestras son grabadas según una escala lineal (como una regla), mientras que A-law y Ω-law usan una escala logarítmica. El efecto neto de estos algoritmos es que con datos de 8 bits se puede llegar a alcanzar el ratio sonido/ruido y el rango de decibeles de una señal de 12 bits PCM. Las tarjetas de sonido comerciales generalmente usan un enfoque más económico en espacio, en

vez de conservar una total fidelidad de sonido, comprimiendo señales de 16 bits PCM en 8 bits de A-law ó Ω-law, mejor que la compresión equivalente 2:1 usando ADPCM.

ANEXO B

Formatos de Audio

Debido a que en los comienzos de la tecnología de audio digital los distintos fabricantes se adhirieron a diferentes estándares, hoy en día se pueden contar por decenas los formatos para audio digital. En la presente relación se incluyen sólo los formatos más populares para las principales plataformas tradicionales de audio: PC compatibles, Apple Macintosh, estaciones de trabajo UNIX, etc. No se incluyen los formatos tipo MIDI, que sólo guardan música de una consola de sintetizador.

Audio Interchange File Format (IFF): formato usado por las Apple Macintosh para guardar audio digitalizado. Soporta una amplia variedad de ratios de muestreo y tamaños de muestreo, de hasta 32 bits por muestra. Este formato soporta loops, es decir, la reproducción repetitiva de un bloque de audio digital, lo que puede ahorrar mucho espacio en temas musicales. El IFF también fue soportado por la Commodore Amiga.

MOD: formato musical nativo de las Commodore Amiga. A diferencia de otros

formatos (no incluidos en este anexo) el MOD contiene fragmentos de audio digitalizados de instrumentos musicales. Al reproducirse una nota de un determinado instrumente, el formato MOD puede incluir distorsiones, reberberaciones y otras manipulaciones del audio incluidas en este formato.

RIFF (RMI): es el Resource Interchange File Format de Microsoft, diseñado para ser el definitivo formato para multimedia de Microsoft Windows, capaz de incrustar "chucks" (término de Microsoft para denominar a los bloques) de diferentes tipos de datos. Entre estos tipos de datos están los bloques de audio digital Wave (WAV) y los MIDI. La belleza del RIFF es que puede incluir bloques de tipos de datos aún no inventados. Lamentablemente el formato RIFF no se popularizado mucho.

Sound (SND): el formato Sound Resource es un formato de audio digital compacto, de sólo 8 bits por muestra, soportado por Apple. Es usado para sonidos cortos, tales como la alerta del parlante interno de la Macintosh, y por aplicaciones Macintosh con necesidades simples de audio digital.

SUN Audio (AU): las estaciones de trabajo Sun Microsystems usan archivos de audio AU, que son formatos de 16 bits comprimidos por los métodos A-law y Ω-law. Este tipo de sonido es muy común en Internet, debido a la fuerte presencia de servidores Sun en los inicios de los servicios FTP, Gopher o WWW.

Voice (VOC): formato de Creative Labs para audio digital. Soporta tanto 8 como 16 bits por muestra, y un amplio rango de ratios de muestreo, desde los 8000 samples/segundo. La data puede ser grabada comprimida o sin comprimir. Los archivos de 8 bits pueden comprimirse en 2, 2.6 ó 4 bits comprimidos / 8 normales, lo que significa tasas de compresión de 4 a1, 3 a 1 y 2 a 1 respectivamente. Los archivos de 16 bits pueden ser comprimidos a 4 bits o a 8 bits, lo que proporcionaría tasas de compresión de 4 a1 y de 2 a 1 respectivamente. Además el formato VOC tiene marcadores para loops, de sincronización de audio con otros elementos multimedia, y marcadores de silencio que reemplazan un cuasi-silencio por un sólo marcador. Actualmente, el formato VOC ha encontrado más acogida en los juegos y aplicaciones DOS.

Wave (WAV): formato de Microsoft dirigido al uso de multimedia en Windows. Puede manejar 8 bis ó 16 bits por muestra, sonido monoaural o stereo, y tasa de muestreo de 11.025 kHz, 22.05 kHz ó 44.1 kHz. Este formato es soportado por casi todas las aplicaciones multimedia en Windows , así como las tarjetas de sonido para este entorno operativo. Además, las APIs multimedia de Windows incorporan funciones de alto y bajo nivel para grabar, reproducir y leer este formato.

ANEXO C

Estructura del Formato Wave

La siguiente es una visión general del formato WAV. Recuérdese que el formato WAV es un tipo de formato diseñado como un "chunck" o bloque del tipo RIFF.

Formato general de un bloque RIFF (y por lo tanto también WAV)

El bloque básico de un archivo RIFF es

<rID><rLong><rData>

Donde:

- <rID> es el identificador del bloque, siempre es la palabra "RIFF" (4 bytes).
- <rLong> es la longitud de la data en el bloque que sigue (4 bytes).
- <rData> es el "chunk" RIFF de formato WAV (de rLong bytes).

Definición de formatoWAV

El formato WAV está subdividido en bloques. Siempre debe contener un bloque

de formato, seguido por un bloque de data (el bloque rData de la sección anterior).

<rData> = <wID><bloque de formato><bloque de data>

Donde:

- <wID> es el identificador del formato WAV, siempre es "WAVE" (4 bytes)

Bloque de formato

El bloque de formato contiene data que especifica el formato de la data contenida en el bloque de data. Su estructura es la siguiente:

<Bloque de formato> = <IDchunk><fLong><wFormatTag> <nCanales> <nSamplesSeg><nBytesSeg><nAlinBloque><ParámetroFormato>

Donde:

- <IDchunk> es el identificador del chunk, siempre es "fmt " (4 bytes)
- <fLong> es la longitud del chunk que sigue (4 bytes)
- <wFormatTag> indica el tipo de WAV (2 bytes). Por ejemplo el formato PCM siempre es 01.
- <nCanales> indica el número de canales (2 bytes). Mono=1, Stereo=2.
- <nSamplesSeg> indica el ratio de muestreo en muestras por segundo, en

el que cada canal de audio debe reproducirse (4 bytes)

- <nBytesSeg> indica el promedio de bytes por segundo que deberían ser transferida la data (4 bytes).

  <nBytesSeg> = nCanales * nSamplesSeg * (nBitsSample/8) .....(*)

- <nAlinBloque> indica el alineamiento en bytes de los datos del bloque de data.
- <ParámetroFormato> contiene cero o más parámetros (2 bytes).

Nótese que el número de bits por muestra (nBitsSample) se debe calcular de la fórmula (*).

Bloque de data WAV

El bloque de datos contiene el audio propiamente dicho. El formato de la data depende de lo que indique el bloque de formato.

<Bloque de data> = <dID><dLong><dData>

Donde:

- <dID> identificador del bloque de data, siempre es "data" (4 bytes)
- <dLong> es la longitud de la data en el chunk que sigue (4 bytes)
- <dData> es la data de la forma de onda (de dLong bytes).

ANEXO D

Transformada Rápida de Fourier

La Série de Fourier (FS) relaciona el dominio continuo en el tiempo con el dominio de frecuencia discreta; y la Transformada de Fourier (FT) relaciona el dominio continuo en el tiempo con el dominio continuo en la frecuencia. La transformada de Fourier discreta en el tiempo (DTFT) relaciona el dominio discreto en el tiempo con el dominio de frecuencia continuo. Aquí se examina la transformada discreta de Fourier (DFT), la cual relaciona dominio discreto en el tiempo con dominio de frecuencia discreto.

Se sabe que la serie de Fourier es una serie de senos y cosenos con frecuencias múltiplos de una frecuencia determinada, que corresponde a la frecuencia de la función original, tal como se muestra a continuación, para una función de periodo T y frecuencia 1/T:

$$f(t) = a_0 + \sum_{n=1}^{\infty} a_n sen(\frac{2\pi nt}{T}) + \sum_{n=1}^{\infty} b_n \cos(\frac{2\pi nt}{T})$$

Pero conocemos que $e^{jx} = \cos(x) + j\,\sin(x)$, y si definimos $c_n = (a_n - j\,b_n)/2$ tenemos que la función se puede expresar como:

$$f(t) = \sum c_n e^{jW_n t} \quad , \quad W_n = 2\pi n/T$$

La transformada de Fourier es una representación derivada de esta última igualdad, excepto que se aplica, estrictamente hablando, a funciones continuas y no periódicas, porque es la extensión de los coeficientes de la igualdad anterior a una serie de términos que abarcan todas las frecuencias posibles en el dominio de los números reales. La transformada discreta de Fourier y su inversa están dadas por:

$$X[k] = \sum_{n=0}^{N-1} x[n] W_N^{kn} \quad , \quad k = 0, 1, ..., N-1$$

$$x[n] = \frac{1}{N} \sum_{n=0}^{N-1} X[k] W_N^{-kn} \quad , \quad n = 0, 1, ..., N-1$$

Donde $W_N = e^{2\,j/N} = e^{j2\pi FT}$, siendo j la unidad de los números imaginarios.

