01149

UNIVERSIDAD NACIONAL AUTONOMA DE MEXICO

FACULTAD DE INGENIERIA

" DISÉÑO DE UN SINTETIZADOR DE VOZ EN ESPAÑOL USANDO EL METODO TD-PSOLA "

T E S I S

QUE PARA OBTENER EL TITULO DE:

MAESTRO EN INGENIERIA

P R E S E N T A :

FERNANDO DEL RIO AVILA

DIRECTOR DE TESIS: DR. ABEL HERRERA CAMACHO

MEXICO, D.F. **MAYO 2005**

El autor desea agradecer el generoso apoyo otorgado
por el CONACYT para la realización de esta tesis.
Beca no. 183674
Sept. 2003 a Julio 2004

Índice

Introducción

El objetivo de esta tesis es desarrollar un sistema de síntesis de voz de vocabulario ilimitado para el idioma español hablado en México. Una de las ventajas de este sistema es la posibilidad de modificar fácilmente la base de datos para sustituir la voz sintetizada. El sistema se presenta con una interfaz, mostrada en la figura I.1, muy cómoda y simple para el usuario y adaptable a diversas aplicaciones. La aportación de este trabajo radica en la combinación del método TD-PSOLA, el cual se rediseñó, y las técnicas de concatenación de difonemas, teniendo como resultado un sintetizador rápido, de bajos requerimientos de memoria y procesador y con buena calidad de audio.

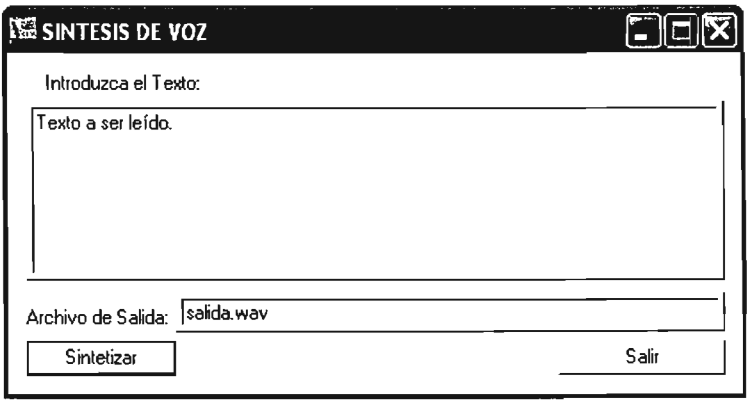

Fig. I.1. Interfaz del sistema de síntesis de voz

Un sistema de síntesis de voz es aquel que genera voz por medios electrónicos. Estos sistemas son también conocidos como sistemas de texto a voz (**TTS**, siglas de las palabras en inglés **T**ext–**T**o-**S**peech). Un sistema de esta índole proporciona otra forma de comunicación entre la computadora y el hombre, siendo de utilidad a minusválidos, invidentes o personas con problemas de comunicación, así como en ambientes de trabajo donde los que laboran ahí tengan su vista ocupada y al mismo tiempo tengan que recibir información del proceso que realizan.

El sistema consta de dos módulos. El primer módulo toma una entrada de texto libre de cualquier longitud, el cual es analizado para generar una salida fonética que indique la forma en que el texto debe ser pronunciado. La salida fonética alimenta al segundo módulo que es el encargado de generar la señal de audio.

El sintetizador de voz es de tipo concatenativo, es decir, toma segmentos de voz previamente grabados que son procesados y unidos entre sí para generar la señal de salida. Los segmentos utilizados son los difonemas, los cuales consisten en tomar dos medios fonemas, desde la parte central del primero hasta la parte central del segundo. Estos fueron seleccionados debido a su numero relativamente pequeño en comparación con otros tipos de segmentos y a que proporcionan una buena calidad de sonido.

Los segmentos son posteriormente procesados por el algoritmo TD-PSOLA, el cual permite modificar sus características de tono y duración directamente en el dominio del tiempo sin necesidad de una parametrización previa de los segmentos. Este algoritmo se utiliza para disminuir las diferencias entre segmentos y mejorar así la calidad del audio de salida.

En el primer capítulo de este trabajo se exponen las características básicas de los sistemas de síntesis de voz y una breve historia del área. En el capítulo 2 se indican las características de algunos sistemas de voz de distribución gratuita. Posteriormente en el capítulo 3 se analiza uno de estos sistemas, Festival, más a detalle para conocer su estructura.

En los capítulos 4 a 6 se detalla el diseño del sistema realizado. En el capítulo 4 se describe la forma en que se generan los segmentos que son utilizados para producir la salida de voz, así como la forma en que se obtienen las marcas de periodo requeridas por el algoritmo PSOLA. En el capítulo 5 se analiza el módulo de conversión de texto a fonemas, así como las reglas requeridas para esta conversión. Finalmente, en el capítulo 6 se describe la conversión de fonemas a voz y se analiza el algoritmo TD-PSOLA, usado para suavizar los puntos de unión y disminuir las discrepancias entre segmentos.

1. Perspectiva general de la síntesis de voz

1.1. Estructura básica de los sintetizadores de voz

Un sintetizador de voz es un aparato o software que genera una salida sonora que simula o imita a la voz humana. Actualmente estos sistemas son muy usados. Por ejemplo, en los sistemas lectores telefónicos de correo electrónico, lectores de libros para ciegos, en películas y música para efectos de sonido y en paquetes para interacción con una computadora.

Estos sistemas constan principalmente de dos bloques [3], que actúan entre sí, para realizar la síntesis de voz, tal como se ilustra en la figura 1.1.

Fig. 1.1. Estructura básica de un sintetizador de voz

El primer bloque toma una entrada de texto y a partir de esta genera una trascripción fonética junto con información sobre duración, entonación, etc. Esta trascripción fonética alimenta al segundo bloque que es el que genera la salida final de voz.

El bloque de procesamiento de lenguaje natural está a su vez dividido en dos bloques de análisis, el de texto y el lingüístico.

El bloque de análisis de texto toma la entrada de texto, a partir del cual se obtiene la salida fonética que será leída.

La parte más importante de este bloque es la normalización del texto. Este proceso consiste en tomar todos aquellos símbolos (números, abreviaciones, etc.) que no indican explícitamente la forma en que deben ser pronunciados y que deben ser expandidos a palabras.

Algunos ejemplos de esta conversión pueden ser:

12 -> doce	12° -> doceavo
1 casa -> una casa	1 coche -> un coche
$10 -> 10 pesos	Ave. -> avenida

Aquí se puede observar que para un mismo símbolo puede haber diferentes formas de leerse, según el contexto en que se encuentra, así como símbolos que se pronuncian en una posición diferente a la escrita.

Una vez que se ha normalizado el texto, se cuenta sólo con palabras que ya tienen una relación directa con su pronunciación. Esta pronunciación se puede obtener por dos métodos básicos[3]:

- Reglas de pronunciación, que almacenan la relación entre símbolos y sonidos. Se debe hacer notar, sin embargo, que la relación exacta es aún un problema abierto y diferentes sistemas ocupan tablas diferentes. Este método será el utilizado en el desarrollo del sistema y su implementación se mostrará en el capítulo 5.

- Tablas de pronunciación, que constan de una lista de palabras y su pronunciación. Estas se utilizan en español normalmente sólo para excepciones, ya que en la mayoría de los casos la pronunciación está basada en las reglas ya mostradas. También son comunes en otros idiomas (como el inglés) donde la relación entre grafías y fonemas no es tan directa.

En el bloque de análisis lingüístico se obtiene la estructura prosódica a partir de la salida fonética previamente obtenida. La estructura prosódica consiste principalmente en la

entonación (que se divide en localización de acentos, la cual es muy sencilla en español ya que se basa en reglas ortográficas bien definidas, y en el cálculo de la variación de la frecuencia fundamental en el tiempo) y en la duración, ritmo y amplitud de las palabras dentro de las frases.

Esta estructura depende normalmente del contexto y la intención, pero debido a la dificultad o imposibilidad de obtenerla a partir de un texto, se utiliza una entonación neutral que se obtiene de análisis estadísticos.

El segundo bloque de la figura 1.1toma la salida del primer bloque y genera la señal de voz. Esta señal puede ser generada por varios métodos. El método más sencillo y el más utilizado actualmente es el concatenativo, el cual consiste en unir segmentos de sonido previamente grabados que son unidos para generar la salida.

Otro método (mostrado en la figura 1.2) es utilizar un modelo entrada-filtro que modela la fuente de sonido y los formantes de la señal de voz sin tomar en cuenta las características físicas que los generan.

En algunos sistemas se mezclan ambos métodos, utilizando el modelo entrada-filtro para reducir el espacio de almacenamiento requerido al almacenar sólo coeficientes, en lugar de una grabación completa de cada segmento.

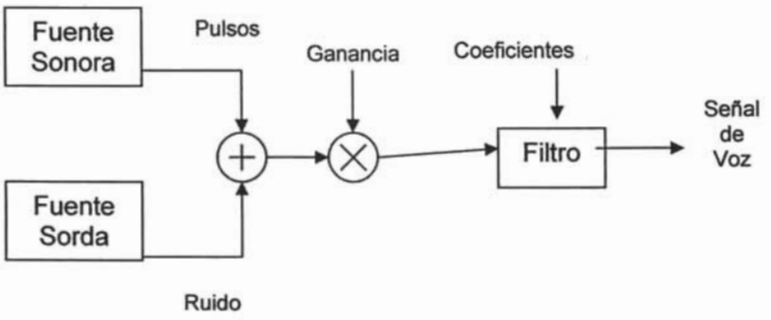

Fig. 1.2. Modelo entrada-filtro[10]

Un tercer método es modelar directamente el sistema productor de voz (articulaciones, cuerdas vocales, etc.). Este sistema en teoría sería el más preciso, pero debido a su extrema complejidad, y a la falta de datos acerca del funcionamiento del sistema de fonación humano, está prácticamente en desuso.

Además, en algunos sistemas actuales se está probando con la generación de una cara que se anima de acuerdo con los fonemas pronunciados. Este sistema parece aumentar la comprensión de la señal de voz, al tener información gráfica adicional además de la información auditiva.

1.2. Historia de la síntesis de voz

1.2.1. Inicios

Aunque desde el siglo XVIII existen aparatos mecánicos que tratan de simular la voz humana mediante medios mecánicos, sólo en la última mitad del siglo XX se volvió posible el hacer sistemas que generaran automáticamente una salida de voz a partir de texto escrito (sistemas de texto a voz).

Aunque los sistemas de síntesis de voz en la actualidad tienen una inteligibilidad muy alta, la calidad del sonido y su naturalidad son aún problemas a resolver. Los sistemas actuales pueden ser usados en muchas aplicaciones donde la naturalidad no es tan importante.

Desde la segunda mitad del siglo XVIII se han construido diversos sistemas capaces de generar una salida de voz de forma artificial, inicialmente mecánicos, luego analógicos y actualmente digitales.

El primer intento registrado fue realizado en 1773 donde Ch. G. Kratzenstein, un profesor de fisiología de Copenhague logró producir sonidos vocálicos a partir de tubos de resonancia conectados a tubos de órgano. [10, 11 y 12]

Simultáneamente a este invento, Wolfgang von Kempelen (Hungaria 1734 – Viena 1804) trabajaba en la construcción de una máquina que pudiera generar sonidos que simularan la voz humana. En 1791 publicó sus descubrimientos y la forma en que se podía construir su máquina de forma que otras personas pudieran continuar su investigación. Esta máquina funcionaba por medio de un fuelle que generaba una salida de aire. Este después era enviado a través de varias tuberías y aberturas que podían ser modificadas manualmente para producir palabras o frases cortas [3, 10 y 11]. En la figura 1.3 se muestra una versión reconstruida de esta maquina.

Durante el siglo XIX se construyeron otros aparatos similares que mejoraron levemente el modelo de Kempelen, al introducir elementos adicionales que simularan la lengua o los cambios en la forma de la boca al hablar.

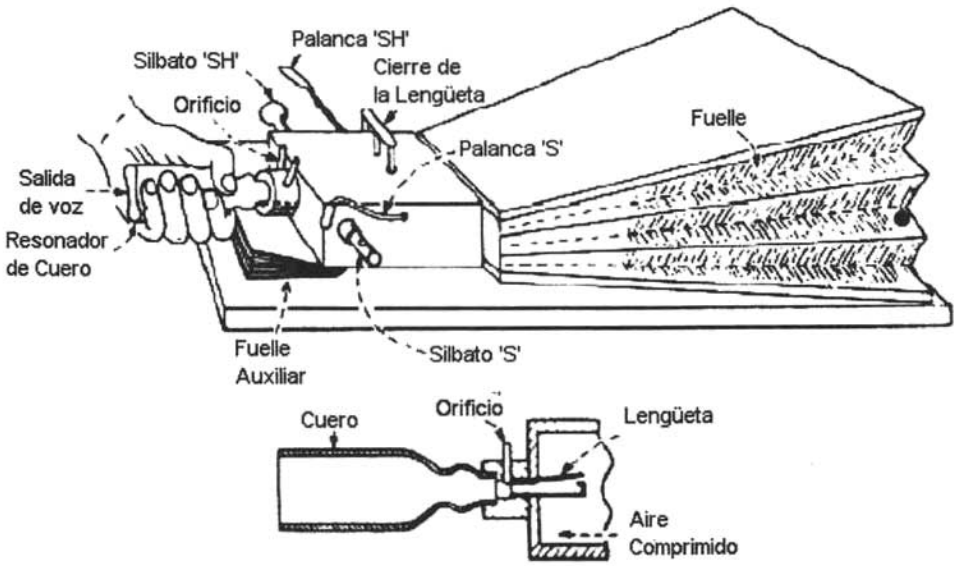

Fig. 1.3. Aparato de von Kempelen [10]

1.2.2. Sintetizadores de voz eléctricos

En 1922, Stewart construyó el primer sistema que intentaba simular la voz mecánica por medios eléctricos [10]. Este dispositivo constaba de dos circuitos resonantes activados por un zumbador (*buzzer)*. Al ser modificadas las frecuencias de resonancia de ambos circuitos se obtenía una señal similar a una vocal.

En la década de los 30's se diseñó el VOCODER en los laboratorios Bell. Este aparato estaba diseñado para analizar la voz humana y obtener parámetros acústicos, a partir de los cuales se podía reconstruir una salida similar a la onda de voz original.

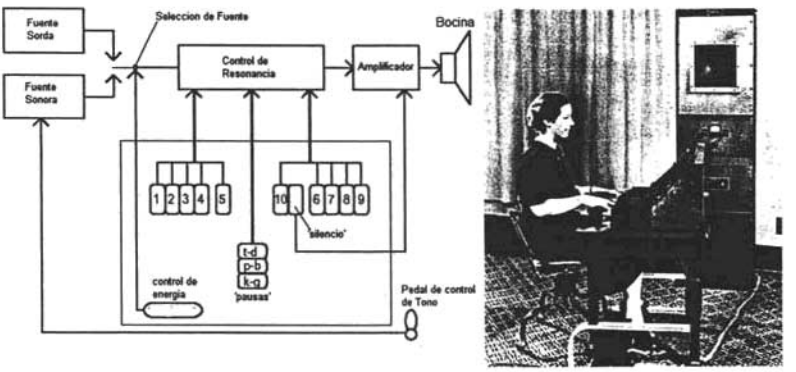

Fig. 1.4. Diagrama del sistema VODER [10] y presentación en la Feria Mundial de Nueva York [12]

A partir de este sistema se construyó una segunda versión que fue mostrada en la Feria Mundial de Nueva York (1939). Este sistema, llamado VODER (mostrado en la figura 1.4) y desarrollado por Homer Dudley generaba una salida de ruido o una salida de audio senoidal de acuerdo con un selector. La frecuencia de esta salida podía ser controlada por un pedal. Esta salida era después filtrada por 10 filtros pasobanda cuya amplitud se modificaba por medio de los dedos. [3,10 y 11]

Aunque la inteligibilidad de este sistema fue mínima, mostró la posibilidad de generar voz por medios eléctricos. Este sistema es la base de la síntesis por formantes. El sistema de

filtros fijos que utiliza el Voder es demasiado limitado para generar las diferentes salidas requeridas, por lo que no se utiliza en los sistemas modernos.

