# Estudio de la definición de visema para un sistema de reconocimiento visual automático del habla

Fernández López, Adriana

Curs 2014-2015

Director: Federico Sukno

GRAU EN ENGINYERIA DE SISTEMES AUDIOVISUALS

Universitat Pompeu Fabra Barcelona
Escola Superior Politècnica

Treball de Fi de Grau

# Estudio de la definición de visema para un sistema de reconocimiento visual automático del habla

Adriana Fernández López

TRABAJO FIN DE GRADO

GRADO EN INGENIERÍA DE SISTEMAS AUDIOVISUALES

ESCUELA SUPERIOR POLITÉCNICA UPF

2015

DIRECTOR DEL TRABAJO

Federico Sukno, Cognitive Media Technologies Research Group

Para ver esta película, debe disponer de QuickTime™ y de un descompresor .

*Dedicado a los dos grandes pilares que me han apoyado durante el transcurso de mi carrera: mi familia y mis amigos.*

## Agradecimientos

Me gustaría expresar mi agradecimiento a Federico Sukno por su implicación en el proyecto y por haberme guiado y aconsejado en todo momento.

## Resumen

Las personas nos comunicamos principalmente mediante el habla, utilizando los recursos auditivos y visuales para interpretarla. El reconocimiento automático del habla se centra principalmente en interpretar las señales de audio, pero en condiciones desfavorables se ha demostrado que el vídeo puede compensar estas condiciones aportando la información oculta en el audio. El estudio del reconocimiento visual automático del habla es todavía un problema abierto en el ámbito científico. Uno de sus problemas principales es la definición de la unidad atómica del vídeo, denominado visema. En este estudio, exploramos diferentes opciones para la definición de un vocabulario de visemas basado en la agrupación de fonemas para la construcción de un sistema de procesado automático visual del habla. El sistema utiliza descriptores SIFT para extraer las características principales de cada fotograma y un modelo oculto de Markov para modelar las restricciones estadísticas de una secuencia de visemas y/o fonemas.

## Resum

Les persones ens comuniquem principalment per mitjà de la parla, utilitzant els recursos auditius i visuals per interpretar-la. El reconeixement automàtic de la parla es basa principalment en interpretar les senyals d'àudio, però en condicions desfavorables s'ha demostrat que el vídeo pot compensar aquestes condicions aportant la informació oculta en l'àudio. L'estudi del reconeixement visual automàtic de la parla es encara un problema obert en l'àmbit científic. Un dels seus problemes principals es la definició de la unitat atòmica del vídeo, l'anomenat visema. En aquest estudi, explorem diferents opcions per a la definició d'un vocabulari de visemas basat en l'agrupació de fonemes per a la construcció d'un sistema de processat automàtic visual de la parla. El sistema fa servir descriptors SIFT per extraure les característiques principals de cada fotograma i un model ocult de Markov per modelar les restriccions estadístiques d'una seqüència de visemas i/o fonemes.

# Índice

Resumen ..... ix

Lista de figuras ..... xiii

Lista de tablas ..... xiv

1. Introducción ..... 1
   1.1. Fonemas y visemas ..... 1

2. TEORÍA ..... 5
   2.1. Algoritmo SIFT (Scale Invariant Feature Transform) ..... 5
      a) Introducción ..... 5
      b) Cálculo del espacio – escala ..... 6
      c) Localización de los keypoints ..... 9
      d) Cálculo de la máxima orientación ..... 9
      e) Cálculo del descriptor SIFT ..... 12
   2.2. Modelos ocultos de Markov (HMM) ..... 13
      a) Introducción ..... 13
      b) Elementos que forman el modelo ..... 14
      c) Las tres grandes cuestiones planteadas en un HMM ..... 14

3. Materiales y herramientas ..... 19

4. Implementación ..... 21
   4.1. Localización de los labios ..... 21
   4.2. Extracción de características ..... 25
   4.3. Clasificador ..... 26
      a) Vocabulario fonético ..... 26
      b) Agrupación ..... 29
         a. Agrupación con los vectores reducidos por PCA ..... 30
         b. Agrupación con los vectores reducidos por PCA y añadiendo aprendizaje supervisado ..... 31
         c. Agrupación con los vectores reducidos por PCA y añadiendo aprendizaje supervisado con LDA ..... 32
      c) Estudio de vocabularios candidatos ..... 35
         a. Primer escenario: ..... 35
         b. Segundo escenario: ..... 37
         c. Tercer escenario: ..... 38

d. Cuarto escenario: ..... 39
e. Quinto escenario: ..... 40
f. Sexto escenario: ..... 41
d) Construcción del modelo oculto de Markov ..... 43

5. Resultados ..... 47
5.1. Elección del vocabulario final ..... 47
5.2. Ejemplo de reconocimiento automático visual del habla ..... 53

6. Conclusiones ..... 55

Bibliografía ..... 61

## Lista de figuras

I. Proceso para evaluar correspondencias entre dos imágenes ..... 6
II Esquema DoG ..... 7
III. Ejemplo imagen $L(w+1) \times (w+1)$ ..... 10
IV. Representación del gradiente..... 10
V Histograma máxima orientación..... 11
VI. Histograma suavizado máxima orientación ..... 11
VII. Representación del gradiente y orientación dominante ..... 12
VIII Gradientes de la imagen y descriptor..... 12
IX. Ejemplo salida del programa Praat..... 19
X. Proceso general de reconocimiento visual automático del habla ..... 21
XI. Polígonos Viola-Jones..... 22
XII. Polígono final..... 22
XIII. Polígono ROI..... 22
XIV. Coordenadas de los polígonos..... 23
XV. Ejemplo ROI con Viola Jones ..... 23
XVI. Ejemplo puntos obtenidos con ASM..... 24
XVII. Puntos usados para el cálculo ROI con ASM ..... 24
XVIII. Ejemplo ROI con ASM ..... 25
XIX. Puntos de los descriptores SIFT ..... 26
XX. Gráfico redundancia máxima..... 31
XXI. Histograma de la cantidad real fonemas..... 33
XXII. Histograma de la cantidad de fonemas según la clasificación basada en LDA ... 34
XXIII. Matriz de confusión normalizada ..... 35
XXIV. Matriz de confusión..... 35
XXV. Estudio escenario 1 ..... 36
XXVI. Estudio escenario 2..... 38
XXVII. Estudio escenario 3..... 39
XXVIII. Estudio escenario 4 ..... 40
XXIX. Estudio escenario 5..... 41
XXX. Estudio escenario 6 ..... 42
XXXI. Esquema HMM ..... 43
XXXII. Probabilidades obtenidas con el vocabulario 1 ..... 48
XXXIII. Probabilidades obtenidas con el vocabulario 2..... 49
XXXIV. Probabilidades obtenidas con el vocabulario 3..... 49
XXXV. Probabilidades obtenidas con el vocabulario 4..... 50
XXXVI. Probabilidades obtenidas con el vocabulario 5..... 51
XXXVII. Probabilidades obtenidas con el vocabulario 6 ..... 51
XXXVIII. Vocabulario de visemas final..... 53
XXXIX. Ejemplo de secuencia de salida del sistema ..... 54

## Lista de tablas

Tabla 1 ..... 8
Tabla 2 ..... 8
Tabla 3 ..... 27
Tabla 4 ..... 27
Tabla 5 ..... 43
Tabla 6 ..... 48
Tabla 7 ..... 48
Tabla 8 ..... 49
Tabla 9 ..... 50
Tabla 10 ..... 50
Tabla 11 ..... 51
Tabla 12 ..... 52
Tabla 13 ..... 52

# 1. Introducción

Los seres humanos nos comunicamos principalmente mediante el habla. La percepción del habla es considerada normalmente como un proceso auditivo. Sin embargo, ha sido demostrado que en la compresión del habla interactúan los sistemas auditivo y visual. La influencia de la visión en la percepción del habla se demostró con el efecto McGurk, en 1976, donde se mostraba al usuario un vídeo que repetía palabras con el fonema /ba/ pero que habían sido grabadas en el vídeo como movimientos de los labios diciendo /ga/, por lo tanto el oyente veía /ga/ y oía /ba/ llegando a la percepción de haber oído el fonema intermedio /da/ (R. Seymour, 2007). De este modo, utilizamos la información visual para entender el habla, especialmente si las condiciones del ambiente son desfavorables, por ejemplo debido al ruido de vehículos, otras personas hablando, etc. Además la interpretación visual del habla es muy importante, especialmente para personas con discapacidad auditiva, ya que éstas utilizan el movimiento de la boca en el lenguaje de signos y en la comunicación entre sordos (G. Potamianos, 2003; E. Antonakos, 2015).

El reconocimiento automático del habla (ASR) utiliza las señales de audio para interpretar la información recibida. Los sistemas actuales de ASR no son robustos si el audio se encuentra corrupto debido a ruidos ambientales, distorsión producida en el canal de transmisión, etc. Se ha demostrado, que la combinación de audio y vídeo permite realizar un mejor reconocimiento del habla que empleando cualquiera de las modalidades por separado (R. Seymour, 2007; W.C. Yau, 2007, Z. Zhou, 2014). Este proceso es conocido como reconocimiento automático audiovisual del habla (AV-ASR). En el AV-ASR se analizan inicialmente ambos canales por separado, por un lado el análisis del audio y por otro el del vídeo, y luego se combinan para obtener mejores resultados que usando únicamente uno de ellos. El audio aporta la mayor parte de la información y de manera más precisa, pero el vídeo la complementa aportando la parte visual que no está presente en la señal de audio. Una de las ventajas del reconocimiento visual del habla es que no se encuentra afectado por el ruido acústico, y además nos aporta información como la localización del hablante, el lugar de articulación y la visibilidad de articuladores como la lengua, los dientes y los labios. La forma de articular puede ayudar a eliminar ambigüedades entre consonantes como “p y k” o “b y d” difíciles de distinguir únicamente con la señal de audio. (G. Potamianos, 2003).

El ASR necesita, para interpretar el audio, la mínima unidad que permita diferenciar una palabra de otra (el fonema). Del mismo modo, para interpretar el video necesitamos la unidad mínima que nos permita interpretarlo de la mejor manera posible (el visema).

## 1.1. Fonemas y visemas

Un fonema es la mínima articulación de un sonido, ya sea vocálico o consonántico, que nos permite distinguir una palabra de otra en un mismo idioma, por ejemplo, en español consideramos los fonemas [p] y [b] distintos, ya que nos permiten diferenciar entre palabras como *peca* y *beca*. (A.C. Gimson, 2008)[1]. Análogamente, la unidad atómica del vídeo es el visema (W.C. Yau, 2007). A diferencia de los fonemas, claramente definidos en un vocabulario y muy consolidados, no se conoce una definición universal de los visemas. Existen discrepancias entre si están relacionados con la posición de la

[1] http://es.wikipedia.org/wiki/Fonema

boca o con el movimiento de ésta. Por este motivo, los investigadores han llegado actualmente a definir diferentes alfabetos de visemas con tamaños de entre 11 y 15 visemas diferentes (P.S Aleksic, 2002). La definición de visemas más usada parte de que éstos son derivados de la agrupación de fonemas teniendo la misma apariencia visual. El número de visemas es inferior al número de fonemas ya que varios fonemas se observan igual. (L. Cappelletta, 2011).

En este documento nos centraremos en el reconocimiento visual automático, concretamente definiremos un vocabulario de visemas basado en fonemas, que nos permita estudiar si es posible, utilizando únicamente el vídeo, descifrar que está diciendo el usuario.

Un sistema de reconocimiento visual tiene los siguientes pasos principales: 1) localizar los labios, 2) utilizar técnicas de extracción de características en los labios y zonas circundantes, y 3) clasificar e interpretar las características obtenidas para leer los labios. (V. Sahu, 2013; R. Seymour, 2007)

Hay diferentes maneras de localizar los labios. Por ejemplo, podemos localizar la cara utilizando un algoritmo de detección facial, como por ejemplo Viola Jones (P. Viola, 2004), y después refinar la detección de la cara para ajustarla únicamente al entorno de la boca. Otra manera es utilizando un modelo basado en forma, como el Modelo Activo de Forma (ASM) (T.F. Cootes, 2004), que nos proporciona puntos característicos de la cara y los labios en cada fotograma para poder así encuadrar los labios más precisamente.

Una vez tenemos la región de los labios localizada tenemos que elegir una técnica de extracción de características. Las técnicas de extracción de características se clasifican en basadas en forma, basadas en píxel (o intensidad) y basadas en movimiento (G. Potamianos, 2003; W.C. Yau, 2007). Las técnicas basadas en forma inicialmente utilizaban el ancho y el alto de la boca, pero actualmente utilizan marcas artificiales o puntos característicos para obtener el contorno de los labios, tal y como se hace, por ejemplo, en ASM. Técnicas basadas en píxeles consideran que el valor del píxel alrededor del área de la boca proporciona información relevante. Las técnicas más utilizadas basadas en intensidad son el análisis de componentes principales (PCA), la transformada discreta del coseno (DCT) y el patrón de flujo óptico (L. Cappelletta, 2011). Las características basadas en intensidad o en forma se extraen de fotogramas estáticos directamente y se pueden ver como características estáticas. Las técnicas basadas en segmentación de movimiento utilizan directamente la dinámica del habla. En este caso, los movimientos pueden ser representados en plantillas espacio-temporales 2D (STT, Spatial-Temporal Templates). Para extraer las características de las STT se utilizan técnicas que combinan la transformada Wavelet discreta estacionaria (SWT) y movimientos Zernike. (W.C. Yau, 2007).

En nuestro caso, hemos decidido extraer características utilizando un sistema basado en intensidad, concretamente utilizando descriptores SIFT (D.G. Lowe, 2005) en cada fotograma, es decir, descriptores que extraen las características más relevantes de la imagen. Los descriptores SIFT describen los puntos indicados y su entorno, por ese motivo, fijando una serie de puntos podemos describir toda el área de los labios. Finalmente, obtenemos un vector formado por descriptores concatenados que describen un fotograma.

Debido a que la dimensionalidad del vector resultante es muy alta se aplica el algoritmo PCA, que nos permite seleccionar las dimensiones más importantes. (R. Seymour, 2007).
El siguiente paso del proceso es el clasificador. En este trabajo hemos decidido utilizar una combinación de Análisis Discriminante Lineal (LDA) y modelos ocultos de Markov como clasificadores, ya que nos permiten diseñar un modelo capaz de clasificar los descriptores SIFT como visemas teniendo en cuenta también sus posibles transiciones.

Un aspecto destacable de este estudio es que nos permitirá verificar si aplicando nuevos métodos de descripción de características se pueden lograr mejores resultados en el reconocimiento visual del habla aportando así mayor información al reconocimiento automático audiovisual.

# 2. TEORÍA

En este apartado explicamos los dos pilares teóricos estudiados en profundidad en este trabajo. Primeramente el algoritmo SIFT que nos permitirá mediante sus descriptores extraer las características de nuestra región de interés; y finalmente, los modelos ocultos de Markov que nos permitirán obtener las secuencias de fonemas más probables pronunciadas en los vídeos.

## 2.1. Algoritmo SIFT (Scale Invariant Feature Transform)

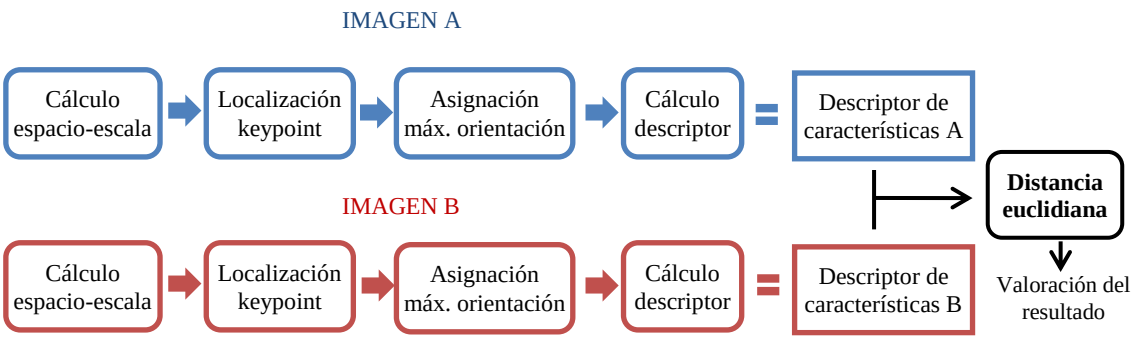
### a) Introducción

Actualmente, la visión por ordenador se encuentra con muchos problemas, uno de ellos es poder describir una imagen con el detalle suficiente para poder encontrar correspondencias con otras imágenes. Las correspondencias son necesarias para el reconocimiento y seguimiento de objetos en diferentes escenas.
Este algoritmo describe las características de la imagen que tienen propiedades adecuadas para establecer correspondencias entre diferentes imágenes de un objeto o escena. Las características son robustas a escala y rotación de la imagen, transformaciones afines, cambios de punto de vista 3D, aparición de ruido y cambios de iluminación.
Las características son suficientemente distintivas, lo que significa que una característica puede encontrar una buena correspondencia con una alta probabilidad sobre una gran base de datos de características de muchas imágenes.
Este algoritmo, en el caso del reconocimiento visual del habla, nos permitirá describir los diferentes fotogramas con vectores de características que serán la base para formar un vocabulario de visemas.
El algoritmo que genera el conjunto de características de la imagen se conoce cómo SIFT (Scale Invariant Feature Tranform) (Lowe, 2004). A continuación, podemos ver los pasos necesarios para su cálculo:

- Algoritmo SIFT:

1. Cálculo del espacio-escala
2. Localización de los Keypoints
3. Asignación de la máxima orientación
4. Cálculo del descriptor del keypoint

A continuación, en la siguiente figura, se muestra el proceso realizado para evaluar las correspondencias entre dos imágenes A y B.
En la práctica, para obtener correspondencias entre imágenes, se calculan las características SIFT de imágenes extraídas de un conjunto de referencia y se almacenan en una base de datos. Seguidamente, se busca la correspondencia con una nueva imagen comparando las características de ésta con las de las imágenes de la base de datos, basándonos en la distancia euclidiana de los vectores de características para evaluar su similitud.




**I. Proceso para evaluar correspondencias entre dos imágenes**

## b) Cálculo del espacio – escala

Utilizamos una cascada de filtros que emplee algoritmos eficientes en la detección de localizaciones candidatas de puntos significativos (keypoints), tal y como se detallará seguidamente.

Mediante el espacio-escala conseguimos representar el mismo keypoint en imágenes de diferentes tamaños, logrando así la invariancia a cambios de escala.
El espacio-escala de una imagen se define como la convolución entre una gaussiana $G$ y la imagen original $I$. La variable sigma corresponde al factor de desenfoque que aplica la gaussiana.

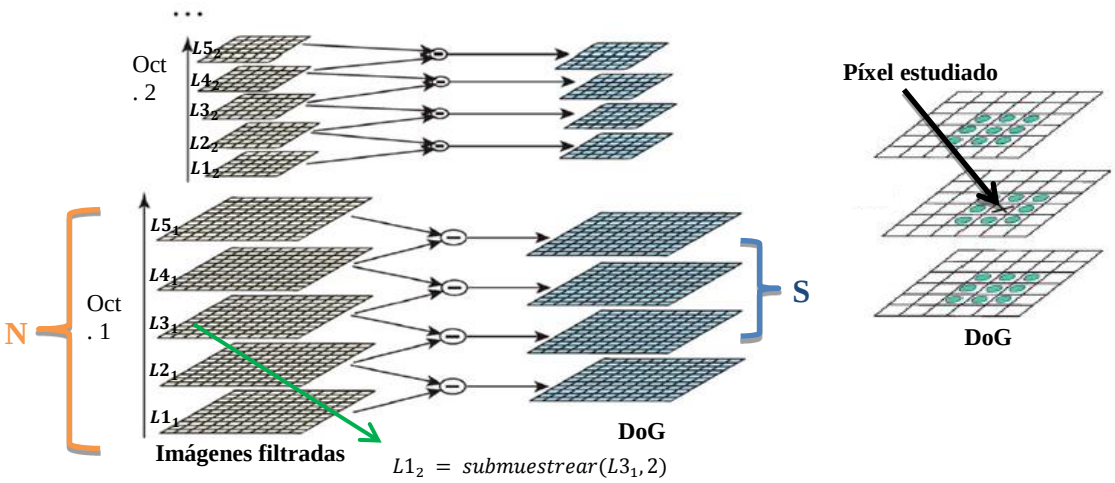
$$L(x, y, \sigma) = G(x, y, \sigma) * I(x, y)$$

Los keypoints no se calculan directamente de las imágenes filtradas $L$. Esto se debe a que necesitamos una función que permita detectar fácilmente los puntos más estables. Los máximos y mínimos de la función Laplaciana de gaussianas proporcionan las características más estables de la imagen, es decir los keypoints, por eso nos interesa utilizar un espacio basado en esta función. La función Laplaciana de gaussianas es computacionalmente compleja, por este motivo se aproximan sus resultados mediante la función diferencia de gaussianas (DoG) que tiene muy poco coste computacional. La DoG se puede calcular como la diferencia de dos escalas vecinas separadas por un factor multiplicativo $k$.

$$D(x, y, \sigma) = \big(G(x, y, k\sigma) - G(x, y, \sigma)\big) * I(x, y) = L(x, y, k\sigma) - L(x, y, \sigma)$$

El espacio de diferencia de gaussianas se construye dividiendo cada octava en un número entero de intervalos $S$ . Cada imagen de la octava debe estar separada de la siguiente por factor constante $k = 2^{1/s}$.

