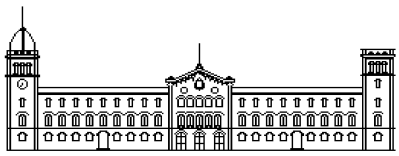


Treball fi de grau

ENGINYERIA INFORMÀTICA

Facultat de Matemàtiques
Universitat de Barcelona

Reconocimiento por forma de la lengua de signos

Enrique Antón López

Director:
Sergio Escalera
Realitzat a: *Departament de Matemàtica Aplicada i Anàlisi. UB*

Barcelona, 8 de juliol de 2010

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 también 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.

Índice de contenido

1.Introducción

1-1.Problema/Motivación

Parafraseando a Roger Penrose *"La inteligencia artificial tiene como objetivo imitar por medio de máquinas las actividades relacionadas a la inteligencia humana"* .Hoy en día los sistemas informáticos y los algoritmos implementados en ellos permiten a dichas maquinas tomar decisiones con cierta “autonomía”.

Estos nuevos avances, mas el progreso realizado en los medios de interacción entre personas y máquinas (Human Computer Interaction) hacen posible día a día crear medios 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 métodos de intercambio de información entre las personas y los sistemas informáticos. Los métodos de intercambio de información generalmente corresponden a sensores que intentan cada vez mas imitar 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 sistemas de reconocimiento de voz, se pueden idear medios de interacción apropiados para la comunidad sorda (aparte del tradicional teclado), que pueden mejorar la interacción de estas con su entorno y con las personas oyentes.
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.Estado del arte

Origen de las lenguas de signos

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".

Imagen 1-2.1.Laminas del abecedario demostrativo de Juan de Pablo Bonet representando las letras A,B,C y D

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 hacia 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.

Lengua de signos americana, (American Sign Language,ASL)

La lengua de signos para el idioma Ingles 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.

Imagen 1-2.3.Alfabeto ASL

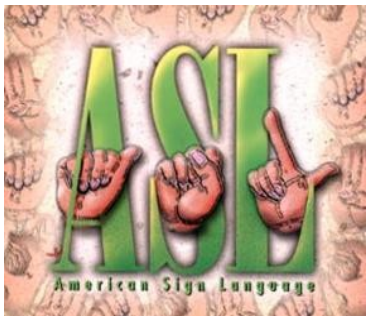


Imagen 1-2.2.Logotipo ASL

Antecedentes relacionados con la captura de gestos humanos

Desde la antigüedad se ha estudiado la interpretación visual de los gestos humanos empezando en la antigua Grecia. Los primeros estudios estaban orientados a la actividad artística, entre ellos los de Leonardo da Vinci . [14], utilizado sobre todo en ese tiempo para la aplicaciones medicas de la época. Mas recientemente, a finales del siglo XIX, con los inicios del cine Eadweard Muybridge *realizo un trabajo en que realizaba capturas de personas filmadas en movimiento* [15].

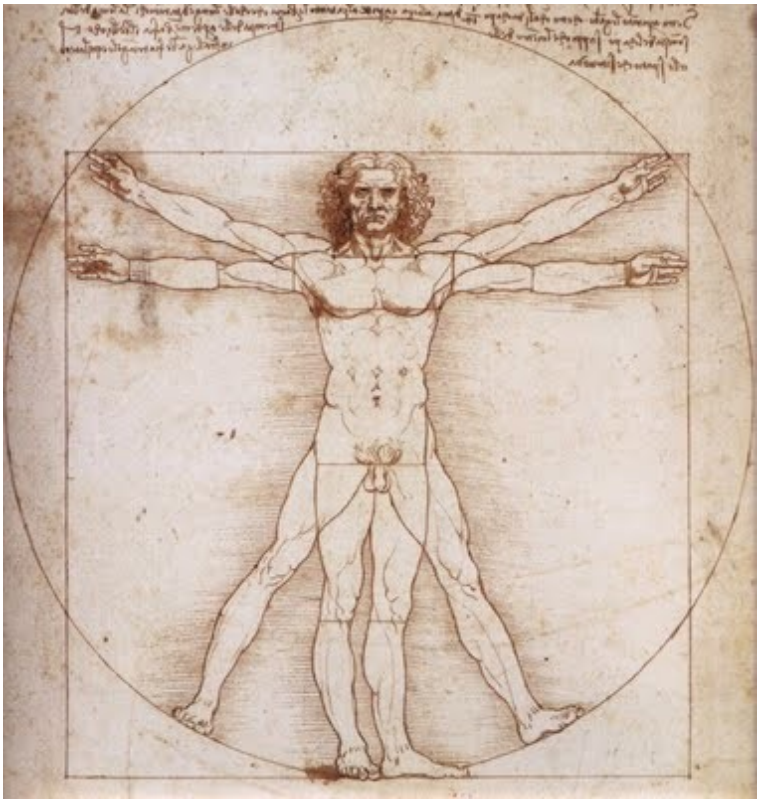


Imagen 1-2.4 Hombre de Vitrubio

Existe una gran cantidad de trabajos previos en cuanto a captura e interpretación de gestos en sistemas informáticos. En las ultimas décadas se han realizado grandes avances en la interacción persona-ordenador, se han introducido nuevos canales de entrada de datos hacia el ordenador aparte de los habituales. Hoy en día es posible introducir ordenes o escribir texto en el ordenador mediante la voz utilizando métodos de obtención de características y clasificación. De la misma manera se puede interactuar mediante cámaras de video y utilizar la información obtenida de estas de múltiples formas, entre ellas vigilancia, utilización del video como interfaz, aplicaciones medicas, utilización en creación de video-juegos y películas, etc... . Existen múltiples antecedentes en reconocimiento y síntesis de gestos mediante video , sobre todo orientados al reconocimiento de signos para personas sordas. Los primeros trabajos en lengua de signos se hicieron para sintetizar los símbolos mediante sistemas informáticos. Schantz y Poizner [2] en 1982 utilizaban la animación de un brazo esquemático que simulaba la lengua de signos americana. Otro proyecto similar fue 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).

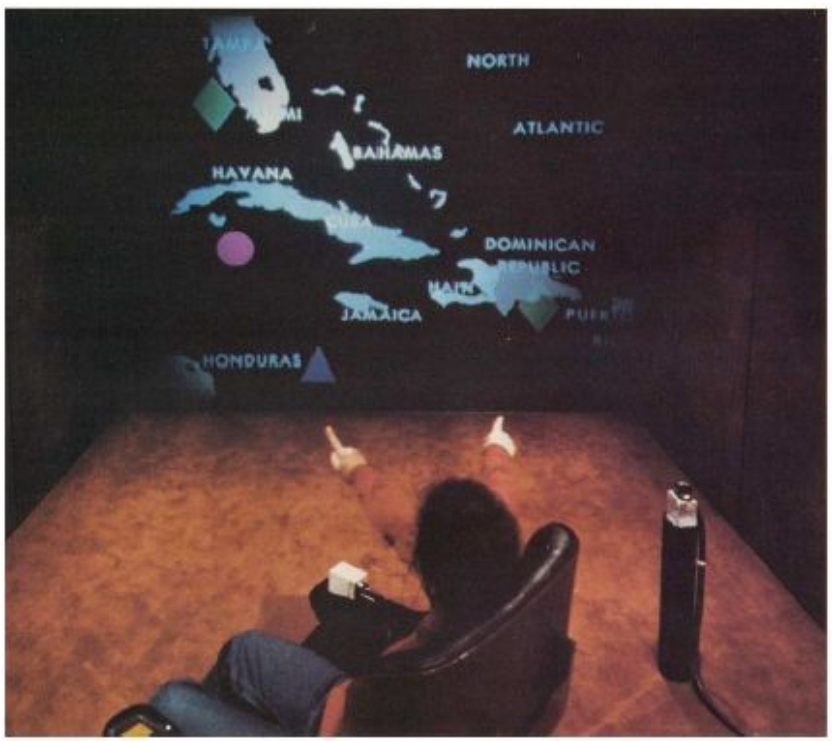


Imagen 1-2.5 Put-that-there

En cuanto a reconocimiento e interpretación de gestos tenemos los trabajos de Richard A. Bolt (Imagen 1-2.5)en 1980, que utilizaba comandos basados en gestos y voz para interactuar con un sistema [16]. .Mas adelante, en 1993 , Thomas Baudel y Michel Baudouin-Lafon crearon un sistema que utilizaba guantes de datos [4] .En 1994, James Davies y Mubarak Shak comenzaban a clasificar signos digitalizando las yemas de los dedos [17].

Los trabajos mas recientes en reconocimiento de la lengua de signos se encuentra el proyecto ILSE, promocionado por la fundacion CARTIF en colaboracion con Telefonica I+D y Redhada.Este sistema interpreta los gestos y los traduce a voz y viceversa [18].
Otro proyecto importante es el creado por Sergio Escalera, Petia Radeva y Jordi Vitrià del Centro de Vision por Computador de la UAB.Este proyecto reconoce frases comunmente utilizadas interpretando los gestos realizados mediante el rostro, las manos y los brazos [19].

Entre los métodos a utilizar 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 artificial.
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].

Metodologías utilizadas en el reconocimiento de símbolos

Para que el sistema pueda reconocer los símbolos se precisan una serie de pasos previos dependiendo del método de clasificación.

El primer paso es procesar la imagen para extraer los contornos. Para obtener la imagen que interviene en la clasificación se utiliza la segmentación o extracción de la región a analizar. Entre los métodos que se pueden utilizar están:

- Segmentación por umbral: Se utiliza para imágenes con iluminación uniforme. Por elección de un umbral a partir del histograma (nº de pixeles para cada intensidad) de forma arbitraria o mediante algoritmos:
  - Método Otsu:Tiene en cuenta la medida de la dispersión del histograma mediante el calculo de la variancia buscando el punto que minimice esta.
  - Método Gonzalez y Woods:A partir de un umbral inicial, se calculan los promedios de los grupos de valores que estén a los lados de este. Se calcula un nuevo umbral a partir de los dos promedios anteriores. Se realizan estos pasos hasta que la diferencia entre dos umbrales consecutivos sea menor a un valor prefijado .Este método es muy similar en resultados al método de Otsu.
  - Método mínimos y máximos:Busca los puntos máximos en el histograma y coloca el umbral en el punto mínimo entre ellos . Para imágenes de iluminación uniforme cada elevación corresponde a un objeto de la imagen,
- Adaptative Median, utilizado en imágenes con iluminación cambiante. Se realiza la búsqueda de umbrales por regiones.
- Mahalanobis Color Segmentation:Utilizada para imágenes en color. Utiliza la distancia Mahalanobis para que pixeles se van a segmentar o no.

La mayoría de métodos de obtención de características requieren la obtención de los contornos de los objetos a clasificar. Uno de los métodos mas completos y mas utilizados para utilización de descriptores es el algoritmo de Canny :

- Algoritmo de Canny:Obtiene una imagen conteniendo los bordes de los objetos que aparecen en la imagen original. Se genera en estos pasos:
  - Suavizado de la imagen: Eliminación de ruidos
  - Calculo de gradientes: Variaciones de intensidad en pixeles vecinos
  - Supresión de gradientes que no lleguen a un máximo determinado.
  - Histéresis de umbral:Aplicación de dos umbrales para obtener los contornos.

Para que el sistema pueda reconocer los símbolos es preciso extraer las características que puedan definirlos. Estas características se llaman descriptores. En función del tipo de modelos se hace uso de unos descriptores u otros:

- HOG (Histograms of oriented gradient): Se basa en la distribución de los gradientes(variaciones de intensidad entre pixeles vecinos) de los contornos. La imagen se divide en sectores, para cada uno de estos se obtiene un histograma de gradientes. La combinación de estos genera el descriptor.
- Zernike:Este descriptor se basa en utilizar los polinomios o momentos de Zernike para describir la imagen.
- Zoning: Consiste en dividir los contornos de la imagen en sectores y guardar la cantidad de pixeles de cada sector.
- Shape context: Basado en forma. Para una serie de puntos de la imagen calcula las distancias a otros puntos.
- CSS:descriptores de curvatura:Obtiene muestras de la curvatura en puntos de interés.
- Blurred Shape Model:Se divide la imagen en regiones y para cada región se calcula la distancia para cada punto a los centros de las regiones adyacentes a la que se encuentra.

  Mas información :[20]

Los descriptores obtenidos son procesados mediante la clasificación. Una vez conseguido el descriptor se compara con los descriptores que se encuentran almacenados en el sistema mediante métodos que nos dan una idea de la similitud entre estos:

- Distancia euclidiana: El mas básico, nos da la distancia entre dos puntos en el espacio cartesiano.
- Distancia Levenshtein: Mide la diferencia entre dos secuencias, usada sobre todo en comparación de textos.
- DTW: Compara la similitud entre dos secuencias que pueden variar en tiempo y velocidad.

Clasificación:

- Bagging: Tiene una fase de generación de modelos. Para n ejemplos en la base de datos y m modelos, desde 1 a m muestrea n ejemplos de la base de datos y a partir de ahí aprende un modelo y lo almacena. En la fase de clasificación toma los modelos y devuelve la clase predicha con mayor frecuencia.
- Boosting: Algoritmo de clasificación con aprendizaje, para cada modelo nuevo tiene en cuenta los anteriores modelos mal clasificados.

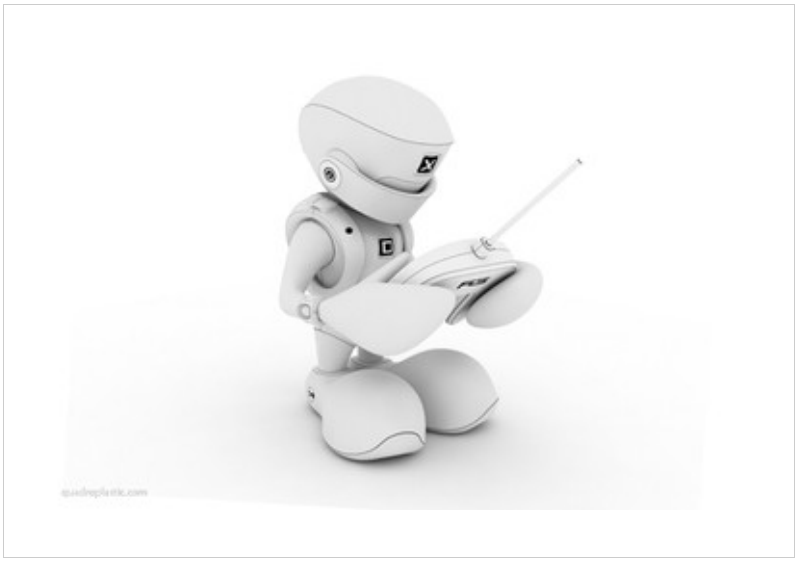
- Adaboost: Variante del anterior.
- K-Nearest Neighbors: Tomando u conjunto de k clases que mas se acercan al modelo a comparar, da por bueno el que mas veces se encuentre en esas k clases.

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 2-0.1.):

1.- Obtención de imágenes a partir de capturas de video.

2.- Filtrado ,segmentación y obtención de bordes de las imágenes para su posterior procesamiento.

3.- A partir de estas imágenes se obtienen una serie de datos característicos.

4.- Clasificación de los datos con la clase correspondiente.