Nótese otra particularidad de las Series de Fourier (que es evidente en las transformadas), que se pueden expresar sólo como senos o sólo como cosenos:

$$\begin{aligned} a_n\, sen(W_n t) + b_n \cos(W_n t) &= K(a_n\, sen(W_n t) + b_n \cos(W_n t))/K \;, K = \sqrt{a_n^2 + b_n^2} \\ &= K(\cos(\theta)\, sen(W_n t) + sen(\theta)\cos(W_n t)) \quad donde\, \theta = \arctan(b_n/a_n) \\ &= K\, sen(\theta + W_n t) \end{aligned}$$

En el diseño de una DFT para un uso particular, se deben escoger los valores para los parámetros N, T y F. N es el número de valores de la secuencia

discreta x[n] sobre los cuáles se realiza la sumatoria para calcular cada valor de la secuencia en el dominio de la frecuencia. También representa el número de valores X[m] producida por la DFT. El conjunto completo de los N valores consecutivos de la secuencia en el tiempo es denominada registro de entrada, y el conjunto de los N valores consecutivos de la frecuencia es llamada registro de salida. T es el intervalo de tiempo entre dos muestras consecutivas de la secuencia de entrada, y F es el intervalo de frecuencia entre dos muestras consecutivas de la secuencia de salida. Estos valores están sujetos a las siguientes restricciones:

1. El producto de F,N y T es igual a 1 (propiedad inherente a la DFT)
2. Como consecuencia del teorema de Nyquist , se tiene que $T= 1/(2f_H)$, donde f_H es la componente de frecuencia significante más alta de la señal continua en el tiempo, es decir la máxima frecuencia formante que se puede registrar discretamente si el tiempo entre muestras es T.
3. La longitud de registro en el tiempo es igual a NT ó 1/F (longitud de onda correspondiente a la frecuencia fundamental).

Algoritmo de la Transformada rápida de Fourier (fft).- la descomposición del cálculo de la DFT en sucesivos cálculos de DFT más pequeñas conduce a una gran eficiencia de cálculo. Los algoritmos basados en esta descomposición son llamados decimación en el tiempo.

Tomaremos N (número de muestras) tal que sea potencia de 2. La

secuencia de x[n] se puede separar en dos secuencia de N/2 puntos, una con los puntos pares y otra con los impares. Luego se tiene que:

$$X[k] = \sum_{par} x[n]W_N^{nk} + \sum_{impar} x[n]W_N^{nk} \quad ;pero \quad W_N^2 = W_{N/2} \ , \ entonces:$$

$$X[k] = \sum_{r=0}^{N/2-1} x[2r]W_{N/2}^{rk} + W_N^k \sum_{r=0}^{N/2-1} x[2r+1]W_{N/2}^{rk} = G[k] + W_N^k H[k]$$

Donde los pares son 2r y los impares son 2r+1, y G[k] y H[k] son símbolos para las sumatorias de pares e impares que además son otra DFT, y son funciones periódicas con período N/2, lo que permite calcularlas sólo entre 0<= k <= N/2 -1, en lugar de ir hasta N-1. Después, se puede descomponer cada una de las DFT pequeñas en otras más pequeñas, hasta llegar a una DFT de dos puntos, que llega a requerir una sola multiplicación de números complejos.

Para el caso general, si se tiene que $N=2^v$ donde $v=\log_2 N$, por el método tradicional de cálculo se tendría que hacer N^2 operaciones de números complejos, mientras que con la transformada rápida sólo deben hacerse $N\log_2 N$. Por ejemplo, para $N= 2^{10}=1024$, el cálculo de la DFT requeriría $2^{20}=1'048,576$ operaciones complejas pero la transformada rápida necesitaría 1024(10)=10,240 operaciones complejas, 100 veces menos.

ANEXO E

Dispositivos MCI

Introducción a los dispositivos MCI

MCI (Media control interface o interfase de control de medios) es un estándar propuesto y adoptado por Microsoft en sus sistemas operativos a partir del Windows 3.1 (aunque tiene antecedentes en el Windows 3.0 con las Multimedia extensions) para el empleo de los dispositivos multimedia. El MCI es una interfase de comandos de alto nivel para el empleo de dispositivos multimedia y los archivos relacionados para tal fin (drivers y archivos de configuración). Además proporciona a las aplicaciones con capacidades independientes del dispositivo para controlar los periféricos de audio y video. Las aplicaciones pueden usar los comandos MCI para controlar cualquier dispositivo multimedia con soporte MCI, incluyendo la grabación y reproducción del sonido.

MCI proporciona comandos estandarizados para reproducir dispositivos multimedia y graba desde ellos a archivo. El control de los dispositivos multimedia incluye comandos como open, play, y close. MCI incluye dos tipos de

interface:

- La interfase de comandos tipo mensaje, que se basa en constantes y estructuras de lenguaje C. El mensaje del que se hable va a uno de los módulos de sistema operativo y provoca un evento del sistema.
- La interfase de cadenas de comandos que provee una versión modo línea de comando de la interfase anterior. Las cadena son de un formato fácil de leer que facilita el control de los dispositivos MCI desde programas en C y herramientas de autoría (programación) multimedia. Las cadenas de comandos duplican la funcionalidad de los comandos tipo mensaje; Windows convierte las cadenas de comandos a comandos tipo mensaje antes de mandarlos al driver MCI para su proceso.

El control de los dispositivos MCI

Para controlar un dispositivo MCI, se debe abrir el dispositivo, mandar los comandos que se deseen ejecutar, y cerrar el dispositivo. Por ejemplo, la siguiente serie de comandos MCI tocan el track 6 de un CD de audio:

```
open cdaudio
set cdaudio time format tmsf
play cdaudio from 6 to 7
close cdaudio
```

El siguiente ejemplo muestra una serie similar de comandos MCI que tocan los primeros 10,000 samples o muestras de un file de sonido WAV (a través de lo que se conoce como un dispositivo waveform ó waveaudio):

```
open c:\mmdata\applause.wav type waveaudio alias sonido
set sonido time format samples
play sonido from 1 to 10000
close sonido
```

Como se puede apreciar de los ejemplos anteriores:

- Los mismos comandos básicos (open, play, and close) se usan en ambos dispositivos.
- El comando open para el dispositivo "waveaudio" incluye la especificación del nombre de un archivo. El dispositivo "waveaudio" es un dispositivo compuesto (asociado con un elemento multimedia, en este caso un archivo), mientras que el dispositivo "cdaudio" es un dispositivo simple (sin asociación con ningún elemento multimedia más).
- Los comandos set especifican en ambos casos el formato en que se va a medir el tiempo, pero el formato de tiempo para el dispositivo "cdaudio" difiere del usado para el dispositivo "waveaudio".
- Los parámetros usados con los flags "from" y "to" tiene un significados específico para cada dispositivo. Para el dispositivo "cdaudio", los

parámetros especifican un rango de tracks o canciones; para el dispositivo "waveaudio", los parámetros especifican un rango de samples o muestras.

Tipos de dispositivos MCI

MCI reconoce un conjunto básico de tipo de dispositivos. Un tipo de dispositivo es un conjunto de drivers MCI que comparten el mismo grupo de comandos y son usados para controlar dispositivos multimedia o archivos multimedia de similar funcionamiento.

Cada vez aumentan más los tipos de dispositivos multimedia, debido a que su implementación puede realizarse para cualquier dispositivo físico o lógico que funcione como MCI (se abra, se reproduzca y/o grabe y se cierre). La siguiente es una lista de los más comunes tipos de dispositivos disponibles actualmente:

Tipo de dispositivo	Descripción
animation	Dispositivo de animación (software)
cdaudio	CD player de audio
dat	Digital audio tape player
digitalvideo	Video digital en una ventana (no basado en GDI[1])

[1] Graphics Device Independant (Independiente del dispositivo gráfico): se dice de los software, lenguajes y aplicativos, que no necesitan de ninguna configuración que dependa del hardware de la trajeta gráfica.

overlay	Dispositivo Overlay (video analógico en una ventana)
scanner	Scanner de gráficos e imágenes.
sequencer	Secuenciador MIDI
vcr	Grabador o reproductor de cintas de video
videodisc	Reproductor Videodisc
waveaudio	Dispositivo de audio que toca archivos de onda digitalizados

Comandos para el audio con forma de ondas (Waveform)

Comando	Descripción
capability	Obtiene las capacidades del dispositivo.
close	Cierra el dispositivo
cue	Prepara para el grabado o la reproducción.
delete	Borra un segemento de datos de un elemento MCI.
info	Obtiene información sobre el dispositivo.
open	Inicializa el dispositivo.
pause	Hace una pausa al reproducir o grabar.
play	Comienza a reproducir un elemento MCI.
record	Comienza a grabar sonidos.
resume	Reinicia la reproducción o grabación del dispositivo.
save	Salva la data a un archivo.

seek	Mueve la posición en el sonido adelante o atrás.
set	Establece el estado operativo del dispositivo.
status	Obtiene información del estado del dispositivo.
stop	Detiene la reproducción o la grabación.

APIs multimedia de Windows

Desde el Windows 3.1 en adelante, se han establecido APIs multimedia que generalmente son llamadas a funciones definidas en un DLL del sistema, tal como la librería MMSystem.dll, o de librerías y headers C como MMSystem.h. Se pueden encontrar aquí diferentes tipos de rutinas, desde funciones de alto nivel a funciones de bajo nivel, de entrada salida para los files RIFF, librerías para el uso de timers manejados por el sistema operativo o para que los comandos MCI se direcciones al dispositivo que les corresponda.

En general, las APIs multimedia de Windows estan mal organizadas, y a menos que se desee un control muy sofisticado sobre los dispositivos multimedia, es preferible usar solamente los comandos MCI.