En la siguiente figura, podemos observar que para la detección de keypoints son necesarias 3 imágenes DoG. La relación entre el número de intervalos de la diferencia de gaussianas y el número de imágenes filtradas es $NL = S + 3$ . Para poder detectar los máximos y mínimos locales en la $D(x, y, \sigma)$ cada píxel de la imagen es comparado con los ocho vecinos más cercanos de su imagen DoG y con los nueve vecinos más cercanos de las imágenes DoG superior e inferior, tal y como podemos ver en la figura siguiente. Un píxel solo se considera un extremo si es mayor a todos sus vecinos o menor a todos ellos.




**II Esquema DoG**

La relación entre una octava y la siguiente es un factor 2:1, esto provoca que una vez filtrada una octava completa se utiliza la imagen filtrada que tenga un valor sigma igual al doble del sigma inicial como imagen inicial de la siguiente octava. Antes de ello submuestreamos esta imagen por un factor dos evitando aliasing, ya que ya ha sido filtrada, y logrando una mejor eficiencia computacional. En la práctica, la sigma correspondiente al doble del tamaño inicial será aquella por debajo de las dos últimas imágenes de la octava.

Seguidamente se explican los cálculos para obtener las imágenes filtradas por gaussianas, basados en la implementación de Andrea Vedaldi (A. Vedaldi, 2007). Tal y cómo se ha explicado, cada imagen filtrada por la gaussiana tiene un valor de suavizado sigma. Una manera de ahorrar coste computacional es filtrar una imagen que ya ha sido filtrada, y evitar tener que filtrar siempre desde la imagen original $I$. Por ese motivo nos interesan los valores de la diferencia de sigmas (*dsigma*) que aplicaremos a cada imagen filtrada para obtener la siguiente.

La coordenada de escala $\sigma$, indica el factor de desenfoque de la gaussiana, se relaciona con las octavas $o$ y los diferentes niveles de la octava $s$ mediante la siguiente ecuación:

$$\sigma(s, o) = \sigma_0 2^{o+s/S} \quad o \in \mathrm{Z}, \quad s = s_{min}, .., s_{max}$$

En esta implementación se consideran los parámetros que utiliza David Lowe (D.G. Lowe, 2004):

$$\sigma_n = 0.5\,; \quad \sigma_0 = 1.6 \cdot 2^{1/S} = 1.6\,k$$

Asumimos que la imagen de entrada $I$ tiene aplicado el suavizado nominal $\sigma_n$, por lo tanto en la primera imagen de la primera octava calculamos la distancia euclidiana entre la sigma nominal y la sigma requerida (final) para conocer el factor de suavizado *dsigma* que debemos aplicar:

$$dsigma_{\,inicial} = \sqrt{(\,\sigma_o \cdot 2^{o_{min}+s/S}\,)^2 - \sigma_n{}^2} \rightarrow$$

$$dsigma_{\,inicial} = \sqrt{(\,\sigma_o \cdot 2^{\,s/S}\,)^2 - \left({}^{\sigma_n}/_{2^{o_{min}}}\right)^2}$$

De este modo corregimos el error de pre-escalado que se puede haber producido en la primera imagen causado por $2^{o_{min}}$.

Una vez calculada la primera imagen de la primera octava las siguientes se calculan aplicando la distancia euclidiana entre la sigma actual y la sigma anterior:

$$dsigma \;=\; \sqrt{(\,\sigma_o \cdot 2^{o_{min}+s/S}\,)^2 - (\,\sigma_o \cdot 2^{o_{min}+(s-1)/S}\,)^2} \;=\; \sigma_o \cdot k^s \sqrt{1 - \left(\frac{1}{k}\right)^2}$$

En la tabla 1 se muestra un ejemplo de la diferencia de sigmas calculadas para 7 octavas con 3 intervalos y 6 imágenes filtradas y en la tabla 2 se muestra un ejemplo de las sigmas reales que se deberían aplicar directamente a la imagen $I$ en el cálculo de cada imagen gaussiana, llamaremos a estos valores *Betas*.

En este caso, la primera imagen de cada octava se corresponde con la cuarta imagen de la octava anterior (Número de intervalos $-$ 2). Si observamos el valor beta 1 en la segunda tabla vemos como se corresponde con el valor real aplicado a la cuarta imagen de la octava anterior (beta 4). Las dsigma correspondientes a la primera escala de las octavas 2-7 tienen valor cero porque sus imágenes son iguales a la cuarta imagen de la octava anterior que se encuentra desenfocada por un valor beta 4, por lo que no tenemos que aplicarle ningún desenfoque más.

**Tabla 1**

| | | **Octavas** | | | | | | |
|---|---|---|---|---|---|---|---|---|
| | | oct. 1 | oct. 2 | oct. 3 | oct. 4 | oct. 5 | oct. 6 | oct. 7 |
| ***Dsigmas*** | $d\sigma_1$ | 1.2490 | 0 | 0 | 0 | 0 | 0 | 0 |
| | $d\sigma_2$ | 1.2263 | 1.2263 | 1.2263 | 1.2263 | 1.2263 | 1.2263 | 1.2263 |
| | $d\sigma_3$ | 1.5450 | 1.5450 | 1.5450 | 1.5450 | 1.5450 | 1.5450 | 1.5450 |
| | $d\sigma_4$ | 1.9466 | 1.9466 | 1.9466 | 1.9466 | 1.9466 | 1.9466 | 1.9466 |
| | $d\sigma_5$ | 2.4525 | 2.4525 | 2.4525 | 2.4525 | 2.4525 | 2.4525 | 2.4525 |
| | $\sigma d_6$ | 3.0900 | 3.0900 | 3.0900 | 3.0900 | 3.0900 | 3.0900 | 3.0900 |

**Tabla 2**

| | | **Octavas** | | | | | | |
|---|---|---|---|---|---|---|---|---|
| | | oct. 1 | oct. 2 | oct. 3 | oct. 4 | oct. 5 | oct. 6 | oct. 7 |
| **Betas** | $\beta\sigma_1$ | 1.2490 | 3.0397 | 6.3214 | 12.7609 | 25.5805 | 51.1902 | 102.3951 |
| | $\beta\sigma_2$ | 1.7503 | 3.9058 | 8.0012 | 16.0960 | 32.2385 | 64.5002 | 129.0120 |
| | $\beta\sigma_3$ | 2.3346 | 4.9803 | 10.1100 | 20.2841 | 40.6252 | 81.2688 | 162.5468 |
| | $\beta\sigma_4$ | 3.0397 | 6.3214 | 12.7609 | 25.5805 | 51.1902 | 102.3951 | 204.7976 |
| | $\beta\sigma_5$ | 3.9058 | 8.0012 | 16.0960 | 32.2385 | 64.5002 | 129.0120 | 258.0299 |
| | $\beta d_6$ | 4.9803 | 10.1100 | 20.2841 | 40.6252 | 81.2688 | 162.5468 | 325.0982 |

En las tablas podemos comprobar como el hecho de calcular las imágenes filtradas partiendo de las anteriores nos facilita en gran medida los cálculos computacionalmente. Cabe destacar que el valor Beta de la cuarta imagen de cada octava es la sigma inicial de la siguiente octava, tal y como se había expuesto anteriormente.

Una vez hemos calculado las imágenes gaussianas debemos calcular el espacio de diferencia de gaussianas tal y como ya se ha explicado.

## c) Localización de los keypoints

Los puntos considerados extremos en la DoG son considerados keypoints candidatos. El siguiente paso es llevar a cabo un ajuste detallado de los datos cercanos a la ubicación, escala. Esto permite rechazar puntos con bajo contraste, es decir puntos sensibles al ruido, o puntos mal localizados a lo largo del borde.
En el proceso de localización de los keypoints se les asigna una localización y una escala o sigma específicas.

## d) Cálculo de la máxima orientación

Mediante la asignación de una orientación dominante a cada keypoint, basada en las propiedades locales de la imagen, el descriptor SIFT puede lograr invarianza a rotaciones si es representado teniendo en cuenta esta orientación.

La escala asignada a cada keypoint se utiliza para seleccionar qué imagen $L$ del espacio-escala se va a utilizar para calcular la máxima orientación del keypoint. Concretamente, siguiendo la implementación de Andrea Vedaldi, se utiliza el redondeo de la escala del keypoint para elegir la imagen que más se ajuste a su escala.

Nos centramos en estudiar el keypoint localmente, por lo tanto estudiamos los píxeles de la imagen seleccionada cercanos al keypoint. Una manera de limitar los píxeles estudiados es aplicando una ventana centrada en el keypoint.

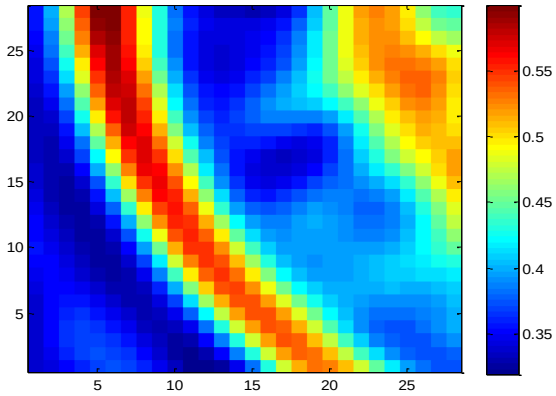
El cálculo del gradiente resulta una herramienta útil porque permite utilizar los datos de la dirección y el ritmo de máxima variación en cada punto para construir un histograma que localice la orientación dominante. Utilizamos la diferencia entre píxeles para calcular la magnitud y la orientación del gradiente, tal y como observamos a continuación:

$$M(x,y) = \sqrt{\begin{array}{l}\left(L_{(w+1)\times(w+1)}(x+1,y) - L_{(w+1)\times(w+1)}(x-1,y)\right)^2 \\ + \left(L_{(w+1)\times(w+1)}(x,y+1) - L_{(w+1)\times(w+1)}(x,y-1)\right)^2\end{array}}$$

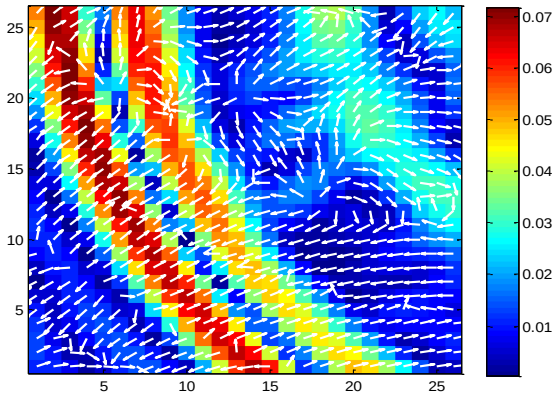
$$\theta(x,y) = tan^{-1}\frac{L_{(w+1)\times(w+1)}(x,y+1) - L_{(w+1)\times(w+1)}(x,y-1)}{L_{(w+1)\times(w+1)}(x+1,y) - L_{(w+1)\times(w+1)}(x-1,y)}$$

La matriz $M_{w\times w}$ almacena los módulos del gradiente de cada keypoint y la matriz $\theta_{w\times w}$ almacena las orientaciones de cada keypoint. La matriz $L_{(w+1)\times(w+1)}$ corresponde al segmento de la imagen $L$ centrado en el keypoint y tiene tamaño $(w+1)\times(w+1)$.

En las siguientes figuras observamos, a la izquierda una imagen $L_{(w+1)\times(w+1)}$ y a la derecha la representación del gradiente de una imagen $L_{(w+1)\times(w+1)}$. La orientación de cada píxel se representa por flechas blancas y la magnitud del gradiente de cada píxel se representa mediante un degradado de colores.




**III. Ejemplo imagen** $L_{(w+1)\times(w+1)}$

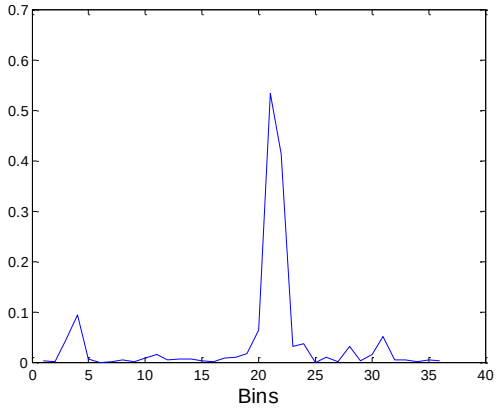



**IV. Representación del gradiente**

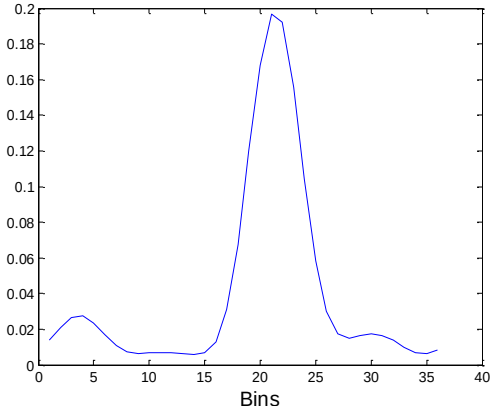
Los valores del módulo y la orientación del gradiente calculados en el vecindario de cada keypoint se utilizarán para formar un histograma que permita obtener su máxima orientación.

Siguiendo el documento de David Lowe (D.G. Lowe, 2004), se utilizan histogramas ponderados por el producto del módulo del gradiente y una gaussiana; con una desviación de 1.5 veces el valor de la escala del keypoint, que da más importancia a los píxeles cercanos al keypoint. Los histogramas representan los 360 grados en 36 bins de 10 grados cada uno. Siguiendo la implementación de Andrea Vedaldi, tras colocar la información en los bins, antes de calcular la máxima orientación, el histograma se suaviza utilizando un filtro de media móvil para reducir ruido.

El filtro de media móvil se aplica un número de veces determinado (típicamente 5) a cada valor del histograma, consiguiendo robustez al ruido. El bin intermedio es el primero en suavizarse y a continuación se suavizan los anteriores hasta suavizarlos todos. La nueva magnitud del bin viene dada por la media entre los bins superior, actual y posterior. En las siguientes figuras podemos observar el histograma antes y después de aplicar el filtro de media móvil. En la imagen de la derecha podemos observar cómo se han suavizado todos los picos de la imagen de la derecha, tal y como esperábamos.




**V Histograma máxima orientación.**




**VI. Histograma suavizado máxima orientación**

La orientación dominante del keypoint corresponde al pico más alto del histograma. Cualquier pico superior al 80% del pico dominante también se considera máxima orientación local y se crea un nuevo keypoint con esta nueva orientación y las mismas coordenadas $x$ e $y$.

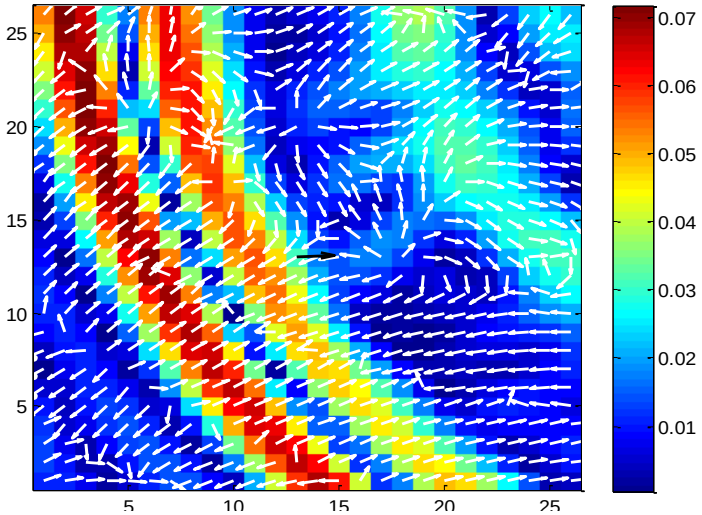
Finalmente, para precisar la máxima orientación se realiza una interpolación entre el bin detectado como máximo y el valor de los bins anterior y posterior. El valor del desplazamiento $d_{bin}$ calculado se suma al valor del bin inicial obteniendo así la orientación dominante del keypoint, tal y cómo observamos a continuación:

$$Orientación_{máx} = \frac{2\pi \cdot (bin + d_{bin} + 0.5)}{NBins}$$

$$d_{bin} = -\frac{1}{2}\frac{hp - hm}{hp + hm - 2h0}$$

El máximo desplazamiento $d_{bin}$ del intervalo se produce cuando uno de los valores $hp$ (valor posterior) o $hm$ (valor anterior) es igual a $h0$ (valor actual) y el otro es nulo: en el caso de $hp = h0$ y $hm = 0$, el desplazamiento del intervalo sería + 0.5 y en el caso de $hm = h0$ y $hp = 0$ sería de 0.5. Tratándose de un histograma los valores $hp$ y $hm$ no pueden ser negativos, además nunca pueden ser mayores que $h0$, ya que en ese caso, $h0$ ya no sería una orientación dominante.

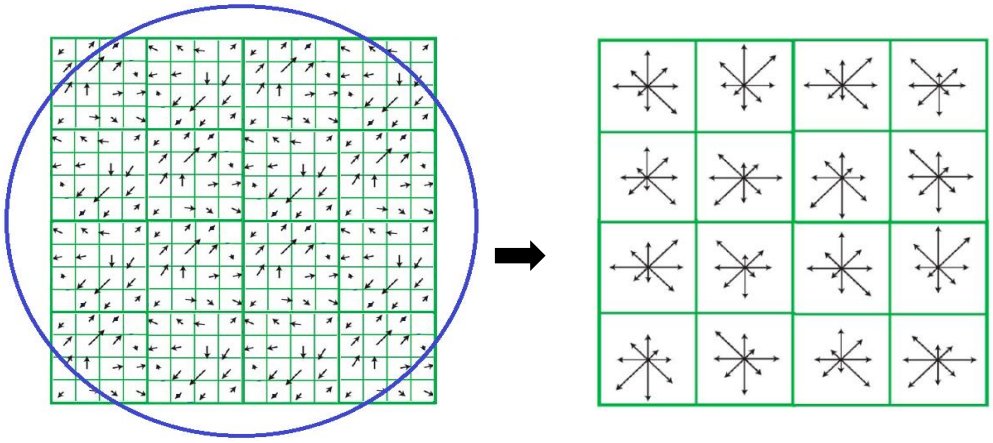
En la siguiente figura se puede observar la representación del gradiente tal y como se ha visto anteriormente, y además se puede observar mediante una flecha negra la representación de la orientación dominante del keypoint.




**VII. Representación del gradiente y orientación dominante**

## e) Cálculo del descriptor SIFT

Una vez tenemos la localización, la escala y la orientación dominante de cada keypoint, ya podemos calcular el descriptor. El descriptor está formado por histogramas de orientaciones de 8 bins de regiones de 4x4 píxeles cercanos al keypoint. En la siguiente figura, se puede ver un ejemplo.



**VIII Gradientes de la imagen y descriptor**

Inicialmente, utilizamos el redondeo de la escala del keypoint para saber qué imagen *L* de la octava debemos utilizar para calcular el modulo y la orientación del gradiente de todos los píxeles alrededor del keypoint.

El filtro gaussiano que pondera el histograma debe tener una desviación igual a la mitad del tamaño de la ventana del descriptor. Este filtro permite evitar cambios repentinos en el descriptor debido a cambios pequeños en la posición de la ventana, además de dar más importancia a los píxeles cercanos al keypoint.
Para lograr invariancia a rotaciones, para cada píxel calculamos la diferencia entre su ángulo y la máxima orientación del keypoint.

Para distribuir el valor de cada muestra del gradiente en los bins del histograma se utiliza interpolación trilineal. Los histogramas resultantes son concatenados formando un descriptor de 4x4x8 = 128 elementos.

Finalmente, el vector de características es modificado para reducir los efectos de cambios de iluminación. Primero el vector es normalizado a la unidad para evitar así cambios de contraste, ya que si multiplicamos el valor de un pixel por una constante, esta se encontrara multiplicada también en el gradiente, y se cancelará al realizar la normalización. Un cambio de luminosidad en el cual una constante es añadida a cada pixel no afectara a los valores del gradiente, ya que estos se calculan utilizando la diferencia de píxeles. Por lo tanto, el descriptor es invariante a cambios de iluminación afines. Se pueden producir también cambios de iluminación no lineales que pueden causar grandes cambios en las magnitudes relativas para algunos gradientes, pero no suele afectar a las orientaciones del gradiente. Para evitar la influencia de estas grandes magnitudes, se eliminan todos los valores del descriptor superiores a 0.2 y luego se renormaliza, obteniendo así el descriptor SIFT definitivo.

## 2.2. Modelos ocultos de Markov (HMM)

### a) Introducción

Los procesos del mundo real producen resultados que se pueden caracterizar como señales, por ejemplo música, caracteres de un alfabeto finito, muestras de voz, etc. Un problema interesante que estudian y han estudiado continuamente los investigadores es cómo caracterizar estas señales reales con modelos. Los modelos son útiles porque nos proporcionan las bases teóricas que describen un sistema de procesado de señal, mediante el cual podemos simular y aprender cómo actúa la fuente que provoca la señal. Los modelos de señales son realmente funcionales ya que trabajan muy bien en la práctica permitiéndonos realizar sistemas de predicción, de reconocimiento, de identificación, etc. Se conocen diferentes tipos de modelos: los modelos deterministas, que utilizan propiedades conocidas de la señal, por ejemplo, si la señal es un seno se determinan la amplitud, la frecuencia y la fase; y los modelos estadísticos, que tratan de caracterizar las propiedades estadísticas de la señal, usando, por ejemplo, procesos Gaussianos, procesos ocultos de Markov, etc. (V.A. Petrushin, 2000; R.L. Rabiner, 1989).