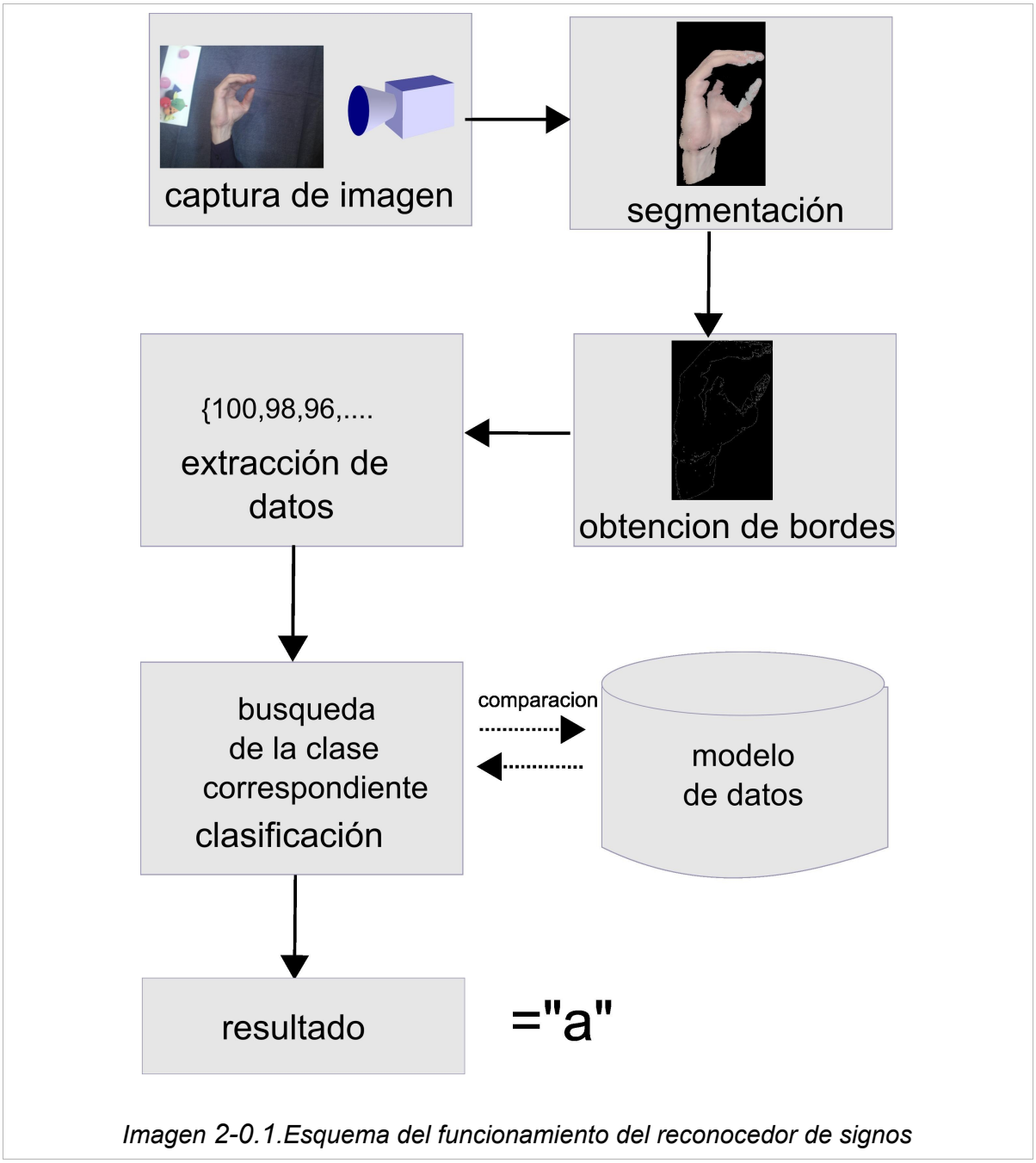


Imagen 2-0.1.Esquema del funcionamiento del reconocedor de signos

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).

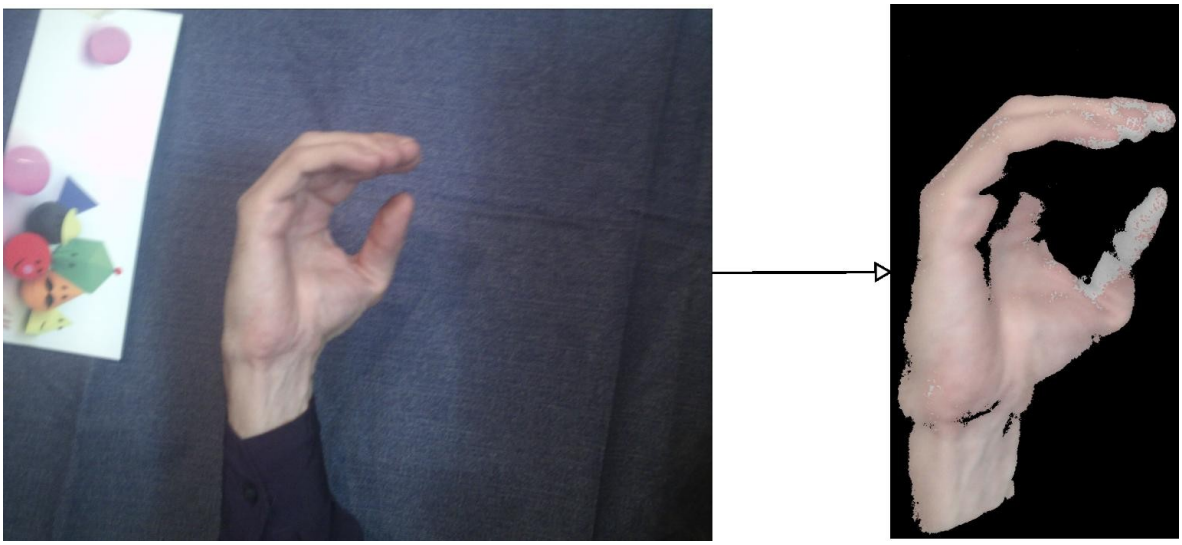


Imagen 2-1.1 Resultado de la segmentación

2.1.1-Elección del espacio de color.

Para este proceso hemos considerado que nos sería más eficiente trabajar sobre el espacio de color HSV (Hue, Saturation, Value – Tonalidad, Saturación, Valor)(Imagen *2-1.1.1*, *2-1.1.2*).

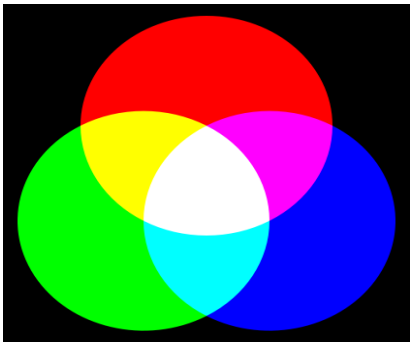


Imagen 2-1.1.1 Mezcla de colores RGB

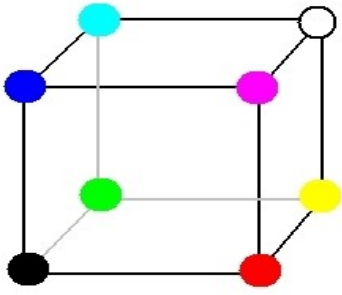


Imagen 2-1.1.2 Cubo de colores RGB

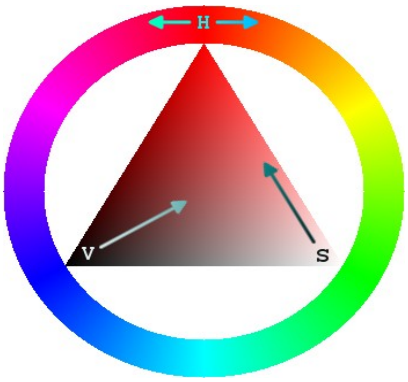


Imagen 2-1.1.3 Representación del espacio HSV

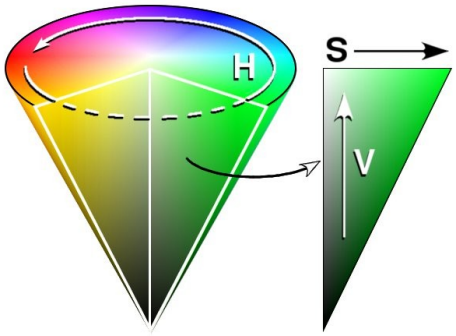


Imagen 2-1.1.4 Representación del espacio HSV

$$M = Max(R, G, B)$$
$$m = min(R, G, B)$$

$$H_1 = \arccos\left[\frac{\frac{1}{2}((R-G)+(R-B))}{\sqrt{(R-G)^2+(R-B)(G-B)}}\right]$$

$$H = H_1 \, si \, , B \leq G$$

$$H = 360^{\circ} - H_{1,} \, si \, B > G$$

$$S = \frac{M-m}{m} \, , V = \frac{M}{255}$$

Imagen 2.1.1.5 .Formula de transformación RGB->HSV

Por lo general las imágenes en color se definen por el espacio de color RGB (red, green, blue) (Imagen 2-1.1.3, 2-1.1.4). 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) .

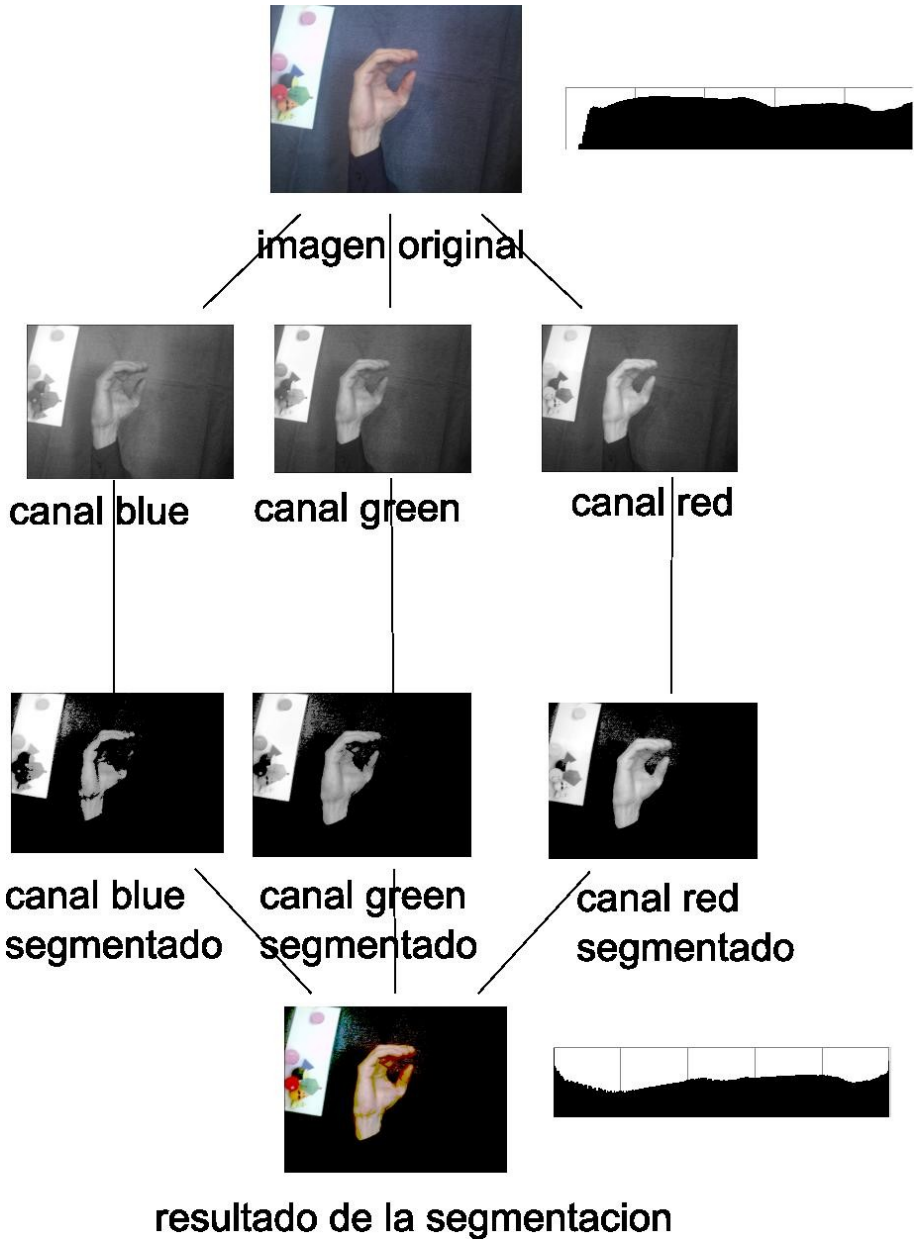


Imagen 2.1.1.6..Resultado de la segmentación en el espacio RGB

El modelo RGB puede ocasionar problemas a la hora de realizar la segmentación de las imágenes(Imagen 2.1.1.6.). 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 modelo HSV nos permite mas tolerancia a cambios de iluminación y fondos no uniformes ya que el canal correspondiente al tono sera el que se tendrá mas en cuenta a la hora de realizar la segmentación dado que sera el que nos diga donde están las zonas que corresponden a la piel.
La función de transformación entre los espacios de color RGB a HSV se observa en la figura 2.1.1.5.

El modelo HSV consta de estos tres canales:

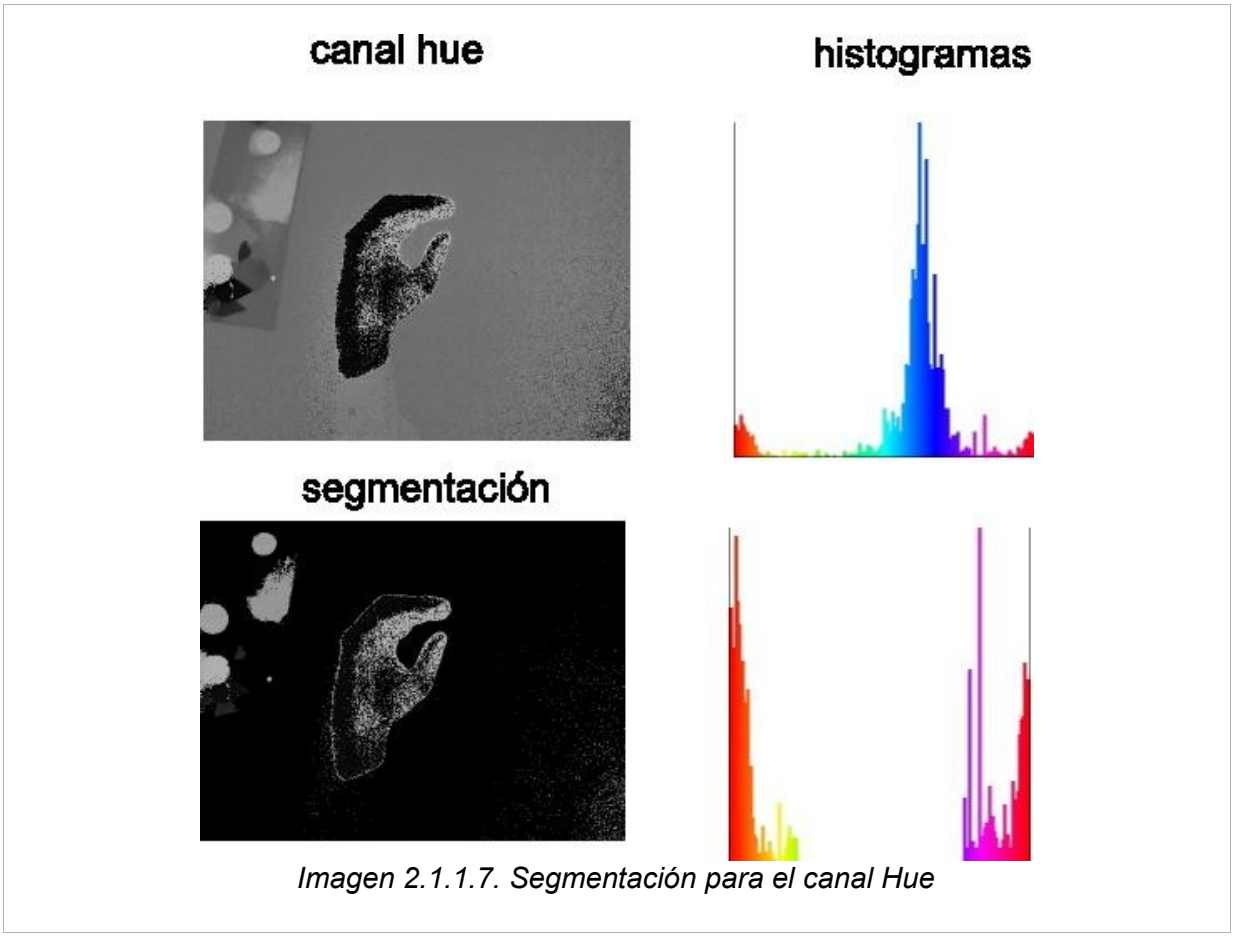


Imagen 2.1.1.7. Segmentación para el canal Hue

• 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, seria el color puro sin ningún tipo de degradación (Imagen 2.1.1.7).Las zonas mas oscuras de la imagen que representa el canal corresponden a los tonos rojo, naranja, amarillo, las intermedias al verde, azul, violeta hasta llegar al rosa-rojo que corresponde a las zonas mas claras.

