



Treball fi de carrera

**ENGINYERIA TÈCNICA EN
INFORMÀTICA DE SISTEMES**

**Facultat de Matemàtiques
Universitat de Barcelona**

**Reconocimiento automático de lenguaje
de signos:Lenguaje ASL**

Enrique Antón López

Directores:

Sergio Escalera

Alberto Escudero Pardo

Realitzat a: Departament de

Matemàtica Aplicada i

Anàlisi. UB

Barcelona, 20 de juliol de 2009

Agradecimientos

A mis padres , a mi hermana Tere, a mi cuñado Jose, a mis sobrinos Alejandro y Gonzalo y sus curiosas historias sobre el Ratoncito Perez , mi familia del pueblo que es demasiado extensa como para nombrarla y sobre todo a mis primas Isa y Montse y a mis tios Obdulia y Ricardo y también a Domingo y a los peques. También al amigo Cobretti, que supo ser un buen amigo en los momentos duros y al Jordan que no se donde se mete y al Granja.Al Pol que ya no esta entre nosotros, a mi amiga Ana que es siempre leal y a sus padres , a Cri Cri, a Jaime y Carlos Herrera y todos los amigos de la asociación. A Andres, a Magda, a Silvia, a Belinda y a sus padres que han sido como una segunda familia.

Y como no a todos los profesores que he tenido en el transcurso de la carrera y sobre todo a Sergio Escalera y Alberto Escudero, que me han dirigido el proyecto con mucha paciencia.

Y tambien a mi gato Claudio que no me ha dado mucha guerra.

Resumen

La interacción hombre-máquina es un problema complejo desde el punto de vista de la inteligencia artificial. El reciente uso de sensores pretende facilitar dicha interacción.

En particular, en este proyecto nos centramos en la visión artificial como mecanismo de simulación del sentido visual humano. El sistema que se presenta será capaz de interpretar los símbolos del alfabeto del lenguaje de signos (en concreto el lenguaje de signos americana ASL) y mostrar el resultado por pantalla, .

Se utilizan técnicas de procesamiento de imágenes para la segmentación de regiones de interés y proyección de contornos para la extracción de características. La clasificación final de los signos viene dada por el algoritmo DTW (Dinamic Time Warping) .

Como resultado, el sistema muestra un alto rendimiento de clasificación en un entorno no controlado clasificando más de 30 clases.

Resum

La interacció home-màquina és un problema complex des del punt de vista de la intel·ligència artificial. El recent ús de sensors pretén facilitar aquesta interacció.

En particular, en aquest projecte ens centrem en la visió artificial com a mecanisme de simulació del sentit visual humà. El sistema que es presenta serà capaç d'interpretar els símbols de l'alfabet del llenguatge de signes (en concret del llenguatge de signes americà ASL) i mostrar el resultat per pantalla.

S'utilitzen tècniques de processament d'imatges per a la segmentació de regions d'interès i projecció de contorns per a l'extracció de característiques. La classificació final dels signes ve donada per l'algoritme DTW (Dinamic Time Warping).

Com a resultat, el sistema mostra un alt rendiment de classificació en un entorn no controlat classificant més de 30 classes.

Abstract

Computer Vision Interaction is a complex problem from the artificial intelligence point of view. The recent use of sensors tries to help that interaction. In particular, this project focuses on artificial vision as a simulation mechanism of the human vision system.

This system is able to recognize symbols from the sign language (in particular, ASL language) and show the result.

We use image processing techniques in order to perform segmentation of regions of interest and contour projection to deal with the feature extraction problem. The final classification is obtained by means of Dynamic Time Warping (DTW).

As a result, the system shows high performance classifying more than 30 sign categories from non-controlled environments.

Indice

Agradecimientos.....	Pagina 3
Resumen.....	Pagina 4
1.Introducción	Pagina 7
1-1.Problema/Motivación.....	Pagina 7
1-2.Antecedentes.....	Pagina 7
1-3.Contribución.....	Pagina 10
1.4.Organización.....	Pagina 11
2- Metodología.....	Pagina 12
2.1-Segmentación.....	Pagina 13
2.2-Extracción de características.....	Pagina 21
2.3-Clasificación.....	Pagina 24
2.4.Diseño y análisis de costes.....	Pagina 26
3-Resultados.....	Pagina 28
3.1.Datos.....	Pagina 28
3.2-Métodos de obtención.....	Pagina 29
3-3. Validación.....	Pagina 30
3-4. Discusiones.....	Pagina 32
4.conclusión y lineas futuras.....	Pagina 33
5.Anexos.....	Pagina 34
6.Bibliografía.....	Pagina 35
Disco compacto.....	Pagina 37

Introducción

1-1.Problema/Motivación

"La inteligencia artificial tiene como objetivo imitar por medio de máquinas las actividades relacionadas a la inteligencia humana" (Penrose) .Hoy en día los sistemas informáticos y los algoritmos permiten a los sistemas integrados tomar decisiones con cierta "autonomía".

Este hecho junto con los nuevos medios de interacción entre personas y máquinas (Human Computer Interacción) hacen posible crear sistemas que puedan complementar o ayudar a un usuario en diversas actividades.

La disciplina de estudio de la interacción hombre-maquina (Human Computer Interaction) estudia los diferentes medios de comunicación entre las personas y los ordenadores. Los medios de comunicación generalmente corresponden a sensores que imitan nuestros sentidos. Los nuevos avances en tecnología informática hacen posibles nuevos medios de interacción con sistemas informáticos aparte de los tradicionales, como reconocimiento de voz, capturas de movimiento, etc. En particular, en este proyecto nos centraremos en la disciplina de la visión artificial, la cual se basa en el análisis de patrones con el objetivo de simular la visión humana.

Del mismo modo que para los oyentes (que son la gran mayoría de los usuarios de ordenadores) existen medios de reconocimiento de voz, existe un vacío en cuanto a medios de interacción apropiados para la comunidad sorda (aparte del tradicional teclado).

Uno de los posibles medios que no necesiten del sonido y que permitan comunicarse con el ordenador puede ser la cámara de vídeo (que actualmente llevan casi todos los ordenadores integrada como webcam).

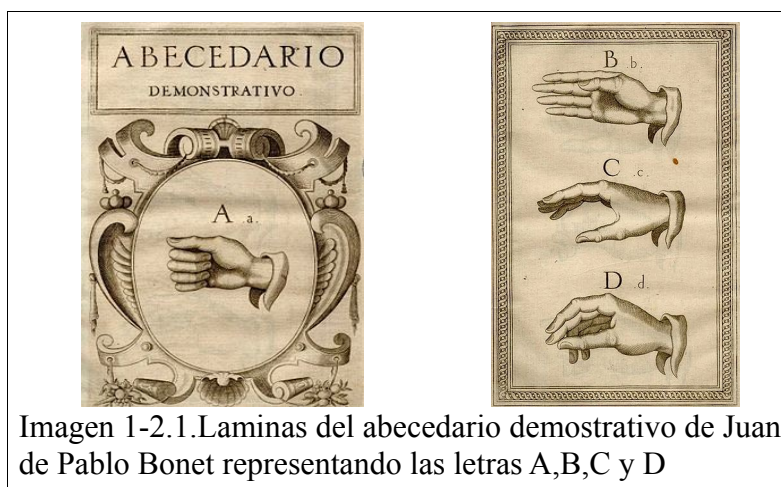
1-2.Antecedentes

Desde hace mucho tiempo los sistemas de signos eran utilizados no sólo por las personas sordas. Entre algunas tribus indias del norte de América eran utilizados para facilitar la comunicación debido a la cantidad de lenguas diferentes que utilizaban.

Entre los primeros estudiosos de los sistemas de signos como medio de comunicación para personas sordas estuvieron [Juan de Pablo Bonet](#),(1573-1633)

pedagogo español que en 1620 escribió “*Reducción de las letras y Arte para enseñar á hablar los Mudos*” [12], libro en el que mostraba un método de comunicación mediante alfabeto de signos para las personas sordas (Imagen 1-2.1.).

A partir del alfabeto publicado por Juan de Pablo Bonet, [Charles-Michel de l'Épée](#) (1712-1789) publica un alfabeto en el que se basan los alfabetos de signos que se usan en la actualidad. Dicho alfabeto fue publicado por su sucesor el [abate Sicard](#) en la forma de “Diccionario general de Signos”.



Otro personaje importante en el campo de la lengua de signos fue Lorenzo Hervas y Panduro, filólogo y lingüista jesuita (1735-1809), que en 1795 escribió “Escuela española de sordomudos, o Arte para enseñarles a escribir y hablar el idioma español” [13] donde hacía un estudio del lenguaje de signos que utilizaban las personas hasta ese momento llamadas “mudas”. Fue en este texto donde este religioso humanista hizo un análisis de este lenguaje y lo equiparaba a las demás lenguas habladas.