Este documento se basa en la aplicación de un tipo de modelo estocástico conocido como modelo oculto de Markov (HMM). En él se modela un proceso de Markov de parámetros desconocidos. Un proceso de Markov es un proceso estocástico sin memoria, es decir la probabilidad condicional sobre el estado presente, futuro y pasado del sistema es independiente. Un HMM estima los parámetros desconocimos mediante los parámetros observables. En un modelo oculto de Markov, el estado no es visible, sino que se observan los símbolos influidos por el estado, de manera que cada estado tiene una distribución de probabilidad sobre los posibles símbolos de salida.

Aplicaremos modelos ocultos de Markov a problemas de reconocimiento del habla. A continuación, explicaremos como funciona un HMM y cómo llegamos a resolver problemas de reconocimiento del habla.

## b) Elementos que forman el modelo

En un modelo oculto de Markov se observa una secuencia de observaciones O = {$O_1$, $O_2$,…, $O_T$} producida en un instante de tiempo t = 1, 2,…, T y que ha sido influenciada por una serie de estados ocultos Q = {$S_1$, $S_2$,…, $S_N$}. El objetivo de nuestro modelo es descubrir cuál es el estado que se oculta tras cada una de las observaciones de la secuencia.

Los elementos necesarios para construir un modelo son:

- N, número de estados del modelo.
- M, número de símbolos observables por los estados.
- V, vocabulario de símbolos observables, V = {$v_1$, $v_2$,… , $v_M$}
- A, distribución de probabilidad de transición de un estado a otro.

$$A = \{a_{ij}\},\ \ a_{ij} = P(q_{t+1} = S_j | q_t = S_i),\ \ 1 \leq i,j \leq N$$

*donde $q_t$ es el estado actual en el que se encuentra el sistema.

- B, distribución de probabilidad de observar un símbolo en el estado *j*.

$$B = \{b_j(k)\},\ \ b_j(k) = P(v_k\ en\ t \mid q_t = S_j),\ \ 1 \leq j \leq N;\ 1 \leq k \leq M$$

- π, distribución de probabilidad inicial de cada estado.

$$\pi = \{\pi_i\},\ \ \pi_i = P(q_1 = S_i),\ \ 1 \leq i \leq N$$

Los elementos que definen un modelo de Markov se agrupan en: $\lambda = (A, B, \pi)$.

## c) Las tres grandes cuestiones planteadas en un HMM

A la hora de resolver un HMM nos surgen tres problemas o cuestiones:

1. ¿Cómo de bien se corresponde el modelo con la secuencia?
2. Dada la secuencia de observaciones, ¿Cómo elegimos una secuencia de estados óptima, es decir, la que mejor describa las respectivas observaciones?
3. ¿Cómo ajustamos λ para describir de forma óptima la secuencia de observaciones?

<u>Solución al primer problema:</u>

Para conocer cómo de bien se corresponde el modelo con la secuencia tenemos que calcular la probabilidad de la secuencia de observaciones O emitida por el modelo λ.

$$P(O|\lambda) = \sum_{Q=1}^{N} P(O|Q,\lambda) \cdot P(Q|\lambda)$$

Intuitivamente, la manera más sencilla de calcular la probabilidad anterior es sumando todas las posibles secuencias de estados. Esta metodología para resolver el problema implica $(2T-1) \cdot \mathrm{N}^{\mathrm{T}}$ productos y $(N^T - 1)$ sumas. Tal y como podemos observar necesitamos un algoritmo más eficiente ya que éste implica demasiados cálculos. De esta manera surge el algoritmo conocido como *Procedimiento de adelante-atrás.*

El procedimiento de adelante-atrás se basa en una técnica conocida como programación dinámica. La programación dinámica realiza cálculos de pequeñas porciones, almacena estos resultados y los utiliza cuando los necesita, sin la necesidad de volverlos a calcular. De esta manera, debemos encontrar una propiedad recursiva que nos permita realizar nuevos cálculos basados en los resultados actuales. (V.A. Petrushin, 2000).

- Procedimiento de adelante-atrás:

Consideramos $\alpha_t(i)$, la probabilidad de la secuencia parcial de observaciones $O_{1\to t} = \{ o_1, o_2, \dots, o_t\}$, producida por todas las posibles secuencias de estados que acaban en el estado *i*, dado el modelo λ. La probabilidad total de la secuencia parcial de observaciones dado el modelo es la suma de todos los $\alpha_t(i)$ desde i = 1, 2,.., N.

$$\alpha_t(i) = P(\, o_1 o_2 \dots o_t, q_t = i|\lambda)$$

$$P(\, o_1 o_2 \dots o_t|\lambda) = \sum_{i=1}^{N} \alpha_t(i)$$

A continuación podemos ver el algoritmo recursivo para calcular $\alpha_t(i)$ y obtener la probabilidad.

Algoritmo adelante:

1) Inicialización:

$$\alpha_1(i) = \pi_i b_i(O_1), \quad 1 \le i \le N$$

2) Recursión:

$$\alpha_{t+1}(j) = \left[\sum_{i=1}^{N} \alpha_t(i) \cdot a_{ij}\right] b_j(O_{t+1}), \quad 1 \le j \le N, \; 1 \le t \le T-1$$

3) Terminación:

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

Del mismo modo, podemos considerar la variable $\beta_t(i)$ como la probabilidad condicional de la secuencia parcial de observaciones desde $O_{t+1}$ hasta el final $O_{t+1\rightarrow T} = \{ o_{t+1}, o_{t+2}, \dots, o_T\}$, producidas por todas las secuencias de estados que empiezan en el estado $i$. El procedimiento de atrás calcula de forma recursiva la variable $\beta_t(i)$ yendo hacia atrás a lo largo de la secuencia de observaciones.

$$\beta_t(i) \ = P( \ o_{t+1} o_{t+2} \dots o_T, q_t = i|\lambda)$$

$$P( \ o_{t+1} o_{t+2} \dots o_T|\lambda) = \sum_{i=1}^{N} \pi_i b_i(o_{t+1}) \beta_t(i)$$

A continuación podemos ver el algoritmo recursivo para calcular $\beta_t(i)$ y obtener la probabilidad.

Algoritmo atrás:

1) Inicialización:

$$\beta_T(i) = \ 1, \quad 1 \ \leq i \ \leq N$$

2) Recursión:

$$\beta_t(i) = \sum_{j=1}^{N} a_{ij} \, b_j(O_{t+1}) \beta_{t+1}(j) \cdot , \quad 1 \ \leq i \leq N, \ \ t = \ T-1, T-2, \dots, 1$$

3) Terminación:

$$P(O|\lambda) = \sum_{i=1}^{N} \pi_i b_i(o_1) \beta_1(i)$$

<u>Solución al segundo problema:</u>

Para solucionar este problema necesitamos un mecanismo que nos permita encontrar la secuencia de estados óptima que mejor describa la secuencia de observaciones O dado el modelo λ.

Este problema se puede resolver utilizando diferentes criterios de optimización. Por ejemplo, podríamos utilizar un criterio que escogiera el estado más probable en cada instante de tiempo, de este modo se maximizaría el número esperado de estados individuales correctos. El criterio más utilizado por los expertos trata de encontrar el camino que define la mejor secuencia de estados, es decir la que maximice $P(Q, O|\lambda)$. Basado en programación dinámica el algoritmo que se emplea se conoce como *Algoritmo de Viterbi* (V.A. Petrushin, 2000; R.L. Rabiner, 1989).

- Algoritmo de Viterbi:

Dada la secuencia de observaciones O, para encontrar la mejor secuencia de estados Q, necesitamos definir una cantidad:

$$\delta_t(i) = \text{máx}_{q_1,q_2,\dots,q_{t-1}} P\,(q_1 q_2 \dots q_t = i,\ o_1 o_2 \dots o_t | \lambda)$$

Esta cantidad $\delta_t(i)$ es la máxima probabilidad a lo largo de un único camino, en el tiempo *t*, que representa a las primeras *t* observaciones y termina en el estado $q_t$.

Algoritmo:

1) Inicialización:

$$\delta_1(i) = \pi_i b_i(O_1), \quad 1 \le i \le N$$

$$\psi(j) = 0$$

2) Recursión:

$$\delta_{t+1}(j) = max_{1\le i\le N}\left[\delta_t(i) \cdot a_{ij}\right] \cdot b_j(O_t), \quad 1 \le j \le N, \ 1 \le t \le T-1$$

$$\psi_{t+1}(j) = argmax_{1\le i\le N}\left|\delta_t(i) \cdot a_{ij}\right|$$

3) Terminación:

$$p^* = max_{1\le i\le N}[\delta_L(i)]$$

$$q_L^* = argmax_{1\le i\le N}[\delta_L(i)]$$

4) Retroceso:

$$q_t^* = \psi_{t+1}(q_{t+1}^*), \ t = T-1, T-2, \dots, 1$$

<u>Solución al tercer problema:</u>

Para resolver este problema necesitamos un método que ajuste los parámetros del modelo λ = (A, B, π), de manera que maximicen la secuencia de observaciones dado el modelo.

Uno de los algoritmos más utilizados para optimizar los parámetros es un proceso iterativo conocido como *método Baum-Welch*.

- Método Baum-Welch:

1) Inicialización: Escogemos aleatoriamente los parámetros del modelo, $A_{ij}$ y $B_i(t)$.

2) Recursión:

Calculamos $\alpha_i(t)\ y\ \ \beta_i(t), \quad i = 1, 2, \dots, N\ y\ t = 1,2, \dots T$

$$A_{ij} = \frac{1}{P(O|\lambda)} \sum_{t=1}^{T-1} \alpha_i(t) a_{ij} b_j(o_{t+1}) \beta_j(t+1)$$

$$B_i(k) = \frac{1}{P(O|\lambda)} \sum_{\{t|o_t=k\}} \alpha_i(t) \beta_i(t)$$

Estimamos los nuevos parámetros del modelo:

$$a_{ij} = \frac{A_{ij}}{\sum_{t=1}^{T} A_{it}}, \quad b_i(k) = \frac{B_i(k)}{\sum_{t=1}^{M} B_i(t)}\ , \quad k = 1,2, \dots M, \quad j = 1,2, \dots, N$$

Calculamos el logaritmo de $P(O|\lambda)$.

3) Terminación:

El algoritmo acaba cuando la diferencia, entre dos valores consecutivos de la función de probabilidad, es inferior a un umbral, o el número máximo de iteraciones es superado (V.A. Petrushin, 2000).

# 3. Materiales y herramientas

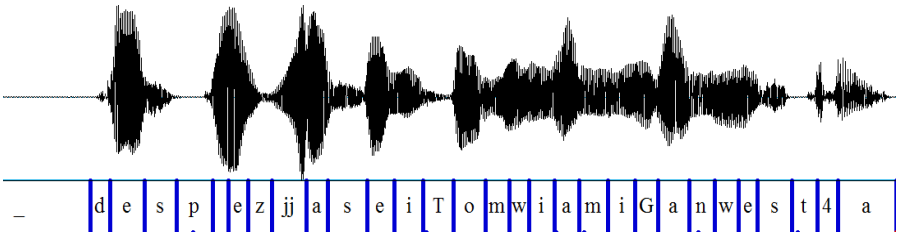
Para el desarrollo de este estudio han sido imprescindibles una serie de materiales y herramientas.

- Base de datos:

Hemos utilizado la base de datos AV@CAR; más precisamente una de las 9 sesiones de grabación de la misma, que contiene vídeos de 19 sujetos diferentes (A. Ortega, 2004). La sesión incluye 25 vídeos por usuario, en los cuales, se dice únicamente una frase. Además se dispone de los archivos de audio y de las transcripciones en ficheros de texto de cada vídeo.

- Praat:

Praat es un programa gratuito que permite describir una señal de audio fonéticamente (P. Boersma, 2015). Únicamente introduciendo la señal de audio, permite escribir en cada instante de tiempo cuál es el fonema que ha sido dicho y después almacenar los datos en un fichero de texto. En el fichero se almacena el símbolo fonético y los instantes inicial y final de éste. A continuación, podemos observar un ejemplo de la salida generada por el programa Praat:




**IX. Ejemplo salida del programa Praat**

Ejemplo: intervalo [3]: xmin = 1.424; xmax = 1.512; texto = "e"

Para agilizar el trabajo Praat dispone de una herramienta llamada Alineamiento sencillo (*EasyAlign)* (J.P. Goldman, 2011). Con esta herramienta, el fichero de audio y el fichero de texto transcrito del audio, podemos obtener el fichero de transcripción fonética automáticamente. Únicamente se han de seguir tres sencillos pasos que se indican en el tutorial del programa. Gracias a este programa, obtenemos la duración de cada fonema durante el vídeo, propiedad que usaremos más adelante para saber qué fonema se ha pronunciado en cada fotograma.

- Matlab:

El software realizado ha sido programado utilizando esta plataforma. [2]

[2] http://es.mathworks.com/products/matlab/

# 4. Implementación

El objetivo de este documento, tal y como ya se ha mencionado, es crear un modelo capaz de reconocer qué se está diciendo en un vídeo utilizando la información visual.

El proceso general que hemos seguido es el siguiente:

**X. Proceso general de reconocimiento visual automático del habla**

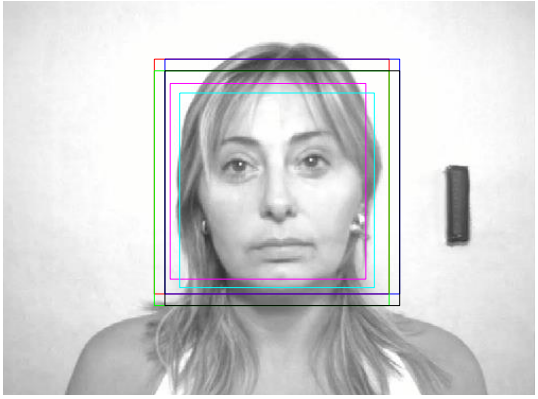
Antes de empezar a trabajar con la base de datos, es necesario separar los datos en dos grupos, el conjunto de entrenamiento y el conjunto de test. Asignamos 20 vídeos de cada uno de los usuarios al conjunto de entrenamiento, es decir, un total de 20 vídeos * 19 usuarios = 380 vídeos de entrenamiento; por el otro lado, asignamos los 5 vídeos restantes de cada usuario al conjunto de test, es decir, 5 vídeos * 19 usuarios = 95 vídeos de test.

A continuación, explicamos el proceso general de reconocimiento visual automático del habla paso por paso.

## 4.1. Localización de los labios

Uno de nuestros objetivos es interpretar qué se ha pronunciado en el vídeo, analizando las características que podamos extraer de él. Inicialmente, debemos detectar la cara del hablante y quedarnos únicamente con la región de la imagen que aporta información relevante. La región que aporta esta información la llamaremos región de interés (ROI). En nuestro caso, la información relevante es aquella correspondiente al habla y nos la proporciona la zona de la boca, por lo tanto ésta es nuestra ROI. Para poder localizar la boca debemos localizar primeramente la cara. Hay varias maneras de localizarla; en este caso hemos estudiado dos maneras distintas: utilizando un software de detección de rostros, el algoritmo conocido como Viola-Jones, y utilizando un programa de seguimiento de forma, concretamente el algoritmo ASM.

Inicialmente, empleamos el software de detección de rostros Viola-Jones (P. Viola, 2004; D. J Kroon, 2010). Este algoritmo es capaz de detectar efectivamente imágenes de rostros frontales a diferentes escalas teniendo en cuenta invariancias a cambios de escala y rotaciones en los ejes *x* e *y* de hasta 45 grados. Viola-Jones nos proporciona las localizaciones x1, x2, y1 e y2 de los polígonos que encuadran la cara en las diferentes escalas, tal y como podemos ver representado en la siguiente figura.



**XI. Polígonos Viola-Jones**

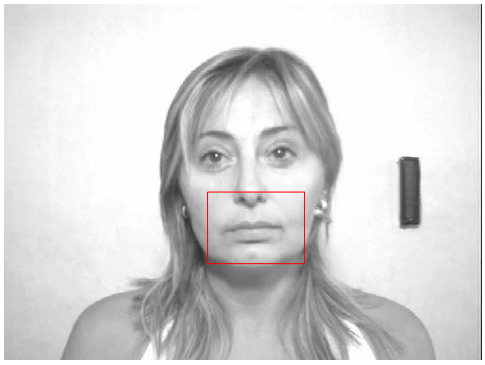
En este momento, tenemos una serie de polígonos que encuadran la cara, pero solamente necesitamos uno para poder ajustar la zona de la boca. Por este motivo, buscamos un criterio que nos permita recuperar, tras aplicar Viola-Jones, un solo polígono que encuadre la cara estudiada. Decidimos realizar la media de todos los polígonos que encuadren la cara para obtener así el polígono final.

Más detalladamente, estudiamos el porcentaje de solapamiento de cada polígono respecto a los demás, es decir, si se intersectan con otros polígonos o están contenidos en otros; seguidamente, aquellos que superen un 85% de solapamiento son seleccionados para calcular el polígono final. Realizamos este estudio para asegurarnos de que solamente cogemos los polígonos que encuadren la cara estudiada, ya que podría aparecer más de una en la imagen. En este ejemplo, todos los polígonos se encuentran suficientemente solapados entre sí, superan el 85% de solapamiento. De manera que realizando la media de las coordenadas de los polígonos podemos obtener de manera sencilla un único polígono que encuadre la cara.

Una vez tenemos los 4 puntos del polígono final que encuadra la cara buscamos un criterio que nos permita encuadrar nuestra ROI para extraer, seguidamente, las características del habla de esta región.

A continuación, en la imagen de la izquierda aparece la detección facial, como resultado de aplicar Viola-Jones y el cálculo del polígono final; y en la imagen de la derecha el encuadre de la boca (ROI) obtenido mediante unos cálculos de proporcionalidad aplicados al polígono de la cara, explicados seguidamente.

**XII. Polígono final**

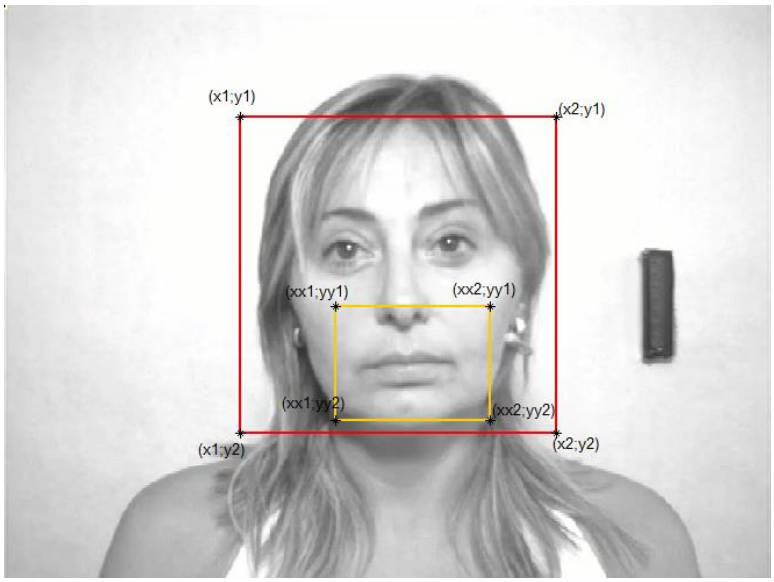


**XIII. Polígono ROI**

Cálculos del polígono de los labios:

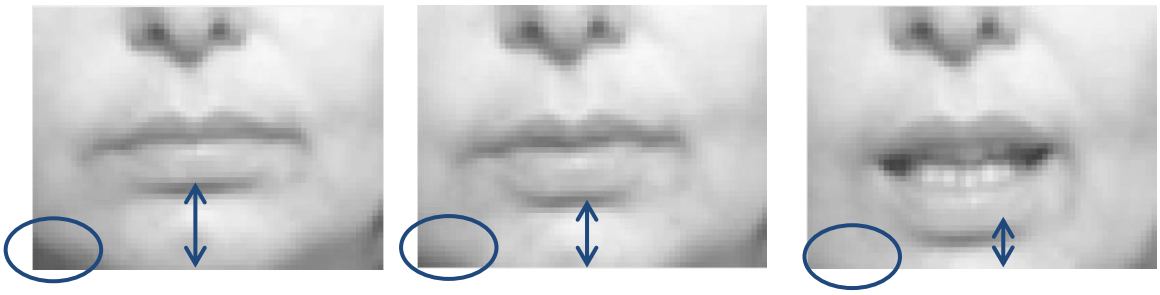
$$xx1 = x1 + \alpha_1 \cdot (x2 - x1) \quad ; \quad yy1 = y1 + \alpha_3 \cdot (y2 - y1)$$
$$xx2 = x2 - \alpha_2 \cdot (x2 - x1) \quad ; \quad yy2 = y2 - \alpha_4 \cdot (y2 - y1)$$

Los factores α se fijan dependiendo de la porción de la cara que queramos encuadrar. En la siguiente imagen podemos observar a qué se corresponden los puntos anteriores.