En 1951 se presentó el reproductor de patrones desarrollado en los laboratorios Haskins de la Universidad de Yale. Este utilizaba espectrogramas, los cuales se iluminaban y eran enviados a un conjunto de celdas fotovoltaicas, cada una de las cuales controlaba la intensidad de una onda fundamental de diferentes frecuencias en saltos de 120 Hz, que podían reconstruir aproximadamente la señal del espectrograma. Franklin Cooper, Alvin Liberman, Pierre Delattre y otros asistentes, experimentaron con espectrogramas reales y con adaptaciones dibujadas a mano para evaluar la importancia de diferentes factores. Este sistema generó una inteligibilidad mucho más alta (más de 90% con espectrogramas reales y más de 80% con los espectrogramas dibujados a mano).[13]

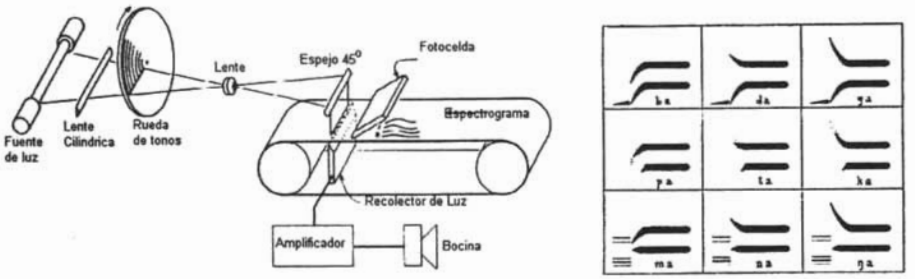

Figura 1.5. Reproductor de patrones y ejemplo de espectros dibujados [13]

Estos dos sistemas funcionaban copiando los patrones espectrales de la voz. Poco después del desarrollo de estos sistemas, se dio un nuevo enfoque a la teoría de síntesis de voz. Este nuevo enfoque fue la generación de una teoría acústica de la forma en que se produce la voz y no sólo en los resultados del proceso. Esta teoría es la base de la síntesis por formantes que se analizará adelante. Según esta teoría, la voz se puede considerar como la salida de un filtro lineal excitado por una o más fuentes, principalmente las cuerdas vocales, y ruido turbulento debido a diferencia de presiones a través de un estrangulamiento [13]. El filtro en este caso, es una simulación de los efectos del tracto vocal. El tracto vocal es simulado por una función de transferencia con pares de polos complejos conjugados que producen los formantes en el espectro de salida. Además de los polos se requiere la

introducción de ceros (antiresonadores) para modelar la absorción de las ramas laterales en algunos fonemas como nasales y fricativos.

En 1953 se crearon los primeros sintetizadores de formantes como el Paramteric Artificial Talker (PAT) construido por Walter Lawrence y el OVE I construido por Gunnar Fant.

El PAT tenía tres resonadores en paralelo. Se tenía una señal de entrada de ruido o periódica y de ahí, a través de patrones dibujados sobre un vidrio que se deslizaba, se controlaban 3 frecuencias del formante, así como las amplitudes del ruido, del fraseo y de la fundamental.

El OVE utilizaba filtros en cascada en lugar de en paralelo. Los dos más bajos eran controlados por movimientos en 2 dimensiones de un brazo mecánico, mientras que la amplitud y la fundamental eran controlados por potenciómetros. Sin embargo, este sistema sólo podía generar vocales.

Es interesante notar que, aunque ambos sistemas parten de la misma teoría y usaban los mismos principios, utilizaron diferentes métodos para llevarla a la práctica, y hasta la fecha aún existe controversia sobre cuál de los dos métodos es mejor, o si la mejor opción es utilizar una mezcla de ambos; teoría propuesta en 1972 por Klatt [13].

Estos sistemas sufrieron varias modificaciones. A PAT se le introdujeron controles individuales para los formantes y un circuito independiente para fricativas, llegando a ser un sistema en cascada. A OVE I se le agregó otra rama estática para simular murmullos nasales, una cascada de dos formantes y un antiformante para simular mejor la función de transferencia del tracto vocal y la excitación de los sonidos fricativos, transformándose en OVE II [10].

Además con mayores o menores modificaciones (por ejemplo, modular la amplitud del ruido en fricativas sonoras o agregar nuevos parámetros) estos dos sistemas continúan siendo la base de los sistemas modernos de síntesis por formantes.

Una de las modificaciones más grandes a esta clase de sistemas fue la introducción de sistemas híbridos (cascada y en paralelo). En el sistema propuesto por Klatt [13] cada sistema se utilizaba para modelar diferentes tipos de sonidos (en paralelo para sonidos sonoros y en cascada para sonidos sordos). Este sistema propuesto por Klatt fue además presentado como un listado en Fortran en 1980, lo que permitió su uso más extendido.

Otro punto importante en la historia de los sintetizadores por formantes ocurrió en una conferencia en Boston en 1972 cuando John Holmes presentó una salida de voz prácticamente indistinguible de una voz natural [13]. Desafortunadamente, esta señal fue generada de forma manual y basada en un proceso de prueba y error de varios meses de duración. Aunque este experimento demostró varios factores importantes en la generación de oraciones, su método no ha podido ser llevado automatizado.

El siguiente avance importante en este tipo de sintetizadores es cambiar el tipo de señal de entrada, de una señal monótona (triangular, tren de pulsos) en una señal que se asemeje más a la señal que entra al tracto vocal.

El primer avance de este tipo se dio en 1975 (Rothenberg), donde se utilizó un sistema de tres parámetros de acuerdo con la apertura de la glotis, la amplitud de la respiración y la frecuencia fundamental. Se han creado métodos que simulan más parámetros, pero hasta la fecha el resultado aún no es completamente natural, debido posiblemente a la falta de conocimiento del modelo real [13].

Aparte de este tipo de sintetizadores, otra línea paralela de investigación es la generación de líneas de transmisión que simulen un tubo similar al tracto vocal. Sin embargo, debido a restricciones en el conocimiento del tracto vocal y en la cantidad de cálculos, se ha avanzado poco en esta área, aunque existen algunos modelos de sintetizadores de este tipo.

Una vez que se obtuvieron sistemas que pudieran simular la voz humana, una aplicación muy importante, que sólo se hizo posible con el advenimiento de computadoras y circuitos integrados, es la generación de una señal de voz a partir de una entrada fonémica o de texto.

El primer programa de este tipo se desarrolló en 1961 (Kelly y Gerstman) con un sintetizador en cascada de tres formantes, cuyos parámetros posteriormente se modificaban a mano[13].

En 1964 apareció otro sistema (Holmes) que funcionaba a partir de síntesis por formantes en paralelo y un conjunto de tablas que permitía generar resultados más complejos como coarticulación y el uso de alófonos. En 1966 (Mattingly) [14] modificó el programa para generar transiciones más realistas, pero obtuvo poca mejoría. El primer uso práctico que se le intentó dar a este sistema, fue una adaptación como parte de una máquina de lectura para ciegos, pero nunca se concretó por falta de recursos.

A finales de los 60's y principio de los 70's se continuó la investigación de los sistemas de síntesis por regla, ajustando diferentes parámetros para hacerlos más similares a la voz natural, principalmente por Klatt [13], dando como resultado el sistema de síntesis del M.I.T. Este sistema conocido como MITalk (1976) fue vendido y cambió de manos varias veces durante los siguientes años. Después de este sistema se desarrolló el Klattalk que continuó siendo mejorado hasta finales de los 80's.

Otro importante desarrollo fue la introducción de sistemas dentro de un circuito integrado. Este es el caso del Votrax SC-01 (1976) para síntesis por formantes y el TMS-5520 de Texas para síntesis por concatenación.

Además, en ese año se introdujo el LPC (Linear Prediction Code) (Mackhoul, e independientemente Markel y Grey) que es la base de la mayoría de los sistemas actuales de síntesis, en las cuales sólo se varía el tipo de entrada y el tipo de filtro donde se alimentan los parámetros .

Otra línea de investigación es tomar segmentos de voz pregrabados como bloques para construir una frase cualquiera. Debido a las características de la voz, no se pueden usar palabras o sílabas (debido a su gran cantidad) ni fonemas (debido a que no toman en consideración los efectos de coarticulación ni de transición entre fonemas). Debido a estos problemas, en 1958 Peterson[13] propuso una unidad denominada difonema, que corresponde al segmento entre el centro de un fonema y hasta el centro del siguiente, lo que permite tomar en cuenta los efectos de transición y coarticulación. En teoría, se requiere el cuadrado del número de fonemas de una lengua, pero hay combinaciones que no pueden existir, con lo que se reduce el número. Se pueden agregar algunos di-fonemas para grabar diferencias entre sílabas acentuadas y no, alófonos, etc. Peterson estimó que se requieren unos 8000 difonemas en inglés, aunque normalmente se utilizan aproximadamente 1000.

En 1961, Sivertsen[13] propuso mezclar difonemas y unidades más largas llamadas diádas, que contienen la mitad final de un fonema, un fonema completo y la mitad inicial del siguiente (VCV) para conservar algunos fenómenos que pueden no estar previstos en los difonemas.

Aunque tienen problemas como discontinuidades en algunas uniones, estos sistemas son muy utilizados debido a su relativa sencillez y alta inteligibilidad. El primer sistema de este tipo fue mostrado en 1967 en el MIT, pero el desarrollo de este sistema se canceló por falta de recursos.

En 1976, Olive y Spickenagle intentaron extraer las características de los fonemas para crear un sistema que generara un catálogo de difonemas de forma automatizada.

En 1979 se introdujo el algoritmo de escalamiento armónico en el dominio del tiempo. En este algoritmo se extraen segmentos en ventanas a intervalos iguales al tono de la señal; estos segmentos se pueden duplicar u omitir para ampliar o disminuir la duración del fonema, así como modificar el tono de la señal. En el caso de que no sea una señal sonora, se utilizan ventanas equiespaciadas a un intervalo predefinido.

Este algoritmo está basado en un experimento de 1959 (Miller, Licklider). En este experimento se demostraba que la señal de voz era redundante dentro de ciertos intervalos de tiempo, por lo que algunos segmentos pueden ser omitidos o repetidos sin alterar la comprensión.

Otro algoritmo basado en este experimento es SOLA [3 y 8] (Synchronized Overlap Add Method – Método de suma de traslape sincronizado), introducido en 1985 (Roucos, Wilgus). Este algoritmo es no recursivo y toma en cuenta las características de la transformada de Fourier de la señal de voz. En este algoritmo, se juntan los dos segmentos a concatenar y se traslapan hasta obtener el punto donde su autocorrelación es máxima. Una vez obtenido este punto, se promedian ambas señales en una ventana de traslape para obtener un punto de unión más suave, manteniendo las características de magnitud, fase y tono de la señal.

Otra ventaja de este algoritmo es que requiere poca carga computacional, por lo que puede ser usado en aplicaciones de tiempo real. Debido a esto, versiones modificadas de este sistema son muy utilizadas en sistemas de síntesis modernos.

Algunos de los algoritmos basados en SOLA son el PSOLA (Pitch Synchronous Overlap Add) y el TD-PSOLA [3] (Time Domain Pitch Synchronous Overlap Add), introducidos en 1987 (Lukaszewicz y Karjalainen), en los cuales no se usa ningún modelo paramétrico, sino se trabaja directamente con la señal original de voz, lo que simplifica aún más los cálculos y se evitan distorsiones ocasionadas por el uso de parámetros en lugar de la señal original. El PSOLA y su versión modificada TD-PSOLA, son variaciones de SOLA, que usan escalamiento armónico, sobreponiendo las señales, promediándolas y después modificando sus propiedades por medio de ventanas centradas en los puntos de amplitud máxima de cada periodo.

Un problema que se presenta con este algoritmo es en las señales que no tienen tono, donde al repetir ventanas para modificar la longitud del fonema, se crea ruido tonal. Una manera de reducir esto es invertir las ventanas que se repiten (Charpentier y Moulines, 1989)[8].

Otro problema que se presenta con este algoritmo, es que se requiere un intenso proceso previo en la base de datos para obtener segmentos que cumplan con las características necesarias.

En el sistema básico se requiere generar gran cantidad de segmentos para descartar aquellos que no cumplan con las características; pero se han generado métodos alternativos de resíntesis de segmentos que permiten modificar los segmentos para que puedan ser utilizados.

Ejemplos de estas modificaciones son el LP-PSOLA (Charpenier y Moulines, 1990) [8], donde se utiliza un predictor lineal y a la salida de éste se aplica el algoritmo PSOLA, y el MBR-PSOLA[3] (Dutoit y Leich en 1993), donde se modifican los parámetros de acuerdo con el tono de cada segmento y finalmente se ajustan para que coincida con el del siguiente segmento. Este proceso sólo se lleva a cabo en segmentos que son sonoros, para evitar los efectos de ruido armónico en segmentos sordos.

1.3. Tipos de síntesis de voz

Actualmente la mayoría de los sistemas de síntesis de voz están basados principalmente en 3 tipos básicos de síntesis[10]:

1- Articulatorios: tratan de modelar directamente el sistema generador de voz.
2- Por formantes: modelan la función de transferencia o de polos de frecuencias del tracto vocal.
3- Por concatenación: utilizan segmentos pregrabados que son unidos (concatenados).

Los dos tipos más usados son por formantes y por concatenación. Aunque el primero fue más usado inicialmente debido a limitaciones en la capacidad de almacenamiento,

actualmente el segundo es más usado debido a que son posibles mayores capacidades de almacenamiento.

1.3.1. Síntesis articulatoria

Este tipo de síntesis trata de modelar los órganos vocales lo más perfectamente posible, por lo que teóricamente es el sistema que podría generar síntesis de más alta calidad, pero a la vez es el sistema más complicado y de más alta carga computacional. Este tipo de síntesis involucra normalmente modelos de las cuerdas vocales humanas, de la lengua (posición, altura, etc.), apertura del velo, presión de los pulmones, apertura glotal, etc.

El modelo de articulación se genera normalmente a partir de radiografías, pero éstas no proporcionan información suficiente para conocer todos los parámetros necesarios. La ventaja de éste es que puede utilizar efectos que difícilmente se podrían incluir en otros sistemas.

Debido a las complejidades de análisis y la carga computacional requerida, este tipo de síntesis ha recibido poca atención, por lo que ha tenido muy poco desarrollo.

1.3.2. Síntesis por formantes

Este es uno de los métodos de síntesis más usados. Se basa en un modelo de entrada-filtro-salida del cual existen básicamente 2 tipos (filtros en cascada y en paralelo, figuras 1.6 y 1.7) y combinaciones de ambos, lo cual produce un mejor resultado. Además, este sistema proporciona mayor flexibilidad que la síntesis por concatenación y una menor dificultad que la síntesis articulatoria.

