Universidad Nacional del Litoral

Facultad de Ingeniería y Ciencias Hídricas

Reconocimiento automático de signos manuales mediante procesamiento digital de video

Proyecto Final de Ingeniería en Informática

Autor: Hernández Vogt, Juan Pablo
Director: Bioing. Martínez, Cesar
Co-Director: Ing. Albornoz, Enrique Marcelo

Santa Fe - 30 de julio de 2010

Dedicado...

a mi mamá Susana, por ser el ejemplo de
respeto, esfuerzo y humildad que
está siempre a mi lado.

a mi abuela Ana, por su paciencia inagotable,
aliento continuo y atención de
cada detalle que necesito.

Agradecimientos

Llegar a escribir estas líneas no hubiera sido posible sin la participación de una enorme cantidad de personas. Quiero brindar un reconocimiento...

A mi familia, que siempre está cerca en cada cosa que realizo y que sin importar los resultados me devuelven amor. Convivir con ellos es una dicha.

A mi novia, que supo ser una fundamental e incondicional colaboradora de este proyecto, y por sobre todo, una gran compañera en la vida.

A los amigos que encontré dentro de la facultad, con los que compartimos muchas horas de estudio y experiencias de vida, me brindaron tantas veces alojamiento y, por sobre todo, palabras de aliento para seguir adelante.

A los profesores de la facultad, muchos de los cuales fueron más que comunicadores de conocimientos específicos y tuvieron para con mi persona cariño y aliento.

Mi más sincera gratitud, a mi director el Bioing. César Martínez, y mi co-director el Ing. Marcelo E. Albornoz, por toda la confianza y el apoyo recibido, por guiarme y por todo el tiempo dispensado en indicar las correcciones necesarias que mejoraron completamente este informe.

Juan Pablo Hernández Vogt
Santa Fe, 30 de julio de 2010

Resumen

En el área del reconocimiento de patrones, un tópico de creciente interés es el reconocimiento de gestos realizados por personas. Los gestos realizados con la mano permiten la representación de un gran número de formas y la interpretación de éstas en un ordenador posibilita una interacción humano-computadora más natural.

Este informe presenta el diseño e implementación de un sistema para el reconocimiento de signos manuales en tiempo real. El procesamiento se realiza sobre el flujo de video obtenido de una cámara web estándar en un ambiente de trabajo natural.

El reconocimiento de un signo consiste en localizar la mano y segmentarla del fondo, para ésto se propone un algoritmo basado en el modelo de color HSV (*Hue*, *Saturation*, *Value*). Luego se analiza su morfología para obtener los parámetros que caracterizan al signo. Finalmente, un clasificador asigna una etiqueta al patrón correspondiente mediante la técnica del vecino más cercano por mínima distancia ponderada.

Para la evaluación de los algoritmos de segmentación y reconocimiento propuestos, se conformó un conjunto de 144 imágenes (a partir de las realizaciones de 16 signos distintos) extraídas del flujo de video, bajo condiciones de iluminación artificial. Se obtuvo una alta tasa de reconocimiento, tanto para distintas realizaciones de un mismo usuario como para realizaciones de diferentes usuarios.

La implementación final del software tiene algoritmos optimizados en C++ que permiten analizar un fotograma en aproximadamente 170 milisegundos en un PC AMD Athlon XP 2800+.

Prefacio

Esta memoria presenta el desarrollo de un sistema que explora las capacidades del procesamiento digital de imágenes para abordar el desafío de una integración humano-computadora prescindiendo de equipamiento especializado o de condiciones de laboratorio estrictas. La misma se encuentra organizada como se describe a continuación.

El capítulo 1 expone la motivación del tema elegido, los esfuerzos realizados por otras personas en la temática y los objetivos planteados al comenzar el trabajo.

En el capítulo 2 se exponen los conceptos y métodos que conforman el marco teórico del procesamiento digital de imágenes.

Los capítulos 3 y 4 exponen el desarrollo del sistema. El primero trata la estrategia propuesta (etapas y recursos), mientras que el segundo aborda la metodología del desarrollo de software que produce una implementación funcional.

A continuación el capítulo 5 presenta los detalles del corpus de imágenes registrado, los experimentos sobre hardware y software, y los resultados del desempeño del método propuesto.

Se consideró oportuno incluir en un capítulo adicional una exitosa implementación del método dentro de una aplicación gráfica real utilizando una biblioteca para GUI libre.

Finalmente, las conclusiones se exponen en el capítulo 7.

Índice general

Índice de figuras

Índice de tablas

CAPÍTULO 1

Introducción

Actualmente, una gran cantidad de dispositivos y máquinas requieren para su operación la ejecución de comandos específicos mediante activación manual, en los que tradicionalmente se utiliza la entrada táctil (teclado, ratón, tablero de control u otro).

El presente Proyecto Final de Carrera presenta un esfuerzo por el desarrollo de una interacción humano-computadora más natural que aproveche la capacidad humana de la gesticulación. Para lograrlo se emplean las potencialidades del procesamiento digital de imágenes las cuales resultan en sistemas más intuitivos y mejor aceptados por personas no expertas.

1.1 Motivación

En el área del reconocimiento de patrones, un tópico de creciente interés es el reconocimiento de gestos. Esta tarea consiste en la captura del movimiento corporal de una persona mediante cámaras u otros dispositivos, y la identificación de poses o movimientos particulares de la cabeza, mano, brazos, etc. Los gestos reconocidos pueden ser empleados como entrada en el control de máquinas, ser traducidos a otra forma de información (p. ej., en la traducción de lenguaje de señas a voz). Migrar la interacción humano-computadora a una manera más natural, empleando la capacidad humana de la gesticulación,

permitiría el desarrollo de interfaces más efectivas y amigables [1] [2].

El análisis y clasificación de gestos de la mano plantea algunas ventajas respecto del análisis de otras partes del cuerpo. La mano facilita la representación de un gran número de formas según la combinación de apertura y cierre de los dedos [3]. Estas características la convierten en una herramienta de interacción muy efectiva, habiendo sido utilizada en ambientes virtuales de navegación [4], simulación quirúrgica [5], generación de arte visual o música [6, 7], entre otros.

1.2 Estado del arte

Un método ampliamente difundido para capturar los movimientos de la mano está dado por los denominados *data gloves*, guantes dotados de sensores magnéticos o electromecánicos que miden localización y ángulos de una gran cantidad de puntos [8]. Sin embargo, a pesar de su efectividad, estos sistemas tienen un alto costo, necesitan un período extenso de calibración y ajuste, y la persona debe ser entrenada ya que su uso condiciona los movimientos naturales del cuerpo [9].

La aplicación de las técnicas de visión computacional intenta reemplazar la utilización de los métodos sensoriales con algoritmos de análisis de imágenes fijas o de video. Estos métodos permiten una mayor comodidad en el uso, son más intuitivos y mejor aceptados por usuarios no expertos [10]. Las técnicas visuales se pueden dividir en enfoques 2D y 3D [11]. El enfoque 3D utiliza un conjunto de videocámaras para reconstruir los gestos de la mano en el espacio, con la desventaja de presentar una alta demanda de potencia de cálculo. El enfoque 2D –una sola cámara– permite formular métodos menos complejos que facilitan la implementación en tiempo real, conservando muchas de las funcionalidades del enfoque anterior.

En los sistemas basados en el enfoque 2D, al menos dos bloques son claramente diferenciables. En primer lugar, una etapa de adquisición y segmentación de la imagen, que consiste en identificar la silueta de la mano y separarla del fondo. La segunda etapa consiste en la extracción de características y la clasificación: aquí se representa la información distintiva de la seña por un conjunto reducido de parámetros que sirve de entrada a un clasificador.

Respecto de la primera etapa, algunos trabajos imponen un alto número de restricciones al procedimiento de adquisición, tales como condiciones de

iluminación ideales, fondos de escena uniformes y dispositivos de captura de alta calidad, entre otras [12, 13]. Otros autores [14], utilizan varias imágenes tomadas desde distintos ángulos para efectuar una segmentación inicial más precisa en una representación tridimensional.

En la segunda etapa se pueden encontrar métodos muy diferentes, donde la aplicabilidad de cada uno está fuertemente asociada a la calidad de la imagen obtenida en la primera etapa. Entre los distintos enfoques de clasificación, se pueden encontrar aproximaciones basadas en redes neuronales [13], modelos ocultos de Markov [15] y teoría de grafos [16], entre otros.

1.3 Objetivos del Proyecto Final

Los objetivos generales de este trabajo son los siguientes:

- Adquirir nuevos conocimientos acerca de aplicaciones específicas innovadoras de técnicas de procesamiento digital de imágenes.
- Aplicar los conocimientos adquiridos en el transcurso de la carrera a un proyecto que utilice técnicas de procesamiento digital de imágenes, que realice un aporte ingenieril a la comunidad en general.

Los objetivos específicos son los siguientes:

- Implementar rutinas de manejo de cámara web para adquisición de flujo de video.
- Diseñar e implementar rutinas básicas para definición de la silueta de la mano.
- Analizar, seleccionar, implementar y adaptar algoritmos de segmentación, morfología, procesamiento en color y otros para extracción de características de la mano.
- Definir e implementar técnicas de reconocimiento de patrones para identificación de la seña.
- Diseñar e implementar una interfaz gráfica que haga accesible al usuario final su utilización.

1.4 Alcances del Proyecto Final

Este proyecto no trata el reconocimiento de gestos dinámicos, formados por una secuencia temporal de signos estáticos. El sistema tampoco plantea la detección de signos realizados con ambas manos simultáneamente o en combinación con gestos faciales.

CAPÍTULO 2

Fundamentos del Procesamiento Digital de Imágenes

Se puede definir al *procesamiento digital de imágenes (PDI)* como el uso de métodos y algoritmos de computadora para la manipulación de imágenes digitales. Entre ellos podemos mencionar la convolución (que es base, a su vez, de muchos otros algoritmos), transformada rápida de Fourier (FFT, del Inglés *Fast Fourier Transformation*), transformada coseno discreta (DCT, del Inglés *Discrete Cosine Transformation*), detección de bordes, mejora del contraste, restauración, etc. En aquellos casos donde se requiere una gran velocidad de procesamiento, estos algoritmos son implementados directamente sobre un hardware dedicado.

En la actualidad el procesamiento digital ha reemplazado al analógico, no sólo por su velocidad sino porque hace posible ejecutar algoritmos más complejos, es más flexible en la actualización de parámetros y/o cambios en el diseño, reduce el costo económico, e incluso permite implementar métodos que serían imposibles en el domino analógico.

Entre las principales aplicaciones del PDI se encuentran: la mejora de la calidad pictórica de imagen (para facilitar la interpretación humana), el procesamiento automático de la escena (mediante el desarrollo de máquinas autómatas), y el almacenamiento y transmisión eficiente de imágenes.

El PDI incorpora conceptos de varias ciencias como física, matemática, computación e ingeniería de software, entre otras. Tiene superposición con metrología, reconocimiento de patrones e inteligencia computacional; por lo tanto su capacidad para desarrollar aplicaciones multidiciplinares es muy

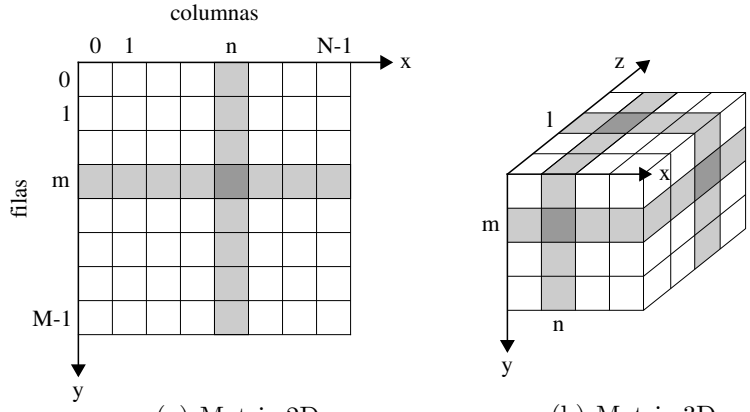
grande. No hay límites claros entre el procesamiento de imágenes y otras áreas relacionadas, más aun, no hay un consenso general respecto de dichos límites [17]. Sin embargo, una forma útil de considerar los niveles de procesamiento es mediante la utilización de un paradigma que los divide en tres tipos:

- Nivel bajo, o procesamiento de imágenes: la entrada y la salida son imágenes (limpieza de ruido, mejora del contraste, transformación espacial).
- Nivel medio, o análisis de imágenes: la entrada es una imagen y la salida son atributos (segmentación, descripción, clasificación).
- Nivel alto, o visión computacional: se procesan funciones cognitivas sobre los atributos de entrada, las salida son decisiones. Es parte de un área mayor llamada inteligencia computacional.

2.1 Representación

A diferencia de los humanos, que forman las imágenes a través del sistema óptico sensando luz visible, las computadoras pueden operar con imágenes generadas por una gran variedad de fuentes. Se utiliza prácticamente todo el espectro electromagnético y fuentes a las que no se asocian imágenes de forma intuitiva como ser el ultrasonido, microscopía electrónica, etc.

Un ejemplo típico de un dispositivo de captura de imágenes es la cámara digital. En ella se dispone una matriz de pequeños sensores que captan la energía lumínica recibida y la transforman en una respuesta eléctrica. Cuantizando cada una de las respuestas se obtiene lo que se denomina *imagen digital*, la cual se repesenta como una matriz $f(x, y)$ con M filas y N columnas (Fig. 2.1(a)). Cada uno de los puntos (x, y) se denomina píxel (del Inglés *picture element* – elemento de imagen) y su amplitud $f(x, y)$ es el valor de la imagen en dicha coordenada. En la notación (x, y) se toma $(0, 0)$ como el origen de coordenadas y $(1, 0)$ como el segundo valor de la primera fila. Para imágenes con más de un valor por píxel (como imágenes a color o las compuestas por sensado en distintas bandas del espectro), se incorpora una nueva dimensión y se representa como una matriz tridimensional $f(x, y, z)$ [18] (Fig. 2.1(b)).



(a) Matriz 2D (b) Matriz 3D

Figura 2.1: Representación de imágenes digitales por medio de arreglos de puntos discretos en una grilla rectangular.

2.2 Modelo de color

Un modelo de color es una forma matemática de describir los colores mediante el uso de números, permitiendo la especificación de un sistema de coordenadas y un subespacio dentro del cual cada color es representado mediante un punto. Entre los modelos más conocidos están: RGB (Red, Green, Blue), CYMK (Cyan, Magenta, Yellow, blacK), HSV (Hue, Saturation, Value), HSI (Hue, Saturation, Intensity), CIE $L^*a^*b^*$ (CIELAB), NCS, Pantone®.

Cada modelo está definido según propósitos específicos. Por ejemplo, para la manipulación de colores en dispositivos hardware como monitores y cámaras digitales, se utiliza RGB; CMYK para la impresión a color. Modelos como HSV o HSI facilitan la manipulación del color a los seres humanos y son utilizados en múltiples algoritmos.

2.2.1 El modelo de color RGB

El modelo RGB describe la composición de un color en términos de la intensidad de los colores primarios (rojo, verde, azul) con los que se forma. Es un modelo de color basado en la síntesis aditiva, con el que es posible representar un color mediante la mezcla por adición de los tres colores primarios de la luz.

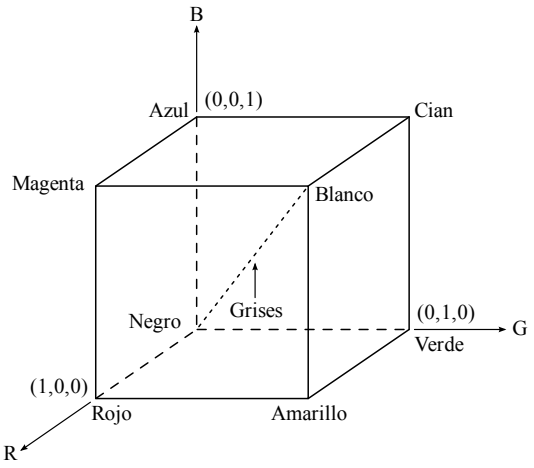


Figura 2.2: Representación espacial del modelo de color RGB.

Este modelo no define por sí mismo lo que significa exactamente rojo, verde o azul, por lo que los mismos valores RGB pueden mostrar colores notablemente distintos en diferentes dispositivos. Aunque utilicen un mismo modelo de color, sus espacios de color pueden variar considerablemente.

Su representación se basa en un sistema de coordenadas Cartesiano donde cada una de las componentes de color se corresponde a cada uno de los ejes. El subespacio de color que interesa se corresponde con el cubo mostrado en la Fig. 2.2. Para indicar con qué proporción mezclamos cada uno, se asigna un valor a cada uno de los colores primarios. De esta manera, por ejemplo, el valor 0 significa que no interviene en la mezcla y, a medida que ese valor aumenta, se entiende que aporta más intensidad a la mezcla. Aunque el intervalo de valores podría ser cualquiera (valores reales entre 0 y 1, valores enteros entre 0 y 37, etc.) es frecuente que cada color primario se codifique con un byte (8 bits). Así el rango de valores de intensidad va desde el 0 al 255 para cada componente de color.

Si bien el modelo RGB es adecuado para especificar el color que se muestra en una pantalla o que es obtenido por un sensor, éste no se condice con la manera natural que se tiene de describir un color. No hay forma de especificar un color parecido al rojo sangre, verde manzana o yema de huevo. Por este motivo se han creado algunos modelos que parten del RGB hacia otros más útiles.

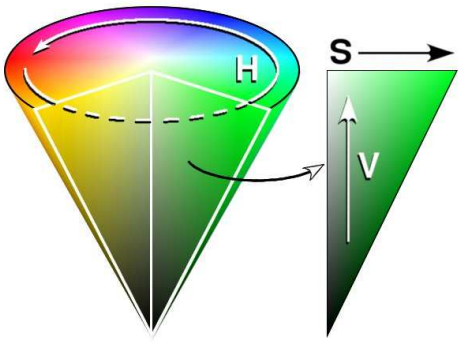


Figura 2.3: Representación espacial del Modelo de color HSV.

2.2.2 El modelo de color HSV

El modelo HSV (cuyas componentes en castellano son Tono, Saturación, Valor), creado en 1978 por Alvy Ray Smith, se define como una transformación no lineal desde el espacio de color RGB, en el que se pueden usar progresiones de color. La clave está en que desacopla la información del color e intensidad y se aproxima así a la forma en que los humanos describen un color. Como resultado, el modelo HSV es una poderosa herramienta para el desarrollo de algoritmos basados en descripciones más naturales e intuitivas del color.

Este modelo puede representarse gráficamente como un cono donde las componentes de éste son (Fig. 2.3):

- **Tono**: el tipo de color (como rojo, azul o amarillo). Se representa con un círculo donde el ángulo indica el valor (de 0° a 360°). Ejemplos: 0° es rojo, 60° es amarillo y 120° es verde.
- **Saturación**: se representa como la distancia al eje de brillo negro–blanco. Los valores posibles van del 0 al 100 %. A este parámetro también se le suele llamar "pureza" por la analogía con la pureza de excitación y la pureza colorimétrica de la colorimetría. Cuanto menor sea la saturación de un color, mayor tonalidad grisácea habrá y más decolorado estará. Por eso es útil definir la insaturación como la inversa cualitativa de la saturación.
- **Valor**: es el brillo del color. Representa la altura en el eje blanco–negro. Los valores posibles van del 0 al 100 %. El 0 siempre es negro. Depen-

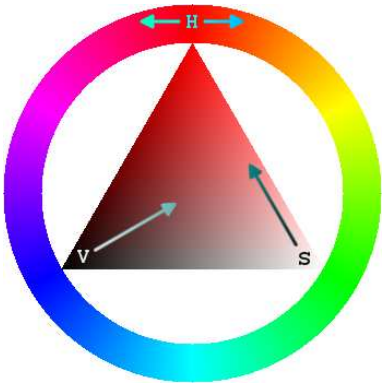


Figura 2.4: Espacio de color HSV como una rueda de color.

diendo de la saturación, 100 podría ser blanco o un color más o menos saturado.

Es común que se desee elegir un color específico para alguna de nuestras aplicaciones y cuando es así resulta muy útil usar la rueda de color HSV (Fig. 2.4). En ella la tonalidad se representa por una región circular y una región triangular separada que puede ser usada para representar la saturación y el valor del color. Normalmente, el eje paralelo a un lado del triángulo denota la saturación, mientras que perpendicularmente se establece el eje corresponde al valor del color. De este modo, un color puede ser elegido al tomar primero la tonalidad de la región circular, y después seleccionar la saturación y el valor del color deseados de la región triangular.

Transformación RGB a HSV

Es posible moverse de un espacio de color a otro mediante ecuaciones de transformación. Sea MAX el valor máximo de las componentes (R, G, B), y MIN el valor mínimo de éstas, las componentes del espacio HSV se pueden calcular como:

$$H = \begin{cases} 0, & \text{si } MAX = MIN \\ 60° \times \frac{G-B}{MAX-MIN} + 0°, & \text{si } MAX = R \\ & \text{y } G \geq B \\ 60° \times \frac{G-B}{MAX-MIN} + 360°, & \text{si } MAX = R \\ & \text{y } G < B \\ 60° \times \frac{B-R}{MAX-MIN} + 120°, & \text{si } MAX = G \\ 60° \times \frac{R-G}{MAX-MIN} + 240°, & \text{si } MAX = B \end{cases}$$

$$S = \begin{cases} 0, & \text{si } MAX = 0 \\ 1 - \frac{MIN}{MAX}, & \text{en otro caso} \end{cases}$$

$$V = MAX$$

Transformación HSV a RGB

Dado un color definido por los valores (H, S, V) en el espacio HSV, donde $H \in [0, 360)$, S y $V \in [0, 1]$, la correspondencia con los valores (R, G, B) del espacio RGB se calculan como [19]:

$$RGB = \begin{cases} (v, t, p), & \text{si } H_i = 0 \\ (q, v, p), & \text{si } H_i = 1 \\ (p, v, t), & \text{si } H_i = 2 \\ (p, q, v), & \text{si } H_i = 3 \\ (t, p, v), & \text{si } H_i = 4 \\ (v, p, q), & \text{si } H_i = 5 \end{cases}$$

donde H_i, f, p, q y t son obtenidos como:

$$H_i = \left[\tfrac{H}{60}\right] \bmod 6,$$
$$f = \tfrac{H}{60} - H_i,$$
$$p = V \times (1 - S),$$
$$q = V \times (1 - f \times S),$$
$$t = V \times (1 - (1 - f) \times S),$$

y $[\cdot]$ es el operador de truncado.

2.3 Operaciones morfológicas

El procesamiento morfológico se aplica para modificar la forma espacial (o estructura) de los componentes en una imagen. Para ello, los operadores morfológicos se definen sobre conjuntos de píxeles. El lenguaje matemático de la teoría de conjuntos ofrece un marco uniforme y potente para el análisis de los problemas en el procesamiento de imágenes [17].

Un caso especial, muy común en muchas aplicaciones, trata sobre las operaciones morfológicas en imágenes binarias. Ahí donde los píxeles de las imágenes valen únicamente *uno* o *cero*[1], sólo pueden utilizarse las funciones lógicas AND (producto), OR (suma) y NOT (negación)[2]. En consecuencia, los operadores morfológicos modifican la forma del objeto agregando o eliminando píxeles según funciones lógicas aplicadas a una pequeña vecindad.

Dentro del procesamiento morfológico existen dos operadores fundamentales llamados *dilatación* y *erosión* con los cuales pueden contruirse otros más complejos. Por su importancia y aplicación directa en el presente proyecto se describen a continuación.

2.3.1 Dilatación y Erosión

La tarea llevada a cabo por estos operadores es similar a la realizada por la convolución, utilizando álgebra Booleana en vez de aritmética. Aprovechando esto, se puede definir una *convolución binaria* [18] que permite explicar su funcionamiento de una manera más intuitiva.

El operador *dilatación* se puede obtener mediante el reemplazo de la multiplicación entre los píxeles de la imagen y una máscara mediante la operación lógica AND, y la sumatoria por la operación OR:

$$g(x,y) = \bigvee_{s=-R}^{R} \bigvee_{t=-R}^{R} m(s,t) \wedge f(x+s, y+t), \tag{2.1}$$

donde $\vee$ y $\wedge$ denotan las operaciones lógicas OR y AND respectivamente. La imagen binaria $f(x,y)$ es convolucionada con una máscara simétrica $m(s,t)$ de $2R+1 \times 2R+1$. Se debe notar que esta formulación no coincide estrictamente con la convolución puesto que la máscara no está espejada respecto a su origen, puede por lo tanto interpretarse como una correlación.

Por su parte, el operador *erosión* se obtiene a partir de la *dilatación* mediante el cambio en la sumatoria por la operación AND:

$$g(x,y) = \bigwedge_{s=-R}^{R} \bigwedge_{t=-R}^{R} m(s,t) \wedge f(x+s, y+t). \tag{2.2}$$

[1]Según el contexto los valores binarios pueden denominarse también como *verdadero/falso*, *blanco/negro*, o cualquier otro par de valores distintos sin perjuicio de su significado.

[2]Estas tres funciones son suficientes puesto que pueden combinarse para formar cualquier otra operación lógica.

Con estas formulaciones se realizan la siguientes tareas: se superpone cada valor de la máscara con el píxel correspondiente de la imagen (centro de $m(s,t)$ ubicado en $f(x,y)$), y se deja *unos* sólo donde máscara e imagen valen *uno*. Luego, según el caso:

- Dilatación: una suma lógica devuelve *cero* sólo si todos los valores del enmascaramiento anterior son *cero*. De esta manera, siempre que exista al menos una coincidencia, la dilatación agregará el píxel (x,y) como objeto en $g(x,y)$ y por tanto el tamaño del objeto crecerá.
- Erosión: un producto lógico devuelve *uno* sólo si todos los valores del enmascaramiento anterior son *uno*. De esta manera, siempre que exista al menos un *cero*, la erosión eliminará el píxel (x,y) como objeto en $g(x,y)$ y por tanto el tamaño del objeto disminuirá.

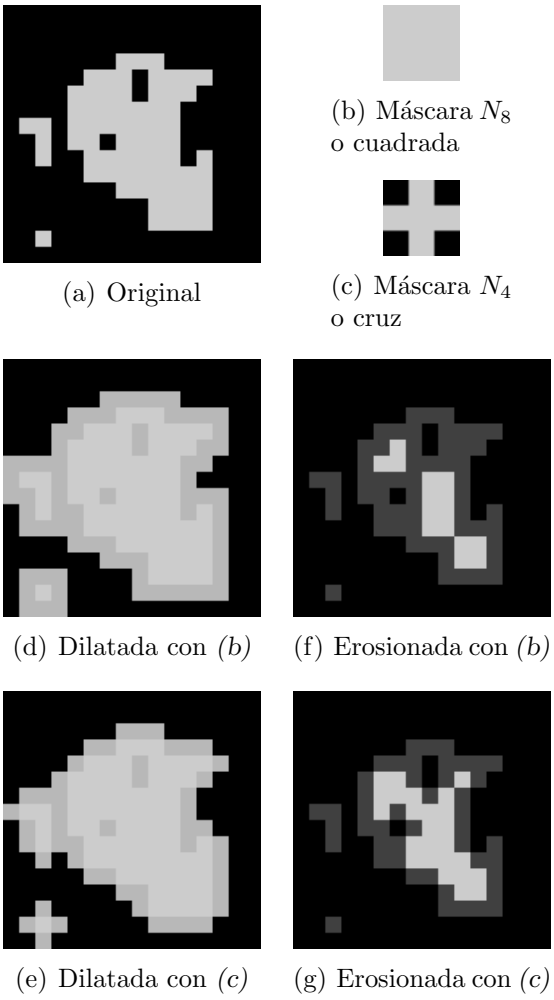
En la Fig. 2.5 se puede ver el resultado de aplicar los operadores morfológicos con dos máscaras distintas.

2.4 Segmentación

La segmentación tiene por objeto subdividir una imagen en sus regiones constitutivas u objetos de interés. Su nivel de detalle y la forma de llevarla a cabo dependen del tipo de problema a resolver, aunque se debe tratar siempre de no superar el nivel de detalle necesario para identificar los elementos.

Los algoritmos de segmentación están basados generalmente en dos propiedades de la intensidad [17]:

- *Discontinuidad:* los cambios abruptos de intensidad se asocian a los límites o bordes de los objetos. Los métodos de detección de puntos, líneas y bordes se desarrollan en base a la convolución de la imagen con máscaras especiales (por ejemplo Prewitt, Sobel y LoG entre otros) y una posterior umbralización. La transformada de Hough es un método alternativo y original para la detección de líneas (extendible a curvas arbitrarias [20]) que también es utilizado como complemento de los métodos anteriores para completar segmentos de líneas que puedan haberse cortado por la presencia de ruido. La detección de líneas y bordes también es abordada desde la teoría Gestalt [21] a través de un punto de vista totalmente distinto donde se asumen principios y leyes de agrupamiento presentes en la percepción visual.



(a) Original

(b) Máscara N_8 o cuadrada

(c) Máscara N_4 o cruz

(d) Dilatada con *(b)*

(f) Erosionada con *(b)*

(e) Dilatada con *(c)*

(g) Erosionada con *(c)*

Figura 2.5: Ejemplos de dilatación y erosión de una imagen *(a)* mediante dos máscaras distintas de 3×3 píxeles. Los colores indican píxel en 1, píxel en 1 (agregado al objeto), píxel en 0 (eliminado del objeto) y píxel en 0.

- *Similitud:* en muchas imágenes los objetos de interés presentan un nivel uniforme de intensidad que es diferente al fondo. La segmentación de regiones similares se realiza de acuerdo a un conjunto de criterios predefinidos. En esta categoría figuran los métodos de umbralización, crecimiento de regiones, separación y unión.

Existen muchos otros algoritmos de segmentación que incluyen estos dos atributos combinados, como watershed para regiones y snakes para siluetas. Por supuesto, están los que intentan responder un problema aun más difícil que es la segmentación de texturas, donde la similitud dentro de una región está dada por la regularidad presente en las variaciones de la intensidad.

En este proyecto la segmentación se realiza con el fin de determinar la silueta de la mano que está realizando un signo. Para esta tarea se ha optado por el método de umbralización cuyo enfoque juega un rol significativo en aplicaciones donde la velocidad es un factor importante. Como complemento se utiliza el crecimiento de regiones con dos propósitos diferentes: aislar la silueta de posibles regiones satélites y el relleno de grietas internas. A continuación se describen estos dos métodos.

2.4.1 Umbralización

La determinación de la pertenencia o no, de un píxel, a una determinada región se basa en el nivel de gris que presenta éste. Para el caso de imágenes a color y multiespectrales el procedimiento se aplica a varias componentes al mismo tiempo.

