UNIVERSIDAD CARLOS III DE MADRD

ESCUELA POLITÉCNICA SUPERIOR

INGENIERÍA DE TELECOMUNICACIONES

PROYECTO FINAL DE CARRERA

PUESTA EN MARCHA DE UN ENTORNO DE EXPERIMENTACIÓN PARA RECONOCIMIENTO DE HABLA EN CABINAS DE AVIÓN

AUTOR: JAVIER DÍEZ BERMÚDEZ DE CASTRO

TUTORA: CARMEN PELÁEZ MORENO

27 de Mayo de 2010

Agradecimientos

En primer lugar he de agradecer a mi tutora, Carmen Peláez, por todo su apoyo y dedicación sin los cuales no hubiera sido posible la realización de éste proyecto. Tengo que agradecer también la atención y ayuda prestadas por David Gelbart en la utilización de su código para la implementación de la substracción espectral. No puedo olvidarme tampoco de agradecer al Ministerio de Educación, Política Social y Deporte la financiación aportada durante la realización de este proyecto.

Quiero agradecer también a mi familia y amigos por haberme apoyado durante todo este tiempo y por lo que en el fondo los considero partícipes de la elaboración de este proyecto.

Resumen

Este proyecto afronta el tema del reconocimiento automático de habla en el escenario de las cabinas de avión. Se trata de un escenario en el cual nos enfrentamos a problemas tales como distintos tipos de ruido (ruido del propio avión, ruido conversacional o ruido de reverberación) así como variabilidad en la lengua nativa de los propios hablantes. En este caso, hemos puesto especial énfasis en buscar soluciones al problema del ruido por reverberación.

Para realizar la investigación se ha realizado la puesta en marcha de un entorno de experimentación haciendo uso de la base de datos HIWIRE[1] sobre el que hemos probado técnicas básicas orientadas a la mejora del reconocimiento en este entorno., en particular hemos probado algunas técnicas simples como la normalización en media y varianza y otras mas complejas como la substracción espectral la cual combinaremos también con un VAD[1].

[1] Voice Activity Detector

Abstract

This project deals with the task of automatic speech recognition (ASR) in aeronautic environments such as airplane's cockpits. Cockpits are scenarios in which we face problems like different kinds of noise (engine noise, conversational noise or reverberant noise) or a high variability due to non-native speakers. In our case we have made an special effort in searching solutions to overcome the problems of reverberant noise

In order to carry on with this investigation we have created an environment of investigation based on the HIWIRE database over which we have tested basic techniques focused on improving ASR performance. More specifically, we have tested basic techniques like mean and variance normalization and some others more complex tike spectral substraction later on combined with VAD[2] .

[2] Voice Ativity Detection

Índice General

Lista de figuras

Lista de tablas

1. Introducción

Hoy en día las tecnologías del habla conforman un campo muy interesante por las posibilidades y soluciones a muy variados tipos de problemas que potencialmente ofrece. El máximo exponente de este tipo de tecnologías sería una interfaz conversacional con la que se pudiera interactuar casi como si de otro interlocutor humano se tratase.

Llegar a ese tipo de solución es algo muy complejo que implica la combinación de muchas tecnologías como pudieran ser la codificación de voz, el reconocimiento de habla, conversión texto habla etc. En este proyecto no vamos a ir tan lejos y vamos a centrarnos en el reconocimiento de habla.

El reconocimiento de habla tiene como objetivo el ser capaces de transcribir la información hablada de un interlocutor. Conseguir ese objetivo es la llave a un número muy elevado de servicios en diferentes campos que en general tienen como meta la automatización de tareas que de otro modo sería mas lentas y/o costosas.

En nuestro caso particular vamos a focalizar nuestra atención hacia la industria aeronáutica en el entorno de las cabinas de avión. La idea es conseguir que el avión sea capaz de reconocer un número determinado de comandos destinados al dialogo entre la tripulación de la aeronave y los controladores de tráfico aéreo de manera que se substituya el tradicional sistema de mensajes escritos por uno en el cual la información sea introducida únicamente por voz.

No obstante, como en todo sistema de reconocimiento nos encontramos diversos problemas comunes a este tipo de sistemas que dificultan su funcionamiento en la práctica como pueden ser las variaciones acústicas entre diferentes hablantes (o incluso para un mismo hablante), recorte o supresión de unidades acústicas, coarticulación etc. Pero seguramente lo más nocivo es el ruido. Existen numerosas fuentes de ruido que perjudican el sistema, nosotros en este proyecto intentamos centrarnos principalmente en el ruido por reverberación y a lo largo de esta memoria haremos un recorrido por las diversas técnicas que hemos evaluado con el fin de combatirlo así como de otro tipo de soluciones que, aunque no han llegado a ser probadas en la práctica en este proyecto, si hemos querido mencionar para dar un poco de perspectiva sobre algunas de las soluciones en las que se trabaja hoy día.

Antes de llegar a todo ello conviene comprender los conceptos fundamentales sobre los que se asienta el reconocimiento de habla, de esta manera en el capítulo 2 se comienza introduciendo los conceptos básicos relacionados con el reconocimiento de habla para posteriormente pasar a comentar la problemática concreta estudiada en el proyecto así como las técnicas con las que se trabajó durante el mismo. Posteriormente en ese mismo capítulo se realiza una pequeña revisión de algunas de las principales técnicas en las que hoy día parece que se están centrando más los esfuerzos de cara a combatir la problemática estudiada en este proyecto. Termina el capítulo con algunos ejemplos de aplicación práctica de sistemas de reconocimiento de habla en la industria aeronáutica.

El tercer capítulo trata de la fase de experimentación del proyecto, se comienza describiendo la base de datos en la cual hemos basado el trabajo y posteriormente se describen las distintas fases y técnicas involucradas en los experimentos incluyendo la funcionalidad del distinto software relacionado en cada momento así como los principales problemas que se nos presentaron en la realización práctica de los experimentos. En la última parte de este capítulo se presentan los distintos resultados obtenidos en forma de tablas y gráficas.

En el cuarto capítulo se describen las principales conclusiones que pudimos obtener tras la realización del proyecto así como las principales líneas de investigación que quedan abiertas tras la finalización del mismo.

La última parte de esta memoria va destinada a los apéndices, en el apéndice A se describe el software HTK en el cual hemos basado nuestro sistema de reconocimiento, en él se comenta la estructura del mismo así como las fases principales que tienen lugar durante el reconocimiento así como las principales herramientas relacionadas durante las mismas.

En el apéndice B se habla del paquete SPRACHcore del cual hicimos uso en alguna fase del proyecto así como de la toolbox VOICEBOX que fue necesaria en la realización de los experimentos del proyecto.

Para terminar, se incluye al final de la memoria el presupuesto final del proyecto.

2. Fundamentos del reconocimiento de habla

En las siguientes secciones expondremos algunos de los principios básicos relacionados con el reconocimiento de habla.

2.1. Producción de la señal de voz

Empezaremos conociendo la composición del aparato fonador encargado de generar el habla humana. Su esquema es el siguiente [2]:

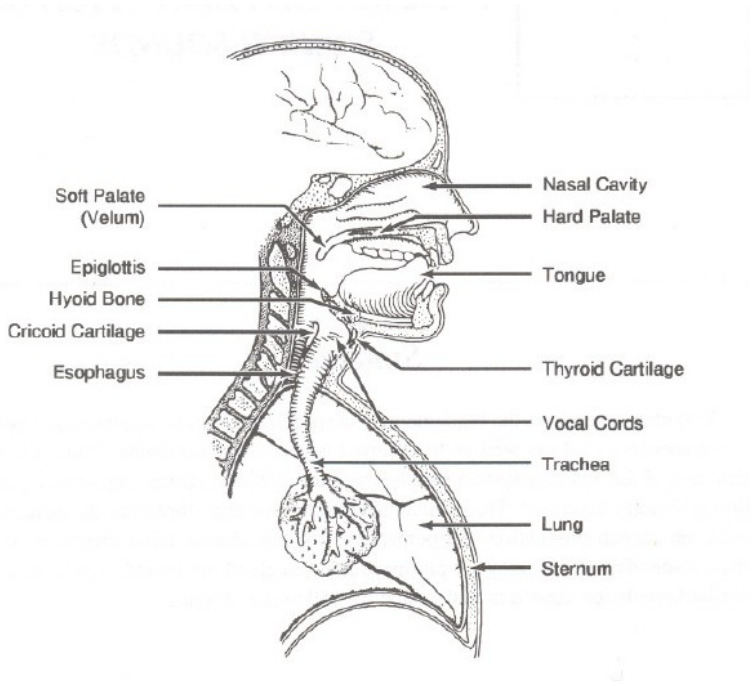

Figura 1: Esquema del aparato fonador

El aire expelido por los pulmones atraviesa dos piezas de ligamentos y músculos simétricos situados en la laringe: las cuerdas vocales.

El tracto vocal es básicamente un tubo acústico de corte transversal no uniforme. Consiste en dos partes:

- Tracto oral, el cual abarca desde los labios a las cuerdas vocales. Se trata de un tracto resonador con varias posibles frecuencias de resonancia. Estas frecuencias son las que conoceremos como formantes.
- Tracto nasal, que controla la nasalidad de los sonidos.

El tracto vocal puede por tanto modelarse mediante un sistema con una determinada función de transferencia. La longitud del tracto vocal de cada hablante

tiene influencia en las labores de reconocimiento de habla y es por ello que una de las técnicas estudiadas en el proyecto fue la normalización del tracto vocal.

Una vez descrito el tracto vocal, falta cuales son sus posibles excitaciones de ese sistema.

Los sonidos producidos en el habla son básicamente de dos tipos, sordos y sonoros. Cada uno de ellos representa un tipo de excitación distinta. Los sonidos sonoros son producidos por la vibración de las cuerdas vocales las cuales con capaces de generar frecuencias de entre 80 Hz y 300Hz dependiendo de muchos factores (edad, sexo, estrés, etc). Por el contrario, en la generación de los sonidos sordos, las cuerdas vocales están abiertas y permiten el paso libre de aire.

De esta manera el proceso de producción de habla en el caso de los sonidos sonoros puede describirse muy bien en el siguiente gráfico:

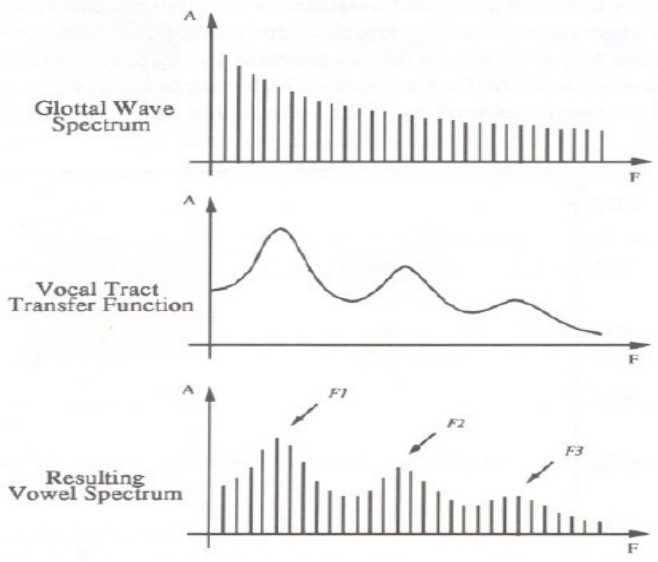

Figura 2: Espectro de la señal de voz[2]

La primera figura representa la secuencia de excitación generada por las cuerdas vocales. La segunda es la función de transferencia del tracto vocal, de manera que el espectro de la señal de voz resultante es el que muestras la última figura. En esta última podemos ver como existen unos determinados armónicos que salen reforzados y que son particulares de cada sonido, son los formantes del sonido.

De esta manera lo que nos interesa obtener de la señal de voz a la hora de llevar a cabo tareas de reconocimiento es la envolvente de dicho espectro.

2.2. Extracción de características

Para poder trabajar con la señal de voz de manera adecuada es necesario parametrizarla. Toda parametrización de voz tiene el mismo fin básico: caracterizar la envolvente espectral de la señal de voz ya que es la que contiene información relevante. Existen diversos tipos de parametrización pero en nuestro caso usaremos la parametrización MFCC[3].

Para llegar al proceso de obtención de los coeficientes MFCC, vamos a explicar primero los principios básicos del análisis cepstral. Como se ha mencionado la voz se compone de dos partes bien diferenciadas, una es la excitación que puede ser sorda o sonora según el tipo de sonido y la otra es la respuesta al impulso del tracto vocal. La señal de voz se obtiene como la convolución de la excitación con la respuesta al impulso del tracto vocal:

$$s(n) = x(n) * h(n) \quad (1)$$

Para el reconocimiento de habla nos interesa quedarnos con la parte que corresponde al tracto vocal. Este proceso de separación a menudo se conoce como deconvolución y es importante notar que no existe manera de realizarlo de manera lineal. Vease la ecuación (1)

Con estas premisas surge el análisis cepstral. La idea de partida del análisis es que dado que ambas señales se convolucionan en el dominio del tiempo, al pasar al dominio de la frecuencia aparecerán multiplicadas:

$$x(n) * h(n) \overset{TF}{\rightarrow} X(\omega)H(\omega) \quad (2)$$

Una vez que nos encontramos en el dominio de la frecuencia realizamos una transformación no lineal como es el logaritmo que nos permitirá separar ambas componentes. Tomando el logaritmo obtenemos:

$$\log | X(\omega)H(\omega) | = \log | X(\omega) | + \log | H(\omega) | = C_x(k) + C_h(k) \quad (3)$$

Si ahora realizamos la transformada inversa conseguimos:

$$\mathrm{DFT}^{-1}\{C_x(k) + C_h(k)\} = c_x(n) + c_k(n) \quad (4)$$

Con lo que hemos conseguido realizar la deconvolución deseada. Ahora solo nos resta separar ambas componentes para quedarnos únicamente con la señal que

[3] Mel Frequency Cepstral Coefficients

corresponde al tracto vocal que es la que contiene información relevante para las tareas de reconocimiento de habla. El proceso de separación se realiza mediante lo que se conoce como liftering que no es mas que un proceso de filtrado llevado al dominio cepstral.

Una vez que hemos comprendido las ideas básicas del análisis cepstral vamos a explicar con más detalle como se obtienen los parámetros MFCC. La idea principal es que ponderamos el espectro mediante un banco de filtros triangulares cuyas longitudes se diseñan inspirándonos en las bandas críticas de percepción del oído humano. El esquema de bloques para la obtención de los coeficientes sería el siguiente:

Figura 3: Esquema obtención parámetros MFCC

La función del filtro de preénfasis es enfatizar las frecuencias altas de manera que se compense el fuerte carácter paso bajo de la excitación glotal. Con esto se reduce el rango dinámico del espectro de manera que se mejoran las características del procesado numérico posterior.

De esta manera el proceso de obtención de los coeficientes sería el siguiente:

1. Cálculo del espectro.

$$\tilde{x}(k) = \sum_{n=0}^{N_W-1} x(n)W(n)\exp(-j\frac{2\pi k}{N_W}) \qquad (5)$$

Siendo la señal de entrada, la ventana elegida (típicamente suele escogerse una ventana Hamming) y la longitud de la ventana.

2. Cálculo de la energía en cada banda.

Realizamos el cálculo de la energía de la siguiente manera:

$$E_j = \sum_{j=0}^{J-1} \varphi_j(k) X_k, \qquad 0 \le j \le J \qquad (6)$$

Siendo $X_k = |\hat{X}(k)|^2 \qquad 0 \le k \le K \qquad \left(K = \frac{N_w}{2}\right)$ y donde J es el número de filtros triangulares que conforman el banco de filtros. Una cualidad importante del banco de filtros es que cumple que:

$$\sum_{k=0}^{K-1} \varphi_j(k) = 1, \qquad \forall j \tag{7}$$

La forma de este banco de filtros es la siguiente:

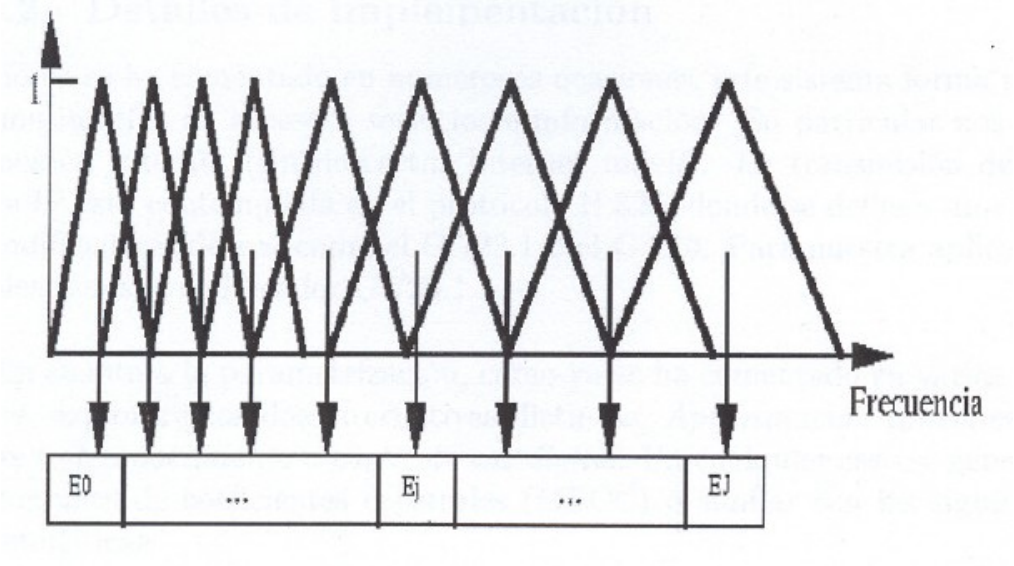

Figura 4: Banco de filtros

El ancho de cada filtro se fija en función de las bandas críticas del oído humano. Así podemos ver que para las altas frecuencias en las cuales el oído tiene mayores dificultades para diferenciar frecuencias cercanas, el ancho del filtro es mayor.

3. Cálculo de los coeficientes MFCC

$$c_m = \beta_c \sum_{j=0}^{J-1} \cos\left(m \frac{\pi}{J}(j + 0.5)\right) \log(E_j) \tag{8}$$

β_c es un factor de amplificación que acomoda el rango dinámico de los coeficientes

Esta parametrización consigue una buena estimación de la envolvente espectral de manera local, no obstante una de las principales características de la señal de voz es su comportamiento dinámico. Es por eso bastante común realizar estimaciones de la derivada local con el fin de captar ese comportamiento dinámico.

Una de las manera más comunes de llevar esto a cabo esto es mediante cepstrum delta y constituye una aproximación suavizada de la derivada local mas que a una simple diferencia entre los coeficientes de tramas vecinas. Estos parámetros pueden expresarse como:

$$\Delta C_i(n) = \frac{\sum_{k=-N}^{N} k C_i(n+k)}{\sum_{k=-N}^{N} k^2} \quad (9)$$

También es posible realizar la segunda derivada, con lo cual obtendríamos lo que se conoce como cepstrum delta-delta. Es importante resaltar que en la práctica estos parámetros no suelen usarse de manera independiente ya que se pierden algunas de las características de la envolvente espectral a largo plazo y los resultados en reconocimiento no son lo suficientemente buenos. Por tanto estos parámetros suelen ser un añadido a, por ejemplo, los MFCC y sirven tal como se ha dicho para poder caracterizar el comportamiento dinámico de la señal de voz. Estas parametrizaciones también se conocen con el nombre de velocidad y aceleración.

2.3. Modelos de lenguaje

Una necesidad básica de un reconocedor de habla que trabaje con un vocabulario extenso es poseer conocimiento del lenguaje en forma de "modelo de lenguaje" [3]. El objetivo básico del modelo de lenguaje es ser capaz de estimar la probabilidad de una secuencia de palabras W dentro del proceso de reconocimiento.

Sabemos que el lenguaje humano posee cierta estructura que hace que las palabras dentro de una frase concreta de alguna forma estén correlacionadas entre si. Esta dependencia ayuda enormemente a la labor de reconocimiento.

El escenario es el siguiente, para una secuencia de palabras $W = w_1 w_2 w_3 ..... w_Q$ pretendemos encontrar $P(W)$. La manera razonable de calcular dicha probabilidad sería la siguiente:

$$P(W) = P(w_1 w_2 w_3 ... w_Q) = P(w_1) P(w_2 \mid w_1) P(w_3 \mid w_1 w_2) ..... P(w_Q \mid w_1 w_2 .... w_{(} \quad (10)$$

Sin embargo es esencialmente imposible estimar todas esas probabilidades condicionadas para todas las palabras en todas las secuencias de palabras de un determinado lenguaje. Por eso en la práctica suelen usarse modelos basados en N-gramas en la cual las probabilidades condicionadas se limitan a las N-1 palabras anteriores, i.e. $P(w_j \mid w_1 w_2 .... w_{j-1}) \approx P(w_j \mid w_{j-N+1} ..... w_{j-1})$.

En la práctica el valor de N no suele ser muy elevado (2 o como mucho 3) e incluso es frecuente encontrar modelos que únicamente especifican si una determinada pareja de palabras es válida o no, i.e,

$$P(w_j \mid w_k) = \begin{cases} 1 & w_k w_j \quad es \quad valido \\ 0 & w_k w_j \quad no \quad es \quad valido \end{cases} \quad (11)$$

Como puede deducirse de lo anterior, en realidad nunca se tiene un conjunto de entrenamiento lo suficientemente extenso como para aprender todas las relaciones

estadísticas entre las distintas palabras en una comunicación oral. Es por eso que suelen usarse técnicas como las siguientes:

Backoff smoothing: En este caso el número de palabras que conformarán la unidad de la cual queremos calcular su probabilidad dependerá del número de muestras que tengamos de esa unidad en el conjunto de entrenamiento. Por ejemplo, si no disponemos de suficientes muestras para calcular la probabilidad de un determinado trigrama, intentaremos separarlo y usar bigramas. Si siguiéramos sin disponer muestras suficientes aun en ese caso, pasaríamos al uso de unigramas (frecuencia relativa de la palabra) .

Deleted interpolation: Esta técnica divide el conjunto de entrenamiento en M partes y para cada una de ellas estima los parámetros deseados para después combinar los M resultados mediante un conjunto de pesos. Para el caso de M=2 por ejemplo, el objetivo sería encontrar el ε optimo para combinar los parámetros de la siguiente manera:

$$\Theta_T = \varepsilon\Theta_1 + (1-\varepsilon)\Theta_2 \qquad \textbf{(12)}$$

Otra particularidad que debe tenerse en cuenta es que en numerosas ocasiones la capacidad de predicción es mayor en palabras que no son consecutivas. En el ejemplo “comer un gran número de hamburguesas” la palabra “comer” es un mejor predictor de “hamburguesas” que de “un”. Si pretendemos modelar estas relaciones de larga distancia incrementando el valor de N, nos encontramos con el problema de escalabilidad descrito anteriormente. Los N-gramas carecen de noción de estructura sintáctica y semántica pero podemos mitigar ese efecto si recurrimos a texto escrito para entrenar los modelos de lenguaje. Sin embargo esto conlleva un nuevo problema: el lenguaje escrito es esencialmente distinto al lenguaje hablado.

2.4. Principios generales de los HMM

Los sistemas de reconocimiento de habla generalmente asumen que la señal de voz es una realización de un mensaje codificado como una secuencia de símbolos tal y como se ve en la siguiente figura [4]:

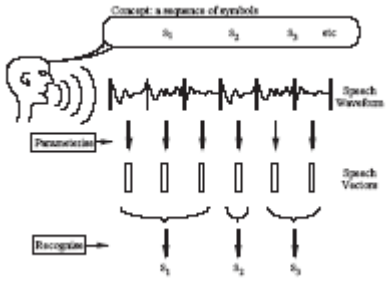

Figura 5: De la voz a la secuencia de símbolos

La señal de voz es convertida a una secuencia de vectores de parámetros equiespaciados en el tiempo (10ms en el caso de nuestros experimentos) y de una duración fija en la cual la señal de voz puede considerarse estacionaria (25ms en nuestro caso). El objetivo del reconocedor es transformar esa secuencia de vectores de parámetros en una secuencia de símbolos, lo cual no es trivial ya que la relación de mapeo no es uno a uno puesto que distintas pronunciaciones de un mismo fonema corresponden al mismo símbolo. Hay muchos factores influyentes en este sentido como las diferencias entre hablantes, estado de ánimo del propio hablante, el propio entorno etc. Además a menudo no es sencillo delimitar con absoluta precisión los límites entre símbolos.

Antes de considerar el escenario de reconocimiento de habla continuo, vamos a estudiar el caso de reconocimiento de palabras aisladas. Supongamos que cada palabra se representa como una secuencia de vectores de parámetros, denominemos esa secuencia como O y la definimos de la siguiente manera:

$$O = o_1, o_2, \ldots, o_T \qquad (13)$$

Donde o_t representa al vector observado en el instante t. En ese caso el problema de encontrar la palabra observada es equivalente a calcular:

$$\arg\max_i \{P(w_i|O)\} \qquad (14)$$

Donde w_i es la i-ésima palabra de todo el vocabulario. Esta probabilidad no puede computarse directamente pero podemos recurrir al teorema de Bayes:

$$P(w_i|O) = \frac{P(O|w_i)P(w_i)}{P(O)} \qquad (15)$$

De esta manera, conocidas a priori las probabilidades $P(w_i)$la palabra pronunciada más probable depende solo de la verosimilitud $P(O \mid w_i)$. Dada la definición de O, la estimación directa de la probabilidad condicionada conjunta $P(o_1, o_2 \ldots \mid w_i)$ a partir de ejemplos de palabras habladas es irrealizable en la práctica. Y es ahí donde resultan útiles los modelos de Markov ya que se substituye el problema de calcular la probabilidad condicionada antes mencionada por el más simple de estimar los parámetros del modelo de Markov.