**XIV. Coordenadas de los polígonos**

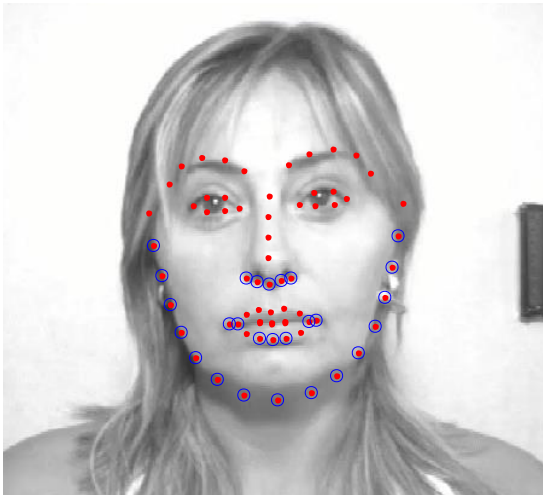
Este mecanismo es poco robusto al movimiento. El detector facial no tiene siempre el mismo tamaño, entonces, dependiendo de la porción seleccionada aparecen los labios más cercanos o más lejanos. Además, no tiene la precisión suficiente para tener en cuenta los movimientos naturales que tiene cualquier usuario al hablar. Si nos fijamos en las siguientes imágenes de la ROI observamos cómo la superficie seleccionada varía considerablemente. Este efecto provoca que descriptores que deberían ser parecidos porque describen la misma posición de la boca no lo sean debido a que no se ha cogido exactamente la misma región de la cara. Por lo tanto, al realizar algunos experimentos preliminares concluimos que este mecanismo no aporta la precisión necesaria para este trabajo.



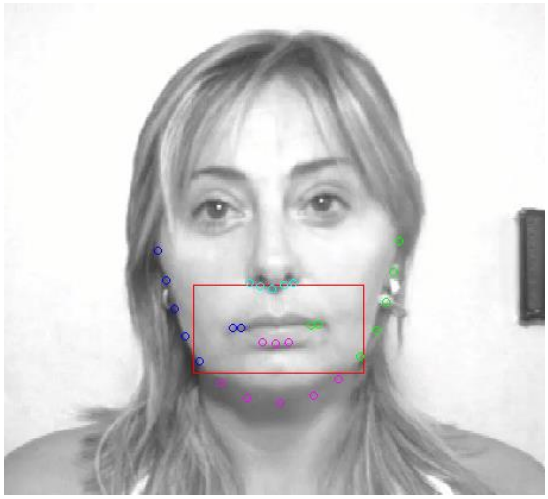
**XV. Ejemplo ROI con Viola Jones**

Debido a este problema buscamos otra manera de localizar la cara y ajustar la zona de los labios. El algoritmo ASM ofrece la posición de los puntos más relevantes en una cara, como por ejemplo el contorno de la cara, de los labios o de los ojos. Decidimos utilizar este algoritmo ya que siempre localiza los mismos puntos de la cara, por ejemplo, siempre encuentra 5 puntos en el contorno de la nariz; y consideramos que este hecho nos permitiría encontrar un mecanismo para fijar el encuadre de los labios con mayor precisión definiendo una región más estable en torno a la boca, y logrando así la robustez al movimiento que estábamos buscando (T.F. Cootes, 2004; F.M. Sukno, 2007).

A continuación, podemos observar el resultado de aplicar ASM a una imagen.



**XVI. Ejemplo puntos obtenidos con ASM**

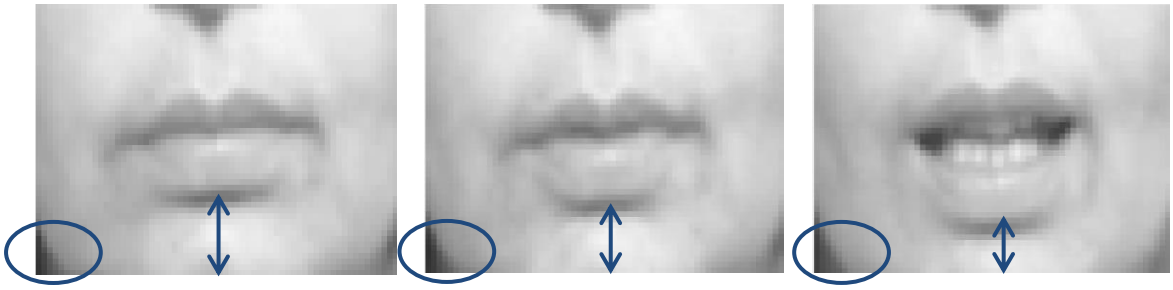


**XVII. Puntos usados para el cálculo ROI con ASM**

En la imagen izquierda podemos observar en color rojo los puntos obtenidos aplicando el algoritmo ASM. Para ajustar el contorno de la boca necesitamos definir un polígono que la encuadre. Se han utilizado los puntos marcados en azul para definirlo, ya que estos puntos se encuentran alrededor de la boca y nos permiten encuadrar su superficie. Por ejemplo, para situar el lateral izquierdo del polígono entre la parte izquierda de la cara y el lateral izquierdo de los labios necesitamos hacer la media entre puntos de las dos posiciones.

Más detalladamente, el polígono obtenido ha sido calculado realizando la media de los puntos seleccionados en azul. Si observamos la imagen de la derecha, las coordenadas x1 y x2 se han calculado haciendo la media de la coordenada *x* de los puntos marcados en azul y verde, en cambio, las coordenadas y1 e y2 han sido calculadas haciendo la media de las coordenadas *y* de los puntos coloreados en cian y magenta respectivamente. El hecho de utilizar mayor cantidad de puntos que los estrictamente necesarios contribuye a la robustez del método frente a posible fallos de alguno(s) de los puntos.

Seguidamente, podemos observar las 3 ROI mostrados anteriormente encuadrados ahora gracias a la ayuda del algoritmo ASM. Si observamos en las esquinas de las tres fotografías vemos como la región seleccionada es prácticamente la misma en las tres. Este mecanismo nos ofrece más robustez ante el movimiento del usuario.



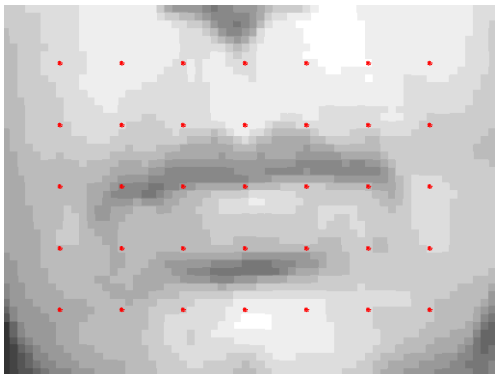
**XVIII. Ejemplo ROI con ASM**

Una vez tenemos la ROI debemos reajustar el tamaño de la imagen para que todas las imágenes contengas el mismo número de pixeles. La resolución tiene que disminuir porque seguidamente describiremos la ROI y cuantos más píxeles tenga la región mayor será la dimensionalidad del descriptor. Por este motivo, tenemos que reducir la resolución lo máximos posible hasta el punto en el que no perdamos detalles necesarios para identificar cuál es la posición de los labios en la imagen. En nuestro caso, tras evaluar visualmente varias opciones, fijamos finalmente la resolución en 48x64 píxeles.

## 4.2. Extracción de características

Una vez hemos localizado la zona de los labios necesitamos describir la ROI con detalle suficiente para después poder compararla con otras ROI y valorar si se trata del mismo visema o no. En nuestro caso, utilizaremos descriptores de características SIFT. El algoritmo SIFT localiza los puntos que considera más relevantes en una imagen, generalmente máximos y mínimos en el espacio-escala, por lo que los puntos descritos varían en cada imagen y no todos los píxeles resultan cubiertos por el descriptor. En nuestro caso, queremos información distribuida de manera uniforme por toda la ROI, para lo cual tendremos que escoger los puntos que serán descritos por el descriptor SIFT externamente, en lugar de hacerlo basándonos en la detección de puntos relevantes.

La resolución elegida para la ROI es de 48x64, valores múltiples de 8, debido a que la distancia entre los puntos elegidos como keypoints ha sido fijada a 8 píxeles. Por lo tanto, calculamos 35 descriptores en cada fotograma, tal y como se indica en rojo en la siguiente imagen. Los descriptores SIFT describen regiones de 16x16 píxeles. Hemos decidido fijar la distancia entre keypoints a 8 para conseguir describir los mismos píxeles de la ROI por varios descriptores. Por ejemplo, los píxeles del centro de la imagen serán descriptos por 4 descriptores distintos, debido al solapamiento de hasta 4 regiones de 16x16 píxeles de descriptores diferentes. Así logramos robustez en la descripción final de la ROI.



**XIX. Puntos de los descriptores SIFT**

La implementación SIFT utilizada es la explicada en el capítulo 2.1, pero adaptada a las condiciones actuales. En este caso, no necesitamos diferentes octavas, ya que los keypoints no son calculados por el algoritmo, sino que los hemos fijado manualmente, tal y como acabamos de explicar.
Los parámetros utilizados en la implementación han demostrado funcionar satisfactoriamente con los valores siguientes:

$$\sigma_0 = 2; \;\; S = 1; \;\; \sigma = 1; \;\; w = 16$$

Una vez tenemos los puntos que actuarán como keypoints de SIFT podemos calcular el descriptor SIFT. Cada descriptor SIFT es un vector de dimensión 128, pero por cada ROI obtenemos 35 descriptores de manera que, si los concatenamos todos, obtenemos un descriptor de la ROI de dimensión 128x35 = 4480.

## 4.3. Clasificador

Una vez realizado el proceso de extracción de características, tenemos que construir el clasificador. El clasificador tratará de clasificar una serie de observaciones, concretamente cadenas de descriptores SIFT, en visemas, basándose en la agrupación de aquellos fonemas que no son distinguibles visualmente (es decir, los fonemas que se pronuncian con la misma forma de los labios y la boca). Por ejemplo, los fonemas “p” y “b” se agruparían en un mismo visema ya que se pronuncian con la misma forma de los labios y la boca.

Inicialmente debemos conocer el vocabulario fonético de la lengua española.

### a) Vocabulario fonético

Para formar el vocabulario fonético nos basamos en el alfabeto fonético SAMPA (J.C. Wells, 1996), ya que corresponde al estándar más utilizado para la notación fonética. El alfabeto SAMPA está formado por los siguientes símbolos:

**Tabla 3**

| Símbolo | Palabra - Transcripción | Símbolo | Palabra - Transcripción |
|---|---|---|---|
| p | padre - **p**aDre | m | mismo – **m**ismo |
| b | vino - **b**ino | n | nunca - **n**unka |
| t | tomo – **t**omo | J | año - a**J**o |
| d | donde – **d**onde | l | lejos - **l**exos |
| k | casa - **K**asa | L | caballo - kaba**L**o |
| g | gata - **g**ata | r | puro – pu**r**o |
| tS | mucho - mu**tS**o | rr | torre - to**rr**e |
| jj | Hielo - **jj**elo | j | rei - rre**j** |
| f | fácil- **f**aTil | w | muy - m**w**i |
| B | cabra - ka**B**ra | i | pico - p**i**ko |
| T | cinco - **T**inko | e | pero - p**e**ro |
| s | sala - **s**ala | a | valle - b**a**Le |
| x | mujer - mu**x**er | o | toro - t**o**ro |
| G | luego - lwe**G**o | u | duro - d**u**ro |
| D | codo - ko**D**o | Gs | examen – e**Gs**amen |
| _ | *silencio* | | |

Por simplicidad en la implementación se han modificado los fonemas “tS” a “C” y “rr” a “R”.

Para poder transcribir una oración a lenguaje fonético tenemos que aplicar una serie de reglas fonéticas, ya que un símbolo del abecedario puede escribirse con distintos símbolos fonéticos, dependiendo de las demás letras de la palabra. (J. Llisterri, 1993).

Seguimos las siguientes reglas:

**Tabla 4**

| Grafema | Reglas de transcripción | Ejemplos |
|---|---|---|
| <a> | **a** | |
| <b> | después de <m> o <n>: **b**<br>otros: **B** | comba – kom**b**a<br>labio - la**B**jo |
| <c> | seguida de <e> o <i>: **T**<br>seguida de <l> o <r>: **G**<br>seguida de <b>, <d>, <m>, <n>, <ñ>, <v>, <g> (precedida de <a>, <o>, <u>) : **G**<br>otros: **k** | celo – **T**elo<br>acné – a**G**né<br>tacto – ta**k**to<br>tecla - te**k**la |
| <ch> | **tS o C** | chelo - **tS**elo |
| <d> | después de <l>, <m> o <n>: **d**<br>otros: **D** | caldo – kal**d**o<br>codo - ko**D**o |
| <e> | **e** | |
| <f> | **f** | |
| <g> | seguida de <r>, <l>, <a>, <o> o <u>: **g**<br>después de <m> o <n> y seguida de <a>, <o> o <u>: **g**<br>seguida de <i> o <e>: **x** | tongo – ton**g**o<br>genio – **x**enjo<br>tigre – ti**G**re<br>lago - la**G**o |

| | | |
|---|---|---|
| | otros: **G** | |
| <h> | en la posición inicial de la palabra seguida de <ie>: **jj**<br>otros: **sin sonido** | hierba – **jj**erBa<br>halo - alo |
| <i> | en posición nuclear en la silaba: **i**<br>en posición no nuclear en la silaba: **j** | tipo – t**i**po<br>cielo - T**j**elo |
| <j> | **x** | |
| <k> | **k** | |
| <l> | **l** | |
| <ll> | **L** | |
| <m> | **m** | |
| <n> | seguida de <p>, <b>, <v>, <m> o <f>: **m**<br>otros: **n** | ánfora – a**m**fora<br>cono - ko**n**o |
| <ñ> | **J** | |
| <o> | **o** | |
| <p> | **p** | |
| <q> | siempre seguida de <u>: **k** | queso - **k**eso |
| <r> | en la posición inicial de la palabra: **rr o R**<br>seguida de <l>, <n> o <s>: **rr o R**<br>otros: **r** | rama – **rr**ama<br>honra – on**rr**a<br>pera - pe**r**a |
| <rr> | **rr** | |
| <s> | **s** | |
| <t> | en la posición final de la silaba: **D**<br>otros: **t** | atleta: a**D**leta<br>toro – **t**oro |
| <u> | sin diéresis procedido por <g> o <q>: **sin sonido**<br>en posición no nuclear en la silaba: **w**<br>en posición nuclear en la silaba: **u** | queso – keso<br>cigüeña: TiG**w**eJa<br>lujo - l**u**xo |
| <v> | después de <m> o <n>: **b**<br>otros: **B** | velo – **b**elo<br>calvo - kal**B**o |
| <w> | después de <m> o <n>: **b**<br>otros: **B** | kiwi - ki**B**i |
| <x> | sin posición inicial y seguido de vocal: **Gs**<br>otros: **s** | examen – e**Gs**amen<br>externo - e**s**terno |
| <y> | en posición inicial de una silaba de dos o más sonidos: **jj**<br>otros: **se procesa como <i>** | yunque – **jj**unke<br>cónyugue: kon**jj**uGe<br>dos i dos - dos **i** dos<br>muy - mu**i** |
| <z> | **T** | zarza - **T**ar**T**a |

Para construir el clasificador necesitamos contar con la información de entrenamiento, en la que sepamos qué fonema se ha pronunciado en cada fotograma.

Inicialmente, construimos una función traductora que nos permita utilizar los ficheros de texto de la traducción de los vídeos, proporcionados por la base de datos, y los

convierta a lenguaje fonético. Seguidamente necesitamos saber cuál es la duración de cada fonema, ayudándonos más adelante, a asignar un fonema a cada fotograma. Para obtener la información que indica la duración de cada fonema utilizamos el programa Praat. Este programa almacena en un fichero de texto qué fonema ha sido pronunciado y su duración, produciendo también el fonema "_" que indica silencio. A continuación, únicamente tenemos que hacer unos cálculos básicos para conocer en qué fotograma se pronuncia cada fonema, tal y como se explica seguidamente.

Como conocemos la duración de cada fonema, podemos calcular los instantes de tiempo inicial y final. También conocemos la cantidad de fotogramas emitidos por segundo que se corresponde con 25 fotogramas por segundo. Entonces, para saber qué fonema se dice en cada fotograma simplemente multiplicamos el número de fotogramas por segundo por el instante inicial y final de cada fonema, y sabremos en qué fotogramas actúa ese fonema.

Fotograma inicial = FS [fotogramas/segundo] * instante inicial [segundos]

Fotograma final = FS [fotogramas/segundo] * instante final [segundos]

Los valores correspondientes a los fotogramas inicial y final calcularlos, tal y como acabamos de explicar, son valores decimales. Esto significa que varios fonemas pueden actuar en un mismo fotograma. Por lo tanto, hemos tenido que definir un criterio para elegir qué fonema asignar a los fotogramas en los cuáles actúen varios fonemas.

Inicialmente, hemos comprobado la duración de los fonemas que actúan en el fotograma, concretamente si la duración de los fonemas era inferior a la duración de un fotograma, es decir inferior a 1/FS = 0.04 segundos. En el caso de que ninguno de los fonemas tuviera una duración inferior a 0.04 segundos se asignaba el fonema de mayor peso al fotograma, es decir, el que actuaba más tiempo en ese fotograma. En cambio, en el caso de que sólo uno de los fonemas fuera inferior a la duración de un fotograma se asignaba ese fonema al fotograma para no perder el fonema pronunciado. Como último caso, si varios fonemas tenían una duración inferior a 0.04 segundos se asignaba el fonema con mayor peso en el fotograma, perdiendo así el fonema con menor peso.
De esta manera, construimos el vector que nos indica para cada fotograma qué fonema se ha pronunciado.

## b) Agrupación

El siguiente paso debería ser agrupar los descriptores SIFT en diferentes grupos formando un vocabulario de visemas. El problema es que la dimensión del vector de cada fotograma es muy grande y no todas las dimensiones tienen la misma importancia. Por este motivo es necesario realizar previamente una extracción de características donde las nuevas características sean una transformación de las originales. Utilizamos análisis de componentes principales (PCA), una técnica que nos permite reducir la dimensionalidad de los datos proyectándolos en una dimensión menor a la original preservando lo máximo posible la varianza en el espacio de alta dimensión. Con tal de reducir la dimensión elegimos el porcentaje de varianza original que queramos que se encuentre representado en los datos. Decidimos representar el 95% de la varianza reduciendo la dimensionalidad de 4480 a 949. Estos vectores de dimensión 1x949 se

utilizan para agrupar los datos y formar el vocabulario de visemas. Para escoger el mecanismo que mejor agrupe nuestros datos hemos explorado tres esquemas distintos que se exponen a continuación.

## a. Agrupación con los vectores reducidos por PCA

Tras obtener los vectores de dimensión reducida necesitamos un algoritmo de agrupamiento. Elegimos el algoritmo *k-means* ya que nos permite agrupar los *n* vectores de 949 dimensiones en *k* grupos, obteniendo un vector de longitud *n* con valores discretos, en el cual cada descriptor pertenece al grupo cuya media se encuentra más cercana. Este algoritmo trata de minimizar la suma de los cuadrados dentro de cada grupo de manera no supervisada.
Para poder utilizar *k-means* es necesario proporcionar un valor *k* que indique el número de grupos en el que el algoritmo debe realizar la partición de los datos. La salida del algoritmo serán las observaciones del modelo oculto de Markov. Tal y como sabemos el número de visemas es inferior al número de fonemas, ya que varios fonemas pueden pronunciarse con una apariencia muy similar de la boca. Por lo tanto el número de grupos *k* tiene que ser inferior a la longitud del vocabulario de fonemas.

Llegados a este punto utilizamos la teoría de la información para fijar un valor que nos permita enviar la mayor cantidad de información útil.

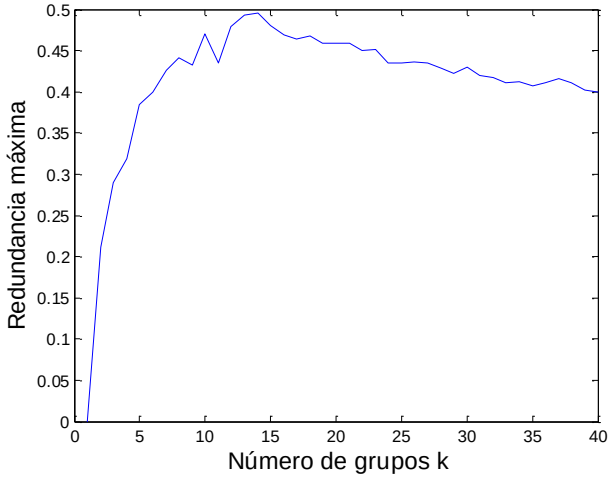
- Calculamos para cada *k* candidata, donde k ∈ [1, Nfonemas]:
    - k-means (k, descriptores SIFT)
    - Entropía de las observaciones, de los estados y conjunta
    - Información mutua
    - Redundancia y redundancia máxima

El valor de la redundancia máxima ha sido el valor más útil, ya que encontramos un valor destacable respecto al resto de medidas que se han comportado de manera aproximadamente monótona. Esta medida obtiene el máximo valor cuando una variable se vuelve completamente redundante con el conocimiento de la otra. A continuación, podemos observar la fórmula matemática del cálculo de la redundancia máxima, en la cual H(X) representa la entropía de las observaciones y H(Y) la entropía de los estados.

$$R_{máx} = \frac{\min(H(X), H(Y))}{H(X) + H(Y)}$$

Para el cálculo de la constante *k* hemos analizado un subconjunto de nuestra base de datos, compuesto por 20 vídeos de los primeros 9 usuarios.