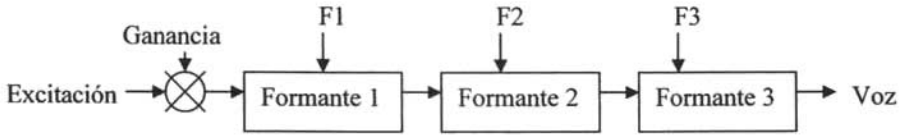

Fig. 1.6. Resonador en cascada[10]

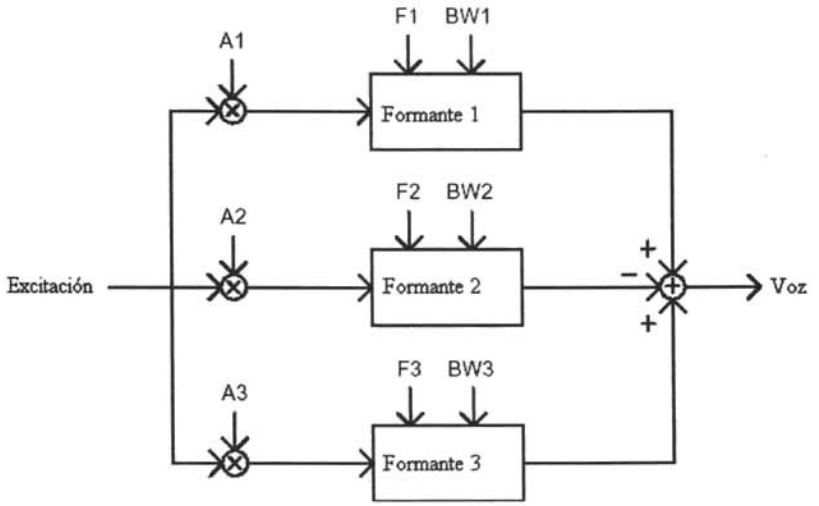

Fig. 1.7. Resonador en paralelo[10]

Este sistema fue muy utilizado en un principio, debido a que tiene pocos requerimientos de memoria y almacenaje. Sin embargo, este método ha sido usado en mucha menor cantidad últimamente y reemplazado en gran medida por los sistemas concatenados, ya que estos tienen una mejor salida de audio. Además, esta se genera de forma más sencilla y la capacidad de almacenamiento que requieren ya no es una limitante.

Normalmente se usan al menos tres formantes para producir la señal de voz, aunque a veces se emplean hasta cinco para mejorar la calidad. Cada formante se modela por medio de un resonador basado en un filtro centrado en la frecuencia del formante, y modelado con un par de polos, con lo que se puede indicar el ancho de banda del filtro.

La síntesis por formantes utiliza cierto conjunto de reglas que determinan los parámetros necesarios para cada sonido. Algunos de estos parámetros pueden ser: frecuencia fundamental (F0), grado de excitación (VO), frecuencias y amplitudes de los formantes (F1,F2,F3,A1,A2,A3), etc.

Los resonadores en cascada funcionan mejor con los sonidos no nasales, pero tienen problemas con las fricativas y oclusivas. Los resonadores en paralelo tienen problemas con las vocales pero funcionan bien para nasales, fricativas y oclusivas. Debido a esto Klatt[13] (1980) ideó un modelo con una mezcla de ambos filtros y 6 formantes, la adición de un ruido de alta frecuencia y un sistema de excitación compleja. Este modelo es el más utilizado en los sistemas comerciales actuales como el MITalk, DECtalk y Prose-2000.

Otro sistema mixto es el PARCAS [10], introducido por Laine en 1982, que es modelado por pares de ecuaciones de transferencia parciales y un conjunto de constantes para mantener las amplitudes balanceadas en diferentes salidas. Se modela por medio de filtros, tanto en cascada como en paralelo, y se usan entradas de pulsos y de ruido, según el fonema a sintetizar. Este sistema se muestra en la figura 1.8.

Una vez que se tiene el conjunto de resonadores a utilizar, éstos se alimentan con un tren de pulsos con una frecuencia igual a la de la fundamental (Fo) para sonidos con fundamental (p. ej. vocales) o con una fuente de ruido para sonidos sin fundamental (p. ej. /s/).

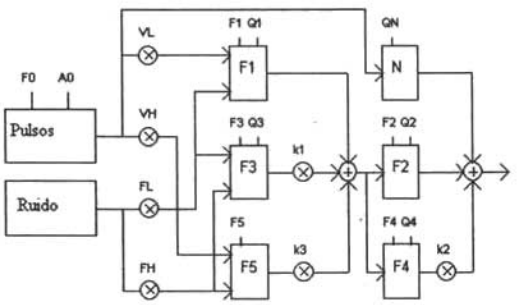

Fig. 1.8. Diagrama de PARCAS (Laine 1982)[10]

En algunos sistemas modernos se introducen señales diferentes a un tren de pulsos para mejorar la salida. Una versión alternativa de este sistema de síntesis es LPC (Predicción lineal). En éste, los coeficientes de los filtros son estimados previamente de segmentos de voz pregrabados (por lo que también tienen relación con la síntesis concatenativa).

La idea básica de LPC es que una muestra de sonido y(k) puede ser aproximada por medio de una combinación lineal de cierto numero de muestras pasadas y coeficientes a(k) precalculados. Esta predicción causa un pequeño error e(n) conocido como señal residual [3].

$$y(n) = e(n) + \sum_{k=1}^{p} a(k)y(n-k)$$

$$e(n) = y(n) - \sum_{k=1}^{p} a(k)y(n-k) = y(n) - \tilde{y}(n)$$

Para calcular los coeficientes, se minimiza el error en una ventana de muestras por medio de la autocovariancia o la autocorrelación, pero en el primer caso el filtro puede resultar inestable.

Una vez calculados estos coeficientes se utiliza el sistema básico de un filtro con entrada de pulsos o ruido, según el tipo de fonema (sonoro o sordo). Esta señal es filtrada por un filtro al que se le alimentan los coeficientes a(k). El numero de k de coeficientes (orden del filtro) es normalmente de entre 10 y 25, dependiendo de la calidad deseada y la velocidad de muestreo de la salida. Estos coeficientes deben ser modificados para cada ventana (aproximadamente entre cada 5 y 10 ms.).

La calidad de este sistema es baja debido a que las oclusivas tienen cambios muy súbitos, difícilmente modelables con un filtro. Además, por ser un sistema todo polo, no puede generar correctamente sonidos de tipo nasal, ya que esto requieren ceros en el filtro para ser modelados.

Para aumentar la calidad de los sistemas LPC se utilizan métodos alternativos, por ejemplo, filtros que aumentan la calidad de ciertas frecuencias a costa de otras, de una forma similar a la curva de respuesta del oído. Además se puede modificar la entrada, para ser alimentado por trenes de pulso de diferentes frecuencias simultáneamente, agregando la señal residual para eliminar el error o por medio de un "libro de códigos", donde se almacenan diferentes tipos de excitaciones según el fonema a generar.

UN método alternativo, conocido como metodo senoidal, consiste en obtener las características espectrales de la señal y su variación en el tiempo por medio de transformadas de Fourier. A partir de esta transformada se obtienen los picos de magnitud frecuencial (w_i) y la fase (ϕ_i) de la señal en estas frecuencias. A partir de estos parámetros se puede reconstruir la señal por medio de una sumatoria de cosenos [10].

$$s(n) = \sum_{i=1}^{L} A_i \cos(w_i n + \phi_i)$$

Debido a sus características este sistema genera problemas en fonemas que no tienen formantes al no existir picos de frecuencias. Este sistema se usa principalmente en sistemas que tratan de sintetizar canto, donde se requiere un control más estricto de las frecuencias y no importa tanto su inteligibilidad. Un ejemplo es el sistema LYRICOS del Instituto de Tecnología de Georgia (http://cslu.cse.ogi.edu/tts/research/sing/sing.html).

1.3.3. Síntesis por concatenación

Este tipo de sistema se basa en conectar segmentos de voz previamente grabados y almacenados. Es el sistema que presenta menor complejidad, aunque tiene la limitación de que no puede simular voces diferentes a las de la persona que originalmente grabó los segmentos.

Uno de los aspectos más importantes de este tipo de síntesis es el de seleccionar un tipo y tamaño de segmento de acuerdo con el tipo de sistema que se quiere desarrollar. En unidades grandes, se requieren menores puntos de concatenación, por lo que se obtiene un mejor control de coarticulación, pero se requiere un número muy grande de unidades. En unidades pequeñas, se tienen más puntos de concatenación, pero se reduce el número de unidades requeridas.

La unidades más grandes son las frases y las palabras. Estas funcionan bien para sistemas que tienen un vocabulario limitado, por ejemplo un sistema que dé la hora del día. Para sistemas de entrada libre hay demasiadas unidades, además de que al ser libre la entrada, puede contener palabras inexistentes o frases mal construidas.

Otra unidad que se podría considerar es la de las sílabas. Su número es considerablemente menor al de palabras o frases, pero aún es demasiado grande (10,000+). Además, no se pueden almacenar los efectos de coarticulación entre sílabas ni la prosodia. El uso de sílabas como unidad es factible sólo en lenguajes silábicos, como por ejemplo el japonés, donde existen menos de 100 sílabas. Debido a estos problemas, no existe al momento ningún sistema de conversión texto-habla que utilice estas unidades.

La siguiente unidad que se puede considerar es la de los fonemas. Su número es bastante reducido (normalmente entre 25 y 50, según el idioma). Los fonemas presentan el problema de la falta de información de coarticulación, por lo que son poco usados; aunque en muchos sistemas se utilizan los fonemas como unidades lógicas que son transformadas a la unidad correspondiente después de su análisis en el proceso de concatenación.

Otra unidad es la demisílaba, que representa la parte inicial y final de las sílabas. Su número es grande pero aceptable (aproximadamente 1000). Estas cubren un buen número de problemas como la coarticulación, algunos alófonos y requieren menos puntos de concatenación que los fonemas. Su número es grande, pero aún aceptable, desafortunadamente, su número no puede ser determinado fácilmente y hay algunas combinaciones que no pueden ser generadas con demisílabas, por lo que normalmente se

usan sólo en sistemas mixtos. Un ejemplo de sistema mixto que utiliza demisílabas es Hadifix (http://www.ikp.uni-bonn.de/dt/forsch/phonetik/hadifix/Hadifix.en.html) de la universidad de Bonn, Alemania.

La siguiente unidad a considerar es la de los difonemas. Un difonema se define como el segmento que inicia en punto central de un fonema y termina en el punto central del siguiente fonema [3]. Esto ayuda a disminuir la distorsión, al estar el punto de concatenación en una zona de relativa estabilidad. Esto, además, permite evitar los problemas de coarticulación, al estar la coarticulación presente implícitamente en los segmentos. El número existente de difonemas es el cuadrado de los fonemas existentes. De estos, existen algunos que pueden ser eliminados al no presentarse en un idioma. Su número está cerca de los 1000 en la mayoría de los idiomas. El número es grande pero aún aceptable, y debido a las ventajas que presenta, es un tipo de unidades muy utilizado. Estas son las unidades que serán utilizadas en el sistema a desarrollar.

Existen otros tipos de unidades menos utilizadas como los trifonemas (contienen un fonema entero entre dos medios fonemas). Estas unidades se utilizan actualmente en la investigación de métodos que mejoran los sistemas de síntesis de voz mediante el uso de unidades de diferentes longitudes.

Otro tipo de unidad que está en investigación actualmente es el microfonema, que es un segmento de fonemas en diferentes contextos. Estos segmentos (llamados prototipos) son después concatenados, y en las uniones se hace una interpolación para disminuir las discontinuidades.

Otro problema básico en estos sistemas es la distorsión que se presenta en los puntos de unión. Esto se ha minimizado un poco con el uso de difonemas, donde la unión se lleva a cabo en un punto más favorable. Otro proceso para eliminar esta distorsión es el uso de filtros o de interpolación que disminuyen las discrepancias entre segmentos, sin alterar demasiado la señal original. Además, se pueden utilizar varias versiones de un mismo difonema para seleccionar al que produzca la menor distorsión en cada punto de unión (por

ejemplo, el sistema Natural Voices de AT&T contiene una base de datos aproximadamente 100 veces más grande que la de otros sistemas difonémicos).

Un tipo de interpolación es el PSOLA [8] (Suma con Traslape Sincronizado al Tono). En éste, se toman los diferentes segmentos y se suman con un cierto traslape, ajustando por medio de ventanas centradas a una distancia igual a la frecuencia fundamental estimada previamente, con lo que, además de mejorar los puntos de concatenación, se puede modificar el tono al modificar la frecuencia e interpolar puntos intermedios o la duración al repetir ventanas. El problema básico de este método ocurre en aquellos fonemas que no tienen frecuencia fundamental, en los cuales se genera un pequeño ruido tonal, debido a la repetición de ventanas iguales.

Este sistema de interpolación es utilizado en los dos sistemas de síntesis de voz gratuitos más importantes, el Festival (desarrollado por Alan Black y Paul Taylor en la Universidad de Edimburgo) y el MBROLA (desarrollado en la Faculté Polytechnique de Mons, Bélgica). Ambos sistemas se encuentran actualmente en desarrollo, pero sólo el sistema Festival se encuentra disponible al público en forma de código fuente. Ambos sistemas proporcionan bases de datos para generar voz en lengua española en variantes española y mexicana.

2. Sistemas de síntesis de voz gratuitos

Se muestran las características básicas de algunos sistemas de síntesis de voz que se pueden obtener de forma gratuita en la red. A continuación se enumeran los programas que serán revisados, y se incluye la dirección donde se encuentran:

1.Festival: http://www.cstr.ed.ac.uk/projects/Festival/download.html
2.CSLU Speech Toolkit: http://cslu.cse.ogi.edu/toolkit/download/index.html (requiere registro)
3.ModelTalker: http://www.asel.udel.edu/speech/DownloadInstr.html (requiere registro)
4.MBROLA: http://tcts.fpms.ac.be/synthesis/mbrola.html
5.EPOS: http://epos.ure.cas.cz/
6.SAM: http://www.members.tripod.com/the-cbm-files/speak/

2.1. Festival

Este sistema fue desarrollado por Alan Black y Paul Taylor en la Universidad de Edimburgo. También contiene partes desarrolladas por CHATR en Japón y la universidad de Carnegie Mellon .

Idiomas soportados: inglés (USA y UK), español (México y España) y alemán. Una parte del soporte para alemán no es libre y se requiere obtener una licencia para su uso. Aunque el soporte de inglés es bastante completo, el soporte en español aún no está completo (por ejemplo no soporta abreviaturas, excepciones, etc.). Se pueden agregar excepciones a mano por medio de un diccionario.

El sistema Festival está diseñado principalmente para Linux, aunque también puede ser compilado para Windows utilizando el compilador gratuito de Cygwin. El sistema básico de síntesis es por medio de una base de datos difonémica. Los difonemas están

almacenados como un conjunto de parámetros LPC y su residuo. Para el proceso de síntesis se van alimentando los parámetros LPC a un filtro que es excitado por el residuo.

2.2. CSLU Toolkit

Este sistema de síntesis utiliza como base una versión modificada del sistema Festival, pero contiene algunas librerías adicionales para desarrollo de aplicaciones y un módulo de animación facial. Además de los idiomas soportados por Festival (ingles, español y alemán) permite sintetizar voz en portugués. Aunque el sistema es gratuito, requiere activarse por medio de Internet durante su instalación.