• 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 2.1.1.8).Las zonas mas oscuras corresponden a los colores menos puros “con mas cantidad de gris”, las mas claras serian los tonos mas puros o sin mezcla de gris.

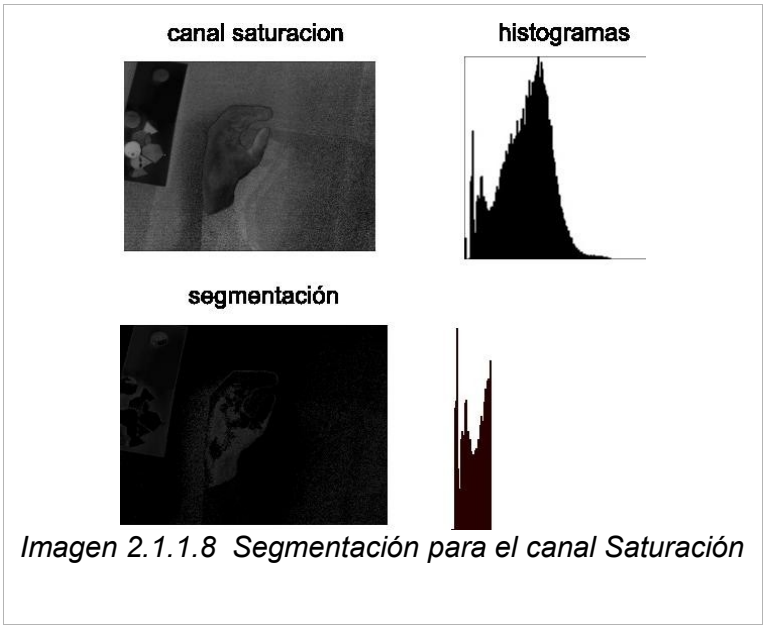


Imagen 2.1.1.8 Segmentación para el canal Saturación

• Canal **Value** (Brillo): El brillo es el valor que va desde el negro hasta el color puro o el blanco (Imagen 2.1.1-.9.Las zonas mas oscuras corresponden a los tonos que mas se acercan al negro, las mas claras son las que mas se acercan al blanco o al tono puro.

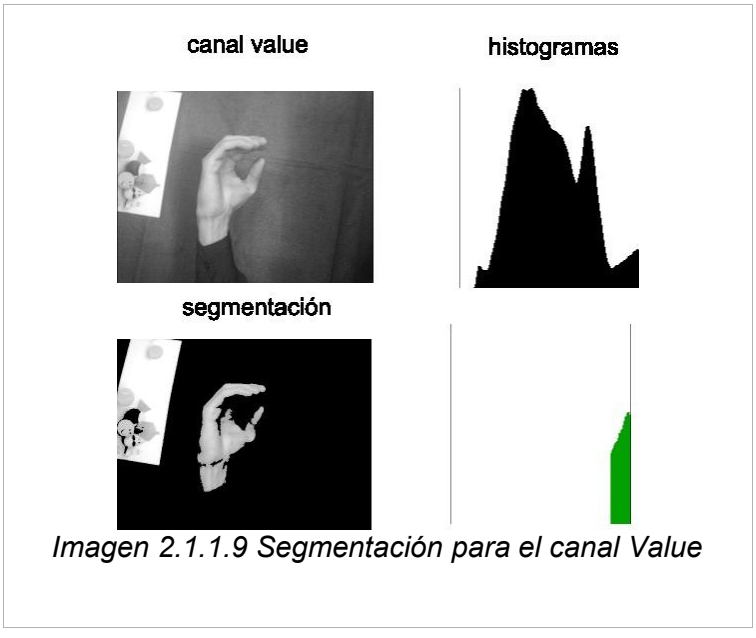


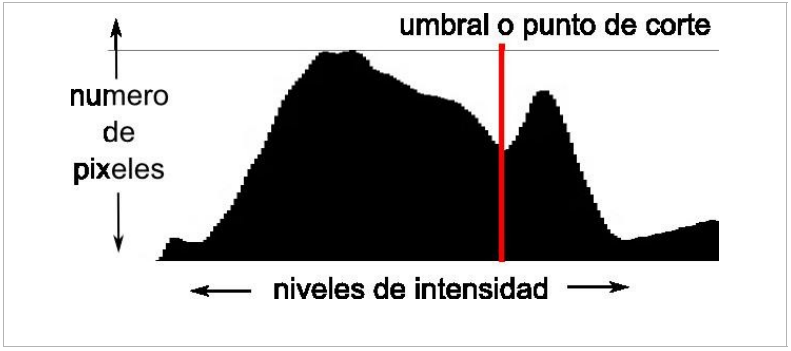
Imagen 2.1.1.9 Segmentación para el canal Value

El proceso de segmentación se realiza según el siguiente algoritmo:
Dado un punto de corte que llamaremos umbral, pondremos todos los pixeles de la imagen que se encuentren por debajo de este al nivel mínimo de intensidad (0, color negro).En el caso de la segmentación binaria los pixeles que se encuentren por encima del umbral se pondrán al nivel máximo (blanco), en la segmentación normal quedan igual.

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

Una ayuda para realizar este proceso es tener en cuenta el histograma de la imagen. El histograma es una representación del numero de pixeles que tienen un valor de intensidad en concreto. Se representa mediante un gráfico en el que el eje de las x (abscisas) corresponde a los niveles de intensidad y el de las y (ordenadas) al de la cantidad de pixeles que toman cada uno de los valores de intensidad.

El umbral para la segmentación se colocara en un punto de las x, una vez que se haya realizado esta el eje de las y a partir de esta estará en 0.



2.1.2-Obtención de la región a segmentar.

Antes de realizar la segmentación es conveniente obtener el recuadro mínimo que contiene a la imagen que nos interesa, para así poder eliminar objetos del fondo que se puedan colar en la segmentación y hagan imposible la extracción de características.
Una manera de hacerlo es realizar una segmentación binaria previa del canal **Hue**, (Imagen 2.1.2.1) teniendo en cuenta solo los tonos que correspondan al color de piel.

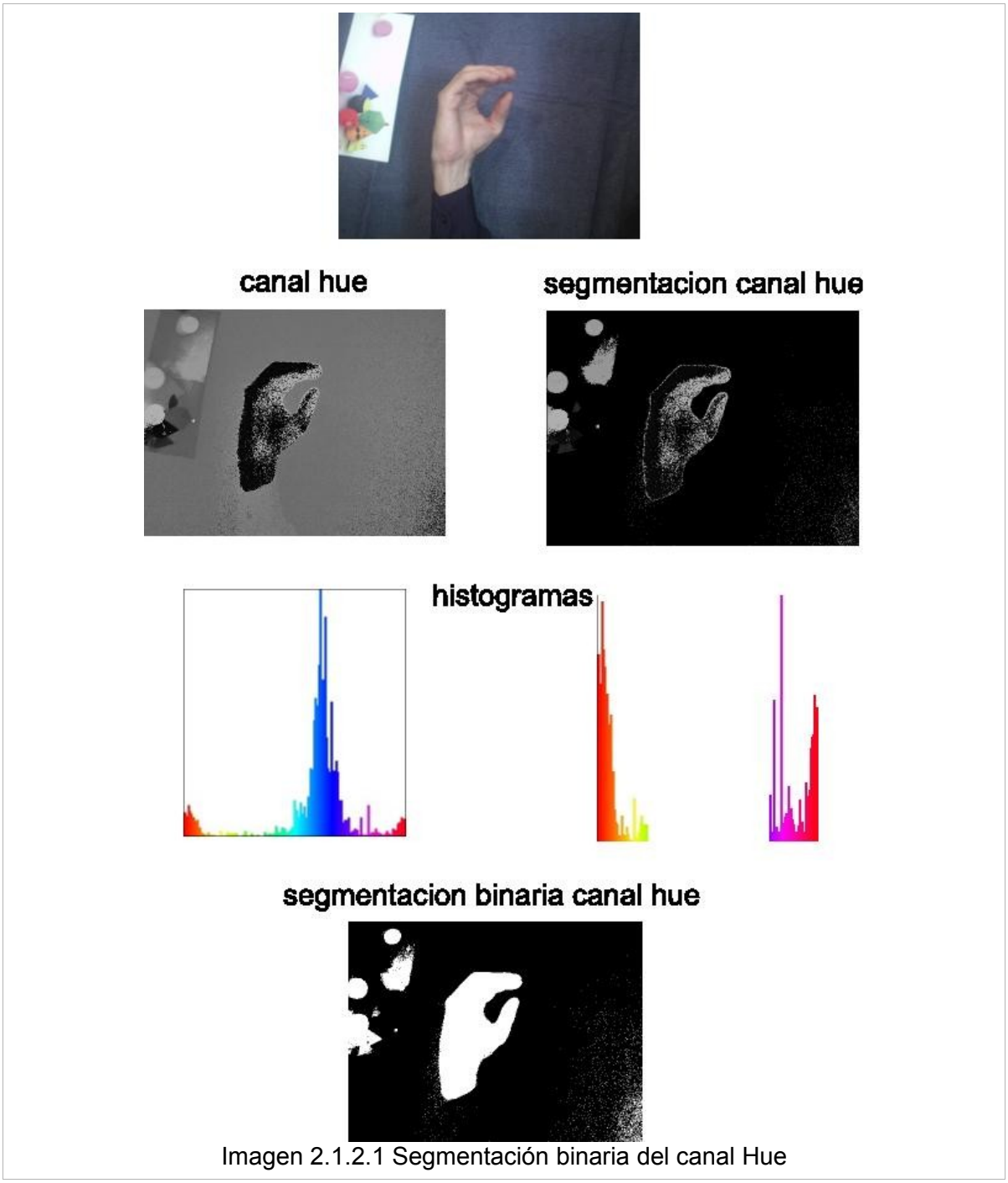


Imagen 2.1.2.1 Segmentación binaria del canal Hue

Esto nos dará una imagen binaria en la que solo debería estar la mano. Un problema que presenta el canal **Hue** es que algunos colores que se acercan mucho al blanco o al negro y no pertenezcan a la mano pueden entrar dentro también de la segmentación. Para que esto no ocurra se recurre a la utilización conjunta del canal **B** del espacio de color **LAB** (Imagen 2.1.2.2, 2.1.2.3)y el canal **Hue**.

El espacio de color **LAB** o **CIELab** consta de tres canales:

- L: luminosidad, va desde 0 (negro), hasta 100 (blanco).
- A:va desde el verde al magenta, de menos a mas.
- B: va desde azul a amarillo.

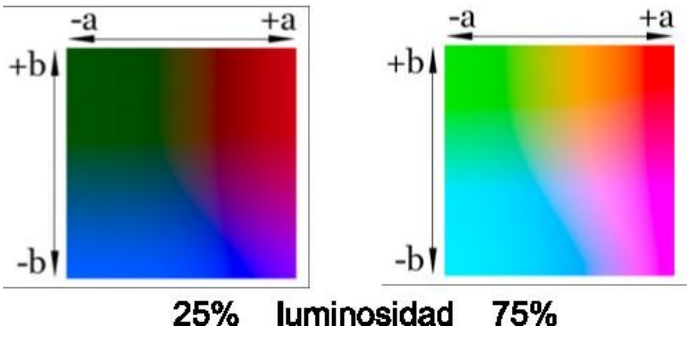


Imagen 2.1.2.2 Paleta LAB

De este espacio utilizamos el canal **B** realizando previamente una segmentación binaria con un umbral aproximado de 127.La elección de este umbral se puede hacer teniendo en cuenta el histograma característico de este canal que consiste en dos picos elevados. La obtención de umbrales se vera mas adelante.

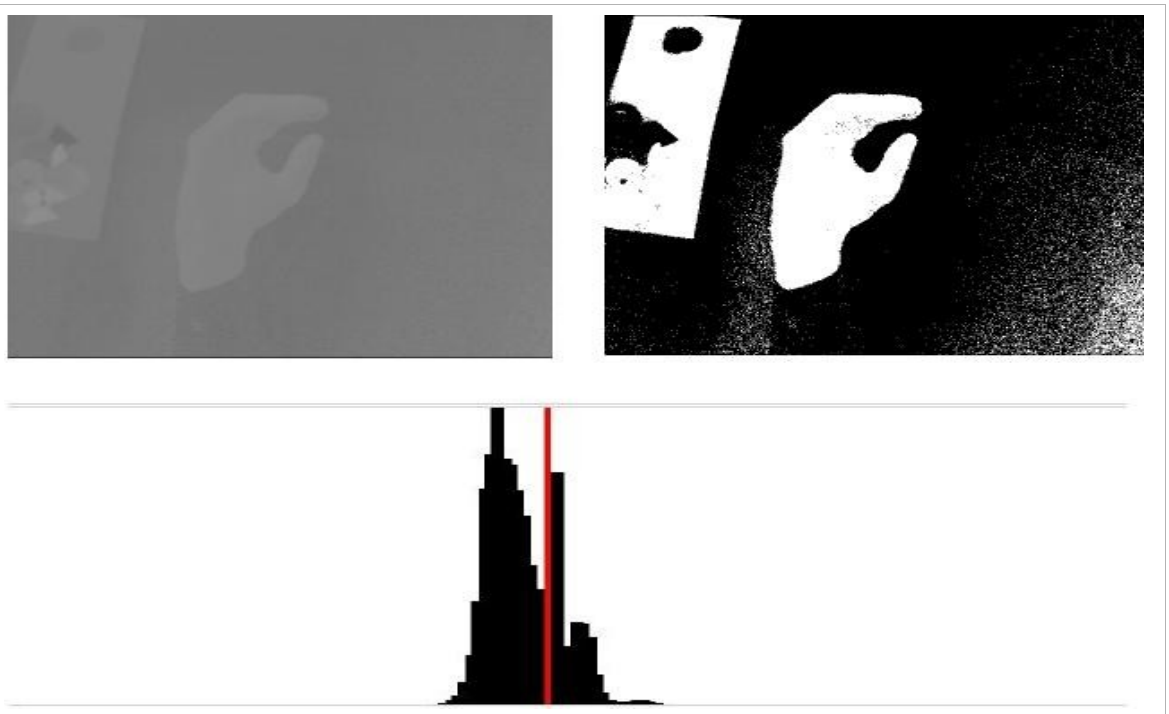


Imagen 2.1.2.3 Histograma LAB

Para el canal Hue se eliminaran todos los tonos que no deben pertenecer al color de piel , que serán los tonos intermedios del histograma

La utilización conjunta de la segmentación de los dos canales mediante una operación AND binaria (Imagen 2.1.2.4)nos da la región a segmentar con bastante exactitud.

Realizamos la proyección del recuento de pixeles por filas y columnas (*Imagen 2.1.2.5*).

