Reconocimiento de Gestos Basado en Visión Estereoscópica y Redes Neuronales

Autor: Victor Daniel Machio

Director: Javier Blanque

Universidad Nacional de Luján

Int. Ruta 5 y 7

6700 Luján, Buenos Aires

República Argentina

Año 2014

Reconocimiento de Gestos Basado en
Visión Estereoscópica y Redes Neuronales

Victor Daniel Machio
Universidad Nacional de Luján
Int. Ruta 5 y 7
6700 Luján, Buenos Aires
República Argentina
vicmp3@gmail.com

Resumen

El presente trabajo introduce las técnicas visión estereoscópica, esqueletización y redes neuronales artificiales con el objetivo de construir un sistema que permita el reconocimiento de los gestos realizados por una mano en una escena determinada.
Para lograr este objetivo se hace uso de la visión estereoscópica que permite la segmentación de la mano del resto de la imagen, luego se obtiene una representación simplificada de la mano buscando su esqueleto y por último se somete el esqueleto a la red neuronal que determina si corresponde a un gesto perteneciente al sistema.

Palabras claves: visión estereoscópica, esqueletos, redes neuronales, color, gestos, reconocimiento de gestos.

Agradecimientos

A mi director de Tesis, profesor Javier Blanque por su tiempo y sugerencias.
A mi familia y amigos por su colaboración y apoyo durante todos mis años de estudio.

Tabla de contenidos

1. Introducción

Hoy en día vivimos en un mundo donde las computadoras son parte de nuestra vida cotidiana, sin darnos cuenta están presentes en la industria, el transporte, la educación y también en nuestro tiempo libre.

Es un hecho que las computadoras realizan tareas repetitivas y procesan un gran volumen de datos con mayor eficiencia que los seres humanos, por este motivo resulta natural querer extender dichas capacidades a tareas más inteligentes como lo es el reconocimiento de objetos y patrones, actividad que la mente humana realiza en forma inconsciente.

Una de las áreas de mayor desarrollo en el último tiempo es la encargada de la interacción humano máquina, *HCI* de sus siglas en inglés (Human-Computer Interfaces). El fin es lograr que las personas se relacionen con las computadoras de una forma más natural y esto puede ser conseguido mediante el uso de gestos dado que es un medio de comunicación muy usado por las personas.

1.1. Objetivos

El objetivo de la tesis es demostrar la factibilidad de la construcción de un sistema capaz de reconocer gestos previamente entrenados. Para esto se realizará una introducción a las técnicas de visión estereoscópica, esqueletización y redes neuronales artificiales, técnicas elegidas para realizar la tarea de reconocimiento de gestos.

1.2. Organización de la Tesis

El trabajo se divide en dos partes. La primer parte formada por los capítulos 2, 3, 4 y 5 presentan la base teórica que permite comprender las técnicas elegidas para la detección de las manos y el reconocimiento de los gestos. La última parte consiste en el desarrollo del sistema propuesto y su integración.

En el capítulo 2, ***Introducción al Reconocimiento de Gestos***, se realiza una introducción a la teoría del reconocimiento de gestos, se detallan las dificultades que se presentan en la actualidad y se enumeran los posibles usos. Además se realiza la presentación del sistema propuesto.

En el capítulo 3, ***Visión Estereoscópica***, se presenta en forma teórica la geometría necesaria por la técnica de visión estéreo para la obtención del mapa de disparidad. Además se describen las tareas intermedias involucradas como la calibración de cámaras, rectificación y búsqueda de correspondencias.

En el capítulo 4, ***Esqueletización***, se describe la técnica que es usada para la representación simplificada del objeto mano a detectar. Se introducen los algoritmos y los problemas más frecuentes.

En el capítulo 5, ***Redes Neuronales Artificiales***, se desarrolla la teoría de las Redes Neuronales Artificiales, sus principales usos, tipos y clasificaciones. También se introducen los métodos de entrenamiento.

En el capítulo 6, ***Integración del sistema***, se realiza la integración de las técnicas descritas en los capítulos anteriores en un sistema de reconocimiento de gestos. Se detallan los pasos seguidos hasta alcanzar el objetivo y los problemas que debieron resolverse.

Finalmente, en el capítulo 7, ***Conclusiones***, se expresan las conclusiones generales y se plantean las ideas para futuros desarrollos.

2. Introducción al Reconocimiento de Gestos

Los gestos constituyen una gran fuente de comunicación entre los humanos de tal forma que una persona sigue haciendo gestos incluso cuando habla por teléfono.

Se denomina gesto a cualquier movimiento realizado mediante el cuerpo, ya sea un movimiento de manos, brazos o cara, con el fin de transmitir algún significado o expresión o para interactuar con el medioambiente [Akl, 2010].

2.1. Reconocimiento de gestos

El reconocimiento de gestos es el proceso de entender y clasificar los movimientos que se realizan con las manos, brazos, cara o incluso la cabeza y que tienen algún significado. Los gestos más expresivos son los movimientos realizados con las manos ya que resultan más naturales e intuitivos para las personas y por lo tanto utilizados más frecuentemente.

Reconocer gestos se ha convertido en una de las áreas más importantes de investigación por su gran importancia en el diseño de interfaces inteligentes entre humanos y computadoras cuyas aplicaciones varían desde el lenguaje de señas, pasando por la rehabilitación médica y hasta realidad virtual [Akl, 2010].

El primer paso para interactuar con los gestos es hacerlos entendibles por las computadoras. Para eso se debe censar la posición del cuerpo humano, su configuración (ángulos y rotaciones) y sus movimientos (velocidad o aceleración).

Para realizar esta tarea existen dos enfoques, el primero consiste en el uso de dispositivos sensores como por ejemplo los guantes, y el segundo consiste en el uso de técnicas de visión por computadora. Ambos enfoques tienen ventajas y desventajas.

El uso de sensores presenta variaciones en precisión, resolución, latencia, rango de movimiento, confort de usuario y costo. Además las interfaces basadas en guantes requieren que el usuario vista un dispositivo y lo lleve consigo junto a los cables que lo conectan a la computadora lo que dificulta la naturalidad de la interacción del usuario con la computadora [Akl, 2010].

Por otro lado las técnicas basadas en visión por computadora tratan este problema, pero se

encuentran con otros como la oclusión. Además, las técnicas de visión por computadora varían entre ellas en el número de cámaras usadas; la velocidad y latencia; la estructura del ambiente como luminosidad y velocidad de movimiento; los requerimientos del usuario como la ropa; las características de bajo nivel usadas en el reconocimiento como bordes, regiones, siluetas, momentos, histogramas; y por último en si se utiliza 2D o 3D.

Por todas estas limitaciones se restringe la aplicación de los sistemas. Por ejemplo si al mirar la TV con la luz apagada uno desea cambiar el volumen con un gesto es difícil reconocer el gesto en esas condiciones de luminosidad. Además resulta poco confortable y natural si se debe mirar directamente a la cámara para completar el gesto [Garg et al., 2009].

2.1.1. Clasificación

Usualmente se considera que gesto y pose es lo mismo pero existe una distinción. La pose corresponde a un gesto estático. Por ejemplo una mano que no involucra movimientos. Por otro lado el gesto se define como un gesto dinámico que es una secuencia de poses conectadas por un movimiento continuo sobre un corto tiempo. Por ejemplo un saludo de despedida moviendo la mano.

Dada esta distinción, el problema de reconocimiento de gestos puede ser separado en dos niveles, un nivel inferior de detección de pose y un nivel superior de reconocimiento de gestos.

Como los gestos son una secuencia de poses conectadas en el tiempo se puede entrenar un reconocedor mediante una posible gramática. De esta forma los gestos de manos pueden verse como un conjunto de poses de mano que se unen para formar una frase al igual que lo hacen las palabras [Garg et al., 2009].

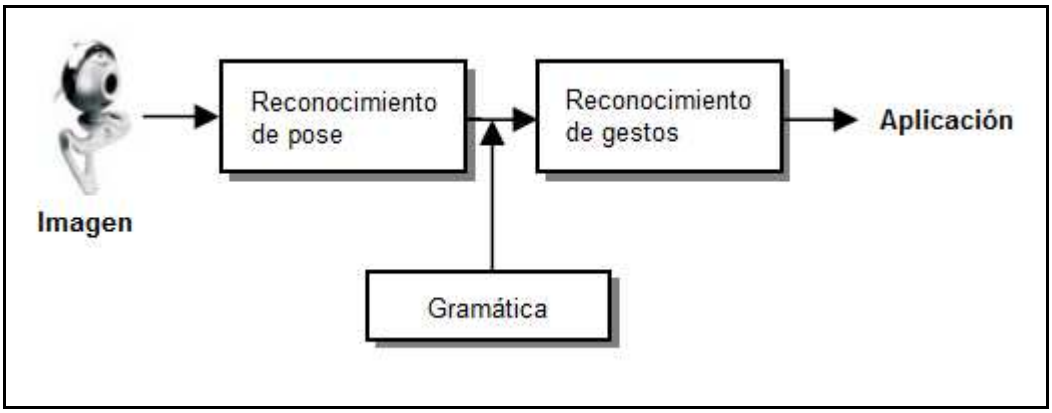


Figura 2.1 - Diagrama del proceso de reconocimiento de gestos [Garg et al., 2009].

2.2. Dificultades

La interpretación de los gestos no resulta sencilla. Generalmente existen varios conceptos asociados a un mismo gesto y viceversa. Por ejemplo, el concepto "*alto*" puede expresarse por un gesto de una mano levantada con la palma hacia adelante, o con un gesto de movimiento exagerado de ambas manos sobre la cabeza. Además los gestos varían entre individuos o para el mismo individuo en distintas oportunidades. Otro inconveniente es que los gestos pueden ser específicos del lenguaje y la cultura. Otras veces pueden verse afectados por el contexto de estar precediendo otro gesto. Además algunos gestos incluso tienen elementos estáticos y dinámicos como el lenguaje de señas.
En el caso del reconocimiento de gestos naturales continuos además se requiere la segmentación temporal y a menudo se necesita especificar el comienzo y fin de un gesto en términos de tiempo y espacio [Mitra et al., 2007].

2.3. Aplicaciones

Específicamente el reconocimiento de gestos puede ser extremadamente útil para:

- Reconocimiento de lenguaje de señas que permita desarrollar ayuda para los discapacitados auditivos. Por ejemplo traducir las señas a texto para que luego algún software "*texto a voz*" lo tome.

- Robots asistivos sociales. Mediante el uso de pequeños sensores y dispositivos apropiados, como acelerómetros y giroscopios, que se encuentren en las vestimentas de un paciente para luego ir leyendo los valores de dichos sensores y

que los robots puedan asistir en la rehabilitación del paciente.

- Desarrollo de interfaces de computadora alternativas. Olvidando el teclado tradicional y mouse para interactuar con la PC, el reconocimiento de gestos puede permitir a los usuarios realizar las tareas frecuentes usando gestos reconocidos mediante el uso de una cámara.

- Juegos interactivos. Puede usarse para controlar interacciones dentro de videojuegos logrando una inmersión del jugador en el juego.

- Control remoto. A través del reconocimiento de gestos se puede controlar distintos dispositivos como TV, puertas automáticas, dispositivos en el auto, etc.

- Otras áreas de aplicación pueden ser: Manejo de objetos en sistemas virtuales, vigilancia inteligente, sistemas médicos, control de robots en ayuda diaria a personas, Interfaz de vehículos, Rehabilitación médica y entrenamiento atlético [Akl, 2010] [Garg et al., 2009].

2.4. Enfoque Propuesto

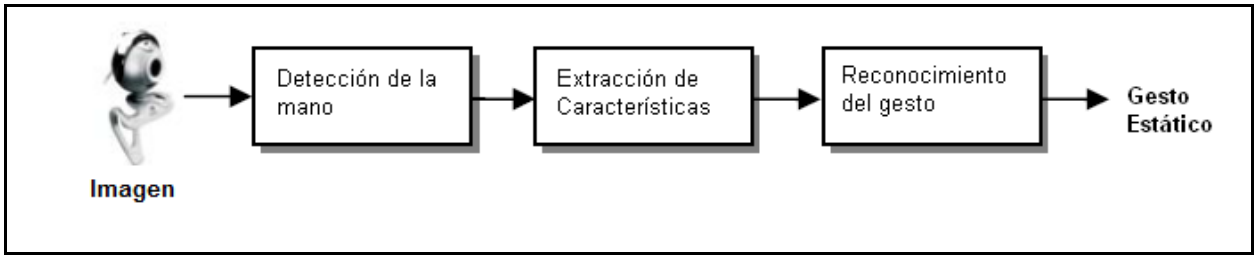


Figura 2.2 - Diagrama con el enfoque propuesto.

El objetivo del sistema es el reconocimiento de gestos estáticos realizados por la mano, para eso se deberán realizar tres tareas:

- Detección de la Mano: La primera tarea que se debe realizar consiste en procesar la imagen que ingresa al sistema para eliminar la información innecesaria y poder separar la mano del resto de la escena.

- Extracción de Características: Una vez obtenida la mano se deben obtener sus características distintivas que permitan luego la comparación con un patrón de

reconocimiento.

- Reconocimiento del Gesto: Se procesan las características obtenidas en el paso anterior en busca de un patrón de reconocimiento para determinar si corresponde a un gesto estático del sistema.

2.4.1. Detección de la Mano

La primera tarea que se debe resolver es la detección de la mano. Para esto se debe separar la escena en información relevante y fondo de la imagen. Existen muchas técnicas de segmentación, entre ellas se encuentran las que usan información 2D o 3D o las que usan características como el color, la forma o el movimiento [Khan et al., 2012].

El color es una técnica muy utilizada y consiste en detectar el objeto mediante la comparación de color. La principal dificultad radica en la elección del espacio de color sobre el cual se trabajará dado que el color de la piel humana varía mucho entre razas o incluso entre individuos de la misma raza. Además se ve influenciado por cambios en la iluminación o características de la cámara y puede ser confundida fácilmente si en el fondo existe una tonalidad similar a la piel humana. Es por este motivo que no resulta una técnica eficiente para utilizar por sí sola.

El uso de la forma cómo característica consiste en extraer los contornos de los objetos presentes en la imagen y compararlos contra una representación de la mano. La principal dificultad que se presenta cuando se parte de imágenes 2D es la oclusión o el punto de vista que pueden conducir a contornos erróneos. La técnica mejora considerablemente combinándola con la visión estereoscópica.

El movimiento es otra técnica que tiene buenos resultados en aquellos escenarios donde el único movimiento presente corresponde al de la mano. Se puede combinar con técnicas como la substracción de fondos para facilitar la detección del movimiento [Garg et al., 2009].

Dentro de las técnicas que usan información 3D se encuentra la visión estereoscópica. Trabajar con una dimensión adicional, denominada distancia o profundidad, permite mejorar la performance dado que se presenta más robustez frente a variaciones de luz o la presencia de oclusiones. Además se tiene una noción del tamaño de los objetos.

Por los motivos enumerados anteriormente se opta por usar un modelo que tome la información 3D de la escena para minimizar los problemas de oclusión y puntos de vista. Esto se logra mediante la implementación de un sistema de Visión Estereoscópico que será explicado en profundidad en el capítulo 3. Luego la detección final se refuerza aplicando la técnica de detección por color.

2.4.2. Extracción de las Características

Una vez resuelta la detección de la mano, y debido a que ésta puede variar en forma y tamaño, se deben extraer de la imagen las características distintivas de la mano.

Para resolver el problema se optó por usar la técnica de esqueletización que devuelve una representación más compacta respetando la forma y conectividad estructural del objeto a la que se denomina esqueleto. La explicación de la técnica se realiza en el capítulo 4.
La principal ventaja de la técnica esqueletización consiste en la reducción de la información del objeto que se debe procesar ganando eficiencia computacional [Martin et al., 2000].

2.4.3. Reconocimiento del Gesto

Una vez obtenidas las características de la mano detectada se deben evaluar con un patrón de reconocimiento para determinar si corresponde a un gesto estático del sistema. Para resolver el problema se elige el uso de una red neuronal artificial debido a su poder y flexibilidad en el reconocimiento de patrones. El capítulo 5 introduce la teoría de ésta técnica [Mitra et al., 2007].

3. Visión Estereoscópica

3.1. Introducción

La visión estereoscópica puede entenderse como la habilidad de inferir información de una escena 3D a partir de dos imágenes tomadas desde distintos puntos de vista. Esto se logra ubicando dos cámaras a la par para obtener así dos vistas distintas de la misma escena. Este proceso es similar al proceso realizado por la visión binocular en las personas [Trucco et al., 1998].

El principio se observa fácilmente si se sostiene la mano extendida con el dedo pulgar levantado, luego se debe mirar fijamente al dedo alternando la vista, es decir primero con un ojo y luego con el otro. En cada vista el dedo aparece desviado cierta cantidad, dicha cantidad se denomina disparidad. La disparidad es lo que permite crear la sensación de profundidad y es el mismo principio usado por la visión estereoscópica.

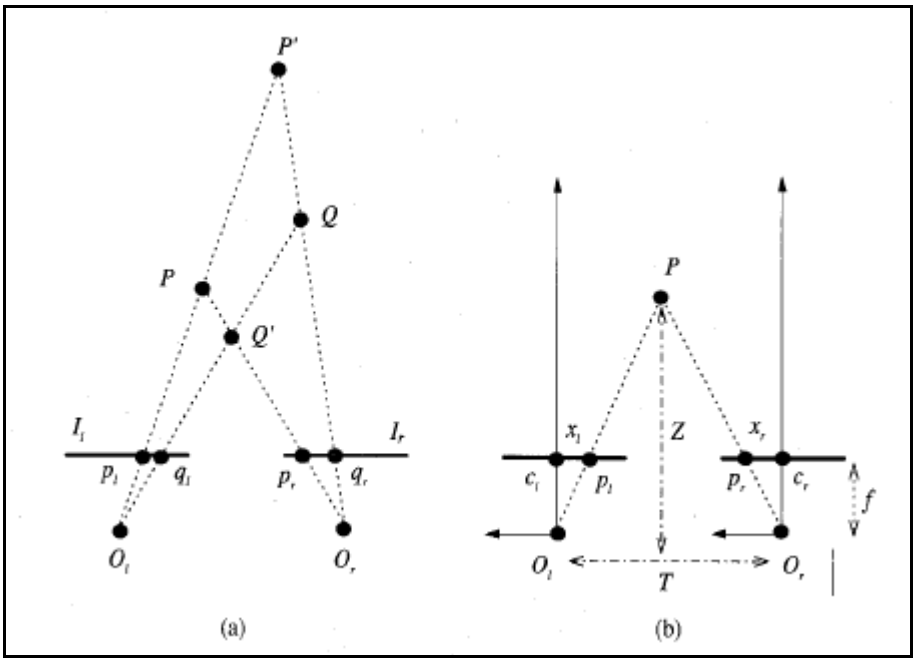


Figura 3.1 - Presentación de un sistema estereoscópico [Trucco et al., 1998].

En la *Figura 3.1 (a)* se presenta un sistema estéreo simplificado visto desde arriba. Los segmentos Ii y Ir representan a las imágenes izquierda y derecha respectivamente. Por otro lado Oi y Or representan los centros de proyección. Los ejes ópticos son paralelos por lo que el punto de intersección se encuentra infinitamente lejos de las cámaras.

El método por el cual se determina la posición en el espacio de las imágenes P y Q es la

triangulación y se calcula mediante la intersección de los segmentos definidos por los centros de proyección y las imágenes P y Q, es decir pi, pr, qi, qr.

El cálculo correcto de la triangulación depende crucialmente de la solución al problema de correspondencia, éste se presenta al intentar emparejar los puntos de las dos imágenes obtenidas, es decir, conociendo las coordenadas de un punto P en la imagen izquierda Ii (pi) se quiere saber cuáles son las coordenadas en la imagen derecha Ir que representan al mismo punto P (pr). Por ejemplo tomando que los puntos correspondientes en las dos imágenes son (pi, pr) para P y (qi, qr) para Q entonces la intersección de los segmentos Oipi - Orpr y Oiqi - Orqr conduce a interpretar los puntos como la proyección de la imagen P y Q. Si por el contrario se toman los pares (pi, qr) y (qi, pr) como los puntos correspondientes, entonces la triangulación devuelve P' y Q' [Trucco et al., 1998].

En la *Figura 3.1 (b)* se asume que el problema de la correspondencia está resuelto y los planos de las imágenes correctamente alineados. Se procede a plantear la ecuación de la triangulación para un punto P cuyas proyecciones son pi y pr, donde la línea base (T) corresponde a la distancia entre los centros de proyección Oi y Or del sistema estéreo; xi y xr son las coordenadas de pi y pr con respecto a los puntos principales ci y cr que corresponden al punto de intersección del segmento principal con el plano de la imagen y pocas veces se encuentra alineado exactamente al centro de imagen ya que depende del eje óptico y lente de la cámara; f es la longitud focal y Z es la distancia entre P y la línea base [Trucco et al., 1998].

De los triángulos (pi, P, pr) y (Oi, P, Or) se obtiene

$$\frac{T+Xi-Xr}{Z-f} = \frac{T}{Z} \quad \text{(a)}$$

y despejando Z

$$Z = f * \frac{T}{d} \qquad \text{(b)}$$

donde $d = Xr - Xi$ es conocido como disparidad y mide la diferencia en la posición retinal entre los puntos correspondientes de las dos imágenes.

Mediante la ecuación (b) se observa que la profundidad es inversamente proporcional a la

disparidad. Esto significa que la disparidad tiende a disminuir cuando el objeto se aleja más de las cámaras cómo se observa en la *Figura 3.2*. En la visión binocular de las personas este efecto se manifiesta como visión doble cuando una persona se acerca un objeto a la vista [Trucco et al., 1998].

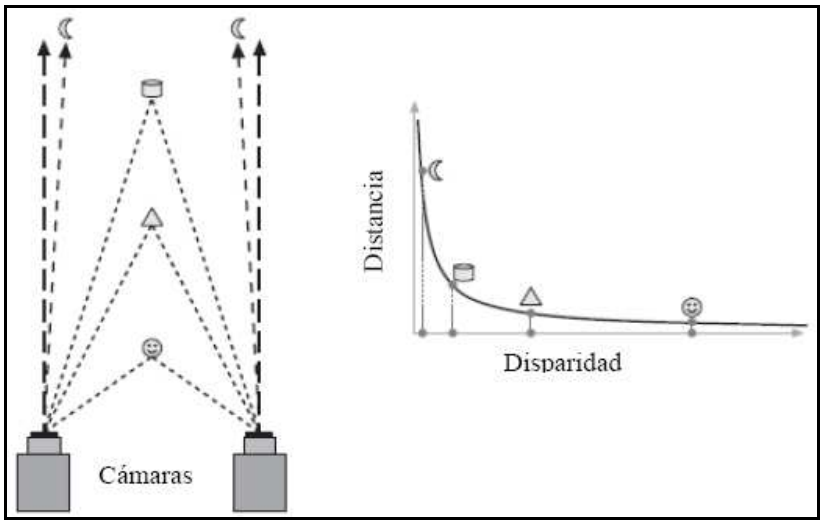


Figura 3.2 - Relación de la Disparidad y la Distancia del objeto [Gary Bradski et al., 2008].

Para poder resolver la ecuación de disparidad se deberá seguir una serie de pasos, entre los principales se distinguen los siguientes [Trucco et al., 1998]:

- Calibración de los parámetros intrínsecos y extrínsecos involucrados en la geometría estereoscópica.

- Rectificación de la geometría epipolar para simplificar la búsqueda que se realiza para resolver el problema de la correspondencia.

- Correspondencia entre los puntos de las imágenes. Este problema se considera la principal dificultad en la visión estéreo.

- Reconstrucción de la escena 3D. Consiste en calcular la profundidad a partir de la disparidad.

3.2. Calibración de las Cámaras

El proceso de calibración es un paso previo importante para poder llevar a cabo la rectificación de las imágenes. Calibrar es el proceso que consiste en obtener los parámetros

intrínsecos y los parámetros extrínsecos de las cámaras así como la posición en la que se encuentra una con respecto a la otra. Dentro de los parámetros intrínsecos o interiores a la cámara se pueden enumerar los factores de escala, la distancia focal, el punto principal y la distorsión; Por otro lado los parámetros extrínsecos o externos describen la posición relativa entre las dos cámaras y se componen de la matriz de traslación que indica el desplazamiento de una cámara respecto a la otra y la matriz de rotación que dice la orientación relativa de las cámaras [Escobedo-Cabello et al., 2010].

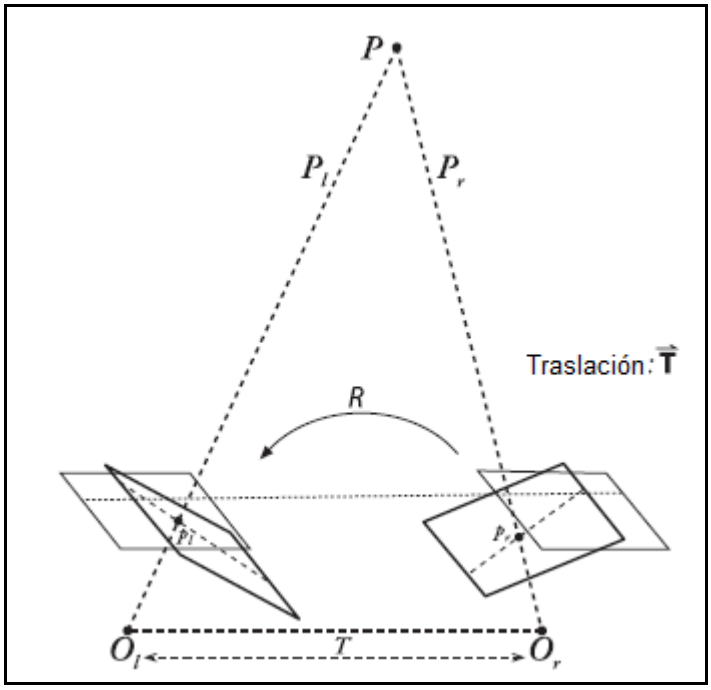


Figura 3.3 - Las cámaras (líneas negras) deben alinearse (líneas grises) tomando como referencia la información de rotación y traslación [Gary Bradski et al., 2008].

Para calibrar se procede al reconocimiento de patrones. Un algoritmo muy utilizado es el propuesto por Z. Zhang que consiste en ir mostrando a ambas cámaras diferentes orientaciones de un tablero de ajedrez [Gary Bradski et al., 2008]. De esta forma con un simple algoritmo de contraste se pueden reconocer las intersecciones blancas-negras de las esquinas del tablero. El tablero usado puede tener cualquier dimensión NxM, de esta manera para cada tablero que mostremos como referencia tendremos NxM puntos conocidos para ambas cámaras. Usando las disparidades de esos puntos entre ambas cámaras se puede calcular la matriz de rotación/traslación que transforma el sistema de coordenadas de la cámara izquierda al sistema de coordenadas de la cámara derecha.

3.3. Geometría Epipolar

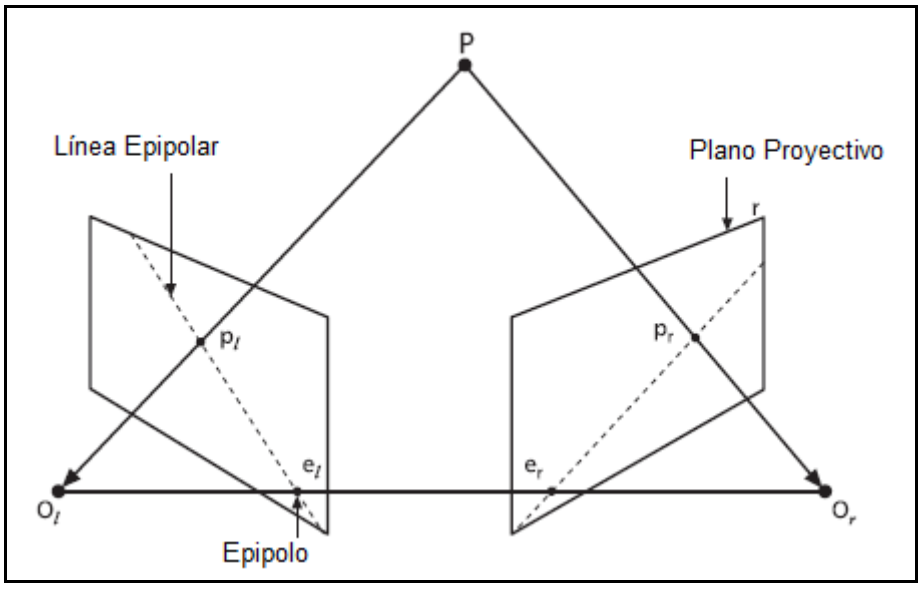


Figura 3.4 - Geometría Epipolar [Trucco et al., 1998].

La Geometría Epipolar se usa en la visión estereoscópica para ayudar a resolver el problema de la correspondencia limitando el espacio de búsqueda de los puntos al relacionar geométricamente los puntos de una imagen con los puntos de la imagen estéreo [Trucco et al., 1998].

En la *Figura 3.4* puede verse el modelo geométrico usado, donde el plano formado por el punto P y los centros ópticos Oi y Or denominado Plano Epipolar interseca a los planos de las imágenes en dos rectas que se denominan Líneas Epipolares.

Luego la búsqueda de un punto en la imagen se restringe a la búsqueda en los puntos que corresponden a una única recta. De cualquier modo, esto significa que por cada píxel de la imagen izquierda habrá que calcular la línea epipolar correspondiente en la imagen derecha. Por este motivo resulta conveniente que cada línea epipolar se encuentra en la misma línea que el píxel al que le corresponde y esto se logra rectificando las imágenes.