La lengua de signos para el idioma Inglés Americano (American Sign Language, ASL) (Imagen 1-2.2., 1-2.3.) es la lengua de signos utilizada en Estados Unidos, el norte de México y la zona anglófona de Canadá. Tiene un origen independiente de la lengua de signos inglesa.

En 1817 Thomas Gallaudet abrió una escuela de sordos en Connecticut (Estados Unidos) donde sentó las bases para la ASL, tomando como referencia la lengua de signos francesa y algunos signos utilizados por las tribus indias de Norteamérica. En 1965

William Stokoe, en el libro “A Dictionary of American Sign Language on Linguistic Principles”[1] describió una gramática precisa para el lenguaje de signos americano.



Hasta ahora se habían hecho intentos de síntesis de la lengua de signos por medio de sistemas informáticos. Los primeros intentos fueron por parte de Schantz y Poizner [2] en 1982 mediante la animación de un brazo esquemático que simulaba la lengua de signos americana. Se han hecho otros trabajos, entre ellos el de Alonso, de Antonio, Fuentes y Montes en 1995 [3], en que se hacía la síntesis en tres dimensiones de una mano que deletreaba la lengua de signos española (LSE).

Entre los trabajos a destacar en el campo que nos interesa, la utilización de visión artificial en el reconocimiento de acciones delante de una cámara existen diversos métodos. Entre ellos están los que se basan en la utilización de guantes [4][5] o los que utilizan la visión.

En los métodos basados en visión existen variantes como detección de las manos mediante el color de la piel [6][7], otros como detección de movimiento [8][9] o también detección de bordes [10].

El elemento común de los métodos anteriormente expuestos es la obtención de un modelo de color de la piel extraído a partir de la cara [8][11] o de un histograma de color de piel fijado con anterioridad [9][10][6].

1-3.Contribución

En este proyecto se ha diseñado un prototipo que permite identificar, uno por uno, cada símbolo del alfabeto de signos americano para personas sordas.

Se pretende iniciar un camino para facilitar la comunicación entre oyentes y personas sordas. Nuestro sistema deberá poder interpretar los símbolos del alfabeto ASL (American Sign Language) y dar su significado.

El programa interpretara los símbolos utilizando una cámara de video integrada en el sistema mediante la cual se realizaran capturas periódicas de los signos que haga el usuario e imprimirá estos de forma consecutiva por pantalla.

1.4.Organización

El contenido de la memoria se estructura en varios apartados:

En el apartado 2. Metodología se explican los pasos a seguir para el procesamiento de las capturas de video.

En el subapartado 2.1 Segmentación se explican los pasos para obtener una imagen de la que podamos extraer características.

En el subapartado 2.2 Extracción de características se explican los pasos y algoritmos necesarios para extraer las características que definen a la imagen

En la ultima parte del apartado de Metodología, Clasificación se explica el método utilizado para realizar la clasificación y que nos permite determinar la clase a la que pertenece la captura que realizamos.

En el apartado 3 Resultados se explican los métodos de obtención de resultados, las variables optimas para obtener estos resultados. En el subapartado 3.3 Validación se muestran los resultados al efectuar una prueba con un texto y los resultados obtenidos

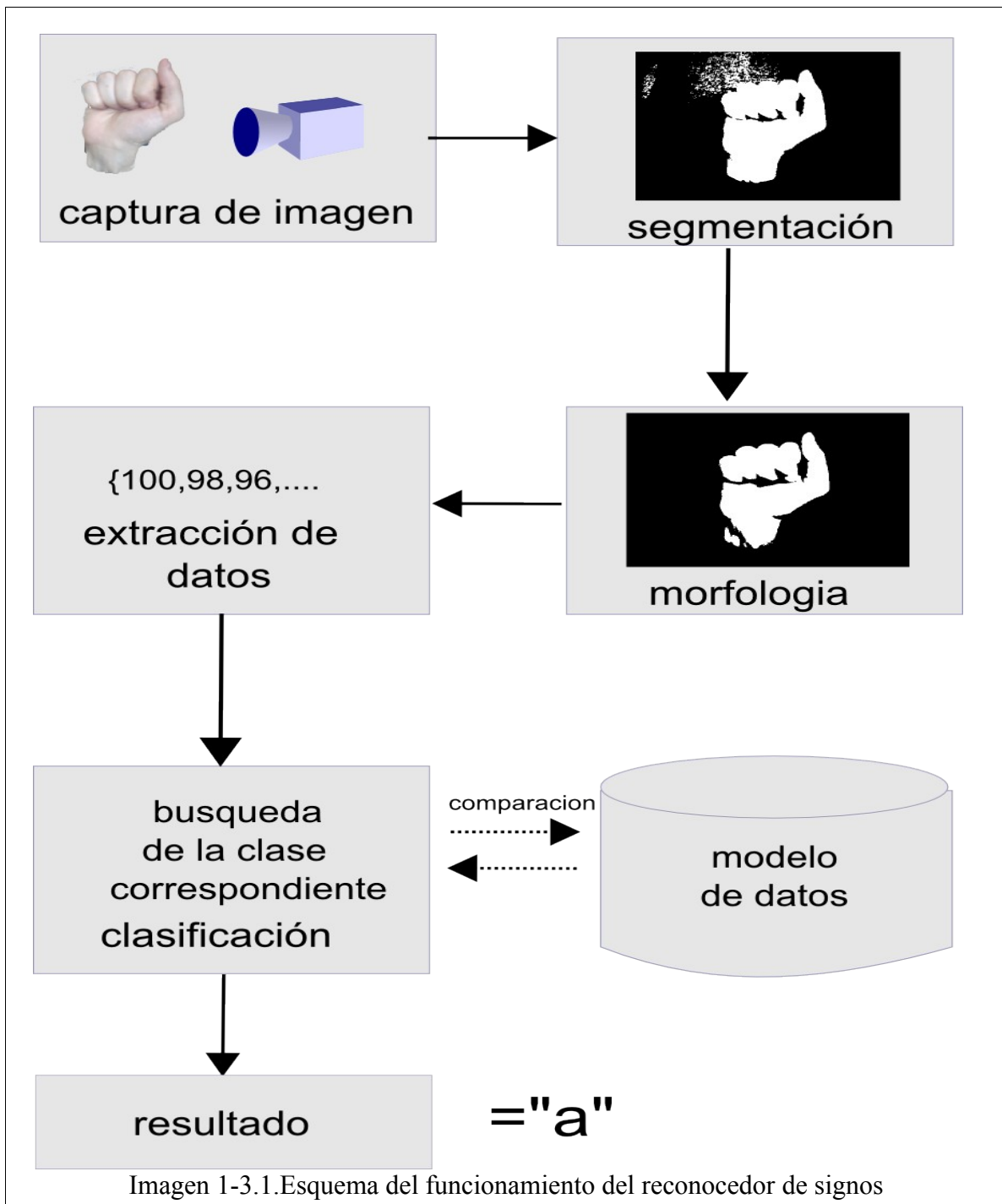
En el apartado 4 se muestran las conclusiones finales sobre la realización del proyecto y posibles aplicaciones y lineas de futuro

En el apartado 5 se hace un inventario de los anexos que se adjuntan al proyecto.

2-Metodología

El sistema implementado se compone de las siguientes etapas(Imagen 1-3.1.):

- 1-Obtención de imágenes a partir de capturas de video.
- 2-Filtraje y segmentación de las imágenes para su posterior procesamiento.
- 3-A partir de estas imágenes se obtendrán una serie de datos característicos.
- 4-Clasificación de los datos con la clase correspondiente.



A continuación se describe en detalle cada uno de los módulos del sistema:

2.1-Segmentación

Sobre la imagen obtenida realizamos un proceso de segmentación que permitirá una mejora en el posterior proceso de extracción de características. La segmentación consiste en extraer una zona determinada de la imagen según diferentes criterios, como por ejemplo su color e intensidad del nivel de gris (Imagen 2-1.1).