Guardamos en un vector para cada elemento de este la cantidad de pixeles cuya intensidad sea mayor que 0 para cada fila de la imagen. Hacemos lo mismo para las

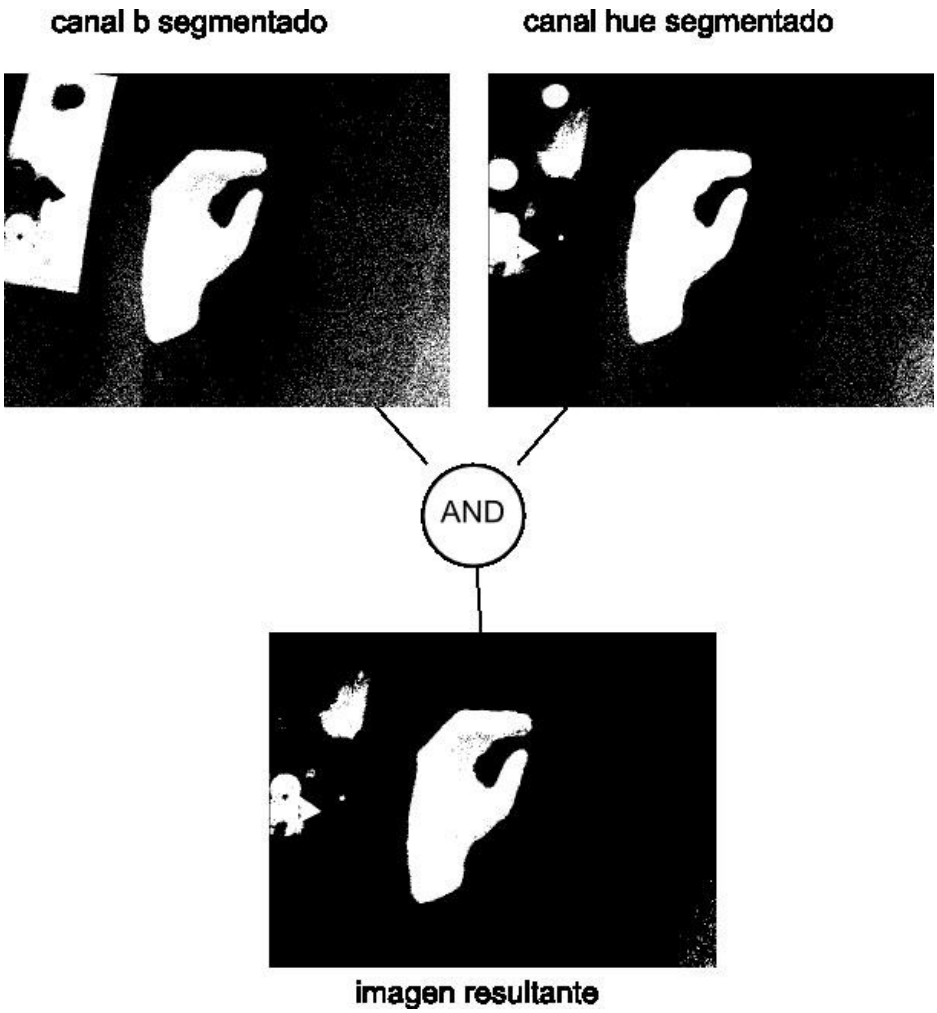


Imagen 2.1.2.4 Creación de la imagen para obtener la region

- columnas. Después se recorren los vectores desde el centro hasta los bordes hasta que se encuentra una secuencia de k pixeles de intensidad igual a 0, esto nos dará cada una de las variables que definen la región que contiene la mano.

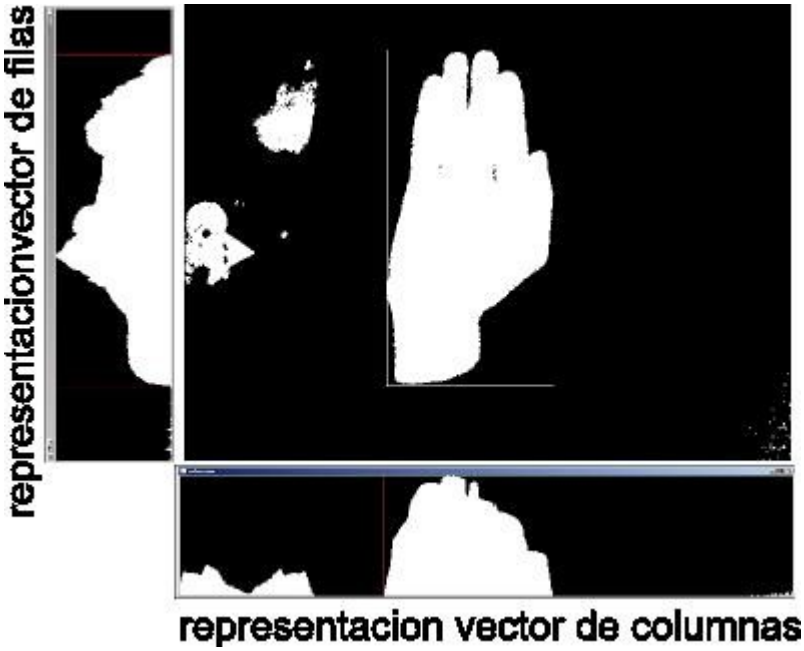


Imagen 2.1.2.5 Representación de los vectores para obtención de la región contenedora

Algoritmo:

```
para i=0 mientras i<imagen->altura, i=i+1
        fila = imagen.obtenerFila(i);
        vector_y[i] = contarPixelesDistintosDeCero(fila)
para i=0 mientras i < imagen->anchura, i=i+1
        columna = imagen.obtenerColumna(i);
        vector_x[i] = contarPixelesDistintosDeCero(columna)
para i= imagen->anchura/2 mientras i > 0, i=i-1
        si vector_x[i] == 0:
                region->x = i
para i= imagen->anchura/2 mientras i > imagen->anchura, i++
        si vector_x[i] == 0:
                region->anchura = i-region->x
para i= imagen->altura/2 mientras i > 0, i=i-1
        si vector_y[i] == 0:
                region->y = i
para i= imagen->altura/2 mientras i > imagen->altura, i++
        si vector_y[i] == 0:
                region-> altura = i-region->y
retorna region
```

2.1.3-Obtención de los umbrales para la segmentación.

Una vez obtenida la mínima región que contiene a la mano hace falta aun poder desechar las zonas que no pertenezcan a la mano y quedarse solo con el contorno y la parte interna de esta para que la obtención de bordes solo nos de los necesarios.

Como ya se vio antes es necesario definir unos umbrales óptimos para poder realizar una segmentación correcta. Estos umbrales pueden variar en cada captura, incluso aunque las condiciones de iluminación sean las mismas.

Existen varios métodos para obtener el umbral optimo en un histograma de forma automática. Entre estos se encuentran el método de Gonzalez y Woods, el método de Otsu, el de los máximos y mínimos.

Para este trabajo se han hecho pruebas con el método de Otsu y el de máximos y mínimos y se han observado mejores resultados con este ultimo método para la mayoría de las capturas.

Método de los máximos y mínimos:

La forma del histograma de una imagen en que se encuentran varios objetos diferenciados nos permite observar que este se encuentra distribuido en forma de picos.

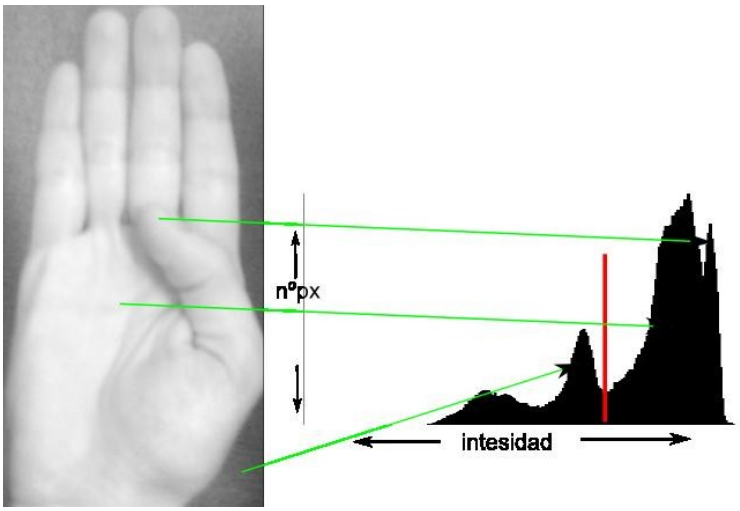


Imagen 2.1.3.1. Distribución del histograma

Cada pico corresponde a zonas de la imagen con intensidad diferente. Por lo general cada una de estas zonas pertenece a un objeto diferente, lo que facilita la extracción de cada uno mediante métodos sencillos como la segmentación por umbral.

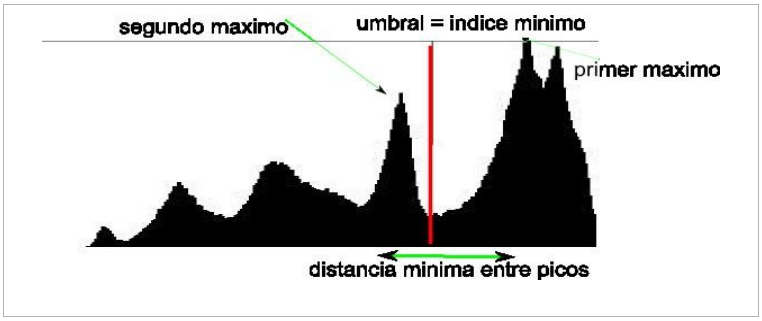
El método de los máximos y mínimos aprovecha esta característica para poder buscar un umbral optimo mediante un sencillo algoritmo.

Algoritmo del método de los máximos y mínimos:

A partir del histograma obtenemos el vector que lo representa. Este vector contiene tantos elementos como niveles de intensidad posibles tenga la imagen (en el caso que nos ocupa 255).Cada elemento del vector contiene un numero que representa el numero de pixeles que contienen ese nivel de intensidad. Por lo tanto el indice de cada posición del vector sera el nivel de intensidad y el contenido de la posición sera el numero de pixeles.

En primer lugar recorrernos cada elemento del vector buscando el mayor y guardamos el indice. Después buscamos el segundo mas grande y guardamos el indice de nuevo teniendo en cuenta que debe haber suficiente separación entre ambos indices para que pertenezcan a picos diferentes. Después recorremos el vector entre ambos indices y buscamos el indice que contenga el valor mínimo, dicho indice sera el umbral adecuado.

Imagen 2.1.3.2 .método máximos y mínimos



Precondición:Vector correspondiente al histograma de la imagen.

Postcondición:Entero comprendido entre los niveles de intensidad.

```
vector_histograma = nuevo vector[niveles de intensidad]
primer_máximo = vector_histograma[0]
indice_primer_máximo = 0
para i = 0, mientras i < max_nivel_intensidad, i++
        si vector_histograma[i] > primer_máximo:
                primer_máximo = vector_histograma[i]
                indice_primer_máximo = i
segundo_máximo = vector_histograma[0]
```

```
indice_segundo_máximo = 0
para i = 0, mientras i < max_nivel_intensidad, i++
        si vector_histograma[i] > segundo_máximo y vector_histograma[i] < primer_máximo y
        valor_absoluto(i-indice_primer_máximo) > distancia entre picos:
                segundo_máximo = vector_histograma[i]
                indice_segundo_máximo = i
si  indice_primer_máximo >  indice_segundo_máximo:
         temp = indice_primer_máximo
        indice_primer_máximo =  indice_segundo_máximo
        indice_segundo_máximo =  temp
mínimo = vector_histograma[indice_primer_máximo]
indice_mínimo = indice_primer_máximo
para i = indice_primer_máximo , mientras i < indice_segundo_máximo , i++
        si vector_histograma[i] < mínimo:
                mínimo = vector_histograma[i]
                indice_mínimo = i
retorna indice_mínimo
```

Método Otsu

Este método tiene en cuenta la dispersión de los valores de intensidad en el vector del histograma mediante el calculo de la variancia. El algoritmo consiste en dividir las intensidades de la imagen en dos clases. La media aritmética de cada una de las clases y la media aritmética del total nos permite calcular la variancia. El umbral optimo sera aquel para el cual esta variancia sea menor.

Aunque este método pueda ser mas efectivo para la generalidad de imágenes uniformemente iluminadas, para este caso concreto no da tan buenos resultados, ya que la mano contiene varias superficies con iluminación distinta. El método de máximos y mínimos nos permite separar el pico del histograma correspondiente al fondo y conseguir así una segmentación aceptable.

2.1.4-Obtención del contorno de la imagen.

Para poder extraer posteriormente de la imagen las características que la definen necesitamos obtener los contornos de la imagen con la mayor claridad posible. Los contornos de la imagen se pueden definir como las transiciones entre regiones de la imagen con diferentes niveles de gris. Estas transiciones por lo general corresponden a los bordes de diferentes objetos, esto seria la situación ideal si no existieran una serie de problemas:

- Las intensidades varían entre un rango discreto de valores.
- Dependiendo del medio utilizado para capturar las imágenes, estas contienen ruido (pequeñas alteraciones en la imagen).
- Ademas de los bordes de los objetos también existen variaciones en la intensidad entre diferentes texturas, en un mismo objeto cambios en a luz, sombras , etc...

El método utilizado en este trabajo se basa en el algoritmo de Canny. Este algoritmo busca los bordes de la imagen y crea una imagen de salida con estos, teniendo en cuenta una serie de parámetros.

El algoritmo de Canny se ejecuta en tres pasos:

- Obtención del gradiente (Imagen 2.1.3.3.b)(variaciones de niveles de intensidad en pixeles consecutivos): Para obtenerlo se suaviza previamente la imagen con un filtro gaussiano. Una vez suavizada esta se obtiene el gradiente mediante un fltro Sobel
- Supresión de no máximos (Imagen 2.1.3.3.c): en este paso se logra el adelgazamiento del ancho de los bordes obtenidos con el gradiente, eliminando los puntos que no llegan a un valor máximo de gradiente.
- Histéresis de umbral (Imagen 2.1.3.3.d): en este paso se aplica una función de histéresis basada en dos umbrales; con este proceso se pretende reducir la posibilidad de aparición de contornos falsos.
-

La biblioteca OpenCV dispone de una función (cvCanny), que implementa todos estos pasos y requiere como parámetros los definidos en el paso de histéresis de umbral. Según el tipo de imágenes a tratar sera mas apropiado utilizar unos valores u otros.

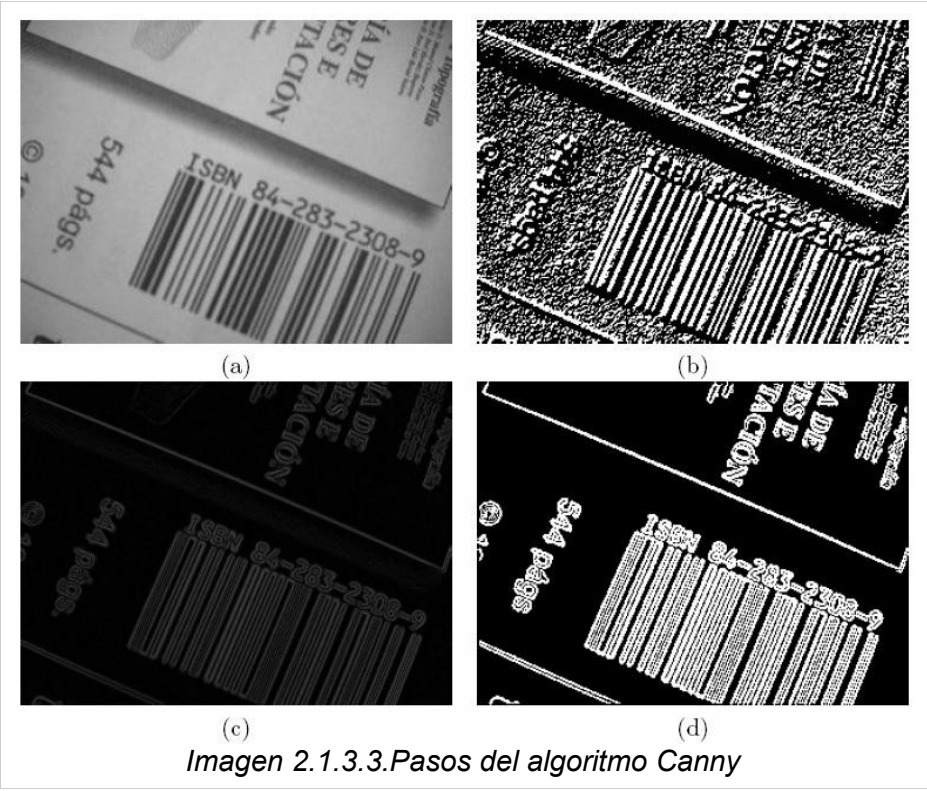


Imagen 2.1.3.3.Pasos del algoritmo Canny

Se pueden utilizar previamente algoritmos de morfología como la erosión y dilatación. Estas funciones se suelen utilizar para imágenes binarias, para eliminar pequeñas regiones o cerrar regiones muy juntas. Aplicando estas a imágenes en gris, sirven para suavizar de forma efectiva el ruido y evitar así la posible aparición de contornos falsos.

