Instituto Politécnico Nacional

Centro de Investigación en Computación

Laboratorio de procesamiento digital de señales

Conversión de voz y separación de locutores

T E S I S

QUE PARA OBTENER EL GRADO DE

MAESTRÍA EN CIENCIAS EN INGENIERÍA DE CÓMPUTO CON OPCIÓN EN SISTEMAS DIGITALES

P R E S E N T A

ING. CÉSAR EDGAR MONTAÑO SÁNCHEZ

DIRECTOR (ES) DE TESIS:
DR. JOSÉ LUIS OROPEZA RODRÍGUEZ
DR. SERGIO SUÁREZ GUERRA

MÉXICO, D.F. **ENERO 2016**

SIP-14 bis

INSTITUTO POLITÉCNICO NACIONAL

SECRETARÍA DE INVESTIGACIÓN Y POSGRADO

ACTA DE REVISIÓN DE TESIS

En la Ciudad de México, D.F. siendo las 10:00 horas del día 14 del mes de diciembre de 2015 se reunieron los miembros de la Comisión Revisora de la Tesis, designada por el Colegio de Profesores de Estudios de Posgrado e Investigación del:

Centro de Investigación en Computación

para examinar la tesis titulada:

"Conversión de voz y separación de locutores"

Presentada por el alumno(a):

Montaño	**Sánchez**	**César Edgar**
Apellido paterno	Apellido materno	Nombre(s)

Con registro:

B	1	3	0	0	7	9

aspirante de: **MAESTRÍA EN CIENCIAS EN INGENIERÍA DE CÓMPUTO CON OPCIÓN EN SISTEMAS DIGITALES**

Después de intercambiar opiniones los miembros de la Comisión manifestaron ***APROBAR LA TESIS***, en virtud de que satisface los requisitos señalados por las disposiciones reglamentarias vigentes.

LA COMISIÓN REVISORA

Directores de Tesis

Dr. Sergio Suárez Guerra

Dr. José Luis Oropeza Rodríguez

Dr. Oleksiy Pogrebnyak

Dr. Ricardo Barrón Fernández

M. en C. Pablo Manrique Ramírez

Dr. Francisco Hiram Calvo Castro

PRESIDENTE DEL COLEGIO DE PROFESORES

Dr. Luis Alfonso Villa Vargas

INSTITUTO POLITECNICO NACIONAL
CENTRO DE INVESTIGACION
EN COMPUTACION
DIRECCION

INSTITUTO POLITÉCNICO NACIONAL

SECRETARÍA DE INVESTIGACIÓN Y POSGRADO

CARTA CESIÓN DE DERECHOS

En la Ciudad de México D.F. el día 07 del mes Enero del año 2016, el (la) que suscribe Montaño Sánchez César Edgar alumno (a) del Programa de Maestría en Ciencias en Ingeniería de Cómputo con opción en Sistemas Digitales con número de registro B130079, adscrito a Centro de Investigación en Computación, manifiesta que es autor (a) intelectual del presente trabajo de Tesis bajo la dirección de Dr. José Luis Oropeza Rodríguez y de Dr. Sergio Suárez Guerra y cede los derechos del trabajo intitulado "Conversión de voz y separación de locutores", al Instituto Politécnico Nacional para su difusión, con fines académicos y de investigación.

Los usuarios de la información no deben reproducir el contenido textual, gráficas o datos del trabajo sin el permiso expreso del autor y/o director del trabajo. Este puede ser obtenido escribiendo a la siguiente dirección b130079@sagitario.cic.ipn.mx. Si el permiso se otorga, el usuario deberá dar el agradecimiento correspondiente y citar la fuente del mismo.

César Edgar Montaño Sánchez

Nombre y firma

Resumen

El análisis de señales de voz con el fin de hacer el reconocimiento del locutor va teniendo mayor número de aplicaciones en diversas áreas de la industria, ejemplos claros de sistemas que ocupan el reconocimiento de locutor son: sistemas de mandos por voz, sistemas de seguridad por autenticación de locutor, sistemas de marcado telefónico, control de robots; entre otros.

De la misma forma los sistemas producción de voz sintética ya tienen un auge aun mayor, dado su uso en sistemas como: contestador telefónico, sistemas de ayuda en dispositivos móviles, síntesis de mensajes de texto, etc.

De la unión de ambos tipos de sistemas surgen los sistemas de conversión de voz, donde se utilizan ambos estudios para realizar un reconocimiento de locutor o locutores para después reproducir el contenido fonético del mensaje con una voz sintetizada. La voz que será sintetizada se producirá con las características fonéticas de cualquiera de los locutores con los que cuente el sistema.

El propósito de este trabajo es disponer de una arquitectura propuesta de caracterización de la voz de locutores, asimismo una vez lograda la correcta caracterización se pretende realizar la imitación de voz de un locutor al repetir el texto acotado por el otro locutor. Los principales parámetros que se analizan en este trabajo son los Codificadores predictivos lineales y los coeficientes cepstrales en la escala de Mel.

Las técnicas empleadas son: Cuantificación vectorial, modelos ocultos de Markov discretos y de densidad continua, además de arquitecturas que combinan ambos análisis. Siendo la técnica de modelos ocultos de Markov de densidad continúa que utiliza coeficientes cepstrales en escala de Mel la que alcanzó el mayor porcentaje de exactitud en los corpus utilizados.

Para la etapa de síntesis se probaron técnicas de síntesis por formantes, síntesis basadas en LPCs y síntesis por concatenación. El método utilizado para la síntesis fue el de concatenación, siendo las unidades lingüísticas utilizadas los di fonos.

Los resultados cualitativos alcanzados tienen porcentaje de efectividad del 98% para la arquitectura total del sistema, dicho porcentaje de efectividad fue registrado con la realización de pruebas MOS, dichos resultados en esta tesis podrán ser utilizados en otras aplicaciones para traducción automática, conversión, doblaje y reconocimiento de voz.

Abstract

The purpose of this thesis is to propose an architecture for voice conversion tasks; also, once the correct characterization is achieved, it is pretended to imitate the voice of one speaker repeating the text bounded by another one.

The main parameters analyzed in this work are: Linear Predictive Coefficients and the Cepstral Coefficients in the Mel scale.

The following techniques were used: Vector Quantization, Discrete and Continuous Density Hidden Markov Models. In addition a set of architectures that combine both analyses. The technique of Continuous Density Hidden Markov Models with cepstral coefficients using Mel scale was used in the corpus because reached the highest accuracy rate.

For the synthesis, techniques like formant synthesis, LPC's synthesis and synthesis by concatenation were tested. The method used for the synthesis was concatenation, linguistic units were di phones.

The speaker and isolated word recognition systems were implemented and applied to a corpus of words, a corpus of words made up places (countries and states), 20 recordings for each word, and two speakers was used for the experiment were the elements involved in the experiment. Performing feature extraction of MFCC type codebooks were established for the speaker recognition and the recognition of isolated words contained in the corpus, the strategy used for speaker recognition was the vector quantization, while for recognition CDHMM isolated word was used. The results shown in this thesis focus on the rate of effectiveness recognition announcer, resulting in 97% effectiveness in speaker recognition, and 99% in isolated word recognition.

Keywords.- Vector quantization, LPC, HMM, DHMM, CDHMM, MFCC, Diphones, speaker recognition, recognition of isolated words and Voice Conversion.

Agradecimientos

En primer lugar quiero agradecer a mi Alma Mater el Instituto Politécnico Nacional el haberme brindado mi formación académica de enseñanza hasta esta estancia, por concederme el honor de pertenecer a esta comunidad y facilitar el desarrollo de mis actividades académicas, deportivas y culturales.

Agradezco a mis padres por darme la vida y apoyarme siempre en todo momento, a mis hermanos por compartir todos los momentos malos y buenos de mi vida con ustedes, gracias por siempre estar conmigo, “este logro es por ellos y para ellos”.

Agradezco a todos mis amigos por su apoyo, consejos y recomendaciones, pero sobre todo por compartir conmigo el muy preciado tiempo de sus vidas.

Agradezco mis directores de tesis el Dr. José Luis Oropeza Rodríguez y Dr. Sergio Suárez Guerra por los conocimientos que me brindaron durante mi trayectoria académica y por el esfuerzo y dedicación para consolidar este proyecto de investigación.

Agradezco a todos los profesores que colaboraron directa o indirectamente en mi formación académica, por su guía, sugerencias y consejos durante este arduo camino.

Agradezco al Centro de Investigaciones en Computación, la oportunidad otorgada para la realización de mis estudios de maestría y las facilidades que permitieron llevar a buen término esta etapa de mi vida.

Agradezco al Consejo Nacional de Ciencia y Tecnología por el apoyo recibido durante mis estudios de maestría.

Índice de Contenido

Índice de Figuras ..... **IV**

Índice de Tablas ..... **VI**

CAPÍTULO 1. INTRODUCCIÓN **1**

1.1 Antecedentes **1**

1.2 Planteamiento del problema. **7**

1.3 Justificación. **8**

1.4 Hipótesis. **8**

1.5 Objetivos. **9**

1.5.1 Objetivo general 9

1.5.2 Objetivos particulares. 9

1.6 Alcances del trabajo **10**

1.7 Contribuciones. **10**

1.8 Organización del trabajo **10**

CAPÍTULO 2. ESTADO DEL ARTE. **12**

2.1 Sistemas de conversión de voz basados en unidades lingüísticas. **13**

2.2 Sistemas de conversión de voz basados en partición de espacios acústicos **14**

2.3 Sistemas de conversión de voz híbridos **15**

2.4 Composición de los sistemas de conversión de voz **16**

2.4.1 Parámetros característicos de señal de voz 16

2.4.2 Modelos para el reconocimiento de voz 16

2.4.3 Modelos para el reconocimiento de locutor 17

2.4.4 Sistemas de texto a voz 17

2.5 Arquitecturas de Reconocimiento de Locutor **18**

2.6 Arquitecturas de sistemas de reconocimiento y conversión de voz. **19**

CAPÍTULO 3. MARCO TEÓRICO **24**

3.1 Pre-procesamiento de señal de voz **26**

3.1.1 Preénfasis 26

3.1.2 Ventaneo 28

3.1.3 Transformada rápida de Fourier (FFT) 30

3.1.4 Transformada discreta del coseno (DCT) 32

3.2. Coeficientes de predicción lineal (LPC) **33**

3.2.1 Algoritmo de Levinson-Durbin 36

3.3 Coeficientes cepstrales o cepstrum **37**

3.4 Coeficientes cepstrales en escala de Mel (MFCC) **38**
3.5 Sistemas de reconocimiento basados en cuantificación vectorial (VQ) **41**
3.5.1 Ventajas de la representación VQ 43
3.5.2 Desventajas de la representación VQ 43
3.5.3 Algoritmo de Lloyd generalizado, o K-means 44
3.6 Modelos ocultos de Markov (HMM) **45**
3.7 Técnicas de síntesis de voz **50**
3.7.1. Síntesis articulatoria 51
3.7.2 Síntesis por formantes 52
3.7.3 Síntesis por concatenación 56
3.8. HTK **60**
CAPÍTULO 4. METODOLOGÍA Y DESARROLLO DE LA INVESTIGACIÓN. **63**
4.1 Elección del corpus de palabras para 2 locutores. **64**
4.2 Reconocimiento del locutor **65**
4.3 Reconocimiento de palabras aisladas **69**
4.4 Síntesis de la palabra reconocida. **73**
CAPÍTULO 5. PRESENTACIÓN Y DISCUSIÓN DE RESULTADOS. **76**
5.1 Características de los corpus de voces. **76**
5.1.1 Características LPC del corpus de Estados 78
5.1.2 Características MFCC del corpus de Dígitos 79
5.2 Resultados de reconocimiento de locutor. **80**
5.3 Resultados en reconocimiento de palabras. **83**
5.4 Resultados dela conversión de voz. **85**
CAPÍTULO 6. CONCLUSIONES Y TRABAJO FUTURO **90**
6.1 Conclusiones. **90**
6.2 Trabajo futuro. **92**
REFERENCIAS BIBLIOGRÁFICA **93**
ANEXO A. Arquitectura del procesador TMS320C6713 ..... **97**
Arquitectura VLIW **101**
Arquitectura VelociTI **103**
Descripción del CORE de Procesador **104**
Fetch, Distpatch y Execute **104**
Registros de propósito general **105**
Unidades Funcionales **106**
Registros de Control **108**
Test **109**
Control de interrupciones **110**

Descripciones de las Caches del Procesador 110
Propiedades y uso del EDMA 111
Timers, EMIFs, McBSPs, GPIOs, HPI, PCI; entre otros. 111

ANEXO B. Procedimiento Para Realizar Reconocimiento de Palabras Con HTK 113

Etapa de entrenamiento usando HTK 114
Etapa de reconocimiento usando HTK 115

ANEXO C. Algoritmo de FFT Mediante Decimación en Frecuencia 117

ANEXO D. Programa para la grabación del corpus de digito en DSP 122

Índice de Figuras

Figura 1. Esquema de un sistema de identificación de locutor. 6
Figura 2. Esquema de un sistema de verificación de locutor. 6
Figura 3. Esquema de conversión de voz basado en selección de unidades. 13
Figura 4. Esquema de conversión de voz basado en partición de espacio 14
Figura 5. Sistema de conversión de voz propuesto por Dutoit (T. Dutoit 2007) 15
Figura 6. Sistema conversor de texto a voz 17
Figura 7. Diagrama General de Reconocimiento de Locutor 18
Figura 8. Tareas fundamentales del reconocimiento de locutor: identificación y verificación 19
Figura 9. Esquema general del sistema de reconocimiento de voz 19
Figura 10. Diagrama a bloques del sistema propuesto en (Eun-Kyoung Kim 1997) 20
Figura 11. Sistema de Conversión de voz mostrado en (Yannis Stylianou 1998) 20
Figura 12. Algoritmo de conversión de voz basada en LPC (Alexander Kain 2001) 21
Figura 13. Conversión de Voz basada en Concatenación (Kei Fujii 2007) 21
Figura 14. Arquitectura general de conversión de locutor implementada 23
Figura 15. Esquema de extracción de características de la señal de voz 24
Figura 16. Filtro de Preénfasis 27
Figura 17. Diagrama a bloques del sistema para un filtro de preénfasis 28
Figura 18. Tipos de ventanas 29
Figura 19. Respuesta en Frecuencia de Ventana de Hamming 29
Figura 20. Algoritmo Levinson-Durbin (Luis, 2010) 36
Figura 21. Banco de filtros para MFCC 39
Figura 22. Algoritmo para cálculo de MFCC 41
Figura 23. Ejemplo de cuantificación vectorial 42
Figura 24. División de espació bidimensional 42
Figura 25. Funcionamiento de cuantificación vectorial 43
Figura 26. División en regiones 44
Figura 27. Modelo oculto de Markov 46
Figura 28. Modelo oculto de Markov de densidad continua (CDHMM) 50
Figura 29. Estructura básica de un sintetizador por formantes en cascada. 53
Figura 30. Estructura básica de un sintetizador de formantes en paralelo 54
Figura 31. Modelo parcas (Laine U., 1982) 55
Figura 32. Módulo de un TTS basado en síntesis por concatenación 57
Figura 33. Concatenación TD-PSOLA de dos unidades acústicas. 60
Figura 34. Diagrama de las librerías de HTK (Roberto Carrillo Aguilar, 2007) 61
Figura 35. Resultados del reconocimiento con HResults en HTK 62

Figura 36. Arquitectura general de conversión de locutor implementada 63
Figura 37. Esquema de reconocimiento de locutor 65
Figura 38. Algoritmo de bipartición 67
Figura 39. Algoritmo para generación de libros código 67
Figura 40. Etapa de reconocimiento mediante cuantificación vectorial 68
Figura 41. Esquema general de reconocimiento por cuantificación vectorial 69
Figura 42. Esquema de reconocimiento de palabras 69
Figura 43. Algoritmo para cálculo de coeficientes MFCC 70
Figura 44. HMM utilizado para la palabra "Berlín" 71
Figura 45. Esquema de síntesis de voz 73
Figura 46. Programa para etiquetado manual 74
Figura 47. Sistema TTS implementado 75
Figura 48. Archivo de configuración para HCOPY 80
Figura 49. Llamada a librería HCOPY de HTK 80
Figura 50. Ejemplo de libro código generado para un locutor 81
Figura 51. Porcentaje de reconocimiento de palabras aisladas en corpus de estados 83
Figura 52. Porcentaje de reconocimiento de palabras aisladas en corpus de dígitos 85
Figura 53. Arquitectura de DSP TMS320C6713 97
Figura 54. Archivos de características MFCC 114
Figura 55. Uso de la función HInit 114
Figura 56. Uso de la función HRest 115
Figura 57. Instrucciones para reconocimiento de palabras aisladas usando HTK 115
Figura 58. Ejemplo del archivo de gramática utilizada por HTK 116
Figura 59. Archivo de resultados de reconocimiento con HResults 116
Figura 60. Calculo de la DFT de N puntos con 2 DFT de (N/2) puntos con N=8 120
Figura 61. Descomposición de una DFT de 8 puntos en cuatro DFT de 2 puntos 120
Figura 62. Esquema para un DFT de 8 puntos 121
Figura 63. Calculo de mariposa en la última etapa 121
Figura 64. Creación de proyecto en CC5 122
Figura 65. Parámetros para el compilador en CC5 122
Figura 66. Parámetros para el linker en CCS5 123
Figura 67. Archivos del proyecto de CCS5 para grabar corpus de dígitos 123

Índice de Tablas

Tabla 1. Programas de reconocimiento del habla de sistemas operativos **3**
Tabla 2. Programas que realizan conversión de voz **4**
Tabla 3. Técnicas de separación de locutores **5**
Tabla 4. Características del corpus de Estados **76**
Tabla 5. Corpus Estados **77**
Tabla 6. Características del corpus de dígitos **77**
Tabla 7. Corpus Dígitos **78**
Tabla 8. Características de pre-procesamiento de señal de voz para LPC´s **78**
Tabla 9. Características de pre-procesamiento de señal de voz para MFCC´s **79**
Tabla 10. Matriz de confusión para reconocimiento de locutor en corpus Estados **81**
Tabla 11. Matriz de confusión para reconocimiento de locutor en corpus Dígitos **82**
Tabla 12. Matriz de confusión para el reconocimiento de palabras "Corpus estados" **84**
Tabla 13. Matriz de confusión para el reconocimiento de palabras "Corpus digitos" **85**
Tabla 14. Ponderación de calidad de audio para prueba MOS **86**
Tabla 15. Prueba MOS para calidad del sintetizador **86**
Tabla 16. Porcentaje de efectividad en la conversión de voz entre locutores **87**
Tabla 17. Porcentaje cuantitativo de efectividad para la conversión de voz **87**
Tabla 18. Comparativa entre objetivos y resultados alcanzados **91**
Tabla 19. Descripción de elementos de TMS320C6713 ..... **98**
Tabla 20. Ventajas y Desventajas de Arquitectura VLIW **102**
Tabla 21. Unidades Operativas **108**
Tabla 22. Registro de Control **109**
Tabla 23. Registros de Control Extendidos **109**
Tabla 24. Interrupciones del TMS320C6713 ..... **110**

CAPÍTULO 1. INTRODUCCIÓN

1.1 Antecedentes

El reconocimiento automático del habla es uno de los temas que aborda principalmente la inteligencia artificial y tiene como objeto de estudio la comunicación hablada entre seres humanos y ordenadores.

La ciencia de la computación, se apoya de las ramas de la lexicología: Onomasiología (sintáctica), semasiología (semántica) y pragmática; además de otras ramas que estudian las señales de voz: acústica, fonética, fonológica, fisiología y el procesamiento de señales. En conjunto pueden realizar la eliminación de incertidumbres o ambigüedades para realizar una excelente interpretación de los mensajes de voz. Mientras que el procesamiento digital de señales es utilizado en áreas afines a la ingeniería de Cómputo.

Por lo tanto, los sistemas de reconocimiento automático del habla son herramientas computacionales capaces de procesar la señal de voz emitida por el ser humano y reconocer la información contenida en ésta, convirtiéndola en texto o emitiendo órdenes que actúan sobre un proceso.

Muchas técnicas de conversión de voz han sido propuestos desde la formulación original del problema de conversión de voz (D. G. Childers, 1985). Childers propuso una técnica que involucra el mapeo de características acústicas de un locutor origen hacia un locutor destino. Un año más tarde Shikano propuso el uso de técnicas de cuantificación vectorial y libros código (K. Shikano, 1986).

El 1990 Abe propuso el término de CVCS (Crosslingual Voice Conversión System) usando sujetos bilingües (M. Abe K. S., 1990). Para finales de la década de los 90´s

Arsal propuso un modelo usando líneas de frecuencia espectral para realizar una representación del envolvente espectral, mostrando los resultados obtenidos en el algoritmo STASC (Arslan, 1999), un año antes Stylianou (Stylianou, 1998) utilizo modelos de mixturas Gaussianas (GMM) combinado con MFCC como alternativa de la envolvente espectral.

En 2001, Toda (T. Toda, 2001) propuso una combinación de representación espectral y técnicas de conversión de voz llamada STRAIGHT (Speech Transformation and Representation using Adaptative Interpolation of weiGHTed spectrum), que permitió la manipulación espectral y parámetros rítmicos.

Más recientemente Rentzos (D. Rentzos, 2003), Ye (Young, 2006), Rao (Yegnanarayana, 2006) y Zhang (M. Zhang, 2009) las contribuciones han sido enfocadas a tecnicas probabilísticos como GMM, Codebooks, RNA y DFW entre otros.

Desde el año 1990 se comercializan sistemas de reconocimiento del habla en los sistemas computacionales, pero no han sido populares debido a que los usuarios prefieren el uso del teclado, el ratón y últimamente las pantallas dactilares para ingresar comandos o texto a los sistemas informáticos. Via voice[1], y NaturallySpeaking[2] son un par de programas de renombre y con una historia en común en este aspecto.

Via voice es un producto desarrollado por IBM el cuál sustituyó a otro desarrollo llamado VoiceType en 1997, mismo año que se introdujo por primera vez al público en general. Dos años más tarde, en 1999, IBM lanzó una versión gratuita de ViaVoice. En 2003, IBM otorgó a ScanSoft, (los creadores de Dragon NaturallySpeaking) los derechos exclusivos de distribución mundial de ViaVoice para Windows y Mac OS X. Dos años más tarde, Nuance se fusionó con ScanSoft.

[1] http://www-01.ibm.com/software/pervasive/viavoice.html

[2] http://www.naturalspeak.com/

Tabla 1. Programas de reconocimiento del habla de sistemas operativos[3]

Microsoft Windows	***Macintosh***	***Linux***
VoiceAttack	*Dragon Dictate*	*Julius*
VAC-Voice activated commands	*MacSpeech Dictate Medical*	*ViaVoice*
Voice Finger	*MacSpeech Dictate Legal*	*CMU Sphinx*
WSRToolkit	*MacSpeech Scribe*	*Open Mind Speech (antes FreeSpeech)*
Trigramtech	*iListen*	*VoxForge*
Vocola	*Speakable items*	*GnomeVoiceControl*
Auditory Sciences	*Via Voice*	*CVoiceControl*
Dragon NaturallySpeaking	*Voice Navigator*	*PerlBox Voice*
Freesr Speech Recognition Software	*M68331 Voice Recognition System*	*Platypus*
SpeechGear's Interact	*Macintosh application MacSpeech*	*Vedic*
Sonic Extractor from Digital Syphon		*Wizzscribe SI*
Speech Magic		*DynaSpeak*
Tazti		*LumenVox Speech Engine*

Por otro lado, la conversión de hablante (también llamada conversión de voz o conversión de locutor), es una técnica usada para cambiar o modificar la identidad del hablante; es decir, lo pronunciado por un locutor es transformado para que suene como si lo hubiera articulado otro.

[3] http://en.wikipedia.org/wiki/List_of_speech_recognition_software

En la actualidad solo existen comercialmente un par de programas que realizan conversión de hablante.

Tabla 2. Programas que realizan conversión de voz

Programa	***Versión***	***Fabricante***	***Página Web***
Voice Changer Software Diamond	*7.0.37*	*AV Soft*	*http://www.audio4fun.com/*
Voxal Voice Changer		*NCH Software*	*http://www.nchsoftware.com/ voicechanger/index.html*

Sin embargo, hay una cantidad enorme de programas que convierten texto al habla, lo cual significa que a partir de un texto, se realiza la síntesis de voz, es decir, el ordenador lee archivos de texto mediante concatenación de letras y a su vez de palabras con bases de datos de voz que por lo general no son las mismas que las personas que escriben dichos textos. Los dos productos más utilizados a nivel comercial son Loquendo[4] y Google Translate[5]

El equipo de investigación de Google ha desarrollado un sistema de traducción estadístico propio que actualmente incluye un sistema de traducción por voz para que las personas con algún tipo de limitación tengan acceso a la información en cualquier idioma. Con el traductor Google los textos escritos en el griego, devanagari, cirílico, árabe y el Chino, son transcritos de forma automática de los equivalentes fonéticos escritos en alfabeto latino.

[4] http://www.nuance.es/empresas/solucion/soluciones-de-atencion-al-cliente/servicios-y-soluciones/soluciones-de-recepcion-de-llamadas/loquendo-small-business-bundle/text-to-speech/index.htm

[5] http://translate.google.com/

A mediados de noviembre de 2009, Google Translate cambió a un aspecto más minimalista. El formulario de traducción cambió de aspecto y posición. Se incorporó una función más rápida donde al escribir cada palabra, automáticamente se va traduciendo por sílaba, a mediados de 2010 se incorporó la función de escuchar cualquier palabra, frase o texto en cualquier idioma.

En la década pasada, se han realizado decenas de técnicas para la separación de locutores, algunas de ellas se mezclan entre sí para realizar una mayor eficiencia (Yuxuan W., 2012).

Tabla 3. Técnicas de separación de locutores

Siglas	***Nombres de la Técnica***
CASA	*Computational Auditory Scene Analysis*
IBM	*Ideal Binary Mask*
AMS	*Amplitude Modulation Spectrogram*
SVMs	*Support Vector Machines*
GMMs	*Gaussian Mixture Models*

La Figura 1 y Figura 2 muestran las estructuras básicas de la identificación del hablante y los sistemas de identificación y verificación. La identificación del locutor es el proceso de determinar que registro de voz proporciona una expresión determinada. La verificación del locutor, por otro lado, es el proceso de aceptar o rechazar la identidad del locutor (Ronald A., 1996). La mayoría de las aplicaciones en las que una voz se usa como la clave para confirmar la identidad de un locutor se clasifican como de verificación del locutor.

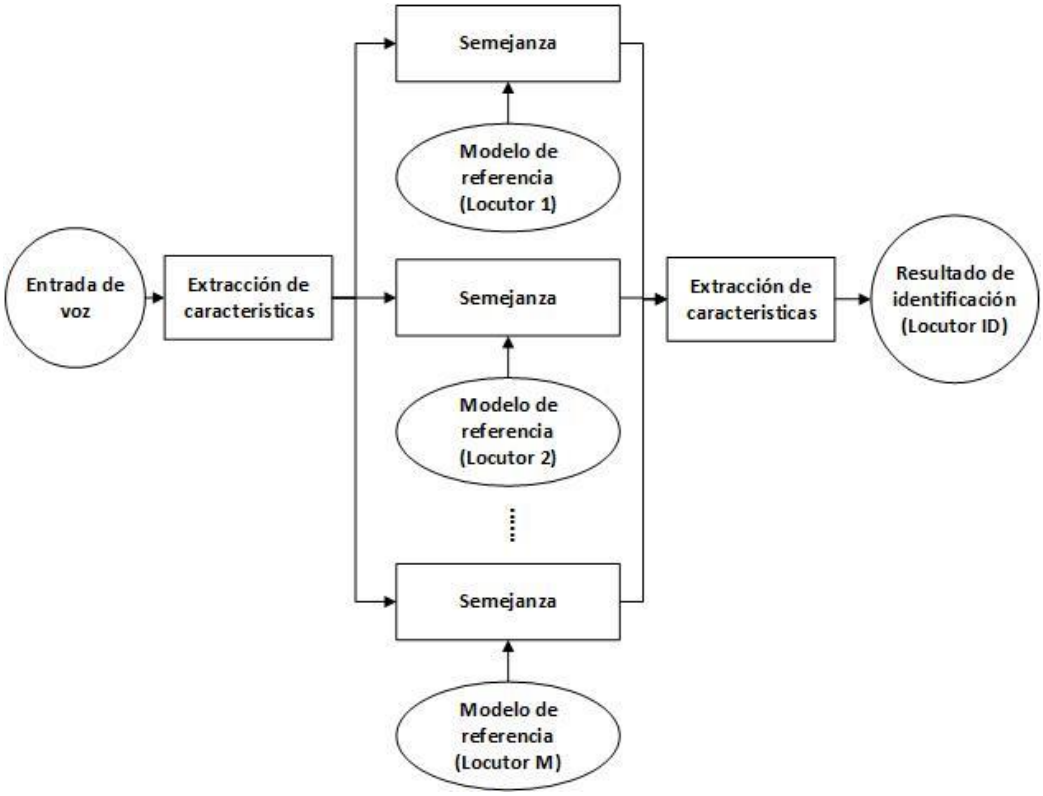


Figura 1. Esquema de un sistema de identificación de locutor.

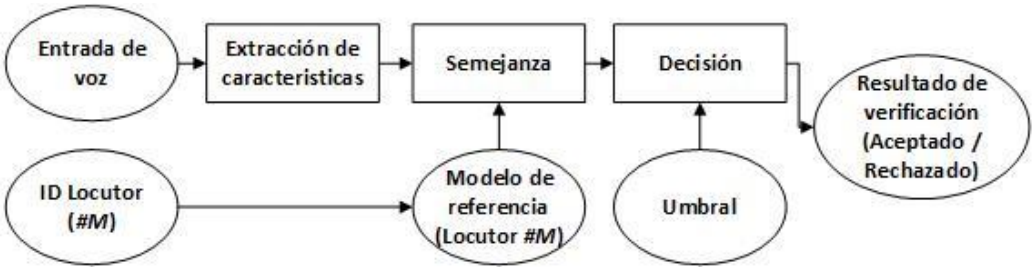


Figura 2. Esquema de un sistema de verificación de locutor.

1.2 Planteamiento del problema.

Recientemente, una gran cantidad de trabajo se ha hecho en la tecnología de voz. La concentración principal está en las técnicas de reconocimiento de voz automática y conversión de texto a voz, conversión de voz es aún un campo poco desarrollado e ingenuo en tecnología del habla (Chollet 1991).

Aunado a esto, se tienen diferentes ramas de la industria en la que se requiere tratamiento de la voz para realizar imitaciones entre locutores, como ejemplo, en la producciones de televisión y cine es muy común que se requiera realizar doblajes haciendo que el diálogo doblado tenga la mayor similitud con el dialogo original.

Otro aspecto importante a destacar es el hecho de extraer el contenido sintáctico de una grabación, actualmente se han diseñado arquitecturas que resuelven parcialmente el problema de conversión de voz, esto es, las arquitecturas estudiadas en el marco teórico muestran la manera de realizar reconocimiento de locutor, reconocimiento de palabras aisladas o síntesis de voz, pero son pocas las arquitecturas que engloban estas tres tareas para poder así realizar la conversión de voz.

Generar una arquitectura capaz de realizar la tarea de conversión de voz entre 2 locutores, implica diseñar un esquema que englobe las tres tareas antes mencionadas; si bien ya existen arquitecturas que realizan conversión de voz, aún faltan una gran cantidad de estudio para subsanar los puntos débiles que estas arquitecturas aún tienen, como son, la eficiencia de conversión, el rendimiento, el costo de procesamiento computacional o el tiempo de procesamiento.

Ayudar a solventar estas problemáticas es el punto a atender en este trabajo de investigación, poniendo mayor atención en alcanzar la mayor eficiencia en la conversión de voz, siendo principal problema a resolver en este trabajo de investigación, ya que así se dará la pauta para poder empotrar la arquitectura dentro un dispositivo embebido en trabajos futuros.

1.3 Justificación.

La conversión de voz es aún un campo poco desarrollado dentro del procesamiento digital de voz, a causa de esto nace la necesidad de resolver problemas puntuales dentro de la comunidad científica y tecnológica que se basan en estudios de este tipo.

La presente tesis nace de la observación y la necesidad de solventar los problemas antes descritos, de manera que se pueda apoyar a las entidades respectivas, y así proponer un punto de partida más, para futuras investigaciones. De igual manera la creación de las bibliotecas de programación implementadas dará soporte a distintas investigaciones e implementación de técnicas de procesamiento de voz sobre los procesadores digitales de señales.

El trabajo pretende dar solución a los planteamientos mostrados mediante el estudio de diferentes arquitecturas en el modelado del sistema con el fin de realizar la tarea de la manera más eficiente, asimismo este trabajo dará base a estudios posteriores en ramas de la informática como la conversión automática del habla y la traducción entre distintos idiomas.

1.4 Hipótesis.

La combinación de distintas técnicas de extracción de características y reconocimiento mediante técnicas de inteligencia artificial, con el fin lograr englobar las 3 tareas principales en la conversión de voz (Reconocimiento de locutor, Reconocimiento de Palabras y Síntesis de voz) sin sacrificar eficiencia y rendimiento computacional, dará lugar a encontrar una forma más eficiente de realizar la conversión voz, trayendo con ello la posibilidad de implementar una arquitectura de conversión de voz en dispositivos embebidos y con ello realizar la tarea de conversión de voz en tiempo real.

1.5 Objetivos.

1.5.1 Objetivo general.

Diseñar e implementar una arquitectura computacional que realice la conversión de voz de un corpus de dos locutores, mediante técnicas de procesamiento digital de voz e inteligencia artificial.

1.5.2 Objetivos particulares.

- Diseñar una arquitectura para realizar conversión de voz entre 2 locutores utilizando características LPCs y MFCC y técnicas de inteligencia artificial, como VQ y HMM.

- Realizar las tareas de entrenamiento y reconocimiento para un corpus de palabras, haciendo uso de CDHMM.