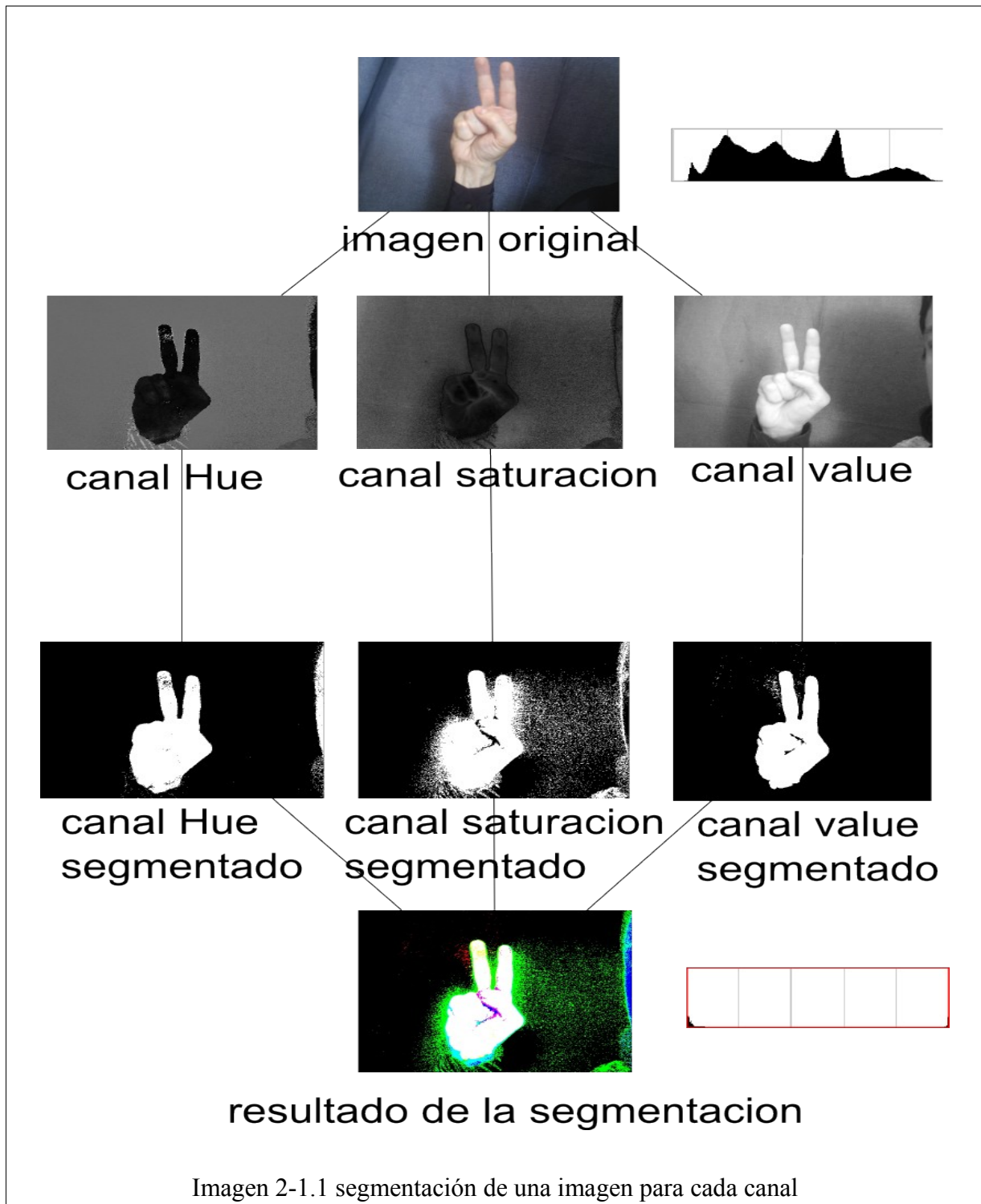


Imagen 2-1.1 segmentación de una imagen para cada canal

Para este proceso hemos considerado que nos sería más eficiente trabajar sobre el espacio de color HSV (Hue, Saturation, Value – Tonalidad, Saturación, Valor)(Imagen 2.1.4, 2.1.5).

Por lo general las imágenes en color se definen por el espacio de color RGB (red, green , blue) (Imagen 2.1.2, 2.1.3). Esto quiere decir que cada píxel (punto de la imagen) tendría un color determinado por la cantidad de cada uno de los colores que le correspondan en cada canal, de la misma forma en que se mezclan en una paleta de colores. La intensidad de cada color (la medida en que se acerca éste al negro o al blanco vendría determinada por la cantidad de cada uno de los colores que pertenezcan a la mezcla)

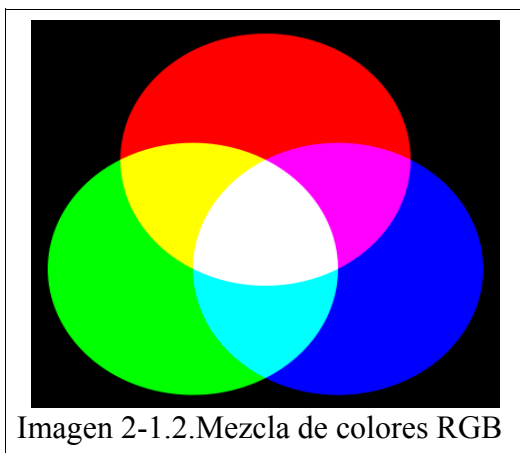


Imagen 2-1.2.Mezcla de colores RGB

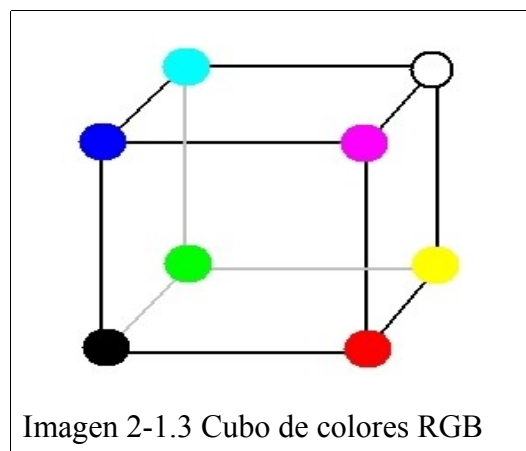


Imagen 2-1.3 Cubo de colores RGB

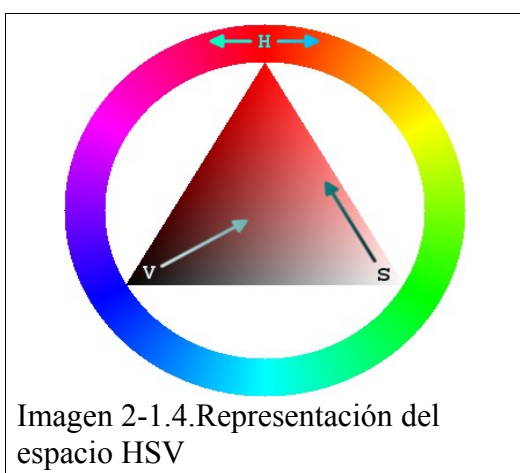


Imagen 2-1.4.Representación del espacio HSV

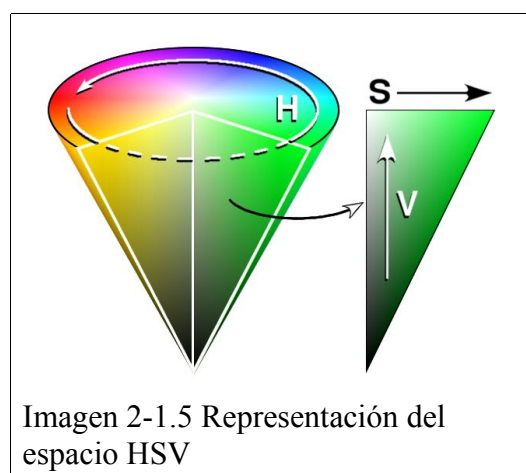
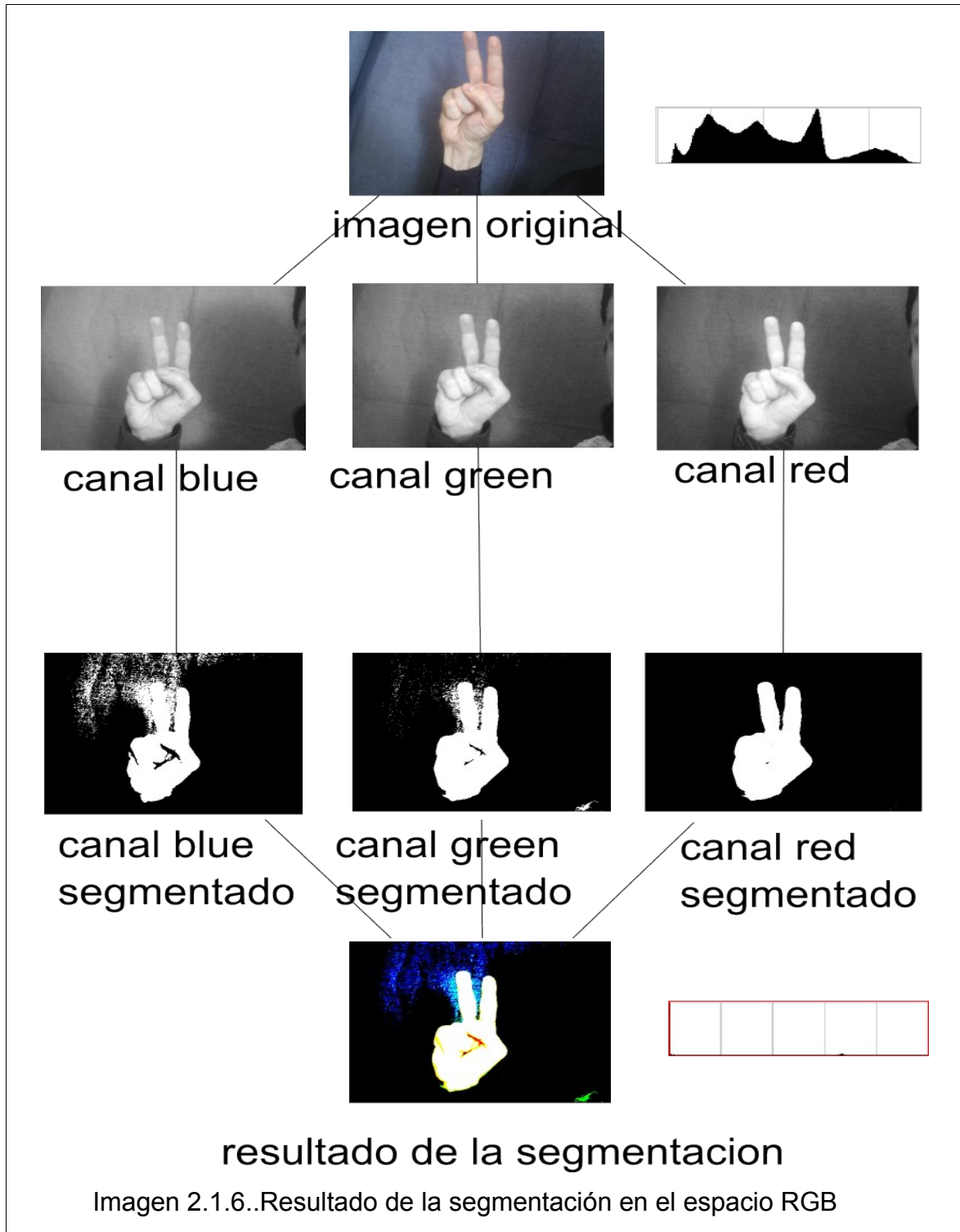


Imagen 2-1.5 Representación del espacio HSV

El modelo RGB puede ocasionar problemas a la hora de realizar la segmentación de las imágenes (Imagen 2.1.6.). Dicho modelo necesitaría observar un conjunto de restricciones que harían que el sistema fuera ineficiente.



Por ejemplo las condiciones de luminosidad de la escena en que se vayan a realizar las tomas son la mayoría de las veces difíciles de controlar. En entornos con diferente luz habría que realizar diferentes ajustes en los valores que se usan al hacer la

segmentación para cada canal.

Sin embargo, realizando la segmentación en el espacio HSV, las condiciones de luz no afectan al resultado.

El cambio entre espacios de colores RGB-HSV se observa en la figura 2.1.7.

$$\begin{aligned}M &= \text{Max}(R, G, B) \\m &= \text{min}(R, G, B) \\H_1 &= \arccos \left[\frac{\frac{1}{2}((R-G)+(R-B))}{\sqrt{(R-G)^2+(R-B)(G-B)}} \right] \\H &= H_1, \text{ si } B \leq G \\H &= 360^\circ - H_1, \text{ si } B > G \\S &= \frac{M-m}{m}, V = \frac{M}{255}\end{aligned}$$

Imagen 2.1.7. Formula de transformación RGB->HSV

Este modelo consta de tres canales:

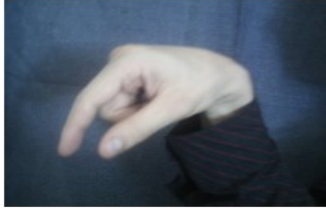
- Canal Hue (Tonalidad): Este canal determina el color del punto, que puede encontrarse en el rango que va del rojo, verde, azul y los colores intermedios, sería el color puro sin ningún tipo de degradación (Imagen 2.1.9).
- Canal Saturation (Saturación): La saturación es un valor que va desde el mínimo (blanco) a un máximo (que sería el color básico del punto determinado por el canal hue) y determina el nivel de gris del punto (Imagen 2.1.10).
- Canal Hue (Brillo): El brillo es el valor que va desde el negro hasta el color puro (Imagen 2.1.11).

El resultado final se observa en la imagen 2.1.8.

Dado el espacio HSV, realizamos la segmentación para cada canal según el siguiente algoritmo:

```
mientras x < imagen->anchura:  
  mientras y < imagen->altura:  
    si ColorPixel(x,y) < umbral:  
      ColorPixel(x,y)=0  
    si ColorPixel(x,y) >= umbral:  
      ColorPixel(x,y)=max  
    fin mientras:  
  fin mientras:
```


imagen original



histogramas en escala logaritmica



segmentación

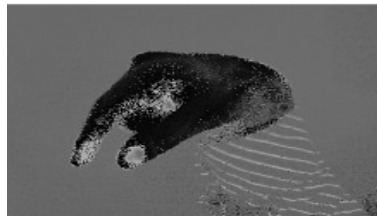


segmentación binaria

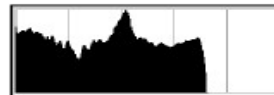


Imagen 2.1.8. Segmentación sobre una imagen

canal hue



histogramas en escala logaritmica



segmentación



segmentación binaria

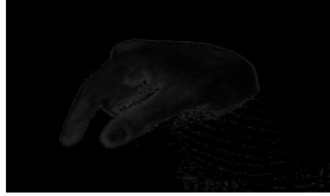


Imagen 2.1.9. Segmentación para el canal Hue

canal saturacion



segmentación



segmentación binaria



histogramas en
escala logaritmica



Imagen 2.1.10. Segmentación para el canal Saturación

canal value



segmentación



segmentación binaria



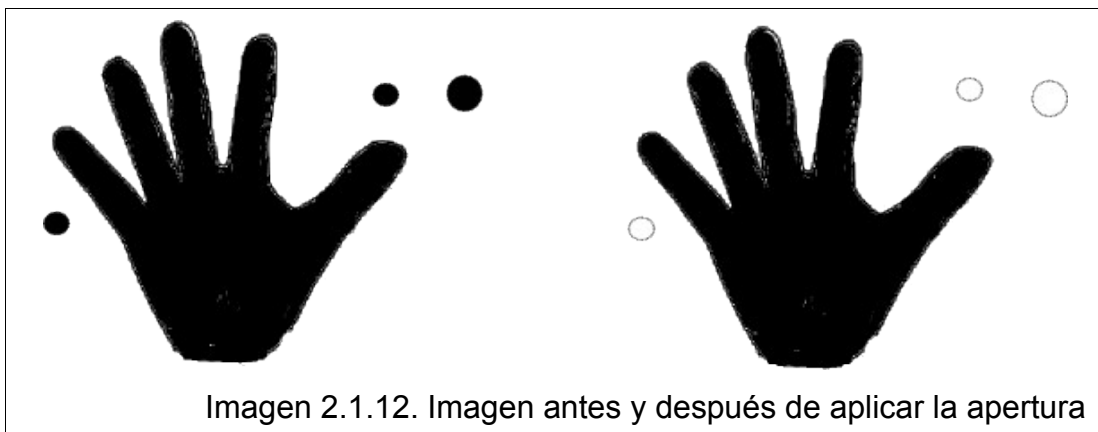
histogramas en
escala logaritmica



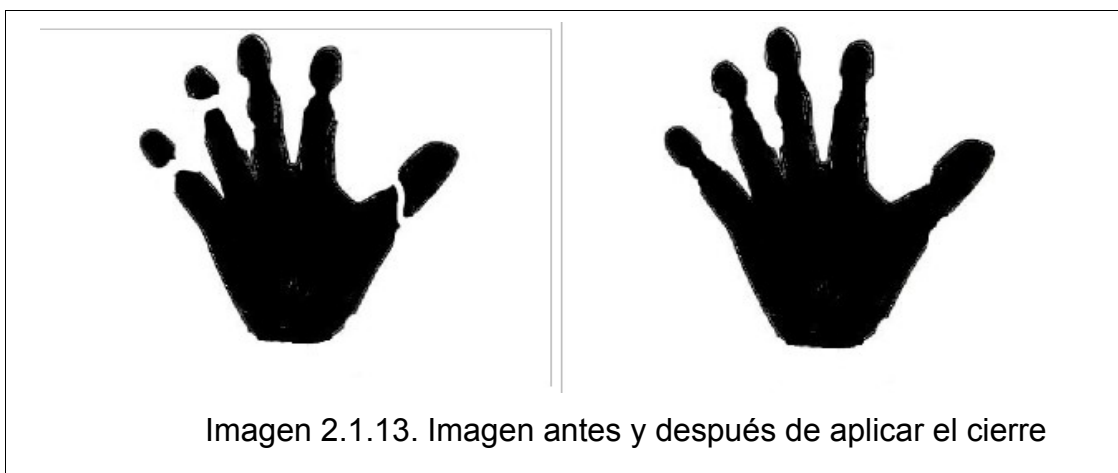
Imagen 2.1.11 Segmentación para el canal Value

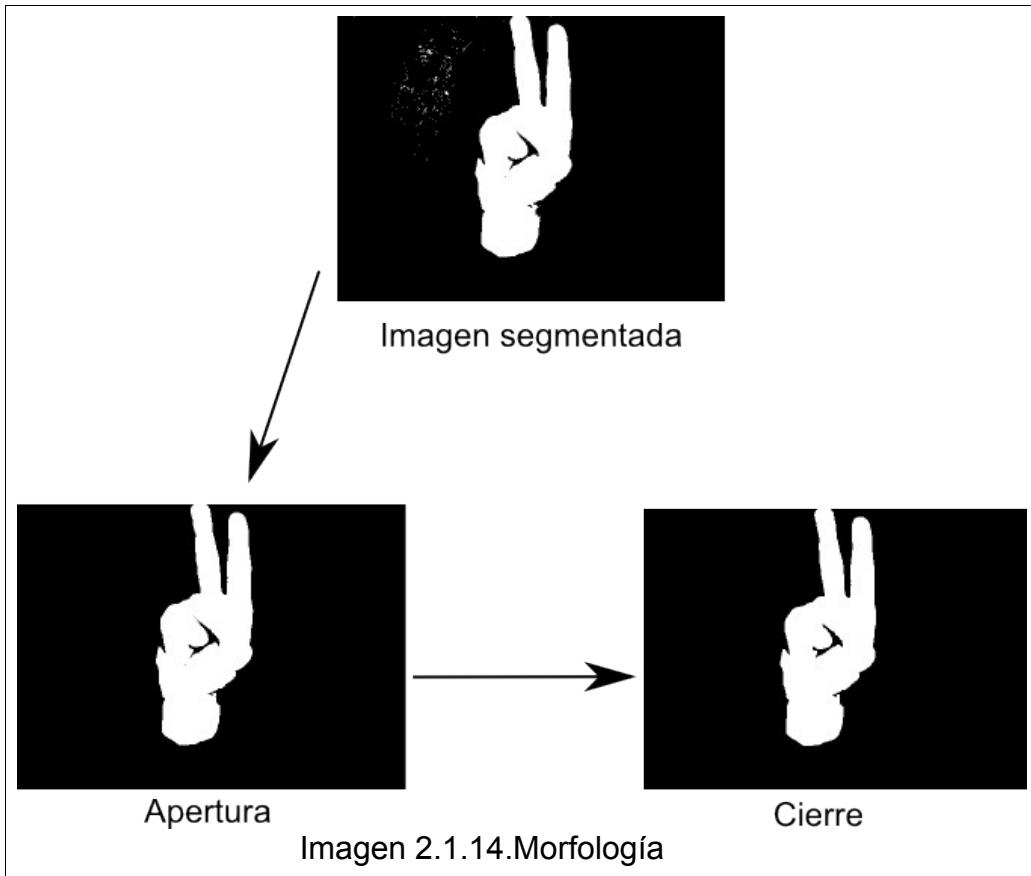
La imagen resultante debe contener solo el contorno, lo cual a veces no es posible ya que pueden quedar algunas pequeñas zonas que no pertenezcan a este, pero que pueden ocasionar problemas a la hora de extraer las características. Para eliminar estas zonas se utiliza un algoritmo de morfología que en un primer paso reduce todos los contornos de la imagen, con lo que se eliminarían las zonas mas pequeñas y seguidamente se aplicaría un segundo paso para retornar la imagen que nos interesa a su estado inicial(Imagen 2.1.14.).

El algoritmo de morfología mas adecuado para eliminar zonas pequeñas sin deformar excesivamente la imagen es la apertura. La apertura (Imagen 2.1.12)es una operación compuesta de erosión y dilatación .Con la erosión reducimos el contorno de la imagen hasta un punto determinado con lo que se elimina lo que no se desea, con la dilatación se vuelve a ampliar el contorno, esta vez sin las zonas eliminadas.



Con el algoritmo de cierre (Imagen 2.1.13)conectamos las zonas que hayan podido quedar desconectadas mediante el algoritmo de apertura. El cierre es la operación contraria a la apertura, primero se amplía el contorno y luego se reduce.



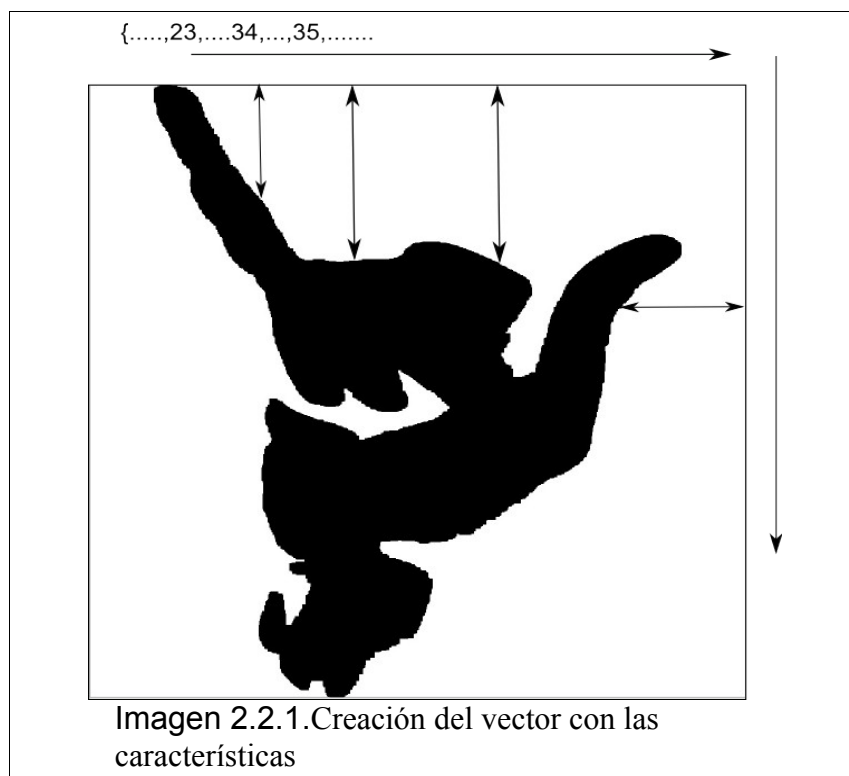


2.2-Extracción de características

Una vez obtenida una imagen segmentada que solo contiene dos valores, negro(valor 0) para las zonas de la imagen que quedan fuera del contorno de la mano y blanco(valor mayor que 0) para el contorno de esta, se necesita obtener una serie de valores que permitan diferenciar a una imagen de otra y poder decidir así que símbolo determina.

Entre las características que pueden definir a una imagen segmentada se encuentra la distancia de cada punto del contorno de la imagen respecto a un punto determinado.

En nuestro caso guardaremos una lista con las distancias desde los extremos de la imagen al punto perpendicular mas cercano perteneciente al contorno de la mano(Imagen 2.2.1.).



La creación del vector se efectúa según el siguiente algoritmo:

```
función creaVector(imagen,vector,tamañoVector):  
  contador=0,x=0,y=0,i=0;  
  mientras x < imagen->anchura:  
    mientras y < imagen->altura:  
      si ColorPixel(x,y) = 0:  
        contador=contador+1  
      sino:  
        fin mientras  
      fin si:  
      y=y+1  
    fin mientras:  
    y=0  
    vector[x]=contador  
    contador=0;  
    x++;  
  fin mientras:  
  x=imagen->anchura-1  
  mientras x < imagen->altura::  
    mientras x >= 0:  
      si ColorPixel(x,y) = 0:  
        contador=contador+1  
      sino:  
        fin mientras  
      fin si:  
      x=x-1  
    fin mientras:  
    x=imagen->anchura-1;  
    vector[imagen->anchura+y]=contador  
    contador=0;  
    y=y+1  
  fin mientras:  
  y==imagen->altura-1  
  
  mientras x >= 0:  
    mientras y >= 0:  
      si ColorPixel(x,y) = 0:  
        contador=contador+1  
      sino:  
        fin mientras  
      fin si:  
      y=y-1  
    fin mientras:  
    y==imagen->altura-1  
    vector[2*imagen->anchura+imagen->altura-x-1]=contador  
    contador=0  
    x=x-1  
  fin mientras:  
  x=0  
  mientras y >= 0:
```

```
mientras x < imagen->anchura:  
    si ColorPixel(x,y) = 0:  
        contador=contador+1  
    sino:  
        fin mientras  
    fin si:  
    x=x-1  
fin mientras:  
  
x=0  
vector[2*imagen->anchura+2*imagen->altura-y-1]=contador  
contador=0;  
y=y+1  
fin mientras:  
    retorna vector  
fin función:
```

2.3-Clasificación

Para la clasificación se necesitara guardar un conjunto de vectores con los que se pueda hacer la comparación después y poder determinar a que símbolo corresponde la imagen que estamos evaluando. Para guardar este conjunto de vectores se harán varias tomas para cada símbolo y se guardaran los vectores obtenidos en un fichero.

Para cada captura que se haga después se realizara una clasificación comparando el vector obtenido con los que se encuentran almacenados en el fichero.

Dicha comparación se efectúa entre este vector y cada uno de estos mediante un algoritmo de clasificación que compara la distancia entre estos.

Un algoritmo eficiente para efectuar esta clasificación es el DTW(Dinamic Time Warping).

El DTW compara la similitud entre dos secuencias. Tenemos $M = (M_1, \dots, M_m)$, que es un modelo de secuencia donde cada M_i es un vector de características. Tenemos $Q = (Q_1, \dots, Q_m)$, que es la secuencia que queremos clasificar. En nuestro caso, el vector de características consiste en la distancias de cada punto del contorno de la imagen al borde de esta, y para cada imagen se define un vector de características diferente. Para poder utilizar el algoritmo de clasificación fijamos el vector de características de manera que contenga las características de la mano, de la forma $Q=(x_1, \dots, x_n)$.

La trayectoria (*warping path*) W define una alineación entre M y Q . Formalmente, $W=w_1, \dots, w_T$, donde $\max(m,n) \leq T \leq m+n-1$. Cada $w_t=(i,j)$ especifica que el vector de características modelo M_i se corresponde con el vector de características Q_j . En este caso tenemos que w_t tiene dos dimensiones temporales (denotadas por i y j). La trayectoria está sujeta a un conjunto de condiciones:

•**Condiciones de límite:** $w_1=(1,1)$ y $w_T=(m,n)$. Esto requiere que en el inicio de la trayectoria debe corresponderse a la primera imagen de la secuencia modelo con la primera imagen de la secuencia a clasificar, y que finalice correspondiendo a la última imagen de la secuencia modelo y la última imagen de la secuencia a clasificar.

•**Continuo temporalmente:** Dados $w_t=(a,b)$ entonces $w_{t-1}=(a',b')$, donde $a-a' \leq 1$ y $b-b' \leq 1$. Esto restringe los pasos permitidos de la trayectoria a las posiciones cercanas en dos imágenes consecutivas de la secuencia.

•**Monótono temporalmente:** Dados $w_t=(a,b)$ entonces $w_{t-1}=(a',b')$, donde $a-a' \geq 0$ y $b-b' \geq 0$. Esto fuerza a la trayectoria de la secuencia a incrementar de forma monótona en dos imágenes consecutivas de la secuencia.

El funcionamiento del algoritmo se muestra en la imagen 2.3.1 Cada celda tiene un coste $C(i,j)$, obtenido de calcular la distancia Euclidiana entre el vector de características modelo (M) y el vector a clasificar (Q). Para obtener la distancia DTW de una celda, se suma el coste de cada celda el valor mínimo ente las celdas vecinas que cumplen las condiciones de continuidad y monotonía enunciadas anteriormente. Una vez se han calculado todas las distancias acumuladas, la distancia DTW es el valor obtenido en la celda $D(m,n)$. Las celdas coloreadas en negro muestran la trayectoria óptima, es decir, aquellas celdas que empezando desde la posición $D(m,n)$ nos han permitido obtener la distancia DTW más pequeña.

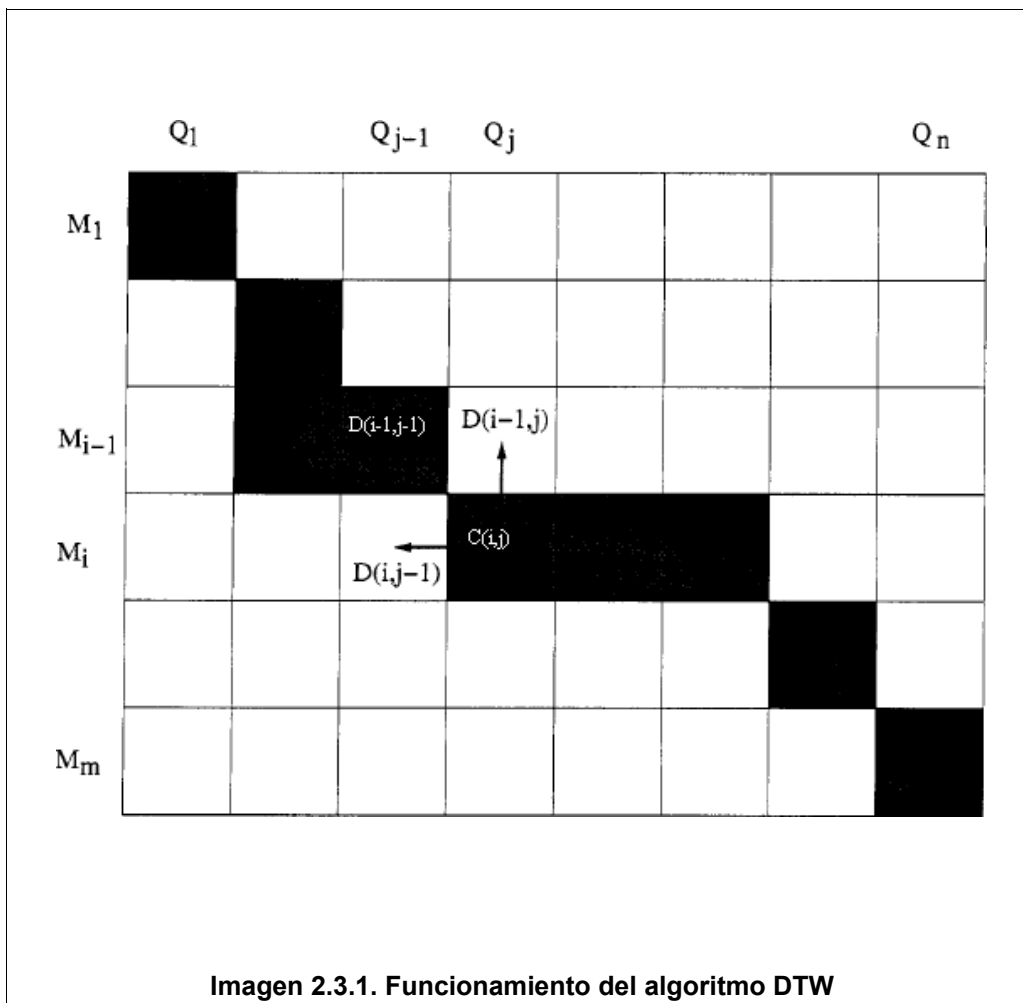


Imagen 2.3.1. Funcionamiento del algoritmo DTW

2.4. Diseño y análisis de costes

Antes de iniciar la fase de implementación se ha realizado la fase de diseño en la que se definen los módulos que van a tomar parte en el programa.

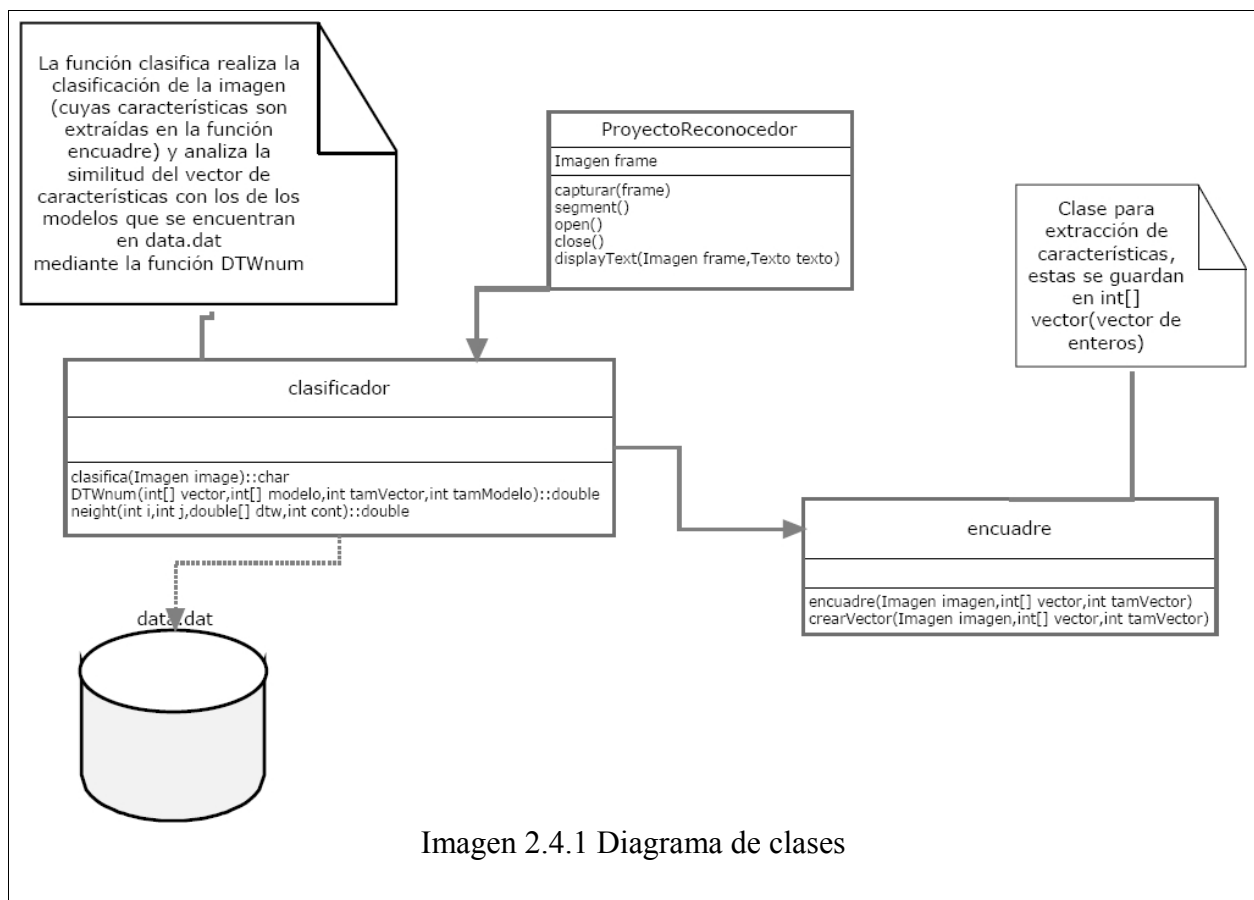
Los módulos se distribuyen de forma sencilla de la siguiente manera, como se puede ver en el diagrama de clases que se encuentra en la imagen 2.4.1.

