Evaluación de la Información Articulatoria como Medio para Mejorar el Desempeño de los Sistemas de Verificación del Hablante

Dagoberto Porras Plata

Trabajo de Grado para optar al título de Magíster en Ingeniería Electrónica

Director

Franklin Alexánder Sepúlveda Sepúlveda

PhD en Procesamiento de Señales

Universidad Industrial de Santander

Facultad de Ingenierías Fisicomecánicas

Escuela de Ingenierías Eléctrica, Electrónica y de Telecomunicaciones

Bucaramanga

2019

Dedicatoria

Este trabajo viene dedicado para todas aquellas personas que apoyaron el desarrollo y ejecución de este trabajo de grado.

En especial a mí familia que me ha enseñado como afrontar la vida.

Agradecimientos

En primer lugar a mi madre Victoria Plata y mi padre Dagoberto Porras, que a lo largo de la vida me han formado y educado para afrontar grandes desafíos y ser un gran ser humano. Además de brindarme su cariño y apoyo todo el tiempo.

A mi hermana, Stefany Porras, que me ha aconsejado y me ha servido como ejemplo, para crecer y tomar buenas decisiones.

A mi hermano, Nicolás Porras, que ha sido un ser humano maravilloso además he contado con todo su apoyo, cariño y gracias por recordarme nunca renunciar a pesar de los obstáculos.

A mi director de este trabajo, el profesor Franklin Alexander Sepúlveda, que con su gran calidad humana y su dedicación, ha hecho de esta una experiencia extraordinaria. Además de brindarme la oportunidad de trabajar en este proyecto.

Al profesor Tamás Gábor Csapó, que hizo posible la pasantía en Budapest y del cuál aprendí una infinidad de cosas.

A todos mis amigos que han sido parte y han hecho posible de cualquier manera la realización de éste proyecto.

Gracias a todos.

Tabla de Contenido

Introducción **18**

1. Objetivos **24**

2. Propuesta de Investigación **24**

2.1. Estructura general de los sistemas de identificación y verificación de hablantes 24

2.2. Imágenes de Ultrasonido 28

2.3. Esquema de la tesis 30

3. Sistema de reconocimiento de hablantes **34**

3.1. Representación de la señal hablada mediante parámetros acústicos 34

3.2. Representación de la señal hablada mediante parámetros articulatorios 40

3.2.1. Extracción de características Articulatorias 43

3.3. Modelado de hablantes 45

3.3.1. Filosofía UBM 47

3.3.2. Modelado basado en GMM 48

3.3.3. Razón de verosimilitud (Likelihood Ratio - LR) 52

3.3.4. Modelado basado en i-vectors+PLDA 53

4. Resultados Experimentales y Discusión **58**

4.1. Configuración del experimento 58

4.1.1. Datos 58

4.1.2. Extracción de características acústicas 62

4.1.3. Fusión de Parámetros articulatorios y características espectrales 62

4.2. Estimación de los modelos 64

4.3. Identificación de hablantes 65

4.4. Verificación de hablantes 67

4.4.1. Experimentos de Verificación 68

5. Inversión Acústico-Articulatorio usando DNN's 71

5.1. Datos 75

5.2. Deep Neural Networks (DNNs) 76

5.3. Arquitectura DNN 77

5.4. Resultados y Discusión del procedimiento de inversión 77

6. Trabajo futuro 79

7. Conclusiones 80

Referencias Bibliográficas 81

Apéndices 94

7.1. GMM-UBM características acústicas (MFCC) 107

7.2. GMM-UBM características articulatorias - poly 112

7.3. GMM-UBM características articulatorias - 5 puntos 121

7.4. GMM-UBM con fusión de características 130

7.5. i-vector + PLDA características articulatorias - poly 135

7.6. i-vector + PLDA Fusión de características MFCC + poly 146

Lista de Figuras

Figura 1. Esquema general de un sistema de Identificación de hablantes. En los recuadros se encuentran las etapas de entrenamiento y pruebas del sistema. 25

Figura 2. Esquema general de un sistema de Verificación de hablantes. En los recuadros se encuentran las etapas de entrenamiento y pruebas del sistema. 26

Figura 3. Esquema general de un sistema de verificación de Hablantes. Se destacan las etapas de Entrenamiento y Test. 27

Figura 4. (arriba) *Nota:* Funcionamiento de un transductor piezo-eléctrico para producir la imagen de ultrasonido. Adaptado de (of Manitoba et al., 2019). (abajo) Cada cristal piezo-eléctrico produce una línea de *scan*, referencias de las imágenes al pie de página. Adaptado de (UTI, 2019). 29

Figura 5. Imagen de Ultrasonido con ruido granular o *speckle* 30

Figura 6. Ejemplo: Contorno extraído de una imagen de la lengua 2D-ultrasonido. 44

Figura 7. Diagrama de representación de la información articulatoria (imágenes de ultrasonido). 45

Figura 8. Esquema de inscripción de los hablantes. 46

Figura 9. Histograma de los fonemas incluidos en la base de datos. 59

Figura 10. Recolección de Datos 61

Figura 11. Esquema de la fusión de características. Cabe destacar que la fusión es realizada al cocatenar los vectores de características acústicas (MFCC) y articulatorias, después de descartar aquellos frames de silencio, que detecta el VAD. Por otra parte, para asegurar el mismo número de frames de MFCC que frames de características articulatorias, se realiza al fijar la ventana de tiempo de extracción de MFCC que la frecuencia de muestreo de la sonda de ultrasonido. 63

Figura 12. Diagrama de bloques del modelado de hablantes utilizando mezclas gaussianas (rojo) e i-vector (naranja). 63

Figura 13. Esquema de SVH con información articulatoria. 69

Figura 14. Curva DET, donde se muestran los resultados experimentales de verificación. 72

Figura 15. Esquema de entrada para la DNN, donde se muestran las dos características articulatorias utilizadas. 74

Figura 16. La calidad de la imagen medida en una frase de prueba. En la leyenda, los términos antes del símbolo '_' indican la imagen utilizada como la verdadera. El término después del símbolo representa la imagen que estamos comparando. por ejemplo, O_rec se refiere a que la imagen original (verdadera) es comparada con la imagen reconstruida por la DNN; Para ser claro, el término PCA es la imagen original pero en una versión reconstruida de la representación con PCA o del espacio EigenTongue (Verdadera). 155

Figura 17. En la parte superior derecha se encuentra una imagen de ultrasonido a partir de la información acústica. La imagen original está en la parte superior izquierda. También se muestran Et1 y Et2 en el fondo 156

Lista de Tablas

Tabla 1. Set de datos: Número de sesiones e imágenes registrados por hablante. 60

Tabla 2. Desempeño logrado por los sistemas identificación entrenados con características acústicas o articulatorias 66

Tabla 3. Resultados de los experimentos de identificación haciendo una fusión de características 66

Tabla 4. Uso del set de datos en el presente trabajo, en las fases de un SVH. 68

Tabla 5. Desempeño del sistema GMM-UBM, para las características propuestas. 70

Tabla 6. SVH con fusión de características, ademas se pueden observar los resultados con los MFCC y el resultado con los parámetros articulatorios poly (1). 70

Tabla 7. Resultados: experimentos utilizando i-vector. 71

Tabla 8. Evaluación de las tres diferentes configuraciones de DNN y características de imágenes de ultrasonido. 79

Lista de Apéndices

pág.

Apéndice A.	Código en python para la toma de datos sincronizada	94
Apéndice B.	Archivo de configuración y Listas para HTK	100
Apéndice C.	VAD y normalización CMVN de características	102
Apéndice D.	Fusión de características	104
Apéndice E.	Implementación GMM-UBM	107
Apéndice F.	Implementación i-vector + PLDA	135

Lista de Abreviaciones

SVH Sistema de Verificación del Hablante

MFCC Coeficientes Ceptrales en la escala de Mel

AF características articulatorias

EMA Articulografía Electromagnética

GMM Modelo de Mezclas Gaussianas

EM Expectation Maximization

ML Maximum Likelihood

MAP Maximum A posteriori

rtMRI vídeos de resonancia Magnética

US Ultrasonido

UBM Modelo de referencia poblacional

LR Razón de verosimilitud

VH Verificación de hablantes

DNN Deep Neural Network

ANN Red Neuronal Artificial

PLDA Probabilistic Linear Discriminant Analysis

EER Equal Error Rate

poly Características Articulatorias descritas por los coeficientes de un polinomio

VAD Detector activo de voz

CMVN Cepstral mean variance normalization

DET Detection error tradeoff graph

UTI Imagen por ultrasonido (Ultrasound tongue Imaging)

Resumen

Título: Evaluación de la información articulatoria como medio para mejorar el desempeño de los sistemas de verificación del hablante [*]

Autor: Dagoberto Porras Plata [**]

Palabras Clave: GMM, UBM, Python, I-vector, Ultrasonido, Verificación del Hablante.

Descripción: Los sistemas barométricos basados en voz son de las formas más preferidos para la gestión de identidad debido a su flexibilidad, velocidad y costo reducido. Sin embargo, se sabe que los sistemas actuales pueden verse afectados por variaciones inesperadas durante la fase de validación, tales como ruido de entorno o cambios en el esfuerzo vocal. En este trabajo se evalúa la información articulatoria del movimiento de la lengua como medio para mejorar el desempeño de los sistemas de verificación del hablante (SVH). Se utilizó una base de datos en español, donde además de las señales de voz, también se adquiere información articulatoria con un sistema de ultrasonido. Se proponen dos grupos de características para representar la información articulatoria y el desempeño obtenido es comparado con un SVH entrenado únicamente con información acústica. Los resultados muestran que las características propuestas contienen gran cantidad de información discriminativa y altamente asociada a la identidad de los hablantes, además que se pueden emplear para complementar y mejorar SVH existentes como por ejemplo combinando dicha información con coeficientes cepstrales. Un resultado adicional que puede encontrarse en éste documento, es un mapeo acústico-articulatorio desarrollado con *Deep Learning*, donde se estiman secuencias de imágenes de ultrasonido a partir de las muestras de audio; como trabajo futuro esta información articulatoria estimada, podría incluirse al SVH propuesto.

---

[*] Trabajo de grado

[**] Facultad de Ingenierías Físico-Mecánicas. Escuela de Ingenierías Eléctrica, Electrónica y telecomunicaciones. Director: Franklin Alexánder Sepúlveda Sepúlveda, PhD en Procesamiento de Señales.

Abstract

Title: Evaluation of articulatory information as a means to improve the performance of speaker verification systems [*]

Author: Dagoberto Porras Plata [**]

Keywords: GMM, UBM, Python, Ultrasound, Speaker Verification.

Description: Speech-based biometrics is one of the most preferred methods for identity management due to its flexibility, speed and reduced cost. However, current state-of-the-art speaker verification (SV) systems are known to be strongly affected by unexpected variability presented during testing, such as environmental noise or changes in vocal effort. In this work we analyze and evaluate articulatory information of the tongue's movement as a mean to improve the performance of speaker verification systems. We use a Spanish database, where besides the speech signals, we also include articulatory information which was acquired with an ultrasound system. Two groups of features are proposed to represent the articulatory information and the obtained performance is compared with an SV system trained only with acoustic information. Our results show that proposed features contain highly discriminative information and it is related to speaker identity, furthermore, these features can be used to complement and improve existing systems by combining such information with cepstral coefficients at the feature level. An additional result that can be found in this document is an acoustic-articulatory mapping developed with Deep Learning, where sequences of ultrasound images are estimated from the audio samples; as future work this estimated articulatory information could be included in the proposed SVH.

---

[*] Master Thesis

[**] Facultad de Ingenierías Físico-Mecánicas. Escuela de Ingenierías Eléctrica, Electrónica y telecomunicaciones. Director: Franklin Alexánder Sepúlveda Sepúlveda, PhD Signal Processing

Introducción

Cada día estamos más familiarizados con sistemas que utilizan la biometría para diferenciar o reconocer a las personas. Por citar un ejemplo, es común que los teléfonos celulares de última generación estén dotados con funciones de reconocimiento de voz, además de mecanismos de reconocimiento facial y dactilar. Especialmente, la voz se destaca por dos grandes rasgos: en primer lugar, la voz representada en el habla es una señal producida naturalmente por los usuarios; y en segundo lugar, la telefonía permite fácilmente la adquisición de la señal de voz debido a que estos sistemas de adquisición son de bajo costo.

El reconocimiento de hablantes es el proceso de discriminar automáticamente a una persona basada en la información incluida en su habla. Este se divide en dos tareas: la identificación y la verificación. La primera consiste en reconocer a un hablante de entre un grupo de diferentes hablantes, y, la verificación determina si el hablante es quien dice ser. El primer caso es un sistema de tipo cerrado, es decir que su respuesta consiste en determinar cuál de entre una cantidad reducida de N posibilidades (personas) es quien corresponde, por lo que se requiere comparar la muestra de prueba con los N modelos de los N hablantes. De otra parte, en el caso de verificación del hablante se busca determinar si es o no la persona en cuestión, dentro de un grupo abierto de posibilidades.

Los sistemas de reconocimiento de hablantes a grandes rasgos se dividen en dos grandes etapas: la representación y el modelado. La representación de la señal se refiere al proceso de extracción de características, donde se obtienen parámetros que representan el comportamiento de la señal del habla. En cuanto a lo que se refiere a la representación, en años recientes se han planteado

diferentes formas de representación de la señal de voz, las cuales se pueden dividir en las siguientes categorías (Kinnunen and Li, 2010): Características espectrales de tiempo corto, características relacionadas con la fuente, características espectro-temporales, y, características de alto nivel. Las características de tiempo corto, como su nombre lo indica, son estimadas en segmentos cortos de entre 20 y 30 ms, generalmente. Aunque características que pertenecen a esta categoría, tales como los MFCC (*Mel-frequency cepstrum coefficients*), son fáciles de estimar y requieren de una menor longitud total de audio, tienden a ser más afectadas por ruido y desacople en el canal respecto a otras características (Kinnunen and Li, 2010; Rao and Sarkar, 2014).

De otra parte, las características relacionadas con la fuente caracterizan el comportamiento de la señal de excitación glotal en sonidos del tipo sonoro (Kinnunen and Li, 2010), las cuales han probado que mejoran el desempeño de sistemas de verificación de hablantes (SVH) (Zheng et al., 2007). En cuanto a las características espectro-temporales se destacan los formantes (Morrison, 2008), los cuales también han mostrado ser de utilidad en los SVH pero su estimación presenta dificultades (Morrison, 2008). Finalmente están las características de alto nivel, las cuales son más robustas contra ruido y desacople del canal, pero son más difíciles de estimar y requieren de una mayor longitud efectiva de audio (Kinnunen and Li, 2010). A este grupo pertenecen fonemas y secuencias de fonemas (Campbell et al., 2007), además de características prosódicas tales como estrés sobre las sílabas, patrones de entonación, rata de habla, entre otros.

Otra fuente de información importante corresponde al fenómeno articulatorio en sí mismo. Una forma de representar este tipo de información son las características articulatorias (AFs,*Articulatory*

Features), las cuales mediante categorías describen el tipo de movimiento y la posición de los diferentes articuladores durante el proceso de producción del habla. Información específica del hablante en particular existe en este tipo de categorías, y por tanto son susceptibles de ser aplicadas en tareas de reconocimiento de hablantes (Leung et al., 2006). En (Leung et al., 2006) se modela la relación entre las categorías articulatorias (manera y punto de articulación) y los fonemas obtenidos a partir de un sistema de reconocimiento de fonemas estándar con el fin de alimentar un sistema de verificación de hablantes. Aunque esta estrategia es exitosa, los AF extraídos son todavía características basadas solamente en la acústica. Por otro lado, varios estudios han demostrado que una fuente importante de variabilidad entre hablantes en la acústica del habla reside en la variabilidad de la morfología del tracto vocal en sí, lo cual la convierte en un buen candidato para propósitos de reconocimiento de hablantes.

Con el fin de representar la forma del tracto vocal se pueden utilizar las siguientes opciones: *vocal tract areas*, parámetros de Maeda, o, parámetros descriptores del movimiento de los articuladores. Este tipo de información ha mostrado utilidad en aplicaciones como reconocimiento de voz (Ghosh and Narayanan, 2011) y reconocimiento de emociones (Kim et al., 2012), además de su utilidad para reconocimiento de personas por las propiedades intrínsecas de este tipo de características; sin embargo, su medición puede ser compleja. Una forma alternativa consiste en estimar el movimiento de los articuladores a partir de la acústica de la voz mediante procedimientos de inversión acústico-articulatoria (Ghosh and Narayanan, 2010; Sepúlveda et al., 2013; Potard et al., 2008). En particular, en (Ghosh and Narayanan, 2011) se propone el uso de las trayectorias de los articuladores estimadas a partir de la señal acústica, como medio para mejorar el desempeño de

los sistemas VH. Posteriormente en (Li et al., 2016a) los autores muestran que el aumento de los MFCC con características obtenidas de la inversión acústica-articulatoria mejora significativamente el rendimiento. La estrategia basada en datos utilizada en (Li et al., 2016a) estima la posición de los articuladores a partir de datos EMA (*Electromagnetic Articulograph* [1]) provenientes de 5 hablantes; y la verificación de hablantes es realizada con un *framework* basado en GMMs (*Gaussian Mixture Models*). A diferencia de estimaciones de la posición de articuladores, en (Zhang et al., 2017a) se utilizan datos EMA directamente, donde se incluyen veinticinco oraciones pronunciadas por seis hablantes. La adquisición con dispositivos EMA es costoso, y por tanto las bases de datos están conformadas por pocos hablantes. A pesar de esto, en (Illa and Ghosh, 2018a) se utilizan datos EMA de 31 sujetos, 18 hombres y 13 mujeres. Los experimentos se realizaron utilizando redes neuronales convolucionales y con memoria a largo y corto plazo como clasificadores, lo que demuestra que el movimiento articulatorio, cuando se utiliza como una característica para la tarea de identificación del hablante, produce una precisión de 98.75 %.

El fenómeno articulatorio también puede ser medido mediante imágenes del tracto vocal. Una de las técnicas corresponde a vídeos de resonancia magnética (rtMRI) del tracto vocal en el plano mid-sagittal (Narayanan et al., 2014). En (Prasad et al., 2015) se utilizan distancias entre las fronteras superior e inferior del tracto vocal, con el fin de representar la forma del tracto vocal mientras se pronuncian cuatro oraciones por cuatro hablantes. Los resultados sugieren que estas

---

[1] Es un método para medir la posición de algunos articuladores de la boca, como la lengua y los labios entre otros. Consiste en ubicar sensores electromagnéticos en puntos de articulación.

características que son estimadas a partir del movimiento articulatorio son más representativas en comparación con los MFCC. Pero este resultado no es definitivo porque es necesario tener en cuenta la mala calidad de audio de las grabaciones de audio durante el proceso de sensado rtMRI.

La adquisición con dispositivos EMA y con rtMRI es costoso, lo que dificulta la toma de datos para una suficiente variedad de hablantes. Además, EMA solo sensa algunos puntos del tracto vocal. En contraste, el ultrasonido (US) puede ser portable y es notablemente más económico; También, puede capturar en tiempo real el movimiento del contorno mid-sagittal, aunque solo de la lengua. Adicionalmente, lo hace de una forma no invasiva e intrusiva, a diferencia de EMA, que requiere introducir sensores electromagnéticos al interior del tracto vocal afectando la naturalidad al hablar. Además, US no sufre de problemas de interferencia como si las tiene rtMRI. Estas propiedades hacen que el sistema US sea más asequible y con mayores posibilidades de masificación de sus aplicaciones, tales como interfaces silenciosas (Hueber et al., 2008; Csapó et al., 2017).

Motivados por las ventajas que supone la utilización de una sonda de ultrasonido y bajo la hipótesis que los movimientos articulatorios reflejan la variabilidad de los hablantes (Li et al., 2016a), en el presente trabajo se propone explorar y analizar la utilidad potencial de la información articulatoria del contorno superior de la lengua para propósitos de verificación e identificación de hablantes. El propósito del presente trabajo es calcular el aporte del movimiento del contorno superior de la lengua como información discriminante entre hablantes. Adicional a este objetivo, en el capítulo 5 quisimos mostrar el resultado de un trabajo realizado por el autor en una pasantía de investigación en *Budapest University of Technology and Economics*. Donde, redes neuronales

aprenden la relación entre la señal acústica e imagen de ultrasonido de la lengua utilizando *Deep Learning*, con el fin de estimar las imágenes de ultrasonido que forman la información articulatoria.

1. Objetivos

Objetivo general

Evaluar y analizar el uso de la información articulatoria del movimiento de la lengua como medio para mejorar el desempeño de los sistemas de verificación del hablante.

Objetivos específicos

Extraer características acústico-articulatorias, a partir de la información recolectada en una base de datos;

Evaluar y comparar el desempeño de un sistema de verificación de hablantes entrenado únicamente con características articulatorias extraídas a partir de la información relacionada con la posición de la lengua recolectada mediante imágenes de ultrasonido durante el proceso de producción del habla;

Evaluar la mejora en desempeño de los sistemas de verificación de hablantes al unir las características acústicas y articulatorias de la información recolectada en una base de datos.

2. Propuesta de Investigación

2.1. Estructura general de los sistemas de identificación y verificación de hablantes

Un sistema de reconocimiento de hablantes es usado para identificar y distinguir personas mediante la extracción y comparación de características incluidas en el habla. La identificación y la verificación de hablantes son las dos tareas fundamentales en este campo. La identificación de hablantes es la tarea de detectar a un hablante responsable de producir un diálogo de prueba. Si la muestra

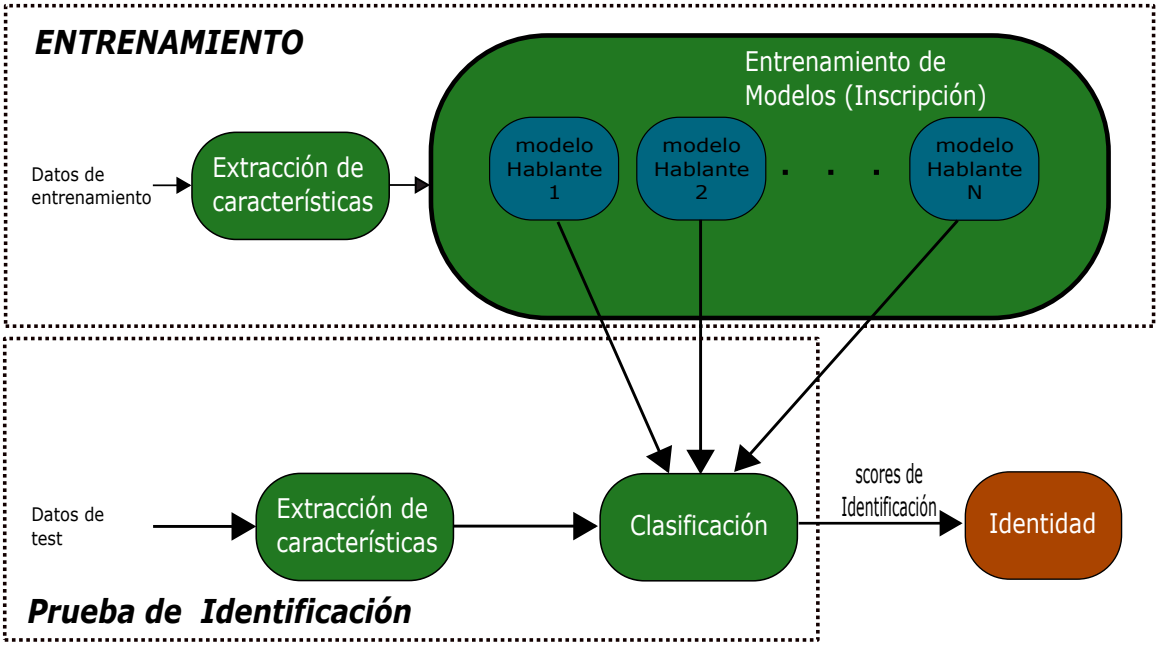


Figura 1. Esquema general de un sistema de Identificación de hablantes. En los recuadros se encuentran las etapas de entrenamiento y pruebas del sistema.

de prueba corresponde a alguno de los hablantes inscritos en el sistema, la identificación es de tipo *closed-set*, y en el caso en que este dialogo no pertenezca al set de hablantes inscritos se denomina *Open-set* (Rao and Sarkar, 2014). El esquema 1 muestra las etapas de una prueba de identificación.

Por otra parte, la verificación de hablantes es el proceso de verificar automáticamente la identidad de una persona por medio de la señal de voz. En otras palabras, lo que se busca es responder si una voz cuestionada corresponde o no con una voz cuya fuente se conoce. El esquema 2 muestra la estructura general de un sistema de verificación de hablantes. En este esquema, se entrena un modelo UBM (*Universal Background Model*) que representa a toda la población, que es diferente del hablante inscrito en el sistema. Además el modelo UBM, sirve como impostor, pues al comparar el hablante inscrito con el resto de población, podríamos responder a la pregunta de

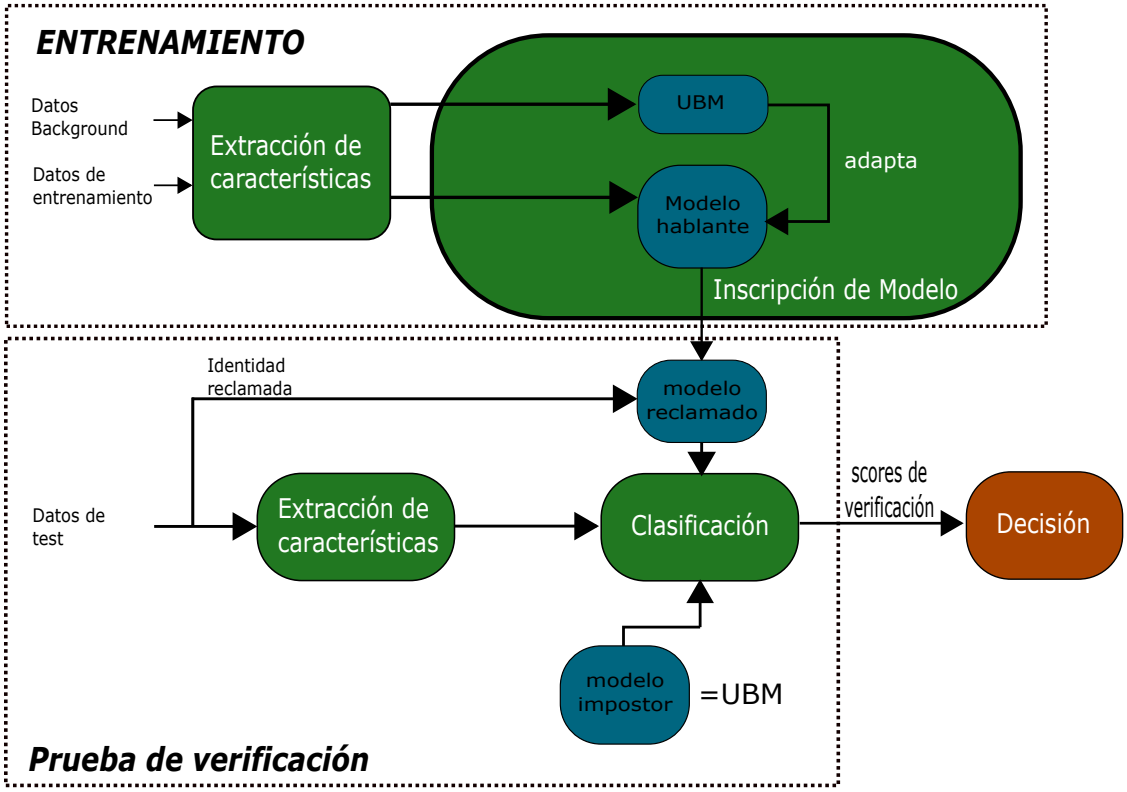


Figura 2. Esquema general de un sistema de Verificación de hablantes. En los recuadros se encuentran las etapas de entrenamiento y pruebas del sistema.

verificación. La estructura de identificación y verificación difieren únicamente en la etapa de prueba como muestra los dos esquemas. Y es consecuente, con lo que se ha mencionado anteriormente pues cada tarea responde a una pregunta diferente.

Esta última área ha experimentado un desarrollo notable en años reciente y es el tema principal del trabajo. A continuación podemos ver algunas estrategias y procedimientos utilizados en verificación automática de hablantes, ademas de una revisión del estado del arte.