Seguidamente, mostramos el gráfico del cálculo de la máxima redundancia. Podemos observar que obtenemos el máximo valor cuando el número de grupos es 14, por lo que fijaremos la longitud de nuestro vocabulario en 14 visemas. Este valor es consistente con el rango de longitudes utilizados en la literatura, que generalmente oscila entre 11 y 15 visemas (L. Cappelleta, 2011; P.S. Aleksic, 2002; J. Jeffers, 1971; A. Goldschen, 1994)




**XX. Gráfico redundancia máxima**

Tras encontrar el valor *k* que mejor agrupa los datos, pasamos a calcular las matrices de emisión, transmisión y probabilidad inicial del HMM, construyendo así el clasificador que clasifique nuestras observaciones (visemas) en estados (fonemas).

## b. Agrupación con los vectores reducidos por PCA y añadiendo aprendizaje supervisado

Los resultados obtenidos aplicando k-means a los descriptores de dimensionalidad reducida no fueron buenos, debido a que el agrupamiento no resultó informativo en términos de nuestras clases de interés, es decir, descriptores que representaban el mismo fonema no se unían en el mismo grupo y los mismos se agrupaban siguiendo un criterio que no se ajustaba a nuestro objetivo. Por este motivo, antes de aplicar un algoritmo de agrupamiento, añadimos información directa del fonema en los vectores reducidos por PCA, es decir, añadimos una etiqueta ponderada que indicaba a qué fonema pertenecía cada vector, supervisando así el aprendizaje. De esta manera, intentamos que el agrupamiento se guiara no solamente por la extracción de características del vídeo, sino también por el fonema conocido, teniendo en cuenta que éste no pudiera determinar por sí solo su grupo de pertinencia. El objetivo era lograr que fonemas iguales se agruparan en un mismo visema y que fonemas que se observan igual también se agrupasen en ese visema. Para ello, fijamos un parámetro λ que pesaba la actuación del aprendizaje supervisado.

Vector descriptor$_{1x(949+Nfonemas)}$ = [Vector descriptor$_{1x949}$, Vector supervisor$_{1xNfonemas}$];

Vector supervisor$_{1xNfonemas}$ = [0… λ…. 0]; la posición no nula indica el fonema.

λ: constante que pesa la importancia de la etiqueta de fonema.

$$\lambda = \alpha \cdot \sqrt{\sum autovalores} \; ; \; \alpha = 0, 0.05, 0.1, 0.15, 0.2, \ldots 1$$

Consideramos λ como un porcentaje de la raíz de la suma de los autovalores obtenidos con PCA. Estudiamos el comportamiento del aprendizaje con diferentes valores de α. En los límites del intervalo al que pertenece α, cuando equivale a 0 o 1, el algoritmo de agrupamiento se comportaba de dos maneras distintas. En el caso de α = 0, el aprendizaje supervisado no actuaba en el entrenamiento produciendo que k-means utilizase solamente el descriptor para clasificar. En cambio; en el caso de α = 1, el peso del factor λ era muy elevado respecto a los valores del descriptor, produciendo que el algoritmo de agrupamiento k-means solo se guiase por la etiqueta del fonema para agrupar.

Por este motivo, calculamos las matrices de emisión, transmisión y probabilidad inicial del HMM con varios valores de $\alpha$ tratando de encontrar un valor que equilibrara las dos medidas y permitiera, seguidamente, descifrar qué se había pronunciado en los vídeos de entrenamiento sin abusar del peso del vector supervisado.

Los resultados obtenidos con el vector reducido con el aprendizaje supervisado tampoco fueron suficientemente buenos, ya que para poder descifrar un mensaje el valor del peso supervisado debía ser muy alto, provocando que el algoritmo k-means únicamente se guiara por la parte supervisada para agrupar, dejando de lado los datos obtenidos de la extracción de características. Este hecho no nos interesa, ya que una vez tuvimos que añadir muestras de test, formadas únicamente por los descriptores, sin aprendizaje, no tenían manera de agruparse correctamente.

### c. Agrupación con los vectores reducidos por PCA y añadiendo aprendizaje supervisado con LDA

Por este motivo decidimos aplicar una tercera metodología, concretamente, tras aplicar PCA, aplicamos un método conocido como análisis discriminante lineal (LDA)[3]. LDA es un algoritmo que reduce la dimensionalidad de los datos aplicando aprendizaje supervisado. Para lograrlo es necesario indicarle a qué etiqueta se corresponde cada muestra, es decir qué fonema es pronunciado en cada fotograma. LDA preserva el máximo posible de información discriminatoria pretendiendo encontrar la mayor separabilidad entre clases, es decir, la mayor diferencia entre las medias de las clases contrarias y la menor diferencia entre las muestras de la misma clase. Como resultado de aplicar LDA los datos originales son proyectados en un espacio de dimensión igual al número de clases menos uno (en nuestro caso, un espacio igual al tamaño del vocabulario de fonemas – 1). Utilizamos LDA ya que consideramos que podía ser capaz de agrupar las muestras teniendo en cuenta tanto el fonema al que pertenecía cada fotograma como el vector descriptor de características SIFT extraído de la ROI.

Una vez convertido el espacio del vector reducido por PCA al vector reducido por LDA necesitamos agrupar las muestras en diferentes grupos obteniendo un vector, de

[3] Hemos utilizado la implementación de Van der Maaten:

Van der Maaten, L.J.P.; Postma, E.O. and van den Herik, H.J. (2009). Dimensionality Reduction: A Comparative Review. Tilburg University Technical Report, TiCC-TR 2009-005.

Van der Maaten, L.J.P. and Hinton, G.E. (2008). Visualizing High-Dimensional Data Using t-SNE. *Journal of Machine Learning Research*: 2579-2605.

longitud igual al número de muestras, con valores discretos que nos indicarán a qué grupo pertenece cada fotograma. En este caso, realizamos una clasificación basándonos en la distancia de Mahalanobis.

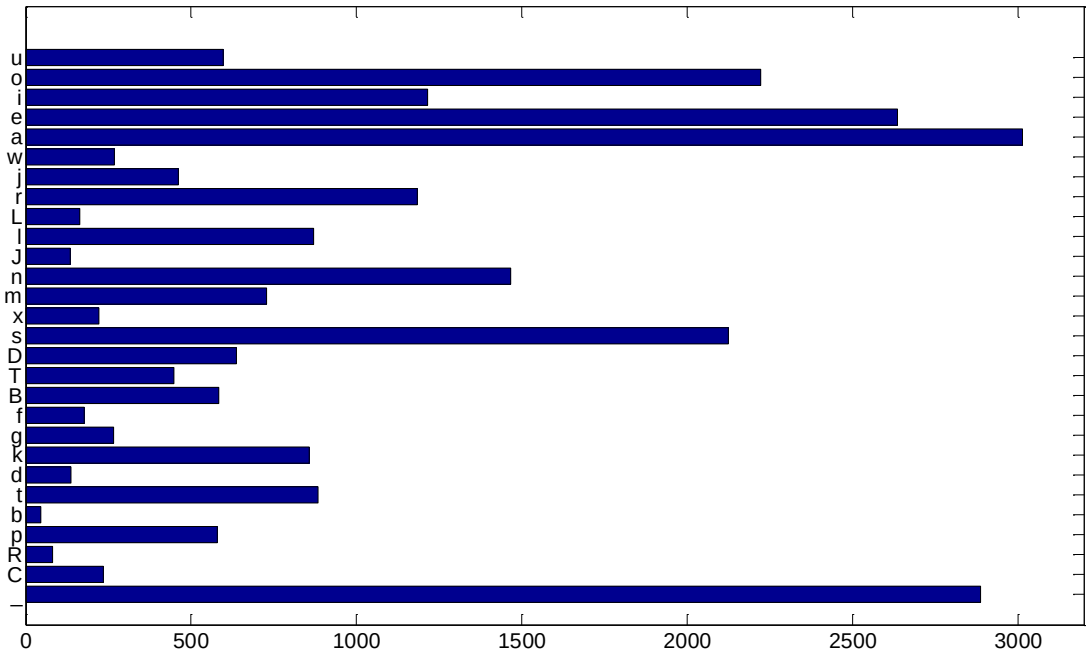
La distancia de Mahalanobis se define como:

$$d_m(\vec{x},\vec{y}) = \sqrt{(\vec{x}-\vec{y})^T\Sigma^{-1}(\vec{x}-\vec{y})}$$

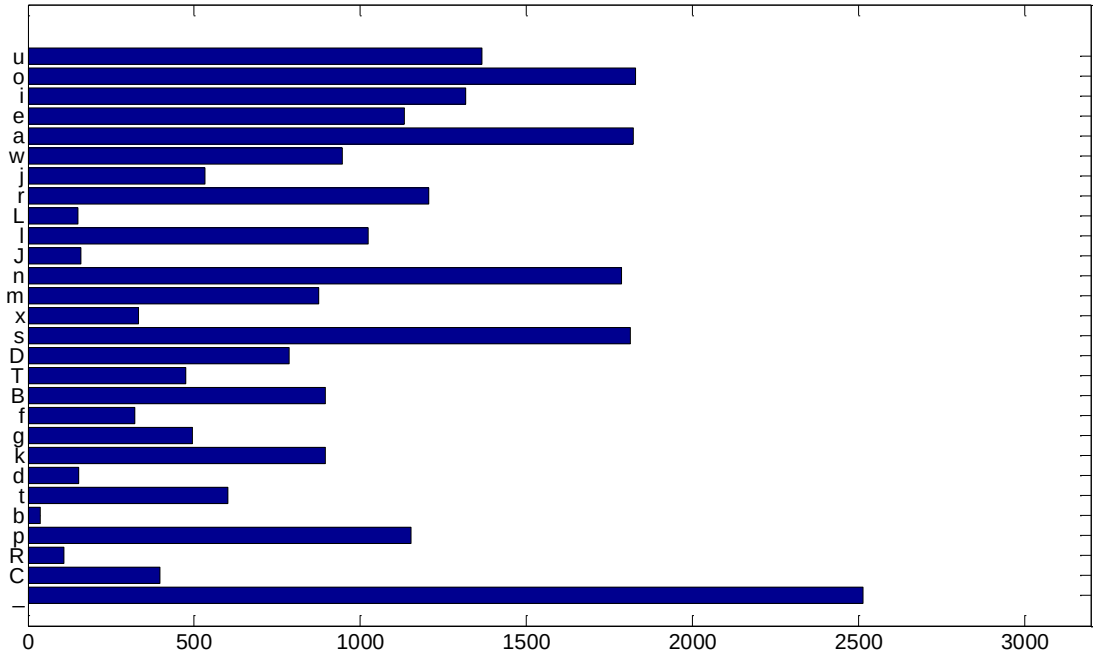
Donde $\vec{x}$ se corresponde con las proyecciones obtenidas mediante LDA, $\vec{y}$ se corresponde con la media de cada uno de los fonemas y $\Sigma^{-1}$ a la matriz de covarianza inversa de las proyecciones obtenidas mediante LDA.

Para obtener el vector que indica la agrupación de las muestras tenemos que calcular, con las proyecciones obtenidas mediante LDA, la media de cada uno de los fonemas del vocabulario y la matriz de covarianza inversa ($\Sigma^{-1}$). Entonces, para cada muestra, buscamos qué grupo tiene la media más próxima a la muestra y asignamos ese grupo a la muestra. De esta manera, obtenemos finalmente una clasificación de las muestras en fonemas.

En las siguientes figuras podemos observar dos histogramas, en la primera se representa el número de muestras que contiene cada fonema de la secuencia real del vídeo y en la segunda se representa el número de muestras de cada fonema asignados en la clasificación.




**XXI. Histograma de la cantidad real fonemas**

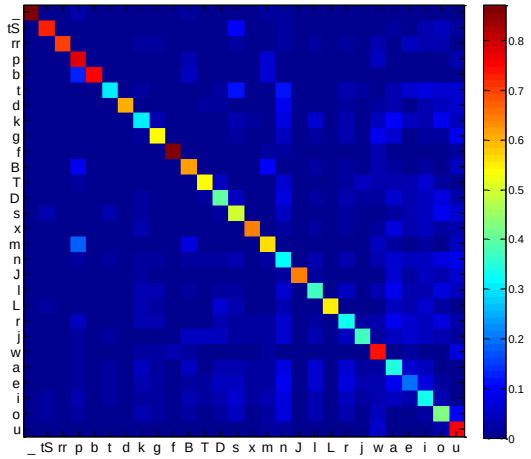



**XXII. Histograma de la cantidad de fonemas según la clasificación basada en LDA**

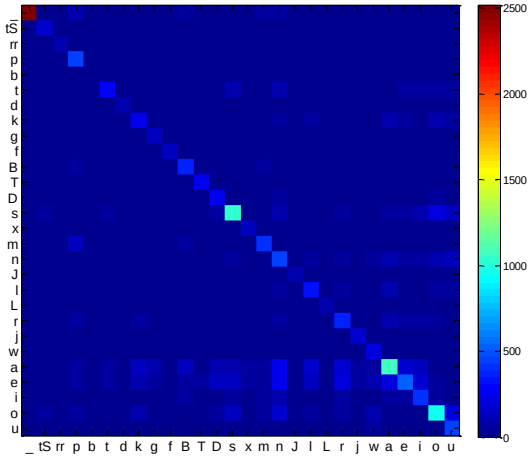
En las figuras anteriores, podemos observar como las muestras clasificadas no han sido clasificadas siempre en los lugares correspondientes a los fonemas originales. Esto es debido a que algunos fonemas se observan igual y pueden ser confundidos con facilidad por el clasificador. Debido a que únicamente tenemos información visual, debemos clasificar la información en función de visemas, no de fonemas. Por ello, tenemos que definir un vocabulario de visemas, de tamaño inferior al tamaño de fonemas, que sea capaz de clasificar los fonemas en grupos de visemas.

Entonces, estudiamos la clasificación de las muestras ayudándonos del conocimiento de su clasificación real. Calculamos la matriz de confusión entre las muestras reales de entrada y las etiquetas en que han sido clasificadas por medio de la distancia de Mahalanobis.

Las siguientes imágenes representan, la matriz de confusión y la matriz de confusión sin normalizar. El eje vertical representa las muestras originales y el eje horizontal las muestras clasificadas. Intuitivamente, vemos que la diagonal de la matriz es la que contiene el mayor número de valores exponiendo que buena parte de las muestras se clasifican correctamente. Si observamos en cualquiera de las figuras, podemos ver cómo los fonemas se confunden a la hora de ser clasificados con fonemas parecidos visualmente, cómo ahora en el caso de los fonemas “b” y “m” que se confunden con el fonema “p”. Este hecho nos hace comprender que estos fonemas tienen relación visual y por lo tanto pueden formar el mismo visema.




**XXIII. Matriz de confusión normalizada**




**XXIV. Matriz de confusión**

Este último método ha sido el que mejores resultados nos ha dado. Por este motivo, partiendo de las proyecciones obtenidas mediante LDA debemos agrupar seguidamente los fonemas en visemas para obtener el vocabulario final. Inicialmente, tenemos un vocabulario de visemas de igual tamaño al de fonemas, pero no conocemos un criterio concreto que nos permita reducir el tamaño del vocabulario de visemas a un tamaño específico. Debido a esto, proponemos una serie de sencillos algoritmos que siguen diferentes criterios de agrupación de fonemas en visemas basándose principalmente en la distribución de los valores de la matriz de confusión normalizada.

## c) Estudio de vocabularios candidatos

Una vez calculada la probabilidad conjunta debemos elegir un criterio de agrupamiento que nos permita unir los fonemas en visemas. Para lograr un vocabulario de visemas capaz de clasificar las diferentes posiciones de la boca y los labios a la hora de hablar, hemos estudiado 6 diferentes escenarios, explicados a continuación.

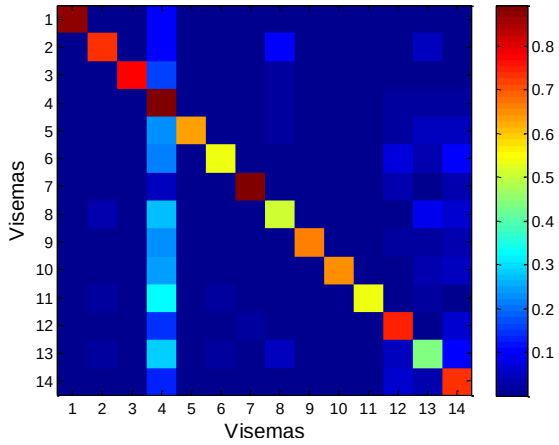
### a. Primer escenario:

Partimos de la matriz de confusión normalizada y de un vocabulario de visemas igual al vocabulario de fonemas. Trabajamos con los 380 vídeos disponibles para el entrenamiento.

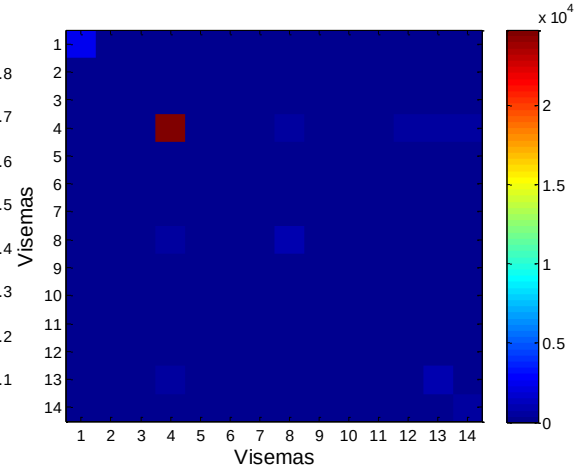
Realizamos un algoritmo iterativo para reducir el tamaño del vocabulario de visemas. El algoritmo encuentra, en cada iteración, el valor máximo fuera de la diagonal. La fila correspondiente a ese valor es el primer visema a agrupar, A, y la columna de ese valor es el segundo visema, B. Entonces se unen los visemas A y B formando un nuevo visema (A, B) en el vocabulario. Las filas y columnas correspondientes a estos visemas son sumadas formando una sola. El razonamiento detrás de este método es que el máximo valor fuera de la diagonal indica los dos fonemas que más confusión generan en el clasificador.
Tras aplicar este algoritmo, podemos observar en las siguientes figuras las matrices de confusión, normalizada y sin normalizar, el tanto por ciento de acierto en cada iteración del algoritmo y el vocabulario final reducido a 14 visemas.

$$Porcentaje\ acierto\ visema = \sum diagonal / Nmuestras$$

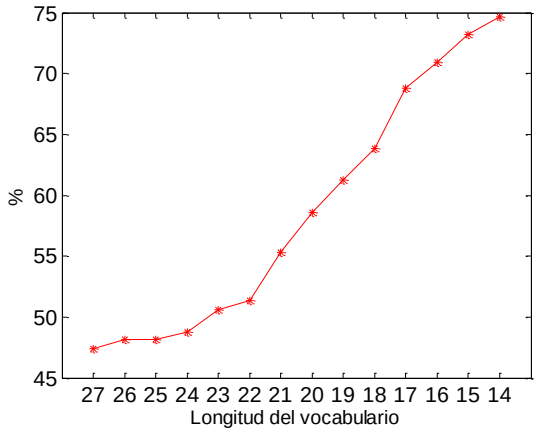
Podemos observar en las siguientes figuras como se han ido agrupando la mayoría de fonemas en un mismo visema; esto es debido a que muchos fonemas tenían un número bastante elevado de muestras colocadas en vocales, concretamente las vocales “e”, “a” e “i”, debido a que algunos fonemas varían visualmente dependiendo de la vocal siguiente o debido a errores de precisión del programa Praat. Además, al unirse dos vocales, éstas han aumentado el número de muestras erróneas de otros fonemas respecto a ellas, agrupándose con ellas en las siguientes iteraciones.



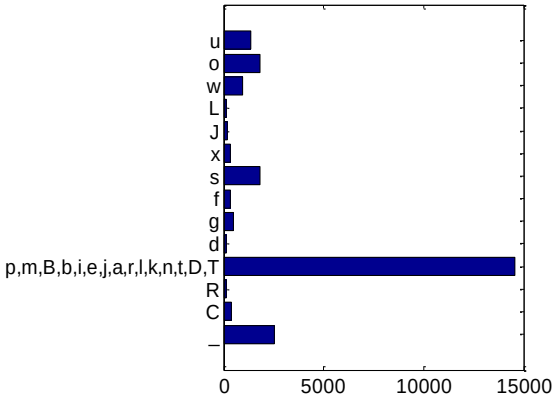

Matriz de confusión normalizada




Matriz de confusión




Tanto por ciento de acertar el visema




Vocabulario de visemas

**XXV. Estudio escenario 1**

Se puede observar que el porcentaje de acierto de visemas es elevado debido a que un único visema está compuesto por 15 fonemas, por lo que acertar el visema en este caso no tiene demasiada dificultad. Tendremos mayores dificultades para acertar cuál es el fonema dicho realmente, ya que con tantas posibilidades de fonemas dentro de un

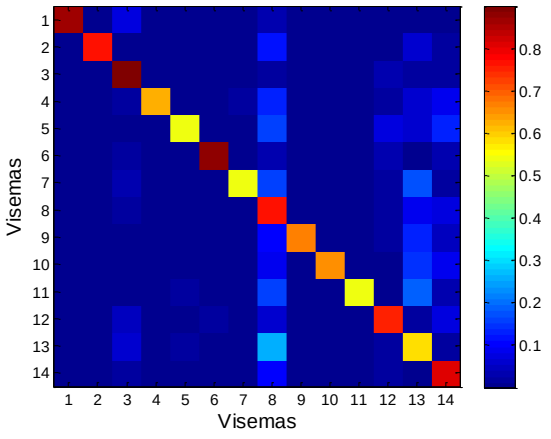
mismo visema será muy difícil acertar el fonema dicho únicamente con la información visual. Por ese motivo, tratamos de explorar el siguiente escenario.