Figura 2.1. Baldi. Módulo de animación de CSLU Toolkit

2.3. Modeltalker

Desarrollado por la Universidad de Delaware, USA. Únicamente soporta inglés. Este sistema funciona también por medio de difonemas. El sistema está pensado para modelar la entonación de diferentes estados de ánimo (alegría, enojo). Desafortunadamente, la base de datos está incompleta, por lo que la calidad del resultado se ve disminuida de forma drástica.

El sistema está dividido en 7 bloques básicos:

- Análisis de texto. Principalmente expande los símbolos y las abreviaturas y convierte los números a palabras. También analiza los comandos que controlan el funcionamiento del sintetizador (entonación, acentos, etc.)

- Módulo de texto a fonemas: A partir del texto analizado genera una salida fonémica, con acentuación y silabificación de palabras. También trata de obtener la pronunciación correcta de aquellas palabras cuya pronunciación varía por su contexto.

- Reglas de partes de habla. Trata de obtener la función de una palabra dentro de la frase (verbo, sustantivo, etc.)

- Análisis prosódico. Analiza la entonación, ritmo, etc. de la frase.

- Análisis del discurso. Trata de obtener la influencia de las frases aledañas en la pronunciación de una frase.

- Cálculo de duración de segmentos. Analiza la duración de diferentes fonemas de acuerdo con la prosodia previamente calculada

- Cálculo de contornos de entonación. Una vez que se tiene el análisis prosódico y la duración de segmentos se genera una curva de entonación para el texto a leer.

2.4. MBROLA

Este sistema soporta el número más grande de idiomas de los sistemas revisados (español (MX, ES), inglés (US, UK), francés, alemán, japonés, etc.) y funciona también por concatenación de difonemas. Este sistema no es realmente un sistema TTS, pues requiere que la entrada haya sido previamente analizada y se haya obtenido la cadena de fonemas a leer. Para poder convertir una entrada de texto a la trascripción fonética requerida, se necesita utilizar un programa adicional que haga la conversión. Para el módulo de TTS se

puede utilizar Festival, que tiene una opción para utilizar MBROLA como método de síntesis. Otro programa que existe es EULER (español, inglés y francés). Este sistema tiene soporte más básico que el de Festival para español.

MBROLA utiliza un método de interpolación lineal por ventanas modificado, donde la longitud de la ventana depende de la frecuencia fundamental del fonema. Debido a esto, los difonemas se encuentran previamente normalizados (todos tienen la misma frecuencia fundamental) para simplificar el cálculo, pero a cambio de una severa distorsión en la salida.

2.5. EPOS

EPOS fue desarrollado en la universidad de Praga y soporta checo, eslovaco, francés y alemán. Este sistema está basado en módulos externos, lo que aumenta su flexibilidad. Contiene módulos de síntesis de voz por difonemas, almacenados directamente como muestras o como coeficientes LPC.

2.6. SAM

Este sistema sólo soporta la lengua inglesa. El programa fue comercial pero, debido a que se encuentra descontinuado, se puede encontrar en la red. Este programa tiene principalmente un valor histórico, ya que fue uno de los primeros sistemas comerciales de síntesis de voz automática (1979). De todos los sistemas analizados, es el único que no utiliza síntesis por concatenación, sino síntesis por formantes.

Fue desarrollado para sistemas con muy poca capacidad de almacenamiento (la plataforma para la que fue diseñado tenía aproximadamente 30 kbytes de RAM y no contaba con unidad de lectura en disco, por lo que requería estar completamente cargado en memoria) y poca capacidad de procesamiento (contaba con un procesador de 1 Mhz.). Debido a estas restricciones, la única solución posible fue la síntesis por formantes.

También se observa que aunque la calidad de la salida de audio es mucho menor, este tipo de síntesis ofrece una mayor flexibilidad, al permitir modificar las propiedades de la boca y la garganta. Actualmente, este tipo de síntesis está prácticamente en desuso, debido a la gran capacidad de almacenamiento en los equipos de cómputo actuales que permite almacenar bases de datos de mayor tamaño.

3. Funcionamiento de Festival

Festival es un sistema de síntesis de voz gratuito de uso sin restricciones, del cual se proporciona el código fuente. Debido a esto, y a diferencia de los demás sistemas de síntesis que contienen algoritmos propietarios de los que se proporciona poca o ninguna información, se puede analizar de forma amplia su funcionamiento.

El sistema Festival se controla por medio de una versión modificada de SCHEMA (una variante de Lisp). Estos comandos se pueden introducir al sistema mediante un interprete de comandos, por medio de archivos de procesamiento por lote o por medio de una biblioteca para C++, con la que se puede integrar el sistema dentro de programas escritos en este lenguaje.

Un ejemplo de comandos en SCHEMA es el siguiente:

```
(voice_abc_diphone)
(SayText "Hola mundo")
```

El primer comando selecciona una voz en español. El sistema incluye voces con pronunciaciones española y mexicana.

El segundo comando toma una cadena, la analiza para generar una salida fonética y a partir de ésta genera una salida de voz. Esto tiene como ventaja que es más sencillo agregar nuevas definiciones de idiomas o modificar las ya existentes, debido a que sólo se requieren crear los scripts de definiciones en lugar de tener que modificar el código fuente de forma directa.

Para llevar a cabo el proceso de síntesis, primero se analiza el texto. El sistema permite agregar filtros para diferentes tipos de texto. Por ejemplo, para un email se puede filtrar la cabecera del mensaje (De:, Para:) o para una página de Internet se pueden eliminar los comandos de HTML.

Una vez filtradas las partes que no se van a sintetizar, se obtiene la representación fonémica del texto. Para esto primero se utiliza un diccionario de pronunciación y sólo en caso de que la palabra no se encuentra en el diccionario se utilizan las reglas de pronunciación (para inglés se incluye un diccionario de pronunciaciones basado en el Oxford Advanced Learners Dictionary para inglés británico y una base de datos de Carnegie Mellon para inglés norteamericano). Para español aún no existe un lexicon compilado. También se tiene que analizar el texto para convertir números y abreviaturas a texto, así como para buscar palabras que pueden tener diferente pronunciación según el contexto y tratar de obtener su pronunciación correcta.

Después se separa el texto en frases para añadir pausas y generar una entonación adecuada según el tipo de frase (pregunta, exclamación, etc.). Desafortunadamente, los dos módulos existentes de análisis para español de Festival carecen de esta sección.

Una vez que se tiene la frase completamente analizada, se obtienen los segmentos de una base de datos de difonemas previamente grabados y almacenados en formato LPC con residuo, los cuales se van procesando de acuerdo con la duración y entonación de cada fonema, y se van uniendo para generar la salida de voz.

3.1. Definición de fonemas

El primer paso para definir una voz en Festival es generar una tabla de fonemas. En la tabla 3.1 se indican las características que se pueden asignar a cada uno de los fonemas. Existen dos versiones de español previamente definidas. La primera es de la Universidad de Edimburgo para español de Castilla. La segunda es de la Universidad de Oregón para español mexicano. En la tabla 3.2 se muestran las definiciones de fonemas de OGI y Edimburgo. Es interesante observar las discrepancias entre ambas. Esto se debe a que no existe una lista estandarizada de fonemas, sino que esta varía según el autor. También se

VARIABLE	VALOR	SIGNIFICADO
VOCAL	+	Es vocal
	-	No es vocal
LONGITUD	S	Corta
	L	Larga
	D	Diptongo
	A	Schwa
	0	No es vocal
ALTURA	1	Alta
	2	Media
	3	Baja
	-	No es vocal
PUNTO ARTICULACION	1	Enfrente
	2	Enmedio
	3	Atrás
	--	No es vocal
REDONDEO LABIOS	+	Si
	-	No
TIPO DE CONSONANTE	S	Stop
	F	Fricativa
	A	Africativa
	N	Nasal
	L	Liquida
	0	No es consonante
PUNTO ARTICULACION	L	Labial
	A	Alveolar
	P	Palatal
	B	Labiodental
	D	Dental
	V	Velar
	0	No es consonante
SONORA	+	Si
	-	No

Tabla 3.1. Variables para la definición de fonemas

puede observar que en los fonemas comunes a ambas versiones existen discrepancias respecto a las características que se les asignan a los mismos. Adicionalmente, la versión de Edimburgo contiene dos fonemas por vocal (acentuado y no acentuado). Además del tipo de fonema, se requiere una tabla donde se indica la longitud promedio de cada uno de los fonemas. Esta duración será modificada posteriormente en el momento de la síntesis para dar ritmo y entonación.

3.2. Reglas de *tokenizacion*

Para analizar un texto el primer paso es reemplazar los números, símbolos, abreviaturas, etc.- en palabras. A esto se le conoce como tokenizacion. En Festival esto se lleva a cabo mediante una tabla de abreviaturas o por medio de reglas. De los dos módulos de análisis para español disponibles para festival sólo la versión de Oregón contiene un diccionario de abreviaturas (ver listado 3.1). En el caso de los símbolos se utiliza un diccionario similar al de las abreviaturas.

```
("Dr"              "doctor")
("Dra"             "doctora")
("AA"              "autores")
("Abr"             "abril")
("[Aa]cept"        "aceptacio'n")
("[Aa]cr"          "acreedor")
("[Aa]\\.[Cc]"     "antesdecristo")
("[Aa]dj"          "adjetivo")
("[Aa]\\.[Mm]"     "aeme")
("[Aa][Mm]"        "aeme")
("[Pp][Mm]"        "peeme")
```

Listado 3.1. Segmento del diccionario de abreviaturas

Nombre		VOCALES										CONSONANTES						Ejemplo	DURACION	
		Vocal		Long.		Altura		P. Articu.		Redondeo Labios		Tipo		P. Artic.		Sonora				
EDI	OGI	EDI	OGI	EDI	OGI	EDI	OGI	EDI	OGI	EDI	OGI	EDI	OGI	EDI	OGI	EDI	OGI		OGI	EDI
#	pau	-	-	0	0	-	-	-	-	-	-	0	0	0	0	-	-	(silencio)	0.1	0.25
a	a	+	+	l	s	3	2	2	2	-	-	0	0	0	0	-	+	casA	0.09	0.08
e	e	+	+	l	s	2	2	1	2	-	-	0	0	0	0	-	+	caucE	0.079	0.08
i	i	+	+	l	s	1	1	1	1	-	-	0	0	0	0	-	+	acIdo	0.08	0.07
o	o	+	+	l	s	2	2	3	2	+	-	0	0	0	0	-	+	cOmida	0.083	0.08
u	u	+	+	l	s	1	1	3	1	+	-	0	0	0	0	-	+	sUbir	0.064	0.07
i0	j	+	-	s	0	1	-	1	-	-	+	0	l	0	d	-	+	acelte	0.042	0.04
u0	w	+	-	s	0	1	-	3	-	+	+	0	s	0	a	-	-	caUce	0.063	0.04
a1		+		l		3		2		-		0		0		-		cAsa		0.09
e1		+		l		2		l		-		0		0		-		cErca		0.09
i1		+		l		1		1		-		0		0		-		cInco		0.08
o1		+		l		2		3		+		0		0		-		cOsa		0.09
u1		+		l		1		3		+		0		0		-		azUl		0.08
p	p	-	-	0	0	-	-	-	-	-	+	s	s	l	l	-	-	Puerta	0.092	0.1
t	t	-	-	0	0	-	-	-	-	-	+	s	s	d	a	-	-	Tiempo	0.098	0.085
k	k	-	-	0	0	-	-	-	-	-	+	s	f	v	p	-	-	Kilo	0.099	0.1
b	b	-	-	0	0	-	-	-	-	-	+	s	s	l	l	+	+	Barco	0.076	0.065
d	d	-	-	0	0	-	-	-	-	-	+	s	s	d	a	+	+	Disco	0.069	0.06
g	g	-	-	0	0	-	-	-	-	-	+	s	f	v	p	+	-	Guerra	0.061	0.08
f	f	-	-	0	0	-	-	-	-	-	+	f	f	b	b	-	-	Familia	0.105	0.1
th		-		0		-		-		-		f		d		-		Cerca		0.1
s	s	-	-	0	0	-	-	-	-	-	+	f	f	a	a	-	+	Subir	0.095	0.11
x	x	-	-	0	0	-	-	-	-	-	+	f	f	v	a	-	-	Jirafa	0.097	0.13
ch	tS	-	-	0	0	-	-	-	-	-	+	a	s	p	l	-	-	CHamaco	0.12	0.135
m	m	-	-	0	0	-	-	-	-	-	+	n	n	l	l	+	+	Marcha	0.073	0.07
n	n	-	-	0	0	-	-	-	-	-	+	n	n	a	l	+	+	Nariz	0.065	0.08
	N		-		0		-		-		+		n		l		+	eNgaño	0.086	
ny	ny	-	-	0	0	-	-	-	-	-	+	n	n	p	l	+	+	aÑos	0.11	0.11
l	l	-	-	0	0	-	-	-	-	-	+	l	l	a	d	+	+	Lupa	0.058	0.08
ll	dZ	-	-	0	0	-	-	-	-	-	+	l	s	p	p	+	-	AYer	0.076	0.105
r	r	-	-	0	0	-	-	-	-	-	+	l	l	a	p	+	+	aRbol	0.098	0.03
rr	rr	-	-	0	0	-	-	-	-	-	+	l	l	a	p	+	+	Real	0.151	0.08

Tabla 3.2. Comparación entre las definiciones de OGI y Edimburgo

Para expandir los números en palabras se requiere un módulo más complejo que el requerido para símbolos o abreviaturas. En este caso existe un diccionario de lectura para diferentes posiciones dentro de una cifra y la expansión se lleva a cabo de forma recursiva separando el numero previamente en bloques.

3.3. Conversión de letras a fonemas

Una vez que se tiene una salida que sólo contiene texto se puede proceder a convertir el texto a una serie de fonemas que indiquen como se leen las palabras.

El primer paso es buscar dentro de un diccionario donde se encuentran las palabras con su pronunciación. Ninguna de las dos definiciones de español que contiene Festival cuentan con un diccionario de excepciones extenso. La versión de Oregón contiene algunos ejemplos para indicar la forma en que se puede generar este diccionario, como se muestra en el listado 3.2.

```
(lex.add.entry
      '("texas" nil (((t e) 1) ((x a s) 0))))
 (lex.add.entry
      '("méxico" nil (((m e) 1) ((x i) 0) ((k o) 0))))
(lex.add.entry
      '("xochimilco" nil (((s o) 0) ((tS i) 0) ((m i l) 1) ((k o) 0))))
(lex.add.entry
      '("atlixco" nil (((a) 0) ((t l i s) 1) ((k o) 0))))
```

Listado 3.2. Segmento del diccionario de lecturas

En ambas definiciones de español de Festival, el módulo de conversión texto-fonemas consiste en un conjunto de reglas que indican que letra o conjunto de letras se transforman en que sonidos o grupo de sonidos por medio de expresiones regulares.

