Revista Ingeniería, Investigación y Desarrollo, Vol. 16 Nº 1, pp. (26-38), 2016

PROTOTIPO DE SILLA DE RUEDAS COMANDADA POR VOZ EMPLEANDO HMM EN UN AMBIENTE CONTROLADO

(Prototype of wheelchair commanded by voice using HMM in a controlled environment)

Oscar Iván Higuera Martínez[1], Helberth Yesid Espitia Flórez[1], Diego F. Méndez Medina [2]
[1] Escuela de Ingeniería Electrónica Uptc. Grupo DSP-UPTC. hyeffe@yahoo.es, oscar.higuera@uptc.edu.co
[2] Escuela de Ingeniería Electrónica Uptc. EIS Ingeniería S.A.S, proyectos@eis-ing.com.

(Recibido junio 28 de 2015 y aceptado agosto 21 de 2015)

Resumen
Este artículo presenta el desarrollo de un sistema de reconocimiento de palabras aisladas independiente del locutor, para comandar una silla de ruedas. Cada palabra se codifica mediante las técnicas de Predicción lineal y Cepstrum real, y la etapa de clasificación se realiza mediante Modelos Ocultos de Markov (HMM). A partir de los resultados se generan órdenes a un sistema móvil (silla de ruedas), el cual es comandado en un ambiente controlado.

Palabras clave: Modelos Ocultos de Markov, procesamiento de voz, predicción lineal, Cepstrum.

Abstract
This article presents the development of a recognition system of speaker independent isolated words to command a wheelchair. Each word is encoded using techniques of Linear Prediction and Real Cepstrum, while the classification step is performed using Hidden Markov Models (HMM). From the results, commands are generated to a mobile system (wheelchair), which is led in a controlled environment.

Key words: Hidden Markov Models, Speech Processing, Linear prediction, Cepstrum.

1. INTRODUCCIÓN

El reconocimiento de voz ha despertado un inusitado interés, tanto en empresas tecnológicas como en universidades. Basta señalar una gran cantidad de diferentes productos de interacción por voz para entender la creciente necesidad: control de acceso, robots industriales, sistemas de ayuda a discapacitados, acceso y navegación por base de datos, operaciones y transacciones comerciales, etc.

El reconocimiento de voz es básicamente un proceso de clasificación de patrones (Faundez, 2001). El objetivo es adquirir un patrón de entrada, en este caso es la señal de voz, y clasificarla como adelante, atrás, izquierda, etc. Los patrones de entrada, pueden ser tratados como palabras, sílabas o fonemas. La principal dificultad del reconocimiento es que la señal de voz es muy variable, debido a la gran cantidad de locutores, diferentes velocidades a la hora de hablar, condiciones acústicas y ambientales, e incluso el estado anímico del locutor.

Un sistema de reconocimiento puede ser clasificado según sus características así: reconocimiento de palabras aisladas, identificación de palabras clave en un discurso continuo, reconocimiento de palabras conectadas, con dependencia o independencia del locutor, según el tamaño del vocabulario y si el reconocedor está en la capacidad de responder en presencia de ruido o solamente en un ambiente controlado. Los sistemas de reconocimiento dependientes del locutor, deben ser entrenados para responder a las características

particulares de la voz de una persona, es decir, para un solo locutor. Algunos trabajos en esta área son (Moralejo, 2010) (Alcubierre, 2005), (Cabas, 2004), (Gold, 2011), (Méndez, 2006), (Burton, 1987), (Rabiner, 1989), (Villamil, 2005), en los cuales le da aplicación al procesamiento de voz.

En este proyecto se da a conocer la experiencia en el desarrollo de un prototipo de reconocimiento de palabras aisladas dependiente del locutor. El prototipo fue desarrollado para ser utilizado en el desarrollo e implementación de una silla de ruedas comandada por voz, este prototipo busca una herramienta de ayuda para personas en situación de discapacidad y de esta forma lograr su independencia.

2. ESTRUCTURA DEL ALGORITMO DE RECONOCIMIENTO

La implementación del algoritmo se divide en tres tareas específicas (Rabiner, 1989), La realización del prototipo, se divide en tres etapas: entrenamiento del *codebook* o libro de códigos, entrenamiento del Modelo Oculto de Markov de cada palabra y reconocimiento de palabras aisladas.

2.1 Entrenamiento del codebook

El entrenamiento del *codebook* se realizó de la siguiente forma: adquisición y muestreo de la señal de voz, cuantificación, filtro pasa bajos, preprocesamiento, extracción de parámetros característicos y cálculo del *codebook*. Es necesario realizar una gran cantidad de repeticiones de cada palabra y almacenar los parámetros característicos de cada una en una matriz para el posterior cálculo de los centroides.

2.1.1 Adquisición. Consiste en captar mediante un micrófono la onda acústica producida por el locutor y digitalizarla para poderla tratar en un computador. Es importante destacar que tanto el micrófono usado como el lugar en el que se realiza la grabación pueden afectar las tasas de reconocimiento. Especialmente si no son los mismos en el proceso de entrenamiento (etapa de aprendizaje del sistema) y de reconocimiento (etapa de funcionamiento del sistema).

2.1.2 Muestreo. La herramienta de adquisición de datos de Matlab (Matlab Data Acquisition ToolBox versión 2.6) está orientada a aprovechar el *hardware* de adquisición de datos para medir y analizar fenómenos físicos.

El sistema de adquisición de datos es una colección de objetos de *software* y *hardware* que permite conectar el computador con el mundo físico. Según el *hardware* conectado, sea interno, externo o insertado dentro de la tarjeta madre, se tienen dentro del Daq Toolbox cuatro subsistemas:

- Entrada análoga (Analog Imput Subsystem)
- Salida análoga
- Entrada y salida digital
- Timer y contadores

El subsistema de entrada análoga tiene como función muestrear y cuantificar la señal análoga empleando uno o más canales. La señal análoga es continua en tiempo y amplitud, el muestreo toma pequeñas muestras de la señal en tiempos discretos, mientras que la cuantificación divide los valores de voltaje en amplitudes discretas.

El muestreo con el DaqToolbox 2.5 Analog Input. El sistema toma muestras del sensor, micrófono, a intervalos constantes. En muchos conversores análogo/digital, como los que se encuentran dentro de las tarjetas de sonido, el muestreo se realiza mediante un circuito de muestreo y retención (Sampling and Hold)

S/H. Usualmente un S/H consta de un *buffer* para la señal, seguido de un selector electrónico conectado a un capacitor. La operación de este sigue los siguientes pasos: en un instante de muestreo dado, el selector conecta el *buffer* y el capacitor a una entrada. El capacitor se carga al voltaje de la entrada. La carga se mantiene hasta que el conversor A/D digitaliza la señal. El proceso completo se repite para el siguiente instante de muestreo. Muy pocas tarjetas de sonido poseen conversores para cada canal y la mayoría multiplexa las entradas a un solo conversor.