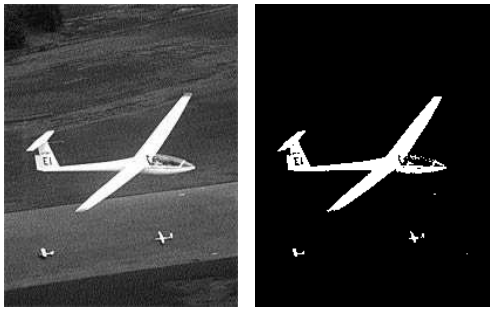
De manera formal, la umbralización puede verse como una operación que involucra una prueba contra un función T de la forma:

$$T = T[x, y, p(x, y), f(x, y)] \tag{2.3}$$

donde $f(x, y)$ es el nivel de gris del punto (x, y) y $p(x, y)$ denota alguna propiedad local en ese punto —por ejemplo, el promedio del nivel de gris de una vecindad centrada en (x, y)—. Una imagen umbralizada $g(x, y)$ puede ser definida como:

$$g(x, y) = \begin{cases} 1 & \text{si } f(x, y) > T \\ 0 & \text{si } f(x, y) \leq T. \end{cases} \tag{2.4}$$

En este caso los píxeles marcados con 1 corresponden a objetos y 0 al fondo. Los valores 0 o 1 son arbitrarios y la binarización puede tomar cualquier otro valor según el caso en particular.



(a) (b)

Figura 2.6: Ejemplo de segmentación por umbralización global. Los objetos claros se separan del fondo mediante la ecuación (2.4), donde $T = 180$. (a) Imagen original, (b) Imagen segmentada.

Si T depende sólo de $f(x, y)$ se la llama umbralización *global*; si en cambio depende tanto de $f(x, y)$ como de $p(x, y)$ recibe la denominación de umbralización *local*. A su vez, si T depende de las coordenadas espaciales x e y, se dice que la umbralización es *dinámica* o *adaptativa*.

La umbralización global es la más simple, y su correcto desempeño depende de cuán bien puede particionarse el histograma de la imagen. La Fig. 2.6 presenta el resultado de la segmentación de las aeronaves y los marcadores de pista mediante este método.

2.4.2 Crecimiento de regiones

Este procedimiento consiste en, a partir de un conjunto de puntos llamados *semillas*, incorporar píxeles contiguos para obtener regiones mayores. Este "crecimiento" se realiza mientras los píxeles adyacentes satisfagan algún criterio de similitud con la semilla (como un rango del nivel de gris o color).

La naturaleza del problema influye en los siguientes aspectos:

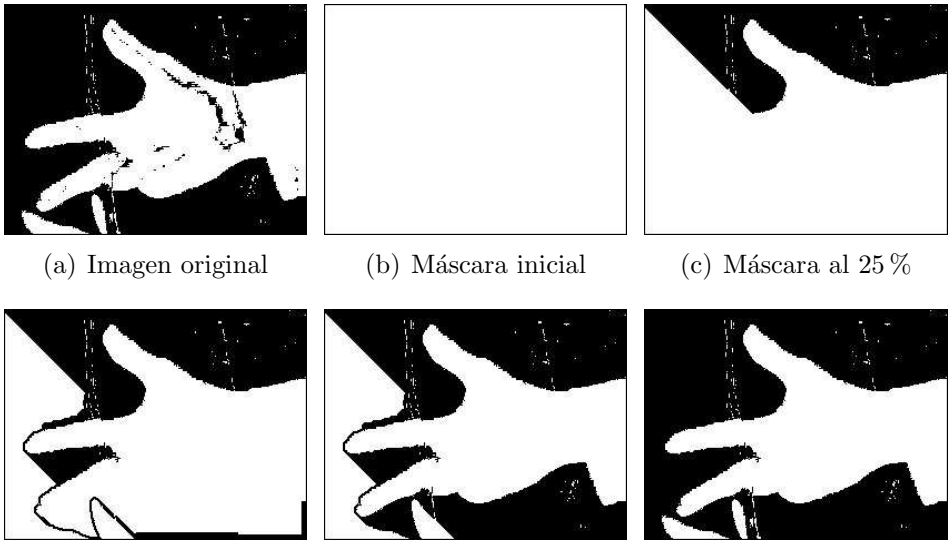
- *Cantidad de semillas:* la información a priori permite definir la utilización de una o varias semillas y la forma de establecer su localización (preestablecida o estimada por algún método).
- *Conectividad:* define cuáles son los píxeles adyacentes a una subregión (generalmente N_8 o N_4, Figs. 2.5(b) y 2.5(c) respectivamente) y determina la dirección del crecimiento.

- *Criterio de similitud:* depende tanto del problema como del tipo de imagen con la cual se trabaja. Las imágenes en color contienen mayor información y por lo tanto alcanzan mejores resultados que sus contrapartes monocromáticas, las cuales son en general más difíciles de procesar correctamente utilizando únicamente el nivel de gris.

  El píxel candidato puede compararse con la semilla (observación local), o bien con el promedio general de la región en crecimiento (observación global).

- *Regla de detención:* esta regla complementa al criterio anterior. Su formulación incrementa el poder del algoritmo de crecimiento, incluyendo conceptos como tamaño y forma de la región. El uso de estos descriptores se basa en un modelo esperado que es asumido. Esta regla no es necesaria cuando el criterio de similitud es suficiente para una buena separación de las características.

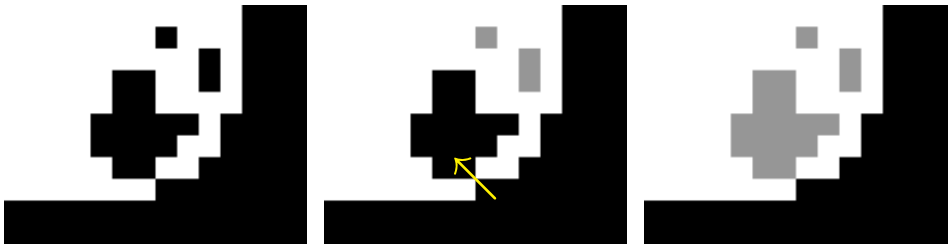
El crecimiento de regiones aplicado a imágenes binarias puede ser utilizado con un propósito distinto a la segmentación. En este trabajo se implementó una técnica llamada *crecimiento inverso* que se utiliza para la eliminación de huecos en regiones internas (operación morfológica). La denominación *inverso* responde a que conceptualmente el crecimiento se realiza en las zonas con ausencia de los objetos de interés. Esta técnica consiste en establecer el borde de la imagen al valor del fondo y realizar un crecimiento de regiones con una sola semilla desde una esquina. Utilizando conectividad N_4 se crece por todos los píxeles iguales al fondo. La preferencia de N_4 se justifica debido a que N_8 rellena más zonas, puesto que conecta píxeles en diagonal y llega a huecos adyacentes como se ve en la Fig. 2.8(b), donde lo indica la flecha. No se utiliza ninguna regla particular de detención; el algoritmo se detiene cuano no encuentran píxeles que segmentar. Cada píxel incorporado (incluyendo al borde) se marca como fondo (cero) en una mascara binaria destino, previamente inicializada con el valor no fondo (uno). A su finalización, la máscara destino contiene las regiones previamente identificadas pero sin huecos o grietas interas (sin aberturas al exterior). En las Figuras 2.7 y 2.8 se ve el detalle del proceso llevado a cabo y el resultado obtenido según el tipo de vecindad respectivamente.



(a) Imagen original (b) Máscara inicial (c) Máscara al 25 %

(d) Máscara al 50 % (e) Máscara al 75 % (f) Máscara final

Figura 2.7: Ejemplo del crecimiento inverso de regiones utilizado para el relleno de huecos internos.



(a) Original (b) Usando N_8 (c) Usando N_4

Figura 2.8: Efecto del tipo de vecindad en el crecimiento de regiones. Los colores indican □ píxel de objeto, ■ píxel de fondo y ■ píxel agregado al objeto.

2.5 Reconocimiento de objetos

El reconocimiento de objetos consiste en poder distinguir, a partir de un conjunto de características o atributos, un objeto de interés respecto de otro. Esto implica dos conceptos muy importantes: extracción de características y clasificación.

El conjunto de características (descriptores) que definen un objeto (también llamado *patrón*) son propiedades que se miden o se generan directamente del propio objeto. Éstos pueden agruparse formando vectores (para valores cuantitativos) y cadenas de letras o árboles (en descripciones estructurales). La naturaleza y cantidad de características que forman un patrón dependen de la técnica que se utilize, la cual a su vez depende del área de aplicación y de la experiencia del usuario en el dominio del problema [22]. Los patrones que comparten propiedades en común pueden organizarse en *clases de patrones*. Si W es el número de clases, las mismas se denotan como $\omega_1, \omega_2, \ldots, \omega_W$. Reconocer un patrón significa asignarlo a una de las clases y los métodos para llevar esta tarea a cabo pueden ser divididos en dos grandes áreas [17]:

a) Decisión teórica: utilizan descriptores que se expresan mediante valores numéricos como longitud, área y dirección, entre otros. Estos enfoques se basan en W funciones de decisión $d_i(\mathbf{x})$ las cuales cumplen con la siguiente propiedad:

$$d_j(\mathbf{x}) > d_i(\mathbf{x}) \;\; i = 1, 2, \ldots, W; i \neq j, \tag{2.5}$$

donde $\mathbf{x} = (x_1, x_2, \ldots, x_n)^T$ representan un vector n-dimensional de cada patrón. Según las características de las funciones discriminantes la condición $>$ puede ser inversa a la formulada. La pertenecia de un patrón desconocido $\mathbf{x}$ a una clase ω_j se determina como aquella función de decisión $d_j(\mathbf{x})$ que obtiene el valor más grande (o pequeño, según el caso) entre todas ellas. Entre los distintos métodos de esta categoría se encuentran: vecino más cercano, correlación, clasificadores estadísticos (aplicables a mediciones e interpretación de fenómenos físicos) y redes neuronales, entre otros.

b) Estructural: consideran los elementos y relaciones estructurales que describan a los objetos de la manera más parecida a la forma real presente en la imagen. Las técnicas empleadas son de las más variadas y su particularidad no permite expresarlas en un marco general. Entre los

métodos se pueden mencionar el emparejamiento de cadenas[3] y el reconocimiento sintáctico de cadenas y árboles, entre otros.

Es así que se puede ver al reconocimiento de objetos como una tarea de *clasificación*. En función del contexto, dicha clasificación puede significar tanto: establecer la existencia de clases o agrupamientos en los datos, como la regla que establece a cual de las clases pertenece una nueva observación. El primer caso toma la forma de *aprendizaje no supervisado* y el segundo de *aprendizaje supervisado* donde un agente externo al sistema es quien define el tipo y el número de clases [23].

2.5.1 Vecino más cercano

Es un método de clasificación supervisado basado en la determinación de una mínima distancia. Dado un objeto, el método supone que los vecinos más cercanos pertenecen a la misma clase dada la proximidad de todos sus atributos. Esta suposición implica que se debe tener cuidado con la elección de los mismos para no permitir que atributos relevantes pierdan importancia frente a otros secundarios. La cantidad de atributos no siempre asegura mejores resultados. El funcionamiento del método se resume en dos fases:

- *Entrenamiento:* consiste en almacenar en una estructura $< \mathbf{x}, \omega >$ los vectores característicos y las etiquetas de las clases para cada uno de los ejemplos de entrenamiento. Dichos vectores (compuestos por los n atributos de cada patrón) forman un espacio n-dimensional que es particionado en regiones de acuerdo a las localizaciones y las etiquetas de cada clase.

- *Clasificación:* la determinación de la clase a la que pertenece un ejemplar $\mathbf{x}_q$ se realiza mediante la búsqueda del vecino que presenta la menor distancia:

$$f(\mathbf{x}_q) = \min_k \{D(\mathbf{x}_q, \mathbf{x}_k)\} =< \mathbf{x}_i, \omega_i > \rightarrow \omega_i, \tag{2.6}$$

  donde k representa cada uno de los vecinos almacenados en el entrenamiento, $D(\cdot)$ es una función distancia (por ejemplo: euclídea, valor absoluto, ponderada), $\mathbf{x}_i$ es el vecino más cercano, ω_i su etiqueta la cual se asigna al ejemplar $\mathbf{x}_q$.

[3]Por "cadena" se hace referencia a "cadenas de caracteres" que es la traducción completa del término Inglés *string*.

CAPÍTULO 3

Desarrollo del método

La palabra *signo* tiene muchas acepciones según las diversas disciplinas que lo traten. En este trabajo cuando decimos *signo* nos estamos refiriendo a la forma (posición, orientación, configuración) que presenta la mano de quien la coloca delante de la cámara, la cual luego de estar definida por un conjunto de características podrá ser identificada respecto de otras y se le podrá asignar un significado específico arbitrario.

El desarrollo del método de reconocimiento de signos manuales provenientes de un flujo de video se realiza en dos etapas. La primer etapa aborda el método de reconocimiento de signos que opera con imágenes estáticas capturadas con una cámara web desarrollado en [24]. Se analizan distintas condiciones de iluminación y de entorno con el objetivo de lograr un método robusto que sea aplicable en situaciones prácticas y sea utilizado por personas sin conocimientos sobre el procesamiento de imágenes. La segunda etapa incorpora dentro de un esquema de procesamiento de video todo el desarrollo anterior, definiendo nuevas etapas a fin de optimizar los recursos y los resultados[1].

[1]Estos resultados fueron presentados en una sesión especial de la ECImag 2009, llevada a cabo en la ciudad de Tandil (Buenos Aires).

3.1 Reconocimiento de un fotograma

Desde la concepción de la idea del proyecto, uno de los puntos de interés ha sido que el reconocimiento de un signo presente en un fotograma[2] permita su utilización fuera del ambiente de laboratorio. Esto implica obtener un reconocimiento exitoso en situaciones que no requieran que los signos sean: realizados sobre un fondo uniforme, de frente y perpendicular a la cámara; con una iluminación predeterminada y orientada para no crear zonas con sombras.

Para esta primera etapa del desarrollo se propuso un método que puede ser dividido en los siguientes bloques fundamentales:

- Adquisición de la imagen.
- Segmentación de la mano.
- Análisis y extracción de características.
- Identificación del signo.

En la Fig. 3.1 puede verse como se relacionan estas etapas.

3.1.1 Adquisición de la imagen

Para la elección del dispositivo de captura, una cámara web, se prioriza que éste sea de uso masivo y de precio accesible. Como requisitos mínimos se establecieron: que opere en resolución de 640×480 píxeles y obtenga 15 fotogramas por segundo (fps). Con estas características se evalúan dos dispositivos de captura: Genius iLook 110 y Genius Slim 1322AF (denominandas iLook y Slim en futuras referencias). La evaluación y selección del modelo se detalla en la sección 5.1.

En las pruebas, las imágenes han sido tomadas a color en espacio de color RGB con una resolución de 640×480 píxeles y se utiliza la mano desnuda, con mangas largas. Una condición adicional impuesta es que tanto las mangas como el fondo tengan un color diferente al de la mano[3].

[2]Imagen tomada con un cámara.

[3]Esta condición puede relajarse mediante una *región de interés*, ver sección 5.6.5.

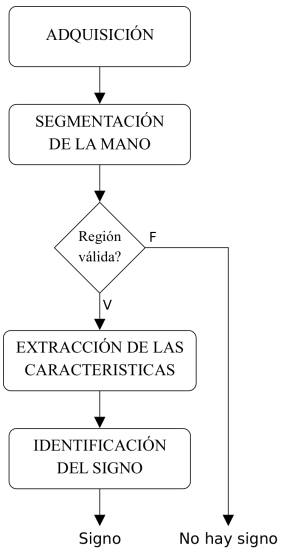


Figura 3.1: Diagrama de flujo del algoritmo propuesto.

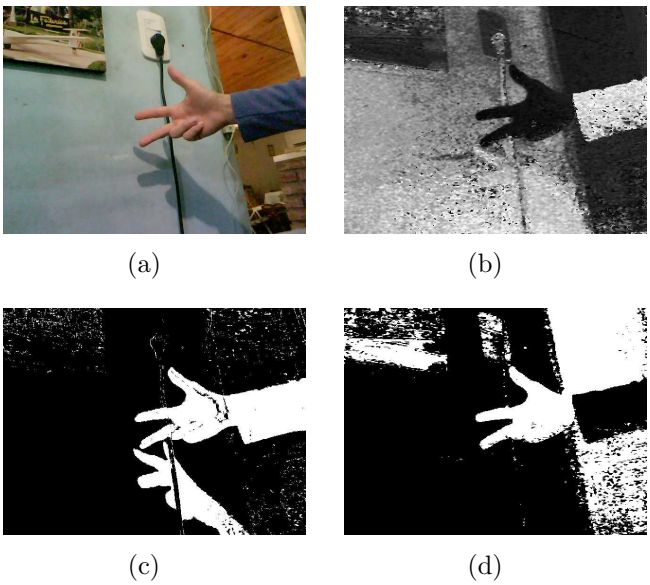
3.1.2 Segmentación de la mano

El primer paso consiste en reconocer a grandes rasgos y a partir de un fondo de referencia obtenido previamente sin la mano, cuál es la perturbación introducida en la escena por la presencia de la misma. Para esto, en cada canal del RGB se calcula el valor absoluto de la diferencia entre ambas imágenes y se umbraliza el resultado, obteniendo así una máscara de cambio para cada canal. La zona de cambio total en la escena se determina como la unión de todas estas máscaras, obteniendo así la máscara de color. Este resultado supone la detección de la mano, el brazo y, en condiciones de luz dirigida, la sombra de éstos (Fig. 3.2(c)).

El siguiente paso consiste en separar la mano del brazo y la sombra. La estrategia para lograrlo es trabajar sobre el canal de tono (Fig. 3.2(b)), separando las regiones cuyo tono se aproxime a uno de referencia (Fig. 3.2(d)) definido previamente[4]. Este método requiere que el brazo no esté desnudo, o si se utiliza un guante, que el mismo sea de color uniforme y diferente al resto del atuendo. Como resultado se obtiene una máscara de diferencia de tono (particularidades de la operación se presentan al final de esta sección).

El paso siguiente consiste en aplicar el operador de dilatación a ambas máscaras (color y tono) para rellenar posibles grietas en el contorno de la

[4]En la sección 4.2.1 se expone una forma práctica para determinar los parámetros.



(a) (b)

(c) (d)

Figura 3.2: (a) Imagen original, (b) Canal de tono, (c) Máscara diferencia de color, (d) Máscara diferencia de tono.

mano (Fig. 3.3(b) y 3.3(c)). Luego se calcula la intersección (un AND lógico) entre ellas y se realiza un crecimiento de regiones a partir de un punto fuera de la mano para obtener la máscara inversa de la misma sin agujeros (Fig. 3.3(d)). En las Figuras 3.3(e) y 3.3(f) se puede ver un ejemplo del error que se produce en la obtención de la máscara al obviar el paso de dilatación, sobre la porción de la imagen que contiene el dedo meñique. Como resultado se tiene una máscara del tamaño de la imagen original, con la silueta de la mano y en algunas situaciones otras regiones aisladas producidas por ruido o sombras (Fig. 3.3(d)).

En algunos casos, debido a las condiciones de iluminación adversas, la umbralización del canal de tono conserva parte del contorno del brazo o de la sombra, conectadas a la mano por una sección apreciablemente angosta (Fig. 3.4(a)). Para solucionar este inconveniente, la estrategia es erosionar la imagen para eliminar el camino que las une, y luego dilatar el resultado para recuperar un tamaño similar al original. De esta forma, al erosionar con una máscara suficientemente grande, se elimina la sección angosta que une ambas regiones (Fig. 3.4(b)). Luego, al dilatar, éstas vuelven a crecer aproximadamente a su superficie original, pero el camino de unión ya no existe (Fig. 3.4(c)).

Asumiendo que la región de la mano es la más extensa entre las presentes,

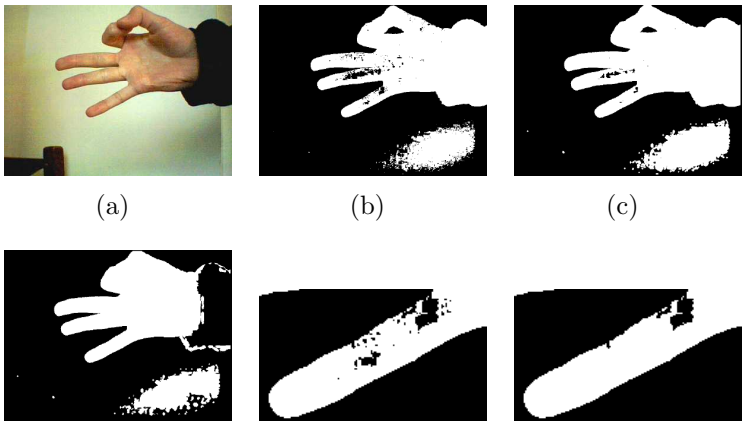


Figura 3.3: Eliminación de grietas internas. (a) Imagen original, (b) Máscara de color, (c) Máscara de color dilatada, (d) Resultado del producto entre máscaras de tono y color, seguido de crecimiento de regiones, (e) Detalle de las imperfecciones presentes en la máscara de color, (f) Problema al aplicar crecimiento de regiones en la máscara de color sin dilatar.

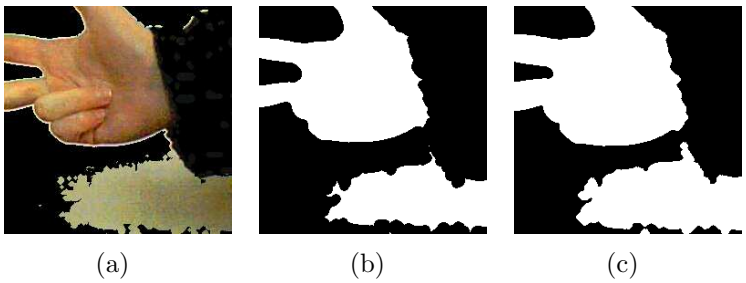
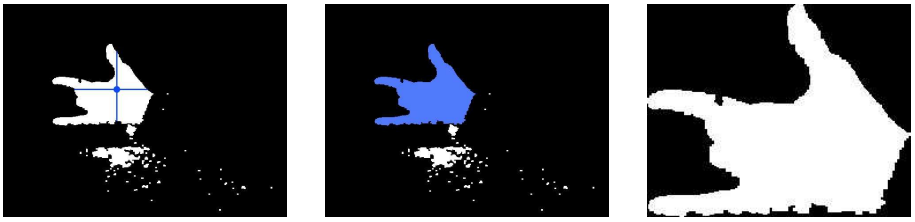


Figura 3.4: Pasos para la eliminación de caminos que unen la región de la mano con otras manchas. (a) Máscara inicial: silueta de la mano conectada a una región indeseada, (b) Máscara erosionada: las regiones se separan, (c) Máscara dilatada.



(a) (b) (c)

Figura 3.5: Determinación de la región que es mano. (a) Intersección de los segmentos más largos, (b) Región seleccionada como mano luego del crecimiento de regiones, (c) Máscara final de la silueta de la mano.

se busca un punto inicial dentro de la misma y se aplica un crecimiento de regiones que obtenga la silueta aislada definitiva. Para encontrar dicho punto, se recorren líneas horizontales y verticales equiespaciadas buscando los segmentos blanco más largos, y se toma como punto central su intersección (Fig. 3.5(a)). En el caso que los segmentos no se crucen en un punto común, se infiere que no existe una única región que sea predominante en ancho y alto, o bien, que no hay regiones en la máscara actual. En estos casos la segmentación concluye informando que no hay signo en la imagen. Si se encuentra el punto, se puede obtener el conjunto de píxeles correspondientes a la región de la mano (Fig. 3.5(b)). Finalmente se pueden definir los límites de la caja frontera[5] como las coordenadas x e y extremas dentro de dicha región. El rectángulo de inclusión delimitado (Fig. 3.5(c)) constituye la zona de análisis en la etapa posterior de extracción de características. La Fig. 3.6 resume el proceso completo de extracción de la silueta.

Particularidades para la máscara diferencia de tonos

Es conocido que el canal de Tono (Hue) del modelo de color HSV es un parámetro circular (o periódico), es decir, los extremos del rango $[0, 360)$ son valores contiguos; por lo tanto merece una atención especial. Para conocer cuáles son los valores cercanos a un determinado valor α, se debe tener en cuenta qué sucede cuando los valores $\theta = \alpha \pm \Delta$ caen fuera del intervalo contemplado por el modelo. Por ejemplo, para el caso $\alpha = 8$ y $\Delta = 25$ el rango calculado $\theta \in (-17, 33)$ debe interpretarse como $\theta \in (343, 360] \cup [0, 33)$.

Una forma alternativa para tratar los distintos posibles desbordes de rango y sus interpretaciones, es efectuar una *rotación del tono* antes de analizar

[5]Denominación dada en el texto a la caja de mínima inclusión, en Inglés *bounding box*.

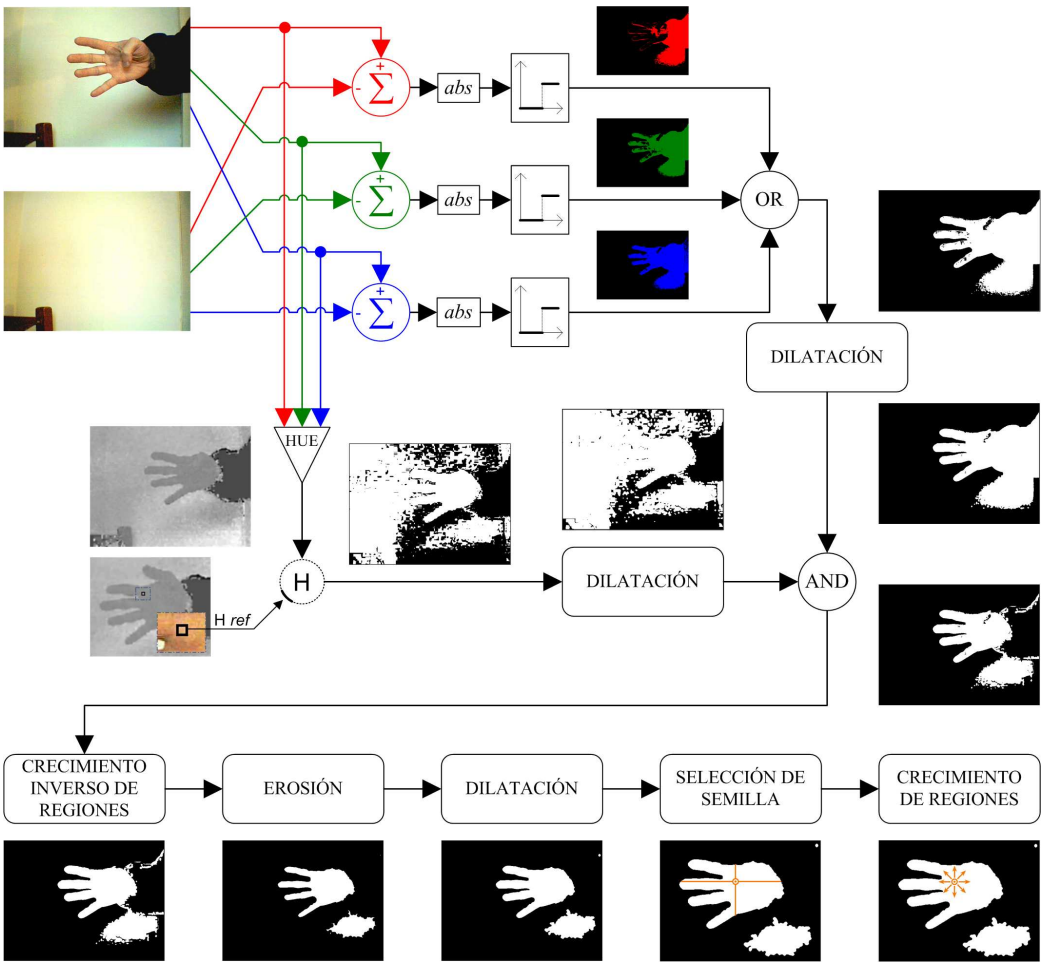


Figura 3.6: Segmentación de la silueta de la mano. A cada canal RGB de la imagen adquirida en presencia de la mano se le sustrae el canal correspondiente de la imagen de fondo, y sobre el valor absoluto de la diferencia se aplica un operador de umbral. Los resultados se combinan mediante una suma lógica para obtener la *máscara de color*. Por otra parte, se aplica un operador de umbral invertido sobre el canal de tono para obtener la *máscara de tono*. Ambas máscaras se dilatan y se combinan mediante un producto lógico, para luego realizar las operaciones finales que obtienen la región de la mano.

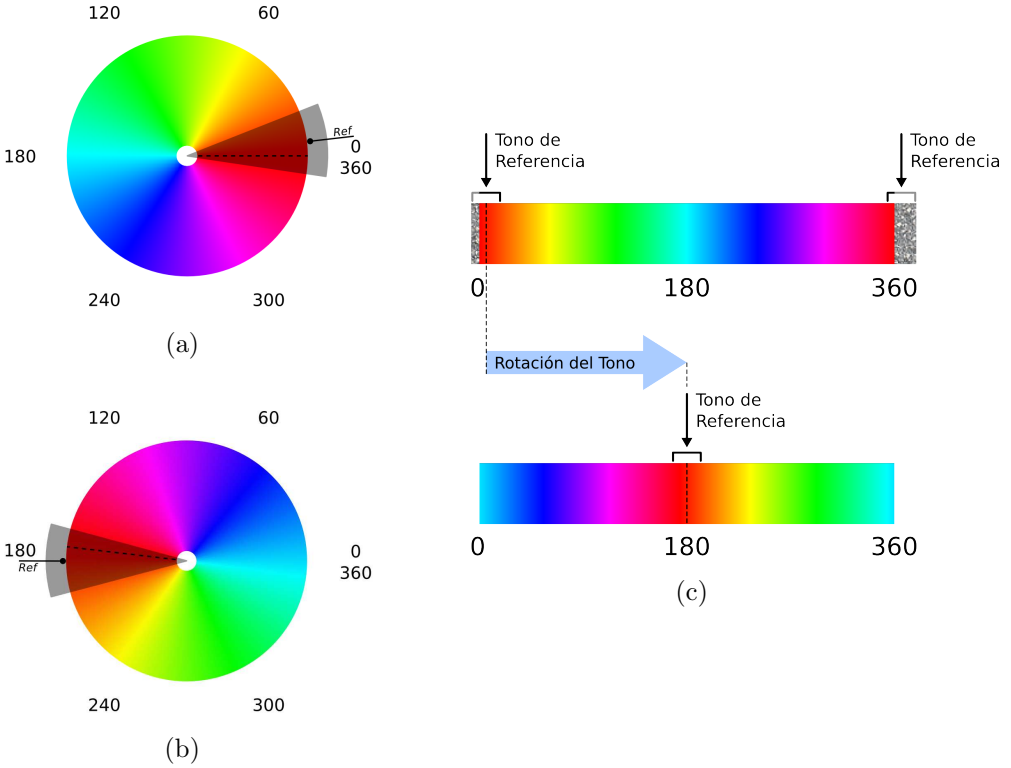
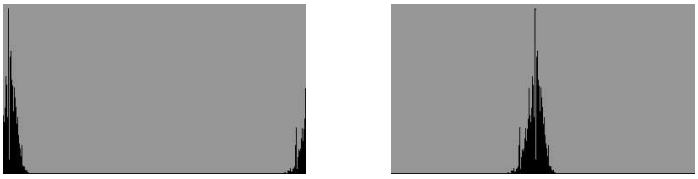


Figura 3.7: Rotación del histograma del canal de tono. (a) Canal de tono original. (b) Canal de tono rotado donde el tono de referencia se mueve a 180°. (c) Vista de la rotación aplicada al patrón cromático.

las diferencias respecto del tono de referencia. La rotación del tono consiste simplemente en desplazar los valores del tono de una imagen tal que el tono de referencia se ubique en 180° . El proceso que genera la máscara diferencia de tono binariza los píxeles manteniéndo sólo aquellos, cuyo tono rotado, esté dentro del rango $180 \pm umbralTono$.