El primer paso consiste en definir las variables básicas que serán utilizadas por el módulo de conversión. Las variables se definen entre paréntesis, empezando por el nombre y seguido por el elemento o elementos que la forman, separados por espacios. Las variables más importantes se muestran en el listado 3.3.

```
(LNS l n s )
(DNSR d n s r )
(EI e i é í)
(AEIOUt á é í ó ú )
(V a e i o u )
(C b c d f g h j k l m n ñ ~ p q r s t v w x y z )
```

Listado 3.3. Definición de variables básicas

El primer segmento de la conversión es el conjunto de reglas para romper diptongos (listado 3.4). Esto permite dividir el texto en sílabas para posteriormente poder encontrar la acentuación en aquellas palabras en las que no existe acento escrito.

```
( "'" V* C* u [ i ] DNSR # = i )
 ( AEIOUt V* C* u [ i ] DNSR # = i )
( u [ i ] DNSR # = il )
 ( "'" V* C* u [ i ] d V # = i )
 ( AEIOUt V* C* u [ i ] d V # = i )
 ( u [ i ] d AEIOUt # = i
 ( u [ i ] d V # = il )
```

Listado 3.4. Reglas para la ruptura de diptongos

Por ultimo se definen las reglas de conversión, donde se define el o los sonidos asignados a una letra o conjunto de ellas. Para crear estas reglas se usa el formato: ([TEXTO A CONVERTIR] = FONEMAS RESULTANTES). Estas se muestran en el listado 3.5.

([a] = a)	([d] = d)	([p h] = f) ([q u] a = k u)
([e] = e)	([f] = f)	([q u] = k)
([i] = i)	([g] "" EI = x)	([q] = k)
([o] = o)	([g] EI = x)	([r r] = rr)
([u] = u)	([g u] "" EI = g)	(# [r] = rr)
(["" a] = al)	([g u] EI = g)	(LNS [r] = rr)
(["" e] = el)	([g ":" u] EI = g u)	([r] = r)
(["" i] = il)	([g ":" u] "" EI = g u)	([s] = s)
(["" o] = ol)	([g "\"" u] EI = g u)	(# [s] C = e s)
(["" u] = ul)	([g "\"" u] "" EI = g u)	(# [s] "" C = e s)
([á] = al)	([g ü] EI = g u)	(# [s] ":" C = e s)
([é] = el)	([g ü] "" EI = g u)	(# [s] "\"" C = e s)
([í] = il)	([g] = g)	([t] = t)
([ó] = ol)	([h] =)	([w] = u)
([ú] = ul)	([j] = x)	([x] = k s)
([":" u] = u)	([k] = k)	([y] # = i)
(["\"" u] = u)	([l l] # = l)	([y] C = i)
([ü] = u)	([l l] = ll)	([y] "" C = i)
([b] = b)	([l] = l)	([y] ":" C = i)
([v] = b)	([m] = m)	([y] "\"" C = i)
([c] "" EI = th)	(["~" n] = ny)	([y] = ll)
([c] EI = th)	([ñ] = ny)	([z] = th)
([c h] = ch)	([n] = n)	
([c] = k)	([p] = p)	

Listado 3.5. Reglas de conversión

4. Construcción de la base de datos

4.1. Selección de unidades

Para los sistemas de síntesis de voz basados en concatenación se requiere la creación de una base de datos que contenga los segmentos pregrabados que serán usados para generar la salida. Los segmentos más comunes son los difonemas, los cuales son usados casi en la totalidad de los sistemas actuales de síntesis de voz, donde sólo varía la forma de almacenamiento y la forma en que son procesados. Estos son los segmentos usados dentro del sistema desarrollado.

En algunos sistemas se utilizan, además de los difonemas, otros segmentos de diferente tamaño para casos específicos, aunque estos casos son pocos debido al aumento de la complejidad al mezclar unidades de diferente tamaño. Estos segmentos adicionales pueden ser unidades con fonemas completos entre los medios fonemas (trifonemas) que son usados en los casos en que la coarticulación u otros efectos de contaminación entre fonemas son muy grandes y no pueden ser representados correctamente con el uso de difonemas. Otro conjunto de unidades usado con cierta frecuencia es el de las palabras completas, las cuales constan normalmente de pronombres, preposiciones y otras palabras que ocurren de forma muy común en el idioma.

4.2. Grabación de unidades

Una vez que el tipo de unidades está decidido, se requiere generar una lista de todos los segmentos necesarios. Cuando se utilizan difonemas, el número teórico de unidades es el cuadrado del número total de fonemas. Sin embargo, en la práctica este número es menor debido a que existen combinaciones inexistentes.

Dos ejemplos de esto serían las combinaciones /r//R/ y /R//r//, que no pueden ser representadas de forma escrita. Otro ejemplo, es el de los alófonos /w/ e /j/ que ocurren sólo en lugares muy específicos, por ejemplo /w/ ocurre en "hue", pero no puede ocurrir en "au".

Una vez que se ha obtenido toda la lista de combinaciones, éstas requieren ser grabadas. Esto se puede llevar a cabo por medio de frases seleccionadas de forma que contengan todas las combinaciones necesarias, o creando palabras sin sentido.

Por el primer método se tiene la ventaja de que pueden obtenerse diferentes versiones del mismo segmento en diferentes contextos, permitiendo seleccionar el mejor, aunque el proceso de selección es muy lento.

El segundo método es mucho más sencillo, al no requerir seleccionar palabras que contengan los segmentos y al ser mucho menor el número de segmentos a catalogar.

Al final se debe de tener al menos una grabación de cada segmento, aunque el tener varias versiones de los segmentos puede ayudar a generar audio de mejor calidad (un ejemplo de esto es en el sistema Naturalvoices de ATT que contiene una gran cantidad de versiones de cada segmento), sin embargo el trabajo de catalogar los segmentos es un proceso demasiado complicado, por lo que muy pocos sistemas utilizan bases tan extensas. Además, el espacio de almacenamiento aumenta considerablemente (la base de datos de Naturalvoices es 50-100 veces mayor que la de los demás sistemas revisados) y aumentan de forma drástica los requerimientos de procesamiento al tener que analizar una base de datos mayor y calcular en cada momento el mejor segmento a utilizar.

Una vez que se tiene la grabación lista, se deben catalogar los segmentos. Por ejemplo, en el caso de los difonemas se deben dar el inicio y el final de cada segmento que correspondan con el punto medio de los fonemas. La división debe ser siempre en el mismo punto del ciclo de los fonemas para minimizar las discontinuidades. Normalmente se utiliza como inicio de ciclo el punto de máxima amplitud. Además, en el caso de los difonemas también se indica el punto de división entre fonemas.

En el caso del sistema desarrollado, la base de datos fue generada a partir de una grabación de palabras inventadas. Esta grabación se llevo a cabo en una PC de escritorio, con una tarjeta de sonido integrada y muestreada a 11 Khz. Y 16 bits por muestra. Cada segmento se encuentra almacenado dentro de un archivo independiente, cuyo nombre corresponde al difonema almacenado. Se consideraron 29 fonemas para generar la base (existen 23 fonemas en la lengua española, pero se consideraron 2 juegos de vocales, acentuadas y no acentuadas, y la pausa como fonemas adicionales).

4.3. Obtención de las marcas de periodo

En varios de los métodos que se utilizan para modificar el tono y el ritmo de la señal de voz se requiere obtener para cada uno de los segmentos pregrabados los puntos en los que comienzan los ciclos en los segmentos sonoros. Para los segmentos no sonoros se utilizan marcas equidistantes. Se necesita almacenar estas marcas junto con los segmentos para su futuro procesamiento.

```
% speech= señal de entrada
% fs_in= frecuencia de muestreo
p1 = round(fs_in/400);
p2 = round(fs_in/60);
spch = speech(☺';
xsamp = length(spch);
N=1:xsamp;
wlen = round((p1+p2)/3); % obtener el tamaño de la ventana de Hanning
spch=spch.^2; % elevar la señal de entrada al cuadrado
ecurve = conv(hanning(wlen)', spch); %convolucionar la señal de entrada
con la ventana de Hanning
ecurve = conv(hanning(wlen)',ecurve); %convolucionar la señal por segunda
vez con la ventana de Hanning
ecurve = ecurve((1:xsamp)+wlen); %recortar el resultado al tamaño de la
señal
```

Listado 4.1. Obtención del contorno de energía

Una manera de obtener este contorno es por medio del contorno de energía [6]. Para obtener este contorno se toma la señal de entrada y se eleva al cuadrado. Esta señal elevada al cuadrado es convolucionada con una ventana de Hanning y el resultado se convoluciona una segunda vez con la ventana de Hanning. En el listado 4.1 se muestra la función encargada de obtener este contorno.

En esta función se define la ventana de Hanning, mostrada en el listado 4.2, como:

$$W(n) = \frac{1 + \cos(\pi\left(\frac{2n}{N} - 1\right))}{2} \quad ; \text{n=-1..N-1}$$

```
function W = hanning(N)
   W1 = (1 + cos(pi*linspace(-1,1,N+2)))'/2;
   W = W1(2:(N+1));
return
```

Listado 4.2. Ventana de Hanning

Para obtener las marcas de periodo se obtienen los máximos locales del contorno de energia, los cuales se encuentran en la misma posición que las marcas de periodo. En la figura 4.1 se muestra el resultado de la doble convolución y en la figura 4.2 los máximos locales sobrepuestos a la señal original. Desafortunadamente el tamaño de la ventana de Hanning se debe obtener a prueba y error pues pueden obtenerse marcas de periodo erróneas debidas a la frecuencia de corte, como se observa en la figura 4.3. En el caso de que la ventana de Hanning sea demasiado pequeña se pierden marcas de periodo, si esta ventana es demasiado grande se pueden obtener ventanas extras, además de obtener un desplazamiento de las marcas, lo que ocasiona que las marcas se obtengan en una posición incorrecta.

Otro problema de este algoritmo es que requiere un módulo que analice la señal para reconocer si el segmento es sonoro o sordo para agregar marcas equiespaciadas dentro de los fonemas sordos, ya que de estos no se obtienen marcas.

Debido a estos problemas, en 1998 Goncharoff propuso un algoritmo con programación dinámica [6], basado en la idea de que, debido a la inercia del aparato de fonación humano, los cambios de frecuencia dentro de una señal no pueden ser súbitos, sino evolucionar de forma lenta. Este algoritmo obtiene resultados mucho mejores y más consistentes, por lo que es el algoritmo usado para generar la base. Además, este algoritmo no requiere marcar los periodos como sonoros o sordos, ya que en los segmentos sordos genera automáticamente marcas equiespaciadas.

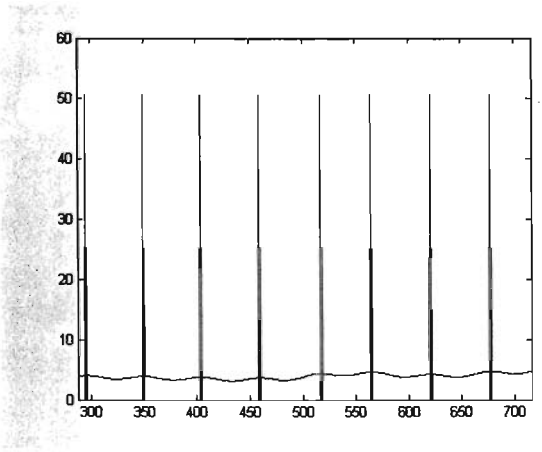

Fig. 4.1. Resultado de la convolución mostrando picos obtenidos.

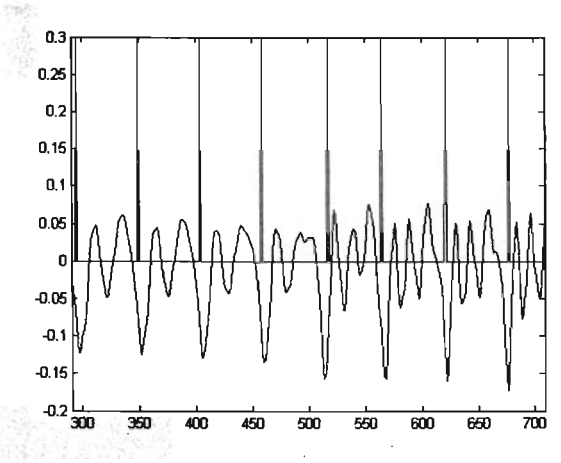

Fig. 4.2. Señal original y marcas de periodo obtenidas

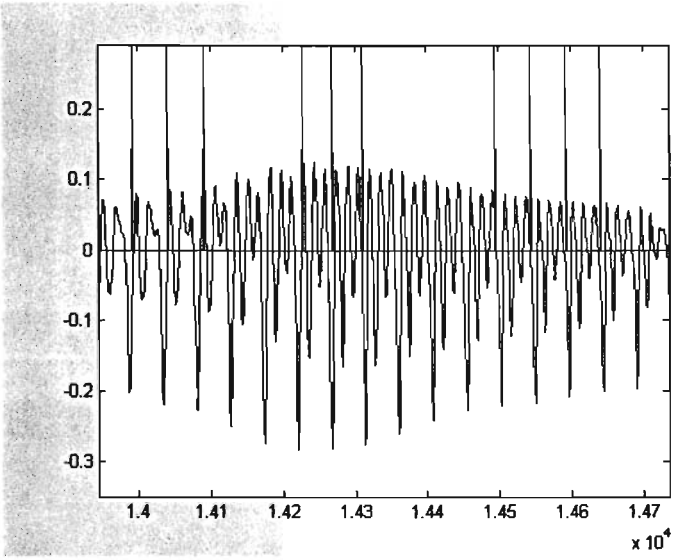

Fig. 4.3. Pérdida de marcas por la frecuencia de corte

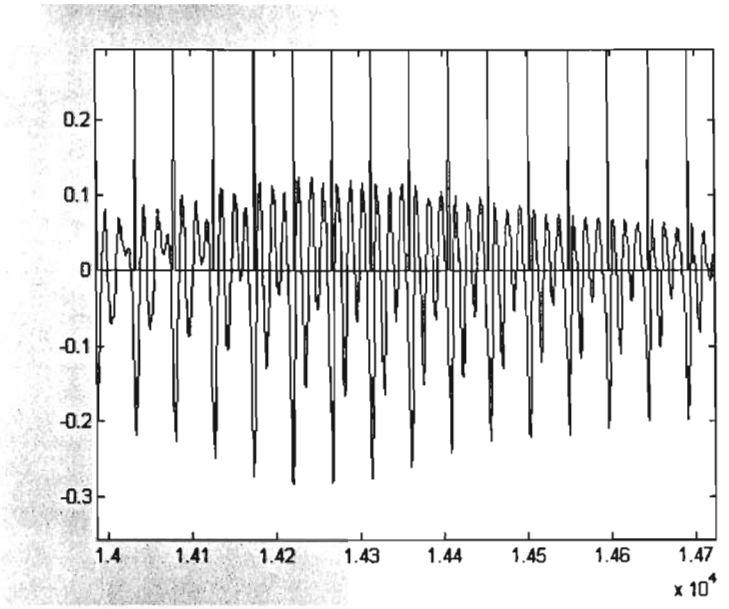

Fig. 4.4. Obtención de marcas de periodo con el algoritmo de Goncharoff

En la figura 4.4 se observa el mismo segmento de las figuras anteriores con las marcas que obtuvo este algoritmo. En este caso la selección de segmentos es la correcta, cada marca se encuentra en el punto de máxima amplitud de cada ciclo. Además se puede observar como cada ciclo es similar en forma al anterior y los cambios ocurren de forma paulatina.

Un algoritmo de programación dinámica es un algoritmo recursivo que utiliza los resultados de iteraciones anteriores para minimizar el tiempo de resolución o, cuando no existe un algoritmo óptimo, analizando diferentes soluciones para obtener la mejor.