2.1.3 Cuantificación. Cuando se realiza la toma, la señal análoga muestreada debe ser convertida a un valor binario que el computador pueda leer. La conversión de una amplitud infinitamente precisa a un número binario es llamada cuantificación. Durante la cuantificación, el conversor A/D emplea un número finito de valores convenientemente espaciados para representar la señal

análoga. El número de diferentes valores es determinado por el número de bits empleados para la conversión. Por defecto, el sistema de adquisición realiza conversión a 16 bit.

2.1.4 Filtro pasabajos. El ancho de banda útil de la voz es de 8 kHz, pero la información relevante está contenida hasta los 4 kHz aproximadamente. Se utiliza un filtro pasabajos con frecuencia de corte en 4 kHz y ganancia de 0 dBs, con el fin de eliminar la información en frecuencias superiores a los 4 kHz, con lo cual se ahorra tiempo de procesamiento, y se evita la producción de *aliasing*. De esta manera, se impide que las réplicas del espectro de la señal, que se producen como consecuencia del muestreo, se solapen ocasionando distorsión.

2.1.5 Preprocesamiento. *El preénfasis*: antes de realizar el procesamiento digital de la señal, es necesario realizar un filtrado paso alto de preénfasis con el objetivo de aumentar la energía relativa de las componentes de alta frecuencia en el espectro de la voz, y adicionalmente suavizar el espectro (Martínez, 2006). El filtro de preénfasis de primer orden con frecuencia de corte en 2500 Hz y ganancia de 6 dBs empleado tiene la forma:

$$H(z) = 1 - a(z) \quad (1)$$

Donde 0.9 < a< 1. La respectiva ecuación de diferencias de H (z) será:

$$S(n) = s(n) - a \times s(n-1) \quad (2)$$

Luego de hacer pruebas con la constante del filtro, se observó que el mejor funcionamiento se obtuvo en la mitad del intervalo recomendado, es decir:

(a = 0.95). El filtro de preénfasis también compensa la caída de 6dB que sufre la voz cuando pasa a través del tracto vocal. En la Figura 1 se muestra el espectro de frecuencia de la palabra “adelante” antes y después de ser preenfatizada. En ella se puede ver cómo después de aplicar el filtro de preénfasis los componentes de frecuencia mayores a 2500 Hz son enfatizados, mientras que los componentes de más baja frecuencia son atenuados.

Ventaneo: la señal de voz varía lentamente en el tiempo, si se analiza en períodos entre 5 y 100 ms., puede observarse que su dinámica es altamente estacionaria, sin embargo, para intervalos del orden de 100 ms. o más, las características de la señal de voz se modifican con cada palabra que es pronunciada. Como consecuencia, el análisis de voz se acostumbra a realizar en intervalos cortos de tiempo. Así, la señal se procesa por tramas, como si cada una de ellas tuviera propiedades estadísticas independientes.

En general, la aplicación de ventanas conduce al aislamiento y, por tanto, a la distorsión del espectro original. Para mantener la distorsión espectral dentro de un mínimo, se requiere el cumplimiento de las siguientes propiedades por parte de la ventana: lóbulo principal reducido, y pequeños o insignificantes lóbulos laterales. Estos dos requerimientos no pueden alcanzarse simultáneamente, por lo cual en la determinación de la ventana más adecuada debe realizarse un compromiso.

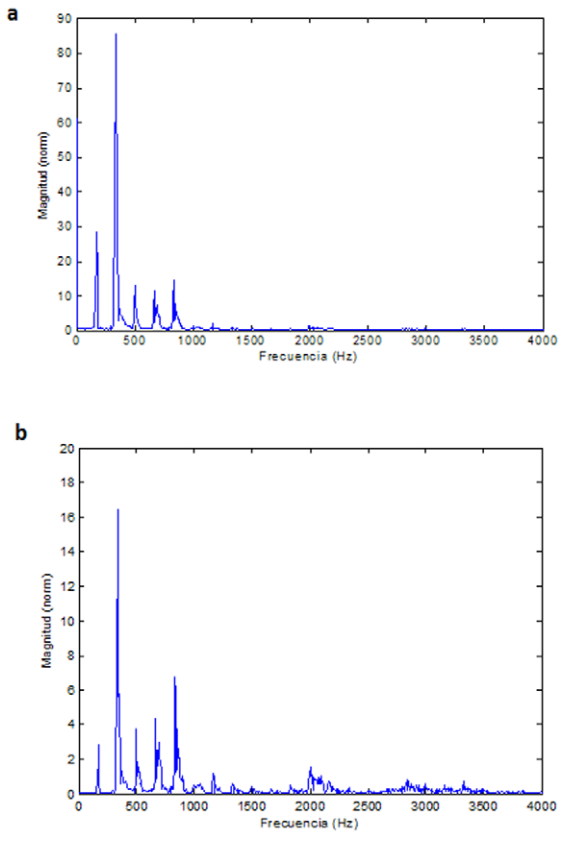

Figura 1. Espectro de frecuencia de la palabra “adelante”. (a). Antes del filtro, (b). Después del filtro de preénfasis.

Esta es la ventana que se usó para el análisis de señales de voz, y se define como (Saito, 1981), (Smith, 1999):

$$w(nT) = 0.54 - 0.46cos\left(2\left(\frac{n}{N}\right)\right) 0 \leq n \leq N \quad (3)$$

El análisis se hizo a partir de la multiplicación de nuestra señal de voz por la fórmula (3).

Segmentación: en el reconocimiento de señales de voz se hace necesario determinar con adecuada precisión los puntos de inicio y final de cada palabra, es decir, deben diferenciarse las partes de señal que llevan información de voz de aquellas que no llevan voz. Este procedimiento evita gastar memoria y tiempo de cálculo en las tramas que no contienen información, impidiendo así obtener resultados erróneos en el análisis de las señales de voz.

El algoritmo utilizado en este trabajo para detectar las regiones de silencio está basado en el cálculo de las variaciones de las muestras de la señal en una trama de voz con respecto a su media. Si la variación es bastante grande, la trama es considerada como una trama de voz, de lo contrario, como de silencio.