- Sintetizar el texto obtenido mediante un analizador sintáctico por di fonemas, el cual estará basado en el algoritmo PSOLA, y convertir la voz del locutor hablante al inferir las características del locutor diferente.

- Implementar la etapa de front-end de la arquitectura en una arquitectura embebida del DSP de Texas Instruments DSK6713 de punto flotante (Lenguaje de programación C).

- Crear el programa de entrenamiento y reconocimiento auxiliándose la herramienta HTK, así como crear el programa de síntesis por concatenación (Lenguaje de programación C).

1.6 Alcances del trabajo.

Con el fin de realizar el sistema de conversión voz, se implementan técnicas de procesamiento de voz e inteligencia artificial, para arquitecturas que den el resultado más eficiente en relación costo-tiempo (términos computacionales), para así realizar la primera implementación en el procesador digital de señales (Front-END).

Para el desarrollo de esta propuesta de investigación, se emplea el lenguaje de programación C.

1.7 Contribuciones.

- Una nueva arquitectura para la conversión de voz
- Creación de bibliotecas de procesamiento de señales en lenguajes C dentro del DSP.
- Implementación de bibliotecas base de HTK dentro del DSP

1.8 Organización del trabajo.

Esta tesis está constituida de 6 capítulos, a continuación se comenta el contenido de los mismos:

Capítulo 1. Introducción. En este capítulo se da una introducción al tema de investigación, se definen el planteamiento del problema, la justificación, la hipótesis, el objetivo general, los objetivos particulares, el alcance del proyecto, las contribuciones y la organización del trabajo para lograr el desarrollo de la tesis.

Capítulo 2. Estado del arte. Se explican algunas de las investigaciones que ya han sido realizadas para la conversión de voz y que fueron aplicadas con sus distintas arquitecturas.

Capítulo 3. Marco Teórico. En este capítulo se explica lo que es el procesamiento digital de voz, así como las diferentes de técnicas utilizadas en el reconocimiento voz

para la construcción de una metodología adecuada que pueda ser aplicada al sistema de conversión de voz.

Capítulo 4. Metodología y desarrollo de la investigación. Dentro de este capítulo se muestran los diagramas de bloques de la arquitectura propuesta, la descripción de cada uno de ellos y el diseño de la metodología.

Capítulo 5. Presentación y discusión de resultados. En este capítulo se muestran los resultados obtenidos después de haber aplicado la metodología propuesta, así como los problemas que se presentaron.

Capítulo 6 Conclusiones, Trabajos futuros y productos de la investigación. En este último capítulo se describen las conclusiones y contribuciones obtenidas tras el desarrollo de la investigación así como las mejoras que en el futuro podrían realizarse.

CAPÍTULO 2. ESTADO DEL ARTE.

Los sistemas de conversión de voz necesitan para llevarse a cabo una serie de técnicas de procesamiento de señales, los cuales toman la señal de voz de un locutor origen y tras una serie de modificaciones la convierte en la voz de un segundo locutor conocido como destino.

Las técnicas que se ocupan se han ido desarrollando en los últimos años, pero en sus inicios el término fue presentado en (S. N. M. Abe 1988), estas técnicas se agrupan en 3 grandes grupos:

- Sistemas que generan la voz del locutor destino con base en unidades lingüísticas almacenas en base de datos previamente grabadas.

- Sistemas basados en algoritmos que generan particiones de los espacios acústicos de cada uno de los locutores para luego realizar la transformación utilizando funciones estadísticas sintetizando así una aproximación de la voz del locutor destino.

- Sistemas de conversión de voz que combinan las ventajas de los dos anteriores para obtener una relación identidad-calidad acústica lo más elevada posible.

Los sistemas de conversión de voz regularmente están integrados por un sistema que transforma el texto a voz (TTS) (Lenzo 1999), (J.A. Gurlekian 2001) y (S. P. Taylor 1998).

Sin embargo, este tipo de sistemas sólo permite que la persona se exprese mediante las voces que tiene por defecto el sistema de síntesis, para ello se utiliza alguno de los 3 tipos de grupos de conversión de voz mencionados anteriormente y detallados a continuación.

2.1 Sistemas de conversión de voz basados en unidades lingüísticas.

Estos sistemas se caracterizan por sintetizar voces con una elevada identidad, ya que la misma es generada partiendo de datos reales del locutor destino, pero la desventaja de estos sistemas es que presentan problemas acústicos debidos al volumen de datos que se utiliza para sintetizar las voces, son ejemplos de este tipo de sistemas (S. N. M. Abe 1988) y (M. Abe 1991).

El rendimiento de este tipo de sistemas está relacionado directamente con las características de la base de datos utilizada. Esto es debido a que si la base de datos tiene una cantidad de información reducida, presentan problemas originados por discontinuidades acústicas debidas al tamaño de la base de datos. Por el contrario, si se amplía el tamaño de la base de datos, se mejorara la calidad del audio resultante, un ejemplo de la arquitectura de este sistema se muestra en Figura 3.

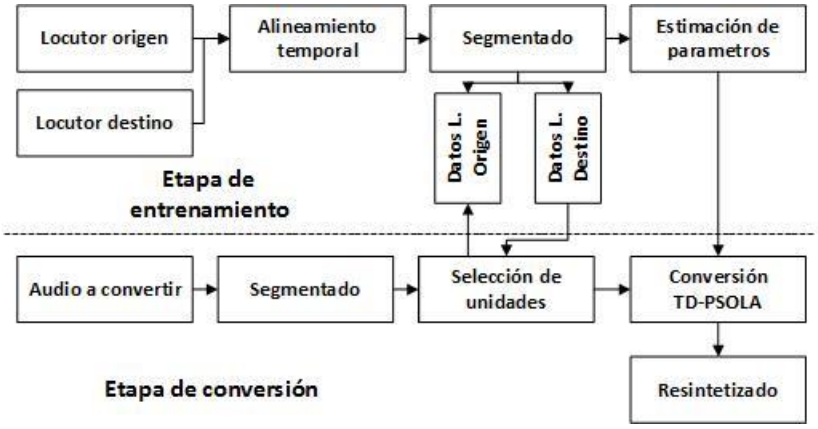


Figura 3. Esquema de conversión de voz basado en selección de unidades.

En la arquitectura anterior se muestran dos etapas principales, la de entrenamiento y la de reconocimiento; en la etapa de entrenamiento se almacenan las unidades lingüísticas de los locutores origen y destino en los bloques “Datos L. Origen” y “Datos L. Destino”. Posteriormente en la etapa de reconocimiento se realiza la selección de las unidades lingüísticas previamente almacenas para generar el re sintetizado.

2.2 Sistemas de conversión de voz basados en partición de espacios acústicos

En este tipo de sistemas, los resultados obtenidos no poseen una identidad tan elevada como en los métodos de selección de unidades, pero no presentan las discontinuidades acústicas de los primeros métodos. Son ejemplos de este tipo de sistemas los basados en GMM (Gaussian Mixture Model) (Barrobés 2006), (Y. Stylianou 1988) y en KH (KHistogramas) (A. A. Uriz 2009).

En la Figura 4, se presenta el sistema de conversión de voz basado en partición de estados.

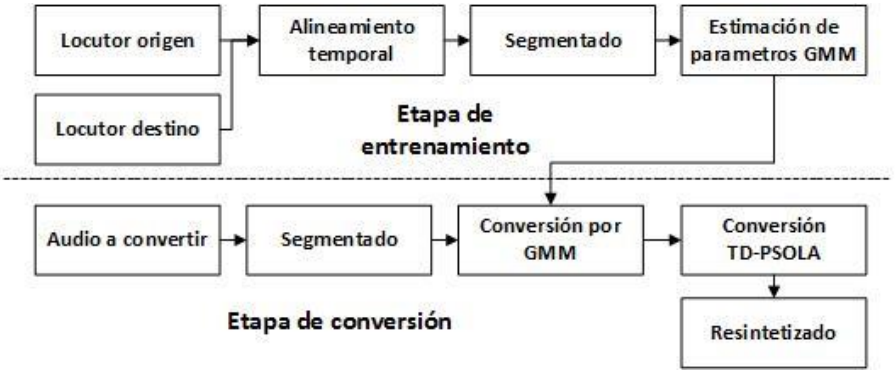


Figura 4. Esquema de conversión de voz basado en partición de espacio

Una característica relevante de este tipo de sistemas es que, dado que los segmentos de la señal de voz a convertir sólo son afectados por una normalización utilizando parámetros obtenidos en la etapa de entrenamiento, su implementación puede ser realizada en tiempo real.

Este esquema a diferencia del presentado en Figura 3 muestra un bloque de estimación de mixturas Gaussianas en la etapa de entrenamiento que son utilizados como base de conocimiento para posteriormente dentro de la etapa de conversión realizar la partición de espacio para el re sintetizado.

2.3 Sistemas de conversión de voz híbridos

Los sistemas dentro de este grupo se caracterizan por estar compuestos por dos etapas de conversión. La primera realiza una conversión por partición del espacio (utilizando un algoritmo como el GMM por ejemplo), mientras que la segunda etapa realiza la conversión mediante un sistema de selección de unidades. Este tipo de sistemas ha tenido un auge en los últimos tiempos debido a los resultados que permiten obtener. Son ejemplos de ellos (Moreno 2007), (T. Dutoit 2007) y (A. Uriz 2009).

En la Figura 5 se presenta el sistema propuesto por (T. Dutoit 2007), el cual se caracteriza por estar compuesto por dos etapas de conversión, una compuesta por un bloque de partición del espacio mediante GMM y una segunda conversión mediante selección de unidades.

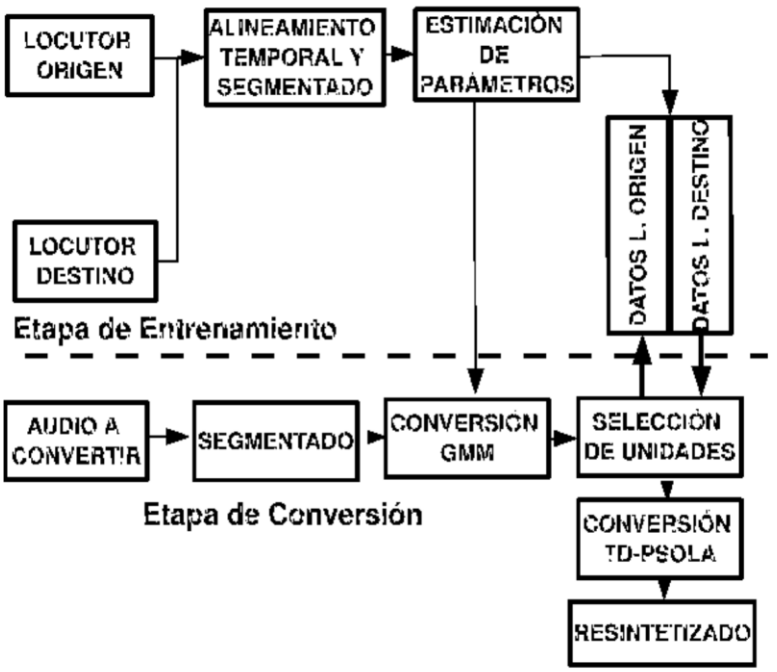


Figura 5. Sistema de conversión de voz propuesto por Dutoit (T. Dutoit 2007)

2.4 Composición de los sistemas de conversión de voz

Los sistemas de conversión de voz involucran distintas técnicas de extracción de características, reconocimiento de locutor, síntesis de voz. Se listarán algunas de las técnicas que involucran los sistemas de conversión de voz:

- Parámetros característicos de señal de voz
- Modelos para el reconocimiento de voz
- Modelos para el reconocimiento de locutor
- Sistemas de texto a voz

2.4.1 Parámetros característicos de señal de voz

Los parámetros característicos de voz que han dominado el área del reconocimiento del habla han sido:

- Coeficientes Cepstrales Reales (RCC), Oppenheim (1969).
- Coeficientes de Predicción Lineal (LPC), Atal y Hanauer (1971).
- Coeficientes Cepstrales de Predicción Lineal (LPCC), Atal (1974).
- Coeficientes Cepstrales en Frecuencia en escala de Mel (MFCC), Davis y Mermelstein (1980).
- Perceptual Linear Prediction Coefficients (PLP), Hermansky (1990).

2.4.2 Modelos para el reconocimiento de voz

- Cuantificación Vectorial, Robert M. Gray (1984) y Linde, Buzo, Gray (.1980).
- NNA (Redes Neuronales Artificiales), Lippman (1988).
- Mixturas Gaussianas, Rabiner (1989).
- HMM (Cadenas Ocultas de Markov), Rabiner (1989).
- Mixturas Gaussianas. Asociadas con HMM, Rabiner (1989).

2.4.3 Modelos para el reconocimiento de locutor

- Cuantificación Vectorial, Robert M. Gray (1984) y Linde, Buzo, Gray (.1980).
- NNA (Redes Neuronales Artificiales), Lippman (1988).
- HMM (Cadenas Ocultas de Markov), Rabiner (1989).
- Mixturas Gaussianas. Asociadas con HMM, Rabiner (1989).

2.4.4 Sistemas de texto a voz

En la Figura 6 se muestra el diagrama de bloques del sistema de conversión de texto a voz descrito en (Lenzo 1999) y fue desarrollado en el Centro de Investigación de Tecnología del Habla de la Universidad de Edimburgo, este diagrama puede ser utilizado para ejemplificar la integración que tienen este tipo de sistema en los sistemas de conversión de voz.

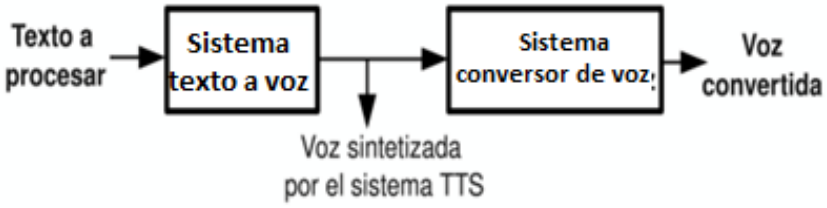


Figura 6. Sistema conversor de texto a voz

Los sistemas de conversión de voz deben de tener la capacidad de extraer el contenido de información que el mensaje del locutor origen conlleva para después generar el texto de dicho mensaje.

Una vez en este punto, solo basta con pasar el texto por un sistema TTS, el cual será alimentado con las características fonéticas del locutor destino para realizar la síntesis y generar la voz final.

2.5 Arquitecturas de Reconocimiento de Locutor

Las arquitecturas de reconocimiento de locutor que se implementan son tan variadas como la cantidad de algoritmos y estrategias existentes para la separación e identificación de las características extraídas de los locutores, sin embargo de manera general se requieren de los siguientes pasos para realizar la tarea de reconocimiento de locutor mostrado en la Figura 7:

- **Entrenamiento**: se extraen parámetros de cada usuario del sistema.
- **Funcionamiento**: a partir de las señales, el sistema tomará la decisión.
- **Actualización**: dar de alta/baja a usuarios y/o mejorar modelos.

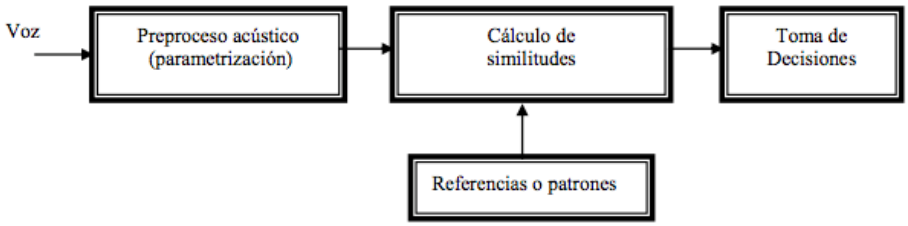


Figura 7. Diagrama General de Reconocimiento de Locutor

Los sistemas de reconocimiento de locutor según su función pueden clasificarse de la siguiente manera:

- **De identificación**: identificar a un locutor de entre varios. En conjunto cerrado: uno de N posibles locutores. Tiene que asignar uno. En conjunto abierto: uno de entre N posibles locutores. Puede que dé como salida que no es ninguno de ellos.

- **De verificación**: verificar si es o no un locutor concreto. Aceptación / Rechazo. Compara la voz de entrada con otras entradas de identidad del locutor (tarjeta magnética, código secreto, etc...)

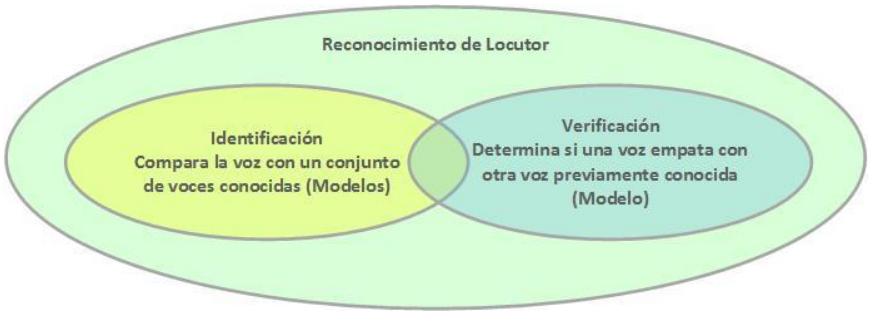


Figura 8. Tareas fundamentales del reconocimiento de locutor: identificación y verificación

En esta tesis se aborda la construcción de un sistema de reconocimiento de locutor mediante la técnica de cuantificación vectorial, para identificar a un locutor (persona) se requiere encontrar un conjunto de características que definan a la persona que está hablando; algunas de éstas son físicas, como el timbre, la entonación o los gestos que hace cuando transmite el mensaje y algunas otras son psicológicas como las palabras que escoge para hacerlo, el acento y emotividad.

2.6 Arquitecturas de sistemas de reconocimiento y conversión de voz.

Se presentan algunas de las arquitecturas que se han utilizado para el reconocimiento de voz, para empezar se presenta en Figura 9 el diagrama general del sistema reconocimiento. Partiendo del esquema mostrado en la Figura 9, se han desarrollado diversos sistemas de conversión de voz, implementando distintas arquitecturas con distintas técnicas, con el fin de obtener mejores resultados cada vez.

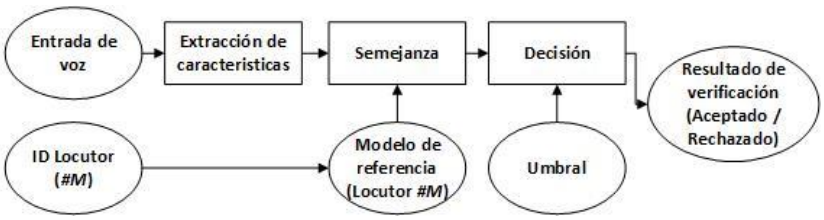


Figura 9. Esquema general del sistema de reconocimiento de voz

En la Figura 10 se muestran la arquitectura de un sistema de conversión de voz utilizado en (Eun-Kyoung Kim 1997), donde se utilizaron HMM y características dinámicas del locutor, logrando un mejor resultado que el obtenido con "natural sound".

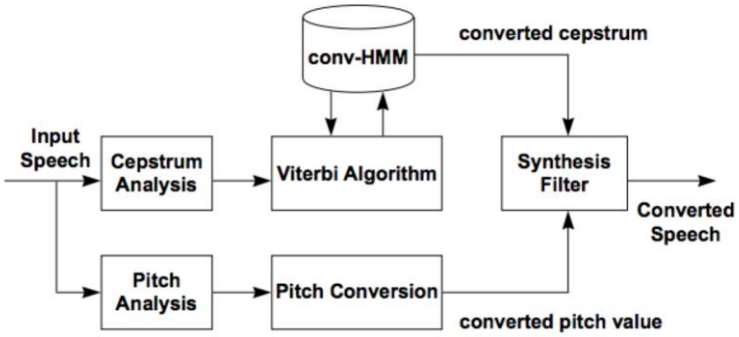


Figura 10. Diagrama a bloques del sistema propuesto en (Eun-Kyoung Kim 1997)

En Figura 11 se muestran la arquitectura de un sistema de conversión de voz utilizado en (Yannis Stylianou 1998), mostrando mejores resultados que con reconocimiento basado cuantificación vectorial pura.

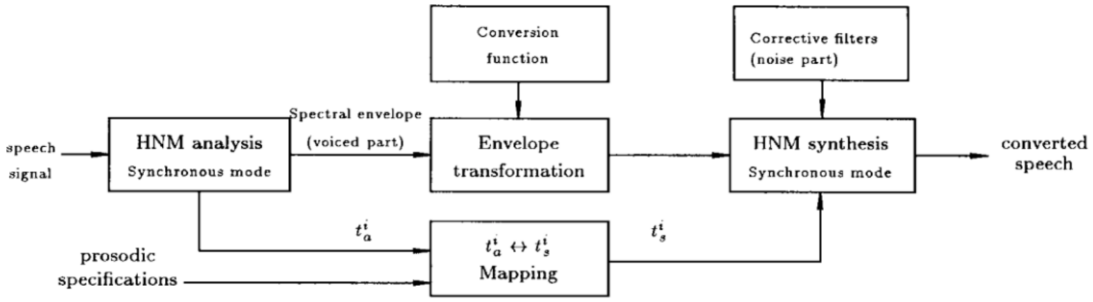


Figura 11. Sistema de Conversión de voz mostrado en (Yannis Stylianou 1998)

En la Figura 12 se muestra otra arquitectura de sistema de conversión de voz donde combinaron VQ y GMM para el reconocimiento, y en la parte de síntesis enfatizaron el trabajo al utilizar síntesis basadas en LPCs.

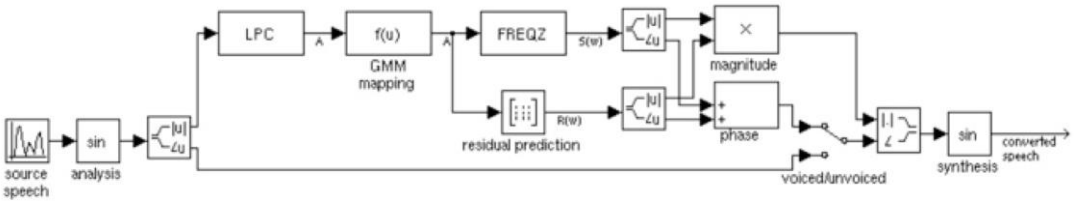


Figura 12. Algoritmo de conversión de voz basada en LPC (Alexander Kain 2001)

En la Figura 13 se muestra otra arquitectura de sistema de conversión de voz donde la parte de síntesis enfatizaron el trabajo al utilizar síntesis basadas en concatenación de fonemas.

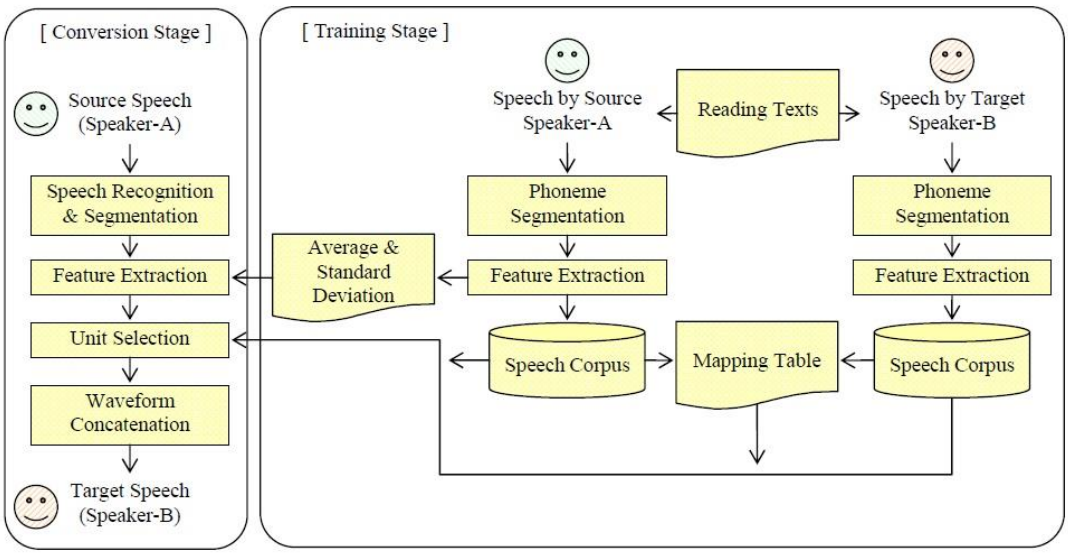


Figura 13. Conversión de Voz basada en Concatenación (Kei Fujii 2007)

Estas son algunas de las arquitecturas que se han utilizado para el reconocimiento de voz, aunque falta por listar otras, tienen la virtud de que las mismas funcionan de una manera adecuada y eficiente dentro de un equipo de cómputo convencional.

Aunado a esto, surge la problemática de ver cómo implementar un sistema de conversión de voz en un sistema embebido para realizar la tarea en tiempo real, esto debido a la relación costo-tiempo que surgen en los dispositivos computacionales embebidos.

En la Figura 14 se presenta la arquitectura diseñada e implementada para satisfacer los problemas planteados, está arquitectura se divide en tres etapas; reconocimiento de locutor, reconocimiento de palabras y síntesis. La etapa de reconocimiento de locutor está implementado mediante VQ generando un libro código por cada locutor, el libro código está integrado por 128 vectores LPC, y se determina la semejanza mediante el uso de distancia euclidiana, el uso de VQ implica menor costo computacional que el uso de HMM mostrado en las arquitecturas antes mencionadas, lo que permite la implementación de esta técnica dentro del procesador digital señales.

La etapa de reconocimiento de palabras se implementa mediante el uso de HMM, generando los modelos de Markov con vectores MFCC, la implementación de esta etapa también es generada en lenguaje de programación C apoyándose de las bibliotecas de HTK.

La etapa final es la de síntesis, está etapa se alimenta de los resultados obtenidos en las dos etapas anteriores, y se realiza mediante la técnica de síntesis por concatenación de unidades lingüísticas con TD-PSOLA; en esta etapa se hace uso de di fonemas para realizar el re sintetizado, y la elección de la base de datos de di fonemas la determina el resultado generado en el reconocimiento de locutor, mientras que el resultado del reconocimiento de palabras determina que unidades lingüísticas se toman de la base de datos de di fonemas; después se concatenan los di fonemas y se eliminan las ambigüedades mediante TD-PSOLA para realizar el re sintetizado.

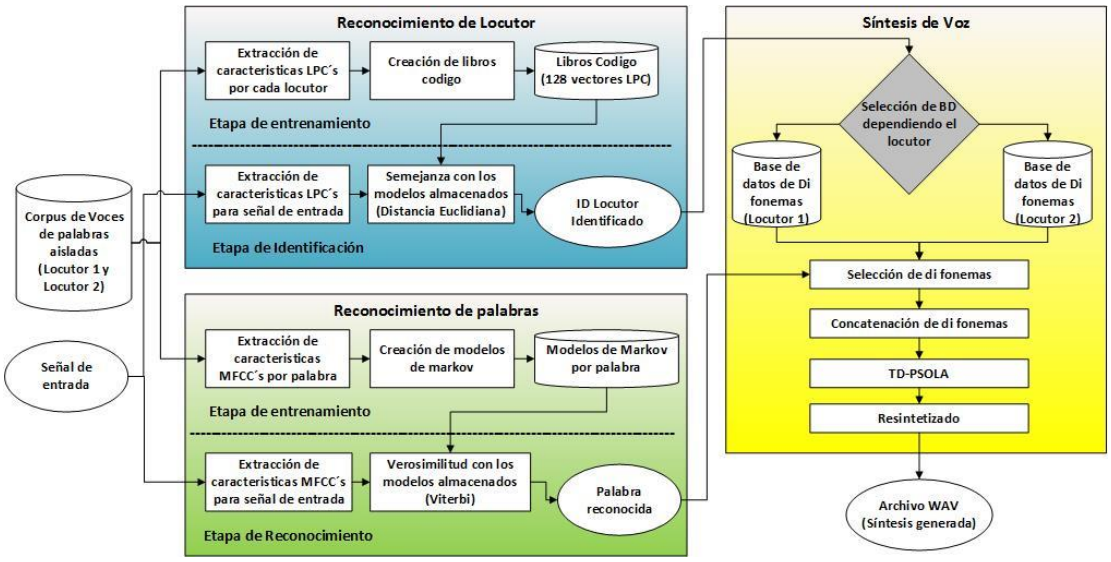


Figura 14. Arquitectura general de conversión de locutor implementada

Este capítulo sirve para dar un pequeño bosquejo de la arquitectura que se diseñó, así como las diferencias con las otras arquitecturas existentes; se realizara el análisis a profundidad de los bloques que componen esta arquitectura, así como los resultados obtenidos de la implementación de la misma en capítulos posteriores.

CAPÍTULO 3. MARCO TEÓRICO

El punto principal para realizar cualquier investigación, tiene la necesidad de poder cuantificar el objeto de estudio y seguir una metodología apropiada para llevar a término la investigación.

Para el caso del procesamiento de voz, es necesario extraer características de la señal de voz para así poder aplicar diversos estudios de reconocimiento, clasificación y síntesis. En Figura 15 se muestra un esquema general que ilustra los pasos a seguir realizar la extracción de características MFCC; esta etapa es conocida como front-end.

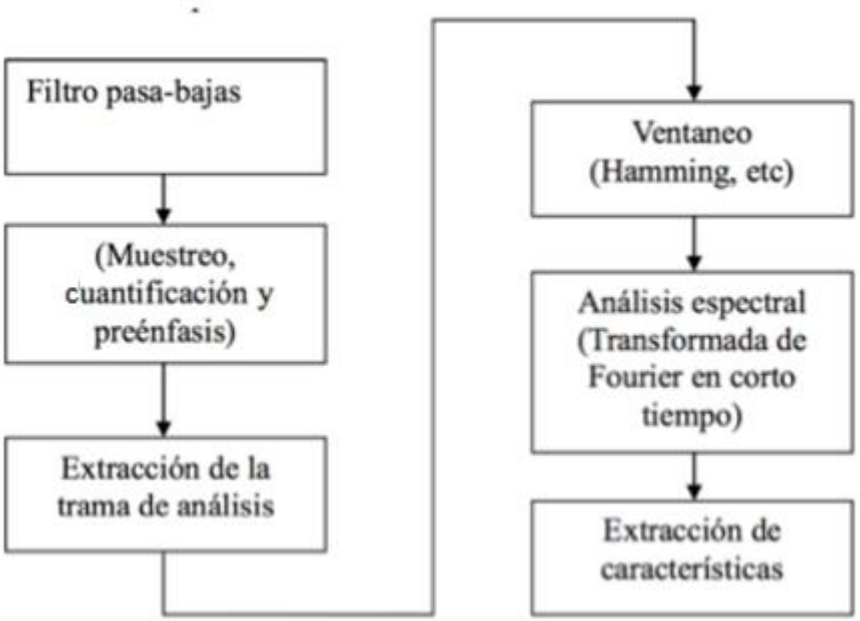


Figura 15. Esquema de extracción de características de la señal de voz

Una vez obtenidas las características de las señales de voz se tiene que seleccionar la técnica para realizar el reconocimiento del locutor y de las palabras aisladas, algunas de ellas se listan a continuación y se profundizará más sobre éstas a lo largo de este capítulo.

- **Dynamic Time Warping**
- **Cuantificación Vectorial**

- **Modelos Ocultos de Markov**

Después de haber reconocido el locutor y la palabra se procede a elegir la técnica de síntesis para imitación fonética entre los locutores. La síntesis de voz puede ser producida por distintos métodos. Todos ellos tienen algunas ventajas y deficiencias. Los métodos se suelen clasificar en tres grupos:

- **Síntesis articulatoria**, que intenta modelar directamente el sistema de producción del habla humana.
- **Síntesis de formantes**, donde se modelan las frecuencias de los polos de la señal de voz o función de transferencia del tracto vocal basado en fuente-filtro-modelo.
- **Síntesis por concatenación,** que utiliza muestras pregrabadas del habla natural.

Por lo tanto se requiere de un conjunto de tareas para llevar a cabo el reconocimiento de voz, las cuales se ejecutarán de preferencia en secuencia e independientemente del objetivo a alcanzar (Guerra, 2004).

Estas tareas se enlistan a continuación:

1. Preparar y crear un Corpus de Voces a utilizar en el trabajo.
2. Definir, preparar y realizar el pre-procesamiento que será aplicado a los archivos del Corpus.
3. Seleccionar, diseñar y programar los algoritmos y métodos que se utilizarán para la extracción de parámetros a utilizar en el reconocimiento de voz.
4. Entrenar el sistema.
5. Sintetizar la palabra reconocida para realizar la imitación.
6. Verificar el grado (%) de aceptación del sistema (reconocimiento).

A lo largo de este capítulo se explican las bases teóricas de la técnicas listadas para los diferentes métodos de procesamiento digital de señales de voz, así como la arquitectura del procesador digital de señales usado y el proyecto HTK, con el fin de generar una arquitectura que se apoye en las combinaciones de estos métodos.

3.1 Pre-procesamiento de señal de voz

Una señal de voz es representada por secuencias de palabras que se describen de la forma $W=(w_1, w_2, w_3, \ldots\ldots, w_t)$, en donde w_t es la palabra genérica pronunciada en el tiempo discreto "t". Las secuencias de palabras son conectadas a través de la voz que es una secuencia de sonidos acústicos X.

En términos matemáticos, el reconocimiento es una función "f" que mapea un X conjunto de datos que pertenece al conjunto completo de secuencias acústicas X a un conjunto W que está incluido en el conjunto ϖ (D. J. Broad 1995)

$$f: X \rightarrow W,\ X \in \chi\ y\ W \in \varpi \qquad (1)$$

Acondicionar una señal para posteriormente procesarlas en necesario para facilitar el trabajo y mejorar el nivel de reconocimiento en el sistema

Su importancia radica en seguir una serie de pasos, para normalizar las características en el dominio del tiempo de las señales de voz grabadas o captadas directamente desde un micrófono.