En la Fig. 3.7 se representa el efecto de la rotación del tono aplicada al plano H y la correspondiente representación lineal del mismo. En ambos casos se denotan el tono de referencia y el umbral de tono utilizados como parámetros en este ejemplo. Un caso real, donde se aplica la rotación, se expone en la Fig. 3.8, donde puede apreciarse el histograma resultante antes y después de la rotación.



(a) (b)

Figura 3.8: Ejemplo de rotación del canal de tono en igual medida que el ejemplo de la Fig. 3.7. (a) Histograma original, (b) Histograma rotado.

3.1.3 Análisis y extracción de características

Las señas se pueden identificar a partir de un grupo de características, las cuales se calculan en base a la máscara de salida obtenida en la etapa previa. Entre las características posibles se consideraron las siguientes:

- Número de dedos ⋆
- Orientación de los dedos
- Relación Ancho / Alto ⋆
- Relación área Mano / área de la caja frontera
- Relación área Izquierda / área Derecha
- Relación área Superior / área Inferior
- Coordenadas del centroide relativas a la caja frontera ⋆
- Distancia radial del centroide relativa a la caja frontera

De todas las propuestas sólo se seleccionan tres de ellas ($\star$), puesto que un objetivo importante es trabajar con la menor cantidad de características posibles y que al mismo tiempo aporten la mayor cantidad de información.

Número de dedos

El significado que representa una seña está generalmente asociado a la cantidad de dedos extendidos que ésta presenta. Por lo tanto, se puede considerar una característica fundamental que no puede ser omitida en ningún proceso de identificación. Se propuso un algoritmo para la determinación del número de dedos separados. Los pasos a seguir son:

1. Calcular los parámetros, medidos en píxeles, que reflejan las dimensiones aproximadas de un dedo. Se calculan en función de las dimensiones D_x y D_y de la caja frontera. Éstos son: longitud mínima permitida para un dedo ($l_{\text{mín}}$), ancho mínimo y máximo que, se estima, puede tener un dedo ($w_{\text{mín}}$ y $w_{\text{máx}}$ respectivamente), distancias mínima y máxima de separación entre falanges de dedos ($d_{\text{mín}}$ y $d_{\text{máx}}$, utilizados en el paso 5) y distancia máxima entre centros de segmentos consecutivos (d_{cen}).

2. Recorrer 50 líneas horizontales y 50 líneas verticales equiespaciadas en la imagen, buscando segmentos cuyas dimensiones se encuentren dentro del intervalo definido por $w_{\text{mín}}$ y $w_{\text{máx}}$ (segmentos finos en Fig. 3.9(a)).

3. Comprobar si existe un conjunto de segmentos válidos en las líneas consecutivas de escaneo, cuyos puntos centrales se hallen a una distancia menor que d_{cen}. Si es así, se construye un vector entre los puntos medios del primer y del último segmento de este conjunto. Si la longitud del mismo es mayor o igual a $l_{\text{mín}}$, se lo incluye en el conjunto de potenciales dedos (segmentos gruesos en Fig. 3.9(a)).

4. Los barridos, horizontal y vertical, pueden detectar ambos un mismo dedo o falange cuando su orientación es cercana a $\pm 45°$. Por esta razón, se debe eliminar del conjunto de dedos potenciales aquellos elementos doblemente identificados (Fig. 3.9(c)). Para ésto, se buscan vectores con coordenadas similares y si el vector promedio de éstos se encuentra completamente contenido en la máscara de la mano, se elimina al menor de ellos. Esta última verificación se utiliza para evitar que dos dedos paralelos muy próximos se interpreten como un único dedo.

5. Buscar vectores que puedan pertenecer a diferentes falanges de un mismo dedo y que hayan sido detectados por separado (Fig. 3.9(d)). Por ejemplo, el caso de un dedo no extendido completamente, cuya primera falange fuera detectada por el barrido en un sentido y la segunda por el otro. Para ésto, se buscan pares de vectores cuyos extremos sean próximos (a una distancia menor a $d_{\text{mín}}$) y tales que los otros dos extremos estén suficientemente separados (por lo menos a una distancia $d_{\text{máx}}$). Por cada par encontrado se calculan sus direcciones, y si resultan ser aproximadamente colineales (sus direcciones difieren en menos de 30°), se reemplaza el par por un único vector cuyas coordenadas surgen de los extremos más distantes del par (Fig. 3.9(e)).

En la Fig. 3.9 se aprecian las etapas del algoritmo.

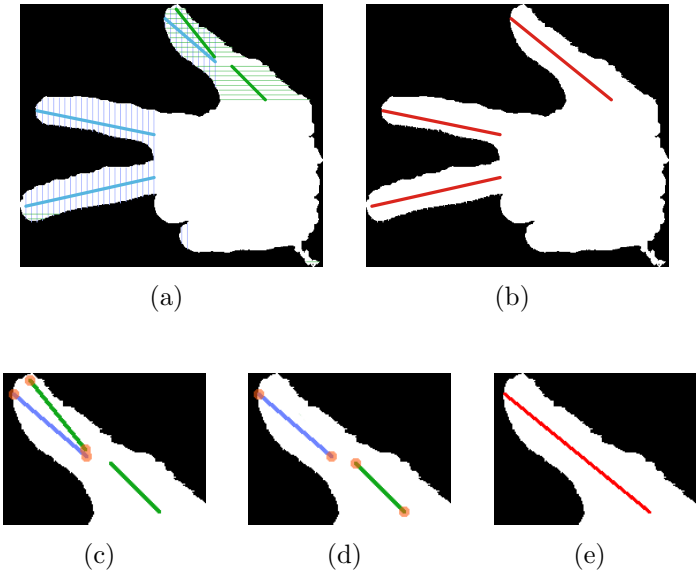
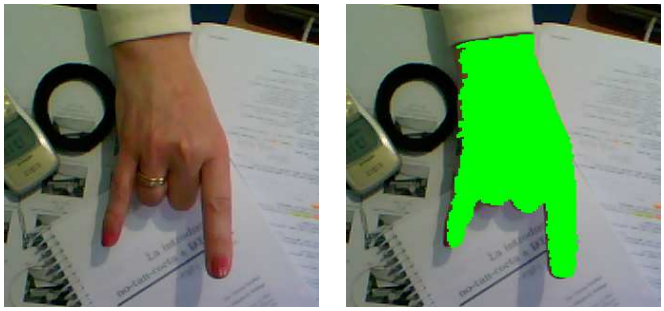


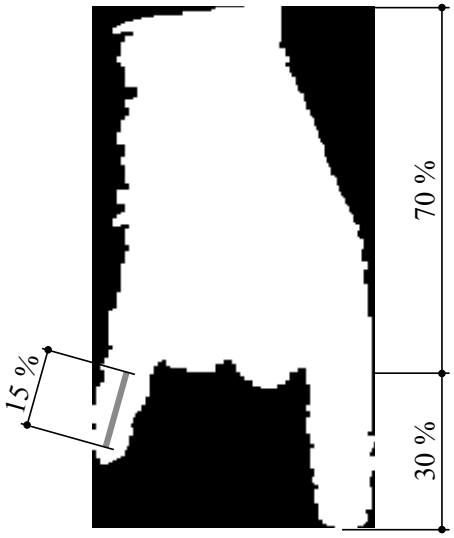
Figura 3.9: (a) Dedos potenciales detectados por barridos horizontal y vertical, (b) Resultado final, (c) Simplificación 1: dedos doblemente identificados, (d) Simplificación 2: segmentos de un mismo dedo, (e) Resultado de la simplificación de un dedo.

Considerando que la longitud de un dedo, en la segmentación, representa aproximadamente el 30 % de la longitud de la mano (Fig. 3.10(c)), y que el dedo podría no presentarse completamente extendido, se opta por utilizar el 15 % de la dimensión máxima de la caja frontera como valor de $l_{\text{mín}}$. Para el caso particular donde una de las dimensiones de la caja frontera sea mayor al doble de la otra (resultando en una relación Ancho/Alto extrema que viciaría cualquier parámetro proporcional), la dimensión máxima a considerar para el cálculo de $l_{\text{mín}}$ es el doble de la mínima, como se establece en la ecuación (3.1).

$$l_{\text{mín}} = \begin{cases} 0{,}15\,\text{máx}(D_x, D_y), & \text{si } D_x < 2D_y \wedge D_y < 2D_x \\ 0{,}15\,\text{mín}(2D_x, 2D_y), & \text{en otro caso.} \end{cases} \tag{3.1}$$



(a) (b)



(c)

Figura 3.10: Estimación de la longitud mínima de un dedo.

Con este valor se calculan, proporcionalmente, todos los demás valores según las siguientes ecuaciones:

$$w_{\text{mín}} = 0{,}2 l_{\text{mín}} \tag{3.2}$$
$$w_{\text{máx}} = 1{,}2 l_{\text{mín}} \tag{3.3}$$
$$d_{\text{cen}} = 0{,}3 l_{\text{mín}} \tag{3.4}$$
$$d_{\text{mín}} = l_{\text{mín}} \tag{3.5}$$
$$d_{\text{máx}} = 2 l_{\text{mín}}. \tag{3.6}$$

Para $w_{\text{máx}}$ se considera una tolerancia mayor, ya que los dedos a $\pm 45°$ representan segmentos más anchos en los barridos horizontales y verticales.

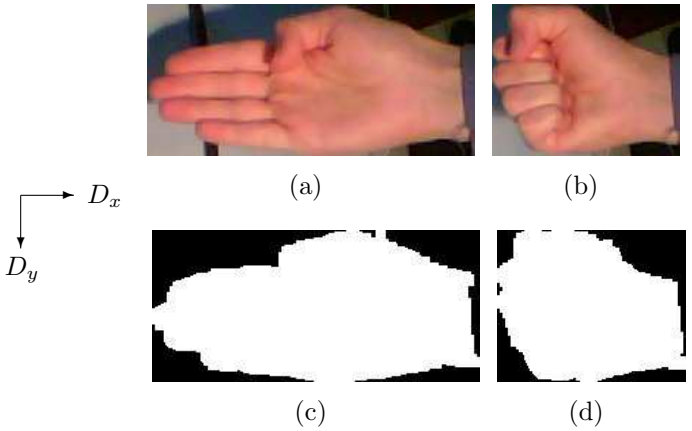


Figura 3.11: Imagen original y silueta detectada de señas con igual número de dedos y ubicación relativa del centroide. En (a) y (c) la relación ancho/alto es 2,17; mientras que en (b) y (d) es 1,24.

Relación Ancho / Alto

Esta relación permite distinguir manos extendidas o apuntando en alguna dirección (Fig. 3.11(a)) frente a manos cerradas (Fig. 3.11(b)). Resulta útil, principalmente en señas, donde no se detectan dedos. Su cálculo es directo mediante el cociente de las dimensiones de la caja de mínima inclusión:

$$R = \frac{D_x}{D_y} \tag{3.7}$$

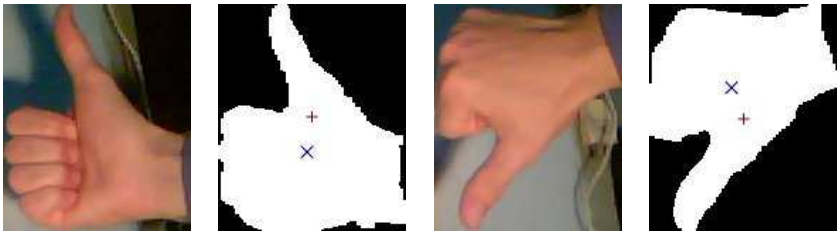
Coordenadas del centroide relativas a la caja frontera

El centroide ubica el centro de masa, dando así una idea de como se distribuye la silueta dentro de su caja frontera. Esta característica se calcula como:

$$C_i^* = \frac{2C_i}{D_i} - 1, \tag{3.8}$$

donde $i \in \{x, y\}$ es la dimensión en estudio (x: horizontal, y: vertical) sobre la cual se obtienen el tamaño de la caja frontera D_i y la coordenada C_i del centroide. Esta última se calcula como:

$$C_i = \frac{\sum_{x=1}^{D_x} \sum_{y=1}^{D_y} i \; m_{xy}}{\sum_{x=1}^{D_x} \sum_{y=1}^{D_y} m_{xy}}, \tag{3.9}$$



(a) (b) (c) (d)

Figura 3.12: Imagen original y silueta detectada con centroide. × centroide y + centro.

donde m_{xy} es la máscara binaria obtenida en la segmentación.

La medida $\left[C_x^*, C_y^*\right]$ obtenida permite diferenciar señas similares con distinta orientación (Fig. 3.12).

3.1.4 Identificación del signo

Las características extraídas conforman lo que denominamos patrón del signo. Éste se compara con un conjunto de patrones que representan las clases del signo a reconocer. Al encontrar el patrón más "parecido" en el conjunto, se le asigna la etiqueta de la clase que identifica a dicho patrón. Esto implica que el usuario deberá entrenar previamente al sistema para que considere las características de cada una de las clases.

El proceso de reconocimiento consiste en seleccionar, del conjunto completo de patrones, aquel cuyas características sean suficientemente similares a las del signo a reconocer. Para ésto se utiliza un conjunto de umbrales configurables que definen las tolerancias de cada característica medida respecto a los valores obtenidos en el entrenamiento. Si más de un signo del conjunto de patrones es seleccionado, se determina el que mejor representa a la imagen analizada como aquel que minimiza un promedio ponderado de las diferencias de las características respecto a los valores de referencia.

Antes de exponer en detalle como se plantea la operación del clasificador, corresponde mencionar las observaciones realizadas. El criterio de similitud entre signos responde al cumplimiendo de similitud por parte de todas las características consideradas. El número de dedos es la más simple puesto que se busca la coincidencia exacta (tolerancia cero). Mientras que, estimar el parecido entre dos relaciones de aspecto requiere un análisis sobre cómo los cambios en las dimensiones originales de una región influyen sobre el coe-

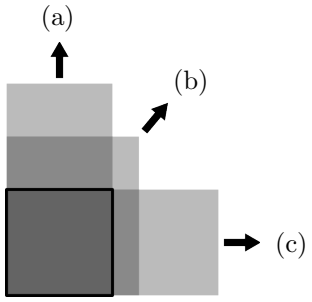


Figura 3.13: Influencia en la relación ancho/alto (R) de las variaciones en una región de referencia. (a) Aumento en altura reduce R, (b) Aumento en ambas direcciones se reduce a uno de los casos a o c, (c) Aumento en el ancho aumenta R.

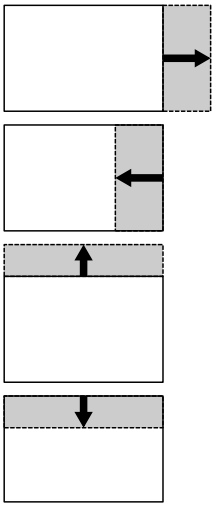
ficiente ancho/alto (Fig. 3.13). Como un aumento en el ancho puede verse como una disminución en el alto, las posibles modificaciones se reducen a dos casos especiales: variación en alto y variación en ancho. Sin embargo, la incidencia de cada variación no es la misma según el sentido y la dimensión de cual se trate. Para obtener los efectos reales de tales varicaciones se ejemplifican en la Fig. 3.14 los posibles cambios (en un 30 %) para una región arbitraria de relación ancho/alto R. De esta se puede concluir que una reducción de un 30 % en cualquiera de sus direcciones se encuentra acotada por los coeficientes 0,70 y 1,43, correspondientes a las ecuaciones (3.11) y (3.13). Así quedan definidas las tolerancias entre las cuales se ha de decidir sobre el parecido de dos relaciones de aspecto.

Por su parte, las coordenadas del centroide relativas a la caja frontera se ubican sobre una región delimitada entre $(-1, 1)$ en x e y, teniendo como origen el centro de la imagen. Dentro de ese plano, dos puntos p y q pueden considerarse parecidos si no distan en más de un determinado valor λ. Para ello se adoptó una medida de distancia definida por la siguiente ecuación:

$$D(p, q) = \text{máx}(|p_x - q_x|, |p_y - q_y|) < \lambda \tag{3.14}$$

En la Fig. 3.15 se ejemplifica el plano completo junto a tres centroides con sus correspondiente regiones, donde la distancia entre dos de ellos, es decir su tolerancia, es menor a 0,20.

Un último aspecto para comentar es la ponderación utilizada para la decisión final. Dado que la escala de las características no es la misma en todas, se hace necesario ponderar previamente cada una de las diferencias (entre el valor actual y el de referencia) para elevar la incidencia de las características con magnitudes pequeñas. Considerando al número de dedos como excluyen-



$$\frac{ancho * (1 + 0{,}3)}{alto} = 1{,}30R \tag{3.10}$$

$$\frac{ancho * (1 - 0{,}3)}{alto} = 0{,}70R \tag{3.11}$$

$$\frac{ancho}{alto * (1 + 0{,}3)} = 0{,}77R \tag{3.12}$$

$$\frac{ancho}{alto * (1 - 0{,}3)} = 1{,}43R \tag{3.13}$$

Figura 3.14: Ejemplo de la variación en la relación ancho/alto a través de aumento y reducción de un 30 % en sus dimensiones.

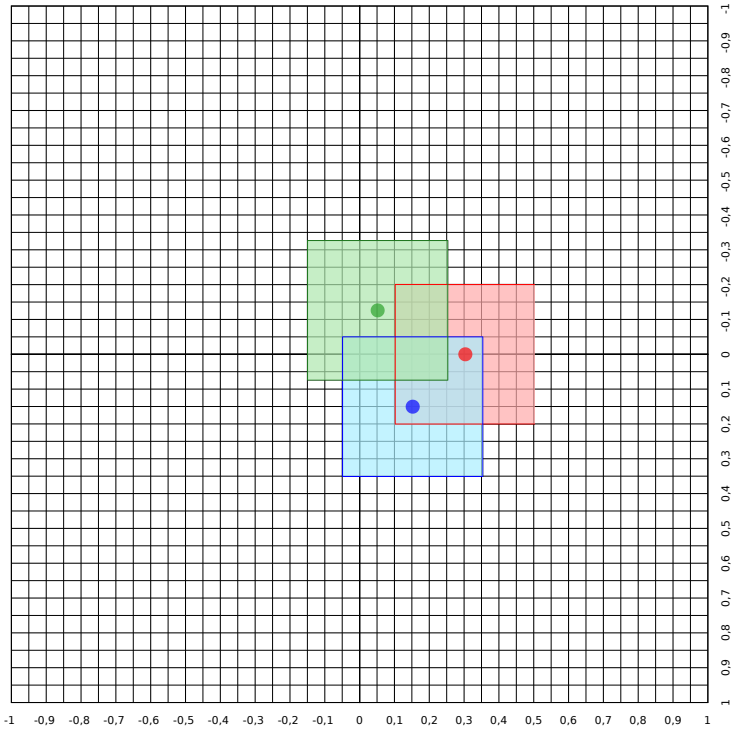


Figura 3.15: Similitud entre centroides. Los puntos son los centroides y las zonas sombreadas representan una tolerancia de 0,20 de sus respectivos centroides.

te, sólo se requieren normalizar las diferencias de la relación de aspecto y las coordenadas del centroide.

Teniendo en cuenta las concideraciones planteadas, se detalla a continuación el proceso llevado a cabo por el clasificador. La primer etapa es la búsqueda de un patrón del conjunto que se parezca al signo actual a identificar. Para ésto las características son evaluadadas en el siguiente orden específico (en caso de no cumplirse una condición se pasa al siguiente patrón del conjunto):

1. Se verifica que el número de dedos sea igual en ambos patrones.

$$Dedos_{actual} = Dedos_{referencia} \tag{3.15}$$

2. Se comprueba que el parecido entre las relaciones ancho/alto esté dentro de una tolerancia $t_{aspec} \in [0, 1)$:

$$(1 - t_{aspec}) \leq \frac{R_{referencia}}{R_{actual}} \leq \frac{1}{(1 - t_{aspec})} \tag{3.16}$$

3. Se comprueba la cercanía entre los respectivos centroides utilizando como tolerancia una distancia $t_{cen} \geq 0$:

$$D(p, q) = \text{máx}(|p_x - q_x|, |p_y - q_y|) \leq t_{cen} \tag{3.17}$$

Un segundo paso consiste en determinar cual de los patrones, que cumplieron todos los criterios de similitud, es el más parecido. El ganador cumple con (3.15), (3.16), (3.17), y se determina según la siguiente ecuación:

$$\min(\alpha \frac{R_{referencia}}{R_{actual}} + \beta D(p, q)) \tag{3.18}$$

El conjunto de pesos y tolerancias pueden ser modificados por el usuario, aunque se encontraron experimentalmente valores que producen una alta efectividad en una amplia variedad de situaciones (ver sección 5.4.1).

Figura 3.16: Flujo de video.

3.2 Reconocimiento en el flujo de video

Un flujo de video es una sucesión de fotogramas provenientes de una fuente que puede ser una cámara o archivo de video (Fig. 3.16). Para su tratamiento se deben incorporar las operaciones de reconocimiento propuestas a la secuencia, y utilizando información de fotogramas pasados, se puede disminuir la carga de procesamiento y sacar provecho de la redundancia temporal.

En esta segunda parte del desarrollo se retoma el proceso de reconocimiento presentado en 3.1 y se elabora un diseño que incorpora nuevas etapas de manera de optimizar los recursos y los resultados. En la Fig. 3.17 se presenta el diagrama de flujo de las etapas involucrados en el procesamiento del video.

3.2.1 Adquisición del fotograma

La adquisición de los datos se realiza a través de un ciclo cerrado que, a intervalos de tiempo regulares[6], extrae un fotograma de la fuente. Ésta es la razón por la cual en la Fig. 3.17 se observan las etapas formando un bucle.

La cámara web seleccionada en este proyecto es la Genius Slim 1322AF que, para una resolución de 640×480 píxeles, opera a 30 fotogramas por segundo (fps). Éste es el valor máximo teórico declarado por la fábrica. En la práctica se experimentó un rendimiento inferior cercano a los 25 fps, los cuales decaen aún más cuando es menor la iluminación. Este efecto se debe a los procesos de corrección de la imagen que se aplican dentro de la cámara y no pueden ser controlados desde el exterior.

[6]La regulariadad se ve afectada según la carga de procesamiento.

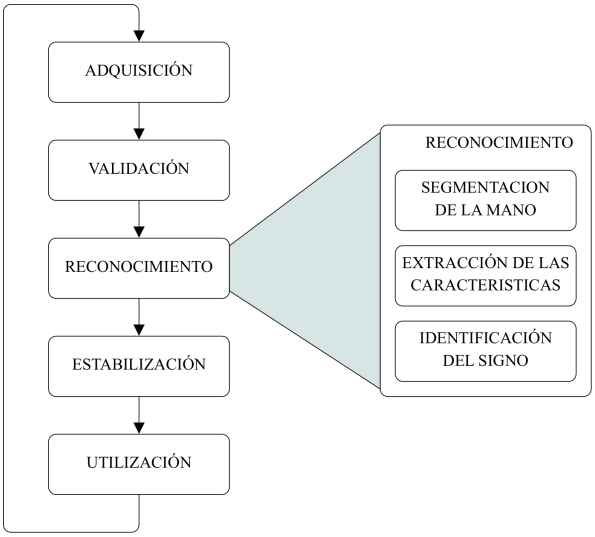


Figura 3.17: Diagrama de flujo del tratamiento del video. Se incorporan las etapas de *Validación*, *Estabilización* y *Utilización*.

3.2.2 Validación del fotograma

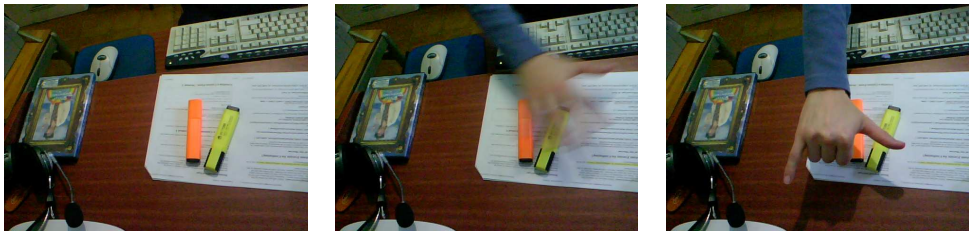
El proceso de reconocimiento de un fotograma insume tiempo, y generalmente es mayor al tiempo en que se mantiene un fotograma hasta que la cámara tiene nueva información disponible. Por lo tanto, con el hardware utilizado, es imposible procesar cada uno de los fotogramas de la secuencia. Aunque fuese posible, ésto sería desperdiciar capacidad de cómputo puesto que existe redundancia temporal, es decir, la probabilidad de un cambio significativo entre dos fotogramas consecutivos es muy baja para el tipo de aplicación que se aborda.

Optimizar el tiempo de procesamiento requiere que sólo se analicen los fotogramas con información importante. Se distinguen tres casos especiales (representados en la Fig. 3.18):

a) Donde se conoce a priori que no hay objetos, entonces se evita realizar el reconocimiento.

b) Donde existen imágenes en transición y las siluetas aparecen borrosas (movimientos veloces). En este caso el reconocimiento puede conducir a resultados erróneos y por tanto es mejor evitarlo.

c) Donde se puede determinar que el fotograma es de alguna manera estable, es decir, que se mantiene relativamente constante entre fotogramas



(a) No hay objetos (b) Imagen borrosa (c) Imagen estable

Figura 3.18: Tipos de imágenes a validar. (a) No hay objetos, se visualiza el fondo configurado, (b) Imagen borrosa, es imposible de reconocer el signo realizado, (c) Imagen estable.

consecutivos, y la imagen contiene una figura con un mínimo de borrosidad. Ésta es la condición adecuada para ejecutar el reconocimiento del signo.

Esta clasificación se puede llevar a cabo dentro del esquema de procesamiento mediante una etapa *Validación del fotograma*, que consta de los siguientes pasos:

1. Se compara el fotograma actual f_{actual} con el fondo de referencia, ambas imágenes en escala de grises. Se calcula la diferencia absoluta que luego se binariza con un umbral fijo ($uBinario$). Se cuenta la cantidad de píxeles distintos de cero, y si éstos superan un umbral de fondo ($uFondo$), se considera que hay una perturbación suficiente en la imagen para seguir con el paso 2. En caso contrario, estamos frente al caso *a)* y se informa que no hay nada para reconocer.

2. Si se detecta un cambio en el fondo, luego de haber estado en reposo, se supone que es parte de un movimiento de aparición (ya que una seña no se presenta instantáneamente en la escena). Por lo tanto, los primeros N fotogramas son omitidos para generar una pequeña espera antes de pasar al paso 3. El último de ellos se almacena en escala de grises como $f_{anterior}$.

3. Se comprueba que los fotogramas consecutivos f_{actual} y $f_{anterior}$ son parecidos. Se calcula su diferencia, se umbraliza y se cuentan los píxeles de la misma manera que en el paso 1. Si la cantidad obtenida no supera el umbral $uCvos$, los fotogramas son muy parecidos y por lo tanto es *válido* iniciar el reconocimiento. El caso contrario sucede cuando el movimiento de los objetos en la escena no se detiene. Finalmente,

para permitir una nueva comparación con el fotograma siguiente, se almacena f_{actual} como $f_{anterior}$ (en escala de grises).

Como se puede apreciar, en cada paso existe una condición que determina si se avanza o no al paso siguiente. Este esquema de decisiones en cascada se representa en la Fig. 3.19.

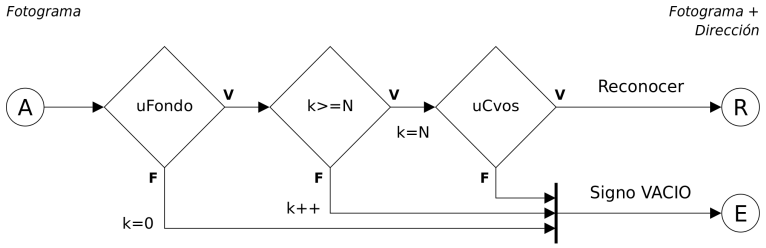


Figura 3.19: Detalle de la etapa validación del fotograma. Ingresa un fotograma desde la etapa de *Adquisición* (A) y según la decisión tomada se envía para su *Reconocimiento* (R) o se informa a la etapa de *Estabilización* (E) que no hay signo.

3.2.3 Reconocimiento del signo

Se observó que las etapas *Segmentación de la mano*, *Análisis y extracción de características* e *Identificación del signo* realizan las tareas específicas que permiten descubrir la información contenida en la imagen, y por tanto pueden ser agrupadas dentro de una tarea general denominada *Reconocimiento del signo*. Ésto posibilita la aplicación del modelo orientado a objetos al momento de desarrollar un software.

A diferencia de las demás etapas, ésta sólo se ejecuta si la etapa anterior determina que el fotograma actual es válido. En la Fig. 3.20 se pueden ver dos caminos alternativos desde la etapa de *Validación*.