3.4. Rectificación

Rectificar un par de imágenes estéreo consiste en transformar cada imagen de tal forma que el par de líneas epipolares se conviertan colineales y paralelas a uno de los ejes de la imagen usualmente el horizontal [Trucco et al., 1998]. Luego de la rectificación, las imágenes quedan en posición frontal paralela como se observa en la *Figura 3.5.*

La importancia de la rectificación es que el problema de correspondencia que involucra una

búsqueda dos dimensional se torna en una búsqueda de una dimensión pues se asegura que la correspondencia de un punto con coordenada vertical yl en la imagen izquierda, se encuentra en la fila de coordenada yr = yl de la imagen derecha denominada "*línea de exploración*" [Trucco et al., 1998].

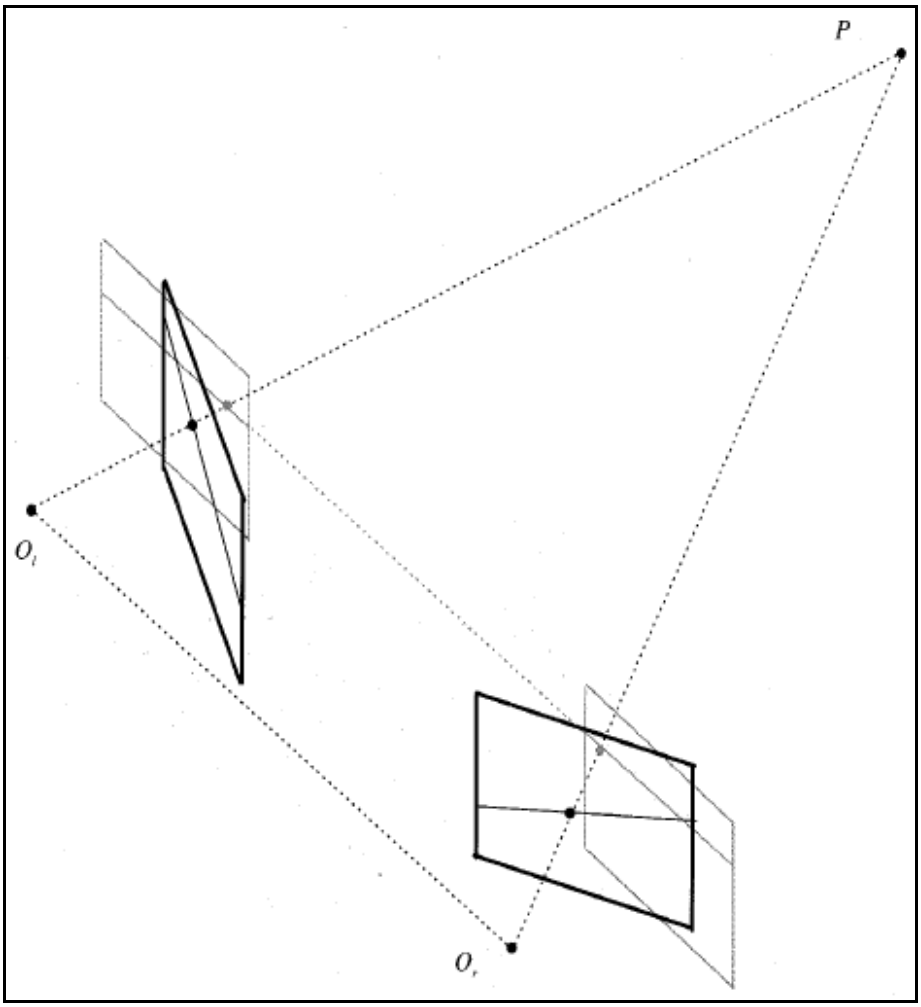


Figura 3.5 - Las líneas epipolares asociadas a un punto P tridimensional en las imágenes originales (líneas negras) se convierten en colineales en las imágenes rectificadas (líneas grises) [Trucco et al., 1998].

3.5. Correspondencia

Un problema que se plantea en el uso de la visión estereoscópica es el denominado problema de correspondencia. Este se presenta al intentar emparejar los puntos de las dos imágenes digitales obtenidas por las distintas cámaras. Es decir, conociendo las coordenadas de un punto P en la imagen izquierda Ii (pi) se quiere saber cuáles son las coordenadas en la imagen derecha Ir (pr) que representan al mismo punto P como puede

observarse en la *Figura 3.1 (a)*.

Así planteada, la correspondencia puede ser vista como un problema de búsqueda, donde dado un elemento en la imagen izquierda se busca el elemento correspondiente en la imagen derecha. En este proceso se involucran dos decisiones [Trucco et al., 1998].

- Que elementos de la imagen coincidir.
- Que medida de similaridad adoptar.

Una clasificación que suele hacerse divide a la correspondencia en dos tipos: el primer método basado en la correlación que realiza la búsqueda en la totalidad de puntos de la imagen y el segundo método basado en las características que realiza la búsqueda en un conjunto de características de la imagen.

No existe un método ideal, la elección del método depende de factores como ámbito de aplicación, requerimientos de software y hardware, etc.

La performance de ambos métodos puede ponerse en riesgo por las oclusiones (puntos sin contrapartida en la otra imagen) y las coincidencias erróneas (debido al ruido en imágenes). Para reducir el efecto de estos fenómenos se pueden usar restricciones como la consistencia (que exista una correspondencia en el par tanto de izquierda a derecha como de derecha a izquierda) y la restricción epipolar.

3.5.1. Método Basado en la Correlación

En este método los elementos a coincidir son ventanas de tamaño fijo, y el criterio de similaridad es una medida de correlación entre las ventanas de las dos imágenes. El elemento correspondiente es dado por la ventana que maximiza el criterio de similaridad en una región de búsqueda [Trucco et al., 1998].

Las ventajas del método radican en que es más fácil de aplicar y devuelve un mapa de disparidad denso. Es necesario contar con imágenes con texturas para un funcionamiento correcto. Por otro lado debido a los efectos de la perspectiva y a los cambios en la dirección de la iluminación, resultan inadecuados para imágenes que son tomadas desde muy distintos puntos de vista. Además los cálculos que deben realizarse para lograr la correlación pueden ser muy costosos.

3.5.2. Método basado en las Características

En este método se restringe la búsqueda por correspondencias a un conjunto disperso de características donde las más comunes son los bordes y esquinas. En lugar de ventanas, se usan propiedades simbólicas y numéricas de características disponibles en los descriptores de características. En lugar de una medida de correlación se usa una medida de distancia entre los descriptores de características. Los elementos correspondientes son aquellos con el par de características más similar, el que tiene menor distancia [Trucco et al., 1998].

Este método resulta más útil cuando existe una información previa de la escena, además puede ser más rápido pero hay que tener en cuenta que hay un costo de producir el descriptor de características. A diferencia del método anterior devuelve un mapa de disparidad disperso y suele ser menos sensible a cambios de iluminación.

3.6. Mapa de Disparidad

Luego de la rectificación de las imágenes y su alineado es posible obtener el mapa de disparidad. Se conoce como disparidad a la distancia de un píxel en una imagen con su correspondiente en la otra imagen [Trucco et al., 1998]. Asimismo, se denomina mapa de disparidad a la matriz constituida por estos valores.

Un mapa de disparidad consiste en una imagen 2D donde el color de cada píxel representa la distancia del punto a la cámara. En otras palabras, los píxeles más claros se encuentran más cerca y los más oscuros más alejados.

Estos mapas resultan útiles porque pueden ser usados para varios propósitos [Dröppelmann et al., 2000]:

- Modelado 3D a partir de imágenes 2D: Cuando se toman dos imágenes 2D de un medioambiente 3D y se calcula el mapa de disparidad se puede crear un modelo 3D de la escena usando la profundidad como la tercera dimensión.

- Rastreo de objetos: Cuando se tiene un mapa de disparidad es más fácil seguir un objeto porque se tiene como adicional la posibilidad de segmentación. Se puede crear segmentos de píxeles que están más cerca basándose en la profundidad de los píxeles y sus adyacencias.

- Reconocimiento de objetos frontales: Cuando se aplica la segmentación basándose en un mapa de disparidad se puede distinguir objetos que están situados enfrente de la escena.

- Como información sobre el medioambiente en la planificación del camino: Un mapa de disparidad provee información adicional para la planificación de caminos

4. Esqueletización

4.1. Introducción

La esqueletización es una técnica que describe las propiedades globales de un objeto reduciéndolo en una representación más compacta. Los esqueletos representan la forma del objeto con una cantidad relativamente pequeña de píxeles expresando la conectividad estructural de sus principales componentes. También son muy útiles a la hora de reconocer, dentro de una imagen, objetos o patrones alargados o con una determinada forma como por ejemplo caracteres, polígonos, patrones cromosómicos, etc. Además proporcionan una abstracción de las características topológicas y geométricas del objeto, de forma que la esqueletización puede verse también como un proceso de compresión de datos [Martin et al., 2000]. Generalmente la obtención del esqueleto suele ser parte de un proceso intermedio que prepara al objeto para un análisis posterior.

4.2. Algoritmos

La primera definición de esqueleto fue realizada por Blum en 1967 y es conocida como Transformación al Eje Medio, *MAF* de sus siglas en inglés (Medial Axis Function). Según Blum el Eje Medio consiste en el conjunto de puntos, que son los puntos centrales de las circunferencias máximas que pueden ser contenidos dentro de la forma o el objeto.

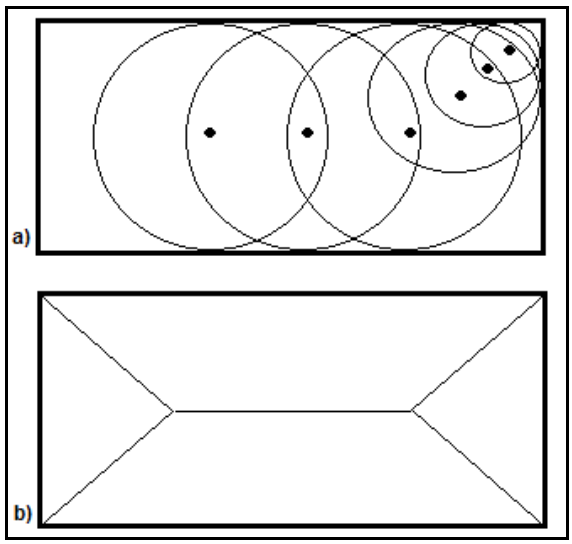


Figura 4.1 - En (a) se visualizan algunas de las circunferencias de radio máximo del objeto. En (b) se visualizan todos los centros de las circunferencias de radio máximo que conforman el esqueleto del objeto.

De esta definición surge una clasificación para los puntos [Chetverikov et al., 1993]. Cada punto del eje medio pertenece a un punto interior, punto ramal o punto final. El punto interior corresponde a un punto cuya circunferencia toca el borde en dos puntos. El punto ramal es aquel cuya circunferencia toca el borde en un número finito de puntos mayor a dos y el punto final es aquel cuya circunferencia coincide en pendiente y curvatura con el borde.

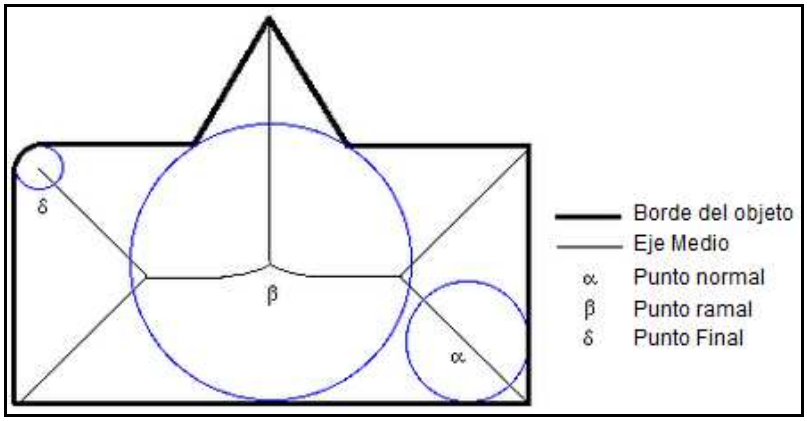


Figura 4.2 - Visualización de la clasificación de los puntos del Eje Medio.

La implementación original de *MAF* insume mucho tiempo por lo que es difícil una implementación directa. Por este motivo se realiza una conversión del espacio continuo al discreto [Martin et al., 2000].

Una buena aproximación a *MAF* se obtiene fácilmente computando primero la distancia de cada píxel del objeto al píxel del borde más cercano y luego calculando el Laplaciano de la distancia. Los píxeles con los valores más grandes pertenecen al eje medio. La manera en que la distancia entre píxeles y borde se miden influye en el resultado final (esqueleto).

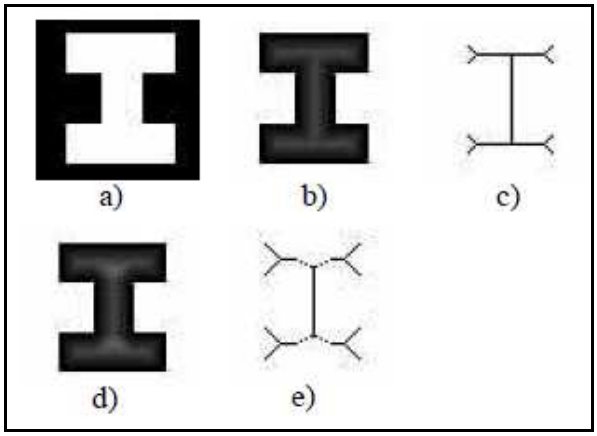


Figura 4.3 - Se observa el proceso de esqueletización usando MAF. a) imagen de ejemplo. b) mapa de distancia para un valor 8. c) esqueleto para una distancia 8. d) mapa de distancia para un valor 4. e) esqueleto para una distancia 4 [Martin et al., 2000].

Otros algoritmos populares dadas su efectividad y confiabilidad son los algoritmos que utilizan plantillas donde por cada coincidencia se elimina el píxel central. Entre estos algoritmos iterativos se encuentran Stentiford y Zhang-Suen

4.2.1. Algoritmo Stentiford

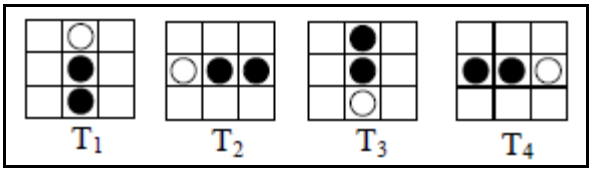


Figura 4.4 - Plantillas para identificar los píxeles a borrar para el método Stentiford. Los cuadrados vacíos corresponden a lugares donde el color del píxel no necesita ser chequeado.

En esta técnica se usa un conjunto de plantillas de 3x3 para escanear la imagen según se observa en la *Figura 4.4*. Los pasos implicados del algoritmo según [Martin et al., 2000] son los siguientes:

1. Encontrar el píxel (i,j) en la imagen donde se obtenga una coincidencia con la plantilla (T1). De acuerdo a esta plantilla todos los píxeles a lo largo de la parte superior de la imagen se eliminan moviéndose de izquierda a derecha y de arriba a abajo.

2. Si el píxel central no es un punto final y no se tiene conectividad = 1 entonces se marca el píxel para su eliminación.

   Píxel Punto Final: Se considera píxel punto final si únicamente está conectado a único píxel. Es decir, si un píxel negro tiene únicamente un píxel negro de los 8 posibles vecinos que podría tener.

   Número de conectividad: Es la medida de cuantos objetos están conectados con un píxel en particular.

3. Se repiten los pasos 1 y 2 para todos los píxeles que coinciden con la plantilla (T1).

4. Se repiten los pasos 1 a 3 para el resto de las plantillas (T2), (T3) y (T4).

(T2) Hará coincidir los píxeles del lado izquierdo del objeto, moviéndose de abajo hacia arriba y de izquierda a derecha.
(T3) seleccionará los píxeles del fondo del objeto, moviéndose de derecha a izquierda y de abajo hacia arriba.
(T4) localizará los píxeles en la derecha del objeto moviéndose de arriba hacia abajo y de derecha a izquierda.

5. Blanquear los píxeles marcados para su borrado.

En la *Figura 4.5* puede verse un ejemplo de dicho proceso.

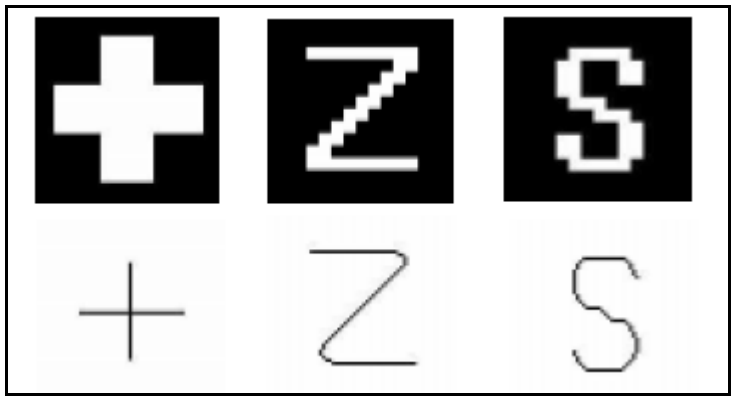


Figura 4.5 - Ejemplo de esqueletización mediante el método Stentiford.

4.2.2. Algoritmo Zhang-Suen

Este algoritmo corresponde a un método en paralelo lo que significa que un valor nuevo obtenido depende del valor previo de la iteración. Es un método rápido y fácil de implementar y el algoritmo consta de dos sub iteraciones [Martin et al., 2000].

En la primera iteración, un píxel I(i,j) se borra si se cumplen las siguientes condiciones:

1. Su número de conectividad es 1.
2. Tiene al menos 2 vecinos de color negro y como máximo 6.
3. Al menos uno de los I(i,j+1), I(i-1,J) y I(i,,j-1) son de color blanco.
4. Al menos uno de kis I(i-1,j), I(i+1,j) y I(i,j-1) son de color blanco.

En la segunda sub iteración cambian las condiciones del paso tres y cuatro:

1. Su número de conectividad es 1.

2. Tiene al menos 2 vecinos de color negro y como máximo 6.
3. Al menos uno de los I(i-1,j), I(i,J+1) y I(i+1,j) son de color blanco.
4. Al menos uno de kis I(i,j+1), I(i+1,j) y I(i,j-1) son de color blanco.

Al final, los píxeles que cumplan las condiciones serán borrados. Si al concluir cualquiera de las sub iteraciones no hay píxeles que borrar, entonces el algoritmo se detiene.

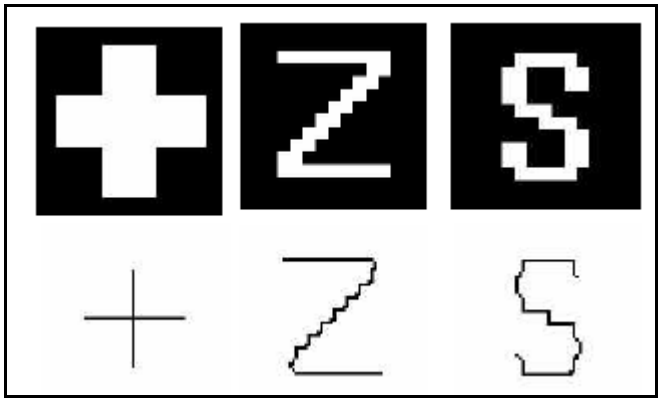


Figura 4.6 - Ejemplo de esqueletización mediante el método Zhang-Suen.

4.3. Problemas comunes

Los esqueletos suelen no ser precisos, entre los inconvenientes más comunes se destacan dos [WWW01]:

- Unicidad: El mismo esqueleto puede pertenecer a dos objetos distintos, *Figura 4.7 (a).*

- Estabilidad: Una perturbación en el borde genera un cambio brusco del esqueleto, *Figura 4.7 (b).*

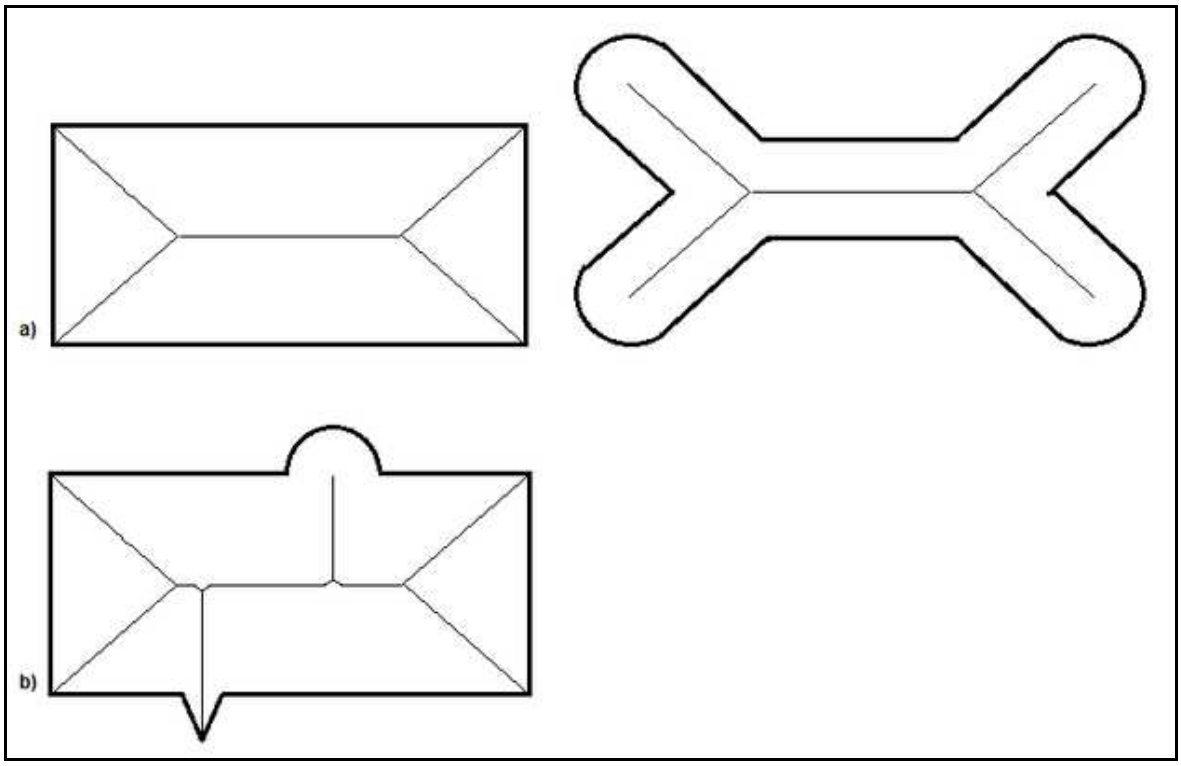


Figura 4.7 - Ejemplo de Unicidad (a) y de Estabilidad (b).

5. Redes Neuronales Artificiales

5.1. Introducción

Las redes neuronales corresponden a un enfoque de la inteligencia artificial que apunta a modelar el cerebro humano. Dentro del cerebro se encuentran las neuronas, que son unidades de procesamiento que operan en paralelo. Se estima que existen diez billones de neuronas en el cerebro humano y alrededor de sesenta trillones de conexiones entre las neuronas. Cada neurona recibe como entrada pequeñas señales eléctricas desde otras neuronas y en contrapartida responde también con señales eléctricas. Estas respuestas son evaluadas en el sentido que una neurona no "*dispara*" una salida a menos que cierto umbral denominado *bias* se haya alcanzado. La medida de evaluación puede ir variando a través de la experiencia. El cerebro es por lo tanto una red de neuronas actuando en paralelo [Shweta et al., 2011].

Similarmente, una red de neuronas artificiales consiste básicamente de neuronas artificiales, que son modelos matemáticos o abstracciones simplificadas de las neuronas biológicas. A diferencia de una neurona biológica, que recibe y emite impulsos eléctricos analógicos a través de la emisión y absorción de señales químicas, la neurona artificial, por ejemplo el *perceptrón*" recibe como entrada valores numéricos y devuelve como salida también valores numéricos.

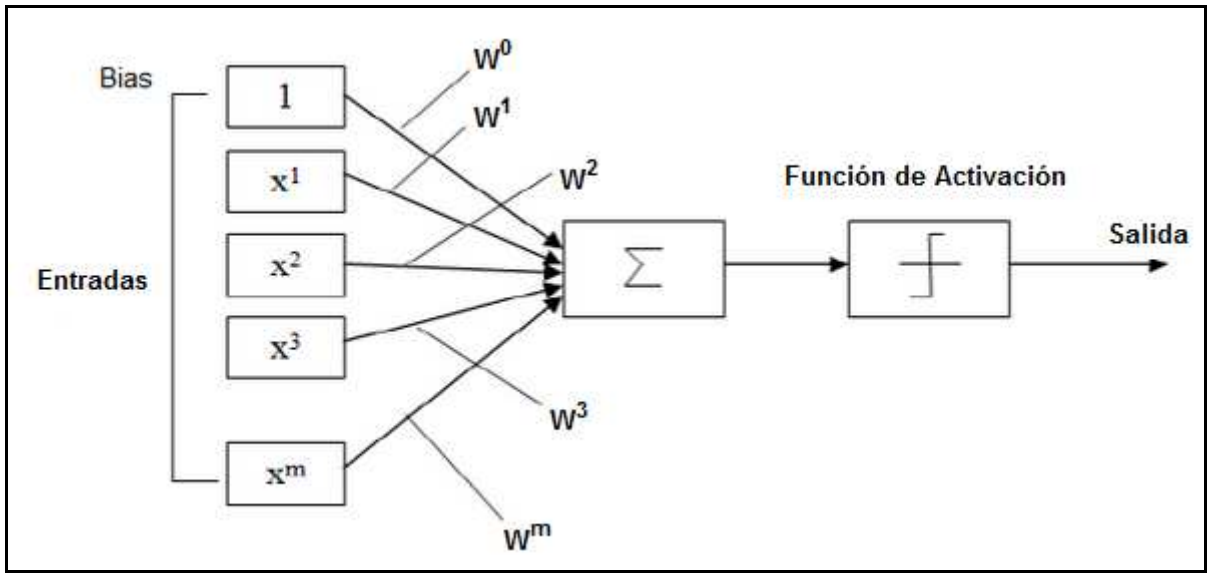


Figura 5.1 - Representación básica de una neurona artificial.

La entrada al perceptrón consiste de un valor numérico multiplicado por un peso más un bias. El perceptrón solo dispara la salida cuando un porcentaje total de las señales de

entrada exceden un cierto umbral. Al igual que las redes biológicas, la salida puede alimentar otros perceptrones.

La entrada ponderada a un perceptrón actúa bajo una función de activación que determina la activación o salida. La razón para la aplicación de una función de activación distinta de la identidad surge de la necesidad de que las neuronas produzcan una salida acotada.

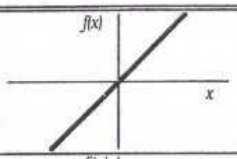
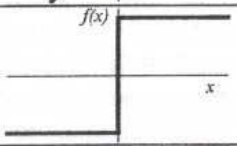
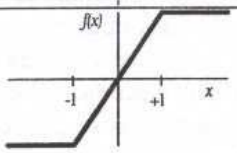
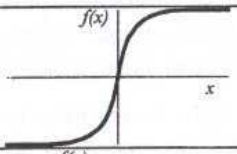
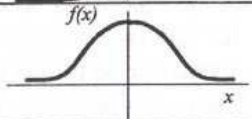
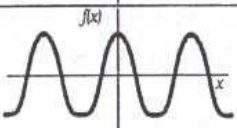
	Función	Rango	Gráfica
Identidad	$y = x$	$[-\infty, +\infty]$	
Escalón	$y = sign(x)$ $y = H(x)$	$\{-1, +1\}$ $\{0, +1\}$	
Lineal a tramos	$y = \begin{cases} -1, & si\ x < -l \\ x, & si\ +l \le x \le -l \\ +1, & si\ x > +l \end{cases}$	$[-1, +1]$	
Sigmoidea	$y = \frac{1}{1+e^{-x}}$ $y = tgh(x)$	$[0, +1]$ $[-1, +1]$	
Gaussiana	$y = Ae^{-Bx^2}$	$[0,+1]$	
Sinusoidal	$y = A\,\mathrm{sen}(\omega x + \varphi)$	$[-1,+1]$	

Figura 5.2 - Funciones de activación comunes [WWW02].

En muchos casos una red de neuronas artificiales es un sistema adaptativo que cambia su estructura basado en información interna y externa que fluye a través de la red durante la fase de entrenamiento. La utilidad de estos modelos yace en el hecho que pueden ser usados para inferir una función a través de observaciones. Esto es particularmente útil en aplicaciones donde la complejidad de los datos o tareas hacen el diseño manual de tal función imposible, muy complejo o ineficiente. Una de las tareas a la que se aplica una red de neuronas artificiales es la clasificación, incluyendo reconocimiento de secuencias y patrones.

5.2. Entrenamiento

Una de las características de las redes neuronales es su capacidad de aprender, es decir poder modificar las matrices de pesos para que la evaluación de una entrada de como resultado un cierta salida deseada. Dicho aprendizaje se logra presentando secuencialmente a la red neuronal vectores de entrada y ajustando los pesos según cierta regla de aprendizaje. El fin es lograr que los pesos converjan a valores estables con un mínimo error. Los métodos de aprendizaje más comunes son el Aprendizaje Supervisado y el Aprendizaje no Supervisado.

5.2.1. Aprendizaje Supervisado

Es un método de entrenamiento que predice los valores de las salidas tomando como referencia datos de entrenamiento conocidos.
Compara las predicciones realizadas con el valor de referencia y aprende de los errores obtenidos. Si la predicción es igual al valor de referencia entonces no se realizan cambios en los pesos del sistema. Por el contrario si la predicción difiere entonces el error es propagado a través del sistema ajustando los pesos correspondientes. El proceso se repite hasta que el error es aceptable de acuerdo a cierto criterio.

5.2.1.1. Algoritmo de Propagación Hacia Atrás o Backpropagation

Es una técnica de aprendizaje supervisado usada para el entrenamiento de redes neuronales artificiales. Fue introducida por Paul Werbos en 1974 y mejorada por David E. Rumelhart, Geoffrey E. Hinton y Ronald J. Williams en 1986. Es más útil para redes "*Pre Alimentadas*", redes que no tienen retroalimentación o conexiones en forma de bucles.