Estos algoritmos en general involucran encontrar el camino a través del cual se obtenga la mejor calificación a través de una matriz de valores, por medio de movimientos predeterminados (derecha, diagonales, abajo). Se calculan las calificaciones que se obtienen a través de diferentes rutas, se almacenan la mejor calificación para cada celda y la ruta para alcanzarla y después se recorre la matriz de forma inversa para obtener la ruta óptima.

Se requiere que la solución pueda ser alcanzada en etapas por medio de decisiones que deben cumplir con el principio del óptimo:

"En una secuencia de decisiones óptima toda sub-secuencia ha de ser también óptima" (Bellman, 1957) [19].

Un algoritmo de programación dinámica tiene los siguientes pasos :

1. Planteamiento de la solución como una sucesión de decisiones y verificación de que ésta cumple el principio de óptimo. Cada etapa requiere una decisión.

2. Definición recursiva de la solución. Cada etapa tiene un conjunto de estados asociados.

3. Cálculo del valor de la solución óptima mediante una tabla en donde se almacenan soluciones a problemas parciales para reutilizar los cálculos.

4. Construcción de la solución óptima haciendo uso de la información contenida en la tabla anterior.

El algoritmo de Goncharoff está dividido en dos etapas. Primero se estima la evolución de la frecuencia en la señal y posteriormente, con esta información, se encuentran los picos donde se deben colocar las marcas.

Primero se obtiene una estimación por medio del contorno de energía, utilizando el método descrito previamente. Esta información se almacena en una matriz que contiene un número de columnas igual a las ventanas analizadas (una cada 40 ms aproximadamente) y con renglones igual al periodo máximo.

Los periodos mínimo y máximo se toman respectivamente como:
Pmin= round(fsamp/400) muestras
Pmax= round(fsamp/60) muestras

donde fsamp corresponde a la frecuencia de muestreo de la señal.

Primero se encuentra el renglón de la matriz dentro del cual se encuentra cada uno de los picos. Se calcula la distancia al pico anterior y, si es menor al mínimo o mayor al máximo se descarta. En caso contrario, se guarda el valor del pico en la columna correspondiente a su distancia. También se calcula la distancia al siguiente pico, por lo que se almacenan dos valores para cada pico. Todas las demás celdas tienen valor cero. En la figura 4.5 se muestra una representación grafica de la matriz para el ejemplo de la figura 4.4

Después se calcula la mejor ruta de acuerdo con los movimientos permitidos (estos son sólo en diagonal a la derecha, con un desplazamiento vertical máximo de tres). Después se interpolan las distancias obtenidas para obtener los puntos aproximados de las marcas, poniendo mayor énfasis en las frecuencias altas.

De aquí se obtiene una segunda matriz, donde para cada marca se toma una ventana de Hanning de tamaño igual al periodo máximo permitido, se almacena un uno en la columna del renglón al que corresponde esa marca y se obtiene la ruta con que se optimizan las posiciones de las marcas.

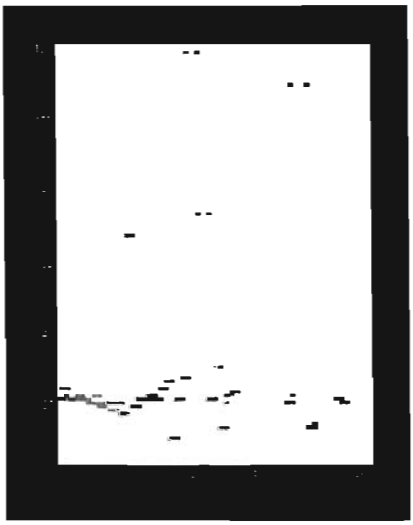

Fig. 4.5. Segmento de la matriz de periodos

5. Conversión de texto a fonemas

Este proceso consta principalmente de dos etapas:

La primera etapa es la normalización del texto en la cual se toma el texto introducido en crudo, se remueve el formato y se convierten los símbolos, abreviaturas y números en su representación en palabras. Además se reemplazan los símbolos de puntuación por pausas.

La segunda etapa consiste en reemplazar el texto ya filtrado en su representación fonética, que indica la forma en que deben ser leídas las palabras

5.1. Normalización del texto

La primera parte de un sistema de síntesis de voz es la etapa de normalización del texto. Este proceso consiste en traducir los símbolos, abreviaturas y números en palabras completas que puedan ser leídas posteriormente. Este proceso es muy complejo y aún no existe una solución total. Lo anterior es debido principalmente a que un mismo símbolo puede traducirse de diferentes maneras en diferentes contextos.

Un ejemplo de esto sería el símbolo '.' que puede interpretarse de varias maneras, como por ejemplo:
Fin de oración: La casa es grande.
Para abreviaturas: Sr. López
Como punto decimal: 123,456.34

5.1.1. Números

La pronunciación de los números puede variar según el contexto en que se encuentran, por lo que el primer paso sería determinar el contexto en el que estos se encuentran. Los casos más importantes son:

Números ordinales: Normalmente se representan añadiendo 'o', 'a', 'er' al final del número :

1º -> primero

2ª -> segunda

1er -> primer

Tiempo: Este se distingue normalmente por tener dos o tres grupos de números de uno o dos dígitos separados por medio de ":". Es uno de los casos más difíciles de considerar, debido a que en ocasiones no se pueden distinguir las unidades usadas, por ejemplo 12:25 puede significar 12 horas con 25 minutos o 12 minutos y 25 segundos.

Fecha: Las fechas también son muy complejas de pronunciar correctamente, debido a la gran cantidad de representaciones diferentes que puede tener, así como la posibilidad de que el mes sea representado por una abreviatura o un número . El caso más difícil de una fecha es cuando el mes se escribe de forma numérica, ya que el día y el año se pronuncian como números ordinales.

Números cardinales: Para los números cardinales el único problema que se debe resolver es con el número '1' que se puede pronunciar de tres maneras dependiendo de si se encuentra solo (uno) o modificando a un sustantivo masculino (un) o femenino (una).

Para obtener la pronunciación de un numero cardinal primero se obtiene el número de cifras. Sólo se obtendrá la pronunciación para números menores a un billón, debido a que números más grandes a este normalmente se escriben como texto. El proceso de lectura de números es recursivo, como se muestra en el listado 5.1.

Si el número contiene punto decimal se divide en dos grupos, antes y después del punto, y se leen de forma independiente por medio de llamadas independientes a la función:
1er. grupo + punto + 2do. grupo

Si el número tiene seis o más cifras se divide en dos grupos, uno con los seis dígitos a la derecha y otro del resto y se manda a llamar de nuevo a la función con cada uno de los grupos. De aquí se genera:
1er. grupo + millones + 2do. Grupo, excepto si el primer grupo consta sólo de '1' en cuyo caso la salida es:
1 millón + pronunciación segundo grupo.

Si el número tiene entre cuatro y seis cifras se divide en dos grupos, uno con los tres dígitos a la derecha y otro del resto y se manda a llamar de nuevo a la función con cada uno de los grupos. De aquí se genera:
1er. Grupo + mil + 2do. grupo

```
CString numero(CString num) {
int a,b;
CString salida="";
TCHAR l1,l2,l3;
b=num.GetLength()-1; % obtener número de dígitos %
a=num.Find('.'); % existe punto decimal? %

% obtener los tres últimos dígitos (en caso de existir) %
l1=num[0];
l2='0';
if (b>0) {l2=num[1];}
l3='0';
if (b>1) {l3=num[2];}

% si existe punto decimal, separar el número en el punto y procesar por
separado %
if (a > 0) {salida=numero(num.Left(a))+"-púnto-"+numero(num.Right(b-
a));b=0;}

% si sólo queda un dígito, leer las unidades %
else if (b==0){
```

Listado 5.1. Función para lectura de números

```
switch(ll) {
case '1': salida="ún-"; break;
case '2': salida="dós-"; break;
case '3': salida="trés-"; break;
case '4': salida="kuátro-"; break;
case '5': salida="sínko-"; break;
case '6': salida="séis-"; break;
case '7': salida="siéte-"; break;
case '8': salida="óCo-"; break;
case '9': salida="nuébe-"; break;
case '0': salida="séro-"; }}

% si hay dos dígitos, leer las decenas y llamar de nuevo a la función
para las unidades %
% en caso de un nombre único (once, doce..) no leer las unidades por
separado %
% si la cifra termina en cero no llamar a la función de unidades %
else if(b==1)
{switch(ll) {
case '1':
            if (l2=='0') {salida=palabra("diez");}
            else if (l2=='1') {salida="ónse");b=0;}
            else if (l2=='2') {salida="dóse";b=0;}
            else if (l2=='3') {salida="trése";b=0;}
            else if (l2=='4') {salida="katórse");b=0;}
            else if (l2=='5') {salida="kínse";b=0;}
            else {salida="diesi";}
            break;
case '2':
            if (l2=='0') {salida="béinte);}
            else {salida="beinti-";}
                  break;
case '3':
      if (l2=='0') {salida="tréinta-";}
      if (l2!='0') {salida=salida+"treintai-";}
                  break;
case '4':
      if (l2=='0') {salida="kuarénta-";}
      if (l2!='0') {salida=salida+"kuarentai-";}
                  break;
case '5':
      if (l2=='0') {salida="sínkuénta-";}
      if (l2!='0') {salida=salida+"sinkuentai-";}
                  break;
case '6':
      if (l2=='0') {salida="sesénta-";}
      if (l2!='0') {salida=salida+"sesentai-";}
                  break;
case '7':
      if (l2=='0') {salida="seténta-";}
      if (l2!='0') {salida=salida+"setentai-";}
                  break;
case '8':
      if (l2=='0') {salida="oCénta-";}
```

Listado 5.1. Función para lectura de números (continuación)

```
      if (l2!='0') {salida=salida+"oCentai-";}
                   break;
case '9':
      if (l2=='0') {salida="nobénta-";}
      if (l2!='0') {salida=salida+"nobentai-";}
                   break;}
      if (l2=='0') {b=0;}      }

% si hay tres digitos leer las centenas y llamar de nuevo a la función
para las decenas %
else if(b==2)
{switch(l1) {
case '1':
      if (l2=='0' & l1=='0') {salida=palabra("cién");b=0;}
      else {salida="siento-";} break;
case '2': salida="dosiéntos-"; break;
case '3': salida=salida+"tresiéntos-"; break;
case '4': salida=salida+"kuatrosiéntos-"; break;
case '5': salida=salida+"kiniéntos-"; break;
case '6': salida=salida+"seisiéntos-"; break;
case '7': salida=salida+"setesiéntos-"; break;
case '8': salida=salida+"oCosiéntos-"; break;
case '9': salida=salida+"nobesiéntos-"; break;}
      if (l2=='0' & l1=='0') {b=0;} }

% si hay entre 4 y 6 dígitos separar en dos grupos y leer por separado %
else if(b>2 & b<6){
            salida=numero(num.Left(b-2))+"míl-";
b=3;}

% si hay entre 7 y 12 dígitos separar en dos grupos y leer por separado %
else if(b>5 & b<12){
            salida=numero(num.Left(b-5))+"míLónes-";
b=6;}
% si hay más de 12 digitos abortar %
else
{salida="un-número-berdaderaménte-gránde-";b=0;}

% si aún no se han leído todas las cifras llamar a la función con la
parte restante %
if (b>0)
      {salida=salida+numero(num.Right(b));}
return(salida);
}
```

Listado 5.1. Función para lectura de números (continuación)

Algunos ejemplos de esta conversión son:

123,456.23 -> sientobeintitrésmíl-kuatrosiéntossinkuentaiséis-púnto-beintitrés

2,000,000.4 -> dós-míLónes-púnto-kuátro

Una vez que se han llevado a cabo estas divisiones preeliminares se tienen uno o más grupos de tres dígitos, de los cuales se obtiene su pronunciación por medio de tablas. Dentro de la función se utiliza la forma en que se leen los números y no la forma en que son escritos.

A continuación se deben obtener a partir del texto ya procesado una secuencia de fonemas que indique la forma en que las palabras escritas deben ser pronunciadas.

Este proceso en el idioma español es sencillo en comparación con otros idiomas, como por ejemplo el inglés o el francés, ya que, a diferencia de estos, en el español existe una relación mucho más directa entre los símbolos utilizados y el fonema que representan, por lo que se requieren muchas menos reglas para obtener los fonemas a ser reproducidos a partir del texto.

Además, en el español existen reglas especificas para obtener la vocal acentuada, así como el acento escrito, lo que también ayuda de gran manera a obtener la entonación de una palabra.

5.1.2. Abreviaturas y símbolos

Los símbolos y abreviaturas se analizan por medio de tablas donde se indica el símbolo o la abreviatura con su lectura correspondiente, por lo que estos se incluyen normalmente dentro de la misma tabla donde se introducen las palabras cuya pronunciación no se ajusta a las reglas normales de conversión de texto a fonemas.

Sin embargo hay que tener cuidado con el símbolo '$', ya que en este caso el símbolo se escribe antes de un conjunto de números, pero se pronuncia después de estos, por lo que éste se trata como un caso especial de lectura de números.

Los símbolos de puntuación son sustituidos por una pausa, que en el sistema se considera como un fonema llamado '-'.

5.2. Conversión del texto filtrado en fonemas

Una vez que se ha filtrado el texto, removiendo símbolos innecesarios y reemplazando todos los símbolos por palabras, se procede a analizar el texto para encontrar la forma en que cada una de estas palabras debe de ser pronunciada. Este proceso se lleva a cabo mediante el uso de las reglas ortográficas del español.

Primeramente se requiere sustituir cada una de las letras por su correspondiente fonema. Una vez que se obtuvo la lectura de las palabras se requiere encontrar la sílaba tónica de la palabra, proceso para el cual se requiere, además, separar la palabra en sílabas para poder encontrar el lugar correcto donde se localiza la acentuación en la palabra.

5.2.1. Los Fonemas de la lengua española

5.2.1.1. Tipos de fonemas de la lengua española

- Vocales: En éstas, el tracto vocal se encuentra abierto y libre de obstáculos, por lo que la única función de la boca es una variación en el timbre. En la lengua española existen 5 vocales principales (a,e,i,o,u), aunque según la región o su posición en la palabra se pueden producir diferencias en su pronunciación (estos son conocidos como alófonos). En la tabla 5.1 se muestran las categorías en las que se subdividen las vocales.

- Diptongos: Es un monosílabo que empieza en la posición de una vocal y cambia hacia la posición de otra vocal diferente. La vocal de mayor abertura es el núcleo silábico y la de menor abertura es la sílaba marginal. Existe una gran controversia sobre se deben considerar como un único fonema o combinación de 2 [23]. Existen tres tipos de diptongos:

		Modo de articulación (apertura)		
		Maxima	Media	Minima
Punto de articulación	Palatales (Parte anterior de la boca)		e	i
	Centrales (Parte central de la boca)	a		
	Velares (Parte posterior de la boca)		o	u

Tabla 5.1 Modo y punto de articulación de las vocales

Crecientes: inician en una vocal débil, seguida de una fuerte (ua, ue, uo, ia, ie, io). En este caso 'u' e 'i' son sustituidos por los alófonos /w/ y /j/ al encontrarse al inicio de una sílaba. (hielo, huevo)

Decrecientes: inician en una vocal fuerte, seguida de una débil (ai, ei, oi, au, eu, ou).

Homogéneos: están formados por dos vocales débiles (iu, ui).