3.1.1 Preénfasis

El preénfasis es realizado con el propósito de suavizar el espectro y reducir las inestabilidades del cálculo asociadas con las operaciones aritméticas de precisión finita, también es utilizado para restaurar la pérdida que sufre la señal de voz en sus componentes de alta frecuencia (Rodríguez, 2013). El preénfasis consiste el pasar toda la señal por un filtro pasa alta de primer orden y con un solo coeficiente.

La ecuación del filtro de preénfasis se muestra en la Figura 16, donde se muestra que el preénfasis es un filtro pasa altas donde acentuará las frecuencias arriba de 2 KHz.

$$y[n] = x[n] - ax[n-1] \quad donde \quad a = 0.95$$

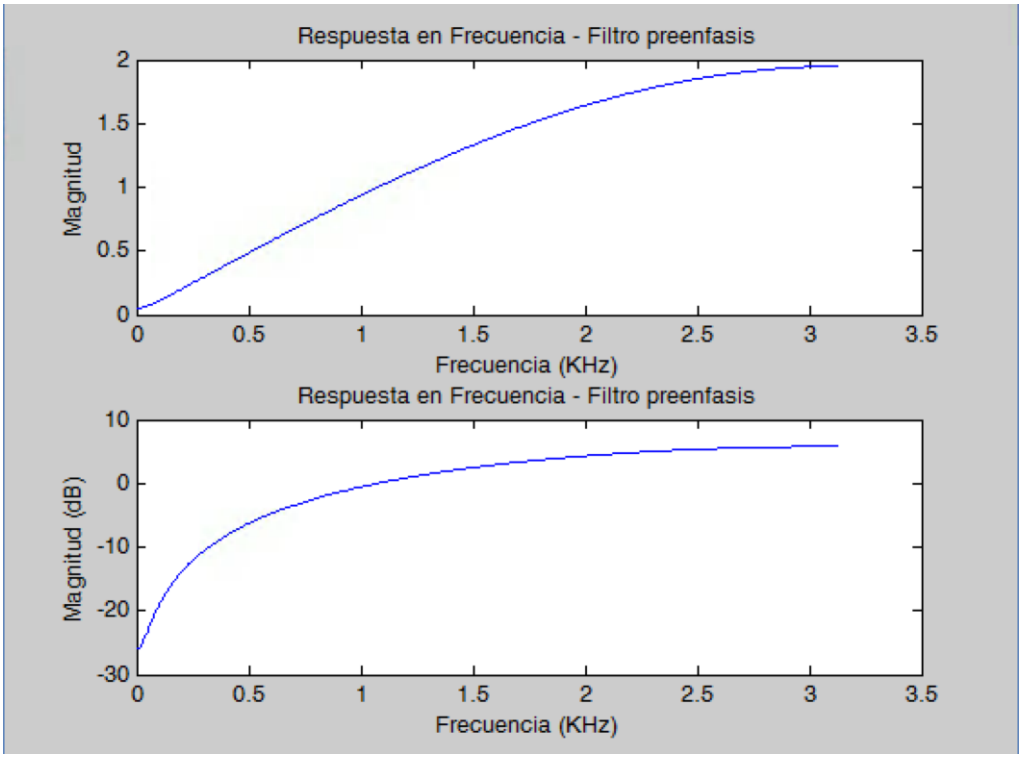


Figura 16. Filtro de Preénfasis

La respuesta en frecuencia se puede obtener mediante el análisis realizado a la ecuación del filtro de preénfasis. Obteniendo la transformada z de la expresión anterior tenemos.

$$Y(z) = X(z) - az^{-1}X(z) = X(z)(1 - z^{-1}) \qquad (2)$$

De (2) se obtiene la respuesta al impulso

$$H(z) = \frac{Y(z)}{X(z)} = 1 - az^{-1} \qquad (3)$$

Con lo anterior se puede obtener la respuesta en frecuencia

$$\boldsymbol{Hacemos} \quad \boldsymbol{z = e^{j\omega}\ tenemos\ H(e^{j\omega}) = 1 - ae^{-j\omega}} \tag{4}$$

Así en la Figura 17 se muestra la implementación por bloques de la ecuación de diferencias obtenida:

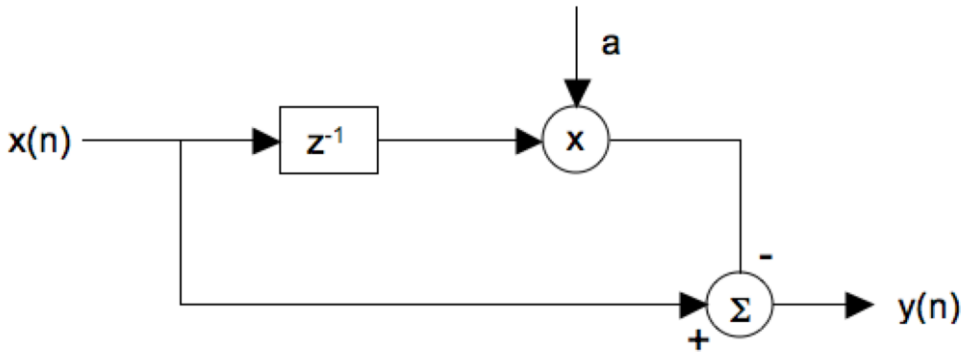


Figura 17. Diagrama a bloques del sistema para un filtro de preénfasis

3.1.2 Ventaneo

El mecanismo de ventaneo muestra como a cada porción de la señal de voz (de un tamaño predefinido por el usuario), se le asigna una ventana, de tal forma que las muestras queden ponderadas con los valores de la función escogida. En este caso, las muestras que se encuentran en los extremos de la ventana tienen un peso mucho menor que las que se hallan en el medio, lo cual es muy adecuado para evitar que el truncamiento característico de los extremos del segmento del bloque varíen la interpolación de lo que ocurre en la parte central, la cual es la más significativa, de las muestras del segmento seleccionado.

La colocación de las ventanas puede realizarse de tal forma que existan solapamientos y, aunque ello repercutirá en los tiempos de respuesta del sistema reconocedor, proporcionará una mejor calidad en los resultados obtenidos.

Las funciones de ventaneo más comunes se muestran en la Figura 18:

Ventana Rectangular	$w(t)=1$ en el intervalo temporal del bloque $(0 \le t \le L-1)$ $w(t)=0$ en el resto de la señal
Ventana Hanning	$w(t)=0.5-0.5*\cos\left(\frac{2\pi t}{L}\right)$ en $(0 \le t \le L-1)$ $w(t)=0$ en el resto de la señal
Ventana Hamming	$w(t)=0.54-0.46*\cos\left(\frac{2\pi t}{L}\right)$ en $(0 \le t \le L-1)$ $w(t)=0$ en el resto de la señal

Figura 18. Tipos de ventanas

La ventana más utilizada, para las tareas de reconocimiento y análisis de señales de voz, es la de Hamming, dado que es la que mejor respuesta en el dominio de la frecuencia aporta para eliminar los ruidos que introduce el truncamiento de la señal en segmentos (José Luis Oropeza Rodríguez S. S., 2006). Para cualquier ventana, su duración determina la cantidad de cambios que se podrán obtener; con una duración temporal larga se omiten los cambios locales producidos en la señal, mientras que con una duración demasiado corta se reflejan demasiado los cambios puntuales y se reduce la resolución espectral.

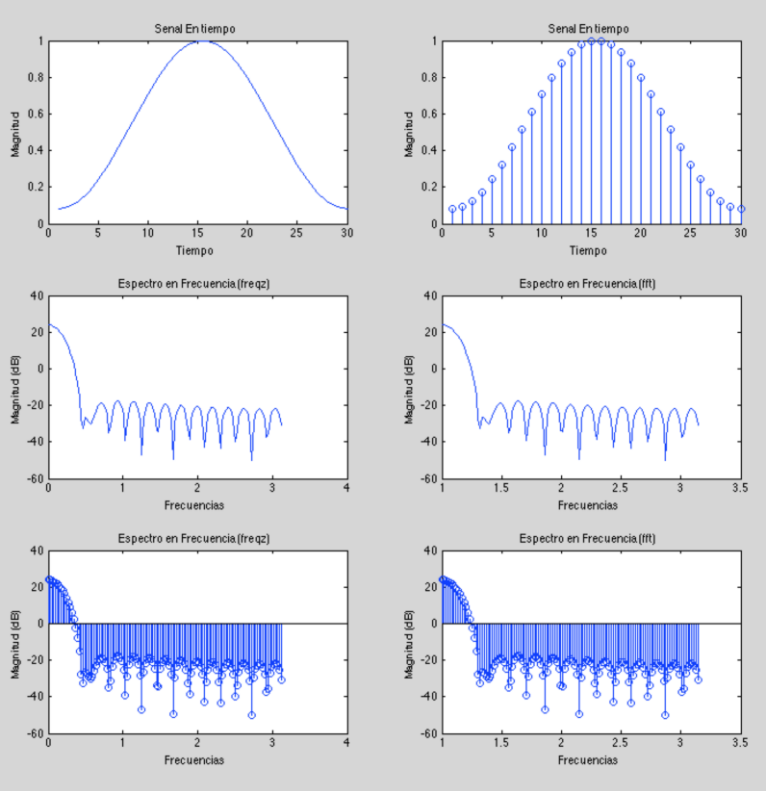


Figura 19. Respuesta en Frecuencia de Ventana de Hamming

3.1.3 Transformada rápida de Fourier (FFT)

La transformada de Fourier es una de las herramientas principales de análisis con que se cuenta hoy en ciencia y tecnología. Su poder radica en la posibilidad establecer relaciones entre puntos de vista muy diferentes relativos a un mismo problema. Así, al visualizar un fenómeno tanto en términos de la función asociada, como de su transformada de Fourier, se tiene frecuentemente un procedimiento de análisis útil para resolver un problema determinado (Witomski, 1999) y (Lanczos, 1966).

Sin entrar en detalles, se plantean aquí las relaciones o fórmulas que permiten ir de un "punto de vista" al "otro". Una se conoce como la integral de Fourier:

$$H(f) = \int_{-\infty}^{\infty} h(t) e^{-j2\pi ft} dt \qquad (5)$$

En una forma similar se define la transformada inversa de Fourier:

$$h(t) = \int_{-\infty}^{\infty} H(f) e^{j2\pi ft} df \qquad (6)$$

Donde las funciones h(t) y H(f) están relacionadas por las ecuaciones (5) y (6), llamándose las dos funciones un par transformado de Fourier, indicando esa relación por la notación

$$h(t) \Leftrightarrow H(f) \qquad (7)$$

Se puede tener una representación alternativa de las dos ecuaciones, haciendo w = 2pf . El par transformado queda entonces definido como:

$$H(\omega) = a_1 \int_{-\infty}^{\infty} h(t) e^{-j\omega t} dt \qquad (8)$$

$$h(t) = a_2 \int_{-\infty}^{\infty} H(\omega) e^{j\omega t} d\omega \qquad (9)$$

Existen aplicaciones que requieren del cálculo de la transformada de Fourier y cuentan para ello con procesadores digitales con el presente trabajo. En vista de su carácter discreto es necesario pensar en un procedimiento que sea apropiado para el cálculo artificial. Haciendo una serie de consideraciones fundamentadas en la transformada continua de Fourier y en sus propiedades, se llega a un algoritmo al que se denomina DFT (también se llega a la FFT) que permite establecer una versión discreta de la transformada continua.

Normalmente los algoritmos de transformadas rápidas de Fourier se introducen como un proceso eficiente de cálculo en el que la suma de la DFT se descompone en dos sumas, cada una con la mitad del número total de muestras. En este punto se plantea el proceso de decimación en el tiempo, consistente en expresar la suma de DFT como dos sumas separadas en las que respectivamente se procesan las muestras pares y las impares.

Los algoritmos de FFT por decimación en tiempo se basan en calcular la DFT dividiendo la sucesión temporal x[n] en sucesiones de menor longitud. Una forma dual se obtiene si se divide la sucesión de las frecuencias X[k] de la DFT en sucesiones más pequeñas; estos algoritmos se denominan de decimación en frecuencia (ANEXO C. Algoritmo de FFT Mediante Decimación en Frecuencia).

3.1.4 Transformada discreta del coseno (DCT)

La Transformada Discreta del Coseno Directa (DCT) es un tipo de transformada real y ortogonal, encargada de compactar la energía para datos altamente correlacionados. Sea una matriz cuadrada ortogonal C, entonces la inversa de dicha matriz será completamente igual a la transpuesta de la misma. O bien, si multiplicamos la matriz ortogonal C con su respectiva transpuesta, el resultado es la matriz identidad.

$$C^{-1} = C^{T} \quad \rightarrow \quad I = C^{-1} * C^{T} \tag{10}$$

De esta manera, es posible transformar un vector columna X de dimensión N, en otro vector columna Y de la misma dimensión, utilizando una matriz ortogonal C de dimensión NxN[18].

$$Y_{Nx1} = C_{NxN} * X_{Nx1} \tag{11}$$

Donde el vector Y será denominado como la transformación unidimensional. De igual manera, existe la respectiva transformación inversa si despejamos X de la ecuación anterior.

$$X_{Nx1} = C^{T}_{NxN} * Y_{Nx1} \tag{12}$$

La DCT es como un proceso de descomposición de un conjunto de muestras discretas, en un nuevo conjunto ponderado de funciones base cosinusoidales. Y por el contrario, al método de reconstruir el conjunto de muestras a partir del nuevo conjunto ponderado de funciones base cosinusoidales, se le conoce como Transformada Discreta del Coseno Inversa (IDCT), la ecuación que representa la transformada discreta de coseno:

$$\mathrm{F(k)} = \mathrm{C(k)} \sum_{n=0}^{N-1} f[n] + \cos\left(\frac{\pi(0.5+n)k}{2N}\right) \quad para\ k = 0.1.2 \ldots\ldots, N-1 \tag{13}$$

Mientras que la ecuación para la transformada inversa del coseno es:

$$\mathrm{f[n]} = \sum_{n=0}^{N-1} C(k) * F(k) + \cos\left(\frac{\pi(0.5+n)k}{2N}\right) \quad para\ k = 0.1.2 \ldots\ldots, N-1 \tag{14}$$

Donde:

f[n] = Representa la secuencia e muestras discreta

F(k) = Representa los coeficientes de la DCT de f[n]

N = Numero de coeficientes de la DCT

$$\mathrm{C(k)} = \begin{cases} \sqrt{\frac{1}{N}}, para\ k = 0 \\ \sqrt{\frac{2}{N}}, para\ 1 \leq k \leq N-1 \end{cases}$$

3.2. Coeficientes de predicción lineal (LPC)

La funcionalidad matemática que da como resultado los coeficientes del filtro, es una relación matricial que se obtiene al realizar la predicción lineal de las muestras de las señales, con lo cual se obtienen los coeficientes de predicción lineal (LPC), también llamados parámetros LPC, que no son otra cosa que los coeficientes del filtro "todo polos" buscado.

El cálculo de parámetros LPC es un procedimiento que se realiza en el dominio del tiempo, y está basado en la consideración de que el término enésimo s(n) de una secuencia, puede ser estimado a partir de los términos anteriores, considerando la sumatoria de los mismos con un peso asociado a cada uno de ellos. ŝ(n) es el término estimado de la secuencia. Este modelo es llamado de predicción lineal (LPC) o auto recursivo.

$$\hat{\mathrm{s}}(\mathrm{n}) = \alpha_1 \mathrm{s}(\mathrm{n}-1) + \alpha_2 \mathrm{s}(\mathrm{n}-2) + \cdots + \alpha_\mathrm{p} \mathrm{s}(\mathrm{n}-\mathrm{p}) \tag{15}$$

Donde $\hat{s}(n)$ es el término estimado de la secuencia. Entonces consideramos la combinación lineal de las muestras anteriores definida como:

$$\hat{s}(n) = \sum_{k=1}^{p} a_k s(n-k) \tag{16}$$

El cálculo del error de predicción (señal real $s(n)$ menos señal estimada $\hat{s}(n)$) se define como:

$$e(n) = s(n) - \hat{s}(n) = s(n) - \sum_{k=1}^{p} a_k s(n-k) \tag{17}$$

Para obtener las ecuaciones que deben ser resueltas para determinar los coeficientes de predicción LPC, definimos los segmentos de voz y de error en un segmento de datos, y los tiempos en función de n, buscamos minimizar la señal de error mínimo cuadrado en el intervalo, ya que el error es mayor según sea menor k y disminuye al aumentar el número de datos considerados para la predicción. A partir del error cuadrático medio M tenemos:

$$M = \sum_{n} e^2(n) = \sum_{n} \left[s(n) - \sum_{k=1}^{p} a_k s(n-k) \right]^2 \tag{18}$$

El método de auto correlación sugiere una manera simple y eficiente de definir los límites de las sumatorias, al asumir que los segmentos de voz $s(n)$ son nulos fuera del intervalo $0 \leq n \leq N-1$. La función de autocorrelación permite medir matemáticamente el parecido existente entre una señal y una versión retrasada en el tiempo de la misma señal. El pico mayor de la función de auto correlación está localizado en la posición cero de desplazamiento. La distancia de la localización del

siguiente pico, máximo global mayor a un 60% del máximo inicial en cero, de la función tras el primer cruce por cero, da una estimación del período básico, y la indicación de la periodicidad de la señal. La función de auto correlación está dada por la ecuación.

$$\boldsymbol{\varphi(k) = R(k) = \sum_{n=0}^{N-1} s(n) \cdot s(n+k)} \tag{19}$$

Si ponemos los términos de la sumatoria de productos en función del auto correlación, tenemos:

$$\boldsymbol{\varphi(j,k) = \sum_{n} s(n-j) \cdot s(n-k)} \tag{20}$$

$$\boldsymbol{\varphi(j,k) = R_n(|j-k|),\ j = 1,2,\cdots,p\ \ y\ \ k = 0,1,\cdots,p} \tag{21}$$

Como la función de auto correlación es simétrica $\boldsymbol{R_n(j) = R_n(-j)}$ las ecuaciones que deben cumplir los parámetros LPC se pueden expresar como:

$$\begin{bmatrix} R_n(0) & R_n(1) & R_n(2) & \cdots & R_n(p-1) \\ R_n(1) & R_n(0) & R_n(1) & \cdots & R_n(p-2) \\ R_n(2) & R_n(1) & R_n(0) & \cdots & R_n(p-3) \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ R_n(p-1) & R_n(p-2) & R_n(p-30) & \cdots & R_n(0) \end{bmatrix} \cdot \begin{bmatrix} a_1 \\ a_2 \\ a_3 \\ \vdots \\ a_p \end{bmatrix} = \begin{bmatrix} R_n(1) \\ R_n(2) \\ R_n(3) \\ \vdots \\ R_n(\mathrm{p}) \end{bmatrix} \tag{22}$$

Esta matriz de $\boldsymbol{p * p}$ valores de autocorrelación es una matriz tipo Toeplitz, donde $\boldsymbol{p}$ es el número de coeficientes a_p, parámetros LPC que se quieren calcular, la cual se puede resolver mediante el método de Levinson-Durbin, obteniendo los parámetros

LPC, los cuales permiten aproximar la señal original minimizando el error cuadrático medio.

3.2.1 Algoritmo de Levinson-Durbin

Algoritmo utilizado en algebra lineal para calcular los coeficientes de una matriz de Toeplitz de forma recursiva, creado por Levinson en 1947 y mejorado por J. Durbin en 1960.

El algoritmo presenta una mejora en la resolución de una ecuación que involucra una matriz de Toeplitz al reducir la complejidad respecto al método Gauss-Jordan, ya que el algoritmo Levinson-Durbin tiene una complejidad o(n^2) mientras que Gauss-Jordan tiene una complejidad o(n^3). En la Figura 20 se muestra el pseudocódigo para resolver la matriz de Toeplitz mediante la aplicación de Levinson-Durbin en donde:

k = Son los coeficientes de reflexión

a = Son los coeficientes de predicción

E = Es el error cuadrático medio

Inicializa el predictor de orden 0: $E_0 = R[0] \qquad a_0^0 = 0$

For $i = 1 \dots p$

$$k_i = \left(R[i] - \sum_{j=1}^{i-1} a^{i-1} R[i-j] \right) / E_{i-1}$$

$$a_j^i = k_i$$

$$a_j^i = a_j^{i-1} - k_i a_{i-j}^{i-1}, \qquad for \;\; 1 \le j \le i-1$$

$$E_i = (1 - k_i^2) E_{i-1}$$

End

Figura 20. Algoritmo Levinson-Durbin (Luis, 2010)

3.3 Coeficientes cepstrales o cepstrum

Se define como la transformada inversa de la amplitud del espectro logarítmica en corto tiempo X(ω) (Noll, 1964). Inherentemente el término cepstrum evoca la transformada inversa del espectro.

La señal de voz x(t) puede ser expresada como la respuesta de la articulación del tracto vocal equivalente al filtro controlado por una fuente pseudoperiodica g(t). Entonces x(t) puede ser controlada por la convolución de g(t) y la respuesta al impulso del tracto vocal h(t) de la forma:

$$\boldsymbol{x(t) = \int_0^t g(\tau)h(t-\tau)d\tau \rightarrow X(\omega) = G(\omega)H(\omega)} \quad (23)$$

Si g(t) es una función periódica, X(ω) es representada por la línea espectral, los intervalos de frecuencia de los cuales son recíprocos del período fundamental de g(t). Por lo que, cuando X (ω) es calculada por la transformada de Fourier de una secuencia en el tiempo muestreada por un período corto de tiempo de la señal de voz, esto exhibe picos con intervalos equivalentes a lo largo del eje x. Su logaritmo X (ω) es de la forma:

$$\boldsymbol{log|X(\omega)| = log|G(\omega)| + log|H(\omega)|} \quad (24)$$

El cepstrum, que es la inversa de la transformada de Fourier es de la forma:

$$\boldsymbol{c(\tau) = F^{-1}log|X(\omega)| = F^{-1}log|G(\omega)| + F^{-1}log|H(\omega)|} \quad (25)$$

El primer y segundo término de la expresión anterior corresponde a la estructura fina espectral y la envolvente espectral, respectivamente. Cuando el valor del cepstrum es determinado por la DFT, es necesario establecer el valor de la transformada, N, suficientemente grande para eliminar el aliasing similar al producido durante el muestreo de la forma de onda el cepstrum se calcula de la forma:

$$\boldsymbol{c_n = \sum_{k=0}^{N-1} log|X(k)|e^{i2\pi kn/N}} \quad (26)$$

3.4 Coeficientes cepstrales en escala de Mel (MFCC)

Un método más eficiente para extraer características y que es el más utilizado actualmente en reconocedores comerciales, son los coeficientes cepstrales en Escala de Mel (MFCC), este método es robusto, además hace uso de la transformada de Fourier para obtener las frecuencias de la señal. El objetivo es desarrollar un conjunto de características basadas en criterios perceptuales, diversos experimentos muestran que la percepción de los tonos en los humanos no está dada en una escala lineal, esto hace que se trate de aproximar el comportamiento del sistema auditivo.

Los coeficientes cepstrales en frecuencia en escala de Mel (MFCC), son una representación definida como el cepstrum de una señal ventaneada en el tiempo, que ha sido derivada de la aplicación de una transformada rápida de Fourier, pero en una escala de frecuencias no lineal, las cuales se aproximan al comportamiento del sistema auditivo humano.

Para obtener con vectores MFCC se tiene que seguir una serie de pasos bien definidos, a continuación se describen los pasos a realizar para obtener los vectores MFCC, dada una transformada discreta de Fourier de una señal de entrada:

$$X[k] = \sum_{n=0}^{N-1} x[n] e^{\frac{-j2\pi nk}{N}} \qquad k = 0,1,\ldots..,N-1 \tag{27}$$

Se distribuye el comportamiento espectral en bandas de frecuencia, mediante un banco de filtros con los que se calcula el promedio del espectro alrededor de cada frecuencia central. Se puede definir a f_l como la frecuencia más baja y a f_h como la frecuencia más alta del banco de filtros en ***Hz***, F_s es la frecuencia de muestreo en ***Hz***, ***M*** el número de filtros y ***N*** el tamaño de la transformada rápida de Fourier.

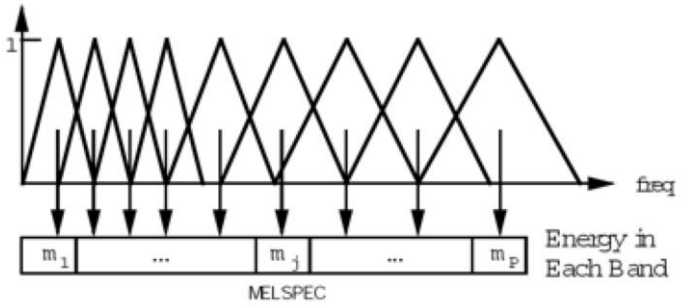


Figura 21. Banco de filtros para MFCC

Normalmente los filtros triangulares son repartidos en el rango de frecuencias completas, desde cero hasta la frecuencia de Nyquist. Sin embargo, un criterio de limitación en banda es comúnmente útil para rechazar frecuencias no deseadas o evitar la construcción de filtros en regiones de frecuencia en las cuales no existe energía de la señal útil. Para el análisis de filtros solamente, las frecuencias de corte más baja y la más alta se pueden establecer utilizando parámetros de configuración LOFREQ y HIFREQ, por ejemplo:

LOFREQ=300

HIFREQ=3400

Pueden ser utilizadas para procesar señales telefónicas. Cuando se especifican las frecuencias de corte baja y alta de esta forma, el número especificado de canales de los bancos de filtro son distribuidos de forma igual a lo largo de toda la escala de Mel resultando en un conjunto de filtros pasa banda, en donde la frecuencia de corte más baja del primer filtro es igual a LFREQ y la frecuencia de corte del último filtro está en HIFREQ.

Las frecuencias centrales de los filtros se calculan con la siguiente formula.

$$f_n = \beta^{-1}[b] = \beta^{-1}\left(\beta(f_l) + n\frac{\beta(f_h)-\beta(f_l)}{M+1}\right) \text{ para } 0 < n < M \quad (28)$$

Donde:

f_l = Frecuencia más baja expresada en Hz

f_H = Frecuencia más alta expresada en Hz

N = Numero de coeficiente a calcular

M = Total de coeficientes Mel Deseados

Y donde Beta equivale a:

$$M(f) = 1125 \ln\left(1 + \frac{f}{700}\right) \tag{29}$$

Y Beta Inversa:

$$\beta^{-1}[b] = 700\left(\exp\left(\frac{b}{1125}\right) - 1\right), \ldots\ldots, 1 \leq b \leq M \tag{30}$$

Una vez que la envolvente del espectro de la señal de voz es multiplicada por el banco, se calcula la energía correspondiente en cada uno de los filtros, después de obtener la energía, debemos de calcularle el logaritmo, pasando por lo tanto al dominio de la potencia espectral logarítmica. El inconveniente de trabajar en este dominio es que los espectros de los filtros en las bandas adyacentes presentan un alto grado de correlación, originando coeficientes espectrales estadísticamente muy dependientes entre ellos.

Para eliminar dicha dependencia o correlación estadística, se hace uso de la Transformada Discreta del Coseno (DCT), que lleva los coeficientes espectrales al dominio de la quefrencia convirtiéndolos en coeficientes cepstrales (MFCC), la transformada DCT es utilizada debido a que solamente estamos trabajando con la parte real de la DFT, por lo que si utilizáramos la IDFT, lo único que se ganaría es desperdiciar recursos computacionales.

Se ha demostrado que las características MFCC, a diferencia de los LPCs tienen mayor efectividad y beneficios para el Reconocimiento Habla, en la Figura 22 se muestra de forma gráfica un resumen de los pasos descritos para calcular los coeficientes ceptrales MEL.

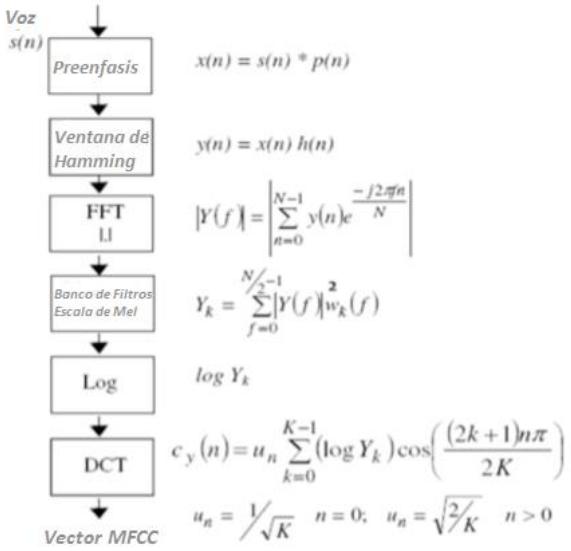


Figura 22. Algoritmo para cálculo de MFCC

3.5 Sistemas de reconocimiento basados en cuantificación vectorial (VQ)

Los sistemas de reconocimiento basados en cuantificación vectorial (Vector Quantization, VQ) hacen uso de las ventajas teóricas formuladas en la Teoría de la Información de Shannon, que establece que la cuantificación escalar es sub-óptima frente a la vectorial; por consiguiente, se trata de cuantificar una señal de entrada mediante la asignación de un vector representativo de la misma.

Así, para una trama de señal dada, que puede tomar infinitos valores de entrada (pensemos, p. e., en un VQ de forma de onda) le asignaremos un vector (trama) representativo de entre un conjunto finito posible.

Al vector representativo lo denominaremos ‘codeword’ (o centroide), y al conjunto finito completo lo denominaremos ‘codebook’ (libro de códigos o muestrario).

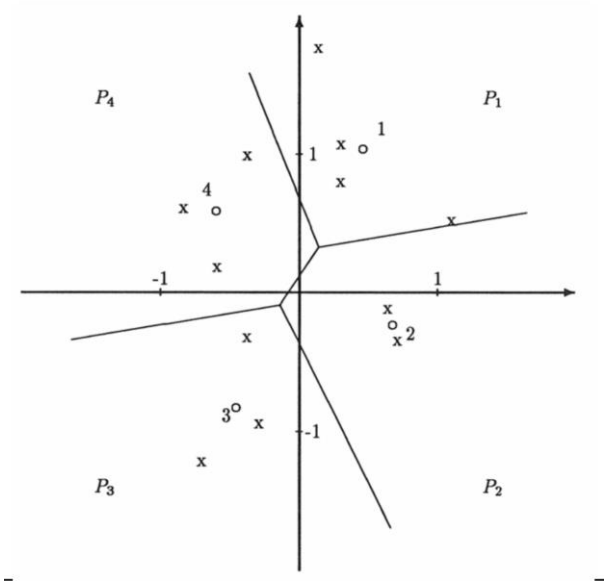


Figura 23. Ejemplo de cuantificación vectorial

Como se observa en la Figura 23, el espacio de representación debe estar dividido en regiones, donde existirá un centroide representativo de cada una de éstas. La Figura 24 muestra una posible división de un espacio bidimensional en regiones:

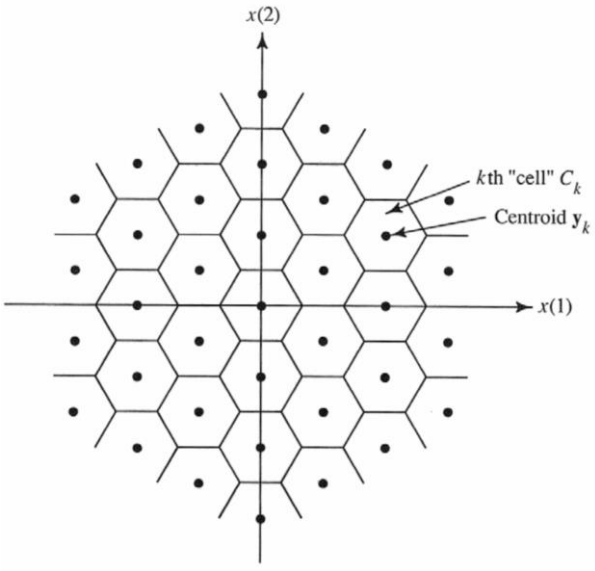


Figura 24. División de espació bidimensional

3.5.1 Ventajas de la representación VQ

1. Reducción drástica de la capacidad de almacenamiento para obtener la información del análisis espectral.
2. Reducción de la complejidad computacional en el cálculo de distancias.
3. Representación discreta de los 'sonidos' de voz.

3.5.2 Desventajas de la representación VQ

1. Introducimos una distorsión espectral al representar cada vector espectral por un representante de entre los del 'codebook' (error de cuantificación).
2. En diseños prácticos hay que obtener un balance entre tres factores:

    - Error de cuantificación (o distorsión espectral): disminuye si aumentamos el tamaño del 'codebook'
    - Elección del vector más próximo: la complejidad del cálculo aumenta con el tamaño del 'codebook'
    - Almacenamiento del 'codebook': en sistemas reales, si aumentamos el tamaño del 'codebook' podemos tener problemas de almacenamiento.

El reconocedor basado en cuantificación vectorial sigue un funcionamiento esquemático como el que se muestra a continuación:

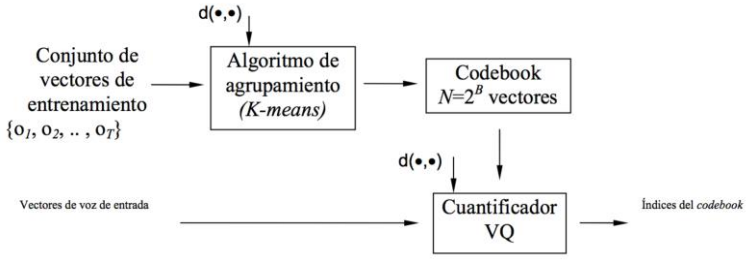


Figura 25. Funcionamiento de cuantificación vectorial

3.5.3 Algoritmo de Lloyd generalizado, o K-means

Consta de cuatro etapas:

1. Inicialización: selección de N vectores como los centroides iniciales del codebook. Hay varias posibilidades: selección aleatoria, segmentación inicial en N tramos iguales con selección aleatoria, ídem con selección promedio, etc.
2. Búsqueda NN(nearest-neighbour): para cada vector de entrenamiento, buscaremos el centroide más próximo, y asociaremos dicho vector a la clase correspondiente representada por el centroide.
3. Actualización de centroides: con los vectores asociados a cada clase, calculamos el nuevo centroide que represente mejor a los elementos de esa clase
4. Iteración: repetimos los pasos (2) y (3) hasta que la distancia promedio (distorsión) caiga por debajo de un umbral predeterminado, o hasta que no haya nuevas reasignaciones de vectores a las clases.