En el modulo principal , Proyecto reconecedor se incluyen las funciones: capturar, que es la que llamamos en la función principal main y es donde se inicia el proceso de la imagen.

En esta función llamamos a la función segment, que realiza la segmentación y a las funciones open y close para realizar la morfología necesaria

Después de este punto se llama a la función clasifica que se encuentra en la clase clasificador .Esta a su vez se sirve de la clase encuadre donde se encuentran las funciones que nos facilitan la extracción de características .

Una vez conseguidas estas características se clasifica la imagen mediante la función DTWnum que hace servir el fichero “data.dat” como base de datos para los modelos a comparar.



En el siguiente diagrama de Gantt (Imagen 2.4.2) se da una estimacion del tiempo empleado en cada una de las fases de la ejecucion del proyecto

Number	Task	Resource	Start	End	Duration	2009				
						February	March	April	May	June
1	Introducción a OpenCV		15/2/2009	25/2/2009	7	■				
2	Análisis		25/2/2009	17/3/2009	14		■			
3	Diseño		17/3/2009	4/4/2009	14			■		
4	Estudio de alternativas		4/4/2009	5/5/2009	21			■		
5	Implementación		5/5/2009	23/5/2009	14				■	
6	Testeo		23/5/2009	12/6/2009	14					■

Imagen 2.4.2 Diagrama de Gantt

La implementación y compilacion del proyecto se ha realizado con la ayuda del entorno de desarrollo Microsoft Visual Studio 2008 Express Edition de descarga gratuita.

Las librerias utilizadas pertenecen a la biblioteca de funciones para procesamiento de imagenes y video OpenCV bajo licencia BSD.

Este documento se ha realizado con el procesador de textos de OpenOffice 3.0 bajo licencia BSD. Los graficos se han realizado con la ayuda del programa InkScape, y Gimp tambien de licencia BSD.

Todo esto se ha realizado en un portatil Intel Centrino 2 bajo Plataforma Windows Vista.

Se han invertido un total de 25 horas aproximadamente en concepto de tutoria. Se ha invertido tambien un total de 200 horas aproximadas en la realizacion del proyecto por parte del alumno. De estas, 60 horas en la etapa de iniciacion a la libreria OpenCV, la fase de analisis y diseño. Otras 50 aproximadamente en el estudio de las posibles alternativas para efectuar el proyecto y las restantes 90 para la implementacion y las pruebas.

3-Resultados

Antes de presentar los resultados obtenidos en este proyecto describimos en detalle los datos, métodos/valores y métricas de evaluación consideradas.

3.1-Datos

Para la implementación del programa se ha utilizado el lenguaje de programación c++ en el entorno de desarrollo Microsoft Visual Studio 2008. Las técnicas de procesamiento de imágenes y de captura de vídeo han sido realizadas con la ayuda de la librería gráfica OpenCV.

El programa realiza capturas obtenidas de una webcam o una cámara de vídeo conectada al ordenador mediante conexión USB. Estas capturas se realizan con un intervalo de 2 ó 3 segundos para que le pueda dar tiempo al usuario de cambiar la posición de la mano para cada símbolo a interpretar.

Cada captura a interpretar pasa por unos determinados procesos para poder ser clasificada:

- conversión de color:** Se realizara la conversión de la imagen a un espacio de color con el que se puedan obtener mejores resultados.
- Separación de canales:** La imagen se divide en los tres canales correspondientes a los canales del espacio de color al que pertenece para que puedan ser procesados independientemente. Estos canales se guardan en tres imágenes por separado.
- Segmentación de imágenes:** Para obtener la región de la que extraemos las características aplicamos un algoritmo de segmentación binaria. Teniendo en cuenta un punto de corte que llamaremos umbral, a cada punto de las imágenes obtenidas con la separación de canales se le dará un valor 0, en el caso de que el valor del punto no llegue a dicho umbral o un valor máximo en el caso contrario.
- Morfología:** Las imágenes obtenidas con la segmentación pueden contener aparte de la zona que nos interesa, pequeñas zonas que nos pueden dar resultados erróneos. Estas zonas si son pequeñas se pueden eliminar con un algoritmo de apertura o de cierre (ver apartado de segmentación de metodología). Los parámetros que se utilicen al aplicar la

morfología deberán ser lo mas ajustados posible para que el contorno que deseamos obtener no se vea deformado y por lo tanto de un resultado falso.

•**Extracción de características:** De la imagen una vez segmentada se extrae la zona que nos interesa, se encuadra en un tamaño de imagen que será igual para todas las capturas que se realicen y cuya anchura será igual a su altura. El vector de datos conseguido tendrá tantos elementos como el tamaño en píxeles del lado de la imagen multiplicado por cuatro.

Las capturas se deben realizar a ser posible con un fondo homogéneo y de un color distinto al de la piel humana. El entorno debería estar iluminado con con luz ambiente suave. Con una iluminación muy fuerte la percepción sobre el color puede resultar engañosa.

Los datos usados en el programa para realizar la clasificación se encuentran en el fichero "data.dat". Estos datos son conjuntos de números que representan las características de las imágenes que se toman como modelo. El fichero de datos contiene varias muestras de características para cada clase, con un total de 34 clases representando a los símbolos del alfabeto y los números.

3.2-Métodos de obtención

Para la obtención de resultados se han ajustado los valores de las variables de cada operación de la siguiente manera:

Para la segmentación por canales se utiliza el espacio de color HSV (Hue, Saturation, Value). El rango de valores que puede tomar cada píxel (lo que determina su intensidad) es de 8 bits, con lo que tenemos 256 valores posibles para cada canal por píxel. Para cada uno de los canales se utilizan estos métodos de segmentación y valores de umbral:

•**Hue:** Utilizamos segmentación binaria inversa, con lo que discriminamos los valores más altos del rango de tonos (colores) y nos quedamos con los colores que nos interesan que son los que van del rojo al amarillo. Al umbral le damos por tanto un valor de 54.

•**Saturación:** Usamos también segmentación binaria inversa ya que los valores más claros son los que se encuentran en el rango más bajo de la escala y son los que nos interesan. Al umbral le damos un valor de 70.

•**Value:** En este caso hacemos una segmentación binaria, discriminamos así los valores mas oscuros de la imagen que son los que están en el nivel más bajo de la escala. Al umbral le damos un valor de 173.

Para la fase de morfología aplicaremos un valor 6 en una primera fase de apertura y un elemento estructurante elíptico, con lo que eliminamos zonas con esta forma y un tamaño de alrededor de 8 píxeles de altura y anchura. Después aplicamos un segundo paso de cierre con un valor 11 para conectar las zonas de la imagen que hayan podido quedar desconectadas de la imagen.

Las imágenes resultantes se reducen a un tamaño de 100x100 píxeles, con lo que obtenemos vectores de datos de 400 elementos. Con un tamaño mayor se incrementaría el tiempo de cálculo para la clasificación y con un tamaño menor se perderían características.

Los vectores se clasifican tomando como modelo los datos que se encuentran en el fichero "data.dat", con un total de 3 muestras para cada uno de los 34 símbolos. Estos símbolos son las letra del alfabeto ASL exceptuando la letra 'j' y la 'z' .La letra 'j' sólo varía de la 'i' en el movimiento, lo que no nos interesa evaluar en este trabajo, lo mismo ocurre con la letra 'z' respecto a la 'x'.

El entorno en que se realiza la validación requiere de un fondo con un tono diferente al de la piel humana, por ejemplo una pizarra.

3-3. Validación

Para realizar la validación se han efectuado varias pruebas.Entre ellas una con el texto "**alfabeto de signos**".