3.2.4 Estabilización del signo

El movimiento natural de la mano o los momentos en que cambia la configuración de la mano para realizar un signo distinto pueden producir pequeñas zonas borrosas no detectadas en la *Validación* y consecuentemente resulten en el reconocimiento inválido del fotograma. Para evitar este problema se

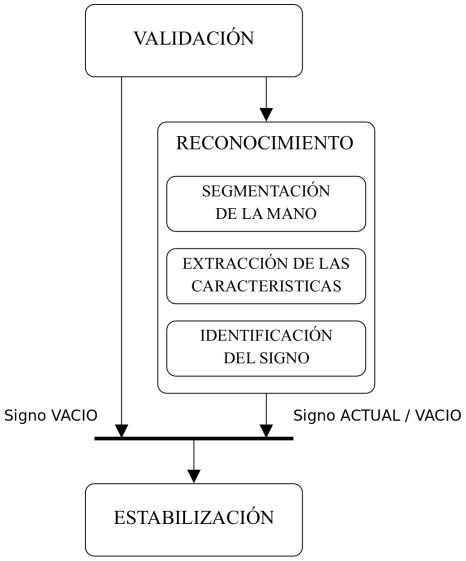


Figura 3.20: Detalle de la etapa del reconocimiento del signo y su relación con las etapas anterior y posterior.

incorpora la etapa de estabilización, previa a la *Utilización*, que considera la información actual respecto al contexto temporal (Fig. 3.21).

Suponga que para un momento dado k, existe una secuencia de signos $\{S_j\}, conj \in [0, k]$, donde cada S_j es el signo en la iteración j del ciclo de tratamiento de video, y S_k corresponde al signo actual. S_k es analizado por medio de una función estabilizadora f_{est} que tiene en cuenta un pasado cercano formado por los tres signos anteriores $\{S_{k-3}, S_{k-2}, S_{k-1}\}$. Dicha función determina si el signo estabilizado S'_k es el signo actual o se debe informar de otro distinto hasta que haya más pruebas que ratifiquen el cambio, evitando así la aparición de los cambios espúreos. La función estabilizadora $f_{est} = F(S_{k-3}, S_{k-2}, S_{k-1}, S_k) = S'_k$ se puede describir de la siguiente forma:

si $(S_k = S_{k-1})$ **entonces** $S'_k = S_k$
sino si $(S_k = S_{k-2})$ **entonces** $S'_k = S_{k-2}$
sino si $(S_{k-1} = S_{k-2})$ **entonces** $S'_k = S_{k-1}$
sino si $(S_{k-1} = S_{k-3})$ **entonces** $S'_k = S_{k-1}$
sino si $(S_{k-2} = S_{k-3})$ **entonces** $S'_k = S_{k-2}$
sino $S'_k = S_{VACIO}$

donde: S_{VACIO} representa la ausencia de signo. Esta estrategia se obtuvo a base de pruebas con distintos casos hipotéticos y el análisis de sus resultados.

En la Fig. 3.22 se expone una representación gráfica de cada una de las

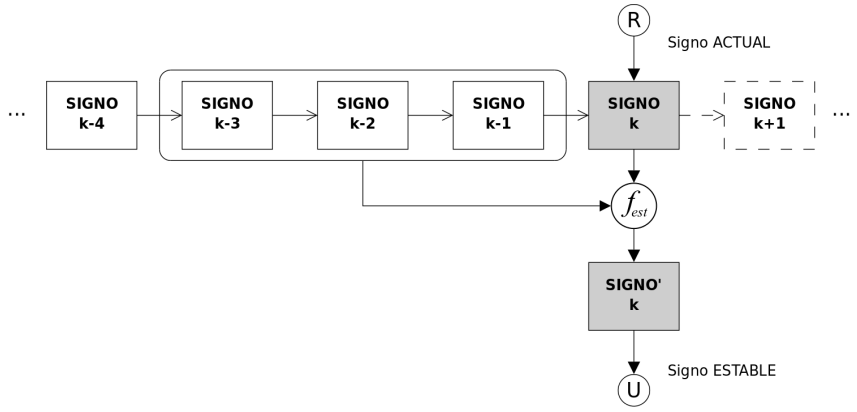


Figura 3.21: Detalle de la etapa estabilización del signo y su relación con las etapas *Reconocimiento* (R) y *Utilización* (U).

decisiones de la función estabilizadora anteriormente descriptas.

Como se vió en 3.2.2, en la etapa de *Validación del fotograma* se divide el flujo de datos en dos direcciones, siendo la etapa *Estabilización del signo* su punto de reunión. Cuando el fotograma actual no es validado, se pasa directamente a la *Estabilización* y se indica la ausencia de un signo mediante la utilización de S_{VACIO}. De este modo, se puede utilizar la función estabilizadora independientemente de la dirección que se toma.

El *Reconocimiento de un fotograma*, por su parte, no siempre devuelve un signo reconocido. Para estos casos donde no se detectó una silueta o las características extraídas no pudieron ser identificadas en el conjunto de patrones conocidos, se utiliza también S_{VACIO} como el elemento nulo de la secuencia de signos.

3.2.5 Utilización del signo

La última etapa dentro del ciclo de tratamiento del video corresponde al uso de la información extraída de la iteración actual. El signo reconocido o ausencia del mismo se presenta a la aplicación para realizar alguna acción específica, como puede ser detectar un cambio de signo, hacer seguimiento de su posición, mover un personaje, ejecutar un comando, etc., dependiendo del contexto.

La utilización de los datos generados está siempre presente en todo algoritmo útil. La mención de ésta dentro del diagrama de flujo se realiza para dejar bien claro en qué momento del ciclo se desarrolla el comportamiento final de la aplicación.

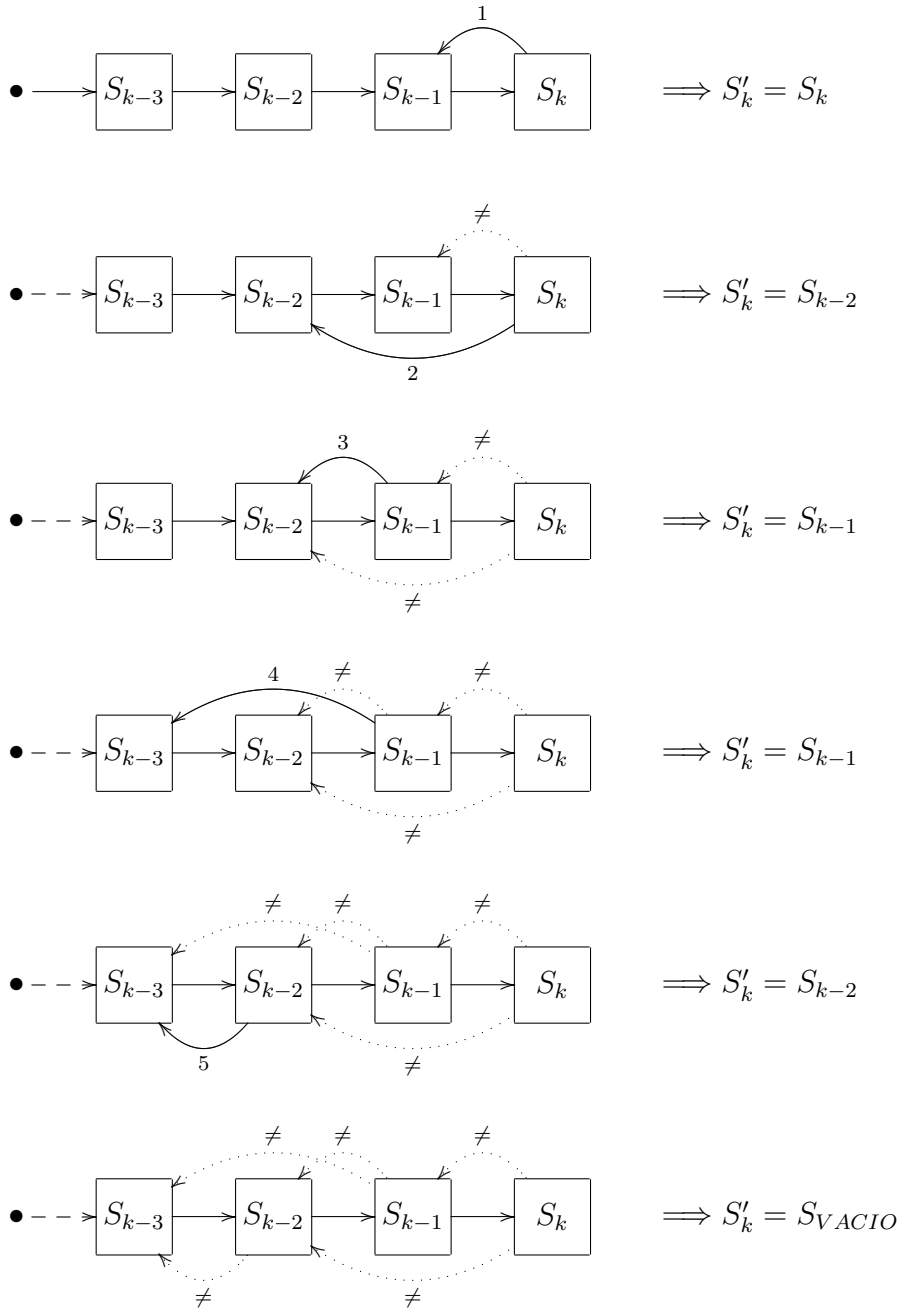


Figura 3.22: Representación gráfica de la función estabilizadora.

CAPÍTULO 4

Desarrollo del software

Existen muchas y variadas formas de llevar a cabo el desarrollo de un software en particular, sin embargo, los distintos procesos de creación de software contienen actividades fundamentales que son comunes a todos ellos:

1. *Especificación de requerimientos:* se analizan y establecen las funcionalidades del software y las restricciones de operación y desarrollo del mismo.

2. *Diseño del software:* se realizan descripciones precisas del paradigma del programa, la estructura, los datos y los algoritmos que serán implementados. Agrega formalidad a las especificaciones.

3. *Implementación:* es el proceso de convertir una especificación en un sistema ejecutable (codificación).

4. *Validación y Verificación:* son las inspecciones y revisiones tendientes a comprobar que el sistema está acorde a lo diseñado y que cumple con las expectativas del cliente.

5. *Evolución o Mantenimiento:* son las modificaciones que se necesitan hacer a un sistema existente para que continúe funcionando o adquiera nuevas capacidades.

La metodología empleada para llevar a cabo estas actividades es la denominada *desarrollo evolutivo*. En este enfoque no se tienen actividades separadas, sinó que las mismas se realizan concurrentemente y presentan una

alta retroalimentación a lo largo de todo el proceso [25]. Sin embargo, se ha realizado un esfuerzo por separar las mismas en procura de una mejor comprensión del proceso de construcción llevado a cabo. En consecuencia todas las actividades se presentan en su estado final de evolución, pudiéndose encontrar en algunas actividades referencias a cuestiones propias que surgieron de otras etapas.

En este capítulo se presentan las tres primeras actividades, mientras que la "Validación" se expone en el próximo capítulo *Experimentos y resultados*. Aquí la "Evolución" no se aplica al proceso de un software nuevo.

4.1 Especificación de requerimientos

La primer tarea que debe realizar un ingeniero al desarrollar software es comprender la naturaleza del problema o problemas que se tienen por resolver. Éstos pueden ser de todo tipo, desde los más triviales donde prácticamente la codificación es inmediata, hasta los más complejos que implican una investigación profunda del tema, el asesoramiento con especialistas, y el trabajo cooperativo con un grupo interdisciplinario. Las especificaciones de lo que hará el software (servicios) como lo que no (restricciones), se conocen con el término de *requerimientos*.

En el sistema propuesto tenemos dos fuentes de requerimientos: a) metodología del reconocimiento de los signos y el procesamiento del video, y b) funcionalidades esperadas del sistema mismo y en interacción con el usuario.

La primera se encuentra documentada en el capítulo anterior y constituye una especificación de las etapas y procedimientos que deben ser implementados. La segunda, que se formuló a través de las diversas iteraciones del desarrollo, es el objeto de la documentación de requerimiento presentada a continuación.

Una técnica sencilla y muy utilizada para relevar requerimientos es la del *caso de uso* (CU). Ésta identifica a los actores involucrados en una interacción particular y le proporciona un nombre. Un caso de uso consiste en una representación gráfica donde los actores se representan como figuras delineadas y cada interacción se representa como una elipse con su nombre. Cada CU tiene asociado una o más descripciones de la interacción. Actualmente es parte de los diagramas UML (del Inglés *Unified Modeling Language* – Lenguaje Unificado de Modelado) [26].

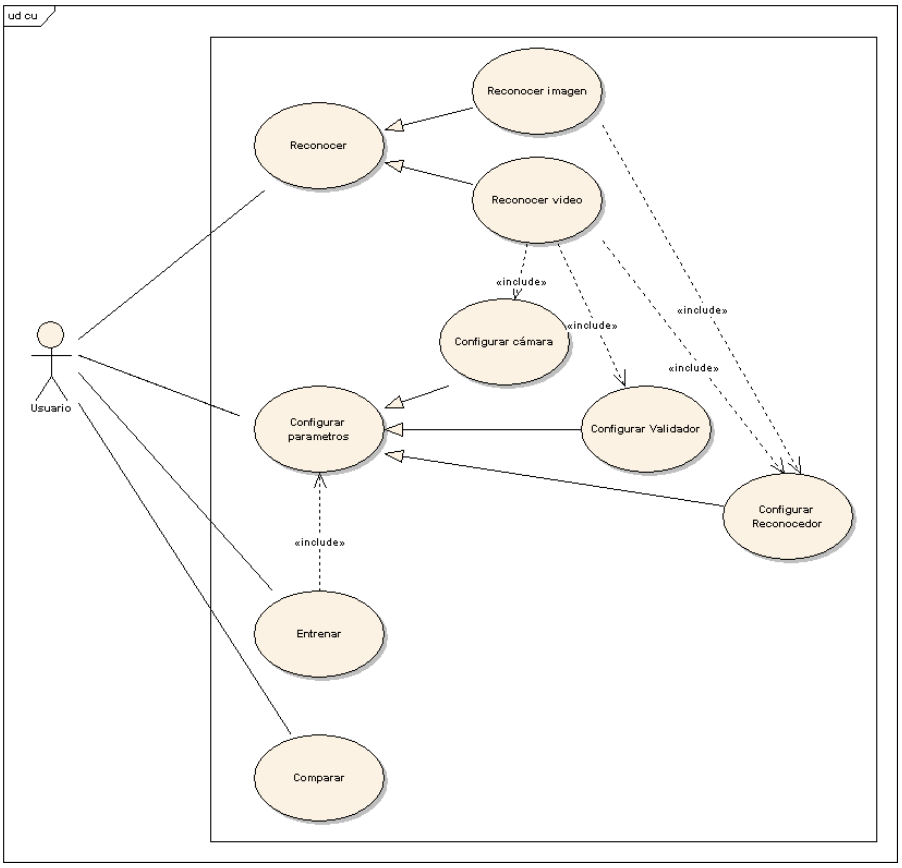


Figura 4.1: Diagrama general de los casos de uso del sistema.

La Fig. 4.1 muestra las interacciones principales entre el sistema y el usuario a través de un diagrama de caso de uso. El objetivo de los mismos se presenta por medio de sus descripciones textuales en lenguaje natural:

CU Reconocer. Este CU es la generalización del reconocimiento. El usuario establece el tipo de reconocimiento.

CU Reconocer imagen. Se ingresan como dato los nombres de las imágenes de fondo y mano que se encuentran almacenadas en archivos. Se configuran los parámetros del algoritmo de reconocimiento mediante el CU Configurar reconocedor. Se ejecuta el reconocimiento. Se muestran los resultados por pantalla y las imágenes son almacenadas como archivos.

CU Reconocer video. Se configuran la cámara y los demás recursos mediante los CU Configurar cámara, Configurar validador y Configurar

reconocedor. Luego se inicia el bucle de la captura de los fotogramas. El comportamiento (inicio y detención del procesamiento, cambios en la configuración, visualización de varibles internas, etc.) se controla por medio de opciones del teclado.

CU Configurar parámetros. Este CU es la generalización de las configuraciones. El usuario establece el valor para cada una de las opciones. De no hacerlo, los elementos del sistema quedan configurados con valores por defecto que permiten una aplicabilidad general.

CU Configurar cámara. Se define el formato de los fotogramas y la velocidad de captura.

CU Configurar reconocedor. Se establecen los parámetros del método de reconocimiento: tono de referencia, umbral de tono, umbral de color, conjunto de patrones utilizados en el etiquetado del signo y umbrales y ponderaciones utilizados por el clasificador. Se disponen también opciones especiales para visualizar el detalle interno de cada etapa del reconocimiento que permita analizar su desempeño.

CU Configurar validador. Se establecen los parámetros de dicha etapa: umbrales diferencia con el fondo y fotogramas consecutivos.

CU Entrenar. Se configuran la cámara y el método de reconocimiento mediante los CU Configurar cámara y Configurar reconocedor. Se presenta al usuario el conjunto de signos que formarán la base de datos de signos y se inicia el bucle de la captura de los fotogramas. Por medio del teclado el usuario puede navegar entre los signos disponibles, reconocer el fotograma actual y agregarlo al conjunto de patrones entrenados, guardar el listado entrenado en un archivo de texto con las imágenes originales, y demás opciones secundarias.

CU Comparar. Se ingresan los nombres de dos archivos de patrones correctamente etiquetados, uno se toma como conjunto entrenado y el otro como los patrones que se quieren identificar. Mediante la comparación del resultado del etiquetado de cada uno de los patrones del conjunto incógnita se determinan medidas del rendimiento del clasificador.

Cada uno de estos CU ha sido desarrollado en profundidad y todas las funcionalidades se encuentran ampliamente detalladas más adelante en la sección “Implementación”. Para no extender demasiado el documento ni ser reiterativo, aquí sólo se presentan como descripciones generales.

4.2 Diseño del software

El diseño del software es un proceso que agrega formalidad y detalle a la estructura del software que se va a realizar. No se obtiene directamente, sinó que se desarrolla de manera iterativa a través de diferentes versiones. Un proceso de diseño incluye la elaboración de modelos (representaciones gráficas) que sirven tanto para el análisis como la documentación de la especificación. En este proyecto se utilizan dos tipos de modelos distintos: flujo de datos y orientado a objetos.

Un *modelo de flujo de datos* se concentra en cómo fluyen los datos a través de una secuencia de pasos y cómo son transformados entre cada uno de ellos.

En un *modelo orientado a objetos* el sistema se piensa en términos de "cosas" (objetos) en lugar de operaciones o funciones. Los distintos objetos del sistema interactúan entre ellos, mantienen su propio estado local y suministran operaciones de esa información del estado. Estos objetos se crean conforme a una definición de clase de objeto que sirve como una plantilla. La notación a utilizar corresponde al UML propuesto en [26], donde una clase se representa con un rectángulo vertical con tres secciones: nombre de la clase en la parte superior, luego los atributos, y en última sección figuran las operaciones asociadas. Las relaciones entre las clases se modelan describiendo las asociaciones por medio de líneas y flechas.

Para la documentación de las etapas del capítulo 3 se ha adoptado una visión funcional del procesamiento de datos, utilizando para ello los diagramas de flujo de datos. Éstos resultan, a criterio del autor, más naturales, intuitivos y fáciles de explicar.

En el desarrollo del software se utiliza el modelo de diseño orientado a objetos al diagrama del procesamiento del video (Fig. 3.17). El resultado obtenido es un conjunto de clases y estructuras que pueden ser utilizadas tanto en conjunto como de manera independiente. De esta manera se obtiene un diseño flexible que a la vez permite la reutilización[1]. A continuación se presentan comentadas las clases más relevantes:

[1]El capítulo 6 presenta un ejemplo de un software aplicativo que reutiliza este diseño.

ReconocedorMano. Clase principal que realiza todas las tareas necesarias para el reconocimiento en un fotograma (sección 3.2.3). Almacena la imagen del fondo, la imagen de la mano y los parámetros de la segmentación. El estado con el que finaliza el proceso de reconocimiento se define en la estructura *ResultadoProcesoMano*, y los datos (características, etiqueta y ubicación) dentro de la clase *Signo*. Se delega a la clase *ContenedorPatron* la base de conocimiento y el método necesario para realizar el etiquetado. Para utilizarse dentro del esquema de procesamiento de video se dispone de la función *noReconocer()* que se invoca cuando el fotograma actual no fuese válido (para actualizar los estados internos). Incluye la *Región de Interés* o ROI (explicado con detalle en 5.6.5). Incorpora opciones de asistencia para el análisis y estudio del desempeño del reconocimiento (por ejemplo, medición de tiempos insumidos y generación de imágenes parciales para cada uno de los distintos pasos internos).

Signo. Es la unidad de información que informa el *ReconocedorMano* al exterior sobre el signo reconocido. Es sencillamente la unión de un *Patron* y una *Ubicación*, pero deja abierta la posibilidad de agregar otros objetos si los requerimientos lo estipularan. Se entendió que las características de una seña son independientes de su posición espacial, por tanto se diseñaron como objetos separados.

Patron. Reune las características extraidas a la silueta de la mano. Su nombre (etiqueta) almacena el significado atribuido por el sistema.

Ubicación. Almacena la posición y dimensión espacial de un signo que le permite al sistema realizar dos o más acciones distintas para un mismo signo según la ubicación de éste en la imagen original.

ContenedorPatron. Administra un conjunto de patrones etiquetados en el que basa el algoritmo de clasificación para la identificación de un patrón dado. Permite llevar a cabo el proceso de entrenamiento (incorporación de nuevos patrones y sus respectivas etiquetas) mediante operaciones de lectura y escritura de archivos.

ValidadorFotograma. Encargada de las operaciones correspondientes a la etapa *Validación del fotograma* (sección 3.2.2). Al igual que *ReconocedorMano*, esta clase incorpora la especificación de la ROI para analizar únicamente aquella región de la imagen que será posteriormente utilizada por el reconocedor.

EstabilizadorSigno. Encargada de las operaciones correspondientes a la etapa *Estabilización del Signo* (sección 3.2.4). Su entrada está formada por el signo de la iteración actual. Cuando el reconocedor puede identificar el signo actual, éste se asigna al estabilizador mediante el método *setSignoActual(Signo)*. En el caso donde el fotograma no es válido o no pudo ser reconocido, la entrada se representa por un *signo vacío* y se indica mediante el llamado al método *setSignoVacio()*.

GraficadorSigno. Esta clase se encarga de dibujar la información del proceso sobre el fotograma actual. Para ésto se la define como «*friend*» de las clases *ReconocedorMano* y *EstabilizadorSigno*, y así puede acceder directamente a todos los datos privados que serán graficados. El contenido está controlado por banderas que indican que debe aparecer en la imagen. Algunas de ellas son: las características del *Signo*, la silueta de la mano, la ROI y el detalle de la función estabilizadora (secuencia de signos de entrada y su resultado).

En la Fig. 4.2 se muestran las clases relacionadas exclusivamente con el reconocedor, y en la Fig. 4.3 el diagrama de las principales clases involucradas en el procesamiento del video.

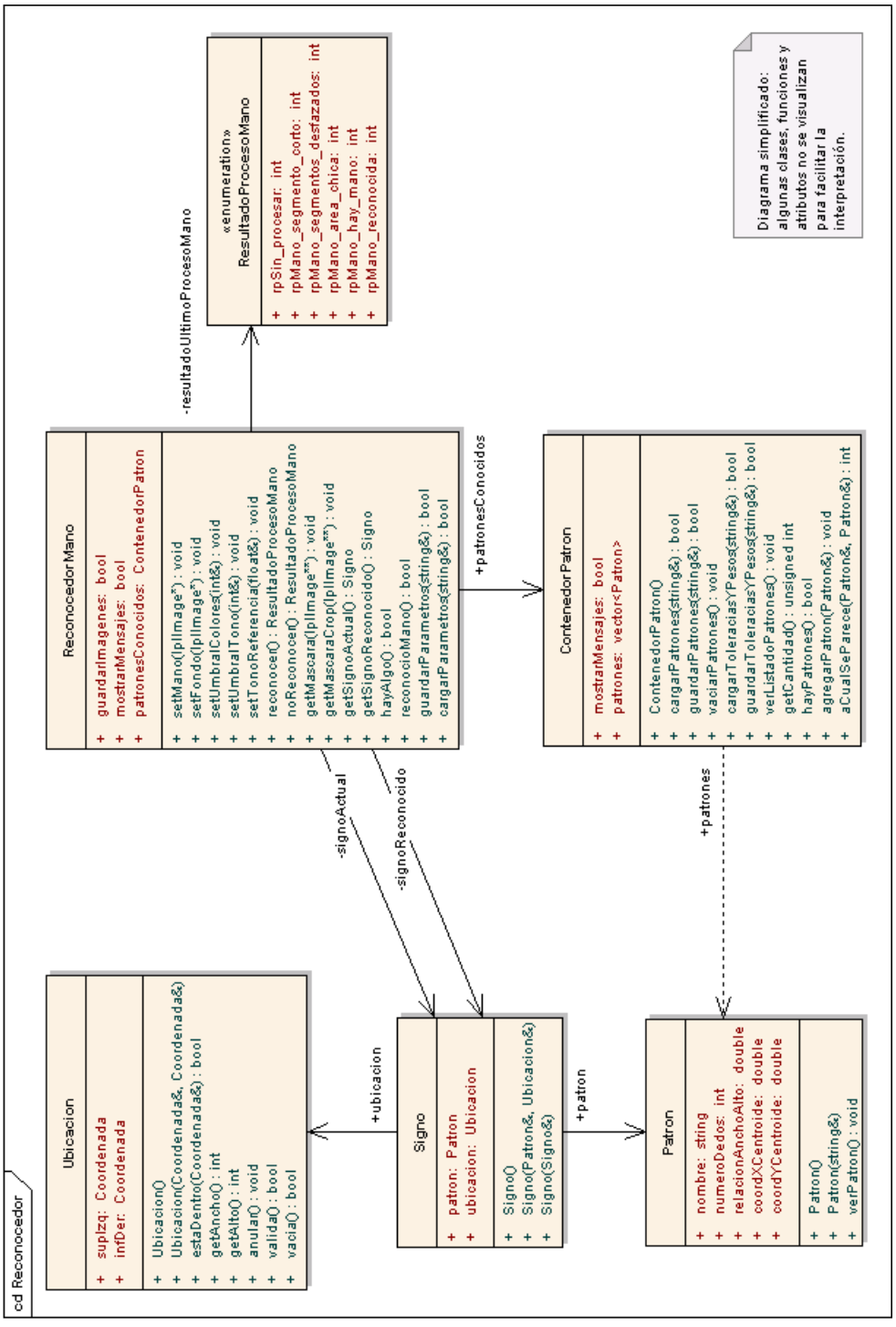


Figura 4.2: Diagrama UML de las clases más relevantes diseñadas para el procesamiento de un fotograma.

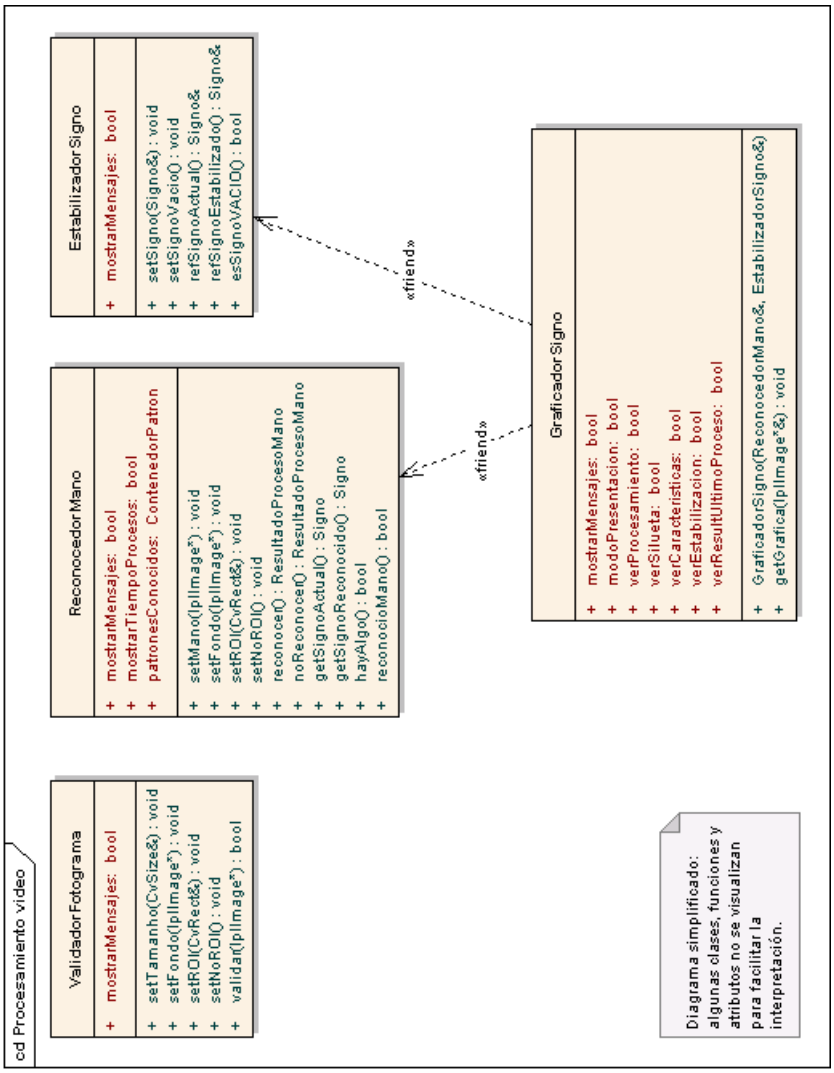


Figura 4.3: Diagrama UML de las clases principales diseñadas para el tratamiento del video.

(a) (b) (c) (d)

Figura 4.4: Selección de la región de la mano para cálculo de parámetros. (a) Imagen del puño y región circular seleccionada, (b) Conjunto de píxeles incluídos en el cálculo, (c) Canal de tono correspondiente, (d) Histograma de los valores de tono.

4.2.1 Asistencia en la configuración

De la experimentación realizada durante el proceso de desarrollo del software (descripta en el capítulo 5) se determina que, los parámetros tono de referencia y umbral de tono del método de reconocimiento, deben calibrarse correctamente para cada situación particular y lograr así un buen desempeño. Para facilitar al usuario el cálculo de los valores adecuados se diseñó el siguiente algoritmo:

1. De una imagen se selecciona una región de la mano que incluya todas las variciones posibles, y siempre dentro de la mano. Si la captura de la imagen es interactiva, se recomienda utilizar la mano cerrada con los dedos hacia la cámara (Fig. 4.4(a)), de esta manera se pueden abarcar zonas internas (palma) y externas (en dedos flexionados) de la mano, las uñas y las sombras (Fig. 4.4(b)).

2. Sobre dicha región se extrae el canal de tono (Fig. 4.4(c)) y se calcula su histograma (Fig. 4.4(d)). En este caso el histograma será una función discreta $h(r_k) = n_k$ donde $r_k \in [0 - 360)$ es el k-ésimo valor de tono y n_k es el número de píxeles con el valor r_k. El número total de píxeles en la región seleccionada es $N = \sum n_k$.