La Figura 26 muestra un ejemplo de división en regiones y cálculo de los centroides óptimos representativos para un caso de espacio bidimensional y asignación aleatoria inicial de los centroides:

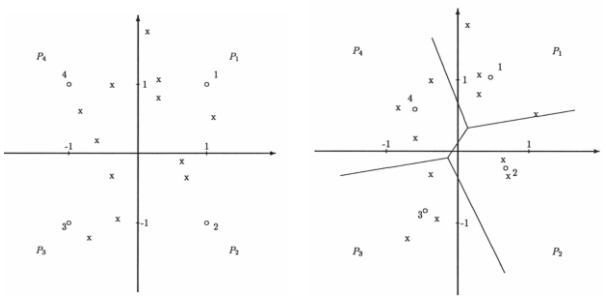


Figura 26. División en regiones

3.6 Modelos ocultos de Markov (HMM)

Otro enfoque alternativo al de medir distancias entre patrones (enfoque topográfico) es el de adoptar un modelo estadístico (paramétrico) para cada una de las palabras del vocabulario de reconocimiento, como son los modelos ocultos de Markov (HMM, del inglés 'Hidden Markov Models') (HUANG, 1992).

En un modelo estadístico se puede considerar que la señal de voz se puede caracterizar de manera apropiada mediante un proceso aleatorio, y que los parámetros del proceso estocástico pueden ser determinados de manera precisa y bien definida.

Las cadenas ocultas de Markov (HMM) es un excelente modelo para este trabajo; la teoría básica de las HMM fue publicada en una serie de artículos clásicos por Baum y sus colegas a finales de los 60 e inicios de los 70 y se utilizó por primera ocasión para el reconocimiento de voz por Baker en CMU, y por Jelinek y colegas en IBM en los años 70 (José Luis Oropeza Rodríguez D. S., 2006).

Estos sistemas han sido posteriores en el tiempo, y hoy día la mayoría de los reconocedores en funcionamiento se basan en esta técnica estadística, ya que aunque sus prestaciones son similares a las de los sistemas basados en VQ requieren menos memoria física y ofrecen un mejor tiempo de respuesta. Tienen como contrapartida, una fase de entrenamiento mucho más lenta y costosa pero como esta tarea se realiza una única vez, y se lleva a cabo en los laboratorios. Es un precio que parece valer la pena pagar.

Un HMM se puede ver como una máquina de estados finitos en que el siguiente estado depende únicamente del estado actual y asociado a cada transición entre estados se produce un vector de observaciones o parámetros. Se puede así decir que un modelo de Markov lleva asociados dos procesos: uno oculto (no observable directamente) correspondiente a las transiciones entre estados, y otro observable (y

directamente relacionado con el primero), cuyas realizaciones son los vectores de parámetros que se producen desde cada estado y que forman la plantilla a reconocer.

Para aplicar la teoría de los HMM en reconocimiento de voz, se representa cada palabra del vocabulario del reconocedor con un modelo generativo (que se calculará en la fase de entrenamiento) y posteriormente, se calcula la probabilidad de que la palabra a reconocer haya sido producida por cada uno de los modelos de la base de datos del reconocedor. Para ello, se asume que durante la pronunciación de una palabra, el aparato fonador puede adoptar sólo un número (finito de configuraciones articulatorias (o estados), y que desde cada uno de esos estados se producen uno o varios vectores de observación (puntos de la plantilla), cuyas características espectrales dependerán (probabilísticamente) del estado en el que se hayan generado Así vista la generación de la palabra, las características espectrales de cada fragmento de señal dependen del estado activo en cada instante, y la evolución del espectro de la señal durante la pronunciación de una palabra depende de la ley de transición entre estados. La representación más usual de un HMM es la utilizada para máquinas de estados finitos, es decir, conjuntos de nodos (que representar a los estados) y arcos (transiciones permitidas entre los estados). Un tipo de HMMs especialmente apropiado para reconocimiento de voz son los modelos "de izquierda a derecha"; modelos en los que una vez que se ha abandonado un estado, ya no se puede volver a él.

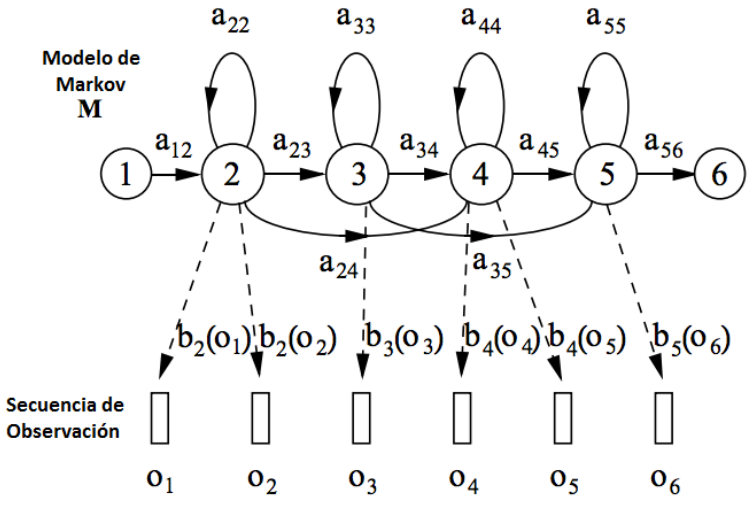


Figura 27. Modelo oculto de Markov

En cuanto a la generación de puntos de la plantilla, en estos modelos se asume que el primer vector de observaciones se produce desde el primer estado, y el último se emite desde el último estado. Recuérdese que la secuencia de estados es la parte oculta del modelo: se conocen los vectores de parámetros, pero no desde qué estado se han producido.

Para realizar la definición formal de los HMM, usamos la notación compacta:

$$\boldsymbol{\lambda} = (\boldsymbol{A}, \boldsymbol{B}, \boldsymbol{\pi}) \tag{31}$$

Para indicar el conjunto de parámetros completos del modelo. Este conjunto de parámetros, por supuesto, definen una medida de probabilidad para O, por ejemplo, $P(O|\lambda)$, donde:

- **λ :** Es representa el modelo
- **A :** Es la matriz de distribución de probabilidad de cada estado de transición
- **B :** Es la matriz de observación de los símbolos de la distribución de probabilidad
- **π :** Es representa la matriz de distribución del estado inicial

Dados los datos de la HMM anterior, los tres problemas básicos que deben ser resueltos por el modelo son los siguientes:

Problema 1: Dada una secuencia de observación O = (o1, o2, ..., oT), y un modelo λ = (A, B, π), como podemos calcular eficientemente $P(O|\lambda)$, la probabilidad de la secuencia de observación producida por el modelo.

Problema 2: Dada la secuencia de observación O = (o1, o2, ..., oT), y el modelo λ, como seleccionamos una secuencia de estados correspondiente Q = (q1, q2, ..., qT) que sea óptima en algún sentido.

Problema 3: Como ajustar los modelos de los parámetros λ=(A, B, π) para maximizar P(O $|$ λ) típicamente este problema es resulta con el algoritmo de Máxima Verosimilitud. En este problema intentamos optimizar los parámetros del modelo para describir de mejor forma como se construye una secuencia de observación dada. La secuencia de observación usada para ajustar los parámetros del modelo es llamada la secuencia de entrenamiento porque es usada para entrenar el HMM. El problema del entrenamiento es una de las aplicaciones más cruciales para el HMM, porque nos permite adaptar de manera óptima los parámetros del modelo que son observados durante el entrenamiento de datos.

El algoritmo de Máxima Verosimilitud (ML – Maximum Likelihood) es utilizado para lograr maximizar P (O $|$ λ), este algoritmo es utilizado para los casos en la que los datos están incompletos (tales como la secuencia oculta de estados) la estimación ML se puede calcular utilizando el algoritmo de Máxima-Expectativa (EM).

En el caso específico de EM para HMM, se conoce como algoritmo de Baum-Welch, desarrollado por Baum y sus colegas (Baum L. E., 1966). El algoritmo EM es una aproximación iterativa para el cálculo de la máxima verosimilitud que se utiliza para encontrar en cada paso una estimación del conjuntos de parámetros λ (Sergio Suárez Guerra, 2007) luego intenta maximizar la verosimilitud de generar los datos de entrenamiento utilizando el modelo, de tal modo que la nueva verosimilitud es mayor o igual a la previa.

Existen 2 principales tipos de HMM, los Modelos Ocultos de Markov Discretos (DHMM) y los Modelos Ocultos de Markov de Densidad Continua (CDHMM).

En los **Modelos Ocultos de Markov Discretos** las observaciones son vectores de símbolos de un alfabeto finito con M+1 elementos diferentes, cada uno de estos se le denomina “codeword”, estos se agrupan en un codebook, V={v_0, …., v_m}, donde la

distribución de probabilidad de un símbolo se define como $B=\{b_{j(k)}\}$ y donde $b_{j(k)}$ es la probabilidad de observación del codebook $v_{k.}$

Para el caso de la señal de voz, está viene representada por una plantilla o secuencia de vectores de características $X=\{X_1, X_2,...X_T\}$, donde cada X_j es un conjunto de parámetros (coeficientes LPC, cepstrum, MFFC, etc) que caracteriza la señal de voz en una ventana de tiempo centrada en $t= i$, y T es el número total de puntos. Estos vectores característicos se hacen pasar a través de un cuantificador vectorial (donde π representa la matriz de distribución del estado inicial de cada vector de parámetros X_i), la señal de voz quedará representada por una secuencia de centroides del cuantificadores de observaciones $O = \{O_1, O_2,...O_T\}$. Así las HMMs que trabajan sobre este tipo de datos son HMM discretos, por lo que B será una matriz con tantas filas como estados tenga el modelo y tantas columnas como centroides tenga el codificador vectorial, en la que cada elemento b_{jk} es la probabilidad de que, estando en el estado i se produzca el centroide k.

Por otra parte, en los **modelos ocultos de Markov de densidad continua**, se asume que las observaciones de los símbolos observables son densidades de probabilidad definidas sobre espacios de observación continuos, por lo que la forma más extendida de distribución es la de una mezcla e funciones de densidad de tipo gaussiano.

En los modelos ocultos de Markov de densidad continua se tiene que un modelo λ viene determinado por los siguientes parámetros:

- **N,** es el número de estados del modelo.
- **Matriz de transiciones**, de dimensión $(N \times N)$. Define la estructura del modelo: cada uno de sus elementos, a_{ij}, define la probabilidad de pasar del estado i al estado j. Normalmente A será bi-diagonal o tri-diagonal, significando que desde cada estado se pueden producir dos o tres tipos distintos de transición.

- **Conjunto de funciones de densidad de probabilidad (fdp),** modelan estadísticamente las observaciones producidas desde cada estado. Habrá pues tantas fdp como estados.
- **π, Vector de dimensión $\mathbf{N}$**. Cada uno de sus elementos, P_i indica la probabilidad de encontrarse inicialmente en el estado i. Para modelos de izquierda a derecha, $P_1 = 1$, y $P_j=0$ para los demás estados.

Para el caso de los DHMM, la señal de voz viene representada por una plantilla o secuencia de vectores de características $X=\{X_1, X_2,...X_T\}$, donde cada X_j es un conjunto de parámetros (coeficientes LPC, Cepstrum, MFCC, etc.) que caracteriza la señal de voz en una ventana de tiempo centrada en t= i, y T es el número total de puntos de la plantilla y B será un conjunto de fdp continuas. En la Figura 28 se muestra de forma gráfica un CDHMM.

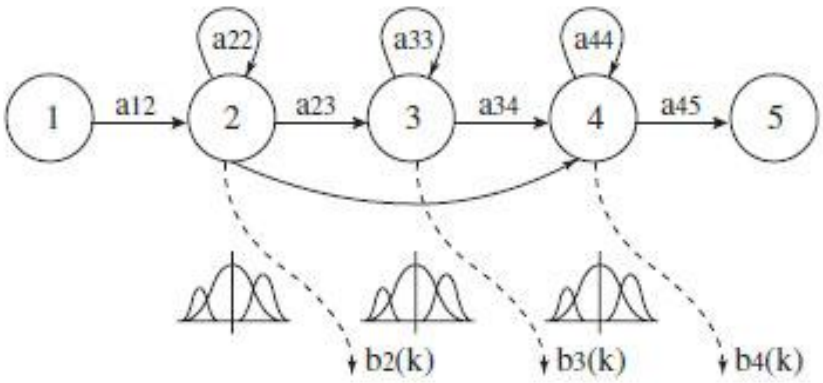


Figura 28. Modelo oculto de Markov de densidad continua (CDHMM)

3.7 Técnicas de síntesis de voz

La síntesis de voz puede ser producida por varios métodos diferentes. Todos ellos tienen algunas ventajas y deficiencias. Los métodos se suelen clasificar en tres grupos:

- **Síntesis articulatoria**, que intenta modelar directamente el sistema de producción del habla humana.

- **Síntesis por formantes**, donde se modelan las frecuencias de los polos de la señal de voz o función de transferencia del tracto vocal basado en fuente-filtro-modelo.
- **Síntesis por concatenación**, que utiliza muestras pregrabadas habla natural.

Los métodos basados en formantes y concatenados son los más comúnmente utilizados en los sistemas de síntesis presentes.

La síntesis basada en formantes fue dominante durante mucho tiempo, pero hoy en día el método de concatenación se está volviendo más y más popular.

El método articulatorio es todavía demasiado complicado de implementar para generar una síntesis de alta calidad, pero puede surgir como un método potencial en el futuro.

3.7.1. Síntesis articulatoria

La síntesis articulatoria intenta modelar los órganos vocales humanos lo más perfectamente posible, por lo que es potencialmente el método más satisfactorio para producir el habla sintetizada de alta calidad. Por otro lado, también es uno de los métodos más difíciles de implementar y la carga computacional también es considerablemente mayor que con otros métodos. Por lo que, se le ha puesto menos atención que otros métodos de síntesis y todavía no ha alcanzado el mismo nivel de éxito.

La síntesis articulatoria típicamente involucra los modelos de los articuladores humanos y las cuerdas vocales. Los articuladores están normalmente modelados con un conjunto de funciones área entre la glotis y la boca. El primer modelo articulatorio se basa en una tabla de funciones de la zona de tracto vocal de la laringe a los labios para cada segmento de fonética (Klatt D., 1987). Para la síntesis basada en reglas de los parámetros de control de articulación pueden ser, por ejemplo, la apertura de labios, protrusión del labio, la altura de punta de la lengua, la posición de la punta de la lengua, la altura de la lengua y la posición de la lengua. Parámetros fona torio o de

excitación pueden ser de apertura de la glotis, tensión de la cuerda, y la presión pulmonar (Kröger B., 1992).

Al hablar, los músculos del tracto vocal causan que los articuladores muevan y cambien la forma del tracto vocal que provoca diferentes sonidos. Los datos para el modelo articulatorio por lo general se derivan de análisis de rayos X de expresión natural. Sin embargo, estos datos están por lo general sólo 2-D cuando el tracto vocal real es, naturalmente en 3-D, por lo que la síntesis articulatoria basada en reglas es muy difícil de optimizar debido a la falta de datos para los movimientos de los articuladores durante el habla. Otra deficiencia en la síntesis articulatoria es que los datos de rayos X no caracterizan las masas o grados de libertad de los articuladores (Klatt D., 1987). Además, los movimientos de la lengua son tan complicados que es casi imposible de modelar con precisión.

Las ventajas de la síntesis articulatoria son que los modelos del tracto vocal permiten un modelado preciso de los transitorios debido a los cambios bruscos de la zona, mientras que los modelos de síntesis de formantes sólo mantienen un comportamiento espectral (O'Saughnessy D., 1987). La síntesis articulatoria es muy raro ver que se implemente en los sistemas actuales, pero dado que los métodos de análisis se están desarrollando rápidamente y los recursos computacionales están aumentando rápidamente, podría ser un método de síntesis potencial en el futuro.

3.7.2 Síntesis por formantes

Probablemente, el método más ampliamente utilizado de síntesis en las últimas décadas ha sido la síntesis de formantes que se basa en la fuente-filtro-modelo. Hay dos estructuras básicas en general, paralelo y en cascada, pero para un mejor rendimiento se utiliza generalmente una combinación de éstos. La síntesis por formantes también proporciona número infinito de sonidos que hace que sea más flexible que los otros métodos de síntesis.

Al menos tres formantes generalmente se requieren para producir el habla inteligible y hasta cinco formantes para producir voz de alta calidad. Cada formante es generalmente modelado con un resonador de dos polos que permite tanto la frecuencia de los formantes y su ancho de banda para ser especificado (Donovan R., 1996).

La síntesis por formantes está basada en reglas por lo que se basa en un conjunto de reglas que se utilizan para determinar los parámetros necesarios para sintetizar una expresión deseada utilizando un sintetizador de formantes (Allen J., 1987) .

Un sintetizador por formantes en cascada (Figura 29) consta de resonadores de tipo pasa banda conectados en serie y la salida de cada resonador de formantes se aplica a la entrada de la siguiente. La estructura en cascada necesita sólo las frecuencias de los formantes como información de control. La principal ventaja de la estructura en cascada es que las amplitudes relativas de los formantes de las vocales no necesitan controles individuales (Allen J., 1987).

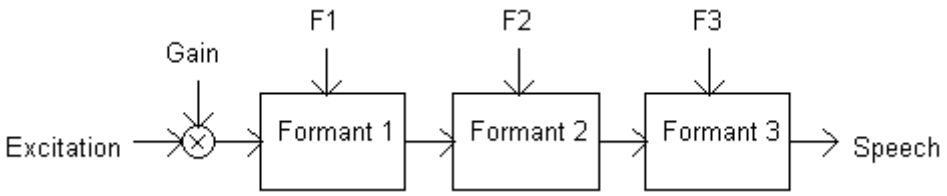


Figura 29. Estructura básica de un sintetizador por formantes en cascada.

La estructura en cascada se ha encontrado mejor para sonidos sonoros no nasales y necesita menos información de control que la estructura paralela, es entonces más sencillo de implementar. Sin embargo, con el modelo de cascada la generación de fricativas y ráfagas explosivas son un problema.

Un sintetizador por formantes en paralelo (Figura 30) consta de resonadores conectados en paralelo. A veces se utilizan resonadores adicionales para las nasales. La señal de excitación se aplica a todos los formantes simultáneamente y sus salidas se suman. Las salidas adyacentes de los resonadores de formantes deben ser

sumadas en fase opuesta para evitar ceros no deseados o anti resonantes en la respuesta de frecuencia (O'Saughnessy D., 1987). La estructura paralela permite controlar el ancho de banda y la ganancia de cada formante de forma individual y por lo tanto necesita también más información de control.

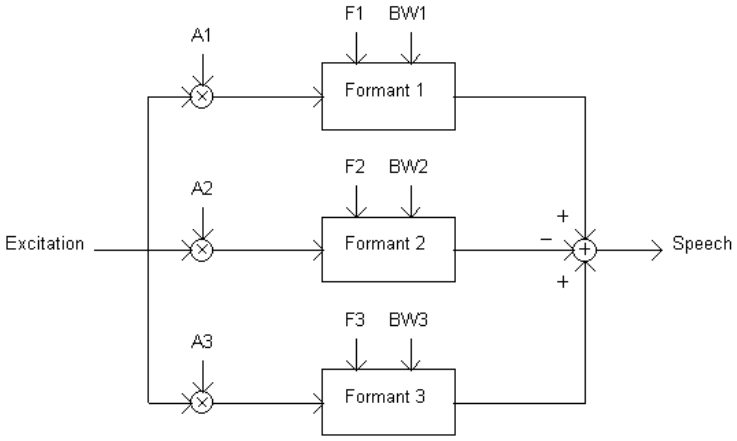


Figura 30. Estructura básica de un sintetizador de formantes en paralelo

La estructura paralela se ha encontrado que es mejor para nasales, fricativas, y detener-consonantes, pero algunas vocales no pueden ser modeladas con un sintetizador de formantes en paralelo.

Ha habido una amplia controversia sobre la calidad entre las características de estas dos estructuras. Es fácil ver los resultados de las dos arquitecturas de forma independiente, pero es difícil lograr hacer que cada arquitectura realice la tarea de forma global, por lo que han hecho algunos esfuerzos para mejorar y combinar estos modelos básicos. En 1980 Dennis Klatt (Klatt, 1980) propuso un sintetizador de formantes más complejo que incorpora tanto la cascada y los sintetizadores en paralelo con resonancias adicionales y anti-resonancias para los sonidos nasalicos.

Una solución es usar los llamados modelos parcas (Paralelo-Cascada) introducido y patentado por (Laine U., 1982). En el modelo, presentado en la Figura 31, la función de transferencia del tracto vocal uniforme se modela con dos funciones de transferencia parciales, cada una incluyendo un segundo formante de la función de

transferencia. Los coeficientes K1, K2 y K3 son constantes elegidas para equilibrar las amplitudes de los formantes de la vocal (Laine U., 1982).

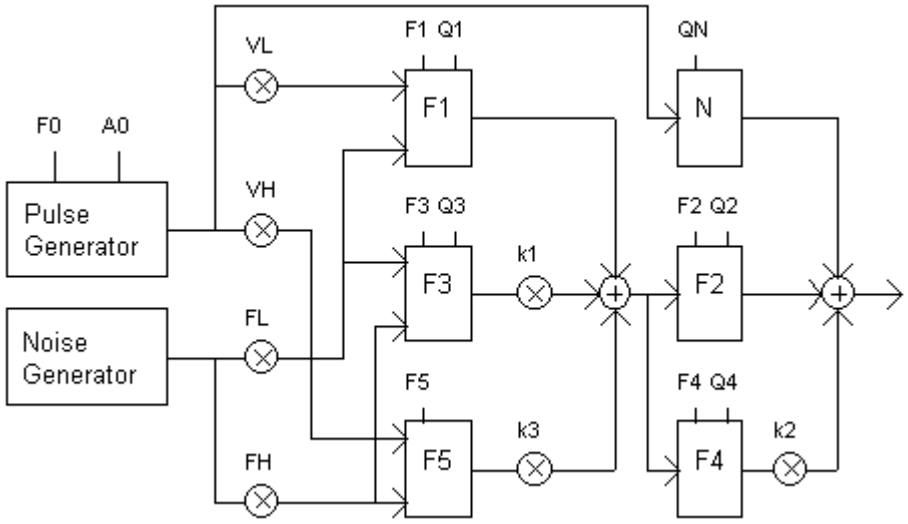


Figura 31. Modelo parcas (Laine U., 1982)

El modelo parcas utiliza un total de parámetros de control 16:

- F0 y A0 - frecuencia y la amplitud de la componente fundamental.
- Fn y Qn - frecuencias formantes y valores de Q (frecuencia formante / ancho de banda)
- VL y VH - amplitud componente sonoro, baja y alta
- FL y FH - amplitud componente sin voz, bajo y alto
- QN - Q-valor de la formante nasal a 250 Hz

La señal de excitación utilizada en la síntesis de formantes consiste en algún tipo de fuente sonora o ruido blanco. Las primeras señales de la fuente sonora utilizados eran simples señales diente de sierra. En 1981 Dennis Klatt introdujo una fuente de sonorización más sofisticada para su sistema Klattalk (Klatt D., 1987). La correcta excitación es importante sobre todo cuando se quiere buen control de las características del habla.

3.7.3 Síntesis por concatenación

En estos sintetizadores se intenta aumentar la calidad de la señal generada por medio de una minimización del ruido de codificación, para lo que se concatenan unidades digitalizadas (pregrabadas) y se ajusta su prosodia original a la de la nueva frase.

La síntesis por concatenación consiste en poner, uno detrás de otro, trozos cortos de grabaciones de un mismo locutor para reproducir la transcripción fonética con las características prosódicas requeridas. La calidad de la voz sintetizada no es igual a la de la voz real. El principal motivo de la pérdida de calidad de los sistemas de concatenación es la coarticulación. El fenómeno de coarticulación provoca que, cuando se concatenan dos segmentos de voz que no eran adyacentes en la grabación original, se produzcan discontinuidades en los puntos de concatenación, y por lo tanto se dé una pérdida de calidad/naturalidad. Hay dos tipos de discontinuidades:

- **Las discontinuidades espectrales**. Son saltos bruscos en la distribución de la frecuencia de la señal, debidos al hecho de que los formantes de las dos unidades no coinciden en el punto de concatenación.

- **Las discontinuidades prosódicas**. Se generan principalmente cuando el tono (f0) o volumen de cada unidad no coincide en el punto de concatenación.

Los sistemas de síntesis por concatenación intentan minimizar estos dos tipos de discontinuidades.

Los sistemas de síntesis por concatenación están compuestos por una base de datos (denominada corpus de unidades acústicas) y tres bloques funcionales: un bloque de selección de unidades acústicas, un bloque de modificación prosódica y un bloque de concatenación.

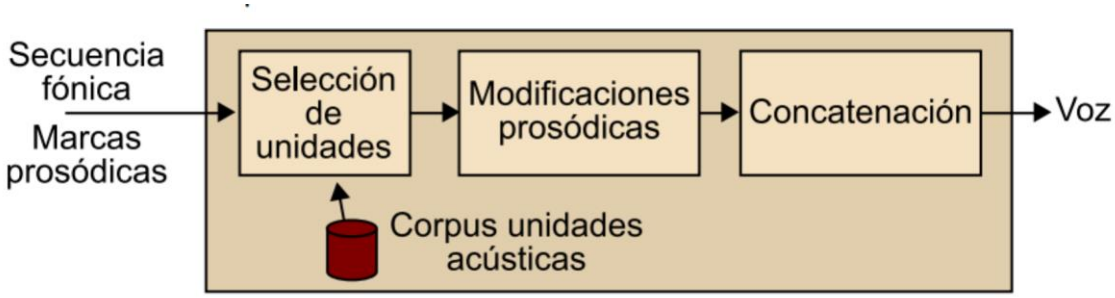


Figura 32. Módulo de un TTS basado en síntesis por concatenación

La generación de la voz empieza cuando el bloque de selección elige los segmentos de voz que se concatenarán, de entre todos los disponibles en el corpus de unidades acústicas. Esta selección se realiza teniendo en cuenta la transcripción fonética y el patrón melódico requerido. A continuación, se modifica la prosodia de cada segmento si es muy diferente de la prosodia que se quiere. En el último paso, se concatenan los segmentos.

El bloque de modificaciones prosódicas no se encuentra siempre en todos los TTS basados en concatenación. Si la dimensión del corpus de unidades acústicas es muy grande, los segmentos seleccionados tienen una prosodia muy parecida a la requerida, y por lo tanto no es necesario modificarla.

Uno de los requisitos principales en la elección de las unidades acústicas que se han de utilizar en un TTS, es que se generen pocas discontinuidades en la concatenación.

Una de las unidades más utilizadas en los TTS son los di fonemas, puesto que tienen el comienzo y final en zonas estables acústicamente y, por lo tanto, proporcionan una distorsión en la concatenación más baja que otras unidades.

Para construir el corpus se debe diseñar un conjunto de frases que incluyan todas las unidades acústicas que se querrán sintetizar y hacer grabaciones con un locutor profesional. Una vez grabado el material, se etiquetan todas las grabaciones junto con

la información prosódica que está asociada a estas (la altura tonal, la duración, la energía, etc.). Finalmente, si la unidad acústica que utilizaremos es el di fonema, se deben indicar los límites de cada segmento de señal perteneciente a un di fonema para guardarlo en el corpus junto con la información prosódica (altura tonal, duración, etc.).

El proceso de creación del corpus de un TTS requiere mucha intervención manual: la elección del locutor, las grabaciones y la revisión de los etiquetados. El proceso de etiquetado es una tarea semiautomática; esto significa que hay una primera fase de etiquetado automático y una segunda fase de revisión manual.

El objetivo de este bloque es elegir la sucesión de unidades acústicas óptima para sintetizar la descripción fonética proporcionada y que, además, se ajuste a la prosodia requerida. La elección de las unidades para una secuencia de fonemas no es trivial, puesto que en la base de datos normalmente existe más de una instancia para cada unidad, cada una de las cuales posee una prosodia diferente. La técnica más utilizada para realizar esta selección es una búsqueda utilizando el algoritmo de Viterbi, que minimiza una función de coste que representa la distorsión introducida

La función de costo está formada por dos términos:

- El costo de unidad representa la diferencia entre los valores de prosodia y de contexto que tiene la unidad elegida respecto a los valores requeridos. Normalmente se calcula de manera proporcional a la diferencia entre las características prosódicas (f0 o , duración, energía, etc.) de la unidad elegida y la descripción requerida de entrada al módulo.

- El coste de concatenación representa la pérdida de calidad debida a la unión de dos unidades. Se calcula como la distancia entre los parámetros espectrales de dos unidades adyacentes. La búsqueda prioriza la selección de unidades adyacentes en las grabaciones originales, siempre que las diferencias

prosódicas no sean muy notables. Por lo tanto, se reducen los puntos de concatenación y se evitan pérdidas de calidad.

La concatenación es el último paso para sintetizar una locución. Su objetivo es unir la secuencia de segmentos acústicos seleccionados previamente de modo que disminuyan las discontinuidades que se dan en los puntos de concatenación. Para reducir las discontinuidades se utilizan técnicas de suavización. Según el dominio en el que se apliquen, se distingue entre concatenación en el dominio temporal y concatenación en el dominio paramétrico.

La concatenación en el dominio temporal se basa en la idea de superponer y sumar o, en inglés, overlap-and-add (OLA). Esta técnica se aplica en dos fases: el análisis (cuando se definen los segmentos de voz que se superpondrán) y la síntesis (cuando se realiza la suma). En la fase de análisis se construyen ventanas de voz multiplicando los segmentos acústicos seleccionados por ventanas superpuestas. Los instantes en los que se hacen estas multiplicaciones se denominan instantes de análisis t_a (se puede ver la parte superior de la Figura 33). Una práctica habitual, denominada time domain pitch synchronous overlap-and-add (TD-PSOLA), es situar los puntos de análisis separados de un período de altura tonal y elegir ventanas de análisis de duración dos períodos de altura tonal.

En la fase de síntesis, los segmentos de voz se sitúan en los instantes de síntesis t_s que les corresponden (dependiendo de qué altura tonal tiene la voz que queremos crear) y se suman (se puede ver la parte inferior de la Figura 33). La Figura 33 muestra un ejemplo de TD-PSOLA para la concatenación de dos unidades acústicas:

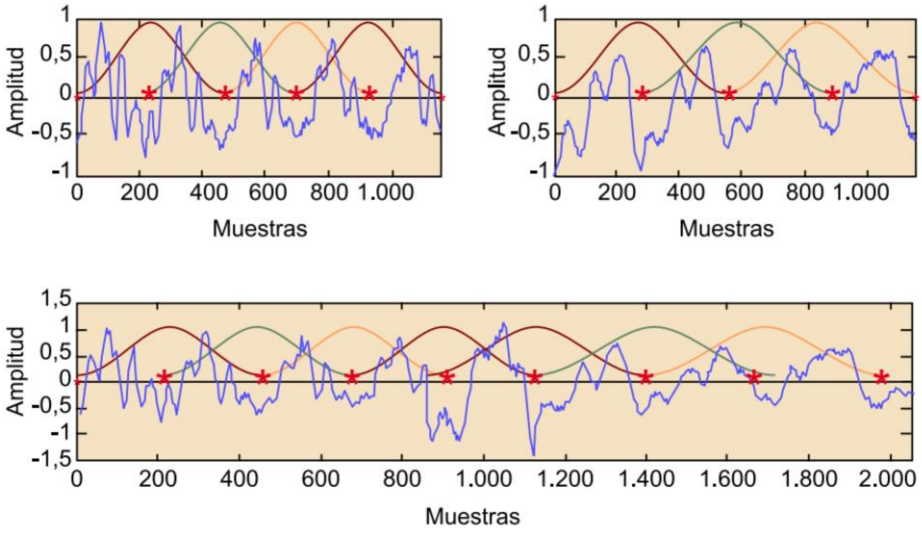


Figura 33. Concatenación TD-PSOLA de dos unidades acústicas.

Arriba: los dos segmentos temporales que se han de concatenar. Abajo: el resultado de la concatenación. Los puntos de análisis y síntesis se han indicado con un asterisco rojo.

En los intervalos que se encuentran alrededor de los puntos de concatenación, el valor de la forma de onda resultante es la interpolación lineal de las dos unidades acústicas.

3.8. HTK

HTK, (Hidden Markov Model Toolkit). Es un conjunto de herramientas de software para diseñar y manipular CHMM. Originalmente fue creado para aplicarlo al desarrollo de sistemas SARH. Ahora puede utilizarse en cualquier área del conocimiento, la única restricción es que el problema a resolver pueda ser enfocado como un proceso de modelación Estocástico de Markov.

HTK dispone de una arquitectura flexible y autosuficiente. La utilización de cualquier herramienta disponible depende de dos aspectos: Módulos de librerías y línea de comandos.

Cada librería es un conjunto de instrucciones para lograr una función específica con las herramientas disponibles. Hay librerías comunes a todos los comandos y otras son

específicas. En Figura 34 (L. R. Rabiner & B. H. Juang, 1993) muestra un esquema general de los módulos de librerías.