La verificación de hablantes puede ser realizada dentro de dos situaciones: 1) Cuando se puede contar con la cooperación del hablante, en ese caso, la persona debe decir una contraseña o

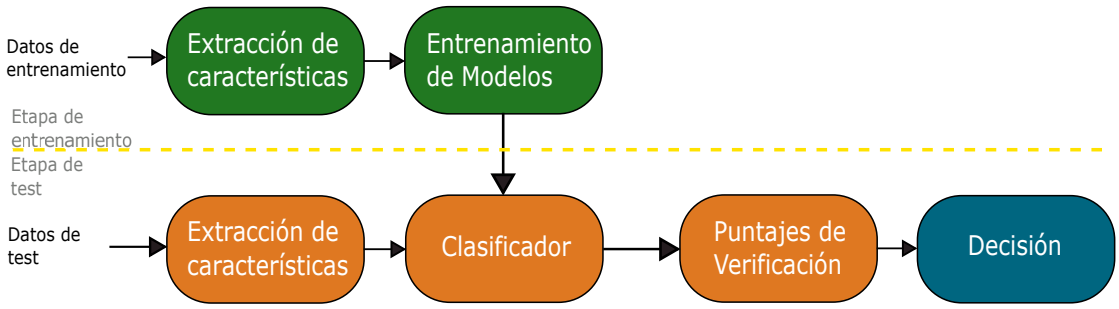


Figura 3. Esquema general de un sistema de verificación de Hablantes. Se destacan las etapas de Entrenamiento y Test.

frases ya establecidas por el sistema; esta verificación es llamada texto dependiente (Bimbot et al., 2004) y es utilizada por un gran número de aplicaciones ya que al contar con un conocimiento a priori del texto, el desempeño del sistema puede mejorar (Bimbot et al., 2004). No obstante, no en todos los casos se cuenta con la cooperación del usuario, de manera que en este escenario el sistema debe ser más flexible, y ser capaz de realizar la tarea de verificación sin contar con la operación del usuario. En estos casos, es llamado texto independiente. Esta última, es la tarea más desafiante (Rao and Sarkar, 2014).

Desde otro punto de vista y como muestra la Figura (El esquema ha sido dibujado en base a los trabajos de (Rao and Sarkar, 2014; Sadjadi et al., 2013)) 3, la línea amarilla destaca dos etapas para el desarrollo de un sistema de verificación de hablantes (VH): entrenamiento y test. Donde es común que representen el comportamiento de la señal de voz (extracción de características) de la misma forma. Luego, en la etapa de entrenamiento se sintetizan estos parámetros y son representados mediante modelos probabilísticos (modelado de hablantes). En la etapa de test, se comparan los parámetros de una muestra de prueba con uno o más modelos de hablantes, con el fin de obtener un puntaje de verificación y basados en este valor puede ser aceptada o rechazada la

identidad del sujeto de prueba.

2.2. Imágenes de Ultrasonido

El ultrasonido es uno de los métodos más atractivos para medir la dinámica de la lengua durante el habla (Stone, 2005; Li et al., 2005; Michailovich and Tannenbaum, 2006; Xu et al., 2016a), ya que es considerado un método no invasivo e inofensivo para el cuerpo humano (Stone, 2005). Las imágenes de ultrasonido son adquiridas por medio de una sonda, la cual contiene un arreglo de cristales piezo-eléctricos que emiten ondas acústicas a frecuencias más altas que el rango audible[2] y a partir de las propiedades de reflexión de la onda de sonido la imagen es construida, la figura 4 muestra dos imágenes que sirven para entender el principio de funcionamiento. En primer lugar, se observa cómo el transductor piezo-eléctrico envía una onda, y por medio del tiempo que tarde una onda en chocar y regresar (eco) a la sonda se puede reconstruir una línea de la imagen. En segundo lugar, para obtener la imagen completa cada cristal crea una línea de *'scan'*, que permite mapear las estructuras cercanas en otras direcciones, como muestra la figura 4 (abajo). Así, con el conjunto de líneas se reconstruye la totalidad de la imagen. Grandes cambios en la densidad por la que viaja la onda crean un fuerte eco (Stone, 2005), por ejemplo la superficie superior de la lengua con el hueso del paladar, esta interfaz entre la lengua-aire-paladar, representa el 99 % de la imagen de ultrasonido (Stone, 2005). En la figura 5, se puede observar una imagen de ultrasonido. Quizás para el lector se le dificulte distinguir el contorno de la lengua, pero esto se debe a la naturaleza del método de ultrasonido que agrega un ruido granular llamado *speckle*, el cual degrada la calidad de la

---

[2] Típicamente en el orden de los MHz (Stone, 2005)

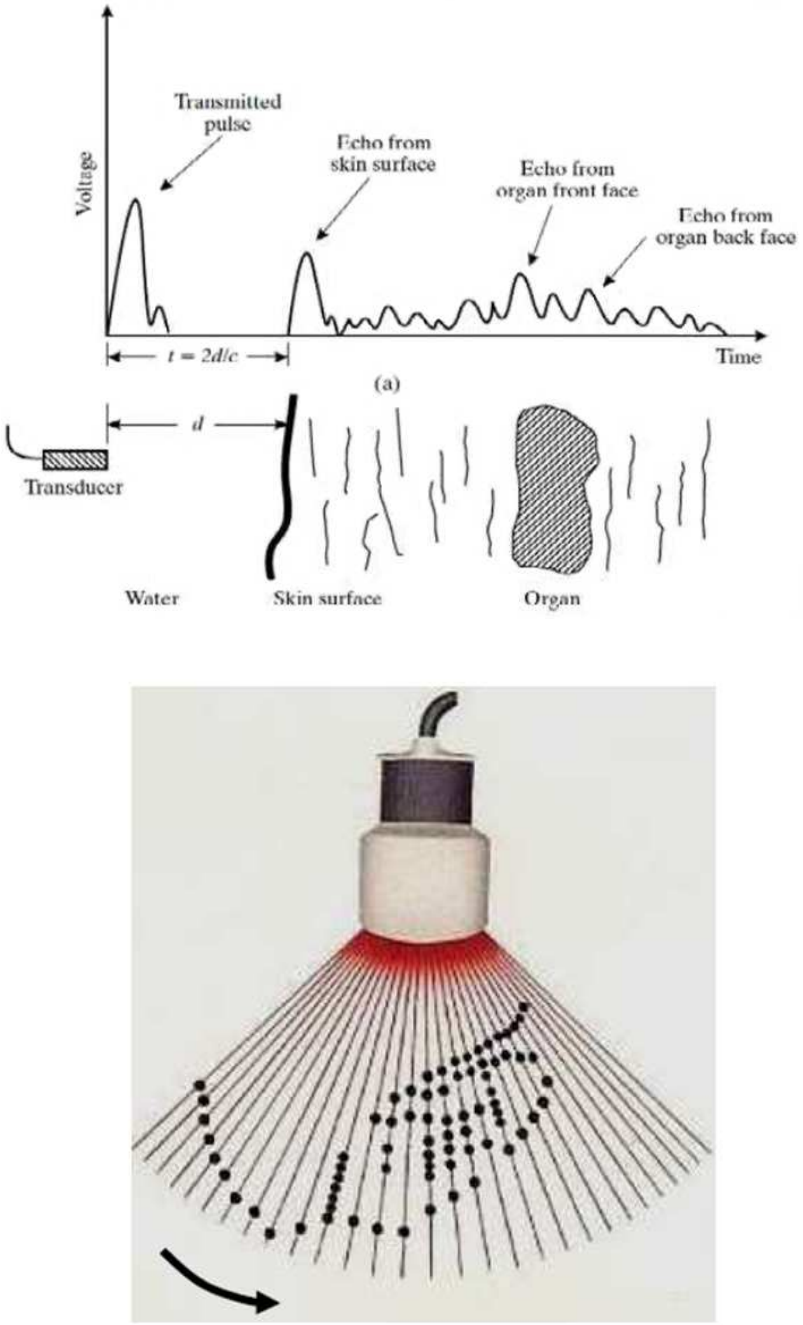


Figura 4. (arriba) *Nota:* Funcionamiento de un transductor piezo-eléctrico para producir la imagen de ultrasonido. Adaptado de (of Manitoba et al., 2019). (abajo) Cada cristal piezo-eléctrico produce una línea de *scan*, referencias de las imágenes al pie de página. Adaptado de (UTI, 2019).

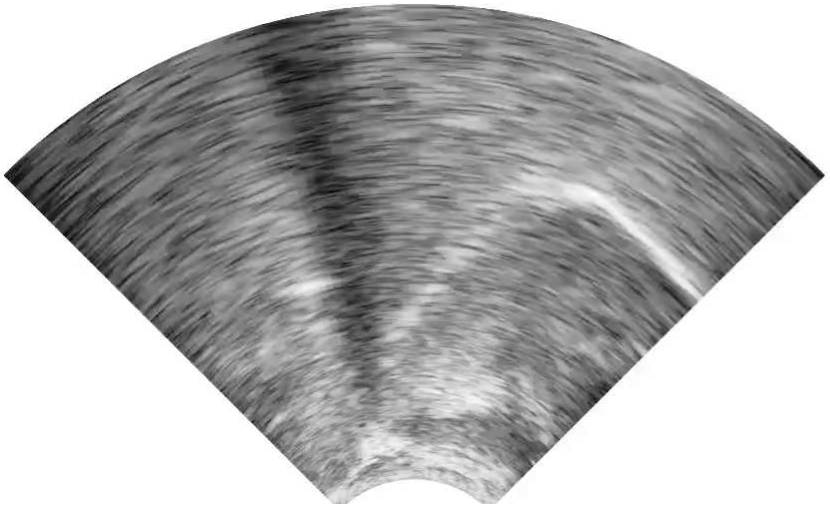


Figura 5. Imagen de Ultrasonido con ruido granular o *speckle*

imagen. Generalmente se usan filtros anisotrópicos, de variación total, con wavelets (Michailovich and Tannenbaum, 2006) o un promedio entre una secuencia de imágenes (Wang et al., 2015) para tratar este problema. Además la calidad de la imagen también puede variar entre sujetos, ya que el contenido graso de la lengua puede atenuar la onda de ultrasonido reflejada (Xu et al., 2016a; Stone, 2005).

2.3. Esquema de la tesis

Para llevar a cabo este estudio, primero se realiza la adquisición de información articulatoria. Esta se obtiene a partir de vídeos 2D de ultrasonido del tracto vocal. Luego es segmentado el contorno superior de la lengua. En este punto, se plantea como primer objetivo representar el contorno de la lengua mediante un conjunto reducido de parámetros. Luego, estos parámetros son utilizados como entrada a un sistema de verificación de hablantes, y el desempeño del sistema es medido con el fin de determinar la capacidad discriminante en las personas de los parámetros articulatorios planteados.

El objetivo principal es evaluar y analizar el uso de la información articulatoria del movimiento de la lengua en los sistemas de verificación de hablantes, aunque además se realiza una prueba adicional de identificación. Nuestro plan para conseguir este objetivo, a grandes rasgos está dividido en tres actividades:

1. Disponer de una base de datos de señales acústicas (voz) en paralelo con la información articulatoria proveniente de una sonda de ultrasonido.

2. Analizar el potencial de la información articulatoria para discriminar a las personas.

3. Comparar el desempeño de sistemas de verificación con información articulatoria

Se ha recolectado una base de datos de voz e imágenes del contorno superior de la lengua con ayuda de una sonda de ultrasonido. La motivación de utilizar la técnica de ultrasonido es: que permite capturar en tiempo real el movimiento del contorno midsagittal de la lengua durante el habla, de una forma no invasiva e intrusiva, a diferencia de otras técnicas como EMA. Además, la portabilidad y el bajo costo, la han hecho muy llamativa para la investigación en el campo del habla (Qin et al., 2008; Preston et al., 2017; Csapó et al., 2017; Xu et al., 2017). Así también, el Grupo CEMOS de la Universidad Industrial de Santander, ha realizado la adquisición de la sonda de ultrasonido PI 7.5 MHz 12-15 fps (*Speech Language Pathology ultrasound system*). Una descripción completa de la adquisición y detalles de la base de datos se encuentra en el capítulo 4. Luego, se proponen dos estrategias para representar la información articulatoria contenida en las imágenes de ultrasonido, tal como se expone en la sección 3.2.1. Una primera aproximación para analizar el potencial de la información articulatoria para discriminar a las personas es entrenar un sistema

de identificación de hablantes. Para esto, la información acústica es representada mediante coeficientes MFCC; La información articulatoria es representada mediante los puntos y coeficientes de un polinomio que se ajustan a la curva que describe el contorno de la lengua. Los experimentos de identificación se realizan utilizando mezclas Gaussianas e i-vector. De manera similar el experimento de verificación de hablantes se lleva a cabo utilizando GMM-UBM e i-vector + PLDA. El desempeño logrado por los sistemas de identificación y verificación sirven como indicador para medir la capacidad de la información articulatoria para discriminar hablantes.

Durante la revisión de la literatura correspondiente, no se encontraron estudios que usen imágenes de ultrasonido como información articulatoria para la verificación y/o identificación de hablantes. Menos aún, se encontró una base de datos de acceso pública de imágenes de ultrasonido y voz sincronizadas. De manera que, los resultados obtenidos son difícilmente comparables con otros trabajos. Como alternativa, comparamos los resultados obtenidos con respecto a los trabajos (Li et al., 2016a; Zhang et al., 2017b), los cuales presentan un objetivo similar, pero utilizando otras formas de información articulatoria. La información articulatoria con la que estos trabajos cuentan, fue adquirida con un sistema EMA, lo cual difiere enormemente de la adquirida con un sistema de ultrasonido.

En resumen, buscamos responder dos preguntas: Primera, ¿es posible discriminar a las personas con la información articulatoria? y segundo, ¿puede la información articulatoria mejorar el desempeño de los sistemas de verificación de hablantes?. La hipótesis de nuestro estudio es que la

información articulatoria útil para discriminar a las personas, pero además que si se fusionan con los tradicionales MFCC en el *framework* convencional GMM-UBM, puede mejorar el desempeño del sistema de verificación del hablante, esto se debe a los movimientos articulatorios, que reflejan la variabilidad de los hablantes (Li et al., 2016a). Para probar esta hipótesis, comparamos el desempeño logrado por los experimentos de verificación e identificación cuando son entrenados con vectores de característica acústicos, articulatorios y la fusión de ambos.

3. Sistema de reconocimiento de hablantes

En este capítulo se documenta la revisión de la literatura en los dos aspectos generales que conforman la tarea de reconocimiento de hablantes. En la primera parte muestra los enfoques utilizados en la representación de la señal acústica de habla y los parámetros articulatorios. En segundo lugar, se puede encontrar una clasificación de los métodos que han sido utilizados para modelar a los hablantes. Finalmente, se describen las mezclas Gaussianas y los *i-vector*, dos técnicas que son empleadas para resolver la pregunta de investigación del presente trabajo.

3.1. Representación de la señal hablada mediante parámetros acústicos

La señal de voz se adquiere mediante un micrófono, sin embargo, gran parte de estos datos es información redundante o que no hace un aporte en la discriminación entre hablantes. Por esto, se utiliza un proceso de extracción de características donde se obtienen aquellos parámetros, de cantidad reducida, que representen el comportamiento de la señal de voz. Estos parámetros se almacenan en un vector de características. Adicional a esto, factores como el ruido ambiental o del canal afecta la señal de voz y la selección de características robustas se convierte en una tarea realmente importante para un sistema de reconocimiento y verificación de hablantes.

Los parámetros utilizados en voz, idealmente, deberán cumplir con las siguientes condiciones (Rose, 2002): 1) gran variabilidad entre-hablantes y baja variabilidad intra-hablante; b) ser robustos ante condiciones de ruido y distorsión por efectos del canal; c) que ocurran de mane-

ra frecuente y natural en el habla; d) fáciles de medir; e) que sean difíciles de imitar por otras personas; f) que no se vean afectados por cambios en la salud y la edad. Adicionalmente, por motivos de estimación de parámetros, la cantidad de características de voz ha de ser pequeño, pues los modelos estadísticos tradicionales no pueden tratar con dimensiones altas (Kinnunen and Li, 2010), Pues el número de muestras requeridas para realizar una estimación confiable de la función de densidad que siguen las características (en la etapa de entrenamiento), crece exponencialmente con la dimensión del vector de características. A este problema se le denomina "la maldición de la dimensionalidad"(Kinnunen and Li, 2010). Las características usadas convencionalmente en reconocimiento de hablantes, pueden ser clasificadas en: espectrales, prosódicas y de alto nivel.

Características espectrales: Se caracterizan por extraer coeficientes a partir del espectro sobre ventanas de tiempo corto. Son los parámetros comúnmente utilizados en tareas relacionadas con verificación e identificación de hablantes. Los MFCC (Gonzalez-Rodriguez et al., 2006), son las características más exitosas y populares, están relacionados con la envolvente espectral, la cual entrega información acerca de la forma del tracto vocal y han mostrado ser de bastante utilidad en propósitos de reconocimiento del hablante. Sin embargo, estas características se ven afectadas por el ruido (Zhang et al., 2017b; Reynolds, 1994; Rao and Sarkar, 2014). No solo los MFCC entregan información de la forma del tracto vocal, están además los PLP (*Perceptual Linear Prediction*) (Hermansky, 1990), LPCC (*Linear Prediction Cepstral Coefficient*) (Makhoul, 1975), entre otros. En cuanto a sistemas comerciales, el software BatVox (AGNITIO, 2015), es uno de los mas

ampliamente utilizados y utiliza a modo de representación de la voz los parámetros MFCC.

Características prosódicas: Están relacionadas con el estrés sobre las sílabas, patrones de entonación, tasa de habla y ritmo. El parámetro prosódico más importante es la entonación (Rao and Sarkar, 2014), el cuál es caracterizado por el contorno de la frecuencia fundamental (*Fo*). En (Mary and Yegnanarayana, 2008; Adami et al., 2003; Ferrer et al., 2007) se pueden encontrar otros ejemplos en los que la utilización de los parámetros prosódicos en sistemas de reconocimiento de hablantes ha sido efectiva. Uno de estos es la combinación de parámetros relacionados con la *Fo* y los espectrales ha mostrado ser relevante para mejorar el desempeño (Kinnunen and Li, 2010; Koolagudi et al., 2012) en presencia de ruido. Sin embargo, las características prosódicas no han sido muy utilizadas en reconocimiento de voz a causa de que la extracción de la información prosódica es considerada una tarea bastante exigente (Rao and Sarkar, 2014), y además el hablante puede manipular deliberadamente estos parámetros (Rao and Sarkar, 2014).

Características de alto nivel: En este tipo de parámetros se aprovecha el tipo de palabras y los sonidos que los hablantes utilizan en una conversación, para determinar así su identidad. Las más comunes son los patrones fonéticos (Campbell et al., 2004), palabras (Doddington, 2001), los gestos prosódicos (Adami et al., 2003) y los movimientos articulatorios (Leung et al., 2006). Particularmente en los trabajos (Zhang et al., 2017b; Li et al., 2016a), se extraen parámetros articulatorios continuos (trayectorias de órganos articulatorios/articuladores), a diferencia de (Leung et al., 2006) dónde se extraen clases (discretas) para describir los movimientos o posiciones de los articuladores. Lo anterior muestra, que se pueden adoptar dos tipos de parámetros articulatorios:

1) continuo, a partir del desplazamiento de los articuladores y 2) discreto, agrupando en clases la configuración de los articuladores.

En síntesis, se han estudiado un gran número de características en distintos niveles, y cada una de estos parámetros aprovecha diversa información producida en el habla, además, han mostrado servir en tareas de verificación de hablantes. Particularmente este trabajo se enfoca en los parámetros articulatorios continuos, y, la eficiencia de estos se compara con los parámetros MFCC. Por lo tanto, no se profundizará en la etapa del modelo de los hablantes y bastará con una breve descripción en la sección 3.3. A continuación, se muestra en detalle las características utilizadas en este trabajo.

Mel Frequency Cepstral Coefficient - MFCC. La técnica de coeficientes cepstrales en la escala de Mel es un método ampliamente utilizado para obtener los vectores característicos de la señal de voz (Patiño Saucedo et al., 2015). Dicha técnica se inspira en el funcionamiento de la cóclea o caracol durante el proceso de la audición humana. En general, la cóclea no puede discernir la diferencia entre dos frecuencias cercanas; cuyo fenómeno se hace más pronunciado a medida que la frecuencia se hace mayor generando un comportamiento aproximadamente exponencial. Por lo tanto se hace uso de la escala de Mel, la cual se aproxima al comportamiento del oído humano. La conversión de frecuencias de una escala lineal a la escala Mel esta dada por (Huang et al., 2001):

$$M(f) = 1125 \ln(1 + f/700) \tag{1}$$

Para el calculo de los MFCC, típicamente se hace por segmentos o ventanas de audio del orden de 20 a 30 ms, desplazadas una respecto a la otra en 10 ms. Luego, sobre estas ventanas se aplican los siguientes pasos:

1. Se calcula la densidad espectral de potencia

2. Se aplica, en el dominio de la frecuencia, un banco de R filtros al espectro de potencia y se obtienen valores de energía asociados a cada rango de frecuencia del banco de filtros, lo cual genera una aproximación del contenido espectral de la señal. Estos filtros, de tipo pasa-banda, están traslapados y distribuidos a lo largo del rango de frecuencias de la señal de voz. Las frecuencias centrales de estros filtros son determinadas de tal manera que todos los filtros compartan el mismo ancho de banda, pero en la escala MEL.

3. se toma el logaritmo de las energías del banco de filtros para formar un vector de longitud R

4. se aplica una transformación discreta del coseno al vector anterior con el fin de decorrelacionar las energías ya que los filtros se traslapan entre si (Patiño Saucedo et al., 2015)

De otra parte, se ha observado un mayor rendimiento en los sistemas de reconocimiento de voz si se calculan no solo los coeficientes MFCC sino también sus primera y segunda derivadas. Con ello se busca además obtener información acerca de la dinámica de la voz de la persona en cuestión (Ichikawa et al., 2010). En particular, estas derivadas se conocen como los componentes de velocidad Δ-MFCC y aceleración ΔΔ-MFCC (Schafer, 2008). Así, el vector de características reúne a los coeficientes MFCC, Δ-MFCC y ΔΔ-MFCC junto al valor de la energía de la señal.

Estos vectores de características son normalizados para obtener media cero y varianza unitaria.

Teniendo en cuenta que en las muestras de audio es común encontrar silencios, que corresponde a información generalmente irrelevante. La mayoría de sistemas incluyen un detector de energía (VAD - *Voice Active Detector*) que tiene como finalidad eliminar los silencios, el cual esta basado en los valores de energía a través del tiempo. Para seleccionar los segmentos correspondientes a la voz (no silencios), se usa una aproximación basado en el valor del *log* de la energía (Biosecure, 2015):

1. Los valores normalizados del *log* de la energía son modelados mediante distribuciones Gaussianas.

2. Aquellos valores de mayor energía son tomados para pasos posteriores mientras que los de menor energía se descartan.

Finalmente las características son normalizadas. La normalización es una técnica de compensación que incrementa la robustez del sistema. Debido a los diversos factores que pueden degradar el rendimiento del sistema tales como condiciones de grabación, ruido y variabilidad del hablante, entre otras, se aplican técnicas para mitigar dichos efectos aproximando la representación de parámetros a distribución gaussiana o normal. Dentro de este tipo de técnicas se mencionan: RASTA (RelAtive SpecTrA) (Hermansky and Morgan, 1994), CMS (*cepstral mean subtraction*) (Naik, 1995) y CMVN (*cepstral mean variance normalization*) (Barras and Gauvain, 2003). En el presente trabajo se utiliza normalización por CMVN, la cual tiene la propiedad de reducir efectos ocasionadas por

la variabilidad del canal.

De otra parte, para mejorar el desempeño del sistema de reconocimiento se emplean técnicas de transformación sobre las características ya obtenidas. Entre estas técnicas se encuentran el JFA (*Joint Factor Analysis*), FA (*Factor Analysis*), PCA (*Principal Component Analysis*) y los i-vector (ver sección 3.3.4, entre otras. En general, estos métodos realizan transformaciones de tipo lineal sobre el espacio de entrada de características de voz con el fin de obtener representaciones que cumplan con criterios adecuados para ser usados en la representación de los modelos de hablantes. Uno de los criterios más útiles para este tipo de tareas es aquel en el que se busca un espacio de salida en el cual las nuevas características están no correlacionadas entre sí. Es decir, cada característica de salida representa una porción de información contenida en la señal de voz.

3.2. Representación de la señal hablada mediante parámetros articulatorios

La información articulatoria puede ser recolectada por medio de: Rayos X, resonancia magnética (MRI), *Electromagnetic Articulography* EMA, ultrasonido (UTI) y con la captura de vídeo del movimiento de los articuladores externos (labios), entre otros métodos. Una vez se tienen los datos articulatorios y al igual que la señal acústica, se extrae la información relevante en un set de parámetros reducidos, es decir, se extraen las características. En cuanto a este procedimiento, podríamos clasificar en dos tipos de características articulatorias: Discretas y Continuas. Para la primera se utiliza el término *características articulatorias* y para la segunda *parámetros articulatorios*, cabe aclarar que solo emplearemos estos términos en esta sección 3.2, pero lo que respecta a todo el trabajo entiéndase como características articulatorias las extraída en forma continua. Las

características articulatorias corresponden a categorías provenientes de la separación de la voz en segmentos dependiendo del punto de articulación (labiales, dentales, alveolares ,...) y modo de articulación (fricativas, explosivas, ...), las cuales son comúnmente obtenidas mediante clasificadores aplicados sobre la señal acústica.

En lo que se refiere al uso de características articulatorias, se han realizado combinaciones de características espectro-temporales con características articulatorias. Por ejemplo en (Leung et al., 2004) se emplean características articulatorias que se derivan de las características espectrales para un sistema de verificación del hablante. De acuerdo a los resultados muestran que las características articulatorias contiene información especifica de los hablantes que son de gran utilidad. Así, la fusión de características articulatorias con los MFCC, mejora el desempeño de los sistemas de verificación del hablante. En (Lombart et al., 2014) se extrae la información articulatoria de la señal acústica de la voz, y se realiza una fusión con los modelos acústicos estándar para obtener un mejor modelo híbrido acústico; por otra parte, se estudia cuales señales de entrada del sistema son mejores para una buena extracción de información articulatoria. Luego, esta información se fusiona con un modelo acústico que es obtenido con redes neuronales.

De otra parte, el término parámetro articulatorio se usa para denotar información de tipo continua; por ejemplo, la posición de los articuladores o la forma del tracto vocal. En los últimos años se ha probado la utilidad de los parámetros articulatorios en problemas de reconocimiento de voz (Ghosh and Narayanan, 2011), reconocimiento de emociones (Kim et al., 2012); y también en

verificación de hablantes (Zhang et al., 2017b; Li et al., 2016a); donde se utiliza directamente la información articulatoria adquirida por un sistema EMA [3].

En síntesis, la información articulatoria ha sido relevante en el campo del reconocimiento de voz y de hablantes, y ha mostrado ser ademas de una alternativa un complemento a las características tradicionales acústicas. Por otra parte, también se pueden extraer parámetros articulatorios a partir de las imágenes de ultrasonido, tal como se propone en el presente trabajo. Para esta tarea han sido utilizados enfoques como EigenTongues (Hueber et al., 2007), dónde se hace una representación por medio de las componentes principales (PCA), o el uso de los contornos de la lengua extraídos de las imágenes de ultrasonido (Hueber et al., 2007; Qin et al., 2008). La sección 3.2 muestra como son extraídos los parámetros articulatorios que son utilizados para estimar los modelos de los hablantes del estudio presente. En la siguiente sección, se presentan algunos enfoques utilizados para estimar estos modelos, sin embargo, solo se profundizará en los empleados para los experimentos de verificación.

Extracción del contorno de la lengua. A partir, de la secuencia de imágenes 2D obtenidas por el sistema de ultrasonido, se propone extraer parámetros articulatorios en dos etapas: 1) Segmentar el contorno de la lengua y 2) Representar el contorno de la lengua con los coeficientes de un polinomio, ó, usando 5 puntos del mismo contorno.

---

[3] *Electromagnetic Articulography*; el cual consiste en localizar bobinas en articuladores de la boca como la lengua y los labios. Así, las bobinas durante el habla producen un campo electro-magnético que induce un corriente en los sensores puestos en los articuladores, y así puede ser registrada la trayectoria de estos, mas información de este sistema en (Schönle et al., 1987)

La etapa de segmentación del contorno de la lengua se realiza con ayuda del software EdgeTrak (Li et al., 2005). El procedimiento es realizado de la siguiente manera: primero, cada imagen de ultrasonido es cargada al software, luego se ajusta el área de interés (donde se encuentra el contorno de la lengua), enseguida se inicializa con 6 puntos a lo largo de la lengua y finalmente el software aplica un algoritmo basado en contornos activos (Kass et al., 1988) que permite obtener el contorno completo; el cual, en el presente trabajo está conformado por $N = 50$ puntos por cada imagen. En este paso final y similar a (Qin et al., 2008), observamos que en la práctica el algoritmo perdía el trayecto de la lengua en muchas ocasiones. Por este motivo, fue necesario verificar manualmente imagen por imagen, y en los casos pertinentes se corrigieron los contornos. Cabe aclarar, que el número de fallos del software utilizado no fue cuantificado en este trabajo. Y tampoco han sido extraídos contornos sin previa inspección. Teniendo en cuenta que esta etapa de etiquetado semi-automático constituye grandes esfuerzos (como trabajo futuro se podría utilizar una herramienta con mejor desempeño y así hacer ésta etapa más eficiente), EdgeTrak ha servido como una interfaz amigable al usuario, que permite obtener los puntos que conforman el contorno de la lengua. En la figura 6 se puede ver un ejemplo del contorno extraído a una imagen de ultrasonido. Por otra parte, el contorno es exportado en un archivo como una matriz (x_m, y_m), dónde $m = \{1, 2, ..., N\}$, y que es utilizado en la extracción de características propuestas.