En la imagen 3.3.1 se observan los resultados de la prueba. En la primera fila (**signos**) vemos los símbolos a efectuar, en la segunda (**capturas**) observamos las imágenes correspondientes al momento de hacer la captura para su procesamiento y clasificación. En la fila (**resultados**) observamos los símbolos que se nos han mostrado por pantalla después de haber sido clasificada la imagen. En la ultima fila (**distancias**) vemos el valor que nos devuelve la función DTW y que es el resultado de la comparación entre el vector extraído de la imagen y el modelo incluido en el fichero "data.dat" y cuya

distancia es la menor de todos los modelos comparados.

El programa ha interpretado la frase “**alfsbet0 dn sigs0s**”, lo que nos da un porcentaje de aciertos de 11 aciertos/16 símbolos, igual al 68.75 %.

































signos								
	a	l	f	a	b	e	t	o
capturas								
resultados	a	l	f	s	b	e	t	0
distancias	0.0126141	0.004728	0.006731	0.009751	0.00561	0.004391	0.002801	0.00496
signos								
	d	e	s	i	g	n	o	s
capturas								
resultados	d	n	s	i	g	s	0	s
distancias	0.00441	0.00357	0.0541	0.002451	0.04865	0.003865	0.00390	0.00597

Imagen 3.3.1 Resultados de la validación

En la siguiente tabla observamos los resultados para las pruebas con otras palabras.

Palabra	Resultado	Distancias	% aciertos
c	c	0.005606	
l	l	0.009662	
a	c	0.0132	
u	u	0.003298	
d	d	0.0051	
i	i	0.00314	
o	0	0.003081	71.42 %
w	w	0.0043	
o	0	0.00257	
l	l	0.00437	
f	f	0.00688	75,00%

3-4. Discusiones

Siendo más precisos este porcentaje del 68.75 % se puede tomar como un porcentaje de aciertos de 13 aciertos/ 16 símbolos = 81.25 %, ya que al '0' se le representa mediante el mismo símbolo que a la 'o' y su interpretación depende sobre todo del contexto.

El porcentaje de aciertos puede variar por factores como el fondo elegido sobre el que se van a realizar las capturas o la iluminación (que puede provocar variaciones en la interpretación del tono de la piel por parte del programa, o provocar interpretaciones erróneas si no se segmenta adecuadamente el canal de Saturación o Valor ya que éste presenta variaciones en el contorno de la mano).

Otro factor que da ciertos problemas es la similitud entre los contornos obtenidos para varios símbolos. Este es el caso por ejemplo de la 'a' y la 's'. La correcta realización de los símbolos por parte del usuario también es importante a la hora de obtener resultados.

El valor distancias es un indicador de la corrección de la realización del símbolo respecto al modelo almacenado, cuanto mas bajo sea este mejor.

4.conclusión y líneas futuras

Los puntos importantes de este proyecto son la segmentación por umbral y la extracción de características que nos permite realizar la clasificación de signos. En nuestro caso realizamos una segmentación simple que nos permite obtener el contorno de la imagen. Se puede mejorar la extracción de características realizando diferentes fases de segmentación para cada captura de imagen teniendo en cuenta las variaciones de tono y luminosidad de la piel de la mano (Imagen 4.1.). Con esto se podrían extraer características específicas para símbolos que contiene diferencias entre sí más allá del contorno. Se puede reducir también el tiempo de cálculo escogiendo características de la imagen que requieran un menor tamaño del vector previo a la fase de clasificación, ya que cuanto mayor es la longitud del vector, mayor es el tiempo de cálculo en fase de clasificación.

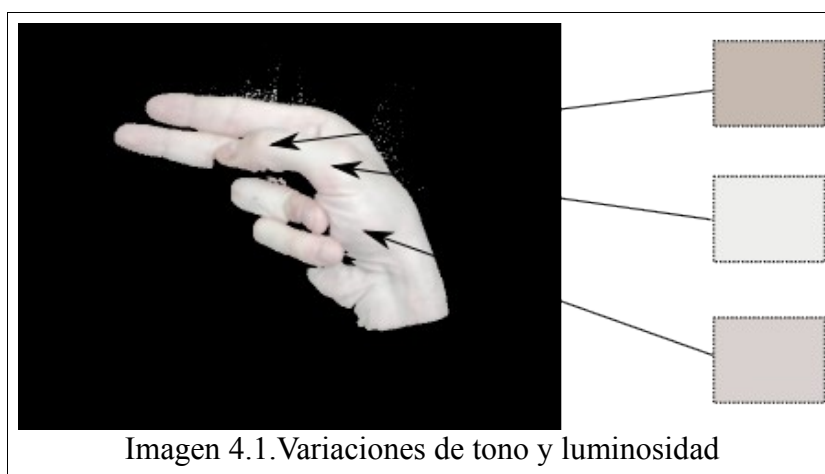


Imagen 4.1. Variaciones de tono y luminosidad

Este programa en concreto está sujeto a una serie de restricciones de uso, como la utilización de fondos e iluminaciones adecuados. Se podría mejorar la adaptabilidad a entornos diferentes permitiendo la variación de parámetros como los valores de umbral o el valores y tipo (binaria o binaria inversa que discrimina zonas diferentes del rango de valores) que nos permitiría usarlo en entornos diferentes o con diferentes tonos de piel.

En nuestro caso nos hemos limitado a los símbolos que constan sólo de una posición. Existen aparte otros símbolos que requieren mas posiciones para ser definidos, lo cual se podría tener en cuenta para una posterior ampliación el considerar varias capturas para un símbolo.

Concluyendo, este trabajo puede ser positivo para facilitar la comunicación entre personas que usan el el lenguaje de signos como medio de comunicación y los que no lo usan, como si de un traductor de idiomas o medio de aprendizaje se tratara.

5.Anexos

La entrega se complementa con la entrega de un disco compacto conteniendo los siguientes elementos:

- Archivo de texto "README" explicando el contenido del CD y funcionamiento del programa e instrucciones de compilación
- Ejecutable del proyecto con las librerías y archivos necesarios para funcionar en plataforma Windows.
- Librería OpenCV (utilizada en la realización del proyecto).
- Código fuente en c++ optimizado para el entorno de desarrollo Microsoft Visual Studio 2008.
- Copia en pdf de la memoria.

6.Bibliografía

- [1] Stokoe, W., Casterline, D. y Croneberg, C. (1965). A Dictionary of American Sign Language on Linguistic Principles. Linstok Press.
- [2] Schantz, M. y Poizner, H. (1982). "A computer program to synthesize American Sign Language". Behavior Research Methods and Instrumentation, 14(5), pp. 467–474. ISSN 0005-7878.
- [3] Alonso, F., de Antonio, A., Fuertes, J.L. y Montes, C. (1995). "Teaching Communication Skills to Hearing-Impaired Children". IEEE MultiMedia, 2(4), pp. 55–67. ISSN 1070-986X.
- [4] T. Baudel, M. Baudouin-Lafon. Charade: remote control of objects using free-hand gestures. En *Comm. ACM* 36(7), páginas 28-35, 1993.
- [5] D.J. Sturman, D.Zeltzer. A survey of glove-base input. En *IEEE Computer Graphics and Applications* 14, páginas 30-39, 1994.
- [6] N. D. Binh, E. Shuichi, T. Ejima. Real-Time Hand Tracking and Gesture Recognition System. En *ICGST International Journal on Graphics, Vision and Image Processing*, Vol. 06, Special Issue on Biometrics, páginas. 31-39, 2006.
- [7] A. Farhadi, D. Forsyth, R. White. Transfer learning in Sign language. En *Computer Vision and Pattern Recognition, 2007. CVPR '07. IEEE Conference*, páginas 1-8, 2007.
- [8] A. Stefan, V. Athitsos, J. Alon, S. Sclaroff. Translation and scale-invariant gesture recognition in complex scenes. En *Proceedings of the 1st international conference on Pervasive Technologies Related to Assistive Environments*, artículo 7, 2008.
- [9] J. Martín, V. Devin, J.Crowley. Active hand tracking. En *Automatic Face and Gesture Recognition*, páginas 574-578, 1998.
- [10] F.Chen, C. Fu, C. Huang. Hand gesture recognition using a real-time tracking method and Hidden Markov Models. En *Image and Video Computing*, 21(8): 745-758, Agosto 2003.
- [11] J.Alon,V.Athitsos,Q.Yan, S.Sclaroff. Simultaneous localization of dynamic hand gestures. En *IEEE Motion Workshop*, páginas 254-260, 2005.

[12]Pablo Bonet, J. de (1620) *Reduccion de las letras y Arte para enseñar á ablar los Mudos*. Ed. Abarca de Angulo, Madrid, ejemplar facsímil accesible en la [Biblioteca Histórica de la Universidad de Sevilla](#)

[13][Escuela española de sordomudos, o Arte para enseñarles a escribir y hablar el idioma español, dividida en dos tomos. Tomo I / obra de Lorenzo Hervás y Panduro.](#) -- Ed. facsímil. [Biblioteca de Signos](#). Original: Madrid, en la Imprenta Real,1795.