3. Con la distribución del tono se calculan los estadísticos *media*, *mediana* y *desvío* según las siguientes ecuaciones:

$$media = \frac{1}{N}\sum r_k h(r_k) \tag{4.1}$$

$$mediana = r_j | \sum_{k=0}^{max(j)} h(r_k) \leq N/2 \tag{4.2}$$

$$desvío = \sqrt{\frac{\sum r_k h(r_k)^2}{N} - media^2} \tag{4.3}$$

4. Los estadísticos se utilizan para la definición de los parámetros. La media determina el valor del tono de referencia y el desvío se utiliza como umbral de tono. En este punto se debe prestar atención (tal como se hiciera en 3.1.2) a la periodicidad del canal de tono. Una distribución concentrada pero con media cercana a cero se verá como dos grupos de datos, separados uno en cada extremo del intervalo $[0, 360)$. Por consiguiente, los estadísticos no pueden ser directamente empleados para la interpetación real de la distribución del tono. Para evitar este inconveniente, se utiliza el histograma que presente el mínimo desvío entre el histograma original y dos versiones del histograma rotado 90° cada una respecto de la anterior[2]. De esta manera se logra que cualquier distribución (se espera una dispersión acotada alrededor de un único valor) se encuentre lejos del extremo y permita definir correctamente los parámetros. Los estadísticos media y mediana calculados para el histograma de mínimo desvío se deben corregir, y esto se logra restando a sus valores la cantidad del desplazamiento realizado (0°, 90°o 180°), unificándolos luego al rango $[0, 360)$ si esto diera negativo. En la Fig. 4.5 se representa un histograma y sus distintas rotaciones para ejemplificar cómo varían sus estadísticos. En este caso en particular, se aprecia una importante disminución en el desvío para la rotación de 30°, pero no suficiente para alejarse completamente del extremo del rango. El mínimo desvío se alcanzó en la rotación de 180°. Se puede observar cómo al acotarse el desvío la media se hace más precisa.

5. El umbral de tono se ajusta con el desvío, que sólo puede tomar valores por encima del desvío mínimo. En una situación real el tono experimentará variaciones mayores, producto de los cambios de posición de la mano respecto de las fuentes de iluminación. Por otra parte, si el

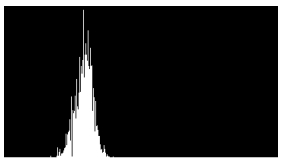
[2]Rotar el histograma es equivalente, y más simple, que rotar el tono y luego calcular su histograma.

(a) Original

	Calculado	Corregido
μ	53,74	–
q_2	19	–
σ	103,28	–

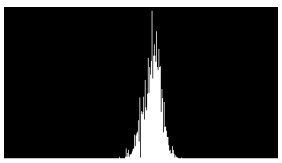
(b) Rotado 30°

	Calculado	Corregido
μ	46,08	16,08
q_2	46	16
σ	15,47	15,47



(c) Rotado 90°

	Calculado	Corregido
μ	105,71	15,71
q_2	106	16
σ	11,9331	11,9331



(d) Rotado 180°

	Calculado	Corregido
μ	195,71	15,71
q_2	196	16
σ	11,9329	11,9329

Figura 4.5: Ejemplo de rotaciones del histograma y su incidencia en los estadísticos ($\mu = media$, $q_2 = mediana$ y $\sigma = desvío$). (a) Histograma original, (b), (c) y (d) Histogramas rotados en 30°, 90° y 180° respectivamente.

desvío es superior a un umbral máximo, se infiere que la región no tiene un tono homogéneo y por lo tanto no se espera una correcta segmentación. En este último caso, la comparación entre la media y la mediana aporta la información necesaria para saber de si la distribución no está centrada en un único valor.

Este algoritmo se lleva a cabo por las siguiente clases (Fig. 4.6):

HistogramaTono. Clase que contiene el histograma (arreglo de 360 elementos), estadísticas (media, mediana, desvío) y el desplazamiento de la rotación en el cual se encuentran los valores del histograma.

AnalizadorHistogramaTono. Clase que permite extraer los píxeles de una imagen según una máscara que define cuales son los píxeles de interés, se calculan sus valores de tono y se genera el histograma que se almacena en la clase *HistogramaTono*. Como complemento de las funcionalidades el histograma puede ser provisto arbitrariamente. Provee las operaciones de cálculo de estadísticos, rotación y generación de una gráfica 2D. El algoritmo descripto es realizado por el método *buscarDesvioMinimo*.

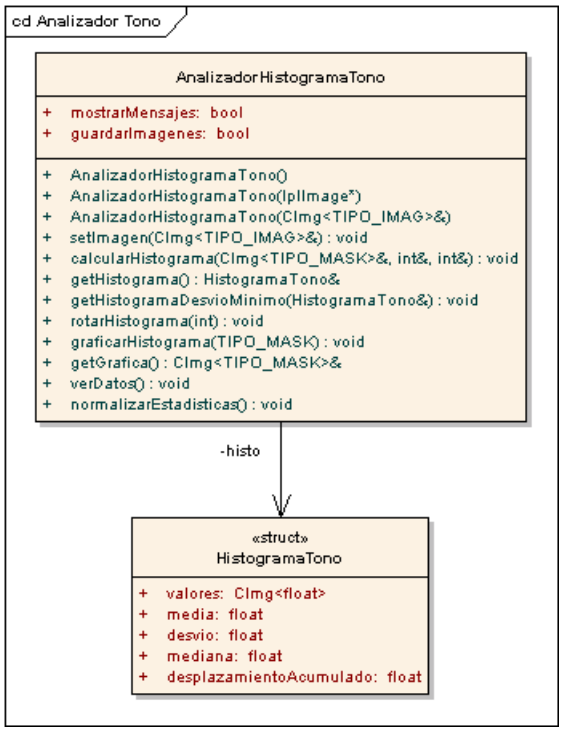


Figura 4.6: Clases que calculan los valores óptimos de los parámetros tono de referencia y umbral de tono.

4.3 Implementación

La implementación se realizó en C++ utilizando bibliotecas y herramientas libres que permiten que el software sea independiente de la plataforma. Entre ellos están los siguientes:

CImg [27]**:** Es una biblioteca para procesamiento de imágenes basada en plantillas de C++. De doble licencia, una de ellas es CeCILL v2.0, compatile con GNU GPL. De esta sólo se utilizó la clase *CImg* como contenedor para las imágenes. Los algoritmos son implementaciones propias adaptadas a las tareas específicas que se realizan en el procesamiento.

OpenCV [28]**:** Es una biblioteca libre de visión artificial originalmente desarrollada por Intel y publicada bajo licencia BSD, que permite que sea usada libremente para propósitos comerciales y de investigación. Sus características son: multiplataforma, licencia BSD, diseñada específicamente para procesamiento de imágenes, posee primitivas para una interfaz gráfica simple y muy buena documentación [29].

Se utilizó en la captura individual de los frames desde la cámara web, visualización y operaciones correspondientes a la etapa de *Validación del fotograma*. No se emplearon las características avanzadas.

Codelite [30]**:** Es un Entorno de Desarrollo Integrado (IDE) multiplataforma que facilita la construccción de proyectos de software e implementa un front-end para el depurador GDB[3]. Muy simple de usar y de aspecto similar al Visual Studio de Microsoft.

La investigación y desarrollo del método de reconocimiento propuesto, como también la forma de interactuar con el dispositivo de adquisición se llevaron a cabo mediante el uso de prototipos. En palabras de Somerville [25]: *Un prototipo es una versión inicial de un sistema de software que se utiliza para demostrar los conceptos, probar las opciones de diseño y, de forma general, enterarse más acerca del problema y sus posible soluciones*. En función del objetivo que se quiera alcanzar, existen dos tipos de prototipos: desechable y evolutivo.

[3]Disponible en `http://www.gnu.org/software/gdb/download/`.

Un *prototipo desechable* se realiza para ayudar a refinar y clarificar las especificaciones del sistema. Este tipo de prototipo se centra en la experimentación. Tiene un período de vida corto, puesto que una vez que se ha evaluado, éste es desechado y no se utiliza para el desarrollo posterior.

Por su parte, un *prototipo evolutivo* se basa en la idea de desarrollar una implementación inicial relativamente simple, la cual se va refinando a través de las distintas etapas. Al final del proceso, el prototipo se convierte en el sistema requerido.

Aquí, el primer paso consistió en definir, implementar y validar las etapas propuestas para el reconocimiento de un fotograma (capítulo 3.1). Para ello se realizó una implementación directa del diagrama flujo de la Fig. 3.1 en prototipos desechables, creando una rutina para cada etapa que se ejecutaban secuencialmente. Estos prototipos permitieron probar distintas estrategias y, a la vez, comprender mejor el problema a resolver. A continuación se desarrolló un prototipo evolutivo (denominado como proyecto manoviva) empleando el modelo orientado a objetos. Las clases diseñadas anteriormente se implementaron en una aplicación de consola que cuenta con tres modos de trabajo: *reconocimiento de signos*, *entrenamiento del sistema*, y *comparación de patrones*. La selección y configuración del modo se realiza por medio de sus parámetros de inicio, listados en la Tabla 4.1.

Reconocimiento

En este modo se permite realizar el reconocimiento de signos manuales en dos contextos: a) procesamiento local, y b) procesamiento del video.

El *procesamiento local* realiza el reconocimiento sobre un archivo de imagen pasado por parámetro. Los resultados y detalles de los algoritmos se almacenan en disco como archivos de texto e imágenes.

El *procesamiento del video* realiza el reconocimiento del flujo de video desde una cámara web (por defecto) o desde un archivo de video. Incluye todas las etapas propuestas en la sección 3.2.

En ambos casos se pueden inicializar, por medio de los parámetros de inicio, los valores del umbral de color, umbral de tono, tono de referencia, ROI y el conjunto de patrones de la base de datos que se desean utilizar para clasificar el signo. Si no se establecen dichos patrones, el proceso de reconocimiento sólo se limita a la extracción y visualización de las características del signo presente.

Parámetro	Por defecto	Descripción
-l	false	Procesar las imágenes <f> y <m> que se pasen como parámetro.
-f		Imagen Fondo.
-m		Imagen Mano.
-vermensajes	false	Mostrar mensajes del reconocedor.
-vertiempos	false	Mostrar los tiempos del procesamiento.
-imagparciales	false	Guardar imágenes parciales del reconocimiento.
-ucolor	35	Umbral de Color.
-utono	25	Umbral de Tono.
-tono	20	Tono de referencia.
-selroi	false	Seleccionar ROI en modo interactivo.
-seltono	false	Seleccionar region para calibrar tono y umbral en modo interactivo.
-parametros	false	Cargar ucolor, utono, tono y ROI desde el archivo parametros.txt.
-p	false	Cargar archivo de patrones <ptrFuente> al reconocedor.
-ptrFuente	ptrFuente.txt	Archivo de patrones.
-ubinario	70	Umbral que binariza las diferencias.
-uconsecutivos	4	Fotogramas que se esperan antes de antes de comparar fotogramas consecutivos.
-dfondo	5000	Mínima diferencia para considerarlo distinto al fondo.
-dseguidos	2000	Máxima diferencia para considerar que dos fotogramas consecutivos son parecidos.
-formato	2	Formato webcam [1:chico, 2:mediano, 3:grande] {352x288, 640x480, 1280x960}.
-codec	MPEG	Sigla del codec de video (4 caracteres) al grabar (en tiempo de ejecución).
-framerate	25	FPS del codec al grabar un video.
-v	false	Procesar un archivo <video> en vez de webcam.
-video		Archivo de video.
-pausa	false	Realiza una pausa después de cada fotograma leído.
-entrenar	false	Entrenar signos: crea el archivo de patrones.
-comparar	false	Comparar patrones de la forma <ptrPrueba> vs. <ptrConocido>: calcula porcentaje de aciertos.
-ptrPrueba	ptrPrueba.txt	Archivo de patrones a reconocer.
-ptrConocido	ptrConocido.txt	Archivo de patrones conocidos.

Tabla 4.1: Parámetros configurables del software.

Algunos ejemplos de ejecución en modo reconocimiento:

1. Procesamiento local:
   `$ manoviva -l -m mano.bmp -f fondo.bmp`

2. Inicializar tono de referencia a 350°:
   `$ manoviva -l -m mano.bmp -f fondo.bmp -tono 350`

3. Procesamiento de video (cámara web):
   `$ manoviva`

4. Cargar los patrones que se utilizarán para identificar el signo:
   `$ manoviva -p -ptrFuente dbpatrones.txt`

5. Procesamiento de video desde el archivo ejemplo.avi:
   `$ manoviva -v -video ejemplo.avi`

El proceso de reconocimiento de un flujo de video es implementado por medio de las clases principales presentadas en la Fig. 4.3 y las funciones específicas que provee la biblioteca OpenCV. Su utilización es muy sencilla puesto que la codificación sigue la forma presentada en el diagrama de flujo de la Fig. 3.17, es decir, se crea un bucle en el cual se realizan los siguientes pasos:

1. Se extrae un nuevo fotograma mediante la primitiva `cvQueryFrame()`.

2. Se encadenan las operaciones de los objetos *ValidadorFotograma*, *ReconocedorMano* y *EstabilizadorSigno*.

3. Se responde al signo estabilizado por medio de alguna función específica.

4. Se espera un tiempo para visualizar los resultados (uso del objeto *GraficadorSigno*) y recibir eventos desde el usuario.

En la Fig. 4.7 se presenta un fragmento de código como ejemplo.

```
...
while (true) {
  // Obtener siguiente fotograma
  frame = cvQueryFrame(capture);

  // Pasar imagen al reconocedor (usado luego por el graficador)
  reconocedor.setMano(frame);

  // Validar fotograma --> Reconocer (si/no) --> Estabilizar signo
  if (validador.validar(frame)) {
    reconocedor.reconocer();

    if (reconocedor.reconocioMano())
      estabilizador.setSigno(reconocedor.getSignoActual());
    else
      estabilizador.setSignoVacio();

  } else {
    reconocedor.noReconocer();
    estabilizador.setSignoVacio();
  }

  // Implementar función de respuesta
  hacerHalgoConSigno(estabilizador.refSignoEstabilizado());

  // Mostrar el resultado del procesamiento
  graficador.getGrafica(frameAux);
  cvShowImage( VENTANA, frameAux );

  // Espera entre frames (milisegundos): 20fps => 50ms
  // y procesar teclado
  tecla = cvWaitKey(50);
  switch ( (char)tecla ) {
    case '?': mostrarMenu(); break;
    ...
  }
}
...
```

Figura 4.7: Código de ejemplo para el procesamiento de un flujo de video.

Al iniciar el procesamiento del flujo de video se abren las dos ventanas que se muestran en la Fig. 4.8. Una de ellas exhibe lo que visualiza la cámara[4] y brinda un conjunto amplio de acciones disponibles desde el teclado. La Tabla 4.2 muestra el menú principal de operaciones que se pueden realizar con el teclado.

La otra ventana muestra tres controles gráficos que facilitan el ajuste a los valores óptimos de los parámetros umbral de color, umbral de tono y tono de refencia; a su vez visualiza el resultado del procesamiento del fotograma actual por medio del objeto *GraficadorSigno*.

El graficador es controlado mediante el submenú de opciones de la Tabla 4.3. La opción principal está determinada por la tecla `m` que establece cual de sus dos formas (o modos) de mostrar la información se utilizará:

- *Presentación:* cuando se encuentra activo, sólo se muestran en pantalla la etiqueta del signo a utilizarse por la aplicación (signo estabilizado) y los marcadores de las esquinas de la región donde fue detectada la mano. De esta manera se tiene una visualización prolija sobre el desempeño del reconocimiento realizado.

- *Procesamiento:* este es el modo alternativo cuando el anterior no está activo. El objetivo de este modo es brindar información al desarrollador. Los datos visualizados son: características del signo actual, región de la mano, resultado del procesamiento (si no hay presencia de mano los datos anterios no aparecen y se lo informa), detalle de la estabilización (secuencia de etiquetas $(S_k, S_{k-1}, S_{k-2}, S_{k-3})$ descendentemente y su resultado S'_k sobre la derecha), y en caso que se haya definido, aparecerá la ROI. Cuando no se carga la base de datos de patrones conocidos, se asigna como etiqueta del signo el nombre clave `__PATRON_MANO_VIVA__`. Estos datos son configurables, pueden o no estar graficados de forma independiente.

En la Fig. 4.9 se exhiben en detalle los resultados de ambos modos para un mismo caso de ejemplo.

[4]En adelante se referirá sólo a la visualización y extracción de fotogramas desde la cámara web, sin embargo las mismas operaciones se realizan cuando el flujo de video proviene de un archivo en disco.

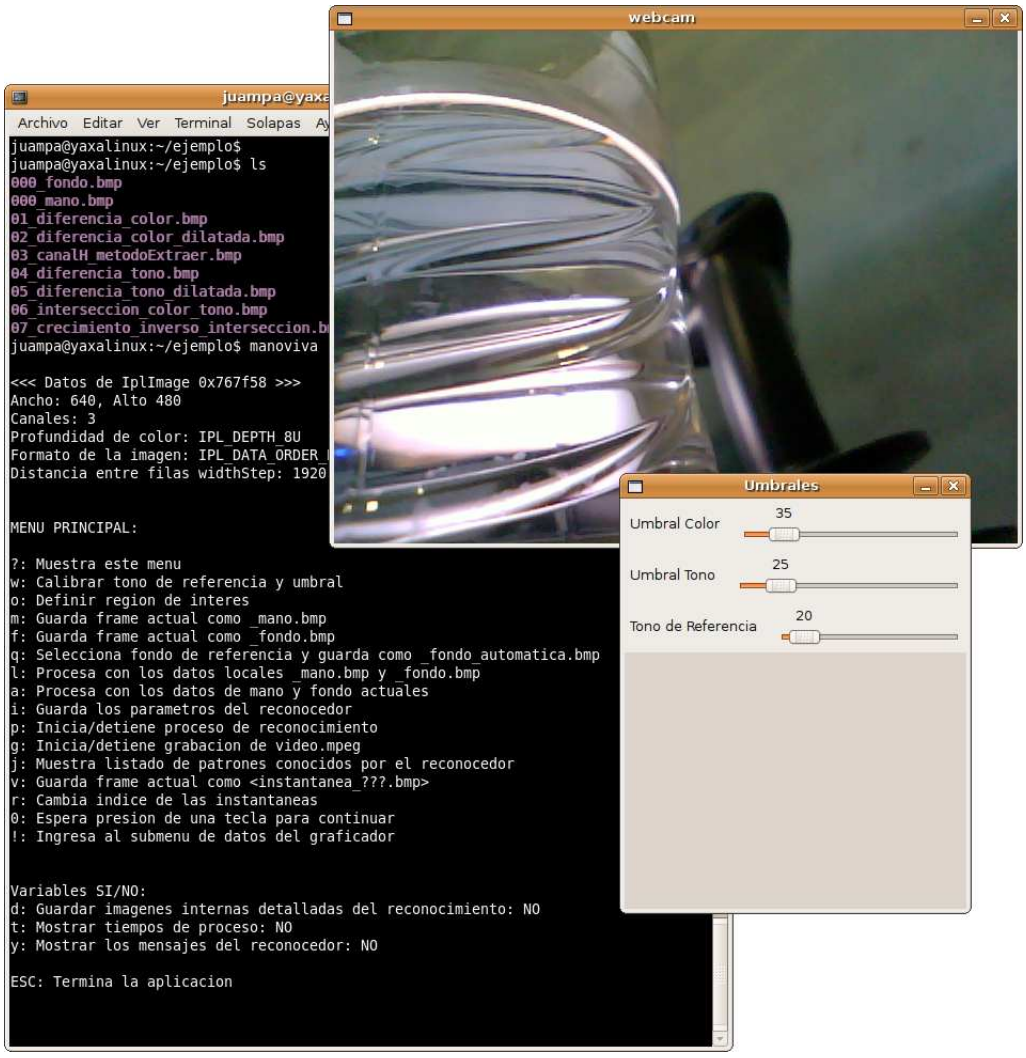


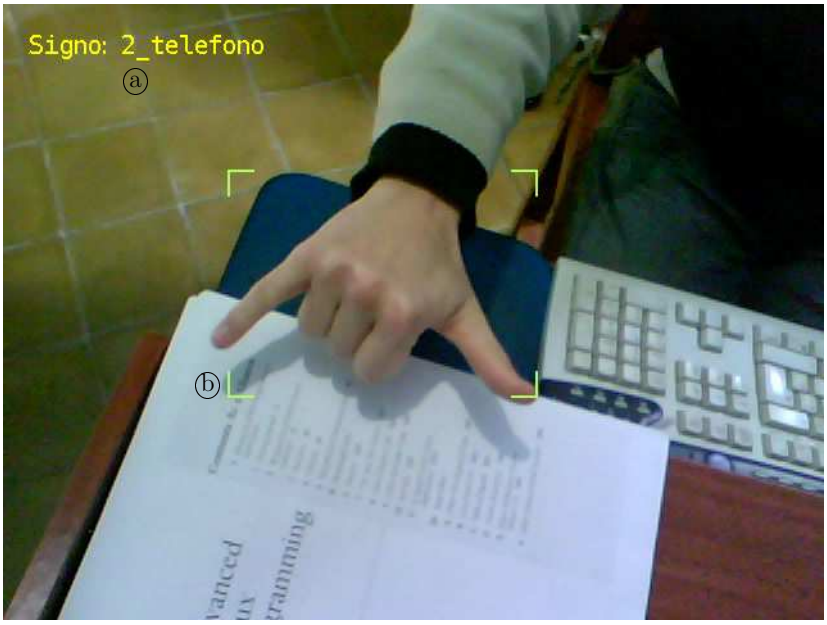
Figura 4.8: Interfaz del modo reconocimiento en procesamiento de video. Se observa la consola mostrando el menú de opciones y las dos ventanas principales: *Webcam* y *Umbrales*.

Tecla	Descripción
?	Muestra este menú.
w	Calibrar tono de referencia y umbral.
o	Definir región de interés.
m	Guarda el frame actual como _mano.bmp.
f	Guarda el frame actual como _fondo.bmp.
q	Guarda el frame actual como _fondo_automatica.bmp.
l	Procesa con los datos locales _mano.bmp y _fondo.bmp.
a	Procesa con los datos de mano y fondo actuales.
p	Inicia/detiene el procesamiento del video.
g	Inicia/detiene la grabación de video.mpeg.
j	Mostrar el listado de patrones conocidos por el reconocedor.
v	Guardar la imagen como <instantanea_???.bmp>.
r	Cambiar el índice de las instantáneas.
0	Espera presión de una tecla para continuar.
!	Ingresa al submenú de datos del graficador.
d	Mostrar imagenes internas detalladas del reconocimiento: <si/no>.
t	Mostrar tiempos de proceso: <si/no>.
y	Mostrar mensajes del reconocedor: <si/no>.
ESC	Termina la aplicación.

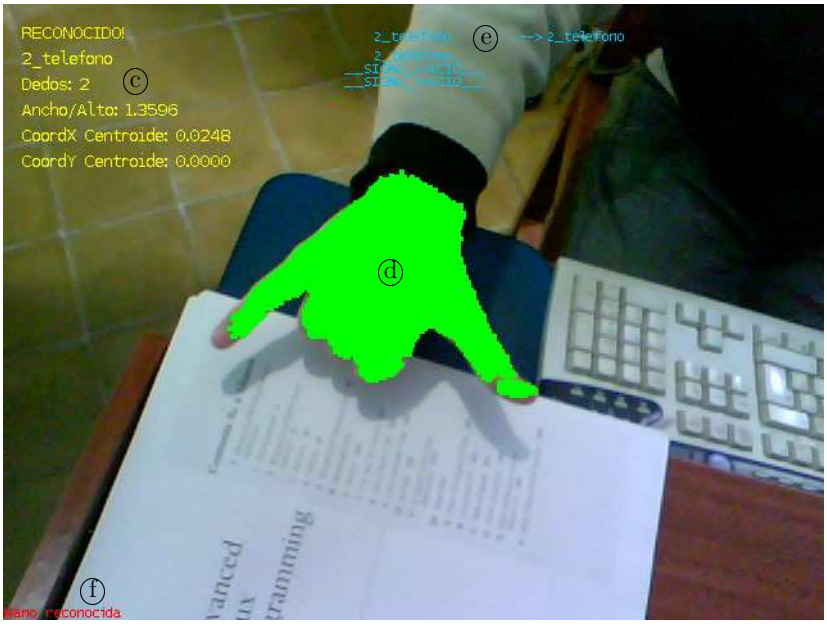
Tabla 4.2: Menú principal de acciones que permiten al usuario interactuar durante el procesamiento de video.

Tecla	Descripción
?	Muestra este submenú.
m	Modo presentación: <si/no>.
s	Silueta: <si/no>.
u	Resultado último proceso: <si/no>.
c	Características: <si/no>.
e	Estabilización: <si/no>.
ESC	Salir del submenú.

Tabla 4.3: Submenú de datos del graficador.



(a) Modo presentación (`verPresentacion = true`)



(b) Modo procesamiento (`verPresentacion = false`)

Figura 4.9: Opciones del graficador, ubicación y colores por defecto de los datos presentados. Referencias: ⓐ Etiqueta del signo estabilizado, ⓑ Esquinas de la región correspondiente a la mano, ⓒ Características del signo, ⓓ Silueta de la mano, ⓔ Detalle de la estabilización, ⓕ Resultado del reconocimiento de la imagen.

(a) (b) (c) (d)

Figura 4.10: Distintos estados en la selección de región de tono. (a) Seleccionando (tamaño de región muy pequeño), (b) Seleccionando, (c) Definida, (d) Moviendo.

Tecla	**Descripción**
?	Muestra este submenú.
c	Calcular los datos de la región seleccionada.
r	Resetear la región de selección.
s	Guargar la imagen original y región de referencia.
ESC	Salir de la calibración.

Tabla 4.4: Menú de calibración de tono de referencia y su umbral.

Lo descripto hasta aquí es la interfaz básica disponible en todo momento. A continuación se presentan dos opciones especiales que necesitan de la interacción del usuario a través de ventanas, ellas son: *calibrar tono de referencia y umbral* (tecla `w`), y *definir región de interés* (tecla `o`). La primera abre una ventana llamada *Calibrar Tono* visualizando el fotograma actual y permite seleccionar (como también mover y eliminar) una región circular de la cual se estimarán el tono de referencia y su umbral en función del histograma, utilizando para ello el algoritmo presentado en la sección 4.2.1. En la Fig. 4.10 se ven los distintos colores que se utilizan para destacar los distintos estados de la región. Una vez definida la región se procede a realizar los cálculos sobre la misma, por medio de la tecla `c` de su menú de teclado (ver Tabla 4.3). Los valores óptimos se muestran por consola, se asignan automáticamente a los controles de la ventana *Umbrales*, y se exponen gráficamente en la ventana emergente *Histograma Tono*. Si el histograma de tono no cumple con el criterio de dispersión acotada propuesto en el punto 5 del algoritmo, se muestran por consola los mensajes sobre tal situación. En las Figs. 4.11 y 4.12 se visualiza la selección de la región y el resultado del cálculo de los parámetros para dos ubicaciones distintas de la región. En consola se muestran los estadísticos de los histogramas rotados según el método propuesto, el resultado obtenido de aquel con el mínimo desvío y los mensajes de conclusión sobre la región. La visualización del histograma se realiza de manera tal que el tono de refe-

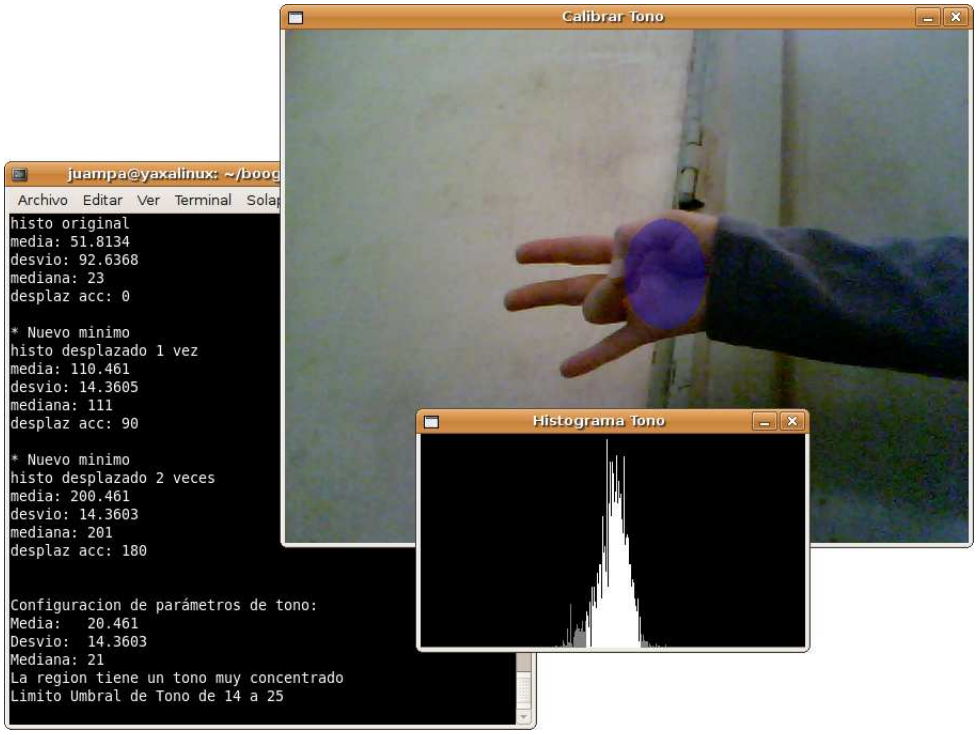


Figura 4.11: Ejemplo de selección de región de tono correcta. El histograma de mínimo desvío se alcanza en la rotación de 180°. El umbral de tono se aumenta al valor mínimo 25 determinado por experiencia.

rencia se ubique en el centro del histograma, y el rango de valores dentro del umbral de tono aparezcan en color blanco, mientras que el resto se colorea en gris.

La segunda opción que se menciona refiere a la definición de la región de interés. Para esto se abre la ventana *Definir ROI* que cuenta con el mismo principio de funcionamiento que la opción anterior. De manera interactiva se selecciona una región rectangular, comprobando al finalizar que no sea menor a un determinado tamaño preestablecido (Fig. 4.13). Esta ventana es inicializada con la ROI actual (en caso de haber sido definida previamente). Sólo posee un par de funciones del teclado, una para remover la región de interés (tecla `r`), y otra para terminar la operación (tecla `ESC`). En la Fig. 4.14 se puede apreciar cómo la definición de una ROI incide sobre el resultado del procesamiento.