En inglés, Backpropagation es una abreviación de "*backwards propagation of errors*" cuya traducción es "*propagación hacia atrás de los errores*" y se utiliza en el entrenamiento de redes multicapas. Como el nombre lo hace notar, en este algoritmo el error (el aprendizaje) se propaga hacia atrás, desde la salida hacia los nodos internos.

El algoritmo consiste en un proceso iterativo donde los pesos se van ajustando en una serie de pasos, es decir se van buscando pesos que minimicen la función de error y esto se logra aplicando la derivada del error con respecto al peso. Es por este motivo que deberá usarse

una función de activación diferenciable para garantizar que la salida también sea una función diferenciable [Rojas, 1996].

Resumen de pasos para aplicar el algoritmo:

1. Inicializar pesos de la red con valores aleatorios.
2. Presentar el patrón de entrada y especificar la salida correspondiente a la entrada.
3. Calcular la salida actual para el patrón de entrada.
4. Calcular el error para todas las neuronas.
5. Actualizar los pesos.
6. Repetir el proceso hasta que el error resulte pequeño.

5.2.2. Aprendizaje No Supervisado

Muchas veces no se dispone de antemano el resultado deseado, por lo que el aprendizaje supervisado no resulta adecuado. En el aprendizaje no supervisado, ideado principalmente por Kohonen, no se presenta ninguna salida o respuesta conocida como parte del proceso de corrección de los pesos. Por el contrario, se busca la consistencia entre las entradas y las salidas, es decir, a entradas iguales deben corresponder salidas iguales o similares. Se agrupan las entradas en clases según criterios estadísticos. Estas redes suelen ser más efectivas para describir información en lugar de predecirla.

5.3. Clasificación de Redes Neuronales Artificiales

Por si sola una neurona artificial no es muy útil para resolver problemas, en cambio la combinación de varias neuronas artificiales da lugar a una red de neuronas artificiales que es capaz de resolver problemas complejos y no lineales. Para un mejor manejo las neuronas se agrupan en capas y la forma en que las neuronas se relacionan en la red neuronal artificial determina la arquitectura de la red [Krenker et al., 2011] [Noor et al., 2012]. A continuación se detalla la principal clasificación:

5.3.1. Redes Neuronales Alimentadas Hacia Adelante

En este tipo de redes la información fluye en una única dirección, comenzando en la capa de entrada y llegando a la capa de salida a través de capas ocultas si existieran. No existen

limitaciones en la cantidad de capas que se pueden usar, el tipo de función de activación o el número de conexiones entre las neuronas [Krenker et al., 2011] [Noor et al., 2012].

La red más simple es la Red Unicapa y consiste en una única capa como puede observarse en la *Figura 5.3.* La red que contiene una o más capas ocultas se denomina Red Multicapa y se observa en la *Figura 5.4*. La existencia de capas ocultas permite resolver problemas más complejos cómo los problemas no lineales.

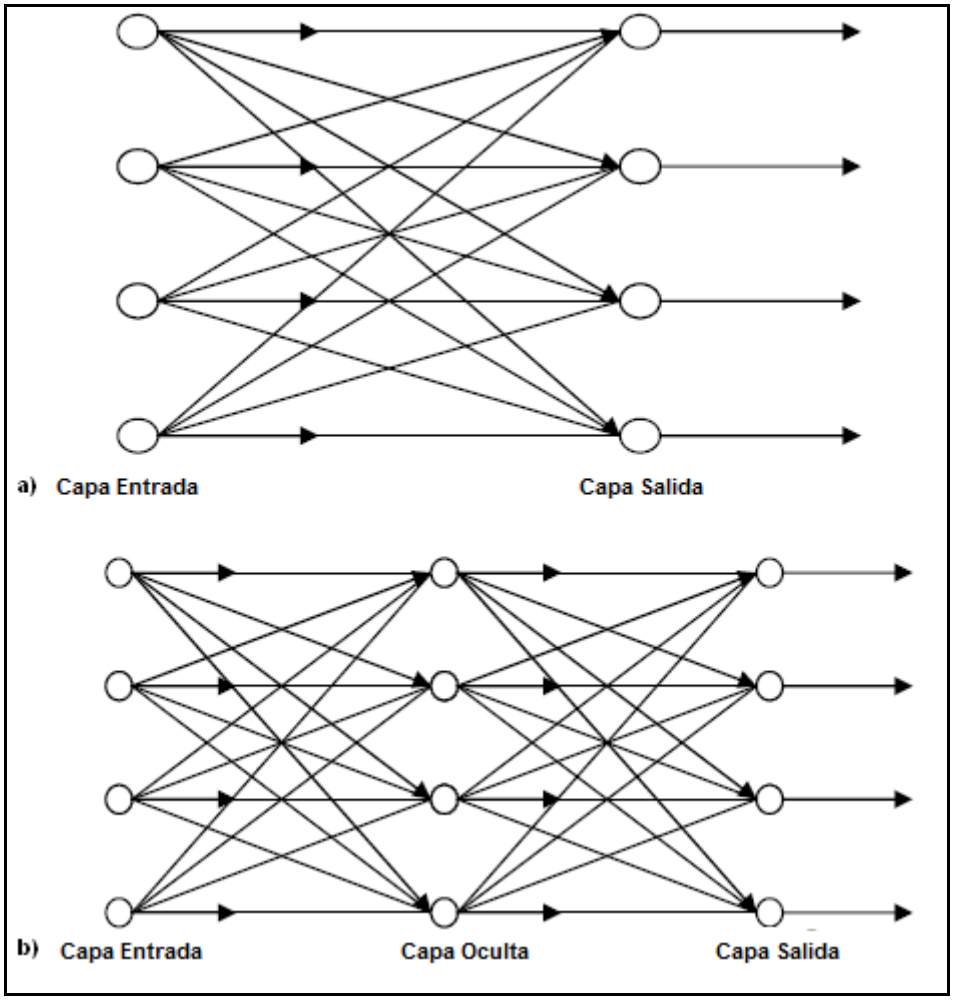


Figura 5.3 - Red Alimentada Hacia Adelante. Unicapa (a). Multicapa (b).

5.3.2. Redes Neuronales Recurrentes

Las Redes Recurrentes son similares a las Redes Alimentadas Hacia Adelante excepto que no tienen la limitación del uso de bucles [Krenker et al., 2011] [Noor et al., 2012]. En este tipo de redes la información no viaja en una única dirección sino que puede realimentar las entradas, esto permite que la red pueda disponer de un comportamiento temporal dinámico y pueda procesar cualquier secuencia de entrada haciendo uso de una memoria interna. Se

comienza aplicando la entrada, se calcula la salida, y a diferencia de las Redes Alimentadas Hacia Adelante, la salida reingresa a la entrada calculándose nuevamente, y así sucesivamente.

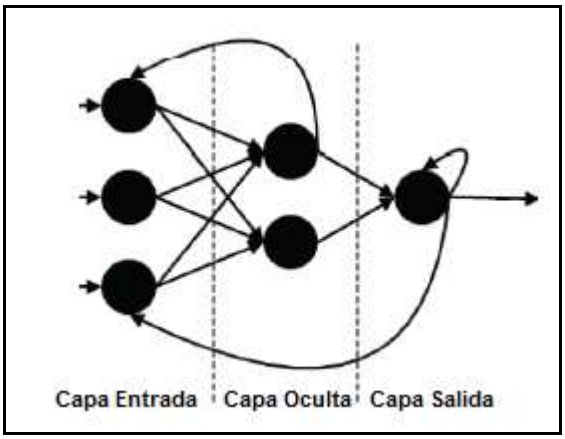


Figura 5.4 - Red Recurrente.

5.3.2.1. Redes Neuronales Completamente Recurrentes

No existe una capa de entrada distintiva, cada neurona tiene entrada del resto de las neuronas pudiendo tener además retroalimentación consigo misma [Krenker et al., 2011].

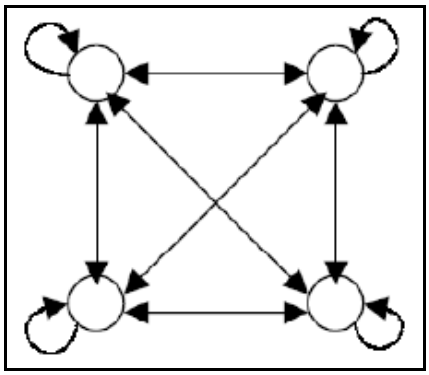


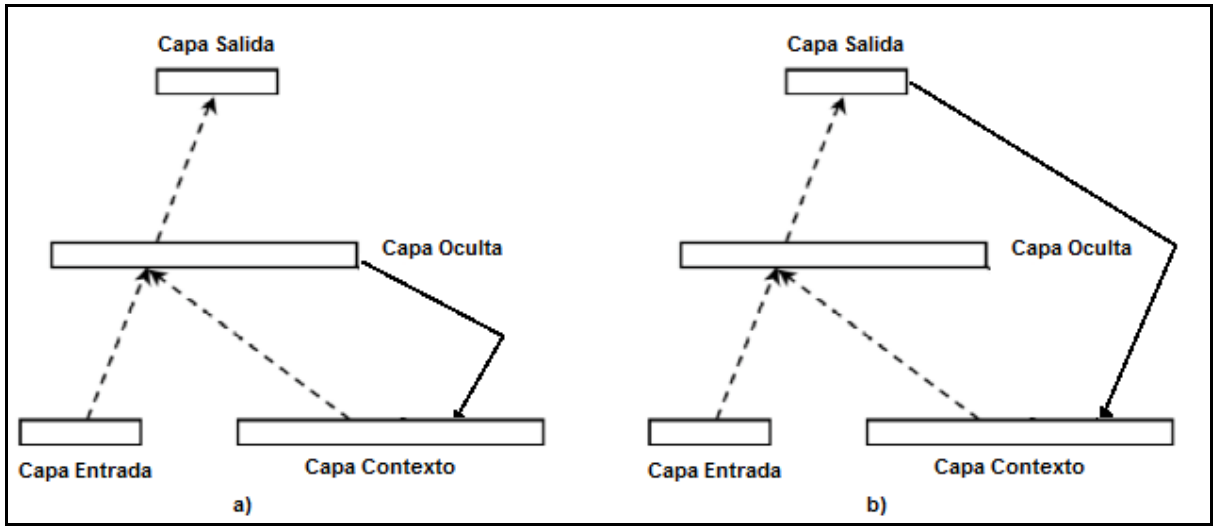
Figura 5.5 - Red Completamente Recurrente.

5.3.2.2. Redes Neuronales Parcialmente Recurrentes

Son un tipo especial de Redes Multicapa que poseen algunas conexiones recurrentes, estas conexiones permiten recordar el estado anterior de ciertas neuronas de la red [Krenker et al., 2011] [Noor et al., 2012]. En la capa de entrada existen unas neuronas especiales, denominadas neuronas de contexto, que son receptoras de las conexiones recurrentes.

Estas neuronas actúan como una memoria, almacenando el estado de las neuronas de una cierta capa en el instante anterior.

Dentro de este tipo de redes las dos más conocidas son las Redes de Elman y las Redes de Jordan. Las Redes de Elman tienen una conexión desde las neuronas de la capa oculta hacia las neuronas de contexto. Las Redes de Jordan se diferencian en que la conexión a las neuronas de contexto se realiza desde las neuronas de la capa de salida.



.Figura 5.6 - Redes Neuronales Parcialmente Recurrente. (a) Red de Elman. (b) Red de Jordan.

5.3.2.3. Redes Neuronales Recurrentes de Tiempo Continuo

En las Redes Neuronales Recurrentes de Tiempo Continuo existen ciclos y pueden o no existir capas diferenciadas. En éste tipo de redes la información fluye por los ciclos permitiendo la posibilidad de capturar y explotar eventos dependientes del tiempo.

A diferencia de las Redes Neuronales Alimentadas Hacia Adelante donde la actualización de las neuronas es síncrona, lo que resulta apropiado para el reconocimiento y clasificación de patrones, en estas redes se usa el tiempo como soporte de la información logrando un funcionamiento asíncrono, con conexiones interneuronales que producen un retardo en las entradas de las neuronas. De esta manera los estados de activación de las neuronas pueden detectar y mantener patrones de activación dependientes del tiempo y que se producen brevemente en el tiempo. Esta información puede modular el comportamiento, producir diferentes acciones para información sensorial similar y ser usada para detectar y representar secuencias de comportamiento [Krenker et al., 2011] [Noor et al., 2012].

5.3.3. Mapas Autoorganizados de Kohonen (SOM)

Los Mapas Autoorganizados, SOM de sus siglas en inglés (Self-Organizing Map), son un tipo de red neuronal desarrollada en 1982 por Tuevo Kohonen. Se denominan así dado que no hay supervisión y el aprendizaje es mediante un aprendizaje competitivo sin supervisión. El término "*mapa*" se refiere a que mapean/asignan los pesos para corresponderse a las entradas y los nodos tratan de actuar como las entradas que se le presentan. Son útiles para clasificación de color y de imágenes [Noor et al., 2012].

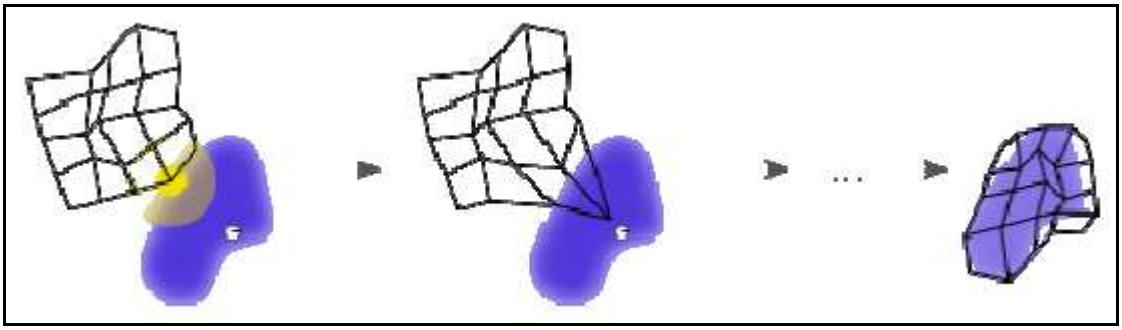


Figura 5.7 - Esquema de Entrenamiento de Mapas Autoorganizados.

En la *Figura 5.7* se visualiza un esquema del entrenamiento donde la región azul representa la distribución de los datos entrenados y el disco blanco pequeño es la muestra de entrenamiento actual. Primero se selecciona el nodo más cercano al nodo de entrenamiento (remarcado en amarillo), luego la grilla se acerca al disco blanco y a su vecindario, por último luego de algunas iteraciones la grilla tiende a aproximarse a la distribución de los datos.

6. Integración del Sistema

6.1. Introducción

En los capítulos anteriores se presentó una serie de técnicas que en conjunto pueden ser utilizadas para el reconocimiento de gestos. En este capítulo se pretende demostrar la eficacia de dichas técnicas mediante la integración en un sistema de reconocimiento de gestos.

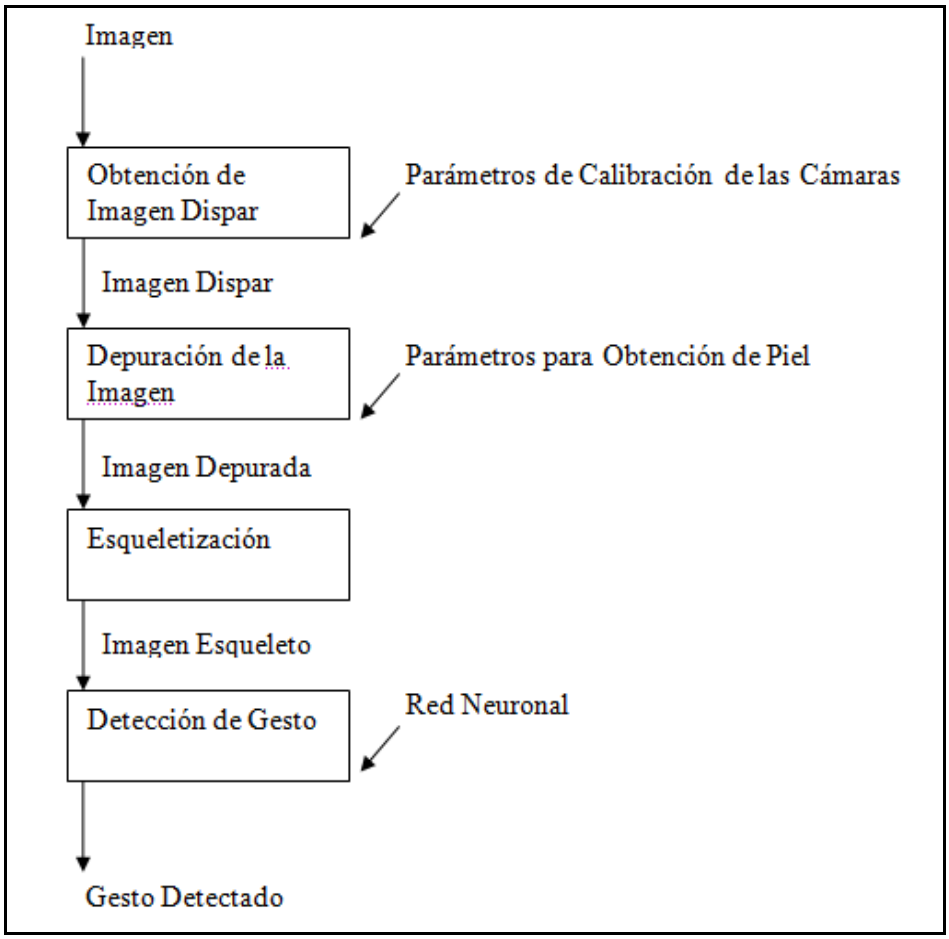


Figura 6.1 - Sistema Propuesto.

En la *Figura 6.1* se visualizan los módulos que componen el sistema propuesto, los mismos son:

- Obtención de Imagen Dispar: Obtiene las imágenes de las cámaras sin procesar, rectifica las imágenes y genera un mapa de disparidad. La visión estereoscópica se logra haciendo uso de los *Parámetros de Calibración de las Cámaras*.

- Depuración de la Imagen: Procesa la *Imagen Dispar* para obtener el objeto que se

encuentra más cercano a las cámaras. Utiliza los *Parámetros de Color de Piel* recibidos para segmentar correctamente la mano en la imagen y aplica una serie de operaciones morfológicas hasta obtener una *Imagen Depurada* que consiste en la mano binarizada.

- Esqueletización: Procesa la *Imagen Depurada* para obtener la *Imagen Esqueleto* equivalente a la mano de la imagen original.

- Detección de Gesto: Evalúa la *Imagen Esqueleto* mediante la red neuronal entrenada y obtiene como resultado el *Gesto Detectado*.

Además de los módulos principales descritos anteriormente, el sistema cuenta con módulos de parametrización necesarios para el correcto funcionamiento del sistema como se observa en la *Figura 6.2.*

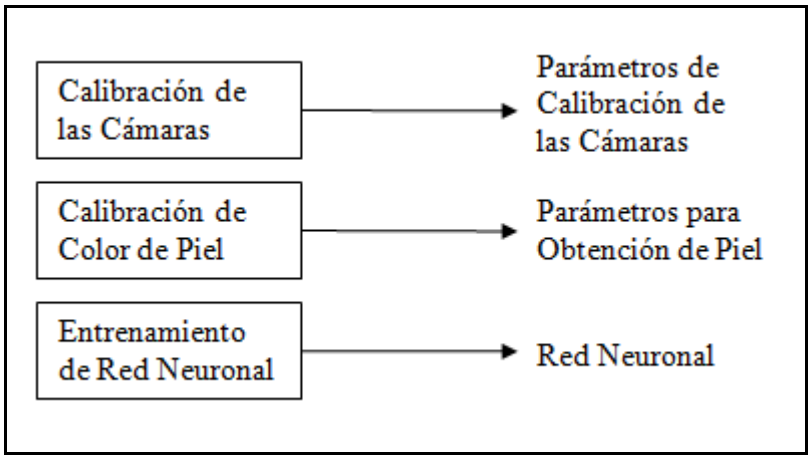


Figura 6.2 - Módulos de Parametrización.

- Calibración de las Cámaras: Calibra las cámaras mediante el procesamiento de una serie de imágenes y construye el sistema estereoscópico. Los *Parámetros de Calibración de las Cámaras* son exportados para su posterior uso en el sistema.

- Calibración de Color de Piel: Obtiene el rango de tono de piel en el espacio de color YCbCr y los almacena cómo *Parámetros para Obtención de Piel* para su posterior utilización en la segmentación de la mano.

- Entrenamiento de Red Neuronal: Captura una secuencia de muestras para cada gesto a detectar, los entrena y construye la *Red Neuronal* del sistema.

6.2. Descripción Detallada del Sistema

6.2.1. Módulos de Parametrización

6.2.1.1. Calibración de las Cámaras y Obtención del Sistema Estéreo

El objetivo del módulo consiste en construir el sistema estereoscópico, y permitir la exportación de los parámetros de calibración para el uso posterior desde la aplicación principal.

Para la construcción del sistema estereoscópico se opta por utilizar dos cámaras web independientes en lugar de un dispositivo con dos lentes. Esta elección presenta una dificultad extra debido a que los lentes de ambas cámaras no se encuentran alineados. La solución del problema se logra luego de la correcta calibración de las cámaras y del proceso de rectificación.

Especificaciones técnicas de las cámaras usadas:

- Modelo: Genius FaceCam 320x
- Interfaz: USB 2.0
- Sensor de Imagen: VGA píxel CMOS
- Resolución de Video: 640 x 480 píxeles / 30fps
- Tipo de Lente: Foco Manual

Figura 6.3 - Modelo Genius FaceCam 320x usado en el sistema estereoscópico.

Las cámaras elegidas para la adquisición de las imágenes corresponden al modelo "*Genius FaceCam 320x*". Dentro de sus ventajas se encuentran la compatibilidad con distintos sistemas operativos, la conectividad mediante USB que hace innecesario el uso de drivers para su utilización y su bajo costo.

El sistema estereoscópico se simula ubicando las dos cámaras distanciadas horizontalmente una de la otra como se observa en la *Figura 6.4,* las cámaras se fijan para impedir que el movimiento descalibre el sistema. La resolución utilizada en la captura de

imágenes es 320 x 240 píxeles.

Figura 6.4 - Disposición de las cámaras para simular el sistema estéreo.

Funcionamiento del Módulo

El primer paso para lograr la calibración de las cámaras consiste en adquirir una colección de imágenes de un patrón conocido, cuyos puntos sean identificables fácilmente. Luego se debe comparar el patrón desde distintos ángulos y calcular la posición y orientación relativa de la cámara para cada imagen como se observa en la *Figura 6.6*.

En principio cualquier objeto podría ser usado para la calibración aunque en la práctica se utiliza regularmente el tablero de ajedrez propuesto por Zhang [Gary Bradski et al., 2008]. La imagen del tablero utilizado se visualiza en la *Figura 6.5*.

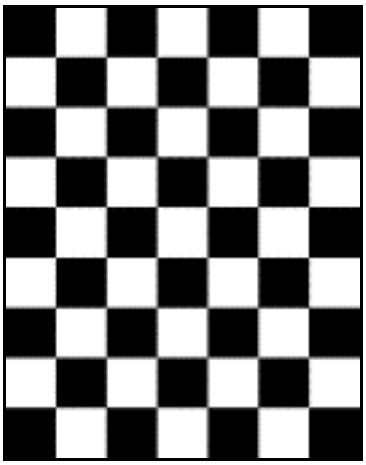


Figura 6.5 - Tablero de ajedrez usado en el proceso de calibración de las cámaras.

La calibración se debe realizar primero para cada cámara independientemente y luego para ambas cámaras juntas, las funciones provistas por EmguCV para esta tarea son

CalibrateCamera y *StereoCalibrate* respectivamente.

En el sistema propuesto se deben adquirir diez imágenes que contengan el tablero de ajedrez para poder iniciar el proceso de calibración, éste valor es ajustable mediante la modificación de una constante del sistema.

Con la calibración realizada se obtienen los parámetros intrínsecos y extrínsecos, matriz de rotación y vector de traslación.

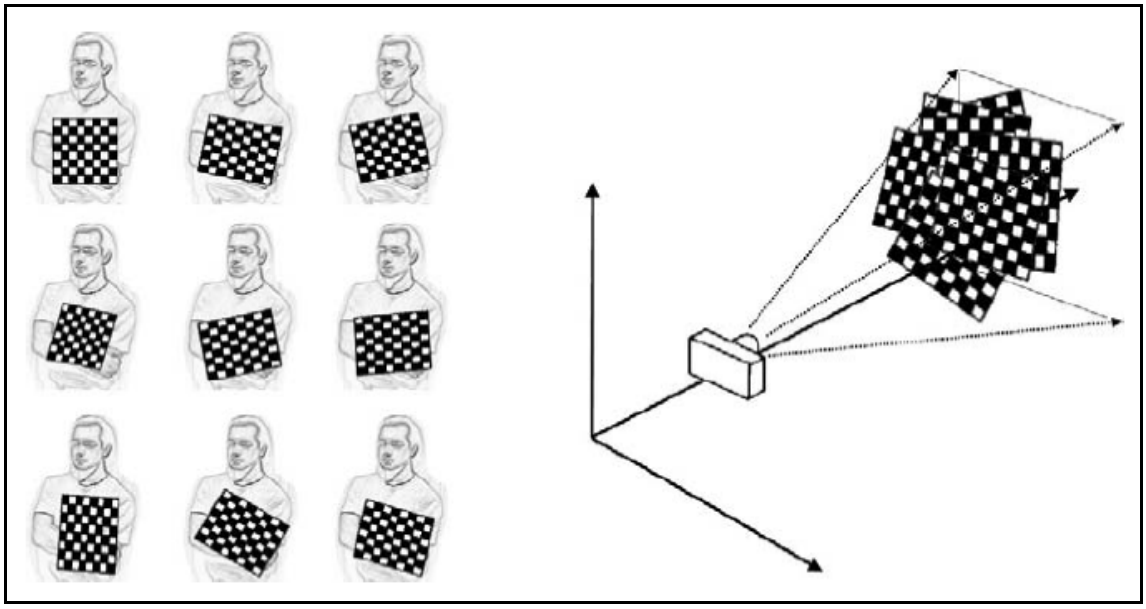


Figura 6.6 - Imágenes de un tablero de ajedrez observadas desde distintas posiciones proveen la información necesaria para obtener las posiciones relativas de las imágenes con respecto a la cámara [Gary Bradski et al., 2008].

En la *Figura 6.7* se observan una serie de imágenes correspondientes a la detección del patrón desde la perspectiva de cada cámara.

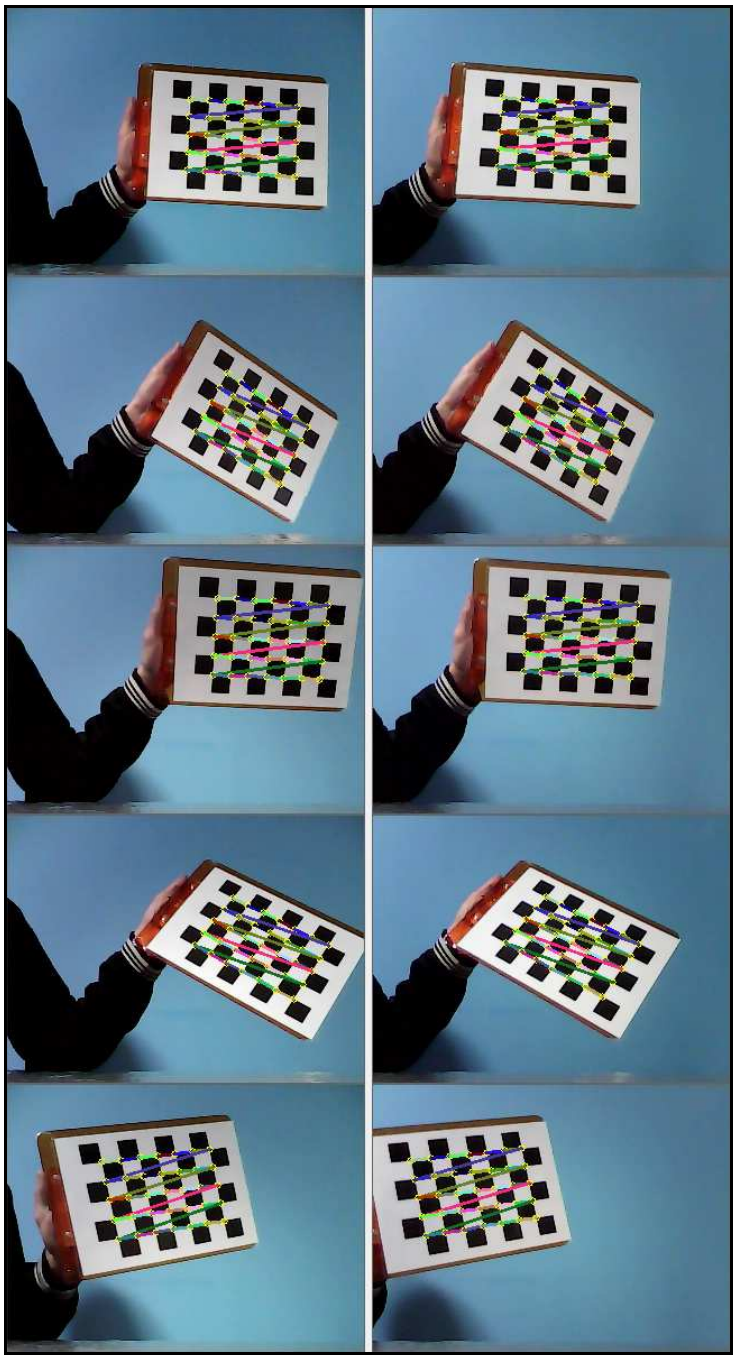


Figura 6.7 - Ejemplos de detección del patrón para ambas cámaras.