3.2.1. Extracción de características Articulatorias. El contorno superior de la lengua se describe con una matriz de puntos que indican la ubicación del contorno dentro de la imagen de US. Por ejemplo, en (Barberena et al., 2017) es trazado el contorno con 42 puntos para encontrar anomalías en algunos fonemas; en (Ménard and Noiray, 2011) se utilizan 100 puntos como con-

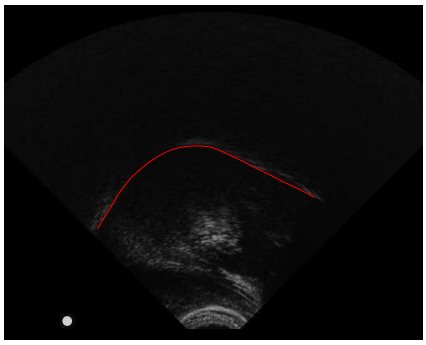


Figura 6. Ejemplo: Contorno extraído de una imagen de la lengua 2D-ultrasonido.

torno de la lengua para desarrollar gestos en francés; Y en (Jaumard-Hakoun et al., 2016; Csapó and Lulich, 2015; Qin et al., 2008) el contorno es descrito con 30, 41 y 24 puntos, en diferentes contextos. Entonces se decidió por utilizar 50 puntos para la extracción del contorno de la lengua.

Teniendo un conjunto de $N = 50$ puntos (x,y) que forman el contorno de la lengua el siguiente paso es representarlo empleando un grupo reducido de parámetros, cómo se muestra en la Figura 7. Para ello se proponen dos enfoques:

- Se ajusta un polinomio de orden 6 a partir de la curva que forma el contorno de la lengua. Se obtienen 7 coeficientes que a su vez son concatenados con el punto inicial (x_i) y final (x_f) en el eje horizontal del contorno, a modo de referencia. Generando un vector de características $p_0, p_1, ..., p_6, x_i, x_f]$ con dimensión 9 para cada imagen de ultrasonido. Los coeficientes son calculados utilizando la función *polyfit* de Matlab.

- Se toman 5 puntos igualmente espaciados con coordenadas ($x_i, y_i; i = 1, 10, 20, 30, 40, 50$), a lo largo del contorno de la lengua. Formando un vector de características ($x_1, x_2, ..., x_5, y_1, y_2, ..., y_5$), con dimensión 10.

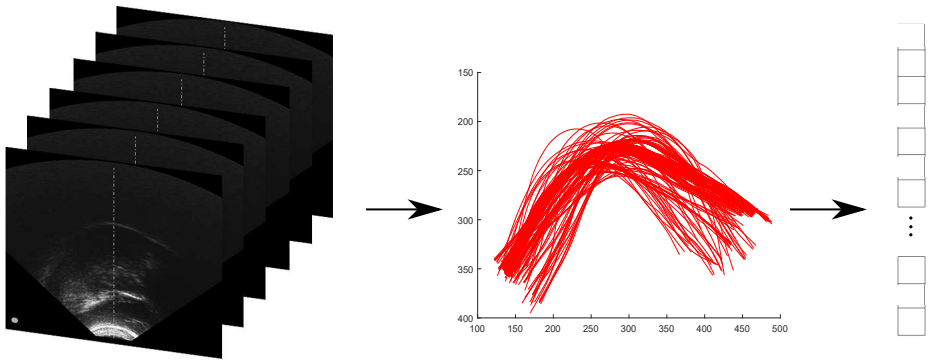


Figura 7. Diagrama de representación de la información articulatoria (imágenes de ultrasonido).

En base al trabajo desarrollado por Stolar y Gick (Stolar and Gick, 2013), que cuantifica la curvatura de la lengua al modelar la forma con una aproximación polinomial de orden 7; por esta razón, se decidió por experimentar con un polinomio de orden 6. De otro lado, el enfoque de tomar los 5 puntos a lo largo del contorno de la lengua está con la metodología de adquisición EMA, la cuál obtiene datos de la posición en algunos puntos de la lengua, generalmente entre 3 y 7 puntos. Adicional a esto, en (de Vasconcelos et al., 2009) representan el contorno de la lengua con 7 puntos y en (Qin et al., 2008) es reconstruido el contorno a partir de 3 puntos. Lo que podría sugerir que en éste intervalo puede encontrarse el número de puntos relevantes para describir la forma de la lengua.

3.3. Modelado de hablantes

En pocas palabras la tarea de verificación de hablantes se resume en responder si un segmento de habla *M* le corresponde o no a una persona *P*. Y para realizar dicha tarea, se construyen modelos de cada hablante con datos de entrenamiento cómo se indica en el esquema 3.

Se utilizan diferentes técnicas de modelado para capturar la distribución de estos paráme-

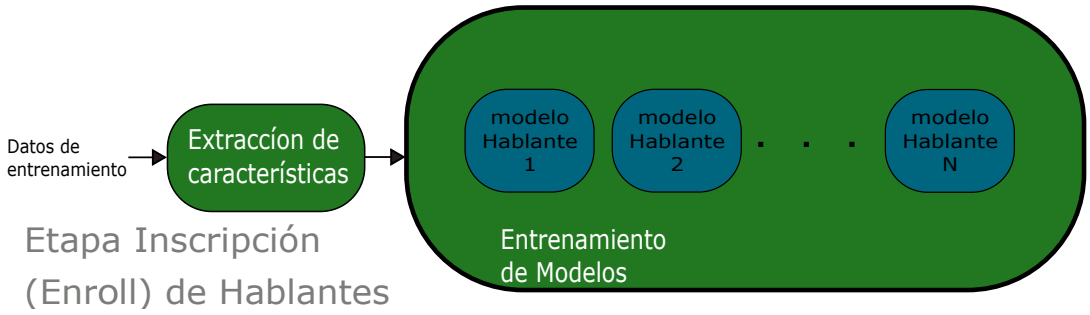


Figura 8. Esquema de inscripción de los hablantes.

tros y así construir los modelos individuales de cada hablante. Esto corresponde a la etapa de entrenamiento e inscripción de hablantes, cómo se indica en el esquema de la Figura 8. Algunas de estas técnicas pueden ser clasificadas de acuerdo a la naturaleza para modelar la distribución de las características (paramétrico y no-paramétrico) o de acuerdo al enfoque en el paradigma de entrenamiento (generativo ó discriminativo) (Rao and Sarkar, 2014).

Modelado paramétrico y no-paramétrico: En el modelado paramétrico se asume la función de densidad de probabilidad que siguen las características, así con los datos de entrenamiento solo se realiza un ajuste en la forma de la función.En contraste, el modelado no paramétrico busca aprender tanto la forma como los parámetros de la función de densidad a partir de los datos de entrenamiento. Algunas técnicas de modelado no-paramétrico en reconocimiento de voz son VQ(*Vector Quantization*) (Burton, 1987), DTW(*Dynamic Time Warping*) (Rabiner and Juang, 1993) y NN(*Nearest Neighbors*) (Higgins, 1990). Por otra parte, los modelos ocultos de Markov (HMM) (BenZeghiba and Bourlard, 2003) y las mezclas Gaussianas (GMMs) (Reynolds et al., 1995), son técnicas de modelado paramétrico ampliamente usadas en verificación de hablantes.

Modelado generativo y discriminativo: El objetivo del modelado generativo es estimar la distribución de las características de cada hablante. A diferencia del modelado discriminativo cómo las máquinas de soporte vectorial (SVM's) (Campbell et al., 2006a) ó las redes neuronales (NN's) (Farrell et al., 1994), que busca estimar la frontera que diferencia las clases (hablantes). Algunos ejemplos de modelado generativo son los ya mencionados GMM's, HMM's y VQ, entre otras. Sin embargo, existen trabajos en los que se combinan modelos generativos y discriminativos para problemas de reconocimiento de hablantes: GMM's + SVM's en (Campbell et al., 2006a) y (Campbell et al., 2006b) que utiliza los modelos GMM's a modo de supervectores de entrada a un clasificador SVM.

Hasta el momento solo se han presentado algunas de las técnicas más conocidas y utilizadas en verificación de hablantes, sin embargo entrar en detalle a cada una de ellas esta afuera del alcance de este documento. Ademas como se menciono anteriormente, el propósito de este trabajo es mostrar el potencial que tienen las características articulatorias en reconocimiento de hablantes. Por lo tanto, las secciones siguientes se limitan a describir las técnicas de modelado utilizadas en los experimentos de este estudio. Una de ellas son las mezclas gaussianas ó GMM's, pues ha sido ampliamente usada y considerada como un *benchmark* para el reconocimiento de hablantes texto-independiente moderno (Rao and Sarkar, 2014). A continuación, se describen esta función de probabilidad.

3.3.1. Filosofía UBM. La tarea de verificación de hablantes se puede resumir en una prueba de hipótesis: 1) H_0 : el sospechoso es la fuente de la grabación analizada, y 2) H_1 : alguna otra

persona es la fuente de la voz analizada. La hipótesis H_0 puede ser modelada con alguna estrategia mencionada en la sección anterior. Sin embargo, en esta sección el interés es por modelar la hipótesis H_1, en la que la muestra analizada no corresponde con el hablante en cuestión, es decir, la muestra de prueba pertenece a otra persona, que se conoce como impostor. Existen dos enfoques para modelar esta hipótesis de impostor. El primero, consiste en usar un conjunto de modelos de hablantes diferentes para cubrir el espacio de la hipótesis alternativa, sin embargo, la desventaja de este enfoque es que su desempeño depende usar un gran número de modelos de hablantes (Bimbot et al., 2004). El segundo enfoque, consiste en reunir un conjunto de muestras de voz de distintos hablantes y entrenar un solo modelo. Conocido como *Universal Background Model* (UBM) (Reynolds et al., 2000). El uso del UBM, se ha convertido en el enfoque predominante en los sistemas de verificación de hablantes (Bimbot et al., 2004). A continuación, se presenta con mayor detalle el entrenamiento del modelo universal con la presunción de que la distribución de características sigue una función de mezclas Gaussianas, conocido como GMM-UBM; y al final de la sección una medida que se utiliza para cuantificar la distancia entre estos dos modelos y que relaciona las dos hipótesis mencionadas.

3.3.2. Modelado basado en GMM. El siguiente apartado ha sido dedicado a la descripción de las Mezclas Gaussianas, pues son el punto de referencia del texto.

Mezclas Gaussianas - GMM's Una función de probabilidad en particular, que representa el comportamiento estadístico de los datos z_t, se puede aproximar mediante la suma ponderada de J

funciones $P(z_t;\theta^j)$, de la forma (Bishop, 2006):

$$P(z_t\cdot) = \sum_{j}^{J} \pi^j P(z_t;\theta^j) \tag{2}$$

donde θ^j es el vector de parámetros que describe $P(z_t;\theta^j)$. Los pesos π^j en (2) son valores escalares normalizados que cumplen $\sum_{j=1}^{J}\pi^j = 1$ y $\pi^j > 0$. Esto asegura que la mezcla descrita por (2) es una verdadera función de densidad de probabilidad (Yannis, 1996).

En el modelo dado por la ecuación (2), $P(z_t;\theta^j)$ corresponde a la distribución Gaussiana multivariada dada por la ecuación (3),

$$P(z_t;\theta^j) \;=\; \mathcal{N}(z_t;\mu_z^j,\Sigma_z^j) \;=\; \frac{1}{(2\pi)^{d/2}} \;\mid\; \Sigma^j \;\mid^{-\frac{1}{2}} \; \exp\left(-\frac{1}{2}(z_t-\mu^j)^{\mathsf{T}}(\Sigma^j)^{-1}(z_t-\mu^j)\right) \tag{3}$$

En este caso el conjunto de parámetros θ^j en (2) corresponde a la media y la covarianza, por tanto $\theta^j = \{\mu_z^j,\Sigma_z^j\}$.

Si se usa una cantidad suficiente de Gaussianas, además del ajuste adecuado de las medias, las covarianzas y los valores de los coeficientes de la combinación lineal, casi cualquier función de densidad de tipo continuo puede ser aproximada (Bishop, 2006). La función de densidad es parametrizada por los pesos, los vectores de medias y las matrices de covarianza de las funciones

Gaussianas . Estos parámetros se representan por,

$$\Theta = \left\{ \pi^j, \mu_z^j, \Sigma_z^j \right\}, \quad j = 1, \ldots, J \tag{4}$$

Una vez se establece la estructura modelo, lo que resta es determinar los parámetros del mismo modelo Θ; para lo cual se aplican procesos de optimización.

Entrenamiento del UBM. El elemento UBM es en esencia un gran modelo de mezclas Gaussianas (GMMs) que representa las propiedades de la voz de la población que se utiliza a modo de referencia y es ampliamente usado en sistemas de verificación del hablante de hoy en día (Hasan and Hansen, 2011). En los sistemas de reconocimiento basados en GMM (*Gaussian mixture model*) se entrena primero un modelo universal (UBM) que sirve como referencia para la posterior comparación. Para el entrenamiento de este modelo global se emplea el criterio ML (*Maximum Likelihood*) junto al algoritmo *Expectation Maximization* (EM). Este último consiste en realizar iteraciones con el fin de encontrar los parámetros del modelo de referencia planteado. Este proceso se realiza maximizando el resultado de la probabilidad calculada por el modelo dados los parámetros actuales denotados por θ. El algoritmo encuentra θ, el cual es el conjunto de parámetros optimizados por medio del siguiente criterio:

$$\theta = \arg \max_{\Theta} E(p(z_t, Y/\Theta)/z_t, \theta) \tag{5}$$

El algoritmo permite encontrar el *Maximum Likelihood* (ML) o la probabilidad *Maximum a Posteriori* (MAP) estimando los datos bajo una hipótesis. El ML para mezclas Gaussianas resulta de tres ecuaciones que generan los parámetros de la mezcla (Gannert, 2007),

$$\theta = (\pi_1, ...\pi_n, \mu_1, ..., \mu_n, \sigma_1, ..., \sigma_n) \tag{6}$$

$$\pi_i^{'} = \frac{1}{n}\sum_j^n p(i|z_j, \theta) \tag{7}$$

$$\mu_i^{'} = \frac{\sum_j^n z_j p(i|z_j, \theta)}{\sum_j^n p(i|z_j, \theta)} \tag{8}$$

$$\sigma_i^{'2} = \frac{\sum_j^n (z_j - \mu_i^{'})(z_j - \mu_i^{'})^T p(i|z_j, \theta)}{\sum_j^n p(i|z_j, \theta)} \tag{9}$$

Donde $\pi_i^{'}$ es el peso para la i-ésima distribución y $p(i|z_j, \theta)$ es el *likelihood* que genera z_j para la distribución i. Posteriormente se adaptan los modelos de cada hablante por medio del criterio MAP (*Maximum a posteriori*) basados en el modelo global obtenido (Gannert, 2007).

Para el caso de la adaptación MAP las ecuaciones son (Gannert, 2007):

$$\pi_i^{'} = \frac{r^{\pi} + N_i}{n + N r^{\pi}} \tag{10}$$

$$\mu_i^{'} = \alpha_i^{\mu} E_i(z_t) + (1 - \alpha_i^{\mu})\mu_i \tag{11}$$

$$\sigma_i^{'2} = \alpha_i^{\sigma} E_i(z_t^2) + (1 - \alpha_i^{\sigma})(\sigma_i^2 + \mu_i^2) - \mu_i^{'2} \tag{12}$$

Siendo N el numero de distribuciones Gaussianas, α es una constante calculada por un factor de

relevancia, r^p. Además las tres ecuaciones se encuentran bajo el criterio dado sobre un grupo de muestras pertenezcan al mismo hablante.

$$\alpha_i^p = \frac{N_i}{N_i + r^p} \tag{13}$$

Así se concluye con la etapa de entrenamiento, que corresponde a la construcción de los modelos de los hablantes y el de la población (UBM). La siguiente etapa corresponde a la prueba de verificación o *test*, donde se obtiene un puntaje de verificación para cada comparación entre un modelo y una muestra.

3.3.3. Razón de verosimilitud (Likelihood Ratio - LR). La prueba de verificación que cuantifica la diferencia entre el modelo y la muestra de *test* en cuestión. Sirve de base para tomar la decisión entre la hipótesis H_0 y H_1, ver sección 3.3.1 es el *Likelihood Ratio* (LR) ó razón de verosimilitud. Esta, se define a partir de la aplicación del teorema de Bayes. Primero, se establecen dos hipótesis: 1) H_0 : el sospechoso es la fuente de la grabación analizada, y 2) H_1 : alguna otra persona es la fuente de la voz analizada. E corresponde a la evidencia. Para tal caso, la expresión que relaciona las hipótesis con la evidencia es la siguiente (Gonzalez-Rodriguez et al., 2006):

$$\underbrace{\frac{Pr(H_0/E)}{Pr(H_1/E)}}_{\text{a posteriori}} = \underbrace{\frac{Pr(E/H_0)}{Pr(E/H_1)}}_{LR} \underbrace{\frac{Pr(H_0)}{Pr(H_1)}}_{\text{a priori}} \tag{14}$$

donde $Pr(A/B)$ denota la probabilidad condicional,siendo A y B cualquiera de las dos hipótesis H_0 o H_1 o la evidencia E . La probabilidad a priori representa la creencia antes de realizarse el experimento, la cual podría ser descrita por el juez, mientras que la razón de verosimilitud es una medida generada por la evidencia. El LR está definido por,

$$LR = \frac{Pr(E/H_0)}{Pr(E/H_1)} \tag{15}$$

En conclusión, la prueba de LR, permite cuantificar las diferencias y similitudes entre los hablantes conocidos y las muestras que se cuestionan. De esta manera, se obtiene un puntaje de verificación, que al final permite aceptar o rechazar la identidad aclamada. En la práctica de esta prueba es común utilizar el logaritmo del LR ó LLR (*Log-Likelihood Ratio*):

$$LLR = logPr(E/H_0) - logPr(E/H_1) \tag{16}$$

En este punto ha sido descrito lo que correspondería el sistema de verificación GMM-UBM empleando la Razón de verosimilitud cómo una medida de distancia entre modelos o clasificador. Un método alternativo son los i-vector, que en numerosos trabajos ha conseguido el mejor desempeño en sistemas de verificación. A continuación, se describe el método para la extracción de i-vector.

3.3.4. Modelado basado en i-vectors+PLDA.

Estado del arte - Situación actual. El uso de los i-vector ha dominado ampliamente las investigaciones en reconocimiento de hablantes (Dehak et al., 2011) en los últimos años. El excelente rendimiento, la compacta representación y el pequeño tamaño de modelo, son algunos de sus atributos que lo han convertido en el estándar actual, como lo indica la evaluación de la NIST 2014 (Greenberg et al., 2014). Los *identity vector* (i-vector) son una representación en dimensión reducida en un sub-espacio que contiene todas los atributos del hablante. Entonces un archivo de audio M es descompuesto en la forma:

$$M = m + Tx \tag{17}$$

donde m el super-vector de medias del modelo UBM, que representa el factor independiente al hablante y x el i-vector (Dehak et al., 2011) con distribución normal estándar $N(0,1)$, que contiene las propiedades únicas del hablante y T es una matriz rectangular que formal el espacio de variabilidad Total. Luego, teniendo m y utilizando el algoritmo EM, se estima el ML ara el entrenamiento de la matriz de variabilidad total T (Sadjadi et al., 2013; Garcia-Romero and Espy-Wilson, 2011).

Extracción de i-vector El procedimiento de extracción es: 1) la señal de audio, es procesada por un detector de voz o VAD. 2) Son extraídas características acústicas, típicamente se usan MFCC junto a sus derivadas. 3) Luego, esta secuencia de vectores de características son representadas por una distribución de mezclas Gaussianas, llamada UBM. 4) Los parámetros de esta distribución son transformados en un i-vector, usando una matriz variabilidad total T. 5) Al i-vector se le sustrae la media global, luego es escalado por la inversa de la raíz cuadrada de la matriz global de covarianza,

y se normaliza a 1 (Garcia-Romero and Espy-Wilson, 2011). También se pueden aplicar distintos métodos para reducir la variabilidad intra-clase cómo LDA y NAP (Campbell et al., 2006a), donde se proyecta a un espacio que maximice la información de una clase, en este caso el i-vector. Finalmente se obtienen los puntajes de verificación, para esto la función más simple es la distancia coseno entre el i-vector (promedio de los i-vector de la etapa de entrenamiento, representa el modelo del hablante) y el i-vector de prueba (Dehak et al., 2011), otra alternativa son las máquinas de soporte Vectorial (SVM) (Cumani et al., 2011) y DNN's (*Deep Neural Newtworks*) (Cumani et al., 2011). Sin embargo, la función utilizada en el estado del arte es la PLDA (*Probabilistic Linear Discriminant Analysis* (Prince and Elder, 2007; Garcia-Romero and Espy-Wilson, 2011)). Esto en cuanto a la etapa del modelado.

El enfoque de i-vector ha sido el estado del arte en los años recientes (ver sección 3.3.4). La extracción de los i-vector consiste en realizar un mapeo de las características tradicionales MFCC (obtenidos de las señales de audio) a un vector de menor dimensión (Dehak et al., 2011). Con este propósito se entrena el UBM, el cuál es usado para el calculo de las estadísticos de Baum-Welch los cuales se definen a partir de la probabilidad a posteriori $P(c|y_t, \Theta_{UBM})$ de cada componente de la mezcla Gaussiana c generada por el vector y_t que representa cada segmento de la muestra de audio. Los estadísticos necesarios de cada muestra de audio son obtenidos así:

$$N_c = \sum_{t=1}^{L} P(c|y_t, \Theta_{UBM}) \tag{18}$$

$$F_c = \sum_{t=1}^{L} y_t P(c|y_t, \Theta_{UBM}) - m_c N_c \tag{19}$$

Dónde L es el número total segmentos de una muestra de audio u y $c = 1, ..., C$ cada componente de mezcla Gaussiana utilizada en el modelo UBM. Además m_c corresponde a la media del modelo UBM.

PLDA. Teniendo los i-vector del conjunto de entrenamiento de cada hablante, se entrena un modelo PLDA (Prince and Elder, 2007), que trata de modelar las diferencias entre distintas clases (individuos). A partir de este punto, se relegan a ser una observación de un modelo generativo probabilístico (Garcia-Romero and Espy-Wilson, 2011). Así, una observación x (i-vector) puede ser descompuesta en (Garcia-Romero and Espy-Wilson, 2011):

$$x = \mu + \phi\beta + \varepsilon \tag{20}$$

Dónde $\mu + \phi\beta$, representa el componente especifico de cada hablante, siendo μ la media del *i-vector* previamente entrenado; ϕ el sub-espacio de *EigenVoices* y β es un vector de variables latentes que sigue una distribución normal y que representa la posición en dicho subespacio del individuo en cuestión; Por último ε, es un termino residual que explica la variabilidad no capturada a través de las variables latentes, y se define a partir de una matriz completa de covarianza Σ. Luego, se estima el ML del factor ϕ utilizando el algoritmo EM (Prince and Elder, 2007).

Por último, los puntajes de verificación son calculados con el LLR (ver sección 3.3.3) de

que dos i-vector correspondientes a la misma persona fueran generados por la misma variable latente de identidad β.

4. Resultados Experimentales y Discusión

4.1. Configuración del experimento

4.1.1. Datos. El set de datos que es utilizado en la parte experimental de este trabajo, ha sido creado por estudiantes de la Universidad Industrial de Santander (www.uis.edu.co).Los datos fueron tomados en un salón de clases de la institución mencionada anteriormente, donde se encontraban 3 personas: 2 encargados de la toma de datos y el sujeto a registrar. Una sesión de grabación es realizada así: Los encargados configuran un casco estabilizador para la sonda de ultrasonido. Luego es suministrado un gel conductor en la barbilla del sujeto. La sonda se posiciona en el casco estabilizador y haciendo contacto en la zona cubierta por el gel conductor. En seguida se hacen unas grabaciones de prueba mediante la cual, el hablante lee unas frases previamente establecidas y así los encargados se aseguran de una correcta toma de muestras. Para prevenir la fatiga del sujeto son recolectadas 15 frases seguido de un descanso y es brindado un refrigerio. Finalmente se recolectan las 15 frases restantes. El set de datos contiene 17 hablantes (9 mujeres y 8 hombres) etiquetados de la forma $\{f01, f02, ..., f09; m01, m02, ..., m08\}$. Los hablantes son originarios de la región de Santander-Colombia y se encuentran en un rango de edad de 20-24 años.

Es registrada de manera paralela la siguiente información: 1) La señal acústica, a 44800 Hz de frecuencia de muestreo; 2) Vídeo de ultrasonido del movimiento de la lengua en el plano mid-sagittal a 14 fps. Cada hablante registra un conjunto de 30 frases diferentes en español, cuya

distribución de fonemas se muestra en la figura 9. Los vídeos de ultrasonido son segmentados en imágenes 2D, frame por frame y a escala de grises con dimensiones (800x600 pixeles). Aproximadamente cada hablante registra 340 segundos de grabación, para un mínimo de 4000 imágenes por hablante. En total se obtienen 101663 imágenes entre todos los hablantes.

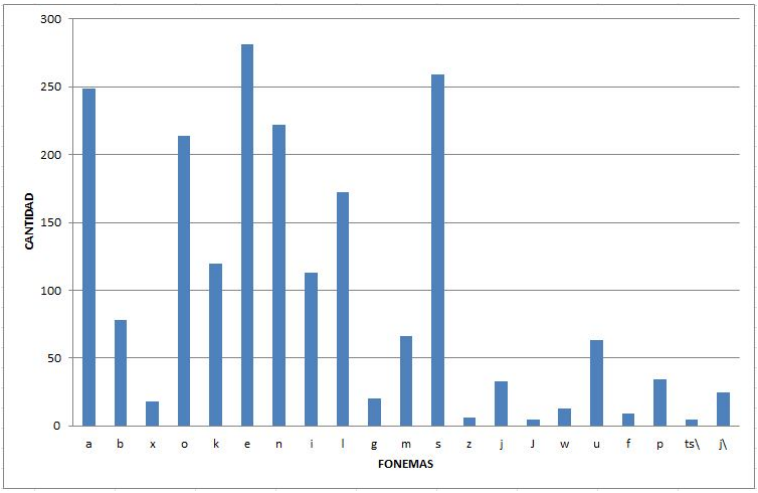


Figura 9. Histograma de los fonemas incluidos en la base de datos.

Se recolectó un set de datos balanceado en género (17 hablantes = 9 mujeres + 8 hombres), que consiste en 30 grabaciones por hablante. La adquisición de audio (voz) y vídeo de ultrasonido (contorno superior de la lengua) fue sincronizada. La señal de audio es adquirida por medio de un micrófono Shure SM58 (`http://www.shure.com/americas/products/microphones/sm/sm58-vocal-microphone`); y, un sistema de ultrasonido PI 7.5 MHz 12-15 fps (*Speech Language Pathology ultrasound system*, puede ser encontrado en el sitio web: `http://seemore.ca/portable-ultrasound-products/pi-7-5-mhz-speech-language-pathology-99-5544-can/`) es utilizado en la adquisición de vídeo de ultrasonido de la lengua. El montaje para la toma de datos se puede observar en la Figura 10. Un factor que puede limitar una buena toma de datos del movimiento de la lengua es la tasa de muestreo. En (Csapó et al., 2017) se utiliza un sistema que puede adquirir a 82 fps, en (Hueber et al., 2007) se utiliza una tasa de muestreo de 30 fps. Sin embargo, en distintos trabajos como (ABBS and COLE, 1982; Fontecave Jallon and Berthommier, 2009; Fontecave and Berthommier,

Hablante	# Sesiones	# Imágenes
f01	30	7132
f02	30	7296
f03	30	7522
f04	30	7352
f05	30	7236
f06	30	6958
f07	30	7558
f08	30	7131
f09	30	4219
m01	30	6813
m02	30	7271
m03	30	7375
m04	30	7193
m05	15	3161
m06	20	4176
m07	29	4587
m08	30	5157
	Total	**101663**

Tabla 1
Set de datos: Número de sesiones e imágenes registrados por hablante.