MCI y la programación en C

La base de la programación de los comandos MCI en C son las estructuras de datos o registros que son diferentes para cada tipo de comando. Por ejemplo, es esta tesis se han usado las siguientes estructuras, entre otras:

- Estructura MCI_SET_PARMS corresponde al comando Set
- Estructura MCI_RECORD_PARMS corresponde al comando Record
- Estructura MCI_OPEN_PARMS corresponde al comando Open

Cadá una de estas estructuras contiene variables que corresponden a los parámetros que el comando asociado utiliza. En vez de pasar los parámetros uno por uno al invocar el comando, se pasa la estructura que contiene a todos ellos. Por ejemplo, si tenemos una variable tipo MCI_OPEN_PARMS llamada MciOpenParm, puede inicializarse así:

```
MciOpenParm.wDeviceID = DeviceId;
MciOpenParm.lpstrDeviceType = "waveaudio";
MciOpenParm.dwCallback = 0;
MciOpenParm.lpstrElementName = "applause.wav";
MciOpenParm.lpstrAlias = 0;
```

En este caso, se está abriendo un dispositivo de sonidos WAV, asignándole un número identificador DeviceID y el archivo de sonido "applause.wav". Para ejecura los comandos, se encesita además de la estructura, ciertos flags de modo del comando, con el propósito de especificar que comando

se desea ejecutar y con que restricciones, como en los siguientes ejemplos:

```
mciSendCommand (0,MCI_OPEN,MCI_OPEN_ELEMENT|MCI_OPEN_TYPE,
MCI_PARM2(MciOp))
mciSendCommand(DeviceId, MCI_RECORD, MCI_WAIT | MCI_TO ,
MCI_PARM2(MciRecord))
```

En la primera función C, se invoca al comando MCI Open, se especifica que los modos en que va a operar el Open son abrir elemento y especificar el tipo, y se le pasa una estructura de parámetros llamada MciOp. En la segunda función, se está emitiendo un comando MCI Recors, para grabar sonidos, se especifica que el modo de grabación es modal ("wait", o sea, el aplicativo dentrá su ejecución hasta que termine la grabación, en lugar de hacer en background) y en modo "hacia" (límite posterior de la grabación en el tiempo, asumiendo que el inicio de la grabación es el tiempo cero del archivo de sonido), y se le para la estructura respectiva MciRecord. Además, en este último caso se está especificando el DeviceID, para que el API sepa a cuál de los dispositivos MCI se está ordenando la acción.

La principal característica de la programación en C de los comandos MCI, es que debemos tener siempre en cuenta que hay que efectuar manualmente cada apertura, inicialización y liberación de los dispositivos MCI. Si se no se abre un dispositivo y se emite un comando para reproducirlo, se obtendrá un error que no

será visible, excepto por el hecho que no se apreciará la reproducción. El control de errores también debe realizarse manualmente. La prograamción MCI es bastante adecuada para sistemas que hacen un uso promedio de los aditamentos multimedia de una computadora personal, o para aquellos casos en que el dispositivo hardware puede provenir de una variedad enorme de marcas y el desarrollador del aplicativo no puede permitirse programar individualmente cada marca de hardware. El caso típico de esto último es la tarjeta de sonido y el lector de CD de audio.

ANEXO F

PROGRAMAS FUENTE

Programa principal "entiende.cpp"

```
/* Project Entiende
        Copyright © 1996. All Rights
Reserved.

        SUBSYSTEM:    entiende.exe
Application
  FILE:       entndapp.cpp
        AUTHOR:       Glen Rodríguez


        OVERVIEW

        Source file for implementation of
EntiendeApp (TApplication).
*/


#include <owl\owlpch.h>
#include <owl\applicat.h>
#include <owl\window.h>
#include <owl\framewin.h>
#include <owl\dc.h>
#include <owl\color.h>
#include <owl\opensave.h>
#include <owl\scroller.h>
#include <owl\button.h>
#include <owl\checkbox.h>
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <math.h>
// #include <dir.h>
// #include <fstream.h>
#include <classlib\bags.h>
#include <dir.h>
#include <complex.h>

#pragma hdrstop

#include "entndapp.h"
#include "entndabd.h"     // Definition of about
dialog.
#include "usuarios.h"
#include "caracter.h"
#include "multimed.h"
#include "complejo.h"

//{{EntiendeApp Implementation}}

//{{DOC_VIEW}}
DEFINE_DOC_TEMPLATE_CLASS(TFileD
ocument, TWindowView, DocType1);
//{{DOC_VIEW_END}}

//{{DOC_MANAGER}}
DocType1 __dvt1("Sonidos WAV (*.WAV)",
"*.wav", 0, "WAV", dtAutoDelete |
dtUpdateDir);
//{{DOC_MANAGER_END}}


const unsigned long max_samples=88200L;
const int samp_pixel=20;
const int periodo_20ms=221;
unsigned char huge datos_ent[max_samples];
int Estado;
int numentrena;
unsigned char num_bits,mono_stereo;
unsigned int sampling, num_ptos_extremos;
unsigned char maxamplitud,minamplitud;
unsigned long
ini_palabra,fin_palabra,pos_actual;
// char semaforo;
float neutral=124.2;
EstUsuario VarUsuario21;
char path_sistema[200];
complex *vocalico;
float far espectro[50];
char vocales[20];
float vocales_m[5];


// FUNCIONES GENERALES

void marca(unsigned int sample_actual)
{
        if (sample_actual<max_matriz_son)
          {

matriz_sonido[sample_actual].sample=pos_act
ual-ini_palabra;

matriz_sonido[sample_actual].amplitud=datos
```

```
_ent[pos_actual-2];
                    if (sample_actual>0)
matriz_sonido[sample_actual].long_onda=matr
iz_sonido[sample_actual].sample-matriz_sonid
o[sample_actual-1].sample;
                                  else
matriz_sonido[sample_actual].long_onda=0;
                    if
(datos_ent[pos_actual-2]>maxamplitud)
maxamplitud=datos_ent[pos_actual-2];
                    if
(datos_ent[pos_actual-2]<minamplitud)
minamplitud=datos_ent[pos_actual-2];
          };
}

float prom_ult40(char sentido) //sentido 1
hacia adelante,2 atras
{
          unsigned long ci;
          float prom;

          if (sentido==1)
          {
                    if (pos_actual==1)
prom=fabs(datos_ent[0]-neutral);
                     else
                    {
                    if
(pos_actual<=periodo_20ms)
                                  {
                                  prom=0;
                                  for
(ci=1;ci<=pos_actual;ci++) prom=prom +
fabs(datos_ent[ci-1]-neutral);

prom=prom/pos_actual;
                                  }
                    else
                                  {
                                  prom=0;
                                  for
(ci=pos_actual-periodo_20ms+1;ci<=pos_actu
al;ci++) prom=prom +
fabs(datos_ent[ci-1]-neutral);

prom=prom/periodo_20ms;
                                  }
                    }
          }
          else  //sentido es 2
          {
                    if (pos_actual==long_file)
prom=fabs(datos_ent[long_file -1]-neutral);
                     else
                    {
                    if
(pos_actual>=(long_file-periodo_20ms+1))
                                  {
                                  prom=0;
                                  for
(ci=pos_actual;ci<=long_file;ci++)
prom=prom + fabs(datos_ent[ci-1]-neutral);

prom=prom/(long_file -pos_actual +1);
                                  }
                    else
                                  {
                                  prom=0;
                                  for
(ci=pos_actual;ci<=pos_actual+periodo_20ms-
1;ci++) prom=prom +
fabs(datos_ent[ci-1]-neutral);

prom=prom/periodo_20ms;
                                  }
                    }
          }
          return(prom);
}

void GetParametros()
{
          char caracter_final='\n';
          int mode= ios::in | ios::out | ios::app ;
          tolera_inicial=100;

f_parametros.open("parametr.dat",mode);
          if (f_parametros)
                    {

f_parametros.getline(Filewav_Temporal,250,c
aracter_final);

f_parametros.getline(ValorNeutralStr,10,caract
er_final);

f_parametros.getline(Mostrar_GrafStr,7,caract
er_final);

f_parametros.getline(tolera_inicialStr,6,caracte
r  final);

f_parametros.getline(path_sistema,199,caracter
_final);
                    alltrim(ValorNeutralStr);
                    alltrim(Mostrar_GrafStr);
                    alltrim(tolera_inicialStr);
                    alltrim(path_sistema);

ValorNeutral=atof(ValorNeutralStr);
                    neutral=ValorNeutral;
```

```
tolera_inicial=atoi(tolera_inicialStr);
                    if
(strcmp(Mostrar_GrafStr,"TRUE")==0)

Mostrar_Graf=1;
                    else

Mostrar_Graf=0;
                    };
        f_parametros.close();
};

void PutParametros()
{
        int mode= ios::out | ios::trunc ;

f_parametros.open("parametr.dat",mode);

f_parametros.write(Filewav_Temporal,strlen(F
ilewav_Temporal));
        f_parametros.put('\n');

f_parametros.write(ValorNeutralStr,strlen(Val
orNeutralStr));
        f_parametros.put('\n');

f_parametros.write(Mostrar_GrafStr,strlen(Mo
strar_GrafStr));
        f_parametros.put('\n');

f_parametros.write(tolera_inicialStr,strlen(toler
a_inicialStr));
        f_parametros.put('\n');

f_parametros.write(path_sistema,strlen(path_si
stema));
        f_parametros.put('\n');
        f_parametros.close();
};

float suaviza_epa(int x, int y)
{
        float aux;

        if ( ((x>0) & (x<199)) & ((y>0) &
(y<63)) )
                    {

aux=16*PalabActModelo->modelopalabra[x][
y];

aux=aux+PalabActModelo->modelopalabra[x+
1][y+1]+PalabActModelo->modelopalabra[x-1
][y-1]+PalabActModelo->modelopalabra[x-1][
y+1]+PalabActModelo->modelopalabra[x+1][
y-1];

aux=aux+PalabActModelo->modelopalabra[x+
1][y]+PalabActModelo->modelopalabra[x-1][y
]+PalabActModelo->modelopalabra[x][y+1]+P
alabActModelo->modelopalabra[x][y-1];
                    aux=aux/24;
                    }
        else
                    {

aux=PalabActModelo->modelopalabra[x][y];
                    };
        return aux;
};

float suaviza_mpa(int x,int y)
{
        float aux;

        if ( ((x>0) & (x<199)) & ((y>0) &
(y<63)) )
                    {

aux=16*PalabraDataBase->modelopalabra[x][
y];

aux=aux+PalabraDataBase->modelopalabra[x
+1][y+1]+PalabraDataBase->modelopalabra[x-
1][y-1]+PalabraDataBase->modelopalabra[x-1
][y+1]+PalabraDataBase->modelopalabra[x+1]
[y-1];

aux=aux+PalabraDataBase->modelopalabra[x
+1][y]+PalabraDataBase->modelopalabra[x-1]
[y]+PalabraDataBase->modelopalabra[x][y+1]
+PalabraDataBase->modelopalabra[x][y-1];
                    aux=aux/24;
                    }
        else
                    {

aux=PalabraDataBase->modelopalabra[x][y];
                    };
        return aux;
};

float suaviza_epl(int x,int y)
{
        float aux;

        if ( ((x>0) & (x<199)) & ((y>0) &
(y<63)) )
                    {
                    //
aux=16*PalabActModelo->modlongonda[x][y]
```

```
aux=8*PalabActModelo->modlongonda[x][y];

aux=aux+PalabActModelo->modlongonda[x+
1][y+1]+PalabActModelo->modlongonda[x-1]
[y-1]+PalabActModelo->modlongonda[x-1][y
+1]+PalabActModelo->modlongonda[x+1][y-
1];

aux=aux+PalabActModelo->modlongonda[x+
1][y]+PalabActModelo->modlongonda[x-1][y]
+PalabActModelo->modlongonda[x][y+1]+Pal
abActModelo->modlongonda[x][y-1];
                    // aux=aux/24;
                     aux=aux/16;
                    }
          else
                    {

aux=PalabActModelo->modlongonda[x][y];
                    };
          return aux;
};

float suaviza_mpl(int x,int y)
{
          float aux;

          if ( ((x>0) & (x<199)) & ((y>0) &
(y<63)) )
                    {
                     //
aux=16*PalabraDataBase->modlongonda[x][y
];

aux=8*PalabraDataBase->modlongonda[x][y];

aux=aux+PalabraDataBase->modlongonda[x+
1][y+1]+PalabraDataBase->modlongonda[x-1]
[y-1]+PalabraDataBase->modlongonda[x-1][y
+1]+PalabraDataBase->modlongonda[x+1][y-
1];

aux=aux+PalabraDataBase->modlongonda[x+
1][y]+PalabraDataBase->modlongonda[x-1][y]
+PalabraDataBase->modlongonda[x][y+1]+Pal
abraDataBase->modlongonda[x][y-1];
                    // aux=aux/24;
                     aux=aux/16;
                    }
          else
                    {

aux=PalabraDataBase->modlongonda[x][y];
                    };
          return aux;
};

int max_ult20(int indice)
{
 int k;
 unsigned char valor_ampl;
 unsigned int posicion;


valor_ampl=matriz_sonido[indice-19].amplitu
d;
 posicion=0;
 for (k=1;k<19;k++)
          {if
(matriz_sonido[indice-19+k].amplitud >
valor_ampl)
          {

valor_ampl=matriz_sonido[indice-19+k].ampli
tud;
           posicion=k;
          };
          };
 posicion=posicion+indice;
 return posicion;
};


//
// Clase para la ventana con resultados y
graficos
//


TBaseWindow::TBaseWindow() :
TWindow(0,0,0)
{
          strcpy(Encabezado,"Esperando
mandato");
          Attr.Style |= WS_HSCROLL;
          Scroller = new TScroller(this, 1, 1, 0,
0);
};

void TBaseWindow::Paint(TDC& dc, BOOL,
TRect&)
{
          unsigned long i,aux;
          TColor color1;
          TColor color_negro=TColor(0,0,0);
          TColor color_azul =TColor(0,0,255);
          TColor color_rojo =TColor(255,0,0);
          switch (Estado)
          {
           case 0 : dc.TextOut(TPoint(5, 5),
```

```
Encabezado); break;
        case 1 : dc.TextOut(TPoint(5, 5),
Encabezado); break;
        case 2 : dc.TextOut(TPoint(5, 5),
Encabezado); break;
        case 3 : dc.TextOut(TPoint(5, 5),
Encabezado);
                                char
contdisplay[4];
                                float
pordiez;
                                for (i=0;
i<long_file; i++)
                                {
                                 if (
((ini_palabra-1)<=i) && (i<=(fin_palabra-1)) )

color1=dc.SetPixel(int(i/samp_pixel),286 -
datos_ent[i] ,color_negro);
                                 else

color1=dc.SetPixel(int(i/samp_pixel),286 -
datos_ent[i] ,color_azul);
                                };
                                for (i=0;
i<num_ptos_extremos; i++)
                                {
                                 aux =
matriz_sonido[i].sample + ini_palabra -1;

color1=dc.SetPixel(int(aux/samp_pixel),286 -
matriz_sonido[i].amplitud ,color_rojo);

color1=dc.SetPixel(int(aux/samp_pixel)-1,286
- matriz_sonido[i].amplitud ,color_rojo);

color1=dc.SetPixel(int(aux/samp_pixel)+1,286
- matriz_sonido[i].amplitud ,color_rojo);

color1=dc.SetPixel(int(aux/samp_pixel),287 -
matriz_sonido[i].amplitud ,color_rojo);

color1=dc.SetPixel(int(aux/samp_pixel),285 -
matriz_sonido[i].amplitud ,color_rojo);
                                };
                                for
(i=2205;i<long_file;i=i+2205)
                                {
                                 char
decimal[2];

pordiez=i/22050;

gcvt(pordiez,1,contdisplay);
                                 pordiez=(i
% 22050)/2205;

gcvt(pordiez,1,decimal);

strcat(contdisplay,".");

strcat(contdisplay,decimal);

dc.TextOut(TPoint(int(i/samp_pixel)-9,300),co
ntdisplay);
                                        };
                                        break;
        default: dc.TextOut(TPoint(5, 5),
Encabezado);
        };
};

void TBaseWindow::AdjustScroller()
{
 TRect  clientRect = GetClientRect();

        TPoint
Range(Max((long_file/samp_pixel)-clientRect.
Width(), 0),

Max(320-clientRect.Height(), 0));
        Scroller->SetRange(Range.x,
Range.y);

 Scroller->ScrollTo(0, 0);
 if (!GetUpdateRect(clientRect, FALSE))
        Invalidate(FALSE);
}

void TBaseWindow::EvSize(UINT SizeType,
TSize& Size)
{
 TWindow::EvSize(SizeType, Size);
 if (SizeType != SIZEICONIC) {
        AdjustScroller();
        Invalidate(FALSE);
 }
}

//
//Clase dialogo usuario 1 y procesos de apoyo
1+2
//

void GetUsuarios()
{
        char caracter_final='\n';
        int sacados1, sacados2;
        num_usuarios=0;
        int mode= ios::in | ios::out | ios::app ;
        f_usuarios.open("usuarios.dat",mode);
        if (f_usuarios)
```

```
                    {
                     while (!(f_usuarios.eof()))
                      {

f_usuarios.get(arr_usuarios[num_usuarios].cod
igo,6,caracter_final);

alltrim(arr_usuarios[num_usuarios].codigo);

sacados1=f_usuarios.gcount();

f_usuarios.get(arr_usuarios[num_usuarios].no
mbre,21,caracter_final);

alltrim(arr_usuarios[num_usuarios].nombre);

sacados2=f_usuarios.gcount();
                     if ((sacados1>0) &&
(sacados2>0)) {num_usuarios++;};
                      };
                    };
          f_usuarios.close();
}

void PutUsuarios()
{
          int i;
          int mode= ios::out | ios::trunc ;
          f_usuarios.open("usuarios.dat",mode);
          for (i=0;i<num_usuarios;i++)
                    {

f_usuarios.write(arr_usuarios[i].codigo,5);

f_usuarios.write(arr_usuarios[i].nombre,20);
                    }
          f_usuarios.close();
};


void GetPalabras()
{
          int sacados1,sacados2,sacados3;
          char palabrastrord[10];
          char archivo[200];
          char caracter_final='\n';
          UltPalabraUsuario=0;
          int mode= ios::in | ios::out | ios::app ;
          strcpy(archivo,path_sistema);
          strcat(archivo,"\\");
          strcat(archivo,UserActivo.codigo);
          strcat(archivo,"\\palabras.dat");
          f_palabras.open(archivo,mode);
          if (f_palabras)
                    {
                    while (!(f_palabras.eof()))
                              {

f_palabras.get(arr_palabras[UltPalabraUsuario]
.palabraid,15,caracter_final);

alltrim(arr_palabras[UltPalabraUsuario].palabr
aid);

sacados1=f_palabras.gcount();

f_palabras.get(arr_palabras[UltPalabraUsuario]
.palabracomm,30,caracter_final);

alltrim(arr_palabras[UltPalabraUsuario].palabr
acomm);

sacados2=f_palabras.gcount();

f_palabras.get(palabrastrord,10,caracter_final);
                              alltrim(palabrastrord);

arr_palabras[UltPalabraUsuario].palabraord=at
oi(palabrastrord);

sacados3=f_palabras.gcount();

arr_palabras[UltPalabraUsuario].puntuacion=0

                              if ((sacados1>0) &&
(sacados2>0) && (sacados3>0))
{UltPalabraUsuario++;};
                              };
                    };
          f_palabras.close();
};

void AppendPalabras()
{
          int dig,sign;
          int mode= ios::out | ios::app ;
          char archivo[200];
          char *numstring;

          strcpy(archivo,path_sistema);
          strcat(archivo,"\\");
          strcat(archivo,UserActivo.codigo);
          strcat(archivo,"\\palabras.dat");
          f_palabras.open(archivo,mode);
          f_palabras.write(dpalabra,14);
          f_palabras.write(daccion,29);


strcpy(arr_palabras[UltPalabraUsuario].palabra
id,dpalabra);

strcpy(arr_palabras[UltPalabraUsuario].palabra
```

```
comm,daccion);

arr_palabras[UltPalabraUsuario].puntuacion=0


numstring=fcvt(UltPalabraUsuario,0,&dig,&si
gn);
        f_palabras.write(numstring,9);
        f_palabras.close();

        FILE *stream;
        strcpy(archivo,path_sistema);
        strcat(archivo,"\\");
        strcat(archivo,UserActivo.codigo);
        strcat(archivo,"\\");

strcat(archivo,arr_palabras[UltPalabraUsuario].
palabraid);
        strcat(archivo,".pal");
        stream = fopen(archivo, "wb");
        fwrite(PalabraDataBase,
sizeof(*PalabraDataBase), 1, stream);
        fwrite(vocales_m, sizeof(vocales_m),
1, stream);
        UltPalabraUsuario++;
        fclose(stream);
};

void LeeModeloPalabra(char filepalabra[8])
{
        FILE *stream;
        char archivo[200];

        strcpy(archivo,path_sistema);
        strcat(archivo,"\\");
        strcat(archivo,UserActivo.codigo);
        strcat(archivo,"\\");
        strcat(archivo,filepalabra);
        strcat(archivo,".pal");
        stream = fopen(archivo, "rb");


fread(PalabraDataBase,sizeof(*PalabraDataBas
e),1,stream);

fread(vocales_m,sizeof(vocales_m),1,stream);
        fclose(stream);
}

void AgregaEntrenaAModelo()
{
 int i,j,k;
 int x,y,z;
 float factorlongitud,factoramplitud;

largo_palabras[PalabraDataBase->numrepetpal
abra]=fin_palabra-ini_palabra;
 PalabraDataBase->numrepetpalabra++;
 if (PalabraDataBase->numrepetpalabra    1)
        {

PalabraDataBase->longbasepalab=fin_palabra-
ini_palabra;
        factorlongitud=1;
        factoramplitud=1;

PalabraDataBase->longprompalabra=fin_palab
ra-ini_palabra;

PalabraDataBase->amplmaxprompal=maxampl
itud-minamplitud;

PalabraDataBase->amplmaxbase=maxamplitud
-minamplitud;
        for (j=0;j<200;j++)
                {
                 for (k=0;k<64;k++)
PalabraDataBase->modelopalabra[j][k]=0;
                 for (k=0;k<17;k++)
PalabraDataBase->modlongonda[j][k]=0;
                };
        }
 else
        {

PalabraDataBase->longprompalabra=PalabraD
ataBase->longprompalabra+
fin_palabra-ini_palabra;

PalabraDataBase->amplmaxprompal=PalabraD
ataBase->amplmaxprompal+maxamplitud-min
amplitud;

factorlongitud=(PalabraDataBase->longbasepal
ab)/(fin_palabra-ini_palabra);

factoramplitud=(PalabraDataBase->amplmaxb
ase)/(maxamplitud-minamplitud);
        };
 for (i=0;i<num_ptos_extremos;i++)
        {

x=int(factorlongitud*matriz_sonido[i].sample /
220);
//          y=int(matriz_sonido[i].amplitud / 4);

y=int(((factoramplitud*(matriz_sonido[i].ampli
tud-neutral))+neutral) / 4);

PalabraDataBase->modelopalabra[x][y]=(Pala
braDataBase->modelopalabra[x][y])+1;
        if ((matriz_sonido[i].long_onda>0) &
```

```
(matriz_sonido[i].long_onda<101))
                    {

z=rango(matriz_sonido[i].long_onda);

PalabraDataBase->modlongonda[x][z]=(Palabr
aDataBase->modlongonda[x][z])+1;
                    };
          };
};


TDUsuarios1::TDUsuarios1(TWindow*
parent, TResId resId)
          : TDialog(parent, resId)
{
 ListaUsuarios= new
TListBox(this,IDC_ListBox, 34, 32, 100, 124);
 // el IDCListBox= 101 es el id del listbox
};

TDUsuarios1::~TDUsuarios1()
{
          ListaUsuarios->Destroy();
          delete ListaUsuarios;
};

void TDUsuarios1::SetupWindow()
{
          int i;
          TDialog::SetupWindow();

          GetUsuarios();
          ListaUsuarios->ClearList();
          for (i=0; i<num_usuarios; i++)

ListaUsuarios->InsertString(arr_usuarios[i].no
mbre,-1);
};

LRESULT TDUsuarios1::EvCommand(UINT
id, HWND hWndCtl, UINT notifyCode)
{
 if ((id==IDC_ListBox) &&
(notifyCode==LBN_SELCHANGE))
          {

ListaUsuarios->GetSelString(UserTemporal.no
mbre,20);

strcpy(UserTemporal.codigo,arr_usuarios[Lista
Usuarios->GetSelIndex()].codigo);

strcpy(UserTemporal.nombre,arr_usuarios[List
aUsuarios->GetSelIndex()].nombre);

SetDlgItemText(IDUSERIDTXT,UserTempor
al.codigo);
          };

 return TDialog::EvCommand(id, hWndCtl,
notifyCode);
}

// Clase dialogo usuario 2

TDUsuarios2::TDUsuarios2(TWindow*
parent, TResId resId)
          : TDialog(parent, resId)
{
 ListaUsuarios= new
TListBox(this,IDC_ListBox, 34, 32, 100, 124);
 // el IDCListBox= 101 es el id del listbox
 Boton1 =new TButton(this,IDC_Editar);
 Boton2 =new TButton(this,IDC_Borrar);
 Boton3 =new TButton(this,IDC_Agregar);
};

TDUsuarios2::~TDUsuarios2()
{
          PutUsuarios();
          ListaUsuarios->Destroy();
          delete ListaUsuarios;
          delete Boton1;
          delete Boton2;
          delete Boton3;
};

void TDUsuarios2::SetupWindow()
{
          int i;
          TDialog::SetupWindow();

          GetUsuarios();
          ListaUsuarios->ClearList();
          for (i=0; i<num_usuarios; i++)

ListaUsuarios->InsertString(arr_usuarios[i].no
mbre,-1);
          Boton1->EnableWindow(0);
          Boton2->EnableWindow(0);
          Boton3->EnableWindow(1);
};

LRESULT TDUsuarios2::EvCommand(UINT
id, HWND hWndCtl, UINT notifyCode)
{
 int ubic_item;

 if ((id==IDC_ListBox) &&
(notifyCode==LBN_SELCHANGE))
          {
```

```
strcpy(UserTemporal.codigo,arr_usuarios[Lista
Usuarios->GetSelIndex()].codigo);

strcpy(UserTemporal.nombre,arr_usuarios[List
aUsuarios->GetSelIndex()].nombre);

SetDlgItemText(IDUSERIDTXT,UserTempor
al.codigo);
            Boton1->EnableWindow(1);
            Boton2->EnableWindow(1);
           };

 if ((id==IDC_Editar) &&
(notifyCode    BN_CLICKED))
           {
            TDUsuarios2_1* Dialogo_2_1 =
new TDUsuarios2_1(this, D_USUARIO2_1,
"Edite si hay cambios en el nombre. Haga OK
para confirmar o Cancel para anular el
cambio.",1);
           if (Dialogo_2_1->Execute() ==
IDOK) {


ubic_item=ListaUsuarios->GetSelIndex();


arr_usuarios[ubic_item]=UserTemporal;


ListaUsuarios->DeleteString(ubic_item);


ListaUsuarios->InsertString(UserTemporal.no
mbre,ubic  item);

                    Boton1->EnableWindow(0);

                    Boton2->EnableWindow(0);


           };
           Dialogo_2_1->Destroy();
           delete Dialogo_2_1;
           };

 if ((id==IDC_Borrar) &&
(notifyCode    BN_CLICKED))
           {
           TDUsuarios2_1* Dialogo_2_1 =
new TDUsuarios2_1(this,
D_USUARIO2_1,"¿Desea eliminar este
usuario? Haga OK para confirmar o Cancel
para anular la eliminación.",2);
           if (Dialogo_2_1->Execute() ==
IDOK) {


                    int i;


ubic_item=ListaUsuarios->GetSelIndex();


                    for
(i=ubic_item;i<num_usuarios-1;i++)


arr_usuarios[i]=arr_usuarios[i+1];


                    num_usuarios--;


ListaUsuarios->DeleteString(ubic_item);


                    Boton1->EnableWindow(0);


                    Boton2->EnableWindow(0);


                    };
           Dialogo_2_1->Destroy();
           delete Dialogo_2_1;
           };

 if ((id==IDC_Agregar) &&
(notifyCode    BN_CLICKED))
           {
           TDUsuarios2_1* Dialogo_2_1 =
new TDUsuarios2_1(this,
D_USUARIO2_1,"Introduzca un nuevo UserId
y su nombre. Haga OK para confirmar o
Cancel para anular al nuevo usuario.",3);
           if (Dialogo_2_1->Execute() ==
IDOK) {


arr_usuarios[num_usuarios]=UserTemporal;
```

```
                num_usuarios++;


ListaUsuarios->InsertString(UserTemporal.no
mbre,-1);


mkdir(UserTemporal.codigo);


          };
          Dialogo_2_1->Destroy();
          delete Dialogo_2_1;
         };

 return TDialog::EvCommand(id, hWndCtl,
notifyCode);
}

// Clase Dialogo 2.1

TDUsuarios2_1::TDUsuarios2_1(TWindow*
parent, TResId resId, char mensaje[200], char
tipo1)
        : TDialog(parent, resId)
{
          strcpy(advertencia,mensaje);
          tipo2=tipo1;
          Campo1 =new
TEdit(this,IDC_EDIT1,6);
          Campo2 =new
TEdit(this,IDC_EDIT2,21);
};

TDUsuarios2_1::~TDUsuarios2_1()
{
        delete Campo1;
        delete Campo2;
};


void TDUsuarios2_1::SetupWindow()
{
        TDialog::SetupWindow();

SetDlgItemText(ID_ADVERTENCIA,adverte
ncia);
        VarUsuario21=UserTemporal;
        if (tipo2!=3)
        {

SetDlgItemText(IDC_EDIT1,VarUsuario21.co
digo);

SetDlgItemText(IDC_EDIT2,VarUsuario21.no
mbre);
        }
        else
        {
         SetDlgItemText(IDC_EDIT1,"");
         SetDlgItemText(IDC_EDIT2,"");
        };