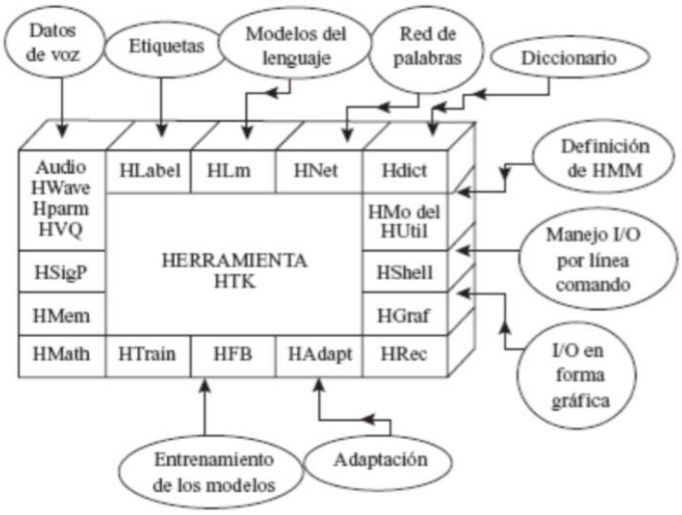


Figura 34. Diagrama de las librerías de HTK (Roberto Carrillo Aguilar, 2007)

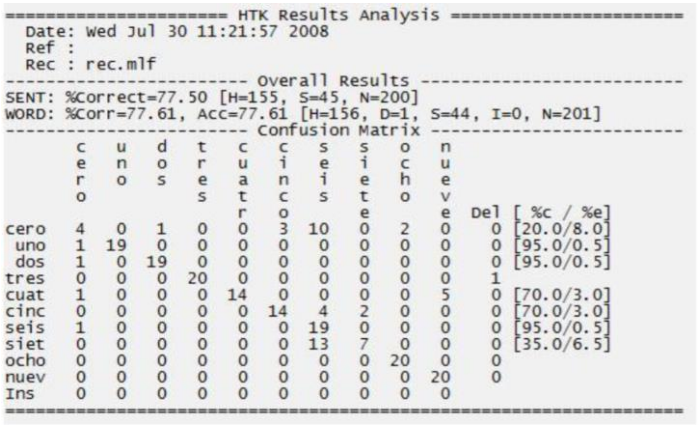
En entrenamiento básicamente, consiste en actualizar los valores de los parámetros del modelo de Markov, mediante diferentes algoritmos y, con ello, conseguir modelos más ajustados a la realidad.

HInit detectar los diferentes fonemas ayudado por las etiquetas, posteriormente, utiliza el algoritmo de Viterbi (L. R. Rabiner & B. H. Juang, 1993), para la estimación de las medias y varianzas.

HRest utiliza el algoritmo de Baum – Welch (L. R. Rabiner & B. H. Juang, 1993), (re estimación de los parámetros de los modelos aislados). Este modo de entrenamiento debe realizarse de modo iterativo, para cada fonema del conjunto.

Al invocar la herramienta HResults, ésta compara las etiquetas de entrada y salida del archivo Master Label File (MLF) o archivo de etiquetas maestro, en este archivo se encuentran las transcripciones de los datos reconocidos con los de los datos de prueba ingresados, realizando el cálculo de las tasas de reconocimiento y error cometido por el reconocedor dando lugar a una matriz de confusión de los distintos HMM. Los

resultados en el HTK, los da a conocer un archivo de salida, siendo fundamental el porcentaje de reconocimiento Figura 35 (Roberto Carrillo Aguilar, 2007).



```
====================== HTK Results Analysis =======================
  Date: Wed Jul 30 11:21:57 2008
  Ref :
  Rec : rec.mlf
------------------------ Overall Results --------------------------
SENT: %Correct=77.50 [H=155, S=45, N=200]
WORD: %Corr=77.61, Acc=77.61 [H=156, D=1, S=44, I=0, N=201]
------------------------ Confusion Matrix -------------------------
       c   u   d   t   c   c   s   s   o   n
       e   n   o   r   u   i   e   i   c   u
       r   o   s   e   a   n   i   e   h   e
       o           s   t   c   s   t   o   v
                       r   o       e       e  Del [ %c / %e]
cero   4   0   1   0   0   3  10   0   2   0    0 [20.0/8.0]
 uno   1  19   0   0   0   0   0   0   0   0    0 [95.0/0.5]
 dos   1   0  19   0   0   0   0   0   0   0    0 [95.0/0.5]
tres   0   0   0  20   0   0   0   0   0   0    1
cuat   1   0   0   0  14   0   0   0   0   5    0 [70.0/3.0]
cinc   0   0   0   0   0  14   4   2   0   0    0 [70.0/3.0]
seis   1   0   0   0   0   0  19   0   0   0    0 [95.0/0.5]
siet   0   0   0   0   0   0  13   7   0   0    0 [35.0/6.5]
ocho   0   0   0   0   0   0   0   0  20   0    0
nuev   0   0   0   0   0   0   0   0   0  20    0
Ins    0   0   0   0   0   0   0   0   0   0
===================================================================
```

Figura 35. Resultados del reconocimiento con HResults en HTK

La primera línea “SENT”, representa el resultado de la tasa de reconocimiento de oraciones. La segunda línea “WORD” presenta el resultado de la tasa de reconocimiento a nivel de palabra. En la segunda línea “Acc”, representa el porcentaje de exactitud de la tasa de reconocimiento. La letra H representa el número de etiquetas reconocidas de forma correcta, S el número de errores por sustitución, N el número total de etiquetas definidas en los archivos de configuración, D es el número de errores por omisión y finalmente I el número de errores por inserción.

CAPÍTULO 4. METODOLOGÍA Y DESARROLLO DE LA INVESTIGACIÓN.

La presente investigación consiste en diseñar e implementar una arquitectura (Figura 36) que realice la conversión de voz de una mejor manera que las arquitecturas que ya han sido propuestas anteriormente, siendo posible llevar dicha arquitectura dentro del DSP a futuro.

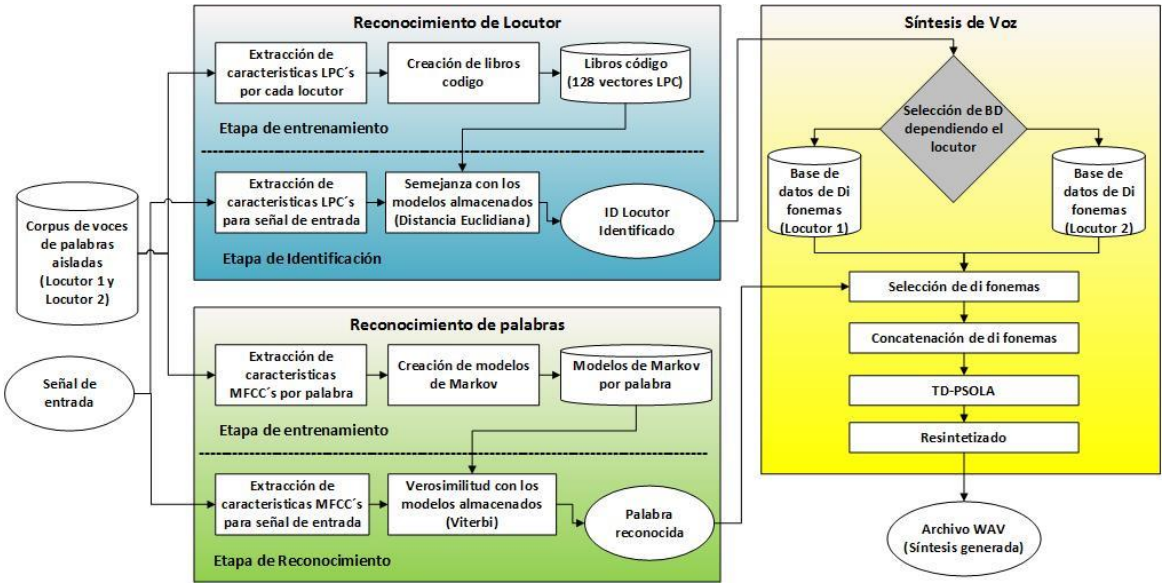


Figura 36. Arquitectura general de conversión de locutor implementada

Una vez planteada la arquitectura general del sistema, se tuvo que dividir los problemas planteados varias fases, en las que se implementaron diversas técnicas de procesamiento de señales para encontrar las técnicas que dieran mejores resultados.

Dichas fases se implementaron de forma independiente una de la otra, con el fin de realizar las pruebas de cada etapa y tener un factor de comparación entre cada técnica empleada, estas son las fases en las que se dividió el proyecto:

- Elección del corpus de palabras para 2 locutores
- Reconocimiento de locutor
- Reconocimiento de palabras aisladas
- Síntesis de la palabra reconocida

Una vez que cada fase se llevó a término de una manera apropiada, se empezó a desarrollar la siguiente y así sucesivamente, con el fin de llevar un desarrollo incremental en la investigación.

Teniendo cada fase de forma funcional (más de una técnica implementada en cada fase), se ensamblan cada una de las fases realizadas para generar la arquitectura global del sistema, y poder realizar las pruebas MOS (Mean Opinion Score) del sistema completo.

Las pruebas MOS consisten el encuestar a cierta cantidad de personas acerca de cómo perciben la calidad del audio y como evalúan el objetivo del sistema, esto lo realizan asignando una ponderación previamente establecida.

En los siguientes subcapítulos se detallan de forma puntual los aspectos involucrados en cada fase de la implementación del sistema, así como las técnicas involucradas.

4.1 Elección del corpus de palabras para 2 locutores.

Los corpus utilizados en este sistema fueron dos, cada uno de ellos consta de 40 grabaciones para cada palabra, cada uno cuanta con características similares y están grabados en formato WAV.

El primero de los corpus está constituido por los nombres de distintos estados del mundo, este corpus fue proporcionado por el laboratorio de procesamiento digital de señales del CIC; los locutores de este corpus son mujeres (Ericka y Guadalupe); el corpus está constituido por 21 estados, dando un total de 840 archivos WAV.

El segundo de los corpus está constituido por los dígitos numéricos; los locutores para este corpus son un hombre y una mujer; siendo 10 palabras, el corpus está formado por un total de archivos de 440 archivos WAV.

4.2 Reconocimiento del locutor

En la Figura 37 se muestra la etapa correspondiente al reconocimiento de locutor, está etapa toma como entrada los corpus de voces para realizar la fase de entrenamiento al generar un codebook por cada locutor y la señal de voz para la fase de identificación de locutor; la salida se genera al comparar la señal de voz de entrada con los codebooks generados en la etapa de entrenamiento identificando al locutor y entregándolo como salida.

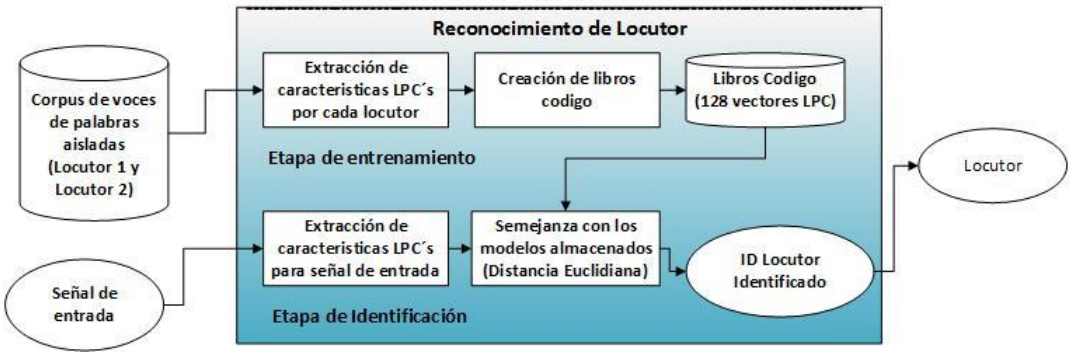


Figura 37. Esquema de reconocimiento de locutor

En ambas etapas es necesario realizar la extracción de características de tipo LPCs; para realizar el reconocimiento del locutor fue necesario generar un libro código para cada locutor conformado por los 128 vectores LPC más representativos generados a través de los vectores LPC resultantes del corpus de palabras correspondientes a cada locutor.

Fueron elegidos 128 vectores para conformar el libro código con base en pruebas realizadas con un número distinto de vectores, siendo 128 vectores los que mejor

porcentaje de reconocimiento daban a la hora de realizar las pruebas, la muestra los resultados de las pruebas realizadas.

Numero de vectores en codebook	% de reconocimiento obtenido
32	82%
64	92%
128	97%
256	90%

Dicho lo anterior para la presente tesis tenemos 2 libros código constituidos por 128 vectores, donde cada vector consta de 12 coeficientes LPC, esto no implica que haya una limitante en cuanto al número de locutores que pueden ser reconocidos por este método, siendo necesario generar tantos libros código como locutores se quieran reconocer.

El proceso de generación de libros código consta de una etapa de entrenamiento donde se generan los propios libros código para fungir como base de conocimiento para el posterior reconocimiento. A continuación se muestran los pasos para la generación de los libros código:

1. Agrupar el número total de vectores LPC correspondientes a cada locutor
2. Calcular un centroide del total de vectores LPC
3. Se empieza construyendo un libro código de tamaño 1, para esto basta con el centroide calculado en el paso anterior
4. En general, si se tiene un libro código óptimo de orden k, se puede construir un libro código óptimo de tamaño 2^k, primero perturbando cada vector código del libro óptimo, para obtener por cada vector dos vectores perturbados, y después, una vez que se tiene un libro código no óptimo de orden 2^k, aplicar el algoritmo de bipartición para optimizarlo, mostrado en la Figura 38.
5. El paso anterior se repite hasta que se obtiene el libro código óptimo del

tamaño deseado.

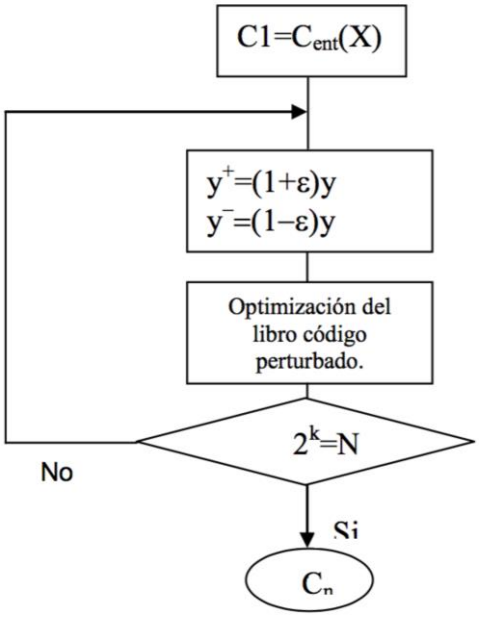


Figura 38. Algoritmo de bipartición

De manera general y gráfica la generación de los libros código se muestra en el siguiente diagrama (Figura 39).

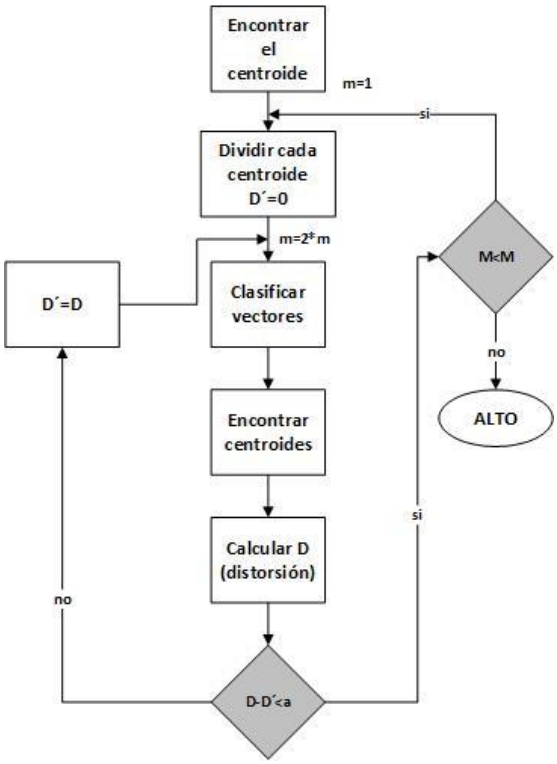


Figura 39. Algoritmo para generación de libros código

Una vez que se ha realizado la etapa de entrenamiento al extraer los libros código, ya es posible realizar pruebas para el reconocimiento del locutor, para esto es necesario tener un señal de entrada en la cual su contenido sintáctico se equivalente a alguna de las palabras contenidas en el corpus.

A esta señal es necesario realizarle una extracción de características, una vez teniendo los vectores LPC de la señal de entrada se procede a realizar una comparación entre cada libro código y los vectores LPC resultantes de la señal de entrada, lo cual genera una medida euclidiana para cada libro código.

La cuantificación vectorial busca encontrar la medida euclidiana mínima en todas las entradas de los libros código, para generar a la salida un índice correspondiente a libro código que génerо la menor distancia euclidiana.

Con el índice resultante podemos determinar cuál es el locutor que más se asemeja a la señal de voz de entrada, en la Figura 40 podemos ver de manera gráfica el proceso de reconocimiento mediante cuantificación vectorial.

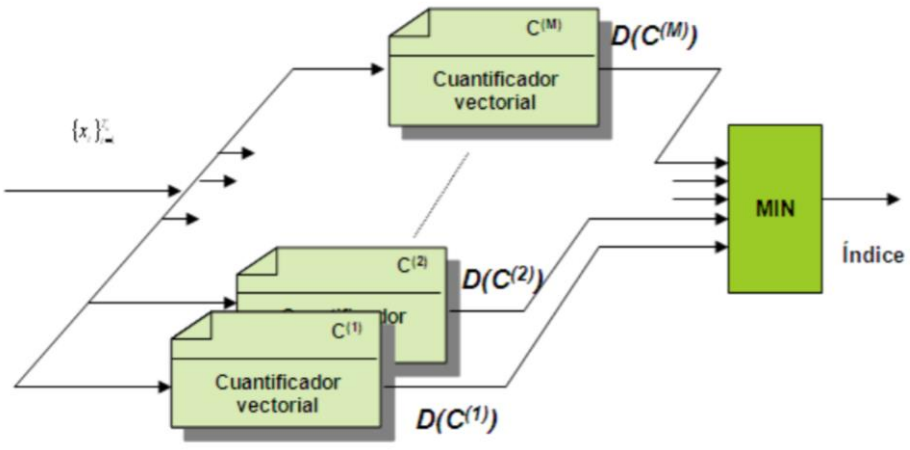


Figura 40. Etapa de reconocimiento mediante cuantificación vectorial

De manera general el diagrama de la Figura 41 nos muestra el proceso general (entrenamiento y reconocimiento) que debe realizarse para generar el reconocimiento mediante cuantificación vectorial.

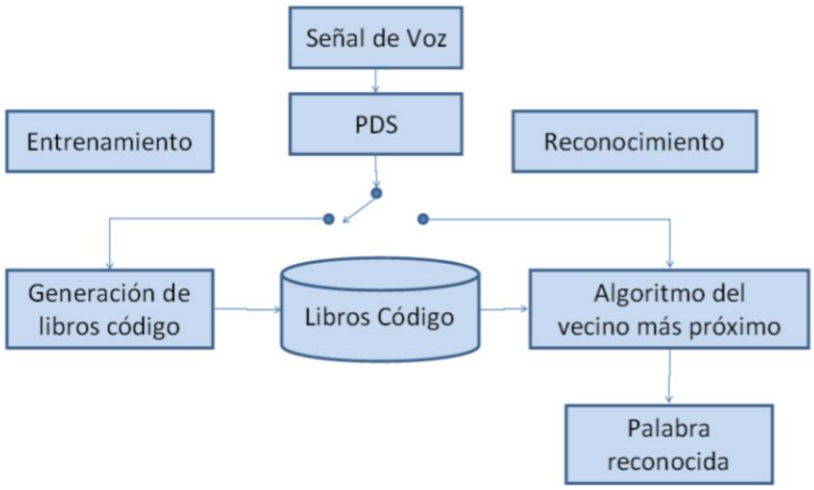


Figura 41. Esquema general de reconocimiento por cuantificación vectorial

4.3 Reconocimiento de palabras aisladas

En la Figura 42 se muestra la etapa correspondiente al reconocimiento de palabras aisladas, está etapa toma como entrada los corpus de voces para realizar la fase de entrenamiento al generar un modelo de Markov por cada palabra del corpus, y la señal de voz para la fase de reconocimiento de la palabra; la salida se genera al verificar la máxima verosimilitud entre la señal de entrada y los modelos obtenidos en la etapa de entrenamiento, para entregar el texto contenido en la grabación gracias a la gramática proporcionada.

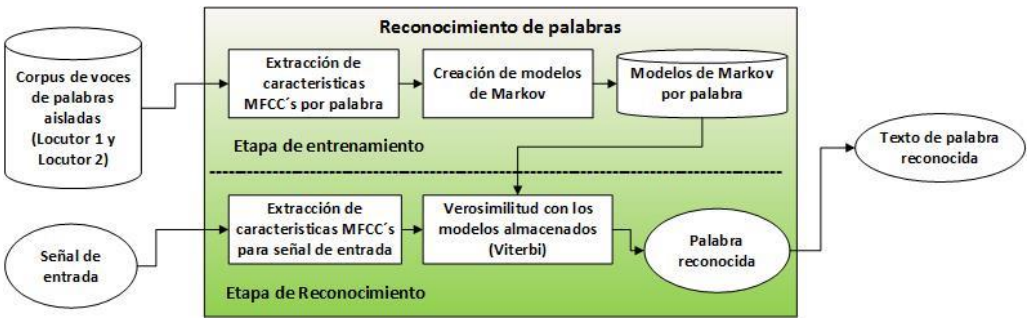


Figura 42. Esquema de reconocimiento de palabras

El reconocimiento de las palabras aisladas se realiza utilizando como características fonéticas, vectores MFCC y como algoritmo de reconocimiento modelos ocultos de Markov.

Para la realización de esta fase del proyecto se hizo uso de las librerías de HTK, ya que gran parte de las técnicas antes mencionadas tienen soporte dentro de las librerías de HTK y están implementadas en C.

Cabe resaltar que para aplicar la teoría de los HMM en reconocimiento de voz, se representa cada palabra del vocabulario del reconocedor con un modelo generativo (Problema 3 de las HMM) y posteriormente, se calcula la probabilidad de que la palabra a reconocer haya sido producida por cada uno de los modelos de la base de datos del reconocedor (Problema 1 de las HMM).

Este capítulo se centrara en describir y dar un ejemplo de los modelos de Markov que se generan con HTK debido a que ya se explicó cómo obtener las características de tipo MFCC en el capítulo 3.4 del marco teórico, aun así la Figura 43 muestra de manera general el algoritmo para la extracción de coeficientes MFCC.

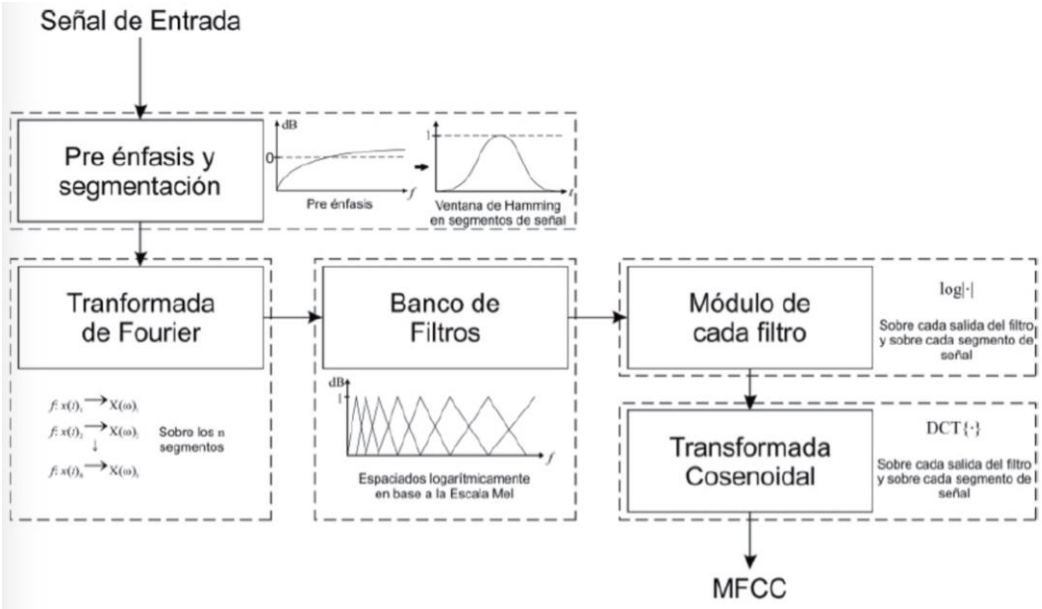


Figura 43. Algoritmo para cálculo de coeficientes MFCC

HTK usa un lenguaje propio para definir los modelos, así todas las herramientas funcionarán correctamente con el mismo HMM. Es necesario conocer este lenguaje para poder entender el prototipo utilizado en esta tesis.

El ejemplo en la Figura 44 corresponde a la palabra “Berlín” definido para el corpus de estados, es un modelo de cinco estados, tres son de emisión (los estados de entrada y salida no emiten). La primera línea indica el nombre del HMM, *~h “nombre”*. Aquí se está definiendo el modelo llamado “hmm0”. En la siguiente línea hay que poner la palabra clave *<BeginHMM>*, que indica el punto a partir del cual se encuentran los parámetros del modelo. Al final de la definición habrá otra palabra clave complementaria *<EndHMM>*.

```
~o <VecSize> 39 <MFCC_D_A >
~h  "Berlin"
<BeginHMM>
  <NumStates> 6

  <State> 2
          <Mean>39
          0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0
          0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0
          0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0
          0.0 0.0 0.0

          <Variance> 39
          1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0
          1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0
          1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0
          1.0 1.0 1.0

  <State> 3
          <Mean> 39
          0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0
          0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0
          0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0
          0.0 0.0 0.0

          <Variance> 39
          1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0
          1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0
          1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0
          1.0 1.0 1.0

  <State> 4
          <Mean>39
          0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0
          0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0
          0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0
          0.0 0.0 0.0

          <Variance>39
          1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0
          1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0
          1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0
          1.0 1.0 1.0
  <State> 5
          <Mean> 39
          0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0
          0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0
          0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0
          0.0 0.0 0.0

          <Variance>39
          1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0
          1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0
          1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0
          1.0 1.0 1.0

  <TransP> 6
          0.0 0.5 0.5 0.0 0.0 0.0
          0.0 0.4 0.3 0.3 0.0 0.0
          0.0 0.0 0.4 0.3 0.3 0.0
          0.0 0.0 0.0 0.4 0.3 0.3
          0.0 0.0 0.0 0.0 0.4 0.6
          0.0 0.0 0.0 0.0 0.0 0.0
<EndHMM>
```

Figura 44. HMM utilizado para la palabra "Berlín"

El orden dentro de la jerarquía es de mayor a menor importancia:

- <BeginHMM>, <EndHMM>
- Parámetros globales, número de estados, definición de cada estado y matriz de transición entre estados.
- Número de flujos de dato y sus pesos.
- Número de componentes mixtas y sus pesos.
- Media y varianza de cada uno de las componentes mixtas.

Al principio se especifican los aspectos globales de todos los modelos que conformen la base, los que suelen indicarse son: Tamaño de los vectores de parámetros *<VecSize> número entero* y el algoritmo con el que se han extraído la información de la forma de onda. El algoritmo se pone entre llaves, como el resto de palabras clave. En este caso *<MFCC>*. Para no dar lugar a errores, el nombre del algoritmo será el indicado durante el proceso de parametrización. En la siguiente línea se deben indicar el número de estados que componen el modelo. A partir de este punto se encuentra la definición de cada estado. La información recogida para cada estado depende de cuántos flujos de datos se utilizan y del número de componentes de mezcla Gaussiana en cada flujo. Existen también casos que mezclan varios flujos de datos y cada uno de ellos cuenta con diversas componentes de mezcla. Una vez alcanzada la componente de mezcla gaussiana como unidad fundamental, los parámetros serán:

- Vector de valores medios.
- Vector de varianza, en alguno de los siguientes formatos:
    - Varianza. El vector es la diagonal de la matriz.
    - Covarianza. La distribución gaussiana se indica como una matriz completa. Estas son simétricas, por lo que sólo se almacena la diagonal superior.

Una vez que se han definido todas las componentes de mezcla de todos los flujos y para todos los estados, el prototipo de HMM se completa con la matriz de transición

de estados <*TransP*> *tamaño*. Esta debe ser una matriz cuadrada, cuyo tamaño hay que indicar siempre de modo explícito, y debe corresponder con el número de estados definidos anteriormente. La suma de todas las componentes de cada fila debe resultar la unidad, excepto para el último estado, cuya suma debe ser siempre cero (no se permite ninguna transición que parta del estado final hacia alguno de los estados anteriores). Finalmente el modelo se termina con <EndHMM>.

Una vez obtenidos y reestimados los modelos para cada palabra, se procede a aplicar el algoritmo de Viterbi que proporcionara el camino de mayor probabilidad (verosimilitud) para la secuencia óptima de estados generando así el texto contenido en la señal de audio.

4.4 Síntesis de la palabra reconocida.

La síntesis toma como entrada los resultados de las etapas de reconocimiento de locutor y reconocimiento de palabras aisladas, el tipo de síntesis con el que se implementó esta fase es a través de concatenación de unidades lingüísticas (Figura 45).

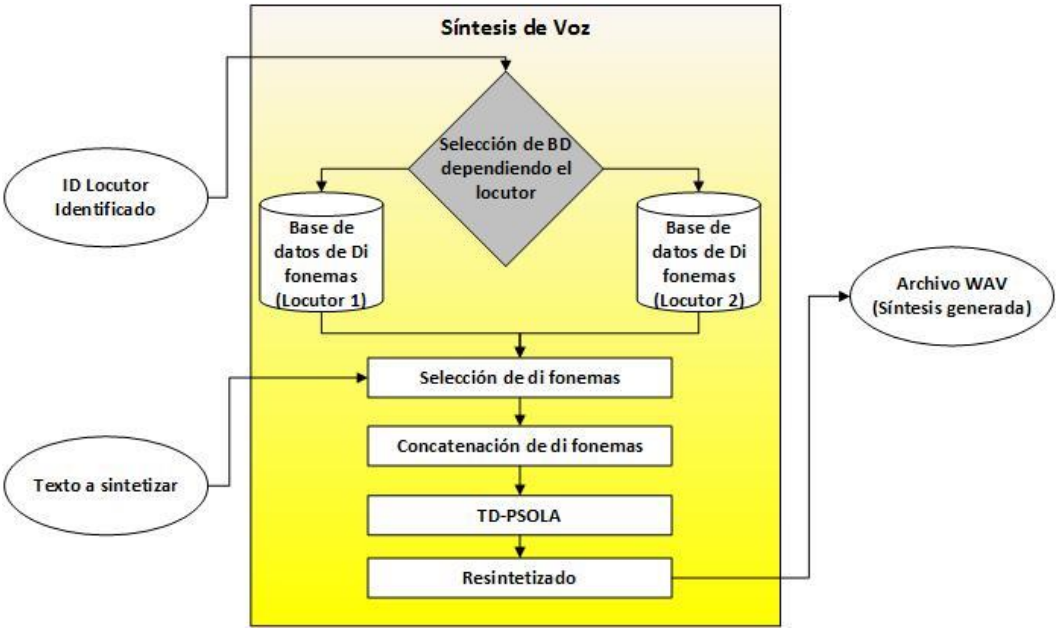


Figura 45. Esquema de síntesis de voz

Cabe resaltar que, si bien, entre más grande sea la unidad lingüística utilizada (silabas, palabras, etc.), mejor será la calidad de síntesis realizada (Suárez, 2003), el tamaño de la base de datos de las unidades lingüísticas será de mayor tamaño.

Las unidades lingüísticas utilizadas son los di fonemas, debido a que la técnica PSOLA tiene una estructura creada para trabajar con dichas unidades lingüísticas, por lo que fue necesario identificar el número total de di fonemas utilizados en el corpus de palabras, después se extraen los di fonemas del corpus de voz por cada locutor, con el fin de tener una base de datos de di fonemas para cada locutor.

La extracción de di fonemas se realizó mediante un programa proporcionado por el laboratorio de procesamiento de señales del CIC implementado en Matlab (Figura 46); al final el resultado es un grupo de archivos de audio en formato raw que constituyen la base de datos de unidades lingüísticas (di fonemas) para cada locutor.

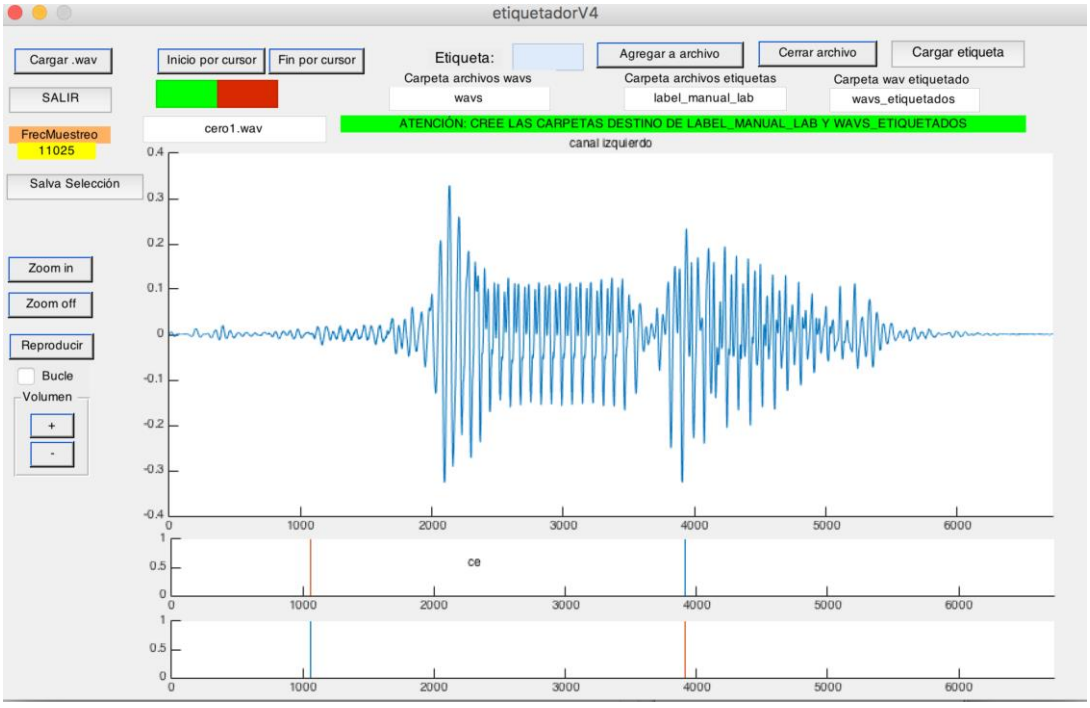


Figura 46. Programa para etiquetado manual

El resto de la fase consiste en implementar un sistema TTS basado en síntesis por concatenación de unidades lingüísticas, para convertir el texto en voz, donde los parámetros de entrada son dos:

- **Texto a sintetizar**: Proporcionado por el reconocimiento de palabras aisladas a través de la gramática utilizada en los archivos de configuración de HTK

- **Base de datos de Di fonemas**: Este proyecto se limita a 2 locutores, el reconocimiento de locutor identifica quien es el locutor que proporciona la señal de entrada, a partir de esto podemos indicar la base de datos de di fonemas con la cual queremos sintetizar, al ser un sistema de conversión de voz se tiene que proporcionar la base de datos de di fonemas del locutor que no fue reconocido para realizar la imitación de voz de una manera adecuada.