El segundo paso consiste en realizar el proceso de rectificación mediante la función *cvStereoRectify* provista por EmguCV para dejar ambas imágenes alineadas.

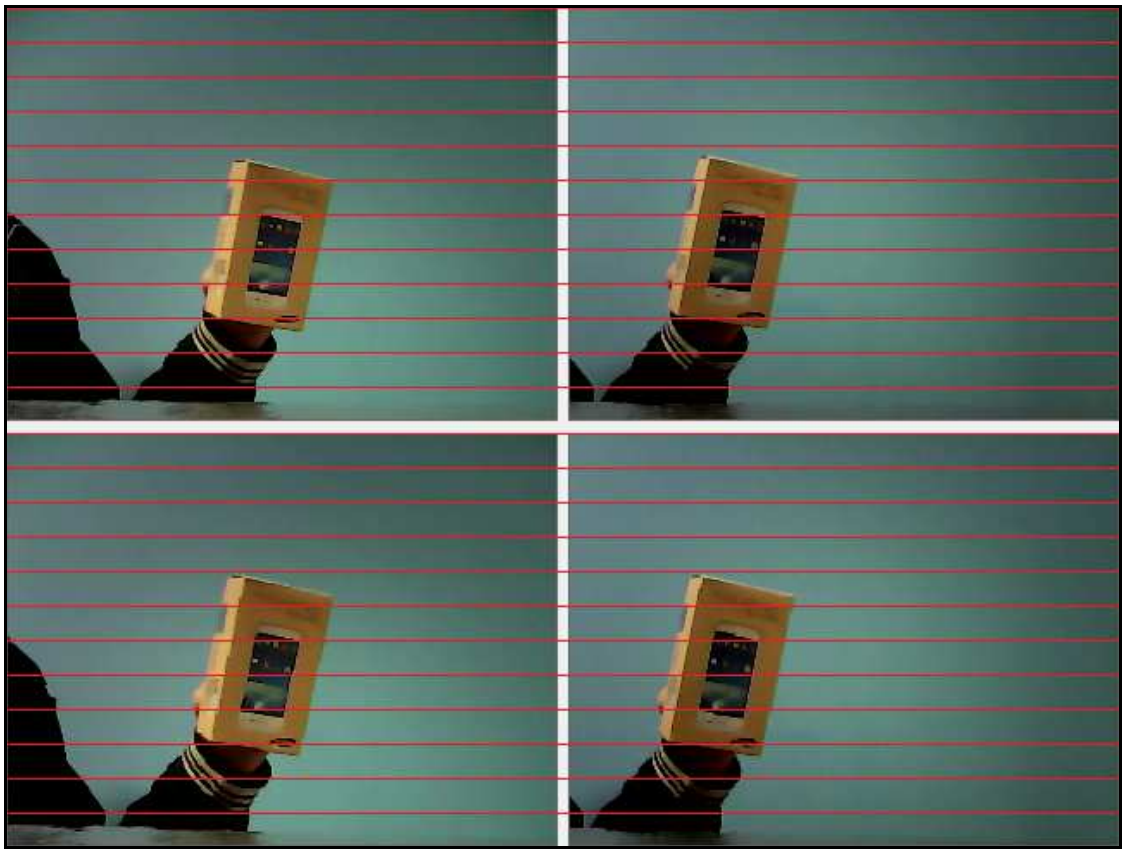


Figura 6.8 - Imágenes sin rectificar (arriba) y luego de la rectificación (abajo).

Por último, sobre las imágenes rectificadas se realiza la búsqueda de correspondencias mediante la función *FindStereoCorrespondence* de EmguCV y se construye el mapa de disparidad correspondiente.

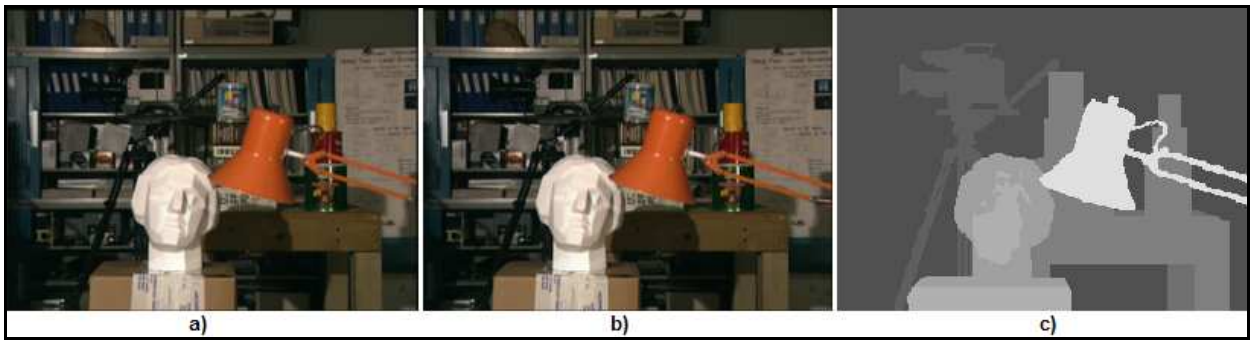


Figura 6.9 - Imágenes de ejemplo tomadas por dos cámaras (a y b). Mapa de disparidad calculado desde las imágenes (c) [Daniel Scharstein et al., 2002].

El resultado de la calibración puede ser exportado para su posterior uso desde la aplicación principal.

Disposición de controles en pantalla

La tarea de calibración se realiza desde una única pantalla observable en la *Figura 6.10*.
A continuación se detalla el uso de los controles de la pantalla:

Cam Izq / **Cam Der**: Permite seleccionar entre las cámaras detectadas por el sistema para ser utilizadas cómo cámara izquierda y cámara derecha respectivamente. Las imágenes sin procesar capturadas por las cámaras se visualizan en los controles ubicados debajo de la selección de cámaras.

Preview Cámara Izquierda / **Preview Cámara Derecha**: Visualizan las imágenes capturadas por las cámaras durante su procesamiento.

Disparity Map Control: Permite la modificación de los parámetros utilizados en el cálculo del mapa de disparidad.

Disparity Map: Visualiza el mapa de disparidad construido.

Output: Detalla las tareas que va realizando el sistema.

Iniciar: Inicializa las variables para comenzar con la tarea de calibración de cámaras.

Calibrar CamL: Inicia el proceso de calibración de la cámara izquierda. Para capturar la imagen visualizada se debe presionar la tecla "*F2*" y si corresponde al tablero de ajedrez es almacenada temporalmente. El proceso debe repetirse hasta obtener diez imágenes, momento en el cual comienza la calibración de la cámara.

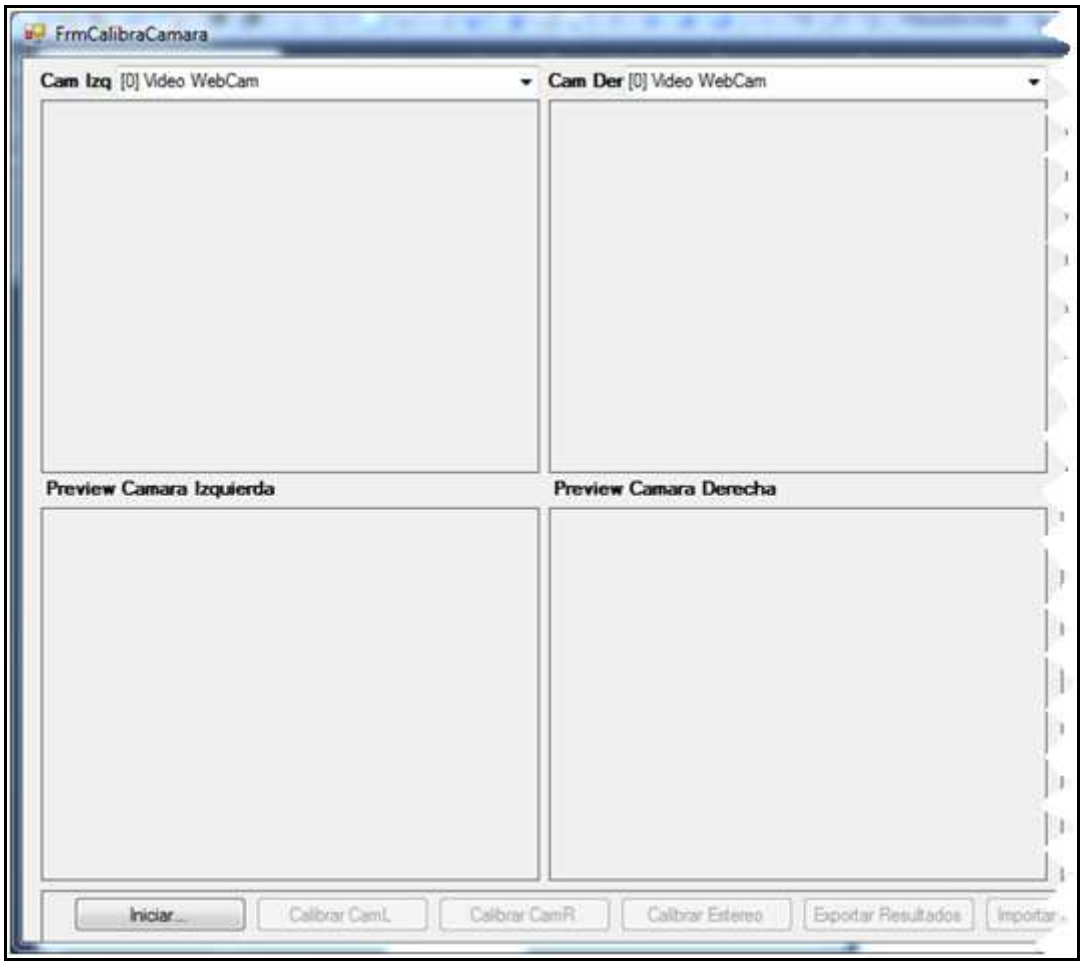
Calibrar CamR: Inicia el proceso de calibración de la cámara derecha. Para capturar la imagen visualizada se debe presionar la tecla "*F2*" y si corresponde al tablero de ajedrez es almacenada temporalmente. El proceso debe repetirse hasta obtener diez imágenes, momento en el cual comienza la calibración de la cámara.

Calibrar Estéreo: Inicia el proceso de calibración de las cámaras como sistema estéreo. Este proceso puede comenzarse si previamente se realizó la calibración individual de las cámaras. Para capturar las imágenes visualizadas por ambas cámaras se debe presionar la tecla "*F2*" y si en ambas cámaras se visualiza el tablero de ajedrez entonces las imágenes son almacenadas temporalmente. El proceso debe repetirse hasta obtener diez imágenes, momento en el cual comienza la calibración de las cámaras.

Exportar Resultados: Exporta los resultados de la calibración en el formato *.xml* para su posterior uso en la aplicación principal. Además exporta las imágenes utilizadas en el proceso de calibración en formato *.jpg*. El directorio de exportación corresponde a "*Exportaciones*" y se ubica dentro del directorio de trabajo de la aplicación. El resultado exportado tendrá validez siempre que la disposición física de las cámaras no varíe.

Importar Resultados: Importa los parámetros de calibración de las cámaras previamente almacenados en el directorio "*Exportaciones*".

Cerrar: Cierra la pantalla.



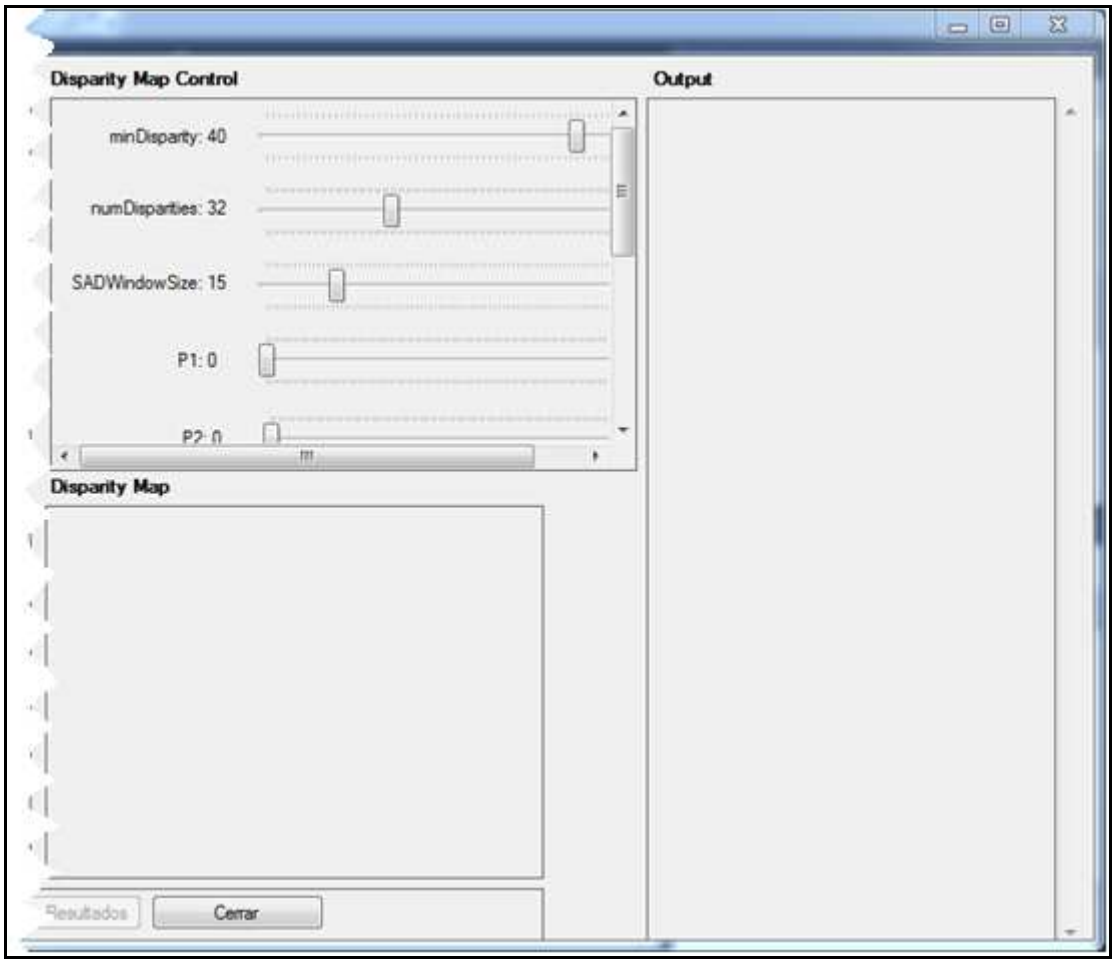


Figura 6.10 - Pantalla Calibración Cámaras.

6.2.1.2. Calibración de Color de Piel

El objetivo del módulo consiste en obtener los valores que representen el color de piel del usuario dentro de la escena para su posterior uso en la segmentación de la mano en la aplicación principal.

Espacio de Color YCbCr

Las primeras pruebas realizadas con el sistema determinaron la dificultad de segmentar correctamente la mano con la información proveniente del sistema estéreo. Por este motivo se decide reforzar la segmentación mediante el uso del color de la piel lo que mejora notablemente los resultados obtenidos al principio.

El uso del color en el área de la visión por computadora es una herramienta poderosa para identificar y extraer objetos en una escena. Los principales motivos de ésta afirmación son,

en primer lugar que el color es un descriptor poderoso de la imagen y ayuda a una identificación más fácil de los objetos, y en segundo lugar a que los seres humanos pueden diferenciar entre miles de tonalidades de color mientras que sólo pueden distinguir entre cerca de dos docenas de niveles de grises [Gonzalez et al., 2002]

Las características que suelen usarse para describir a los colores son los valores de Brillo, Tono y Saturación [Gonzalez et al., 2002]:

- El Brillo consiste en la luminosidad / oscuridad relativa del color y se puede expresar como un porcentaje siendo 0% el color negro y 100 % el color blanco.

- El Tono se denomina al color transmitido a través del objeto. Se mide en un ángulo entre 0º y 360º. Se identifica con el nombre del color como por ejemplo rojo.

- La Saturación o cromatismo consiste en la fuerza o pureza del color, es decir la proporción de blanco en proporción al tono. Se mide con un porcentaje siendo 0% gris y 100% saturación completa.

El Tono y la Saturación definen la Cromaticidad por lo que un color puede ser caracterizado por el Brillo y Cromaticidad.

Las imágenes capturadas por las cámaras web corresponden al espacio de color RGB. Este espacio de color tiene como desventaja que es fácilmente influenciado por variaciones de la luz y existencia de sombras, por este motivo conviene convertir a otro espacio de color que sea menos sensible a estos cambios.

Dos espacios de color muy utilizados son [Jack, 2008]:

- HSI de sus siglas en inglés (Hue, Saturation Intensity), que describe el color con valores intuitivos, basados en cómo las personas perciben el color, donde el valor Tinta define al color dominante de un área, la Saturación mide el colorido del área en relación a su brillo y la Intensidad se relaciona a la luminancia del color. La transformación entre RGB y HSI es no lineal

- YCbCr, donde el color se define en términos de luminancia y crominancia. El valor Y determina la luminancia y los valores Cb y Cr determinan el tono del color, donde Cb indica la ubicación del color en una escala entre el azul y el amarillo y Cr lo ubica

entre el rojo y verde. La transformación entre RGB y YCbCr es lineal.

Se opta por trabajar en el espacio de color YCbCr por su simplicidad en la transformación y porque la separación explícita del color en valores de crominancia y luminancia lo hace atractivo para la tarea de detección del color debido a que se presenta más robustez frente a cambios de luz [Jack, 2008].

La conversión de RGB a YCbCr se puede realizar mediante la fórmula de la *Figura 6.11* donde los valores R,G y B oscilan en el rango [0, 1], el valor Y en el rango [16, 235] y los valores Cb y Cr en el rango [16, 240].

$$\begin{bmatrix} Y \\ C_b \\ C_r \end{bmatrix} = \begin{pmatrix} 65.481 & 128.553 & 24.966 \\ -39.797 & -74.203 & 112 \\ 112 & -93.786 & -18.214 \end{pmatrix} * \begin{bmatrix} R \\ G \\ B \end{bmatrix} + \begin{bmatrix} 16 \\ 128 \\ 128 \end{bmatrix}$$

Figura 6.11 - Fórmula de conversión del espacio RGB al espacio YCbCr.

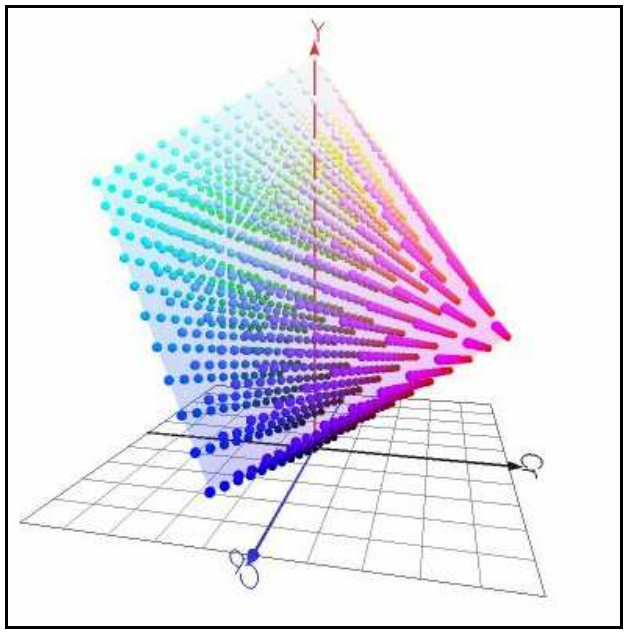


Figura 6.12 - Espacio de color YCbCr.

Disposición de controles en pantalla

El fin del módulo es procesar una imagen que contenga una porción de piel del usuario, convertirla al espacio de color YCbCr y luego obtener los valores promedios Cb y Cr que identifiquen el color piel, en la *Figura 6.14* se visualiza un ejemplo del proceso.

La tarea se realiza desde una única pantalla observable en la *Figura 6.13*.
A continuación se detalla el uso de los controles de la pantalla:

Imagen Original: Visualiza la imagen capturada por la cámara y muestra un área rectangular resaltada en el centro de la imagen que corresponde al sector que va a ser considerado para la calibración del color visualizado en el control "*Template a Calibrar*".

Imagen Procesada: Visualiza la imagen capturada por la cámara luego del proceso de calibración. Mediante los controles "*Ver YCbCr*" y "*Ver Piel*" se puede cambiar entre ver la imagen en el espacio de color YCbCr o ver la porción de imagen que corresponde al color piel respectivamente.

Template a Calibrar: Visualiza el área rectangular de la imagen antes de la calibración, luego de la conversión al espacio de color YCbCr y luego de aplicar el filtro por color de piel.

Histograma YCbCr: Visualiza la distribución de los canales Cb y Cr luego de la conversión de la imagen original al espacio de color YCbCr.
Los controles "*Cb Min/Max*" y "*Cr Min/Max*" permiten modificar los valores promedios obtenidos para ajustar la porción de la imagen que corresponde a la piel.

Calibrar: Inicia el proceso.

Recalcular Histograma: Vuelve a calcular el histograma para los valores configurados en los controles "*Cb Min/Max*" y "*Cr Min/Max*"

Confirmar: Cierra la pantalla devolviendo los valores de calibración obtenidos para ser utilizados por la aplicación principal.

Cerrar: Cierra la pantalla sin confirmar los resultados.

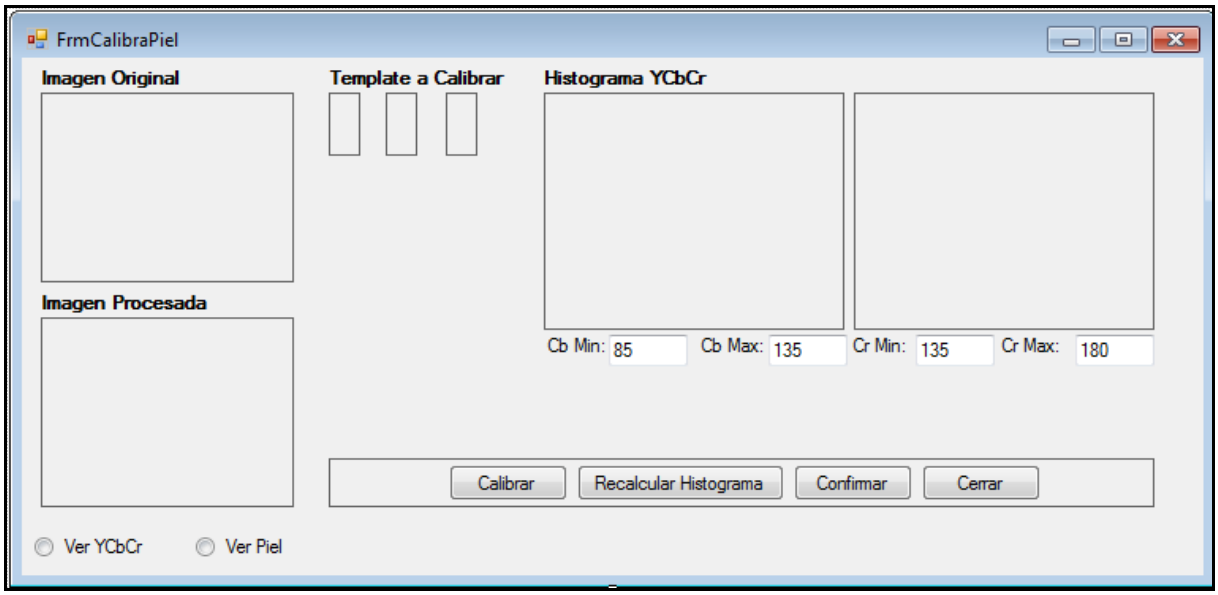


Figura 6.13 - Pantalla Calibración Piel.

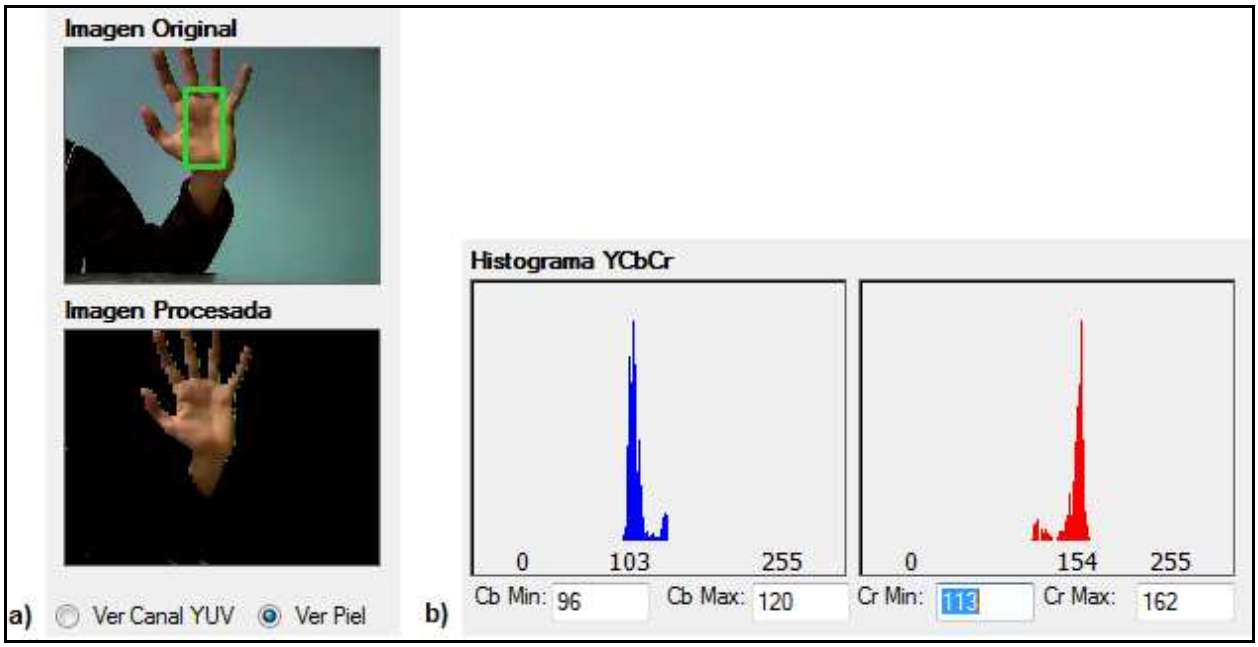


Figura 6.14 - Calibración de Piel. Imagen Original y Piel detectada (a). Histograma con los valores de distribución de los canales Cb y Cr (b)

6.2.1.3. Entrenamiento de Red Neuronal

El objetivo del módulo consiste en crear una red neuronal que permita reconocer los gestos y su posterior exportación para el uso desde la aplicación principal.

Se opta por trabajar con una red Multicapa, un tipo de red Pre Alimentada entrenada por el algoritmo de Propagación Hacia Atrás. Éste tipo de redes es sencillo de implementar y los

resultados de reconocimiento son aceptables.

En el sistema propuesto se reconocerán 3 gestos y cada uno deberá entrenarse con 80 muestras del gesto, estos valores son ajustables mediante la modificación de dos constantes del sistema. En la sección *6.3 Resultados* se explican las pruebas realizadas para determinar el número de muestras usadas en el entrenamiento.

La *Figura 6.15* presenta los 3 gestos reconocidos por el sistema y su esqueleto correspondiente. Para el sistema los gestos se denominan "*Gesto 0*" *Figura 6.15 (a)*, "*Gesto 1*" *Figura 6.15 (b)* y "*Gesto 2*" *Figura 6.15 (c).*

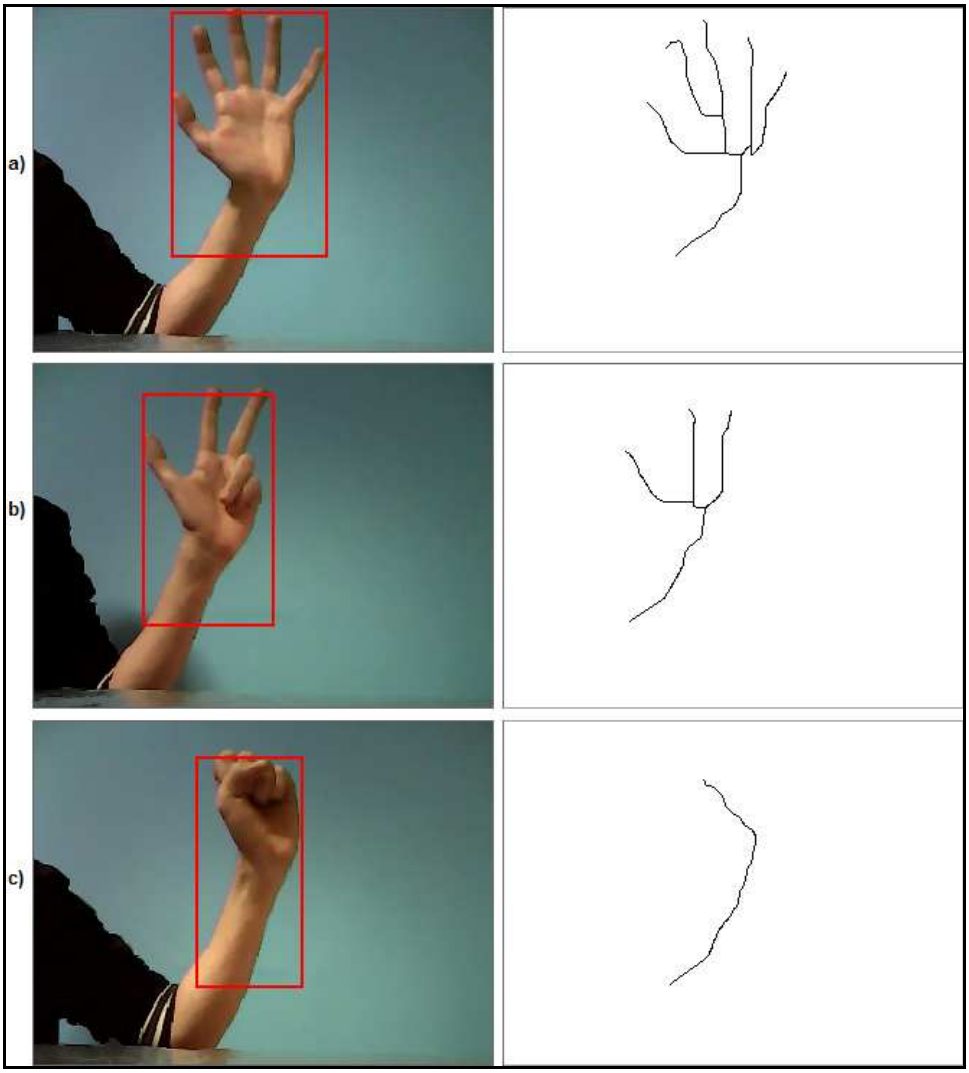


Figura 6.15 - Gestos entrenados en el sistema y el esqueleto correspondiente.