2.2-Extracción de características

Una vez obtenida la imagen que contiene los contornos externos e internos de la mano, se necesita obtener una serie de valores que permitan diferenciar a una imagen de otra y poder decidir así que símbolo determina.

Se pueden elegir varias características que nos permitan diferenciar una imagen de otra. En este caso al tener la imagen que contiene los bordes externos e internos de la mano podemos aprovechar toda la información que nos puede ofrecer esta disposición mediante las distancias entre los pixeles contenidos. Uno de los métodos que puede aprovechar mejor estas características es el algoritmo **"Blurred Shape Model".** La ventaja de este método es que pequeñas diferencias entre imágenes correspondientes al mismo símbolo no afectan al resultado. Esto es importante ya que permite una mayor flexibilidad en el sistema.

2.2.1-Descripción del algoritmo "Blurred Shape Model".

- Dividimos la imagen a segmentar en x*y recuadros (Imagen 2.2.1.1, 2.2.1.2).
- Inicializamos un vector de x*y elementos, tantos como recuadros.

Vector = { 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0}

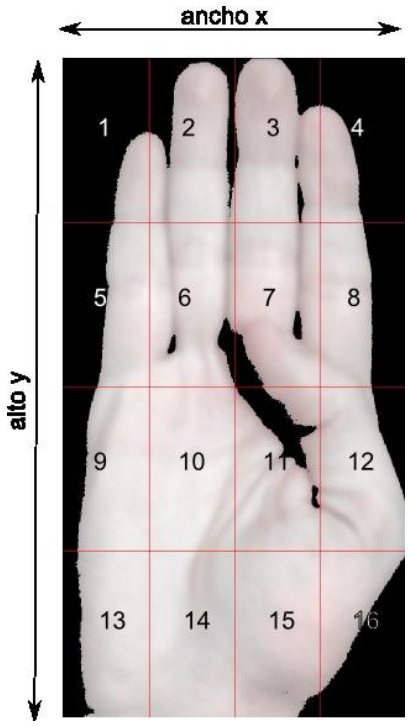


Imagen 2.2.1.1 Configuración del vector de características

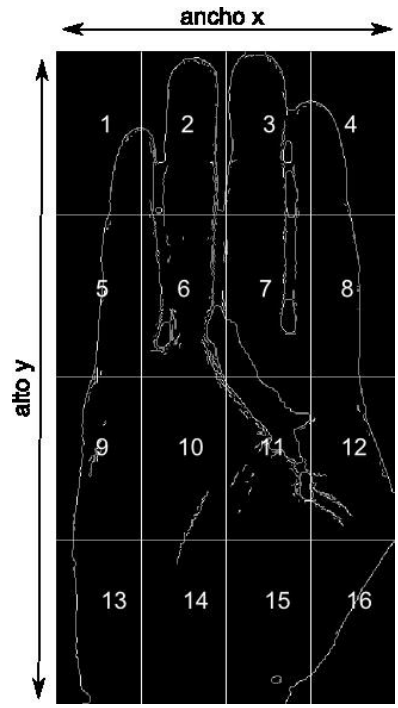


Imagen 2.2.1.2 Imagen sobre la que se aplica el algoritmo BSM

- Para cada pixel con valor de intensidad distinto de 0 calcular las distancias desde este hasta el centro de las regiones adyacentes a la que se encuentra el punto incluido el centro de la región en que se encuentra el pixel *Imagen (2.2.1.3)*.

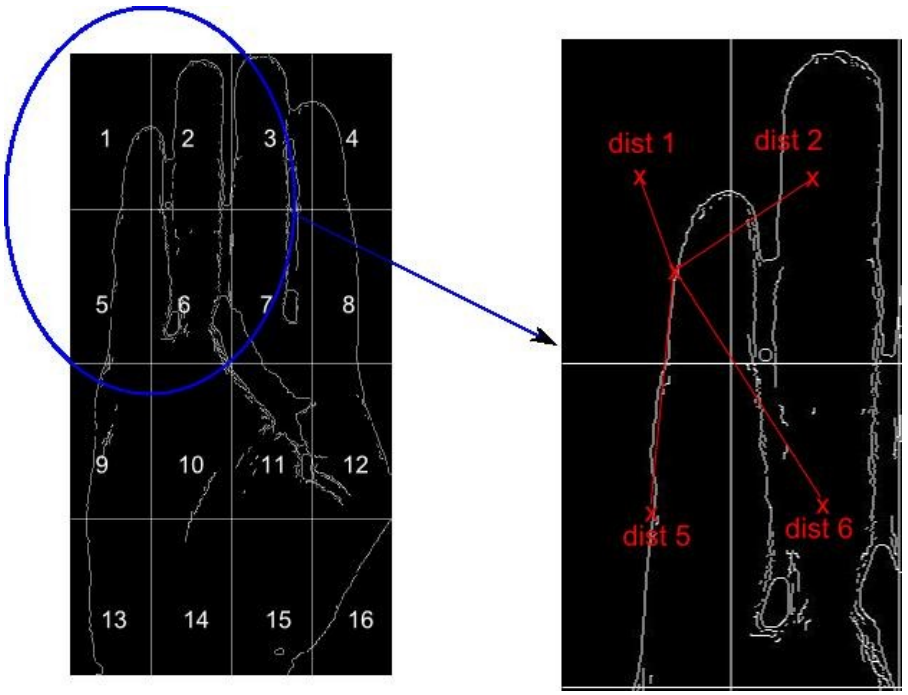


Imagen 2.2.1.3 Calculo de las distancias para cada pixel del contorno

- Los centros de las regiones, que llamaremos centroides, irán cambiando según en que región se encuentre el pixel para el que calculamos las distancias. El calculo de estos centroides sera:
-
  - Para la región contenedora: x = ancho región / 2, y = alto región /2
  - Para cada centroide de as regiones adyacentes el calculo se hará sumando o restando el ancho o el alto de la región, según se encuentren a la derecha o a la izquierda, arriba o abajo, esto se ve mas detalladamente mas adelante en el algoritmo que se adjunta.
  - Si la región en que se encuentra el pixel esta en las esquinas o en los bordes solo se tendrán en cuenta como adyacentes las regiones que se encuentran dentro de las dimensiones de la imagen, no se hará el calculo de las distancias a puntos externos a esta.

- Guardar estas distancias en un vector temporal, en las posiciones correspondientes a las regiones:

  vector temporal = {dist 1,dist 2, 0, 0, dist 5, dist 6 ,0,0,0,0,0,0,0,0,0,0}

- Calcular la suma de la distancia total:
  - $D\,total = \sum_{i=0}^{i=N} dist_i$
- Dividir cada elemento del vector temporal por esta suma:
  - $vector\ temporal = \{\frac{dist_1}{D\,total}, \frac{dist_2}{D\,total}, 0, 0, \frac{dist_5}{D\,total}, \frac{dist_6}{D\,total}, 0,0,0,0,0,0,0,0,0,0\}$

- Calcular la inversa para cada elemento:
  - $vector\ temporal = \{\frac{1}{dist_1}\frac{,1}{dist_2}, 0, 0, \frac{1}{dist_5}\frac{,1}{dist_6}, 0,0,0,0,0,0,0,0,0,0\}$

- Una vez hechos todos estos pasos se actualiza el vector de salida sumando elemento por elemento del vector temporal a este.

- Al finalizar el calculo de las distancias para todos los puntos se vuelve a calcular la suma de las distancias del vector principal y se divide cada unos de los elementos por esta. La suma total de todos los elementos deberá ser igual a 1.

Algoritmo detallado:

Precondición:imagen binaria conteniendo los bordes.
Postcondición:vector cuya suma total sea igual a 1.

```
Inicializar vector de salida:
vector = nuevo vector (nº regiones ancho * nº regiones alto)
Inicializar vector temporal:
temporal  = nuevo vector (nº regiones ancho * nº regiones alto)
para i = 0, mientras i < nº regiones ancho, i++
        para j = 0, mientras j < nº regiones alto, j++
                //calcular centroides
                cCx = i * ancho región + ancho región /2
                cCy = j * alto región + alto región / 2
                // centroides de regiones adyacentes
                cNWx = cCx - ancho_región
                cNWy = cCy - alto_región
                cNx = cCx
                cNy = cCy - alto_región
                cNEx = cCx +ancho_región
                cNEy = cCy - alto_región
                cWx = cCx - ancho_región
                cWy = cCy
                cEx = cCx + ancho_región
                cEy = cCy
                cSWx = cCx - ancho_región
                cSWy = cCy + alto_región
                cSx = cCx
                cSy = cCy + alto_región
                cSEx = cCx + ancho_región
                cSEy = cCy + alto_región
                para x = i * ancho_región, mientras x < (i+1) * ancho región, x++
                        para y = j * alto_región, mientras y < (j+1) * alto región, y++
                                si ColorPixel > 0:
                                        //calcular solo las distancias a los centroides que esten dentro
                                        //de la imagen
                                        temporal[i*ancho_región + j] = distancia (cC,pixel)
                                        temporal[(i-1)*ancho_región + (j-1)] = distancia (cNW,pixel)
                                        temporal[i*ancho_región + (j-1)] = distancia (cN,pixel)
                                        temporal[(i+1)*ancho_región + (j-1)] = distancia (cNE,pixel)
                                        temporal[(i-1)*ancho_región + j] = distancia (cW,pixel)
                                        temporal[(i+1)*ancho_región + j] = distancia (cE,pixel)
                                        temporal[(i-1)*ancho_región + (j+1)] = distancia (cSW,pixel)
                                        temporal[i*ancho_región +( j+1)] = distancia (cS,pixel)
                                        temporal[(i+1)*ancho_región + (j+1)] = distancia (cSE,pixel)
                                suma=calcularSumatorioVector(temporal)
                                dividirVector (temporal, suma)
                                vector = vector + temporal
suma=calcularSumatorioVector(vector)
dividirVector (vector, suma)

retorna vector
```

El otro algoritmo del que se ha hecho uso en este proyecto es el Zoning. De la misma manera que en el anterior dividimos la imagen en un numero NxN de cuadriculas y guardamos en cada posición de un vector de NxN elementos el numero de pixeles distintos de 0 que contenga cada cuadricula.

```
sizeCopia=imagenEntrada->altura/N
para i = 0, mientras i < NxN, i++
        copiaImagen=clonar(imagenEntrada)
        xmin=(i%N)* sizeCopia
        ymin=(i/N)* sizeCopia
        copiaImagen=recortarImagen(xmin,ymin,sizeCopia,sizeCopia)
        nºpixeles=calcularHistograma(copiaImagen)
        vector[i]= nºpixeles
retorna vector
```

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 $M = (M_1,\ldots,M_m)$, que es un modelo de secuencia donde cada M_i es un vector de características. Tenemos $Q = (Q_1, \ldots,Q_m)$, 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 $Q=(x_1,\ldots,x_n,)$.

La trayectoria (*warping path) W* define una alineación entre *M* y *Q*. Formalmente, $W=w_1, \ldots,w_T$, 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.

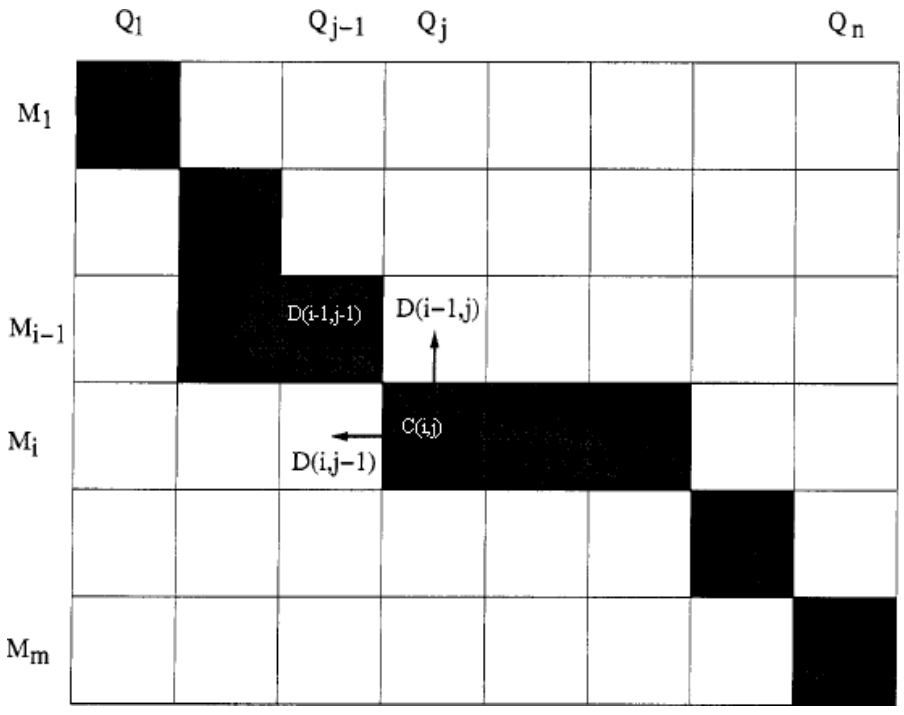


Imagen 2.3.1. Funcionamiento del algoritmo DTW

2.4.Diseño y análisis de costes

2.4.1 Explicación del diseño

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 módulo principal , **recSignosASL** se incluyen las funciones:

processes, que es la que llamamos en la función principal **main** y es donde se inicia el procesado de la imagen y obtención del carácter tras la clasificación.

En esta función llamamos a la función **segmentation**, que realiza la segmentación y a las funciones **erodeDilate** para suavizar la imagen, a la función descriptor (**BSM** o **Zoning** dependiendo del descriptor), que extrae el vector de características y a la función **classifierKNN** que busca el carácter que mas le corresponde según el archivo de muestras.

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 los ficheros que se encuentran en la careta **"\files"** como base de datos para los modelos a comparar.

En **morphology** se encuentran las funciones de procesamiento de imagen y segmentación. En **analisys** los algoritmos necesarios para calcular los histogramas necesarios y los arrays asociados que nos ayudaran a calcular los umbrales para segmentación.

En **features** se encuentran los descriptores utilizados, **BSM** y **Zoning** y en **classifier** se realiza la clasificación de los descriptores tomando como referencia el archivo de datos correspondiente al tipo de descriptor utilizado.

Seguidamente se muestran los diagramas de clases y de flujo del programa.