        switch (tipo2)
        {
        //EDITAR
         case 1: Campo1->SetReadOnly(1) ;

Campo2->SetReadOnly(0) ;break;
        //BORRAR
         case 2: Campo1->SetReadOnly(1) ;

Campo2->SetReadOnly(1) ;break;
        //AGREGAR
         case 3: Campo1->SetReadOnly(0) ;

Campo2->SetReadOnly(0) ;break;
        };
};

LRESULT
TDUsuarios2_1::EvCommand(UINT id,
HWND hWndCtl, UINT notifyCode)
{
 if ((id==IDOK) &&
(notifyCode==BN_CLICKED))
          {

GetDlgItemText(IDC_EDIT1,VarUsuario21.co
digo,6);

GetDlgItemText(IDC_EDIT2,VarUsuario21.n
ombre,21);
           alltrim(VarUsuario21.codigo);
           alltrim(VarUsuario21.nombre);
           UserTemporal=VarUsuario21;
          };

 return TDialog::EvCommand(id, hWndCtl,
notifyCode);
}


// Clase TDPalabras1 = Palabras 1

TDPalabras1::TDPalabras1(TWindow* parent,
TResId resId)
        : TDialog(parent, resId)
{
```

```
};

TDPalabras1::~TDPalabras1()
{
};

LRESULT TDPalabras1::EvCommand(UINT
id, HWND hWndCtl, UINT notifyCode)
{
 if ((id==IDOK) &&
(notifyCode==BN_CLICKED))
          {

GetDlgItemText(IDC_IDPALABRA,dpalabra,
15);

GetDlgItemText(IDC_ACCION,daccion,30);
          numentrena=
GetDlgItemInt(IDC_NUMENTRENA);
          alltrim(dpalabra);
          alltrim(daccion);
          };

 return TDialog::EvCommand(id, hWndCtl,
notifyCode);
}

// Clase Opciones

TDOpciones::TDOpciones(TWindow* parent,
TResId resId)
        : TDialog(parent, resId)
{
        Chequeo1=new
TCheckBox(this,4091,"Mostrar
Gráficos",5,10,200,26,0);
};

TDOpciones::~TDOpciones()
{
        delete Chequeo1;
};

void TDOpciones::SetupWindow()
{

SetDlgItemText(ID_INPUTTEMPORAL,File
wav_Temporal);

SetDlgItemText(IDC_VALORCERO,ValorNe
utralStr);
        if (Mostrar_Graf==1)
                Chequeo1->Check();
                else Chequeo1->Uncheck();
};

LRESULT TDOpciones::EvCommand(UINT
id, HWND hWndCtl, UINT notifyCode)
{

 if ((id==IDOK) &&
(notifyCode==BN_CLICKED))
          {
          char str_a_dec[8];
          UINT estado_check;

GetDlgItemText(ID_INPUTTEMPORAL,File
wav_Temporal,250);

GetDlgItemText(IDC_VALORCERO,str_a_de
c,8);
          neutral=atof(str_a_dec);
          ValorNeutral=neutral;
          strcpy(ValorNeutralStr,str_a_dec);

estado_check=Chequeo1->GetCheck();
          if (estado_check==BF_CHECKED)
                {
                 Mostrar_Graf=1;

strcpy(Mostrar_GrafStr,"TRUE");
                }
                else
                {
                 Mostrar_Graf=0;

strcpy(Mostrar_GrafStr,"FALSE");
                }
          };

 return TDialog::EvCommand(id, hWndCtl,
notifyCode);
}

// Clase Dialogo Puntuacion

TDPuntuacion::TDPuntuacion(TWindow*
parent, TResId resId)
        : TDialog(parent, resId)
{
 ListaPuntuacion= new TListBox(this,101, 30,
28, 220, 145);
}

TDPuntuacion::~TDPuntuacion()
{
        ListaPuntuacion->Destroy();
        delete ListaPuntuacion;
}

void TDPuntuacion::SetupWindow()
{
```

```
        int i,j,temp;
        char texto[25];
        char str[20];
        int arr_reordenado[50];
        TDialog::SetupWindow();

        ListaPuntuacion->ClearList();
        for (i=0; i<UltPalabraUsuario; i++)
        {
         arr_reordenado[i]=i;
        };
        for (i=0; i<UltPalabraUsuario; i++)
        {
         for (j=i+1; j<UltPalabraUsuario; j++)
         {
         if
(arr_palabras[arr_reordenado[i]].puntuacion>a
rr_palabras[arr_reordenado[j]].puntuacion)
          {
                temp=arr_reordenado[i];

arr_reordenado[i]=arr_reordenado[j];
                arr_reordenado[j]=temp;
          };
         }
        };

        for (i=0; i<UltPalabraUsuario; i++)
        {

strcpy(texto,arr_palabras[arr_reordenado[i]].pa
labraid);
          strcat(texto,"   ");

sprintf(str,"%.4f",arr_palabras[arr_reordenado[
i]].puntuacion);
          strcat(texto,str);

ListaPuntuacion->InsertString(texto,-1);
        };
}

LRESULT TDPuntuacion::EvCommand(UINT
id, HWND hWndCtl, UINT notifyCode)
{
 return TDialog::EvCommand(id, hWndCtl,
notifyCode);
}


//
// Clase para la ventana principal de comandos
y hints
//

TMainWindow::TMainWindow(TWindow*
VentTrabajo) : TDecoratedFrame(0,
"Entiende" , VentTrabajo , TRUE)
{
        Attr.X=0;
        Attr.Y=0;
        Attr.W=640;
        Attr.H=439;
};


//////////////////////////////////////////////////
// EntiendeApp
// =======
//
DEFINE_RESPONSE_TABLE1(TBaseWindo
w, TWindow)
        EV_WM_SIZE,
END_RESPONSE_TABLE;

DEFINE_RESPONSE_TABLE1(EntiendeApp
, TApplication)
        EV_COMMAND(CM_RECVOZ,
CmRecVoz),
        EV_COMMAND(CM_RECFILE,
CmRecFile),

EV_COMMAND(CM_USUARIOCAMBIAR
_ACTIVO, CmCambiaUsActivo),

EV_COMMAND(CM_USUARIOEDITAR_U
SUARIOS, CmEditaUsuario),

EV_COMMAND(CM_HELPABOUT,
CmHelpAbout),

EV_COMMAND(CM_VOCABULARIOANA
DIR_PALABRA, CmAnadirPalabra),

EV_COMMAND(CM_VOCABULARIOREE
NTRENAR_PALABRA, CmReentPalabra),

EV_COMMAND(CM_VOCABULARIOEDIT
AR_PALABRA, CmEditarPalabra),
        EV_COMMAND(CM_OPCIONES,
CmOpciones),
END_RESPONSE_TABLE;


EntiendeApp::EntiendeApp () :
TApplication("Entiende")
{

//        DocManager = new
TDocManager(dmSDI | dmMenu);

        // INSERT>> Your constructor code
```

```
here.

        Estado=0; //estado 0 es inactividad
        long_file=0;
        GetParametros();
        PalabraDataBase= new
ConceptoPalabra;
}


EntiendeApp::~EntiendeApp ()
{
        // INSERT>> Your destructor code
here.
        delete PalabraDataBase;
}


void EntiendeApp::SetupSpeedBar
(TDecoratedFrame *frame)
{
        //
        // Create default toolbar New and
associate toolbar buttons with commands.
        //
        TControlBar* cb = new
TControlBar(frame);
        cb->Insert(*new
TButtonGadget(CM_RECVOZ,
CM_RECVOZ));
        cb->Insert(*new
TButtonGadget(CM_RECFILE,
CM_RECFILE));
        cb->Insert(*new TSeparatorGadget);
        cb->Insert(*new
TButtonGadget(CM_HELPABOUT,
CM_HELPABOUT));

        // Add fly-over help hints.

cb->SetHintMode(TGadgetWindow::EnterHint
s);
        cb->SetCursor(this, CURSOR_1);

        frame->Insert(*cb,
TDecoratedFrame::Top);
        barra_sup=cb;
}


//////////////////////////////////////////////////////
// EntiendeApp
// ========
// Application intialization.
//
void EntiendeApp::InitMainWindow ()
{
        PBaseWindow VentanaBase=new
TBaseWindow();
        VentanaBase->SetCursor(this,
CURSOR_1);
        PMainWindow frame = new
TMainWindow(VentanaBase);

        nCmdShow = nCmdShow !=
SW_SHOWMINIMIZED ?
SW_SHOWNORMAL : nCmdShow;

        //
        // Assign ICON w/ this application.
        //
        frame->SetIcon(this,
IDI_SDIAPPLICATION);

        //
        // Menu associated with window and
accelerator table associated with table.
        //
        frame->AssignMenu(SDI_MENU);

        //
        // Associate with the accelerator
table.
        //
        frame->Attr.AccelTable =
SDI_MENU;

        SetupSpeedBar(frame);

        TStatusBar *sb = new
TStatusBar(frame, TGadget::Recessed,


TStatusBar::CapsLock      |


TStatusBar::NumLock


TStatusBar::ScrollLock


TStatusBar::Overtype);
        sb->SetCursor(this, CURSOR_1);
        frame->Insert(*sb,
TDecoratedFrame::Bottom);
        barra_inf=sb;

        VentDisplay = VentanaBase;
        MainWindow = frame;
}
```

```
//////////////////////////////////////////////////////
// EntiendeApp
// ==========
// Menu Help About entiende.exe command
void EntiendeApp::CmHelpAbout ()
{
        //
        // Show the modal dialog.
        //

EntiendeAboutDlg(MainWindow).Execute();
}


void EntiendeApp::Evalua_archivo()
{
        const float umbral_inicio=2;
        const float umbral_final=2.5;
        unsigned char sample,ant_y;
        unsigned long i;
        unsigned char header[44];
        unsigned int cont_extremos,j;
        char signo, ant_signo,
signo_max,signo_max_ant;
        unsigned char es_vocal;
        float maximoneutral;
        complex valorcomplejo;
        unsigned char maxult20,maxpenul20;

        // inicio cuerpo de evalua_archivo

f_ent.open(file_completo,ios::in|ios::binary);
//f_ent : stream del file de entrada .wav
        if (!f_ent)
                {

MainWindow->MessageBox("Error al tratar de
abrir el archivo .wav","Error", MB  OK);

strcpy(VentDisplay->Encabezado,"Esperando
Mandato");
                Estado=0;
                long_file=0;

VentDisplay->AdjustScroller();
                VentDisplay->Invalidate();

VentDisplay->UpdateWindow();
                }
        else
                {
                for(i=1;i<45;i++)
header[i]=0;
                i=0;
                while ((f_ent.get(sample))
&& (i<44) )
                        {
                        header[i]=sample;
                        i++;
                        }
                num_bits=header[34];
                sampling=header[25]*256 +
header[24];
                mono_stereo=header[22];
                if ((num_bits!=8) ||
(sampling!=22050) || (mono_stereo!=1))
                {

MainWindow->MessageBox("Este programa
solo reconoce archivos de 8 bits/sample, 22050
Hz y Monoaural","Atención!!!", MB_OK);

strcpy(VentDisplay->Encabezado,"Esperando
Mandato");
                        Estado=0;
                        long_file=0;

VentDisplay->AdjustScroller();

VentDisplay->Invalidate();

VentDisplay->UpdateWindow();
                }
                else
                {

long_file=(header[7]*16777216L)+
(header[6]*65536L)+ (header[5]*256L)+
header[4]-36L;
                i=0;
                while (f_ent.get(sample)
&& (i<max_samples))
                {

datos_ent[i]=sample;
                        i++;
                        }
                if ( (i<max_samples) &&
!(i=long_file) )

MainWindow->MessageBox("      Posible
Corrupción del file.\nNo se garantiza la
precisión del reconocimiento.","Advertencia",
MB_OK);
                if (long_file>i)
{long_file=i;
MainWindow->MessageBox("Archivo mayor
a 4 segundos","Advertencia", MB_OK);};
                }
                f_ent.close();
                maxamplitud=0;
```

```
                minamplitud=255;

VentDisplay->SetCursor(this, CURSOR_2);
                barra_sup->SetCursor(this,
CURSOR_2);
                barra_inf->SetCursor(this,
CURSOR_2);
                for
(pos_actual=tolera_inicial;pos_actual<=long_fi
le;pos_actual+=40)
                        {
                         if
(prom_ult40(1)>=umbral_inicio)
{ini_palabra=pos_actual;


pos_actual=long_file+1;};
                        };
                for
(pos_actual=long_file-100;pos_actual>=60;pos
_actual-=40)
                        {
                         if
(prom_ult40(2)>=umbral_final)
{fin_palabra=pos_actual;


pos_actual=59;};
                        };

ant_y=datos_ent[ini_palabra-2]; //ini-1
                ant_signo=0;
                signo=0;
                cont_extremos=0;
                for
(pos_actual=ini_palabra;pos_actual<=fin_pala
bra;pos_actual++)
                        {
                         if
(datos_ent[pos_actual-1]>ant_y) signo=1;
                                 else {if
(datos_ent[pos_actual-1]==ant_y)
signo=ant_signo; else signo=-1;};
                        if ( (signo!=0) &&
(signo!=ant_signo) )
                                {
                                 if
((datos_ent[pos_actual-1]-neutral>5) ||
(prom_ult40(1)>=umbral_inicio))
                                        {

marca(cont_extremos);

cont_extremos++;
                                        };
                                };
ant_y=datos_ent[pos_actual-1];
                                ant_signo=signo;
                                };

num_ptos_extremos=cont_extremos;

                es_vocal=0;
                strcpy(vocales,"");
                maxult20=126;
                maxpenul20=126;
                signo_max=0;