- Consonantes: Las consonantes se clasifican de acuerdo con las categorías mostradas en la tabla 5.2 [23]. La relación de estos fonemas con el texto escrito se analiza en la siguiente seccion.

	Bilabial		Labiodental		Interdental		Dental		Alveolar		Palatal		Velar	
	sor.	son.	sor.	son.	sor.	son.	sor.	son.	sor.	son.	sor.	son.	sor.	son.
Oclusiva	p	b					t	d					k	g
Nasal		m								n		ñ		
Vibrante simple										ř				
Vibrante múltiple										r				
Fricativa			f		θ				s				x	
Lateral Africativas										l	c	λ		

Tabla 5.2. Fonemas de la lengua española

De acuerdo con la manera de articulación se clasifican en:

Oclusivas: El tracto vocal en el punto de articulación y el pasaje se encuentran cerrados. Se produce una exhalación cortante con característica de respuesta transitoria.

Fricativas: El tracto vocal está abierto parcialmente con el velo cerrado. Se genera ruido en el punto de articulación.

Africativas: Existe un cierre inicial del tracto vocal seguido de una expiración gradual que produce turbulencia.

Semivocales: El tracto vocal está parcialmente abierto en el punto de articulación y no se produce turbulencia. Estas se dividen en vibrantes y laterales.

Nasales: El tracto vocal está cerrado y el velo abierto.

También pueden clasificarse por el punto en donde se articulan. Sin embargo, este punto es aproximado y varia entre autores, ya que debido a los fonemas adyacentes puede no alcanzarse perfectamente el punto de articulación y sufrir variaciones o ser variable con el tiempo:

Bilabiales: Se pronuncian con los labios.

Labiodentales: La punta de la lengua hace contacto con la parte posterior del diente incisivo superior.

Interdentales: La lengua se sitúa entre los dientes.

Alveolares: La punta de la lengua se acerca o toca la punta alveolar en el techo de la boca.

Palatares: La lengua se apoya en el paladar.

Velares: La lengua toca el velo del paladar.

Por la acción de las cuerdas vocales:

Sonoras: Se producen vibraciones en las cuerdas vocales.

Sordas: En estas consonantes no existe vibración en las cuerdas vocales.

5.2.1.2. La relación entre letras y fonemas

En las siguientes dos tablas se muestran la relación entre las grafías y los fonemas usadas en el sistema desarrollado. Para facilidad en el procesamiento se sustituyeron los nombres de algunos fonemas, debido al no existir algunos símbolos dentro del estándar ASCII.
/ɾ/ se sustituye por /R/
/ʎ/ se sustituye por /L/

Grafía	**Sonido**
a	/a/
b	/b/
d	/d/
e	/e/
f	/f/
h	Mute
j	/x/
k	/k/
m	/m/

Grafía	**Sonido**
n	/n/
ñ	/ñ/
o	/o/
p	/p/
s	/s/
t	/t/
ü	/u/
v	/b/
z	/s/

Tabla 5.1. Grafías con relación directa a fonemas

En la tabla 5.1 se observan aquellas letras con una relación directa con un fonema. En la tabla 5.2 se muestran aquellas letras que pueden tener más de una lectura de acuerdo con su posición.

Grafía	Reglas
c	/k/ si está seguida de "a","o" o "u" /s/ si está seguida de "e" o "i" /c/ si está seguida de "h". En este caso "ch" se pronuncia con un solo fonema
g	/g/ si está seguida de "a", "o", "u", "üe", "üi" /x/ si está seguida de "e", "i" /g/ si está seguida de "ue" or "ui", la "u" no sera pronunciada.
l	En caso de existir dos 'l' juntas y no se encuentran al final de una palabra se pronuncian en conjunto con el fonema /y/. /l/ en otros casos.
q	/k/ Si existe una 'u' seguida de vocal, la 'u' no será pronunciada.
r	/r/ entre dos vocales /R/ en otro caso. Si existen dos "r" juntas se pronuncian en conjunto /R/
s	/s/ En caso de encontrarse seguida de 'h' se pronuncian en conjunto /sh/.

Tabla 5.2. Grafías con más de un posible fonema

x	/k//s/ También puede pronunciarse como /x/ o /sh/ No existen reglas definidas para estas pronunciaciones, por lo que se deben manejar como excepciones.
w	No se usa en palabras españolas y tiene varias pronunciaciones. Debe manejarse como excepción.
y	/y/ excepto al final de palabra o seguida de consonante /i/ al final de palabra o antes de una cononante
i	/j/ al inicio de sílaba en diptongos (alófono) /i/ en otro caso
u	/w/ al inicio de sílaba en diptongos (alófono) /u/ en otro caso

Tabla 5.3. Grafías con más de un posible fonema (continuación)

Nótese que en la tabla 5.2 se muestran los alófonos /j/ y /w/ para las letras 'i' y 'u' respectivamente, los cuales no son fonemas diferentes sino modificaciones de un mismo fonema, que se producen debido al contexto, para facilitar su pronunciación, auque en muchos sistemas no se toman en consideración, estos dos alófonos son muy comunes, por lo que en la mayoría de los sistemas sí son considerados.

A continuación se dan algunos ejemplos de esta conversión. Ya se tomó en cuenta el módulo de acentuación, descrito en la siguiente sección, en estos ejemplos.

Calle -> káLe
Cereza -> serésa
Chicle -> Cíkle
Guerra -> géRa
Geranio -> jeránio
Quinqué -> kinké
Iraq -> irák

5.2.2. Acentuación en la lengua española

En la lengua española las palabras se pueden clasificar, de acuerdo con la sílaba en la que recae la acentuación, en cuatro tipos diferentes.

Palabras Agudas: La sílaba que lleva el énfasis es la última y llevan acento escrito sólo si terminan en n, s o vocal.

Palabras graves: La sílaba que lleva el énfasis es la penúltima y llevan acento escrito sólo si no terminan en n, s o vocal.

Palabras esdrújulas: La sílaba que lleva el énfasis es la antepenúltima y siempre llevan acento escrito.

Palabras sobre-esdrújulas: Llevan el énfasis antes de la penúltima sílaba y siempre llevan acento escrito.

De acuerdo con estas reglas existen dos casos en los que la acentuación no se encuentra escrita de forma explícita:

- Si la palabra termina en n, s o vocal: En este caso será una palabra grave con acentuación en la penúltima sílaba.

- Si la palabra no termina en n, s o vocal: En este caso será una palabra aguda con acentuación en la última sílaba.

Si la palabra lleva acento escrito ya no se requiere hacer nada para encontrar el acento tónico. En caso contrario sólo se requiere conocer la letra final de la palabra para conocer la sílaba que lleva el acento. Una vez que se obtiene esta letra se puede proceder a obtener la última o la penúltima sílaba de la palabra y finalmente se escribe el acento para facilitar el procesamiento en los demás módulos. Este proceso se muestra en el listado 5.2.

```
/* encontrar la vocal acentuada */

/* paso 1 -> si ya está acentuada no hacer nada*/
if (!palabra_acentuada(salida)) {

/* paso 2 -> penúltima sílaba?? */

found=0;
if (nsv(salida)) {
      for (a=0;a<=b;a++) {
            l2=0;
            l3=0;
            p1=0;
            p2=0;
            l1=salida[a];
            if(a<b) l2=salida[a+1];
            if(a<(b-1)) l3=salida[a+2];
            if(a>0) p1=salida[a-1];
            if(a>1) p2=salida[a-2];
            if (vocal(l1) & !found) {
                  remember=a;
                  look=0;
                  if((a<=(b-2)) & (l2=='u' | l2=='i')) look=a+2;
                  else if(a<b) look=a+1;
                  l1=0;
                  while(look<=b & (!vocal(salida[look]))) look=look+1;
                  l1=salida[look];
                  if(look>=(b-1) & vocal(l1)) found=1;
                  } } }

/* paso 3 -> en caso contrario el acento será en la ultima sílaba */
  if (!found) {
            if (b==0 & vocal(salida[0])) {remember=0;found=1;}
if(b>0){
      if (vocal(salida[b-1])) {remember=b-1;found=1;}
      else if(b>1 & vocal(salida[b-2]) & (!vocal(salida[b-1]) |
salida[b-1]=='i' | salida[b-1]=='u') & !vocal(salida[b])) {remember=b-
2;found=1;}
}}

/* una vez encontrada la vocal acentuada, escribir el acento */
if (found==1) salida=poner_acento(salida,remember);
}
```

Listado 5.2. Encontrar vocal acentuada

5.2.3. Separación silábica en el español

Para separar las palabras en sílabas existen 10 reglas básicas [10 y 23]:

REGLA 1.- En las sílabas, siempre tiene que haber por lo menos una vocal. Sin vocal no hay sílaba.

REGLA 2.- Existen conjuntos de consonantes que deben ser mantenidos juntos y pertenecen siempre a la misma sílaba: br, bl, cr, cl, dr, fr, fl, gr, gl, kr, ll, pr, pl, tr, rr, ch,sh.

REGLA 3.- Cuando una consonante se encuentra entre dos vocales, se une a la segunda vocal.

Ejemplo: une -> u-ne

REGLA 4.- Cuando hay dos consonantes entre dos vocales, cada vocal se une a una consonante excepto si son consonantes consideradas inseparables (ver regla 2)

Ejemplos: componer -> com-po-ner
Aprender -> a-pren-der

REGLA 5.- Si son tres las consonantes colocadas entre dos vocales, las dos primeras consonantes se asociarán con la primera vocal y la tercer consonante con la segunda vocal excepto si la segunda y tercera consonantes están dentro del grupo de inseparables. En las combinaciones de cuatro consonantes adyacentes la frontera silábica se situará entre la segunda y la tercera consonantes y ambos grupos deben pertenecer a la regla 2.

Ejemplos: transporte -> trans-por-te
Cumple -> cum-ple
Inscripción -> ins-crip-ción

REGLA 6.- Las palabras que contienen una h precedida o seguida de otra consonante se dividen separando ambas letras, excepto en aquellas combinaciones que pertenezcan a la regla 2.

Ejemplo. Anhelo -> an-he-lo

REGLA 7.- El diptongo es la unión inseparable de dos vocales. Se pueden presentar tres tipos de diptongos:

1) Una vocal abierta + una vocal cerrada
2) Una vocal cerrada + una vocal abierta
3) Una vocal cerrada + una vocal cerrada

Son diptongos sólo las siguientes parejas de vocales: ai, au, ei, eu, io, ou, ia, ua, ie, ue, oi, uo, ui, iu, ay, ey, oy.

Ejemplo: jaula -> jau-la

La unión de dos vocales abiertas o semiabiertas no forma diptongo, es decir, deben separarse en la segmentación silábica. Pueden quedar solas o unidas a una consonante.

Ejemplo: aéreo -> a-é-reo

REGLA 8.- La h entre dos vocales, no destruye un diptongo.

Ejemplo: ahuyentar -> ahu-yen-tar

REGLA 9.- La acentuación sobre la vocal cerrada de un diptongo provoca su destrucción.

Ejemplo: María -> Ma-rí-a

REGLA 10.- La unión de tres vocales puede formar un triptongo. La única disposición posible para la formación de triptongos es la que indica el esquema:
Vocal cerrada + vocal abierta o semiabierta + vocal cerrada

En caso de ser una combinación diferente a esta, el grupo debe ser separada en dos sílabas.

Sólo las siguientes combinaciones de vocales forman un triptongo: iai, iei, uai, uei, uau, iau, uay, uey.

De acuerdo con estas reglas existen 4 tipos de sílabas:

1) V -> vocal (1 ó 2)

2) VC -> vocal (1 ó 2) + consonante (1 ó 2)

3) CV -> consonante (1 ó 2) + vocal (1,2 ó 3)

4) CVC -> consonante (1 ó 2) + vocal (1,2 ó 3) + consonante (1 ó 2)

6. Conversión de fonemas en voz

La conversión de fonemas a voz consta principalmente del módulo de procesamiento de señales y del módulo de concatenación.

El módulo de procesamiento de señales es el encargado de modificar la duración y el tono de los segmentos pregrabados, mientras que el módulo de concatenación requiere solamente tomar estos segmentos ya procesados y unirlos directamente entre sí para generar la señal de salida

6.1. Modificación de la duración y tono de la señal

Una vez que se tiene una salida fonética que indique la duración y la frecuencia deseadas se debe tomar los segmentos previamente grabados y modificarlos para obtener la señal deseada. Esto permite dar una mayor naturalidad a la salida de voz, así como disminuir en cierta medida las discontinuidades entre segmentos.

Sin embargo, se debe intentar disminuir estas discontinuidades desde la fase de grabación para minimizar lo más posible la cantidad de procesamiento requerido, debido a que estos métodos generan distorsiones en la señal que se hacen más notorias entre mayor sea el cambio que se deba efectuar y que pueden disminuir de manera significativa la comprensión del texto.

Los métodos más importantes para disminuir discontinuidades se pueden dividir en 3 grupos: de predicción lineal, espectrales y de dominio temporal. Los primeros dos métodos requieren obtener cierto número de parámetros de la señal, los cuales son modificados para generar una nueva señal, mientras que el tercero trabaja modificando directamente la señal de voz.

En el caso de este sistema se utilizó el método TD-PSOLA, ya que éste genera buenos resultados con una relativa simplicidad en relación con otros sistemas. Además, se cuenta con la ventaja de que, a diferencia de otros métodos, no se requiere parametrizar la señal, con lo que se reducen las distorsiones a la señal de forma drástica.

6.1.1. Método TD-PSOLA

El nombre TD-PSOLA se deriva de Time Domain Pitch Sinchronous Overlap Add (Suma con traslape sincronizada al periodo en el dominio del tiempo) y es uno de los métodos más recientes de síntesis, al haber sido introducido al inicio de los 90's [8].

Una característica muy importante de este método es que funciona directamente sobre la señal de audio original, en lugar de en una señal parametrizada, como en la mayoría de los demás métodos, con lo que se evita introducir distorsiones debidas al cálculo de los parámetros.

Para poder utilizar este método se requiere tener dentro de la base de datos para cada segmento una lista donde se incluye el inicio de cada ciclo dentro de los segmentos (marcas de periodo). El inicio de cada ciclo se ubica en un máximo local dentro de la señal. Para aquellos segmentos no sonoros se agregan marcas a una distancia predeterminada. También se debe agregar a cada marca una indicación de si corresponde a un segmento sonoro o no.

El método consiste en tomar ventaneados de la señal original, centrados en cada una las marcas de periodo y con longitud de dos periodos. Después se calcula una nueva posición para las marcas de acuerdo con la nueva frecuencia que se desea y se reacomodan los segmentos, previamente multiplicados por una ventana de Hanning, sumando los traslapes. Además para modificar la duración se pueden repetir o eliminar ciertos segmentos de la señal.

Este proceso se puede observar en la figura 6.1, donde se muestra cómo de la señal original (parte central) se reacomodan los centros de las ventanas para aumentar (parte superior) o disminuir (parte inferior) la frecuencia. Para mantener el tiempo se eliminan o repiten ventanas de la señal.