La red neuronal construida consta de una capa inicial formada por 150 neuronas, una capa intermedia formada por 5 neuronas y una capa de salida formada por 3 neuronas que corresponden a los gestos a reconocer.

Para simplificar la complejidad en el entrenamiento de la red neuronal se realiza un pre procesado de la imagen esqueleto para obtener una imagen equivalente con tamaño de 15x15 píxeles, la cantidad de píxeles que corresponde a las 150 neuronas de la capa inicial.

El pre procesamiento consta de dos pasos visualizados en la *Figura 6.16*:

- Redimensionar el esqueleto a un tamaño de 150x150 píxeles. Primero se recorta la imagen para obtener una imagen formada únicamente por el esqueleto, luego se ubica el esqueleto centrado en una imagen vacía de tamaño 150x150 píxeles.

- Transformar la nueva imagen a un tamaño de 15x15 píxeles. La imagen resultado del paso anterior se sitúa sobre una cuadrilla con tamaño 15x15 y cada área de la cuadrilla que tenga un píxel del esqueleto se rellena.

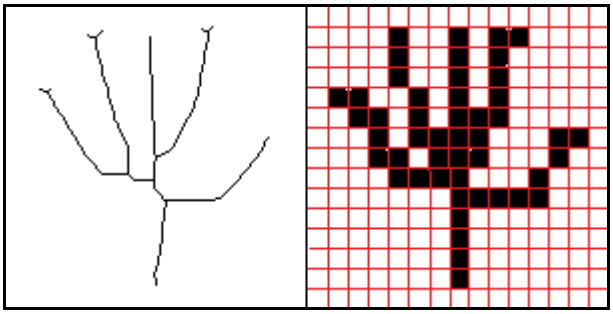


Figura 6.16 - Esqueleto redimensionado a un tamaño de 150x150 píxeles (izquierda). Esqueleto transformado a un tamaño de 15x15 píxeles (derecha)

Disposición de controles en pantalla

La tarea se realiza desde una única pantalla observable en la *Figura 6.17*.
A continuación se detalla el uso de los controles de la pantalla:

Cam Izq / **Cam Der**: Permite seleccionar entre las cámaras detectadas por el sistema para ser utilizadas cómo cámara izquierda y cámara derecha respectivamente. Las imágenes sin procesar capturadas por las cámaras se visualizan en los controles ubicados debajo de la selección de cámaras.

Preview Cámara Izquierda / **Preview Cámara Derecha**: Visualizan las imágenes capturadas por las cámaras durante su procesamiento.

Output: Detalla las tareas que va realizando el sistema.

Disparity Map: Visualiza el esqueleto de la imagen y cada paso previo para su construcción si se selecciona desde el control "*Visualizar*".

Visualizar: Permite visualizar cada una de las etapas en la segmentación de la mano.

Valores YCbCr: Permite variar los valores promedios Cb y Cr utilizados para detectar la porción de la imagen que corresponde a la piel.

Gestos: Muestra el listado de gestos para los cuales se deberá obtener las muestras correspondientes que luego serán entrenadas en la red neuronal.

Iniciar: Inicializa las variables y construye el sistema estéreo utilizado para segmentar la mano que luego será entrenada por la red neuronal.

Capturar Gesto: Captura la imagen y la almacena como muestra para el gesto seleccionado.

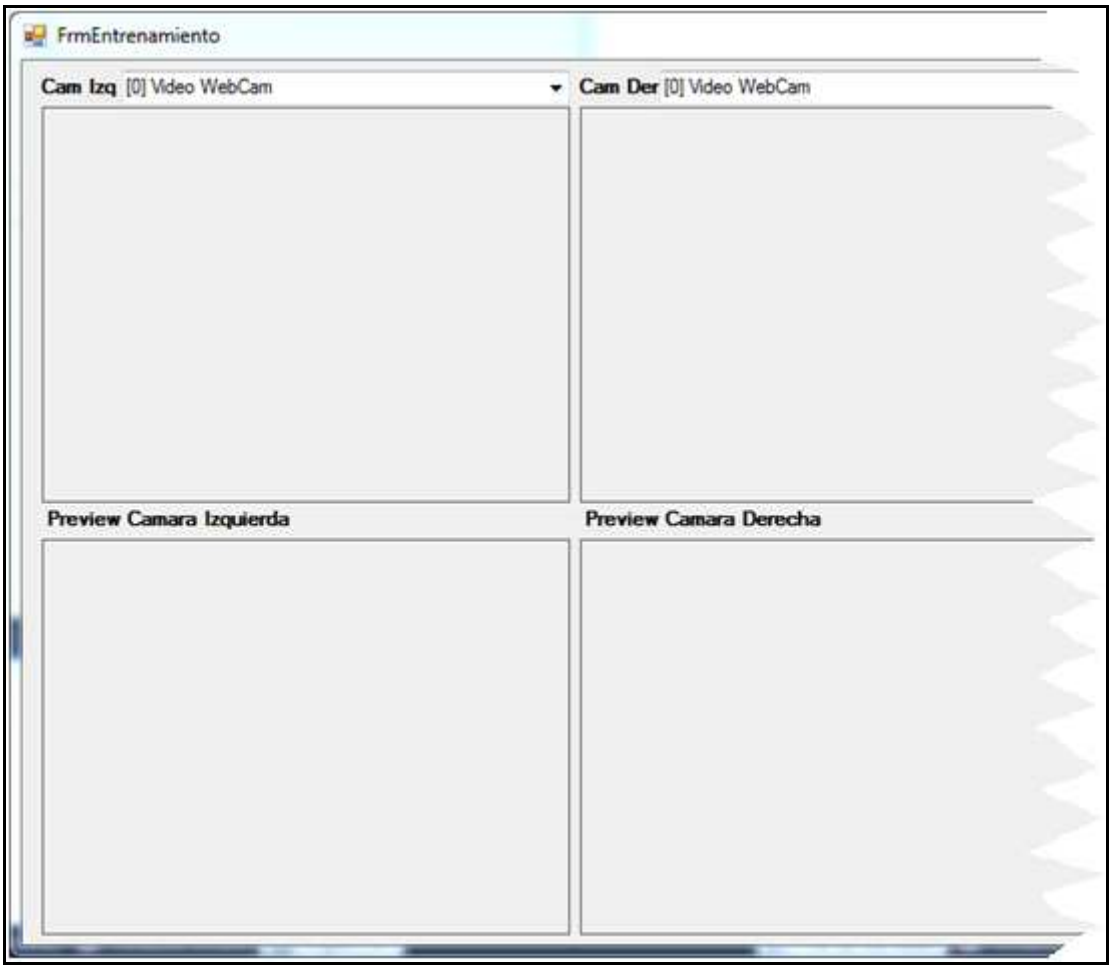
Importar Gestos: Permite cargar un conjunto de muestras previamente almacenado en la carpeta "*ExportacionesANN*" del directorio de trabajo de la aplicación. Las imágenes deben tener extensión *.png* y deben existir tantas imágenes como gestos y muestras se hayan configurado en el sistema.

Entrenar Red: Inicia el proceso de entrenamiento de la red neuronal. Es necesario que se hayan obtenido todas las muestras para los gestos para comenzar el proceso.

Exportar Red: Exporta la red neuronal en formato *.txt* para su posterior uso en la aplicación principal. Además exporta las muestras utilizadas en el proceso de entrenamiento en formato *.png*. El directorio de exportación corresponde a "*ExportacionesANN*" y se ubica dentro del directorio de trabajo de la aplicación.

Testear Gesto: Verifica la imagen en la red neuronal para determinar si corresponde a un gesto válido.

Cerrar: Cierra la pantalla.


FrmEntrenamiento
Cam Izq [0] Video WebCam
Cam Der [0] Video WebCam
Preview Camara Izquierda
Preview Camara Derecha

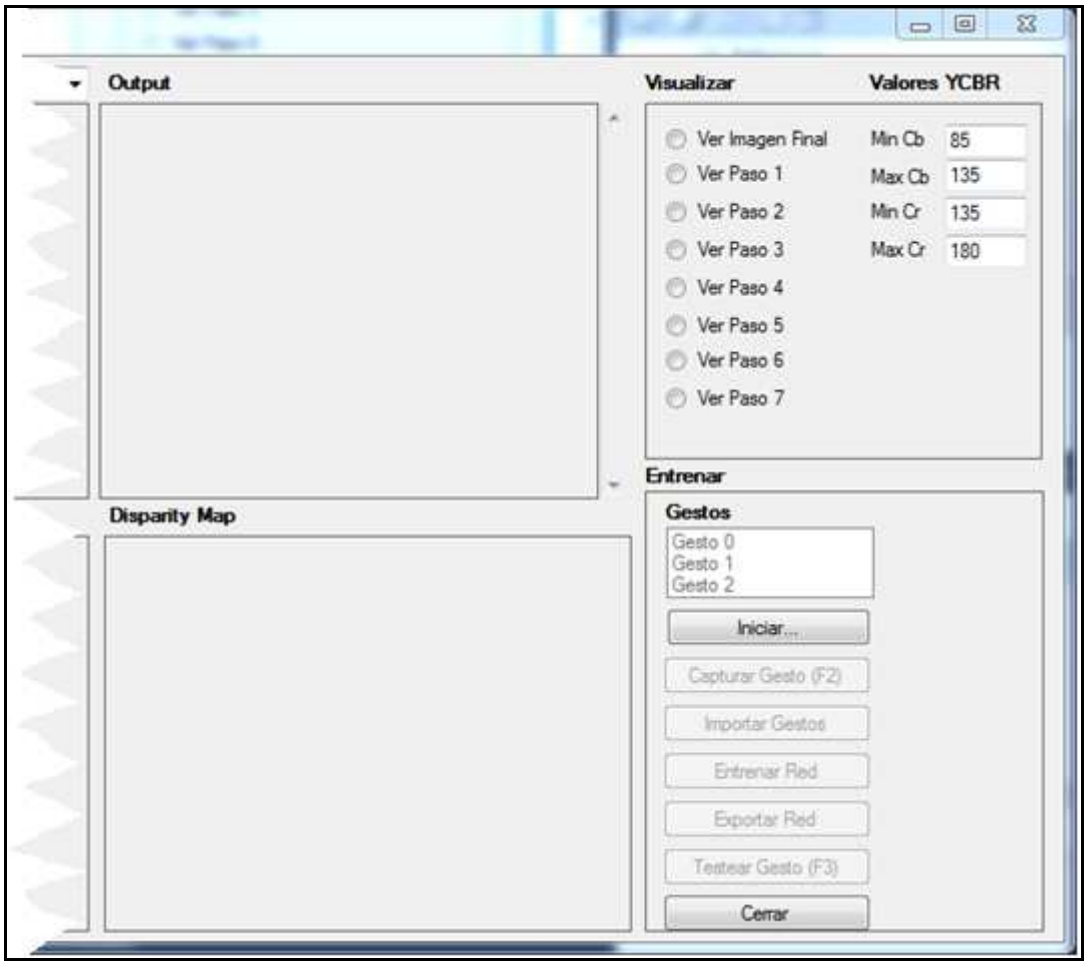


Figura 6.17 - Pantalla Entrenamiento Red Neuronal.

6.2.2. Módulos de Sistema

La aplicación principal captura las imágenes de las cámaras sin procesar y determina si se encuentra presente un gesto. Para lograr esta tarea son necesarios una serie de pasos que consisten en la obtención de un mapa de disparidad, la obtención y depuración de la mano y la posterior evaluación en la red neuronal. Desde la aplicación además se permite el ingreso a los módulos de parametrización del sistema.

Disposición de controles en pantalla

La tarea se realiza desde una única pantalla observable en la *Figura 6.18*.
A continuación se detalla el uso de los controles de la pantalla:

Cam Izq / **Cam Der**: Permite seleccionar entre las cámaras detectadas por el sistema para ser utilizadas cómo cámara izquierda y cámara derecha respectivamente. Las imágenes sin

procesar capturadas por las cámaras se visualizan en los controles ubicados debajo de la selección de cámaras, "*Cámara Izquierda*" y "*Cámara Derecha*" respectivamente.

Disparity Map: Visualiza el esqueleto de la imagen y cada paso previo para su construcción si se selecciona desde el control "*Visualizar*".

Output: Detalla las tareas que va realizando el sistema.

Visualizar: Permite visualizar cada una de las etapas en la segmentación de la mano.

Iniciar: Inicializa las variables, construye el sistema estéreo utilizado de acuerdo a los parámetros de calibración almacenados y carga la red neuronal previamente entrenada para poder reconocer si una imagen corresponde a un gesto válido del sistema.

Calibrar Cámara: Realiza el llamado al módulo "*Calibración de las Cámaras y Obtención del Sistema Estéreo*".

Calibrar Piel: Realiza el llamado al módulo "*Calibración de Color de Piel*".

Entrenar Red: Realiza el llamado al módulo "*Entrenamiento de Red Neuronal*".

Detectar Gesto: Permite verificar si la imagen visualizada en el control "*Disparity Map*" corresponde a un gesto válido para el sistema. El resultado se visualiza en el control "*Output*".

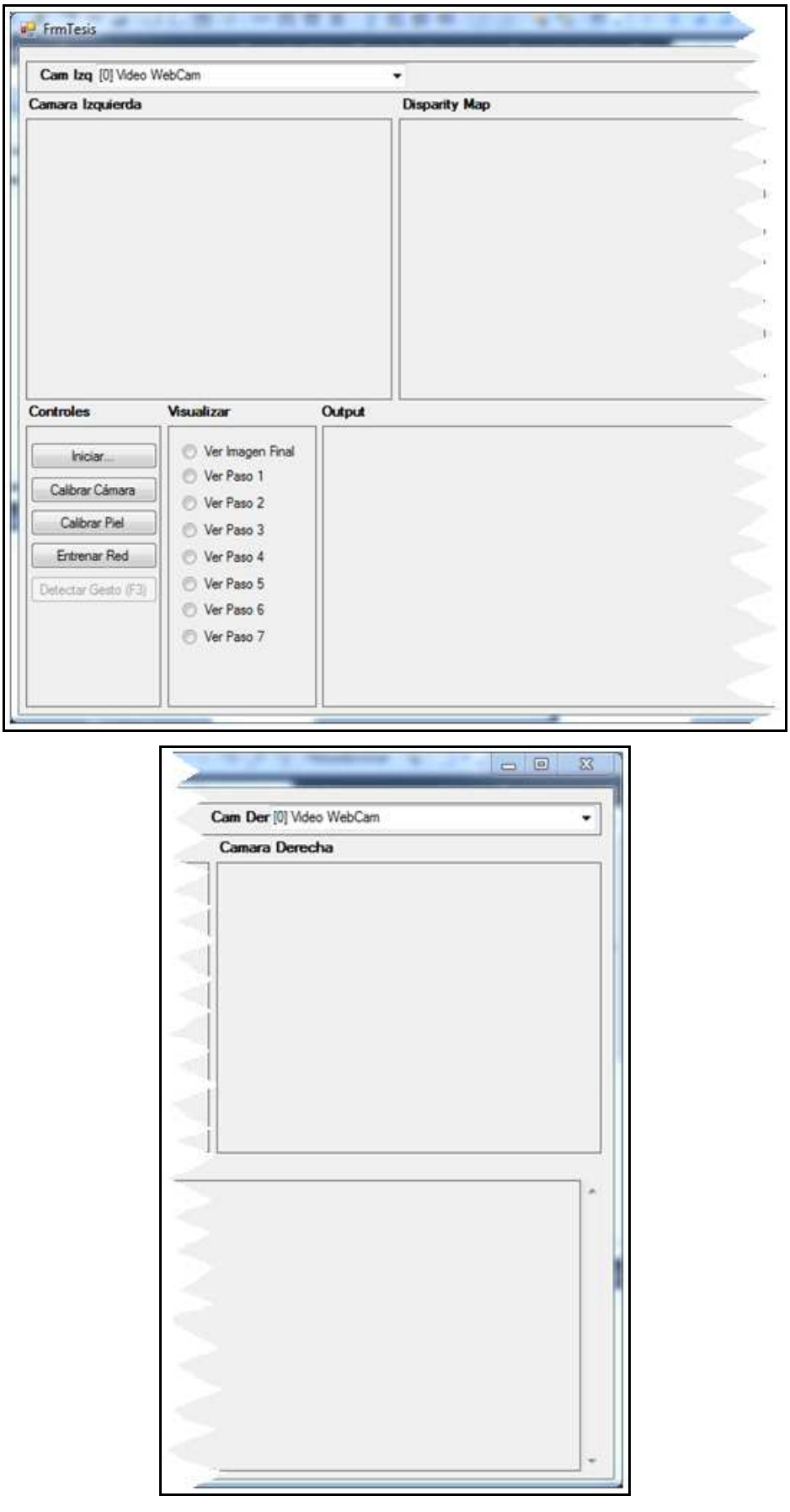


Figura 6.18 - Pantalla Principal.

6.2.2.1. Obtención de Imagen Dispar

Las imágenes obtenidas por las cámaras se rectifican utilizando los Parámetros de Calibración de las Cámaras y se obtiene un mapa de disparidad de la escena como se observa en la *Figura 6.19.*

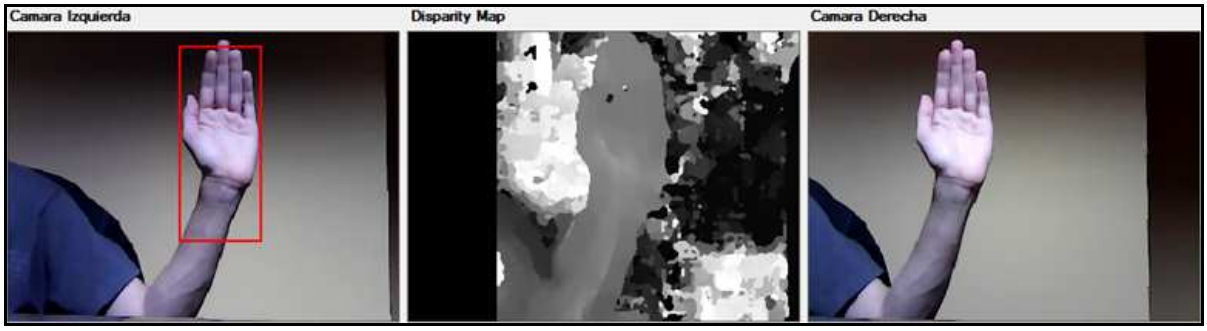


Figura 6.19 - Creación del mapa de disparidad de las imágenes.

6.2.2.2. Depuración de la Imagen

Se procesa el mapa de disparidad obtenido para quedarse con una imagen que contiene únicamente a la mano. A continuación se detallan cada una de las tareas del proceso de depuración y que pueden visualizarse mediante el control "*Visualizar*" observado en la *Figura 6.20*:

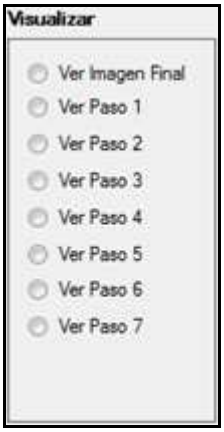


Figura 6.20 - Control que permite visualizar cada uno de los pasos involucrados en la depuración de la imagen.

- Paso 1: Corresponde al mapa de disparidad generado. *Figura 6.21 (b)*

- Paso 2: El mapa de disparidad se procesa para dejar visibles los objetos que se

encuentran aproximadamente a 1 metro de distancia de las cámaras. Cómo el color del mapa de disparidad es una noción de la distancia de los objetos, se determinó que filtrando los píxeles pertenecientes al rango de color 110 - 180 se obtienen resultados aceptables. A continuación se erosiona y dilata para quitar el ruido presente. *Figura 6.21 (c)*

- Paso 3: Se binariza la imagen y se refuerza el quitado de ruido eliminando las regiones pequeñas presentes en la imagen.

- Paso 4: Si hubiera más de una región se toma la de mayor tamaño y se rellenan los huecos para obtener una región sólida de color blanco. *Figura 6.21 (d)*

- Paso 5: Se procesa la imagen que dio origen al mapa de disparidad en busca de áreas con color piel. A continuación se aplica la operación AND entre la imagen que contiene áreas con color piel y la región sólida de color blanco. El resultado es reforzar mediante el uso de color la segmentación de la mano. *Figura 6.21 (e)*

- Paso 6: Se vuelven a rellenar los huecos que pudieran existir y se quitan las regiones pequeñas que pudieron haberse creado.

- Paso 7 / Imagen Final: Se aplica el algoritmo para la detección del esqueleto de la mano. *Figura 6.21 (f)*

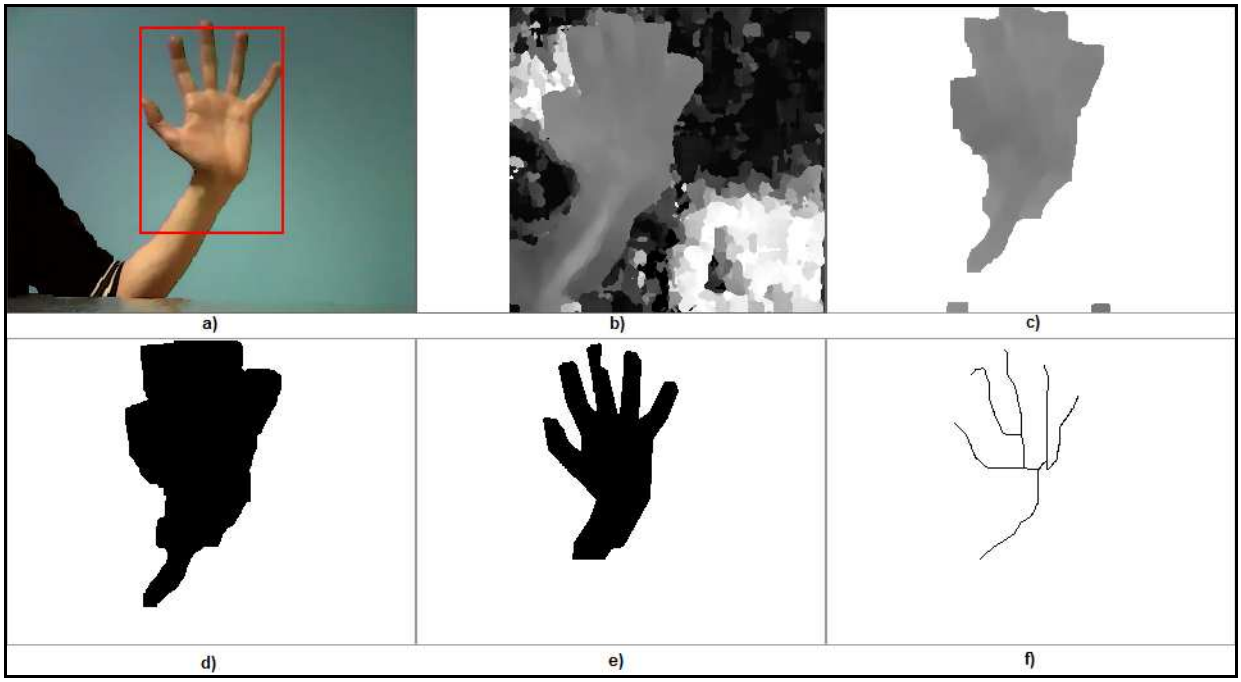


Figura 6.21 - Etapas de la detección de la mano. 1) Captura de la mano (a). 2) Construcción

del mapa de disparidad (b). 3) Segmentación del mapa de disparidad para los objetos próximos (c). 4) Binarización y depuración (d). 5) Refuerzo de la segmentación aplicando la técnica del color de piel (e). 6) Esqueletización de la mano obtenida (f).

6.2.2.3. Esqueletización

La mano binarizada se procesa para obtener el esqueleto. Para esto se implementa el algoritmo de Zhang Suen explicado la sección *4.2.2. Algoritmo Zhang-Suen*.

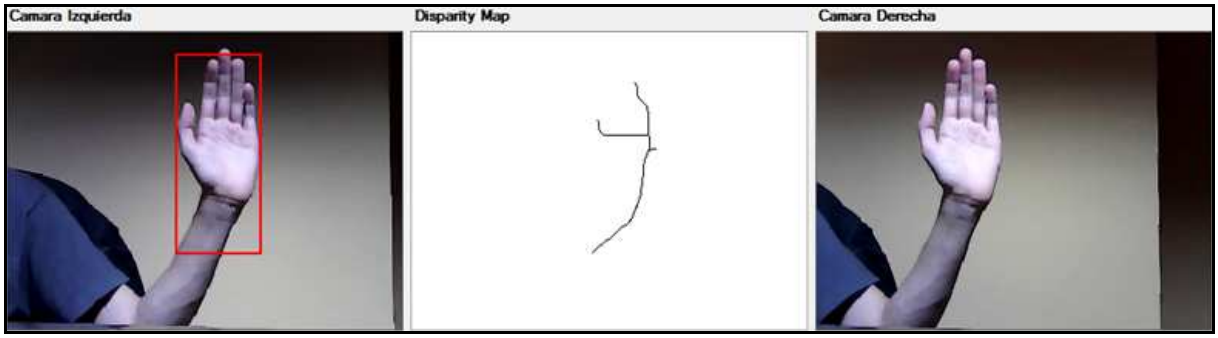


Figura 6.22 - Obtención del esqueleto de la mano.

6.2.2.4. Detección de Gesto

Se evalúa el esqueleto en la red neuronal y se obtiene si la imagen corresponde a un gesto válido en el sistema.

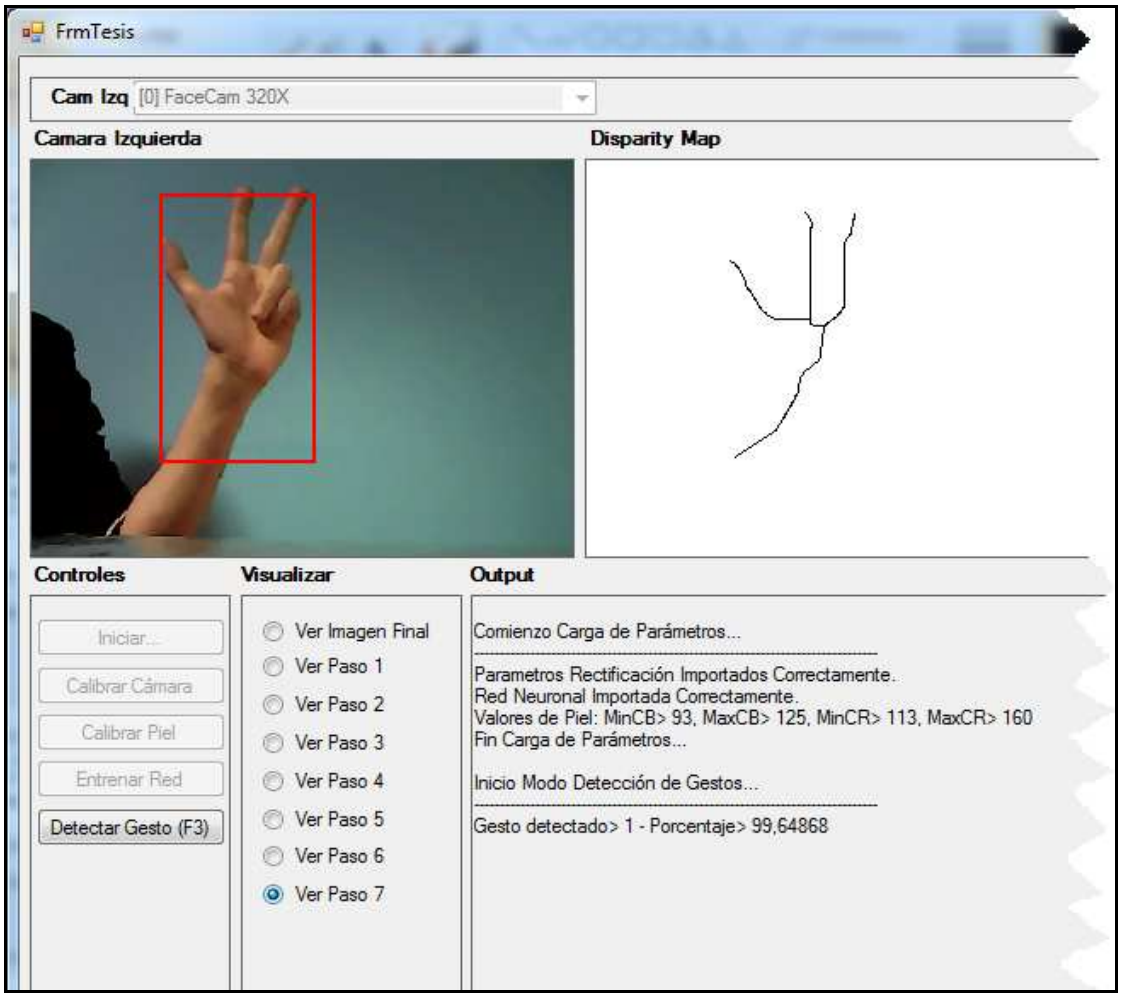


Figura 6.23 - Reconocimiento del gesto.