                signo_max_ant=0;
                for
(j=19;j<num_ptos_extremos;j++)
                {

maximoneutral=matriz_sonido[j].amplitud-neu
tral;

maxult20=matriz_sonido[max_ult20(j)].amplit
ud;
                 if (maxult20=maxpenul20)
signo_max=signo_max_ant;
                        else
maxult20>maxpenul20 ? signo_max=1 :
signo_max=-1;
                if
((maximoneutral>0.5*(maxamplitud-neutral))
&& (es_vocal==0) &&
                (signo_max=-1) &&
(signo_max_ant=1) &&
(ini_palabra+matriz_sonido[j].sample+512<fin
_palabra))
                {
                        es_vocal=1;
                        for (i=0;i<512;i++)
                         {
                          buffer[i]=0;

buffer[i]=datos_ent[ini_palabra+matriz_sonido
[j].sample+i];

buffer_DFT[i]=complex(0,0);
                                };

vocalico=FFT(buffer,buffer_DFT);

                // FILE *stream_txt;
                // int dig;
                // char complejo_str[25];
                // dig=15;
                // stream_txt =
fopen("d:\\fft.txt", "wt");
                // for (i=0;i<256;i++)
                //        {
```

```
                    //
valorcomplejo=vocalico[i];
                    //
gcvt(real(valorcomplejo),dig,complejo_str);
                    //
fwrite(complejo_str,strlen(complejo_str),1,stre
am_txt);
                    // fwrite(" , ",sizeof(" ,
"),1,stream  txt);
                    //
gcvt(imag(valorcomplejo),dig,complejo_str);
                    //
fwrite(complejo_str,strlen(complejo_str),1,stre
am_txt);
                    //
fwrite("\n",sizeof("\n"),1,stream_txt);
                    //          };
                    // fclose(stream_txt);

                              for (i=0;i<50;i++)
                              {

valorcomplejo=vocalico[i];
                               espectro[i]=0;

espectro[i]=sqrt(pow(real(valorcomplejo),2)+p
ow(real(valorcomplejo),2)) ;
                              };

strcat(vocales,QueVocalEs(espectro));
                              j=j+50;
                              }; // fin if complejo

maxpenul20=maxult20;

signo_max_ant=signo_max;
                              if
((maximoneutral<0.5*(maxamplitud-neutral))
&& (es_vocal==1) && (maximoneutral>0))
es_vocal=0;
                    }; // fin for j=1 to
num_ptos_extremos


VentDisplay->SetCursor(this, CURSOR_1);
                    barra_inf->SetCursor(this,
CURSOR_1);
                    barra_sup->SetCursor(this,
CURSOR_1);
                    Estado=3;

VentDisplay->AdjustScroller();
                    VentDisplay->Invalidate();

VentDisplay->UpdateWindow();
                    }
}

void EntiendeApp::Compara_archivo()
{
          int i,j,k,x,y,z,col1,col2,factor_rango;
          float
parecido1,parecido2,parecido3,desv_st_correg;
          // parecido1 y factor1: amplitudes.
parecido2 y factor2: long. de onda

setea_factores(factor1,factor2,factorsilencio,fa
ctor_rango);
          factorlongitud=1;
          factoramplitud=1;
          PalabActModelo= new
ConceptoPalabra;
          for (j=0;j<200;j++)
                    {
                     for (k=0;k<64;k++)
                              {

PalabActModelo->modelopalabra[j][k]=0;
                              };
                    for (k=0;k<17;k++)
                              {

PalabActModelo->modlongonda[j][k]=0;
                              };
                   };

factorlongitud=(PalabraDataBase->longbasepal
ab)/(fin_palabra-ini_palabra);

factoramplitud=(PalabraDataBase->amplmaxb
ase)/(maxamplitud-minamplitud);
          if (factorlongitud<0.7)
factorlongitud=0.7;
          if (factorlongitud>1.3)
factorlongitud=1.3;
          for (j=0;j<num_ptos_extremos;j++)
          {
           x=int(factorlongitud *
matriz_sonido[j].sample / 220);
           y=int((factoramplitud *
(matriz_sonido[j].amplitud-neutral) + neutral) /
4);

PalabActModelo->modelopalabra[x][y]=(Pala
bActModelo->modelopalabra[x][y])+1;

z=rango(matriz_sonido[j].long_onda);

PalabActModelo->modlongonda[x][z]=(Palab
ActModelo->modlongonda[x][z])+1;
          };
```

```
        for (i=0;i<UltPalabraUsuario;i++)
                {

LeeModeloPalabra(arr_palabras[i].palabraid);
                parecido1=0;
                parecido2=0;
                parecido3=0;
                for (j=0;j<200;j++)
                        {
                        for (k=0;k<64;k++)
                                {

parecido1=parecido1+funcCompara(suaviza_e
pa(j,k),suaviza_mpa(j,k),factorlongitud);
                                //
parecido1=parecido1+funcCompara(PalabAct
Modelo->modelopalabra[j][k],PalabraDataBas
e->modelopalabra[j][k],factorlongitud);
                                };
                        col1=0;
                        col2=0;
                        for (k=0;k<17;k++)
                                {

parecido2=parecido2+funcCompara(suaviza_e
pl(j,k),suaviza_mpl(j,k),factorlongitud);
                                //
parecido2=parecido2+funcCompara(PalabAct
Modelo->modlongonda[j][k],PalabraDataBase-
>modlongonda[j][k],factorlongitud);

col1=col1+(PalabActModelo->modlongonda[j]
[k]);

col2=col2+(PalabraDataBase->modlongonda[j
][k]);
                                };
                        if ((col1<=4) &
(col2>4)) parecido3=parecido3+factorsilencio;
                        if ((col2<=4) &
(col1>4)) parecido3=parecido3+factorsilencio;
                        };

arr_palabras[i].puntuacion=(factor1*parecido1
)+(factor2*parecido2)+parecido3-(factor1+fact
or2)*(sqrt(parecido1*parecido2))/4;

arr_palabras[i].puntuacion=arr_palabras[i].pun
tuacion+comparavocales(vocales, vocales_m);

desv_st_correg=PalabraDataBase->desvstanpal
abra;
                // minimo=4%
maximo=10%
                if
(PalabraDataBase->desvstanpalabra >
(0.1*PalabraDataBase->longprompalabra))
desv_st_correg=0.1*PalabraDataBase->longpr
ompalabra;
                        if
(PalabraDataBase->desvstanpalabra <
(0.04*PalabraDataBase->longprompalabra))
desv_st_correg=0.04*PalabraDataBase->longp
rompalabra;
                        if
(fabs(((PalabraDataBase->longprompalabra)-(f
in_palabra-ini_palabra))/3)>desv_st_correg)
arr_palabras[i].puntuacion=factor_rango*arr_p
alabras[i].puntuacion;
                        };
        TDPuntuacion* Dial_Puntuacion=
new
TDPuntuacion(MainWindow,D_PUNTUACIO
N);
        if
(Dial_Puntuacion->Execute()==IDOK)
        {
        };
        delete PalabActModelo;
}

void EntiendeApp::CmRecVoz()
{

strcpy(VentDisplay->Encabezado,"Reconocien
do de Grabación en Vivo");

strcpy(file_completo,Filewav_Temporal);
        Estado=1;     //estado 1 es grabando
        long_file=0;
        VentDisplay->AdjustScroller();
        VentDisplay->Invalidate();
        VentDisplay->UpdateWindow();

GrabarSonido(file_completo,VentDisplay,this,
barra_sup,barra_inf);
        Estado=2;     //estado 2 es cargando
file
        VentDisplay->Invalidate();
        VentDisplay->UpdateWindow();
        Evalua_archivo();
        Compara_archivo();
};

void EntiendeApp::CmRecFile()
{
         char file_escogido[MAXPATH];
         char path_file[MAXPATH];

 static TOpenSaveDialog::TData data (

OFN_HIDEREADONLY|OFN_FILEMUSTE
```

```
XIST|OFN_NOREADONLYRETURN,
            "Archivos .WAV (*.WAV)|*.wav|",
            0,
            ""
            "WAV"
  );
  if (TFileOpenDialog(MainWindow,
data).Execute() == IDOK) {

TOpenSaveDialog::GetFileTitle(data.FileName
, file_escogido, MAXPATH);
            strcpy(path_file,"Reconociendo de ");
            strcat(path_file,data.FileName);

strcpy(VentDisplay->Encabezado,path_file);
            strcpy(file_completo,data.FileName);
            Estado=2;     //estado 2 es cargando
file
            VentDisplay->Invalidate();
            VentDisplay->UpdateWindow();
            Evalua_archivo();
            Compara_archivo();
  }
};

void EntiendeApp::CmEditaUsuario()
{
        TDUsuarios2* Usuarios2 = new
TDUsuarios2(MainWindow, D_USUARIO2);
            Usuarios2->Execute();
            Usuarios2->Destroy();
            delete Usuarios2;
};

void EntiendeApp::CmCambiaUsActivo()
{
            TDUsuarios1* Usuarios1 = new
TDUsuarios1(MainWindow, D_USUARIO1);
            if (Usuarios1->Execute()==IDOK)
                        {
                                    char titulo[32];

UserActivo=UserTemporal;

strcpy(titulo,"Entiende: ");

strcat(titulo,UserActivo.nombre);

MainWindow->SetCaption(titulo);
                                    GetPalabras();
                        };
            Usuarios1->Destroy();
            delete Usuarios1;
};

void EntiendeApp::CmAnadirPalabra()
{
            int i;
            TDPalabras1* D_AnadirPalabra= new
TDPalabras1(MainWindow,D_PALABRAS1);
            if
(D_AnadirPalabra->Execute()==IDOK)
                        {
                        char file_enesimo[50];
                        char convertido[3];
                        char mensaje[30];

PalabraDataBase->longprompalabra=0;

PalabraDataBase->desvstanpalabra=0;

PalabraDataBase->numrepetpalabra=0;

PalabraDataBase->amplmaxprompal=0;

PalabraDataBase->amplmaxbase=0;
                        for (i=0;i<numentrena;i++)
                                    {

strcpy(file_enesimo,path_sistema);

strcat(file_enesimo,"\\");

strcat(file_enesimo,UserActivo.codigo);

strcat(file_enesimo,"\\temp");

itoa(i+1,convertido,10);

strcat(file_enesimo,convertido);

strcat(file_enesimo,".wav");

strcpy(mensaje,"Muestra No.");

strcat(mensaje,convertido);
                                    strcat(mensaje," ");

strcat(mensaje,dpalabra);

MainWindow->MessageBox(mensaje,"Grabaci
ón",MB_OK);

GrabarSonido(file_enesimo,VentDisplay,this,b
arra_sup,barra_inf);

strcpy(file_completo,file_enesimo);
                                    Evalua_archivo();

AgregaEntrenaAModelo();

mezclavocales(vocales_m,vocales,numentrena)
```

```
                        };

PalabraDataBase->longprompalabra=(Palabra
DataBase->longprompalabra)/(PalabraDataBas
e->numrepetpalabra);

PalabraDataBase->amplmaxprompal=(Palabra
DataBase->amplmaxprompal)/(PalabraDataBa
se->numrepetpalabra);
                        if
(PalabraDataBase->numrepetpalabra >1)
                        {
                         int np;
                         for
(np=0;np<PalabraDataBase->numrepetpalabra;
np++)

PalabraDataBase->desvstanpalabra=(largo_pal
abras[np]-(PalabraDataBase->longprompalabra
))*(largo_palabras[np]-(PalabraDataBase->lon
gprompalabra));

PalabraDataBase->desvstanpalabra=sqrt((Palab
raDataBase->desvstanpalabra)/(-1 +
PalabraDataBase->numrepetpalabra));
                        }
                        else
PalabraDataBase->desvstanpalabra=0.75;
                         AppendPalabras();
                        };
          D_AnadirPalabra->Destroy();
          delete D_AnadirPalabra;
};

void EntiendeApp::CmReentPalabra()
{
};

void EntiendeApp::CmEditarPalabra()
{
};

void EntiendeApp::CmOpciones()
{
          TDOpciones* D_Opciones= new
TDOpciones(MainWindow,IDD_DIALOGOS
ETTINGS);
          if (D_Opciones->Execute()==IDOK)
                    {
                     PutParametros();
                    };
          D_Opciones->Destroy();
          delete D_Opciones;
};

void EntiendeApp::InitInstance ()
{
           TApplication::InitInstance();

          // NO Accept files via drag/drop in
the frame window.

MainWindow->DragAcceptFiles(FALSE);
}

// Estados
// 0 inactivo
// 1 grabando en vivo a file temporal
// 2 cargando contenido de file a memoria
+procesado
// 3 displayando grafico de ondas

int OwlMain (int , char* [])
{
          EntiendeApp    App;
          int            result;

          result = App.Run();
          return result;
}
```