El proceso de síntesis está implementado en su totalidad en lenguaje de programación C; pero con motivos de pruebas unitarias para esta fase, se realizó una interfaz gráfica del sistema TTS, esta interfaz es mostrada en la Figura 47.

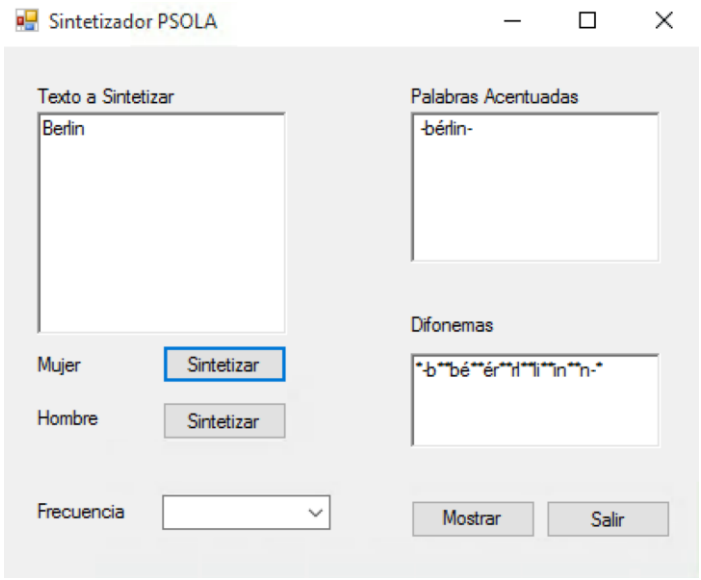


Figura 47. Sistema TTS implementado

CAPÍTULO 5. PRESENTACIÓN Y DISCUSIÓN DE RESULTADOS.

La presente investigación busca una arquitectura capaz de competir con las arquitecturas presentadas en el marco teórico para sistemas de conversión de voz, como se menciona anteriormente, la metodología separa el proyecto en diferentes etapas, con el motivo de generar pruebas unitarias en cada etapa de la arquitectura para verificar su porcentaje de efectividad.

Cada etapa de la arquitectura puede verse como una caja negra, recibe un flujo de información y genera un resultado que sirve para alimentar a la siguiente etapa, de esta manera se crea una metodología incremental que va armando la arquitectura, dando la flexibilidad de realizar pruebas unitarias (por cada etapa) y teniendo la capacidad de implementar distintas técnicas.

En los siguientes subcapítulos se reportan los resultados de las pruebas realizadas en cada etapa para los 2 distintos corpus utilizados y los resultados finales para el sistema de conversión de voz, siendo el corpus de dígitos el utilizado para generar estos resultados finales.

5.1 Características de los corpus de voces.

El primero de los corpus está constituido por los nombres de distintos estados del mundo, se utiliza con motivos de pruebas ya que al no ser grabado con el DSP TMS320C6713 genera un mayor porcentaje de error en instancias finales del sistema, en la Tabla 4 se muestran las características con las que fue grabado el corpus.

Tabla 4. Características del corpus de Estados

Característica	***Detalle de la característica***
Numero de palabras	*21*
Frecuencia de muestreo	*11025 Hz*

Formato de la grabación	*WAV*
Bits por muestra	*8*
Canales de audio	*1 (mono)*

El número total de archivos WAV que constituye el corpus es de 840 grabaciones, en la Tabla 5 se muestran las palabras con los que está conformado el corpus.

Tabla 5. Corpus Estados

Berlín	*Los Ángeles*	*Querétaro*
Buenos Aires	*México*	*Rio de Janeiro*
Chicago	*Monterrey*	*Roma*
Guadalajara	*Paris*	*Santiago*
La Paz	*Poza Rica*	*Sinaloa*
León	*Puebla*	*Tijuana*
Toronto	*Veracruz*	*Yucatán*

El segundo de los Corpus está constituido por los dígitos numéricos, este corpus es utilizado para generar los resultados finales de la presente tesis ya que fue grabado con el procesador digital de señales TMS320C6713 con el que se realizan la etapa de reconocimiento en tiempo real, la Tabla 6 se muestran las características con las que fue grabado el corpus.

Tabla 6. Características del corpus de dígitos

Característica	***Detalle de la Característica***
Numero de palabras	*10*
Frecuencia de muestreo	*8000 Hz*
Formato de la grabación	*WAV*
Bits por muestra	*8*
Canales de audio	*1 (mono)*

El número total de archivos WAV que constituye el corpus es de 440 grabaciones, en la Tabla 7 se muestran las palabras con los que está conformado el corpus.

Tabla 7. Corpus Dígitos

Cero	*Uno*
Dos	*Tres*
Cuatro	*Cinco*
Seis	*Siete*
Ocho	*Nueve*

5.1.1 Características LPC del corpus de Estados

Las características de tipo LPC fueron extraídas generando todo el código en lenguaje de programación C y teniendo en la etapa de pre-procesamiento de la señal voz las siguientes características.

Tabla 8. Características de pre-procesamiento de señal de voz para LPC´s

Característica	***Detalle de la característica***
Filtro de preénfasis	*A 2 kHz*
Tipo de ventana	*Hamming*
Tamaño de ventana	*256 muestras*
Traslape de ventana	*128 muestras*
No coeficientes LPC	*12*

El proceso de extracción de características LPCs se describe a continuación de manera general, el proceso detallado de cada técnica aplicada se describe en el capítulo 3.2 del marco teórico.

Para cada ventana generada de cada grabación se aplica el siguiente proceso con el fin de extraer 12 coeficientes LPC:

1. Se aplica un filtro de pre énfasis a toda la grabación de la palabra
2. Se divide la grabación en ventanas de 256 muestras con un traslape del 50%
3. Se aplica una ventana de Hamming a cada bloque de 256 muestras
4. Se extraen los primeros 13 coeficientes de auto correlación
5. Se aplica el algoritmo de Levison-Durbin para extraer 12 coeficientes LPC

Todos los vectores generados en esta fase darán lugar a un libro código utilizado en el reconocimiento de locutor.

5.1.2 Características MFCC del corpus de Dígitos

Las características de tipo MFCC fueron extraídas apoyándose de la librería de HTK "HCopy", la etapa de pre-procesamiento de la señal voz para el cálculo de los coeficientes MFCC tiene las siguientes características.

Tabla 9. Características de pre-procesamiento de señal de voz para MFCC´s

Característica	***Detalle de la Característica***
Filtro de preénfasis	*A 2 kHz*
Tipo de ventana	*Hamming*
Tamaño de ventana	*300 nanosegundos*
Traslape de ventana	*150 nanosegundos*
No Coeficientes LPC	*12*

La utilización de las librerías de HTK requiere de un archivo de configuración como el mostrado en la Figura 48.

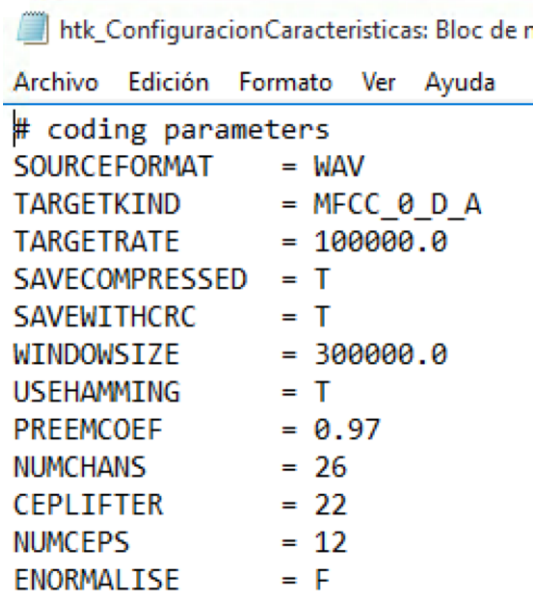


Figura 48. Archivo de configuración para HCOPY

Para el uso de HCOPY se muestra la llamada a la biblioteca en la Figura 49

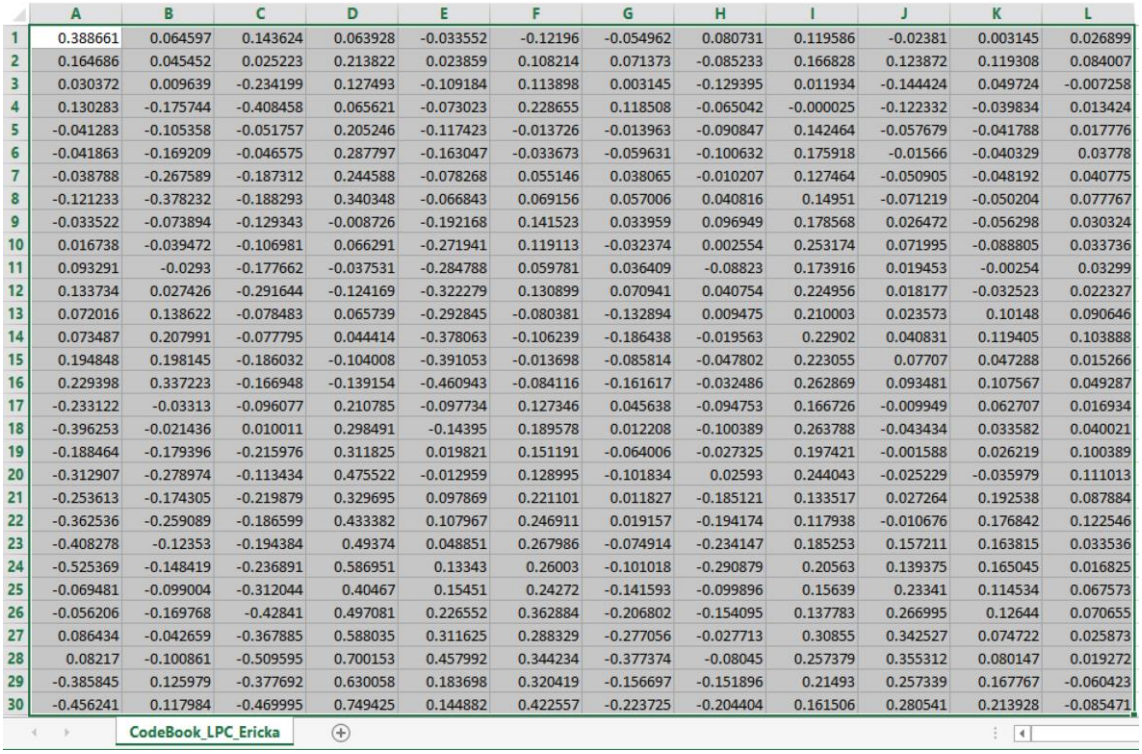
```
%HTK%\Hcopy -C htk_ConfiguracionCaracteristicas.txt -S htk_MapeoMFCC.txt
```

Figura 49. Llamada a librería HCOPY de HTK

Estas características serán utilizadas en el reconocimiento de palabras aisladas junto a los modelos ocultos de Markov.

5.2 Resultados de reconocimiento de locutor.

El reconocimiento con cuantificación vectorial se implementó utilizando libros código con 128 vectores LPC (Figura 50), para el corpus de voz de estados, fue probado con 10 grabaciones de cada palabra del corpus para cada locutor, siendo un total de 210 grabaciones pertenecientes al locutor A (Ericka) y 210 grabaciones pertenecientes al locutor B (Guadalupe).



	A	B	C	D	E	F	G	H	I	J	K	L
1	0.388661	0.064597	0.143624	0.063928	-0.033552	-0.12196	-0.054962	0.080731	0.119586	-0.02381	0.003145	0.026899
2	0.164686	0.045452	0.025223	0.213822	0.023859	0.108214	0.071373	-0.085233	0.166828	0.123872	0.119308	0.084007
3	0.030372	0.009639	-0.234199	0.127493	-0.109184	0.113898	0.003145	-0.129395	0.011934	-0.144424	0.049724	-0.007258
4	0.130283	-0.175744	-0.408458	0.065621	-0.073023	0.228655	0.118508	-0.065042	-0.000025	-0.122332	-0.039834	0.013424
5	-0.041283	-0.105358	-0.051757	0.205246	-0.117423	-0.013726	-0.013963	-0.090847	0.142464	-0.057679	-0.041788	0.017776
6	-0.041863	-0.169209	-0.046575	0.287797	-0.163047	-0.033673	-0.059631	-0.100632	0.175918	-0.01566	-0.040329	0.03778
7	-0.038788	-0.267589	-0.187312	0.244588	-0.078268	0.055146	0.038065	-0.010207	0.127464	-0.050905	-0.048192	0.040775
8	-0.121233	-0.378232	-0.188293	0.340348	-0.066843	0.069156	0.057006	0.040816	0.14951	-0.071219	-0.050204	0.077767
9	-0.033522	-0.073894	-0.129343	-0.008726	-0.192168	0.141523	0.033959	0.096949	0.178568	0.026472	-0.056298	0.030324
10	0.016738	-0.039472	-0.106981	0.066291	-0.271941	0.119113	-0.032374	0.002554	0.253174	0.071995	-0.088805	0.033736
11	0.093291	-0.0293	-0.177662	-0.037531	-0.284788	0.059781	0.036409	-0.08823	0.173916	0.019453	-0.00254	0.03299
12	0.133734	0.027426	-0.291644	-0.124169	-0.322279	0.130899	0.070941	0.040754	0.224956	0.018177	-0.032523	0.022327
13	0.072016	0.138622	-0.078483	0.065739	-0.292845	-0.080381	-0.132894	0.009475	0.210003	0.023573	0.10148	0.090646
14	0.073487	0.207991	-0.077795	0.044414	-0.378063	-0.106239	-0.186438	-0.019563	0.22902	0.040831	0.119405	0.103888
15	0.194848	0.198145	-0.186032	-0.104008	-0.391053	-0.013698	-0.085814	-0.047802	0.223055	0.07707	0.047288	0.015266
16	0.229398	0.337223	-0.166948	-0.139154	-0.460943	-0.084116	-0.161617	-0.032486	0.262869	0.093481	0.107567	0.049287
17	-0.233122	-0.03313	-0.096077	0.210785	-0.097734	0.127346	0.045638	-0.094753	0.166726	-0.009949	0.062707	0.016934
18	-0.396253	-0.021436	0.010011	0.298491	-0.14395	0.189578	0.012208	-0.100389	0.263788	-0.043434	0.033582	0.040021
19	-0.188464	-0.179396	-0.215976	0.311825	0.019821	0.151191	-0.064006	-0.027325	0.197421	-0.001588	0.026219	0.100389
20	-0.312907	-0.278974	-0.113434	0.475522	-0.012959	0.128995	-0.101834	0.02593	0.244043	-0.025229	-0.035979	0.111013
21	-0.253613	-0.174305	-0.219879	0.329695	0.097869	0.221101	0.011827	-0.185121	0.133517	0.027264	0.192538	0.087884
22	-0.362536	-0.259089	-0.186599	0.433382	0.107967	0.246911	0.019157	-0.194174	0.117938	-0.010676	0.176842	0.122546
23	-0.408278	-0.12353	-0.194384	0.49374	0.048851	0.267986	-0.074914	-0.234147	0.185253	0.157211	0.163815	0.033536
24	-0.525369	-0.148419	-0.236891	0.586951	0.13343	0.26003	-0.101018	-0.290879	0.20563	0.139375	0.165045	0.016825
25	-0.069481	-0.099004	-0.312044	0.40467	0.15451	0.24272	-0.141593	-0.099896	0.15639	0.23341	0.114534	0.067573
26	-0.056206	-0.169768	-0.42841	0.497081	0.226552	0.362884	-0.206802	-0.154095	0.137783	0.266995	0.12644	0.070655
27	0.086434	-0.042659	-0.367885	0.588035	0.311625	0.288329	-0.277056	-0.027713	0.30855	0.342527	0.074722	0.025873
28	0.08217	-0.100861	-0.509595	0.700153	0.457992	0.344234	-0.377374	-0.08045	0.257379	0.355312	0.080147	0.019272
29	-0.385845	0.125979	-0.377692	0.630058	0.183698	0.320419	-0.156697	-0.151896	0.21493	0.257339	0.167767	-0.060423
30	-0.456241	0.117984	-0.469995	0.749425	0.144882	0.422557	-0.223725	-0.204404	0.161506	0.280541	0.213928	-0.085471

CodeBook_LPC_Ericka

Figura 50. Ejemplo de libro código generado para un locutor

La matriz de confusión de la Tabla 10 muestra que se alcanza un 93% en el reconocimiento de locutor, también muestra el número de grabaciones que fueron reconocidas para cada locutor y el porcentaje de eficiencia en esta etapa de la arquitectura del sistema para el corpus de estados.

Tabla 10. Matriz de confusión para reconocimiento de locutor en corpus Estados

Palabra/Locutor	Ericka	Guadalupe	Porcentaje de Eficiencia
Berlín	9	11	95%
Buenos Aires	10	10	100%
Chicago	8	12	90%
Guadalajara	11	9	95%
Lapazo	10	10	100%
León	9	11	95%
Los Ángeles	10	10	100%
México	10	10	100%
Monterrey	7	13	85%
Paris	10	10	100%
Poza Rica	3	17	65%
Puebla	5	15	75%
Querétaro	10	10	100%

Rio Janeiro	9	11	95%
Roma	8	12	90%
Santiago	10	10	100%
Sinaloa	10	10	100%
Tijuana	10	10	100%
Toronto	8	12	90%
Veracruz	7	13	85%
Yucatán	10	10	100%
Total			93%

Para el corpus de dígitos se realiza el mismo procedimiento, es decir, el sistema se alimenta con 10 grabaciones de cada digito por locutor, dando un total de 100 grabaciones por cada locutor.

Para el caso del corpus de dígitos se realizó la prueba con un locutor de género masculino y un locutor de género femenino, la matriz de confusión de la Tabla 11 muestra que se alcanza un 97% en el reconocimiento de locutor, también muestra el número de grabaciones que fueron reconocidas para cada locutor y el porcentaje de eficiencia en esta etapa de la arquitectura del sistema para el corpus de estados.

Tabla 11. Matriz de confusión para reconocimiento de locutor en corpus Dígitos

Palabra/Locutor	Hombre	Mujer	Porcentaje de Eficiencia
Cero	10	10	100%
Uno	10	10	100%
Dos	10	10	100%
Tres	10	10	100%
Cuatro	10	10	100%
Cinco	12	8	90%
Seis	10	10	100%
Siete	14	6	80%
Ocho	10	10	100%
Nueve	10	10	100%
Total			97%

5.3 Resultados en reconocimiento de palabras.

El reconocimiento de palabras aisladas se realizó con HMM y características de tipo MFCC, utilizando las librerías de HTK se proporcionan mediante la función HResults los porcentajes de efectividad para el reconocimiento de palabras aisladas.

Para el corpus de voz de estados, el sistema de reconocimiento fue probado con las 40 grabaciones de cada palabra del corpus, siendo un total de 840 grabaciones, recordemos que 420 pertenecen al locutor A (Ericka) y 420 pertenecen al locutor B (Guadalupe).

La Figura 51 muestra el porcentaje de efectividad en el reconocimiento de las palabras aisladas, alcanzando un 99% de efectividad para el corpus de estados, asimismo la Tabla 12 muestra la matriz de confusión para el corpus de estados.

```
====================== HTK Results Analysis =======================
  Date: Sat Dec 13 17:09:28 2014
  Ref :
  Rec : htk_rec_Corpus_Completo.mlf
------------------------ Overall Results --------------------------
SENT: %Correct=99.76 [H=838, S=2, N=840]
WORD: %Corr=99.76, Acc=99.76 [H=838, D=0, S=2, I=0, N=840]
===================================================================
```

Figura 51. Porcentaje de reconocimiento de palabras aisladas en corpus de estados

Donde:

- N = Número total de palabras procesadas
- H = Número de palabras reconocidas correctamente
- S = Número de palabra no reconocidas

Tabla 12. Matriz de confusión para el reconocimiento de palabras “Corpus estados”

	Berlín	Buenos Aires	Chicago	Guadalajara	La Paz	León	Los Ángeles	México	Monterrey	Paris	Poza Rica	Puebla	Querétaro	Rio Janeiro	Roma	Santiago	Sinaloa	Tijuana	Toronto	Veracruz	Yucatán	% Efectividad
Berlín	40	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	100
Buenos Aires	0	40	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	100
Chicago	0	0	40	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	100
Guadalajara	0	0	0	40	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	100
La Paz	0	0	0	0	40	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	100
León	0	0	0	0	0	40	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	100
Los Ángeles	0	0	0	0	0	0	40	0	0	0	0	0	0	0	0	0	0	0	0	0	0	95
México	0	0	2	0	0	0	0	38	0	0	0	0	0	0	0	0	0	0	0	0	0	100
Monterrey	0	0	0	0	0	0	0	0	40	0	0	0	0	0	0	0	0	0	0	0	0	100
Paris	0	0	0	0	0	0	0	0	0	40	0	0	0	0	0	0	0	0	0	0	0	100
Poza Rica	0	0	0	0	0	0	0	0	0	0	40	0	0	0	0	0	0	0	0	0	0	100
Puebla	0	0	0	0	0	0	0	0	0	0	0	40	0	0	0	0	0	0	0	0	0	100
Querétaro	0	0	0	0	0	0	0	0	0	0	0	0	40	0	0	0	0	0	0	0	0	100
Rio Janeiro	0	0	0	0	0	0	0	0	0	0	0	0	0	40	0	0	0	0	0	0	0	100
Roma	0	0	0	0	0	0	0	0	0	0	0	0	0	0	40	0	0	0	0	0	0	100
Santiago	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	40	0	0	0	0	0	100
Sinaloa	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	40	0	0	0	0	100
Tijuana	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	40	0	0	0	100
Toronto	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	40	0	0	100
Veracruz	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	40	0	100
Yucatán	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	40	100
Total																						99.76

Para el corpus de voz de dígitos, el sistema de reconocimiento fue probado con las 40 grabaciones de cada palabra del corpus, siendo un total de 400 grabaciones, donde 200 pertenecen al locutor A (Hombre) y 200 pertenecen al locutor B (Mujer).

La Figura 52 muestra el porcentaje de efectividad en el reconocimiento de las palabras aisladas, alcanzando un 99% de efectividad para el corpus de dígitos, asimismo la Tabla 13 muestra la matriz de confusión para el corpus de dígitos.

```
====================== HTK Results Analysis =======================
  Date: Sun Jan 17 16:08:02 2016
  Ref :
  Rec : htk_rec_Corpus_Completo.mlf
------------------------ Overall Results --------------------------
SENT: %Correct=99.50 [H=398, S=2, N=400]
WORD: %Corr=99.50, Acc=99.50 [H=398, D=0, S=2, I=0, N=400]
```

Figura 52. Porcentaje de reconocimiento de palabras aisladas en corpus de dígitos

Tabla 13. Matriz de confusión para el reconocimiento de palabras “Corpus digitos”

	Cero	Uno	Dos	Tres	Cuatro	Cinco	Seis	Siete	Ocho	Nueve	% Efectividad
Cero	40	0	0	0	0	0	0	0	0	0	100
Uno	0	40	0	0	0	0	0	0	0	0	100
Dos	0	0	40	0	0	0	0	0	0	0	100
Tres	0	0	0	40	0	0	0	0	0	0	100
Cuatro	0	0	0	0	40	0	0	0	0	0	100
Cinco	0	0	0	0	0	40	0	0	0	0	100
Seis	0	0	0	0	0	0	40	0	0	0	100
Siete	0	0	0	2	0	0	0	38	0	0	95
Ocho	0	0	0	0	0	0	0	0	40	0	100
Nueve	0	0	0	0	0	0	0	0	0	40	100
Total											99.5

5.4 Resultados dela conversión de voz.

Los resultados presentados en esta sección corresponden a la totalidad de sistema, es decir, se evalúa la arquitectura total del sistema de conversión de voz teniendo como entrada el corpus de dígitos, y siendo alimentado directamente desde el DSP con grabaciones en tiempo real.

La forma de evaluar el sistema es mediante pruebas de rendimiento estadístico MOS que estudia la calidad de la voz, y los resultados mostrados dependerán de la retroalimentación de los probadores.

Las pruebas MOS consisten el encuestar a cierta cantidad de personas acerca de cómo perciben la calidad del audio y como evalúan el objetivo del sistema, esto lo realizan asignando una ponderación previamente establecida.

Para la evaluación de este sistema de conversión de voz se encuesto a 15 personas, con un total de 20 pruebas para evaluar conversión de voz, 10 para locutor A y 10 para locutor B, se realizaron 2 tipos de pruebas:

- Para evaluar la calidad del audio que genera el sintetizador
- Evaluar la capacidad del sistema de realizar la conversión de voz entre 2 locutores (Hombre / Mujer).

Las siguientes tablas muestra la ponderación para la evaluación de los sistemas

Tabla 14. Ponderación de calidad de audio para prueba MOS

Descripción	Ponderación
Excelente calidad en audio	5
Buena calidad en audio	4
Regular calidad en audio	3
Regular calidad en audio	2
Mala calidad en audio	1

La evaluación de la calidad de sintetizador arrojo los siguientes resultados:

Tabla 15. Prueba MOS para calidad del sintetizador

Encuestado	**Ponderación Promedio (20 Eventos)**
Persona 1	4.4
Persona 2	4.7
Persona 3	4.5
Persona 4	3.9
Persona 5	4.2
Persona 6	4.5
Persona 7	3.9
Persona 8	4.4
Persona 9	4.5
Persona 10	3.9
Persona 11	3.85
Persona 12	4.4
Persona 13	4.5
Persona 14	3.9
Persona 15	4.0
Ponderación Total Promedio	**4.23**

Los resultados de las pruebas para verificar el funcionamiento del sistema de conversión de voz se muestra en la Tabla 16.

Tabla 16. Porcentaje de efectividad en la conversión de voz entre locutores

Encuestado	Funciona	No Funciona
Persona 1	20	0
Persona 2	20	0
Persona 3	20	0
Persona 4	19	1
Persona 5	20	0
Persona 6	18	2
Persona 7	20	0
Persona 8	20	0
Persona 9	19	0
Persona 10	20	0
Persona 11	19	1
Persona 12	20	0
Persona 13	20	0
Persona 14	19	1
Persona 15	20	0
Porcentaje total de efectividad	98.3%	1.6%

La los resultados de la tabla anterior aunados a los resultados de las pruebas unitarias en las etapas de la arquitectura, muestra que se ha conseguido generar una arquitectura capaz de competir con las presentadas en el capítulo 2.

Debido a que el resultado de la conversión de voz de esta arquitectura, es dependiente de los resultados obtenidos del reconocimiento de locutor y del reconocimiento de palabras, podemos generar la medición cuantitativa que muestra el resultado cuantitativo de la conversión de voz, como se muestra en la Tabla 17.

Tabla 17. Porcentaje cuantitativo de efectividad para la conversión de voz

Palabra/Locutor	Hombre	Mujer	Porcentaje de Eficiencia
Cero	10	10	100%
Uno	10	10	100%
Dos	10	10	100%
Tres	10	10	100%
Cuatro	10	10	100%

Cinco	12	8	90%
Seis	10	10	100%
Siete	15	5	75%
Ocho	10	10	100%
Nueve	10	10	100%
Total			96.5%

Es importante recalcar que las motivaciones de las investigaciones mostradas en el capítulo 2 son distintas unas de las otras, también las técnicas que se utilizaron para cada investigación son distintas, en las próximas líneas discutiremos los aspectos más importantes de las arquitecturas mostradas en el estado del arte, así como los resultados que estas alcanzaron.

En el artículo “Hidden Markov Model Based Voice Conversion Using Dynamic Characteristic of Speaker”, el principal propósito es el de plantear una técnica de conversión de voz de voz basado en HMM mediante el modelado de características dinámicas del locutor, en la investigación se hizo uso de LPC´s y un sintetizador basado en coeficientes ceptrales.

Los resultados fueron comparados con un sistema de conversión de voz basado en cuantificación vectorial con libros código de 512 vectores, mientras que para la implementación de la arquitectura propuesta en este artículo se usaron HMM de 32 estados.

Las pruebas MOS que se realizaron para este artículo tienen una ponderación máxima de 3.5, las pruebas realizadas muestran que con técnicas tradicionales de cuantificación vectorial se alcanzó 3.2 en la prueba MOS, mientras que la implementación de la arquitectura con HMM y características dinámicas del locutor se alcanzó 3.41.

En el artículo “Design and evaluation of a voice conversion algorithm based on spectral envelope mapping and residual prediction”, se propone un algoritmo basado en el espectro LPC y predicción residual, para generar el sintetizado.

Este articulo pretende generar una base de datos fonéticamente rica para generar de manera adecuada los enunciados finales, las base de datos seleccionadas fueron Timit y Harvard, para dar un total de 1170 enunciados.

Estas dos comparaciones, son las que mejores resultados dieron en las arquitecturas presentadas en el estado del arte, por lo que los resultados alcanzados en la arquitectura propuesta en esta tesis, del 98% de efectividad pruebas cualitativas, y 96% en pruebas cuantitativas, muestran que es un arquitectura viable y capaz de competir con las ya existentes, por supuesto, es motivo para tomar en cuenta las diferentes técnicas ocupadas. Recordemos que uno de los objetivos de la presente tesis era el de tener una arquitectura hibrida, y enfocada a su posterior migración al dispositivo DSP, dicho objetivo es alcanzado, al tener toda la arquitectura implementada en su totalidad en lenguaje de programación C, lista para poderse migrar al DSP.

CAPÍTULO 6. CONCLUSIONES Y TRABAJO FUTURO

6.1 Conclusiones.

Los resultados obtenidos en este trabajo muestran que podemos utilizar cuantificación vectorial para la identificación de locutores tomando como base libros código generados a partir del corpus para cada locutor.

Los porcentajes de efectividad obtenidos muestran que para corpus con palabras aisladas, la efectividad del reconocimiento está a la altura de sistemas de reconocimiento de locutor con estrategias más complejas como mixturas gaussianas o HMM, teniendo a su vez un menor coste computacional que las dos estrategias mencionadas.

El reconocimiento de palabras aisladas se realizó mediante HMM dando porcentajes de reconocimiento arriba del 99% con lo cual queda claro que, el reconocimiento de voz utilizando características de tipo MFCC y reconocimiento de patrones con HMM sigue siendo uno de los métodos más viables para realizar esta tarea.

Recordando la hipótesis inicial para este sistema, se requiere generar una arquitectura para el reconocimiento de voz sin sacrificar la eficiencia y rendimiento computacional.

Al termino del trabajo de investigación se concluye que se alcanzaron los objetivos planteados, ya que la arquitectura propuesta alcanzo resultados satisfactorios en términos de las pruebas MOS realizados, en cuanto a la eficiencia y rendimiento computacional, al tener la totalidad de la arquitectura propuesta implementada en lenguaje C, se optimizaron los recursos de memoria y complejidades espaciales dentro de la arquitectura, por tanto la implementación tiene una mayor eficiencia y rendimiento que otras arquitecturas antes implementadas en lenguajes de más alto nivel como Matlab, Java, C# y VB.

La Tabla 18 muestra una comparativa entre los objetivos particulares y los resultados que se obtuvieron en este trabajo de investigación.

Tabla 18. Comparativa entre objetivos y resultados alcanzados

Objetivo	**Resultado**
Diseñar una arquitectura para realizar conversión de voz entre 2 locutores utilizando características LPCs y MFCC y técnicas de inteligencia artificial, como VQ y HMM.	La arquitectura mostrada en Figura 7 satisface las necesidades del objetivo planteado, alcanzando porcentajes cualitativos de efectividad del 98% y cuantitativos del 96%.
Realizar las tareas de entrenamiento y reconocimiento para un corpus de palabras, haciendo uso de CDHMM.	Haciendo uso de HTK se generaron los modelos de Markov de densidad continua para el reconocimiento de palabras aisladas.
Sintetizar el texto obtenido mediante un analizador sintáctico por di fonemas, el cual estará basado en el algoritmo PSOLA, y convertir la voz del locutor hablante al inferir las características del locutor diferente.	Se creó un sistema TTS utilizando síntesis por concatenación de di fonemas, y eliminando ambigüedades con la técnica TD-PSOLA; en lenguaje C
Implementar la etapa de front-end de la arquitectura en una arquitectura embebida del DSP de Texas Instruments DSK6713 de punto flotante (Lenguaje de programación C).	Se implementó la fase de front-end (Extracción de características), dentro del DSP DSK6713.
Crear el programa de entrenamiento y reconocimiento auxiliándose la herramienta HTK, así como crear el programa de síntesis por concatenación (Lenguaje de programación C).	La etapa de reconocimiento de palabras aisladas se implementó utilizando las librerías de HTK, alcanzando una efectividad de reconocimiento de 99%

6.2 Trabajo futuro.

El sistema realizado es un sistema hibrido, donde parte del procesamiento se realiza dentro del procesador digital de señales (DSP) y otra parte dentro de la PC. Resulta factible pensar en generar la expansión de memoria para que todo el trabajo quede incluido dentro del procesador digital de señales, para que éste realice de forma independiente todo el procesamiento de reconocimiento y síntesis de voz.