6.3. Resultados

Para evaluar el sistema construido se crean cuatro redes neuronales entrenadas con distintas cantidades de muestras para cada gesto:

- Red A: Cada gesto a entrenar consta de 10 muestras.
- Red B: Cada gesto a entrenar consta de 20 muestras.
- Red C: Cada gesto a entrenar consta de 40 muestras.
- Red D: Cada gesto a entrenar consta de 80 muestras (se ingresan dos veces las muestras de la Red C).

Luego se verifica la eficacia de cada red neuronal para reconocer los gestos, en la evaluación se utilizan 30 ejemplos de gestos que consisten en 10 ejemplos del *"Gesto 0"*, 10 ejemplos del *"Gesto 1"* y 10 ejemplos del *"Gesto 2".* Los ejemplos utilizados se visualizan en la sección *6.3.1. Figuras Utilizadas para el Entrenamiento de la Red Neuronal.*

A continuación se detalla el resultado de las pruebas indicando el porcentaje de detección de cada ejemplo evaluado.

	Red A	**Red B**	**Red C**	**Red D**
Gesto 0 - Ejemplo 1	100%	99,36%	100%	**100%**
Gesto 0 - Ejemplo 2	99,98%	97,87%	100%	**100%**
Gesto 0 - Ejemplo 3	84,98%	88,11%	100%	**100%**
Gesto 0 - Ejemplo 4	99,92%	99,57%	99,94%	**100%**
Gesto 0 - Ejemplo 5	100%	100%	100	**100%**
Gesto 0 - Ejemplo 6	100%	99,82%	97,73%	**100%**
Gesto 0 - Ejemplo 7	99,66%	99,43%	99,62%	**99,96%**
Gesto 0 - Ejemplo 8	sin detección	falsa detección Gesto 1	falsa detección Gesto 1	falsa detección Gesto 1
Gesto 0 - Ejemplo 9	93,67%	**99,98%**	89,59%	98,11%
Gesto 0 - Ejemplo 10	87,12%	falsa detección Gesto 1	98,74%	**99,09%**
Gesto 1 - Ejemplo 1	100%	89,12%	99,78%	**100%**
Gesto 1 - Ejemplo 2	99,60%	99,09%	**99,99%**	98,72%
Gesto 1 - Ejemplo 3	**99,95%**	99,91%	98,83%	96,30%
Gesto 1 - Ejemplo 4	99,69%	91,44%	98,36%	**100%**
Gesto 1 - Ejemplo 5	99,65%	99,54%	**100%**	99,66%
Gesto 1 - Ejemplo 6	99,71%	falsa detección Gesto 0	99,95%	**100%**
Gesto 1 - Ejemplo 7	100%	100%	100%	**100%**
Gesto 1 - Ejemplo 8	**100%**	98,52%	99,94%	93,47%
Gesto 1 - Ejemplo 9	100%	98,15%	**100%**	95,41%
Gesto 1 - Ejemplo 10	99,21%	**99,74%**	97,50%	82,85%
Gesto 2 - Ejemplo 1	97,65%	95,47%	99,35%	**99,51%**
Gesto 2 - Ejemplo 2	100%	99,98%	**100%**	99,90%
Gesto 2 - Ejemplo 3	**100%**	94,79%	99,93%	99,99%
Gesto 2 - Ejemplo 4	98,34%	99,73%	**99,99%**	99,94%
Gesto 2 - Ejemplo 5	98,39%	**99,88%**	99,49%	99,80%
Gesto 2 - Ejemplo 6	96,42%	99,66%	**99,78%**	99,72%

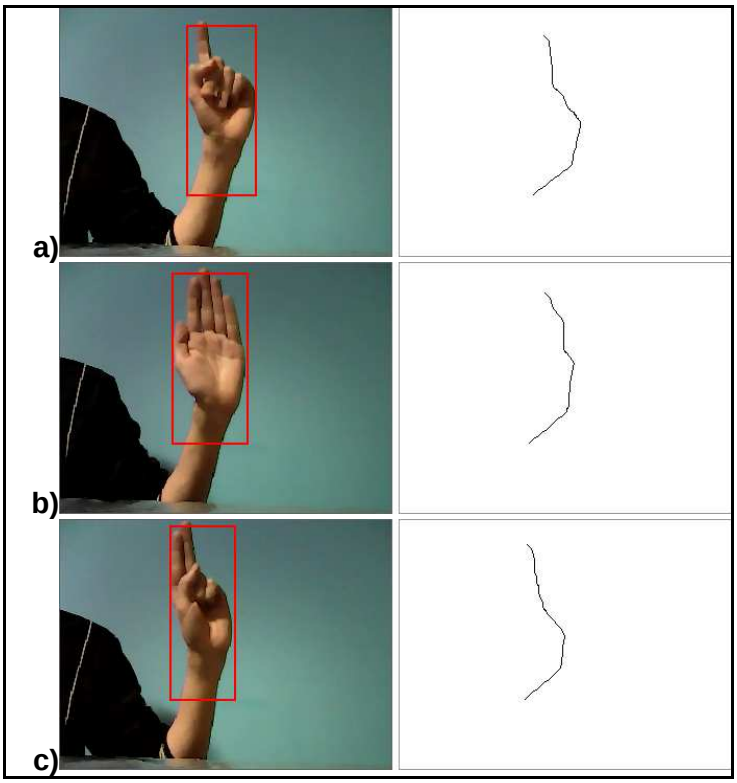
Gesto 2 - Ejemplo 7	100%	100%	100%	**100%**
Gesto 2 - Ejemplo 8	**99,89%**	90,28%	98,11%	99,02%
Gesto 2 - Ejemplo 9	sin detección	sin detección	falsa detección Gesto 1	**83,47%**
Gesto 2 - Ejemplo 10	**94,32%**	sin detección	sin detección	90,88%

Tabla 6.1 - Comparación de los resultados de reconocimiento.

Como se observa en la *Tabla 6.1*, a medida que se entrenan más muestras se obtienen mejores porcentajes de reconocimiento siendo la red neuronal D la que tiene resultados óptimos para reconocer los 3 gestos del sistema.

Un problema detectado en las pruebas se visualiza en la *Figura 6.24*. Esta dificultad está relacionada al problema de unicidad de los esqueletos explicado en la sección *4.3. Problemas comunes, al* existir varios gestos que producen el mismo esqueleto la red neuronal no es capaz de distinguirlos correctamente.

Otro Problema se relaciona con la estabilidad del esqueleto, la menor imperfección a la hora de segmentar la mano produce cambios bruscos del esqueleto que son difíciles de distinguir por la red neuronal.



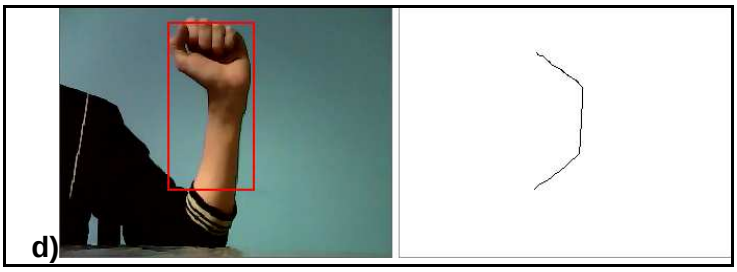


Figura 6.24 - Gestos con el mismo esqueleto.

6.3.1. Figuras Utilizadas para el Entrenamiento de la Red Neuronal

A continuación se presentan las muestras utilizadas para el entrenamiento de la red neuronal D, donde la *Figura 6.25* corresponde a las muestras utilizadas en el entrenamiento del *"Gesto 0"*, la *Figura 6.26* corresponde a las muestras utilizadas en el entrenamiento del *"Gesto 1"* y la *Figura 6.27* corresponde a las muestras utilizadas en el entrenamiento del *"Gesto 2"*.

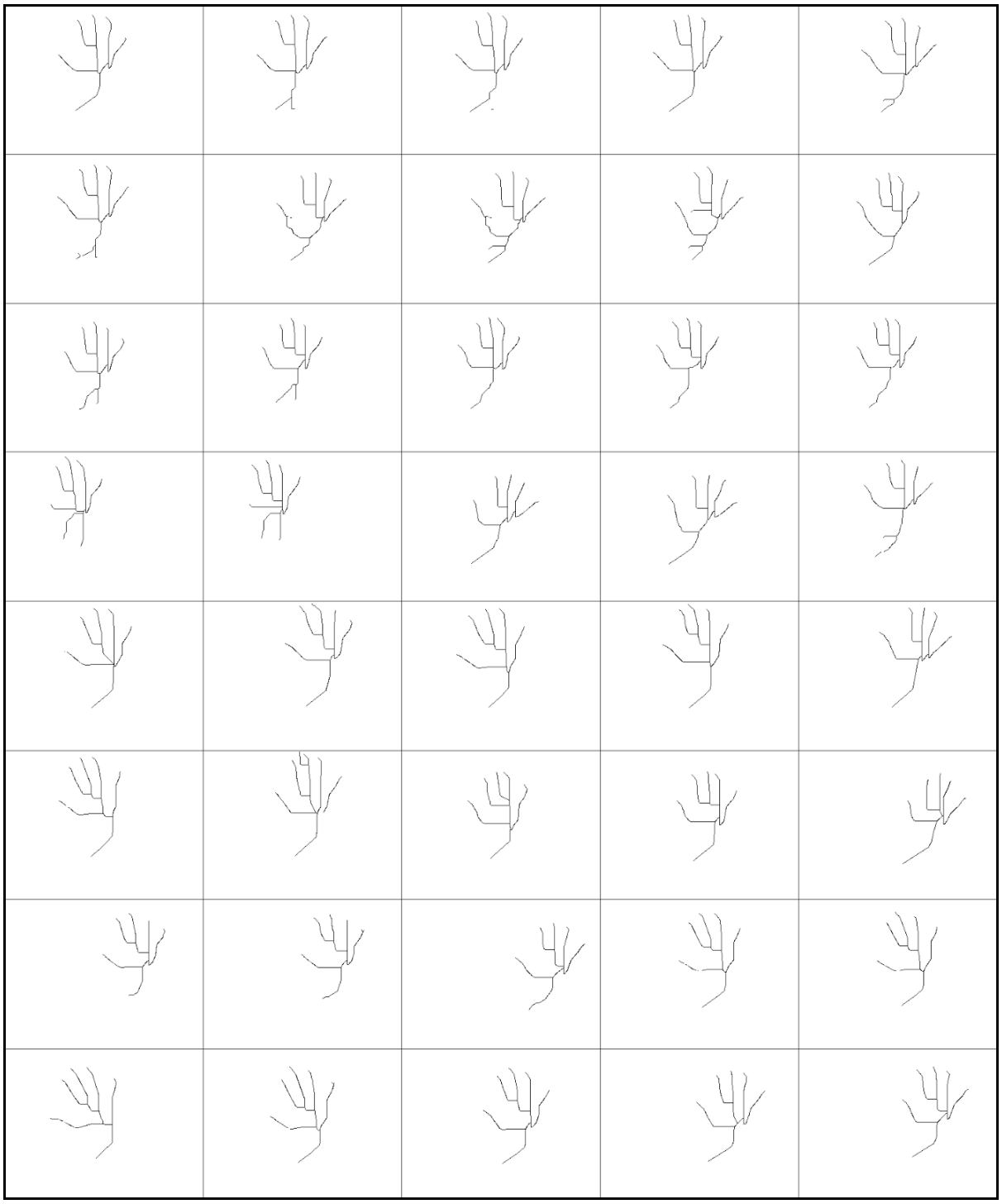


Figura 6.25 - Muestras utilizadas para el entrenamiento del “Gesto 0”. Al entrenamiento ingresan dos copias de cada una de las muestras visualizadas para completar las 80 muestras a entrenar.

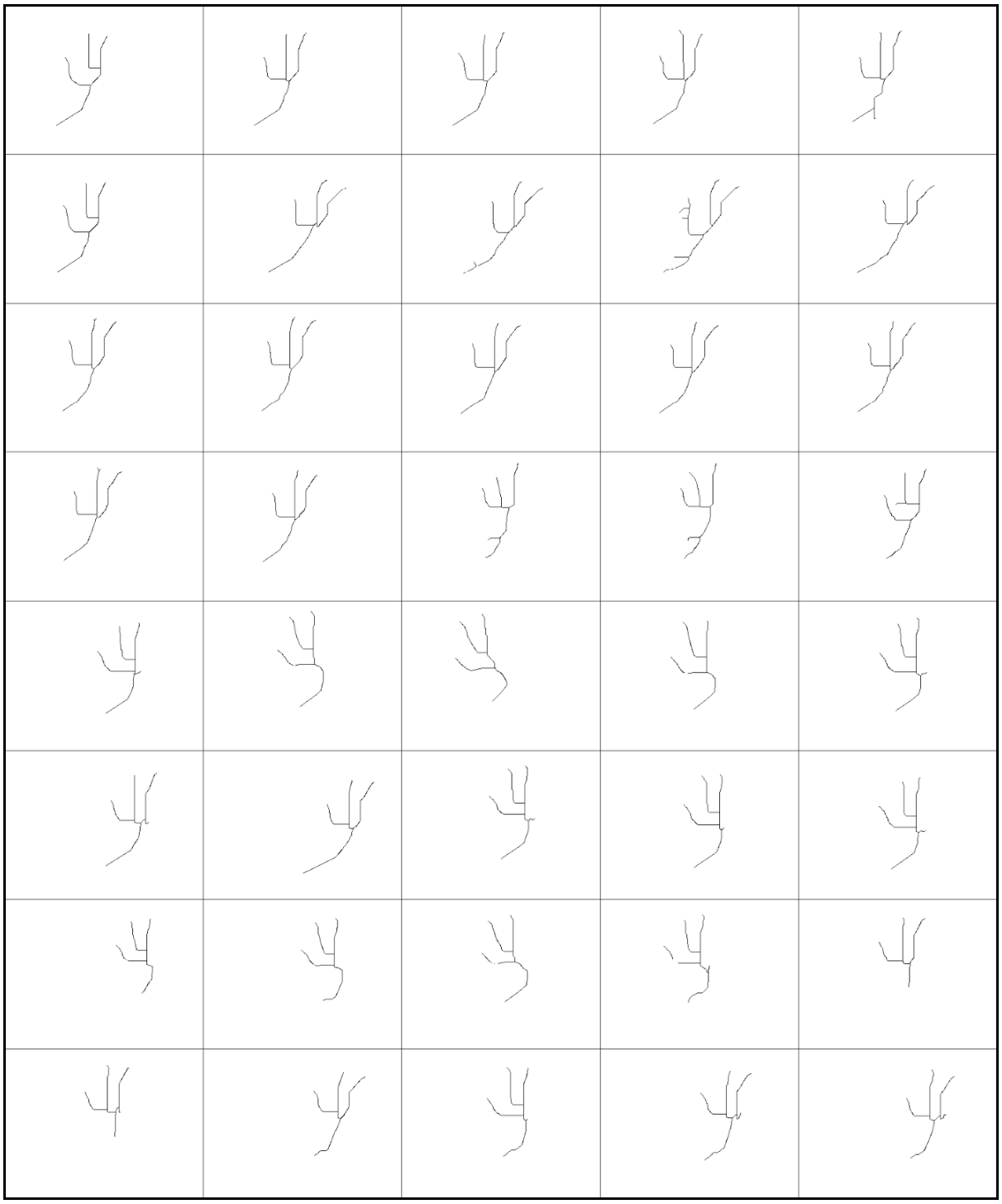


Figura 6.26 - Muestras utilizadas para el entrenamiento del "Gesto 1". Al entrenamiento ingresan dos copias de cada una de las muestras visualizadas para completar las 80 muestras a entrenar.

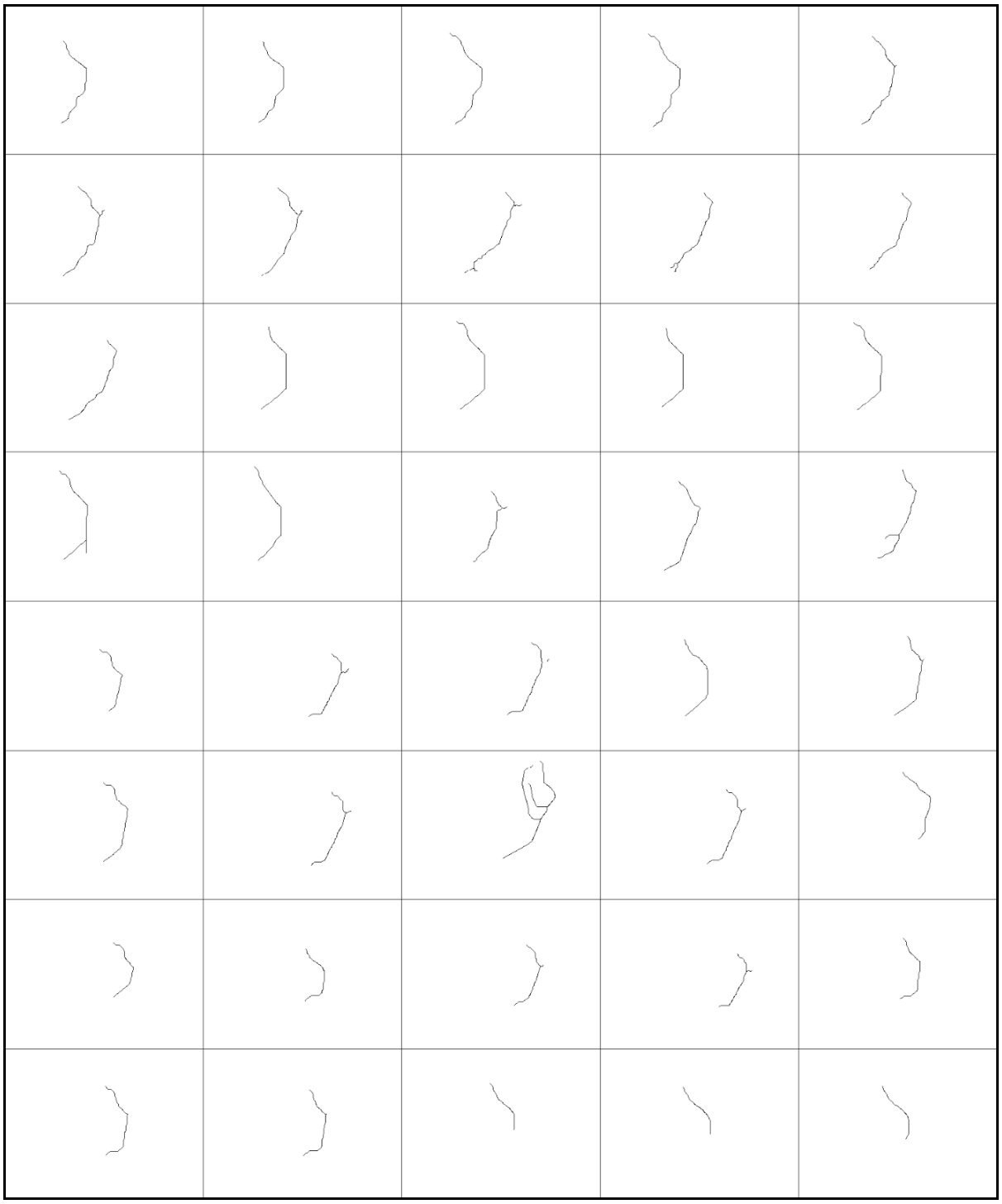
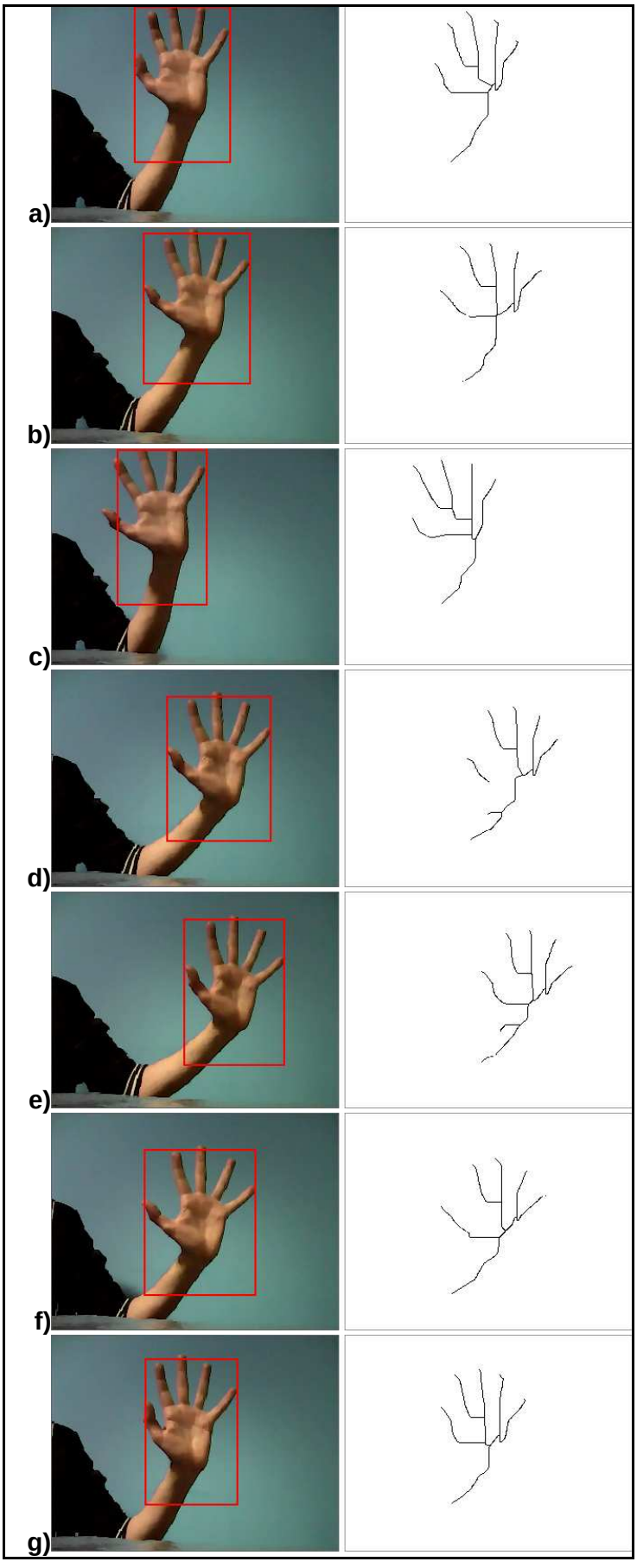


Figura 6.27 - Muestras utilizadas para el entrenamiento del "Gesto 2". Al entrenamiento ingresan dos copias de cada una de las muestras visualizadas para completar las 80 muestras a entrenar.

A continuación se presentan los 30 ejemplos de gestos utilizados en la evaluación de la red neuronal. La *Figura 6.28* corresponde a los ejemplos del *"Gesto 0"*, la *Figura 6.29* al *"Gesto 1"* y la *Figura 6.30* al *"Gesto 2"*.


a)
b)
c)
d)
e)
f)
g)

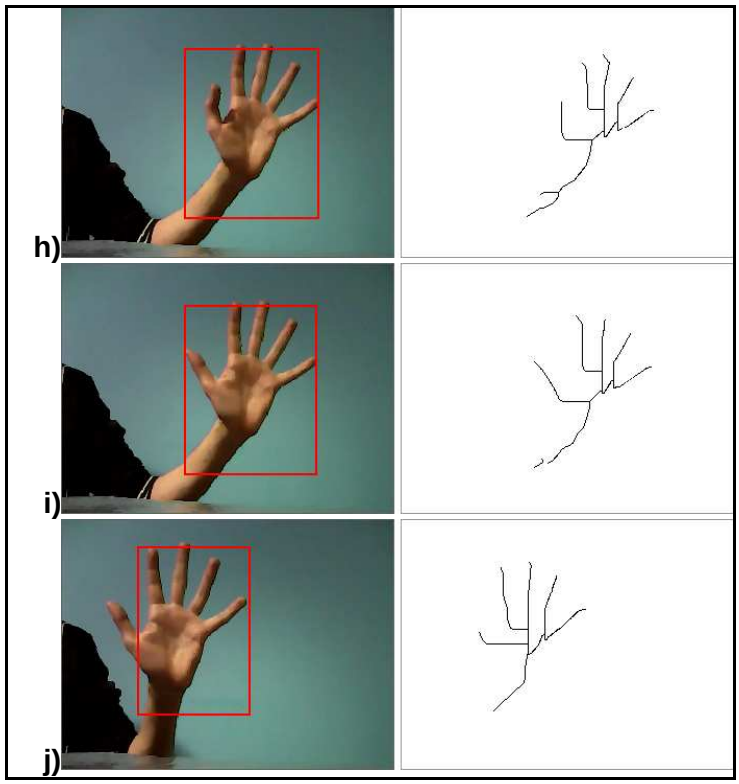
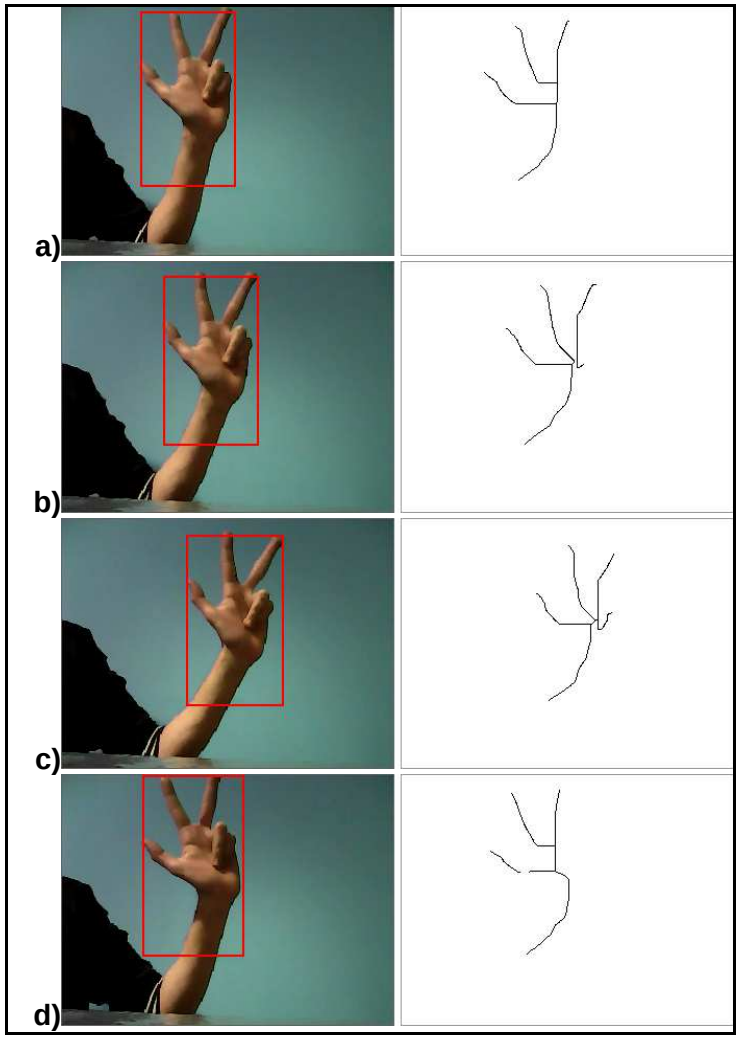


Figura 6.28 - Ejemplos utilizados para evaluar el "Gesto 0".