## b. Segundo escenario:

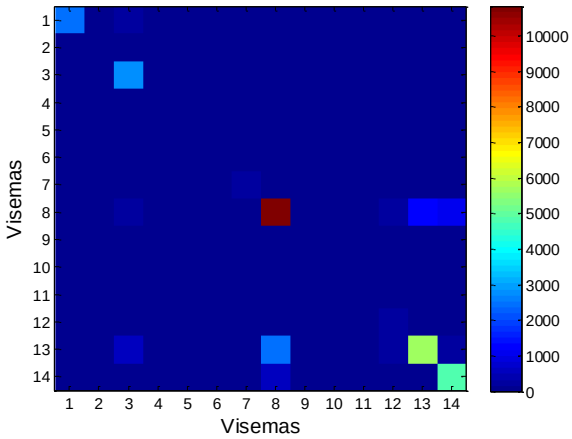
Partimos de la matriz de confusión normalizada y de un vocabulario de visemas igual al vocabulario de fonemas. Tras observar los resultados del primer escenario, pensamos restringir el algoritmo anterior no permitiendo que las vocales se agrupasen con otras consonantes, ya que al parecer esto condiciona a que todos los fonemas se agrupen en uno. Además, también condicionamos que dos grupos formados por dos o más fonemas no puedan agruparse, pretendiendo evitar macro grupos de fonemas.

De esta manera, el algoritmo encuentra el valor máximo fuera de la diagonal, del cual la fila corresponde al visema A y la columna al visema B. Entonces se unen los visemas A y B formando el visema (A, B) únicamente si A y B son ambas consonantes o ambas vocales y al menos uno de los dos no es un visema "compuesto" (es decir, formado por grupos de fonemas). Las filas y columnas correspondientes a estos visemas son sumadas formando una sola.

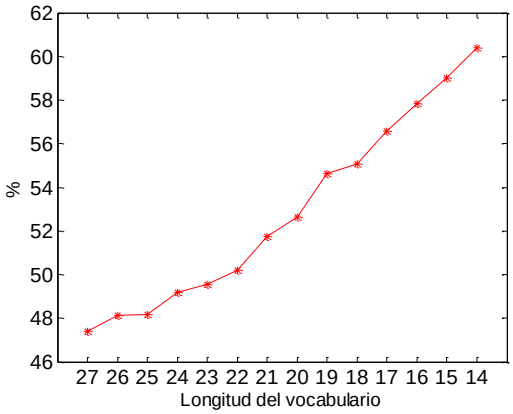
Seguidamente, podemos observar como eliminando la agrupación de vocales con consonantes obtenemos más variedad de grupos de fonemas. En la matriz de confusión, podemos observar como el visema 8 es el que contiene más fonemas, se puede reconocer ya que es el que tiene el mayor número de muestras, aproximadamente 10000. Los visemas con mayor número de fonemas son las más difíciles de clasificar, tal y como sucede también con el fonema 13 que contiene 4 fonemas. Podemos observar la relación entre fonemas y visemas en la figura del vocabulario de fonemas. Si observamos el porcentaje de acierto de visemas, podemos ver como a medida que reducimos el vocabulario aumenta hasta sobrepasar el 60% de aciertos. El porcentaje de acertar el visema ha disminuido respecto el caso anterior, pero la distribución del vocabulario es más realista y, especialmente, más informativa en términos de fonemas.



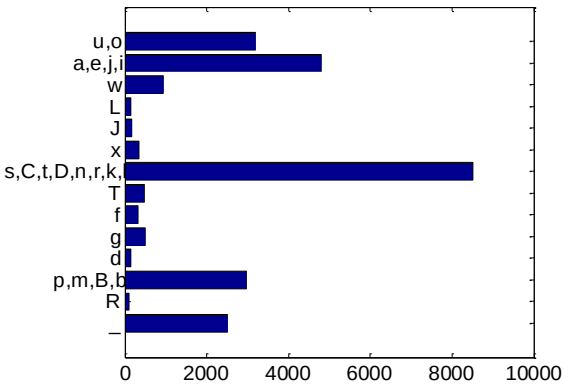

Matriz de confusión normalizada




Matriz de confusión




Tanto por ciento de acertar el visema




Vocabulario de visemas

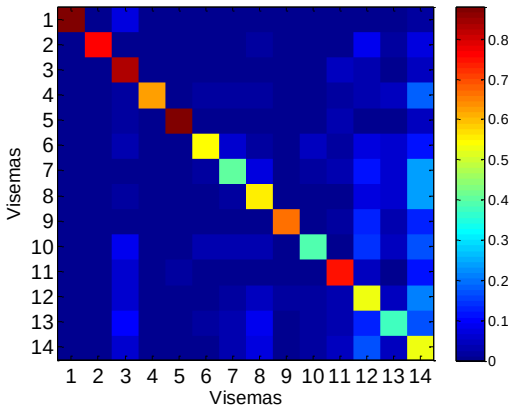
**XXVI. Estudio escenario 2**

Seguimos estudiando más escenarios para decidir, más adelante, qué vocabulario resulta más adecuado para descifrar las frases pronunciadas en cada vídeo.

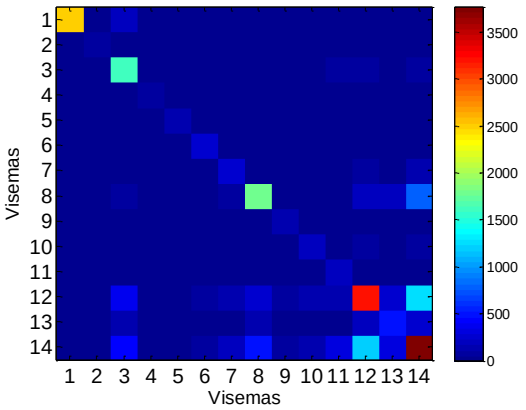
## c. Tercer escenario:

En este caso, también partimos de la matriz de confusión normalizada y de un vocabulario de visemas igual al vocabulario de fonemas.

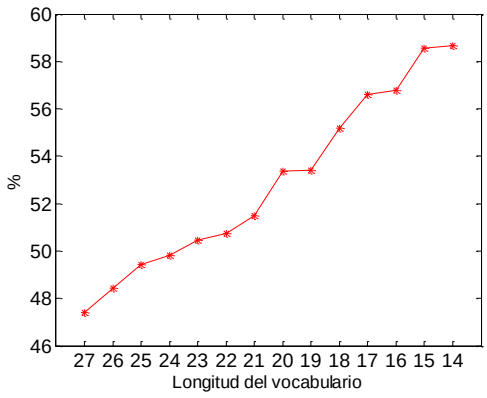
Consideramos un algoritmo que encuentre, para cada fila de la matriz, el valor máximo fuera de la diagonal. Seguidamente almacenamos el valor y las coordenadas de este valor, la fila corresponde al visema en el que se encuentra y la columna al visema con el cual se va a unir. Una vez almacenados todos los valores los ordenamos de mayor a menor y agrupamos en cada iteración cada una de las filas y columnas correspondientes, reduciendo el vocabulario hasta el tamaño deseado. Con la condición de que grupos de dos o más fonemas no puedas agruparse con otros de igual o mayor tamaño.
A continuación, mostramos las figuras correspondientes a las matrices de confusión, el cálculo del porcentaje de acierto y el vocabulario obtenido.



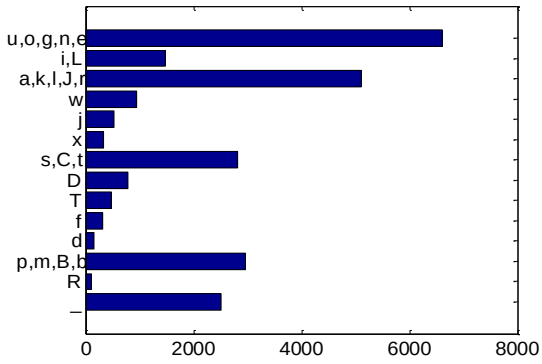

Matriz de confusión normalizada




Matriz de confusión




Tanto por ciento de acertar el visema




Vocabulario de visemas

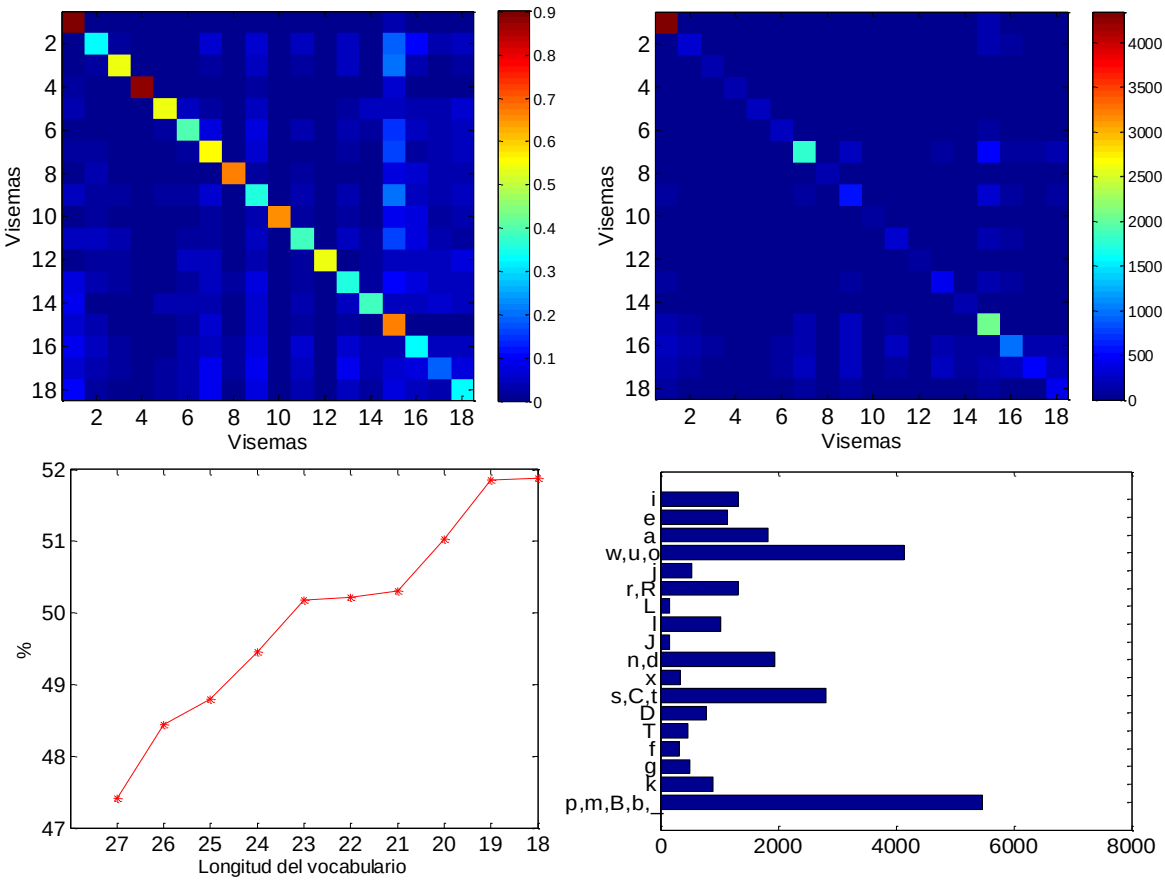
**XXVII. Estudio escenario 3**

En las figuras anteriores, podemos observar como el porcentaje de acierto de visemas se ha reducido respecto al escenario 2, no superando el 60% de acierto, pero en cambio, el vocabulario obtenido es más homogéneo, teniendo únicamente visemas de tamaño máximo 5. Si nos fijamos en las matrices de confusión, podemos observar como encontramos mayor dispersión que en el caso anterior; esto es debido a que visemas de un fonema ahora están formados por varios fonemas, aumentando el número de muestras mal clasificadas en ese visema. Los grupos donde aparecen las vocales son aquellos con más muestras y además son aquellos a los que otros grupos tienen más muestras mal clasificadas.

## d. Cuarto escenario:

Este escenario es similar al anterior, incluye todas las características y condiciones del escenario 3, pero además no permite que vocales y consonantes se unan entre ellas.

En las siguientes figuras, encontramos el vocabulario de visemas donde podemos observar como tenemos 18 visemas. Esto es debido a que este algoritmo sólo ha iterado una vez la matriz de confusión, por lo tanto únicamente almacenamos una vez los valores máximos de cada fila. Entonces, al recorrer todas las filas hemos logrado reducir el vocabulario a 18 visemas. La matriz de confusión es bastante dispersa, especialmente en el fonema formado por “w,u,o”. La probabilidad de acertar el visema utilizando en este algoritmo es inferior a los casos anteriores, superando ligeramente el 50% de acierto.




Matriz de confusión normalizada
Tanto por ciento de acertar el visema

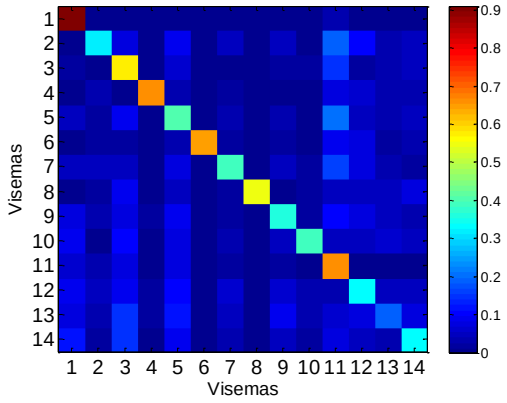
Matriz de confusión
Vocabulario de visemas

**XXVIII. Estudio escenario 4**

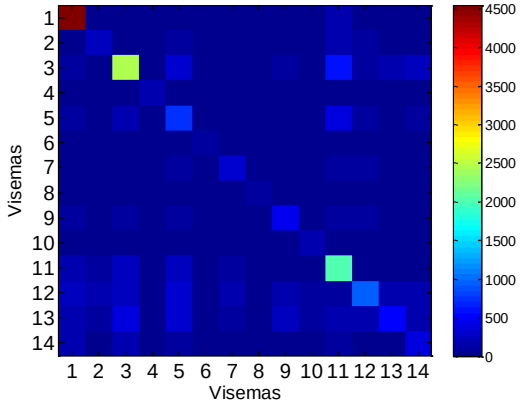
## e. Quinto escenario:

Este escenario es equivalente al escenario anterior, pero añadiendo una nueva restricción. Una vez acabada la primera iteración del algoritmo donde se han asignado los máximos fuera de la diagonal de la matriz de confusión de tamaño 28, si el vocabulario tiene un tamaño superior al que deseamos (en este caso, 14 visemas), recalculamos la matriz de confusión, los máximos de esta nueva matriz y reducimos el vocabulario con estos nuevos valores.

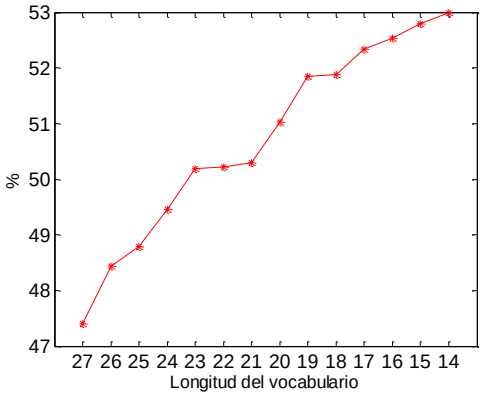
En las siguientes figuras, podemos observar como la precisión a la hora de acertar un fonema ha aumentado respecto al caso anterior, debido a que hemos reducido el tamaño del vocabulario de 18 visemas a 14. Los fonemas se encuentran muy repartidos, no logrando visemas de más de 5 fonemas. Observamos como la matriz de confusión tiene bastante dispersión fuera de la diagonal.



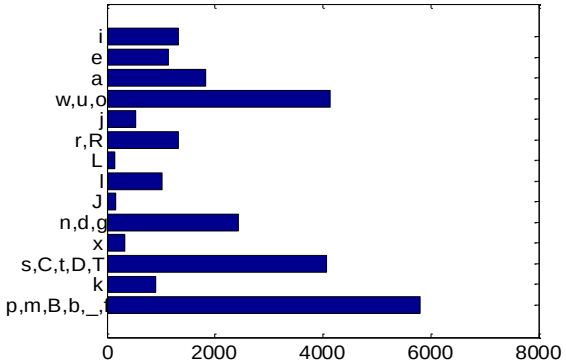

Matriz de confusión normalizada




Matriz de confusión




Tanto por ciento de acertar el visema




Vocabulario de visemas

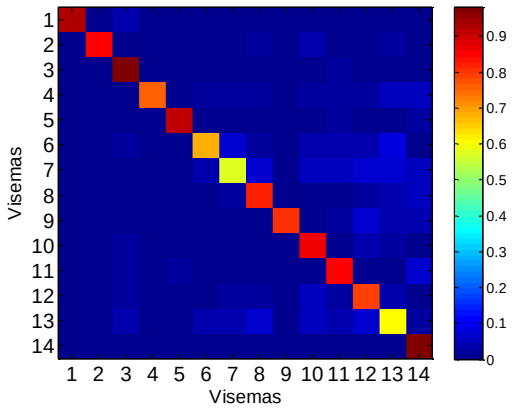
**XXIX. Estudio escenario 5**

## f. Sexto escenario:

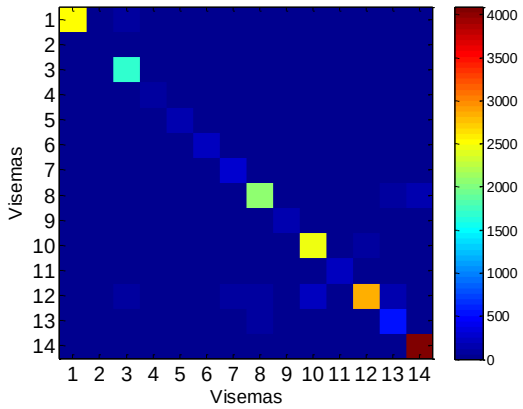
Partimos de la matriz de confusión normalizada y del vocabulario inicial con 28 fonemas.

El criterio seguido en este escenario es agrupar dos visemas siempre que uno tenga el máximo fuera de la diagonal en el otro. Entonces, añadimos las muestras del visema A a la fila en la que el valor se corresponde con la diagonal del visema B. De esta manera evitamos el error causado al sumar filas y columnas de dos visemas, donde son aumentados la diagonal y los valores de otros visemas mal clasificados. Además reducimos los valores mal clasificados en el visema con el que te unes, ya que al aumentar su centro y normalizar los valores fuera de la diagonal disminuyen, logrando que menos visemas quieran unirse a este grupo.

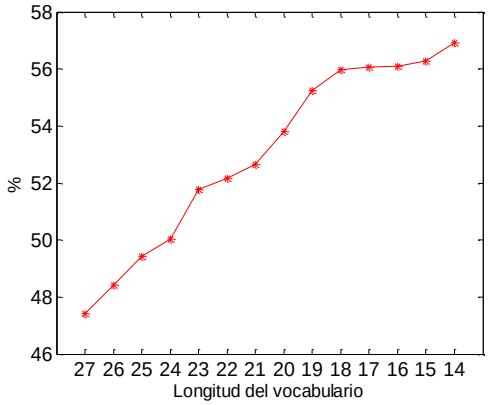
En las figuras siguientes, podemos observar como en el caso de las matrices de confusión la dispersión observada en casos anteriores ha disminuido significativamente. Además hemos recuperado porcentaje de acierto de visemas llegando al 57% de acierto. El tamaño de los visemas que contiene el vocabulario no es superior a 5.



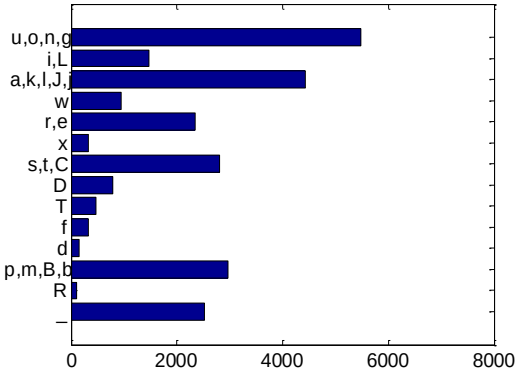

Matriz de confusión normalizada




Matriz de confusión




Tanto por ciento de acertar el visema




Vocabulario de visemas

**XXX. Estudio escenario 6**

En la siguiente tabla se muestran los seis vocabularios estudiados. Podemos observar como el vocabulario 1 es un mal candidato a ser el vocabulario de visemas final, ya que está formado por un macro visema (con 15 fonemas) que nos dificultaría acertar el fonema pronunciado. En el vocabulario 2, tratamos de reducir el macro visema evitando que vocales y consonantes estén agrupadas en un mismo visema, logrando que el macro visema se convierta principalmente en dos visemas de 8 y 4 fonemas que se habían unido por culpa de las muestras erróneas comunes destinadas a las vocales. En el vocabulario 6, tratamos de reducir los macro visemas evitando sumar los errores al agrupar visemas. En los vocabularios 3, 4 y 5 se han aplicado los criterios de agrupación partiendo siempre de la matriz de confusión inicial; debido a que se pretendía evitar que la actualización de la matriz de confusión condicionara que visemas que anteriormente no tenían máximo en ciertos visemas ahora los tuvieran.

**Tabla 5**

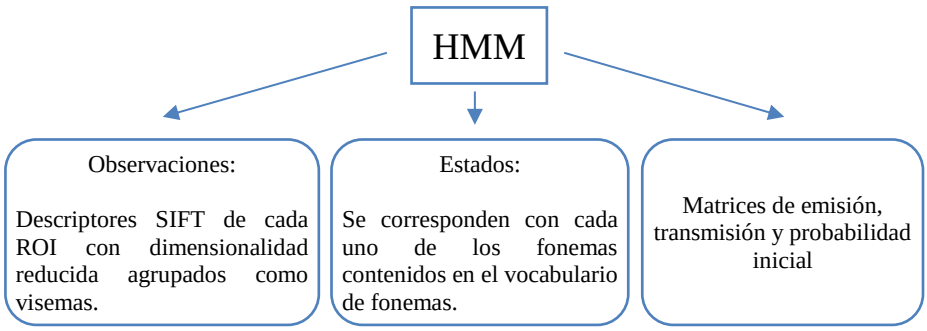
| Tabla comparativa de vocabularios de visemas | | | | | |
|---|---|---|---|---|---|
| **Caso 1** | **Caso 2** | **Caso 3** | **Caso 4** | **Caso 5** | **Caso 6** |
| **u** | **u,o** | **u,o,g,n,e** | **i** | **i** | **u,o,n,g** |
| **o** | **a,e,j,i** | **i,L** | **e** | **e** | **i,L** |
| **w** | **w** | **a,k,l,J,r** | **a** | **a** | **a,k,l,J,j** |
| **L** | **L** | **w** | **w,u,o** | **w,u,o** | **w** |
| **J** | **J** | **j** | **j** | **j** | **r,e** |
| **x** | **x** | **x** | **r,R** | **r,R** | **x** |
| **s** | **s,C,t,D,n,r,k,l** | **s,C,t** | **L** | **L** | **s,t,C** |
| **f** | **T** | **D** | **l** | **l** | **D** |
| **g** | **f** | **T** | **J** | **J** | **T** |
| **d** | **g** | **f** | **n,d** | **n,d,g** | **f** |
| **p,m,B,b,i,e,j,a,r,l,k,n,t,D,T** | **d** | **d** | **x** | **x** | **d** |
| **R** | **p,m,B,b** | **p,m,B,b** | **s,C,t** | **s,C,t,D,T** | **p,m,B,b** |
| **C** | **R** | **R** | **D** | **k** | **R** |
| **_** | **_** | **_** | **T** | **p,m,B,b,_,f** | **_** |
| | | | **f** | **º** | |
| | | | **g** | | |
| | | | **k** | | |
| | | | **p,m,B,b,_** | | |

Necesitamos fijar un único vocabulario que nos permita entrenar nuestro modelo de Markov para poder después descifrar qué dice cualquier vídeo que introduzcamos.

Primeramente construiremos el modelo oculto de Markov y seguidamente, añadiremos los 95 vídeos destinados a test para analizar con cuál de los vocabularios se obtienen mejores resultados. Finalmente, fijaremos un único vocabulario de visemas.

## d) Construcción del modelo oculto de Markov

Este es el momento de construir el clasificador basado en HMM. Tal y como se ha comentado anteriormente, un modelo oculto de Markov necesita un conjunto de observaciones y una serie de estados. El modelo construido sigue la siguiente estructura:




**XXXI. Esquema HMM**

Donde las matrices de emisión, transmisión y probabilidad inicial se calculan del siguiente modo:

- Matriz de emisiones, *B*:

La matriz de emisiones se corresponde con la probabilidad de observar un símbolo estando en cada uno de los estados.

- Conocemos:

- El vector contenedor de los fonemas pronunciados en cada fotograma, es decir, el valor de la secuencia real de estados (*frame_phoneme*).
- Los posibles estados, correspondientes al vocabulario de fonemas (*vocabulary_phonemes*).
- Las posibles observaciones, correspondientes al vocabulario de visemas (*vocabulary_viseme*) .
- El vector discreto que contiene la asignación de a qué visema pertenece cada fotograma (*clusters*).

Con estos datos, podemos calcular la matriz de emisiones, simplemente debemos seguir este sencillo algoritmo:

```
for i = 1:length(vocabulary_phoneme)
        phonemes = find(strcmp(frame_phoneme, vocabulary_phoneme (i)));
        %Buscamos para cada fonema todos los fotogramas que lo tienen asignado y
        almacenamos sus posiciones en phonemes.
   for j = 1: length(vocabulary_viseme)
      visemes = find( clusters(phonemes) == j);
     %Bucamos en el vector clusters qué visemas han sido asignados al fonema
     estudiado.
      if(~isempty(visemes))
        emision(j,i) = length(visemes)/length(phonemes);
        %Calculamos la probabilidad de obtener cada uno de los visemas dados cada
        uno de los fonemas.
      end
   end
end
```

- Matriz de transmisión, A:

La matriz de transmisión se corresponde con la probabilidad de ir al estado *x*, estando actualmente en el estado *y*, donde *x* e *y* corresponden a todas los posibles combinaciones de estados.

- Conocemos:

- El vector contenedor de los fonemas dichos en cada fotograma, es decir, el valor de la secuencia de estados real (*frame_phoneme*).

- Los posibles estados, correspondientes al vocabulario de fonemas (*vocabulary_phonemes*).

Para calcular esta matriz únicamente tenemos que calcular para cada fonema del vocabulario de fonemas, qué fonemas le siguen en la secuencia real y contabilizarlo como probabilidad. Tal y como se indica en el siguiente algoritmo:

```
for i = 1:length(vocabulary_phonemes)
   ph = find(strcmp(frame_phoneme, vocabulary_phonemes {i}));
  %Buscamos para cada fonema todos los fotogramas que lo tienen asignado y
  almacenamos sus posiciones en ph.
   ph_n = ph + 1; %Almacenamos las posiciones de los fonemas siguientes.
   if(ph(end) == length(frame_phoneme))
      ph_n = ph_n(1:end-1);
       %En el caso de que ph contenga la posición del último fotograma eliminamos la
       última posición del vector ph_n ya que no habrá fonema siguiente.
   end
   next_ph = frame_phoneme(ph_n); %Almacenamos los fonemas siguientes
   for j = 1:length(vocabulary_phonemes)
      ph2 = find(strcmp(next_ph, vocabulary_phonemes {j}));
         %Buscamos cuantas veces aparece cada fonema siguiente.
      transmision(i,j) = length(ph2)/length(ph);
       %Calculamos la probabilidad de qué después de un fonema aparezca cada uno de
       los fonemas posibles.
   end
end
```

- <u>Matriz de probabilidad inicial, π:</u>

La probabilidad inicial se ha calculado utilizando una *caja de herramientas* de HMM. Concretamente, utilizando una serie de funciones que partiendo de un conjunto de secuencias de entrenamiento, la matriz de transmisión y la matriz de emisiones permiten calcular la matriz π. Nos interesa tener un conjunto de secuencias de entrenamiento muy elevado para poder obtener matrices que se ajusten lo mejor posible a nuestros datos. Hemos utilizado 380 secuencias diferentes para calcular la matriz. En nuestro caso, la matriz de probabilidad inicial ha tenido poca importancia, ya que los vídeos siempre empiezan en silencio y por lo tanto todas las secuencias han empezado en el estado "_".

<u>Decodificación de las secuencias:</u>

Finalmente, se utiliza el algoritmo de Viterbi, proporcionado por la misma caja de herramientas, para obtener la secuencia de estados que ha emitido con mayor probabilidad la secuencia de observaciones introducida. Obtenemos así, la secuencia de fonemas que aproxima la frase que se ha pronunciado en el vídeo.

# 5. Resultados

En este apartado elegiremos el vocabulario de visemas que mejor interprete la información de los datos de entrada. También mostraremos un ejemplo de la salida del todo el sistema de reconocimiento visual automático.

## 5.1. Elección del vocabulario final

En este momento tenemos seis vocabularios de visemas candidatos, pero necesitamos definir únicamente uno. Hemos considerado que la mejor manera de decidir cuál será el vocabulario final es testear cada uno de ellos con las muestras de test. De esta manera, podremos calcular las probabilidades de acierto de visemas y de fonemas con muestras que no han sido entrenadas en el sistema y ver qué vocabulario nos permite recuperar más información de los vídeos de entrada.

Para cada vocabulario candidato realizamos:

- Entrenamiento del Modelo de Markov con el vocabulario específico.
- Añadimos las muestras de test de todos los usuarios, las convertimos en los diferentes espacios hasta obtener su proyección en el espacio LDA.
- Calculamos probabilidad de acierto de visemas y de fonemas.

Inicialmente, entrenamos el modelo de Markov, tal y como se ha explicado anteriormente, calculando las matrices de transmisión, emisión y probabilidades iniciales.

Seguidamente, debemos transformar los 95 vídeos reservados para test. Partimos de un vídeo y tenemos que asignarle un visema a cada uno de sus fotogramas. Para realizar este proceso realizamos unos pasos similares a los realizados con el conjunto de entrenamiento. El primer paso es obtener la ROI mediante el algoritmo ASM, el segundo calcular el descriptor SIFT de dimensión 4480 y el tercero transformar éste en los espacios de PCA y LDA.

Los dos primeros pasos se realizan exactamente igual para el conjunto de test que para el conjunto de entrenamiento. En cambio, la conversión a PCA se realiza restando la media al descriptor SIFT y seguidamente multiplicando el resultado por una matriz de coeficientes que transforma la dimensión de 4480 a 949, ambas matrices obtenidas en el entrenamiento de los datos al aplicar PCA. Una vez hemos reducido la dimensionalidad del vector a 949 transformamos estos vectores al espacio LDA del mismo modo, restando la media y multiplicando por las matrices de proyección, obtenidas también durante el entrenamiento.
Finalmente, debemos aplicar la distancia de Mahalanobis a cada descriptor para asignarle el visema más probable.
De esta manera, ya hemos obtenido el vector discreto que contiene la estimación automática del visema observado en cada fotograma.

El último paso realizado para cada vocabulario con el conjunto de test es decodificar los fonemas o visemas asignados a cada fotograma con el algoritmo de Viterbi y recuperar así los fonemas o visemas más probables pronunciados en cada fotograma. De esta

manera, obtenemos nuestra estimación final de la frase que ha sido pronunciada en el vídeo, utilizando la información visual y estadística de transiciones entre estados.

Podemos calcular las probabilidades de acierto de fonemas y de visemas ya que conocemos la secuencia final obtenida con Viterbi y conocemos la secuencia real de fonemas obtenida con la ayuda del programa Praat. Por lo tanto, únicamente debemos contar cuantas veces acertamos los fonemas y visemas con respecto al vocabulario que estemos estudiando.

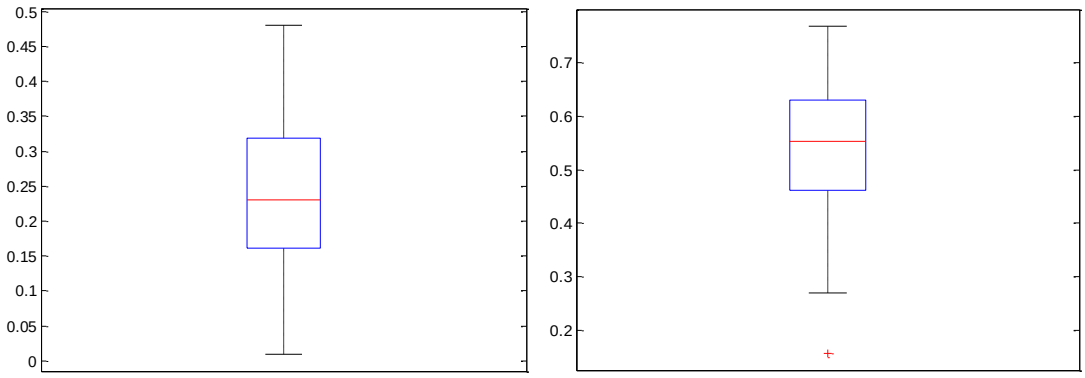
Seguidamente mostramos los resultados de las probabilidades obtenidas con los diferentes vocabularios.

El vocabulario candidato del primer escenario es el siguiente:

**Tabla 6**

| Vocabulario de visemas 1 | | | | | | |
|---|---|---|---|---|---|---|
| u | o | w | L | J | x | s |
| f | g | d | p,m,B,b,i,e,j,a,r,l,k,n,t,D,T | R | C | _ |

Los resultados obtenidos tras realizar los pasos que acabamos de explicar utilizando el vocabulario de visemas 1 son:




Probabilidad de acertar el fonema  Probabilidad de acertar el visema

**XXXII. Probabilidades obtenidas con el vocabulario 1**

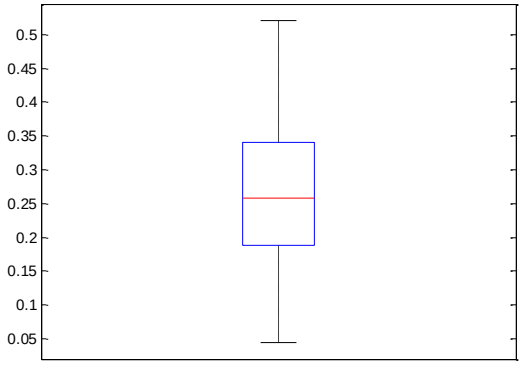
En la figura derecha, podemos observar como la probabilidad de acertar el visema utilizando el vocabulario 1 es de 0.552; y en la izquierda como la probabilidad de acertar el fonema es de 0.230.

El vocabulario candidato del segundo escenario es:

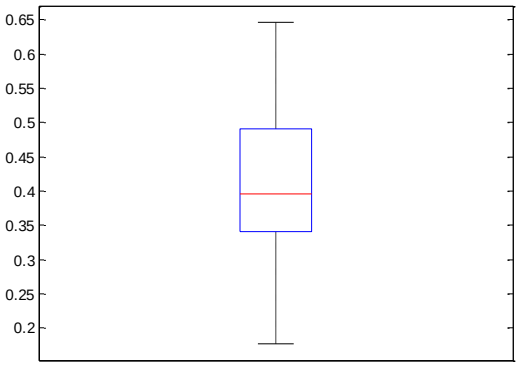
**Tabla 7**

| Vocabulario de visemas 2 | | | | | | |
|---|---|---|---|---|---|---|
| u,o | a,e,j,i | w | L | J | x | s,C,t,D,n,r,k,l |
| T | f | g | d | p,m,B,b | R | _ |

Los resultados obtenidos tras realizar los pasos explicados anteriormente utilizando el vocabulario de visemas 2 son:




Probabilidad de acertar el fonema




Probabilidad de acertar el visema

**XXXIII. Probabilidades obtenidas con el vocabulario 2**

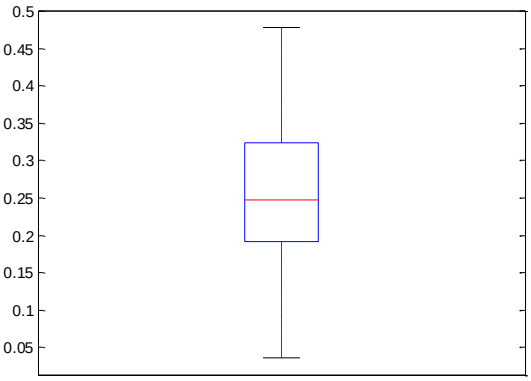
En la figura derecha, podemos observar como la probabilidad de acertar el visema utilizando el vocabulario 2 es de 0.396; y en la izquierda como la probabilidad de acertar el fonema es de 0.258.

El vocabulario candidato del tercer escenario es:

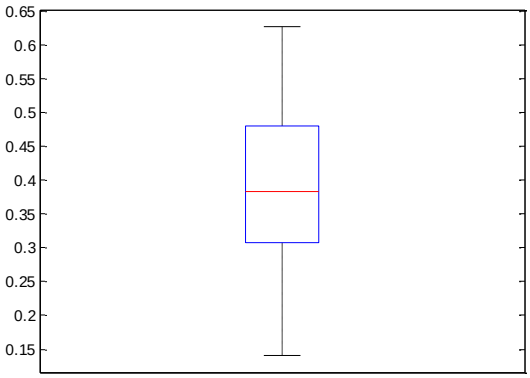
**Tabla 8**

| Vocabulario de visemas 3 | | | | | | |
|---|---|---|---|---|---|---|
| u,o,g,n,e | i,L | a,k,l,J,r | w | j | x | s,C,t |
| D | T | f | d | p,m,B,b | R | _ |

Los resultados obtenidos tras realizar los pasos explicados anteriormente utilizando el vocabulario de visemas 3 son:




Probabilidad de acertar el fonema




Probabilidad de acertar el visema

**XXXIV. Probabilidades obtenidas con el vocabulario 3**

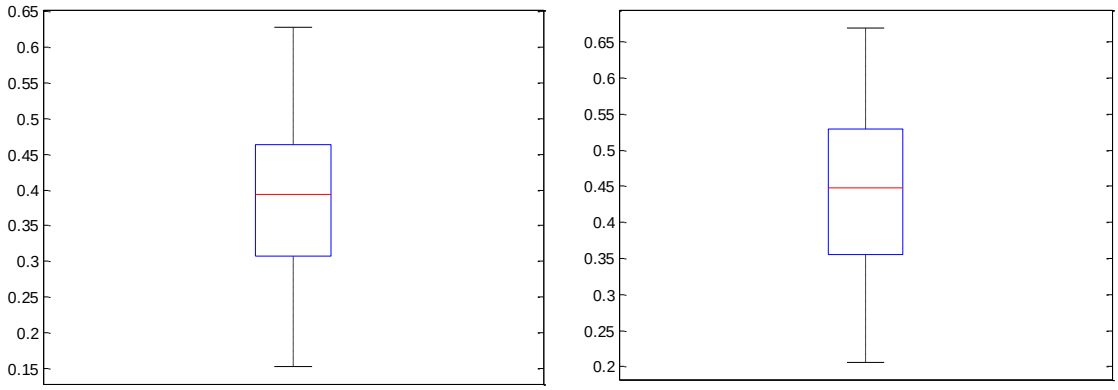
En la figura derecha, podemos observar como la probabilidad de acertar el visema utilizando el vocabulario 3 es de 0.383 y en la izquierda como la probabilidad de acertar el fonema es de 0.247.

El vocabulario candidato del cuarto escenario es:

**Tabla 9**

| Vocabulario de visemas 4 | | | | | |
|---|---|---|---|---|---|
| i | e | a | w,u,o | j | r,R |
| L | l | J | n,d | x | s,C,t |
| D | T | f | g | k | p,m,B,b,_ |

Los resultados obtenidos tras realizar los pasos explicados anteriormente utilizando el vocabulario de visemas 4 son:




Probabilidad de acertar el fonema Probabilidad de acertar el visema

**XXXV. Probabilidades obtenidas con el vocabulario 4**

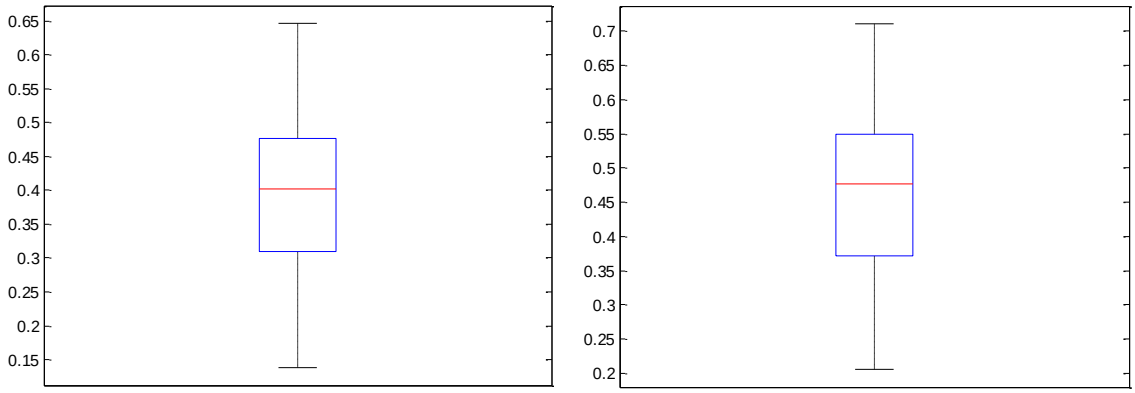
En la figura derecha, podemos observar como la probabilidad de acertar el visema utilizando el vocabulario 4 es de 0.448; y en la izquierda como la probabilidad de acertar el fonema es de 0.394.

El vocabulario candidato del quinto escenario es:

**Tabla 10**

| Vocabulario de visemas 5 | | | | | | |
|---|---|---|---|---|---|---|
| i | e | a | w,u,o | j | r,R | L |
| l | J | n,d,g | x | s,C,t,D,T | k | p,m,B,b,_,f |

Los resultados obtenidos tras realizar los pasos explicados anteriormente utilizando el vocabulario de visemas 5 son:




Probabilidad de acertar el fonema  Probabilidad de acertar el visema

**XXXVI. Probabilidades obtenidas con el vocabulario 5**

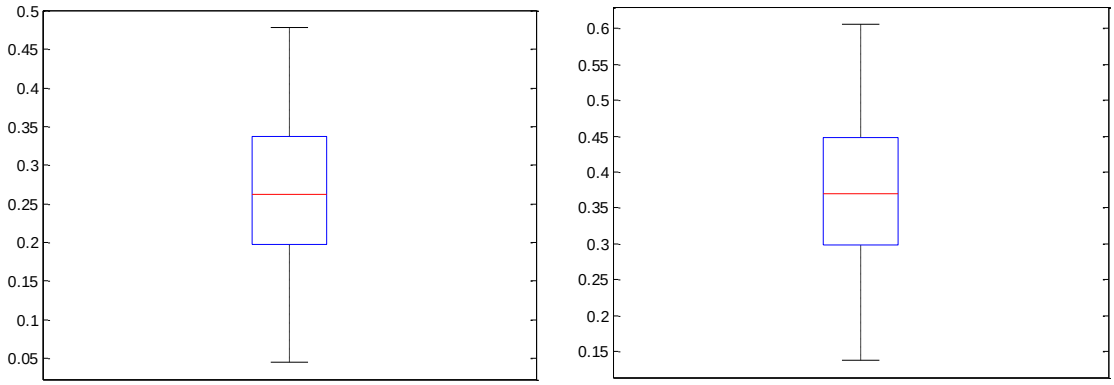
En la figura derecha, podemos observar como la probabilidad de acertar el visema utilizando el vocabulario 5 es de 0.476; y en la izquierda como la probabilidad de acertar el fonema es de 0.382.

El vocabulario candidato del sexto escenario es:

**Tabla 11**

| Vocabulario de visemas 6 | | | | | | |
|---|---|---|---|---|---|---|
| u,o,g,n | i,L | a,k,l,J,j | w | r,e | x | s,C,t |
| D | T | f | d | p,m,B,b | R | _ |

Los resultados obtenidos tras realizar los pasos explicados anteriormente utilizando el vocabulario de visemas 6 son:




Probabilidad de acertar el fonema  Probabilidad de acertar el visema

**XXXVII. Probabilidades obtenidas con el vocabulario 6**

En la figura derecha, podemos observar como la probabilidad de acertar el visema utilizando el vocabulario 6 es de 0.369 y en la izquierda como la probabilidad de acertar el fonema es de 0.262.

En los resultados mostrados arriba observamos como al evaluar los datos de test con los diferentes vocabularios de visemas, obtenemos que los vocabulario 4 y 5 son los que

ofrecen mejores resultados, es decir, descifran mayor cantidad del mensaje contenido en los vídeos, llegando en promedio a acertar el 40% de fonemas y entre el 44% y 47% de visemas. Por lo tanto, tenemos que fijar uno de ellos como vocabulario final de visemas, concretamente, fijamos el vocabulario 5 ya que se corresponde con el tamaño de vocabulario esperado y además aporta un porcentaje significativamente más alto en el acierto de visemas.

Una vez calculadas las probabilidades de acierto observamos la secuencia estimada final para ver si realmente somos capaces de interpretar (o leer) que se ha pronunciado en el vídeo. El primer resultado derivado de este análisis fue que el sistema era mucho más eficiente para detectar correctamente los momentos de silencio que para acertar los fonemas o visemas que habían sido pronunciados. Este hecho era de esperar, ya que en la mayor parte de los vídeos el fonema más frecuente es el del silencio, por lo tanto el clasificador ha sido mucho más entrenado para identificar este fonema. Sin embargo, es interesante también analizar hasta qué punto el vocabulario seleccionado nos permite acertar el mayor número de fonemas o visemas pronunciados en los vídeos; pudiendo así descifrar la mayor parte del mensaje posible. Llegados a este punto, calculamos, para cada vocabulario, la probabilidad de acertar el fonema o el visema considerando los silencios de forma separada al resto.

A continuación, observamos en las tablas 11 y 12 las probabilidades calculadas:

**Tabla 12**

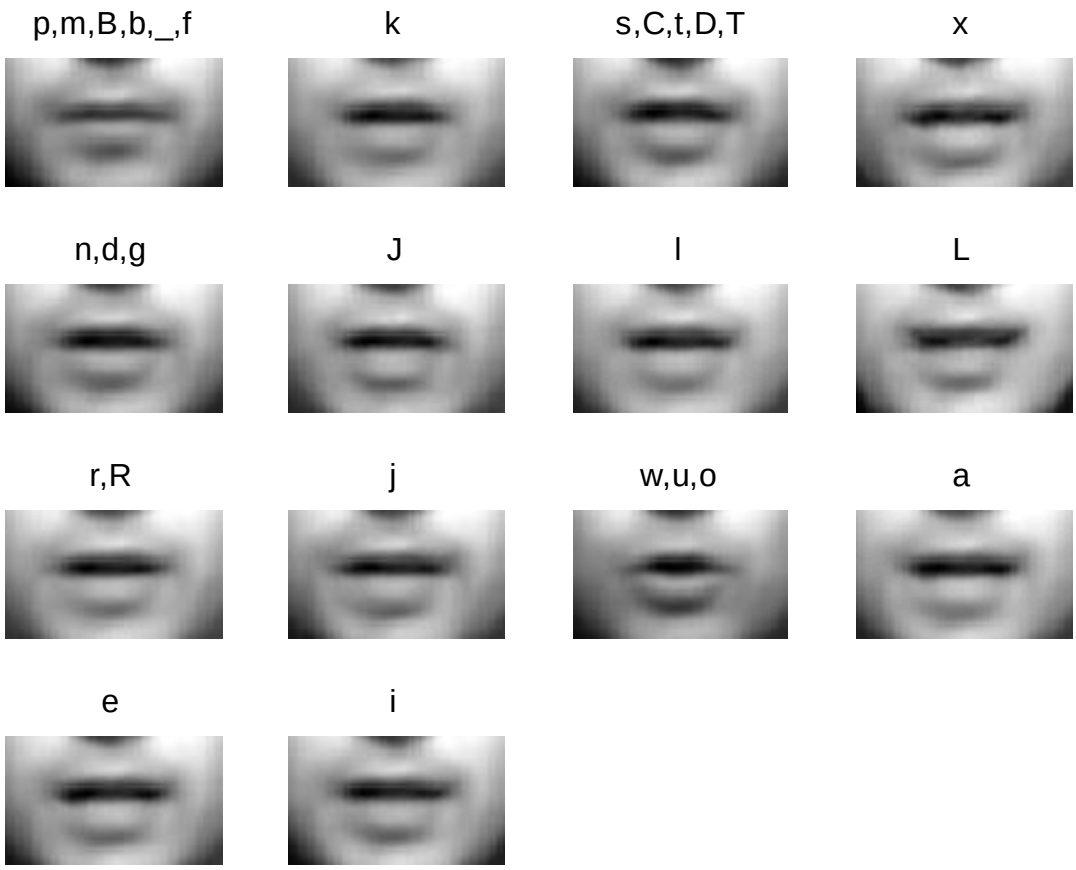
| | Probabilidad de acertar el fonema "_" (silencio) | Probabilidad de acertar los fonemas restantes | Probabilidad de acertar todos los fonemas |
|---|---|---|---|
| Vocabulario 1 | 0.388 | 0.118 | 0.230 |
| Vocabulario 2 | 0.386 | 0.164 | 0.258 |
| Vocabulario 3 | 0.387 | 0.166 | 0.247 |
| Vocabulario 4 | 0.687 | 0.194 | 0.394 |
| Vocabulario 5 | 0.648 | 0.193 | 0.382 |
| Vocabulario 6 | 0.387 | 0.175 | 0.262 |

**Tabla 13**

| | Probabilidad de acertar el visema "_"(silencio) | Probabilidad de acertar los visemas restantes | Probabilidad de acertar todos los visemas |
|---|---|---|---|
| Vocabulario 1 | 0.425 | 0.577 | 0.552 |
| Vocabulario 2 | 0.386 | 0.346 | 0.396 |
| Vocabulario 3 | 0.387 | 0.338 | 0.383 |
| Vocabulario 4 | 0.717 | 0.235 | 0.448 |
| Vocabulario 5 | 0.705 | 0.258 | 0.476 |
| Vocabulario 6 | 0.387 | 0.298 | 0.369 |

En las tablas anteriores, podemos observar como vuelven a ser los vocabularios 4 y 5 los que ofrecen probabilidades más altas en el acierto de fonemas, confirmando los resultados anteriores. Si nos fijamos son capaces de reconocer muy bien el silencio con cerca de un 70% de acierto y de reconocer cualquier otro fonema con aproximadamente un 20%. Finalmente, confirmamos que el vocabulario 5 sigue siendo aquel que mejor estima las frases pronunciadas en el vídeo y, por lo tanto, el vocabulario final.

Seguidamente mostramos cada uno de los visemas del vocabulario final calculados como la media entre las imágenes ROI de toda la secuencia de entrenamiento.

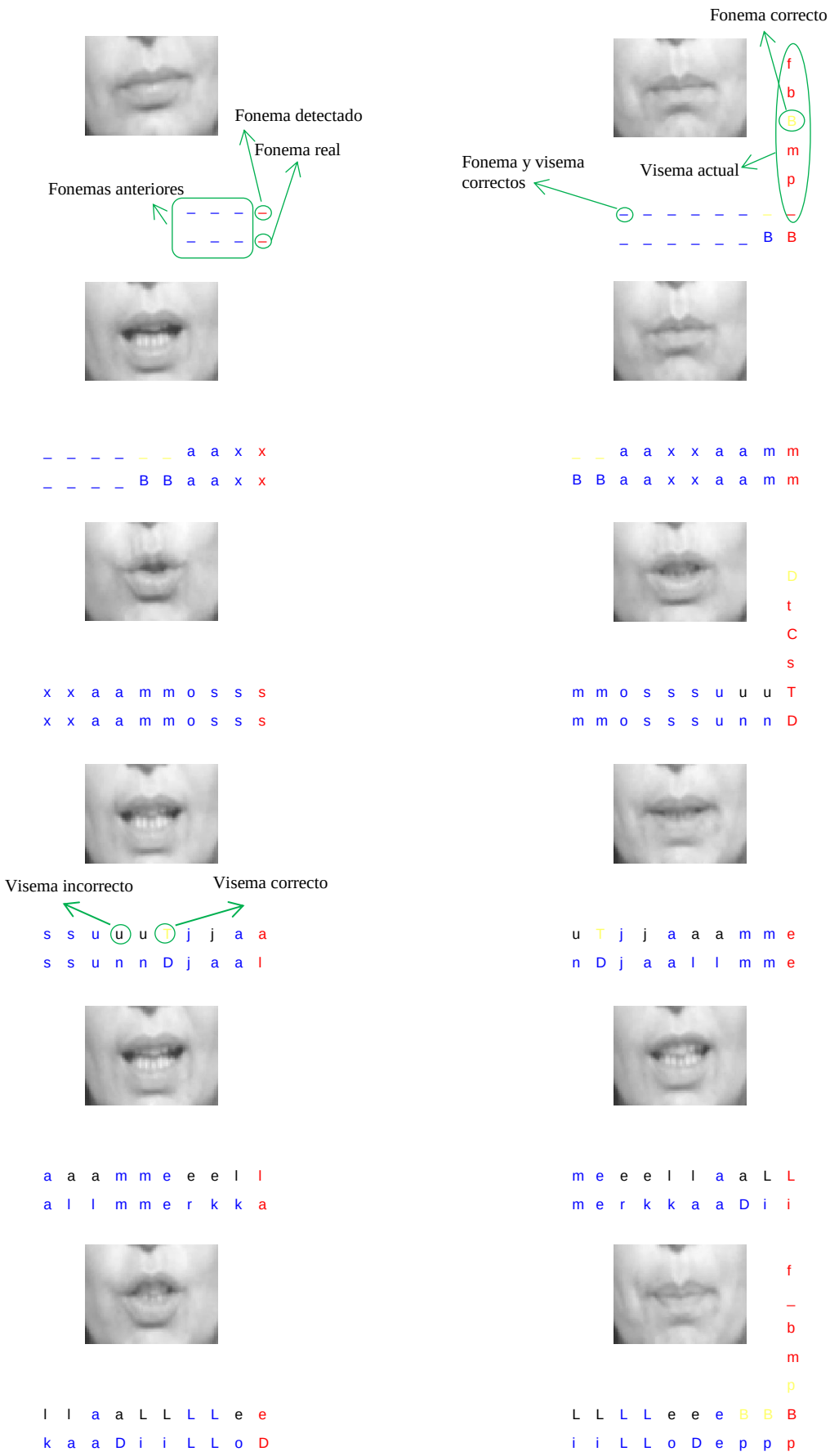



**XXXVIII. Vocabulario de visemas final**

## 5.2. Ejemplo de reconocimiento automático visual del habla

En este apartado, mostramos un ejemplo del resultado de nuestro sistema de reconocimiento automático visual del habla.

Se muestra una secuencia de 12 figuras correspondientes a un vídeo de entrenamiento en el que el usuario dice: “Bajamos un día al mercadillo de Palma”. En cada figura aparece la ROI, la secuencia fonética reconocida por el sistema y la secuencia fonética real del vídeo. Concretamente, podemos observar el fonema pronunciado en ese instante (en color rojo) y los 9 fonemas anteriores. Cada figura ha sido capturada cada 4 fotogramas. En las figuras, hemos indicado en rojo el fonema detectado actual, pero además si el fonema actual no se correspondía con el fonema real, también hemos mostrado en rojo todos los fonemas contenidos en el visema correspondiente a ese fonema. En el caso de que en ese visema se encontrase el fonema real lo hemos marcado en amarillo. Para los fonemas anteriores hemos establecido diferentes colores dependiendo de la salida que había tenido el sistema, es decir, cuando tanto el visema como el fonema eran correctos hemos marcado el fonema en azul; cuando solamente el visema era correcto lo hemos marcado en amarillo; y cuando la salida no se correspondía con el visema real lo hemos marcado en color negro.




**XXXIX. Ejemplo de secuencia de salida del sistema**

# 6. Conclusiones

Una vez analizados los resultados, hemos podido verificar varias de las hipótesis expuestas en la introducción.

La base de datos utilizada nos ha permitido entrenar el clasificador de manera que éste ha sido capaz de interpretar con bastante precisión los tiempos de silencio (65% de acierto), pero no así los fonemas relacionados con el habla propiamente (20% de acierto). Sería interesante realizar otro estudio paralelo en el cual la base de datos contenga información únicamente del habla para comparar los diferentes resultados obtenidos, y quizá poder formar un sistema compuesto por ambos resultados.

También hemos podido estudiar cómo el tamaño del vocabulario y la clasificación de fonemas en visemas han sido fundamentales para obtener resultados útiles que nos permitan estimar el habla a partir exclusivamente de vídeos. Hemos llegado a la conclusión de que un vocabulario de 14 visemas nos permite descifrar hasta el 40% de los fonemas.

Finalmente, hemos llegado a la conclusión de que con los métodos empleados (descriptores SIFT estáticos y HMM) no hemos sido capaces de interpretar el habla de las personas utilizando únicamente los recursos visuales. Por tanto, la información que hemos podido extraer de los recursos visuales sería más útil en situaciones en las que el audio se encuentre corrupto y el vídeo sea capaz de aportar o reforzar la información perdida. Por ejemplo, con la presencia de ruido en el audio, el vídeo puede evitar que tratemos de reconocer habla en ausencia de ella. Además, aunque el vídeo tenga menor precisión para reconocer habla que el audio, puede ayudar a desambiguar similitudes entre fonemas como “p” y “k”, “b” y “d” o “b” y “g”. Por ejemplo, en los vídeos que hemos testeado el valor de confusión entre “b” y “d” y “b” y “g” es 0 y entre “k” y “p” es 0.0012.

Como continuación de este proyecto sería interesante realizar una serie de estudios. Fijándonos en la base de datos utilizada tenemos que destacar que en ella los usuarios hablaban con naturalidad, es decir, no se les había pedido que remarcasen la pronunciación. Por lo tanto, sería interesante estudiar, con una base de datos en la cual los usuarios vocalizasen muy bien lo que van a decir, cuáles serían los resultados que lograríamos en el caso ideal de que todos los usuarios pronunciasen muy bien.
Otro punto a estudiar sería la técnica de extracción de características. En nuestro caso hemos utilizado descriptores SIFT de características estáticos, pero no hemos probado de extraer y analizar la información del movimiento, tal y como también se hace en la literatura; y tampoco hemos probado de extraer la información estática utilizando otro tipo de descriptores.
Consideramos importante también, una vez tenemos el sistema de reconocimiento del habla, añadir reglas semánticas que nos permitan dar mayor coherencia al habla reconocida.
Además sería interesante finalmente poder continuar este proyecto creando un sistema AV-ASR mucho más robusto que nos permita interpretar el habla en la mejor medida posible.

## Bibliografía


Lowe, D.G. (2004). Distinctive image features from scale-invariant keypoints. *Internation Journal of Computer Vision*, 91-110.

Vedaldi, A. (2007). An open implementation of SIFT detector and descriptor. California: UCLA CSD. Technical Report 070012

Aleksic, P. S.; Williams, J. J.; Wu, Z. and Katsaggelos, A. K. (2002). Audio-Visual Continuous Speech Recognition Using MPEG-4 Compliant Visual Features. *EURASIP Journal on Aplied Signal Processing*, 1213-1227.

Zhou, Z.; Zhao. G.; Hong, X. and Pietikäinen, M. (2014). A review of recent advances in visual speech decoding. *Image and Vision Computing*, 590-605.

Cootes, T. F. and Taylor, C. J. (2004). Statistical models of Appearance for Computer Vision. Manchester M13 9PT, U.K.

Sukno, F. M.; Ordás, S.; Butakoff, C.; Cruz, S. and Frangi, A. F. (2007). Active Shape Models with Invariant Optimal Features: Application to Facial Analysis. *IEEE Transactions on pattern analysis and machine intelligence*, vol. 29, no. 7.

Wells, J. C. (1997). SAMPA computer readable phonetic alphabet. Handbook of standards and resources for spoken language systems 4.

Boersma, P. and Weenink, D. (2015) Praat – doing phonetics by computer. Recovered 5 June 2015, from http://www.praat.org/

Goldman, J. Ph. (2011). EasyAlign: an automatic phonetic alignment tool under Praat Proceedings of InterSpeech, Firenze, Italy. Recovered 5 June 2015, from http://latlcui.unige.ch/phonetique/easyalign.php

Llisterri, J. and Mariño, J. B. (1993). Spanish adaptation of SAMPA and automatic phonetic transcription. Catalunya, España ESPRIT PROJECT 6819

Yau, W.C.; Kumar, D. K. and Weghorn, H. (2007). Visual Speech Recognition Using Motion Features and Hidden Markov Models.

Seymour, R.; Stewart, D. And Ming, J. (2007). Comparison of Image Transform-Based Features for Speech Recognition in Clean and Corrupted Videos. EURASIP Journal on Image and Video Processing, vol. 2008.

Sahu, V. and Sharma, M. (2013). Result Based Analysis of Various Lip Tracking Systems. *International Conference on Green High Performance Computing.*

Cappelletta, L. and Naomi, H. (2011). Viseme definitions comparison for visual-only speech recognition. *European Signal Processing Conference*.

Petrushin, V.A. (2000) . Hidden Markov Models: Fundamentals and Applications.

Potamianos, G.; Neti, C.; Gravier, G; Garg, A. and Senior, A.W. (2003). Recent Advances in the Automatic Recognition of Audiovisual Speech.

Rabiner, R. L. (1989) A Tutorial on Hidden Markov Models and Selected Applications in Speech Recognition.

Ortega, A.; Sukno, F.; Lleida, E.; Frangi, A.; Miguel, A.; Buera, L. and Zacur, E. (2004) AV@CAR: A Spanish multichannel multimodal corpus for in-vehicle automatic audio-visual speech recognition. *In IV Inernational Conference on Language Resources and Evaluation*, vol. 3, 763-767

Antonakos, E.; Roussos, A. and Zafeiriou, S. (2015). A Survey on Mouth Modeling and Analysis for Sign Language Recognition. *Proceedings of IEEE International Conference on Face and Gesture*. (in press)

Gimson, A.C. (2008) The Pronunciation of English. Cruttenden, A., ed., Hodder.

Viola, P. and Jones, M.J. (2004). Robust Real-Time Face Detection. *International Journal of Computer Vision*, 137-154.

Kroon, D.J. (2010). Viola Jones Object Detection. Recovered 5 June 2015, from http://www.mathworks.com/matlabcentral/fileexchange/29437-viola-jones-object-detection/content/ObjectDetection.m

Goldschen, A.; Garcia, O.; and Petajan, E. (1994). Continuous optical automatic speech recognition by lipreading. vol. 1, pp. 572 –577.

Jeffers, J. and Barley,M. (1971). Speechreading (Lipreading). Charles C Thomas Pub Ltd.