Estas dos opciones interactivas para configurar el reconocedor fueron agregadas en el procesamiento local por medio de los parámetros de inicio `-seltono` y `-selroi` (Tabla 4.1).

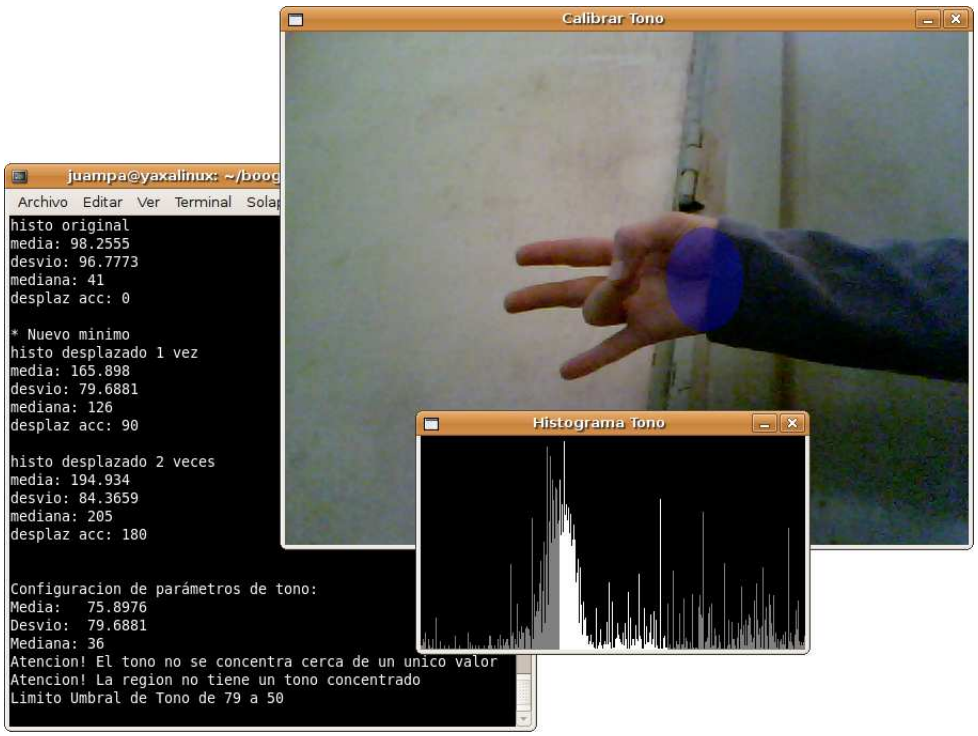


Figura 4.12: Ejemplo de selección de región de tono no adecuada. El histograma de mínimo desvío se alcanza en la rotación de 90°. El umbral de tono se limita al valor máximo 50 e informa que la región no es homogénea.

(a) (b)

Figura 4.13: Distintos estados en la selección de la ROI. (a) Seleccionando, (b) Definida o Moviendo.

(a) (b)

Figura 4.14: Ejemplo de procesamiento con y sin ROI. (a) ROI definida, (b) Sin ROI, se procesa toda la imagen.

Entrenamiento

El sistema utiliza un entrenamiento supervisado, es decir, la definición de los tipos (clases) de signos se realiza de forma manual por un supervisor antes de utilizar el sistema. Para este proyecto se ha establecido un conjunto fijo de 16 signos diferentes (el detalle completo de los signos propuestos se puede ver en la sección 5.3).

Un entrenamiento se realiza mediante el siguiente comando:

```
$ manoviva -entrenar
```

La mecánica de trabajo propuesta sigue los siguientes cuatro pasos:

Paso 1: Identificar el entrenamiento.

En primera instancia el software pide un nombre para el sujeto que realiza los signos y un identificador para el conjunto entrenado. De esta manera se pueden ordenar las imágenes originales, auxiliares, y los resultados, dentro de una estructura de directorios de la siguiente manera: desde el directorio donde se ejecuta la aplicación (por ejemplo local) se crea el árbol de directorios entrenamiento $\rightarrow$ sujeto$N$ $\rightarrow$ conjunto$K$ (Fig. 4.15(a)), el contenido es alojado en el directorio conjunto$K$ (Fig. 4.15(b)).

Paso 2: Definir el fondo y parámetros comunes a todas las capturas.

La interfaz presenta las mismas dos ventanas auxiliares del modo reconocimiento necesarias para visualizar la cámara y configurar los parámetros (Fig. 4.8). Cambia únicamente la forma de responder al teclado para realizar las acciones propias del entrenamiento listadas en la Tabla 4.5. En este paso se toma una imagen del fondo sobre el cual se van a realizar los

(a) (b)

Figura 4.15: Estructura de directorios creada para almacenar los datos del entrenamiento. (a) Árbol de directorios, (b) Ejemplo del contenido alojado.

Tecla	Descripción
?	Muestra este menu.
i	Mostrar signo actual.
a	Pasar al singo anterior.
s	Pasar al signo siguiente.
d	Mostrar todos los signos disponibles.
c	Mostrar los patrones conocidos actualente.
y	Mostrar mensajes del reconocedor.
k	Mostrar mensajes del contenedor de patrones.
o	Imagenes internas detalladas del proceso de reconocimiento.
f	Seleccionar imagen de fondo.
e	Entrenar el frame actual.
g	Guardar el entrenamiento y configuracion.
r	Recargar el archivo de patrones.
v	Vaciar los patrones entrenados.
w	Seleccionar region para tono de referencia.
z	Actualizar los parametros del reconocedor.
ESC	Salir del entrenamiento.

Tabla 4.5: Menú principal del modo entrenamiento.

signos y se definen los valores para los umbrales de color y tono, y el tono de referencia que serán constantes a lo largo de todo el entrenamiento. Se cuenta aquí también con la opción de seleccionar una región de la cual estimar los parámetros tono de referencia y umbral de tono implementados en el modo reconocimiento.

Paso 3: Entrenar cada uno de los signos propuestos.

A continuación se procede al entrenamiento de cada uno de los signos. En consola se muestra el signo esperado, avanzando automáticamente al siguiente luego de su entrenamiento. El usuario en cualquier momento puede iterar sobre dicha lista para entrenar nuevamente un signo si observa un error en la imagen procesada, como puede ser una imagen borrosa por movimiento o equivocación al realizar el signo. Para cada signo existe una única entrada dentro de una lista, por lo que el reentrenamiento de un signo reemplaza los datos del anterior. Para cada signo entrenado se almacenan en el directorio correspondiente el fotograma capturado, una copia con la impresión de las características extraídas, y una máscara binaria de la silueta de la mano (Fig. 4.15(b)). El nombre de estos archivos se corresponden a [signo].bmp, enmascarada_[signo].bmp y mascaracrop_[signo].bmp; donde [signo] corresponde al nombre o etiqueta que se ha definido para cada uno de los signos del listado propuesto.

Paso 4: Guardar los datos y terminar la aplicación.

Finalizado el entrenamiento resta volcar en disco la lista de las características extraídas para cada signo y los valores de los parámetros utilizados en el procesamiento. Estos datos se guardan en los archivos de texto patrones.txt y parametros.txt respectivamente. En la Fig. 4.16 se presenta el contenido de estos archivos correspondientes a una sesión de entrenamiento real.

Comparación

En este modo de trabajo se comparan dos conjuntos de patrones aprendidos en distintos entrenamientos y se determina la efectividad del método clasificador de patrones.

Como se vió, cada entrenamiento produce un archivo de a lo sumo 16 patrones (uno por cada signo); sin embargo, las comparaciones pueden realizarse con conjuntos de cantidades arbitrarias. Para ésto la lectura de los archivos se realiza secuencialmente hasta encontrar la palabra clave `CHAU` (omitiendo todo el resto del contenido) o alcanzar el fin del archivo. Dicha palabra clave se incluye por defecto al guardar el entrenamiento (ver línea 18 de la Fig. 4.16(a)), su ausencia no invalida la carga del contenido.

```
1 # Patron: <nombre> <nro. dedos> <Rel. Ancho/Alto> <Coord. X Centroide>
  <Coord. Y Centroide>
2 1_indice 1 0.490842 0.0447761 -0.208791
3 1_pulgar_der 1 1.31481 0.220657 0.0617284
4 1_pulgar_izq 1 1.44056 -0.223301 0.00699301
5 1_pulgar_vert 1 0.722222 0.102564 -0.212963
6 2 2 0.47012 0.101695 -0.115538
7 2_ele 2 0.819608 -0.244019 -0.254902
8 2_telefono 2 1.22162 -0.079646 0.0486486
9 2_curva 2 0.527881 -0.0704225 -0.115242
10 2_cuernito_vert 2 0.539062 0.0144928 -0.09375
11 3 3 0.715356 -0.256545 -0.093633
12 3_menores 3 0.589212 0 -0.0705394
13 4 4 0.636691 -0.00564972 -0.0215827
14 5 5 0.863309 -0.0583333 -0.0503597
15 punho 0 0.692308 -0.015873 0.010989
16 palma_horizontal 0 1.98592 0.0141844 0.028169
17 palma_vertical 0 0.448399 -0.111111 -0.0320285
18 CHAU
```

(a) Patrones entrenados

```
1 # Los parametros son: Umbral Color, Umbral Tono y Tono de Referencia
2 # Region de Interes (ROI): ROI Coord_SupIzq(x,y) Coord_InfDer(x,y) ; noROI
3 35 25 15
4 noROI
```

(b) Parámetros del reconocimiento

Figura 4.16: Contenido de los archivos de texto del entrenamiento. a) Patrones de 16 signos entrenados, uno por cada línea, b) Valores del reconocedor y ROI.

Para realizar esta comparación se puede ejecutar alguno de los siguientes comandos:

1. Utilizar por defecto los archivos ptrPrueba.txt y ptrConocido.txt:
   `$ manoviva -comparar`

2. Indicar cual es el nombre de los archivos con los datos:
   `$ manoviva -comparar -ptrPrueba` prueba
   `-ptrConocido` conocido

3. Comparar entrenamientos de dos sujetos distintos, suponiendo que se está en el directorio entrenamiento:
   `$ manoviva -comparar`
   `-ptrPrueba` sujeto1/conjunto1/parametros.txt
   `-ptrConocido` sujeto3/conjunto2/parametros.txt

Los *patrones conocidos* son aquellos que se tomarán como la base de conocimiento para realizar la identificación[5]. Si el patrón a comparar se parece a uno de ellos, entonces se reconoce el signo como aquel más parecido. El detalle del método de etiquetado se puede revisar en la sección 3.1.4. Los *patrones de prueba* son los candidatos que se desean reconocer. Éstos simulan ser los casos reales que se darán dentro del proceso normal de una aplicación. Dado que ámbos conjuntos de patrones poseen el nombre correcto (designado en el entrenamiento), se pueden calcular los siguientes estadísticos:

a) **Porcentaje de reconocidos**: patrones que han podido ser etiquetados correcta o incorrectamente. Todos deben ser etiquetados, sinó se estaría en presencia de problemas con la elección de los coeficientes del clasificador o signos del entrenamiento mal realizados.

b) **Porcentaje de reconocidos idénticamente**: el nombre propuesto por el proceso de etiquetado para un patrón de prueba coincide realmente con su nombre original. Es siempre menor o igual que el porcentaje de reconocidos.

La efectividad del clasificador se extrae del porcentaje de reconocidos idénticamente. Los resultados analizados más adelante (en la sección 5.5) son extraídos de éste cálculo.

[5]Representan el entrenamiento del sistema.

CAPÍTULO 5

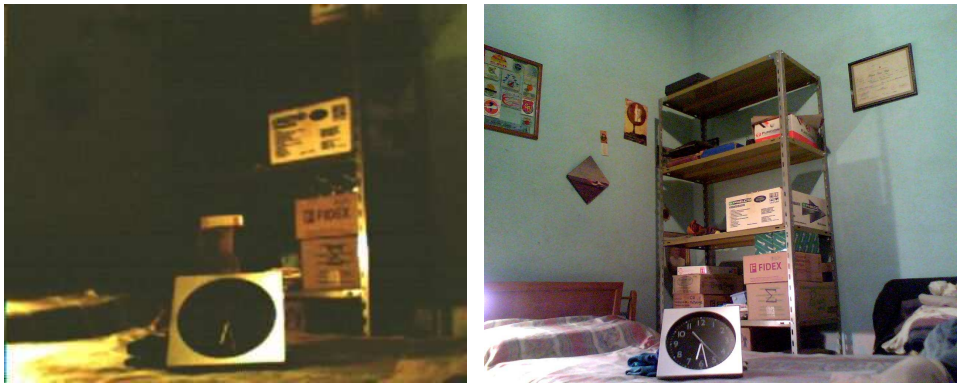
Experimentos y resultados

En este capítulo se presentan los experimentos sobre el hardware y el software que exploran las capacidades del método propuesto y guían el proceso de optimización, los detalles del corpus de imágenes registrado, y los resultados del desempeño obtenido.

5.1 Evaluación de las cámaras web

Para determinar la incidencia del dispositivo hardware, se compararon las imágenes de una misma escena capturadas al mismo tiempo (Fig. 5.1), ubicando ambas cámaras prácticamente en el mismo lugar (una cámara al lado de la otra). Se preparó una escena en una habitación ubicando distintos objetos a diferentes distancias, con colores y texturas variadas; iluminada únicamente con luz artificial (una lámpara incandescente de 60 Watts en el centro del techo y un velador con una lámpara incandescente de 40 Watts).

Para el modelo iLook la calidad de la imagen es evidentemente inadecuada, no refleja las condiciones reales de iluminación y distorsiona los colores obteniendo un tinte amarillento en toda su extensión (Fig. 5.1(a)). Por su parte, el modelo Slim obtiene una imagen que abarca una mayor superficie (mayor apertura de la lente), una claridad muy superior y conserva los colores reales (Fig. 5.1(b)).



(a) (b)

Figura 5.1: Imágenes tomadas por 2 cámaras simultáneamente desde una posición similar. (a) Genius iLook 110, (b) Genius Slim 1322AF.

Estos resultados posicionan al modelo Slim como el adecuado para emplear como dispositivo de adquisición de imágenes. La configuración de trabajo es la siguiente: resolución de 640×480 píxeles, utilizando el controlador UVC [1] sobre la distribución GNU/Linux Ubuntu 8.04.

5.2 Preproceso de la imagen

Las imágenes presentan una mínima degradación, observable al convertirlas al espacio HSV. Ésta es visible en los canales de Tono y Saturación debido a la compresión que hace el empaquetado 4:2:2 del formato de video propio de la cámara. Se puede observar que en estos dispositivos está presente un ruido de color generado por la electrónica propia de la cámara, típicamente gaussiano y de energía significativa en condiciones de baja iluminación. Esto motivó el análisis con métodos de eliminación de ruido, filtros de promediado y mediana, ambos con máscaras de 5×5 [17]. Como el proceso de extracción de la silueta de la mano incluye una umbralización y procesos de dilatación y erosión, la incidencia de estos filtros en el resultado final es despreciable. Por lo tanto, y pensando en las optimizaciones necesarias para la implementación en tiempo real, se consideró innecesario agregar una etapa de preproceso a las imágenes adquiridas.

[1] http://linux-uvc.berlios.de/ USB Device Class Definition for Video Devices

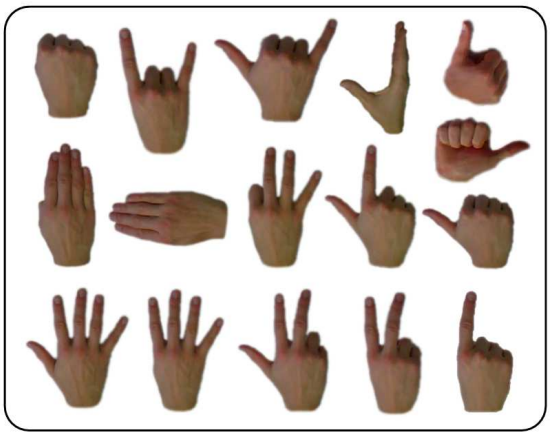


Figura 5.2: Conjunto de los 16 signos diferentes en la base de datos.

5.3 Base de datos

La base de datos utilizada para el entrenamiento y pruebas de sistema fue conformada por un conjunto de 144 imágenes, pertenecientes a las tres interpretaciones de 16 signos realizadas por tres sujetos. Los signos propuestos se pueden observar en la Fig. 5.2.

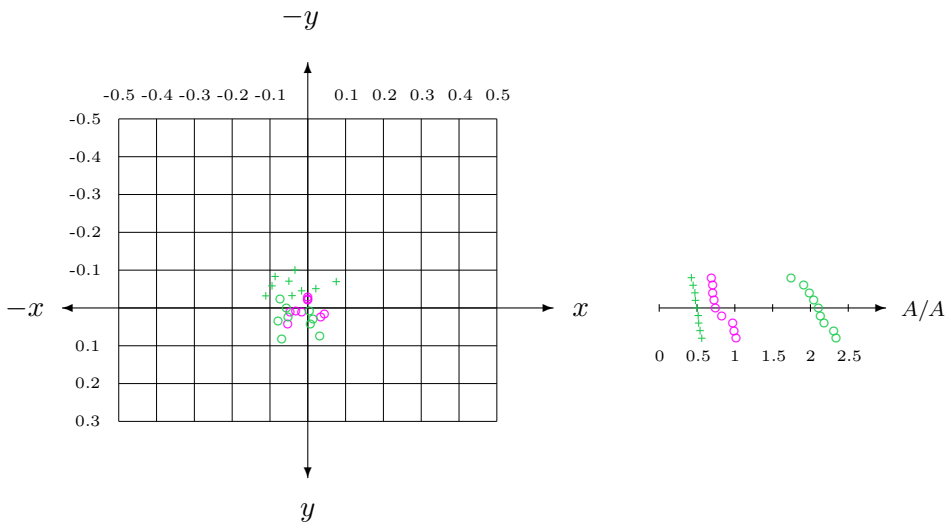
Las imágenes fueron adquiridas en un ambiente interior con un ligero aporte de luz artificial (iluminación no dedicada). Al capturar las imágenes se priorizó que el sujeto realizara las señas sin condicionamientos previos, guiándose por imágenes de referencia, y con la mayor naturalidad posible sombre distintos contextos espaciales y de iluminación. En la Fig. 5.3 se pueden observar tres de las nueve imágenes utilizadas en el entrenamiento del signo 3_menores (la descripción de los signos se presenta en la Tabla 5.1). Debido a la naturalidad, las diferentes realizaciones de una misma seña presentan leves variaciones como ser separación y ángulo entre dedos, inclinación de la mano, etc., aún perteneciendo a un mismo sujeto.

Para los 16 signos posibles se calcularon todas las características propuestas en la sección 3.1.3. Todos los valores calculados se exponen en las Figs. 5.4, 5.5, 5.6, 5.7, 5.8 y 5.9, agrupadas por la cantidad de dedos en el signo, con el objeto de ver su distribución. De éstas se puede observar que siempre existe una característica bien separada de la otra cuando dos signos tienen valores similares en las coordenadas del centroide o en la relación ancho/alto (por ejemplo los signos *Puño* y *Palma horizonal* en la Fig. 5.4).

Figura 5.3: Ejemplo de distintas realizaciones de un signo.

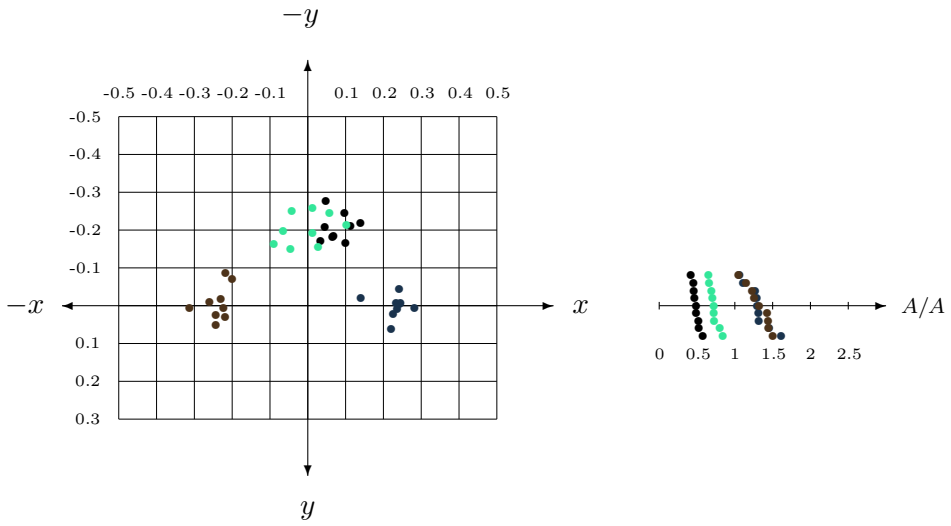
Signo	Etiqueta	Signo	Etiqueta
	1_indice		1_pulgar_der
	1_pulgar_izq		1_pulgar_vert
	2_cuerno_vert		2_curva
	2_ele		2
	2_telefono		3_menores
	3		4
	5		palma_horizontal
	palma_vertical		punho

Tabla 5.1: Descripción de los signos.



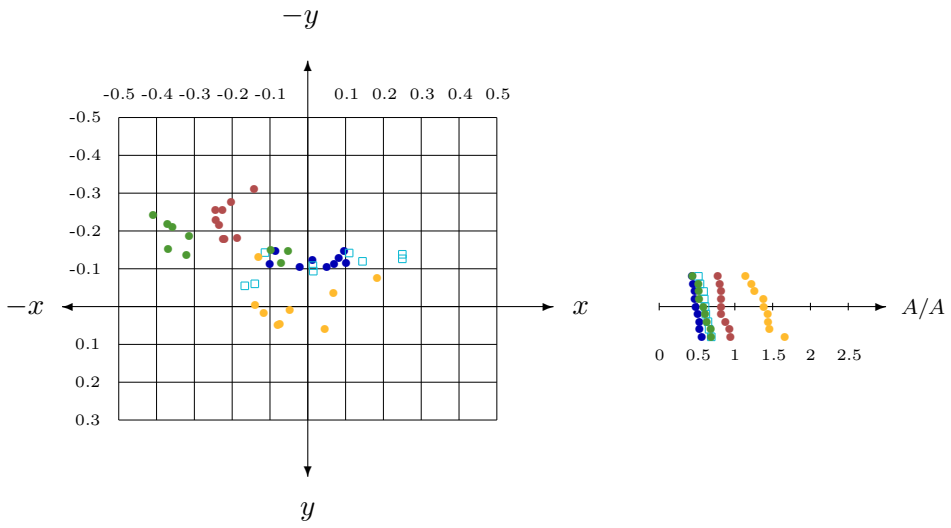
(a) Coordenadas del centroide (b) Relación Ancho/Alto

Figura 5.4: Distribución de parámetros para signos sin dedos. Donde: ○ Puño, + Palma vertical, ○ Palma horizontal.



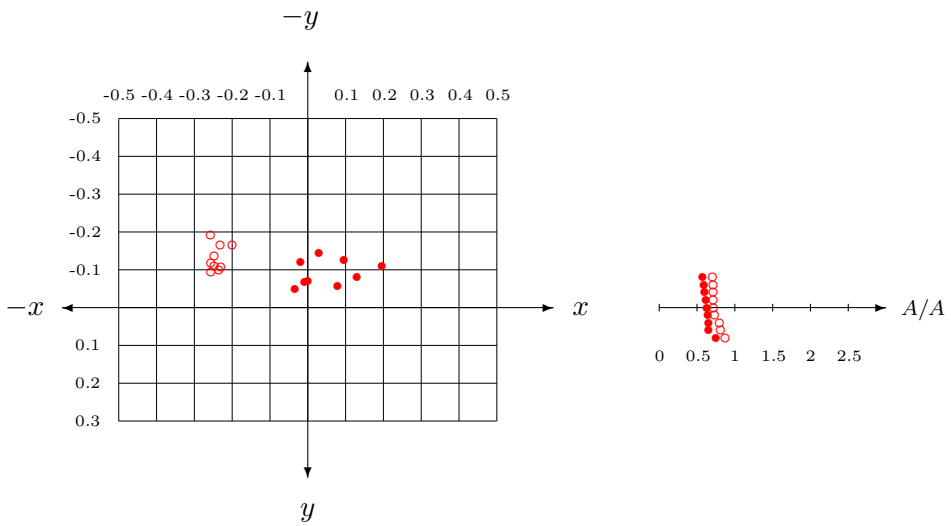
(a) Coordenadas del centroide (b) Relación Ancho/Alto

Figura 5.5: Distribución de parámetros para signos de un dedo. Donde: ● Índice, ● Pulgar hacia la derecha, ● Pulgar hacia la izquierda, ● Pulgar vertical.



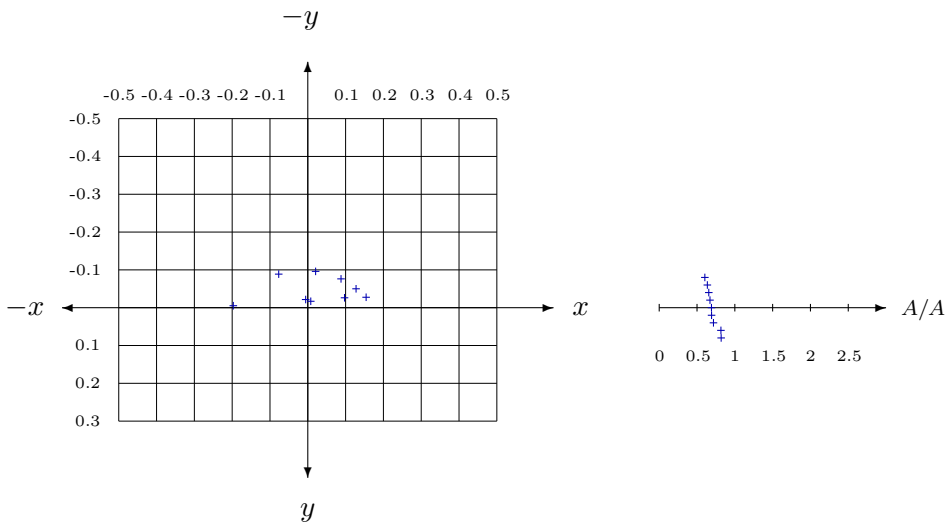
(a) Coordenadas del centroide (b) Relación Ancho/Alto

Figura 5.6: Distribución de parámetros para signos de dos dedos. Donde: ● Dos, □ Cuernito, ● Ele, ● Teléfono, ● Curva.



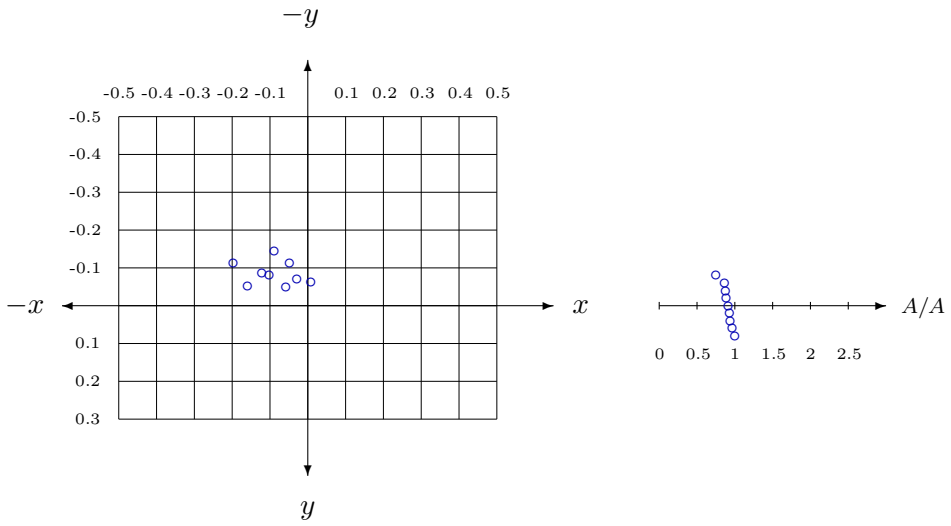
(a) Coordenadas del centroide (b) Relación Ancho/Alto

Figura 5.7: Distribución de parámetros para signos de tres dedos. Donde: ○ Tres, ● Tres menores.



(a) Coordenadas del centroide (b) Relación Ancho/Alto

Figura 5.8: Distribución de parámetros para signos de cuatro dedos. Donde: + Cuatro.



(a) Coordenadas del centroide (b) Relación Ancho/Alto

Figura 5.9: Distribución de parámetros para signos de cinco dedos. Donde: ○ Cinco.

5.4 Configuración y calibración

Para la mayoría de los parámetros (como las tolerancias y los pesos del vector de características y el umbral de color RGB) se encontraron valores que funcionaron adecuadamente en todos los casos de prueba. Sin embargo, los parámetros relacionados al tono (su valor de referencia y umbral) demostraron ser críticos para el rendimiento del sistema. Con la selección de una región, y no de una vecindad de un punto, se resolvió la forma de estimar un umbral de tono adecuado a cada situación en particular.

Las experiencias también muestran que se debe tener cuidado en la manera en que se ilumina una escena, como se expone más adelante.

5.4.1 Parámetros del reconocedor

Los umbrales de diferencia de color y diferencia de tono, el tono de referencia y la imagen de fondo deben ser obtenidos durante un proceso de calibración, previo al uso del sistema. Como se introdujo previamente, el fondo es una imagen patrón adquirida en ausencia de la mano. Especificar el umbral de color no es trivial, aunque se puede estimar una aproximación inicial que depende de las condiciones de iluminación, en los experimentos se utilizó un valor de 35. El tono de referencia se obtiene como el promedio de los tonos de una región de la mano, seleccionada por el usuario. A su vez, de la misma se puede estimar un valor para el umbral de tono utilizando la dispersión de los valores respecto de su media (sección 4.2.1).

Como se vió en la etapa de *Identificación de signo* (sección 3.1.4), el algoritmo propuesto define un conjunto de tolerancias y pesos para la selección de la etiqueta correspondiente. Del entrenamiento realizado se establecieron experimentalmente los siguientes valores:

- Tolerancia para número de dedos: 0 (estricto).
- Tolerancia para relación de aspecto: 0,30 (30 %).
- Tolerancia para similitud de centroides: 0,20.
- Ponderación para relación de aspecto: 1.
- Ponderación para similitud de centroides: 5.