Así mismo queda la base para realizar trabajos sobre conversión automática de voz entre lenguas, al poderse empotrar todo el sistema en un dispositivo embebido, es teóricamente posible realizar esta tarea en tiempo real.

La presente investigación deja una arquitectura funcional para la conversión de voz entre locutores, dicha arquitectura opera y compite con las arquitecturas antes propuestas.

Así mismo se deja la base para generar una biblioteca de funciones propias del laboratorio de PDS, al haber analizado y reescrito una parte del código de HTK para la implementación de este proyecto.

El seguimiento de la presente investigación dará lugar a la inclusión de técnicas de procesamiento de lenguaje natural, para realizar conversión de voz entre distintos idiomas en tiempo real, así como para la investigación a fondo de distintas técnicas de síntesis de voz.

REFERENCIAS BIBLIOGRÁFICA

A. A. Uriz, P. A.-G. (2009). A Comparison Between GMM and non-GMM models applied in Voice Conversion. *Proceeding of 10º Argentine Symposium on Technology*, (págs. 31-44).

A. Uriz, P. A. (2009). Voice Conversion using K-Histograms and Frame Selection. *Proceeding of Interspeech 2009*.

Alexander Kain, M. W. (2001). Design and evaluation of a voice conversion algorithm based on spectral envelope mapping and residual prediction. *ICASSP*, 813-816.

Allen J., H. S. (1987). *From Text to Speech: The MITalk System.*

Arslan, L. (1999). "Speaker transformation algorithm using segmental codebooks (STASC). *Speech Communication, 28*, 211-226.

Barrobés, H. D. (2006). Voice conversion applied to text-to-speech synthesis. *PhD Thesis (Universitat Politècnica de Catalunya)*.

Baum L. E., P. T. (1966). *Statistical Inference for Probabilistic Functions of Finite State Markov Chains* (Vol. 37). Ann Math.

Cambridge University Engineering Deparment. (2009). *The HTK Book.*

Chollet, C. M. (1991). Speech recognition in adverse environments: Speech enhancement and spectral transformations. *IEEE ICASSP*, 925–928.

D. G. Childers, B. Y. (1985). Voice conversion: Factors responsible for quality. *ICASSP*, (págs. 748-751).

D. J. Broad, F. C. (1995). Formant estimation by linear transformation of the LPC cepstrum . *Journal of the Acoustic Society of America , 86*, 2013-2017.

D. Rentzos, S. V. (2003). Transformation of speaker characteristics for voice conversion. *IEEE WASRU*, 706-711.

Donovan R. (1996). *Trainable Speech Synthesis.* PhD. Thesis. Cambridge University Engineering Department, England.

Eun-Kyoung Kim, S. L.-H. (1997). HIDDEN MARKOV MODEL BASED VOICE CONVERSION USING DYNAMIC CHARACTERISTICS OF SPEAKER. *EUROSPEECH*.

Guerra, S. S. (2004). Una Metodología para realizar trabajos de reconocimiento de voz. *Centro de investigación en Computación (CIC) – IPN*.

HUANG, X. (1992). *The SPHINX-II Speech Recognition System: An Overview.* Pittsburg: CMU.

J.A. Gurlekian, L. C. (2001). *El alfabeto fonético SAMPA y el diseño de corpora fonéticamente balanceados* (Vol. 47). ASALFA.

José Luis Oropeza Rodríguez, D. S. (2006). Algoritmos y métodos para el reconocimiento de voz en español mediante sílabas. *Centro de Innovación y Desarrollo Tecnológico en Cómputo y Centro de investigación en Computación – IPN*.

José Luis Oropeza Rodríguez, S. S. (2006). Digit recognition using LPC template-based approach for the Spanish language . *CIDETEC.* Mexico D.F.

K. Shikano, K. L. (1986). Speaker adaptation through vector quantization. *ICASSP. 11.* n ICASSP.

Kei Fujii, J. O. (2007). High-Individuality Voice Conversion Based on Concatenative Speech Synthesis. *International Journal of Electrical, Computer, Energetic, Electronic and Communication Engineering, 1*.

Klatt. (1980). Software for a Cascade/Parallel Formant Synthesizer. *Journal of the Acoustical Society of America, JASA, Vol. 67*, 971-995.

Klatt D. (1987). Review of Text-to-Speech Conversion for English. *Journal of the Acoustical Society of America, JASA vol. 82 (3)*, pp.737-793.

Kröger B. (1992). Minimal Rules for Articulatory Speech Synthesis. *Proceedings of EUSIPCO92*, *1*, págs. 331-334.

L. R. Rabiner & B. H. Juang. (1993). *Fundamentals of Speech Recognition.* New Jersey: Prentice Hall .

Laine U. (1982). PARCAS, a New Terminal Analog Model for Speech Synthesis. *Proceedings of ICASSP 82 (2).*

Lanczos, C. (1966). Discourse on Fourier Series. Edinburg: Olyver and Boyd.

Lenzo, A. W. (1999). *Building Synthetic Voices.*

Luis, H. A. (2010). *Control por comandos de voz de un robot Khepera II para la manipulación de objetos.* Tesis, Mexico D.F.

M. Abe. (1991). A Segment-based approach of Voice Conversion. *Proc. of the IEEE International Conference on Acoustic* (págs. 765-768). Speech and Signal Processing.

M. Abe, K. S. (1990). Crosslanguage voice conversion. *ICASSP*, (págs. 345-348).

M. Abe, S. N. (1988). Voice Conversion through vector quantization. *Proc. of the IEEE International Conference on Acoustic,* (págs. 655- 658). Speech and Signal Processing.

M. Zhang, J. T. (2009). Phoneme cluster based state mapping for textindependent voice conversion. *ICASSP*, 4281– 4284.

Moreno, D. E. (2007). Weighted Frequency Warping for Voice Conversion. *Proceedings of theInterspeech 2007*.

Noll, A. (1964). Short-time spectrum and 'cepstrum' techniques for vocal-pitch detection. *Journal of Acoustic Society*, 296–309.

O'Saughnessy D. (1987). *Speech Communication - Human and Machine.* Addison-Wesley.

Roberto Carrillo Aguilar. (2007). Diseño y manipulación de modelos ocultos de markov, utilizando herramientas htk. una tutoría. *Ingeniare. Revista chilena de ingeniería , 15*(1), pág. 18.26.

Rodríguez, J. L. (2013). Speech recognition applied to control a simple robot using cellular phone with Android Operating System . *ELECTRO.*

Ronald A., J. M. (1996). *Survey of the State of the Art in Human Language Technology.* Cambridge University Press.

S. P. Taylor, P. T. (1998). The Architecture Of The Festival Speech Synthesis. *The Third ESCA Workshop in Speech Synthesis*, 147-151.

Sergio Suárez Guerra, J. L. (2007). Speech Recognition Using Energy, MFCCs and RO Parameters to Classify Syllables in the Spanish Language . *ICASSP.*

Stylianou, Y. (1998). Continuous probabilistic transform for voice conversion. *IEEE TSAP*, *6*, págs. 131-142.

Suárez, S. O. (2003). Pruebas y validación de un sistema de reconocimiento del habla basado en sílabas con un vocabulario pequeño. *Congreso Internacional de Computación CIC2003 .* Mexico D.F.

T. Dutoit, A. H. (2007). Towards a voice conversionsystem based on frame selection. *Proceedings of the IEEE International Conference on Acoustics* (págs. 513-516). Speech and Signal Processing.

T. Toda, H. S. (2001). Voice conversion algorithm based on gaussian mixture model with dynamic frequency warping of straight spectrum. *Power [dB], 30*, 40.

Witomski, C. G. (1999). Fourier Analysis and Applications. Filtering, Numerical Computation, Wavelets. *Texts in Applied Mathematics* (pág. 30). Springer.

Y. Stylianou, O. C. (1988). Continuous probabilistic transform for voice conversion. *Proceedings of the IEEE International Conference on Acoustics. 6*, págs. 131-142. Speech and Signal Processing.

Yannis Stylianou, O. C. (March de 1998). Continuous Probabilistic Transform for Voice Conversion. *IEEE TRANSACTIONS ON SPEECH AND AUDIO PROCESSING*.

Yegnanarayana, K. R. (2006). Voice conversion by prosody and vocal tract modification. *9th ICIT*, 111–116.

Young, H. Y. (2006). "Quality-enhanced voice morphing using maximum likelihood transformations. *IEEE TASLP, 14*, 1301-1312.

Yuxuan W., K. H. (2012). Acoustic Features for Classification Based Speech Separation. *Proceedings of Interspeech*, 1532-1535.

ANEXO A. Arquitectura del procesador TMS320C6713

El DSP utilizado para la implementación del sistema de conversión de voz es el TMS320C6713, Figura 53, para lo cual se presenta su modo de operación y su arquitectura para el correcto funcionamiento:

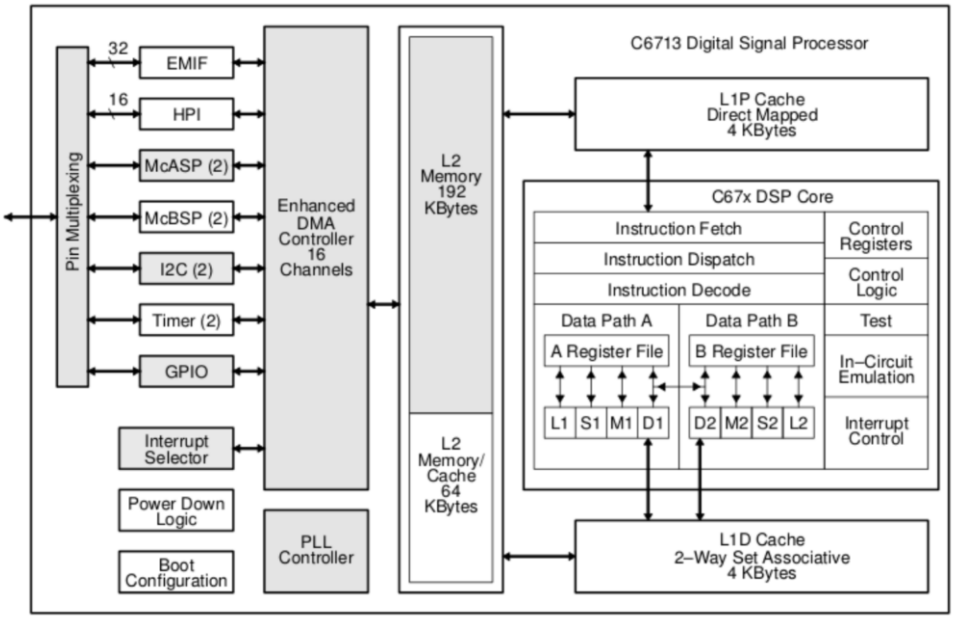


Figura 53. Arquitectura de DSP TMS320C6713

El C6713 consiste de cuatro partes principales:

1. El núcleo (C67x DSP Core)

2. La memoria
    a. La memoria de cache L1. Formada por dos bancos
        i. 4 Kbytes de memoria para programa (L1P Cache)
        ii. 4 Kbytes de memoria de datos (L1D Cache)
    b. La memoria de cache L2 de programa y datos

3. Los periféricos

4. Las rutas de datos
    a. Trayectoria A (Data Path A)
        i. Con 16 Registros de propósito general de 32 bits cada uno.
        ii. Cuatro ALU conocidas mejor como unidades funcionales, las cuales pueden operar en paralelo
    b. Trayectoria B (Data Path B)
        i. Con 16 Registros de propósito general de 32 bits cada uno.
        ii. Cuatro ALU conocidas mejor como unidades funcionales, las cuales pueden operar en paralelo.

Tabla 19. Descripción de elementos de TMS320C6713

Característica	Descripción
El TMSC6713 es un procesador en punto flotante	Procesador especializado en el cálculo de operaciones de coma flotante, las operaciones en coma flotante, se tratan de forma completamente distinta a las operaciones enteras con registros dedicados y tiempos de ciclos diferentes.
Basado en la arquitectura VLIW (Very Long Instruction Word) de 256 bits	Pueden recuperarse de memoria hasta 8 instrucciones de 32 bits a la vez.
Memoria interna incluye una arquitectura de dos niveles de caché.	El primer nivel de cache, sin una dirección en el mapa de memoria, consiste de un banco de 4KB para instrucciones (Caché de programa) y de un segundo banco de 4KB para datos (Cache de datos). El segundo nivel de caché, ocupando los primeros 256 kB en el mapa de memoria, consiste de dos bancos que

	suman 256KB. El primer banco puede ocupar hasta 192KB en tanto que el segundo banco puede ocupar hasta 64KB. Esta división del cache L2 permite dos accesos de lectura o dos accesos de escritura, al mismo tiempo.
Cuenta con dos bancos de registros de propósito general.	El banco A tiene 16 registros numerados como A0, A1, ..., A15 y el banco B contiene 16 registros numerados como B0, B1, ..., B15.
Hay ocho ALU, cada ALU tiene dos registros de entrada en donde se copian los operados y un registro de salida en donde se almacena el resultado. Estos registros adicionales liberan los registros de propósito general para ser usados por otras ALU	Hay cuatro ALU o unidades operativas (L1, S1, M1, D1) que toman sus operados del banco de registros A. Hay cuatro ALU o unidades operativas (L2, S2, M2, D2) que toman sus operados del banco de registros B. Mediante dos buses cruzados X1 y X2, las unidades operativas pueden operar sobre los registros Del banco opuesto.
En el encapsulado del DSP se incluyen también diversos circuitos integrados que implementan puertos de comunicación para varios protocolos.	EMIF (External Memory Interface File) es un circuito integrado que permite conectar varios tipos de circuitos de memoria externa. El DSK incluye un circuito integrado

	con SDRAM de 16MB y Flash ROM de 512KB. McASP (Multichannel “A “ Serial Port) Se incluyen dos circuito integrado de este puerto McBSP (Multichannel “B “ Serial Port) Se incluyen dos CI de este puerto I2C es un puerto de comunicación serial para comunicar sensores y circuitos de control basados en micro controlador. Timer, se incluyen dos circuitos contadores de las pulsaciones de reloj. ◦ GPIO (GeneralPurpose IO).
Mediante un circuito EMIF (Extende Memory Interface File) se pueden conectar bancos de memoria externos.	El EMIF CE0 conecta un circuito integrado de SDRAM de 16MB. El EMIF CE1 conecta un circuito integrado de Flash ROM de 512KB. Este banco se divide en dos bancos. El primer banco de 256KB es para aplicaciones del usuario. El segundo banco de 256KB contiene un código de arranque para la DSK.

Arquitectura VLIW

El TMS320C6713 cuenta con una arquitectura VLIW, dicha arquitectura al igual que los procesadores superescales también corresponden a los procesadores de emisión múltiple, es decir, además de que se ejecutan varias instrucciones en paralelo, también se emiten varias instrucciones en el mismo ciclo de reloj.

Pero a diferencia de los procesadores superescalares, el encargado de planificar, en cada ciclo, las instrucciones que van a ejecutarse en paralelo, es el compilador. El compilador se encarga de averiguar las dependencias entre la secuencia de instrucciones y de ir formando paquetes de emisión que contienen varias instrucciones. Todo ello, sin alterar, claro está, la semántica del programa.

El paquete de emisión es tratado por las etapas de Extracción y Emisión del procesador como si fuera una única instrucción. Desde la etapa de emisión se emite el paquete a una estación en la que se descompone en las varias instrucciones que contiene y cada una se envía a su unidad funcional correspondiente, con lo que se ejecutan todas ellas en paralelo.

Un paquete de emisión está formado por varios campos, donde cada uno de ellos corresponde a una unidad funcional del procesador. El compilador va formando grupos de instrucciones sin dependencias y cada una de ellas se pone en el campo correspondiente a su unidad funcional. Al llegar el paquete de emisión al procesador, no hay que ocuparse de las dependencias y riesgos estructurales, pues ya están resueltas de antemano, y en cada ciclo se van emitiendo estas macro-instrucciones o paquetes. Es decir, la unidad de emisión es el paquete.

Los distintos campos del paquete de emisión se corresponden con las unidades funcionales de que dispone el procesador. Así, es normal que se disponga de una unidad de aritmética entera, otra para multiplicaciones y divisiones, otra de coma flotante, otra de carga/almacenamiento, otra de tratamiento de bifurcaciones.

A continuación se listan las ventajas y desventajas de esta arquitectura.

Tabla 20. Ventajas y Desventajas de Arquitectura VLIW

Ventajas	Desventajas
Se apoya en el compilador para generar, el código paralelo, con lo que libera a la arquitectura VLIW de las dos partes más complejas de un procesador superescalar	La arquitectura VLIW simplemente mueve la complejidad del hardware (paralelización de instrucciones) al software, por lo que se requiere construir un compilador mucho más complejo que los convencionales.
El compilador puede formar una ventana de instrucciones mucho más grande que la del procesador superescalar con lo que resulta más fácil encontrar instrucciones que puedan ejecutarse en paralelo.	No resulta fácil rellenar todos los campos del paquete de emisión con instrucciones, por lo que a menudo hay campos vacíos. Lo que esto significa es que se desaprovechan los recursos del procesador.
El hardware es mucho más simple, ya que el compilador asume las tareas de crear un flujo de instrucciones paralelas.	Debido al problema anterior, el tamaño del código se incrementa, pues hay muchos huecos vacíos y desaprovechados en los paquetes de emisión.
Al eliminar las unidades encargadas de buscar las dependencias de datos y	Cuando la tecnología de integración de los chips permita la construcción de

control y de ocuparse de los riesgos estructurales, se deja mucho espacio libre en el área del chip, lo que permite emplearlo para disponer de un mayor banco de registros.	procesadores con más unidades funcionales, los programas compilados para las generaciones anteriores de procesadores, no se ejecutarán o no aprovecharán los nuevos procesadores, ya que la codificación de los paquetes de emisión depende del número de unidades funcionales.

Arquitectura VelociTI

La arquitectura VelociTI es una arquitectura VLIW, por lo que contiene las características antes mencionadas con algunas diferencias, las cuales se listan a continuación

- La arquitectura VelociTI al igual que otra VLIW permite paralelamente, buscar, decodificar y ejecutar múltiples instrucciones que componen la palabra VLIW.

- La arquitectura VelociTI establece el conjunto de instrucciones más simple para los ciclos del procesador.

- En cuestión del conjunto de instrucciones, la arquitectura VelociTI reduce el conjunto de éstas, haciéndolas simples, atómicas e independientes, logrando en cuestión del conjunto de instrucciones una arquitectura RISC.

- Esta es una arquitectura load-store, ya que las operaciones de memoria son desacopladas de las operaciones aritméticas

- Las instrucciones más frecuentes pueden ser ejecutadas con números más grandes en las unidades funcionales.

- El pipeline esta desprotegido y totalmente expuesto al compilador.

Descripción del CORE de Procesador

La UCP o núcleo del DSP contiene:

- Unidad de búsqueda de instrucciones
- Unidad de despacho de instrucciones
- Unidad de decodificación de instrucciones
- 32 registros de propósito general de 32 bits
- 8 unidades funcionales o ALU
- Registros de control
- Un control de interrupciones
- Circuitería para control y observación del DSP.

Fetch, Distpatch y Execute

Dado que se tiene una arquitectura VLIW, estas tres operaciones del Core del procesador se van a realizar en paquetes de instrucciones, paquetes que van a quedar definidos en tiempo de compilación, ya que la arquitectura VLIW es una arquitectura y depende del compilador.

El paquete de emisión es tratado por las etapas de fetch y dispach del procesador como si fuera una única instrucción. Desde la etapa de dispatch se emite el paquete a una estación en la que se descompone en las varias instrucciones que contiene y cada

una se envía a su unidad funcional correspondiente, con lo que se ejecutan todas ellas en paralelo.

Un paquete de emisión está formado por varios campos, donde cada uno de ellos corresponde a una unidad funcional de procesador. El compilador va formando grupos de instrucciones sin dependencias y cada una de ellas se pone en el campo correspondiente a su unidad funcional. Al llegar el paquete de emisión al procesador, no hay que ocuparse de las dependencias y riesgos estructurales, pues ya están resueltas de antemano, y en cada ciclo se van emitiendo estas macro-instrucciones o paquetes. Es decir, la unidad de emisión es el paquete.

Los distintos campos del paquete de emisión se corresponden con las unidades funcionales de que dispone el procesador. Así, es normal que se disponga de una unidad de aritmética entera, otra para multiplicaciones y divisiones, otra de coma flotante, otra de carga/almacenamiento, otra de tratamiento de bifurcaciones

Registros de propósito general

El DSP cuenta con dos bancos de registros de propósito general. El banco A (Register File A) tiene 16 registros numerados como A0, A1, ..., A15 y el banco B (Register File B) contiene 16 registros numerados como B0, B1, ..., B15.

- Los registros de propósito general pueden ser usador para manejar datos.
    - Los registros de propósito general pueden ser usados para manejar punteros a datos en memoria
    - El uso específico de los registros de propósito general se muestra a continuación:
    - Los registros A0, A1, B0, B1, B2 pueden ser usados como registros de condición para ciclos.
    - Los registros A4, A5, A6, A7 pueden ser usados para direccionamiento circular (búfer circular)

- Los registros B4, B5, B6, B7 pueden ser usados para direccionamiento circular (búfer circular)
- Los registros A10, A11, A12, A13, A14, A15 se llaman registros de usuario y pueden ser usados para manejar datos y para manejar punteros a datos en memoria: registros para el usuario. Estos registros se guardan en la pila cuando se invoca una subrutina.
- Los registros B10, B11, B12, B13, B14 se llaman registros de usuario y pueden ser usados para manejar datos y para manejar punteros a datos en memoria (direccionamiento indirecto): registros para el usuario. Estos registros se guardan en la pila cuando se invoca una subrutina.
- B15: Apuntador de la pila

- Los registros de propósito general soportan datos de 32 bits para punto fijo y para punto flotante.

- Empleando dos registros de propósito general se pueden soportar datos de 40 bits para punto fijo.

- Los 32 bits menos significativos son colocados en un registro par y los restantes ocho bits más significativos son colocados en las localidades menos significativas del próximo registro superior (registro impar).

Unidades Funcionales

Se tienen ocho unidades operativas denominadas como: .L1, .S1, .M1, .D1 y .L2 .S2, .M2, .D2. Cada ALU tiene dos registros de entrada en donde se copian los operados y un registro de salida en donde se almacena el resultado.

Hay cuatro unidades operativas (.L1, .S1, .M1, .D1) o ALU que toman sus operados en el banco de registros A.

- L1
  - Realiza operaciones lógicas.
  - Realiza operaciones aritméticas (suma y resta) en punto fijo.
  - Realiza operaciones aritméticas (suma y resta) en punto flotante.

- S1
  - Realiza operaciones de manipulación de bits
  - Realiza operaciones de corrimiento de bits.
  - Realiza operaciones de salto.
  - Generación de constantes.
  - Realiza operaciones aritméticas (suma y resta) en punto fijo.
  - Realiza operaciones aritméticas (suma y resta) en punto flotante.

- D1
  - Realiza operaciones de lectura de datos (load).
  - Realiza operaciones de escritura de datos (store).
  - Realiza operaciones aritméticas (suma y resta) en punto fijo.

- M1
  - Realiza operaciones de multiplicación en punto fijo.
  - Realiza operaciones de multiplicación en punto flotante.
  - Calcula el recíproco

Hay cuatro unidades operativas (.L2, .S2, .M2, .D2) o ALU que toman sus operados en el banco de registros B. Sus funciones son idénticas a las de las unidades 1.

Mediante dos buses cruzados X1 y X2, las unidades operativas pueden tomar y operar sobre los registros del banco opuesto.

Las unidades .L1, .L2, .D1, .D2 tienen un puerto extra de 8 bits para operaciones de 40 bits.

Debido a que cada unidad tiene su propio conjunto de puertos, las 8 unidades pueden ser usadas en paralelo en cada ciclo.

Tabla 21. Unidades Operativas

Unidad	Operaciones Logicas	Suma punto fijo	suma punto flotante	Multiplicación punto fijo	Multiplicación punto flotante	Desplaz. de bits	Saltos	Lectura escritura
L	*	*	*					
S		*	*			*	*	
D		*						*
M				*	*			

Registros de Control

Hay un banco de registros que controlan e modo de operación del DSP. Una unidad (.S2) puede leer de y escribir hacia los registros de control. La Tabla 22 describe los registros de control. Cada registro es accesado mediante la instrucción MVC.

El C67x posee tres registros de configuración adicionales, para soportar operaciones de punto flotante. Los registros especifican los modos de redondeo de punto flotante para las unidades .M y .L También poseen campos de bits para advertir si src1 y src2 son NaN (no es un número) o números des normalizados.

Además si resulta overflow o underflow, es inexacto, infinito o inválido. Hay campos que advierte si una división por cero fue ejecutada o si una comparación fue ejecutada con un NaN.

Tabla 22. Registro de Control

Abreviatura	Nombre	Descripción
AMR	Regitro de modo de direccionamiento	Especifica si utiliza direccionamiento lineal o circular oara cada uno de los ocho registros, también contiene el tamalo para el direccionamiento circular.
CSR	Registro de control de estado	Contiene el bit de interrućión global, los bits de control del cache y tros bits de contro de estado misceláneos.
IFR	Registgro de bandera de interrupción	Despliega el estado de las interrupciones.
ISR	Registros para activar interrupción	Permite activar interrupciones manualmente.
ICR	Registro para interrución	Permite limpiar interrupciones pendientes manualmente.
IER	Registro para retorno de interrupción	Permite habilitar /deshabilitar interrupciones individuales.
NRP	Puntero de retorno de interrupción no mascarable	Contiene la dirección de retorno de una interrupción no mascarable.
PCEI	Contador del programa, fase E1	Contiene la dirección del paquete fetch (contiene le paquete de ejecución del pipeline) en la etapa E1.

Tabla 23. Registros de Control Extendidos

Abreviatura	Nombre	Descripción
FADCR	Registro de configuración del sumador del punto flotante.	Especifica el modo de underflow, modo de redondeo, NaN y otras excepciones para la unidad .L
FAUCR	Regitro de configuración auxiliar de punto flotante.	Especifica modos de underflow, modos de redondeo, NaN y otras excepciones para la unidad .S
FMCR	Regsitro de configuración del cmultiplicador de punto flotante.	Especifica modos de underflow, modos de redondeo, NaN y otras excepciones para la unidad .M

Test

Este pequeño modulo incorporado en el procesador se encarga de ejecutar una rutina de autoevaluación que consta de 16 pasos de verificación para determinar que el procesador está operando correctamente.

Control de interrupciones

El procesador tiene un total de 16 interrupciones a nivel de hardware, el control de interrupciones va a ser el encargado de gestionar y en dado momento ligar dichas interrupciones a su correspondiente ISR (Interrup Service Routine), para su posterior gestión.

Dentro de las interrupciones por hardware, tiene 16 diferentes tipos, de las que solo permite elegir 12 interrupciones y también permite cambiar la polaridad de entrada para las interrupciones externas.

Tabla 24. Interrupciones del TMS320C6713

Interrupción	Nombre
HWI RESET	Reset
HWI NMI	Reset
HWI RESERVED0	Reset
HWI RESERVED1	Reset
HWI INT4	Pin externo 4
HWI INT5	Pin externo 5
HWI INT6	Pin externo 6
HWI INT7	Pin externo 7
HWI INT8	Controlador EDMA
HWI INT9	Transmisión McSPS 0
HWI INT10	EMIF
HWI INT11	Recepción McBSP 0
HWI INT12	McBSP 1
HWI INT13	Host
HWI INT14	Timer 0
HWI INT15	Timer 1

Descripciones de las Caches del Procesador

El primer nivel de cache, sin una dirección en el mapa de memoria, consiste de un banco de 4KB para instrucciones (Caché de programa) y de un segundo banco de 4KB para datos (Cache de datos).

El segundo nivel de caché, ocupando los primeros 256 kB en el mapa de memoria, consiste de dos bancos que suman 256KB. El primer banco puede ocupar hasta 192KB en tanto que el segundo banco puede ocupar hasta 64KB. Esta división del cache L2 permite dos accesos de lectura o dos accesos de escritura, al mismo tiempo.

Propiedades y uso del EDMA

Este controlador tiene mejorado las funciones de transferir datos al mapa de memoria, sin la intervención del CPU. EDMA puede obtener o mover datos de la memoria interna, periféricos internos o de dispositivos externos. Tiene 16 canales independientes, todos ellos programables, de tal forma que cada uno de ellos tiene 16 diferentes tipos de operaciones. También cuenta con un espacio de RAM para soportar múltiples configuraciones de futuras transferencias.

Timers, EMIFs, McBSPs, GPIOs, HPI, PCI; entre otros.

HPI (Interface al Puerto Host)

El HPI es un puerto paralelo de 16 bits por medio del cual, un procesador Host puede accesar directamente al espacio de memoria del CPU. El dispositivo host tiene la facilidad de acceso debido a que es el maestro de la interfaz. El dispositivo Host y el CPU pueden intercambiar información a través de la memoria interna o externa. El Host tiene acceso directo a los periféricos de memoria mapeada.

EMIF (Interfase de Memoria Externa)

El EMIF soporta una interfaz de baja adherencia (glueless) para varios dispositivos externos. Los tipos de memoria que soporta son:

- SRAM de ráfaga síncrona (SBSRAM).
- DRAM síncrona (SDRAM).
- Dispositivos asíncronos, incluyendo SRAM, ROM y FIFO’s.
- Dispositivo externo de memoria compartida.

McBSP

El McBSP está basado en las interfaces estándar del puerto serial encontrado en las plataformas TMS320C2000 y C5000. Este puerto almacena muestras seriales en un buffer de memoria automáticamente, con o sin la ayuda del controlador EDMA.

El DSP6713 tiene dos puertos serial multicanal con buffer (McBSP0 y McBSP1). El McBSP0 es el que controla de manera unidireccional y es congelado a través de cinco registros que controlan el lujo a través del McBSP1. El McBSP1 es usado como un canal de comunicación bidireccional de datos y proporciona:

- Comunicación Full - Dúplex.
- Registros de datos de doble buffer para flujo continúo de datos.
- Tramado independientes para recepción y transmisión.
- Interfaz directa a códec estándar, chips de interface analógico (AICs) y otros dispositivos A/D y D/A conectados serial mente.
- El McBSP tiene las siguientes capacidades:
- Transmisión y recepción multicanal de 128 canales.
- Selector para determinar el tamaño del dato (8, 12, 16, 20, 24 y 32 bits).
- Transferencia inicial de 8 bits con LSB o MSB.
- Polaridad programable para tramas sincronizadas.
- Reloj interno altamente programable y generador de tramas.

TIMER

El C6713 tiene dos Timer's de propósito general que se utilizan para:

- Eventos del Timer.
- Eventos de contador.
- Generador de pulsos.
- Interrupción del CPU.
- Enviar eventos de sincronización al controlador EDMA.

ANEXO B. Procedimiento Para Realizar Reconocimiento de Palabras Con HTK

El reconocimiento de las palabras aisladas se realiza utilizando como características fonéticas, vectores MFCC y como algoritmo de reconocimiento modelos ocultos de Markov.

Para la realización de esta fase del proyecto se hizo uso de las librerías de HTK, ya que gran parte de las técnicas antes mencionadas tienen soporte dentro de las librerías de HTK y están implementadas en C.

Como obtener las características de tipo MFCC ya fue explicado en el capítulo 3.4, por lo que este capítulo se enfoca a ver cuál es el proceso de entrenamiento y reconocimiento de palabras haciendo uso de HMM con las librerías de HTK.

El reconocimiento de locutor de basa en Modelos ocultos de Markov de densidad continúa, las funciones utilizadas de las librerías HTK para el reconocimiento de palabras aisladas son:

- HCopy
- HInit
- HRest
- HParse
- HVite
- HResults

Al igual que el reconocimiento de locutor, tenemos que dividir en 2 etapas el reconocimiento de palabras aisladas, etapa de entrenamiento y etapa de reconocimiento, el manejo de las funciones y las tareas que ejecuta cada función son explicados en los siguientes subcapítulos.

Etapa de entrenamiento usando HTK

Una vez obtenidas las características de tipo MFCC tendremos a la salida archivos con la extensión ".MFCC", como se muestra en la Figura 54 que contienen los coeficientes MFCC.