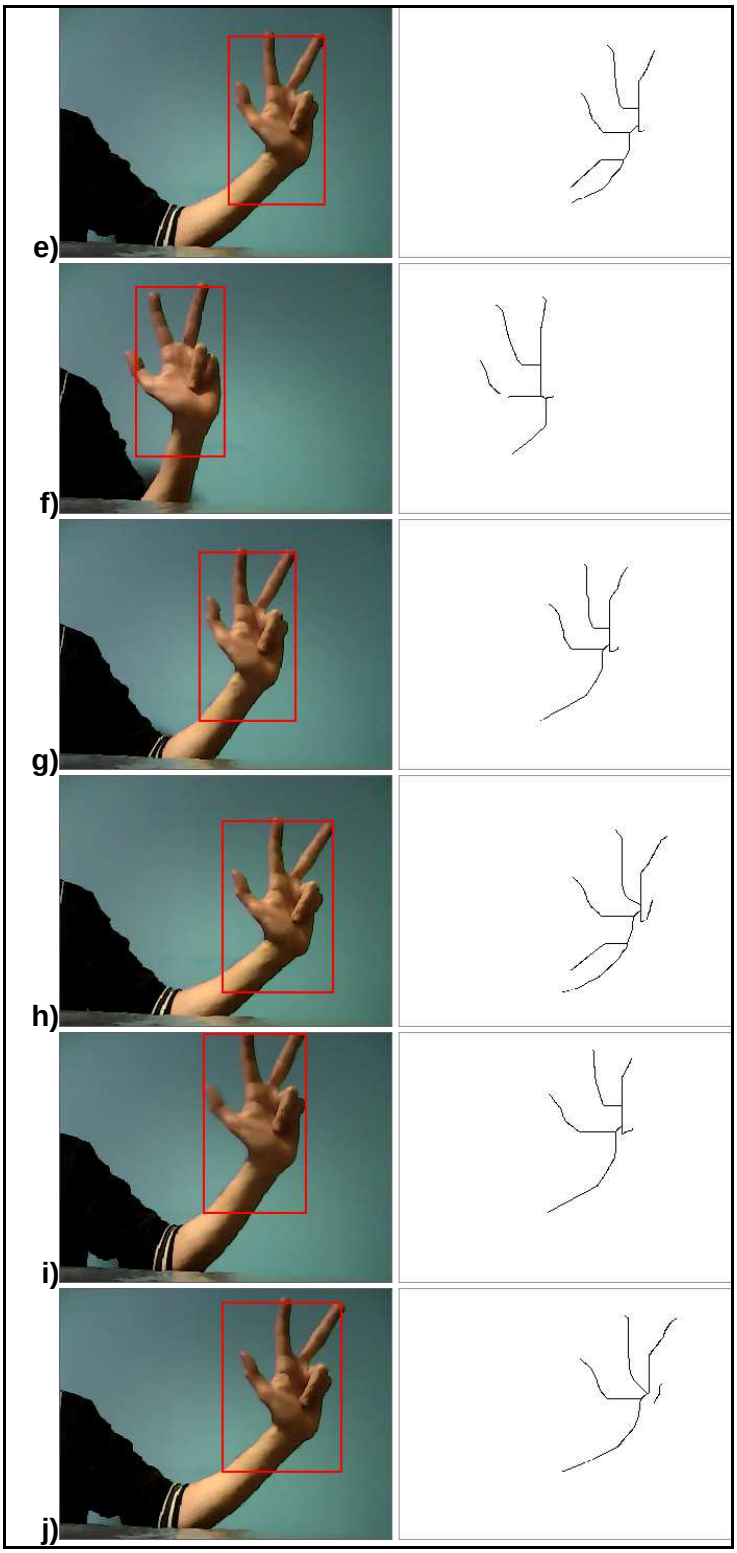
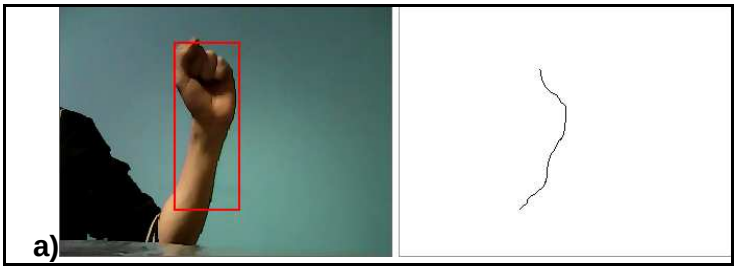
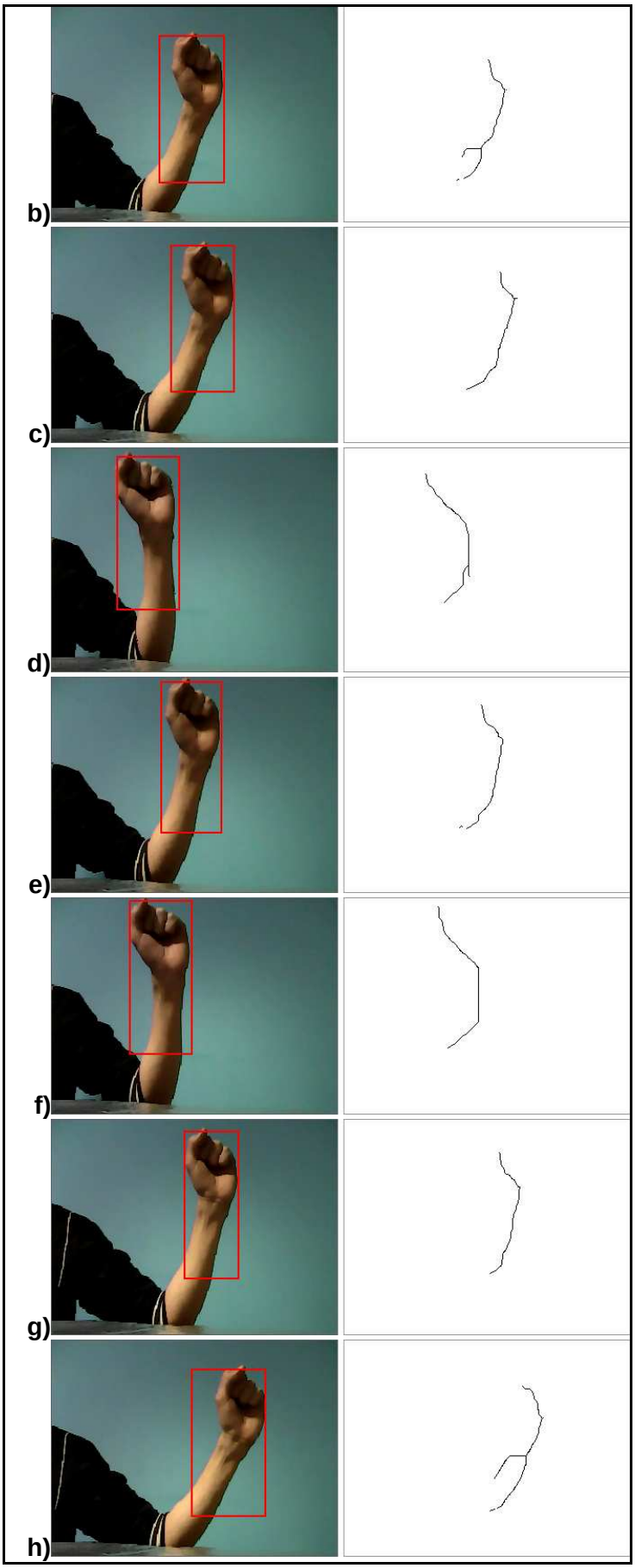


Figura 6.29 - Ejemplos utilizados para evaluar el "Gesto 1".




b)
c)
d)
e)
f)
g)
h)

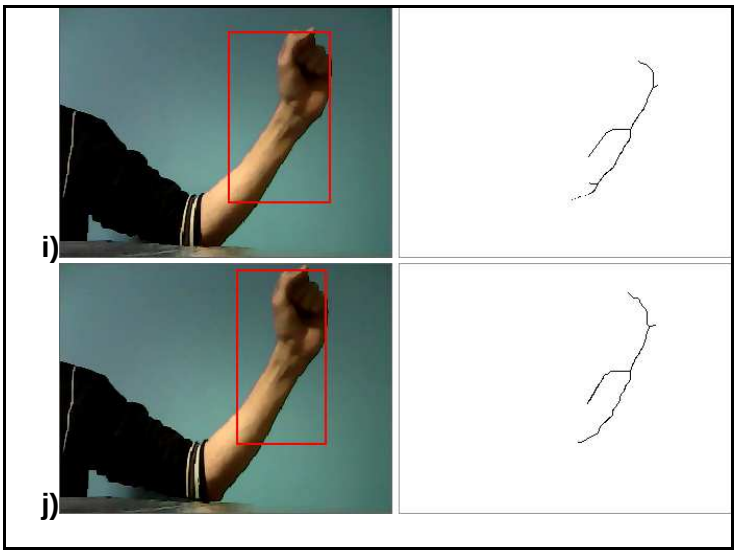


Figura 6.30 - Ejemplos utilizados para evaluar el “Gesto 2”.

7. Conclusiones

En esta tesis se presentó un sistema reconocedor de gestos a partir de las técnicas de visión estereoscópica, esqueletización y redes neuronales. La motivación del trabajo se debió a que es cada vez más frecuente la incorporación de la interpretación de gestos en aplicaciones o aparatos cotidianos.

La visión estereoscópica presenta como ventaja el bajo costo de su implementación y la existencia de buenos resultados cuando se aplica en ambientes pocos ocluidos y cuando la mano se encuentra bien posicionada con respecto a las cámaras, cómo es el caso planteado en el trabajo. Las primeras pruebas realizadas determinaron que si bien la detección de la posición de la mano mediante esta técnica era precisa, su contorno se distorsionaba imposibilitando el reconocimiento del gesto, éste problema suscitó el uso del color para ayudar a la obtención precisa de la mano.

La técnica de esqueletización resulta adecuada para reducir la información de la mano que permitirá luego su reconocimiento mediante la red neuronal, logrando de esta manera mayor eficiencia en el procesamiento del sistema. El principal problema que surgió en las pruebas se debe a la no unicidad de los esqueletos generados por lo que no resulta eficiente si se desea reconocer un gran número de gestos que tengan un esqueleto similar.

La red neuronal es una herramienta poderosa para el reconocimiento de patrones. Su implementación en el sistema permite clasificar un conjunto de información que representa a gestos realizados por la mano para su posterior uso en el reconocimiento de gestos desde el sistema. La precisión en el reconocimiento mejora notablemente a medida que se incrementa la información clasificada.

Por lo explicado anteriormente el sistema resulta ser una implementación de bajo costo que permite el reconocimiento de gestos sencillos casi en tiempo real.

7.1. Trabajos futuros

A continuación se enumera una serie de trabajos futuros que se pueden realizar para extender la funcionalidad del sistema:

- Sería de gran utilidad poder mejorar la velocidad de reconocimiento para permitir el uso del sistema en tiempo real. Esto puede ser logrado aprovechando las características del paralelizado de las tareas de depuración de la imagen.

- La mejora en la precisión del reconocimiento puede llevarse a cabo mediante un entrenamiento más completo de la red neuronal usando un mayor conjunto muestras.

- Extender la funcionalidad del sistema para el reconocimiento del lenguaje de señas. Esta característica necesita incorporar la posibilidad de reconocer ambas manos al mismo tiempo y extender el número de gestos reconocidos por sistema. Para esto será necesario reforzar la detección de las características distintivas de la mano que se van a reconocer como por ejemplo mediante la detección de bordes o contorno.

- Es de gran utilidad para el reconocimiento del lenguaje de señas la incorporación del reconocimiento de gestos dinámicos. Se debe resolver la detección del inicio / fin del gesto, esto puede ser llevado a cabo por otra red neuronal que se encargue de esta tarea.

8. Anexos

8.1. EmguCV

EMGUCV

EmguCV es un envoltorio (wrapper) multiplataforma de la librería de procesamiento de imágenes OpenCV para usar con el framework .NET. Su uso permite el llamado de las funciones de OPENCV desde lenguajes compatibles con .NET como C#, VB, C++ [WWW03].

Figura 8.1 - EmguCV.

OPENCV

OpenCV (Open Source Computer Vision) es una librería libre multiplataforma, orientada a la visión artificial, originalmente desarrollada por Intel y que contiene funciones que abarcan una gran gama de áreas dentro del proceso de visión como reconocimiento de objetos, visión estéreo, redes neuronales [WWW04].

Figura 8.2 - OpenCV.

Uno de los objetivos de la librería consiste en proveer una infraestructura de visión por computadora fácil de usar que permite construir aplicaciones sofisticadas rápidamente.
Dentro de la librería existen más de 500 funciones que abarcan un amplio abanico de áreas como visión estéreo, robótica, imágenes médicas, uso de cámaras, reconocimiento de patrones, etc [Gary Bradski et al., 2008].

A continuación se enumeran los tipos de datos y funciones presentes en la librería y que son utilizadas en la aplicación [WWW05].

- Gray: Clase que define el color gris.

```
public Gray(
      System.double intensity
)

intensity: indica la intensidad del color
```

- Bgr: Clase que define el color RGB.

```
public Bgr(
      System.double blue,
      System.double green,
      System.double red
)

blue: cantidad de color azul
green: cantidad de color verde
red: cantidad de color rojo
```

- MCvScalar: Clase que representa a un vector.

```
public MCvScalar(
      System.double v0
)

v0: representa el tamaño del vector
```

- MCvPoint3D32f: Clase que representa un punto 3D con coordenadas de punto flotante.

```
public MCvPoint3D32f(
      System.float x,
      System.float y,
      System.float z
)

x: valor de coordenada x
y: valor de coordenada y
z: valor de coordenada z
```

- LineSegment2DF: Clase que representa a una línea.

```
public LineSegment2DF(
      System.Drawing.PointF p1,
      System.Drawing.PointF p2
)

p1: punto inicial de la línea
p2: punto final de la línea
```

- MCvTermCriteria: Clase que representa el criterio de terminación para un algoritmo iterativo.

```
public MCvTermCriteria(
      System.int maxIteration,
      System.double eps
)
```

maxIteration: *máximo número de iteraciones permitidas*
eps: *epsilón que determina la precisión*

- Matrix: Clase que permite el manejo de matrices.

```
public Matrix(
      System.int32 rows,
      System.int32 cols
)
```

rows: *número de filas*
cols: *número de columnas*

- Capture: Clase que captura imágenes desde una cámara o video.

```
public Capture(
      System.int32 camIndex
)
```

camIndex: *identifica la cámara a utilizar*

- RetrieveBgrFrame: Método que obtiene la imagen capturada.

```
public virtual Image<Bgr, byte> RetrieveBgrFrame()
```

- SetCaptureProperty: Método que establece las propiedades para la captura de video.

```
public void SetCaptureProperty(
      Emgu.CV.CvEnum.CAP_PROP property,
      System.double value
)
```

property: *identificador de la propiedad*
value: *valor que debe tomar la propiedad*

- Start: Método que Inicia la captura.

```
public void Start()
```

- Image: Clase que permite el manejo de imágenes.

```
public Image(
      System.int32 width,
      System.int32 height
)
```

width: *ancho de la imagen*
height: *alto de la imagen*

- And: Método que realiza la operación morfológica AND.

```
public void _And(
      Emgu.CV.CvArray<TDepth> src2
)
```

src2: *imagen con la cual realizar la operación AND*

- Canny: Método que busca y devuelve los bordes de la imagen.

```
public Image<TColor, TDepth> Canny(
      System.double thresh,
      System.double threshLinking
)
```

thresh: *umbral para la búsqueda de los segmentos iniciales*
threshLinking: *umbral para la vinculación de los bordes*

- Convert: Método que convierte la imagen a otro color y profundidad.

```
public Image<TOtherColor, TOtherDepth> Convert<TOtherColor,
TOtherDepth>()
```

- Copy: Método que realiza una copia de la imagen.

```
public Image<TColor, TDepth> Copy()
```

- CopyBlank: Método que crea una imagen de igual tamaño.

```
public Image<TColor, TDepth> CopyBlank()
```

- Draw: Método que dibuja un circulo / segmento en la imagen.

```
public virtual void Draw(
      Emgu.CV.Structure.CircleF / LineSegment2DF circle / line,
      Emgu.CV.TColor color,
```

```
      System.Int32 thickness
)
```

circle / line: *círculo / línea a dibujar*
color: *color del dibujo*
thickness: *grosor de la línea*

- FindCornerSubPix: Método que itera hasta encontrar la ubicación más precisa de los píxeles de la esquinas.

```
public void FindCornerSubPix(
      System.Drawing.PointF[][] corners,
      System.Drawing.Size win,
      System.Drawing.Size zeroZone,
      Emgu.CV.Structure.MCvTermCriteria criteria
)
```

corners: *coordenadas de los puntos de las esquinas*
win: *tamaño medio de la ventana de búsqueda*
zeroZero: *tamaño medio de la zona muerta sobre la región de búsqueda en la que no se aplica la fórmula*
criteria: *criterio de terminación para el proceso refinador de los puntos*

- FindContours: Método que busca contornos en la imagen.

```
public Contour<Point> FindContours(
      Emgu.CV.CvEnum.CHAIN_APPROX_METHOD method,
      Emgu.CV.CvEnum.RETR_TYPE type,
      MemStorage stor
)
```

method: *método de aproximación para el contorno*
type: *tipo de recuperación del contorno*
stor: *almacenamiento en memoria*

- Not: Método que calcula la imagen complementaria.

```
public Image<TColor, TDepth> Not()
```

- Or: Método que realiza la operación morfológica OR.

```
public Image<TColor, TDepth> Or(
      Image<TColor, TDepth> img2
)
```

img2: *imagen con la cual realizar la operación OR*

- Sub: Método que sustrae una imagen de la imagen actual.

```
public Image<TColor, TDepth> Sub(
      Image<TColor, TDepth> img2
)

img2: imagen a ser sustraída de la imagen
```

- ThresholdBinary: Método que convierte la imagen a partir de un umbral.

```
public void ThresholdBinary(
      Emgu.CV.TColor threshold,
      Emgu.CV.TColor max_value
)

threshold: valor de umbral para la comparación
max_value: valor para los píxeles que superen el umbral
```

- ToBitmap: Método que convierte la imagen en un Bitmap.

```
public System.Drawing.Bitmap ToBitmap()
```

- Dilate: Método que aplica la operación morfológica de dilatación.

```
public void _Dilate(
      System.Int32 iterations
)

iterations: número de iteraciones para realizar la operación
```

- Erode: Método que aplica la operación morfológica de erosión.

```
public Image<TColor, TDepth> Erode(
      System.Int32 iterations
)

iterations: número de iteraciones para realizar la operación
```

- Contour: Clase que representa una colección de puntos pertenecientes al contorno.

```
public Contour(
      Emgu.CV.MemStorage storage
)

storage: almacenamiento en memoria a ser utilizado
```

- ApproxPoly: Método que aproxima la curva.

```
public Contour<T> ApproxPoly(
      System.Double accuracy,
```

```
      Emgu.CV.MemStorage storage
)

accuracy: precisión deseada en la aproximación
storage: almacenamiento en memoria a ser usado
```

- IntrinsicCameraParameters: Clase que representa a los parámetros intrínsecos de la cámara.

```
public IntrinsicCameraParameters()
```

- InitUndistortMap: Método que calcula el mapa sin distorsiones.

```
public void InitUndistortMap(
      System.Int32 width,
      System.Int32 height,
      out Emgu.CV.Matrix<float> mapx,
      out Emgu.CV.Matrix<float> mapy
)

width: ancho de la imagen
height: alto de la imagen
mapx: matriz con el mapa de coordenadas x (salida)
mapy: matriz con el mapa de coordenadas x (salida)
```

- ExtrinsicCameraParameters: Clase que representa a los parámetros extrínsecos de la cámara.

```
public ExtrinsicCameraParameters()
```

- StereoSGBM: Clase que representa un mapa de disparidad utilizando el algoritmo "*semi-global block matching*".

```
public StereoSGBM(
      System.Int32 minDisparity,
      System.Int32 numDisparities,
      System.Int32 SADWindowSize,
      System.Int32 P1,
      System.Int32 P2,
      System.Int32 disp12MaxDiff,
      System.Int32 preFilterCap,
      System.Int32 uniquenessRatio,
      System.Int32 speckleWindowSize,
      System.Int32 speckleRange,
      System.Boolean fullDP
)

minDisparity: valor mínimo posible de disparidad
numDisparities: disparidad máxima menos disparidad mínima
SADWindowSize: tamaño de bloques comparados
P1: primer parámetro para controlar la suavidad de la disparidad
```

P2: *segundo parámetro para controlar la suavidad de la disparidad*
disp12MaxDiff: *máxima disparidad permitida*
preFilterCap: *valor de truncamiento de los píxeles para la imagen pre filtrada*
uniquenessRatio: *margen en porcentaje para determinar cómo realizar el matcheo de puntos*
speckleWindowSize: *tamaño máximo de las regiones de disparidad para considerar puntos de ruido*
speckleRange: *máxima variación de disparidad para considerar los puntos de ruido.*
fullDP: *si se selecciona se indica que se utiliza el algoritmo dinámico two-pass*

- FindStereoCorrespondence: Método que calcula el mapa de disparidad para las imágenes rectificadas.

```
public void FindStereoCorrespondence(
      Emgu.CV.Image<Gray, byte> left,
      Emgu.CV.Image<Gray, byte> right,
      Emgu.CV.Image<Gray, short> disparity
)
```

left: *imagen izquierda*
right: *imagen derecha*
disparity: *mapa de disparidad calculado (salida)*

- ANN_MLP: Clase que representa una red neuronal Multicapa, un tipo de red Pre Alimentada

```
public ANN_MLP(
      Emgu.CV.Matrix<int> layerSize,
      Emgu.CV.ML.MlEnum.ANN_MLP_ACTIVATION_FUNCTION activeFunction,
      System.Double fParam1,
      System.Double fParam2
)
```

layerSiza: *tamaño de las capas*
activeFunction: *indica la función de activación utilizada*
fParam1: *parámetro alpha de la función de activación*
fParam2: *parámetro beta de la función de activación*

- Predict: Método que procesa y evalúa un ejemplo en la red neuronal.

```
public Predict(
      Emgu.CV.Matrix<float> samples,
      Emgu.CV.Matrix<float> outputs
)
```

samples: *ejemplo a procesar*
outputs: *resultado de la predicción*

- Train: Método que entrena la red neuronal.

```
public Train(
      Emgu.CV.Matrix<float> trainData,
      Emgu.CV.Matrix<float> responses,
      Emgu.CV.Matrix<float> sampleWeights,
      Emgu.CV.Matrix<float> sampleIdx,
      MCvANN_MLP_TrainParams parameters,
      ANN_MLP_TRAINING_FLAG flag
)
```

trainData: *muestras a entrenar*
responses: *resultado esperado para las muestras a entrenar*
sampleWeights: *puede dejarse nulo, matriz de pesos para identificar muestras con mayor importancia que otras*
sampleIdx: *puede dejarse nulo, identifica las muestras de interés*
parameters: *parámetros de entrenamiento*
flag: *flag modificador del entrenamiento*

- Load: Método que carga en memoria una red previamente almacenada.

```
public void Load(
      System.String fileName
)
```

fileName: *archivo que contiene la red previamente almacenada*

- Save: Método que almacena una red neuronal para su posterior uso.

```
public void Save(
      System.String fileName
)
```

fileName: *archivo que contiene la red previamente almacenada*

- CalibrateCamera: Función que calcula los parámetros intrínsecos y extrínsecos.

```
public static void CalibrateCamera(
      MCvPoint3D32f[][] objectPoints,
      System.Drawing.PointF[][] imagePoints,
      System.Drawing.Size imageSize,
      Emgu.CV.IntrinsicCameraParameters intrinsicParam,
      Emgu.CV.CvEnum.CALIB_TYPE flags,
      out Emgu.CV.ExtrinsicCameraParameters[] extrinsicParams
)
```

objectPoints: *colección de puntos 3D con patrones de calibración*
imagePoints: *colección de puntos 2D con patrones de calibración*
imageSize: *tamaño de la imagen*
intrinsicParam: *colección de parámetros intrínsecos*
flags: *parámetros de calibración*
extrinsicParams: *colección de parámetros extrínsecos (salida)*

- FindChessboardCorners: Función que determina si la imagen contiene un patrón del tablero de ajedrez y localiza los puntos internos del tablero.

```
public static bool FindChessboardCorners(
      Emgu.CV.Image<Gray, byte> image,
      System.Drawing.Size patternSize,
      Emgu.CV.CvEnum.CALIB_CB_TYPE flags,
      out System.Drawing.PointF[] corners
)
```

image: *imagen origen*
patternSize: *número de ángulos internos del tablero de ajedrez (filas x columnas)*
flags: *parámetros modificadores*
corners: *colección de puntos detectados (salida)*

- StereoCalibrate: Función que realiza la calibración estéreo de dos cámaras.

```
public static void StereoCalibrate(
      Emgu.CV.Structure.MCvPoint3D32f[][] objectPoints,
      System.Drawing.PointF[][] imagePoints1,
      System.Drawing.PointF[][] imagePoints2,
      Emgu.CV.IntrinsicCameraParameters intrinsicParam1,
      Emgu.CV.IntrinsicCameraParameters intrinsicParam2,

      System.Drawings.Size imageSize,
      Emgu.CV.CvEnum.CALIB_TYPE flags,
      Emgu.CV.Structure.MCvTermCriteria termCrit,
      out Emgu.CV.ExtrinsicCameraParameters extrinsicParams,
      out Emgu.CV.Matrix<double> foundamentalMatrix,
      out Emgu.CV.Matrix<double> essentialMatrix
)
```

objectPoints: *colección de puntos 3D con patrones de calibración*
imagePoints1: *colección de puntos 2D con patrones de calibración observados por la cámara 1*
imagePoints2: *colección de puntos 2D con patrones de calibración observados por la cámara 2*
intrinsicParam1: *parámetros intrínsecos*
intrinsicParam2: *parámetros intrínsecos*
imageSize: *tamaño de la imagen*
flags: *parámetros modificadores*
termCrit: *criterio de terminación para el algoritmo iterador*
extrinsicParams: *parámetros extrínsecos*
foundamentalMatrix: *matriz fundamental*
essentialMatrix: *matriz esencial*

- cvRemap: Función que aplica una transformación geométrica a la imagen origen utilizando el mapa recibido como parámetro.

```
public static void cvRemap(
      System.IntPtr src,
      System.IntPtr dst,
      System.IntPtr mapx,
      System.IntPtr mapy,
      System.Int32 flags,
      Emgu.CV.Structure.MCvScalar fillval
)
```

src: *imagen origen*

dst: *imagen destino*
mapx: *mapa de coordenadas x*
mapy: *mapa de coordenadas y*
flags: *valor que identifica un método de interpolación*
fillval: *valor usado para completar puntos aislados*

- cvStereoRectify: Función que realiza la rectificación de una par de cámaras calibradas.

```
public static void cvStereoRectify(
      System.IntPtr cameraMatrix1,
      System.IntPtr cameraMatrix2,
      System.IntPtr distCoeffs1,
      System.IntPtr distCoeffs2,
      Emgu.CV.MCvSize imageSize,
      System.IntPtr R,
      System.IntPtr T,
      System.IntPtr R1,
      System.IntPtr R2,
      System.IntPtr P1,
      System.IntPtr P2,
      System.IntPtr Q,
      Emgu.CV.CvEnum.STEREO_RECTIFY_TYPE flags
      System.double alpha
      System.Drawing.Size newImageSize
      ref System.Drawing.Rectangle validPixROI1
      ref System.Drawing.Rectangle validPixROI2
)
```

cameraMatrix1: *matrices de la cámara 1*
cameraMatrix2: *matrices de la cámara 2*
distCoeffs1: *vector de distorsión de la cámara 1*
distCoeffs2: *vector de distorsión de la cámara 2*
imageSize: *tamaño de la imagen usada para la calibración estéreo*
R: *matriz de rotación para el sistema de coordenadas de las cámaras 1 y 2*
T: *matriz de traslación para el sistema de coordenadas de las cámaras 1 y 2*
R1: *matriz de rotación de la cámara 1 (salida)*
R2: *matriz de rotación de la cámara 2 (salida)*
P1: *matriz de proyección de la cámara 1 luego de la rectificación (salida)*
P2: *matriz de proyección de la cámara 2 luego de la rectificación (salida)*
Q: *matriz de transformación (salida)*
flags: *parámetros de operación*
alpha: *parámetro de escalamiento de la imagen*
newImageSize: *resolución de la imagen luego de la rectificación*
validPixROI1: *rectángulos opcionales dentro de la imagen rectificada*
validPixROI2: *rectángulos opcionales dentro de la imagen rectificada*

- cvInitUndistortRectifyMap: Función que computa los mapas de distorsión para ayudar a corregir las imágenes.

```
public static void cvInitUndistortRectifyMap(
      System.IntPtr cameraMatrix,
      System.IntPtr distCoeffs,
      System.IntPtr R,
      System.IntPtr newCameraMatrix,
```

```
      System.IntPtr mapx,
      System.IntPtr mapy
)
```

cameraMatrix: *matriz de la cámara*
distCoeffs: *vector de distorsión*
R: *matriz de rotación*
newCameraMatrix: *nueva matriz de la cámara*
mapx: *mapa de coordenadas x (salida)*
mapy: *mapa de coordenadas y (salida)*

Para poder hacer uso de las cámaras desde EmguCV primero se debe obtener un listado de las cámaras conectadas a la computadora. Esto se logra mediante el uso de una librería externa cuyo propósito es ofrecer funciones de *DirectShow* desde aplicaciones .NET. La librería es *DirectShowLib-2005* [WWW06].

La lista de dispositivos se obtiene mediante el uso de la siguiente función:

- GetDevicesOfCat: Función que devuelve una lista de dispositivos de una categoría determinada.

```
public static DirectShowLib.DsDevice[] GetDevicesOfCat(
      System.Guid FilterCategory
)
```

FilterCategory: *categoría de búsqueda*

8.2. Aforge.NET

AForge.NET es una librería de visión por computadora e inteligencia artificial desarrollada para el framework .NET cuyas funcionalidades son el procesamiento de imagen, redes neuronales, programación genética y lógica difusa. Se opta por usar la librería por el mejor procesamiento en imágenes y la gran cantidad de filtros que brinda [WWW07].

Figura 8.3 - Aforge.NET.

A continuación se enumeran los tipos de datos y filtros presentes en la librería y que son utilizadas en la aplicación [WWW08].

- Blob: Clase que representa a una región "*blob*", una parte de la imagen. Dentro de la clase se encapsula tanto la región como la información sobre su posición dentro de la imagen original.

```
public Blob(
      AForge.Imaging.Blob source
)

source: blob para inicializar
```

- BlobCounter: Clase que cuenta y extrae las regiones "*blobs*" en la imagen.

```
public BlobCounter()
```

- ExtractBlobsImage: Método que extrae una región de una imagen dada.

```
public void ExtractBlobsImage(
      System.Drawing.Bitmap image,
      AForge.Imaging.Blob blob,
      System.Boolean extractInOriginalSize
)

image: imagen desde donde extraer el blob
blob: blob a extraer
extractInOriginalSize: especifica el tamaño del blob a extraer
```

- GetObjectsInformation: Método que obtiene una colección con la información de las regiones (no incluye la imagen).

```
public AForge.Imaging.Blob[] GetObjectsInformation()
```

- GetObjectsRectangles: Método que obtiene una colección con la información de coordenadas de las regiones.

```
public System.Drawing.Rectangle[] GetObjectsRectangles()
```

- ProcessImage: Método que procesa la imagen para construir el mapa de regiones.

```
public void ProcessImage(
      System.Drawing.Bitmap image
)

image: imagen origen a procesar
```

- Crop: Filtro que recorta una imagen devolviendo una nueva imagen que contiene únicamente el área correspondiente al rectángulo especificado como parámetro.

```
public Crop(
      System.Drawing.Rectangle rect
)

rect: rectángulo a recortar
```

- Grayscale: Filtro que realiza la conversión de la imagen recibida a escala de grises.

```
public Grayscale(
      System.double red,
      System.double green,
      System.double blue
)

red: valor del coeficiente rojo
green: valor del coeficiente verde
blue: valor del coeficiente azul
```

- Threshold: Filtro que realiza la binarización de la imagen usando el umbral recibido como parámetro. Todo píxel con intensidad igual o superior al umbral se convierte en blanco. El resto de los píxeles se convierte en negro.

```
public Threshold(
      System.Int32 threshold
)

threshold: valor del umbral
```

- Merge: Filtro que toma dos imágenes del mismo tamaño y produce una imagen donde cada píxel corresponde al máximo valor de los píxeles correspondientes de cada imagen.

```
public Merge(
      AForge.Imaging.UnmanagedImage unmanagedOverlayImage
)
```

unmanagedOverlayImage: *imagen a superponer*

- Shrink: Filtro que remueve los píxeles del borde de la imagen que tengan un determinado color generando una imagen de menor tamaño.

```
public Shrink(
      System.Drawing.Color colorToRemove
)
```

colorToRemove: *color a quitar en los bordes*

8.3. Diagrama de Clases

El detalle del diagrama de clases del sistema propuesto puede visualizarse en la *Figura 8.4*.