5.4.2 Incidencia de la iluminación sobre el tono

El proceso de segmentación por color se tornó crítico respecto a las condiciones de iluminación, sobre todo en situaciones extremas. Una importante cantidad de iluminación direccional puede provocar sombra en algunas zonas de la mano, afectando a la imagen capturada de forma "peligrosa", comprometiendo la segmentación. En la Fig. 5.10 se muestra una escena con luz natural excesiva que produce una imagen con demasiado contraste, la mano presenta una parte muy brillosa (donde incide directamente la luz que ingresa por la ventana) y el resto con poca intensidad. Intuitivamente, la segmentación no debería presentar inconvenientes, sin embargo, el canal de tono está severamente comprometido. Se observa que, sobre la extensión de la mano, el canal de tono no es homogéneo. Existen zonas con píxeles negros, grises claros y oscuros, y en menor medida blancos. El tono se define como una medida circular, por lo tanto los valores extremos, 0° y 360°, representan el mismo valor. Para visualizar los datos se debe mapear esta escala a una escala lineal entre 0 y 255, lo que presenta inconvenientes: el salto entre negro y blanco aparece como una gran diferencia en la escala lineal cuando en realidad no lo es. Para entender por qué falla la segmentación examinamos tres zonas distintas y sus correspondientes histogramas (Fig. 5.11):

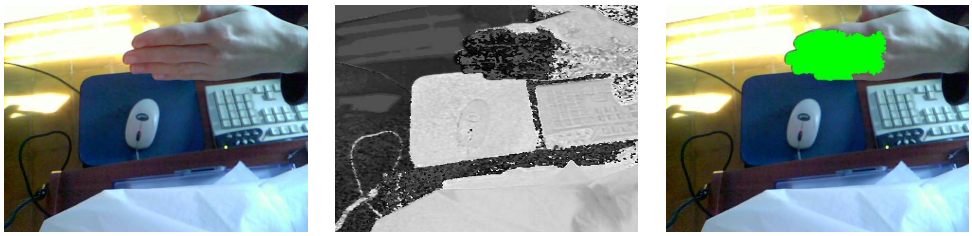
a) Todos los valores son oscuros. Si bien no se concentran en un único valor, el umbral de tono abarca los píxeles suficientes para crear una region homogénea, luego de la dilatación.

b) Ésta es una zona de transición donde se observa una dispersión total del tono. No existe ningún valor de umbral acotado que permita una extracción satisfactoria de la silueta.

c) Zona con alta concentración del tono, se puede extraer correctamente la silueta de la mano. El proceso implica una variación del tono de referencia a 261° (185 en escala lineal de grises).

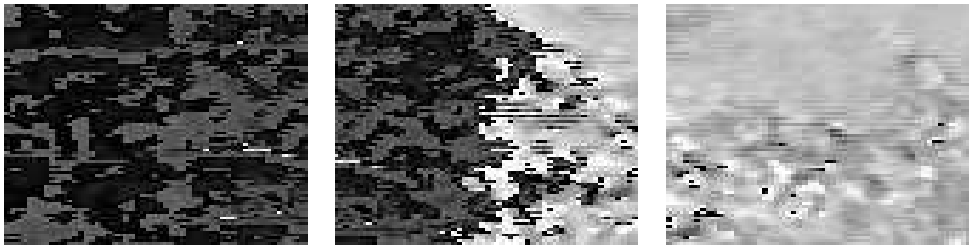
Las características del tono en estas condiciones de iluminación no pueden ser reparadas con ningún filtro o procesamiento extra. Éste problema debe considerarse como la escena donde el sistema no puede utilizarse.

También se estudió el fenómeno opuesto, se procedió a disminuir la iluminación de la escena. En la Fig. 5.12 se muestra que, al contrario de lo intuído a priori, una iluminación deficiente no altera tanto la homogeneidad del canal de tono (Fig. 5.13) y la segmentación puede llevarse a cabo satisfactoriamen-

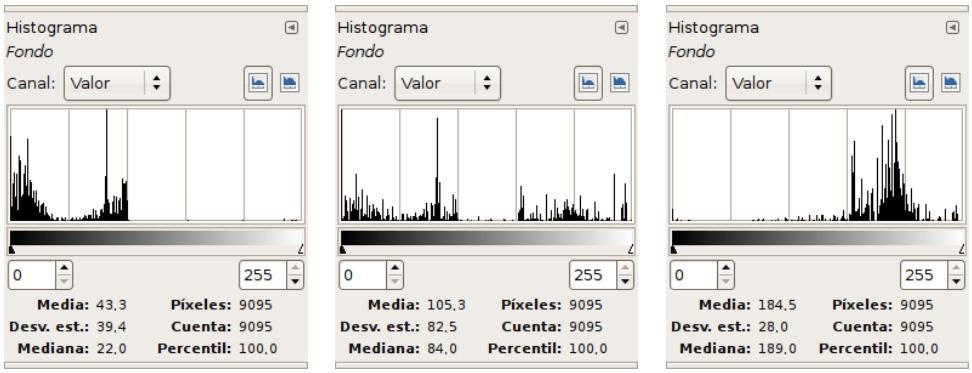


(a) Imagen original (b) Canal de tono (c) Máscara sobreimpresa

Figura 5.10: Gran incidencia de la luz en la segmentación.



(a) Muestra 1 (b) Muestra 2 (c) Muestra 3

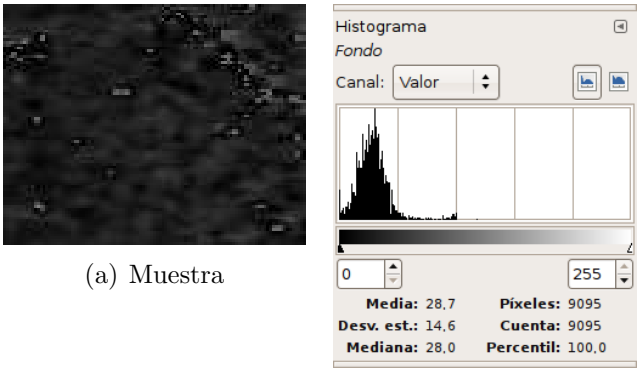


(d) Histograma de *(a)* (e) Histograma de *(b)* (f) Histograma de *(c)*

Figura 5.11: Variabilidad del tono en alta iluminación. (a), (b) y (c) son muestras del canal de tono en tres partes de una misma mano, (d), (e) y (f) son sus respectivos histogramas. En (a) y (c) los valores de tono son uniformes, (c) presenta el límite entre ellas.

(a) Imagen original (b) Canal de tono (c) Máscara sobreimpresa

Figura 5.12: Débil incidencia de la luz en la segmentación.



(a) Muestra

(b) Histograma de *(a)*

Figura 5.13: Variabilidad del tono con baja iluminación.

te. La dispersión del tono se mantiene más acotada ante una deficiencia, y no un exceso, de iluminación.

5.5 Efectividad en el reconocimiento

En los experimentos se aplicó el método de validación cruzada para la obtención de los resultados [31]. Para ellos se utilizó el software desarrollado en la modalidad *Comparación* descripta en la sección 4.3.

En una primer etapa y por cada sujeto, se realizaron 3 experimentos donde dos de sus conjuntos se usaron como *conocidos* y el restante como *prueba*, variando el conjunto de prueba. El desempeño obtenido fue del 95,16 % sobre todos los experimentos de todos los sujetos, con el detalle dado en la Tabla 5.2.

	Sujeto A	Sujeto B	Sujeto C
Experimento 1	100,00	87,75	100,00
Experimento 2	100,00	93,75	100,00
Experimento 3	87,50	100,00	87,50
Promedio	95,83	93,83	95,83

Tabla 5.2: Experimentos individuales por sujeto, con validación cruzada.

Del análisis de las identificaciones fallidas se encontró que la causa principal de los fallos se debe a un cambio en el ángulo de inclinación de la mano que desplaza las coordenadas del centroide, llevando en algunos casos a situaciones extremas donde se exceden las tolerancias asignadas en el clasificador.

Para comprobar la independencia de los resultados respecto de los individuos, se realizaron experimentos utilizando las tres realizaciones de dos sujetos para el entrenamiento y las tres realizaciones del restante para la prueba. Los detalles pueden verse en la Tabla 5.3.

Entrenamieno	Prueba	Desempeño
Sujeto B y Sujeto C	Sujeto A	93,75
Sujeto A y Sujeto C	Sujeto B	95,83
Sujeto A y Sujeto B	Sujeto C	85,42
Promedio		91,67

Tabla 5.3: Experimentos con validación cruzada de sujetos.

De los resultados de los experimentos se puede concluir que el método permite un reconocimiento independiente del sujeto de alrededor del 92 %.

Todos los signos han sido realizados con la mano derecha, sin embargo, ésto no implica que sea necesario un nuevo entrenamiento para utilizar la mano izquierda. Bastaría con agregar una opción que contemple la simetría del centroide.

5.6 Optimizaciones de los algoritmos

La implementación en un sistema en tiempo real requiere de la optimización de distintos aspectos que contribuyen a disminuir el tiempo total em-

pleado en el procesamiento de cada fotograma. Más allá de las optimizaciones propias del compilador g++, se trabajó sobre los siguientes aspectos técnicos y conceptuales.

5.6.1 Erosión y Dilatación

Las operaciones de erosión y dilatación de las máscaras durante la segmentación de la silueta de la mano son las que más tiempo consumen, sobre todo porque se utilizan máscaras de convolución de hasta 11×11 píxeles. Las primeras pruebas consistieron en comparar cuanto tiempo era necesario para una tarea en particular, utilizando codificaciones propias de los métodos versus funciones de una librería específica y bien depurada, como lo es OpenCV. El desempeño para las primeras fue muy pobre, requiriendo el doble del tiempo.

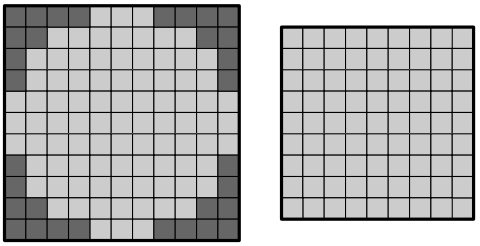
Al revisar la implementación se determinó que el problema radicaba en utilizar una máscara circular en lugar de una cuadrada, como lo hacían los métodos contra los que se comparó. Ésto tiene una importante incidencia que se detalla a continuación:

- La convolución con un kernel arbitrario implica que, dentro de la iteración de cada uno de los píxeles de la imagen, se analice el contenido de cada uno de los píxeles del kernel. Esto es $n \times n$ accesos a memoria en un kernel cuadrado de tamaño n.
- Cuando la máscara binaria es cuadrada hay dos ventajas. Por un lado no es necesario almacenar el kernel en memoria, sólo basta su dimensión. Por otro lado, no se hace necesaria su iteración sobre cada elemento, con la correspondiente pérdida de tiempo.

A partir de estas observaciones se procedió a implementar los cambios de máscaras redondas por cuadradas. En la Fig. 5.14 se presenta un ejemplo del cambio realizado.

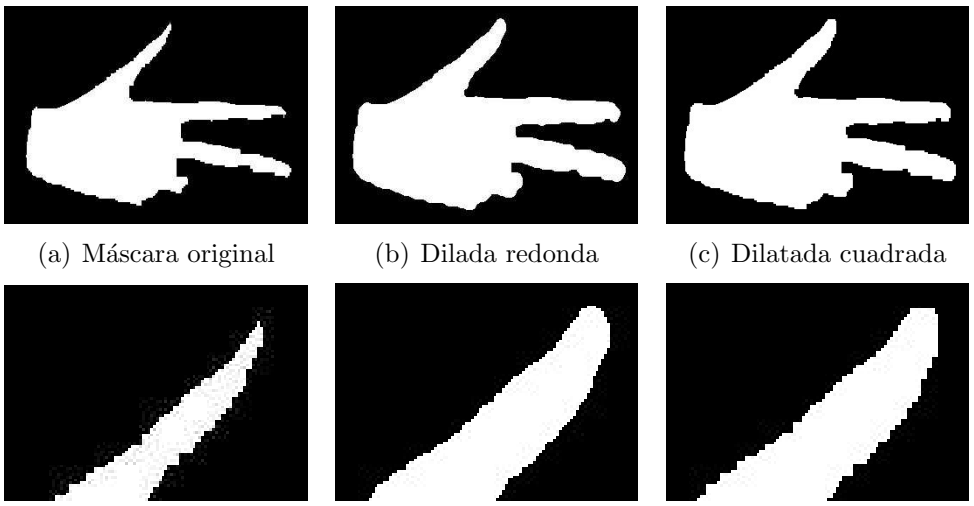
La razón, por la que al principio se decidió utilizar una máscara redonda, fue obtener siluetas con bordes suaves antes de extraer las características de la silueta de la mano (Fig. 5.15). Esta cualidad pareció importante al principio, pero no aportó ventajas a los métodos desarrollados en este proyeto.

Otro punto importante que merece atención es la implementación de las operaciones de dilatación y erosión en los bordes de la imagen original. A medida que el centro del kernel se aproxima al borde, algunas filas y/o columnas de éste se localizan fuera del plano de la imagen. Entre las distintas



(a) (b)

Figura 5.14: Máscaras redondas vs. cuadradas que produce resultados similares. ☐ Pixel en 1, ■ Pixel en 0. (a) Máscara redonda de 11×11. (b) Máscara cuadrada de 9×9.



(a) Máscara original (b) Dilada redonda (c) Dilatada cuadrada

Figura 5.15: Comparación de máscaras para Dilatación. (a) Máscara original, (b) Dilatación utilizando la máscara redonda 5.14(a), (c) Dilatación utilizando la máscara cuadrada 5.14(b).

maneras de resolver esta cuestión, se optó por limitar el área de aplicación al centro de la imagen, a una distancia no menor que $(n-1)/2$ píxeles del borde. Los píxeles que no son procesados son puestos a cero. Con ésto se ahorra sobrecargar las funciones con expresiones condicionales y aumentar la velocidad final sin perder precisión.

Con los cambios propuestos se logró una reducción del tiempo de cálculo original a la mitad. Así el desempeño se equiparó al obtenido por OpenCV.

5.6.2 Conversión RGB a HSV

La máscara de tono requiere la conversión del espacio de color RGB al HSV. En un principio se utilizó una función propia de la clase CImg para esta tarea. Luego, para disminuir la sobrecarga en la segmentación, se implementó una función específica para el cálculo de la componente Tono. Ésta inmediatamente crea su máscara de tono asociada, de acuerdo a sus parámetros tono de referencia y umbral de tono. De esta manera se evita la escritura en un buffer que sólo almacena dicho canal (y no se volverá a usar) y se agrupa en una sóla etapa la diferencia respecto del tono de referencia.

5.6.3 Gestión de Memoria

Para mejorar la velocidad de respuesta de los algoritmos se utilizaron estructuras de datos creadas una sola vez, en el constructor del objeto, y son reutilizadas incluso para distintos propósitos. De esta manera se evitan las alocaciones de memoria repetitivas y se gana en localidad espacial por el acceso reiterado a los mismos segmentos de direcciones.

Por utilizar funciones que manipulan datos globales dentro de su clase, la codificación debe hacerse con más atención, puesto que se presupone el contenido de cada una de las estructuras. Este incremento en la complejidad se refleja en una disminución de los tiempos de procesamiento.

5.6.4 Modelo de fondo

En el algoritmo se considera que el fondo es constante. Sin embargo esto no siempre es asi. Si el usuario mueve un objeto de la escena es probable que aparezca también la mano. Esto está generalmente resuelto por la separación

planteada por la segmentación en color, pero influye sobre la validación del fotograma. También pueden darse cambios por otros factores: si la iluminación del lugar está dada por efecto natural (luz entrando desde una ventana), ésta cambiará mientras transcurren las horas. Para aplicaciones de exterior como pueden ser sistemas de vigilancia o seguridad, se debe tener en cuenta una herramienta comocida como "modelo de fondo" que permita adaptar gradualmente el fondo de referencia para poder realizar una correcta separación entre frame actual y fondo. Este proyecto, que busca una utilización en lugar con iluminación controlada (mas o menos constante) no incluye modelos de fondo, para su uso basta realizar un "refresco" del fondo cuando se deciden cambiar el fondo o las condiciones de iluminación de la escena.

5.6.5 Región de interés

El campo visual de la cámara no siempre se utiliza completamente para realizar los signos. Entonces, no se justifica procesar toda la imagen cuando se conoce de antemano qué partes no contendrán información de interés. También, puede ser de interés omitir aquellas zonas de la imagen que compliquen el reconocimiento, por ejemplo zonas donde hay continuas variaciones de la escena u objetos con el mismo tono de la mano. Por éstos motivos se decidió conveniente brindar al usuario la opción para definir una "Región de interés". Una ROI consiste en una región rectangular definida en el interior del área del fotograma de la cámara. Para el procesamiento sólo se considerará la información contenida dentro de este área. La implementación de una ROI no sólo hace un aporte al concentrar esfuerzos en un área útil, sino que mejora el rendimiento por disminuir la cantidad de píxeles que necesitan ser procesados. Así por ejemplo, una disminución en píxeles de Δw, Δh en el ancho y alto respectivamente (ver Fig. 5.16), corresponde a una reducción total R en los píxeles del fotograma, dada por la siguiente fórmula:

$$R = h\Delta w + w\Delta h - \Delta w\Delta h, \tag{5.1}$$

donde $w =$ es el ancho y $h =$ es el alto del fotograma original.

Si medimos la reducción de los píxeles en relación al tamaño de la imagen original, se tiene:

$$R^* = \frac{R}{w \cdot h} = \frac{\Delta w}{w} + \frac{\Delta h}{h} - \frac{\Delta w}{w}\frac{\Delta h}{h} \tag{5.2}$$

$$= \alpha_w + \alpha_h - \alpha_w\alpha_h \tag{5.3}$$

Por ejemplo, para una reducción relativa de un cuarto en el ancho y alto del fotograma ($\alpha_w = \alpha_h = 1/4$) se tiene $R^* = \frac{1}{4} + \frac{1}{4} - \frac{1}{4} \cdot \frac{1}{4} = \frac{7}{16}$, o sea, una reducción del 43,8 %. La Fig. 5.16 muestra este resultado.

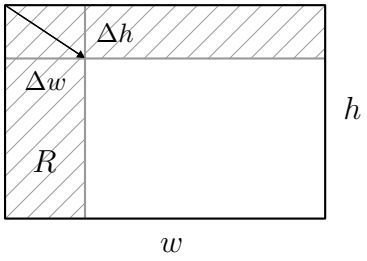


Figura 5.16: Ejemplo de una reducción R en el tamaño de una imagen, con $\Delta w = 0{,}25w$ y $\Delta h = 0{,}25h$.

5.7 Velocidad de ejecución

Un requerimiento fundamental para la aplicabilidad del método de reconocimiento en un sistema de tiempo real, es que el usuario perciba la respuesta a sus acciones de la manera más inmediata posible. Para este proyecto, donde el hardware utilizado no es especializado, se estableció a priori que sería deseable poder reconocer tres fotogramas por segundo. Para conocer el tiempo insumido por la etapa de reconocimiento se realizaron pruebas donde se consideró:

¿Qué se mide? Se mide el tiempo del reconocimiento. Para ello se utiliza un caso de prueba complejo que incluya: la extracción de la silueta de una imagen de 640×480 píxeles, la selección de la región correcta entre varias candidatas, conteo de cinco dedos, y el etiquetado del signo después de clasificarlo sobre el conjunto completo de 144 patrones.

¿Qué no se mide? Se deja fuera de la medición el tiempo de la carga de las imágenes de mano y fondo, la configuración de los parámetros del reconocedor, y la carga de los patrones conocidos.

¿Con qué se mide? La medición se realiza mediante la inclusión de funciones especiales justo antes y después de llamar a la rutina *reconocer()* de la clase *ReconocedorMano*. Para ésto se utilizó la función *clock()*

provista por la biblioteca estándar de C++ en su archivo de cabecera `<ctime>`. Se debe notar que esta función cuenta tanto el tiempo del proceso como el utilizado por el sistema.

¿Cómo se mide? El tiempo insumido en el reconocimiento se calcula como el promedio de las mediciones de reiteradas pruebas. Para este propósito se utiliza el modo *Reconocimiento* del software manoviva que, en su procesamiento local (imágenes en archivos), informa al usuario del tiempo insumido según las pautas descriptas anteriormente.

Para tener una idea completa se hicieron pruebas en dos sistemas operativos distintos: Ubuntu GNU/Linux 8.04 y MS Windows XP; con y sin carga de aplicaciones multimedia en los mismos. El hardware utilizado cuenta con un microprocesador AMD Athlon XP 2800+ (1000 MHz, caché 512 KB), 1024MB de RAM, placa madre ASUS K8V-MX (chipset VIA) y la cámara web seleccionada Genius Slim 1322AF. Los resultados de la velocidad de ejecución se presentan en la Tabla 5.4, medidos en fotogramas por segundo (fps), junto al tiempo insumido en milisegundos (ms) equivalente.

Carga del sistema	Ubuntu 8.04	Windows XP
Baja	168,33ms / 5,94fps	117,00ms / 8,55fps
Alta	181,67ms / 5,50fps	140,05ms / 7,14fps

Tabla 5.4: Velocidad de ejecución en el reconocimiento.

Como se puede observar, el sistema operativo Windows XP obtiene una mejor velocidad en la ejecución. Este comportamiento o sensación de respuesta más lenta se comprobó en otras aplicaciones multiplataforma, lo que indica que la diferencia en el rendimiento no es un asunto propio de la implementación de este software.

CAPÍTULO 6

Tutor de signos

Como complemento y más allá de los objetivos propuestos para el presente Proyecto Final de Carrera, se implementó una aplicación con entorno gráfico específico que ejemplifica una aplicación real que incluye todo lo desarrollado. Se propuso un software tutor cuyo objetivo es brindar una tutoría interactiva y el aprendizaje didáctico de los signos [32].

Las aplicaciones con GUI (del Inglés *Graphical User Interface* – Interfaz Gráfica de Usuario) son manejadas por eventos. Esto significa que la aplicación reside en un bucle a la espera de eventos que provengan del usuario o de alguna otra fuente (como el aviso de redibujado de la pantalla o la conexión de un socket), y entonces distribuye el evento a la función apropiada que lo maneje. Este mecanismo, como se comentará más adelante, cambia conceptualmente la forma en que se debe escribir el código respecto de una aplicación de consola.

La motivación por este software surge de dos consideraciones. Por un lado, el interés particular de aprender una nueva biblioteca libre que permitiese construir interfaces gráficas para distintas plataformas. Mientras que el otro aspecto fue determinar la viabilidad de incorporar, a la propia gestión de mensajes de la GUI, toda la carga del procesamiento del video. Ésto sin degradar la interacción con los componentes de la interfaz grafica y manteniendo una respuesta rápida a las acciones del sujeto que realiza los signos delante de la cámara.

6.1 Materiales

Todas las bibliotecas utilizadas para su implementación son libres y permiten que el software sea multiplataforma[1]. A las herramientas de software utilizadas en el desarrollo del prototipo manovia del Capítulo 4 (*CImg*, *OpenCV* y *CodeLite*) se agregan las siguientes:

wxWidget [33]**:** Brinda un API para crear una GUI utilizando los controles nativos de la plataforma subyacente en cada caso. Caracterísiticas: multiplataforma, licencia GLPv2 (salvo wxWindows que tiene prohibida su modificación), utilizado por muchas aplicaciones (xCHM, bacula, aresgalaxy, filezilla entre otras), muy buena documentacion y gran cantidad de ejemplos [34].

wxFormBuilder [35]**:** Es un constructor de interfaces gráficas de usuario desarrollado *en* y *para* wxWidget. Características: multiplataforma, de código abierto, genera código en C++ y en XRC (archivos XML con la especificación de los elementos que se cargan dinámicamente y permiten separar la interfaz de usuario del código funcional), simple e intuitivo de utilizar.

6.2 Consideraciones en el diseño

Esta aplicación es fruto de un desarrollo basado en la reutilización de componentes existentes. En este caso, los componentes son las clases de la sección 4.2 y recursos especiales que provee wxWidget para facilitar algunas tareas al programador (como cuadros de diálogos estándar y administrador de archivos de configuración, entre otros).

Aprovechando las ventajas que provee una GUI para seleccionar colores y opciones, se decidió hacer algunas modificaciones menores en los objetos encargados del procesamiento para que incluyan una mayor personalización. Ésto se aprecia en las descripciones de la interfaz realizadas más adelante.

[1]Se comprobó que la compilación sobre MS Windows XP es directa pero, por diferencias en la implementación de los métodos de dibujo en pantalla, fueron necesarios algunos cambios específicos para su correcto funcionamiento sobre la plataforma WIN32.

El mecanismo de gestión de eventos de una aplicación con GUI cambia completamente la forma en que se escribe el código respecto de una aplicación de consola. El bucle del procesamiento de video (Fig. 3.17) no puede implementarse directamente puesto que dificultaría la gestión de los eventos, la interfaz no se actualizaría correctamente y por tanto el usuario percibiría que la aplicación tiene un mal funcionamiento o en el peor de los casos está fuera de servicio. En este nuevo escenario la solución diseñada presenta las siguientes características:

- Se crea un hilo de ejecución secundario para extraer los fotogramas desde la cámara web a intervalos regulares de $Q = 1/fps$ milisegundos. Para cada fotograma se envía un evento al gestor de la aplicación y se hace una espera de Q ms.
- El evento de un nuevo fotograma se procesa por una función especial, que realiza las operaciones necesarias según el estado de la aplicación.
- El fotograma es almacenado en una variable compartida por el hilo principal y el hilo de la cámara, por lo tanto, se la debe proteger mediante un *mutex*[2] para evitar que los datos queden corruptos por acciones concurrentes.
- El procesamiento de un fotograma puede durar más tiempo de lo que tarda el hilo de la cámara en hacer una nueva extracción desde el flujo de video. Para evitar la acumulación de eventos mientras se procesa el fotogramas actual, se utiliza el propio mutex como bandera dentro del hilo de la cámara. Mientras el mutex esté bloqueado no se debe extraer ningún fotograma, pues el anterior está siendo procesado, por lo tanto se hace una espera de Q ms.

En la Fig. 6.1 se presenta cómo son las relaciones entre los distintos elementos para que el procesamiento de video sea posible en una aplicación con interfaz gráfica. El *FramePrincipal* procesa los fotogramas y los eventos del usuario. Un objeto *mutex* protege la variable compartida que almacena el fotograma. El hilo (thread) secundario *HiloDeLaCamara* obtiene nuevos fotogramas desde la cámara a intervalos regulares y avisa al frame principal por medio de un evento. El método *procesar()* incluye las etapas *Validación*, *Reconocimiento*, *Estabilización* y *Utilización*.

[2]Acrónimo de *MUTually EXclusive*. Un mutex es un objeto de sincronización cuyo estado indica que los datos adquiridos por un hilo no pueden ser compartidos con otro.

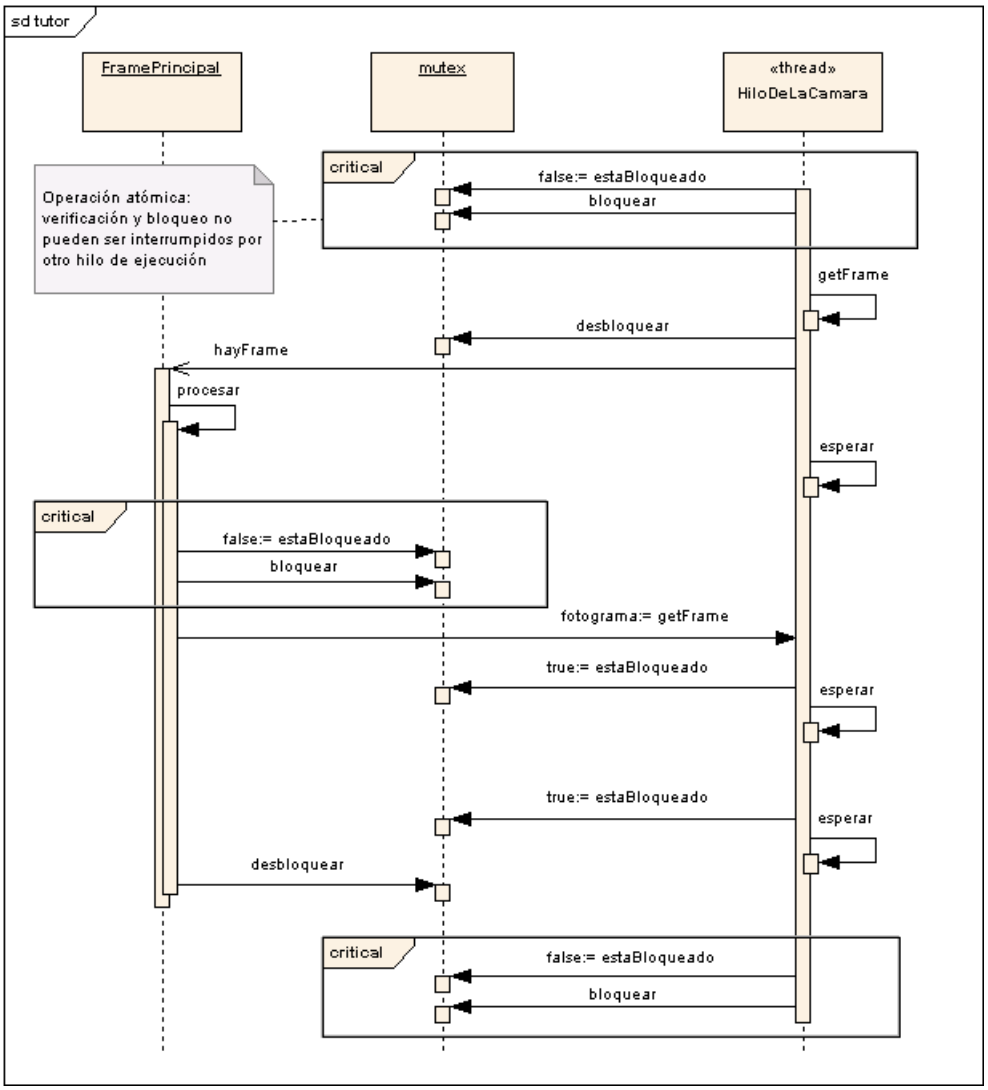
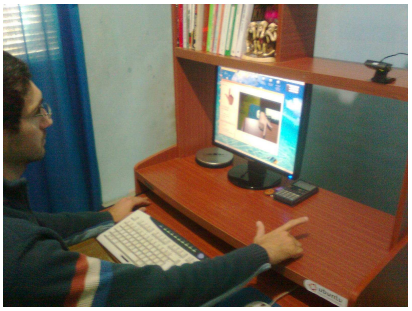


Figura 6.1: Esquema de procesamiento en la aplicación con GUI.

(a) Cámara al frente



(b) Cámara arriba

(c) Detalle de *(a)*

(d) Detalle de *(b)*

Figura 6.2: Posición de la cámara respecto del usuario. (a) Al frente, (b) Arriba (posición elegida para el Tutor) (d) y (e) Detalle de la posición de la cámara web.