2.4.2 Diagrama de clases

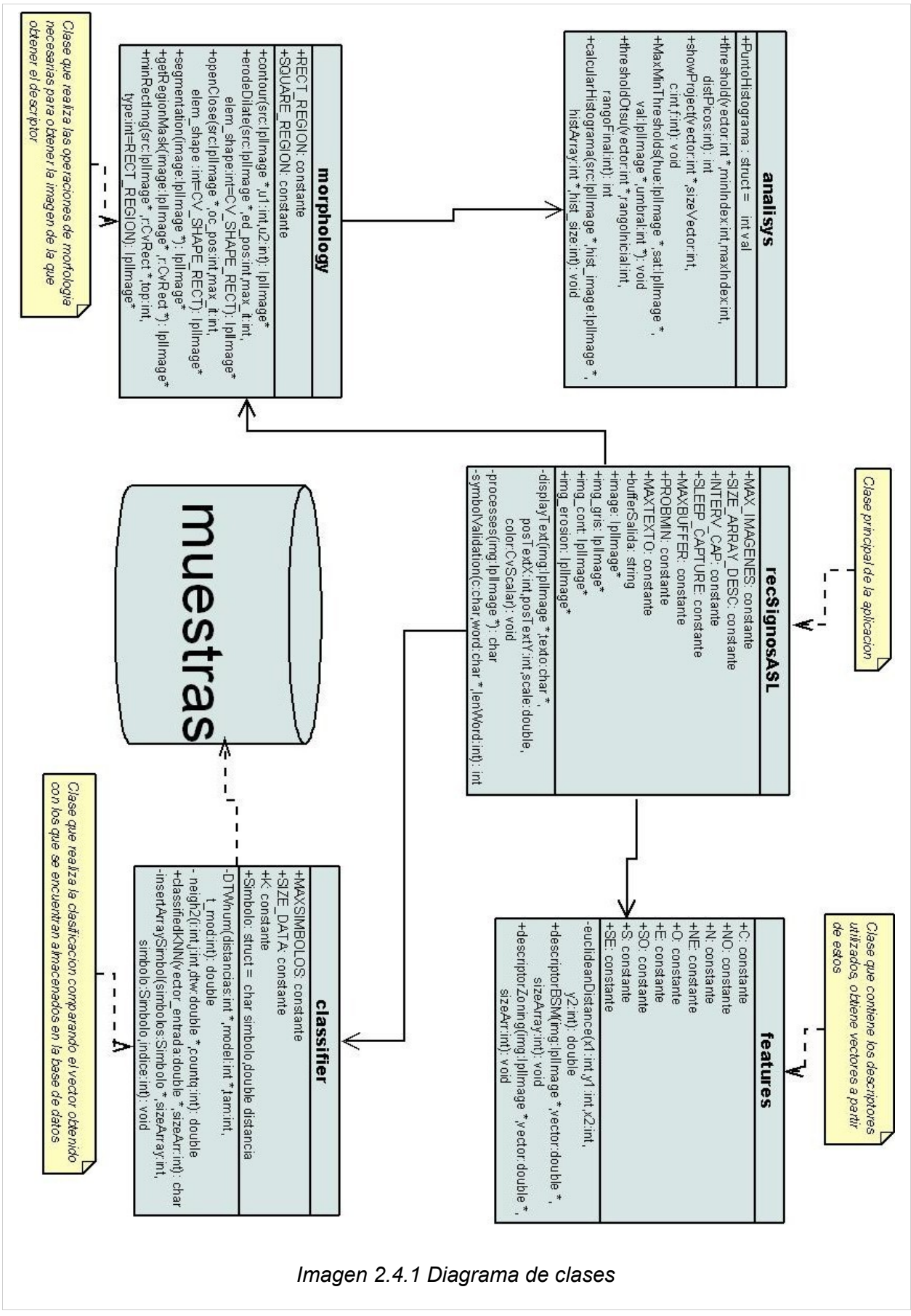


Imagen 2.4.1 Diagrama de clases

2.4.3 Diagramas de flujo

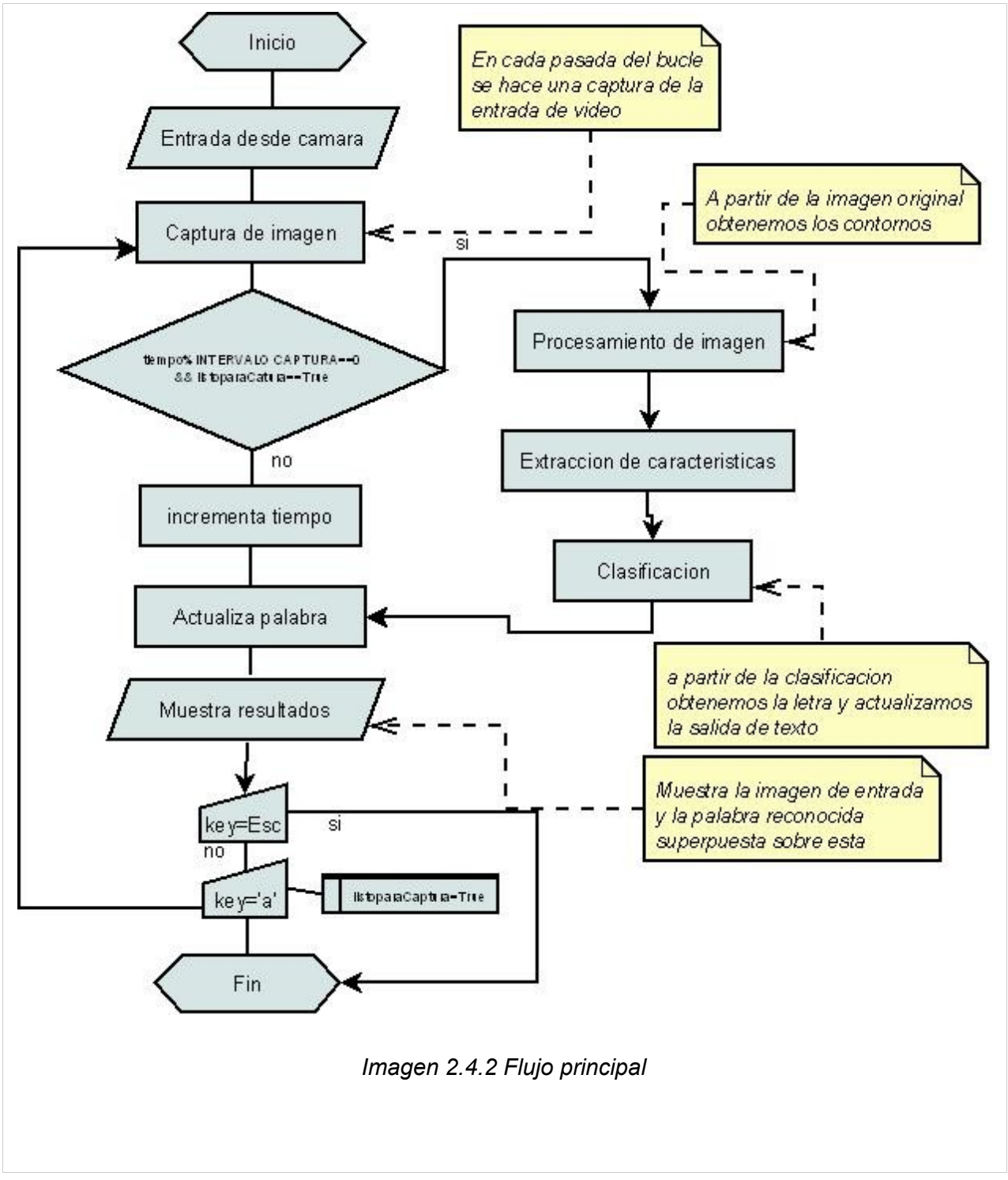


Imagen 2.4.2 Flujo principal

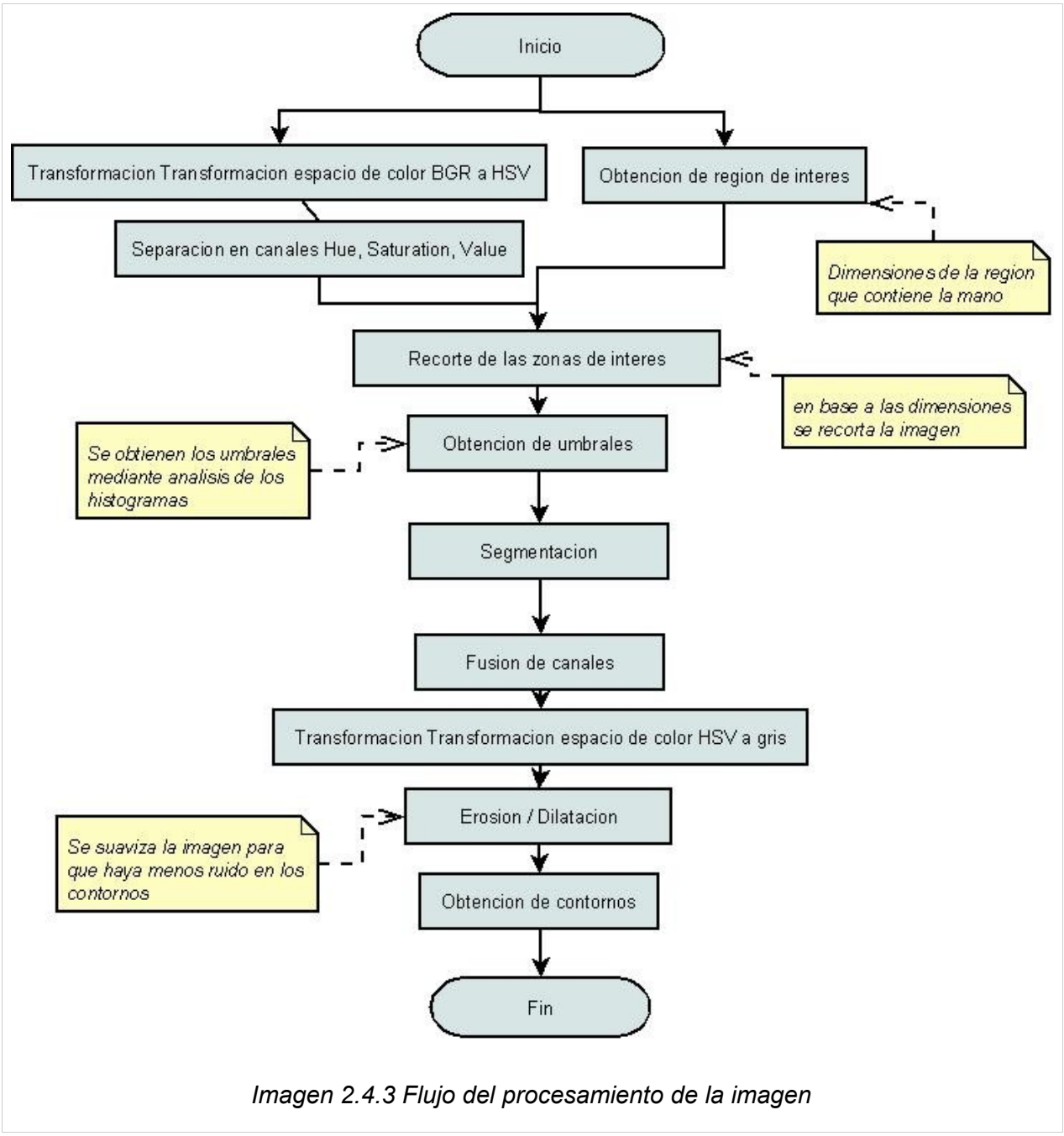


Imagen 2.4.3 Flujo del procesamiento de la imagen

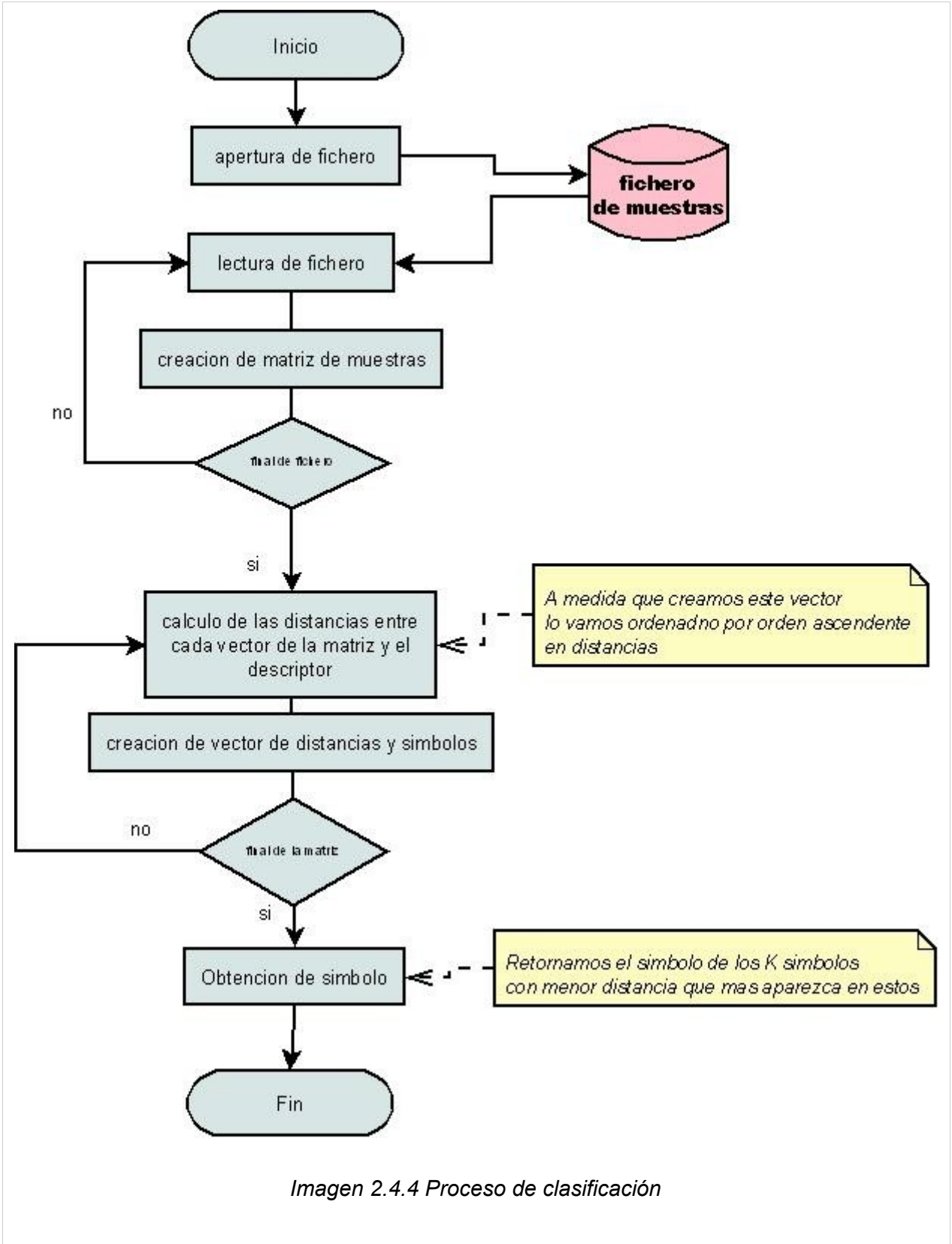


Imagen 2.4.4 Proceso de clasificación

En el siguiente diagrama de Gantt (Imagen 2.4.5) se da una estimación del tiempo empleado en cada una de las fases de la ejecución del proyecto

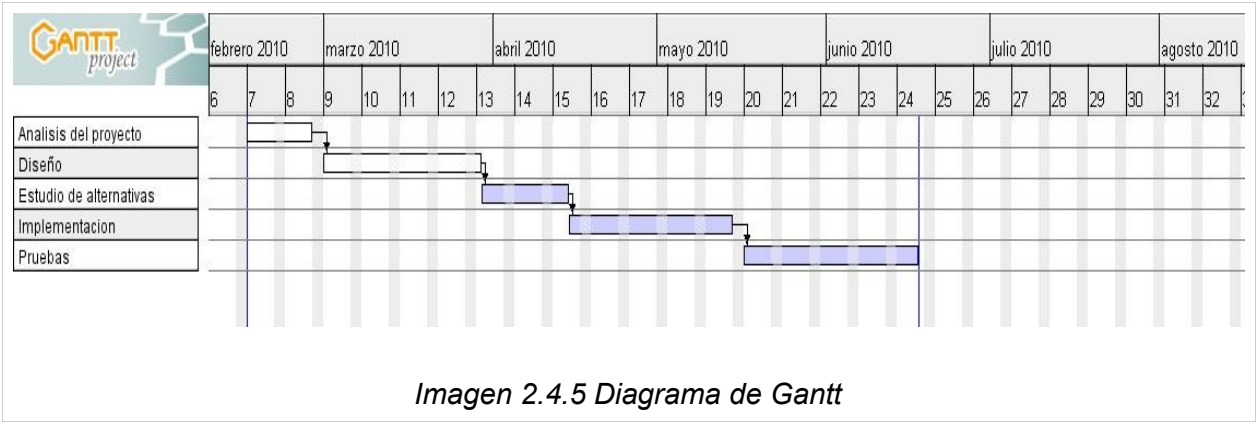


Imagen 2.4.5 Diagrama de Gantt

La implementación y compilación del proyecto se ha realizado con la ayuda del entorno de desarrollo Microsoft Visual Studio 2010 Express Edition de descarga gratuita.
Las librerías utilizadas pertenecen a la biblioteca de funciones para procesamiento de imágenes y video OpenCV bajo licencia BSD, versión 2.1.
Este documento se ha realizado con el procesador de textos de OpenOffice 3.1 bajo licencia BSD. Los gráficos se han realizado con la ayuda del programa InkScape, y Gimp también de licencia BSD. También se ha hecho uso del programa CamStudio también gratuito para realizar las capturas en video de la ejecución del programa.