2009; Ghosh and Narayanan, 2010), se afirma que el 99 % de la dinámica de la lengua puede ser muestreada de 6-10 Hz. Adicional a esto, se muestra en (Lofqvist, 2007) que hablar de manera cadenciada en el momento de realizar la toma de datos, ayuda a disminuir el ancho de banda utilizado para muestrear el movimiento de la lengua.

Por otra parte, se pueden mitigar algunos errores o ruido en la posición de la sonda disponiendo de un casco estabilizador (Scobbie et al., 2008; Whalen et al., 2005). Esto permite libertad para el hablante y ademas garantiza uniformidad en la toma de muestras. Por este motivo, en el presente trabajo se utiliza un casco manufacturado por *Articulate Instruments* (Scobbie et al., 2008). Finalmente, las señales de audio se almacenaron en formato *.wav* y el



Figura 10. Recolección de Datos

vídeo de ultrasonido, es guardado en una secuencia de imágenes en escala de grises en formato *.jpg*.

4.1.2. Extracción de características acústicas. Para llevar acabo este procedimiento se utiliza una *Toolkit* llamada *HTK* (Young et al., 2002). La cual es una herramienta utilizada en investigación en el campo de reconocimiento de voz. Además esta diseñada para utilizarse como una interfaz de línea de comandos, y cada librería cuenta con sus argumentos particulares. Particularmente en la extracción de los MFCC, se utiliza la librería *HCopy* con el siguiente comando:

```
HCopy -A -D -T 1 -C hcopy_AVS_pruebas -S lista_para_HTK.lst
```

Dónde las opciones -A -D -T 1, son opciones comunes para todas las librerías de *HTK* y su función es para la impresión de datos en pantalla. La opción -C es para indicar el archivo de configuración y -S la lista de archivos que utiliza el comando, allí en esa lista se encuentran los archivos de entrada, es decir los audios en formato *.wav* y el nombre de los archivos de salida, los cuales serían los vectores de características MFCC, en formato *.HTK*. En el Anexo 2 se encuentra información más detallada, como lo es el archivo de configuración y las listas de archivos.

Luego, con el propósito de eliminar los segmentos de silencio, se utiliza la función *vadsohn* de la toolbox de Matlab *voicebox* (Brookes et al., 1997). De esta manera, se remueven las características contenidas en las ventana de silencio que contiene el archivo de audio respectivo. Finalmente la normalización CMVN es realizada mediante la toolbox *MSR Identity Toolbox* (Sadjadi et al., 2013) en Matlab. El *script* utilizado para realizar esta tarea puede ser encontrado en el Apéndice 3.

4.1.3. Fusión de Parámetros articulatorios y características espectrales. La Fusión de parámetros articulatorios se puede llevar acabo en dos instancias como se indica en el trabajo (Li et al., 2016a): A nivel de características y/o a nivel de puntajes de verificación. En el presente trabajo solo se implementa la fusión a nivel de características. Para esto, se concatenan los parámetros articulatorios con los MFCC, como se indica en la Figura 11. Cabe resaltar que la frecuencia de muestreo de los dispositivos de adquisición de información acústica y articulatoria es diferente 44100Hz y 14fps respectivamente. Para obtener el mismo número de características acústicas que articulatorias en el eje temporal, se amplía el tamaño de la ventana en la que se extraen los MFCC a 78 ms. Esto podría incurrir en una pérdida de la información dinámica en la señal de voz ampliando tanto la ventana temporal de

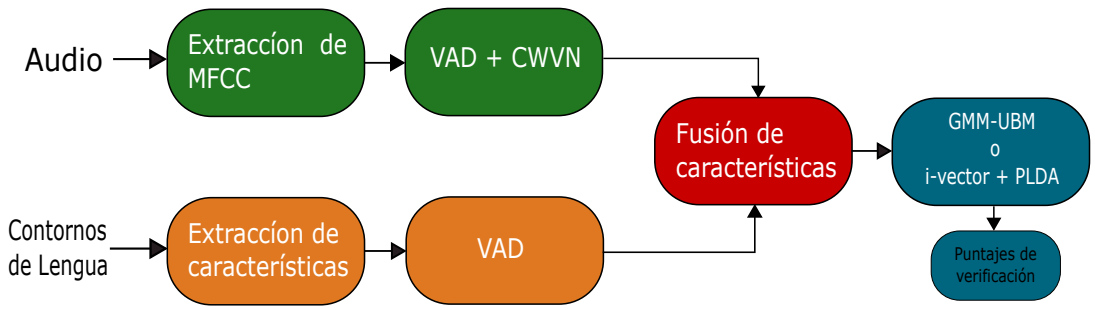


Figura 11. Esquema de la fusión de características. Cabe destacar que la fusión es realizada al cocatenar los vectores de características acústicas (MFCC) y articulatorias, después de descartar aquellos frames de silencio, que detecta el VAD. Por otra parte, para asegurar el mismo número de frames de MFCC que frames de características articulatorias, se realiza al fijar la ventana de tiempo de extracción de MFCC que la frecuencia de muestreo de la sonda de ultrasonido.

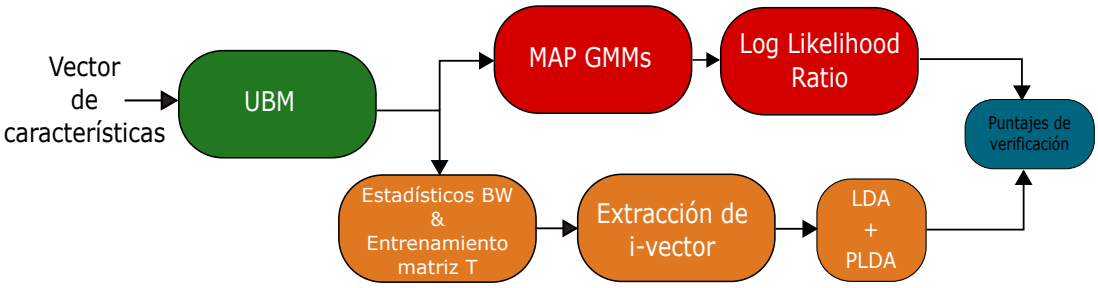


Figura 12. Diagrama de bloques del modelado de hablantes utilizando mezclas gaussianas (rojo) e i-vector (naranja).

análisis, como lo sugiere el trabajo de (Eringis and Tamulevičius, 2014), en el cuál se experimenta algunos parámetros para el cálculo de los MFCC, como el ancho de de la ventana, aplicado al reconocimiento de palabras. Los resultados indican que con el aumento del tamaño de la ventana empeora la tasa de acierto. También en (Kelly and Gobl, 2011) se hace un análisis del cálculo de los MFCC dependiendo del tamaño de la ventana de observación, en este caso para distinguir los sonidos vocales, fricativos sonoros y sordos. Para los dos primeros casos, los resultados indican que con el aumento de la ventana empeora la habilidad para detectar estos sonidos, sin embargo con las fricativas sordas mejora. Luego de utilizar el VAD, también son descartados aquellos *frames* (contornos) que corresponden a silencios, ver en el apéndice 4, para encontrar la implementación de la fusión de características.

4.2. Estimación de los modelos

El diagrama de bloques mostrado en la figura 12, indica los procedimientos en la estimación de modelos de mezclas Gaussianas (GMM) e i-vector. Así, Teniendo vectores de característica acústicos (MFCC), articulatorios y la fusión de ambos es entrenado un modelo global (UBM) con 16 distribuciones Gaussianas utilizando la *MSR Identity Toolbox*. En base al trabajo desarrollado por Reynolds (Reynolds et al., 1995), donde estudia el desempeño de un sistema de reconocimiento de hablantes a partir del número de componentes Gaussianas en un set de datos con 16 hablantes. Los resultados muestran un máximo de rendimiento con 16 componentes Gaussianas. Por otra parte, se realizaron experimentos con mas de 16 componentes, pero los modelos no lograban entrenarse completamente lo que reducía el desempeño a un error del 50 %. Por este motivo no fueron incluidos estos resultados en el trabajo y 16 fue el número de componentes seleccionadas.

Luego utilizando cada enfoque:

Mezclas Gaussianas: Los modelos individuales son adaptados a partir del UBM con el criterio MAP y el set de datos de entrenamiento (individual), utilizando un factor de relevancia de 10. Esta implementación se puede observar en el Apéndice 5.

i-vector: A partir del UBM, se extraen los estadísticos de Baum-Welch y es entrenada la matriz de variabilidad total (T) de dimensión 16. Posteriormente se extraen los i-vectors utilizando el set de datos de entrenamiento individual. Finalmente la dimensión de los *i-vector* es reducida a 14 con LDA. El modelo PLDA (entrenado con los datos del UBM) sirve para el cálculo de los *scores*. Las rutinas de implementación se pueden ver en el Apéndice 6

Luego se realizan los pruebas de identificación y verificación cómo se indica a continuación. Los puntajes de verificación e identificación son calculados con el *Log Likelihood Ratio*. El desempeño es calculado para cada sistema, es decir el conjunto de: vectores de característica de entrenamiento y pruebas; los modelos de los hablantes y las pruebas de verificación o identificación realizadas. El desempeño de referencia en ambas tareas es el obtenido por una configuración base, que corresponde a modelos de mezclas Gaussianas entrenados con vectores de características

MFCC.

El capítulo contiene principalmente los resultados obtenidos por los sistemas de verificación e identificación. Para esto, el desempeño que logra cada uno de los sistemas de identificación y verificación es tomado como un índice para evaluar la información articulatoria es importante definir la métrica para cuantificar el desempeño de dichos sistemas. Los experimentos de identificación y verificación fueron realizados con la MSR identity toolbox en MATLAB (Sadjadi et al., 2013). Es claro que contar con una estrategia para adquirir la información articulatoria es poco práctico, a menos que esté relacionada con aplicaciones basadas en interfaces silenciosas. Para resolver este problema se estima la información articulatoria mediante un mapeo acústico-articulatorio.

4.3. Identificación de hablantes

Tomar un vector de características de prueba y compararlo con todos los modelos de hablantes inscritos en el sistema, describe una prueba de identificación de hablantes. Son seleccionados 20 archivos de todos los hablantes (17) como set entrenamiento. El set de pruebas corresponde a 10 archivos de 15 hablantes, es decir, 150 archivos de test. No se han tenido en cuenta los hablantes $\{m05, m06\}$ cómo test. En total se realizan 2250 (150x15) comparaciones.

En la tarea de identificación de hablantes se compara un archivo de prueba con todos los modelos de hablantes inscritos, el puntaje más alto obtenido por alguno de esto modelos es identificado como el hablante al que corresponde la muestra de prueba. Entonces se presentan dos casos: 1) cuando la muestra de audio consigue el mayor puntaje con el modelo que corresponda y 2) cuando el puntaje más alto no corresponde al hablante al que pertenece la muestra. A partir de los experimentos positivos o cuando el sistema comete un error, se calcula la tasa de acierto. La cual se define como:

$$TasadeAciertos = 100 * \frac{IP}{IP + FI} \tag{21}$$

Dónde, IP se refiere al número de identificaciones realizadas exitosamente y FI a los errores o falsas identificaciones. La tabla 2 indica que sin importar si el método corresponde a GMM-UBM o i-vector + PLDA, las características acústicas presentan mejor desempeño que las articulatorias. Aunque el desempeño logrado por los sistemas que fueron

entrenados con los coeficientes del polinomio de la lengua en la tabla 2 es cercano al punto de referencia. Un factor que puede marcar la diferencia entre las características acústicas y articulatorias es la dimensión, pues, el vector que contiene los mfcc tiene una dimensión de 39 mientras que 1) y 2) de la tabla 2 tiene dimensión 9 y 10 respectivamente. Así que, este factor puede ser un motivo por el que los mfcc sigan obteniendo los mejores resultados, a pesar de esto, el desempeño logrado por la información articulatoria es solo un poco inferior (entre un 4 y 8 % con el método GMM-UBM), teniendo en cuenta que solo se dispone de una parte del tracto vocal. Adicional a esto, la tasa de aciertos con i-vector es menor, y esto puede suponer que no es suficiente la cantidad de datos para la extracción de los i-vector (Li et al., 2016b).

Características	**Método**	**Tasa de aciertos**
1) $p_0, p_1, ..., p_6, x_i, x_f$	GMM-UBM	**94 %**
2) $x_1, x_2, ..., x_5, y_1, y_2, ..., y_5$	GMM-UBM	90 %
3) $MFCC$	GMM-UBM	**98,67 %**
1) $p_0, p_1, ..., p_6, x_i, x_f$	i-vector + PLDA	92 %
2) $x_1, x_2, ..., x_5, y_1, y_2, ..., y_5$	i-vector + PLDA	68 %
3) $MFCC$	i-vector + PLDA	**98 %**

Tabla 2
Desempeño logrado por los sistemas identificación entrenados con características acústicas o articulatorias

Características	**Método**	**Tasa de aciertos**
1) $p_0, p_1, ..., p_6, x_i, x_f$ + MFCC	GMM-UBM	98 %
2) $x_1, x_2, ..., x_5, y_1, y_2, ..., y_5$ + MFCC	GMM-UBM	96 %
1) $p_0, p_1, ..., p_6, x_i, x_f$ + MFCC	i-vector + PLDA	98 %
2) $x_1, x_2, ..., x_5, y_1, y_2, ..., y_5$ + MFCC	i-vector + PLDA	92 %

Tabla 3
Resultados de los experimentos de identificación haciendo una fusión de características

Por otra parte, los resultados de la tabla 3 sugieren que el vector de características acústico-articulatorias

no mejoran el desempeño logrado por el sistema base o bien se estanca en el 98 %. Esto puede suceder, porque al concatenar el vector de características acústicas de dimensión 39 con un vector de características articulatorias de dimensión 9 ó 10, puede sobre dimensionar el problema de reconocimiento de hablantes. Además, puede que las pruebas que son 2250 no sean suficientes para evaluar la fusión de características.

4.4. Verificación de hablantes

Una prueba de verificación consiste en tomar un vector de características de *test* y compararlo con un modelo que representa a la población (UBM) y un modelo de un hablante objetivo. Así, la prueba de verificación indica la probabilidad de que el vector de características corresponda o no al hablante objetivo. Ahora, para la realización de los experimentos con los hablantes recolectados en la base de datos, se selecciona un conjunto de archivos o muestras de *test* que son transformados en vectores de características. Luego, los vectores de características son comparados con los modelos de hablantes objetivos, que corresponden al mismo hablante, para producir pruebas verdaderas (con etiquetas '1') y ademas se comparan con los demás modelos que sirven como impostores, de esta manera se generan pruebas falsas (con etiquetas '0'). Para nuestros experimentos son inscritos en el sistema 15 hablantes. De esta manera, se obtienen los puntajes de verificación. Los modelos individuales de cada hablante son adaptados utilizando 24 frases de 2 hablantes, diferentes a los hablantes del set de entrenamiento como se indica en la tabla 4. De otra parte, se toman las 6 frases restantes de los 2 hablantes utilizados en la fase de *enroll*. Estas 6 frases son utilizadas para llevar a cabo los test de verificación (diferentes a las sesiones utilizadas para la inscripción/*enroll* de usuarios).

Se usa validación cruzada, utilizando la misma cantidad de hablantes en set de entrenamiento y en test. Los hablantes del set de entrenamiento son intercambiados con los del set de pruebas para realizar la validación cruzada. Adicional a esto, se repite el experimento con el mismo UBM pero con 10 diferentes configuraciones aleatorias entre las frases utilizadas para entrenamiento y para test. La única restricción que hicimos, fue utilizar los hablantes $\{m05, m06\}$, sólo en la fase de entrenamiento (UBM) dado que, solo cuentan con 15 y 20 sesiones respectivamente (ver tabla 1). En total, para cada experimento se realizan 105 experimentos y 1050 pruebas, y el resultado se obtiene mediante el promedio del EER obtenido en cada prueba.

	UBM	Experimentos	
		Enroll	test
# Hablantes	15	2	2
# Sesiones	todas	24	6

Tabla 4
Uso del set de datos en el presente trabajo, en las fases de un SVH.

4.4.1. Experimentos de Verificación. En una prueba de verificación como se indica en 4.4 son comparados los vectores de características de *test* con los modelos de los hablantes inscritos en el sistema. De esta manera, se obtienen los puntajes de verificación. No obstante, el resultado del sistema consiste en dar respuesta si el archivo de prueba corresponde o no a la persona con la que ha sido puesta en comparación. Para esto se fija un umbral de decisión, sin embargo el umbral no tiene gran relevancia al momento de analizar el desempeño logrado por el sistema, puesto que este umbral es seleccionado en el punto en el que los falsos rechazos y las falsas aceptaciones se minimicen. Así, el desempeño de un SVH es cuantificado con el EER (*Equal Error Rate*), el cual es una de las medidas más comunes en sistemas biométricos, e indica el valor para el cual la proporción de falsas aceptaciones es igual a la proporción de falsos rechazos del sistema. Entre más bajo sea el EER, más preciso es el sistema biométrico.

Anteriormente, en la sección 4.4 se describe con mayor detalle el cálculo de los resultados de verificación por medio de validación cruzada. Así, el resultado de cada prueba dentro de la validación cruzada corresponde a un porcentaje EER. Así que el promedio de todos los EER resultantes proporciona el desempeño general del sistema.

Sistema Base-Referencia. Con el objetivo de tener un resultado de referencia se realiza un experimento con una configuración tradicional, es decir utilizando únicamente la información acústica, representada en los parámetros MFCC + Δ + $\Delta\Delta$ sobre ventanas de 25ms y un desplazamiento de 10ms. Los hablantes son modelados con GMM's y posteriormente se realizan los experimentos de verificación e identificación cómo corresponda. Finalmente los resultados obtenidos con las configuraciones alternativas se pueden comparar con el desempeño logrado por este sistema base.

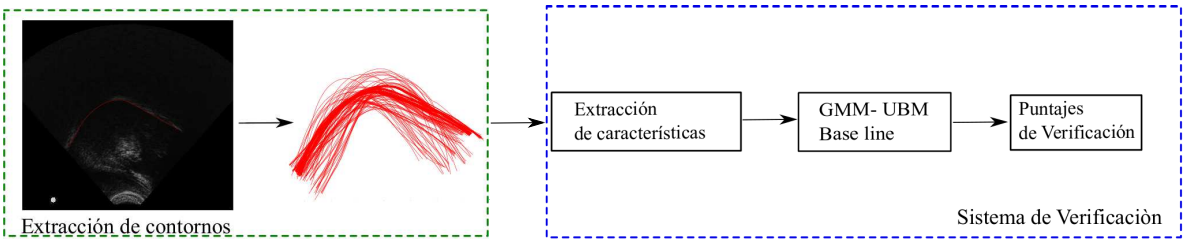


Figura 13. Esquema de SVH con información articulatoria.

SV únicamente con parámetros articulatorios. El esquema mostrado en la Figura 13, muestra un resumen de los pasos, realizados para llevar acabo el experimento únicamente con parámetros articulatorios y en la tabla 5 muestra los resultados de verificación utilizando las características articulatorias, además podemos compararlo con el desempeño obtenido únicamente con parámetros MFCC. Se observa un desempeño de 2.83 % para el caso de características articulatorias. En (Zhang et al., 2017b), se alcanzan desempeños del $4.7 - 6.8\,\%$ utilizando la información de la punta y la mitad de la lengua recolectada por EMA. En la tabla 5 casilla 3), se tiene como referencia el resultado del SVH utilizando como entrada las características acústicas MFCC + Δ + $\Delta\Delta$. Las mencionadas anteriormente, obtuvieron un EER del 3.79 %, lo que supera el desempeño alcanzado por las características articulatorias. Sin embargo y al igual que los experimentos de identificación (sección 4.3), la dimensión del vector de características acústicas es de 39 a diferencia del vector de características articulatorias que es de 9 y 10 para 1) y 2) de la tabla 5, respectivamente. Así que, esta importante diferencia en la dimensión puede ser un motivo por el que los MFCC sigan obteniendo los mejores resultados. Además, es notable que el desempeño de la información articulatoria es relativamente cercano a los MFCC, teniendo en cuanta una amplia diferencia en la dimensión y que solo se tiene acceso a una parte del tracto vocal. Cabe resaltar que en las columnas LI y LS de las tablas 5 7 6 corresponden al límite inferior y superior del intervalo de confianza calculado con un nivel de confianza del 95 %.

SV con fusión de características. A partir de los resultados anteriores, se realiza la fusión de características como se describe en 4.1.3, donde, se menciona que la ventana de extracción de los MFCC es de 78 ms, lo cuál es diferente del sistema base que son de 25ms. En la tabla 6, se puede observar que el desempeño logrado por el SVH del 1.29 %. Esto representa una mejora respecto al enfoque poly (1) e incluso de los MFCC, que logran un desempeño

Características	**Método**	**EER**	**LI**	**LS**
1) $p_0, p_1, ..., p_6, x_i, x_f$	GMM-UBM	**2.83 %**	2.42 %	3.24 %
2) $x_1, x_2, ..., x_5, y_1, y_2, ..., y_5$	GMM-UBM	4.94 %	4.37 %	5.51 %
3) $MFCC$	GMM-UBM	**3.79 %**	3.35 %	4.23 %

Tabla 5
Desempeño del sistema GMM-UBM, para las características propuestas.

del 3.79 %. Esto indica que entrenar con información acústica y articulatoria, es decir con la fusión de características, es un resultado prometedor. Y estos resultados son congruentes con lo mostrado en el trabajo (Li et al., 2016a), en el cual también presentan una mejora en desempeño, cuando se entrena con la fusión de las características.

Características	**Dimensión**	**EER**	**LI**	**LS**
1) $p_0, p_1, ..., p_6, x_i, x_f$	9	2.83 %	2.42 %	3.24 %
2) $MFCC$	39	3.79 %	3.35 %	4.23 %
3) $p_0, p_1, ..., p_6, x_i, x_f$ + MFCC	48	**1.29** %	0.98 %	1.60 %

Tabla 6
SVH con fusión de características, ademas se pueden observar los resultados con los MFCC y el resultado con los parámetros articulatorios poly (1).

Experimentos con i-vector. En este apartado se encuentran los experimentos con el enfoque que se encuentra en el estado del arte *i-vector*. El desempeño logrado se muestra en la tabla 7. En 1) se realiza el experimento sólo con características articulatorias, particularmente el enfoque *poly*, obteniendo un EER de 4.82 %. Al realizar la fusión de características, se obtiene un mejor desempeño alrededor de 2,26 puntos porcentuales. Lo que sigue corroborando la hipótesis, que la fusión mejora el desempeño. Sin embargo, el mejor desempeño es logrado por la configuración anterior, es decir utilizando GMM-UBM. Y esto puede puede ser debido a que el enfoque i-vector, para lograr los más altos desempeños requiere de grandes cantidades de datos de entrenamiento (Li et al., 2016a).

Características	Método	EER	LI	LS
1) $p_0, p_1, ..., p_6, x_i, x_f$	i-vector + PLDA	4.82 %	4.27 %	5.37 %
2) MFCC	i-vector + PLDA	3.22 %	2.79 %	3.65 %
3) $p_0, p_1, ..., p_6, x_i, x_f$ + MFCC	i-vector + PLDA	2.56 %	2.15 %	2.97 %
4) $p_0, p_1, ..., p_6, x_i, x_f$ + MFCC	GMM-UBM	**1.29** %	0.88 %	1.70 %

Tabla 7
Resultados: experimentos utilizando i-vector.

Finalmente en la figura 14, se muestra una curva DET [4] que reúne todos los resultados de los experimentos de verificación. El particularmente el EER, es el punto en que las falsas aceptaciones y los falsos rechazos es igual. Así, la curva qué mas abajo se encuentre, es la que mejor desempeño obtuvo. En este caso, la fusión de características articulatorias *poly* con los MFCC y modelados con mezclas Gaussianas presentan el mejor desempeño.

5. Inversión Acústico-Articulatorio usando DNN's

Los sistemas de reconocimiento de voz son muy prácticos, ya que, los dispositivos de adquisición como los micrófonos, los cuales permiten capturar la señal de habla, son de bajo costo y fáciles de implementar; además las personas pueden cooperar de una manera muy natural para producir los datos requeridos. Esto supone que la propuesta de este trabajo, que consta de utilizar información articulatoria obtenida de imágenes ultrasonido, resulte ser poco practico para los sistemas de reconocimiento de voz, al menos, desde el punto de vista de una aplicación móvil. Este problema puede ser resuelto al estimar la información articulatoria a partir de la señal de voz.

Los movimientos articulatorios y la señal acústica del habla están directamente relacionados en el proceso de producción del habla. La inversión acústica a articulatoria (AAI) (Richmond, 2002) se asocia a la estimación de

[4] Las curvas DET permiten visualizar el desempeño del sistema en base a la relación que se presenta entre los dos tipos de error que se pueden generar: Falsas aceptaciones y falsos rechazos. Los ejes de las curvas DET son representados usando frecuentemente una escala logarítmica (Martin et al., 1997)

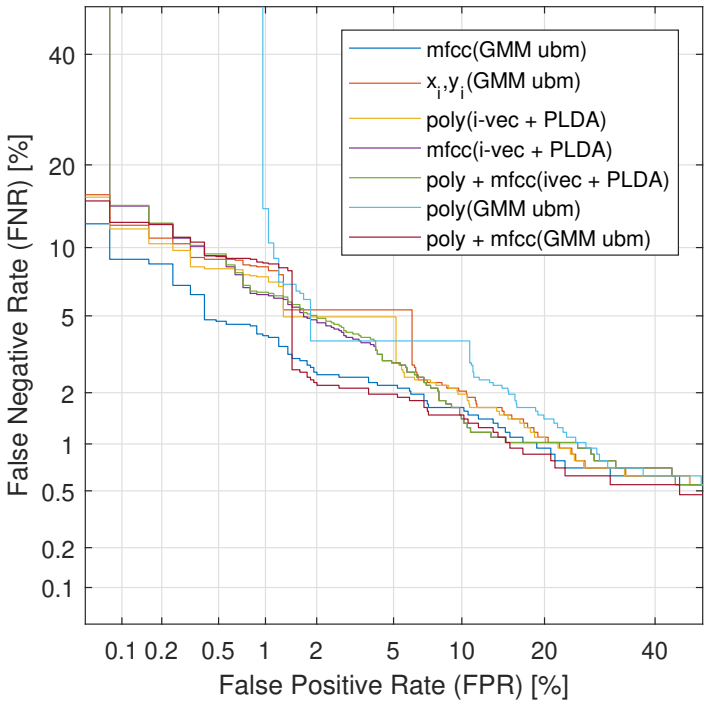


Figura 14. Curva DET, donde se muestran los resultados experimentales de verificación.

los movimientos articulatorios a partir de las características acústicas del habla. Recientemente, ha habido un interés significativo en la AAI, porque el aprendizaje de la relación entre la información articulatoria y la acústica podría mejorar el rendimiento en varias tareas como el reconocimiento de voz (Frankel et al., 2000; King et al., 2007), síntesis de voz (Ling et al., 2009), verificación de hablantes (Li et al., 2016b), cabezas parlantes (Wang et al., 2010), entre otros. La inversión acústica-articulatorio se puede dividir en dos enfoques: independiente del sujeto (SI-AAI) ó dependiente del hablante (SD-AAI) (Illa and Ghosh, 2018b). El primero está más allá del alcance de este trabajo.

Se han propuesto varios métodos para abordar el problema SD-AAI, incluidos los *codebooks* (Ouni and Laprie, 2005), los Modelos de Mezcla Gaussianos (GMM) (Toda et al., 2008), los Modelos Ocultos de Markov (HMM) (Zhang and Renals, 2008). Durante los últimos años, las Deep Neural Networks (DNN) han producido los resultados de referencia en el estado del arte en el reconocimiento de voz (Mohamed et al., 2012), entre otros campos. Por lo tanto, algunos colegas comenzaron a investigar y utilizar arquitecturas profundas para realizar inversión

acústico-articulatoria (Uria et al., 2012; Wu et al., 2015) incluso teniendo tal éxito como obtener la precisión de vanguardia (Liu et al., 2015) en la tarea de AAI. Definitivamente, todos estos enfoques necesitan datos paralelos acústico-articulatorios para entrenar el modelo AAI. Por lo tanto, estos estudios se basan en datos de Articulografía Electromagnética (EMA) (Toda et al., 2008; Zhang and Renals, 2008; Richmond, 2006; Uria et al., 2012; Wu et al., 2015; Liu et al., 2015) sin embargo, las imágenes de ultrasonido (UTI) son una técnica de mayor costo-beneficio si tomamos en cuenta el costo del equipo, portabilidad, seguridad y estructuras visualizadas.

En este capítulo, se presenta la estimación de la información articulatoria a partir de la acústica. El cuál ha sido el resultado de una pasantía de investigación del autor del presente trabajo en colaboración con la *Budapest University of Technology and Economics*. Durante el período de estancia se implementaron dos redes neuronales profundas (DNN) diferentes para modelar la relación acústico-articulatoria. Este campo se conoce como inversión acústica-articulatoria (AAI) (Richmond, 2002). Hay una gran cantidad de trabajos relacionados con esta tarea, sin embargo, encontramos que esos esfuerzos se enfocan en la Articulografía Electromagnética (EMA). Por otra parte, la imagen por ultrasonido (UTI) es una técnica de mayor relación costo-beneficio si tenemos en cuenta el costo del equipo, la portabilidad, la seguridad y las estructuras visualizadas.

Nuestro objetivo es entrenar una DNN para obtener imágenes de UTI a partir de señales de audio. También probamos dos enfoques para representar la información articulatoria: El espacio EigenTongue y la imagen de ultrasonido en bruto. Como medida de calidad objetiva para las imágenes UTI reconstruidas, utilizamos MSE, SSIM y CW-SSIM. Nuestros resultados experimentales muestran por un lado que CW-SSIM podría ser el más coherente en el contexto de UTI. Y también, probamos con tres configuraciones de sistema diferentes, que son: DNN simple conformado por 2 capas ocultas con 64x64 píxeles de un archivo UTI como entrada; El mismo DNN simple pero con imágenes de ultrasonido proyectadas en el espacio EigenTongue; Y un DNN más complejo compuesto de 5 capas ocultas con archivos UTI proyectados en el espacio EigenTongue.

De acuerdo con nuestra revisión de la literatura, solo registramos este trabajo (Wei et al., 2016) donde uti-

lizaron Deep Auto-encoders para obtener las representaciones de datos, luego con una arquitectura DNN obtuvieron un mejor rendimiento que el método GMM. El mapeo entre Imágenes de ultrasonido y características acústicas es realizado, pero exclusivamente en vocales chinas. En consecuencia, motivado por el éxito conocido en la precisión de las arquitecturas profundas y el deseo de ir más allá de la AAI que las vocales, presentado en (Wei et al., 2016). El documento actual se centra en mostrar que con redes neuronales profundas totalmente conectadas y simples se puede predecir información articulatoria a partir de características espectrales. Finalmente, comparamos nuestros resultados con tres medidas de calidad de imagen (MSE, SSIM, CW-SSIM). Dónde, se muestra que todas esas configuraciones presentan un rendimiento muy similar.

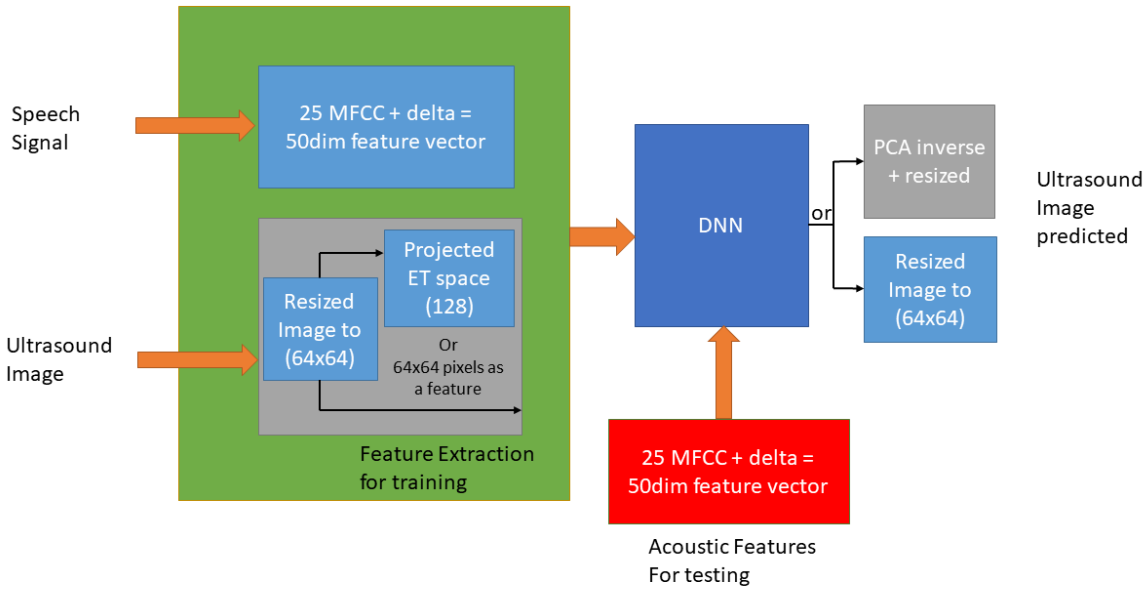


Figura 15. Esquema de entrada para la DNN, donde se muestran las dos características articulatorias utilizadas.

5.1. Datos

La configuración para la adquisición de datos en este trabajo es la misma que en (Csapó et al., 2017). Se realizó sobre un sujeto femenino húngaro (42 años) con habilidades normales para hablar mientras se leían oraciones en voz alta en la base de datos de PPSD (Olaszy, 2013) en idioma húngaro (en total 209 oraciones, de las cuales 200 se usaron para entrenamiento y validación y nueve para la prueba). El movimiento de la lengua también se registró en la orientación midsaggital utilizando un sistema de ultrasonido "Micro"(Articulate Instruments Ltd.) con un transductor de ultrasonido convexo de 20 mm de radio de 20 mm / elemento de 2-4 MHz a 64 cps. Durante las grabaciones, el transductor se fijó con un auricular de estabilización de ultrasonido (Articulate Instruments Ltd.). La grabación de los labios de vídeo también se realizó (Csapó et al., 2017), pero no se usó en el estudio actual. La señal de voz se grabó con un micrófono de condensador omnidireccional Audio-Technica - ATR 3350 que se localizo aproximadamente a 20 cm de los labios. Tanto la señal del micrófono como las señales de sincronización de ultrasonido se digitalizaron utilizando una tarjeta de sonido externa M-Audio - MTRACK PLUS a una frecuencia de muestreo de 22050 Hz. El ultrasonido y las señales de audio se sincronizaron utilizando la salida de sincronización de cuadros del equipo con el software Articulate Assistant Advanced (Articulate Instruments Ltd.). En los experimentos a continuación y similares a (Tóth et al., 2018), se usaron los datos fuente del ultrasonido.

Representación de la información articulatoria. Las imágenes de ultrasonido originales sin procesar, en escala de grises son de 64x842 píxeles se re-dimensionaron a 64x64 usando una interpolación bicúbica. Esta reducción no afectó significativamente el contenido visual de las imágenes, sin embargo permite un conjunto de datos más compacto, lo que disminuye el tiempo de entrenamiento, la síntesis y el riesgo de sobre-entrenamiento del DNN.

Para obtener las características de las imágenes de ultrasonido, probamos con dos enfoques, como se puede ver en la Figura 15. Primero, todos los 64x64 píxeles de la UTI se usan como un vector de características. Y segundo, realizamos el método de extracción de características 'Eigentongue' (ET) (Hueber et al., 2007), que consiste en encontrar un conjunto de imágenes ortogonales (*EigenTongues*) que constituyen un sub-espacio para la representación de

todas las configuraciones posibles de la lengua . En general, las eigentongues se extraen aplicando PCA en los datos de entrenamiento y los eigenvectors obtenidos se definen como eigentongue, en la parte inferior de la figura 17 puede ver dos ET principales. Se supone que los componentes extraídos codifican la cantidad máxima de información relevante presente en las imágenes, como la posición de la lengua, el hueso hioides y los músculos también están presentes.

Por cada 12 ms de tramas de audio (dictadas por una velocidad de ultrasonido de 82 fps), son extraídos 25 coeficientes en escala MEL (MFCC). También se incluye la primer derivada Δ, conocido como el componente de velocidad. En un vector de características acústicas totalmente de dimensión 50 se calcula para representar la información de acústica (McFee et al., 2015).

5.2. Deep Neural Networks (DNNs)

Las DNN's como su nombre lo indica, son redes neuronales artificiales con múltiples capas entre la capa de entrada y la de salida, llamadas capas ocultas. La función de estas capas ocultas es la de aprender automáticamente características que puedan representar los datos de entrada. Además, al aumentar el número de capas ocultas se pueden obtener relaciones más complejas, que permitan hacer un correcto mapeo entre la entrada y la salida. Por ejemplo, una DNN puede reconocer un automóvil de una fotografía, mediante el aprendizaje de una representación más simple de dicho objeto, con características como las ruedas, las puertas y ventanas que lo conforman. Sin embargo, la mayoría de veces, la salida de las capas ocultas puede ser un concepto más abstracto que no tiene un sentido para nosotros.

Por otra parte, existen distintas arquitecturas que hacen parte del *Deep Learning*. Por ejemplo RNN's, LSTM, CNN's, Autoencoders, entre otras. La arquitectura más básica y común es la *feed forward neural network*, y es llamada así porque, el flujo de la información es desde la capa de entrada, pasando por las capas ocultas hasta la capa de salida, en otras palabras, hacía adelante y sin ninguna retro-alimentación. Algunos retos importantes en el campo del Deep learning, es el sobre-entrenamiento (*overfitting*) y el costo computacional. A partir de allí, han surgido métodos para tratar con estas preocupaciones como métodos de regularización y optimización de hiper-párametros, donde, se ajusta el número de neuronas usadas en cada capa, el número de capas, entre otros parámetros. Y el costo computacional es aliviado al usar tarjetas gráficas de procesamiento ó GPUs y seleccionando un *batch* apropiado para el procesamiento,

entre otros métodos. En síntesis, esta sección muestra un breve resumen de las DNNs, ya que entrar en los detalles de esta temática no hace parte del propósito de este documento, para mayor información acerca de Deep Learning se pueden referir a este libro (Goodfellow et al., 2016).

5.3. Arquitectura DNN

El objetivo principal de este trabajo es obtener imágenes de ultrasonido a partir de los MFCC. Para este propósito, entrenamos dos arquitecturas DNN: una con 5 capas ocultas, con cada capa oculta formada por 5000 neuronas rectificadas; y la otra con 2 capas ocultas, cada capa oculta consiste en 1000 neuronas rectificadas. La dimensión de entrada (MFCC) es 50 y la salida para los 2 enfoques: 128 para los ETs y 64x64 píxeles. La Figura 15 muestra esto. Además, para obtener mejores resultados, se realizó una optimización de hiper-parámetros, los detalles se encuentran en la sección Experimental. Finalmente, las arquitecturas de DNN son entrenadas con 200 archivos que contienen 56773 imágenes de ultrasonido para entrenamiento y 6309 para validación (10 % del número total de imágenes).

5.4. Resultados y Discusión del procedimiento de inversión

Según nuestra encuesta bibliográfica, creemos que nuestro trabajo no puede compararse con ninguno anterior. En efecto el trabajo más similar es (Wei et al., 2016) sin embargo, solo se realiza AAI sobre vocales. Y esperábamos ir más allá de ese enfoque, realizando una AAI en oraciones completas. Además, inspirado en (Xu et al., 2016b), donde se muestra que el índice CW-SSIM parece ser la mejor medida de calidad para las imágenes de ultrasonido. Así que decidimos incluir dos índices de calidad que son el SSIM y el CW-SSIM (Sampat et al., 2009).

Para medir la calidad de la imagen de ultrasonido reconstruida se utilizaron tres métricas. Cada medida de calidad se calcula sobre cada fotograma, donde la imagen verdadera y es la imagen original re-dimensionada y $\hat{y}$ es la imagen estimada de la arquitectura DNN.

- Mean Square Error (MSE)

$$MSE(y,\hat{y}) = \frac{1}{n_{samples}} \sum_{i=0}^{n_{samples}-1} (y_i - \hat{y}_i)^2 \tag{22}$$

where $n_{samples}$ son los 64x64 pixeles.

- Índice de similitud estructural (SSIM) (Wang et al., 2004), este índice mide tres tipos de impacto visual de cambios en la iluminación l, contraste c y estructura s entre dos imágenes.

$$SSIM(y,\hat{y}) = [l(y,\hat{y})]^{\alpha}[c(y,\hat{y})]^{\beta}[s(y,\hat{y})]^{\gamma} \tag{23}$$

En nuestro experimento, el índice SSIM se calcula mediante la función de ponderación gaussiana simétrica circular 11x11, con una desviación estándar de 1.5 píxeles. Las estadísticas para los cambios de impacto visual se calculan como sigue en (Xu et al., 2016b).

- La similitud estructural de wavelet compleja (CW-SSIM) (Sampat et al., 2009) es una extensión del método SSIM al dominio de wavelet complejo, que es una medida de similitud de imagen novedosa, robusta a pequeñas distorsiones.

$$CWSSIM(y,\hat{y}) = \frac{2|\sum_{l=1}^{L} w_{y,l} w_{\hat{y},l}^{*}| + K}{\sum_{l=1}^{L} |w_{y,l}|^2 + \sum_{l=1}^{L} |w_{\hat{y},l}|^2 + K} \tag{24}$$

donde w representa los complejos coeficientes de wavelet de las dos imágenes. También de * indica que el complejo conjugado de w y K es una pequeña constante estabilizadora positiva (Xu et al., 2016b). Tanto en SSIM como en CWSSIM, el índice 1 significa que el contenido de dos imágenes comparadas es el mismo, y 0 es el valor mínimo. En la figura 16 podemos ver las tres medidas de calidad aplicadas a una sola frase de nuestro experimento. Allí, la línea roja muestra lo que esperamos que sea el mejor resultado para ser coherente, porque comparamos la imagen original con la versión de PCA o ET recuperada del original. Que se ve igual que el original (el mismo contorno de la lengua y las estructuras visualizadas) pero sin el ruido que contiene la imagen original. Entonces, la figura 16 sugiere que el índice CW-SSIM podría ser la mejor medida en nuestros experimentos porque es más coherente que los otros índices.

Comparamos 3 sistemas de inversión acústica a articulatoria en 1 hablante húngaro. Los resultados se muestran en la Tabla 8. Podemos ver que no hay mucha diferencia en las medidas de calidad entre los sistemas propuestos. Sin embargo, nuestros resultados experimentales sugieren que con un DNN simple de 2 capas y 1000 unidades, podemos obtener mejores resultados en comparación con una arquitectura mas compleja de 5-DNN y 5000 unidades.

Capas Ocultas UTI características	MSE		SSIM		CW-SSIM	
	Media	Desv.Std.	Media	Desv.Std.	Media	Desv.Std.
5 x 5000 unidades 128 ETs	85,54	11,32	0,69	0,09	0,79	0,06
2 x 1000 unidades 64*x*64 pixeles	86,02	11,04	0,71	0,08	0,81	0,06
2 x 1000 unidades 128 ETs	85,03	11,43	0,72	0,09	0,81	0,06

Tabla 8
Evaluación de las tres diferentes configuraciones de DNN y características de imágenes de ultrasonido.

Además, no es necesario proyectar las imágenes de ultrasonido en un espacio EigenTongue, porque el rendimiento es muy similar. Además, se muestra un cuadro reconstruido a partir de la información acústica en la Figura 17. Finalmente se realizó una optimización hiper-parámetros. Se probaron varias configuraciones, incluido el número de unidades en 5 capas ocultas. También se probaron los optimizadores 'rmsprop', 'adam', 'sgd' y el tamaño del lote. Sin embargo, los resultados son muy similares a los mostrados en este trabajo.

6. Trabajo futuro

La base de datos es un resultado de este trabajo y en colaboración con dos estudiantes de pregrado. Además se va a ofrecer para estudios académicos y de investigación, para esto se va a habilitar un repositorio en línea que permita descargar los datos recolectados. Como trabajo futuro planteamos ampliar la base de datos que fue recolectada. Además de la adquisición de un dispositivo que facilite la sincronización de la toma de muestras de audio e imágenes de ultrasonido, así mismo, se recomienda utilizar una sonda de ultrasonido con mejor resolución temporal. Lo que además de recolectar mayor información de la dinámica de la lengua también permite aumentar las ventanas temporales en la extracción de los MFCC.

Por otra parte, se ha planteado implementar una inversión acústico-articulatoria, es decir que a partir de las características acústicas, estimar la información articulatoria. De manera, que el resultado de esta posibilite una implementación práctica. Pues la adquisición de ambas señales en tiempo real no es muy práctico (Li et al., 2016a). En este punto, ya se ha adelantado gran parte en el marco de la realización de una pasantía en el grupo SmartLab, BME en la ciudad de Budapest, Hungría. Donde hemos implementado tres tipos diferentes de sistemas AAI. Para todos los casos

utilizamos arquitecturas DNN. Los resultados sobre las medidas de calidad utilizadas indican que con un DNN simple que consiste en 2 capas ocultas, se realiza la mejor precisión en la tarea de inversión articulatoria. Sin embargo, no hay mucha diferencia en comparación con las otras configuraciones del sistema, que incluyen probar con: EigenTongue y la UTI como representación de las imágenes articulatorias. Además, realizamos una optimización de hiperparámetro para obtener mejores resultados, pero nuevamente no hemos encontrado una mejora significativa. En conclusión, creemos que se puede obtener una mejora significativa, incluida la información temporal. Para este propósito, sugerimos utilizar arquitecturas profundas como las capas LSTM o BLSTM. Por otra parte, el índice CW-SSIM parece ser más coherente que las otras medidas de calidad. Como trabajo futuro, proponemos comparar los resultados presentados en este trabajo con un enfoque de GMM e incluir información temporal a las características.

7. Conclusiones

El presente trabajo ha evaluado y analizado el uso el uso de la información articulatoria del movimiento de la lengua en los sistemas de verificación del hablante. De acuerdo a los resultados obtenidos, es posible concluir que las características articulatorias sirven para resolver la tarea de verificación. También, que la fusión de las características acústicas y articulatorias mostró una mejoría del desempeño del SVH desarrollado. Por otra parte, se recolecto una base de datos en paralelo de imágenes de ultrasonido y voz que sirvió para obtener los resultados experimentales que sugieren y soportan la hipótesis que la información articulatoria puede mejorar los sistemas de verificación de hablantes.

Fueron propuestos dos enfoques para la extracción de las características articulatorias a partir de la secuencia de imágenes 2D de ultrasonido correspondiente al contorno superior de lengua, en un plano midsadgital. Los resultados en materia de verificación de hablantes, indican que los coeficientes calculados por medio de un ajuste polinomial de curvas de orden 6 y con un desempeño en EER de 2.83 %, esta por encima que utilizar 5 puntos del contorno de la lengua. Además, las características polinomiales muestran ser prometedores para ser utilizados en materia de VH, ya que el desempeño logrado es sólo un poco superior al de los MFCC con un EER de 3.79 %.

Así mismo, fue implementada una fusión de características acústicas y articulatorias, creando vectores de características híbridas que sirven de entrada para un SVH. En este punto se utilizaron dos métodos: GMM-UBM e i-vector+PLDA. El desempeño logrado por este enfoque de fusión de características y el método GMM-UBM muestran los mejores resultados en materia de desempeño, con un EER de 1.29 %, incluso sobrepasa al EER alcanzado por los MFCC. Creemos, que el desempeño de i-vector se encuentra limitado por la cantidad de datos de entrenamiento, y que a pesar de ser el enfoque que se encuentra en el estado del arte, no alcanza el mejor rendimiento. Sin embargo, al comparar los resultados de los experimentos únicamente con i-vector, también muestran que la fusión de las características mejora el desempeño del SVH.

Los resultados en las pruebas de identificación sugieren que la representación polinomial del contorno de la lengua es mas discriminante, pues este enfoque logra una precisión del 94 %, frente al 92 % de las 5 puntos; esto en el mejor de los casos que fue utilizando modelos de Mezclas Gaussianas. Por otra parte, al realizar la fusión de las características acústicas y articulatorias el sistema sufrió una degradación en el desempeño, lo que contrapone la posición tomada en las pruebas de verificación. De esta manera, encontramos que la información articulatoria por sí sola, sirve para discriminar a las personas. Y además que la fusión con los MFCC, mejoraron en nuestro caso el desempeño sólo de los sistemas de verificación de hablantes. Pero difícilmente se puede generalizar dicho resultado. Finalmente, es una novedad utilizar la información articulatoria recolectada con un sistema de ultrasonido con propósitos de VH, pues en la revisión de la literatura no se encontró ningún trabajo que lo hiciera. Y así mismo, gran parte de los resultados obtenidos en el presente trabajo fueron presentados en el marco de una conferencia internacional llamada *Beszédkutatás -2018*, realizada en Hungría, Octubre 18-19th del 2018.

Referencias Bibliográficas

(citado en 2019). Building a raspberry pi based ultrasound imaging development platform [online]. Disponible en: https://kelu124.gitbooks.io/echomods/RPI.html.

ABBS, J. and COLE, K. (1982). Consideration of bulbar and suprabulbar afferent influences upon speech motor coordination and programming. In *Speech Motor Control*, pages 159–186. Pergamon.

Adami, A. G., Mihaescu, R., Reynolds, D. A., and Godfrey, J. J. (2003). Modeling prosodic dynamics for speaker recognition. In *Acoustics, Speech, and Signal Processing, 2003. Proceedings.(ICASSP'03). 2003 IEEE International Conference on*, volume 4, pages IV–788. IEEE.

AGNITIO (2015). Batvox data sheet. Available from Internet: http://www.agnitio-corp.com/sites/default/files/BATVOX_Datasheet.pdf.

Barberena, L. d. S., Simoni, S. N. d., Souza, R. C. S. d., Moraes, D. A. d. O., Berti, L. C., and Keske-Soares, M. (2017). Ultrasound analysis of tongue contour for the sound [j] in adults and children. In *CoDAS*, volume 29. SciELO Brasil.

Barras, C. and Gauvain, J.-L. (2003). Feature and score normalization for speaker verification of cellular data. In *2003 IEEE International Conference on Acoustics, Speech, and Signal Processing, 2003. Proceedings.(ICASSP'03).*, volume 2, pages II–49. IEEE.

BenZeghiba, M. F. and Bourlard, H. (2003). On the combination of speech and speaker recognition. In *Eighth European Conference on Speech Communication and Technology*.

Bimbot, F., Bonastre, J.-F., Fredouille, C., Gravier, G., Magrin-Chagnolleau, I., Meignier, S., Merlin, T., Ortega-García, J., Petrovska-Delacrétaz, D., and Reynolds, D. A. (2004). A tutorial on text-independent speaker verification. *EURASIP Journal on Advances in Signal Processing*, 2004(4):101962.

Biosecure (2007 [cited 2015]). Reference system based on speech modality alize/lia_ral [online]. Available from Internet: http://www-clips.imag.fr/geod/User/laurent.besacier/NEW-TPs/TP-Biometrie/tools/CommentsLBInstall/doc.pdf.

Bishop, C. M. (2006). *Pattern Recognition and Machine Learning*. Springer.

Brookes, M. et al. (1997). Voicebox: Speech processing toolbox for matlab. *Software, available [Mar. 2011] from www. ee. ic. ac. uk/hp/staff/dmb/voicebox/voicebox. html*, 47.

Burton, D. (1987). Text-dependent speaker verification using vector quantization source coding. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, 35(2):133–143.

Campbell, W. M., Campbell, J. P., Gleason, T. P., Reynolds, D. A., and Shen, W. (2007). Speaker verification using support vector machines and high-level features. *IEEE Transactions on Audio, Speech, and Language Processing*, 15(7):2085–2094.

Campbell, W. M., Campbell, J. P., Reynolds, D. A., Jones, D. A., and Leek, T. R. (2004). Phonetic speaker recognition with support vector machines. In *Advances in neural information processing systems*, pages 1377–1384.

Campbell, W. M., Campbell, J. P., Reynolds, D. A., Singer, E., and Torres-Carrasquillo, P. A. (2006a). Support vector machines for speaker and language recognition. *Computer Speech & Language*, 20(2-3):210–229.

Campbell, W. M., Sturim, D. E., and Reynolds, D. A. (2006b). Support vector machines using gmm supervectors for speaker verification. *IEEE Signal Processing Letters*, 13(5):308–311.

Csapó, T. G., Grósz, T., Gosztolya, G., Tóth, L., and Markó, A. (2017). Dnn-based ultrasound-to-speech conversion for a silent speech interface. In *Proc. Interspeech 2017*, pages 3672–3676.

Csapó, T. G., Grósz, T., Gosztolya, G., Tóth, L., and Markó, A. (2017). Dnn-based ultrasound-to-speech conversion for a silent speech interface. *Proc. Interspeech, Stockholm, Sweden*, pages 3672–3676.

Csapó, T. G. and Lulich, S. M. (2015). Error analysis of extracted tongue contours from 2d ultrasound images. In *Sixteenth Annual Conference of the International Speech Communication Association*.

Cumani, S., Brümmer, N., Burget, L., and Laface, P. (2011). Fast discriminative speaker verification in the i-vector space. In *Acoustics, Speech and Signal Processing (ICASSP), 2011 IEEE International Conference on*, pages 4852–4855. IEEE.

de Vasconcelos, M. J. M., Ventura, S. M. R., Tavares, J. M. R., and Freitas, D. R. S. (2009). Analysis of tongue shape and motion in speech production using statistical modeling. In *SEECCM 2009-2nd South-East European Conference on Computational Mechanics*.

Dehak, N., Kenny, P. J., Dehak, R., Dumouchel, P., and Ouellet, P. (2011). Front-end factor analysis for speaker verification. *IEEE Transactions on Audio, Speech, and Language Processing*, 19(4):788–798.

Doddington, G. (2001). Speaker recognition based on idiolectal differences between speakers. In *Seventh European Conference on Speech Communication and Technology*.

Eringis, D. and Tamulevičius, G. (2014). Improving speech recognition rate through analysis parameters. *Electrical, Control and Communication Engineering*, 5(1):61–66.

Farrell, K. R., Mammone, R. J., and Assaleh, K. T. (1994). Speaker recognition using neural networks and conventional classifiers. *IEEE Transactions on speech and audio processing*, 2(1):194–205.

Ferrer, L., Shriberg, E., Kajarekar, S., and Sonmez, K. (2007). Parameterization of prosodic feature distributions for svm modeling in speaker recognition. In *Acoustics, Speech and Signal Processing, 2007. ICASSP 2007. IEEE International Conference on*, volume 4, pages IV–233. IEEE.

Fontecave, J. and Berthommier, F. (2009). Quasi-automatic extraction of tongue movement from a large existing speech cineradiographic database. *Evaluation*, 2:8–11.

Fontecave Jallon, J. and Berthommier, F. (2009). A semi-automatic method for extracting vocal tract movements from X-ray films. *Speech Communication*, 51(2):97–115.

Frankel, J., Richmond, K., King, S., and Taylor, P. (2000). An automatic speech recognition system using neural networks and linear dynamic models to recover and model articulatory traces.

Gannert, T. (2007). *A Speaker Verification System Under The Scope: Alize*. PhD thesis, Masters thesis, TMH.

Garcia-Romero, D. and Espy-Wilson, C. Y. (2011). Analysis of i-vector length normalization in speaker recognition systems. In *Twelfth Annual Conference of the International Speech Communication Association*.

Ghosh, P. K. and Narayanan, S. (2010). A generalized smoothness criterion for acoustic-to-articulatory inversion. *The Journal of the Acoustical Society of America*, 128:2162–2172.

Ghosh, P. K. and Narayanan, S. (2011). Automatic speech recognition using articulatory features from subject-independent acoustic-to-articulatory inversion. *The Journal of the Acoustical Society of America*, 130(4):EL251–EL257.

Gonzalez-Rodriguez, J., Drygajlo, A., Ramos-Castro, D., Garcia-Gomar, M., and Ortega-Garcia, J. (2006). Robust estimation, interpretation and assessment of likelihood ratios in forensic speaker recognition. *Computer Speech & Language*, 20(2-3):331–355.

Goodfellow, I., Bengio, Y., and Courville, A. (2016). *Deep Learning*. MIT Press. `http://www.deeplearningbook.org`.

Greenberg, C. S., Bansé, D., Doddington, G. R., Garcia-Romero, D., Godfrey, J. J., Kinnunen, T., Martin, A. F., McCree, A., Przybocki, M., and Reynolds, D. A. (2014). The nist 2014 speaker recognition i-vector machine learning challenge. In *Odyssey: The Speaker and Language Recognition Workshop*, pages 224–230.

Hasan, T. and Hansen, J. (2011). A study on universal background model training in speaker verification. *IEEE Transaction Audio, Speech Language Processing*, 19(7):1890–1899.

Hermansky, H. (1990). Perceptual linear predictive (plp) analysis of speech. *the Journal of the Acoustical Society of America*, 87(4):1738–1752.

Hermansky, H. and Morgan, N. (1994). Rasta processing of speech. *IEEE transactions on speech and audio processing*, 2(4):578–589.

Higgins, A. (1990). Yoho speaker verification. In *Speech Research Symposium, Baltimore, MD*.

Huang, X., Acero, A., and Hon, H. (2001). *Spoken Language Processing: A guide to theory, algorithm, and system development*. Prentice Hall.

Hueber, T., Aversano, G., Cholle, G., Denby, B., Dreyfus, G., Oussar, Y., Roussel, P., and Stone, M. (2007). Eigentongue feature extraction for an ultrasound-based silent speech interface. In *2007 IEEE International Conference on Acoustics, Speech and Signal Processing-ICASSP'07*, volume 1, pages I–1245. IEEE.

Hueber, T., Chollet, G., Denby, B., Dreyfus, G., and Stone, M. (2008). An ultrasound-based silent speech interface. *The Journal of the Acoustical Society of America*, 123:3938.

Ichikawa, O., Fukuda, T., and Nishimura, M. (2010). Dynamic features in the linear-logarithmic hybrid domain for automatic speech recognition in a reverberant environment. *IEEE Journal of Selected Topics in Signal Processing*, 4(5):816–823.

Illa, A. and Ghosh, P. K. (2018a). Inferring speaker identity from articulatory motion during speech. Workshop on Machine Learning in Speech and Language Processing, 2018.

Illa, A. and Ghosh, P. K. (2018b). Low resource acoustic-to-articulatory inversion using bi-directional long short term memory. *Proc. Interspeech 2018*, pages 3122–3126.

Jaumard-Hakoun, A., Xu, K., Roussel-Ragot, P., Dreyfus, G., and Denby, B. (2016). Tongue contour extraction from ultrasound images based on deep neural network. *arXiv preprint arXiv:1605.05912*.

Kass, M., Witkin, A., and Terzopoulos, D. (1988). Snakes: Active contour models. *International journal of computer vision*, 1(4):321–331.

Kelly, A. C. and Gobl, C. (2011). The effects of windowing on the calculation of mfccs for different types of speech sounds. In *International Conference on Nonlinear Speech Processing*, pages 111–118. Springer.

Kim, J., Ghosh, P., Lee, S., and Narayanan, S. S. (2012). A study of emotional information present in articulatory movements estimated using acoustic-to-articulatory inversion. In *Signal & Information Processing Association Annual Summit and Conference (APSIPA ASC), 2012 Asia-Pacific*, pages 1–4. IEEE.

King, S., Frankel, J., Livescu, K., McDermott, E., Richmond, K., and Wester, M. (2007). Speech production knowledge in automatic speech recognition. *The Journal of the Acoustical Society of America*, 121(2):723–742.

Kinnunen, T. and Li, H. (2010). An overview of text-independent speaker recognition: From features to supervectors. *Speech Communication*, 52(1):12 – 40.

Koolagudi, S. G., Rao, K. S., Reddy, R., Kumar, V. A., and Chakrabarti, S. (2012). Robust speaker recognition in noisy environments: Using dynamics of speaker-specific prosody. In *Forensic Speaker Recognition*, pages 183–204. Springer.

Leung, K.-Y., Mak, M.-W., and Kung, S.-Y. (2004). Applying articulatory features to telephone-based speaker verification. In *Acoustics, Speech, and Signal Processing, 2004. Proceedings.(ICASSP'04). IEEE International Conference on*, volume 1, pages I–85. IEEE.

Leung, K.-Y., Mak, M.-W., Siu, M.-H., and Kung, S.-Y. (2006). Adaptive articulatory feature-based conditional pronunciation modeling for speaker verification. *Speech Communication*, 48(1):71–84.

Li, M., Kambhamettu, C., and Stone, M. (2005). Automatic contour tracking in ultrasound images. *Clinical linguistics & phonetics*, 19(6-7):545–554.

Li, M., Kim, J., Lammert, A., Ghosh, P. K., Ramanarayanan, V., and Narayanan, S. (2016a). Speaker verification based on the fusion of speech acoustics and inverted articulatory signals. *Computer Speech & Language*, 36:196 – 211.

Li, M., Kim, J., Lammert, A., Ghosh, P. K., Ramanarayanan, V., and Narayanan, S. (2016b). Speaker verification based on the fusion of speech acoustics and inverted articulatory signals. *Computer speech & language*, 36:196–211.

Ling, Z.-H., Richmond, K., Yamagishi, J., and Wang, R.-H. (2009). Integrating articulatory features into hmm-based parametric speech synthesis. *IEEE Transactions on Audio, Speech, and Language Processing*, 17(6):1171–1185.

Liu, P., Yu, Q., Wu, Z., Kang, S., Meng, H., and Cai, L. (2015). A deep recurrent approach for acoustic-to-articulatory inversion. In *Acoustics, Speech and Signal Processing (ICASSP), 2015 IEEE International Conference on*, pages 4450–4454. IEEE.

Lofqvist, A. (2007). Tongue movement kinematics in long and short Japanese consonants. *Journal of the Acoustical Society of America*, 122(1):512–518.

Lombart, J., Miguel, A., and Lleida, E. (2014). Articulatory feature extraction from voice and their impact on hybrid acoustic models. In *Advances in Speech and Language Technologies for Iberian Languages*, pages 138–147. Springer.

Makhoul, J. (1975). Linear prediction: A tutorial review. *Proceedings of the IEEE*, 63(4):561–580.

Martin, A., Doddington, G., Kamm, T., Ordowski, M., and Przybocki, M. (1997). The det curve in assessment of detection task performance. Technical report, DTIC Document.

Mary, L. and Yegnanarayana, B. (2008). Extraction and representation of prosodic features for language and speaker recognition. *Speech communication*, 50(10):782–796.

McFee, B., Raffel, C., Liang, D., Ellis, D. P., McVicar, M., Battenberg, E., and Nieto, O. (2015). librosa: Audio and music signal analysis in python. In *Proceedings of the 14th python in science conference*, pages 18–25.

Ménard, L. and Noiray, A. (2011). The development of lingual gestures in speech: Comparing synthesized vocal tracts with natural vowels. *Faits de langue*, 37:189–202.

Michailovich, O. V. and Tannenbaum, A. (2006). Despeckling of medical ultrasound images. *ieee transactions on ultrasonics, ferroelectrics, and frequency control*, 53(1):64–78.

Mohamed, A.-r., Dahl, G. E., Hinton, G., et al. (2012). Acoustic modeling using deep belief networks. *IEEE Trans. Audio, Speech & Language Processing*, 20(1):14–22.

Morrison, G. (2008). Forensic: voice comparison using likelihood ratios based on polynomial curves fitted to the formant trajectories of australian english /ai/. *International Journal of Speech, Language & the Law*, 15(2).

Naik, D. (1995). Pole-filtered cepstral mean subtraction. In *icassp*, volume 95, pages 157–160.

Narayanan, S., Toutios, A., Ramanarayanan, V., Lammert, A., Kim, J., Lee, S., Nayak, K., Kim, Y.-C., Zhu, Y., Goldstein, L., Byrd, D., Bresch, E., Ghosh, P., Katsamanis, A., and Proctor, M. (2014). Real-time magnetic resonance imaging and electromagnetic articulography database for speech production research (tc). *The Journal of the Acoustical Society of America*, 136:1307.

of Manitoba, U., Services, I., and Technology (citado en 2019). Ultrasound imaging overview (online). Disponible en: http://umanitoba.ca/faculties/health_sciences/medicine/units/cacs/sam/8478.html.

Olaszy, G. (2013). Precíziós, párhuzamos magyar beszédadatbázis fejlesztése és szolgáltatásai [development and services of a hungarian precisely labeled and segmented, parallel speech database]. *Beszédkutatás 2013 [Speech Res. 2013]*, pages 261–270.

Ouni, S. and Laprie, Y. (2005). Modeling the articulatory space using a hypercube codebook for acoustic-to-articulatory inversion. *The Journal of the Acoustical Society of America*, 118(1):444–460.

Patiño Saucedo, A., Sepúlveda Sepúlveda, A., Cajas, G., and Ferney, D. (2015). Phoneme recognition system using articulatory-type information. *Tecciencia*, 10(19):11–14.

Potard, B., Laprie, Y., and Ouni, S. (2008). Incorporation of phonetic constraints in acoustic-to-articulatory inversion. *Journal of the Acoustical Society of America*, 123(4):2310–2323.

Prasad, A., Periyasamy, V., and Ghosh, P. K. (2015). Estimation of the invariant and variant characteristics in speech articulation and its application to speaker identification. In *2015 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 4265–4269.

Preston, J. L., Byun, T. M., Boyce, S. E., Hamilton, S., Tiede, M., Phillips, E., Rivera-Campos, A., and Whalen, D. H. (2017). Ultrasound images of the tongue: A tutorial for assessment and remediation of speech sound errors. *Journal of visualized experiments: JoVE*, (119).

Prince, S. J. and Elder, J. H. (2007). Probabilistic linear discriminant analysis for inferences about identity. In *Computer Vision, 2007. ICCV 2007. IEEE 11th International Conference on*, pages 1–8. IEEE.

Qin, C., Carreira-Perpinán, M. A., Richmond, K., Wrench, A., and Renals, S. (2008). Predicting tongue shapes from a few landmark locations. In *Ninth Annual Conference of the International Speech Communication Association*.

Rabiner, L. R. and Juang, B.-H. (1993). *Fundamentals of speech recognition*, volume 14. PTR Prentice Hall Englewood Cliffs.

Rao, K. S. and Sarkar, S. (2014). *Robust speaker recognition in noisy environments*. Springer.

Reynolds, D. A. (1994). Experimental evaluation of features for robust speaker identification. *IEEE Transactions on Speech and Audio Processing*, 2(4):639–643.

Reynolds, D. A., Quatieri, T. F., and Dunn, R. B. (2000). Speaker verification using adapted gaussian mixture models. *Digital signal processing*, 10(1-3):19–41.

Reynolds, D. A., Rose, R. C., et al. (1995). Robust text-independent speaker identification using gaussian mixture speaker models. *IEEE transactions on speech and audio processing*, 3(1):72–83.

Richmond, K. (2002). Estimating articulatory parameters from the acoustic speech signal.

Richmond, K. (2006). A trajectory mixture density network for the acoustic-articulatory inversion mapping. In *Ninth International Conference on Spoken Language Processing*.

Rose, P. (2002). *Forensic Speaker Identification*. Forensic Science Series. Taylor & Francis.

Sadjadi, S. O., Slaney, M., and Heck, L. (2013). Msr identity toolbox v1. 0: A matlab toolbox for speaker-recognition research. *Speech and Language Processing Technical Committee Newsletter*, 1(4):1–32.

Sampat, M. P., Wang, Z., Gupta, S., Bovik, A. C., and Markey, M. K. (2009). Complex wavelet structural similarity: A new image similarity index. *IEEE transactions on image processing*, 18(11):2385–2401.

Schafer, R. (2008). *Homomorphic Systems and Cepstrum Analysis of Speech*. Springer.

Schönle, P. W., Gräbe, K., Wenig, P., Höhne, J., Schrader, J., and Conrad, B. (1987). Electromagnetic articulography: Use of alternating magnetic fields for tracking movements of multiple points inside and outside the vocal tract. *Brain and Language*, 31(1):26–35.

Scobbie, J. M., Wrench, A. A., and van der Linden, M. (2008). Head-probe stabilisation in ultrasound tongue imaging using a headset to permit natural head movement. In *Proceedings of the 8th International seminar on speech production*, pages 373–376.

Sepúlveda, A., Guido, R. C., and Castellanos-Dominguez, G. (2013). Estimation of relevant time-frequency features using Kendall coefficient for articulator position inference. *Speech Communication*, 55(1):99–110.

Stolar, S. and Gick, B. (2013). An index for quantifying tongue curvature. *Canadian Acoustics*, 41(1).

Stone, M. (2005). A guide to analysing tongue motion from ultrasound images. *Clinical linguistics & phonetics*, 19(6-7):455–501.

Toda, T., Black, A. W., and Tokuda, K. (2008). Statistical mapping between articulatory movements and acoustic spectrum using a gaussian mixture model. *Speech Communication*, 50(3):215–227.

Tóth, L., Gosztolya, G., Grósz, T., Markó, A., and Csapó, T. G. (2018). Multi-task learning of speech recognition and speech synthesis parameters for ultrasound-based silent speech interfaces. In *Proc. Interspeech 2018*, pages 3172–3176.

Uria, B., Murray, I., Renals, S., and Richmond, K. (2012). Deep architectures for articulatory inversion. In *Thirteenth Annual Conference of the International Speech Communication Association*.

Wang, H., Wang, S., Denby, B., and Dang, J. (2015). Automatic tongue contour tracking in ultrasound sequences without manual initialization. In *2015 Asia-Pacific Signal and Information Processing Association Annual Summit and Conference (APSIPA)*, pages 200–203. IEEE.

Wang, L., Qian, X., Han, W., and Soong, F. K. (2010). Synthesizing photo-real talking head via trajectory-guided sample selection. In *Eleventh Annual Conference of the International Speech Communication Association*.

Wang, Z., Bovik, A. C., Sheikh, H. R., and Simoncelli, E. P. (2004). Image quality assessment: from error visibility to structural similarity. *IEEE transactions on image processing*, 13(4):600–612.

Wei, J., Fang, Q., Zheng, X., Lu, W., He, Y., and Dang, J. (2016). Mapping ultrasound-based articulatory images and vowel sounds with a deep neural network framework. *Multimedia Tools and Applications*, 75(9):5223–5245.

Whalen, D. H., Iskarous, K., Tiede, M. K., Ostry, D. J., Lehnert-LeHouillier, H., Vatikiotis-Bateson, E., and Hailey, D. S. (2005). The haskins optically corrected ultrasound system (hocus). *Journal of Speech, Language, and Hearing Research*, 48(3):543–553.

Wu, Z., Zhao, K., Wu, X., Lan, X., and Meng, H. (2015). Acoustic to articulatory mapping with deep neural network. *Multimedia Tools and Applications*, 74(22):9889–9907.

Xu, K., Gábor Csapó, T., Roussel, P., and Denby, B. (2016a). A comparative study on the contour tracking algorithms in ultrasound tongue images with automatic re-initialization. *The Journal of the Acoustical Society of America*, 139(5):EL154–EL160.

Xu, K., Gábor Csapó, T., Roussel, P., and Denby, B. (2016b). A comparative study on the contour tracking algorithms in ultrasound tongue images with automatic re-initialization. *The Journal of the Acoustical Society of America*, 139(5):EL154–EL160.

Xu, K., Roussel, P., Csapó, T. G., and Denby, B. (2017). Convolutional neural network-based automatic classification of midsagittal tongue gestural targets using b-mode ultrasound images. *The Journal of the Acoustical Society of America*, 141(6):EL531–EL537.

Yannis, S. (1996). *Harmonic plus Noise Models for Speech, Combined with Statistical Models, for Speech and Speaker Modification*. PhD thesis, l'Ecole Nationale Supérieure des Télécommunications.

Young, S., Evermann, G., Gales, M., Hain, T., Kershaw, D., Liu, X., Moore, G., Odell, J., Ollason, D., Povey, D., et al. (2002). The htk book. *Cambridge university engineering department*, 3:175.

Zhang, L. and Renals, S. (2008). Acoustic-articulatory modelling with the trajectory hmm.

Zhang, Y., Long, Y., Shen, X., Wei, H., Yang, M., Ye, H., and Mao, H. (2017a). Articulatory movement features for short-duration text-dependent speaker verification. *International Journal of Speech Technology*, 20.

Zhang, Y., Long, Y., Shen, X., Wei, H., Yang, M., Ye, H., and Mao, H. (2017b). Articulatory movement features for short-duration text-dependent speaker verification. *International Journal of Speech Technology*, 20(4):753–759.

Zheng, N., Lee, T., and Ching, P. C. (2007). Integration of complementary acoustic features for speaker recognition. *IEEE Signal Processing Letters*, 14(3):181–184.

Apéndices

Apéndice A. Código en python para la toma de datos sincronizada

Se anexa el código desarrollado con el propósito de adquirir la información acústica y articulatoria en paralelo:

```
import pyaudio
import wave
import time
from pynput import keyboard


print('::::::␣Toma␣de␣Datos␣2.0␣␣::::::::␣\n')
name_wav_file = (raw_input("Ingresa␣el␣nombre␣del␣archivo␣de␣audio␣:␣"))+'.wav'
print("Has␣insertado␣", name_wav_file)


class Recorder(object):
    '''A recorder class for recording audio to a WAV file.
    Records in mono by default.
    '''

    def __init__(self, channels=1, rate=44100, frames_per_buffer=1024):
        self.channels = channels
        self.rate = rate
        self.frames_per_buffer = frames_per_buffer

    def open(self, fname, mode='wb'):
```

```
        return RecordingFile(fname, mode, self.channels, self.rate,
                             self.frames_per_buffer)


class RecordingFile(object):
    def __init__(self, fname, mode, channels,
                 rate, frames_per_buffer):
        self.fname = fname
        self.mode = mode
        self.channels = channels
        self.rate = rate
        self.frames_per_buffer = frames_per_buffer
        self._pa = pyaudio.PyAudio()
        self.wavefile = self._prepare_file(self.fname, self.mode)
        self._stream = None

    def __enter__(self):
        return self

    def __exit__(self, exception, value, traceback):
        self.close()

    def record(self, duration):
        # Use a stream with no callback function in blocking mode
        self._stream = self._pa.open(format=pyaudio.paInt16,
                                     channels=self.channels,
```

```
                                        rate=self.rate,
                                        input=True,
                                        frames_per_buffer=self.frames_per_buffer)
        for _ in range(int(self.rate / self.frames_per_buffer * duration)):
            audio = self._stream.read(self.frames_per_buffer)
            self.wavefile.writeframes(audio)
        return None


    def start_recording(self):
        # Use a stream with a callback in non-blocking mode
        self._stream = self._pa.open(format=pyaudio.paInt16,
                                        channels=self.channels,
                                        rate=self.rate,
                                        input=True,
                                        frames_per_buffer=self.frames_per_buffer,
                                        stream_callback=self.get_callback())
        self._stream.start_stream()
        print('----->Esta grabando')
        return self


    def stop_recording(self):
        self._stream.stop_stream()
        print('------->Finalizado')
        return self
```

```
    def get_callback(self):
        def callback(in_data, frame_count, time_info, status):
            self.wavefile.writeframes(in_data)
            return in_data, pyaudio.paContinue
        return callback


    def close(self):
        self._stream.close()
        self._pa.terminate()
        self.wavefile.close()

    def _prepare_file(self, fname, mode='wb'):
        wavefile = wave.open(fname, mode)
        wavefile.setnchannels(self.channels)
        wavefile.setsampwidth(self._pa.get_sample_size(pyaudio.paInt16))
        wavefile.setframerate(self.rate)
        return wavefile


def on_press(key):
    print('alphanumeric key {0} pressed'.format(key))


space_counts = 0
rec = Recorder(channels=2)
```

```
recfileh = rec.open(name_wav_file, 'wb')#NOMBRE DEL ARCHIVO WAV ::-> Cambiar


def on_release(key):
    global space_counts
    #print('{0} released'.format(key))
    if key == keyboard.Key.esc:
        # Stop listener
        return False
    if key == keyboard.Key.space:
        tic = time.time()
        space_counts = 1 + space_counts
        print('HOLI oprimiste {0}'.format(key))
        print(space_counts)
        if space_counts > 1:
            #with rec.open('nonblocking.wav', 'wb') as recfileh:
                recfileh.stop_recording()
                space_counts = 0
             #print(space_counts)
        else:
                recfileh.start_recording()
                toc = time.time()
                print (toc-tic, 'sec Elapsed')
                #time.sleep(5)
                #recfileh.stop_recording()
```

```
# Collect events until released
with keyboard.Listener(
        on_press=on_press,
        on_release=on_release) as listener:
    listener.join()
```

Apéndice B. Archivo de configuración y Listas para HTK

En la extracción de características se utiliza la librería HTKCopy que pertenece al sofware HTK, con el comando:

```
HCopy -A -D -T 1 -C hcopy_AVS_pruebas -S lista_para_HTK.lst
```

Algunas opciones fueron descritas en la sección 4.1.2. A continuación se muestra el archivo de configuración utilizado con el nombre *hcopy_AVS_pruebas*:

```
SOURCEFORMAT = WAV
TARGETKIND = MFCC_0_D_A
SOURCERATE = 226.757369
TARGETRATE = 780000.0
SAVECOMPRESSED = F
SAVEWITHCRC = F
WINDOWSIZE = 1560000.0
USEHAMMING = T
PREEMCOEF = 0.97
NUMCHANS = 27
#CEPLIFTER = 10
NUMCEPS = 12
LOFREQ = 300
HIFREQ = 4000
ENORMALIZE = T     # log energy is normalized to 1.0 for each utterance
ZMEANSOURCE = T    # bias is subtracted from waveform if any
```

Y en seguida basta con mostrar las dos primeras líneas del archivo de lista, pues las otras lineas siguen el mismo formato, pero con el nombre del archivo correspondiente. El archivo de lista *lista_para_HTK.lst*:

D:\Proyecto_Maestria\Datos\UIS_colombian\wav\f01_rec_01.wav D:\Proyecto_Maestria\Datos\UIS_colombian\f

D:\Proyecto_Maestria\Datos\UIS_colombian\wav\f01_rec_02.wav D:\Proyecto_Maestria\Datos\UIS_colombian\f

Apéndice C. VAD y normalización CMVN de características

Script de Matlab utilizado para utilizar el VAD de la toolbox voicebox y la normalización de la MSR toolkit:

```
% Este programa implementa extraccion de caracteristicas.
% Busca tener la suficiente flexibilidad como para imitar
% diferentes configuraciones sugeridas, especialmente la
% inclusion del modulo VAD.

clc; clear all; close all;

path(path,'D:\Proyecto_Maestria\Tools\voicebox');     % VAD
path(path, 'D:\Proyecto_Maestria\Tools\MSR_Identity_Toolkit_v1\code'); % readhtk, writehtk

%en el archivo sig estan los nombres fuente y destino de la extraccion de caracteristicas
spk_feature_folder = 'lst\';
%spk_listas = {'lista_para_HTK_Noisy.lst'};
spk_listas = {'lista_para_HTK.lst'};
spk_feature_lst = strcat(spk_feature_folder, spk_listas);


fid = fopen(spk_feature_lst{1});
C = textscan(fid, '%s %s\n');  % en C{1}: .wav files ; en C{2} .htk files
fclose(fid);
Wav_files = C{1};    Htk_files = C{2};

for(cont=1:length(Wav_files))
```

```
    audio_file_name = char(Wav_files(cont));
    [wav, Fs] = audioread(audio_file_name);     signal = wav(:, 1);
    % se lee las caracteristicas estimadas usando htk.
    htk_name = char(Htk_files(cont));
htk_name
    [data, frate, feakind] = htkread(htk_name);    % del MSR
    data = cmvn(data, 'true'); % aplica cepstral mean a variance normalization.
   % delete(htk_name);
  %htk_name
    % ahora el VAD.
    vs = vadsohn(signal, Fs, 'a');    % aplica el VAD de voicebox.
    veces = ceil((frate/1e7)*Fs);
    temp = decimate(vs, veces);    VAD = temp(2: length(temp)-1);    VAD = VAD';
clear temp;  clear vs;
    vad_index = find(VAD>0);
    mfcc_feat = data(:, vad_index(1:length(vad_index)-1));
    % se guarda en formato htk
  % htkwrite(htk_name, mfcc_feat, frate, feakind);
end
```

Apéndice D. Fusión de características

Se anexa el script utilizado para realizar la fusión de características, despuÃ©s de utilizar el VAD y la normalización CMVN, de características.

```
% Fusion de caracteristicas acusticas y articulatorias

clc; clear all; close all;

path(path,'D:\Proyecto_Maestria\Tools\voicebox');     % VAD
path(path, 'D:\Proyecto_Maestria\Tools\MSR_Identity_Toolkit_v1\code'); % readhtk, writehtk

% en el archivo sig estan los nombres fuente y destino de la extraccion de caracteristicas
spk_feature_folder = 'lst\';
%spk_listas = {'lista_para_HTK_Noisy.lst'};
spk_listas = {'lista_para_HTK.lst'};
spk_feature_lst = strcat(spk_feature_folder, spk_listas);

%caracteristicas articulatorias
load features\POl_xi_xf_features.mat
%trainSpeakerData_POL = trainSpeakerData;
%load features\xy_features.mat

fid = fopen(spk_feature_lst{1});
C = textscan(fid, '%s      %s\n');  % en C{1}: .wav files ; en C{2} .htk files
fclose(fid);
Wav_files = C{1};    Htk_files = C{2};
```

```
temp_15 = trainSpeakerData(15,:);
temp_9 = trainSpeakerData(9:17,:);
temp_9(7,:) = [];
trainSpeakerData(9,:) = temp_15;
trainSpeakerData(10:17,:) = temp_9;

clear temp_15 temp_9

art_fea=trainSpeakerData';
art_fea = art_fea(:);
art_feat = art_fea(~cellfun('isempty',art_fea));

for(cont=1:length(Wav_files))
    audio_file_name = char(Wav_files(cont));
    [wav, Fs] = audioread(audio_file_name);     signal = wav(:, 1);
    %se lee las caracteristicas estimadas usando htk.
    htk_name = char(Htk_files(cont));
htk_name
    [data, frate, feakind] = htkread(htk_name);    %del MSR
    data = cmvn(data, 'true'); %aplica cepstral mean a variance normalization.
minimum = min(length(data),length(art_feat{cont}));
data = data(:,1:minimum);
art_featn = art_feat{cont}(:,1:minimum);
    delete(htk_name);
htk_name
```

```
    % ahora el VAD.
     vs = vadsohn(signal, Fs, 'a');    % aplica el VAD de voicebox.
     veces = ceil((frate/1e7)*Fs);
     temp = decimate(vs, veces);    VAD = temp(2: length(temp)-1);    VAD = VAD';
clear temp;  clear vs;
     vad_index = find(VAD>0);
     max_vad_index = find(vad_index>length(data));
     if isempty(max_vad_index)
     mfcc_feat = data(:, vad_index(1:length(vad_index)-1));
     art_feat_f = art_featn(:,vad_index(1:length(vad_index)-1));
     else
     mfcc_feat = data(:, vad_index(1:min(max_vad_index)-1));
     art_feat_f = art_featn(:,vad_index(1:min(max_vad_index)-1));
     end
     mfcc_art = [mfcc_feat;art_feat_f];
    % se guarda en formato htk
  htkwrite(htk_name, mfcc_art, frate, feakind);
end
```

Apéndice E. Implementación GMM-UBM

A continuación se presenta cada uno de los códigos en que se implementan los sistemas de VH utilizados en los experimentos.

7.1. GMM-UBM características acústicas (MFCC)

```
clc
clear all

%path(path, 'E:\Proyecto_Maestria\Tools\MSR_Identity_Toolkit_v1\code'); % las rutinas de M
path(path,'D:\Proyecto_Maestria\Tools\MSR_Identity_Toolkit_v1\code'); % las rutinas de MSR
%% Step0: Opening MATLAB pool
nworkers = 12;
nworkers = min(nworkers, feature('NumCores'));
isopen = parpool(3);

%% Step1: Training the UBM
dataList_folder = 'C:\Users\salpub83fdhb2\Dropbox\Mis␣Documentos␣Maestria\MATLAB\Proyecto\
for k1=1:16
for k2=(k1+1):17
j=1;
    if ((k1==14)|(k1==15)|(k2==14)|(k2==15))==1
        continue
    else
%k1=1; k2=15;
list_all = strcat('lista_UBM_k1_',int2str(k1),'_k2_',int2str(k2),'.lst');
```

```
dataList_all = {strcat(dataList_folder, list_all)};
%%%%%% nombres modelos

nmix         = 16;
model_folder = 'models\GMM_UBM\';
models_name = strcat(num2str(nmix),'gmm','_1UBM.mat');

dataList = dataList_all;
final_niter = 10;
ds_factor   = 1;
UBM = gmm_em(dataList{1}, nmix, final_niter, ds_factor, nworkers);
%load('models/GMM_UBM/UBM_256gmm_1UBM.mat')
%save('models\UBM_s1.mat','UBM')
%save(strcat(model_folder,'UBM_',models_name),'UBM')

%% Step2: Adapting the speaker models from UBM
fea_dir = 'D:\Proyecto_Maestria\Datos\UIS_colombian\features\';
fea_ext = '.htk';

enroll_list_folder = 'C:\Users\salpub83fdhb2\Dropbox\Mis␣Documentos␣Maestria\MATLAB\Proyec
enroll_list_all =strcat('lista_enroll_k1_',int2str(k1),'_k2_',int2str(k2),'.lst');
enroll_list = {strcat(enroll_list_folder, enroll_list_all)};

fid = fopen(enroll_list{1}, 'rt');
```

```
C = textscan(fid, '%s %s\n');
fclose(fid);
model_ids = unique(C{2}, 'stable');
model_files = C{1};
nspks = length(model_ids);
map_tau = 10.0;
config = 'mwv';
gmm_models = cell(nspks, 1);
for spk = 1 : nspks,
    ids = find(ismember(C{2}, model_ids{spk}));
    spk_files = model_files(ids);
    gmm_models{spk} = mapAdapt(spk_files, UBM, map_tau, config);
end
%% Step3: Scoring the verification trials
trial_list =  strcat('lst\lista_test_k1_',int2str(k1),'_k2_',int2str(k2),'.lst');

fid = fopen(trial_list, 'rt');
C = textscan(fid, '%s %%%%%% %s\n');
fclose(fid);
test_files = C{1};
test_IDs = C{2};

N_ref = length(gmm_models(:, 1));
N_test = length(test_files(:, 1));
kk = 1;
```

```
for(ii=1: N_ref)
    for(jj=1:N_test)
        trials = [ii jj];
        gmmScores(kk) = score_gmm_trials(gmm_models, test_files, trials, UBM);
        answers(kk) = strcmp(model_ids(ii), test_IDs(jj));
        kk = kk + 1;
    end
end

%==================================================================
%% Step4: Computing the EER and plotting the DET curve
%labels = C{3};
labels = answers;
eer = compute_eer(gmmScores, labels, true);
eer
res_name = strcat('Resultados\RES_',num2str(nmix),'gmm','_1UBM_k1_',int2str(k1),'_k2_',int2
save(res_name,'eer','labels','gmmScores')
    end
end
end

%% -> prueba por generos
% fid = fopen('lst\lista_test_genders.lst');
% C = textscan(fid, '%s\t%s\n');
```

```
% fclose(fid)
% genders_test = C{2};
%
% fid = fopen('lst\lista_enroll_genders.lst');
% C = textscan(fid, '%s\t%s\n');
% fclose(fid)
% genders_enroll = C{2};
% kk = 1;
%
% for(ii=1: N_ref)
%     for(jj=1:N_test)
%         if strcmp(genders_enroll{ii},genders_test{jj}) == 1
%             trials = [ii jj];
%             gmmScores_g(kk) = score_gmm_trials(gmm_models, test_files, trials, UBM);
%             answers_g(kk) = strcmp(model_ids(ii), test_IDs(jj));
%             kk = kk + 1;
%         end
%     end
% end
% %
%  figure(3);
%  eer_g = compute_eer(gmmScores_g, answers_g, true);
%  eer_g
%
% save('Resultados\gmmtest_sg_test.mat','gmmScores_g','answers_g')
```

```
delete(isopen)
```

7.2. GMM-UBM características articulatorias - poly

```
clc; clear all; close all;


%Script para realizar un experimento de verificacion
% loading the features
load datax\dx_f01.mat; dx_f01=datax; clear datax;
load datax\dx_f02.mat; dx_f02=datax; clear datax;
load datax\dx_f03.mat; dx_f03=datax; clear datax;
load datax\dx_f04.mat; dx_f04=datax; clear datax;
load datax\dx_f05.mat; dx_f05=datax; clear datax;
load datax\dx_f06.mat; dx_f06=datax; clear datax;
load datax\dx_f07.mat; dx_f07=datax; clear datax;
load datax\dx_f08.mat; dx_f08=datax; clear datax;

load datax\dx_m01.mat; dx_m01=datax; clear datax;
load datax\dx_m02.mat; dx_m02=datax; clear datax;
load datax\dx_m03.mat; dx_m03=datax; clear datax;
load datax\dx_m04.mat; dx_m04=datax; clear datax;
load datax\dx_m05.mat; dx_m05=datax; clear datax;
load datax\dx_m06.mat; dx_m06=datax; clear datax;

load datax\dx_py_m01.mat; dx_py_m01=datax; clear datax;
load datax\dx_py_m02.mat; dx_py_m02=datax; clear datax;
```

```
load datax\dx_py_f01.mat; dx_py_f01=datax; clear datax;

load features\trainSpeakerData

path(path,'D:\Proyecto_Maestria\Tools\MSR_Identity_Toolkit_v1\code'); % las rutinas de MSR
%% Step0: Opening MATLAB pool
 nworkers = 3;
 nworkers = min(nworkers, feature('NumCores'));
 isopen = parpool(3);

%% Step1: Training the UBM
nDims = 9; % dimensionality of feature vectors
nChannels = 30; % Number of channels (sessions) per speaker
nSpeakers = 17;
%trainSpeakerData = cell(nSpeakers, nChannels);
nChannels_test=6;

for n=1:nChannels
    trainSpeakerData{1,n} = [trainSpeakerData{1,n};dx_f01(n).xi;dx_f01(n).xf];
%d1)
    trainSpeakerData{2,n} = [trainSpeakerData{2,n};dx_f02(n).xi;dx_f02(n).xf];
%2)
    trainSpeakerData{3,n} = [trainSpeakerData{3,n};dx_f03(n).xi;dx_f03(n).xf];
%3)
    trainSpeakerData{4,n} = [trainSpeakerData{4,n};dx_f04(n).xi;dx_f04(n).xf];
```

```
%4)
    trainSpeakerData{5,n} = [trainSpeakerData{5,n};dx_f05(n).xi;dx_f05(n).xf];
%5)
    trainSpeakerData{6,n} = [trainSpeakerData{6,n};dx_f06(n).xi;dx_f06(n).xf];
%6)
    trainSpeakerData{7,n} = [trainSpeakerData{7,n};dx_f07(n).xi;dx_f07(n).xf];
%7)
    trainSpeakerData{8,n} = [trainSpeakerData{8,n};dx_f08(n).xi;dx_f08(n).xf];
%8)

    trainSpeakerData{9,n} = [trainSpeakerData{9,n};dx_m01(n).xi;dx_m01(n).xf];
%9)
    trainSpeakerData{10,n} = [trainSpeakerData{10,n};dx_m02(n).xi;dx_m02(n).xf];
%d10)
    trainSpeakerData{11,n} = [trainSpeakerData{11,n};dx_m03(n).xi;dx_m03(n).xf];
%d11)
    trainSpeakerData{12,n} = [trainSpeakerData{12,n};dx_m04(n).xi;dx_m04(n).xf];
%d12)


    trainSpeakerData{15,n} = [trainSpeakerData{15,n};dx_py_f01(n).xi;dx_py_f01(n).xf]; %d13
    trainSpeakerData{16,n} = [trainSpeakerData{16,n};dx_py_m01(n).xi;dx_py_m01(n).xf]; %d14
    trainSpeakerData{17,n} = [trainSpeakerData{17,n};dx_py_m02(n).xi;dx_py_m02(n).xf]; %d15
```

```
        if n>15
            trainSpeakerData{13,n} = [];
        else
            trainSpeakerData{13,n} = [trainSpeakerData{13,n};dx_m05(n).xi;dx_m05(n).xf];
%d10)
        end

    if n>20
        trainSpeakerData{14,n} = []; %20 sesiones
    else
        trainSpeakerData{14,n} = [trainSpeakerData{14,n};dx_m06(n).xi;dx_m06(n).xf];
%d11)
    end

    speakerID(1,n) = 1; %Keep track of who this is
    speakerID(2,n) = 2; %Keep track of who this is
    speakerID(3,n) = 3; %Keep track of who this is
    speakerID(4,n) = 4; %Keep track of who this is
    speakerID(5,n) = 5; %Keep track of who this is
    speakerID(6,n) = 6; %Keep track of who this is
    speakerID(7,n) = 7; %Keep track of who this is
    speakerID(8,n) = 8; %Keep track of who this is
    speakerID(9,n) = 9; %Keep track of who this is
    speakerID(10,n) = 10; %Keep track of who this is
    speakerID(11,n) = 11; %Keep track of who this is
```

```
    speakerID(12,n) = 12; %Keep track of who this is
    speakerID(13,n) = 13; %Keep track of who this is
    speakerID(14,n) = 14; %Keep track of who this is
    speakerID(15,n) = 15; %Keep track of who this is
    speakerID(16,n) = 16; %Keep track of who this is
    speakerID(17,n) = 17; %Keep track of who this is

end
save features\speakerID.mat speakerID
save features\POl_xi_xf_features.mat trainSpeakerData
 for n=1:nChannels_test
     testSpeakerData{1,n} = trainSpeakerData{1,n+24};
     testSpeakerData{2,n} = trainSpeakerData{2,n+24};
     testSpeakerData{3,n} = trainSpeakerData{3,n+24};
     testSpeakerData{4,n} = trainSpeakerData{4,n+24};
     testSpeakerData{5,n} = trainSpeakerData{5,n+24};
     testSpeakerData{6,n} = trainSpeakerData{6,n+24};
     testSpeakerData{7,n} = trainSpeakerData{7,n+24};
     testSpeakerData{8,n} = trainSpeakerData{8,n+24};

     testSpeakerData{9,n} = trainSpeakerData{9,n+24};
     testSpeakerData{10,n} = trainSpeakerData{10,n+24};
     testSpeakerData{11,n} = trainSpeakerData{11,n+24};
     testSpeakerData{12,n} = trainSpeakerData{12,n+24};
```

```
        testSpeakerData{13,n} = trainSpeakerData{13,n+24};
        testSpeakerData{14,n} = trainSpeakerData{14,n+24};
        testSpeakerData{15,n} = trainSpeakerData{15,n+24};
        testSpeakerData{16,n} = trainSpeakerData{16,n+24};
        testSpeakerData{17,n} = trainSpeakerData{17,n+24};


%---Ids
        test_speakerID(1,n) = 1; %Keep track of who this is
        test_speakerID(2,n) = 2; %Keep track of who this is
        test_speakerID(3,n) = 3; %Keep track of who this is
        test_speakerID(4,n) = 4; %Keep track of who this is
        test_speakerID(5,n) = 5; %Keep track of who this is
        test_speakerID(6,n) = 6; %Keep track of who this is
        test_speakerID(7,n) = 7; %Keep track of who this is
        test_speakerID(8,n) = 8; %Keep track of who this is
        test_speakerID(9,n) = 9; %Keep track of who this is
        test_speakerID(10,n) = 10; %Keep track of who this is
        test_speakerID(11,n) = 11; %Keep track of who this is
        test_speakerID(12,n) = 12; %Keep track of who this is
        test_speakerID(13,n) = 13; %Keep track of who this is
        test_speakerID(14,n) = 14; %Keep track of who this is
        test_speakerID(15,n) = 15; %Keep track of who this is
        test_speakerID(16,n) = 16; %Keep track of who this is
        test_speakerID(17,n) = 17; %Keep track of who this is
    end
```

```
% seleccion de los datos para entrenar el UBM

for k1=1:16
for k2=(k1+1):17
j=1;
    if ((k1==13)|(k1==14)|(k2==13)|(k2==14))==1
        continue
    else
for n =1:17
    if ((n==k1)|(n==k2))==0
      trainSpeakerData_K(j,:) = trainSpeakerData(n,:);
      j = j+1;
    end
end
testSpeakerData_K(1,:)= testSpeakerData(k1,:);
testSpeakerData_K(2,:)= testSpeakerData(k2,:);
test_speakerID_K(1,:) = test_speakerID(k1,:); %Keep track of who this is
test_speakerID_K(2,:) = test_speakerID(k2,:); %Keep track of who this is

%seleccion para los datos de prueba


nmix         = 16;
model_folder = 'Experimento\models\';
```

```
models_name = strcat(model_folder,num2str(nmix),'gmm','_1UBM_k1_',int2str(k1),'_k2_',int2st

final_niter = 10;
ds_factor   = 1;

A = trainSpeakerData_K(:);
A = A(~cellfun('isempty',A));

ubm = gmm_em(A, nmix, final_niter, ds_factor, nworkers);
save(models_name,'ubm')
%% Step2: Adapting the speaker models from UBM
% Step2: Now adapt the UBM to each speaker to create GMM speaker model.

map_tau = 10.0;
config = 'mwv';
gmm = cell(2, 1);

if k1 == 16
    C = trainSpeakerData(k1, 1:24) ;
    C = C(~cellfun('isempty',C));
    gmm{1} = mapAdapt(C, ubm, map_tau, config);
else
gmm{1} = mapAdapt(trainSpeakerData(k1, 1:24), ubm, map_tau, config);
end
```

```
if k2 == 16
    B = trainSpeakerData(k2, 1:24) ;
    B = B(~cellfun('isempty',B));
    gmm{2} = mapAdapt(B, ubm, map_tau, config);
else
gmm{2} = mapAdapt(trainSpeakerData(k2, 1:24), ubm, map_tau, config);
end

%%
% Step3: Now calculate the score for each model versus each speaker's
% data.
% Generate a list that tests each model (first column) against all the
% testSpeakerData.

%trials = zeros(nSpeakers*nChannels_test*nSpeakers, 2);
%answers = zeros(nSpeakers*nChannels_test*nSpeakers, 1);
kk = 1;
test_speakerID_K=test_speakerID_K(:);

for ii=1:length(unique(test_speakerID_K, 'stable'))
   for jj=1:length(test_speakerID_K)
       trials(kk,:) = [ii jj];
       answers(kk) = test_speakerID_K(ii) == test_speakerID_K(jj);
       kk = kk + 1;
   end
```

```
end
gmmScores = score_gmm_trials(gmm, testSpeakerData_K(:), trials, ubm);

%% Step4: Computing the EER and plotting the DET curve
%labels = C{3};
labels = answers;
eer = compute_eer(gmmScores, labels, true);
eer
% save res eer labels gmmScores
res_folder = 'Experimento\res\';
res_name = strcat(res_folder,'RES_',num2str(nmix),'gmm','_1UBM_k1_',int2str(k1),'_k2_',int2
save(res_name,'eer','labels','gmmScores')
clear test_speakerID_K
end
end
end
delete(isopen)
```

7.3. GMM-UBM características articulatorias - 5 puntos

```
clc; clear all; close all;

%Script para realizar un experimento de verificacion
% loading the features
load datax\dxy_f01.mat; dxy_f01=dataxy; clear dataxy;
load datax\dxy_f02.mat; dxy_f02=dataxy; clear dataxy;
```

```
load datax\dxy_f03.mat; dxy_f03=dataxy; clear dataxy;
load datax\dxy_f04.mat; dxy_f04=dataxy; clear dataxy;
load datax\dxy_f05.mat; dxy_f05=dataxy; clear dataxy;
load datax\dxy_f06.mat; dxy_f06=dataxy; clear dataxy;
load datax\dxy_f07.mat; dxy_f07=dataxy; clear dataxy;
load datax\dxy_f08.mat; dxy_f08=dataxy; clear dataxy;

load datax\dxy_m01.mat; dxy_m01=dataxy; clear dataxy;
load datax\dxy_m02.mat; dxy_m02=dataxy; clear dataxy;
load datax\dxy_m03.mat; dxy_m03=dataxy; clear dataxy;
load datax\dxy_m04.mat; dxy_m04=dataxy; clear dataxy;
load datax\dxy_m05.mat; dxy_m05=dataxy; clear dataxy;
load datax\dxy_m06.mat; dxy_m06=dataxy; clear dataxy;

load datax\dxy_py_m01.mat; dxy_py_m01=dataxy; clear dataxy;
load datax\dxy_py_m02.mat; dxy_py_m02=dataxy; clear dataxy;
load datax\dxy_py_f01.mat; dxy_py_f01=dataxy; clear dataxy;

path(path,'D:\Proyecto_Maestria\Tools\MSR_Identity_Toolkit_v1\code'); % las rutinas de MSR
%% Step0: Opening MATLAB pool
 nworkers = 3;
 nworkers = min(nworkers, feature('NumCores'));
 isopen = parpool(3);

%% Step1: Training the UBM
```

```
nDims = 10; % dimensionality of feature vectors
nChannels = 30; % Number of channels (sessions) per speaker
nSpeakers = 17;
trainSpeakerData = cell(nSpeakers, nChannels);
nChannels_test=6;

for n=1:nChannels
    trainSpeakerData{1,n} = [dxy_f01(n).x dxy_f01(n).y]';     %1)
    trainSpeakerData{2,n} = [dxy_f02(n).x dxy_f02(n).y]';     %2)
    trainSpeakerData{3,n} = [dxy_f03(n).x dxy_f03(n).y]';     %3)
    trainSpeakerData{4,n} = [dxy_f04(n).x dxy_f04(n).y]';     %4)
    trainSpeakerData{5,n} = [dxy_f05(n).x dxy_f05(n).y]';     %5)
    trainSpeakerData{6,n} = [dxy_f06(n).x dxy_f06(n).y]';     %6)
    trainSpeakerData{7,n} = [dxy_f07(n).x dxy_f07(n).y]';     %7)
    trainSpeakerData{8,n} = [dxy_f08(n).x dxy_f08(n).y]';     %8)

    trainSpeakerData{9,n} = [dxy_m01(n).x dxy_m01(n).y]';     %9)
    trainSpeakerData{10,n} = [dxy_m02(n).x dxy_m02(n).y]';    %10)
    trainSpeakerData{11,n} = [dxy_m03(n).x dxy_m03(n).y]';    %11)
    trainSpeakerData{12,n} = [dxy_m04(n).x dxy_m04(n).y]';    %12)

    %trainSpeakerData{10,n} = [dxy_f07(n).x dxy_f07(n).y]';    %13)
    %trainSpeakerData{11,n} = [dxy_m04(n).x dxy_m04(n).y]';    %14)
```

```
    trainSpeakerData{15,n} = [dxy_py_f01(n).x dxy_py_f01(n).y]'; %d15)
    trainSpeakerData{16,n} = [dxy_py_m01(n).x dxy_py_m01(n).y]'; %d16)
    trainSpeakerData{17,n} = [dxy_py_m02(n).x dxy_py_m02(n).y]'; %d16)

if n>15
        trainSpeakerData{13,n} = [];
    else
        trainSpeakerData{13,n} = [dxy_m05(n).x dxy_m05(n).y]';     %d10)
    end

    if n>20
    trainSpeakerData{14,n} = []; %20 sesiones
    else
    trainSpeakerData{14,n} = [dxy_m06(n).x dxy_m06(n).y]';     %d11)
    end

    speakerID(1,n) = 1; %Keep track of who this is
    speakerID(2,n) = 2; %Keep track of who this is
    speakerID(3,n) = 3; %Keep track of who this is
    speakerID(4,n) = 4; %Keep track of who this is
    speakerID(5,n) = 5; %Keep track of who this is
    speakerID(6,n) = 6; %Keep track of who this is
    speakerID(7,n) = 7; %Keep track of who this is
    speakerID(8,n) = 8; %Keep track of who this is
    speakerID(9,n) = 9; %Keep track of who this is
```

```
    speakerID(10,n) = 10; %Keep track of who this is
    speakerID(11,n) = 11; %Keep track of who this is
    speakerID(12,n) = 12; %Keep track of who this is
    speakerID(13,n) = 13; %Keep track of who this is
    speakerID(14,n) = 14; %Keep track of who this is
    speakerID(15,n) = 15; %Keep track of who this is
    speakerID(16,n) = 16; %Keep track of who this is
    speakerID(17,n) = 17; %Keep track of who this is
end
save features\xy_features.mat trainSpeakerData
 for n=1:nChannels_test
    testSpeakerData{1,n} = trainSpeakerData{1,n+24};
    testSpeakerData{2,n} = trainSpeakerData{2,n+24};
    testSpeakerData{3,n} = trainSpeakerData{3,n+24};
    testSpeakerData{4,n} = trainSpeakerData{4,n+24};
    testSpeakerData{5,n} = trainSpeakerData{5,n+24};
    testSpeakerData{6,n} = trainSpeakerData{6,n+24};
    testSpeakerData{7,n} = trainSpeakerData{7,n+24};
    testSpeakerData{8,n} = trainSpeakerData{8,n+24};

    testSpeakerData{9,n} = trainSpeakerData{9,n+24};
    testSpeakerData{10,n} = trainSpeakerData{10,n+24};
    testSpeakerData{11,n} = trainSpeakerData{11,n+24};
    testSpeakerData{12,n} = trainSpeakerData{12,n+24};
```

```
        testSpeakerData{13,n} = trainSpeakerData{13,n+24};
        testSpeakerData{14,n} = trainSpeakerData{14,n+24};
        testSpeakerData{15,n} = trainSpeakerData{15,n+24};
        testSpeakerData{16,n} = trainSpeakerData{16,n+24};
        testSpeakerData{17,n} = trainSpeakerData{17,n+24};


%---Ids
        test_speakerID(1,n) = 1; %Keep track of who this is
        test_speakerID(2,n) = 2; %Keep track of who this is
        test_speakerID(3,n) = 3; %Keep track of who this is
        test_speakerID(4,n) = 4; %Keep track of who this is
        test_speakerID(5,n) = 5; %Keep track of who this is
        test_speakerID(6,n) = 6; %Keep track of who this is
        test_speakerID(7,n) = 7; %Keep track of who this is
        test_speakerID(8,n) = 8; %Keep track of who this is
        test_speakerID(9,n) = 9; %Keep track of who this is
        test_speakerID(10,n) = 10; %Keep track of who this is
        test_speakerID(11,n) = 11; %Keep track of who this is
        test_speakerID(12,n) = 12; %Keep track of who this is
        test_speakerID(13,n) = 13; %Keep track of who this is
        test_speakerID(14,n) = 14; %Keep track of who this is
        test_speakerID(15,n) = 15; %Keep track of who this is
        test_speakerID(16,n) = 16; %Keep track of who this is
        test_speakerID(17,n) = 17; %Keep track of who this is
    end
```

```
% seleccion de los datos para entrenar el UBM


for k1=1:16
for k2=(k1+1):17
j=1;
    if ((k1==13)|(k1==14)|(k2==13)|(k2==14))==1
        continue
    else
for n =1:17
    if ((n==k1)|(n==k2))==0
      trainSpeakerData_K(j,:) = trainSpeakerData(n,:);
      j = j+1;
    end
end
testSpeakerData_K(1,:)= testSpeakerData(k1,:);
testSpeakerData_K(2,:)= testSpeakerData(k2,:);

test_speakerID_K(1,:) = test_speakerID(k1,:); %Keep track of who this is
test_speakerID_K(2,:) = test_speakerID(k2,:); %Keep track of who this is

%seleccion para los datos de prueba


nmix         = 16;
```

```
model_folder = 'Experimento\models\';
models_name = strcat(model_folder,num2str(nmix),'gmm','_1UBM_k1_',int2str(k1),'_k2_',int2st

final_niter = 10;
ds_factor   = 1;

A = trainSpeakerData_K(:);
A = A(~cellfun('isempty',A));

ubm = gmm_em(A, nmix, final_niter, ds_factor, nworkers);
save(models_name,'ubm')
%% Step2: Adapting the speaker models from UBM
% Step2: Now adapt the UBM to each speaker to create GMM speaker model.

map_tau = 10.0;
config = 'mwv';
gmm = cell(2,1);

if k1 == 16
    C = trainSpeakerData(k1, 1:24) ;
    C = C(~cellfun('isempty',C));
    gmm{1} = mapAdapt(C, ubm, map_tau, config);
else
gmm{1} = mapAdapt(trainSpeakerData(k1, 1:24), ubm, map_tau, config);
end
```

```
if k2 == 16
    B = trainSpeakerData(k2, 1:24) ;
    B = B(~cellfun('isempty',B));
    gmm{2} = mapAdapt(B, ubm, map_tau, config);
else
gmm{2} = mapAdapt(trainSpeakerData(k2, 1:24), ubm, map_tau, config);
end

%%
% Step3: Now calculate the score for each model versus each speaker's
% data.
% Generate a list that tests each model (first column) against all the
% testSpeakerData.

%trials = zeros(nSpeakers*nChannels_test*nSpeakers, 2);
%answers = zeros(nSpeakers*nChannels_test*nSpeakers, 1);
kk = 1;
test_speakerID_K=test_speakerID_K(:);

for ii=1: length(unique(test_speakerID_K, 'stable'))
   for jj=1:length(test_speakerID_K)
       trials(kk,:) = [ii jj];
       answers(kk) = test_speakerID_K(ii) == test_speakerID_K(jj);
       kk = kk + 1;
```

```
    end
end
gmmScores = score_gmm_trials(gmm, testSpeakerData_K(:), trials, ubm);

%% Step4: Computing the EER and plotting the DET curve
%labels = C{3};
labels = answers;
eer = compute_eer(gmmScores, labels, true);
eer
% save res eer labels gmmScores
res_folder = 'Experimento\res\';
res_name = strcat(res_folder,'RES_',num2str(nmix),'gmm','_1UBM_k1_',int2str(k1),'_k2_',int2
save(res_name,'eer','labels','gmmScores')
clear test_speakerID_K
end
end
end
delete(isopen)
```

7.4. GMM-UBM con fusión de características

```
clc
clear all
close all

%path(path, 'E:\Proyecto_Maestria\Tools\MSR_Identity_Toolkit_v1\code'); % las rutinas de M
```

```
path(path,'D:\Proyecto_Maestria\Tools\MSR_Identity_Toolkit_v1\code'); % las rutinas de MSR

%% Step0: Opening MATLAB pool
nworkers = 12;
nworkers = min(nworkers, feature('NumCores'));
isopen = parpool(3);

%% Step1: Training the UBM
dataList_folder = 'C:\Users\salpub83fdhb2\Dropbox\Mis␣Documentos␣Maestria\MATLAB\Proyecto\
% for k1=1:16
% for k2=(k1+1):17
k1=1; k2=3;
j=1;
%     if ((k1==14)|(k1==15)|(k2==14)|(k2==15))==1
 %         continue
 %    else
%k1=1; k2=15;
list_all = strcat('lista_UBM_k1_',int2str(k1),'_k2_',int2str(k2),'.lst');
dataList_all = {strcat(dataList_folder, list_all)};
%%%%%%-> nombres modelos

nmix         = 16;
model_folder = 'models\GMM_UBM\';
models_name = strcat(num2str(nmix),'gmm','_1UBM.mat');

dataList = dataList_all;
```

```
final_niter = 10;
ds_factor   = 1;
UBM = gmm_em(dataList{1}, nmix, final_niter, ds_factor, nworkers);
%load('models/GMM_UBM/UBM_256gmm_1UBM.mat')
%save('models\UBM_s1.mat','UBM')
%save(strcat(model_folder,'UBM_',models_name),'UBM')

%% Step2: Adapting the speaker models from UBM
fea_dir = 'D:\Proyecto_Maestria\Datos\UIS_colombian\features\';
fea_ext = '.htk';

enroll_list_folder = 'C:\Users\salpub83fdhb2\Dropbox\Mis␣Documentos␣Maestria\MATLAB\Proyec
enroll_list_all =strcat('lista_enroll_k1_',int2str(k1),'_k2_',int2str(k2),'.lst');
enroll_list = {strcat(enroll_list_folder, enroll_list_all)};

fid = fopen(enroll_list{1}, 'rt');

C = textscan(fid, '%s␣%s\n');
fclose(fid);
model_ids = unique(C{2}, 'stable');
model_files = C{1};
nspks = length(model_ids);
map_tau = 10.0;

config = 'mwv';
```

```
gmm_models = cell(nspks, 1);
for spk = 1 : nspks,
    ids = find(ismember(C{2}, model_ids{spk}));
    spk_files = model_files(ids);
    gmm_models{spk} = mapAdapt(spk_files, UBM, map_tau, config);
%     for ll=1:nmix
%         if isnan(gmm_models{spk}.mu(1,ll)) == 1,
%             gmm_models{spk}.mu(:,ll) = UBM.mu(:,ll);
%         end
%         if isnan(gmm_models{spk}.sigma(1,ll)) == 1,
%             gmm_models{spk}.sigma(:,ll) = UBM.sigma(:,ll);
%         end
%     end
end
%% Step3: Scoring the verification trials
trial_list =  strcat('lst\lista_test_k1_',int2str(k1),'_k2_',int2str(k2),'.lst');

fid = fopen(trial_list, 'rt');
C = textscan(fid, '%s␣␣␣␣␣%s\n');
fclose(fid);
test_files = C{1};
test_IDs = C{2};

N_ref = length(gmm_models(:, 1));
N_test = length(test_files(:, 1));
```

```
kk = 1;


for(ii=1: N_ref)
    for(jj=1:N_test)
        trials = [ii jj];
         gmmScores(kk) = score_gmm_trials(gmm_models, test_files , trials , UBM);
         answers(kk) = strcmp(model_ids(ii), test_IDs(jj));
         kk = kk + 1;
    end
end


%=================================================================
%% Step4: Computing the EER and plotting the DET curve
%labels = C{3};
labels = answers;
eer = compute_eer(gmmScores, labels , true);
eer
res_name = strcat('Resultados\RES_',num2str(nmix),'gmm','_1UBM_k1_',int2str(k1),'_k2_',int2
save(res_name,'eer','labels','gmmScores')
  % end
%end
%end
delete(isopen)
```

Apéndice F. Implementación i-vector + PLDA

A continuación se presenta cada uno de los códigos en que se implementan los sistemas de VH utilizados en los experimentos de los i-vector. en esta sección se encuentra cada código en dos partes: El entrenamiento y el test.

7.5. i-vector + PLDA características articulatorias - poly

Primero el entrenamiento:

```
clc; clear all; close all;
% TRAINING
% script de simulacion de un sistema de verificacion de hablantes
% concaracteristicas articulatorias, bajo la metodologia
% i-vector


%load features\trainSpeakerData %pol features
load features\POl_xi_xf_features%pol & xi, xf features
%load features\xy_features.mat % xy features --> Ndim=10
load features\speakerID
path(path,'D:\Proyecto_Maestria\Tools\MSR_Identity_Toolkit_v1\code'); % las rutinas de MSR
%% Step0: Opening MATLAB pool
 nworkers = 3;
 nworkers = min(nworkers, feature('NumCores'));
 isopen = parpool(3);


 %% Step1: Training the UBM


nDims = 9; % dimensionality of feature vectors
```

```
%nDims = 10; % dimensionality of feature vectors
nChannels = 30; % Number of channels (sessions) per speaker
nSpeakers = 17;
nChannels_test=6;
for n=1:nChannels_test
    testSpeakerData{1,n} = trainSpeakerData{1,n+24};
    testSpeakerData{2,n} = trainSpeakerData{2,n+24};
    testSpeakerData{3,n} = trainSpeakerData{3,n+24};
    testSpeakerData{4,n} = trainSpeakerData{4,n+24};
    testSpeakerData{5,n} = trainSpeakerData{5,n+24};
    testSpeakerData{6,n} = trainSpeakerData{6,n+24};
    testSpeakerData{7,n} = trainSpeakerData{7,n+24};
    testSpeakerData{8,n} = trainSpeakerData{8,n+24};

    testSpeakerData{9,n} = trainSpeakerData{9,n+24};
    testSpeakerData{10,n} = trainSpeakerData{10,n+24};
    testSpeakerData{11,n} = trainSpeakerData{11,n+24};
    testSpeakerData{12,n} = trainSpeakerData{12,n+24};

    testSpeakerData{13,n} = trainSpeakerData{13,n+24};
    testSpeakerData{14,n} = trainSpeakerData{14,n+24};
    testSpeakerData{15,n} = trainSpeakerData{15,n+24};
    testSpeakerData{16,n} = trainSpeakerData{16,n+24};
    testSpeakerData{17,n} = trainSpeakerData{17,n+24};
```

```
%---Ids
    test_speakerID(1,n) = 1; %Keep track of who this is
    test_speakerID(2,n) = 2; %Keep track of who this is
    test_speakerID(3,n) = 3; %Keep track of who this is
    test_speakerID(4,n) = 4; %Keep track of who this is
    test_speakerID(5,n) = 5; %Keep track of who this is
    test_speakerID(6,n) = 6; %Keep track of who this is
    test_speakerID(7,n) = 7; %Keep track of who this is
    test_speakerID(8,n) = 8; %Keep track of who this is
    test_speakerID(9,n) = 9; %Keep track of who this is
    test_speakerID(10,n) = 10; %Keep track of who this is
    test_speakerID(11,n) = 11; %Keep track of who this is
    test_speakerID(12,n) = 12; %Keep track of who this is
    test_speakerID(13,n) = 13; %Keep track of who this is
    test_speakerID(14,n) = 14; %Keep track of who this is
    test_speakerID(15,n) = 15; %Keep track of who this is
    test_speakerID(16,n) = 16; %Keep track of who this is
    test_speakerID(17,n) = 17; %Keep track of who this is
end
speakerID(13,16:30) = 0;
speakerID(14,21:30) = 0;
speakerID(16,20) = 0;

%k1=1;  k2=2;%---------->k1k2
for k1=1:16
```

```
for k2=(k1+1):17
j=1;
    if ((k1==13)|(k1==14)|(k2==13)|(k2==14))==1
        continue
    else
j=1;



 for n =1:17
    if ((n==k1)|(n==k2))==0
      trainSpeakerData_K(j,:) = trainSpeakerData(n,:);
      speakerID_K(j,:) = speakerID(n,:);
      j = j+1;
    end
 end


testSpeakerData_K(1,:)= testSpeakerData(k1,:);
testSpeakerData_K(2,:)= testSpeakerData(k2,:);
test_speakerID_K(1,:) = test_speakerID(k1,:); %Keep track of who this is
test_speakerID_K(2,:) = test_speakerID(k2,:); %Keep track of who this is


test_speaker_filename = strcat('Experimento_ivec\test_speakers\','ts_k1_',int2str(k1),'_k2_
save(test_speaker_filename,'testSpeakerData_K','test_speakerID_K')
%% 2.1 Entrenaando el UBM
%nombre de modelos y config del UBM
```

```
nmix         = 16;  tv_dim =16;
model_folder = 'Experimento_ivec\models\';
models_name = strcat(model_folder,'ivec_UBM',num2str(nmix),'gmm',int2str(k1),'_k2_',int2str

final_niter = 10;
ds_factor   = 1;

A = trainSpeakerData_K(:);
A = A(~cellfun('isempty',A));
spk_labs = speakerID_K(:);
spk_labs(spk_labs == 0) = [];

ubm = gmm_em(A, nmix, final_niter, ds_factor, nworkers);
save(models_name,'ubm')
%==================================================================
%Step 2.2 :  Calculate the statistics needed for the i-Vector model

niter  = 5;
stats = cell(length(A), 1);
%parfor index_file = 1 : length(feat_Files)
for index_file = 1:length(A)
    [N, F] = compute_bw_stats(A{index_file}, ubm);
    stats{index_file} = [N; F];
end
```

```
%========================================================================
% Step 2.3: Learning the total variability subspace
T = train_tv_space(stats, ubm, tv_dim, niter, nworkers);
save(strcat(model_folder,'T_k1_',int2str(k1),'_k2_',int2str(k2),'.mat'),'T')

% Step 2.4: Now compute the ivectors, LDA and GPLDA
dev_ivs = zeros(tv_dim, length(A));
tic
%parfor index_file = 1 : length(feat_Files),
for index_file = 1 : length(A),
    dev_ivs(:, index_file) = extract_ivector(stats{index_file}, ubm, T);
end
t1=toc;

V = lda_MSR(dev_ivs, spk_labs);
save(strcat(model_folder,'V_k1_',int2str(k1),'_k2_',int2str(k2),'.mat'),'V')

spk_names = unique(spk_labs);     nSpeakers = length(spk_names);
lda_dim = min(200, nSpeakers-1);
lda_dim = min([lda_dim, nSpeakers-1,size(dev_ivs,1)]);
dev_ivs = V(:, 1 : lda_dim)' * dev_ivs;
nphi    = lda_dim;        niter   = 10;
PLDA = gplda_em(dev_ivs, spk_labs, nphi, niter);

save(strcat(model_folder,'PLDA_k1_',int2str(k1),'_k2_',int2str(k2),'.mat'),'PLDA')
```

```
lda_dim

    end

end

end

delete(isopen)
```

Y ahora el script de test:

```
clc; clear all; close all;

% TEST

% script de simulacion de un sistema de verificacion de hablantes

% concaracteristicas articulatorias, bajo la metodologia

% i-vector

%load features\trainSpeakerData %pol features

load features\POl_xi_xf_features%pol & xi, xf features

%load features\xy_features.mat % xy features --> Ndim=10

load features\speakerID

path(path,'D:\Proyecto_Maestria\Tools\MSR_Identity_Toolkit_v1\code'); % las rutinas de MSR

nmix         = 16;   tv_dim =16;

%% Step0: loading test speaker data

%--> for k1 k2

%k1=1; k2=16;

for k1=1:16

for k2=(k1+1):17

j=1;
```

```
    if ((k1==13)|(k1==14)|(k2==13)|(k2==14))==1
        continue
    else
lda_dim = 14;
model_folder = 'Experimento_ivec\models\';
models_name = strcat(model_folder,'ivec_UBM',num2str(nmix),'gmm',int2str(k1),'_k2_',int2str
test_speaker_filename = strcat('Experimento_ivec\test_speakers\','ts_k1_',int2str(k1),'_k2_


load(models_name) % load UBM model
load(strcat(model_folder,'T_k1_',int2str(k1),'_k2_',int2str(k2),'.mat')) % load T matrix
load(test_speaker_filename)% load test data
%%%-> aqui voy arreglar el problema de la 16; que no tiene archivo (20)
enrol_speakerID_k1 = speakerID(k1,1:24);
enrol_speakerID_k2 = speakerID(k2,1:24);
model_files_k1 = trainSpeakerData(k1, 1:24);
model_files_k2 = trainSpeakerData(k2, 1:24);

if k1 == 16
    model_files_k1 = model_files_k1(~cellfun('isempty',model_files_k1));
    enrol_speakerID_k1(20) = 0;
   % enrol_speakerID_k1(enrol_speakerID_k1 == 0) = [];
    model_IDs = unique([enrol_speakerID_k1;speakerID(k2,1:24)],'stable');
elseif k2 == 16
    model_files_k2 = model_files_k2(~cellfun('isempty',model_files_k2));
```

```
    enrol_speakerID_k2(20) = 0;
 %   enrol_speakerID_k2(enrol_speakerID_k2 == 0) = [];
    model_IDs = unique([speakerID(k1,1:24);enrol_speakerID_k2],'stable');
else
    model_files = [trainSpeakerData(k1, 1:24);trainSpeakerData(k2, 1:24)];
    enrol_speakerID = [speakerID(k1,1:24);speakerID(k2,1:24)];
    model_IDs = unique([speakerID(k1,1:24);speakerID(k2,1:24)],'stable');
end

model_IDs(model_IDs == 0) = [];
model_files = [model_files_k1(:);model_files_k2(:)];
enrol_speakerID = [enrol_speakerID_k1(:);enrol_speakerID_k2(:)];
enrol_speakerID(enrol_speakerID == 0) =[];

N_spks = length(model_IDs);
model_ivs1 = zeros(tv_dim, N_spks);
model_ivs2 = model_ivs1;

for spk = 1 : N_spks
    ids = find(ismember(enrol_speakerID, model_IDs(spk)));
    spk_files = model_files(ids);    % los archivos para el spk hablante
    N = 0; F = 0;
%    length(spk_files)
    for ix = 1 : length(spk_files),
        [n, f] = compute_bw_stats(spk_files{ix}, ubm);
```

```
        N = N + n; F = f + F;
        model_ivs1(:, spk) = model_ivs1(:, spk) + extract_ivector([n; f], ubm, T);
    end
    model_ivs2(:, spk) = extract_ivector([N; F]/length(spk_files), ubm, T); % stats averag
    model_ivs1(:, spk) = model_ivs1(:, spk)/length(spk_files); % i-vector averaging!
end

%====================================
% test

test_files = testSpeakerData_K(:);
test_IDs = test_speakerID_K(:);

test_ivs = zeros(tv_dim, length(test_files));
for tst = 1 : length(test_files),
    [N, F] = compute_bw_stats(test_files{tst}, ubm);
    test_ivs(:, tst) = extract_ivector([N; F], ubm, T);
end

%save('ivs\enroll_data.mat','model_ivs1','model_ivs2', 'model_files','model_IDs')
%save('ivs\test_data.mat','test_ivs', 'test_IDs','test_files')

% reduce the dimensionality with LDA
load(strcat(model_folder,'V_k1_',int2str(k1),'_k2_',int2str(k2),'.mat'))
```

```
model_ivs1 = V(:, 1 : lda_dim)' * model_ivs1;
model_ivs2 = V(:, 1 : lda_dim)' * model_ivs2;
test_ivs = V(:, 1 : lda_dim)' * test_ivs;

% Step5: Now score the models with all the test data.

N_ref = length(model_ivs1(1, :));
N_test = length(test_ivs(1, :));
load(strcat(model_folder,'PLDA_k1_',int2str(k1),'_k2_',int2str(k2),'.mat'))

kk = 1;
for(ii=1: N_ref)
   for(jj=1:N_test)
      ivScores_1(kk) = score_gplda_trials(PLDA, model_ivs1(:, ii), test_ivs(:, jj));
      ivScores_2(kk) = score_gplda_trials(PLDA, model_ivs2(:, ii), test_ivs(:, jj));
      answers(kk) = strcmp(int2str(model_IDs(ii)), int2str(test_IDs(jj)));
      kk = kk + 1;
   end
end

figure(1);
eer_iv1 = compute_eer(ivScores_1, answers, true);
eer_iv1

figure(2);
```

```
eer_iv2 = compute_eer(ivScores_2, answers, true);
eer_iv2
clear model_files enrol_speakerID model_IDs testSpeakerData_K  test_speakerID_K T ubm V PLD
%%--> saving the results
res_folder = 'Experimento_ivec\res\';
res_name = strcat(res_folder,'RES_',num2str(nmix),'gmm','_1UBM_k1_',int2str(k1),'_k2_',int2
save(res_name,'eer_iv1','eer_iv2','answers','ivScores_1','ivScores_2')
    end
end
end
```

7.6. i-vector + PLDA Fusión de características MFCC + poly

Primero el entrenamiento:

```
clc
clear all

path(path,'D:\Proyecto_Maestria\Tools\MSR_Identity_Toolkit_v1\code'); % las rutinas de MSR

%% Step0: Opening MATLAB pool
nworkers = 3;
nworkers = min(nworkers, feature('NumCores'));
isopen = parpool(3);
%if ~isopen, parpool(nworkers); end

% ========================================================
```

```
%% Step1: Training the UBM
dataList_folder = 'C:\Users\salpub83fdhb2\Dropbox\Mis␣Documentos␣Maestria\MATLAB\Proyecto\
 for k1=1:16
 for k2=(k1+1):17
%k1=1; k2=3;
j=1;
    if ((k1==14)|(k1==15)|(k2==14)|(k2==15))==1
        continue
    else
%k1=1; k2=15;
list_all = strcat('lista_UBM_k1_',int2str(k1),'_k2_',int2str(k2),'.lst');
dataList_all = {strcat(dataList_folder, list_all)};
%%%%%% -> nombres modelos


nmix         = 16; tv_dim =16;
model_folder = 'models\i_vec\';
models_name = strcat(model_folder,'ivec_UBM',num2str(nmix),'gmm_k1_',int2str(k1),'_k2_',in


dataList = dataList_all;
%nmix         = nmix;
final_niter = 10;
ds_factor    = 1;
UBM = gmm_em(dataList{1}, nmix, final_niter, ds_factor, nworkers);
save(models_name,'UBM')
```

```
%=================================================================
% Step 2.1 : Calculate the statistics needed for the i-Vector model.
%tv_dim = tv_dim;
niter  = 5;
fid = fopen(dataList{1});     % usa los mismos archivos del UBM.
C = textscan(fid, '%s \n');
fclose(fid);
feat_Files = C{1};      % cada file es C{1}{n_file}
stats = cell(length(feat_Files), 1);
%parfor index_file = 1 : length(feat_Files),
  for index_file = 1:length(feat_Files)
     [N, F] = compute_bw_stats(feat_Files{index_file}, UBM);
     stats{index_file} = [N; F];
  end

% Step 2.2: Learning the total variability subspace
T = train_tv_space(stats, UBM, tv_dim, niter, nworkers);
save(strcat(model_folder,'T_k1_',int2str(k1),'_k2_',int2str(k2),'.mat'),'T')

% Step 2.3: Now compute the ivectors, LDA and GPLDA
fid = fopen(strcat('lst\lista_UBM_IDs_k1_',int2str(k1),'_k2_',int2str(k2),'.lst'));
C = textscan(fid, '%s      %s\n');
fclose(fid);
feat_Files = C{1};
dev_ivs = zeros(tv_dim, length(feat_Files));
```

```
tic
%parfor index_file = 1 : length(feat_Files),
for index_file = 1 : length(feat_Files),
    dev_ivs(:, index_file) = extract_ivector(stats{index_file}, UBM, T);
end
t1=toc;

% reduce the dimensionality with LDA
spk_labs = C{2};
%save('ivs\dev_data_500_256GMM.mat','dev_ivs', 'spk_labs','feat_Files')

V = lda_MSR(dev_ivs, spk_labs);
save(strcat(model_folder,'V_k1_',int2str(k1),'_k2_',int2str(k2),'.mat'),'V')

spk_names = unique(spk_labs);     nSpeakers = length(spk_names);
lda_dim = min(200, nSpeakers-1);
lda_dim = min([lda_dim, nSpeakers-1,size(dev_ivs,1)]);
dev_ivs = V(:, 1 : lda_dim)' * dev_ivs;
nphi    = lda_dim;        niter   = 10;
PLDA = gplda_em(dev_ivs, spk_labs, nphi, niter);

%save PLDA PLDA
save(strcat(model_folder,'PLDA_k1_',int2str(k1),'_k2_',int2str(k2),'.mat'),'PLDA')

lda_dim
```

```
    end
end
end
delete(isopen)
```

Y ahora el script de test:

```
clear all
clc

path(path,'D:\Proyecto_Maestria\Tools\MSR_Identity_Toolkit_v1\code'); % las rutinas de MSR

%%%%%%%%%%
for k1=1:16
for k2=(k1+1):17
%k1=1; k2=3;
j=1;
    if ((k1==14)|(k1==15)|(k2==14)|(k2==15))==1
        continue
    else
enroll_list_folder = 'C:\Users\salpub83fdhb2\Dropbox\Mis␣Documentos␣Maestria\MATLAB\Proyec
enroll_list_all =strcat('lista_enroll_k1_',int2str(k1),'_k2_',int2str(k2),'.lst');
enroll_list = {strcat(enroll_list_folder, enroll_list_all)};

%%%parametros
nmix         = 16; tv_dim = 16;
```

```
model_folder = 'models\i_vec\';
models_name = strcat(model_folder,'ivec_UBM',num2str(nmix),'gmm_k1_',int2str(k1),'_k2_',in

tv_dim = tv_dim;
nSpeakers = 17;
lda_dim = 14;

lda_dim = min([lda_dim, nSpeakers-1,tv_dim]);
fid = fopen(enroll_list{1});
C = textscan(fid, '%s      %s\n');
fclose(fid)

load(strcat(model_folder,'ivec_UBM',num2str(nmix),'gmm_k1_',int2str(k1),'_k2_',int2str(k2)
load(strcat(model_folder,'T_k1_',int2str(k1),'_k2_',int2str(k2),'.mat'))

model_IDs = unique(C{2}, 'stable');    % el nombre los speakers estan en segunda columna.
model_files = C{1};
N_spks = length(model_IDs);
model_ivs1 = zeros(tv_dim, N_spks);
model_ivs2 = model_ivs1;

for spk = 1 : N_spks
    ids = find(ismember(C{2}, model_IDs{spk}));
    spk_files = model_files(ids);    % los archivos para el spk hablante
```

```
    N = 0; F = 0;
%    length(spk_files)
    for ix = 1 : length(spk_files),
        [n, f] = compute_bw_stats(spk_files{ix}, UBM);
        N = N + n; F = f + F;
        model_ivs1(:, spk) = model_ivs1(:, spk) + extract_ivector([n; f], UBM, T);
    end
    model_ivs2(:, spk) = extract_ivector([N; F]/length(spk_files), UBM, T); % stats averag
    model_ivs1(:, spk) = model_ivs1(:, spk)/length(spk_files); % i-vector averaging!
end

%====================================
%ahora sigue la parte de test.
trial_list =  strcat('lst\lista_test_k1_',int2str(k1),'_k2_',int2str(k2),'.lst');

fid = fopen(trial_list, 'rt');
C = textscan(fid, '%s %s\n');
fclose(fid);
test_files = C{1};
test_IDs = C{2};

test_ivs = zeros(tv_dim, length(test_files));
for tst = 1 : length(test_files),
    [N, F] = compute_bw_stats(test_files{tst}, UBM);
    test_ivs(:, tst) = extract_ivector([N; F], UBM, T);
```

```
end

%save('ivs\enroll_data.mat','model_ivs1','model_ivs2', 'model_files','model_IDs')
%save('ivs\test_data.mat','test_ivs', 'test_IDs','test_files')

% reduce the dimensionality with LDA

load(strcat(model_folder,'V_k1_',int2str(k1),'_k2_',int2str(k2),'.mat'))

model_ivs1 = V(:, 1 : lda_dim)' * model_ivs1;
model_ivs2 = V(:, 1 : lda_dim)' * model_ivs2;
test_ivs = V(:, 1 : lda_dim)' * test_ivs;
% Step5: Now score the models with all the test data.

N_ref = length(model_ivs1(1, :));
N_test = length(test_ivs(1, :));
kk = 1;
load(strcat(model_folder,'PLDA_k1_',int2str(k1),'_k2_',int2str(k2),'.mat'))

for(ii=1: N_ref)
   for(jj=1:N_test)
      ivScores_1(kk) = score_gplda_trials(PLDA, model_ivs1(:, ii), test_ivs(:, jj));
      ivScores_2(kk) = score_gplda_trials(PLDA, model_ivs2(:, ii), test_ivs(:, jj));
      answers(kk) = strcmp(model_IDs(ii), test_IDs(jj));
      kk = kk + 1;
```

```
    end
end

figure(1);
eer_iv1 = compute_eer(ivScores_1, answers, true);
eer_iv1

figure(2);
eer_iv2 = compute_eer(ivScores_2, answers, true);
eer_iv2
clear model_files enrol_speakerID model_IDs testSpeakerData_K  test_speakerID_K T ubm V PLD
%%--> saving the results
res_folder = 'res_ivec\';
res_name = strcat(res_folder,'RES_',num2str(nmix),'gmm','_1UBM_k1_',int2str(k1),'_k2_',int2
save(res_name,'eer_iv1','eer_iv2','answers','ivScores_1','ivScores_2')
    end
end
end
```

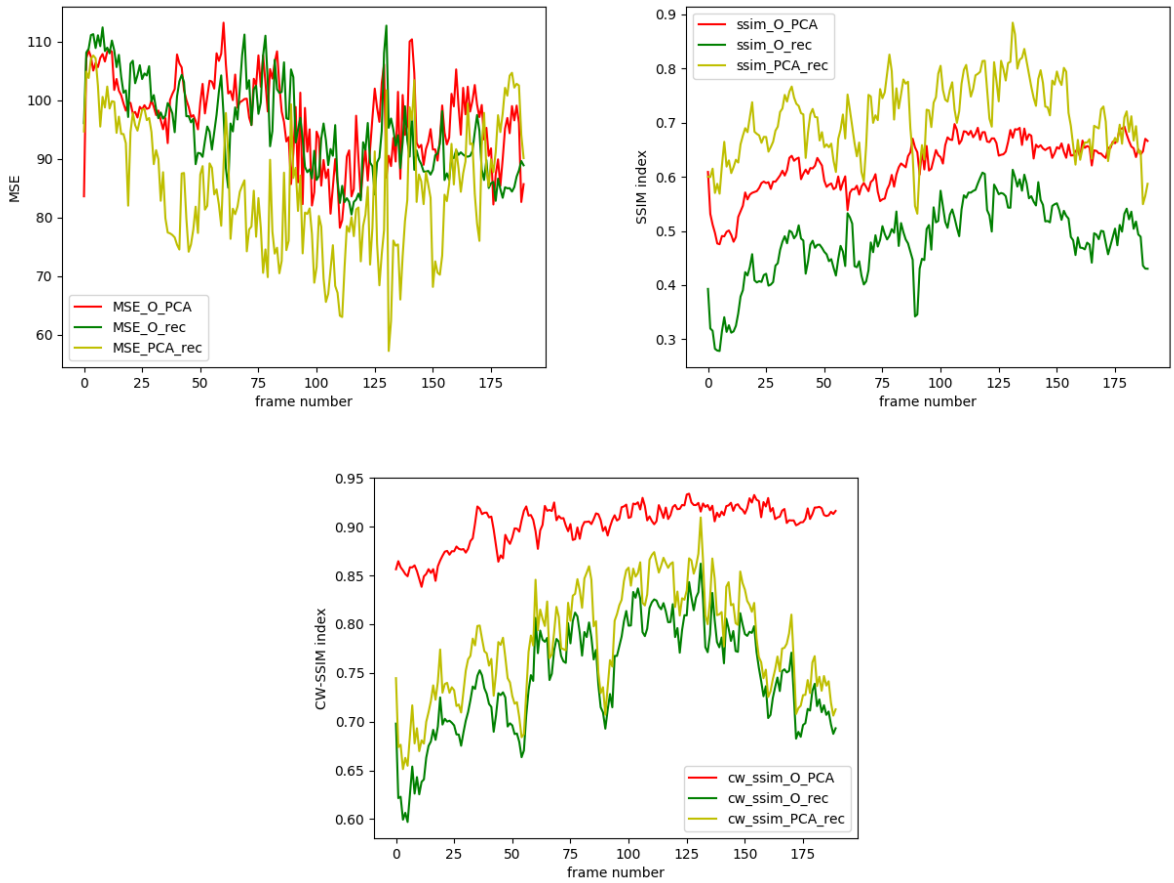


Figura 16. La calidad de la imagen medida en una frase de prueba. En la leyenda, los términos antes del símbolo '_' indican la imagen utilizada como la verdadera. El término después del símbolo representa la imagen que estamos comparando. por ejemplo, O_rec se refiere a que la imagen original (verdadera) es comparada con la imagen reconstruida por la DNN; Para ser claro, el término PCA es la imagen original pero en una versión reconstruida de la representación con PCA o del espacio EigenTongue (Verdadera).

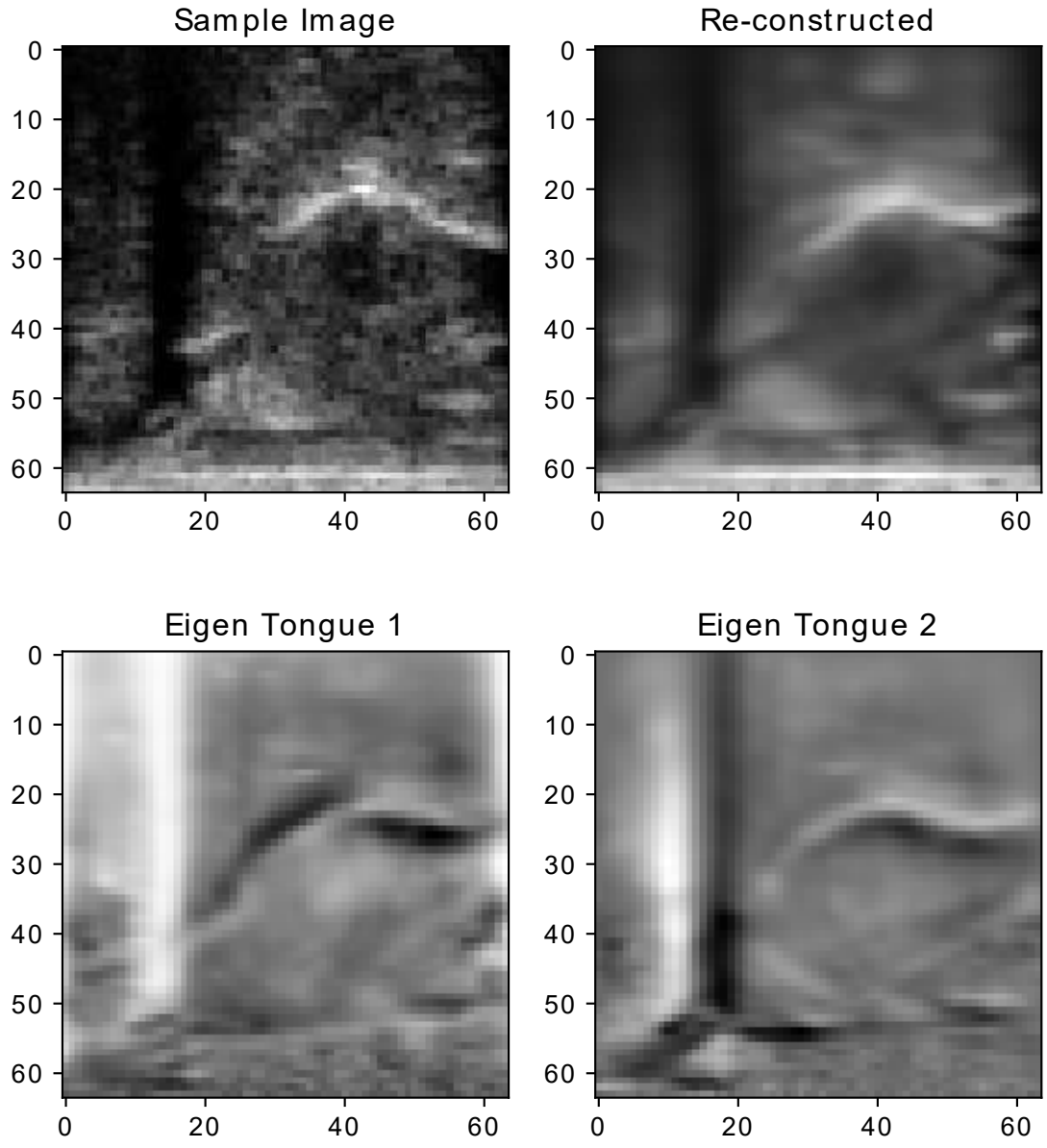


Figura 17. En la parte superior derecha se encuentra una imagen de ultrasonido a partir de la información acústica. La imagen original está en la parte superior izquierda. También se muestran Et1 y Et2 en el fondo