6.3 Localización física de la cámara

La posición de la cámara respecto del usuario es un detalle importante a tener en cuenta. Si la misma se coloca al frente, el antebrazo debe estar en posición vertical para la realización de los signos (Fig. 6.2(a)). Esta posición resulta incómoda en pocos minutos, sintiéndose la fatiga en el codo y/o en el hombro. Para lograr una situación más confortable se optó por ubicar la cámara desde el frente, por sobre el plano de ejecución (horizontal en este caso), y enfocada hacia abajo (Fig. 6.2(b)). De este modo se logra una posición más natural para la realización del signo, con mínimo esfuerzo, y permite que el usuario apoye parte de su brazo para descanzar del peso.

Como el reconocimiento de los fotogramas se realiza de acuerdo a la posición específica de la cámara respecto de la mano, el entrenamiento del sistema debe realizarse bajo la misma configuración que la de utilización. Para el conjunto de signos propuesto en la sección 5.3 se tuvo en cuenta este aspecto, de manera tal de no necesitar nuevos entrenamientos para la implementación del

(a) Desde el usuario (b) Desde la cámara

Figura 6.3: Visualización del signo desde el usuario y la cámara.

tutor. Como se puede observar en la Fig. 6.3, los signos son visualizados por la cámara de manera rotada a como lo hace el usuario. El usuario ve los dedos sobre el margen izquierdo del signo (Fig. 6.3(a)), mientras que la cámara ve el signo rotado 180° respecto del usuario, con los dedos sobre el margen derecho (Fig. 6.3(b)). Por este motivo se incluyó la opción de visualizar los signos desde los puntos de vista de la cámara y del usuario (Fig. 6.5(c)). Ésto ayuda a comunicar al usuario la forma correcta de realizar el signo y sobre todo no confundirlo, si éste es un niño pequeño.

6.4 Partes principales de la interfaz

En la Fig. 6.4 se muestra al tutor en operación y se destacan las tres partes principales que conforman la interfáz con el usuario:

- *Amarillo*: listado de signos disponibles.
- *Verde*: mensajes del tutor hacia al usuario.
- *Azul*: región que visualiza lo que la webcam está enfocando.

Su funcionamiento es simple, consiste en seleccionar un signo del listado, incluído en el programa, y realizarlo frente a la cámara. El tutor indica en pantalla si lo está haciendo correctamente o no.

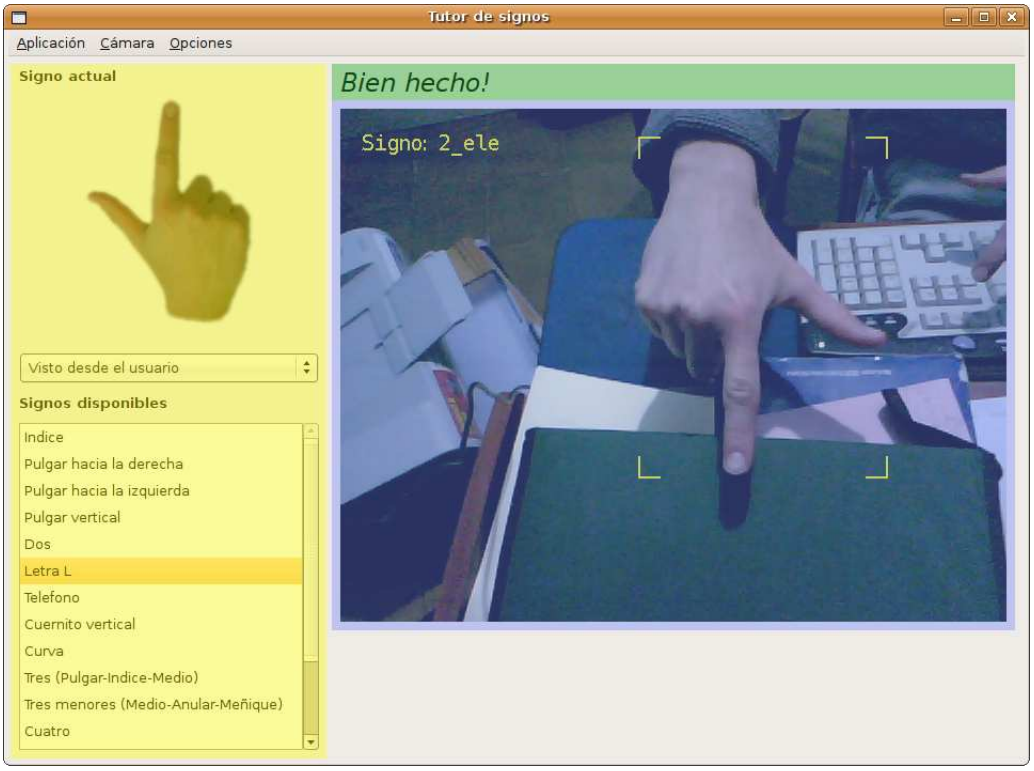
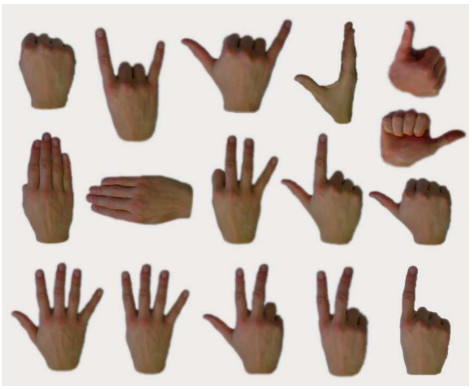


Figura 6.4: Partes de la interfaz gráfica del tutor.

(a) Panel de signos



(b) Signos disponibles

(c) Opciones de visualización

Figura 6.5: Signos disponibles

6.4.1 Signos disponibles

Para esta aplicación se dispuso de la base de datos conformada por las realizaciones de 16 signos distintos (ver sección 5.3. En cualquier momento el usuario puede seleccionar un signo del panel lateral, que cuenta con una imagen de referencia, su descripción textual y la posibilidad de visualizar el signo desde la perspectiva del usuario o desde la cámara según la comodidad del usuario. El detalle del panel se puede ver en la Fig. 6.5(a).

6.4.2 Mensajes del tutor

Cuando el tutor recibe información de que se ha reconocido un signo, éste verifica, que sea el mismo signo que el elegido en el panel de signos

disponibles. Si coinciden informa mostrando el texto *Bien hecho!* sobre el área de mensajes, si son distintos comunica al usuario *No está haciendo el signo correcto*. Estos casos se pueden observar en la Fig. 6.6. Mientras no se detecte signo alguno aparecerá el mensaje *Realice un signo*.

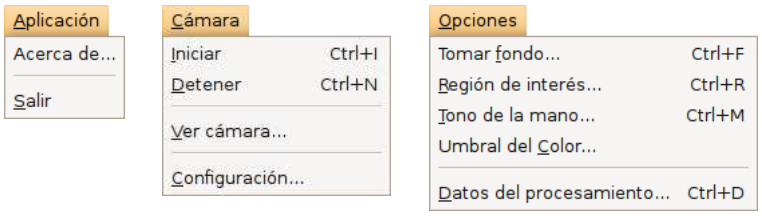
Como se vió en la sección 3.1.2, para el reconocimiento es necesario que se defina un fondo de referencia. Por lo tanto, mientras no se haya configurado ésto, el tutor visualizará el video sin procesarlo e informará al usuario con el mensaje *Debe elegir primero una imagen del fondo*. Los umbrales de color, tono y tono de referencia vienen predefinidos de manera que, aun sin estar calibrados previamente durante el uso del tutor, permiten la ejecución del reconocedor.

6.4.3 Visualización

Además de visualizar el video de la cámara web es posible dibujar sobre cada fotograma. La visualización por defecto incluye la etiqueta o nombre del signo, los marcadores de las esquinas que ubican la zona rectangular ocupada por el signo, y la ROI si ha sido definida.

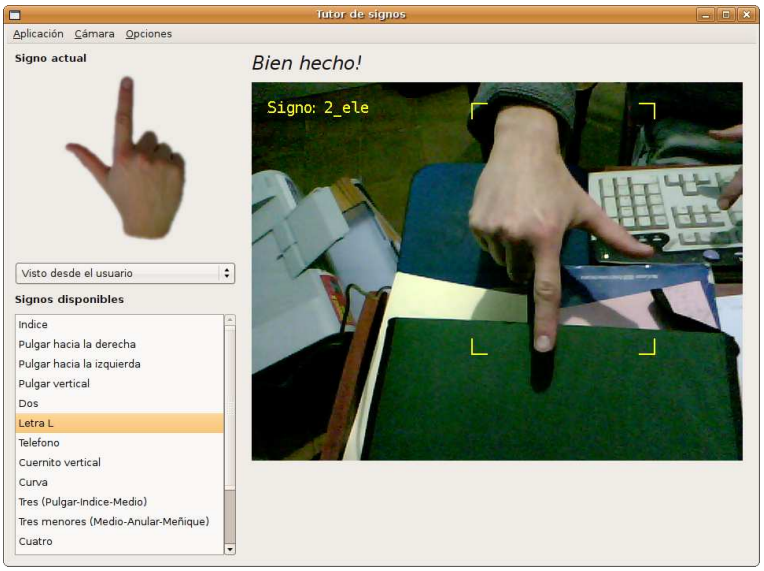
6.5 Configuración y calibración

A través del menú principal se accede a las opciones de configuración de la cámara y calibración de los parámetros del reconocedor. Sus elementos son los siguientes:

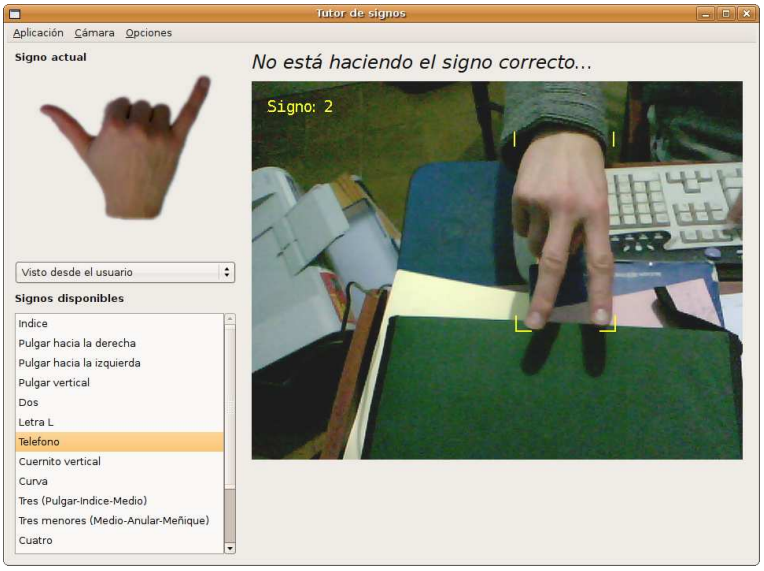


Menú Aplicación

- Acerca de: Muestra información sobre la aplicación y el desarrollador.
- Salir: Detiene el procesamiento de video, guarda la configuración actual y termina la ejecución del tutor.



(a) Ejecución correcta



(b) Ejecución incorrecta

Figura 6.6: Mensajes en la ejecución del signo elegido.

Figura 6.7: Configuración de la cámara web.

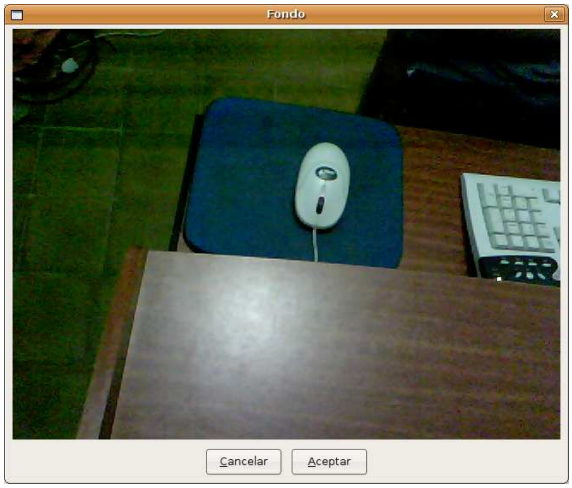


Figura 6.8: Selección del fondo de referencia.

Menú Cámara

- Iniciar: Inicia la captura de video desde la cámara web con los parámetros establecidos en su configuración.
- Detener: Detiene la captura de video.
- Ver cámara: Visualiza en una ventana aparte el video sin datos adicionales. Es una vista adicional en casos donde el visor interno muestra muchos datos adicionales.
- Configuración: Permite configurar la resolución de la cámara (640×480 y 352×288 píxeles) y la velocidad (1 a 30 fps) con la que se extraen los fotogramas desde la cámara). Ver Fig. 6.7.

Menú Opciones

- Tomar fondo: Muestra el fotograma actual y pregunta al usuario si desea o no establecerla como fondo de referencia. Ver Fig. 6.8.
- Región de interés: Permite definir, mover o eliminar una región rectangular que representa la región de interés del procesamiento. Se utiliza clic y arrastre para definir una nueva región o mover (si el primer clic fue dentro de área previamente definida). Ver Fig. 6.9.

Figura 6.9: Selección de la región de interés. Metodología: clic, arrastrar y soltar. Clic sobre la ROI definida permite mover su posición.

- Tono de la mano: Visualiza el fotograma actual, sobre éste el usuario debe seleccionar un área con la que se realizará el cálculo del tono de referencia y el umbral de tono. Al definir o mover la región seleccionada a otra posición, se presentan los valores actuales de media, desvío y mediana. Además se grafica el histograma del canal de tono (desplazado para ubicar el tono de referencia en su centro), los valores dentro del umbral de tono aparecen en blanco y los que lo sobrepasan aparecen en gris. En caso de determinar que las condiciones de la región no son buenas, se mostrará la leyenda *Región no uniforme, seleccione otra*. Ver Figs. 6.10 y 6.11.
- Umbral de color: Este parámetro indica un límite mínimo para la diferencia, por lo que a mayor valor del umbral menos diferencia será detectada. Normalmente no es necesario cambiarlo (se permite su modificación con propósitos experimentales) puesto que está establecido por defecto en un valor aplicable a variadas situaciones. Ver Fig. 6.12.
- Datos del procesamiento: Permite configurar que información mostrar, generada por los objetos involucrados en el procesamiento del video. De esta manera se puede conocer que está sucediendo internamente, estudiarlo y perfeccionarlo. El *Graficador* presenta datos en pantalla sobre el fotograma actual, que se muestra en la zona de visualización de la cámara web. El *Validador*, *Reconocedor* y *Estabilizador* envían datos a la consola. Ver Figs. 6.13 y 6.14.

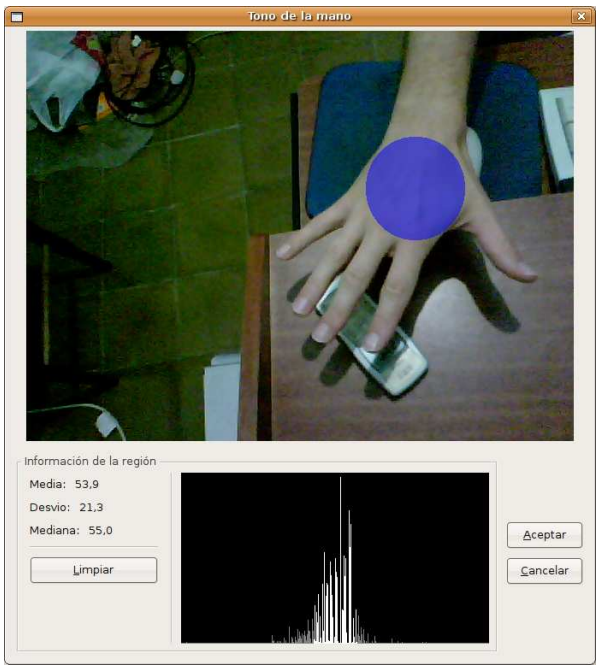


Figura 6.10: Calibración correcta del tono de referencia y umbral de tono.

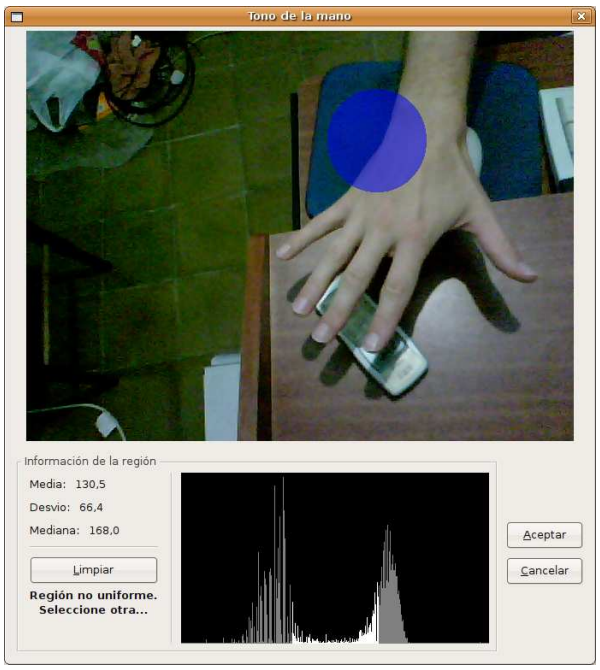


Figura 6.11: Calibración incorrecta del tono de referencia y umbral de tono.

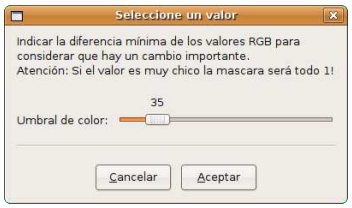


Figura 6.12: Calibración del umbral de color.

(a) Presentación (b) Datos en general

Figura 6.13: Datos del procesamiento en pantalla.

(a) Desactivados (b) Activados

Figura 6.14: Datos del procesamiento en consola. (a) Supresión global de todos los avisos, (b) Selección individual de datos.

Figura 6.15: Archivos del subdirectorio practica/, esenciales para operar el tutor de signos.

6.6 Personalización

El sistema se diseñó para ser flexible y personalizable. Los signos y los parámetros se pueden cambiar sin necesidad de volver a compilar el código fuente.

Dentro del subdirectorio practica/ se encuentran los archivos esenciales del tutor (Fig. 6.15):

- El listado de los signos disponibles que aparecen en el panel de signos, se ubica dentro del archivo listado.txt. Por cada línea se coloca una descripción larga y la etiqueta del signo, separados por un punto y coma.
- La imagen con la silueta de cada signo debe estar en un archivo independiente, cuyo nombre esté formado por la etiqueta que figura dentro del archivo listado.txt ([etiqueta_signo].png).
- La base de datos de patrones conocidos por el reconocedor se carga desde el archivo patrones.txt.

Por otro lado, el tutor utiliza un archivo config.ini con la configuración de los datos propios del uso del sistema. Este archivo es de texto plano, donde sus líneas tienen la forma `clave=valor` y se agrupan dentro de otras líneas de la forma `[grupo]`. Mediante esta estructura se almacena la configuración de la cámara, parámetros del *Validador*, colores que utiliza el *Graficador* y los

valores de umbral de color, umbral de tono, y tono de referencia del *Reconocedor*. Todos los valores son guardados automáticamente al cerrar la aplicación. Si existe este archivo al inicio de la ejecución, será creado automáticamente con sus valores por defecto.

6.7 El tutor en otra plataforma

Aquí se presentan imágenes (Figs. 6.16, 6.17 y 6.18) de algunas de las ventanas de la interfáz gráfica del tutor trabajando sobre Microsoft Windows XP. Se pueden apreciar las similitudes con las interfaces anteriores correspondientes al entorno GNOME. El resultado final depende de la experiencia que tenga el programador en el uso de los "sizers" de wxWidget. Esto es, objetos para ubicar subventanas dentro de ventanas a través de un algoritmo de distribución similar a los incluídos en otros toolkits como el AWT de Java, el GTK o el Qt.

Figura 6.16: Tutor en Windows.

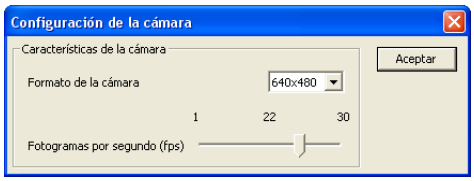


Figura 6.17: Configuración de la cámara web en Windows.

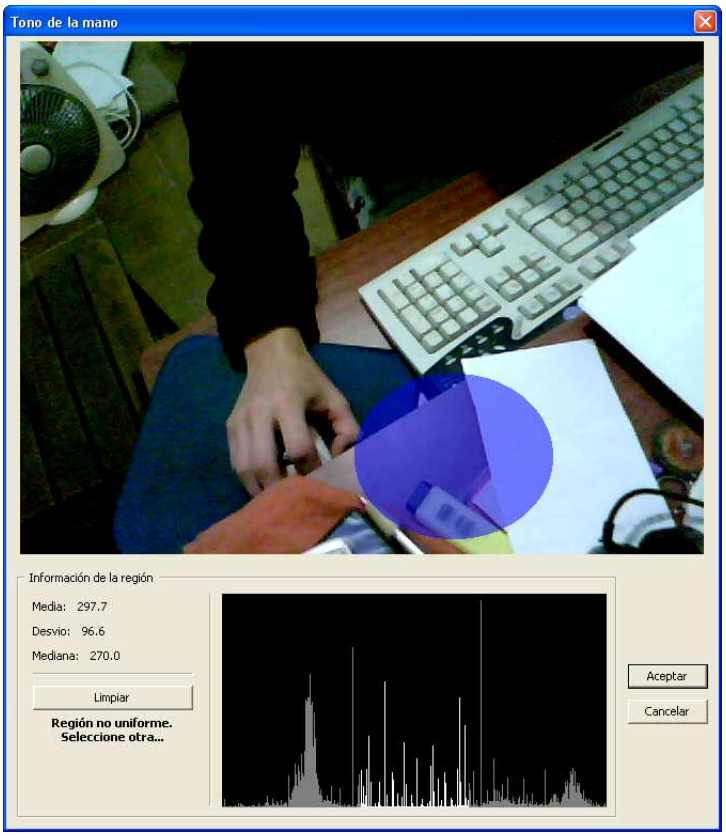


Figura 6.18: Calibración del tono de referencia y umbral de tono en Windows.

CAPÍTULO 7

Conclusiones y desarrollos futuros

En este trabajo se presenta el desarrollo de un método integral para detección de signos manuales, basado en un enfoque de procesamiento y análisis.

El sistema obtiene fotogramas de un flujo de video de una cámara de bajas prestaciones, y aplica ténicas de procesamiento de imágenes para segmentar la mano del resto de los objetos (brazo y fondo) y obtener un patrón formado por un conjunto de tres características medidas sobre la silueta de la mano. Un clasificador asocia el patrón a una clase, la cual identifica el signo realizado por el usuario. El procesamiento individual de cada fotograma está relacionado a su pasado cercano, condicionando el resultado final.

Se observó que el factor más influyente sobre el desempeño del reconocedor es la correcta separación de la silueta de la mano en la imagen, por lo que se puso especial énfasis en este aspecto. Este trabajo, a diferencia de otros similares, impone un menor número de restricciones sobre el entorno y las condiciones de utilización del sistema.

Los métodos utilizados no revisten una gran complejidad computacional y el tiempo de proceso ha sido reducido considerablemente desde las primeras implementaciones a través de las optimizaciones propuestas, superando el requerimiento de velocidad mínima incluso en el sistema operativo que presentó el más bajo desempeño.

En términos generales, el sistema presenta un rendimiento aceptable en función de las tasas de reconocimiento alcanzadas, aún en condiciones de iluminación sub-óptimas. A diferencia de lo esperado, la iluminación blanca

provista por una lámpara de bajo consumo sobre la zona de ejecución resultó contraproducente para la segmentación. El mejor resultado se obtuvo con el aporte indirecto de bombillas incandescentes comunes.

Para el sistema orientado a un usuario final, se logró una forma simple y práctica de determinar los detalles de calibración de los parámetros más criticos.

Se puede mencionar que una extensión natural de las funcionalidades de este sistema estaría dada por el reconocimiento de gestos dinámicos (gesticulaciones) en el flujo de video, donde cada gesto se compone de un movimiento completo de la mano. El empleo de gestos dinámicos posibilita aumentar el número de comandos que pueden ser reconocidos, establecer variantes para cada gesto particular, incrementando las prestaciones del sistema.

Respecto al clasificador, otros métodos más robustos de reconocimiento de patrones podrían ser necesarios para escalar el sistema, por ejemplo redes neuronales del tipo Perceptrón multicapa, que permitirían incorporar características no utilizadas en este proyecto.

Bibliografía

[1] A. J. Dix, J. Finlay, G. Abowd, and R. Beale. *Human-Computer Interaction*. Prentice Hall, New Jersey, 2003.

[2] S. Mitra and T. Acharya. Gesture Recognition: A Survey. *IEEE Trans on Systems, Man and Cybernetics, Part C: Applications and Reviews*, 37(3):311–324, 2007.

[3] A. Erol, G. Bebis, M. Ñicolescu, R. Boyle, and X. Twombly. Vision-based hand pose estimation: A review. *Computer Vision and Image Understanding*, 108:52–73, 2007.

[4] D. Bowman. Principles for the design of performance-oriented interaction techniques. In K. M. Stanney, editor, *Handbook of Virtual Environments: Design, Implementation and Applications*, pages 201–207. Lawrence Erlbaum Associates, Hillsdale, NJ, 2002.

[5] A. Liu, F. Tendick, K. Cleary, and C. Kaufmann. A survey of surgical simulation: applications, technology, and education. *Presence: Teleoperators and Virtual Environments*, 12(6):599–614, 2003.

[6] C. von Hardenberg and F. Bérard. Bare-hand human-computer interaction. In *PUI '01: Proceedings of the 2001 workshop on Perceptive user interfaces*, pages 1–8, New York, NY, USA, 2001. ACM.

[7] L. Tarabella. Handel, a *Free-Hands* Gesture Recognition System. In *Computer Music Modeling and Retrieval*, volume 3310 of *Lecture Notes in Computer Science*, pages 139–148. Springer, 2005.

[8] J. Weissmann and R. Salomon. Gesture recognition for virtual reality applications using datagloves and neural networks. In *International Joint Conference on Neural Networks*, pages 2043–2046. IEEE, 1999.

[9] R. G. O'Hagan, A. Zelinsky, and S. Rougeaux. Visual gesture interfaces for virtual environments. *Interacting with Computers*, 14:231–250, 2002.

[10] Ying Wu and Thomas S. Huang. Vision-Based Gesture Recognition: A Review. In *Gesture-Based Communication in Human-Computer Interaction*, pages 103–115. Springer Berlin, 1999.

[11] Thomas B. Moeslund and Erik Granum. A Survey of Computer Vision-Based Human Motion Capture. *Computer Vision and Image Understanding*, 81(3):231–268, 2001.

[12] L. Gupta and S. Ma. Gesture-based interaction and communication: Automated Classification of Hand Gesture Contours. *IEEE Trans on Systems, Man, and Cybernetics, Part C*, 31(1):114–120, February 2001.

[13] Ekaterini Stergiopoulou and Nikos Papamarkos. A new technique for hand gesture recognition. In *Proceedings of the International Conference on Image Processing (ICIP)*, pages 2657–2660. IEEE, 2006.

[14] Xiaoming Yin and Ming Xie. Hand gesture segmentation, recognition and application. In *Proceedings of the International Symposium on Computational Intelligence in Robotics and Automation*, pages 438–443. IEEE, 2001.

[15] M. Ashraful Amin and Hong Yan. Sign Language Finger Alphabet Recognition from Gabor-PCA Representation of Hand Gestures. In *Proceedings of the International Symposium on Computational Intelligence in Robotics and Automation*. IEEE, 2007.

[16] Ben W. Miners, Otman A. Basir, and Mohamed S. Kamel. Understanding hand gestures using approximate graph matching. *IEEE Transactions on Systems, Man, and Cybernetics, Part A*, 35(2):239–248, 2005.

[17] R. González and R. Woods. *Digital Image Processing*. Prentice Hall, 2002.

[18] B. Jähne. *Digital Image Processing*. Springer-Verlag, 2002.

[19] A. R. Smith. *The HSV-RGB Transform Pair*. Website, 2009. `http://www.alvyray.com/Papers/hsv2rgb.htm`.

[20] R. Duda and P. Hart. Use of the Hough Transformation to detect lines and curves in pictures. *Comm. ACM*, 15(1):11–15, June 1972.

[21] A. Desolneux, L. Moisan, and J.-M. Morel. *From Gestalt Theory to Image Analysis, A Probabilistic Approach*. 2006.

[22] A. Konar. *Artificial Intelligence and Soft Computing: Behavioral and Cognitive Modeling of the Human Brain*. CRC Press LLC, 2000.

[23] R. J. Henery. In *Machine Learning, Neural and Statistical Clasification*, pages 8–16. Editors: D. Mitchie and D.J. Spiegelhalter and C.C. Taylor, 1994.

[24] J. P. Hernández Vogt, R. J. Godoy, and P. Novara. Reconocimiento de señas de una mano. *37º JAIIO, en categoría Trabajo de Cátedra*, Septiembre 2008.

[25] I. Sommerville. *Ingeniería de software*. Addison-Wesley, 6th edition, 2002.

[26] G. Booch, J. Rumbaugh, and I. Jacobson. *El Lenguaje Unificado de Modelado*. Addison-Wesley Logman, 1999.

[27] D. Tschumperlé. *The CImg Library. C++ Template Image Processing Toolkit.* Version 1.30. Website, 2008. `http://cimg.sourceforge.net`.

[28] G. Bradski and Others. *OpenCV.* Version 1.1.0. Website, 2009. `http://sourceforge.net/projects/opencvlibrary/`.

[29] G. Bradski and A. Kaehler. *Learning OpenCV*. O'Really Media Inc., 2008.

[30] E. Ifrah. *CodeLite.* Version 2.1.0. Website, 2009. `http://www.codelite.org`.

[31] R. Duda, P. Hart, and D. Stork. *Pattern classification.* Wiley-Interscience, 2nd. edition, 2000.

[32] J. P. Hernández Vogt. Software interactivo para el aprendizaje de señas manuales mediante procesamiento de video. *XIII EJI de la UNL y IV EJI de Universidades de Santa Fe*, Septiembre 2009. Director: C. E. Martinez, Co-Director: E. M. Albornoz. 1º Premio en el Área Ingeniería.

[33] J. Smart, R. Roebling, V. Zeitlinm, and Others. *wxWidgets: A portable GUI toolkit.* Version 2.8.10. Website, 2009.
`http://www.wxwidgets.org`.

[34] J. Smart and K. Hock. *Cross-Platform GUI Programming with wxWidgets.* Prentice Hall Professional Technical Reference, 2005.

[35] J. A. Hurtado and Others. *An Open Source GUI Builder for wxWidgets.* Version 3.0.57. Website, 2009.
`http://www.wxformbuilder.org`.