En sistemas de reconocimiento de habla basados en Modelos Ocultos de Markov (HMMs) se asume que las observaciones corresponden a realizaciones generadas por un modelo de Markov como el que puede verse a continuación:

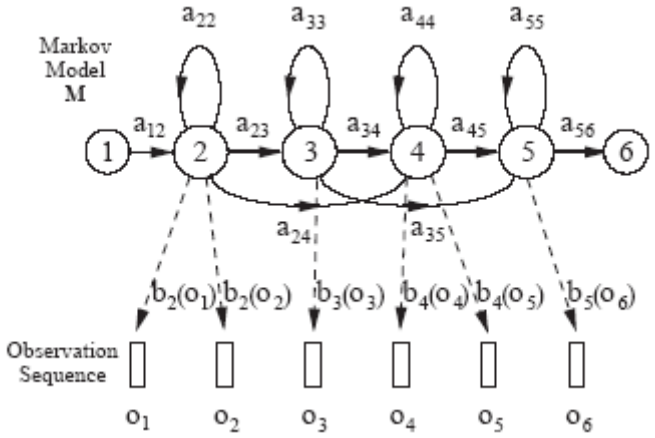

Figura 6: Modelo de Markov[4]

Un modelo de Markov es una máquina de estados finitos que cambia de estado en cada unidad de tiempo ty cada vez que se entra en el estado j se genera un vector $o_t$con densidad de probabilidad $b_j(o_t)$. Así mimo la transición del estado $i$al estado $j$viene gobernada por la probabilidad a_{ij}.

De esta manera la probabilidad conjunta de que el modelo M genere la secuencia Opuede calcularse simplemente como el producto de las probabilidades de transición y las probabilidades de salida. Por ejemplo para la secuencia Xde la figura anterior tendríamos:

$$P(O,X|M) = a_{12}b_2(o_1)a_{22}b_2(o_2)a_{23}b_3(o_3)\ldots \quad (16)$$

En la práctica solo la observación O es conocida mientras que la secuencia de estados Xpermanece oculta. Es la razón por la que se les conoce como Modelos ocultos de Markov.

Dado que X es desconocido, la verosimilitud deseada se calcula sumando a través de todas las posibles secuencias de estados, es decir:

$$P(O|M) = \sum_{X} a_{x(0)x(1)} \prod_{t=1}^{T} b_{x(t)}(o_t) a_{x(t)x(t+1)} \quad (17)$$

A pesar de que esta expresión no es manejable en la práctica de manera directa, existen procedimientos recursivos que permiten calcularla de manera bastante eficiente.

Por tanto, si la ecuación (14) puede calcularse, el problema de reconocimiento esta resuelto. Dado un conjunto de modelos $M_i$correspondiente a palabras w_i, la ecuación (14) puede resolverse usando (15) y asumiendo que:

$$P(O|w_i) = P(O|M_i). \quad (18)$$

Para todo esto suponemos que los parámetros $\{a_{ij}\}$y $\{b_j o_t\}$son conocidos para cada modelo M_iy esto se consigue mediante el entrenamiento de dichos modelos. Podemos resumir la mecánica del reconocimiento de palabras aisladas mediante la siguiente figura, en ella el vocabulario consta de tres palabras (“one”, “two” y “three”) para las cuales disponemos de tres ejemplos de cada una para la fase de entrenamiento:

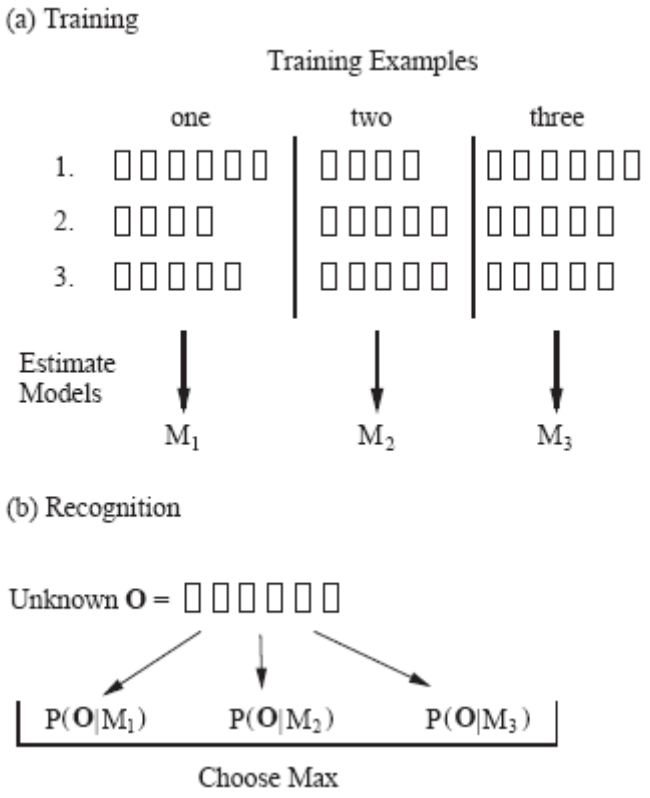

Figura 7: Entrenamiento del reconocedor[4]

La forma de las distribuciones $\{b_j o_t\}$en HTK puede ser discreta pero en nuestro caso usamos funciones de densidad continuas representadas por mezclas de gaussianas siendo la fórmula para el cáculo de $b_j o_t$la siguiente:

$$b_j(o_t) = \prod_{s=1}^{S} \left[\sum_{m=1}^{M_s} c_{jsm} \mathcal{N}(o_{st}; \mu_{jsm}, \Sigma_{jsm}) \right]^{\gamma_s} \quad (19)$$

Donde $M_x$es el número de componentes de la mezcla en el stream s, c_{jsm}es el peso del m-ésimo componente y $N; \mu, \Sigma$es una gaussiana multivariable con vector de medias μy matriz de covarianza Σ, es decir:

$$\mathcal{N}(o; \mu, \Sigma) = \frac{1}{\sqrt{(2\pi)^n |\Sigma|}} e^{-\frac{1}{2}(o-\mu)' \Sigma^{-1} (o-\mu)} \quad (20)$$

Donde $n$corresponde a la dimensión de o. El exponente $\gamma_s$se usa para dar un mayor énfasis a un stream en particular, no obstante no existen herramientas en HTK que permitan estimar ese parámetro, por lo que únicamente puede modificarse de forma manual.

2.4.1. Re-Estimación Baum-Welch

Para determinar los parámetros de los HMMs es necesario realizar una primera aproximación a su valor real y posteriormente ir refinándolos hasta llegar a una versión mas precisa (en el sentido de máxima verosimilitud) de los mismos. Para ello se utiliza la re-estimación Baum-Welch.

El objetivo es estimar las medias y varianzas de un HMM para el cual la salida de cada estado se modela como una distribución gaussiana de la forma:

$$b_j(o_t) = \frac{1}{\sqrt{(2\pi)^n |\Sigma_j|}} e^{-\frac{1}{2}(o_t-\mu_j)' \Sigma_j^{-1} (o_t-\mu_j)} \quad (21)$$

Si solo hubiera un estado $j$en el HMM la estimación de media y varianza correspondería con las medias:

$$\hat{\mu}_j = \frac{1}{T} \sum_{t=1}^{T} o_t \quad (22)$$

$$\hat{\Sigma}_j = \frac{1}{T} \sum_{t=1}^{T} (o_t - \mu_j)(o_t - \mu_j)' \quad (23)$$

Pero en la práctica disponemos de múltiples estados por lo que no hay una asignación directa entre vectores de observaciones y estados individuales puesto que la secuencia de los mismo es desconocida. De manera que las expresiones anteriores se modifican de la siguiente manera:

$$\hat{\mu}_j = \frac{\sum_{t=1}^{T} L_j(t) o_t}{\sum_{t=1}^{T} L_j(t)} \quad (24)$$

$$\hat{\boldsymbol{\Sigma}}_j = \frac{\sum_{t=1}^{T} L_j(t)(\boldsymbol{o}_t - \boldsymbol{\mu}_j)(\boldsymbol{o}_t - \boldsymbol{\mu}_j)'}{\sum_{t=1}^{T} L_j(t)} \quad (25)$$

Donde $L_j t$es un elemento de ponderación que denota la probabilidad de estar en el estado $j$en el instante t. Estas últimas expresiones son las que se utilizan para realizar la re-estimación Baum-Welch.

Ahora solo necesitamos calcular $L_j t$ para lo cual se utiliza el algoritmo *Forward-Backward*. Sea la probabilidad[4] *forward* o *de ida* definida de la siguiente manera para un modelo Mcon Nestados:

$$\alpha_j(t) = P(\boldsymbol{o}_1, \ldots, \boldsymbol{o}_t, x(t) = j|M). \quad (26)$$

Es decir, $\alpha_j t$ es la probabilidad conjunta de observar los primeros t vectores de habla y encontrarnos en le estado j en el instante t y es posible calcularla mediante la siguiente fórmula recursiva:

$$\alpha_j(t) = \left[\sum_{i=2}^{N-1} \alpha_i(t-1)a_{ij}\right] b_j(\boldsymbol{o}_t). \quad (27)$$

Siendo las condiciones iniciales:

$$\alpha_1(1) = 1 \quad (28)$$

$$\alpha_j(1) = a_{1j} b_j(\boldsymbol{o}_1) \quad (29)$$

Para $1 < j < N$ tenemos:

$$\alpha_N(T) = \sum_{i=2}^{N-1} \alpha_i(T)a_{iN}. \quad (30)$$

Y como de la definición de $\alpha_j t$ tenemos que:

$$P(\boldsymbol{O}|M) = \alpha_N(T). \quad (31)$$

el cálculo de la probabilidad *forward* nos permite obtener la verosimilitud $P(O \mid M)PO \mid M$.

[4] Dado que las distribuciones de salida son densidades, no son realmente probabilidades, simplemente es una convención a la hora de denominarlas.

La probabilidad *backward* o *hacia atrás* puede calcularse mediante la siguiente fórmula recursiva:

$$\beta_i(t) = \sum_{j=2}^{N-1} a_{ij} b_j(o_{t+1}) \beta_j(t+1) \quad (32)$$

Con la condición inicial dada por:

$$\beta_i(T) = a_{iN} \quad (33)$$

Y para $1 < j < N$:

$$\beta_1(1) = \sum_{j=2}^{N-1} a_{1j} b_j(o_1) \beta_j(1). \quad (34)$$

Mientras que la probabilidad *forward* es una probabilidad conjunta, la probabilidad *backward* es una probabilidad condicionada. De esta manera podemos definir la probabilidad de ocupación de un estado mediante el producto de las dos:

$$\alpha_j(t)\beta_j(t) = P(\boldsymbol{O}, x(t) = j|M). \quad (35)$$

Y de aquí obtenemos:

$$\begin{aligned} L_j(t) &= P(x(t) = j|\boldsymbol{O}, M) \\ &= \frac{P(\boldsymbol{O}, x(t) = j|M)}{P(\boldsymbol{O}|M)} \\ &= \frac{1}{P}\alpha_j(t)\beta_j(t) \end{aligned} \quad (36)$$

Donde $P = PO \mid M$.

Ahora que tenemos toda la formulación, los pasos a seguir para realizar la re-estimación son lo siguiente:

- Para cada vector que requiera ser re-estimado reservamos espacio tanto para el numerador como para el numerador de las ecuaciones (22) y (23). Los denominaremos acumuladores.
- Calcular las probabilidades *forward* y *backward* para todos los estados j e instantes t.
- Para cada estado je instante t, usar la probabilidad $L_j t$y el vector observado en ese instante $o_t$para actualizar el valor de los acumuladores para ese estado.
- Usar el valor final de los acumuladores para calcular los nuevos valores de los parámetros.

- Si el valor de $P = O \mid M$para esta iteración no es mayor que el valor de la iteración previa, detener el proceso. En caso contrario, repetir todo el proceso anterior usando los nuevos parámetros re-estimados.

2.4.2. Reconocimiento: Algoritmo de Viterbi

Para el reconocimiento lo preferible es usar un sistema de máxima verosimilitud de las secuencias ya que así es mas sencillo generalizar para el caso de reconocimiento continuo de habla. Dicha verosimilitud se calcula usando el mismo algoritmo que el usado para el cálculo de las probabilidades *forward* y *backward* excepto que las operaciones de suma se substituyen por operaciones de argumento máximo.

Para un modelo M, $\varphi_j t$ representa la máxima verosimilitud de observar los vectores o_1 al o_t y estar en el estado j en el instante t la cual puede calcularse de manera recursiva a partir de la siguiente expresión:

$$\phi_j(t) = \max_i \{\phi_i(t-1)a_{ij}\}\, b_j(\boldsymbol{o}_t). \qquad (37)$$

Siendo

$$\phi_1(1) = 1 \qquad (38)$$

$$\phi_j(1) = a_{1j} b_j(\boldsymbol{o}_1) \qquad (39)$$

Para $1 < j < N$ la máxima verosimilitud $PO \mid M$viene dada por:

$$\phi_N(T) = \max_i \{\phi_i(T)a_{iN}\} \qquad (40)$$

Por cuestiones prácticas en vez de calcular las verosimilitudes de manera directa, se calculan las log-verosimilitudes. De esta manera la ecuación (37) se transforma en:

$$\psi_j(t) = \max_i \{\psi_i(t-1) + log(a_{ij})\} + log(b_j(\boldsymbol{o}_t)). \qquad (41)$$

Esta recursión representa la base del algoritmo de Viterbi. Como puede verse en la Figura 8 el algoritmo tiene como objetivo encontrar el mejor camino a través de una matriz en la que la dimensión vertical representa los estados del HMM mientras que la horizontal representa las tramas de habla, es decir el tiempo. Cada punto de la figura representa la log-probabilidad de observar dicha trama en dicho tiempo y cada segmento entre dos puntos representa la log-probabilidad de transición. La log-probabilidad de cualquier camino se calcula como la suma de las log-probabilidades de de transición y de salida a lo largo de todo el recorrido.

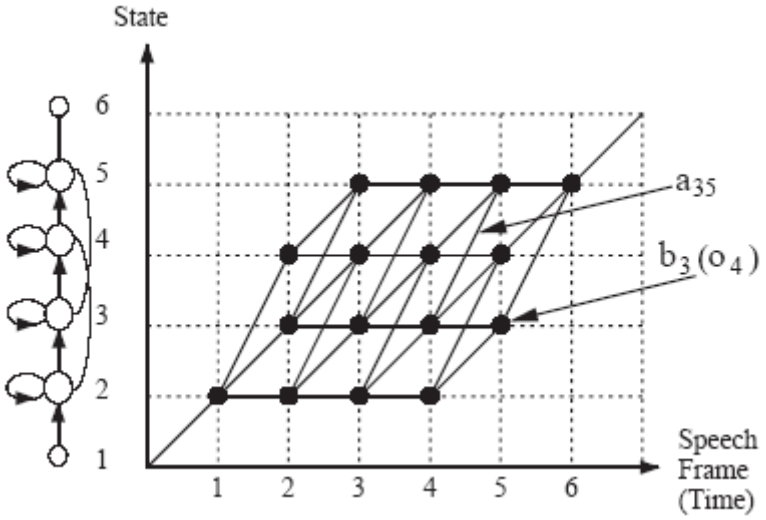

Figura 8: Viterbi[4]

2.4.3. Reconocimiento Continuo de Habla

Hasta ahora hemos hablado de reconocimiento de palabras aisladas, pero lo que realmente nos interesa es el reconocimiento continuo de ellas para poder trabajar con frases enteras.

En la práctica este reconocimiento continuo se lleva a cabo concatenando los diferentes HMM de cada símbolo (palabras o fonemas). Sin embargo existen algunos problemas prácticos como la delimitación de estos símbolos ya que en principio dichos límites no son conocidos. En la fase de entrenamiento es posible delimitar dichos símbolos de manera que el entrenamiento se basaría en palabras aisladas tal y como se ha explicado anteriormente, aunque esta práctica no es excesivamente recomendable ya que puede dar lugar a modelos defectuosamente estimados. De hecho, en el software que nosotros usamos y del que hablaremos a continuación (HTK), a este tipo de entrenamiento basado en palabras aisladas se lo considera únicamente como una posible fase de arranque de la fase de entrenamiento, pero no como una opción de entrenamiento completa en si misma.

La mecánica del entrenamiento continuo de habla se basa en el cálculo en paralelo de todos los parámetros de todos los HMM, los pasos serían los siguientes:

- Reservar espacio para todos los acumuladores y todos los parámetros de todos los HMM.
- Obtener la siguiente muestra de entrenamiento.
- Construir un HMM compuesto por la concatenación de todos los HMMs correspondientes a la transcripción de los símbolos de la muestra de entrenamiento.
- Calcular las probabilidades *forward* y *backward* para el HMM compuesto. En este punto hay que incluir estados intermedios entre distintos símbolos que no corresponden a segmentos de habla, lo cual modifica ligeramente el cálculo de ambas probabilidades.

- Usar dichas probabilidades para calcular las probabilidades de ocupación de cada estado en cada instante y actualizar los acumuladores tal y como se explicó anteriormente.
- Repetir el punto 2 hasta agotar las muestras de entrenamiento.
- Usar los acumuladores para calcular las nuevas estimaciones de los parámetros de todos los HMMs.
- Estos pasos deben repetirse tantas veces como se desee hasta conseguir la convergencia buscada.

3. Reconocimiento de habla en cabinas de avión

El uso de reconocedores de habla en el mundo de la aviación no es nuevo puesto que ya a mediados de los años 80, el programa Advanced Fighter Technology Integration (AFTI) [11] integraba sistemas de reconocimiento en los aviones de combate F-16 para tareas como la fijación de frecuencias de radio, control de displays de vuelo o el establecimiento de coordenadas para el manejo de la nave en modo autopiloto [11][12]. Una particularidad de estos sistemas era que su vocabulario era bastante reducido con el fin de resultar lo más fiable posible.

En la actualidad existen varios ejemplos de aplicaciones de sistemas de reconocimiento automático de habla en la práctica, muchos de ellos relacionados con el ámbito militar. En el caso de los aviones de combate hay que tener en cuenta efectos que no se producen en las cabinas de los aviones comerciales como por ejemplo el ruido producido en las máscaras de oxígeno o el efecto Lombard, producido por el incremento del esfuerzo vocal cuando se intenta compensar un nivel elevado de ruido de fondo [16]. Este último tiene graves efectos sobre los reconocedores de habla pues modifica características básicas de la señal como la frecuencia fundamental o la propia articulación de los fonemas.

Dentro del mencionado ámbito militar, actualmente existen aviones de combate en servicio que implementan este tipo de sistemas, como por ejemplo el Eurofighter Typhoon. Desarrollado en un programa internacional en el que participan Reino Unido, Alemania, Italia y España, posee un sistema DVI[5] que permite al piloto la introducción de diversos comandos simplemente con la voz, pudiendo por ejemplo asignar objetivos con dos comandos de voz [13]. Se trata de un sistema dependiente del hablante de manera que el entrenamiento del sistema se realiza con muestras de los comandos pronunciadas por cada piloto. En España este modelo entró en servicio en 2004 [14] y esta prevista la llegada de 87 de estos aviones, lo cual supondría el 14% de la flota total existente.

Otro proyecto importante en la implementación de sistemas de reconocimiento de habla en aviones de combate es el "Joint Strike Fighter" de la Fuerza Armada de los Estados Unidos[28] [29]. Este proyecto se encuentra realizando pruebas para la implementación de un sistema de reconocimiento de habla en los aviones F-35 como alternativa al clásico control a través del panel de mandos. El sistema se basa en software Dynaspeak [30] y hace uso de un micrófono corto incluido en la máscara de oxígeno, en este caso el software está pensado para ser independiente del hablante.

Existe otro avión de combate para el cual se han realizado estudios en el campo del reconocimiento de habla es el JAS 39. En él se ha estudiado el efecto de las fuerzas G a la hora de implementar un sistema de reconocimiento[16].

---

[5] Direct Voice Inpunt

Mas centrada en la aviación comercial existe una patente que implementa tanto un sistema de reconocimiento de habla como uno de conversión texto-habla con el objetivo de manejar de manera más cómoda y rápida diversas tareas de manejo y monitorización del avión [17].

En nuestro proyecto nos hemos centrado en al ámbito de los aviones comerciales, concretamente nuestra base de datos está grabada en la cabina de un Boeing 737. Dentro de este ámbito uno de los problemas mas importantes que presenta el entorno de las cabinas de avión para tareas de reconocimiento de habla es el ruido debido a las reverberaciones. La señal de voz que capta el micrófono está formada por la onda que sigue el camino directo más una cantidad variable de versiones de ella misma atenuadas y retardadas, producto de las reflexiones que se producen dentro de la cabina. Este fenómeno es el que usualmente se conoce como reverberación y resulta en una distorsión en el dominio del tiempo que afecta a la inteligilibilidad de la señal de voz.

El efecto de la reverberación puede describirse como la convolución de la señal de habla limpia con la respuesta al impulso de la sala en la que se habla (RIR[6]). La duración de la RIR normalmente se sitúa entre los 200ms y los 1000ms lo cual excede significativamente la longitud típica de las ventanas de análisis que se suele situar entre los 10ms y los 40ms. Por esta razón no es factible el uso de modelos tradicionales de adaptación intra-trama y es necesario el uso de técnicas que exploten la información contenida en varias tramas tal y como hacemos por ejemplo en la técnica de substracción espectral como se verá más adelante.

La respuesta al impulso simulada de un entorno que presenta reverberación es la siguiente [5]:

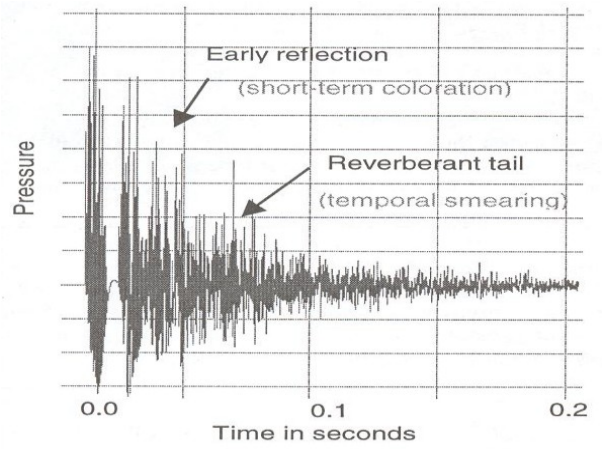

Figura 9: Señal con reverberación

Como podemos ver por dicha respuesta la reverberación se puede descomponer en dos efectos principales. El primero de ellos es que las señales reflejadas pueden llegar a enmascarar sonidos subsiguientes, este tipo de problema suele agravarse si la distancia

[6] Room Impulse Response

entre el hablante y el micrófono que debe captar la señal es elevada. Por esa razón los mayores problemas suelen darse en las consonantes finales ya que se ven enmascaradas por la energía reflejada de toda la señal de voz precedente. En este punto es importante introducir el concepto de tiempo de reverberación T_{60}, el cual se define como el tiempo en que tarda en decaer la energía de la onda sonora 60dB desde el momento en que se apaga la fuente de sonido.

El segundo efecto se analiza mejor en el dominio de la frecuencia ya se traduce en el enmascaramiento del espectro de la onda directa por un ruido cuyo espectro es similar en forma al de una señal de voz, lo que se percibe como una coloración del ruido.

A continuación haremos una revisión de las técnicas fundamentales disponibles en la literatura encaminadas a paliar los efectos de la reverberación sobre el reconocimiento automático de habla.

3.1. De-Reverberación en el dominio del tiempo

Puesto que el efecto de la reverberación es la suma de varias componentes de la señal original que recorren distintos caminos, se puede pensar en eliminar dichas componentes en el dominio del tiempo . Para ello un esquema tipo sería el siguiente [25]:

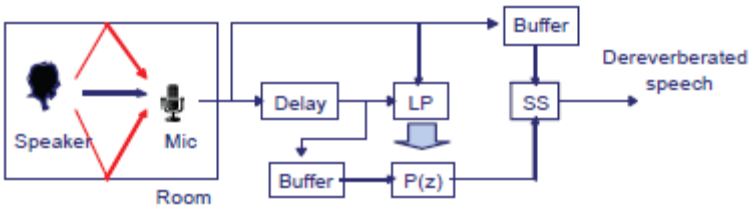

Figura 10: De-reverberación en el dominio del tiempo

La señal observada, $x[n]$ se retarda t muestras y se procesa mediante predicción lineal para calcular el filtro de predicción $P(z)$ el cual se aplica a la muestra $x[n-t]$ para producir una réplica de la reverberación incluida en $x[n]$. Finalmente dicha réplica es substraida de $x[n]$ para obtener la salida de-reverberada.

Sin embargo si la substracción se realiza en el dominio del tiempo, los resultados no serán buenos en el caso de las componentes a largo plazo ya que se genera incluso reverberación extra, tal y como se ilustra en la Figura 10 en la cual T representa el tiempo y A corresponde a la amplitud de la función de transferencia [25]

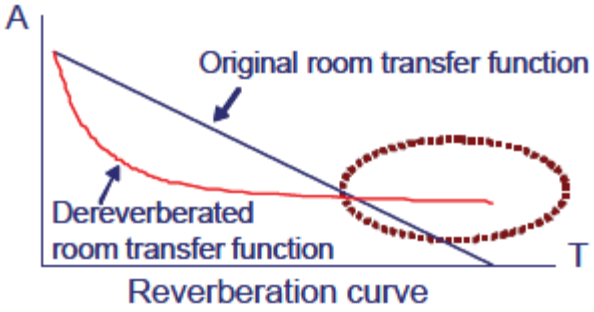

Figura 11: Efectos de la de-reverberación en el tiempo

Podemos ver que para las componentes a corto plazo sí se consigue mejorar, no obstante para las componentes a largo plazo no sólo no se mejora sino que incluso se tiende a empeorar ya que se genera reverberación extra.

Es por eso que para combatir los efectos de la reverberación se suele trabajar en el dominio de la frecuencia mediante la substracción espectral ya que de esa manera se consiguen paliar los efectos negativos de trabajar en el dominio del tiempo.

3.2. Normalización de media y varianza

Como hemos explicado, la parametrización usada en nuestros experimentos fue MFCC. En nuestro caso usamos vectores de 39 componentes, de los cuales 13 corresponden a los componentes básicos (12 mas 1 parámetro de energía), 13 parámetros de velocidad y otros 13 de aceleración.

Dada la naturaleza de los parámetros es posible realizar con ellos una serie de manipulaciones muy interesantes. Puesto que el efecto del canal de transmisión se traduce en multiplicar el espectro de la señal de voz por la función de transferencia del canal, en el dominio cepstral esta multiplicación se transforma en una suma que puede ser en parte compensada restando la media cepstral de todos los vectores de parámetros. Es lo que se conoce como "Normalización de la media cepstral"

Dado que en la práctica esta media debe estimarse sobre un número limitado de parámetros, la compensación no podrá ser exacta, pero esta simple técnica es muy efectiva en la práctica para amortiguar los efectos causados por ejemplo por el uso de diferentes micrófonos en el espectro de la señal recibida.

Esta es una técnica muy común que no va encaminada específicamente a combatir el efecto de la reverberación, pero que si mejora las prestaciones del sistema y por ello la usamos en todos los experimentos, incluyendo el experimento de referencia. No obstante debemos indicar que la versión de normalización de media que usamos de manera común a todos los experimentos únicamente se basaba en los datos de un único fichero de audio, mientras que posteriormente modificamos dicha normalización en media para que se realizase a partir de todo el material de audio de un determinado hablante.

Además de la normalización en media, también la varianza puede ser normalizada, siendo además muy recomendable aplicar ambas normalizaciones. Para mejorar la robustez del reconocimiento, tanto la media como la varianza deben ser calculadas sobre el mayor número posible de datos. Es por esto que para ambos casos hicimos uso de todo el material de audio de cada uno de los hablantes, de manera que hicimos una normalización en media y varianza para cada uno de los hablantes de la base de datos.

3.3. Substracción espectral

La substracción espectral es una técnica ya más enfocada al propio problema del ruido de reverberación.

El principio de esta técnica es el siguiente, cuando se graba audio en una sala en la cual tenemos reverberación presente, el efecto de la reverberación puede modelarse como un canal que distorsiona la señal de audio tal y como explicamos anteriormente, es decir, la señal que observamos es la resultante de la convolución de la señal deseada

con la respuesta al impulso del canal[8] con el cual estamos modelando el efecto de la reverberación.

Suponiendo que el canal sea lineal e invariante en el tiempo, el espectro recibido $X(\omega)$ es igual al producto del espectro de audio $S(\omega)$ y el espectro del canal $C(\omega)$, es decir:

$$X(\omega) = S(\omega) \cdot C(\omega) \quad \textbf{(42)}$$

En la práctica no trabajamos directamente con la Transformada de Fourier de la señal recibida $X(\omega)$ si no con la DFT $X(n,\omega)$ siendo n el índice de tiempo al rededor del cual se calcula una versión enventanda de la DFT, es decir, para cada instante de tiempo n calculamos la DFT correspondiente a la ventana de tiempo centrada en el instante n y cuya longitud es uno de los parámetros a fijar en la parte de procesado.

Si la ventana de análisis es lo suficientemente grande y suave (en nuestro caso ventana Hamming), la propiedad del producto sigue cumpliéndose de manera aproximada por lo que asumiendo que el canal es invariante en el tiempo (no depende de n) tenemos que:

$$X(n,\omega) \approx S(n,\omega) \cdot C(\omega) \quad \textbf{(43)}$$

Tomando logaritmos en ambos lados de (43) obtenemos la siguiente expresión:

$$\log X(n,\omega) \approx \log S(n,\omega) + \log C(\omega) \quad \textbf{(44)}$$

A partir de (44) se podría en teoría eliminar de manera aproximada la influencia de $C(\omega)$ junto con cualquier porción constante de la señal de voz si eliminamos la media sobre n de $\log X(n,\omega)$.

Es la misma lógica que se usa en la normalización de la media cepstral antes expuesta pero con la diferencia de que en el aquel caso las ventanas típicas son de 20ms y por tanto se centra en los efectos a corto plazo. Sin embargo el tiempo de reverberación en una sala, por ejemplo, suele estar mas cercano al segundo y sus efectos son a largo plazo por lo que se hace necesaria una ventana de análisis mucho mayor.

Para poder integrar la substracción espectral con los sistemas ASR[9] actuales, hicimos uso de un programa aparte programado en C que producía una señal de voz resintetizada a la cual se le había realizado la substracción espectral [6]. Esa señal es la que posteriormente se entregaba a nuestro sistema de ASR.

---

[8] la RIR (Room Impulse Response)

[9] Reconocimiento automático del habla (Automatic Speech Recognition)

Las labores de análisis espectral se realizan mediante una DFT con enventanado de Hamming de longitud variable y con una separación entre ventanas de N/4 siendo N el número de muestras usadas en la DFT. De esta manera para una ventana de 1’024 seg corresponderían 8192 muestras a una frecuencia de muestreo de 8KHz.

Puesto que apenas existen diferencias prácticas en prestaciones cuando se calcula o no la normalización en fase [7], únicamente realizamos la normalización en magnitud.

3.4. VAD como apoyo a la substracción espectral

Para intentar mejorar las prestaciones de la técnica de substracción espectral decidimos intentar apoyarla mediante el uso de un VAD[10].

Tal y como se ha explicado en el apartado anterior, podemos modelar el efecto de la reverberación como el efecto de un canal. Efecto que pretendemos estimar y posteriormente substraer en el dominio de la frecuencia del espectro de señal recibido.

Para poder mejorar la estimación de dicho espectro podemos plantearnos estudiar únicamente aquellas tramas en las cuales no haya señal de voz, ya que de esta manera en dichas tramas sólo estará presente el efecto de la reverberación que pretendemos combatir y por tanto la estimación que hagamos de él será mejor.

Para la aplicación del VAD nos apoyamos en el VAD implementado en la Release 8 del codificador AMR-NB[11] del grupo 3GPP [9] y de las dos especificaciones de VAD que permite hicimos uso de la opción 1.

El esquema que realiza la decisión del VAD es el siguiente:

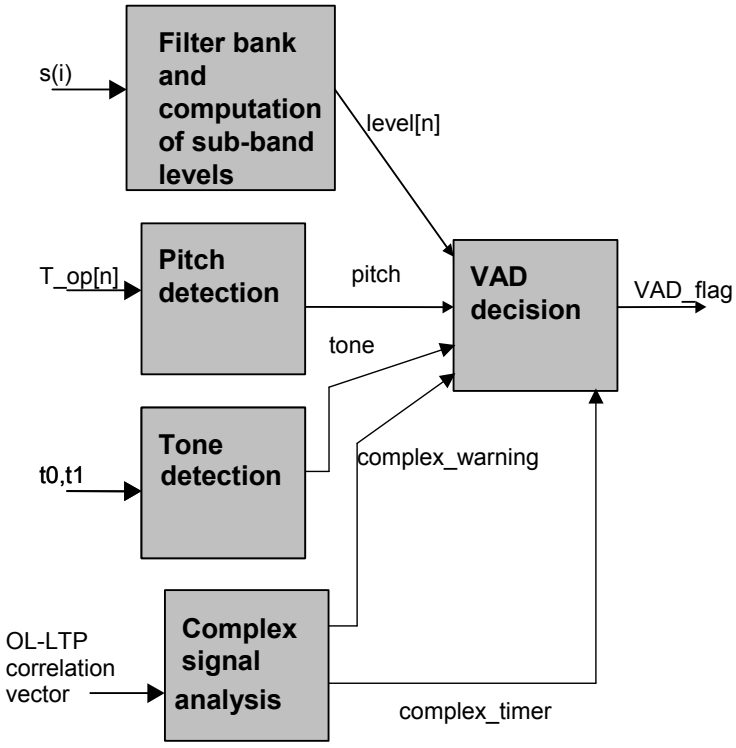

Figura 12: Esquema del VAD[9]

[10] Voice Activity Dectector

[11] Adaptive Multi-Rate - Narrow Band

El algoritmo del VAD usa diferentes parámetros del codificador de voz para calcular un flag booleano (VAD flag) que indica si la trama es de voz o no. Dichos parámetros tal y como se puede ver en la anterior figura son:

- Nivel de sub-bandas
- Indicador de frecuencia fundamental (pitch)
- Presencia de tonos de información.
- Flag de señal compleja.

Las muestras de la trama se dividen en sub-bandas y se calcula el nivel de señal en cada una de ellas (level[n]). La detección de frecuencia fundamental (pitch) se realiza mediante lazos abiertos y el detector de pitch nos indica si hay o no tal señal. El bloque de detección de tonos nos indica si se han encontrado tonos de información los cuales son detectados en base a la ganancia de pitch en los análisis en lazo abierto del codificador. El bloque de detección de señal compleja calcula un flag que nos indica la presencia o no de algún tipo de señal correlada compleja como podría ser el caso de la música.

La potencia de la trama bajo estudio se calcula de la siguiente manera:

$$pow\_sum = \sum_{i=-L\_NEXT}^{L\_FRAME-L\_NEXT-1} s(i) * s(i) \qquad (45)$$

Siendo L_FRAME la longitud en muestras de una trama. En este caso, puesto que la longitud de trama es de 20ms y la frecuencia de muestreo es de 8KHz tenemos 160 muestras por trama. L_NEXT es el número de muestras que el codificador toma para el cálculo, en este caso 40.

La diferencia entre los niveles de señal entre la trama entrante y ruido se estima de la siguiente manera:

$$snr\_sum = \sum_{n=1}^{9} MAX(1.0, \frac{level[n]}{bckr\_est[n]})^2 \qquad (46)$$

Siendo *level[n]* el nivel de señal en la banda n y *bckr_est[n]* el nivel de ruido. Para realizar la decisión del VAD se toma el valor de la variable *snr_sum* y se compara con un determinado umbral. Este umbral puede ajustarse hasta conseguir la sensibilidad deseada frente al nivel de ruido de fondo.

El umbral de decisión depende del nivel medio de ruido, el cual se calcula mediante la siguiente expresión:

$$noise\_level = \sum_{n=1}^{9} bckr\_est[n] \qquad (47)$$

Puede entonces calcularse el umbral del VAD mediante la siguiente formula:

$$vad\_thr = VAD\_SLOPE * (noise\_level - VAD\_PI) + VAD\_THR\_HIGH \quad (48)$$

Donde _SLOPE, VAD_P1, y VAD_THR_HIGH son constantes ajustables en función de la sensibilidad deseada.

Se aplica entonces el decisor:

```
if (snr_sum > vad_thr)

vadreg = 1

else

vadreg = 0
```

Finalmente se refina esa decisión en función del carácter de las últimas tramas para poder detectar por ejemplo niveles bajos de potencia al final de las tramas de habla, subjetivamente importantes pero de difícil detección.

3.5. Normalización del tracto Vocal

Tal y como se explicó en la introducción, cada hablante posee un tracto vocal de diferente longitud de manera que el espectro de la señal sonora presenta características que dependen del mismo. En reconocimiento de habla, la técnica VTLN intenta eliminar las características individuales de cada hablante para mejorar el rendimiento del sistema de reconocimiento.

Una manera de implementar la normalización del tracto vocal (VTLN[12]) es escalar el eje de frecuencias en el banco de filtros.

HTK[13] soporta un escalado lineal simple controlado por un factor de escalado α. Dicho factor ha de ser encontrado mediante técnicas de búsqueda y es necesario establecer dicho factor de escalado para cada uno de los hablantes de manera independiente [4].

[12] Vocal Tract length Normalization

[13] Ver Anexo A

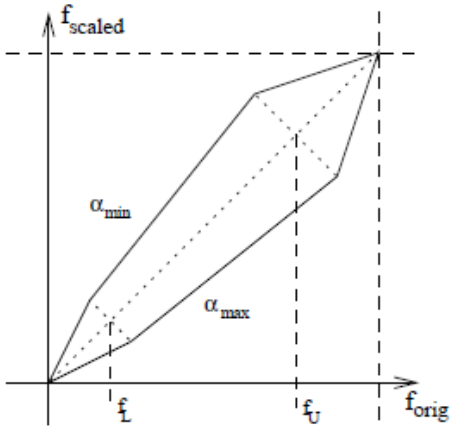

Figura 13: Función de escalado para VTLN

Para la aplicación de esta técnica es por tanto necesario encontrar un parámetro de escalado específico para cada uno de los hablantes de la base de datos. En HTK esto implica generar de manera automática un archivo de configuración específico para cada uno de los hablantes. Esto unido a la cantidad de tiempo de simulación que sería necesario para la estimación de cada parámetro hizo que nos centrásemos en otras técnicas. Al final del apéndice A se comentan los pasos que deberían seguirse para realizar la implementación práctica de esta técnica en HTK.

3.5.1. VTLN en el dominio del tiempo

La forma más habitual de uso de la normalización del tracto vocal es en el dominio de la frecuencia, no obstante es posible realizar esta técnica en el dominio del tiempo con la idea de poder acelerar el proceso de normalización [10].

Como hemos dicho, normalmente se trabaja en el dominio de la frecuencia manipulando el espectro de nuestra señal de habla tanto en módulo como en fase, lo cual implica trasladar la señal de voz del dominio del tiempo al dominio de la frecuencia y posteriormente devolverla al dominio del tiempo por medio de la DFT y la DFT inversa respectivamente.

No obstante existen sistemas con menos recursos para los cuales todo este procesado resulta computacionalmente demasiado costoso por lo que surge la motivación de aplicar todo el procesado necesario únicamente en el dominio del tiempo y la razón por la que surge la idea del TD-VTLN (time domain VTLN).

Las normalización suele basarse en funciones de escalado las cuales pueden describirse en base a parámetros que nos indiquen características como la linealidad o no linealidad de dicha función. En general una función de escalado se puede definir de la siguiente manera:

$\tilde{\omega}\omega \mid \xi_1, \xi_2, \ldots; 0 \leq \omega, \tilde{\omega} \leq \pi$, donde $\xi_1, \xi_2 \ldots$son los parámetros de escalado y ω es la frecuencia normalizada con π correspondiente a la mitad de la frecuencia de muestreo de acuerdo al criterio de Nyquist.

Un ejemplo de función de escalado seria la siguiente:

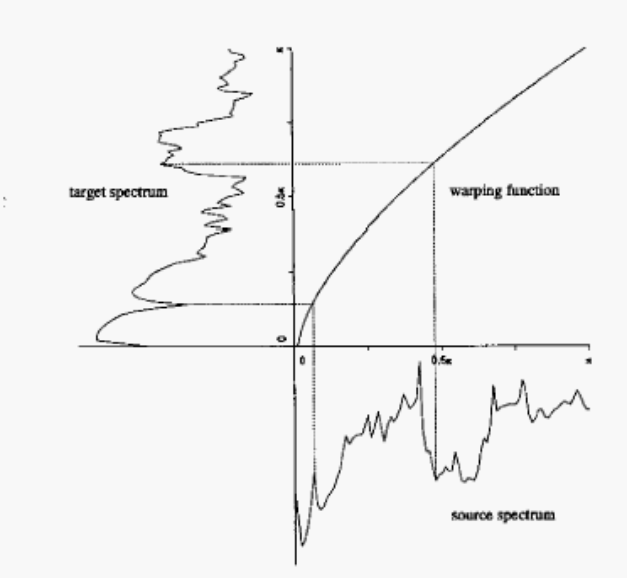

Figura 14: Función de escalado general[10]

Para intentar comprender qué es lo que ocurre en el dominio del tiempo vamos a escoger como ejemplo una función de escalado muy común en incluso ligeramente mas compleja de lo que nos permite HTK.

Vamos a suponer una función de escalado lineal pero con varios segmentos

$\tilde{\omega}(\omega \mid \omega_1', \tilde{\omega}_1') = \alpha_i \omega + \beta_i \tilde{\omega}\omega \mid \omega_1^I, \omega_1^{-I} = \alpha_i \omega + \beta_i$ para $\omega_i \leq \omega < \omega_{i+1}; i = 0 \ldots I$

con $\alpha_i = \frac{\tilde{\omega}_{i+I} - \tilde{\omega}_i}{\omega_{i+I} - \omega_i}$, $\beta_i = \tilde{\omega}_{i+1} - \alpha_i \omega_{i+1}$ y $0 = \omega_0 < \omega_1 < \cdot < \omega_I < \omega_{I+1} = \pi$, para $\tilde{\omega}_i$ equivalente

Podemos ver un ejemplo de esta función monótona y acotada en la siguiente figura:

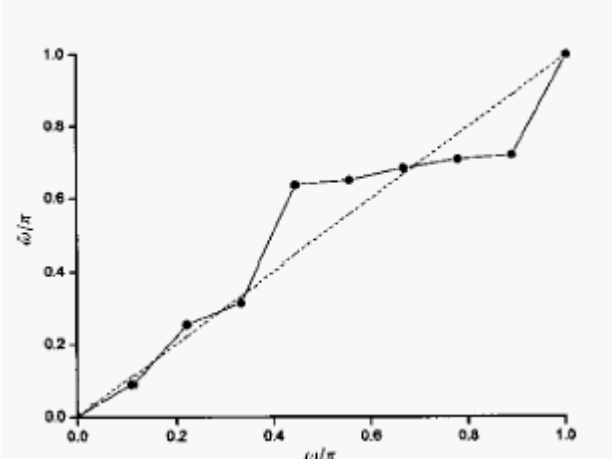

Figura 15: Función de escalado por segmentos[10]

Ahora vamos a calcular la expresión del espectro escalado $\tilde{X}$ derivado de X aplicando la función de escalado $\tilde{\omega}$. Se mantiene la siguiente relación:

$$\tilde{X}(\tilde{\omega}(\omega)) = X(\omega) \Rightarrow \tilde{X}(\omega) = X\left(\tilde{\omega}^{-1}(\omega)\right) \quad (49)$$

Calculamos la función inversa de $\tilde{\omega}$:

$$\tilde{\omega}^{-1}\omega = \frac{\omega - \beta_i}{\alpha_i} \quad (50)$$

$$\frac{\omega_i - \beta_i}{\alpha_i} \leq \omega < \frac{\omega_{i+I} - \beta_i}{\alpha_i}; \qquad i = 0 \ldots I \quad (51)$$

La ecuación puede rescribirse como:

$$\tilde{\omega}^{-1}(\omega) = \sum_{i=0}^{I} \frac{\omega - \beta_i}{\alpha_i} R(\alpha_i \omega + \beta_i \mid \omega_i, \omega_{i+1}) \quad (52)$$

Siendo:

$$R(\omega \mid \omega', \omega'') = rect\left[\frac{\omega - \frac{\omega'' + \omega'}{2}}{\omega'' - \omega'}\right]; \omega' < \omega''$$

$$= \begin{cases} 1 & : \quad \omega' < \omega < \omega'' \\ 0.5 & : \quad \omega = \omega' \vee \omega = \omega'' \\ 0 & : \quad en\_otro\_caso \end{cases} \quad (53)$$

Substituyendo (52) en (49) obtenemos:

$$\widetilde{X}(\omega)=\sum_{i=0}^{I} X\left(\frac{\omega-\beta_i}{\alpha_i}\right) R(\alpha_i\omega+\beta_i \mid \omega_i,\omega_{i+1}) \quad (54)$$

Ahora vamos a ver que ocurre en el dominio del tiempo, para ello vamos a explotar varias propiedades de la DFT, en particular las propiedades de escalado, desplazamiento, convolución y linealidad.

Para empezar vamos a describir el primer término en el dominio del tiempo:

$$u(t):=F^{-1}\left\{X\frac{\omega-\beta}{\alpha}\right\}t=\alpha e^{i\beta t}x\alpha t \quad (55)$$

El segundo término lo obtenemos gracias a la correspondencia en tiempo de la función *rect* en el dominio de la frecuencia:

$$F^{-1}\{\mathrm{rect}w\}t=\frac{T}{\pi t}\sin\frac{t}{2T}, \quad (56)$$

Donde T es la longitud de la trama en tiempo. Utilizando las propiedades de escalado y desplazamiento podemos obtener una expresión algo compleja para la versión en tiempo rt derivada de $R\alpha\omega+\beta$.

Por último, una multiplicación entre los espectros de dos señales corresponde a una convolución de las mismas en el dominio del tiempo de manera que obtenemos:

$$F^{-1}\left\{X\frac{\omega-\beta}{\alpha}\cdot R\alpha\omega+\beta\right\}t=u*rt. \quad (57)$$

Es importante decir que puesto que trabajamos con señales discretas en el dominio del tiempo, el escalado se llevaría a cabo mediante interpolación por spline cúbico con un número de puntos que dependerían de la longitud de la trama. Dependiendo del parámetro α la interpolación cubrirá una parte del tiempo de trama ($\alpha<1$) o bien cubrirá un intervalo de tiempo superior al tiempo de trama ($\alpha>1$) por lo que daría lugar a tramas de diferentes longitudes que podrían unirse mediante el procesado TD-PSOLA[15] que habría que ejecutar $I+1$veces.

Lo verdaderamente interesante de esta técnica como se ha comentado es el ahorro computacional que supone no tener que realizar tanto la DFT como la DFT inversa que tienen complejidad $O(T^2)$. Sin embargo tenemos que hacer una operación de convolución la cual también tiene complejidad $O(T^2)$, por lo que en principio no habría ahorro en el tiempo de cómputo. No obstante para el caso particular en el cual $I=0$, es decir, cuando la función de escalado únicamente posee un único segmento.

En ese caso particular el ahorro computacional es notable tal y como se puede ver a continuación:

[15] Time Domain pitch-synchronous overlap an add

	FD-VTLN	TD-VTLN
DFT	$4T^2 - 2T$	-
Spline interpolation	40T	40T
IDFT	$4T^2 - 2T$	-
PSOLA	4T	4T
Total	$8T^2 + 40T$	44T

Tabla 1: Coste computacional VTLN

Este ahorro computacional facilita la implementación de sistemas cuyo tiempo de respuesta pueda ser crítico, lo cual puede ser una necesidad importante en aplicaciones relacionadas con la aeronáutica como es nuestro caso, es por ello que resulta interesante mencionar esta posibilidad aun cuando no hayamos podido implementarla en este proyecto.

3.6. Otras técnicas modernas relacionadas con la problemática abordada

Dado que el objetivo del proyecto era la puesta en marcha del entorno de experimentación con las técnicas más básicas no se han implementado técnicas sofisticadas. Sin embargo, hemos creído oportuno citar en esta sección otras alternativas.

Las técnicas actuales más efectivas para combatir el efecto de la reverberación están enfocadas al uso de múltiples micrófonos en recepción para la captura de la señal de voz. Desgraciadamente, durante el transcurso de este proyecto no fue posible obtener una base de datos de estas características para el entorno que consideramos. Aun así incluimos una revisión de las técnicas básicas aplicables en dicho caso en la sección 3.6.2.

No queremos, por lo tanto, dejar de lado la situación en la que únicamente haya un micrófono ya que dependiendo del tipo de aplicación en ocasiones no se podrá disponer de otro esquema en recepción. Es además, habitual que las bases de datos que se poseen para entornos específicos como el que nos ocupa sean muy pequeñas y tengan que complementarse con otras bases de datos ya existentes. Así, no queremos dejar de reseñar una de las técnicas más efectivas en dicho caso: la adaptación de los modelos HMM..

3.6.1. Adaptación de los HMM

Una nueva aproximación para combatir los efectos de la reverberación es adaptar tanto la energía y los parámetros espectrales de los HMM así como sus derivadas temporales a las modificaciones del entorno [33]. El único parámetro necesario para realizar la adaptación es el tiempo de reverberación T_{60}. A pesar de que T_{60} es dependiente de la frecuencia, en la práctica normalmente se realiza la aproximación de considerarlo independiente de la misma.

Normalmente el problema de la reverberación va asociado a otros efectos como el ruido de fondo, de manera que existe la necesidad de combinar la adaptación a las condiciones de reverberación con la adaptación al ruido de fondo. Dicha adaptación al ruido se basa en la estimación del propio ruido de fondo antes de que se comience a detectar la señal de voz.

3.6.1.1. Modelado de los efectos

La respuesta al impulso se modela como una exponencial decreciente en el tiempo con la siguiente expresión:

$$E(t) = E_0 \cdot e^{-\frac{6 \cdot \ln(10)}{T_{60}} t} \quad (58)$$

$$h^2(t) \sim e^{-\frac{6 \ln(10)}{T_{60}} t} \quad (59)$$

Siendo E_0 la energía de la excitación.

Como vemos, el único parámetro que es necesario estimar es el tiempo de reverberación T_{60} el cual depende de factores como el tamaño de la sala en la que produzca la señal de voz o las características de absorción de sus muros y suele tomar valores de entre 0.2 y 0.4 segundos para habitaciones pequeñas , 0.4 y 0.8 para salas grandes e incluso puede estar por encima del segundo para salas muy grandes como pudiera ser el interior de una iglesia.

A partir de la respuesta al impulso de la sala[16], podemos obtener la función de transferencia de la misma sin más que aplicar la transformada de Fourier a dicha respuesta al impulso. El contorno de esa función de transferencia normalmente tiene variaciones muy rápidas en frecuencia aunque de cara al procesado únicamente resulta de interés la envolvente de la misma, especialmente en los casos en que se usan bancos de filtros para realizar la parametrización de la señal de audio..

En la Figura 16 puede verse que la reverberación lleva a una extensión artificial de la contribución de cada sonido. Este efecto pude describirse en frecuencia como una modulación del espectro y matemáticamente pude obtenerse que la función de transferencia de dicha modulación pude describirse mediante la expresión (60) [26]:

[16] la RIR

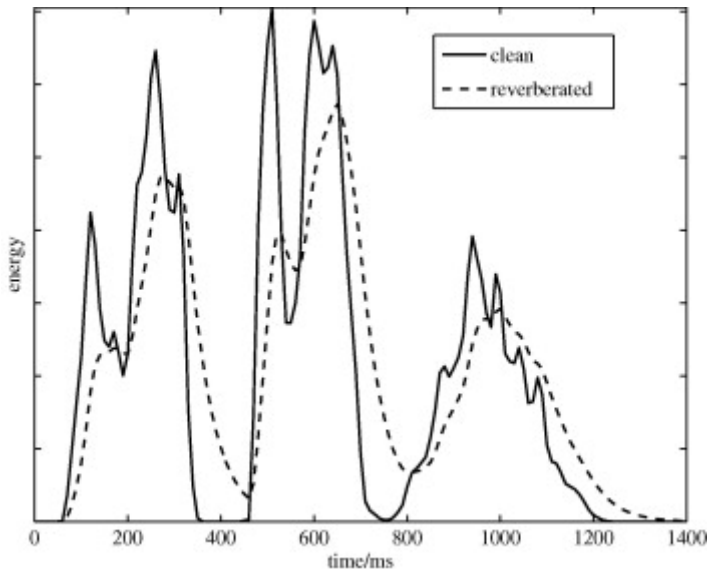

Figura 16: Comparación espectro normal y con reverberación [33]

$$m(F) = \frac{1}{\sqrt{1+(2\cdot\pi\cdot\frac{T_{60}}{6\cdot\ln(10)})^2}} \qquad (60)$$

En la Figura 17 podemos apreciar la característica paso-bajo de la función de modulación para distintos valores de T_{60}, cuanto mayor sea el valor de T_{60} menor será la frecuencia de corte de la función de modulación. Esta contribución extra de los sonidos precedentes puede conseguir que los parámetros acústicos de los sonidos de baja energía se vean enmascarados por los de un sonido precedente de alta energía.