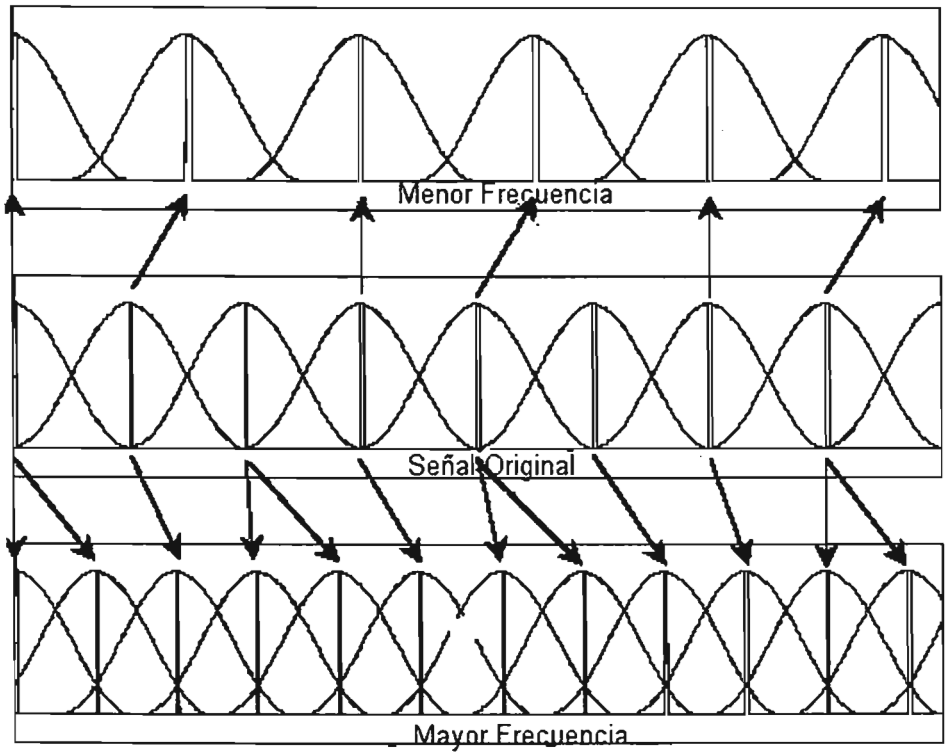

Fig. 6.1. Re-síntesis utilizando el algoritmo PSOLA

Aquí se muestra una versión del algoritmo, esta versión causa un cambio de duración y frecuencia lineal, pero se puede utilizar cualquier tipo de curva. La función se encuentra implementada en Matlab y tiene la siguiente forma:

```
function y = tdpsola(s,pscale,pscale2,tscale,tscale2,fs);
```

Donde 'y' es la señal de salida, 's' la señal de entrada, 'pscale' y 'pscale2' la variación de la frecuencia fundamental inicial y final en porcentaje , 'tscale' y 'tscale2' la variación de la

duración al inicio y al final en porcentaje, 'fs' es la frecuencia de muestreo de la señal de entrada.

```
pm_ps=pm;
pshift=0;
lp=length(pm);
for I = 1:lp
      if (i>1)
            T0=pm(i)-pm(i-1);
      else
            T0=pm(i)-0;
      end
      pin=pscale+ ((pscale2-pscale)*i/lp);
      pshift=pshift-round(T0*(1-(1/pin))); %aqui
      pm_ps(i)=pm(i)+pshift;
end
```

Listado 6.1. Calculo de nueva posición de marcas de periodo

El primer paso es encontrar la nueva posición de las marcas. A partir de la nueva frecuencia se encuentra la duración del nuevo ciclo y sólo se van colocando las marcas a esta nueva distancia de la marca anterior (ver listado 6.1).

Una vez que se tiene la posición de las nuevas marcas, se obtiene la nueva duración de la señal y, de acuerdo con la nueva frecuencia, se calcula cuántos ciclos se deben agregar o eliminar para obtener esta nueva duración. Se debe tomar en cuenta que al modificar la frecuencia de los segmentos se modifica su duración, por lo que el número de segmentos depende de esta nueva frecuencia (ver listado 6.2).

Una vez que se tienen las nuevas marcas se debe agregar también la posición de los ciclos que deben de ser agregados. En esta sección se crea una matriz que indica la posición de inicio y final de cada ventana en la señal original y la posición en que se encontrará en la nueva señal (ver listado 6.3). Aquí puede haber más de una entrada por cada ventana si ésta

se requiere repetir o puede no haber una entrada si la ventana va a ser eliminada. Nótese que el tamaño de la ventana debe ser un múltiplo entero de ciclos, en este caso 2.

```
useds=zeros(1,length(pm_ps)-2);
tin=tscale+ ((tscale2-tscale)/lp);
tot=tin*(pm(2)-pm(1))/(pm_ps(2)-pm_ps(1))
for I = 1 : length(useds)
      tin=tscale+ ((tscale2-tscale)*i/lp);
      avg=pm_ps(i+1)-pm_ps(i);
      new_tscale=tin*(pm(i+1)-pm(i))/avg;
      if (new_tscale>1)
            while(tot>1)
                  useds(i)=useds(i)+1;
                  tot=tot-(pm_ps(i+1)-pm_ps(i))/avg;
            end
            tot=tot+new_tscale
      else
            useds(i)=1;
            while(tot<1)
                  useds(i)=useds(i)-1;
                  tot=tot+(pm_ps(i+1)-pm_ps(i))/avg;
            end
            tot=tot-(1-new_tscale)
      end
end
```

Listado 6.2. Obtener ventanas a duplicar o eliminar

Una vez que se tiene la longitud y posición de las nuevas ventanas, se va tomando cada una de las ventanas originales, se multiplica por una ventana de Hanning, mostrada previamente en el listado 4.2, para disminuir las discontinuidades en el punto de unión y se van sumando los valores que se traslapen (ver Listado 6.4).

```
Start=1;
count=1;
for i=1:length(useds)
      if (useds(i)>0)
            final(count,:)=[start pm(i) pm(i+2) 0];
            count=count+1;
            start=start+pm_ps(i+1)-pm_ps(i)+1;
      end
      for j=2:useds(i)
            final(count,:)=[start pm(i) pm(i+2) mod(j,2)];
            count=count+1;
            start=start+pm_ps(i+1)-pm_ps(i)+1;
      end
end
numfrm=size(final,1);
```

Listado 6.3. Asignar ventanas a marcas de periodo

```
A=final(:,1) + (final(:,3)-final(:,2)+1);
ylen=max(A);
y=zeros(ylen,1);
w=zeros(size(y));
for i = 1 : numfrm
   start=final(i,1);
   len=final(i,3)-final(i,2)+1;
   wgt=hanning(len);
   frm=s(final(i,2):final(i,3));
   if (final(i,4))
     frm=wrev(frm);
   end
   y(start:start+len-1)=y(start:start+len-1)+frm.*wgt;
   w(start:start+len-1)=w(start:start+len-1)+wgt;
end
```

Listado 6.4. Sumar ventanas para generar la nueva señal

Por último se divide la señal resultante entre el valor del traslape de las ventanas en cada punto para obtener la amplitud final en cada punto (Listado 6.5).

En la figura 6.2 se muestra un ejemplo de señal con este algoritmo. En el primer ejemplo la duración es la misma pero las separación entre armónicos es la doble. En el segundo caso la frecuencia es la misma, pero se ha duplicado el número de muestras.

```
for i=1:ylen
      if w(i)==0
            w(i)=1;
      end
      y(i)=y(i)/w(i);
end
```

Listado 6.5. Normalizar señal

Para reducir las discontinuidades con este algoritmo se toma la frecuencia fundamental de los segmentos que se van a unir y se calcula su diferencia. Una vez que se tiene esta diferencia se obtienen las frecuencias fundamentales que se deben de tener al inicio y al final del segmento para minimizar discontinuidades y se ejecuta el algoritmo con estos valores.

Después de obtener el segmento modificado en frecuencia, se toma la amplitud de cada segmento en el punto de unión y se modifica, de forma que coincida en ambos segmentos. Debido a que el punto de corte es en un máximo local se asegura que se obtendrá una curva suave.

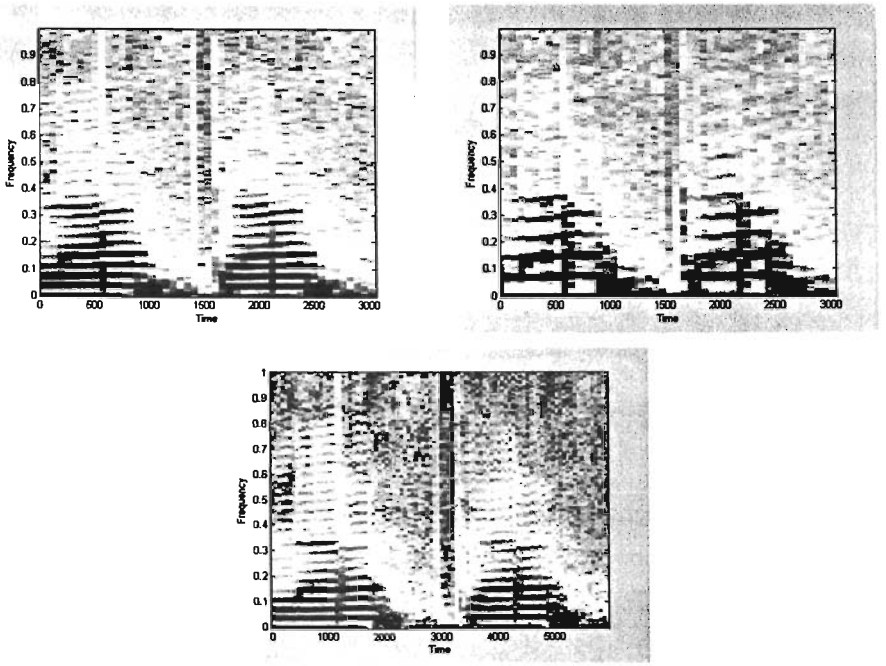

Fig. 6.2. Algoritmo PSOLA. (a) Señal original, (b) Doble de la frecuencia, misma duración, (c) Misma frecuencia, doble duración

Conclusiones

En esta tesis se llevaron a cabo una serie de mejorías a un sistema de síntesis de voz por concatenación de difonemas, el cual a partir de un texto cualquiera en lengua española puede generar una salida de voz.

En este sistema se utilizaron difonemas, debido a que estos son capaces de dar buenos resultados con una cantidad aceptable de segmentos. El número teórico de difonemas es el cuadrado de los fonemas existentes en el español (23^2=529), pero debido a que existen muchas combinaciones que no se pueden dar (por ejemplo rrr o nñ) el número real es menor. Sin embargo, en el caso de este sistema se utilizaron dos versiones de cada vocal (acentuada o no), con lo que el número de sonidos utilizados asciende de 23 a 28, lo que incrementa considerablemente la cantidad de unidades requeridas, pero disminuye la necesidad de modificar la señal de entrada original y, por tanto, disminuye también considerablemente las distorsiones añadidas por el sistema a la señal de salida.

Se generó un módulo de conversión de texto a fonemas, para permitir entradas de texto de tipos diferentes al texto llano (principalmente para el análisis de números). Además, se añadió un módulo de procesamiento de señales cuya función es modificar los segmentos pregrabados para disminuir las discontinuidades en los puntos de unión inherentes a los sintetizadores de voz por concatenación y mejorar la calidad del audio generado. Este módulo también puede ser utilizado para modificar las características de tono y velocidad de la señal de salida de forma independiente, por lo que al modificar el tono no se modifica la duración y viceversa.

Comparando este sistema con otras opciones gratuitas existentes, basadas en algoritmos similares, se observa que aunque las discontinuidades no se disminuyen de forma tan efectiva como en éstos, la distorsión generada al modificar la señal es mucho menor, lo que ayuda a mejorar la comprensión del audio.

Bibliografía

I- Libros y material impreso

(1) del Río Ávila, Fernando **Diseño de un Sintetizador de Voz por Difonemas**
UNAM, Tesis de Licenciatura, 2002

(2) Keller, E. et al **Improvements in Speech Synthesis.**
Willey and Sons, 2002, Inglaterra

(3) Dutoit, Thierry **An introduction to text-to-speech synthesis.**
Kluwer Academic,1997, Holanda

(4) Holmes, J.N. **Speech synthesis and Recognition.**
Von Nostrand Reinbold,1988,Inglaterra

(5) Goldstein, Bruce **Sensation and Perception.**
5th Edition, Brooks Cole Publishing Co., 1999, U.S.A.

(6)Goncharoff, Vladimir / Gries, Patrick **An Algorithm For Accurately Marking Pitch Pulses In Speech Signals.**
Proceedings of the IASTED International Conference in Signal and Image Processing (SIP'98) October 28-31, 1998 - Las Vegas, Nevada, USA pags. 281-234

(7) Bailly, G. / Benoit, C. **Talking Machines: Theories, Models and Designs.**
Ed. North Holland, 1992, Holanda

(8) Moulines, E. / Charpentier, F. **Pitch Synchronous Waveform Processing Techniques for Text-to-Speech Synthesis Using Diphones.**
Speech Communication 9, pags. 453-467, 1990

(9) Apuntes de la materia "Procesamiento Digital de Voz"
Herrera Camacho, Abel

II- Referencias en Internet

(10) Lemmetty, Sami **Review of Speech Synthesis Technology.**
http://www.acoustics.hut.fi/~slemmett/dippa/

(11) **Zur Geschichte der Sprachsynthese 1770-1970.**
http://www.ling.su.se/staff/hartmut/kempln.htm

(12) G. Stork, David **HAL's Legacy: 2001's Computer as Dream and Reality.**
http://mitpress.mit.edu/e-books/Hal/

(13) Klatt, Dennis H. **Review of text-to-speech conversion for English**
http://www.mindspring.com/~ssshp/ssshp_cd/dk_737a.htm

(14) Mattingly, Ignatius **Speech Synthesis for Phonetic and Phonological Models**
http://www.mindspring.com/~ssshp/ssshp_cd/im_home.htm

(15) Lleida Solano, Eduardo **Conversion Texto-Voz.**
http://www.gtc.cps.unizar.es/~eduardo/investigacion/voz/ctv.html

(16) Cano,Rafael **Apuntes de Gramatica Española.**
http://members.ferrara.linux.it/elfenor/espanol/gramatica.htm

(17) Castejon Lapeyra, F. Et al **Un conversor de texto-voz para español.**
http://www.tid.es/presencia/publicaciones/comsid/esp/articulos/vol52/artic8/8.html

(18) Figueroa Mora, Karina **Tesis de Licenciatura.**
http://www.tid.es/presencia/publicaciones/comsid/esp/articulos/vol52/artic8/8.html

(19) Guerequeta, Rosa / Vallecillo, Antonio **Técnicas de diseño de algoritmos**
Servicio de Publicaciones de la Universidad de Málaga. 1998
http://polaris.lcc.uma.es/~av/Libro/indice.html

(20) Donovan, Robert Edward **Trainable Speech Synthesis**
Cambridge University, Doctor of Philosophy dissertation
http://citeseer.ist.psu.edu/donovan96trainable.html

(21) Türk, Oytun **New Methods For Voice Conversion**
Boğaziçi University, Master of Science Thesis
http://www.busim.ee.boun.edu.tr/~speech/thesis/oytun_turk.pdf

(22) Black, Allan W. / Lenzo, Kevin A. **Building Voices in the Festival Speech Synthesis System**
Carnegie Mellon University, 2000
http://www.ling.ohio-state.edu/courses/materials/795X/festvox/festvox_toc.html

(23) Ríos Mestre, Antonio **La Transcripción Fonética Automática Del Diccionario Electrónico De Formas Simples Flexivas Del Español: Estudio Fonológico En El Léxico**
Universitat Autònoma de Barcelona
http://elies.rediris.es/elies4/index.htm