BIBLIOGRAFIA

- Análisis de Fourier - Hwei Hsu. (Adison Wesley Iberoamericana, 1987).
- Artículo: Compresión eficiente de la voz. Ings. José Lira, Catherine Castro y Carlos Silva. (Edición XII CONIMERA, 1995).
- Artículo: Reconocimiento de Palabras aisladas a Vocabulario limitado empleando Predicción lineal y Alineamiento Temporal. Ing. Andrés Flores y Dr. Jorge del Carpio. (Edición XII CONIMERA, 1995).

  Sound Blaster: The Official Book - Peter Ridge, David Golden, Ivan Luk y Scott Sindorf (Osborne - Mc Graw Hill, 1994).

  Física: Campos y Ondas - M. Alonso y E. Finn (Fondo Educativo Interamericano).
- Física para las ciencias de la vida. Alan H. Cromer (Editorial Reverté, 1994).
- The Feynman Lectures on Physics, vol.I - R.Feinman, R.Leighton y M.Sands (Adisson Wesley).
- Borland C++ Handbook - Chris Pappas y William Murray (Osborne - McGraw Hill , 1992).
- Diccionario Español Inglés de la Universidad de Chicago - Carlos Castillo y Otto Bond (Pocket Books, 1977).
- Páginas Web sobre Transformada rápida de Fourier y análisis de ondas por Fourier en general (http://www.spd.eee.strath.ac.uk/~interact/fseries.html).
- Páginas Web de la Univ. de Illinois (http://hawaii.cogsci.uiuc.edu./speech.html).
- Páginas Web de la Univ. de Berkeley (http://www.icsi.berkeley.edu/real/real-intro.html).
- Páginas Web de la Univ. de Stanford - Applied Speech Tech. Lab. (http://www.csli.stanford. edu/users/bscott/astl.html).