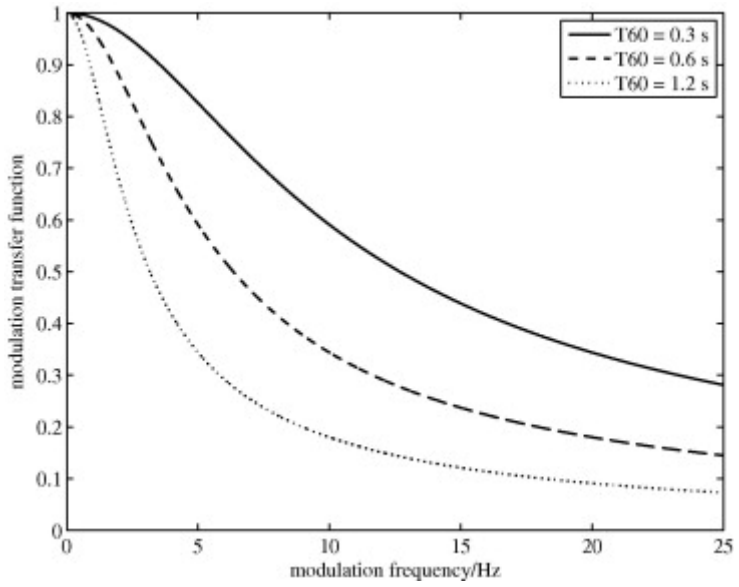

Figura 17: Función de modulación para varios valores de T_{60} [33]

3.6.1.2. Adaptación de los parámetros estáticos

Tal y como describimos en la sección 2.4, cada palabra pude modelarse con un HMM de varios estados. Cada estado tiene una duración media $dur(S_i)$ que puede obtenerse a partir de la probabilidad de mantenerse en ese estado $p(S_i \mid S_i)$ de la siguiente manera:

$$\mathrm{dur}(S_i) = \frac{1}{1-p(S_i|S_i)} \cdot t_{\mathrm{shift}} \quad (61)$$

La contribución de una determinada excitación acústica acaecida en el primer estado hacia siguientes puede estimarse integrando la ecuación (59) tomando como intervalo de integración la duración del estado.

$$\alpha_{i,1} = \int_{t_s(S_i)}^{t_e(S_i)} h^2(t)\partial t \quad (62)$$

siendo:

$$t_s(S_i) = \sum_{j=1}^{i-1} \mathrm{dur}(S_j) \quad (63)$$

y:

$$t_e(S_i) = t_s(S_i) + \mathrm{dur}(S_i) \quad (64)$$

con:

$$\int_0^{\infty} h^2(t)\partial t = 1 \quad (65)$$

Normalmente cada estado de los HMM's se define mediante un conjunto de parámetros como pueden ser por ejemplo los parámetros MFCC's. Estos parámetros representan las medias de distribuciones gaussianas, y sus correspondientes varianzas terminan por definir por completo dichas distribuciones.

La media del parámetro de energía de un determinado estado S_i puede adaptarse teniendo en cuenta las contribuciones de los estados precedentes de la siguiente manera:

$$E(S_i) = \alpha_{i,i} \cdot E(S_i) + \alpha_{i,i-1} \cdot E(S_{i-1}) + \alpha_{i,i-2} E(S_{i-2}) + \ldots = \sum_{j=1}^{i} \alpha_{i,j} \cdot E(S_j) \quad (66)$$

En la práctica únicamente suelen ser relevantes 2 o 3 estados anteriores al actual dependiendo del tiempo de reverberación y la duración media de los estados.

De la misma manera pueden adaptarse las medias de las densidades espectrales de potencia. En el caso de usar parámetros MFCC, el primer paso es volver a llevarlos al dominio de la frecuencia mediante IDCT:

$$\left\{C_0, C_1, C_2 \ldots\ldots, C_{NR_{cep}}\right\} \xrightarrow{IDCT} \left\{\log(|X_1|), \log(|X_2|), \ldots\ldots, \log(|X_{NR_{mel}}|)\right\} \xrightarrow{EXP} \left\{|X_1|, |X_2| , \ldots\ldots |X_{NR_{mel}}|\right\} \quad (67)$$

Siendo NR_cep el número de coeficientes y NR_mel el número de bandas en la escala de frecuencia MEL.

Entonces podemos adaptar la densidad espectral de potencia de manera análoga al parámetro de energía:

$$\tilde{|X_k(S_i)|^2} = \alpha_{i,i} \cdot |X_k(S_i)|^2 + \alpha_{i,i-1} \cdot |X_k(S_{i-1})|^2 + \alpha_{i,i-2} \cdot |X_k(S_{i-2})|^2 + \ldots\ldots = \sum_{j=1}^{i} \alpha_{i,j} \cdot |X_k(S_j)|^2 \quad (68)$$

Con $1 \le k \le NR\_mel$

Posteriormente es necesario obtener nuevamente los parámetros MFCC a partir del espectro adaptado $\tilde{X}$

Las varianzas no suelen adaptarse ya que tienen una influencia menor en el rendimiento del reconocimiento [27].

La Figura 18 muestra el resultado que se obtiene en el HMM de una palabra determinada tras el procesado comparándolo con el HMM de la misma palabra entrenado en condiciones libres de reverberación y comparándolo también con el HMM de la misma palabra entrenado en condiciones de reverberación:

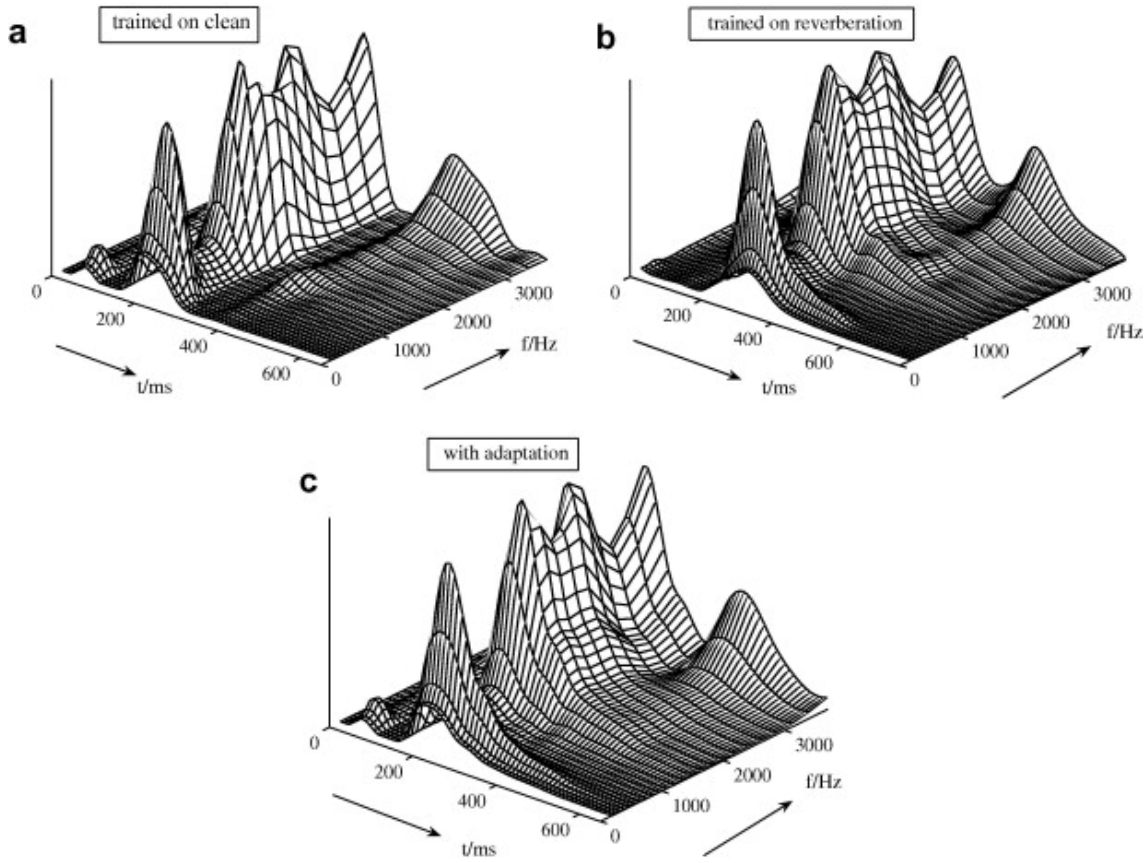

Figura 18: Comparación de HMMs [33]

Puede observarse en dicha figura que tras la técnica de adaptación, el resultado es muy similar a lo que obtendríamos en caso de haber entrenado los modelos directamente en condiciones de reverberación.

3.6.1.3. Adaptación de los parámetros delta.

De la misma manera que los parámetros estáticos se ven afectados por los efectos de la reverberación, también se ven afectados los parámetros delta. Esto puede observarse por ejemplo viendo las gráficas de la Figura 18 los "valles" existentes en el HMM entrenado sin reverberación son de alguna manera "rellenados" por efecto de las colas que genera la reverberación en el HMM adaptado, lo cual indica que las derivadas en esas zonas se ven modificadas.

Los parámetros Delta de una determinada trama se calculan mediante la diferencia ponderada de los parámetros estáticos de las tramas siguiente y precedente. De esta manera la energía logarítmica de las deltas se calcula de la siguiente manera:

$$\Delta \log E(t_i) = \frac{\sum_{j=1}^{s} j \cdot [\log E(t_{i+j}) - \log E(t_{i-j})]}{\text{norm}} \qquad (69)$$

con:

$$\text{norm} = 2 \cdot \sum_{j=1}^{3} \quad j^2 \qquad (70)$$

De esta manera podemos obtener las energías delta medias tanto para el HMM limpio como para el adaptado $\Delta \log\{E_{clean}(t_i)\}$ y $\Delta \log\{E_{adapted}(t_i)\}$.Con ellos es posible obtener la diferencia entre ambas:

$$\Delta\log\{E_{\text{diff}}(t_i) = \Delta\log\{E_{\text{adapted}}(t_i) - \Delta\log\{E_{\text{clean}}(t_i) \qquad (71)$$

Finalmente para conseguir la versión adaptada de los parámetros delta, se le añade una versión ponderada de esta diferencia a los parámetros delta de la versión sin reverberación:

$$\Delta\log\{E(t_{S_i}, \text{mix}_j) = \Delta\log E_{\text{clean}}(t_{S_i}, \text{mix}_j) + \beta \cdot \log\{E_{\text{diff}}(t_{S_i}) \qquad (72)$$

Esto se realiza para cada estado del HMM y para cada mezcla de gaussianas. Un valor común para el parámetro β es 0.7.

Para los parámetros cepstrales el proceso es análogo y la Figura 19 resume el proceso de obtención de la diferencia media entre los parámetros cepstrales de los HMM sin reverberación y el adaptado:

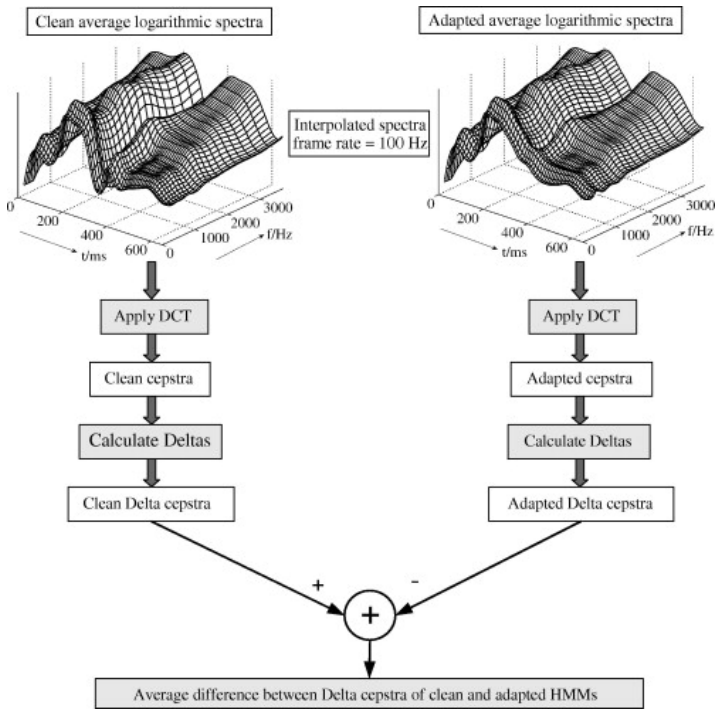

Figura 19: Esquema para la obtención de la diferencia media entre parámetros[33]

Y los parámetros cepstrales se obtendrían de la siguiente manera a partir de dicha diferencia:

$$\Delta C_m(S_i, \text{mix}_j) = \Delta C_{m_{clean}}(S_i, \text{mix}_j) + \beta \cdot \Delta C_{m_{diff}}(S_i) \quad (73)$$

Los parámetros deta-delta se calculan a partir de los delta de la misma manera en que los delta se obtienen de los estáticos.

3.6.1.4. Estimación de T_{60}

Como hemos comentado, el único parámetro que es necesario estimar es el tiempo de reverberación (T_{60}). El reconocimiento de una determinada muestra se realiza mediante un conjunto de HMM adaptados en los cuales el valor de T_{60} se ha estimado a partir del reconocimiento de la muestra anterior. Después de dicho reconocimiento se realiza una búsqueda del conjunto de HMM adaptados que proporcione máxima verosimilitud para un reconocimiento forzado del mismo conjunto de HMM ya reconocidos.

Para determinar el valor de T_{60} que se usará para la muestra siguiente, se va variando su valor en escalones típicamente de $\pm 20ms$ y con dicho valor se reconstruye la anterior secuencia de HMM pero adaptada al nuevo valor de T_{60} y se calcula la probabilidad de que el último vector de parámetros coincida con la nueva secuencia de HMM adaptados. La búsqueda no se suele extender más de $\pm 40ms$ ya que la variación temporal de T_{60} suele ser muy reducida.

A continuación podemos ver un esquema que ilustra el proceso de estimación:

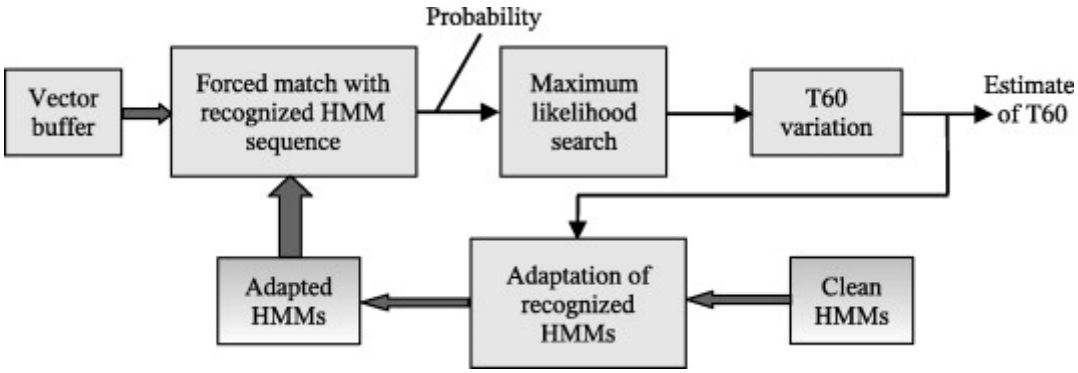

Figura 20 Esquema para la estimación de T_{60} [33]

3.6.2. Técnicas multimicrófono

Es muy común emplear varios micrófonos para la recepción de la señal de manera que en función de la colocación de los mismos, recibimos diferentes versiones de la señal de voz. Cada una de ellas tendrá unas características diferentes en términos de retardo y reflexiones lo cual permite desarrollar diversas técnicas de dereverberación que aumentarán las prestaciones de los sistemas de reconocimiento de habla que se vean afectados por la misma.

El objetivo de la dereverberación es reducir la energía de las componentes reverberantes en la señal recibida con lo que se minimizan los efectos nocivos de la misma, para ello existen diversas aproximaciones.

Un método habitual es usar un array de micrófonos: se estiman las direcciones de llegada de la señal directa y se mejora la calidad de la señal recibida variando la directividad del array de micrófonos. Para conseguir resultado satisfactorios en cuanto a dereverberación, ésta técnica requiere el uso de un número elevado de micrófonos ya que de otra manera no se conseguirá suficiente ganancia directiva.

Otra aproximación consiste en encontrar el filtro inverso de la habitación que provoca la reverberación para poder deconvolucionar la señal recibida y recuperar la señal original. No obstante este método funcionara de manera efectiva únicamente cuando se cumplan determinadas condiciones. Si las diversas funciones de transferencia (RTF's[17]) desde la fuente hacia cada uno de los micrófonos son conocidas, se puede construir un filtro inverso que recupere de manera totalmente precisa la señal original. La existencia de dicho filtro inverso está garantizada si el número de micrófonos es superior al número total de fuentes y si las diferentes RTF's no tienen ceros comunes . No obstante en la práctica no podemos suponer que las funciones de transferencia son invariantes en el tiempo ya que por ejemplo los propios interlocutores podrían variar su posición con lo cual las condiciones de cada canal variarían. Además hay que tener en cuenta que estimar de manera exacta las distintas funciones de transferencia es algo muy complicado en la práctica por lo que llevar a cabo este tipo de dereverberación es una tarea compleja.

Precisamente por la dificultad del último método, se han estudiado otros métodos con una idea similar pero en los que el concepto de filtro inverso se relaja hacia la idea de un filtro de dereverberación y en los que no se tienen ningún tipo de conocimiento a priori de las RTF's. A éste tipo de dereverberación se lo conoce como dereverberación ciega . En lo sucesivo nos referiremos a dicha versión relajada como "filtro inverso" aunque estrictamente no lo sea.

[17] Room Transfer Function

3.6.2.1. Dereverberación Ciega

Supongamos que una únca fuente de voz es capturada por L micrófonos siendo $L > 1$ y que la señal tiene una pequeña componente adicional de ruido, sean $s_t, x_t^{(l)}$ y $b_t^{(l)}$ las secuencias digitalizadas de la fuente, la señal observada y el ruido respectivamente donde t y l son el tiempo y el índice de los micrófonos receptivamente. Entonces el proceso de observación puede modelarse como:

$$x_t^{(l)} = \sum_{n=0}^{M-1} a_n^{(l)} s_{t-n} + b_t^{(l)} \qquad (74)$$

Donde $a_m^{(l)}$ es la RIR de longitud M desde la fuente al $l_{\text{ésimo}}$ micrófono. El objetivo por tanto es estimar un filtro dereverberador que produzca una señal de salida que contenga la menor cantidad de reverberación posible inducida por la RIR (o equivalentemente la RTF) y cuyas características espectrales mas se asemejen a las de una señal de voz sin reverberación.

La señal mejorada se obtendría mediante:

$$y_t = \sum_{l=1}^{L} \sum_{n=0}^{K-1} w_n^{(l)} x_{t-n}^{(l)} \qquad (75)$$

Siendo $w_n^{(l)}$ el filtro de dereverberación de longitud K. La siguiente figura ilustra el modelo de observación y dereverberación:

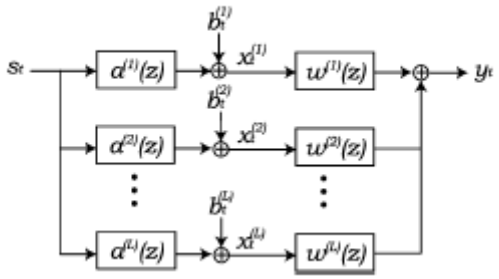

Figura 21: Filtro dereverberador[43]

Algunos de los problemas a los que se enfrenta la dereverberación ciega son los siguientes:

- Tratamiento de fuentes coloreadas:

  Cuando la fuente es un proceso blanco, el filtro inverso puede estimarse fácilmente buscando un filtro lineal que "blanquee" la señal observada, sin embargo dicho blanqueo eliminará o reducirá la inherente correlación de las características de señal original.

- Sensibilidad al ruido:

En general, un gran número de ceros de las diferentes RTF's se encuentran muy cercanos los unos a los otros. En esas condiciones un filtro inverso tenderá a amplificar el ruido en la señal reconstruida.

- Variación temporal de las RTF's:

Las RTF's siempre sufren variaciones temporales en condiciones normales (variación posicional de la fuente, fluctuaciones de temperatura en la habitación...). Si no se consigue actuar correctamente en relación a estas variaciones, se cometerán errores graves en la estimación del filtro.

- Procesado en tiempo real:

En muchas aplicaciones el procesado en tiempo real es un requerimiento básico, por lo que debe tenerse en cuenta a la hora aplicar el sistema en la práctica

.

3.6.2.2. Otras técnicas

Existen diferentes maneras de abordar el problema de la dereverberación, las técnicas habituales mas importantes son las siguientes:

- Método del subespacio

El método del subespacio puede estimar el filtro inverso independientemente de las características de la fuente. El autovector de la matriz de covarianza de la señal observada cuyo autovector sea el menor de todos, se toma como estimación de la RTF y el filtro inverso es calculado como el inverso de dicha estimación. Existen técnicas que dotan a este método de mayor robustez respecto al ruido y a la variación temporal de las RTF's. Sin embargo este método habitualmente presenta inestabilidades numéricas conforme el orden del canal crece y además se ve bastante afectado por el ruido por lo que no suele ser una técnica muy eficiente cuando se emplea en escenarios reales.

- Dereverberación basada en deconvolución

Este método estima el filtro inverso de las RTF's como el filtro que "blanquea" la señales observadas o las transforma en secuencias independientes. Por ejemplo en se supone que un pequeño filtro blanqueador puede eliminar el efecto de las características de la fuente de la señal observada y por tanto el residuo que queda de las diferentes señales esta compuesto de manera aproximada por las RTF's convolucionadas con una secuencia independiente de manera que aplicando técnicas de deconvolución de orden superior a los residuos, pueden obtenerse las inversas de las RTF's.

Una de las técnicas basadas en este principio es el MCLP[18], se le considera un método "robusto" de deconvolución ciega para comunicaciones digitales. En este caso el cálculo del filtro inverso se realiza mediante la aproximación de suponer a la fuente como un proceso Autoregresivo (AR) excitado por ruido blanco gaussiano. Bajo esa suposición, un filtro que "blanquee" la señal tiene una respuesta al impulso que es el resultado de la convolución del filtro inverso de la RTF y la fuente del proceso AR. Por tanto el filtro inverso de cada RTF se obtiene mediante la descomposición apropiada del filtro blanqueador.

Otros métodos que involucran el uso de análisis de predicción son los basados en MSLP[19]. MSLP intenta suprimir las componentes reflejadas asumiendo que hay un parámetro de retardo que separa el efecto de la colorización de la señal y el de la reverberación. Bajo ciertas suposiciones, MSLP produce una predicción fiable de la amplitud de dichas reflexiones y eso combinado con la substracción espectral ofrece mejoras en las prestaciones del ASR. Una de las virtudes del

[18] Multi Channel Linear Prediction

[19] Multistep Linear Prediction

MSLP es que es una técnica que trabaja de manera estable, efectiva y a un costo computacional muy bajo incluso con un único micrófono.

- HERB[20]

HERB se sirve del hecho de que la señal de voz tiene una inherente estructura armónica de manera que estima el filtro dereverberador como aquel que recupera dicha estructura armónica en la señal observada. HERB es capaz de conseguir una alta capacidad de dereverberación pero requiere de unos tiempos de observación de la señal muy elevados lo que lo limita en la aplicación práctica en determinados ámbitos

[20] Harmonicity based dereverberation

4. Experimentos

En este apartado describiremos el proceso de puesta en marcha de un entorno de experimentación de reconocimiento de habla en cabinas de avión que hemos llevado a cabo. Además ofreceremos los resultados de los distintos experimentos realizados en base a las técnicas mencionas en el apartado 3 así como la funcionalidad del diverso software involucrado. Para la gran mayoría de los experimentos hicimos uso del software HTK [20] del cual hablaremos algo más en el Apéndice A. Además también hicieron falta programas en C y MATLAB para completar alguna de las fases tal y como comentaremos más adelante.

Este capítulo está organizado de la siguiente manera: en primer lugar describiremos la base de datos con la cual hemos trabajado, en primer lugar comentando brevemente los objetivos del proyecto dentro del cual se enmarca y posteriormente describiremos con mayor detalle las características del material de audio que contiene. A continuación nos centraremos en el desarrollo de la parte práctica del proyecto empezando por explicar como se ha desarrollado la fase de entrenamiento, el material de audio adicional que necesitamos para la misma así como la funcionalidad de los distintos scripts involucrados en el mismo. Posteriormente nos centraremos en describir de manera práctica la implementación de las diversas técnicas que pudimos probar así como los principales problemas que nos encontramos en la aplicación de las mismas.

4.1. Puesta en marcha del entorno de experimentación

Empezaremos describiendo la base de datos que utilizaremos y el origen de la misma. A pesar del progreso realizado en las últimas décadas, la tecnología existente en materia de reconocimiento automático de habla todavía no es lo suficientemente fiable como para que pueda usarse con garantías en multitud de aplicaciones del "mundo-real", lo cual incluye a la industria aeronáutica.

El proyecto EUIST HIWIRE [18] tiene como objetivo realizar avances significativos en la robustez, naturalidad y flexibilidad de la interacción vocal entre humanos y máquinas para aplicaciones aeronáuticas.

Con ese fin, en la base de datos se intenta reflejar dos de los problemas típicos en un entorno usual en aeronáutica como es la cabina de un avión como son el ruido y el habla no nativa [22]. Para ello en la grabación de la base de datos se han superpuesto el ruido propio de la cabina del avión grabado previamente durante un vuelo real y para la grabación de las distintas frases que componen la base de datos se han usado hablantes de varias nacionalidades no inglesas y hablando todos ellos en lengua inglesa de manera que todo el material de la base de datos corresponde a habla no nativa.

[22] Entendemos por habla no nativa como el caso en el cual una determinada persona usa una lengua o idioma que no se corresponde con la de su país de procedencia

La base de datos HIWIRE [1] contiene órdenes habladas correspondientes a entradas del CPDLC [24].

CPDLC es un medio de comunicación entre el controlador aéreo y la tripulación del avión que reemplaza la tradicional comunicación por voz por el uso de mensajes de texto. Un ejemplo de comandos comunes usados en este sistema son los que se muestran a continuación:

Assigned level minus nine
HF2 seven two seven zero
Position bravo zulu whiskey lima one
Request direct to delta sierra india due to weather
ETA one india nine alpha yankee at three nine minutes
We can accept plus five at zero hours
Weather radar on
FD off
Wilco

Figura 22: Ejemplo de CPDLC [1]

De esta manera desaparecen los problemas debidos a la pobre calidad del audio en el canal HF y el riego de error en la interpretación de los comandos se reduce en gran medida.

Al usar la voz como entrada del sistema, el piloto simplemente pronuncia el comando que quiere introducir en el sistema y automáticamente dicho comando es introducido en la ventana de datos del sistema de comunicaciones.

La longitud de los comandos pueden variar desde una única palabra a mas de doce, el numero total de palabras distintas es 133 y la perplejidad de la gramática es 14.9. El número total de frases distintas que componen la base de datos es 311.

[24] Controller Pilot Data Link Communications

4.1.1. Material de audio

Para la realización de este proyecto se ha contado con una primera parte de la base de datos HIWIRE que fue puesta a disposición del público en el año 2007. Desgraciadamente, no hemos podido contar con la segunda parte que consideramos más interesante para este proyecto por estar grabada con múltiples micrófonos y en condiciones más realistas.

En particular, el material de audio de esta primera parte de la base de datos se divide en dos diferentes tipos de voz grabada: (i) el grupo original de frases grabadas en una habitación en silencio mediante micrófono corto. A este grupo se le denomina "clean" en la base de datos y es el considerado limpio de ruido. Existe otro grupo (ii) de frases obtenidas mediante la adición de ruido grabado en una cabina real de avión al grupo de frases limpio de ruido (i.e "clean"). A este segundo grupo de frases se le considera la parte "ruidosa" de la base de datos.

La parte ruidosa de la base de datos a su vez se subdivide en tres diferentes grupos, cada uno de ellos se compone del grupo original de frases con una determinada relación señal a ruido (SNR) que toma los valores de 10dB, 5dB y -5dB, a cada uno de estos grupos se les hace referencia en la base de datos como "bajo nivel de ruido" (LN), "medio nivel de ruido" (MN) y "alto nivel de ruido" (HN) respectivamente. Cada frase limpia de ruido tiene por tanto su correspondiente versión "ruidosa" en cada uno de los tres diferentes niveles de ruido.

Un total de 8099 frases en lengua inglesa pronunciadas por 81 hablantes no nativos procedentes de Francia, Grecia, Italia y España componen la base de datos. De cada hablante se han grabado 100 frases en la misma sesión y su distribución puede verse en la siguiente tabla:

Country	# Speakers	# Utterances
France	31	3100
Greece	20	2000
Italy	20	2000
Spain	10	999
Total	81	8099

Figura 23: Tabla de hablantes y frases[1]

El grupo limpio de ruido de la base de datos se ha grabado en un PC usando una frecuencia de muestreo de 16KHz con 16 bits por muestra en formato PCM Windows Wav. La SNR media estimada de estas grabaciones es aproximadamente 30dB.

Como se ha comentado, la parte "ruidosa" de la base de datos se ha generado añadiendo artificialmente diferentes niveles de ruido manteniendo constante el nivel de la señal de voz limpia de ruido. El ruido ha sido grabado en la cabina de un Boeing 737 mediante un micrófono AKG Q300 localizado en el tablero de mandos de la cabina. La grabación fue realizada durante un vuelo normal y tiene un comportamiento bastante estacionario. Podemos ver una representación de este ruido tanto en tiempo como en frecuencia en la siguiente figura:

La representación en tiempo y frecuencia del ruido grabado en la cabina del avión es la siguiente:

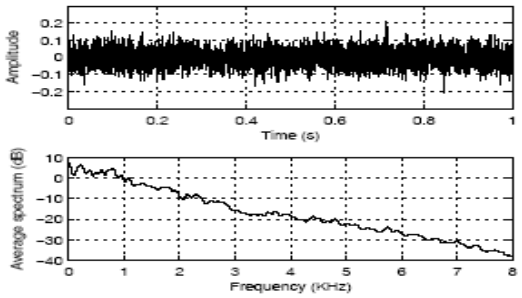

Figura 24: Caracterización del ruido[1]

Partiendo de la estimación inicial de la SNR de la parte "limpia" de la base de datos, el nivel de ruido se ajusta para obtener los deseados niveles de SNR para la parte "ruidosa" de la base de datos de manera que se obtienen aproximadamente los 10dB, 5dB y -5dB anteriormente mencionados. A continuación podemos ver un ejemplo de las diferentes versiones de una misma frase en cada uno de los diferentes niveles de ruido antes mencionados:

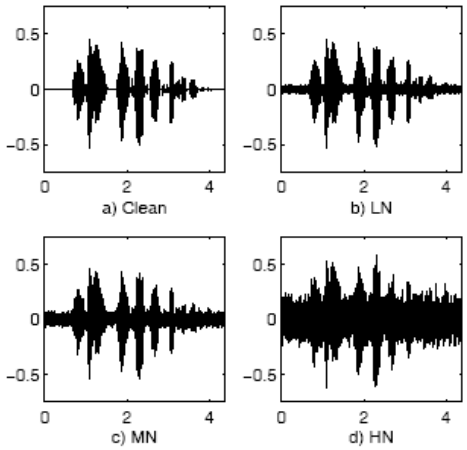

Figura 25: Ejemplo de frase con distintos niveles de ruido[1]

4.1.2. Fase de entrenamiento

El entrenamiento de nuestros modelos no se realiza a partir de la base de datos HIWIRE ya que esta no posee material de audio suficiente para ello. Es por ello que para dicha labor hacemos uso de la base de datos TIMIT [19]. Esta base de datos consta de un total de 6300 frases de 630 diferentes hablantes con 10 frases cada uno de ellos.

La distribución del material de audio en TIMIT es la siguiente:

Región	Hombres	Mujeres	Total
New England	31 (63%)	18 (27%)	49 (8%)
Northern	71 (70%)	31 (30%)	102 (16%)
North Midland	79 (67%)	23 (23%)	102 (16%)
South Midland	69 (69%)	31 (31%)	100 (16%)
Southern	62 (63%)	36 (37%)	98 (16%)
New York City	30 (65%)	16 (35%)	46 (7%)
Western	74 (74%)	26 (26%)	100 (16%)
Army Brat	22 (67%)	11 (33%)	33 (5%)
Total	438 (70%)	192 (30%)	630 (100%)

Tabla 2: Distribución de TIMIT

Para poder realizar el entrenamiento, la propia base de datos HIWIRE proporciona varios scripts que permiten completar el entrenamiento a partir del material de audio de TIMIT.

Los scripts involucrados son los siguientes:

dolbl.sh*:* Es el encargado de las labores de etiquetado. En el caso de mi distribución de Linux tuve que cambiar todos los comandos "awk" del script por "gawk" instalando previamente el paquete correspondiente a ese comando. Fue la única manera de hacerlo funcionar.

dotrain.sh*:* Es el encargado de ejecutar las herramientas HTK adecuadas para el entrenamiento de los modelos. La única modificación necesaria en este script es el ajuste de la variable HTKBIN a la ruta en la que se encuentren instaladas las herramientas HTK. El script genera de manera iterativa modelos con distinto número de gaussianas en escalas correspondientes a potencias de 2. Es decir, primero genera el juego de modelos para el caso de una sola gaussiana, posteriormente hace lo propio para el caso de 2 gaussianas, despues 4, 8 y así iterativamente hasta 256. Para nuestro caso particular elegimos usar modelos de 32 gaussianas ya que consideramos que utilizar modelos con un numero superior no iba a suponer una mejora substancial teniendo en cuenta el tiempo extra de simulación que supondría tener que trabajar con dichos modelos. Consideramos por tanto que 32 gaussianas suponía un buen compromiso entre prestaciones del sistema y tiempo de simulación.

doall.sh*:* Este script se encarga de generar las listas adecuadas a partir de los datos de la base de datos y de llamar a los dos scripts anteriores. Es el script principal de la fase de entrenamiento. La única modificación necesaria en este script es el ajuste de la variable DIRTIMIT a la dirección en la cual se encuentra el materia de audio de TIMIT.

Si además queríamos variar el tipo de parametrización, era necesario cambiar tanto el fichero ***proto*** en el cual se define el tipo de HMM como el fichero ***config.parming*** en el cual se define la configuración de la parametrización. En nuestro caso trabajamos con parametrización MFCC en vectores de 39 elementos repartiéndose en 13 parámetros estáticos, 13 de velocidad y 13 de aceleración. El tipo de enventanado fue Hamming con una longitud de 32ms por ventana y siendo el tiempo entre tramas de 10ms.

Adicionalmente a los ajustes mencionados para los tres scripts también hubo que pasar los nombres tanto de los directorios como de los archivos de la base de datos TIMIT de mayúsculas a minúsculas ya que los scripts estaban pensados para trabajar con material etiquetado en minúsculas mientras que toda la base de datos se encontraba en mayúsculas. Para realizar esta labor se creo el script ***mayusmin.sh***.

4.2. Experimento de referencia

Una vez entrenados los modelos, el siguiente paso consiste en parametrizar los datos de audio con los que queremos trabajar (HIWIRE) para poder evaluar las prestaciones del sistema de reconocimiento. Aquí se presentaba de nuevo el problema de las mayúsculas y minúsculas mencionado en la sección anterior.

Para la realización la parametrización la propia base de datos HIWIRE posee los scripts ***docode.csh*** y ***work_code.csh*** y únicamente es necesario modificar el fichero ***config_hcopy*** en caso de querer modificar parámetros de configuración en la parametrización. En cualquier caso la parametrización debe ser la misma que en la fase de entrenamiento.

Una vez parametrizado todo el material de audio ya podemos comenzar con las tareas de reconocimiento del experimento básico. Para ello la base de datos dispone del script ***work_eval1.csh*** el cual al terminar la simulación nos genera todos los ficheros de resultados.

En este punto se aplicó la técnica de normalización de la media cepstral explicada en la sección 3.2 y que estuvo presente en todos los experimentos dada su sencillez de aplicación

4.3. Normalización de media y Varianza

Tal y como explicamos en la sección 3.2, el paso previo a la aplicación de la normalización era el calculo offline de las medias y varianzas para los vectores de parámetros de cada hablante.

Tanto la media como la varianza se estiman *offline* antes de comenzar el reconocimiento y se almacenan en ficheros independientes, uno para el vector de medias y otro para el de varianzas, y esto para cada uno de los hablantes de la base de datos.

Una vez obtenidos los ficheros de media y varianza, es necesario configurar adecuadamente la herramienta HCopy[25] antes de realizar la parametrización de todos los ficheros de audio. Para ello necesitamos añadir 5 variables del fichero de configuración de HCopy:

- *CMEANDIR*: Indica la ruta en la que se encuentran los ficheros de medias calculados previamente.

[25] Ver anexo A

- *CMEANMASK*: Es una expresión regular en la cual hay que emplear los caracteres especiales **?**,***** y **%** que sirve para poder diferenciar a cada uno de los hablantes.

- *VARSCALEDIR*: Análogo a CMEANDIR pero para los ficheros de varianzas.

- *VARSCALEMASK*: Análogo a CMEANMASK pero para los ficheros de varianzas.

- *VARSCALEFN*: Es la dirección del fichero que contiene el vector de varianza global y contra el que se hace la normalización. Es decir, a partir del los ficheros indicados en la variable VARSCALEDIR se normaliza a 1.0 y posteriormente se reescala a los valores contenidos en el fichero indicado por VARSCALEFN.

Con el fin de obtener los ficheros de medias y varianzas de cada hablante desarrollamos un script que calculaba la media y varianza para cada hablante a partir del total de sus frases en la sección "clean" o sin ruido de la base de datos.

El script desarrolado fue ***calcvar.sh*** y permite el calculo separado o conjunto de la media y/o varianza para cada uno de los hablantes generando un conjunto de ficheros con las medias y otro conjunto con las varianzas de manera que para cada hablante al final disponemos de un vector de medias y otro de varianzas. El script se basa en el uso de la herramienta HCompV de HTK para la cual es necesario crear un pequeño fichero de configuración indicando la longitud de los vectores. La ejecución de la herramienta es la siguiente:

```
HCompV -C $CONFIG -c $OUTDIR -k $PATTERN -q $TYPE
```

Siendo las opciones:

`-C`: Indica el fichero de configuración de la herramienta HCompV.

`-c`: Indica el directorio de salida

`-k`: Indica la máscara con la que se identificará a cada hablante.

`-q`: Indica el tipo de cálculo a realizar. Sus valores pueden ser "m" si queremos calcular la media, "v" para la varianza o "mv" para calcular ambos a la vez.

Una vez obtenidos los ficheros de media y varianza tenemos que ajustar los parámetros que se indicaron en la sección 3.2 y volver a ejecutar el experimento principal para obtener los resultados.

4.4. Substracción espectral

Tal como se explicó en la sección 3.3, la substracción espectral va mas enfocada a solucionar el problema del ruido. Para la aplicación práctica de esta técnica nos basamos en el código desarrollado por David Gelbart [21] el cual se basa en su publicación sobre la substracción espectral aplicada a sistemas de reconocimiento [6].

Para las labores de análisis, tal y como comentamos anteriormente usamos una DFT con enventanado de Hamming de longitud variable y con una separación entre ventanas de L/4 siendo L el número de muestras usadas en la DFT. De esta manera para una ventana de 1'024 seg corresponderían 8192 muestras a una frecuencia de muestreo de 8KHz.

Existían implementaciones tanto en C++ como en MATLAB pero optamos por la primera al ser la más rápida y un mejor uso de la memoria según su propio autor. El programa usado fue ***meansub.c*** .

Para poder compilar el programa eran necesarias dos librerías adicionales: la librería ***fxt*** necesaria para los cálculos de FFT y la librería ***dpwelib*** perteneciente al proyecto SPRACHcore [22].

meansub.c genera ficheros de audio a los cuales realiza la substracción espectral a partir de los ficheros de audio originales. El programa únicamente normaliza la log-magnitud del espectro dejando intacta la fase del espectro tal y como se explicó en la sección 3.3. De esta manera al comienzo de la fase de análisis se separan la magnitud y la fase del espectro en variables independientes. Para cada trama, se calcula la media aritmética de la log-magnitud espectral de esa trama conjuntamente con las W anteriores y las W posteriores, donde W es un número entero que fijamos antes del comienzo de la fase de análisis. La media así calculada se substrae en la propia trama de manera que tenemos la nueva magnitud del espectro al cual se le añade la fase que habíamos dejado intacta, de manera que obtenemos el audio resintetizado que será el que se introducirá a nuestro sistema de reconocimiento.

En la aplicación de esta técnica probamos dos variantes. En la primera de ellas realizábamos la substracción espectral de manera independiente a cada uno de los ficheros de audio mientras que en la segunda realizábamos el cálculo a partir de todos los ficheros de audio concatenados de un mismo hablante y probando distintas longitudes de ventanas. Hay que explicar aquí que la longitud media de los ficheros de audio se situaba en torno a los 3 o 4 segundos y por tanto el tiempo de análisis estaba limitado a la duración de cada fichero de audio. En que en la segunda variante trabajamos con la concatenación de todos los ficheros de audio de un mismo hablante por lo que la longitud total es mucho mayor, el tiempo de análisis para este caso estaba fijado en 12 segundos. Independientemente del tiempo total de análisis, ***meansub.c*** trabaja en base a tramas de audio de una longitud determinada, esta longitud será otro parámetro de experimentación.

Esta segunda variante era posible debido a que todos los ficheros de audio para un mismo hablante están grabados en la misma estancia y bajo las mismas condiciones (i.e no varia la respuesta del canal).

Como hemos comentado, dentro de la segunda variante, realizamos diversas pruebas variando la longitud de la ventana de trabajo (el tiempo total de análisis siempre se mantuvo fijo a 12segundos). Inicialmente dicha longitud estaba fijada a 0,512seg e hicimos experimentos para longitudes de 0,512seg, 0,256seg y 0,128seg.

En la aplicación práctica de esta técnica un problema al que nos enfrentamos fue que ***meansub.c*** generaba ficheros de audio con muestras de 32bits en vez de 16bits a semejanza de los ficheros de entrada.

La razón que da el autor del código original es que el procesado puede aumentar la magnitud de las muestras de audio de manera que de mantener la representación de las muestras en 16bits podrían producirse efectos de *overflow*. Sin embargo en nuestro caso no apreciamos ese tipo de problema por lo que reducir nuevamente el número de bits de las muestras procesadas de 16bits a 32bits no supone ningún tipo de pérdida de información.

El principal problema era que HTK supone que las muestras son siempre de 16bits, y por tanto no era posible el procesado directo mediante HTK de los ficheros a los cuales habíamos realizado la substracción espectral ya que las muestras procesadas tenían 32bits. El resultado era que HTK interpretaba que había el doble de muestras de las que realmente teníamos.

Así mismo, los ficheros de salida eran ficheros sin formato por lo que antes de trabajar con ellos era necesario realizar dos pasos:

1. Darles un formato (en nuestro caso, WAV)
2. Pasar las muestras de 16bits a 32bits

El primer paso lo realizábamos con el programa de conversión sox [23] mientras que para el segundo utilizábamos MATLAB.

Para el segundo paso se desarrollaron dos programas en MATLAB que permitiesen reducir el número de bits por muestra de 32 a 16. Se eligió MATLAB porque resultaba relativamente sencillo manipular los ficheros de audio en formato wav gracias a la toolbox VOICEBOX [24] que incluye los programas readwav.m y writewav.m que nos permiten leer y escribir ficheros en formato wav. Las funciones programadas fueron conversion.m y convertir.m siendo la primera de ellas la función principal que invocamos para realizar la conversión. Únicamente es necesario tener en cuenta la ordenación de bits de los ficheros con los que vamos a trabajar para poder pasarla como parámetro a converision.m para que ésta pueda realizar la conversión correctamente.

Una vez convertidos los ficheros solo nos resta parametrizarlos y ejecutar nuevamente el experimento principal para obtener los resultados.

4.4.1. VAD como apoyo a la substracción espectral

Tal y como se explicó en la sección 3.4, probamos a intentar mejorar las prestaciones de la técnica de substracción espectral combinándola con un VAD con la idea de mejorar la estimación del espectro reverberante y por tanto mejorar las prestaciones de la substracción espectral.

Para ello nos apoyamos en nos apoyamos en el VAD implementado en la Release 8 del codificador AMR-NB del grupo 3GPP tal y como comentamos anteriormente. El código del codificador era muy cerrado al estar estandarizado por lo que diversas variables como la longitud de la ventana o el tiempo entre tramas eran inamovibles sin cambiar casi en su totalidad el programa lo cual no era viable. Por ello únicamente hicimos una pequeña modificación en el código para extraer los resultados del VAD y poder aplicarlos posteriormente al programa con el que realizábamos la substracción espectral.

meansub.c permitía el paso como parámetro un fichero de 1's y 0's que identificaban las tramas etiquetadas como voz de las que no lo eran. La clave por tanto era ser capaces de obtener dicho fichero para cada uno de los ficheros de audio de la base de datos. El primer paso fue extraer a partir del código del codificador AMR la secuencia de 1's y 0's que diferenciaba a las tramas de voz de las que no lo eran, para ello solo hubo que modificar la función *vad_decision* del fichero ***sp_enc.c.*** El resultado final del análisis del VAD era una secuencia de flags booleanos en función del tipo de trama (1 si detectaba actividad vocal, 0 en caso contrario) que se almacenaba en la variable "mem_vadreg". Se modificó el programa para poder extraer dicha secuencia con la idea de generar un fichero de texto que sería el que posteriormente se pasaría como parámetro a ***meansub.c***.

En la aplicación práctica de esta técnica tuvimos problemas con la longitud de las tramas puesto que como hemos dicho los parámetros del codificador AMR eran totalmente fijos en lo que a longitud de trama y tiempo entre tramas se refiere (20ms y 10ms respectivamente), por lo que no era ni sencillo ni recomendable modificar el código original para que trabajase con valores distintos. De cara a la posterior integración con el programa de substracción espectral era vital que al menos el tiempo entre tramas fuese idéntico en ambos casos ya que en caso contrario los flags del VAD obtenidos mediante el codificador AMR no se corresponderían con las tramas analizadas por el programa de substracción espectral.

El problema fue que el algoritmo usado por ***meansub.c*** obligaba a que el tiempo entre tramas fuese exactamente cuatro veces menor que el tiempo de trama por lo que para conseguir un tiempo entre tramas de 10ms necesitábamos analizar tramas de 40ms, lo que no se correspondía de manera exacta con la longitud utilizada en el análisis del codificador AMR. No obstante a efectos prácticos esas diferencias no eran cruciales ya que el máximo error cometido posible se produciría en los primeros y últimos 20ms de cada palabra lo cual es asumible para nuestros propósitos.

De todo lo anterior hay que sacar una conclusión importante, la ventana de trabajo nos viene fijada por las características del propio VAD. No tiene demasiado sentido pensar en aplicar un VAD con ventanas de análisis de longitud elevada ya que por

ejemplo en una ventana de 0.5s puede haber multitud de tramas tanto de voz como de silencio, es necesaria una ventana algo más corta para aumentar la precisión en la discriminación del tipo de trama. Sin embargo para el resto de experimentos si podemos fijar una ventana más larga basándonos en que los efectos de la reverberación típicamente son a largo plazo. Por tanto estamos ante la disyuntiva de que probablemente con esta técnica podamos conseguir una estimación más clara del espectro de ruido pero perdiendo a cambio la visión de los efectos a largo plazo que suele ser la más importante para el ruido por reverberación que, como hemos comentado, en ocasiones puede presentar un tiempo de reverberación (T_{60}) cercano al segundo. En cualquier caso consideramos interesante realizar el experimento y analizar los resultados obtenidos.

A continuación vemos dos figuras que muestran los resultados de aplicar el VAD a uno de los ficheros de la base de datos. La segunda figura es un detalle de la primera:

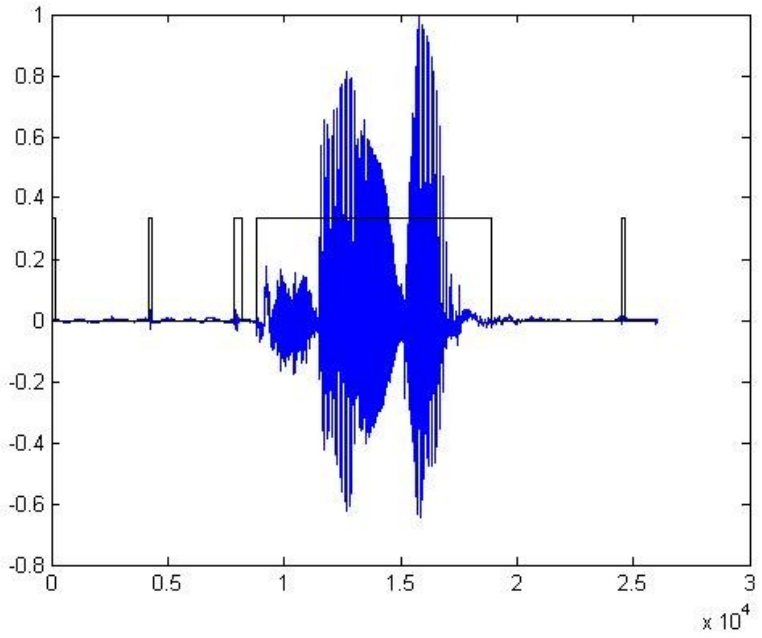

Figura 26: Aplicación del VAD (1)

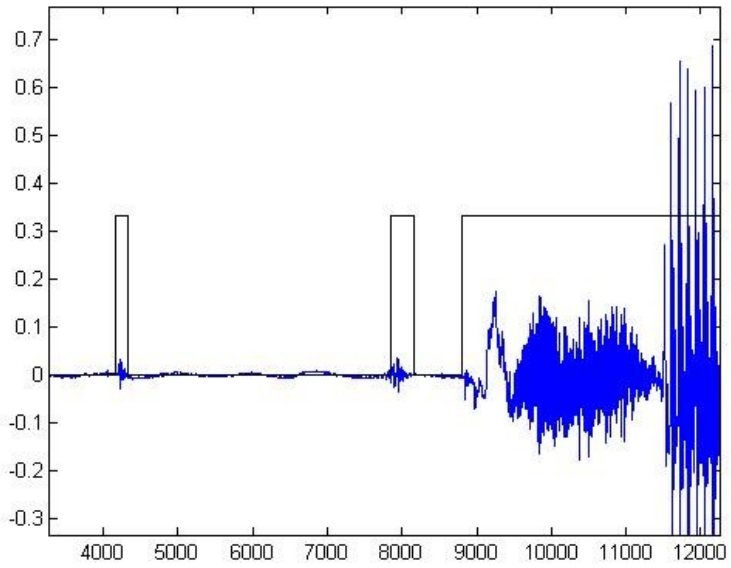

Figura 27: Aplicación del VAD (2)

Las gráficas fueron obtenidas a partir de MATLAB. Los intervalos en los que el VAD detecta voz aparecen resaltados con un pulso positivo. Lo que realmente nos interesaba era discriminar correctamente los momentos en los que no hay voz para poder realizar la estimación del espectro a partir de ellos. Como puede observarse, en general la discriminación es suficientemente buena por lo que seguramente explorar la opción 2 del VAD no nos hubiera llevado a mejoras significativas en el rendimiento final de nuestra aplicación por lo que no se consideró interesante probar esa segunda opción.

4.5. Resultados

A continuación podemos ver los resultados obtenidos para el experimento de referencia en los diferentes niveles de ruido que presenta la base de datos.Para todas las tablas de resultados que siguen se cumple lo siguiente:

Clean	NumFrases	NumPalabras
Francés	3099	9157
Griego	2000	6104
Italiano	1983	6037
Español	999	3048
Sum/Avg	8081	24346

Clean	Palabras correctas(%)	Susbstit(%)	Elimin(%)	Inserciones(%)	ErrorPalabra(%)	ErrorFrase(%)	Precisión
Francés	85,45	13,81	0,73	0,6	15,15	37,11	84,86
Griego	83,99	15,3	0,7	2,42	18,43	42,8	81,58
Italiano	82,9	16,19	0,91	3,06	20,16	42,94	79,84
Español	85,17	13,71	1,12	0,62	15,45	38,14	84,55
Sum/Avg	84,42	14,75	0,81	1,65	17,22	40,07	82,77

Tabla 3: Experimento de referencia. Sin ruido.