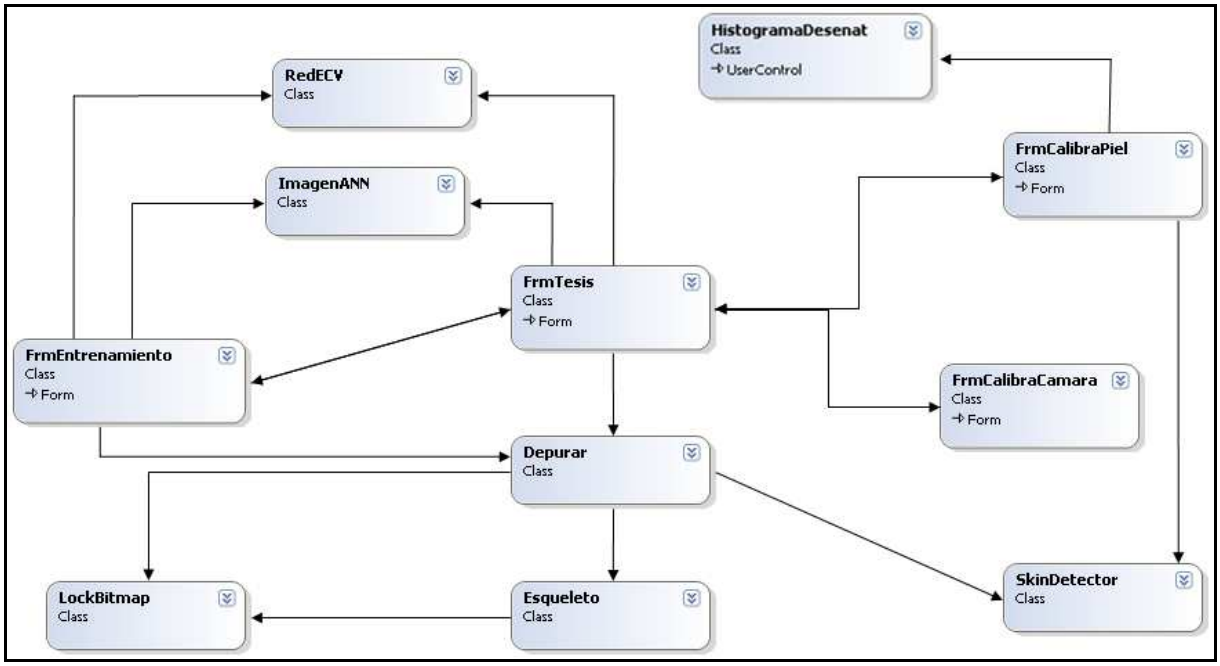


Figura 8.4 - Diagrama de Clases.

A continuación se detalla cada una de las clases del sistema propuesto:

- RedECV: La clase tiene como objetivo crear y entrenar una red neuronal. Una vez que la red ha sido creada permite verificar si un gesto se encuentra entrenado en la red. La red creada se puede guardar y exportar para un posterior uso.
  El detalle de la clase se visualiza en la *Figura 8.5 (a).*

- ImagenANN: El objetivo de la clase es procesar una imagen para su posterior ingreso a la red neuronal. La imagen se recorta y luego se achica para hacer más eficiente el entrenamiento de la red neuronal.
  El detalle de la clase se visualiza en la *Figura 8.5 (b).*

- FrmTesis: Formulario principal que permite reconocer un gesto detectado por el sistema. Además tiene opciones para ingresar a los formularios de parametrización de las opciones del sistema.
  El detalle de la clase se visualiza en la *Figura 8.6.*

- FrmCalibraCamara: Formulario que se utiliza para la construcción del sistema

estereoscópico.
El detalle de la clase se visualiza en la *Figura 8.7.*

- FrmEntrenamiento: Formulario que permite la creación y entrenamiento de la red neuronal.
  El detalle de la clase se visualiza en la *Figura 8.8.*

- FrmCalibraPiel: Formulario que permite calibrar el color de la piel para su posterior uso en la segmentación de la mano.
  El detalle de la clase se visualiza en la *Figura 8.9 (a).*

- Depurar: El objetivo de la clase es segmentar la mano en la imagen y obtener su esqueleto. Para esto se procesa tanto la imagen capturada por las cámaras cómo el mapa de disparidad calculado. Cada hito en el proceso de segmentación es devuelto para ser visualizado en la aplicación. Las tareas realizadas por la clase son la binarización del mapa de disparidad, segmentación por umbral, detección de mayor región, segmentación por color de piel y obtención del esqueleto.
  El detalle de la clase se visualiza en la *Figura 8.9 (b).*

- SkinDetector: La clase realiza la conversión del espacio de color RGB al espacio de color YCbCr para luego segmentar el área correspondiente al color piel.
  El detalle de la clase se visualiza en la *Figura 8.10 (a).*

- Esqueleto: La clase recibe una imagen binarizada y devuelve el esqueleto luego de la aplicación del algoritmo de Zhang Suen.
  El detalle de la clase se visualiza en la *Figura 8.10 (b).*

- LockBitmap: Cuando se trabaja con imágenes los métodos que proporciona Visual C# no son muy eficientes, por ese motivo se utiliza una clase que accede directamente a los bytes de la imagen para acelerar su procesamiento. La clase es de código abierto y el desarrollo fue realizado por Vano Maisuradze [WWW09].
  El detalle de la clase se visualiza en la *Figura 8.5 (c).*

- HistogramaDesenat: Cuando se realiza la conversión del espacio de color RGB al espacio de color YCbCr se necesita visualizar gráficamente la distribución de los píxeles en los canales Cb y Cr para elegir correctamente los valores que ayuden a

segmentar la piel. Para esto se hace uso de un control que permite visualizar un histograma. La implementación del control corresponde a un desarrollo de código abierto realizado por: Alexandru Ghiondea [WWW10].

El detalle de la clase se visualiza en la *Figura 8.10 (c).*

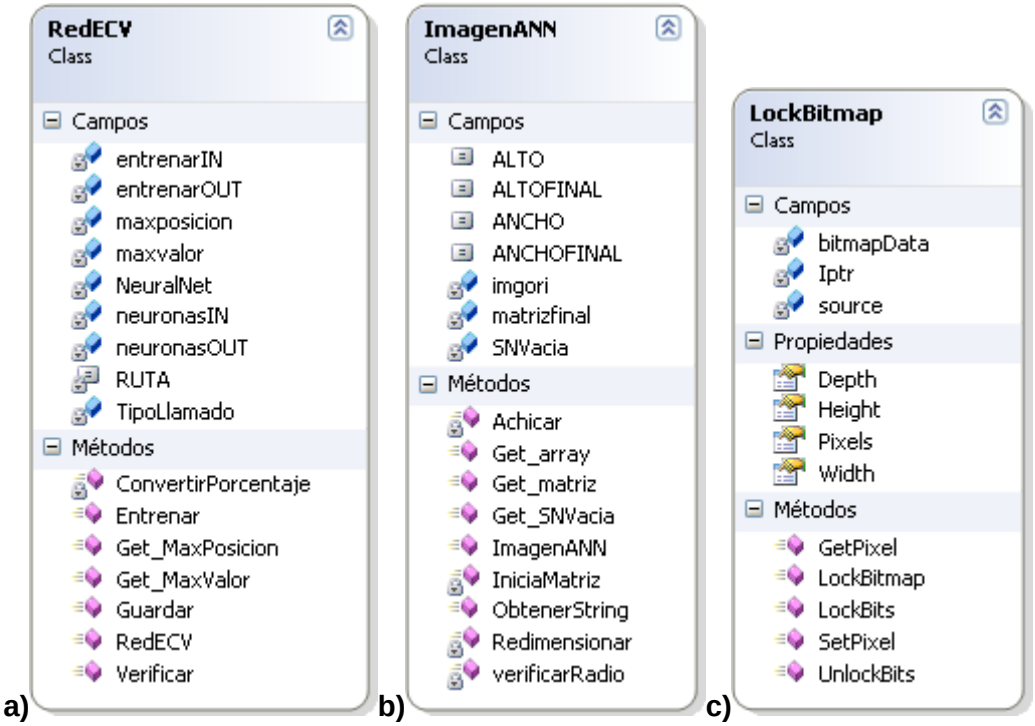


Figura 8.5 - Detalle de las Clases RedECV, ImagenANN y LockBitmap.

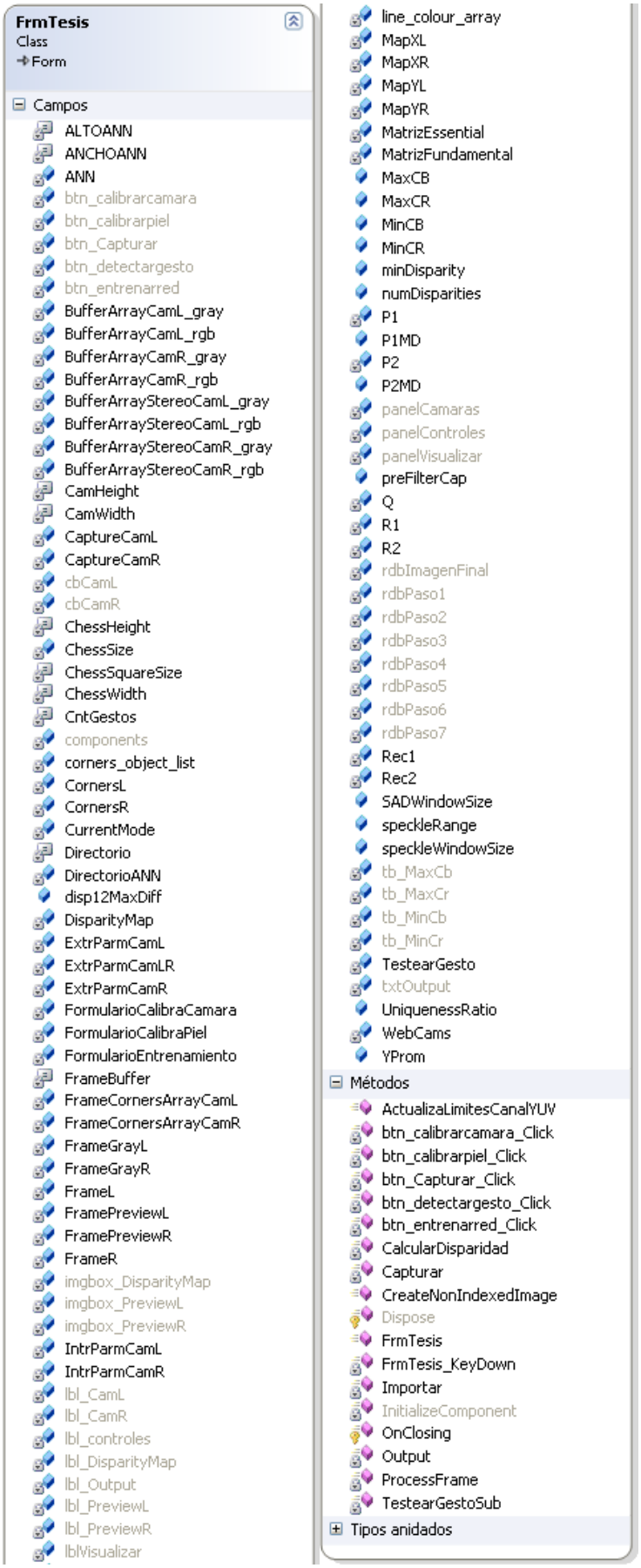


Figura 8.6 - Detalle de la Clase FrmTesis.

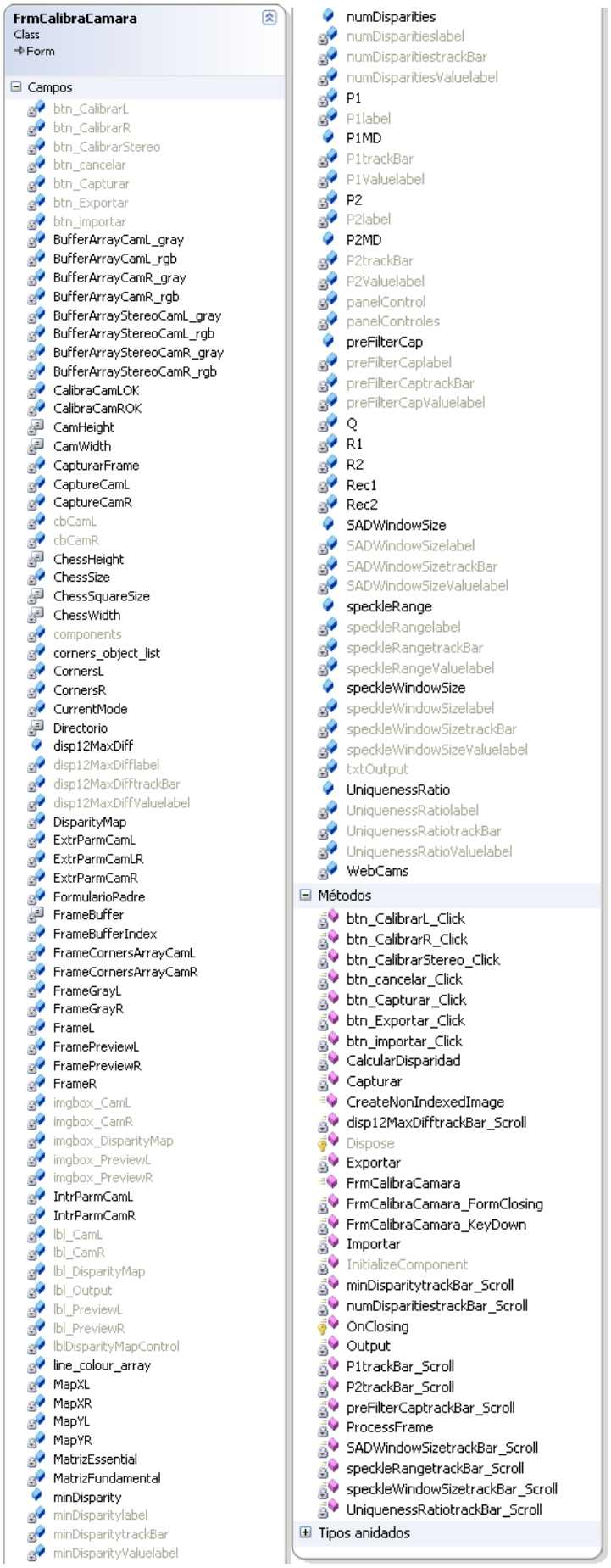


Figura 8.7 - Detalle de la Clase FrmCalibraCamara.

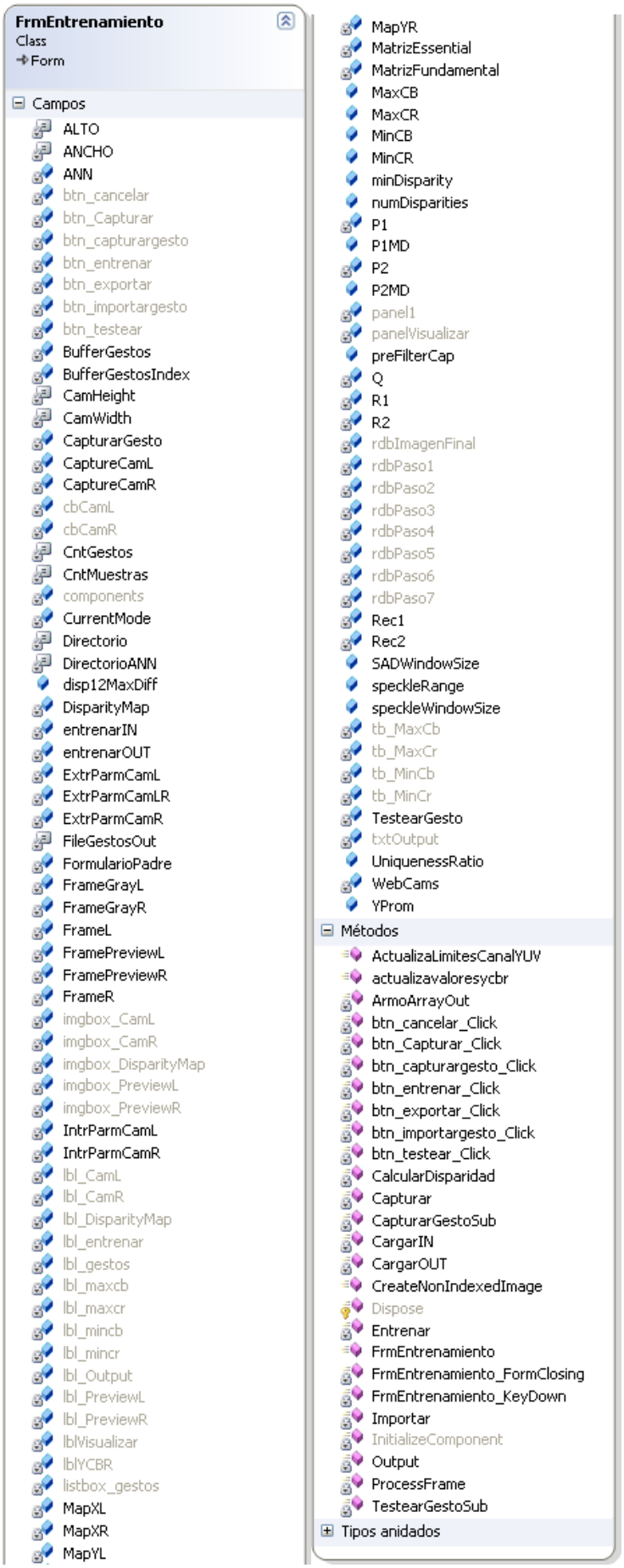


Figura 8.8 - Detalle de la Clase FrmEntrenamiento.

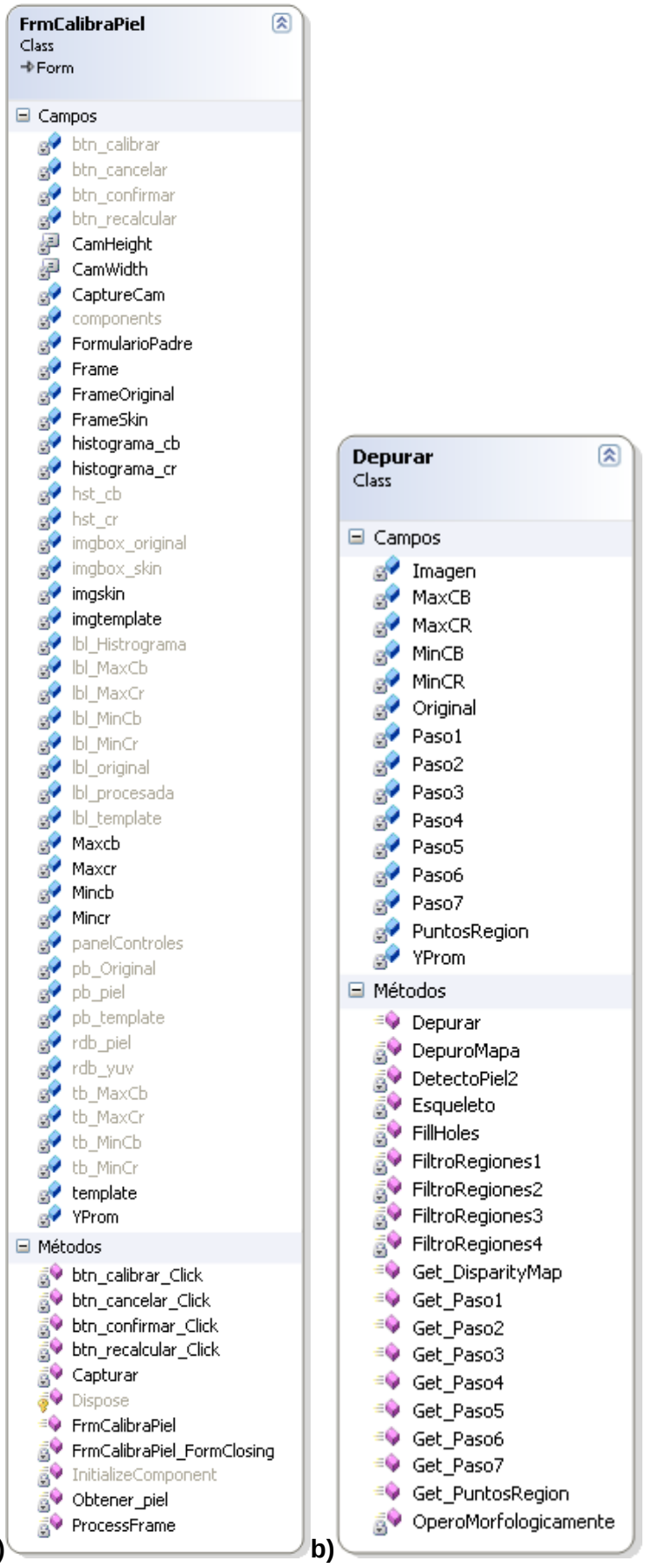


Figura 8.9 - Detalle de las Clases FrmCalibraPiel y Depurar.

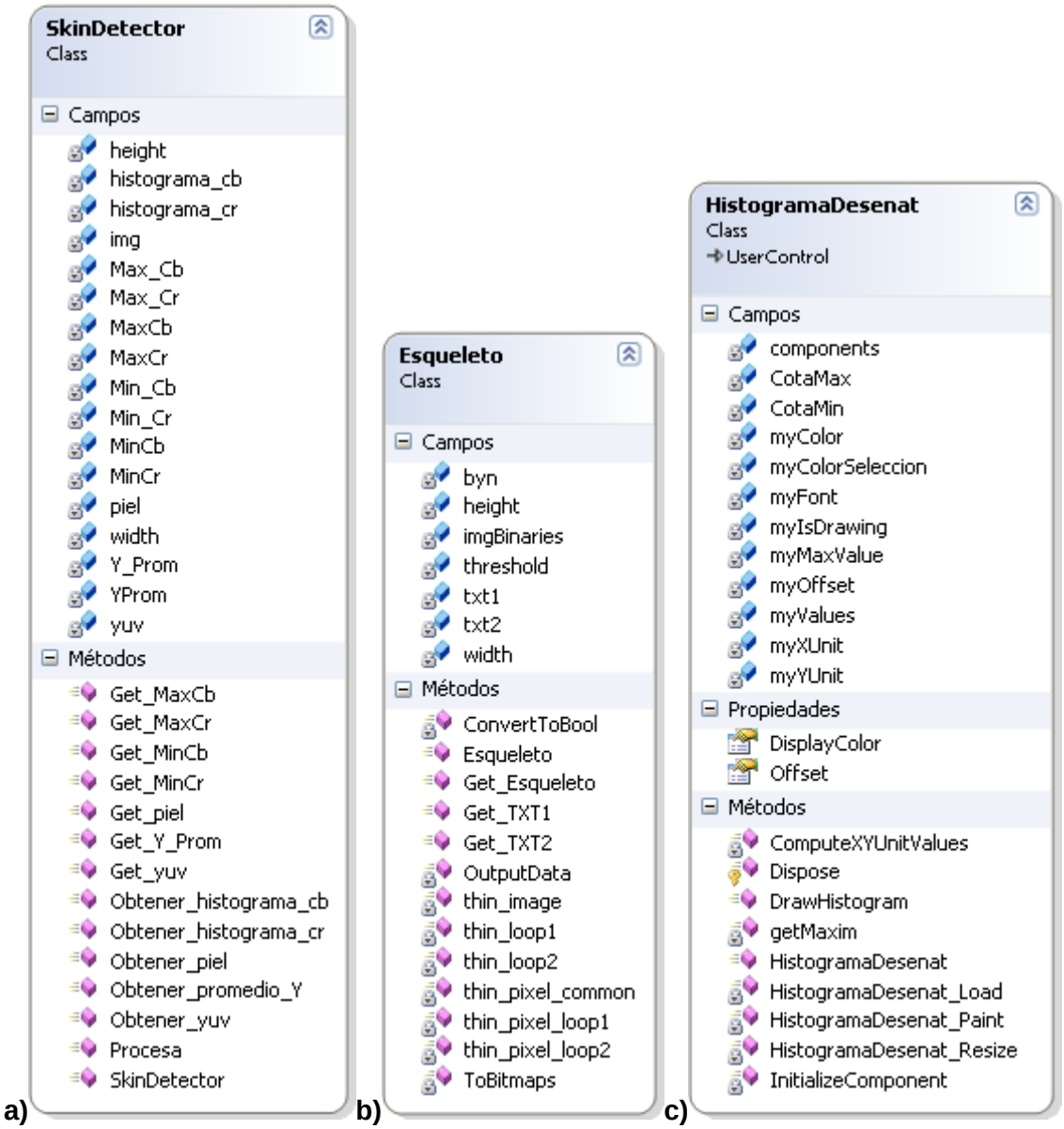


Figura 8.10 - Detalle de las Clases SkinDetector, Esqueleto y HistogramaDesenat.

9. Bibliografía

[Akl, 2010] Ahmad Akl. “A Novel Accelerometer-based Gesture Recognition System”. University of Toronto, 2010.

[Chetverikov et al., 1993] Dmitry Chetverikov, Walter G. Kropatsch. “Computer Analysis of Images and Patterns”. 5th International Conference, Budapest, 1993.

[Daniel Scharstein et al., 2002] D. Scharstein and R. Szeliski. “A taxonomy and evaluation of dense two-frame stereo correspondence algorithms.”, International Journal of Computer Vision, 2002.

[Dröppelmann et al., 2000] Sebastian Dröppelmann, Moos Hueting, Sander Latour, Martijn van der Veen. "Stereo Vision using the OpenCV library". 2000.

[Escobedo-Cabello et al., 2010] Jesús Arturo Escobedo-Cabello, Volodymyr Ponomaryov, Enrique Escamilla-Hernández. “Aplicación de un algoritmo de cálculo de disparidad para la estimación de profundidades usando cámaras estéreo.”, Científica, Vol.14 Núm. 2, pp. 67-73, ESIME Instituto Politécnico Nacional MÉXICO, abril-junio 2010.

[Garg et al., 2009] Pragati Garg, Naveen Aggarwal, Sanjeev Sofat. "Vision Based Hand Gesture Recognition". World Academy of Science, Engineering and Technology, 2009.

[Gary Bradski et al., 2008] Gary Bradski, ADrian Kaehler “OReilly Learning OpenCV”, O'Reilly Media, 2008.

[Gonzalez et al., 2002] Rafael C. Gonzalez, Richard E. Woods. “Digital Image Processing, second edition”. Prentice Hall, 2002.

[Jack, 2008] Keith Jack. “Digital Video and DSP: Instant Access”. Newnes, 2008.

[Khan et al., 2012] Rafiqul Zaman Khan, Noor Adnan Ibraheem "Comparative Study Of Hand Gesture Recognition System", Department of Computer Science, India, 2012.

[Krenker et al., 2011] Andrej Krenker, Janez Bester and Andrej Kos. “Introduction to the Artificial Neural Networks”. ISBN: 978-953-307-243-2, InTech, 2011.

[Martin et al., 2000] Alberto Martin, Sabri Tosunoglu. "Image Processing Techniques for Machine Vision". Florida International University, 2000.

[Mitra et al., 2007] Sushmita Mitra, Tinku Acharya. “Gesture Recognition: A Survey”. IEEE Transactions on Systems, Man and Cybernetics - Part C. 2007.

[Noor et al., 2012] Noor A. Ibraheem, Rafiqul Z. Khan. “Vision Based Gesture Recognition Using Neural Networks Approaches: A Review”. International Journal of human Computer Interaction (IJHCI), 2012.

[Rojas, 1996] Raúl Rojas. "Neural Networks, A Systematic Introduction". Springer, Berlin, 1996.

[Shweta et al., 2011] Shweta K. Yewale, Pankaj K. Bharne. “Artificial Neural Network Approach for Hand Gesture Recognition”. International Journal of Engineering Science and Technology (IJEST), 2011.

[Trucco et al., 1998] Emanuele Trucco, Alessandro Verri. "Introductory Techniques for 3D Computer Vision". Capítulo 7. Prentice Hall, 1998.

[WWW01] http://www.inf.u-szeged.hu/~palagyi/skel/skel.html

[WWW02] http://softwarelibre.unsa.edu.ar/docs/descarga/2003/curso/htmls/redes_neuronales/c35.html

[WWW03] http://www.emgu.com/wiki/index.php/Main_Page

[WWW04] http://en.wikipedia.org/wiki/OpenCV

[WWW05] http://www.emgu.com/wiki/files/2.4.2/document/Index.html

[WWW06] http://directshownet.sourceforge.net/about.html

[WWW07] http://en.wikipedia.org/wiki/AForge.NET

[WWW08] http://www.aforgenet.com/framework/docs/

[WWW09] http://www.codeproject.com/Tips/240428/Work-with-bitmap-faster-with-Csharp

[WWW10] http://www.codeproject.com/Articles/12125/A-simple-histogram-displaying-control