Nombre	Fecha de modifica...	Tipo	Tamaño
Sinaloa34.mfcc	13/12/2014 17:09	Archivo MFCC	6 KB
Sinaloa35.mfcc	13/12/2014 17:09	Archivo MFCC	5 KB
Sinaloa36.mfcc	13/12/2014 17:09	Archivo MFCC	7 KB
Sinaloa37.mfcc	13/12/2014 17:09	Archivo MFCC	6 KB
Sinaloa38.mfcc	13/12/2014 17:09	Archivo MFCC	6 KB
Sinaloa39.mfcc	13/12/2014 17:09	Archivo MFCC	6 KB
Sinaloa40.mfcc	13/12/2014 17:09	Archivo MFCC	6 KB
Tijuana1.mfcc	13/12/2014 17:09	Archivo MFCC	5 KB
Tijuana2.mfcc	13/12/2014 17:09	Archivo MFCC	5 KB
Tijuana3.mfcc	13/12/2014 17:09	Archivo MFCC	5 KB
Tijuana4.mfcc	13/12/2014 17:09	Archivo MFCC	5 KB

Figura 54. Archivos de características MFCC

El siguiente paso es generar las matrices para los modelos ocultos de Markov, esto se logra aplicando la siguiente instrucción.

```
%HTK%\HInit -A -T -D 1 -S htk_MFCC.txt -l Berlin -M %CarpetaEntrenamiento%\hmm0 Berlin.txt
%HTK%\HInit -A -T -D 1 -S htk_MFCC.txt -l BuenosAires -M %CarpetaEntrenamiento%\hmm0 BuenosAires.txt
%HTK%\HInit -A -T -D 1 -S htk_MFCC.txt -l Chicago -M %CarpetaEntrenamiento%\hmm0 Chicago.txt
%HTK%\HInit -A -T -D 1 -S htk_MFCC.txt -l Guadalajara -M %CarpetaEntrenamiento%\hmm0 Guadalajara.txt
%HTK%\HInit -A -T -D 1 -S htk_MFCC.txt -l LaPaz -M %CarpetaEntrenamiento%\hmm0 LaPaz.txt
%HTK%\HInit -A -T -D 1 -S htk_MFCC.txt -l Leon -M %CarpetaEntrenamiento%\hmm0 Leon.txt
%HTK%\HInit -A -T -D 1 -S htk_MFCC.txt -l LosAngeles -M %CarpetaEntrenamiento%\hmm0 LosAngeles.txt
%HTK%\HInit -A -T -D 1 -S htk_MFCC.txt -l Mexico -M %CarpetaEntrenamiento%\hmm0 Mexico.txt
%HTK%\HInit -A -T -D 1 -S htk_MFCC.txt -l Monterrey -M %CarpetaEntrenamiento%\hmm0 Monterrey.txt
%HTK%\HInit -A -T -D 1 -S htk_MFCC.txt -l Paris -M %CarpetaEntrenamiento%\hmm0 Paris.txt
%HTK%\HInit -A -T -D 1 -S htk_MFCC.txt -l PozaRica -M %CarpetaEntrenamiento%\hmm0 PozaRica.txt
%HTK%\HInit -A -T -D 1 -S htk_MFCC.txt -l Puebla -M %CarpetaEntrenamiento%\hmm0 Puebla.txt
%HTK%\HInit -A -T -D 1 -S htk_MFCC.txt -l Queretaro -M %CarpetaEntrenamiento%\hmm0 Queretaro.txt
%HTK%\HInit -A -T -D 1 -S htk_MFCC.txt -l RioJaneiro -M %CarpetaEntrenamiento%\hmm0 RioJaneiro.txt
%HTK%\HInit -A -T -D 1 -S htk_MFCC.txt -l Roma -M %CarpetaEntrenamiento%\hmm0 Roma.txt
%HTK%\HInit -A -T -D 1 -S htk_MFCC.txt -l Santiago -M %CarpetaEntrenamiento%\hmm0 Santiago.txt
%HTK%\HInit -A -T -D 1 -S htk_MFCC.txt -l Sinaloa -M %CarpetaEntrenamiento%\hmm0 Sinaloa.txt
%HTK%\HInit -A -T -D 1 -S htk_MFCC.txt -l Tijuana -M %CarpetaEntrenamiento%\hmm0 Tijuana.txt
%HTK%\HInit -A -T -D 1 -S htk_MFCC.txt -l Toronto -M %CarpetaEntrenamiento%\hmm0 Toronto.txt
%HTK%\HInit -A -T -D 1 -S htk_MFCC.txt -l Veracruz -M %CarpetaEntrenamiento%\hmm0 Veracruz.txt
%HTK%\HInit -A -T -D 1 -S htk_MFCC.txt -l Yucatan -M %CarpetaEntrenamiento%\hmm0 Yucatan.txt
```

Figura 55. Uso de la función HInit

Para la reestimación de las matrices de los modelos ocultos se hará uso de la función HRest, su uso se muestra a continuación.

```
%HTK%\HRest -A -T -D 1 -S htk_MFCC.txt -l Berlin -M %CarpetaEntrenamiento%\hmm1 %CarpetaEntrenamiento%\hmm0\Berlin
%HTK%\HRest -A -T -D 1 -S htk_MFCC.txt -l BuenosAires -M %CarpetaEntrenamiento%\hmm1 %CarpetaEntrenamiento%\hmm0\BuenosAires
%HTK%\HRest -A -T -D 1 -S htk_MFCC.txt -l Chicago -M %CarpetaEntrenamiento%\hmm1 %CarpetaEntrenamiento%\hmm0\Chicago
%HTK%\HRest -A -T -D 1 -S htk_MFCC.txt -l Guadalajara -M %CarpetaEntrenamiento%\hmm1 %CarpetaEntrenamiento%\hmm0\Guadalajara
%HTK%\HRest -A -T -D 1 -S htk_MFCC.txt -l LaPaz -M %CarpetaEntrenamiento%\hmm1 %CarpetaEntrenamiento%\hmm0\LaPaz
%HTK%\HRest -A -T -D 1 -S htk_MFCC.txt -l Leon -M %CarpetaEntrenamiento%\hmm1 %CarpetaEntrenamiento%\hmm0\Leon
%HTK%\HRest -A -T -D 1 -S htk_MFCC.txt -l LosAngeles -M %CarpetaEntrenamiento%\hmm1 %CarpetaEntrenamiento%\hmm0\LosAngeles
%HTK%\HRest -A -T -D 1 -S htk_MFCC.txt -l Mexico -M %CarpetaEntrenamiento%\hmm1 %CarpetaEntrenamiento%\hmm0\Mexico
%HTK%\HRest -A -T -D 1 -S htk_MFCC.txt -l Monterrey -M %CarpetaEntrenamiento%\hmm1 %CarpetaEntrenamiento%\hmm0\Monterrey
%HTK%\HRest -A -T -D 1 -S htk_MFCC.txt -l Paris -M %CarpetaEntrenamiento%\hmm1 %CarpetaEntrenamiento%\hmm0\Paris
%HTK%\HRest -A -T -D 1 -S htk_MFCC.txt -l PozaRica -M %CarpetaEntrenamiento%\hmm1 %CarpetaEntrenamiento%\hmm0\PozaRica
%HTK%\HRest -A -T -D 1 -S htk_MFCC.txt -l Puebla -M %CarpetaEntrenamiento%\hmm1 %CarpetaEntrenamiento%\hmm0\Puebla
%HTK%\HRest -A -T -D 1 -S htk_MFCC.txt -l Queretaro -M %CarpetaEntrenamiento%\hmm1 %CarpetaEntrenamiento%\hmm0\Queretaro
%HTK%\HRest -A -T -D 1 -S htk_MFCC.txt -l RioJaneiro -M %CarpetaEntrenamiento%\hmm1 %CarpetaEntrenamiento%\hmm0\RioJaneiro
%HTK%\HRest -A -T -D 1 -S htk_MFCC.txt -l Roma -M %CarpetaEntrenamiento%\hmm1 %CarpetaEntrenamiento%\hmm0\Roma
%HTK%\HRest -A -T -D 1 -S htk_MFCC.txt -l Santiago -M %CarpetaEntrenamiento%\hmm1 %CarpetaEntrenamiento%\hmm0\Santiago
%HTK%\HRest -A -T -D 1 -S htk_MFCC.txt -l Sinaloa -M %CarpetaEntrenamiento%\hmm1 %CarpetaEntrenamiento%\hmm0\Sinaloa
%HTK%\HRest -A -T -D 1 -S htk_MFCC.txt -l Tijuana -M %CarpetaEntrenamiento%\hmm1 %CarpetaEntrenamiento%\hmm0\Tijuana
%HTK%\HRest -A -T -D 1 -S htk_MFCC.txt -l Toronto -M %CarpetaEntrenamiento%\hmm1 %CarpetaEntrenamiento%\hmm0\Toronto
%HTK%\HRest -A -T -D 1 -S htk_MFCC.txt -l Veracruz -M %CarpetaEntrenamiento%\hmm1 %CarpetaEntrenamiento%\hmm0\Veracruz
%HTK%\HRest -A -T -D 1 -S htk_MFCC.txt -l Yucatan -M %CarpetaEntrenamiento%\hmm1 %CarpetaEntrenamiento%\hmm0\Yucatan
```

Figura 56. Uso de la función HRest

La función anterior reestima la matrices de probabilísticas de los modelos de Markov, por lo que será necesario repetir su uso 2 veces más. Con lo anterior tendremos los modelos de Markov entrenados.

Teniendo un modelo de Markov entrenado por cada palabra, se tiene la base de conocimiento necesaria para realizar el reconocimiento de la palabra.

Etapa de reconocimiento usando HTK

La etapa de reconocimiento toma como base las matrices de los Modelos de Markov entrenados, se toma una señal de voz como entrada la cual es procesada para obtener las características y poder comparar contra los modelos previamente entrenados.

La Figura 57 muestra de manera general los pasos y las funciones necesarias para realizar el reconocimiento.

```
%HTK%\Hparse -A -D -T 1 htk_Gramatica.txt htk_red.slf
%HTK%\Hcopy -C htk_ConfiguracionCaracteristicas.txt -S htk_MapeoMFCC.txt
%HTK%\HVite -A -T -D 1 -S htk_MFCC_Para_Reconocimiento.txt -H htk_hmmdefs.hmm
%HTK%\hresults -A -T -D 1 -e ??? sil htk_ListaHMM.txt htk_rec.mlf > ..\htk_Res
```

Figura 57. Instrucciones para reconocimiento de palabras aisladas usando HTK

La función HParse toma como entrada un archivo donde se especifica la gramática del lenguaje (Figura 58) y genera los autómatas necesarios para la implementación de la gramática proporcionada.

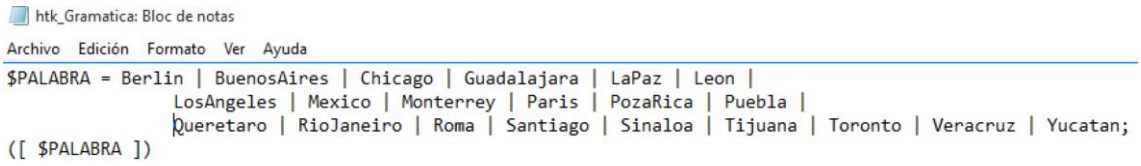


Figura 58. Ejemplo del archivo de gramática utilizada por HTK

La función HCopy genera las características de tipo MFCC del mismo modo que se realiza en la etapa de entrenamiento, solo que ahora aplicada a la señal de entrada.

La función HVite implementa y hace uso del algoritmo de Viterbi para realizar la comparación de los modelos entrenado con las características extraídas para establecer cuál de los modelos tiene la máxima verosimilitud con la señal de entrada y dar por asentado la palabra reconocida.

La función HResult simplemente genera una presentación de los resultados del reconocimiento, mostrando el porcentaje del reconocimiento, la Figura 59 muestra un ejemplo del archivo generado por Hresult.

```
====================== HTK Results Analysis =======================
  Date: Sat Dec 13 17:09:26 2014
  Ref :
  Rec : htk_rec.mlf
------------------------ Overall Results --------------------------
SENT: %Correct=99.76 [H=419, S=1, N=420]
WORD: %Corr=99.76, Acc=99.76 [H=419, D=0, S=1, I=0, N=420]
===================================================================
```

Figura 59. Archivo de resultados de reconocimiento con HResults

ANEXO C. Algoritmo de FFT Mediante Decimación en Frecuencia

A continuación se realiza un análisis de cómo obtener la FFT mediante un algoritmo de decimación en frecuencia, de manera dual podría deducirse la FFT mediante decimación en tiempo. Para el cálculo de la FFT **se requiere que el tamaño del segmento de análisis sea de tamaño N en potencia de 2**, y se distinguen las muestras pares e impares de la transformada X[k].

$$\boldsymbol{X[k] = \sum_{n=0}^{N-1} x[n] e^{j\frac{2\pi}{N}nk} \quad con \quad k = 0, 1, \dots, N-1} \tag{32}$$

Donde las muestras pares son

$$\boldsymbol{X[2r] = \sum_{n=0}^{N-1} x[n] e^{j\frac{2\pi}{N}n(2r)} \quad con \quad r = 0, 1, \dots, (\frac{N}{2} - 1)} \tag{33}$$

Donde (33) Puede se expresada en 2 sumatorias

$$\boldsymbol{X[2r] = \sum_{n=0}^{(\frac{N}{2}-1)} x[n] e^{j\frac{2\pi}{N}n(2r)} + \sum_{n=\frac{N}{2}}^{N-1} x[n] e^{j\frac{2\pi}{N}n(2r)}} \tag{34}$$

Sustituyen en (34) tenemos

$$\boldsymbol{X[2r] = \sum_{n=0}^{(\frac{N}{2}-1)} x[n] e^{j\frac{2\pi}{N}n(2r)} + \sum_{n=\frac{N}{2}}^{N-1} x[n + (N/2)] e^{j\frac{2\pi}{N}[n+N/2](2r)}} \tag{35}$$

Por razones de periodicidad en $e^{j\frac{2\pi}{N}n(2r)} = W_N^{n(2r)}$

$$\boldsymbol{W_N^{[n+(N/2)](2r)} = e^{j\frac{2\pi}{N}[n+(N/2)](2r)} = e^{j\frac{2\pi}{N}n(2r)}e^{j\frac{2\pi}{N}Nr} = W_N^{n(2r)}} \tag{36}$$

Dado que $e^{j\frac{2\pi}{N}n(2r)} = W_N^{n(2r)} = e^{j\frac{2\pi}{N/2}nr} = W_{N/2}^{nr}$ la ecuación (35) puede reescribirse

$$\boldsymbol{X[2r] = \sum_{n=0}^{(N/2)-1} (x[n] + x[n+(N/2)])e^{j\frac{2\pi}{N/2}nr} \quad con \quad r = 0, 1, \dots, (\frac{N}{2} - 1)} \tag{37}$$

La ecuación anterior nos da las muestras pares de DFT, ahora solo bastaría calcular las muestras impares de la DFT, y éstas se calculan a partir de la siguiente ecuación

$$\boldsymbol{X[2r+1] = \sum_{n=0}^{N-1} x[n]e^{j\frac{2\pi}{N}n(2r+1)} \quad con \quad r = 0, 1, \dots, (\frac{N}{2} - 1)} \tag{38}$$

La ecuación (38) puede reescribirse de la siguiente manera

$$\boldsymbol{X[2r+1] = \sum_{n=0}^{(\frac{N}{2}-1)} x[n]e^{j\frac{2\pi}{N}n(2r+1)} + \sum_{n=\frac{N}{2}}^{N-1} x[n]e^{j\frac{2\pi}{N}n(2r+1)}} \tag{39}$$

La segunda suma de la ecuación (39) es:

$$\begin{aligned}\sum_{n=\frac{N}{2}}^{N-1} \boldsymbol{x[n]e^{j\frac{2\pi}{N}n(2r+1)}} &= \sum_{n=0}^{\frac{N}{2}-1} \boldsymbol{x[n+(N/2)]e^{j\frac{2\pi}{N}[n+N/2](2r+1)}} \\ &= \boldsymbol{e^{j\frac{2\pi}{N}[N/2](2r+1)}} \sum_{n=0}^{\frac{N}{2}-1} \boldsymbol{x[n+(N/2)]e^{j\frac{2\pi}{N}n(2r+1)}} \\ &= \sum_{n=0}^{\frac{N}{2}-1} \boldsymbol{x[n+(N/2)]e^{j\frac{2\pi}{N}n(2r+1)}}\end{aligned} \tag{40}$$

Sustituyendo (40) en (39) tenemos

$$\boldsymbol{X[2r+1]} = \sum_{n=0}^{(N/2)-1} \boldsymbol{(x[n]+x[n+(N/2)])e^{j\frac{2\pi}{\frac{N}{2}}n(r+1)}} \tag{41}$$

Notando que $e^{j\frac{2\pi}{N}2} = W_N^2 = e^{j\frac{2\pi}{N/2}} = W_{N/2}$ la ecuación (35) puede reescribirse

$$\boldsymbol{X[2r+1]} = \sum_{n=0}^{\frac{N}{2}-1} \{\boldsymbol{x[n]x[n+(N/2)]e^{j\frac{2\pi}{N}n}}\ \}\boldsymbol{e^{j\frac{2\pi}{N/2}nr}}\ \boldsymbol{con}\ \ \boldsymbol{r=0,1,\dots,(\frac{N}{2}-1)} \tag{42}$$

La ecuación (42) es la DFT de (N/2) puntos, que se obtiene restando a la primera sucesión de entrada la segunda y multiplicando la sucesión resultante por $e^{j\frac{2\pi}{n}n}$.

Por lo que de acuerdo con (37) y (42) tenemos que

$$\boldsymbol{g[n]} = \boldsymbol{x[n]} + \boldsymbol{x\left[n+\frac{N}{2}\right]} \quad \boldsymbol{y} \quad \boldsymbol{h[n]} = \boldsymbol{x[n]} + \boldsymbol{x\left[n+\frac{N}{2}\right]} \tag{43}$$

La DFT puede calcularse formando primero las sucesiones g[n] y h[n] , calculando luego $h[n]e^{\frac{2\pi}{N}n}$ y finalmente calculando la DFT de (N/2) puntos de esas 2 sucesiones, con lo que se obtienen las muestras pares e impares de la DFT, solamente que aplicarle el algoritmo de bit-reverse para ordenar adecuadamente la transformada

Para ilustrar el método tomemos un ejemplo de 8 muestras, es decir N = 8, como se muestran en las siguientes figuras

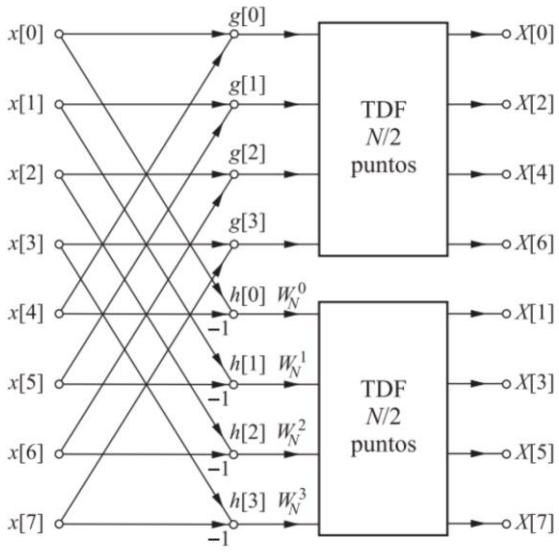


Figura 60. Calculo de la DFT de N puntos con 2 DFT de (N/2) puntos con N=8

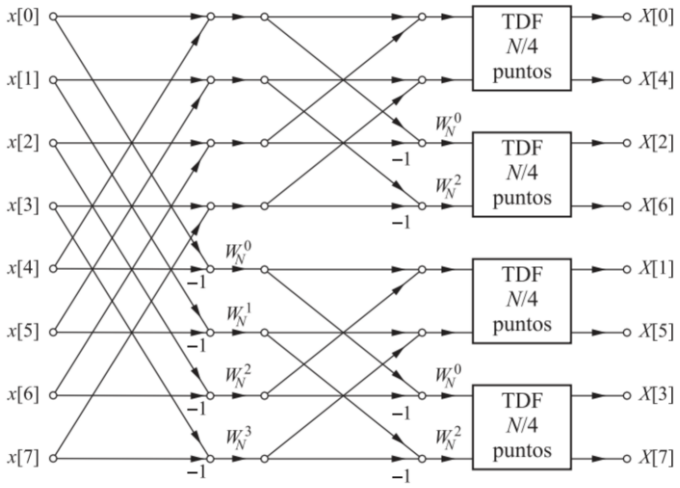


Figura 61. Descomposición de una DFT de 8 puntos en cuatro DFT de 2 puntos

Con lo que tenemos una DFT descompuesta en unidades básica y donde el esquema para una N=8 es del tipo

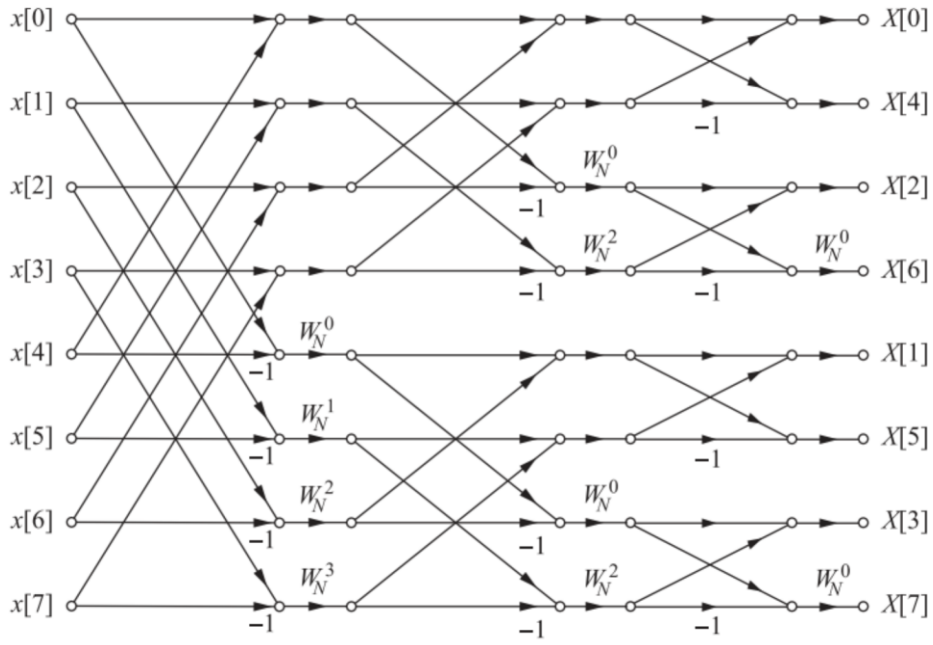


Figura 62. Esquema para un DFT de 8 puntos

Donde a unidad básica de mariposa es de esta forma:

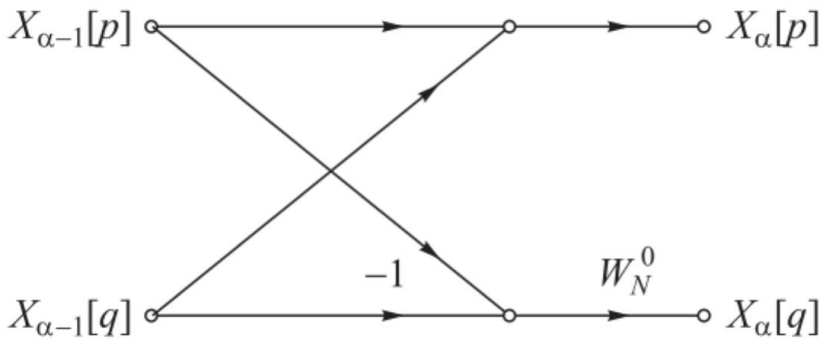


Figura 63. Calculo de mariposa en la última etapa

A continuación se muestran los pasos a seguir que se utilizaron generar el programa con el que se grabó el corpus de dígitos con el DSP.

Generar un proyecto dentro de “Code Composer 5 (CCS5)”:

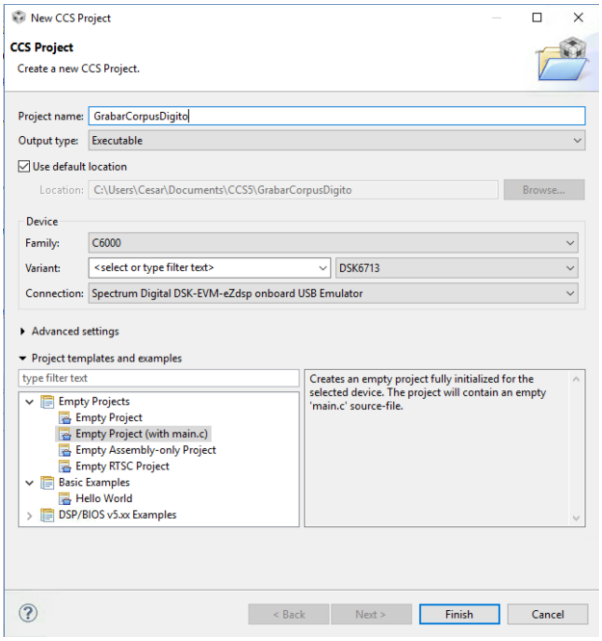


Figura 64. Creación de proyecto en CC5

Modificar parámetros en el menú “Propiedades” del proyecto, de manera que las líneas de compilación y del linker queden de la siguiente manera.

Para la Compilación:

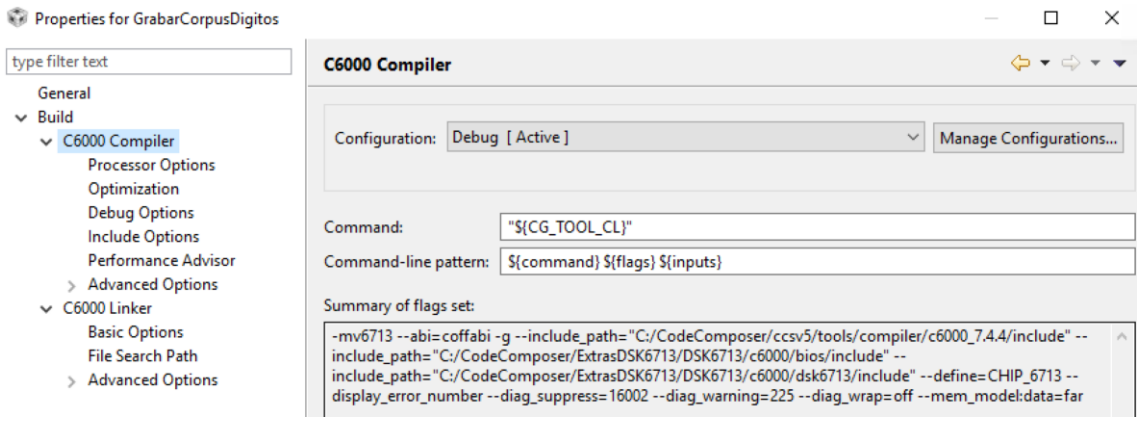


Figura 65. Parámetros para el compilador en CC5

Para el Linker:

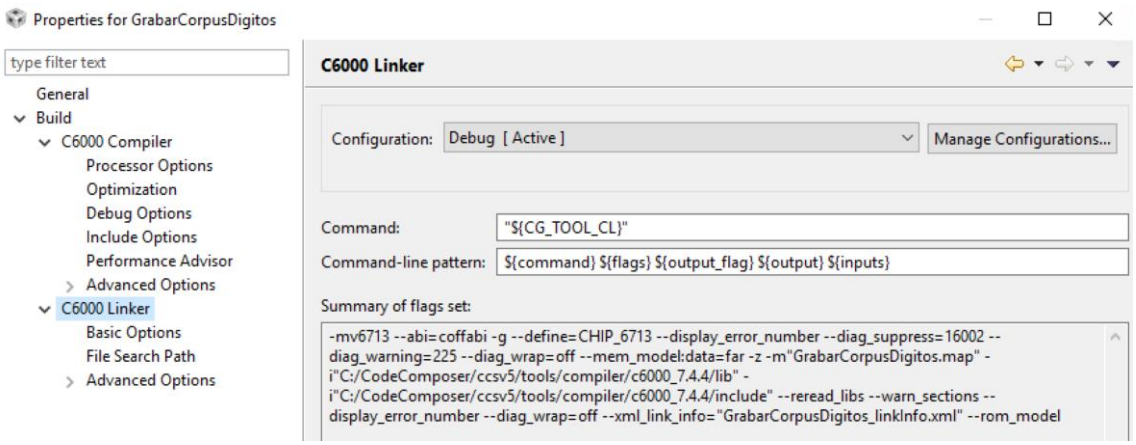


Figura 66. Parámetros para el linker en CCS5

Incluir los siguientes archivos al proyecto

- C6713dsk.cmd : Mapea las regiones de memoria física dentro del proyecto.
- C6713dskinit.c, C6713dskinit.h : Dan soporte a los periféricos de la tarjeta.
- Vectors_intr.asm : Habilita el uso de interrupciones.

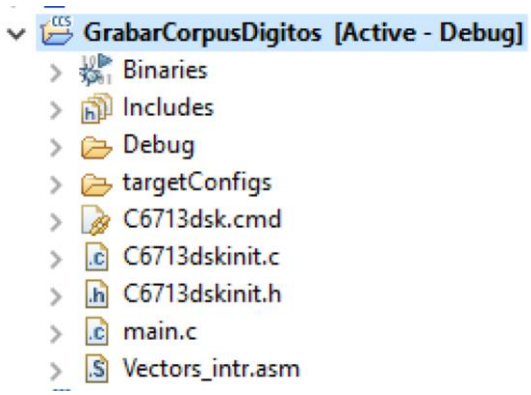


Figura 67. Archivos del proyecto de CCS5 para grabar corpus de dígitos

Reemplazar el código del archivo archivo main.c, con el siguiente código:

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include "C6713dskinit.h"
#include "dsk6713_led.h"
#include "dsk6713_dip.h"

/******* Macros y variables globales *************/
#define DSK6713_AIC23_INPUT_MIC  0x0015
#define DSK6713_AIC23_INPUT_LINE                0x0011
#define FREQ_MUESTREO                           8000
#define TAM_GRABACION                           FREQ_MUESTREO
#define FACTOR                                  1
#define ENERGIA_SILENCIO                        0.1

Uint32 fs               =       DSK6713_AIC23_FREQ_8KHZ;                //Tasa de Muestreo
Uint16 inputsource      =       DSK6713_AIC23_INPUT_MIC;                //Entrada de Audio

typedef struct
{
        char chunk_id[4];
        int chunk_size;
        char format[4];
        char subchunk1_id[4];
        int subchunk1_size;
        short int audio_format;
        short int num_channels;
        int sample_rate;
        int byte_rate;
        short int block_align;
        short int bits_per_sample;
        char subchunk2_id[4];
        int subchunk2_size;
} encabezadoWAV;

encabezadoWAV encabezado, encabezadoAux;
int cont                =       0;
short isGrabando        =       0;      //0:Sin grabar          1:Grabando
short isReproduciendo   =       0;      //0:Sin reproducir      1:Reproduciendo
int bandera             =       1;
int cambioDeDigito      =       0;

FILE *fptr;
short aux;
int numeroDigito                = -1;
int numeroGrabacion     = 21;


#pragma DATA_SECTION(buffer,".senal")
short buffer[TAM_GRABACION];
char digitos[10][7] = { "cero\0",   "uno\0",   "dos\0",   "tres\0", "cuatro\0",
                        "cinco\0", "seis\0",  "siete\0", "ocho\0",  "nueve\0"};


/***************************** Funciones  *******************************/
encabezadoWAV cabeceraDeWAV(int fs, int tamanoDatos, short noCanales, short bitsPorMuestra)
{
        encabezadoWAV header;
        strcpy(header.chunk_id, "RIFF");
        header.chunk_size = tamanoDatos + sizeof(encabezadoWAV)-8;
        strcpy(header.format, "WAVE");
        strcpy(header.subchunk1_id, "fmt ");
        header.subchunk1_size = 16;
        header.audio_format = 1;
        header.num_channels = noCanales;
        header.sample_rate = fs;
        header.byte_rate = (fs * noCanales * bitsPorMuestra)/8;
        header.block_align = (noCanales * bitsPorMuestra)/8;
        header.bits_per_sample = bitsPorMuestra;
        strcpy(header.subchunk2_id, "data");
        header.subchunk2_size = tamanoDatos;

        return header;
}

void grabar(int muestra){
        char nombreArchivo[20];

        DSK6713_LED_off(3);
        cambioDeDigito = 0;
        if( DSK6713_DIP_get(0)==0  || isGrabando ){
            DSK6713_LED_on(0);
            isGrabando = 1;
            if( cont < TAM_GRABACION ){
                        buffer[cont] = muestra;
                        cont++;
            }
            else{
                        puts("Grabando, Espere...");

                        sprintf(nombreArchivo, "%s%d.wav", digitos[numeroDigito], numeroGrabacion++);
                        fptr=fopen(nombreArchivo,"wb");
                        fwrite(&encabezado, sizeof(encabezadoWAV), 1, fptr);
                        fwrite(buffer, sizeof(short), TAM_GRABACION, fptr);
                        fclose(fptr);

                        DSK6713_LED_off(0);
```

```
                                isGrabando = 0;
                                cont = 0;
                                printf ("Grabacion Terminada\tArchivo: %s\n", nombreArchivo);
                        }
                }
                return;
}

short reproducir(){
                short muestra;
                DSK6713_LED_on(1);
                isReproduciendo = 1;
                if( cont < TAM_GRABACION ){
                        muestra = buffer[cont];
                        cont++;
                        return muestra;
                }
                else{
                        DSK6713_LED_off(1);
                        isReproduciendo = 0;
                        cont = 0;
                        return (short)0;
                }
}

void cambiarDigito(){
                if( !cambioDeDigito ){
                        DSK6713_LED_on(3);
                        numeroDigito++;
                        numeroGrabacion = 21;
                        if( numeroDigito > 9 ){
                                bandera = 0;
                        }
                        else{
                                printf("\n/*******************************************************/\n");
                                printf("Cambio de Digito Realizado A : %s\n", digitos[numeroDigito]);
                                printf("/*******************************************************/\n");
                        }
                }
                cambioDeDigito = 1;
}

/***************************** Rutina de interrupcion  *****************************/
interrupt void c_int11() //interrupt service routine
{
                short muestraSalida;
                aux = input_left_sample();
                muestraSalida = aux;

                if( DSK6713_DIP_get(0)==0  || isGrabando )
                        grabar(aux);
                else if( DSK6713_DIP_get(1)==0 || isReproduciendo )
                        muestraSalida = reproducir();
                else if( DSK6713_DIP_get(3)==0 )
                        cambiarDigito();

                output_left_sample(muestraSalida);
                return;
}


/******************************* Programa Principal  *******************************/
void main()
{
                encabezado = cabeceraDeWAV(FREQ_MUESTREO, TAM_GRABACION*sizeof(short), 1, 16);
                DSK6713_init();
                DSK6713_LED_init();
                DSK6713_DIP_init();
                cambiarDigito();
                comm_intr();                                                                    //Inicializa
DSK,codec,McBSP
                while(bandera);
                printf("Programa terminado\n");
}
```