LN	Palabras correctas(%)	Susbstit(%)	Elimin(%)	Inserciones(%)	ErrorPalabra(%)	ErrorFrase(%)	Precisión
Francés	32,63	36,96	30,41	0,58	67,95	80,83	32,05
Griego	43,99	36,62	19,4	1,05	57,06	80	42,93
Italiano	44,03	34,34	21,63	1,37	57,35	74,23	42,66
Español	50,1	34,02	15,88	0,75	50,66	72,47	49,35
Sum/Avg	40,39	35,86	23,73	0,91	60,51	77,97	39,48

Tabla 4: Experimento de referencia. Bajo ruido.

MN	Palabras correctas(%)	Susbstit(%)	Elimin(%)	Inserciones(%)	ErrorPalabra(%)	ErrorFrase(%)	Precisión
Francés	12,88	36,63	50,5	0,28	87,41	92,03	12,59
Griego	20,97	39,04	39,99	0,61	79,64	91,95	20,36
Italiano	25,91	36,16	37,93	0,68	74,77	85,93	25,23
Español	27,49	36,84	35,66	0,46	72,97	85,59	27,04
Sum/Avg	19,88	37,13	42,97	0,48	80,60	89,71	19,40

Tabla 5: Experimento de referencia: Ruido medio.

HN	Palabras correctas(%)	Susbstit(%)	Elimin(%)	Inserciones(%)	ErrorPalabra(%)	ErrorFrase(%)	Precisión
Francés	1,52	33,53	64,96	0,01	98,49	98,48	1,5
Griego	0,61	32,91	66,48	0	99,39	99	0,61
Italiano	2,67	34,5	62,83	0,1	97,43	97,83	2,57
Español	1,15	32,38	66,47	0,07	98,92	98,4	1,08
Sum/Avg	1,531	33,47	65,00	0,037	98,50	98,43	1,490

Tabla 6: Experimentote referencia: Ruido alto.

En las tablas se muestran los resultados para cada una de las nacionalidades de los hablantes[26], puede verse una explicación algo más detallada del significado de cada uno de los parámetros en la sección 6.1.4, pero el parámetro en el que nos vamos a basar principalmente a la hora de medir las prestaciones del sistema es la precisión (*accuracy*) media tomando todas las nacionalidades y por tanto será el que usemos en las representaciones gráficas principalmente.

A continuación se muestra la representación gráfica de la precisión media en el experimento de referencia para todas las nacionalidades:

[26] Recordemos que la totalidad de la base de datos esta grabada en lengua inglesa siendo la única diferencia la nacionalidad origen de cada uno de los hablantes.

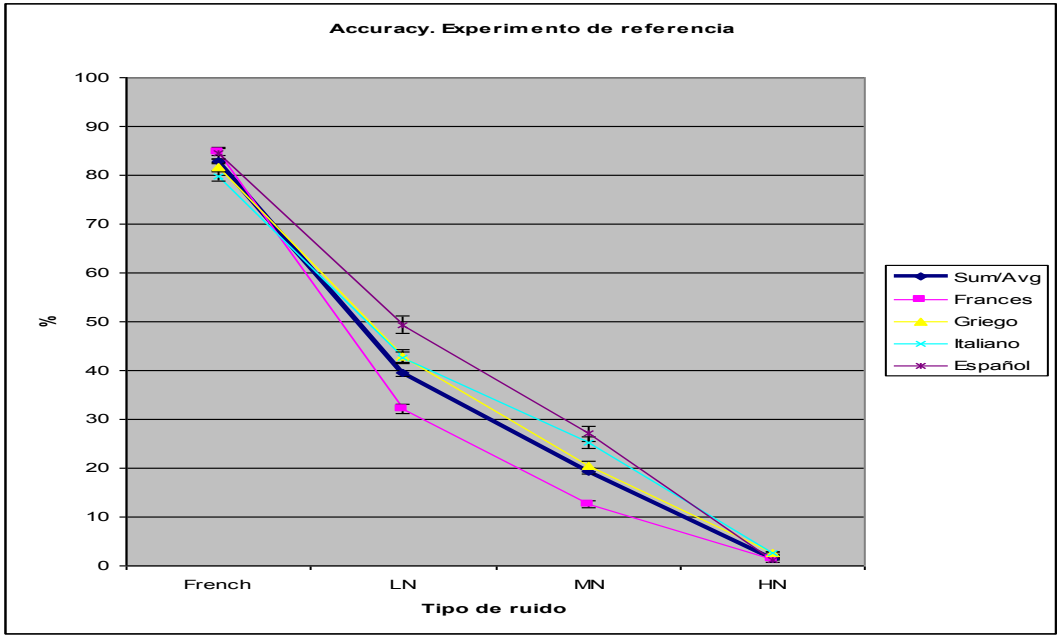

Figura 28: Precisión. Experimento de referencia.

De esta última gráfica puede observarse lo nocivo que resulta el efecto del ruido en las tareas de reconocimiento hasta el punto que para condiciones de ruido elevado (HN) el reconocimiento es prácticamente imposible. Pero la conclusión más importante que puede extraerse es que la nacionalidad del hablante resulta decisiva en relación a las prestaciones del sistema ya que entre hablantes de nacionalidad española y francesa por ejemplo existe una diferencia cercana al 20% en cuanto a precisión. Es por ello que podemos concluir que un parámetro fundamental a tener en cuenta en el diseño de este tipo de sistemas es la nacionalidad de los futuros usuarios pudiendo considerarse la realización de algún tipo de adaptación a la lengua nativa de cada hablante.

A continuación podemos ver los resultados obtenidos tras aplicar la normalización de media y varianza:

Clean	Palabras correctas(%)	Susbstit(%)	Elimin(%)	Inserciones(%)	ErrorPalabra(%)	ErrorFrase(%)	Precisión
Francés	85,6	13,6	0,81	0,8	15,2	36,5	84,79
Griego	84,49	14,91	0,61	1,83	17,35	40,95	82,65
Italiano	84,28	14,94	0,78	3,46	19,18	40,24	80,82
Español	85,04	13,71	1,25	0,92	15,88	38,44	84,12
Sum/Avg	84,93	14,26	0,80	1,722	16,79	38,75	83,20

Tabla 7: Normalización en media y varianza. Sin ruido.

LN	Palabras correctas(%)	Susbstit(%)	Elimin(%)	Inserciones(%)	ErrorPalabra(%)	ErrorFrase(%)	Precisión
Francés	41,88	35,48	22,64	0,73	58,85	75,9	41,15
Griego	58,47	26,67	11,86	1,56	43,09	68,95	59,91
Italiano	54,89	30,81	14,3	1,51	46,61	67,88	53,38
Español	61,19	27,85	10,96	0,89	39,7	63,96	60,3
Sum/Avg	51,56	31,21	16,48	1,14	49,57	70,73	51,16

Tabla 8: Normalización en media y varianza. Bajo ruido.

MN	Palabras correctas(%)	Susbstit(%)	Elimin(%)	Inserciones(%)	ErrorPalabra(%)	ErrorFrase(%)	Precisión
Francés	18,93	37,36	43,42	0,44	81,85	89,77	18,78
Griego	36,76	37,99	25,25	1,25	64,48	84,95	35,51
Italiano	35,5	34,54	29,97	0,78	65,28	80,53	34,71
Español	38,78	35,6	25,62	0,69	61,91	79,98	38,09
Sum/Avg	29,87	36,60	33,42	0,75	71,01	85,09	29,21

Tabla 9: Normalización en media y varianza. Ruido medio.

HN	Palabras correctas(%)	Susbstit(%)	Elimin(%)	Inserciones(%)	ErrorPalabra(%)	ErrorFrase(%)	Precisión
Francés	1,89	33,49	64,62	0,01	98,12	98,03	1,88
Griego	2,46	34,14	63,4	0,11	97,66	98,3	2,35
Italiano	5,63	35,02	59,35	0,23	94,6	96,07	5,4
Español	2,4	32,97	64,63	0,1	97,7	97,8	2,3
Sum/Avg	3,01	33,96	63,02	0,09	97,09	97,58	2,912

Tabla 10: Normalización en media y varianza. Alto ruido.

Ahora compararemos gráficamente la precisión media aplicando esta técnica y el obtenido en el experimento de referencia:

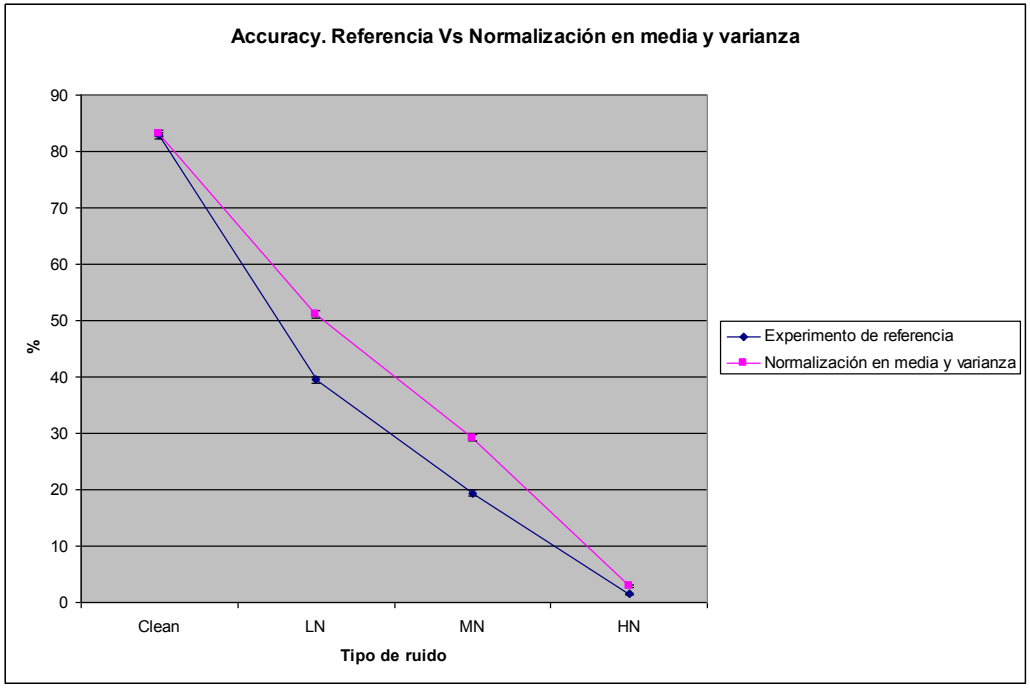

Figura 29: Experimento de referencia Vs Normalización en media y varianza

Podemos ver que con esta técnica conseguimos mejorar las prestaciones del sistema en cada una de las circunstancias de ruido. No obstante tanto para el caso sin ruido como para el caso con el nivel más alto de ruido las diferencias son prácticamente imperceptibles. Para las situaciones de bajo y medio nivel de ruido obtenemos una mejora entorno al 10% lo cual es una mejora notable.

Podemos por tanto concluir que este tipo de normalización es una técnica muy recomendable. No obstante debemos tener en cuenta que para poder aplicarla hemos tenido que calcular los vectores de media y varianza de manera offline lo cual puede suponer una limitación a la hora de

implementar esta técnica en la práctica.

Las siguientes tablas muestran los resultados obtenidos tras aplicar la técnica de substracción espectral pero usando únicamente una sola frase en la estimación del espectro. Después de estas podemos ver las tablas con los resultados de aplicar esta técnica pero aumentando el tiempo total del análisis.

Debemos tener en cuenta para el primer caso que la duración media de cada archivo de audio se situaba en torno a los tres o cuatro segundos mientras que para el segundo concatenábamos todos los archivos de audio de un mismo hablante para poder tomar periodos de análisis mayores, en nuestro caso para el análisis se tomaban periodos de doce segundos.

Independientemente del tiempo total de análisis en cada caso, el programa de substracción trabajaba en base a tramas de una longitud determinada. Para poder realizar la comparación entre ambos casos, la ventana de análisis es la misma (0.512s).

Clean	Palabras correctas(%)	Susbstit(%)	Elimin(%)	Inserciones(%)	ErrorPalabra(%)	ErrorFrase(%)	Precisión
Francés	65,2	27,78	7,02	0,8	35,6	65,12	64,4
Griego	62,76	32,06	5,18	5,1	42,33	75,2	57,66
Italiano	66,84	27,94	5,22	2,87	36,03	65,2	63,97
Español	66,77	27	6,23	1,35	34,58	66,67	65,42
Sum/Avg	65,19	28,78	6,02	2,44	37,24	67,82	62,75

Tabla 11: Substracción espectral usando una frase. Sin ruido.

LN	Palabras correctas(%)	Susbstit(%)	Elimin(%)	Inserciones(%)	ErrorPalabra(%)	ErrorFrase(%)	Precisión
Francés	19,85	38,81	41,33	0,41	80,56	90,67	19,45
Griego	36,2	39,19	24,61	0,8	64,6	85,79	35,4
Italiano	31,49	37,42	31,09	0,89	69,41	85,12	30,6
Español	36,84	38,25	24,9	0,66	63,81	85,19	36,19
Sum/Avg	28,85	38,49	32,64	0,65	71,80	87,42	28,20

Tabla 12: Substracción espectral usando una frase. Ruido bajo.

MN	Palabras correctas(%)	Susbstit(%)	Elimin(%)	Inserciones(%)	ErrorPalabra(%)	ErrorFrase(%)	Precisión
Francés	7,38	36,02	56,6	0,21	92,83	96,26	7,17
Griego	16,64	38,34	45,02	0,52	83,88	94,45	16,12
Italiano	18,57	37,73	43,7	0,45	81,88	91,78	18,12
Español	18,04	36,88	45,08	0,16	82,12	92,89	17,88
Sum/Avg	13,73	37,12	49,14	0,33	86,60	94,29	13,39

Tabla 13: Substracción espectral usando una frase. Ruido medio.

HN	Palabras correctas(%)	Susbstit(%)	Elimin(%)	Inserciones(%)	ErrorPalabra(%)	ErrorFrase(%)	Precisión
Francés	0,94	33,66	65,4	0,02	99,08	99,03	0,92
Griego	0,49	32,68	66,83	0	99,51	99,05	0,49
Italiano	1,71	33,76	64,54	0,05	98,34	98,59	1,65
Español	0,43	32,35	67,22	0	99,57	99,3	0,43
Sum/Avg	0,95	33,28	65,76	0,01	99,06	98,96	0,93

Tabla 14: Substracción espectral usando una frase. Ruido alto.

Clean	Palabras correctas(%)	Susbstit(%)	Elimin(%)	Inserciones(%)	ErrorPalabra(%)	ErrorFrase(%)	Precisión
Francés	80,47	17,87	1,66	0,57	20,09	44,3	79,9
Griego	71,51	25,59	2,9	3,41	31,9	62,15	68,1
Italiano	76,56	21,4	2,04	2,98	26,42	52,34	73,58
Español	78,51	19,09	2,4	1,05	22,54	49,85	77,46
Sum/Avg	77,05	20,79	2,15	1,923	24,86	51,37	75,12

Tabla 15: Substracción espectral, ventana de 0.512s. Sin ruido

LN	Palabras correctas(%)	Susbstit(%)	Elimin(%)	Inserciones(%)	ErrorPalabra(%)	ErrorFrase(%)	Precisión
Francés	29,09	36,52	34,39	0,56	71,46	82,67	28,53
Griego	40,35	38,17	21,47	1,03	60,68	82,09	39,33
Italiano	39,95	35,68	24,37	1,24	61,29	77,26	38,71
Español	46,1	34,74	19,16	0,69	54,59	75,58	45,41
Sum/Avg	36,64	36,50	26,85	0,85	64,21	80,32	35,78

Tabla 16: Substracción espectral, ventana de 0.512s. Ruido bajo.

MN	Palabras correctas(%)	Susbstit(%)	Elimin(%)	Inserciones(%)	ErrorPalabra(%)	ErrorFrase(%)	Precisión
Francés	11,15	35,82	53,03	0,21	89,06	93	10,94
Griego	19,2	38,38	42,41	0,66	81,45	92,7	18,55
Italiano	23,17	36,39	40,43	0,6	77,42	87,59	22,58
Español	23,82	36,15	40,03	0,36	76,54	87,49	23,46
Sum/Avg	17,65	36,63	45,70	0,43	82,77	90,91	17,22

Tabla 17: Substracción espectral, ventana de 0.512s. Ruido medio.

LN	Palabras correctas(%)	Susbstit(%)	Elimin(%)	Inserciones(%)	ErrorPalabra(%)	ErrorFrase(%)	Precisión
Francés	1,47	33,46	65,06	0	98,53	98,45	1,48
Griego	0,61	32,81	66,58	0	99,39	98,95	0,61
Italiano	2,48	34,16	63,36	0,05	97,57	98,08	2,43
Español	0,98	32,25	66,77	0	99,02	98,8	0,98
Sum/Avg	1,44	33,32	65,23	0,01	98,56	98,52	1,43

Tabla 18: Substracción espectral, ventana de 0.512s. Ruido alto.

En la comparación gráfica de los dos casos nuevamente nos fijamos en en la precisión media obtenida en cada caso:

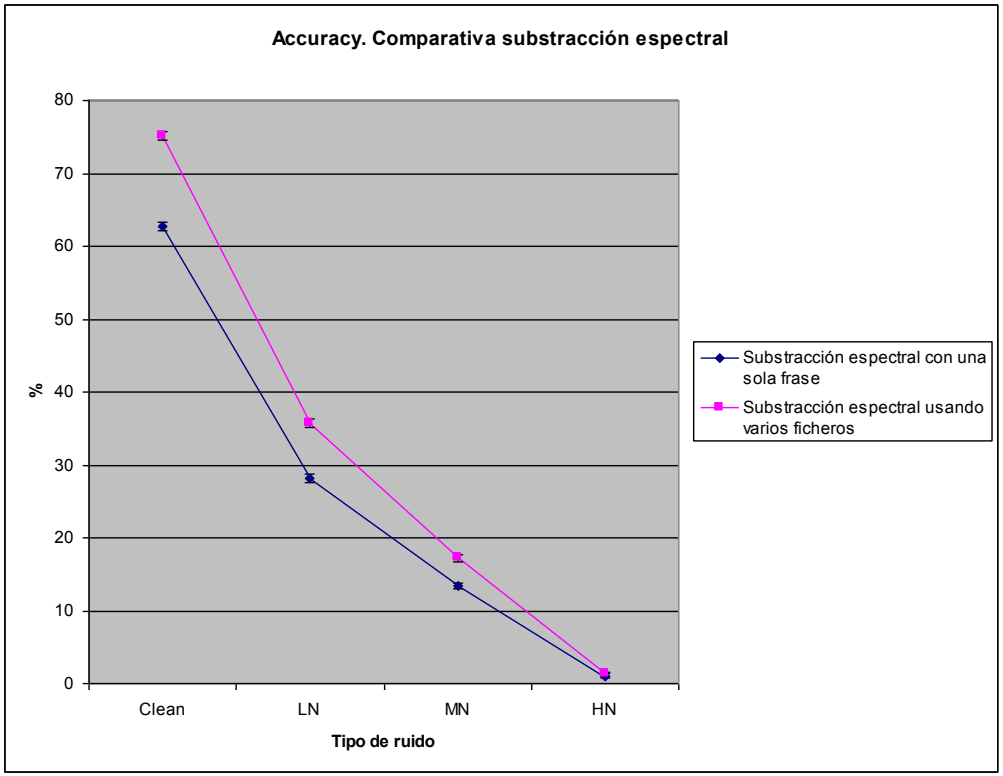

Figura 30: Comparativa substracción espectral uno y varios ficheros.

Podemos ver que los resultados mejoraban al usar varios ficheros involucrados en el análisis, lo cual confirma la hipótesis de partida indicada la sección 4.4 según la cual al estar todos los ficheros de audio para un mismo hablante grabados en la misma estancia y bajo las mismas condiciones, la respuesta del canal no variaba y era posible realizar la concatenación de todos los ficheros de audio para mejorar el análisis.

Nuevamente podemos ver que para el caso de mayor cantidad de ruido no existen diferencias ya que el deterioro es tan grande que prácticamente no permite obtener resultados concluyentes.

A continuación vamos a mostrar los resultados obtenidos al variar la ventana de análisis en la substracción espectral y posteriormente compararemos gráficamente los resultados obtenidos para cada una de ellas:

Clean	Palabras correctas(%)	Susbstit(%)	Elimin(%)	Inserciones(%)	ErrorPalabra(%)	ErrorFrase(%)	Precisión
Francés	81,72	16,95	1,33	0,6	18,88	43,11	81,12
Griego	73,54	23,87	2,59	2,69	29,14	59,45	70,85
Italiano	77,95	20,29	1,76	2,82	24,86	51,34	75,13
Español	78,44	19,03	2,53	0,79	22,34	50,35	77,65
Sum/Avg	78,36	19,78	1,89	1,68	23,31	50,06	76,67

Tabla 19: Substracción espectral, ventana de 0.256s. Sin ruido.

LN	Palabras correctas(%)	Susbstit(%)	Elimin(%)	Inserciones(%)	ErrorPalabra(%)	ErrorFrase(%)	Precisión
Francés	30,62	36,43	32,95	0,52	69,9	81,7	30,1
Griego	41,68	37,63	20,69	1,18	59,5	81,45	40,5
Italiano	41,43	35,37	32,21	1,23	59,8	76,7	31,19
Español	46,95	34,61	18,44	0,82	53,87	74,37	46,13
Sum/Avg	38,028	36,24	27,94	0,89	62,86	79,50	34,92

Tabla 20: Substracción espectral, ventana de 0.256s. Ruido bajo.

MN	Palabras correctas(%)	Susbstit(%)	Elimin(%)	Inserciones(%)	ErrorPalabra(%)	ErrorFrase(%)	Precisión
Francés	12,05	35,78	52,18	0,2	88,15	92,32	11,84
Griego	19,99	38,34	41,68	0,57	80,59	92,3	19,41
Italiano	24,48	36,11	39,41	0,63	76,15	86,79	23,85
Español	25,2	35,96	38,85	0,46	75,26	87,29	24,73
Sum/Avg	18,69	36,51	44,79	0,42	81,74	90,33	18,25

Tabla 21: Substracción espectral, ventana de 0.256s. Ruido medio.

HN	Palabras correctas(%)	Susbstit(%)	Elimin(%)	Inserciones(%)	ErrorPalabra(%)	ErrorFrase(%)	Precisión
Francés	1,53	33,45	65,02	0,03	98,5	98,55	1,5
Griego	0,59	32,99	66,42	0	99,41	99,1	0,59
Italiano	2,62	34,22	63,16	0,08	97,47	98,13	2,54
Español	1,08	32,28	66,63	0,03	98,95	98,7	1,06
Sum/Avg	1,50	33,38	65,10	0,03	98,52	98,60	1,47

Tabla 22: Substracción espectral, ventana de 0.256s. Ruido alto.

Clean	Palabras correctas(%)	Susbstit(%)	Elimin(%)	Inserciones(%)	ErrorPalabra(%)	ErrorFrase(%)	Precisión
Francés	81,96	16,75	1,29	0,61	18,65	42,98	81,35
Griego	73,15	24,21	2,64	2,41	29,26	60,05	70,74
Italiano	77,69	20,69	1,62	2,68	25	51,64	75,01
Español	76,54	20,37	3,08	0,72	24,18	53,15	75,83
Sum/Avg	78,06	20,01	1,92	1,57	23,51	50,58	76,48

Tabla 23: Substracción espectral, ventana de 0.128s. Sin ruido.

LN	Palabras correctas(%)	Susbstit(%)	Elimin(%)	Inserciones(%)	ErrorPalabra(%)	ErrorFrase(%)	Precisión
Francés	30,73	36,54	32,73	0,51	69,78	81,48	30,22
Griego	41,68	37,7	20,62	1,15	59,47	81,39	40,53
Italiano	42,14	34,9	22,96	1,33	59,19	75,95	40,81
Español	47,24	34,68	18,08	0,89	53,64	74,17	46,35
Sum/Avg	38,28	36,19	25,52	0,91	62,63	79,19703	37,364393

Tabla 24: Substracción espectral, ventana de 0.128s. Ruido bajo.

MN	Palabras correctas(%)	Susbstit(%)	Elimin(%)	Inserciones(%)	ErrorPalabra(%)	ErrorFrase(%)	Precisión
Francés	12,45	36,08	51,47	0,22	87,77	92,26	12,23
Griego	19,89	38,2	41,91	0,57	80,68	92,3	19,32
Italiano	24,85	36,16	38,99	0,58	75,73	86,13	24,27
Español	25,89	35,96	38,16	0,59	74,7	86,89	25,29
Sum/Avg	18,99	36,60	44,39	0,44	81,44	90,10	18,55

Tabla 25: Substracción espectral, ventana de 0.128s. Ruido medio.

HN	Palabras correctas(%)	Susbstit(%)	Elimin(%)	Inserciones(%)	ErrorPalabra(%)	ErrorFrase(%)	Precisión
Francés	1,59	33,41	65	0	98,41	98,48	1,59
Griego	0,59	32,86	66,55	0	99,41	99,1	0,59
Italiano	2,72	34,49	62,8	0,08	97,37	98,18	2,63
Español	1,12	32,32	66,57	0,03	98,92	98,7	1,08
Sum/Avg	1,56	33,40	65,03	0,02	98,46	98,58	1,53

Tabla 26: Substracción espectral, ventana de 0.128s. Ruido alto

El resultado de enfrentarlas gráficamente es el siguiente:

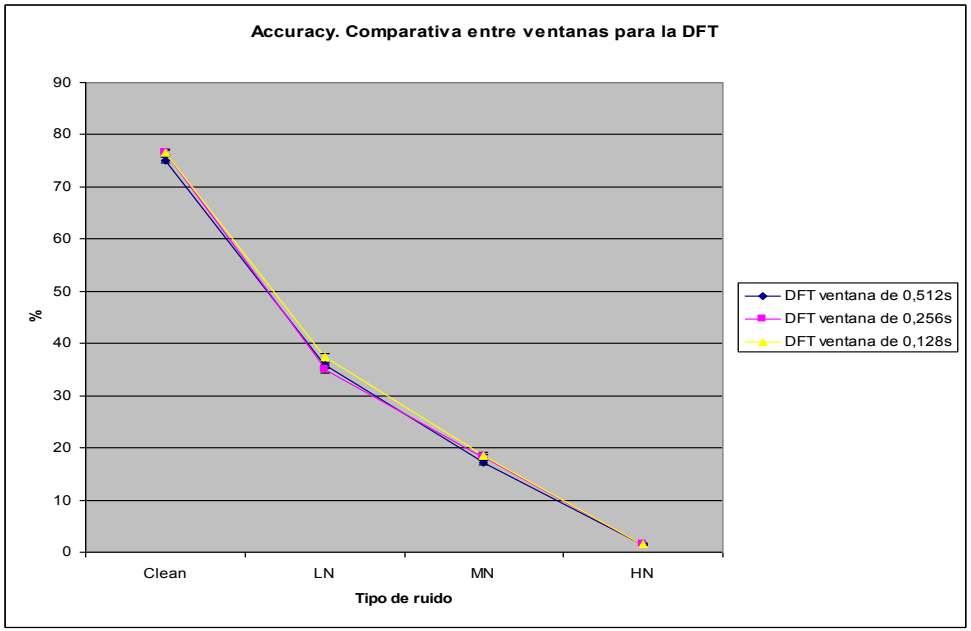

Figura 31: Comparativa entre longitudes de ventanas de análisis.