La región de silencio es detectada de la siguiente forma: se calcula la media de las muestras pertenecientes a una trama y se realiza la sumatoria del valor absoluto de la diferencia de cada muestra y la media. Entonces, si esta suma excede un umbral predefinido, la trama es considerada como una trama de voz, de lo contrario como una trama de silencio (Saito, 1981), (Smith, 1999). Este proceso se representa en las ecuaciones 4 y 5:

$$\mu = \frac{1}{N}\sum_{n=1}^{N} S_k(n) \quad (4)$$

$$\Omega = \sum_{n=1}^{N} |S_k(n) - \mu| \quad (5)$$

Donde Sk(n) son las muestras de la señal de la trama k, μ es la media y Ω es el resultado de la sumatoria, la cual es comparada con el umbral.

El umbral se calcula observando los valores obtenidos de Ω para diferentes muestras de voz (sonora y sorda) y muestras de silencio. En la Tabla 1 se indican los valores típicos obtenidos de Ω.

Tabla 1. Valores típicos de Ω

	Muestras sonoras			Muestras Sordas			Muestras de Silencio
	/a/	/e/	/u/	/f/	/s/	/g/	/ /
Ω	21.26	28.62	14.53	11.78	10.06	9.21	6.8

Como se observa, el valor mínimo de Ω para las muestras de voz es 9.21, mientras que para las muestras de silencio el máximo es de 6.8; por tanto, el umbral que se toma es de 8.00, permitiendo así tolerar algunos niveles de ruido.

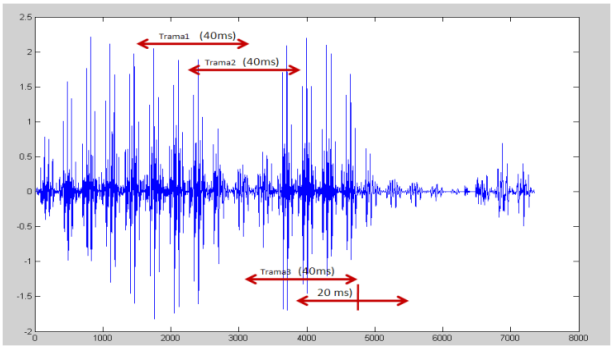

Figura 2. Señal de voz dividida en tramas de 40 ms. con traslape de 20 ms. y ventaneo con Hamming.

2.1.6 Extracción de parámetros característicos. La extracción de características por medio de coeficientes LPCC permite parametrizar la señal con un número pequeño de patrones, con los cuales es posible reconstruirla adecuadamente, además no requiere demasiado tiempo de procesamiento (Deller, 1987), (Rabiner, 1989), lo que es importante a la hora de la implementación. Para la selección del número de coeficientes LPCC se utilizó la siguiente recursión:

$$p \approx \left(\frac{F_s}{1200\,Hz}\right) + (2\ o\ 3) \quad (6)$$

$$F_s(frecuencia\ de\ muestreo) = 8000 \quad (7)$$

$$p \approx \left(\frac{8000}{1200\,Hz}\right) + (2\ o\ 3) \approx 10\ coeficientes\ LPCC \quad (8)$$

Una vez que la señal ha sido preprocesada y segmentada en tramas, se calculan los 10 primeros coeficientes LPC de cada trama por medio del método de autocorrelación y el algoritmo Levinson Durbin. Luego calculamos los

coeficientes LPC-Cepstrum con la siguiente recursión, en la cual partimos de los 10 coeficientes LPC hallados con la función de Matlab LPC.

$$cn = \begin{cases} 0 & n \leq 0 \\ -a_1 & n = 1 \\ -a_n - \sum_{k=1}^{n-1} -c(k)a_{n-k} & n > 1 \end{cases} \quad (9)$$

Donde los a_n son los coeficientes LPC de la trama analizada y los c_n son los coeficientes LPCC necesarios para caracterizar la señal de voz.

2.2 Módulo de entrenamiento del codebook

2.2.1 Cálculo del codebook. Anteriormente conseguimos reducir la información de la señal de voz que se adquirió a 8000 muestras por segundo cuantificadas a 16 bits/muestra, con tramas de 40ms. (con 20ms. de solapamiento) y 10 coeficientes LPCC, se obtuvo una reducción de datos de 32 (8000*0.04/10). De esta forma disminuyó la complejidad computacional del proceso de entrenamiento y reconocimiento. A pesar de esta reducción, era necesario tener una menor cantidad de datos en el sistema. En este caso, para reducir el volumen de información necesario para transmitir o almacenar la información de los coeficientes LPC-Cepstrum, recurrimos a la cuantificación vectorial.

Encontramos que según el siguiente teorema, el proceso de cuantificación vectorial es la forma más eficiente de cuantificar una señal vectorial.

Teorema **(**Larunbat, 2006): para cualquier sistema de codificación dado, que mapee una señal vectorial dada en N palabras binarias y reconstruya el vector aproximación desde alguna de estas palabras binarias, existe un cuantificador vectorial con tamaño de *codebook* N, que tiene exactamente el mismo rendimiento, esto es, para cualquier entrada produce la misma salida que el sistema de codificación dado.

Para el reconocimiento de palabras aisladas entrenamos el *codebook* con las palabras del vocabulario por reconocer. El proceso consiste en promediar vectores de características semejantes, para formar un *codebook* de 6 bits (64 vectores). El vector resultante de cada uno de los promedios será el centroide (codeword o centro de masas) de la agrupación (o clúster) de vectores usados para calcularlo, y será el que mejor los represente en su media.

Empleamos 6 bits o 64 centroides, ya que experimentando con *codebooks* más grandes (128 centroides) se puede obtener un mejor modelado, pero se aumentó significativamente el número de símbolos en las secuencias de entrenamiento de los HMM, y los tiempos necesarios para parametrizar y reconocer las palabras. Esto es debido a que la estimación de cada centroide es equivalente a la estimación de una media, y la variabilidad en la estimación de una media es inversamente proporcional al número de vectores usado.

La dimensión del *codebook* necesaria para conseguir una cuantificación adecuada es un asunto esencial en el diseño del *codebook* y define la potencia de cálculo requerida.

Vectores de observación: transformamos las palabras de entrenamiento (adelante, reversa, izquierda, derecha, frenar y Markov) en un conjunto de NumV vectores, siendo NumV el número de tramas por palabra. Cada uno de estos vectores es de dimensión 10 y contiene los coeficientes LPC-Cepstrum calculados para la trama correspondiente. Los vectores de todas las palabras fueron almacenados en una matriz, llamada MAVECT, que constituye el grupo de entrenamiento.