Todo esto se ha realizado en un portátil Intel Centrino 2 bajo Plataforma Windows Vista.
Se han invertido un total de 25 horas aproximadamente en concepto de tutoria. Se ha invertido también un total de 200 horas aproximadas en la realización del proyecto por parte del alumno. De estas, 60 horas en la etapa de iniciación a la librería OpenCV, la fase de análisis y diseño. Otras 50 aproximadamente en el estudio de las posible alternativas para efectuar el proyecto y las restantes 90 para la implementación 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 2010. Las técnicas de procesamiento de imágenes y de captura de vídeo han sido realizadas con la ayuda de la librería gráfica OpenCV versión 2.1.La referencia de uso se puede consultar en http://opencv.willowgarage.com/documentation/cpp/index.html

El programa realiza capturas obtenidas de una webcam o una cámara de vídeo conectada al ordenador mediante conexión USB. También se han realizado pruebas a partir de videos filmados previamente. Estas capturas se realizan con un intervalo que puede variar de una décima de segundo a 1 segundo aproximadamente. Para las validaciones se usan diversos métodos según se hagan las capturas a partir de video o en tiempo real .Para los videos se tomaran una serie fija de capturas, una vez completadas se dará por valido el símbolo que mas veces aparezca. En tiempo real se dará por bueno el símbolo que se haya obtenido mediante clasificación N veces, a partir de ahí el usuario puede interpretar un nuevo símbolo .

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. 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. Dicho umbral es obtenido de forma automática mediante algoritmos simples de búsqueda de umbrales a partir de la forma del histograma de la imagen. Aun así las capturas han de realizarse en un entorno mínimamente controlado, con el fondo inmediatamente posterior a la mano mas oscuro que esta y de un tono fuera del rango de los posibles valores asignados a la piel. El resultado de los tres canales se fusiona y se pasa a una imagen en gris que solo deberá contener la mano con el fondo en negro.

• **Suavizado de la imagen :** Se realiza un suavizado previo de la imagen para eliminar ruidos antes de buscar los contornos mediante las operaciones de morfología erosión y dilatación, en este orden. A pesar de que el algoritmo utilizado para la detección de bordes utiliza un suavizado previo mediante un operador morfológico, la utilización única de este algoritmo sin el paso anterior genera problemas para todos los parámetros utilizados.

• **Detección de bordes :** Para esto utilizamos un algoritmo que viene implementado directamente en la librería OpenCv. Este algoritmo llamado de Canny realiza la búsqueda de los bordes de la imagen mediante varios pasos teniendo en cuenta dos parámetros de umbral. En nuestro caso estos valores serán aproximadamente de 50 y 26.

• **Extracción de características:** A partir de la imagen obtenida mediante la detección de bordes obtenemos los descriptores que guardaremos en vectores. El vector de datos conseguido tendrá tantos elementos como el tamaño en pixeles del lado de la imagen multiplicado por cuatro. Para el descriptor Zoning las imágenes de las que obtenemos el vector serán cuadradas, de un tamaño igual al máximo entre la altura y la anchura. Para el descriptor Blurred Shape Model , las imágenes también serán de un tamaño de 4x4, a partir del menor rectángulo contenedor, así como el vector obtenido

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 directa. 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 los ficheros “dataBSM.dat” y “dataZoning.dat”, para BSM y Zoning respectivamente, que se encuentran en la carpeta “\files”. 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 30 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 pixel. Para cada uno de los canales se utilizan estos métodos de segmentación y valores de umbral:

•**Hue:** Utilizamos segmentación 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 40.

•**Saturación:** Usamos también segmentación 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. Obtenemos el umbral mediante el método de máximos y mínimos.

•**Value:** En este caso hacemos una segmentación , discriminamos así los valores mas oscuros de la imagen que son los que están en el nivel más bajo de la escala. Obtenemos el umbral de la misma forma que el anterior.

Para la extracción de contornos se aplica la función cvCanny incluida en la biblioteca OpenCV. Los parámetros para histéresis de umbral se sitúan en los valores de 50 para el primer umbral y 26 para el segundo.

De las imágenes resultantes se extraen vectores de 4x4 elementos. Esto implica que de cada imagen se obtiene el descriptor para 4x4 regiones. Este tamaño permite que el tiempo de calculo sea menor y obtiene una buena clasificación. También permite obtener la clasificación a partir de un tamaño mayor de muestras.

Los vectores se clasifican tomando como modelo los datos que se encuentran en los ficheros “dataBSM.dat” y “dataZoning.dat” situados en la carpeta “\files”, con un total de 15 muestras para cada uno de los 30 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'**.También se incluyen los números **'3','4','5','7','8','9'**.Los números **'1','2'**, y **'6'** se expresan mediante el mismo símbolo que las letras **'d', 'v'** y **'w'** respectivamente

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

El clasificador utilizado se basa en el algoritmo **K nearest neighbors** .Este algoritmo compara el descriptor obtenido con cada modelo de la base de datos. De entre los K modelos que mas se acercan damos por valido el que mas se repita en estos. El tamaño de K elegido influye en la fiabilidad del resultado. Dependiendo de la base de datos, del numero de modelos por cada clase y de las variaciones en las condiciones de captura para cada modelo, entre otras cosas, es conveniente lograr un valor de K eficiente (Imagen 3.3.1).

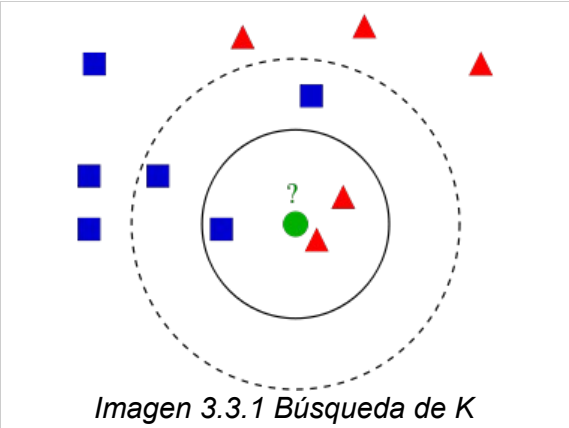


Imagen 3.3.1 Búsqueda de K

El carácter obtenido mediante la clasificación se pasa a una función que va guardando los resultados en un buffer. De las N clasificaciones guardadas en este buffer se da por valida la clase que mas aparezca con una probabilidad establecida,Este método mejora la posibilidad de obtener la clase valida.

Los parámetros que se pueden variar en la ejecución del programa son estos(*Imagen 3.3.2*):

- Tipo de clasificación:BSM o Zoning.
- Rango de iluminación:Nos permite colocar los valores máximo y mínimo a la hora de buscar el umbral para el canal Value. Este canal es el que se ve afectado por las diferentes condiciones de iluminación.
- Intervalo de captura: Tiempo de espera entre cada clasificación.
- Vecinos en la clasificación: Nos permite variar el valor de K (elementos mas cercanos al modelo).
- Buffer de validación:Tamaño del string sobre el cual buscaremos el símbolo que se mostrara como valido.
- Probabilidad de ocurrencias de símbolo: Veces que tiene que estar una clase en el buffer para que se de por valida , de 0 a 1 (ejemplo , buffer de 5 caracteres con probabilidad de 0.8, si aparece una a 4 veces es valida, si aparece 3 no).

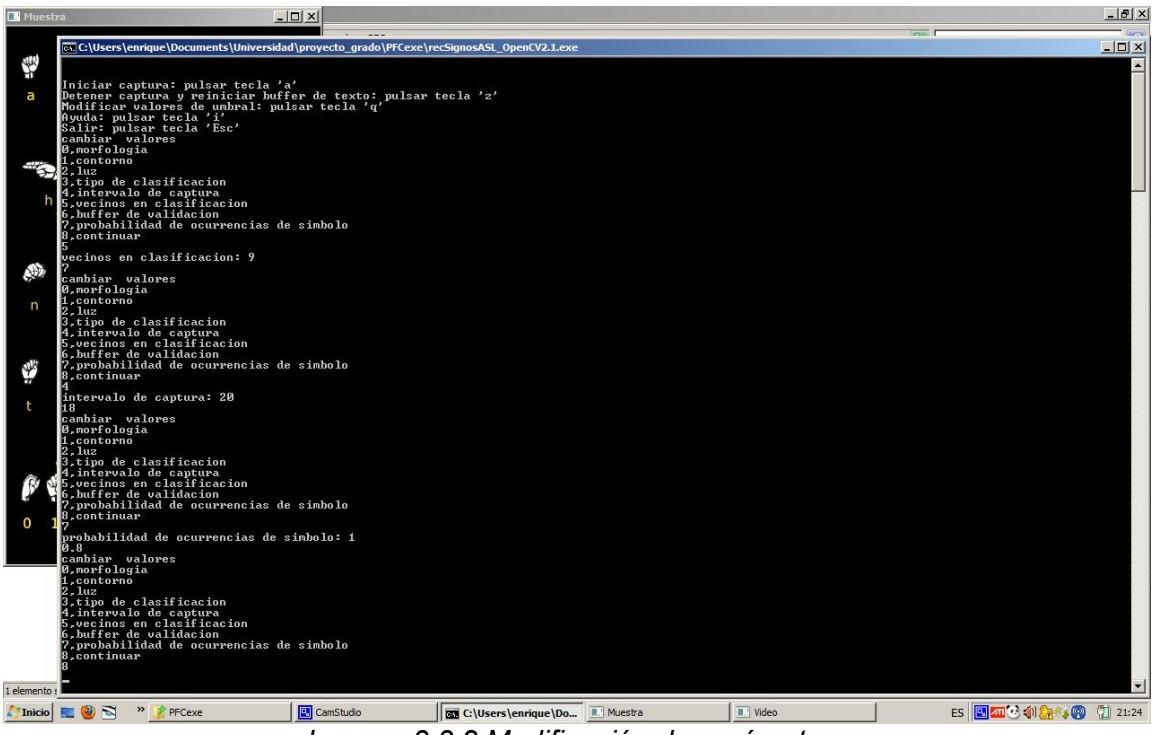


Imagen 3.3.2 Modificación de parámetros

En la imagen 3.3.3 se muestra una captura de la ejecución del programa. En la ventana principal se muestra la captura de video y el texto. En la parte inferior de la ventana se encuentran todos los símbolos que han resultado en el clasificador, en la parte superior, los símbolos que se dan por validos (en rojo). Aquí vemos como en la parte inferior de la ventana el string finaliza en 'ooooo' y l ultima letra de la palabra superior es 'o'.A medida que se llena la pantalla se van vaciando por el principio los strings

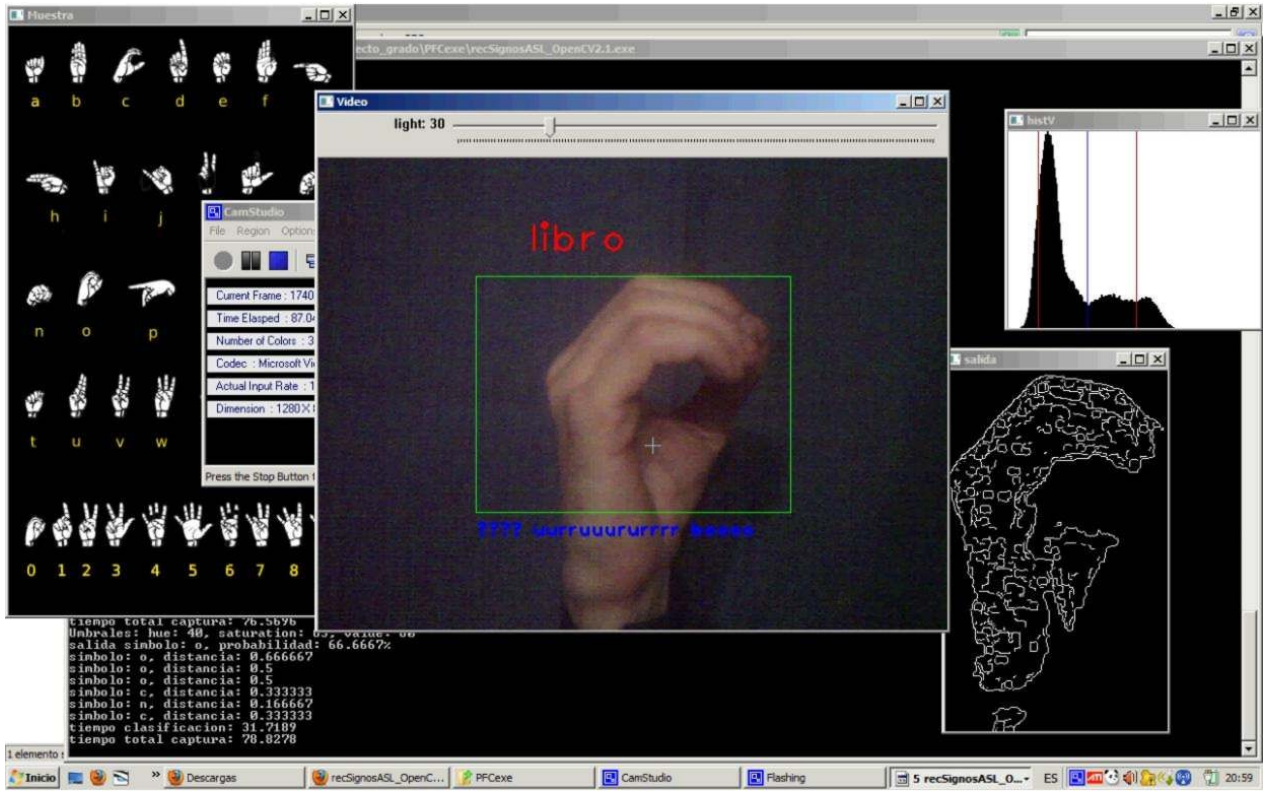
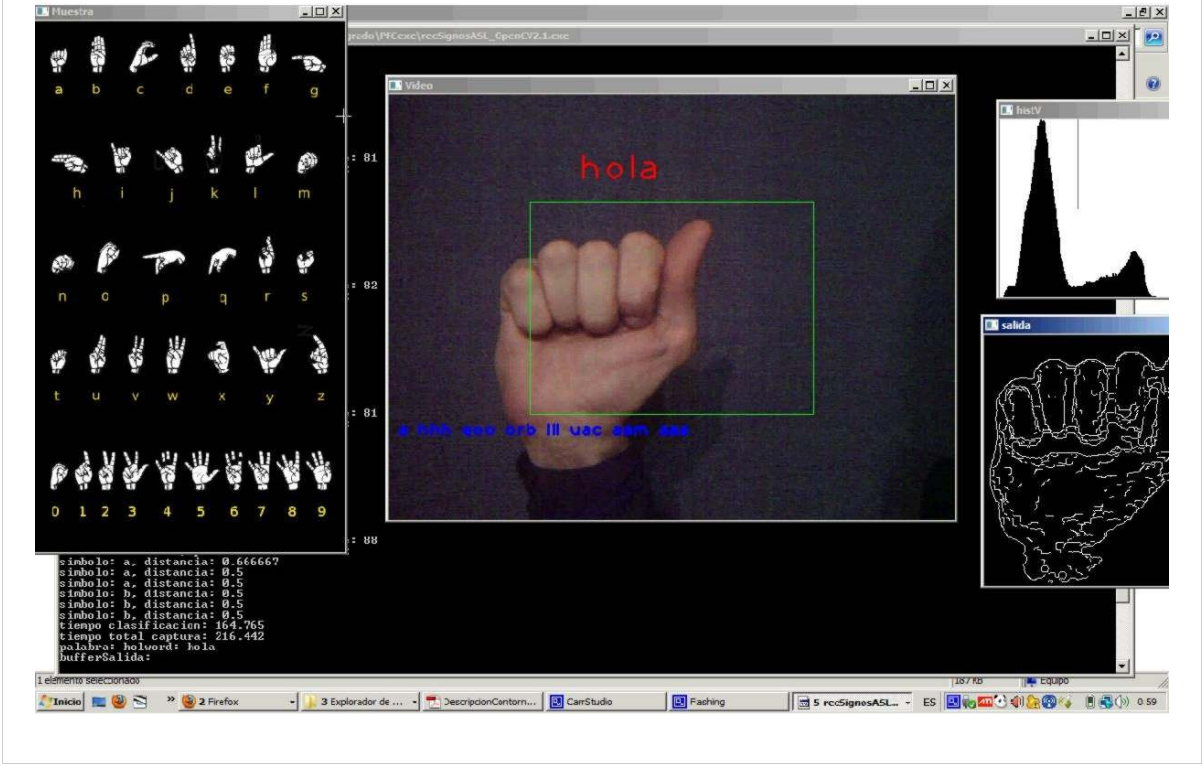