Aunque las diferencias son pequeñas (entorno al 4%). Los resultados nos indicaron que obtenemos mejores prestaciones para ventanas mas pequeñas. Esto seguramente sea debido a que en un entorno relativamente pequeño como es una cabina de avión, el tiempo re reverberación no es muy elevado por lo que los efectos a corto plazo son mas importantes que los efectos a largo plazo de manera que aumentar la longitud de la ventana de análisis no comporta mayores beneficios. No obstante los resultados debieran ser distintos en caso de que el efecto de la reverberación fuese mayor, tendiendo a mejorar al aumentar la longitud de la ventana cuanto más elevado fuera el tiempo de reverberación T_{60} .

Las siguientes tablas muestran los resultados obtenidos al incluir el uso del VAD en la técnica de substracción espectral:

Clean	Palabras correctas(%)	Susbstit(%)	Elimin(%)	Inserciones(%)	ErrorPalabra(%)	ErrorFrase(%)	Precisión
Francés	82,07	16,8	1,14	0,73	18,66	42,59	81,33
Griego	77,28	21,05	1,67	2,9	25,62	54,25	74,38
Italiano	79,56	19,23	1,21	2,92	23,36	48,81	76,64
Español	79,66	18,11	2,23	0,89	21,23	47,15	78,77
Sum/Avg	79,9	18,61	1,42	1,82	21,85	47,56	78,14

Tabla 27: Substracción espectral usando VAD: Sin ruido.

LN	Palabras correctas(%)	Susbstit(%)	Elimin(%)	Inserciones(%)	ErrorPalabra(%)	ErrorFrase(%)	Precisión
Francés	29,23	36,63	34,14	0,54	71,3	82,41	28,69
Griego	40,29	36,86	22,85	1,12	60,83	82,09	39,17
Italiano	40,09	35,02	24,9	1,29	61,21	76,7	38,79
Español	44,95	35,47	19,59	0,98	56,04	75,88	43,96
Sum/Avg	36,57	36,14	27,27	0,92	64,34	80,12	35,64

Tabla 28: Substracción espectral usando VAD: Ruido bajo.

MN	Palabras correctas(%)	Susbstit(%)	Elimin(%)	Inserciones(%)	ErrorPalabra(%)	ErrorFrase(%)	Precisión
Francés	11,58	35,93	52,5	0,27	88,7	92,74	11,3
Griego	18,46	37,34	44,2	0,51	82,04	92,85	17,95
Italiano	23,22	36,06	40,72	0,53	77,31	86,64	22,69
Español	23,39	36,25	40,35	0,43	77,03	88,39	22,97
Sum/Avg	17,59	36,35	46,05	0,412	82,81	90,73	17,18

Tabla 29: Substracción espectral usando VAD: Ruido medio.

HN	Palabras correctas(%)	Susbstit(%)	Elimin(%)	Inserciones(%)	ErrorPalabra(%)	ErrorFrase(%)	Precisión
Francés	1,38	33,54	65,09	0	98,62	98,58	1,37
Griego	0,49	32,49	67,02	0	99,51	99,1	0,49
Italiano	2,34	34,09	63,57	0,05	97,71	97,93	2,29
Español	0,79	32,22	66,99	0	99,21	98,6	0,79
Sum/Avg	1,32	33,25	65,42	0,01	98,68	98,55	1,30

Tabla 30: Substracción espectral usando VAD: Ruido alto.

En la siguiente gráfica se comparan los resultados a la hora de aplicar o no aplicar el uso del VAD a esta técnica:

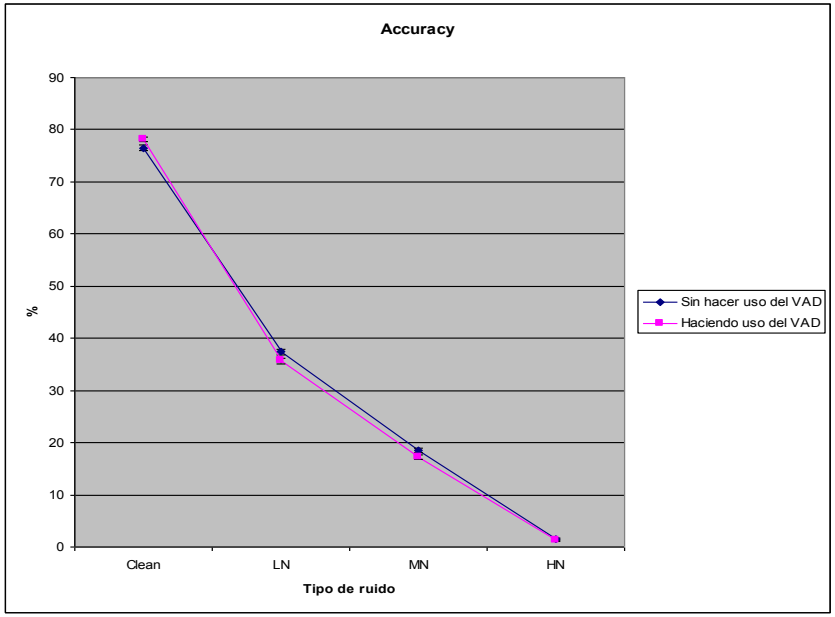

Figura 32: Uso del VAD en la substracción espectral.

El resultado es que no obtenemos mejora al aplicar esta técnica si no mas bien lo contrario, lo cual no nos extraña del todo tal y como comentamos en el apartado 4.4.1 y que se resume en el hecho de que para conseguir discriminación correcta en el VAD el software encargado de realizar la substracción espectral nos obliga a trabajar con ventanas pequeñas, con lo que estamos dejando de lado los efectos a largo plazo que son los más importantes en términos de reverberación. A pesar de estos resultados, no podemos afirmar con rotundidad que esta técnica no sea efectiva aunque si podemos concluir que para poder llegar a serlo sería necesario encontrar la manera de conseguir una buena discriminación en

el VAD sin que esto nos obligase a trabajar con ventanas excesivamente pequeñas, de manera que no perdiéramos los efectos a largo plazo del ruido por reverberación.

Por último vamos a ver una gráfica comparativa de todas las técnicas y variantes estudiadas hasta ahora:

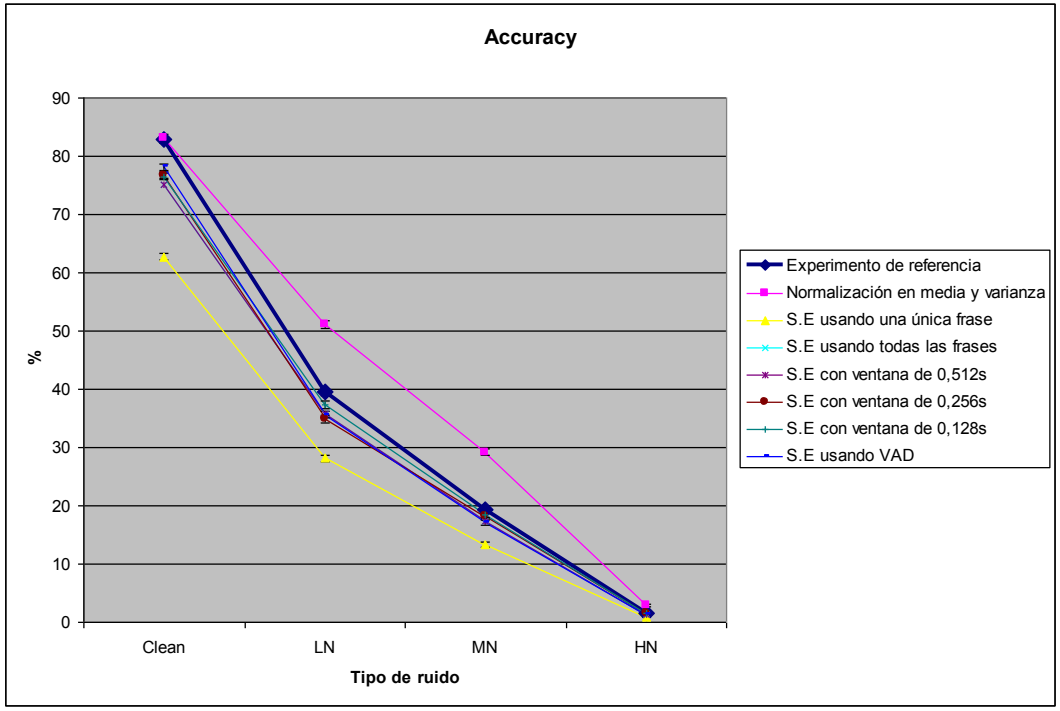

Figura 33: Comparativa final.

Como podemos observar al final con lo que hemos conseguido mejores resultados ha sido con la normalización en media y varianza mientras que con el uso de la técnica de substracción espectral no hemos conseguido unos resultados tan positivos. La consecuencia seguramente es que esa técnica esta más enfocada al ruido por reverberación, el cual pretendía ser uno de los principales objetos de estudio de este proyecto, pero el tipo de ruido preponderante en la base de datos no era de este tipo. Como explicamos al hablar de HIWIRE, los distintos niveles de ruido

presentes en la base de datos se obtienen a partir de la inclusión de ruido real presente en la cabina del avión durante un vuelo normal, pero este ruido no es ruido de reverberación ya que este se genera a partir de las diferentes reflexiones que puede sufrir la señal de voz desde que es emitida hasta que es capturada por el micrófono. Es decir, la reverberación se genera en el momento en el que se empieza a hablar mientras que el ruido introducido en HIWIRE procede de una grabación en cabina en la cual obviamente no estaban presentes las señales de voz de los hablantes de la base de datos ya que ambas cosas se grabaron por separado para sumarse posteriormente.

Eso no quiere decir que no exista ruido por reverberación, el problema es que al estar grabada toda la base de datos mediante micrófono corto, la incidencia del ruido por reverberación es mucho menor que la del ruido ambiente grabado en cabina por lo que las técnicas específicas para combatir ese tipo de ruido no se muestran demasiado eficaces en esta base de datos.

5. Conclusiones y líneas futuras

5.1. Conclusiones

En este proyecto fin de carrera hemos intentado analizar los principales problemas que se presentan en el entorno de las cabinas de avión para el reconocimiento de habla así como intentar encontrar y evaluar posibles soluciones a dichos problemas.

De todos ellos decidimos centrarnos principalmente en el ruido por reverberación ya que dada la fisonomía de las cabinas de los aviones sobre todo comerciales, es un tipo de ruido que siempre estará presente en mayor o menor medida, por lo que resulta interesante estudiar posibles soluciones al respecto dado que la tendencia actual es la de ir incorporando poco a poco sistemas de reconocimiento de habla dentro de la industria aeronáutica.

Este tipo de sistemas agiliza y facilita la interacción de la tripulación con el avión de manera que si se consiguen sistemas fiables, descargarán de trabajo a la tripulación en cuanto a manejo de paneles e instrumentación lo que potencialmente puede redundar a un menor nivel de estrés y ello podría llegar a reducir el número de errores asociados al factor humano con lo que se incrementaría la seguridad en los vuelos.

Como material de trabajo dispusimos de una parte de la base de datos HIWIRE la cual no obstante no posee suficiente material de audio como para poder ser usada en la fase de entrenamiento, por lo que para esa parte nos apoyamos en la base de datos TIMIT.

En primer lugar decidimos probar con la normalización de media y varianza para cada uno de los hablantes y posteriormente experimentamos con la técnica de substracción espectral para la cual probamos diversas variantes en torno al tiempo de análisis y ventana de trabajo. Posteriormente a dicha técnica le añadimos el uso de un VAD con la idea de intentar mejorar la estimación del espectro reverberante. Las principales conclusiones que pudimos extraer son las siguientes:

- Respecto al experimento de referencia pudimos comprobar que existe una fuerte dependencia en las prestaciones del sistema con la lengua nativa de los hablantes lo cual debe ser tenido muy en cuenta en la aplicación práctica de este sistema.

- La aplicación de la normalización de media y varianza se mostró muy efectiva en cuanto a la mejora de las prestaciones de sistema. No obstante hay que tener en cuenta que para poder aplicarla hemos tenido que calcular de manera offline los vectores de medias y varianzas lo cual podría suponer una limitación en la aplicación práctica del sistema.

- Respecto a la aplicación de la técnica de substracción espectral pudimos comprobar que para conseguir mejores resultados necesitábamos ampliar el tiempo de análisis hasta mas allá de la duración de un solo fichero a base de concatenar varios, lo cual era posible gracias a que las condiciones en las

que estaba grabados eran idénticas para todos los ficheros de un mismo hablante. La ventana de trabajo para la que obtuvimos mejore resultados fue para la de 0.128 segundos lo cual concluimos que .es debido a que es la longitud de ventana que mejor se ajusta al tiempo de reverberación correspondiente a la cabina del avión. En otro tipo de escenario en el cual el tiempo de reverberación fuese más elevado seguramente una ventana mas grande hubiera funcionado mejor.

- En lo que respecta a la aplicación de un VAD como complemento a la técnica de substracción espectral, no conseguimos mejora alguna de los resultados. Esto seguramente es debido a que al tener que usar el VAD nos vemos obligados a trabajar con ventanas mucho más pequeñas de las que son recomendables cuando se pretenden combatir los efectos de la reverberación ya que estos son siempre a largo plazo. No consideramos que sea una limitación de la técnica en sí, si no de la manera en la cual tuvimos que implementarla. Por lo que pensamos que, con las modificaciones adecuadas, seguramente sea posible obtener mejores resultados con esta técnica.

5.2. Líneas futuras

Tras la finalización del proyecto aún quedan líneas abiertas sobre las cuales puede profundizarse:

- Una técnica que aunque pudimos estudiar, al final no pudimos llegar a implementar es la normalización del tracto vocal (VTLN). Al final del apéndice A se comentan brevemente los pasos que habrían de seguirse para poder implementar esta técnica.

- Otra interesante variación consistiría en probar con otro tipo de parametrización diferente a la utilizada en este proyecto. Concretamente existen estudios en los que se indica que parámetros RASTA resultan particularmente interesantes en los sistemas que sufren ruido por reverberación [8] y de hecho las herramientas de SPRACHcore permiten trabajar con estos parámetros.

- Al hilo de una de las conclusiones de este proyecto, sería muy interesante contemplar la posibilidad realizar una adaptación a la lengua nativa del hablante ya que los resultados de los experimentos han demostrado que ésta es un parámetro fundamental en el rendimiento del sistema de reconocimiento.

- Respecto a la inclusión del VAD como apoyo a la substracción espectral sería interesante encontrar una manera efectiva de ser capaces de trabajar con ventanas de mayor duración sin perder discriminación en el VAD.

- De cara a esta base de datos, y dado que inicialmente decidimos poner nuestros esfuerzos en combatir el ruido por reberveración, no hicimos hincapié en incluir sistemas de cancelación de ruido mas convencionales. Dada la naturaleza del ruido presente en la base de datos, sería interesante combinar las técnicas probadas con alguna técnica de cancelación de ruido como pudiera ser el filtrado wienner.

- Otra línea interesante para poder seguir trabajando en el tema del ruido por reverberación sería el disponer de la segunda parte de la base de datos HIWIRE en la cual se pueda disponer de audio grabado con múltiples micrófonos ya que tal y como se ha comentado en la sección 3.6, la mayoría de los esfuerzos por mejorar las prestaciones en este tipo de sistemas hoy día pasan por hacer uso de técnicas multimicrófono que con la parte de HIWIRE de la que disponemos fue imposible evaluar al estar grabada con un único micrófono.

6. Apéndice A

6.1. HTK

HTK es el software de reconocimiento en el que se basaron los experimentos tal y como hemos visto, la versión que utilizamos fue la 3.4. La mayoría de la funcionalidad de HTK se divide en distintos módulos. Gracias a esta arquitectura modular se garantiza que las interfaces de cada una de las herramientas hacia el exterior funcionan de la misma manera, lo que facilita el manejo del software y además proporciona una estructura central de funciones de uso común.

La siguiente figura muestra la estructura típica de software de HTK indicando las interfaces de entrada/salida entre los distintos módulos [4]:

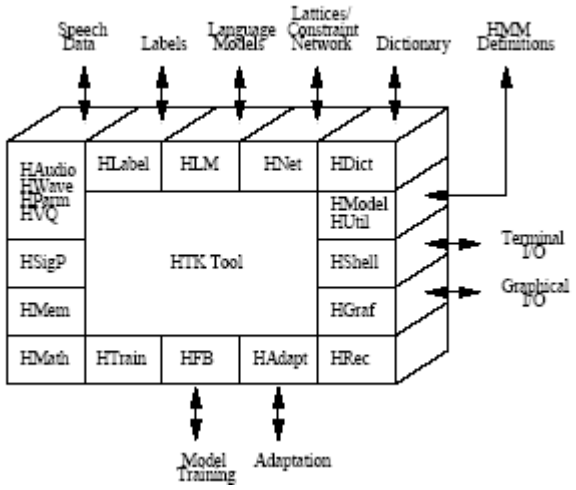

Figura 34: Estructura de HTK

La funcionalidad de los distintos módulos puede resumirse de la siguiente manera:

HSELL controla la interacción entrada tanto con el usuario como con el sistema operativo.

MEM se encarga de toda la gestión de memoria.

HMath proporciona todo el apoyo matemático.

HSigP se encarga de todas las operaciones de procesado necesarias para el análisis de habla.

HLabel proporciona la interfaz para los ficheros de etiquetas. Soporta múltiples tipos de formatos permitiendo importar información desde otros sistemas.

HLM proporciona la interfaz con los ficheros de modelos de lenguaje.

HNet es la interfaz para redes.

HDict es a su vez la interfaz para los diccionarios.

HVQ se encarga los libros de codificación VQ.

HModel se ocupa de las definiciones de los HMM.

HWave se encarga de gestionar tanto la entrada de audio como de la salida. Al igual que HLabel, soporta múltiples formatos de entrada.

HAudio se encarga de soportar la entrada directa de audio.

HGraf proporciona una interfaz sencilla de gráficos interactivos.

HUtil ofrece una gran cantidad de rutinas para manipular los HMM's.

HTrain y HFB contienen todas las herramientas para el entrenamiento de modelos que tiene HTK.

HAdapt proporciona soporte para las diferentes herramientas de adaptación que posee HTK.

HRec contiene las principales funciones para realizar las tareas de reconocimiento.

El ajuste de todos estos módulos se realiza ajustando las diferentes variables de control contenidas en los correspondientes ficheros de configuración.

Las herramientas de HTK están diseñadas para ser usadas por medio de la tradicional línea de comandos. Cada herramienta puede tener varias opciones pudiendo llevar cada una de ellas uno o varios argumentos adicionales, la letra que representa a una determinada opción esta precedida por un guión de manera que una invocación típica de una herramienta HTK tendría un formato similar a este:

```
HFoo -T 1 -f 34.3 -a -s myfile file1 file2
```

La herramienta HFoo tendría en este caso dos argumentos principales llamados file1 y file2 y otros cuatro opcionales que como hemos dicho serían aquellos que van precedidos por un guión, en este ejemplo serían –T, -f, -a y –s. Adicionalmente algunas de estas opciones tienen argumentos los cuales siempre van detrás de la letra que identifica la opción y separados por un espacio. Así, la opción –T tendría como argumento el entero 1 y la opción –f tendría 34.3.

Adicionalmente a los argumentos indicados explícitamente mediante línea de comandos, el funcionamiento de una determinada herramienta puede controlarse mediante parámetros incluidos en un fichero de configuración como en el siguiente ejemplo:

```
HFoo -C config -f 34.3 -a -s myfile file1 file2
```

En este caso la opción –C tiene como argumento el nombre de un fichero que contiene varios parámetros que servirán para configurar la herramienta. Es posible usar varios ficheros de configuración usando repetidas veces la opción –C.

Aunque podría parecer que este manejo de las herramientas por línea de comandos en vez de usar algún tipo de interfaz gráfica es un poco anticuado, tiene numerosas ventajas como por ejemplo que es posible escribir shell scripts que nos permitan ejecutar de manera automática todos y cada uno de los diferentes pasos que son necesarios cuando se realizan experimentos a gran escala.

6.1.1. Funcionamiento

El uso de las herramientas de HTK típicamente se agrupa en varios pasos que podríamos agrupar en cuatro fases principales: Preparación de datos, entrenamiento, fase de test y análisis. Esta distribución esquemática puede representarse mediante la siguiente figura:

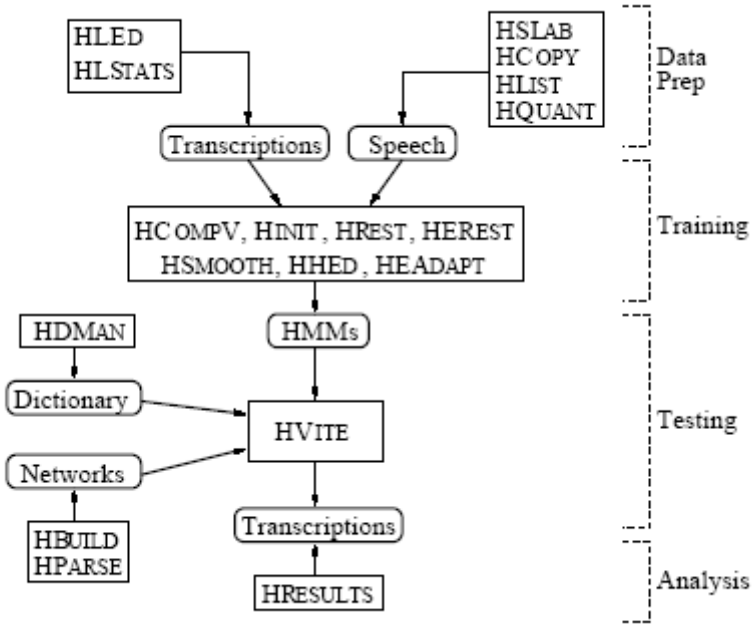

Figura 35: Fases del análisis con HTK

A continuación describimos cada una de las fases así como las principales herramientas involucradas en cada una de las etapas.

6.1.2. Preparación de datos

Para poder construir un grupo de HMMs es necesario tener varios ficheros con material de audio con sus correspondientes transcripciones.

Normalmente el material de audio se obtiene de bases de datos típicamente almacenadas en soportes físicos como DVDs. Ese es el caso particular de nuestro proyecto.

Pero antes de poder usar ese material de audio, es necesario parametrizarlos adecuadamente así como las transcripciones deben ser convertidas a ficheros de etiquetas adecuados.

HTK puede parametrizar las formas de onda *al vuelo* aunque lo más habitual y lo más eficiente en la práctica es parametrizarlas únicamente una vez. Esa es la técnica usada en nuestro caso ya que trabajamos con una base de datos grabada previamente.

La herramienta que usamos para realizar la parametrización es HCopy y para ello es necesario preparar un fichero de configuración en el cual se indican los diferentes parámetros necesarios como por ejemplo el formato de audio de entrada, el tipo de parametrización que queremos y las características de la misma.

Otras herramientas útiles en la fase de preparación son HList que aparte de ser la que nos permitiría convertir los datos de entrada *al vuelo*, puede ser usada para comprobar los resultados de cualquier conversión antes de comenzar a procesar grandes cantidades de datos. HLEd es otra herramienta útil en esta fase ya que nos permite realizar transformaciones en los ficheros de etiquetas lo cual es útil ya que típicamente las etiquetas usadas en las transcripciones originales pueden no ser del todo adecuadas ya que podrían por ejemplo usar un conjunto de fonemas distinto. Finalmente HLStats puede manejar y mostrar estadísticas de los ficheros de etiquetas

6.1.3. Herramientas de entrenamiento

Una vez hemos preparado los datos el siguiente paso es definir la topología de cada uno de los HMMs escribiendo la definición de un prototipo. HTK permite construir los HMMs con cualquier topología que deseemos, las definiciones de los HMM se almacenan en ficheros de texto por lo que luego pueden ser editados en caso de ser necesario. Así mismo, la distribución estándar de HTK incluye varios ejemplos de prototipos y scripts para generar las topologías más comunes de manera automática.

A excepción de las probabilidades de transición, el valor de todos los parámetros dados en la definición de los prototipos se ignoran. El objetivo de un prototipo es únicamente especificar las características generales y la topología de un determinado HMM. El valor de estos parámetros se calcula más tarde mediante las herramientas de entrenamiento.

Las etapas de la fase de entrenamiento se ilustran bien en la siguiente figura:

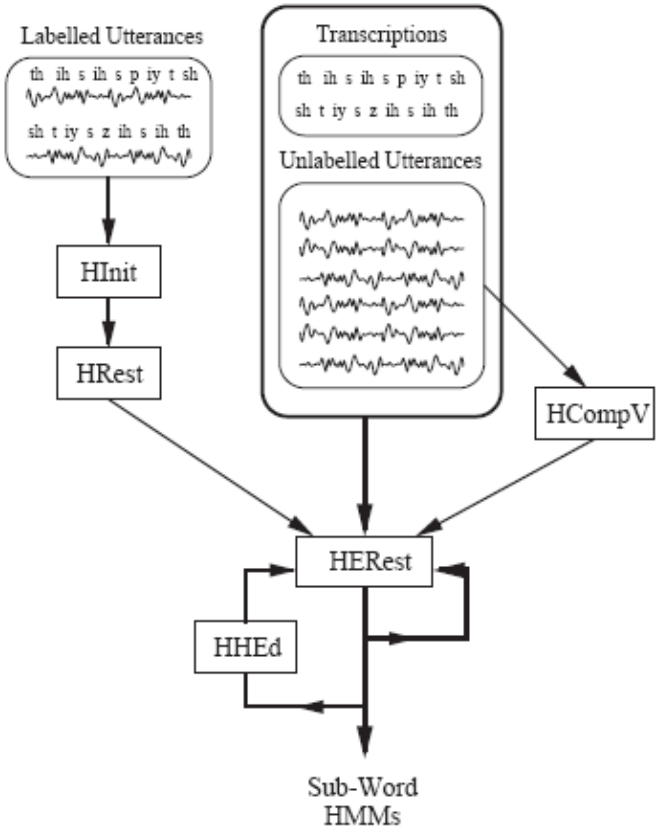

Figura 36: Fases del entrenamiento

En primer lugar hay que crear un conjunto inicial de modelos. Si existe material de habla para el cual los límites de los fonemas están bien marcados, es posible usar ese material como *datos de arranque*. En este caso, las herramientas HInit y HRest proporcionan un modelo de entrenamiento de *palabras sueltas* usando todo el material etiquetado de los datos de arranque. Cada uno de los HMM se genera de manera individual. HInit lee todo el material de los datos de arranque y separa todas las muestras de un determinado fonema, entonces calcula de manera iterativa un conjunto de valores iniciales de los parámetros usando un método k-medias segmental. En el primer ciclo se segmenta el material de entrenamiento de manera uniforme, cada modelo se empareja con el correspondiente segmento de datos y se estiman entonces las medias y las varianzas.

En nuestro caso, usamos modelos de mezcla de gaussianas y la principal diferencia con el método anterior es que en este caso se usa una versión modificada de k-medias con clustering.

En el segundo y sucesivos ciclos, la segmentación uniforme se substituye por el alineamiento de Viterbi. Los valores iniciales de los parámetros los calcula HInit y posteriormente son reestimados por HRest. Nuevamente se usan datos de arranque completamente etiquetados pero substituyendo el método k-medias segmental por la reestimación Baum-Welch.

Cuando no poseemos datos de arranque puede usarse un arranque tipo *flat start* en el cual los modelos de los fonemas se inicializan de forma que sean idénticos y los estados tengan medias y varianzas iguales a la media y varianza globales de los datos de habla. La herramienta que nos permite realizar este tipo de arranque es HCompV.

Una vez que se ha creado el conjunto inicial de modelos, la herramienta HRest se utiliza para realizar el entrenamiento sobre el total de datos de entrenamiento disponibles. En nuestro caso, para las labores de entrenamiento usamos una base de datos distinta de HIWIRE ya que esta no disponía de material suficiente para esta tarea. En su lugar usamos la base de datos TIMIT. HRest realiza la reestimación Baum-Welch para todo el conjunto de HMM de manera simultánea. Para cada muestra de entrenamiento, los correspondientes modelos de fonemas se concatenan y se ejecuta el algoritmo forward-backward para obtener todas las estadísticas necesarias para cada HMM. Una vez que se ha procesado todo el material de entrenamiento, el conjunto de estadísticas obtenidas se utiliza para re-estimar los parámetros de los HMM.

La filosofía de HTK es que los HMM se vayan refinando de manera incremental, de manera que se vayan incluyendo un mayor número de mezcla de gaussianas de manera iterativa.

6.1.4. Herramientas de reconocimiento

HTK dispone de una herramienta de reconocimiento llamada HVite que permite realizar reconocimiento usando modelos de lenguaje y mallas. HLRecsore es una herramienta que permite manipular las mallas generadas por HVite para por ejemplo aplicar un modelo de lenguaje más complejo. Existe un reconocedor adicional que sirve como extensión de HTK llamado HDecode pero no hicimos uso de él por lo que no profundizaremos en ello.

HVite usa el algoritmo *token passing* para realizar el reconocimiento de habla basado en el algoritmo de Viterbi. HVite coge como entrada una red que describe las secuencias de palabras permitidas, un diccionario que defina como se pronuncia cada una de las palabras y un conjunto de HMM. Convierte entonces la red de palabras en una red de fonemas y después adjunta la definición apropiada de HMM a cada instancia de fonema. El reconocimiento puede realizarse tanto a partir de una lista de ficheros con material de habla como de audio directo, en nuestro caso al usar una base de datos como fuente de información, nos situamos en el primer caso.

Las redes de palabras necesarias para usar HVite normalmente son simples lazos en los que cada palabra puede ir a continuación de cualquiera o bien pueden ser grafos dirigidos que representen una gramática de estados finitos. Las redes de palabras se almacenan en HTK usando el formato estándar para mallas. De esta manera pueden ser modificadas mediante un simple editor de texto, no obstante puede resultar un método de edición algo tedioso por lo que HTK dispone de dos herramientas de apoyo para crear redes de palabras. De esta manera HBuild permite crear sub-redes y usarlas en redes de más alto nivel y aunque se utiliza la misma notación a bajo nivel, se evita mucha duplicación, ademas HBuild puede modificar las transiciones de los lazos de palabras para incorporar las probabilidades de los bigramas.

Como alternativa a especificar la red de palabras de forma directa, se puede usar una notación de gramática de un nivel más alto. Esta notación esta basada en EBNF[27]. Más tarde la herramienta HParse se encarga de convertir la descripción en EBNF a la red de palabras equivalente.

Otra herramienta relacionada con esta fase de reconocimiento es HLRescore que permite la manipulación de mallas como por ejemplo las generadas por HVite y realizar operaciones sobre ellas como por ejemplo calcular las estadísticas de la malla o convertir las mallas a redes de palabras equivalentes.

Por último, cabe mencionar la herramienta HDMan que permite gestionar todo lo relacionado con los diccionarios como por ejemplo la creación de grandes diccionarios a partir de la unión de numerosas fuentes.

Una vez que hemos construido el reconocedor basado en HMMs, hay que evaluar su rendimiento. Normalmente esto se hace comparando la salida del reconocedor con las correctas transcripciones del material de habla con el que se esta trabajando. Esta comparación se lleva a cabo mediante la herramienta HResult.

Una vez que lancemos la herramienta HResult obtendremos el número de errores de substitución (S), errores de eliminación (D) y errores de inserción (I).

Un error de substitución se produce cuando el reconocedor decodifica una palabra y ésta no se corresponden con el valor correcto, un error de eliminación se produce cuando eliminamos una palabra y un error de inserción se produce cuando insertamos una o varias palabras de más en la decodificación.

A partir de esos datos podemos obtener dos parámetros de evaluación adicionales como el porcentaje de acierto y la precisión (*accuracy)*:

$$\text{Percent Correct} = \frac{N - D - S}{N} \times 100\%$$

$$\text{Percent Accuracy} = \frac{N - D - S - I}{N} \times 100\%$$

Siendo *N* el número total de etiquetas presentes en las transcripciones de referencia, que en nuestro caso es el número de palabras.

Dependiendo de cómo configuremos HResult (mediante un fichero de configuración como se explicó anteriormente), podemos obtener los resultados en dos tipos de formatos distintos:

[27] Extended Backus Naur Form

```
====================== HTK Results Analysis =======================
  Date: Sat Sep  2 14:14:22 1995
  Ref : refs
  Rec : results
------------------------ Overall Results --------------------------
SENT: %Correct=98.50 [H=197, S=3, N=200]
WORD: %Corr=99.77, Acc=99.65 [H=853, D=1, S=1, I=1, N=855]
===================================================================
```

En este formato la primera parte nos da información acerca de la hora y el nombre de los ficheros usados. La línea etiquetada como SENT nos indica el número total de frases que fueron decodificadas de manera correcta mientras que la que se etiqueta como WORD nos da las estadísticas para las palabras de manera individual.

Pero también podemos obtener los resultados en formato NIST[28]:

```
,-----------------------------------------------------------.
| HTK Results Analysis at Sat Sep  2 14:42:06 1995          |
| Ref: refs                                                 |
| Rec: results                                              |
|===========================================================|
|           # Snt |  Corr    Sub    Del    Ins    Err  S. Err |
|-----------------------------------------------------------|
| Sum/Avg |  200  |  99.77   0.12   0.12   0.12   0.35   1.50 |
`-----------------------------------------------------------'
```

La siguiente figura representa de manera esquemática este proceso de reconocimiento:

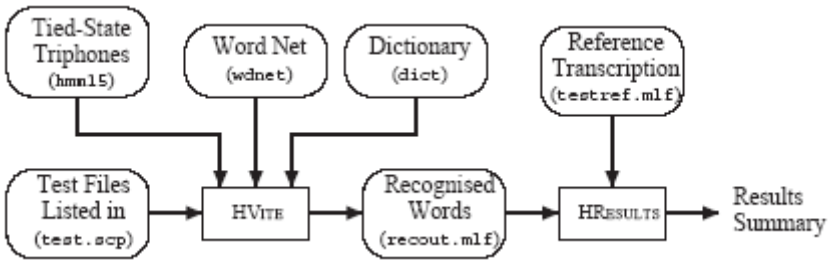

Figura 37: Fases del reconocimiento

[28] National Institute of Standars and Technology

6.1.5. Pasos para la implementación de VTLN en HTK

En la sección 3.5 comentamos que no pudimos llegar a implementar el VTLN en nuestro proyecto, aún así aquí mencionaremos los pasos que serían necesarios para aplicar esta técnica así como una interesante variación de la misma que resultaría útil para dispositivos con poca capacidad de procesado.

Lo primero sería calcular el factor de escalado óptimo para cada uno de los hablantes de la base de datos. Existen varias maneras de estimar los parámetros de la función de escalado, por ejemplo estimación mediante un decisor de máxima verosimilitud (ML) tal como se describe en en el cual es necesario calcular el Jacobiano de la transformación aunque es posible simplificar el cálculo obviando el término del Jacobiano y usar solo la distancia.

Otra posibilidad sería usar un análisis HDA[29] descrito en [36]. También es posible usar un sistema discriminativo estándar como la máxima información mutua (MMI), pero en este último caso si la transformación no resulta acotada se ha demostrado que es necesario el uso de interpolación con la matriz ML estimada para resultar útil .

Con los diferentes parámetros calculados, habría que generar con cada uno de ellos un fichero de configuración para HCopy que nos serviría para poder realizar la parametrización de nuestros ficheros de audio, típicamente siguiendo un script programado específicamente para dicha labor.

[29] Heteroscedastic discriminant analysis

7. Apéndice B

7.1. SPRACHcore

En la realización del proyecto fue necesario hacer uso en determinados momentos del paquete software SPRACHcore [22]. El software de SPRACHcore proviene del proyecto SPRACH el cual nace de la colaboración entre ICSI[30], la Univerisdad de Cambridge, la Univerisdad de Sheffield, la Faculte Polytechnique de Mons y el INESC de Lisboa.

SPRACH es el acrónimo de *SPeech Recognition Algorithms for Connectionist Hybrids,* se trata de un proyecto dedicado a la investigación de sistemas de reconocimiento híbridos HMM/ANN[31] estudiados anteriormente en el proyecto WERNICKE.

El paquete software SPRACHcore no se ha actualizado como tal desde 2004, en su lugar se han ido lanzando nuevas versiones de sus componentes pero de manera individual. El paquete incluye tanto herramientas de reconocimiento basadas en redes neuronales como herramientas de extracción de características. En el paquete podemos encontrar el código fuente de todas y cada una de las herramientas, las cuales están diseñadas para poder ser utilizadas en sistemas Unix.

Existen dos variantes principales del paquete SPRACHcore:

SPRACHcore-nogui: Incluye todas las herramientas de línea de comandos sin incluir toda la interfaz gráfica. La razón de no incluir la interfaz gráfica es que es la parte que presenta mayores problemas en la instalación y a su vez es la más prescindible. Esta fue la versión con la que decidimos trabajar en el proyecto.

SPRACHcore: Esta versión incluye todas las herramientas de SPRACHcore-nogui añadiendo además toda la funcionalidad gráfica.

El contenido de SPRACHcore-nogui es:

intvec - Librería de manejo de vectores de enteros.

fltvec - Librería de manejo de vectores de coma flotante.

quicknet - Entrenamiento y evaluación de redes neuronales (libreria y programas)

rasta - Cálculo de parametros rasta.

---

[30] International Computer Science Institute (Berkley).

[31] ANN: Artificial Neural Network.

dpwelib - Interfaz E/S universal de archivos de audio.

feacalc - Cálculo de parámetros como RASTA, PLP y otros (usa el código de rasta como libreria)

feacat - Utilidad de converisón de tipos de datos.

pfile_utils - Herramientas de transformación para archivos pfile

bedk_frontend - Herramienta relacionada con MSG (Modulation-filtered SpectroGram) [32]

ffwd - forward-pass pkg para redes neuronales.

labels2pfile - Script tcl para la conversión entre formatos de etiquetas.

simplebn

dr_scripts - Scripts Perl para el entrenamiento y alineado(El alineado requiere Softsound's efsgd, no incluido)

randlines - Herramienta para la aleatorización de líneas en ficheros de texto.

noway - Decodificador a posteriori basado en HMM de Steve Renals

simpleui - Demostración simple usando audio_fe, rasta, quicknet

El paquete completo SPRACHcore incluye además las siguientes herramientas:

aprl

guitools

tclsh

libdat

otcl

farray_otcl

pfif_otcl

sound_otcl

gdtcl

dpweutils_tcl

dpwetcl

audio_frontend

berpbackend

berpdemo

sprachdemo

thislgui

Del paquete SPRACHcore-nogui finalmente solo necesitamos hacer uso de la librería dpwelib necesaria para poder compilar la libreria fxt relacionada con la herramienta de substracción espectral tal y como se comenta en 4.4.

En un principio intentamos también trabajar con las herramientas feacat y feacalc para resolver el problema que se nos presentaba en el número de bits por muestra cuando trabajábamos con meansub.c, pero finalmente optamos por resolver dicho problema mediante procesado con MATLAB.

7.2. Toolbox VOICEBOX

Para solventar los problemas comentados en la sección 4.4 referentes al número de bits por muestra, recurrimos a herramientas de esta toolbox [24] para desarrollar programas con los que conseguir nuestro propósito.

VOICEBOX es una toolbox de procesado de habla basada en programas MATLAB mantenidos y escritos en su mayoría por Mike Brookes, Department of Electrical & Electronic Engineering, Imperial College, Exhibition Road, London SW7 2BT, UK.

La mayoría de los programas requieren una versión de MATLAB 6.5 o superior siendo necesaria algun tipo de pequeña modificación si quiere usarse en versiones previas. En la realización del proyecto usamos MATLAB 7.0 y no tuvimos nigún tipo de problema en el uso de las mismas.

8. Presupuesto

En este apéndice se presentan justificados los costes globales de la realización de este Proyecto Fin de Carrera. Tales costes, imputables a gastos de personal y de material, se pueden deducir de las tablas 31 y 32.

En la tabla 31 se muestran las fases del proyecto y el tiempo aproximado para cada una de ellas. Así pues, se desprende que el tiempo total dedicado por el proyectando ha sido de 1120 horas, de las cuales aproximadamente un 10% han sido compartidas con el tutor del proyecto, por lo que el total asciende a 1230 horas. Teniendo en cuenta que la tabla de honorarios del Colegio Oficial de Ingenieros de Telecomunicación establece unas tarifas de 78 €/hora, el coste de personal se sitúa en 96096 €.

Fases	**Horas**
Documentación	250
Desarrollo del Software	200
Experimentación y toma de resultados	400
Redacción de la memoria del proyecto	270

Tabla 31: Fases del proyecto

En la tabla 32 se recogen los costes de material desglosados en equipo informático, documentación y gastos varios no atribuibles (material fungible, llamadas telefónicas, desplazamientos...). Ascienden, pues, a un total de 1650 €.

Concepto	**Importe(€)**
Ordenador de sobremesa	900
Documentación	150
Gastos varios	650

Tabla 32: Costes de material

A partir de estos datos, el presupuesto total es el mostrado en la tabla 33..

Concepto	**Importe(€)**
Gastos personales	96096
Gastos de material	1650
Base imponible	97746
I.V.A (16%)	15639.36
Total	113385.36

Tabla 33: Presupuesto

9. Referencias

[1] J.C. Segura, et al " The HIWIRE database, a noisy and non-native English speech corpus for cockpit communication", Dept. of TSTC, ETSIIT, Univ. of Granada, Spain

[2] Rabiner, Lawrence. “Fundamentals of speech recognition” Englewood Cliffs: Prentice-Hall , 1993

[3] Gold, Ben. “Speech and audio signal processing : processing and perception of speech and music” John Wiley & Sons, 2000.

[4] Steve Young, Gunnar Evermann,Mark Gales,Thomas Hain,Dan Kershaw,Xunying (Andrew) Liu,Gareth Moore,Julian Odell,Dave Ollason,Dan Povey,Valtcho Valtchev,Phil Woodland. “The HTK Book” Revisado para la versión 3.4 de HTK, Diciembre de 2006.

[5] Junqua, Jean-Claude. “Robustness in automatic speech recognition : fundamentals and applications” Kluwer Academics 1996.

[6] Gelbart, D.; Morgan, N.; , "Evaluating long-term spectral subtraction for reverberant ASR," *Automatic Speech Recognition and Understanding, 2001. ASRU '01. IEEE Workshop on* , vol., no., pp. 103- 106, 2001
URL: http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=1034598&isnumber=22205

[7] C. Avendano, S. Tibrewala, and H. Hermansky, “Multiresolution Channel Normalization for ASR in Reverberant Environments”, in *EUROSPEECH 1997*, Rhodes, Greece, 1997.

[8] Kingsbury, B.E.D.; Morgan, N. "Recognizing reverberant speech with RASTA-PLP". Acoustics, Speech, and Signal Processing, 1997. ICASSP-97., 1997 IEEE International Conference on 21-24 Apr 1997 On page(s): 1259 - 1262 vol.2. Meeting Date : 21 Apr 1997-24 Apr 1997.

[9] 3GPP TS 26.094: "AMR Speech Codec; Voice Activity Detection".

[10] Sundermann, D.; Bonafonte, A.; Ney, H.; Hoge, H., "Time domain vocal tract length normalization," Signal Processing and Information Technology, 2004. Proceedings of the Fourth IEEE International Symposium on , vol., no., pp. 191-194, 18-21 Dec. 2004
URL:http://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=1433719&isnumber=30910

[11] Donald R. McMonagle, "Advanced Fighter Technology Integration (Afti) F-16-The Pilot Interface". United States Air Force, Flight Test Center. October 1984.

[12] F-16 AFTI Advanced Fighter Tecnology Integration http://www.dfrc.nasa.gov/gallery/photo/F-16AFTI/index.html . Última modificación: 22 de Diciembre de 2004. Responsable NASA oficial: Marty Curry. Fecha de última visita 3/05/2010

[13] Eurofighter Typhoon Direct Voice Input: http://www.eurofighter.com/et_as_vt_dv.asp. Fecha de última visita: 3/05/2010

[14] Ejercito del aire, Eurofighter Typhoon. Ministerio de Defensa, Gobierno de España: http://www.ejercitodelaire.mde.es/ea/pag?idDoc=6485A0861301E048C12570D700463CE4&idRef=4D3582D4F64BAB7EC125745000327771. Fecha de última visita: 3/05/2010

[15] Wikipedia: Ejercito del Aire de España: http://es.wikipedia.org/wiki/Ej%C3%A9rcito_del_Aire_Espa%C3%B1ol#Avi.C3.B3n_Eurofighter_Typhoon. Fecha de última visita: 3/05/2010

[16] Christine England, "Speech recognition in the JAS 39 Gripen aircraft-adaptation to speech at diferent G-loads". Master Thesis in Speech Technology. Royal Institute of Technology, Department of speech, music and hearing. Estocolmo, 11 Marzo 2004.

[17] Joseph L.Komer, Joseph E.Gepner, Charles Gregory Sherwood. "Automatic Speech Recognition System and method for aircraft" United States Patent Application Publication. US 2010/00330400 A1. Feb. 4, 2010.

[18] EU-IST HIWIRE web site: http://www.hiwire.org Fecha de última visita: 3/05/2010

[19] TIMIT: http://www.ldc.upenn.edu/Catalog/readme_files/timit.readme.html Fecha de última visita: 3/05/2010

[20] HTK web-site http://htk.eng.cam.ac.uk/ Fecha de última visita: 3/05/2010

[21] Material para substracción espectral http://www.icsi.berkeley.edu/speech/papers/asru01-meansub-corr.html Fecha de última visita: 3/05/2010

[22] SPRACHcore http://www.icsi.berkeley.edu/~dpwe/projects/sprach/index.html. International Computer Science Institute, Berkeley CA. Última actualización: 14/05/1999 a las 00:37:04. Fecha de última visita: 3/05/2010

[23] SOX http://sox.sourceforge.net/ Fecha de última visita: 3/05/2010

[24] VOICEBOX http://www.ee.ic.ac.uk/hp/staff/dmb/voicebox/voicebox.html Fecha de última visita: 3/05/2010

[25] Miyoshi, M.; Kinoshita, K.; Yoshioka, T.; Nakatani, T., "Principles and applications of dereverberation for noisy and reverberant audio signals," *Signals, Systems and Computers, 2008 42nd Asilomar Conference on* , vol., no., pp.793-796, 26-29 Oct. 2008
URL: http://www.ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=5074518&isnumber=5074342

[26] Houtgast et al., 1980

[27] Gales,1995; Hirsch 2001a

[28] Joint Strike Fighter http://es.wikipedia.org/wiki/Lockheed_Martin_F-35_Lightning_II#Programa_Joint_Strike_Fighter. Última modificación: 8/04/2010 a las 00:57. Fecha de última visita: 3/05/2010

[29] Joint Strike Fighter http://www.jsf.mil/index.htm Fecha de última visita: 3/05/2010

[30] Dynaspeak http://www.speechatsri.com/products/dynaspeak.shtml Fecha de última visita: 3/05/2010

[31] Federal Aviation Administration, "Controller-Pilot Datalink Communications (CPDLC)", http://hf.tc.faa.gov/capabilities/cpdlc.htm U.S. Department of Transportation, Federal Aviation Administration Fecha de última visita: 3/05/2010

[32] Brian Kingsbury. ·Perceptually-inspired signal processing strategies for robust speech recognition in reverberant environments. PhD Dissertation". University of California at Berkeley, December 1998

[33] Hans-Günter Hirsch, Harald Finster, "A new approach for the adaptatin of HMMs to reverberation and background noise"Niederrhein Unibersity of Applied Sciences, Department of Electrical Engineering and Computer Science, Reinarzstr, 49 47805 Krefeld, Germany Sept 2007

[34] E J. Charpentier and M, G. StelIa, "Diphone Synthesis Using an Overlap-Add Technique for Speech Waveforms Concatenation," in Pmc. ofrhe ICASSP '86, Tokyo, Japan, 1986.

[35] M. J. F. Gales, "Maximum likelihood linear transformations for HMM-ased peech recognition," Computer Speech and Language, vol. 12, no. 2, pp. 75 – 98, Apr. 1998.

[36] N. Kumar and A. G. Andreou, "Heteroscedastic discriminant analysis and reduced rank HMMs for improved speech recognition," Speech Communication, vol. 26, no. 4, pp. 283–297, Dec. 1998.

[37] L. F. Uebel and P. C. Woodland, "Discriminative linear transforms for speaker adaptation," in ISCA ITRWorkshop on Adaptation Methods in Speech Recognition, Sophia Antipolis, France, Aug. 2001, pp. 61–64.

[38]

[39] "Token Passing: a Conceptual Model for Connected Speech Recognition Systems", SJ Young, NH Russell and JHS Thornton, CUED Technical Report F_INFENG/TR38, Cambridge University, 1989.

[40] Sehr, A.; Kellermann, W., "Strategies for modeling reverberant speech in the feature domain," Acoustics, Speech and Signal Processing, 2009. ICASSP 2009. IEEE International Conference on , vol., no., pp.3725-3728, 19-24 April 2009 URL:http://www.ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=4960436&isnumber=4959496

[41] Kondoz, A.M. "Digital speech : coding for low bit rate communication systems" 1996

[42] Jurafsky, Daniel S. "Speech and language processing : an introduction to natural language processing, computational linguistics, and speech recognition" 2000.

[43] Nakatani, T.; Juang, B.-H.; Yoshioka, T.; Kinoshita, K.; Delcroix, M.; Miyoshi, M., "Speech Dereverberation Based on Maximum-Likelihood Estimation With Time-Varying Gaussian Source Model," Audio, Speech, and Language Processing, IEEE Transactions on , vol.16, no.8, pp.1512-1527, Nov. 2008 URL: http://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=4648935&isnumber=4648209

[44] J.L Flanagan, "Computer-steered microphone arrays for sound transduction in large rooms," *J.Acoust. Soc. Amer.*, vol78 no.11 pp. 1508-1518,1985

[45] G. Elko, "Superdirective microphone arrays," in Acoustic Signal Processing for Telecommunication, S. Gay and J. Benesty, Eds. Norwell, MA: Kluwer, 2000, pp. 181–235.

[46] M. Miyoshi and Y. Kaneda, "Inverse filtering of room acoustics," IEEE Trans. Acoust., Speech, Signal Process., vol. 36, no. 2, pp. 145–152, Feb. 1988.

[47] T. Hikichi, M. Delcroix, and M. Miyoshi, "Inverse filtering for speech dereverberation less sensitive to noise and room transfer function fluctuations," EURASIP J. Adv. Signal Process., vol. 2007, 2007, article ID 34013.

[48] B. Gillespie, H. Malvar, and D. Florencio, "Speech dereverberation via maximum-Kurtosis subband adaptive filtering," in Proc. 2001 IEEE Int. Conf. coust., Speech, Signal Process. (ICASSP'01), 2001, vol. 6, pp. 3701–3704.

[49] Kinoshita, K.; Delcroix, M.; Nakatani, T.; Miyoshi, M., "Suppression of Late Reverberation Effect on Speech Signal Using Long-Term Multiple-step Linear Prediction," Audio, Speech, and Language Processing, IEEE Transactions on , vol.17, no.4, pp.534-545, May 2009 URL: http://www.ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=4787207&isnumber=4787206

[50] Kinoshita, K.; Nakatani, T.; Miyoshi, M., "Fast Estimation of a Precise Dereverberation Filter based on Speech Harmonicity," *Acoustics, Speech, and Signal Processing, 2005. Proceedings. (ICASSP '05). IEEE International Conference on* , vol.1, no., pp. 1073-1076, March 18-23, 2005
URL: http://www.ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=1415303&isnumber=30650