En el transcurso de esta investigación recopilamos varios grupos de entrenamiento, que se almacenaron como bases de datos en archivos .mat., los cuales fueron empleados en el desarrollo del trabajo. Algunas características fueron palabras por reconocer, múltiples repeticiones en un ambiente con poco ruido. Se observó que entrenar con palabras que no se van a reconocer produce centroides inútiles y hace que la información que sí es necesario clasificar se organice con un número menor de centroides.

Medición de la distancia entre vectores: fue necesario introducir el concepto de distancia, ya que tanto en el entrenamiento de los *codebooks,* como en el proceso de cuantificación, precisamos evaluar el parecido entre dos vectores de parámetros. Para medir la distancia entre un par de vectores utilizamos la distancia euclidea cuadrática, ecuación (10).

$$d(V_1, V_2) = \sqrt{(A_1 - A_2)^2 + (B_1 - B_2)^2 + \cdots + (Z_1 - Z_2)^2}$$

Con: $\begin{cases} V_1 = (A_1, B_1, \ldots, Z_1) \\ V_2 = (A_2, B_2, \ldots, Z_2) \end{cases}$ (10)

Encontramos que esta medida de distancia requiere de cálculos sencillos, lo cual nos representa menor tiempo de procesamiento en comparación con otro tipo de medida de distancia entre vectores. En Rabiner (1989) se recomienda emplear la medida de distancia Mahalanobis para la medida de distancias y la evaluación de la distorsión generada. Sin embargo, esta requiere el cálculo de la inversa de la matriz de covarianza de cada par de vectores, lo cual significa mayor cantidad de operaciones y tiempo de procesamiento.

2.2.2 Procedimiento de clasificación. Se empleó el algoritmo LBG (por Linde Buzo Gray), optimizado mediante algoritmo de Lloid.

El algoritmo LBG fue aplicado en 6 iteraciones o niveles para producir 2^6 centroides. Comenzamos con un centroide original, calculado con la media de los vectores de entrenamiento. En la primera iteración obtuvimos 2 centroides provisionales, sumando y restando un vector aleatorio de norma menor que uno al centroide original. Luego, se calculó la distancia de cada uno de los vectores de entrenamiento a estos nuevos centroides y se reorganizó en dos nuevos clústeres. De la media de los vectores dentro de los nuevos clústeres hallamos los vectores definitivos de la primera iteración. En las siguientes iteraciones a cada centroide que calculamos anteriormente se le sumó y restó un vector aleatorio de magnitud menor que uno, se calcularon distancias, se reorganizaron los clúster y se hallaron los centroides. Siguiendo este procedimiento, en cada iteración obtuvimos el doble de centroides que en iteración anterior (principio de división binaria)

En nuestro LBG de 6 niveles, además de la división binaria, hicimos varias pruebas para hallar el vector aleatorio, con el fin de obtener un vector que no nos dejara clústeres vacíos y nos entregara una buena distribución de centroides. Un método para realizar el *splitting* simple, calcula la desviación estándar y la media de los elementos de cada clúster para luego obtener nuevos centroides sumando la desviación estándar a los antiguos centroides en la dirección que nos indica la media. En la búsqueda de nuestro vector aleatorio se generó un arreglo de números aleatorios escogidos dentro de una distribución normal con la media aritmética y la desviación estándar del clúster y lo dividimos por 4 veces su valor máximo, para que su norma fuera mucho menor que uno. Este fue el que mejor desempeño tuvo en las pruebas realizadas.

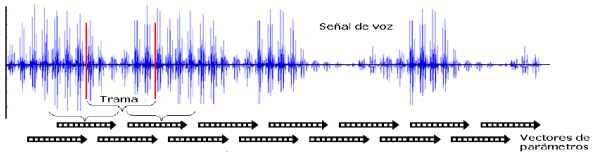

Figura 3. Principio de división binaria

El tipo de *codebook* que se obtuvo cuando realizamos el LBG, necesita ser optimizado. Para mejorar la distribución y la distorsión media de nuestro *codebook* empleamos el algoritmo de Lloid. Posteriormente, en el reconocimiento se comprobó que los reconocedores con *codebooks* implementados con Lloid tenían mejor desempeño que los que solo se habían trabajado con el LBG.

2.2.3 Entrenamiento del Modelo Oculto de Markov para cada palabra. Empleamos Modelos Ocultos de Markov (HMM) para modelar el mecanismo de producción del habla. Su capacidad de discriminación se debe a que están constituidos por un conjunto de estados, comparables a los estados que atraviesa el tracto vocal cuando hablamos. Cada uno de los estados produce un conjunto de posibles salidas, asimilables a los sonidos que configuran la señal de voz.

Por cuestiones de funcionamiento del reconocedor, empleamos dos modelos adicionales a las instrucciones para el comando de la silla: un modelo basura y un modelo que reconoce la respiración. En definitiva, se entrenaron los siguientes modelos: “Adelante”, “Reversa”, “Izquierda”, “Derecha”, “Frenar”, “Markov (fue escogido para indicar cuando el programa debe comenzar a reconocer)”, Modelo Basura (está destinado a reconocer como ‘basura’ palabras fuera del vocabulario del reconocedor) y el Modelo de la Respiración (se destinó a reconocer los sonidos propios de la respiración y ruidos involuntarios al exhalar).

Para dar solución al problema del aprendizaje o entrenamiento se buscó maximizar la probabilidad de

que una secuencia de observaciones haya sido generada por un HMM optimizando sus parámetros. Esto significa que para entrenar los modelos necesitábamos de algún tipo de estimación inicial de los parámetros.

Se hizo una estimación de los parámetros iniciales distinta para cada uno de los HMM, dependiendo de la cantidad de estados, de la duración de la palabra y de cómo fue reentrenado el modelo al comprobar su funcionamiento con el reconocedor. Los parámetros A y B son probabilidades almacenadas en forma de matrices. Las primeras estimaciones se hicieron con matrices equiprobables y aparecieron probabilidades muy pequeñas; posteriormente se emplearon matrices con valores arbitrarios que se fueron refinando a medida que se volvían a entrenar los modelos.

Por la topología empleada, el entrenamiento debe realizarse con múltiples observaciones $\boldsymbol{O} = [\boldsymbol{O}^{(1)}, \boldsymbol{O}^{(2)}, \boldsymbol{O}^{(3)}, \dots, \boldsymbol{O}^{(k)}]$.

A continuación se describe cómo implementamos el algoritmo Forward-Backward (adelante-atrás) para múltiples observaciones, también conocido como el algoritmo de Baum-Welch para reestimar los parámetros del HMM.

Por conveniencia, el parámetro pi se tomó iniciando el modelo en el primer estado, lo cual nos permitió evadir su reestimación en el algoritmo Baum Welch (Ecuaciones 11 y 12).

Los valores reestimados de A y B se obtienen de:

$$\bar{a}_{ij} = \frac{\textit{número esperado de transiciones desde el estado } S_i \textit{ al estado } S_j}{\textit{número esperado de transiciones desde } S_i}$$

(11)

$$\bar{b}_{ij} = \frac{\textit{número esperado de veces en el estado } j \textit{ viendo el simbolo } V_k}{\textit{número esperado de veces en el estado } j}$$

(12)

Estas expectativas fueron calculadas a partir de las definiciones de la variable en avance o alfa y la variable en retroceso o beta. La variable en avance α se define como la probabilidad de la secuencia de observación parcial (hasta un tiempo t) y estado S_i dado un modelo λ, y se calculó mediante un procedimiento recursivo (Rabiner, 1989):

Inicialización: $\alpha_1(j) = \pi_1 b_j(O_1), 1 \le i \le N$

Inducción: $\alpha_{t+1}(j) = \left[\sum_{i=1}^{N} \alpha_t(j)\, a_{ij}\right]_1 b_j(O_{t+1})$

$$1 \le t \le T-1$$

$$1 \le j \le N$$

Terminación: $P(O/\lambda) = \sum_{i=1}^{N} \alpha_T(i)$

La variable en retroceso β se define como la probabilidad de la secuencia de observación parcial (desde un tiempo t+1 hasta el final) dado un estado S_j y el modelo λ, al igual que α se calcula mediante un procedimiento recursivo:

Inicialización: $\beta_T(i) = 1, \qquad 1 \le i \le N$

Inducción: $\beta_t(i) = \sum_{j-1}^{N} a_{ij} b_j(O_{t+1}) \beta_{t-1}(j)$

$$t = T-1, T-2, \dots, 1, 1 \le j \le N,$$

Las ecuaciones (11) y (12) en términos de alfa y beta para múltiples observaciones se convierten la reestimación dada por la ecuaciones 13 y 14.

$$\overline{a_{ij}} = \frac{\sum_{k=1}^{K} \frac{1}{P_k} \sum_{t=1}^{T_k-1} \alpha_t^k(i) a_{ij} b_j\left(O_{t+1}^{(k)}\right) \beta_{t+1}^k(j)}{\sum_{k=1}^{K} \frac{1}{P_k} \sum_{t=1}^{T_k-1} \alpha_t^k(i) \beta_t^k(i)}$$

(13)

$$\overline{b_j}(l) = \frac{\sum_{k=1}^{K} \frac{1}{P_k} \sum_{\substack{t=1 \\ \text{siempre que} \\ O_t = V_k}}^{T_k-1} \alpha_t^k(i) \beta_t^k(i)}{\sum_{k=1}^{K} \frac{1}{P_k} \sum_{t=1}^{T_k-1} \alpha_t^k(i) \beta_t^k(i)}$$

(14)

Con estos cálculos hicimos el entrenamiento siguiendo estos pasos: i. Estimamos un conjunto inicial de parámetros {a,b}. ii. Calculamos las reestimaciones de a y b de acuerdo con las fórmulas. iii. Sustituimos los antiguos valores de a y b por los valores reestimados. iv. Se regresa al paso 2 hasta que se cumplió el criterio de convergencia o de terminación.

Al trabajar con múltiples observaciones no se tuvo un criterio de convergencia, sino un criterio de terminación del algoritmo. Experimentalmente se observa que algunos modelos presentaban buenas probabilidades en el reconocimiento con pocas iteraciones en el algoritmo de entrenamiento. Ajustando el número de estados, la cantidad de observaciones, la cantidad de centroides y las matrices iniciales (A, B) se pudo maximizar la probabilidad que el modelo haya producido la secuencia por reconocer.

2.3 Reconocimiento de palabras aisladas. La implementación de la etapa de reconocimiento se lleva a cabo de la siguiente forma: adquisición y muestreo de la señal de voz, cuantificación, filtro pasa bajos, preprocesamiento, extracción de parámetros característicos, cuantificación vectorial, cálculo de probabilidades, elección de la mayor probabilidad y salida de datos por el puerto paralelo.

Teniendo en cuenta que el algoritmo de reconocimiento no solo tiene que ser capaz de reconocer palabras aisladas, sino también de rechazar palabras pronunciadas en una conversación cotidiana, se entrena un modelo llamado modelo *basura* con 300 secuencias de observación de entrada para el entrenamiento del modelo, un *codebook* de 64 centroides, modelo HMM de seis estados.

Este modelo garantiza que ninguna palabra ajena al vocabulario de comando de la silla de ruedas vaya a generar un movimiento indeseado, estas palabras pronunciadas en una conversación cotidiana son reconocidas como modelo basura sin generar ninguna orden en el puerto paralelo.

Para evitar movimientos indeseados, se entrena el modelo para la palabra Markov, una palabra que no es usada frecuentemente. Al pronunciar Markov no activamos el reconocimiento, sino la salida de datos por puerto paralelo; mientras la palabra Markov no es pronunciada, el sistema reconoce todas las palabras normalmente, pero no envía ningún comando de acción para el sistema móvil. Si el usuario en una conversación cotidiana pronuncia la palabra adelante sin desear movimiento alguno, la silla no se mueve porque requiere del comando de activación del movimiento, en este caso la palabra Markov.

Luego de la pronunciación del comando de activación, el prototipo de reconocimiento de palabras aisladas está listo para comandar el movimiento de la silla de ruedas. Por ejemplo, pronunciamos la palabra izquierda y el prototipo halla la probabilidad de que cada uno de los modelos entrenados haya generado la secuencia de entrada generada por la pronunciación de la palabra, se comparan las probabilidades obtenidas y se escoge la mayor. Estas probabilidades se encuentran organizadas en un vector y al escoger la mayor probabilidad retornamos también su posición dentro del vector, así al enviar los datos por puerto no hay confusión.

Dependiendo de palabra reconocida se envían los siguientes datos por el puerto paralelo:

Adelante → 0001
Reversa → 0010
Izquierda→ 0100
Derecha → 1000
Frenar → 0000

2.3.1 Cálculo de probabilidad. Se calcula la probabilidad de la secuencia generada por alguno de los modelos de las palabras entrenadas. Esta probabilidad se calcula con el procedimiento *forward* que se mostrará a continuación Rabiner (1989):

Consideramos la variable *forward* definida así:

$$\propto_t (i) = P(O_1, O_2, \ldots\ldots, O_t, q_t = S_i/\lambda) \qquad (15)$$

Resolvemos para $\propto_t (i)$ de la siguiente forma:

1) Inicialización: $\alpha_1(i) = \pi_i b_i(O_1), \qquad 1 \leq i \leq N$

(16)

2) Inducción:

$$\propto_{t+1}(j) = \left[\sum_{i=1}^{N} \propto_t (i) a_{ij}\right] b_j(O_{t+1}),$$
$$1 \leq t \leq T-1, 1 \leq i \leq N \quad (17)$$

3) Terminación:

$$P(O/\lambda) = \sum_{i=1}^{N} \propto_T (i) \quad (18)$$

En este caso, por definición tenemos que:

$$\propto_T (i) = P(O_1, O_2, \ldots\ldots, O_t, q_T = S_i/\lambda)) \quad (19)$$

Y como observamos en la ecuación $P(O/\lambda)$ es justamente la suma de todos los $\propto_T (i)'s$. En la anterior recursión tenemos que a_{ij} es la matriz de probabilidad de transición entre estados entrenada para cada palabra, π_i es el vector de distribución de estado inicial el cual definimos así $\pi_i = [1,0,0, \ldots\ldots N]$, para asegurar que siempre empieza la secuencia en el estado uno, condición importante en el reconocimiento de voz y $b_j(O_t)$ matriz de probabilidad de emisión de cada símbolo de la secuencia en un estado determinado.

Para determinar los elementos de esta matriz es necesario tener una secuencia de observaciones de entrada compuesta por los centroides representativos de los vectores de cada observación y b_{jk} matriz de probabilidad de emisión de cada símbolo en un estado cualquiera entrenada para cada palabra. La matriz $b_j(O_t)$ se obtiene de la siguiente forma:

Secuencia de observación = [7, 4, 18,45, 33,64,…O_t ,……,45], $1 \leq t \leq T$

$$b_{jk}) = \begin{bmatrix} b_{11} & b_{12} & \cdots & b_{1k} \\ \vdots & & \ddots & \vdots \\ b_{j1} & b_{j2} & \cdots & b_{jk} \end{bmatrix} 1 \leq j \leq N \quad \text{y}$$
$$1 \leq k \leq 64 \quad (20)$$

Siendo N el número de estados y k el número de centroides.

Según la secuencia de observación tenemos que $O_1 = 7, O_2 = 4, O_3 = 18$ y $O_T = 45$, entonces la matriz de probabilidad de emisión de cada símbolo de la secuencia en un estado determinado es:

$$b_j(O_t) = \begin{bmatrix} b_{1\,7} & b_{1\,4} & b_{1\,18} & \cdots & b_{1\,45} \\ & \vdots & & \ddots & \vdots \\ b_{j\,7} & b_{j\,4} & b_{j\,18} & \cdots & b_{j\,45} \end{bmatrix} \quad (21)$$

Así obtenemos una matriz de dimensión N (número de estados) por T (longitud de la secuencia de entrada), donde cada elemento corresponde a la probabilidad determinada para dicha observación en la matriz b_{jk}.

2.3.2 Optimización del entrenamiento de los HMM. Como se explicó, no hay un método analítico para determinar los parámetros iniciales del modelo oculto de Markov de cada palabra, por lo tanto realizamos pruebas variando los siguientes parámetros: i. Número de secuencias de observación para el entrenamiento de los HMM de cada palabra, ii. Cantidad de estados del modelo, iii. Número de centroides, iii. Componentes de la matriz a_{ij}.

Se toma un total de 240 muestras, 40 muestras de cada palabra y a partir de estas pronunciaciones se calcula el error en el reconocimiento de cada palabra. Las modificaciones se realizaron en el algoritmo de entrenamiento de los HMM y el algoritmo de cálculo del *codebook.*

Prueba 1: se modifica la cantidad de secuencias de observación para el entrenamiento del HMM de cada palabra. Esta prueba se realiza con matrices a_{ij} y b_{jk} aleatorias, modelos de 6 estados y un *codebook* de 64 centroides.

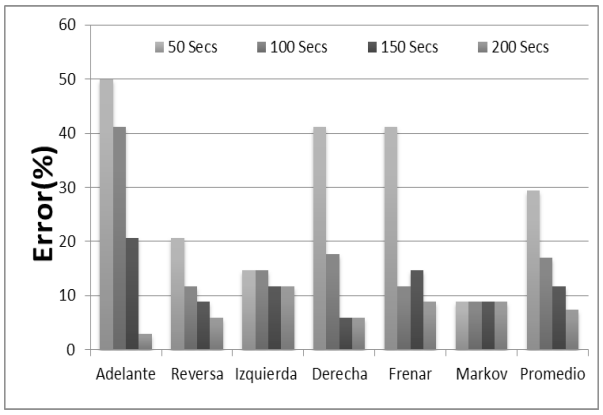

Figura 4. Cuadro comparativo de errores de la prueba 1.

Observando los resultados de la figura 4, se concluye que el error promedio total más bajo fue el obtenido con 200 secuencias de observación en el entrenamiento de los HMM. Con un número elevado de secuencias de observación para el entrenamiento del modelo de cada palabra se representa una mayor variabilidad, capacitando al modelo para responder satisfactoriamente a cualquier modificación en el estilo de habla. Sin embargo, con más de 200 secuencias de observación no se obtiene una reducción del error promedio considerable y sí se aumenta el tiempo de entrenamiento.

Prueba 2: se altera la cantidad de estados de los HMM de cada palabra en su entrenamiento. Esta prueba se realiza con matrices a_{ij} y b_{jk} aleatorias, 200 secuencias de observación y un *codebook* de 64 centroides.

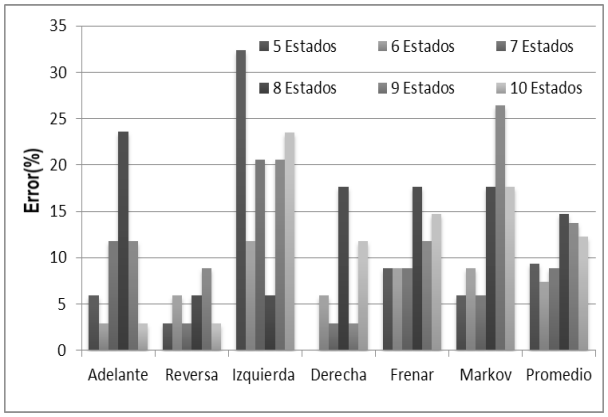

Figura 5. Cuadro comparativo de errores de la prueba 2

Como se puede observar en la figura 5 los mejores resultados en cuanto a disminución total del error se obtuvieron entrenando los HMM de cada palabra con 6 estados. Sin embargo, se realiza una prueba escogiendo el modelo entrenado para cada palabra con la cantidad de estados que obtuvieron menor promedio de error. Es decir, los modelos de las palabras no tienen el mismo número de estados para su entrenamiento, eso depende del promedio de error para cada modelo. Esta prueba se lleva a cabo con matrices a_{ij} y b_{jk} aleatorias, 200 secuencias de observación y un *codebook* de 64 centroides.

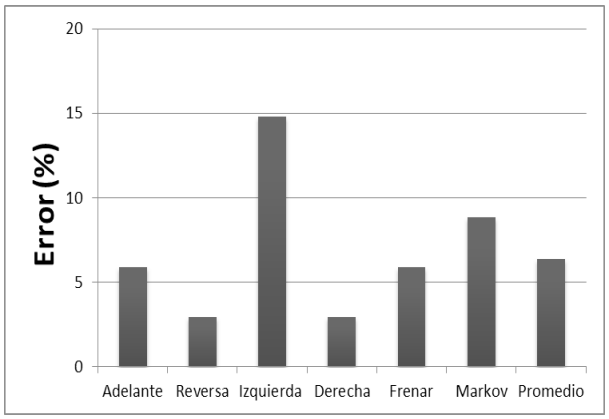

Figura 6. Cuadro comparativo de errores utilizando modelos con número de estados combinado.

Los promedios de error obtenidos en la prueba de la figura 6 no son los mismos de la prueba de la figura 5, porque al combinar los modelos el cálculo de las probabilidades cambia, es decir, puede aumentar o disminuir el error promedio en cada palabra. Como observamos (4) el mejor funcionamiento lo obtuvimos utilizando una combinación de modelos con distinto número de estados, esta combinación será utilizada en lo que sigue para pruebas posteriores.

Prueba 3: se cambia el tamaño del *codebook* tomando los parámetros que mejor funcionaron hasta el momento.

Esta prueba se realiza con matrices a_{ij} y b_{jk} aleatorias, 200 secuencias de observación y una combinación de modelos con distinto número de estados.

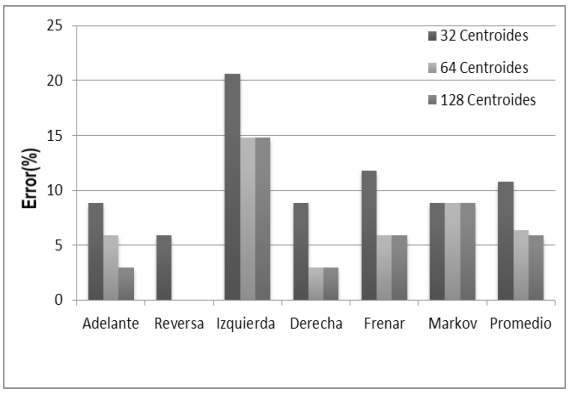

Figura 7. Cuadro comparativo número de centroides.

Observando los resultados de la figura 7, el mejor funcionamiento fue obtenido utilizando un *codebook* con 128 centroides característicos, pero para el reconocimiento su tiempo de ejecución aumentó casi el doble, como se muestra en la siguiente tabla. Esta medida del tiempo de ejecución se toma desde el final del proceso de adquisición hasta la elección de la máxima probabilidad. El tiempo de ejecución se encontró utilizando una herramienta de matlab llamada *Profiler***,** que mide el tiempo de ejecución de cada instrucción del programa por separado.

Como se observa en la Tabla 2, una disminución tan pequeña en el error promedio no justifica una elevación del tiempo de ejecución del 78.75 %; así que el *codebook* que mejor cumplió con el compromiso entre funcionamiento y velocidad de ejecución fue el *codebook* implementado con 64 centroides característicos.

Tabla 2. Tiempo de ejecución del programa con variación de centroides.

64 centroides		128 centroides
Palabra	Tiempo de ejecución (s)	Tiempo de ejecución (s)
Adelante	1.079	1.903
Reversa	1.047	2.032
Izquierda	1.219	2.084
Derecha	1.172	2.045
Frenar	1.063	1.947
Markov	0.969	1.672
Total	1.09	1.948

Prueba 4: se modifica el contenido de la matriz a_{ij} en el entrenamiento del modelo de cada palabra para poder obtener un mejor rendimiento en cuanto a eficiencia en el reconocimiento de palabras aisladas. Esta prueba se realiza con 200 secuencias de observación, una combinación de modelos con distinto número de estados, un *codebook* con 64 centroides característicos y una matriz b_{jk} constante. La matriz b_{jk} contiene componentes aleatorios que al modificarlos no muestran un cambio sustancial en la disminución del error, por lo cual todos los modelos se entrenaron con la misma matriz. Luego de hacer varias pruebas, se determinan las matrices a_{ij} utilizadas para el entrenamiento del modelo de cada palabra.

El promedio de error para cada palabra y el error total se muestran en la Tabla 3. El modelo definitivo se alcanza después de probar el reconocedor en funcionamiento, si el error de reconocimiento se acerca a cero, no se incrementa el número de observaciones y se archivan los parámetros.

Se puede observar que la disminución del error total es significativa, así que el algoritmo de reconocimiento de palabras aisladas que se implementó presenta las siguientes características: i. 10 coeficientes característicos; ii. *Codebook* de 64 centroides; iii. Coeficientes LPCC; iv. Algoritmo LBG para el entrenamiento del *codebook, v.* 200 secuencias de observación para el entrenamiento de los HMM de cada palabra; vi. Número de estados combinados para el entrenamiento de los HMM de cada palabra.

Tabla 3. Error en el reconocimiento de palabras aisladas modificando la matriz a_{ij} en el entrenamiento de los HMM de cada palabra.

Palabra	# de estados del modelo	Error (%)
Adelante	6	0
Reversa	5	2.94
Izquierda	8	5.88
Derecha	5	5.88
Frenar	6	5.88
Markov	*5*	2.94
Error promedio total		3.92

La eficiencia del algoritmo de palabras aisladas utilizando Modelos Ocultos de Markov es de 96.08 %, esta eficiencia fue obtenida luego de realizar las pruebas mostradas anteriormente. Se obtuvo una eficiencia alta teniendo en cuenta que para el comando de un sistema

móvil por medio de palabras aisladas es necesario que el prototipo cumpla un compromiso entre velocidad de procesamiento y eficiencia en el reconocimiento.

2.4 Interfaz de comunicación entre el computador y el sistema móvil

La interfaz de comunicación entre el computador y el sistema móvil propuesta inicialmente con un microcontrolador, fue cambiada teniendo en cuenta aspectos como la reducción en la circuitería y en los costos de construcción, por un arreglo de reles conectados en puente H para controlar la inversión de giro de los dos motores.

El puerto paralelo está conectado a una tarjeta de acople de tierras y posteriormente a los reles que, dependiendo de la orden pronunciada, se encargan de conectar las baterías a los motores. La silla de ruedas también incluye dentro de su estructura física un cargador de baterías acoplado. Además se desarrolló un manual de operación de la silla de ruedas, allí se incluyen los pasos necesarios para hacer un correcto uso de la silla de ruedas (en la Figura 8 se observa la silla construida en el prototipo).

Figura 8. Silla construida para manejo del prototipo.

3. CONCLUSIONES

La aplicación de los HMM al comando de dispositivos por voz es una aplicación que lleva grandes prestaciones en su desempeño, además de ser fácil de implementar y entrenar. Así mismo, en la generación del codebook se debe llegar a un compromiso entre error por distorsión, almacenamiento y costo computacional. En aplicaciones que no demande cumplir un compromiso de cómputo se podría generar un codebook con un mayor número de centroides y así representar más características fonéticas con cada uno, aumentando la eficiencia de los coeficientes LPCC, característica primordial a tener en cuenta en el momento de la aplicación.

El entrenamiento de los HMM hecho a partir de múltiples observaciones es capaz de producir mejores resultados con una iteración en el algoritmo de avance y retroceso. El modelo en proceso de entrenamiento se va acercando al modelo definitivo a medida que se reestiman parámetros con cada nueva repetición de la palabra, a diferencia del entrenamiento con una sola pronunciación en el cual el modelo definitivo se logra cuando convergen los parámetros re-estimados en muchas iteraciones del algoritmo.

El algoritmo de clasificación de vectores LBG funciona de manera excelente en comparación con k-means debido a que parte del principio de división binaria sin memoria, es decir, a medida que aumenta el número de clústeres los anteriores desaparecen y se realiza la reubicación de vectores con respecto a los centroides de los nuevos clústeres. Esto garantiza que la distorsión por cuantificación va a ser mínima.

4. REFERENCIAS

Alcubierre J.M., Mínguez J., Montesano L., Montano L., Saz 2º., Lleida 2E. (2005) Silla de Ruedas Inteligente Controlada por Voz. Primer Congreso Internacional de Domótica, Robótica y Teleasistencia para todos. Fundación ONCE, Madrid, España.

Alonso Gonzales Itzar Goretti. (2002) Reconocedor de dígitos, Universidad de las Palmas de Gran Canaria.

Álvarez Mauricio, Suárez Julio F., Castellanos Germán. (2003) Selección de características para el Reconocimiento de Voz con Modelos Ocultos de Markov. Universidad Nacional de Colombia, Sede Manizales.

Burton David K., (1987) Text-Dependent Speaker Verification Using Vector Quantization Source Coding, IEEE Transactions on Speech and Audio Processing, Vol. 35, No. 2. Doi: 10.1109/TASSP.1987.1165110.

Cabas Vanegas Edgar de Jesús, Barrera Alarcón Yecid Fernando (2004). Diseño e Implementación de un Prototipo que realice la Verificación Automática de Personas por su Voz. Tesis de grado. Ingeniería Electrónica, Universidad Pedagógica y Tecnológica de Colombia, Seccional Sogamoso.

Larunbat B. (2006) Cuantificación Vectorial, Instituto de Ingeniería Eléctrica - Facultad de Ingeniería - Montevideo, Uruguay.

Deller John R., Proakis G., Hansen H.L., (1993) Discrete-time processing of speech signals, Prentice-Hall.

Departamento de Ingeniería Electrónica y Comunicaciones Centro Politécnico Superior Universidad de Zaragoza, (2006). Tecnologías de la voz, Práctica IV, Análisis y Síntesis LPC de la señal de voz.

Faundez Zanuy Marcos, (2001) Tratamiento digital de voz e imagen, Marcombo Editores.

Gold Ben., Morgan Nelson. (2011), Speech And Audio Signal Processing, Processing and Perception of Speech and Music, John Wiley & Sons, Inc.

Hasegawa-Johnson Mark, (2000) Lecture Notes in Speech Production, Speech Coding, and Speech Recognition. University of Illinois at Urbana-Champaign.

Maldonado José Luciano (2003) La estadística como herramienta para el desarrollo de sistemas automáticos reconocedores del habla. Instituto de Estadística Aplicada y Computación Facultad de Ciencias Económicas y Sociales Universidad de Los Andes.

Martínez Fernando, Portale Gustavo, Klein Hernán, Olmos Osvaldo. (2006), Reconocimiento de voz, apuntes de cátedra para Introducción a la Inteligencia Artificial. Universidad Tecnológica Nacional.

Méndez Ortiz Freddy, Vecino Pico Hugo. (2006) Propuesta, validación y prueba de una arquitectura para modelado de portales WEB semánticos basados en interacción por voz. Grupo de tecnologías de información, Universidad Autónoma de Bucaramanga.

Moralejo, L. Ostermann, S. Sanz, C. (2010) Adaptación a Jclic para alumnos con deficiencia motriz, mediante comandos por voz. Memorias Virtual Educa 2010, Santo Domingo, República Dominicana.

Rabiner L.R., (1989) "A Tutorial on Hidden Markov Models and Selected Applications in Speech Recognition," Proceedings of the IEEE, vol. 77, no. 2. Pp. 257-286. doi. 101109/5.18626

SAITO Shuko, Nakata Kazuo. (1981), Fundamentals of Speech Signal Processing, Academic Press.

SMITH Sreven W. (1999), "The Scientist and Engineer's Guide to Digital Signal Processing". 2da. edición, San Diego CA. California Technical Publishing.

Villamil Espinosa Iván Horacio. (2005). Aplicaciones en Reconocimiento de Voz Utilizando HTK. Trabajo de grado, Departamento de electrónica. Pontificia Universidad Javeriana Colombia.