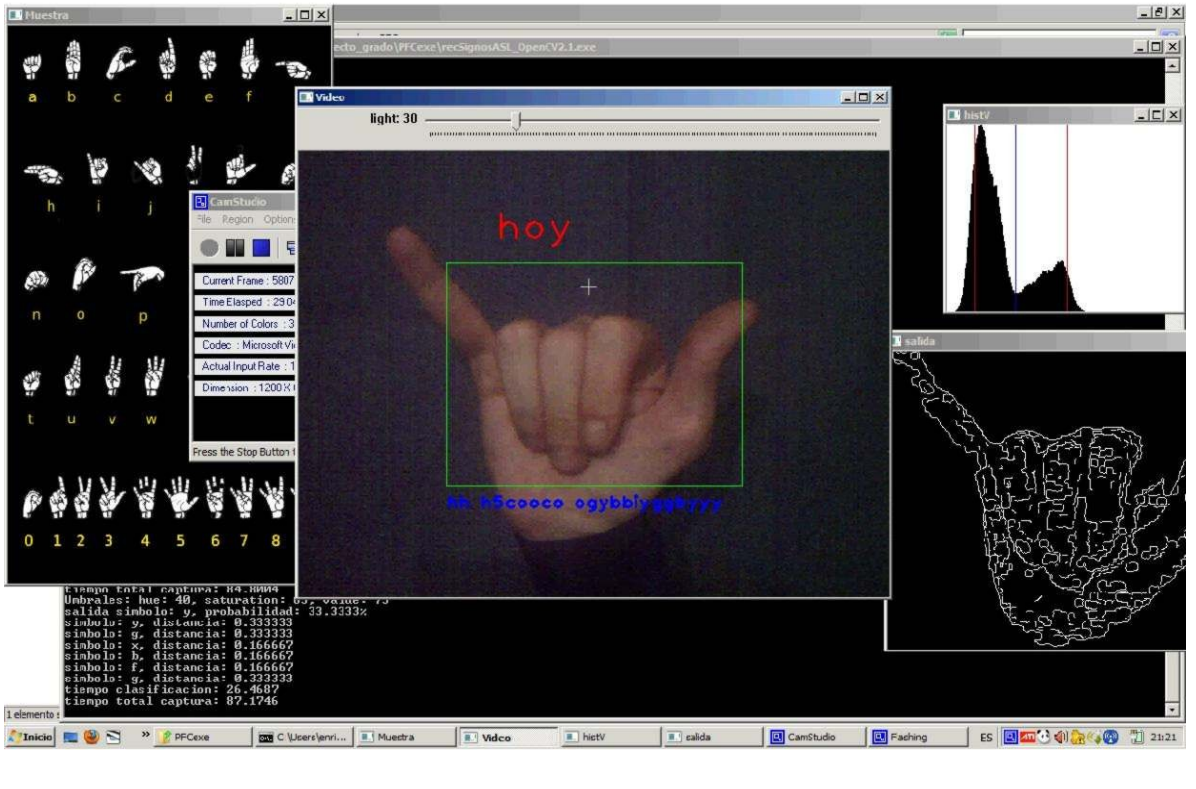
Imagen 3.3.3.Ejecución del programa.

En la parte derecha se muestra una ventana con el contorno de la imagen clasificada. El contorno nos permite ver que la segmentación se ha hecho correctamente.

Se han realizado varias pruebas con el descriptor **Zoning** y con el descriptor **Blurred Shape Model**. A continuación se muestran los resultados.

A continuación se muestran varias capturas de algunas de las pruebas realizadas. El fallo en la clasificación generalmente solo incrementa el tiempo hasta que se validan las letras. Un valor grande de buffer hace menos probable que se de un símbolo erróneo por valido, con valores pequeños esto si ocurre.


Muestra
Video
hola
histV
salida
simbolo: a, distancia: 0.666667
simbolo: a, distancia: 0.5
simbolo: a, distancia: 0.5
simbolo: b, distancia: 0.5
simbolo: b, distancia: 0.5
simbolo: b, distancia: 0.5
tiempo clasificacion: 164.765
tiempo total captura: 216.442
palabra: holword: hola
bufferSalida:


Muestra
Video
light: 30
hoy
histV
salida
CamStudio
Current Frame : 5807
Time Elasped : 290
Number of Colors : 3
Codec : Microsoft Vi
Actual Input Rate : 1
Dimension : 1200 X
Press the Stop Button
Umbrales: hue: 40, saturation:
salida simbolo: y, probabilidad: 33.3333%
simbolo: y, distancia: 0.333333
simbolo: g, distancia: 0.333333
simbolo: x, distancia: 0.166667
simbolo: b, distancia: 0.166667
simbolo: f, distancia: 0.166667
simbolo: g, distancia: 0.333333
tiempo clasificacion: 26.4687
tiempo total captura: 87.1746

3-4. Comentarios

Para los dos descriptores utilizados se ha encontrado como mas optimo el valor de 5 en la K para el algoritmo de clasificación , para un total de 15 modelos por clase.
Para símbolos muy parecidos es mucho mas problemático el descriptor Zoning .El Blurred Shape Model muestra efectividad incluso con símbolos como la 'g' y la 'h' o la 'o' y la 'c' y los valida sin ningún problema (Imágenes 3.4.1, 3.4.2) .

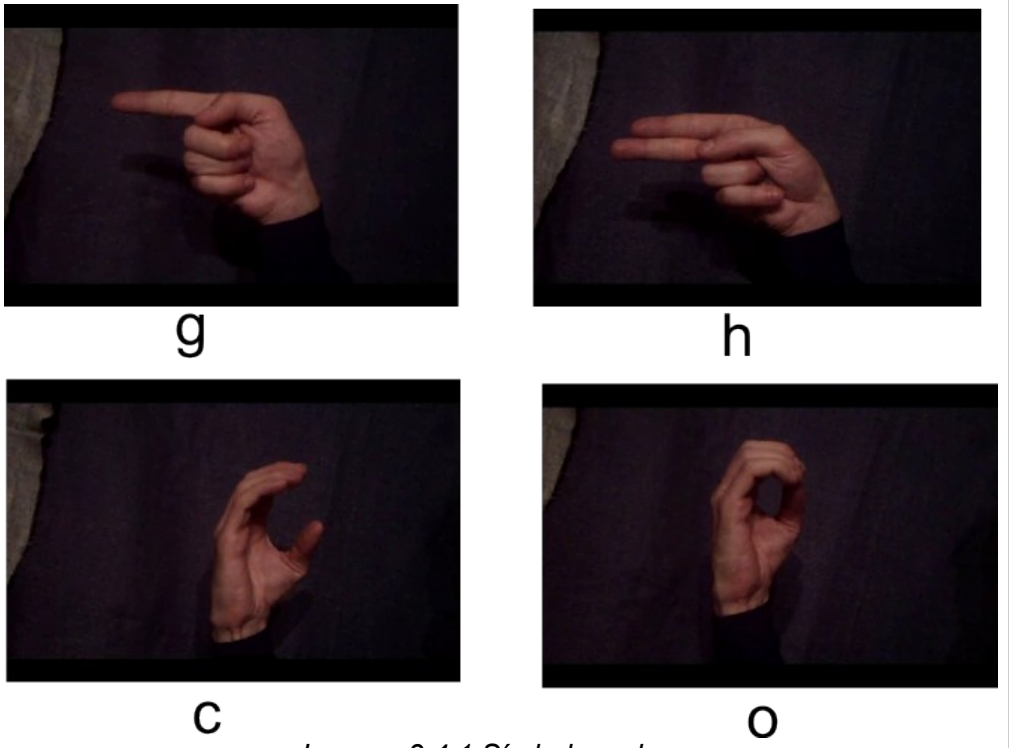


Imagen 3.4.1 Símbolos g,h,c,o

El descriptor Zoning para es muy poco efectivo con signos mínimamente similares, teniendo en cuenta que el tamaño del descriptor es de 4x4 elementos , esto aumenta las probabilidades de error .Para reducir esto es preciso aumentar el tamaño del descriptor lo que implica mayor tiempo de calculo.

Imagen 3.4.2 Simbolos g,h,c,o , salida en ejecución

4.conclusión y lineas futuras

Para contornos de imágenes obtenidos de manera correcta, el algoritmo Blurred Shape Model demuestra ser mas eficiente que otros descriptores, incluso para símbolos muy parecidos en forma. El tamaño del descriptor también hace que el tiempo de calculo en la clasificación no sea muy elevado.

Para obtener las características de forma aceptable es preciso obtener el contorno de la imagen a procesar de forma totalmente limpia. La parte mas importante del proceso es la segmentación. Separar la imagen del fondo de forma conveniente es complicado en entornos no controlados. El algoritmo de segmentación utilizado se muestra eficiente en bastantes condiciones, pero aun tiene limitaciones, como el uso de un tipo de fondo adecuado. El uso de otros algoritmos como Adaptative Median, segmentación de la piel por probabilidades o distancia Mahalanabis, hace mas robusto este proceso, aunque mas costoso en tiempo de calculo.

El algoritmo de clasificación utilizado para los símbolos una vez clasificados es muy efectivo ya que realiza varias clasificaciones antes de dar por buena una, pero dependiendo de la cantidad de modelos y de los parámetros puede resultar muy lento. El algoritmo del que hace uso (K-nearest neighbors) muestra su eficiencia a través del ensayo y error, no aprende de si mismo, aunque es efectivo para este caso particular .Introduciendo métodos de aprendizaje se hace la clasificación mas rápida y eficiente con el tiempo.

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.

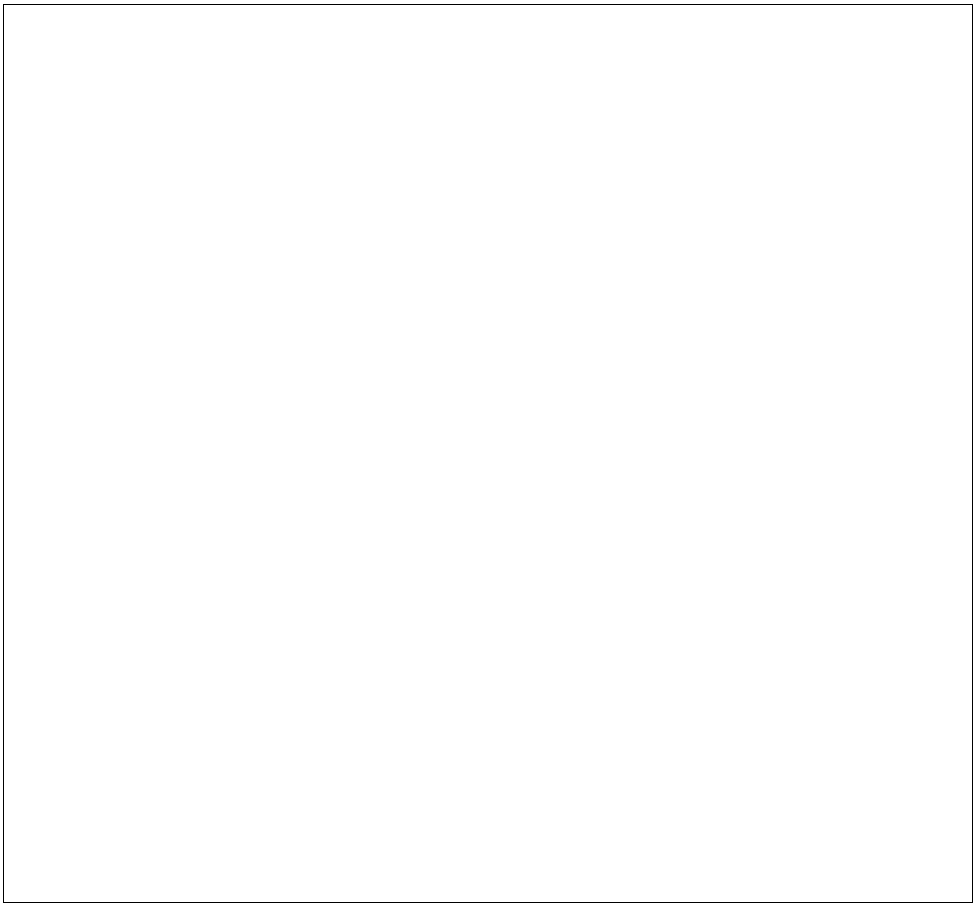
Concluyendo, este trabajo puede ser positivo para facilitar la comunicación entre personas que usan el el lenguaje de signos como medio de comunicación y los que no lo usan, como si de un traductor de idiomas o medio de aprendizaje se tratara.

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 2.1 (utilizada en la realización del proyecto).
- Código fuente en c++ optimizado para el entorno de desarrollo Microsoft Visual Studio 2010.
- Copia en pdf de la memoria.

6.CD



6.Bibliografia

[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 and M. Baudouin-Lafon, “Charade: Remote Control of Objects Using Free-Hand Gestures,” Comm. ACM, vol. 36, no. 7, pp. 28-35, 1993. Download PDF

[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) *Reduction 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.

[14] Mario Satz (1982)"Diseños Anatómicos de Leonardo da Vinci". Ed. Bencard.

[15] E. Muybridge (1955). "The Human Figure in Motion". Dover Publications.

[16] Bolt, R.A. (1980) *Put-that-there: Voice and Gesture in the graphics interface* Download PDF

[17] http://www.springerlink.com/content/p646513106164853/

[18]http://sociedadinformacion.fundacion.telefonica.com/DYC/SHI/seccion=1188&idioma=es_ES&id=2009100116310003&activo=4.do?elem=3997.

[19] http://www.universia.es/portada/actualidad/noticia_actualidad.jsp?noticia=106503

[20] http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.72.5279&rep=rep1&type=pdf

ENGINYERIA TÈCNICA EN INFORMÀTICA DE SISTEMES

UNIVERSITAT DE BARCELONA

Treball fi de carrera presentat el dia de de 200
a la Facultat de Matemàtiques de la Universitat de Barcelona,
amb el següent tribunal:

Dr. President

Dr. Vocal 1

Dr. Secretari

Amb la qualificació de: