UNIVERSIDAD DEL MAR
CAMPUS PUERTO ESCONDIDO

RECONOCIMIENTO DE ROSTROS EN UN AMBIENTE DE ILUMINACIÓN CONTROLADO NO INVARIANTE A EXPRESIONES FACIALES

T E S I S

QUE PARA OBTENER EL TÍTULO DE
LICENCIADO EN INFORMÁTICA

PRESENTA
CÉSAR GEOVANI PEREYRA RAMOS

DIRECTOR DE TESIS
M. EN C. JORGE OCHOA SOMUANO

PUERTO ESCONDIDO, OAXACA 2011

Dedicatoria

Mi tesis de licenciatura está dedicada a las siguientes personas, que me brindaron la motivación y la entrega para realizar este proyecto de tesis:

Guillermina Ramos Martínez

Mi mamá

Jaime Pereyra Cortés

Mi papá

M. en C. Manuel Alejandro Valdés Marrero

Jefe de Carrera de Informática

M. en C. Jorge Ochoa Somuano

Profesor Investigador y Director de Tesis

M. en C. Isidro Moctezuma Cantorán

Profesor Investigador y Revisor de Tesis

M. en C. José Francisco Delgado Orta

Profesor Investigador y Revisor de Tesis

I.S.C. Saúl Gómez Carreto

Profesor Investigador y Revisor de Tesis

Agradecimientos

En primer lugar quiero agradecer a la Universidad del Mar campus Puerto Escondido, donde he adquirido los conocimientos, habilidades y cualidades en el área de informática que se han adherido a mi persona y a mi razón, que contribuyó a mi formación académica es estos cortos cinco años, pero que han sido suficientes para hacer de mi, un profesional responsable con la universidad y la sociedad.

Al Programa Nacional de Becas para la Educación Superior (PRONABES), por la concesión de la beca de titulación a través del programa becanet – superior.

No muy separadamente, un total agradecimiento a mis maestros del área que con sus correcciones y consejos han formado en mí una actitud de lucha, entrega, responsabilidad, honestidad, constancia y paciencia durante mi vida académica en esta Universidad, pero principalmente agradezco como alumno y como persona a mi director de tesis M. en C. Jorge Ochoa Somuano que me ha mantenido avante en mis metas, a él quiero manifestarle mi gratitud y mi total admiración por su paciencia, su orientación y el interés mostrado para terminar mi proyecto de licenciatura, todo esto es por ustedes maestros, base y más allá de mis conocimientos y de lo que me quedó de cada uno de ustedes.

Agradezco a mi familia por su motivación, a mis padres, mi hermana y sobrina por su total entrega, principalmente a mi madre Guillermina Ramos Martínez quien me ha enseñado a mirar más allá de mi vista, a seguir después de haber caído, a ti Madre te agradezco tu paciencia y constancia en el seguimiento de mis estudios, por haber creído en mí y que ahora te regreso con uno de mis primeros grandes logros en mi vida, esto es por ellos por lo que queda después de todo, mi familia.

Agradezco el tiempo brindado para la toma de las imágenes y crear así la base de conocimiento a mis siguientes compañeros: Abimael, Reyna, Carlos, Francisco, Gemma, Iris Amiel, José, Omar, Cirino, Valentín, Armando, Juan José, Victoria, Jorge, Ricardo.

Pero en la cima de todo esto agradezco a Dios por darme la inteligencia para no darme por vencido en la carrera a pesar de los obstáculos, agradezco infinitamente por darme salud y humildad para seguir adelante.

Resumen

El objetivo de este trabajo de licenciatura titulado “Reconocimiento de rostros en un ambiente de iluminación controlado no invariante a expresiones faciales”, consistió en la implementación de una RNA Perceptron con cuatro neuronas para el reconocimiento de rostros, teniendo como apoyo una base de conocimientos compuesta por las distancias más significativas del rostro de una persona tomadas a partir de una imagen BMP de 378 x 512 pixeles.

El sistema cuenta con un módulo de preprocesamiento en el cual se elimina ruido y se realzan detalles de la imagen, para el posterior cálculo de las características significativas del rostro de la persona, también tiene un módulo que se encarga de entrenar la RNA tipo Perceptron con sus cuatro neuronas de entrada. Por consecuencia, también cuenta con un módulo de reconocimiento en el cual se abre una imagen para su identificación en el sistema, que muestra la imagen guardada de la persona reconocida.

La implementación de esta metodología para el reconocimiento de rostros dio como resultado un 92 % de efectividad en el reconocimiento de imágenes de la base de conocimiento, teniendo en cuenta que el gesto de la persona no varíe mucho.

Abstract

The goal of this underwork entitled “Faces Recognition in a controlled lighting environment not invariant to facial expressions”, was the implementation of a perceptron artificial neural network with four neurons for the recognition of faces, supported by a knowledge database consisting of the most significant distances from the face of a person taken from a 378 x 512 pixels bitmap image.

The system features a pre-processing module which removes noise and enhance image details, for the subsequent calculation of the significant features of the person face. Also, it has a module which is responsible for training the perceptron artificial neural network with four input neurons. Consequently, it also has a recognition module in which an image is opened for identification by the system, which displays the saved image of the recognized person.

The implementation of this methodology for face recognition resulted in a 92 % success in recognizing images of the knowledge base, considering that the gesture of the person does not vary much.

CONTENIDO

LISTADO DE FIGURAS

LISTADO DE TABLAS

LISTADO DE ECUACIONES

LISTADO DE CÓDIGO FUENTE

GLOSARIO DE TÉRMINOS

ALU	Arithmetic Logic Unit. Unidad aritmética lógica, es la encargada de manejar la mayor parte de las operaciones de procesamiento (Norton 1999).
BLUETOOTH	Es la norma que define un estándar global de comunicación inalámbrica, que posibilita la transmisión de voz y datos entre diferentes equipos mediante un enlace por radiofrecuencia (Blauden electronics 2008).
CCD	Charge Coupled Device. Dispositivo acoplado por carga, graba imágenes en un chip semiconductor fotosensitivo del tamaño de una estampilla de correo (Pajares & De la Cruz 2008).
CLASE	Los elementos u objetos que comparten una serie de características comunes se pueden asociar de forma natural en clases o grupos. Las relaciones y los atributos compartidos entre estos elementos son la base del conocimiento heredado (Pajares & Santos 2006).
CMY	Modelo de color con información de: Cyan (cian), Magenta (magenta) y Yellow (amarillo) (Pajares & De la Cruz 2008).
CUANTIZACIÓN	El objetivo es aproximar una muestra de entrenamiento dada a una distribución desconocida utilizando un número pequeño de vectores prototipo (Pajares & De la Cruz 2008).
DETECCIÓN	Poner de manifiesto, utilizando medios adecuados, lo que no puede ser observado directamente (*Diccionario de la lengua española* 1997).
ENTROPÍA	Permite evaluar la degradación de la energía de un sistema (*Larousse enciclopédico universal* 2000).
GANANCIAS	Con frecuencia se añade al conjunto de pesos de la neurona un parámetro adicional que se denomina bias o ganancia, éste representará el umbral de disparo de la neurona, es decir, el nivel mínimo que debe alcanzar una neurona para que ésta se active (Martín & Sanz 2007).

GUI	Graphical User Interface. Interfaz gráfica de usuario, es un medio de comunicación efectiva entre un humano y una computadora (Pressman 2005).
HSI	Modelo de color que tiene información de: Hue (matiz), Saturation (saturación) e Intensity (intensidad) (Pajares & De la Cruz 2008).
HSV	Modelo de color con la siguiente información: Hue (matiz), Saturation (saturación) y Value (valor) (Pajares & De la Cruz 2008).
IMAGEN	Representación grabada, pintada, dibujada o esculpida de una persona o cosa (*Diccionario de la lengua española* 1997).
IMAGEN DIGITAL	Matriz de valores en dos dimensiones (2D), que se almacenan en medios electrónicos (Pajares & De la Cruz 2008).
OBJETO	Es el conocimiento que se representa mediante tuplas o registros de información de cada elemento (imágenes, formas de ondas o señales o cualquier tipo de medidas que necesitan ser clasificadas) (Pajares & De la Cruz 2008). Cada una de estas tuplas contiene un conjunto de campos o columnas que definen atributos específicos y valores de ese elemento. Pudiéndose utilizar cálculos relacionales para manipular los datos, basándose en relaciones definidas entre ellos y así buscar información en la tabla (Pajares & Santos 2006).
PDA	Personal Digital Assistant. Asistente personal digital, son mucho más pequeñas que las computadoras portátiles, se usan para aplicaciones especiales, como crear hojas de cálculo pequeñas, mostrar números telefónicos y direcciones importantes y dar seguimiento a fechas y agendas (Norton 1999).
PESO	También conocido como peso sináptico, define la intensidad de interacción entre la neurona presináptica (las que envían las señales) y las neuronas postsinápticas (las que las reciben) (Martín & Sanz 2007).

RAM Random Access Memory. Memoria de acceso aleatorio, el propósito de la RAM es conservar programas y datos mientras están en uso (Norton 1999).

RGB El modelo de color RGB, iniciales en inglés de los colores primarios, Red (rojo), Green (verde) y Blue (azul), modelo de color orientado al hardware como: monitores de color y una amplia gama de videocámaras (Pajares & De la Cruz 2008).

SIGMOIDAL Esta función resulta adecuada para modelar propiedades como: grande, mucho o positivo. Se caracteriza por tener un valor de inclusión distinto de 0 para un rango de valores por encima de cierto punto (límite inferior), siendo 0 por debajo de este límite y 1 para valores mayores a un límite superior (Martín & Sanz 2007).

SINÁPSIS Formalmente puede interpretarse como el producto escalar de los vectores de entrada y pesos (Martín & Sanz 2007).

SINERGIA Acción combinada de diversas acciones tendentes a lograr un efecto único. Asociación de varias organismos para lograr una función (*Larousse enciclopédico universal* 2000).

SISTEMA EXPERTO Sistema basado en el conocimiento, desarrollo de mecanismos de búsquedas de propósito general, es el uso de conocimiento específico del dominio que facilita el desarrollo de etapas de razonamiento más largas, resolviendo así recurrentes en dominios de conocimiento restringido o resolver soluciones complejas (Russell & Norving 2004).

SURFACE Producto de Microsoft que permite al usuario manejar contenidos digitales con movimientos de las manos u objetos (Microsoft 2010).

XYZ Este modelo de color en la componente Y estaría la luminosidad y en XZ la coloración (De la Escalera 2009).

YIQ Este modelo de color se usa en la televisión comercial, Y por luminance (luminancia), I por In-phase (fase) y Q por Quadrature (cuadratura), es decir, Y para la luminancia y dos para la información del color (I y Q)

(Pajares & De la Cruz 2008).

YUV Este modelo se caracteriza por tener la información de: Y por luminance (luminancia) y dos para la información del color U y V (diferentes gamas de brillo entre los colores azul y rojo, respectivamente) (Pajares & De la Cruz 2008).

CAPÍTULO 1. INTRODUCCIÓN

El presente trabajo de licenciatura consiste en la realización de un software a manera de prototipo que permita reconocer rostros a partir de distancias de sus rasgos más significativos, dichas distancias se entrenan con una RNA Perceptron. El software fue realizado en Borland C++ Builder 5.0.

Problemática

Actualmente la mayoría de las personas han escuchado en las noticias sobre la clonación de tarjetas de crédito, el robo de identidad, los secuestros y robos *express*, imágenes donde se ve claramente al asaltante de un cajero o banco, un policía corrupto, un funcionario público pasando por alto la ley (conversando por teléfono de los actos que realizó, de lo que vendió o del fraude que cometió, etc.). Se pueden mencionar muchos ejemplos similares, pero la realidad es que en la mayoría de los casos no se atrapa al delincuente por la falta de tecnología que permita la identificación de las personas que van contra la ley.

Este tipo de sucesos, hace decidir a la sociedad pagar lo que sea necesario para tener seguridad en sus hogares, negocios, escuelas y empresas; principalmente en actividades donde involucren dinero, patrimonio, pasatiempos y diversiones. Es por ello que la mayoría se interesa en los diferentes mecanismos que le brinden seguridad, por ejemplo: un sistema de reconocimiento de rostros.

Metodología de solución

Adquirir las imágenes con tomas de fotografías a diferentes personas en un ambiente controlado, sobre todo en iluminación, además manteniendo un mismo tipo de expresión facial.

- Generar una base de imágenes para la fase de pruebas y experimentación.
- El sistema tiene tres módulos: El primer módulo es para la carga de una imagen, un segundo módulo interno de preprocesamiento y procesamiento de las imágenes y un último módulo de aprendizaje y reconocimiento.
- Aplicar algoritmos de procesamiento de imágenes digitales para resaltar detalles importantes de la imagen o para eliminar ruido, es decir, se le hará un preprocesamiento.
- Aplicar métodos de cálculo de distancia euclideana entre las partes de la cara (largo y ancho: ojo izquierdo, ojo derecho, nariz, boca y una combinación de éstas). Cabe aclarar que este proceso se debe realizar manualmente, es decir, el usuario debe presionar en el punto inicial, arrastrar hasta el punto final y soltar; esos dos puntos principales son los que delimitan a las partes del rostro a las que se va a calcular la distancia euclideana, por ejemplo: si se desea calcular el ancho del ojo izquierdo el usuario debe presionar en el punto A, arrastrar hasta en el punto B y soltar (Fig. 1.1), el sistema calculará la distancia euclideana (Ec. 1) del punto A al punto B de manera automática y así sucesivamente para las demás distancias.
- Generar una base de datos con diferente información que represente diversas medidas del rostro, métricas de la nariz, ojos, boca y combinaciones entre éstas.

- Implementar algoritmos de inteligencia artificial para las etapas de aprendizaje y reconocimiento.
- Describir y explicar los resultados que se obtuvieron con las pruebas realizadas, así como comentarios y conclusiones generadas en esta etapa.

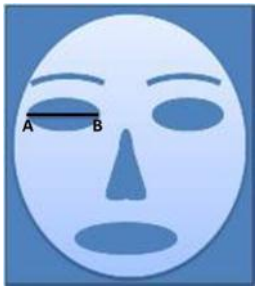


Figura 1.1. Cálculo de la distancia euclideana del punto A al punto B.

$$d(A,B) = \sqrt{(A_x - B_x)^2 - \left(A_y - B_y\right)^2} \quad (1)$$

Organización del documento de tesis

La estructura general del presente trabajo de tesis se compone de cinco capítulos, a continuación se describen los cuatro restantes.

Capítulo 2. Antecedentes. En este capítulo se estudian y analizan las técnicas y metodologías que se han implementado en otros trabajos de reconocimiento de rostros, esto para retomar alguna técnica utilizada y decidir el camino de solución, analizando cómo se han planteado, desarrollado y solucionado problemas similares. Sin lugar a dudas, con este análisis se tiene un panorama más claro tanto del dominio del problema como de las herramientas que se utilizaron y que dan solución a este trabajo de investigación.

De la misma forma se abordan temas como:

- Justificación.
- Planteamiento del problema.
- Objetivos.
- Alcances y límites de estudio.

Capítulo 3. Marco teórico. Se describen las técnicas, herramientas y metodologías implementadas en el presente trabajo de tesis, también se describe la base teórica del

reconocimiento de rostros. En este capítulo se ordena todo el proceso teórico que respalda la metodología de solución.

Capítulo 4. Desarrollo del tema. Se describe a detalle el sistema implementado que resolvió el problema del reconocimiento de rostros en un ambiente de iluminación controlado no invariante a expresiones faciales, es decir, se describe el funcionamiento de los tres módulos: preprocesamiento, entrenamiento y reconocimiento. Éstos ayudan a la interpretación de los resultados que da el sistema y por tanto se muestran las pruebas realizadas con la metodología de solución utilizada en este trabajo de tesis.

Capítulo 5. Conclusiones y trabajos futuros. Es el capítulo donde se describen las conclusiones de los resultados de las 6 pruebas realizadas, de igual manera se mencionan sugerencias sobre posibles actualizaciones que se le pueden hacer al sistema y de allí proponer nuevos módulos e incluso sistemas novedosos que incorporen nuevas metodologías o la aplicación con otras técnicas para el mejoramiento y resolución de nuevas problemáticas.

En este último capítulo se explican los resultados de las 6 pruebas, el porcentaje de certeza y de error de cada prueba, así como el número de imágenes utilizadas en cada módulo del sistema y especialmente de las imágenes utilizadas en el módulo de reconocimiento, es decir, en este apartado se evaluó la eficiencia del sistema.

Anexo A. Imágenes originales para las pruebas. En este apartado se muestran las imágenes utilizadas en las pruebas, en la tabla I se describen cada una de las pruebas y las imágenes utilizadas en los módulos.

Tabla I. Caracteristicas de las pruebas realizadas al sistema.

			Imágenes utilizadas en el módulo de	
Prueba	Personas	Imágenes por personas	Preprocesamiento	Reconocimiento
1	10	2	20	10
2	10	2	20	10
3	10	2	20	5
4	10	3	30	10
5	10	3	30	10
6	10	4	40	10

Complementando la información de las últimas columnas de la tabla I, en la tabla II se especifican las imágenes que se utilizaron en dichos módulos. Las imágenes utilizadas se encuentran en los anexos A y B.

Tabla II. Imágenes utilizadas en las pruebas.

Imágenes utilizadas en los módulos de		
Prueba	Preprocesamiento	Reconocimiento
1	A.1 y A.2	A.2
2	A.1 y A.2	A.3
3	A.1 y A.2	A.5
4	A.1, A.2 y A.3	A.1
5	A.1, A.2 y A.3	A.4
6	A.1, A.2, A.3 y A.4	A.2

Anexo B. Imágenes para el preprocesamiento y reconocimiento. En este apartado se muestran las imágenes procesadas en el módulo de preprocesamiento, se muestran las imágenes después de aplicarles la escala de grises y filtro de Sobel. En este anexo también se muestran las imágenes utilizadas en el módulo de reconocimiento de cada prueba.

Anexo C. Funcionamiento del sistema. En este apartado se describe detalladamente cada uno de los módulos del sistema, se ilustran los pasos que se deben realizar para llevar a cabo una prueba, desde el momento que se abre una imagen para calcular sus distancias, mostrar los datos, pasar por la etapa de preprocesamiento, de entrenamiento de los datos y el reconocimiento de un rostro.

Anexo D. Código fuente. En este anexo se muestran y describen los códigos medulares del sistema. Se mencionan los códigos de abrir una imagen, funciones de la red neuronal, así como el código de las acciones de los botones principales de cada módulo.

Anexo E. Contenido del CD. En esta parte del documento se ilustra y describen los archivos que se encuentran en el CD-ROM, en éste se encuentran el código fuente del proyecto, el archivo ejecutable, el documento de la tesis en archivo pdf, las imágenes de cada prueba y el instalador del proyecto.

CAPÍTULO 2. ANTECEDENTES

Hoy día se encuentran diversos trabajos que abordan métodos de solución con redes neuronales, con procesamiento de imágenes digitales y con otras técnicas muy efectivas en el reconocimiento de rostros. A continuación se presentan proyectos relacionados con este tema.

2.1. Estado del arte y trabajos relacionados

Actualmente no existen trabajos relacionados con el tema de redes neuronales en la Universidad del Mar, sin embargo, existe una tesis (Toscano 2009), donde se utilizó procesamiento de imágenes para localizar un objeto en movimiento a partir de una serie de imágenes.

Analizando diferentes trabajos y estudiando las metodologías utilizadas por ellos, es como se adquieren más ideas y métodos para resolver ciertos problemas que tendrá este proyecto de tesis, por lo cual es relevante estudiar y describir algunos artículos.

Detección y seguimiento de objetos invariantes en color en una secuencia de imágenes (Toscano 2009)

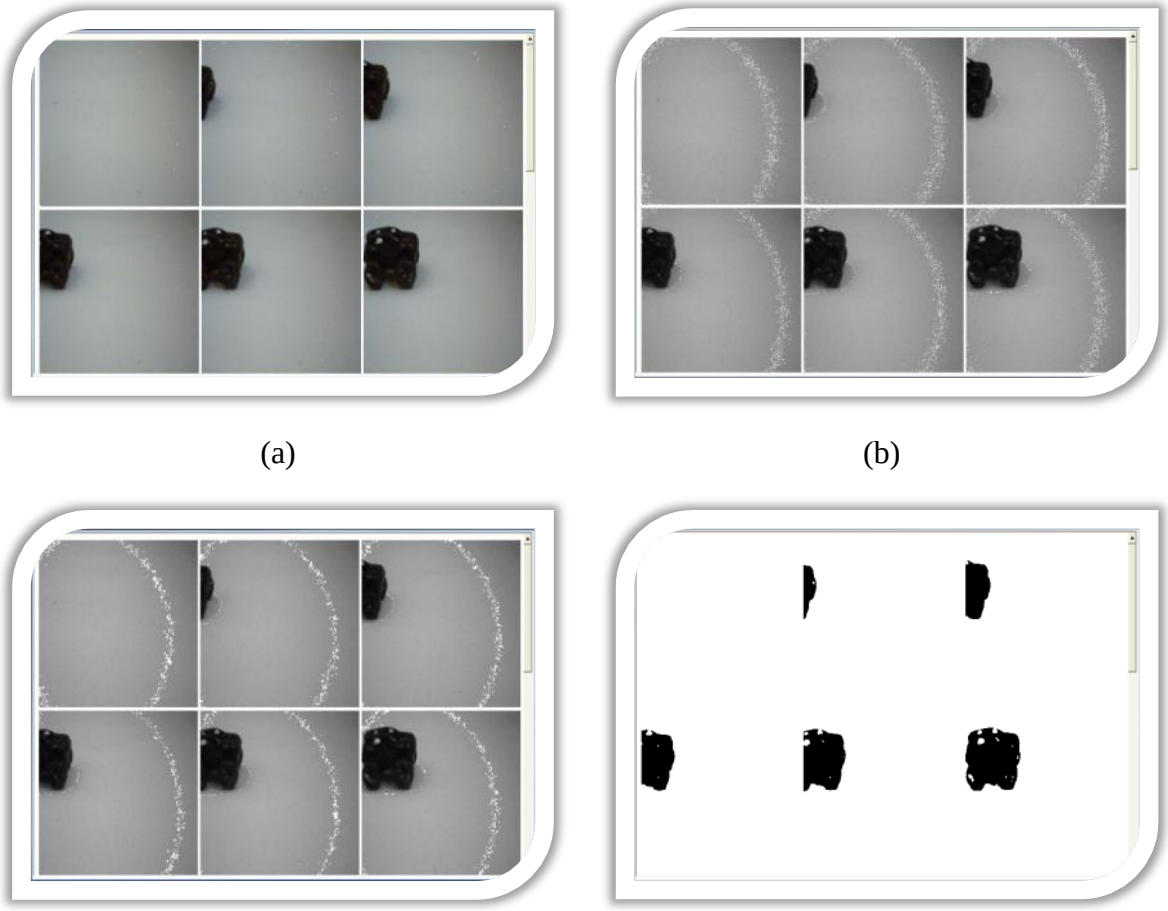
Esta tesis proporciona ideas muy puntuales a considerar, el trabajo consistió en el desarrollo de un software que permite detectar objetos en movimiento a partir de secuencia de imágenes y por lo tanto, realizar un seguimiento del objeto durante una trayectoria lineal, con la finalidad de resolver situaciones enfocadas en el área de la detección y seguimiento de objetos, mediante la utilización de métodos de inteligencia artificial y de visión por computadora. En el desarrollo de este trabajo se aplicaron diferentes técnicas de procesamiento de imágenes entre las cuales se pueden mencionar las siguientes: aplicación de la escala de grises, filtro de la mediana, binarización, operadores de Sobel y resta de imágenes. A raíz de la aplicación de estas técnicas, se determinó que el sistema desarrollado tiene una efectividad general del 99.7 %, siempre y cuando la secuencia de imágenes se obtenga bajo un ambiente controlado, esto quiere decir, que los colores de los objetos deben tener un contraste notable respecto al fondo en cada una de las imágenes de la secuencia.

Las técnicas y herramientas utilizadas antes mencionadas, en conjunto con un ambiente controlado en el momento de la toma de las imágenes, logran la efectividad alcanzada por este trabajo. Estas técnicas siguen un ciclo general de proceso que se basa en las etapas fundamentales del procesamiento digital de imágenes. Es preciso mencionar un ejemplo que describe y muestra los procesos realizados.

En la escena se muestra un automóvil en color negro sobre un fondo blanco. A continuación se especifican los procesos realizados, así como los resultados obtenidos en cada uno de estos. La figura 2.1a, muestra la secuencia de imágenes en original.

Una vez cargada la secuencia de imágenes, se le aplica el proceso de escala de grises y como resultado de este proceso se obtiene la secuencia presentada en la figura 2.1b.

Con la intención de eliminar el ruido que se presenta en cada una de las imágenes de la secuencia, se aplica el filtro de la mediana a la secuencia de imágenes en escala de grises, el resultado se aprecia en la figura 2.1c.



(a) (b)

(c) (d)

Figura 2.1 Imágenes de un objeto en movimiento. (a) Secuencia original de imágenes, (b) Imágenes en escala de grises, (c) Resultado del filtro de la mediana y (d) Binarización de las imágenes.

Los resultados son poco perceptibles pero se si observan con detenimiento en las figuras 2.1b y 2.1c se pueden apreciar las diferencias. El siguiente paso fue la aplicación del filtro de binarización a la secuencia de imágenes de la figura 2.1c, con la finalidad de dejar cada una de las imágenes en dos tonalidades de color (Fig. 2.1d), para este ejemplo, blanco y negro; para con ello poder realizar la correcta detección de bordes con los operadores de Sobel que se aplicaron en el siguiente paso (Fig. 2.2).

Por último, se procede a realizar la detección y seguimiento de los objetos aplicando el procedimiento de la resta de la imágenes a las imágenes de la figura 2.2, en esta secuencia de imágenes se localiza el automóvil, pero solamente se visualizan los bordes del mismo desplazándose en las imágenes de la secuencia y también se cuenta con el fondo pero en

color negro. Este último proceso tiene como finalidad realizar la correcta detección y seguimiento de los objetos.

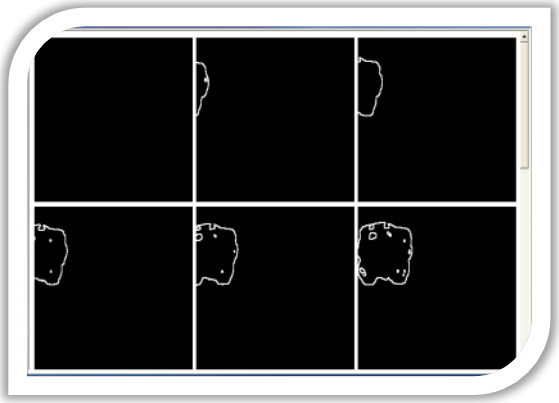


Figura 2.2. Aplicación de los operadores de Sobel.

Para apreciar la detección y seguimiento del objeto se colocó un polígono de cuatro lados en color rojo, a partir de los resultados obtenidos. Como resultados obtenidos exclusivamente para este caso, se puede determinar una efectividad del 100 % y un porcentaje de error del 0 %, en un tiempo de ejecución de 40.8 segundos. El criterio para el cálculo del error se tomó con base en el área de los objetos rodeados por el polígono de color rojo, respecto al objeto en su totalidad.

Es importante mencionar que se consideraron los métodos utilizados en esta tesis para el desarrollo de este proyecto de investigación, porque han reportado buena efectividad y aplican metodologías generales en el procesamiento digital de imágenes y especialmente por el ambiente controlado que ayuda a eliminar una gran cantidad de ruido en las imágenes.

Reconocimiento de rostros utilizando secuencias de histogramas como tramas espacio-temporales (Moreno et al. 1999)

Este artículo describe un modelo de reconocimiento de rostros humanos, utilizando redes neuronales espacio-temporales (STN, por sus siglas en inglés). Esta red neuronal incorpora una capa de entrada, que se encarga de transformar la imagen de entrada al sistema en una secuencia de vectores normalizados que serán aplicados a una red espacio-temporal. Finalmente, la red tiene una capa de salida que utiliza unidades *outstar* de Grossberg.

Este trabajo propone una solución al reconocimiento de rostros, que convierte la imagen bidimensional en una secuencia espacio-temporal de vectores. Se trata de un modelo que, a partir de la imagen, calcula el histograma de cada una de las filas que la componen. Una imagen de *NF* filas y *NC* columnas con *NG* niveles de gris, dará lugar a *NF* histogramas, cada uno correspondiendo a una fila, de *NG* componentes cada uno. La secuencia ordenada de estos *NF* histogramas puede utilizarse para identificar una imagen. Aunque en teoría es posible que dos imágenes diferentes tengan la misma secuencia ordenada de histogramas, la posibilidad de que esta situación se produzca en un proceso de identificación es casi nula. En una secuencia de histogramas normalizados, cada histograma corresponde a una fila de la imagen. En ella, se ofrece una información bastante robusta acerca de la luminosidad de la misma, la solución propone analizar el rostro como una secuencia del nivel de luminosidad desde la parte superior a la parte inferior del mismo. Por sus características, las redes espacio-temporales son una buena herramienta para reconocer estas secuencias ordenadas con una gran tolerancia a fallos, por lo que se considera adecuada para la naturaleza de la señal de entrada (secuencia de vectores de histogramas normalizados).

La solución presentada se describe como sigue: la imagen es la entrada a la primera capa (capa de entrada), que calcula los histogramas correspondientes a cada una de las filas de la imagen, y procede a normalizar dichos histogramas. Una capa interna recibe la salida de la capa de entrada. Dicha capa interna es una red espacio-temporal. Finalmente, una capa de salida (*outstar*), tiene como función elegir el rostro ganador de entre todos los rostros con los que se entrenó.

Para la verificación de resultados se preparó una serie de pruebas, formada por 16 series de 6 imágenes cada una. Las imágenes tienen una dimensión de 112 filas por 92 columnas y 256 niveles de gris. Dichas series se muestran en la figura 2.3.

El aprendizaje se ha realizado mostrando al sistema únicamente la primera imagen de cada serie. Los resultados obtenidos se indican en la tabla III. Un símbolo “x” indica que la imagen de la serie correspondiente no ha sido reconocida.

Tabla III. Resultado de las imágenes reconocidas en toda la serie.

	Series															
Imagen	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
1																
2																
3			x									x	x	x	x	x
4			x	x				x	x			x	x		x	x
5			x	x			x	x	x			x		x	x	x
6			x	x				x	x						x	x

Como se puede observar, 6 de las 16 series son reconocidas totalmente. Todas ellas tienen en común que la luminosidad se mantiene constante en toda la serie.

Algunas veces, en series en las que se aprecia un cambio importante de luminosidad, se producen fallos en la identificación; véase como ejemplo en la figura 2.3, la serie 16 en las imágenes 4 y 5. Otra situación que provoca fallos es que en una misma serie el individuo pueda aparecer con o sin gafas. Véanse como ejemplo, las series 4 y 13 de la figura 2.3, aunque hay casos en que no representa un problema.

Este método presenta una gran tolerancia a fallos, obteniéndose muy buenos resultados cuando se produce un giro o movimiento de la cabeza que no afecta a la luminosidad de la imagen. Véanse como ejemplo las series 1, 5 y 11 de la figura 2.3. Las series elegidas son representativas de un gran abanico de rostros posibles (mujeres, hombres, con o sin barba, calvos o con pelo, con gafas o sin ellas, rubios o morenos, etc.), obteniendo un resultado global más que aceptable. Así, de 96 imágenes, sólo se han presentado 29 fallos en la identificación, lo cual da aproximadamente un 70 % de aciertos. Hay que tener en cuenta que no se ha realizado ningún tipo de preprocesamiento a las imágenes para intentar corregir o mejorar ciertas características, por ejemplo la luminosidad.

Este artículo permite comprender que la gama de soluciones para el reconocimiento de rostros es inmenso, que en términos generales las imágenes que se van a utilizar necesitan una normalización. En otras palabras, las imágenes deben tener características específicas o tener una serie de estándares referentes a la dimensión, color, brillo, fondo, iluminación, etc.

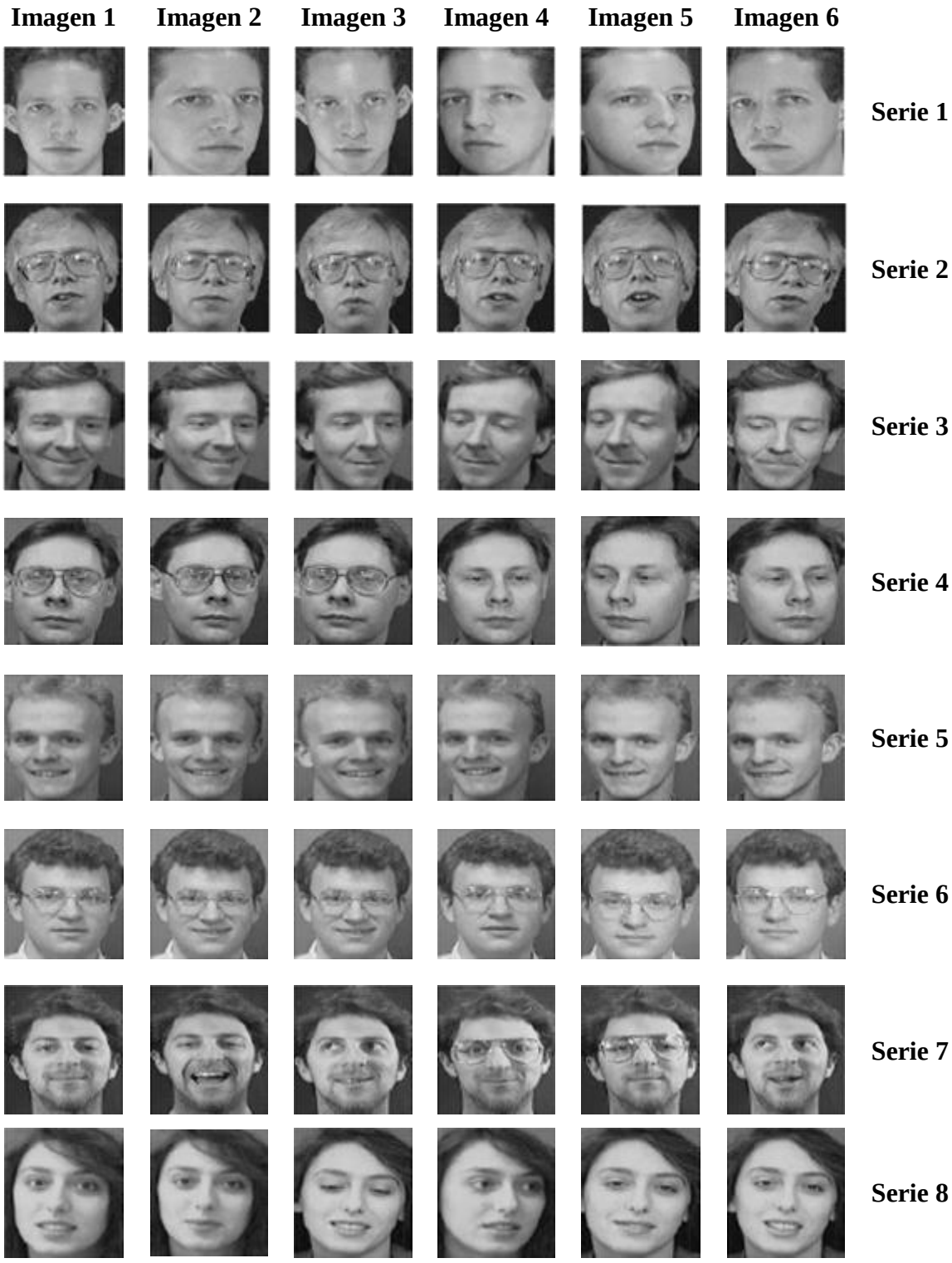


Figura 2.3. Series de imágenes.

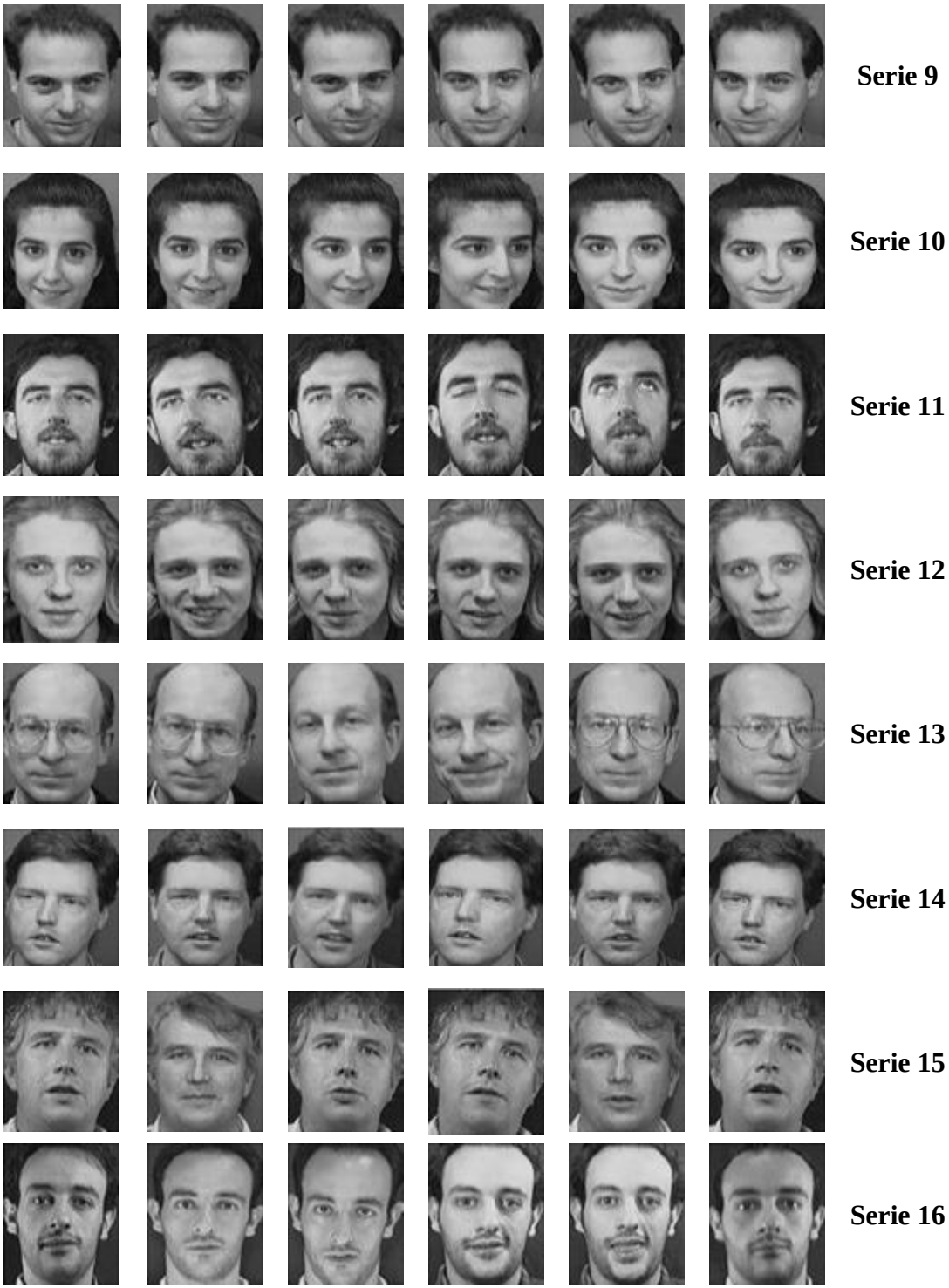


Figura 2.3. Series de imágenes (continuación).

Reconocimiento holístico de rostros a través de análisis multivariado y algoritmos genéticos: resultados preliminares (Kuri 2002)

En este trabajo se presenta una nueva técnica que utiliza un enfoque holístico para el reconocimiento de rostros, sin tener que recurrir a la obtención de características geométricas o de regiones. La caracterización del rostro se realiza a partir de un muestreo aleatorio de diversos atributos seleccionados para los pixeles constituyentes de la imagen del rostro. A partir de dicha información, se constituyen tres grupos de datos (espectros), correspondientes a los valores de las frecuencias bajas, gradiente y entropía de la imagen.

El reconocimiento de un rostro, formulado desde el punto de vista de la clasificación de patrones, se puede enfocar como "un problema de aprendizaje". Dicha problemática se puede resolver a través del uso de una función de aproximación multivariada que proporcione el mínimo error de ajuste con respecto al espacio muestral.

Una función de aproximación es una forma de producir un clasificador a partir de un conjunto de datos de entrenamiento. A través de un algoritmo genético se obtienen los coeficientes y el grado de tres polinomios de aproximación que minimizan el error de ajuste de los datos correspondientes a los valores de los tres espectros (frecuencias bajas, gradiente y máxima entropía) por medio de los cuales se caracterizan los rostros de la base de entrenamiento. Un determinado rostro a identificar se caracteriza también por medio de los tres polinomios mencionados y se compara contra la base de entrenamiento mediante una métrica de Mahalanobis, con lo cual se logra el reconocimiento de rostros utilizando una sola imagen por sujeto para el entrenamiento, con una precisión del 97.5 %.

En esta investigación se utiliza un enfoque de aprendizaje supervisado. El aprendizaje supervisado consiste en la adquisición de reglas de clasificación a partir de ejemplos, en este caso, se utilizó un conjunto de 40 imágenes frontales de rostros (base de datos), las cuales se caracterizaron como ejemplos de entrenamiento. A partir de esta base de datos se realiza una comparación con el rostro a identificar para encontrar similitudes a un nivel holístico (global), sin tener que recurrir a la extracción de características geométricas del rostro o a la utilización de "plantillas" de ciertas regiones (ojos, nariz, boca, etc.), métodos tradicionalmente utilizados en el reconocimiento de rostros.

En el aprendizaje supervisado, se proporcionan ejemplos de la forma (x_i, y_i) y se puede asumir una función de aprendizaje *f*, tal que, $f(x_i) = y_i$. El objetivo consiste en

encontrar la función *f*, de tal manera que dicha función capture "los patrones generales" presentes en los datos de entrenamiento y se pueda aplicar para predecir valores de *y*, a partir de diversos valores de *x*.

Generalmente, cada x_i es una descripción de algún objeto, situación o evento, al igual que las y_i. La función se puede hacer extensiva al manejo de diversas variables en un espacio "n-dimensional".

Los valores utilizados para la función, que caracterizan un rostro, pueden ser atributos de los pixeles de cada imagen (coordenadas, nivel de gris o color, gradiente, ruido, etc.). Utilizando el enfoque anterior, se está en posibilidad de caracterizar un rostro de una manera holística, sin tener que especificar elementos geométricos o plantillas de regiones.

Como se ha mencionado anteriormente, a través de una función se pretende caracterizar un determinado rostro. Específicamente, se seleccionó una familia de funciones polinomiales de aproximación, cuyo propósito fue caracterizar un rostro a partir de una serie de atributos de los pixeles de cada imagen. De estos atributos, se seleccionó uno como variable dependiente y otro (u otros, en el caso multivariado) se plantearán como variables independientes.

La aproximación del polinomio (a los atributos seleccionados) tradicionalmente se ha realizado mediante la técnica de regresión lineal o regresión múltiple. Sin embargo, para aplicar dicho método se presupone que los datos cumplen con ciertas características: las variables deben tener una distribución normal, las distribuciones deben tener la misma varianza. Para un valor de la variable independiente, la distribución de los valores de la variable dependiente debe tener una media que se encuentre en la línea de regresión, etc.

Muchos problemas de la vida real, entre ellos los datos de un rostro humano, no cumplen con las características anteriormente descritas. Por lo tanto, una forma de resolver el problema, es obtener un aproximado, visto como un problema de optimización. El objetivo consistió en encontrar la forma y los valores de los coeficientes del polinomio que mejor caracterizan la interrelación entre la(s) variable(s) independiente(s) y la dependiente bajo una determinada norma. La solución a la problemática anterior cae dentro del área que se ha denominado **optimización combinatoria**, y es un problema de difícil resolución por métodos tradicionales.

Para realizar la búsqueda y optimización respectiva, se utilizó un método propuesto por Kuri, denominado **algoritmo genético de orden**, el cual permite encontrar la forma y los valores de los coeficientes del polinomio de aproximación en el espacio de búsqueda, de tal forma que se minimice el máximo error absoluto de aproximación entre los datos y la función aproximante. La forma polinomial que es posible obtener es del tipo mostrado en la ecuación 2.

$$\boldsymbol{f(v_1, \ldots \ldots v_p) = \sum_{i_1=0}^{g_1} \ldots \ldots \ldots \sum_{i_p=0}^{g_p} C_{i_1 \ldots\ldots\ldots i_p} v_1^{i_1} \ldots \ldots v_p^{i_p},} \tag{2}$$

donde:

V_i = serie de atributos de los pixeles de cada imagen.
C_i = matriz de covarianza para cada imagen.

Por tanto, en la fase de entrenamiento, se caracterizarán mediante polinomios los rostros de la base de datos. Esta aproximación se realizará a partir de diversos atributos de cada rostro, los cuales se obtuvieron de una muestra de pixeles de cada rostro.

Posteriormente, un determinado rostro a identificar, se caracterizará también mediante una familia de polinomios, los cuales se comparan con los polinomios que corresponden a los rostros de la base de datos (entrenamiento) utilizando la métrica de Mahalanobis. La cantidad *r* (Ec. 3)

$$\boldsymbol{r^2 = (x - m_x)^T C_x^{-1} (x - m_x),} \tag{3}$$

es llamada la distancia de Mahalanobis, a partir del vector de características *x* al vector de medias m_x , en donde C_x es la matriz de covarianza para *x*. Se puede utilizar la distancia de Mahalanobis en un clasificador de distancia mínima en la siguiente forma: dado que $m_1, m_2, \ldots . m_c$ con los valores "medios" para las "c" clases, y $C_1, C_2, \ldots . C_c$ son las correspondientes matrices de covarianza, se puede clasificar un vector de características *x* al medir la distancia de Mahalanobis desde *x*, a cada una de los valores "medios" de las clases.

Entonces se asignará *x* a la clase para la cual la distancia es mínima (Fig. 2.4).

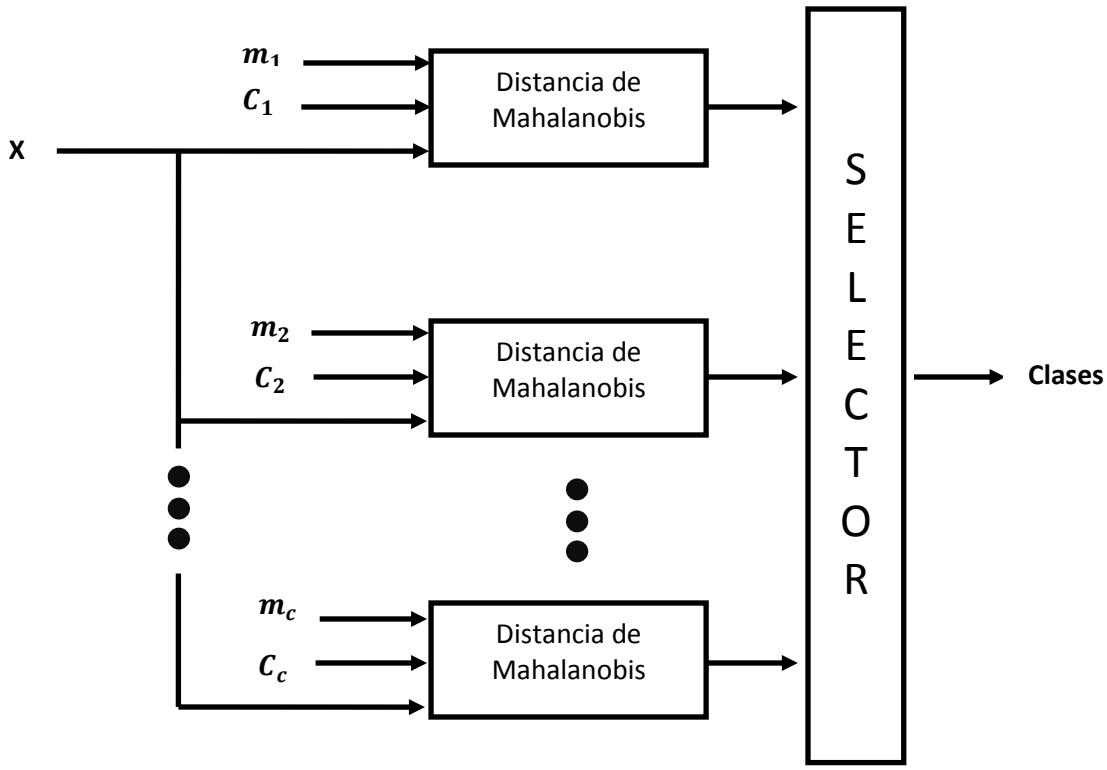


Figura 2.4. Clasificador Mahalanobis de distancia mínima.

Dicho clasificador se ha utilizado en la investigación para realizar el reconocimiento de un determinado rostro (x), identificando si éste se encuentra en la base de datos de rostros (denominada ORL generada en los Laboratorios AT&T de la Universidad de Cambridge, Inglaterra).

Los resultados que se obtuvieron en el reconocimiento de rostros con la base ORL fue de 97.5 % de precisión, lo cual se considera altamente satisfactorio. Sobre todo considerando que únicamente se utilizó una sola imagen por sujeto en la fase de entrenamiento (a diferencia de la mayoría de los métodos reportados en la literatura que requieren varias imágenes para el entrenamiento). Por lo tanto, se puede concluir que el método es el más robusto.

Sistema de reconocimiento de rostros (Villa 2007)

El objetivo de este trabajo fue crear un sistema computacional capaz de reconocer rostros a partir de imágenes faciales capturadas a través de una cámara web. El sistema compara

paramétricamente la imagen adquirida con aquellas almacenadas en una base de datos (usuarios registrados).

Para ello, se desarrolló una aplicación conformada por dos partes, la primera un programa desarrollado en Matlab, utilizando la interfaz visual *Guide* y la segunda por hardware, la cual está integrada por una computadora conectada a una cámara web para la captura de las imágenes de los rostros. La cámara web está sobre una plataforma, la cual a su vez tendrá una base donde la persona colocará el rostro para que pueda capturar la imagen respectiva. Además se coloca una lámpara fluorescente en la parte superior para asegurar una adecuada iluminación. El diseño consistió en una caja, donde se encontraba la cámara web en un extremo y en el otro el soporte del rostro. El esquema de la base propuesta se muestra en la figura 2.5.

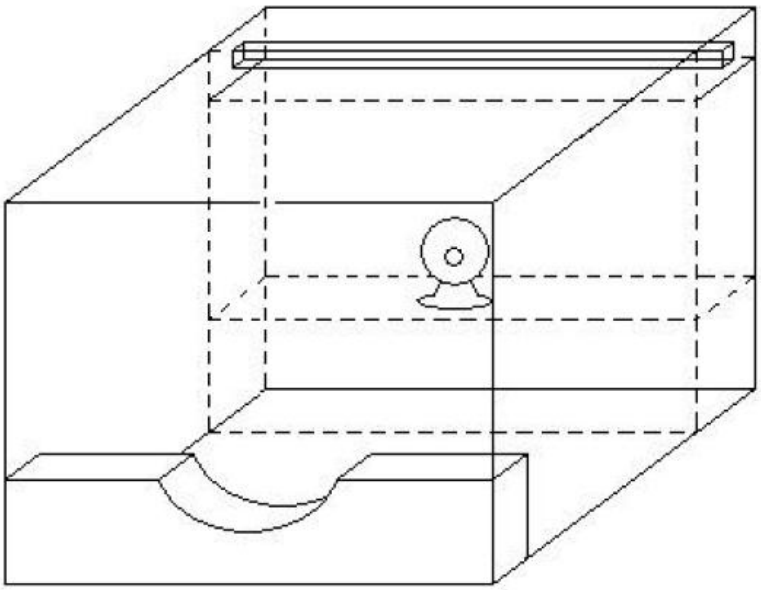


Figura 2.5. Base para la captura de imágenes.

En la parte de software, se ha implementado un algoritmo que permite segmentar las imágenes capturadas para obtener únicamente la imagen del rostro. De esa forma se descarta información de otro tipo que puede ser no relevante para los objetivos del programa. Posteriormente, se procede con la descomposición de las imágenes para luego aplicarles la técnica de Análisis de Componentes Principales (ACP), con lo cual finalmente se procede a realizar el reconocimiento.

En general los temas involucrados son los siguientes:

Segmentación de imágenes. Una vez que la imagen es capturada con la cámara web, se procede con el algoritmo de segmentación, el cual está basado en las proyecciones de las derivadas de las filas y columnas de los valores de la imagen.

Previamente, se le aplica el filtrado de mediana a la imagen, para quitar el ruido y suavizar la imagen. Luego se procede a recuantizar la imagen a dos bits, para que los cambios sean más bruscos y de esta manera las derivadas se detecten mejor. Luego, se obtiene el negativo de la imagen recuantizada para que el fondo y todo elemento no perteneciente al rostro sean oscuros y con ello facilitar la segmentación (Fig. 2.6).

(a) (b) (c)

Figura 2.6. Segmentación de imágenes. (a) Filtrado de la mediana, (b) Recuantizada a dos bits, (c) Negativa.

Descomposición *wavelet*. La descomposición *wavelet* es una técnica que se utiliza para descomponer la imagen de entrada en cuatro sub-imágenes. En el programa se utiliza la descomposición *wavelet* para reducir las dimensiones de las imágenes sin perder la información facial necesaria. El resultado de la descomposición está conformado por cuatro sub-imágenes, que contienen detalles principales, detalles verticales, detalles horizontales y detalles diagonales. Para este trabajo, se descompuso la imagen dos veces, por lo cual se obtienen cuatro sub-imágenes, cuyas dimensiones son cuatro veces menor que la imagen original; sin embargo, los detalles principales no se pierden. Para el programa, se utiliza la sub-imagen con los detalles principales y se le normaliza, a fin de que el rango de grises, se encuentre entre 0 y 255.

En la figura 2.7 se observa la descomposición usando *wavelets* de una imagen facial.

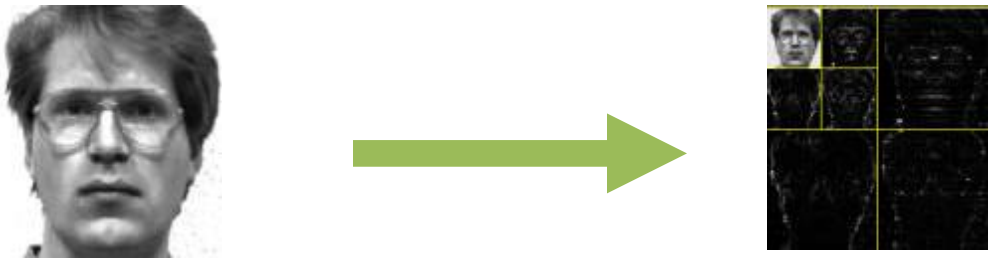


Figura 2.7. Descomposición *wavelet* de una imagen facial.

Análisis de componentes principales. Es un método matemático que da origen a la técnica *eigenfaces*. El método consiste en la recolección de imágenes de rostros de varias personas que son luego combinadas y convertidas en una matriz.

Los vectores que conforman esta matriz son los denominado vectores *eigenfaces*. Estos pueden ser combinados adecuadamente para reconstruir cualquier imagen facial del conjunto.

Los resultados reportados fueron de un 92.26 % de imágenes reconocidas, por lo cual se considera un buen resultado.

Lo importante de este artículo, y que se consideró para esta tesis, es que uno de los métodos para eliminar ruido de la imagen es el "filtrado de la imagen", también se consideró que estandarizar las imágenes es importante ya que se eliminan pixeles innecesarios de la imagen, es decir, normalizar la imagen para que sólo muestre la cara y no dejarle más información para procesar.

Se consideró que el método de ACP, es un tema avanzado que requiere de más tiempo de estudio, para resolver el problema de reconocimiento de rostros utilizando este método.

Detección de caras y localización de características faciales para reconocimiento biométrico (Landesa & Alba 2007)

El objetivo fundamental de este trabajo es el desarrollo algorítmico de un módulo de detección de caras, orientado a su integración en un sistema remoto de reconocimiento biométrico y monitorización de usuarios, a través de imágenes tomadas con una cámara web y transmitidas a través de Internet. Este tipo de aplicación presenta claras restricciones

de tiempo real, de forma que la eficiencia probada del algoritmo de Viola-Jones es la piedra angular sobre la que se sustenta el diseño preliminar del sistema.

Desde este punto de partida, la manera más intuitiva de conectar la secuencia de vídeo procedente de la cámara web con el módulo de reconocimiento, fue aplicando un detector de caras sobre cada *frame*. Sin embargo, los algoritmos de reconocimiento requieren cierta normalización de la cara (fotométrica y geométrica) previa al proceso de identificación, para compensar determinadas variaciones no deseadas en las condiciones de iluminación (intensidad, temperatura de color, directividad, número de fuentes, etc.) y en la geometría de la cara (escala, pose, orientación, margen de recorte, etc.).

La normalización geométrica es la encargada de gestionar este último grupo de problemas, que en última instancia pueden tratarse partiendo de la localización de las características faciales (ojos, nariz y boca). En esta etapa los autores decidieron aplicar el algoritmo de Viola-Jones para encontrar dichas características faciales dentro de cada región (cara) candidata.

La tarea consistió en detectar las caras presentes en cada imagen (frame), y dentro de cada una de ellas encontrar la posición y tamaño de sus ojos, nariz y boca (Fig. 2.8). Este conjunto de datos conformó la información de entrada al proceso de normalización geométrica, previo a la etapa de reconocimiento (ambas tareas desempeñadas en otro módulo de desarrollo).

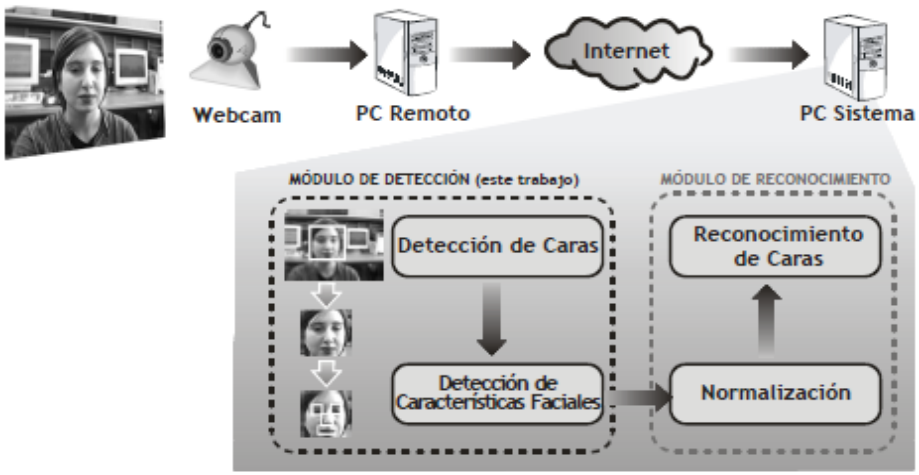


Figura 2.8. Esquema del sistema global de identificación biométrica remota.

Es importante destacar que todo el módulo fue desarrollado desde cero, programando tanto la plataforma de entrenamiento como el detector en Matlab, para

facilitar la experimentación e introducción de variaciones y parámetros de forma flexible en todo el desarrollo. El objetivo de esta fase preliminar fue únicamente algorítmico, de modo que sólo se evaluaron las prestaciones del sistema, dejando aparte las cuestiones relacionadas con la implementación en tiempo real para cuando el detector sea implementado en un lenguaje de programación apto para ese tipo de aplicaciones (C, C++, etc.).

El esquema algorítmico que se implementó para construir los detectores está inspirado en el modelo de Viola-Jones y en las posteriores modificaciones propuestas por Lienhart. Estos detectores se basan en una cascada de clasificadores que es explorada por toda la imagen en múltiples escalas y localizaciones. Cada etapa de la cascada se basa en el uso de simples características tipo Haar (eficientemente computadas utilizando la imagen integral) seleccionadas y combinadas mediante AdaBoost durante el entrenamiento. La eficiencia de este esquema reside en el hecho de que los negativos (la inmensa mayoría de las ventanas a explorar) van siendo eliminados progresivamente (Fig. 2.9), de forma que las primeras etapas eliminan un gran número de ellos (los más fáciles) con muy poco procesamiento. Esto permite que las etapas finales tengan tiempo suficiente para encargarse de clasificar correctamente los casos más difíciles.

Figura 2.9. Cascada de detectores propuesta por Viola-Jones.

Se utilizaron 7200 caras diferentes, 2400 seleccionadas aleatoriamente de tres bases de datos diferentes: AR Face Database, XM2VTS Face Database y BioID Face Database, la mitad de ellas previamente reflejadas.

El tamaño de detección base es 24x24 pixeles, el mismo utilizado por Viola-Jones. También utilizaron imágenes (no caras) de múltiples colecciones de imágenes descargables de Internet. Cada etapa de la cascada fue entrenada con 5000 caras y 5000 no caras (obtenidas de los falsos positivos de las etapas previas) y probada sobre un conjunto de pruebas compuesto de 1000 caras y 1000 no caras. La cascada fue dimensionada con 11

etapas, exigiendo una tasa mínima de detección del 91.54 % y una tasa máxima de falsos positivos de $1.2603x10^{-6}$ (ambas calculadas sobre los conjuntos de pruebas).

En el entrenamiento para los detectores de ojos, nariz y boca (Fig. 2.10), se utilizaron recortes de las tres mismas bases de datos mencionadas anteriormente. Al igual que en el detector de caras, estos recortes fueron extraídos de 2400 imágenes escogidas aleatoriamente de cada base de datos, siendo la mitad de ellas previamente reflejadas. El conjunto de negativos es el mismo para detección de caras.

Figura 2.10. Imágenes de entrenamiento de caras, ojos, nariz y boca.

Para el análisis del funcionamiento del detector de caras y compararlo con otros sistemas, tomaron un conjunto de fotografías de la CMU Frontal Face Database, compuesto de 130 imágenes con 507 caras. La tabla IV resume los resultados obtenidos.

Tabla IV. Prestaciones del detector sobre la base de datos CMU.

Prestaciones	Resultados
Tasa de detección	71.94 % (364 caras de 507)
Tasa de falsos positivos	$1.96x10^{-6}$(60 falsos positivos)
Acierto	99.9993 %

Las prestaciones del detector, aunque sensiblemente inferiores a las del sistema original de Viola-Jones (para una tasa similar de falsos positivos alcanzan un 92 % de detección) es bastante satisfactorio, teniendo en cuenta el estado preliminar de este trabajo y que el detector de Viola-Jones utiliza un total de 6060 características tipo Haar en 38 etapas, frente a las utilizadas en esta investigación, 321 en 11 etapas.

Para comprobar el método de localización de características faciales se escanearon los detectores de ojos, nariz y boca sobre un subconjunto de 137 caras de la CMU Frontal Face Database.

Las caras se corresponden con aquellas de tamaño mayor o igual a 48x48 pixeles que el detector es capaz de encontrar, y la decisión de las localizaciones correctas o incorrectas se hace comparando los centros de los positivos detectados con la información de etiquetado de la CMU.

Landesa & Alba (2007) reportan en este artículo que el sistema localiza correctamente todas las características faciales (Fig. 2.11) en el 72.26 % de las caras y sólo falla en una característica en un 20.44 % de ellas.

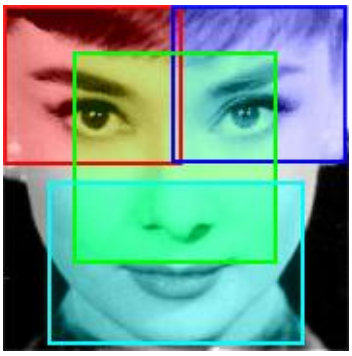


Figura 2.11. Zonas de interés ojo izquierdo, ojo derecho, nariz y boca.

Este artículo indica las capacidades del modelo de Viola-Jones, debido a las características del sistema que se realizó, se necesitaba un modelo robusto para desarrollar un sistema con los requerimientos que se mencionan en el artículo. En la tabla V se muestra una comparativa de las prestaciones obtenidas por el sistema de Viola-Jones.

Tabla V. Resultado de localización.

Prestaciones	**Ojos**	**Nariz**	**Boca**
Tasa de localización correcta	94.16 %	94.89 %	81.02 %
Tasa de localización incorrecta	5.84 %	2.92 %	10.95 %
Tasa de no detección	0 %	2.19 %	8.03 %
Porcentaje total	100 %	100 %	100 %

Las ideas a destacar del artículo que se implementaron en esta tesis son la normalización que realizan a la cara (imagen), normalización geométrica y fotométrica, la primera enfocada a la geometría de la cara (escala, pose, orientación, margen de recorte, etc.), la segunda enfocada a las variaciones no deseadas en las condiciones de iluminación (intensidad, temperatura de color, directividad, número de fuentes, etc.).

No se consideró usar el método de Viola-Jones, debido a que los requerimientos del sistema a desarrollar son de otra naturaleza (características), que se pueden resolver con otros métodos. Otra de las ideas que se tomó en cuenta es un segundo entrenamiento de la red neuronal, para comparar los resultados más óptimos y definir el número de veces para entrenar la red neuronal.

2.2. Justificación

Las herramientas de software que brindan este tipo de solución y que se encuentran en el mercado tienen costos muy elevados.

El proyecto final podría utilizarse como base para proyectos futuros para aplicarse en la Universidad del Mar, con el fin de identificar a sus alumnos, maestros o empleados en general, ya que dicha institución no cuenta con un software especializado o enfocado a esta parte de seguridad.

En dependencias de gobierno, instituciones privadas, empresas, universidades, etc., hay áreas de seguridad a las cuales sólo personal autorizado puede entrar, y con sólo una fotografía del rostro del empleado o de algún cliente, se podría autentificar si la persona tiene acceso a esa área de la empresa o institución.

Con la ayuda de una imagen del rostro de una persona, se calcularían las métricas necesarias del rostro para identificar a la persona, o deducir que no pertenece al personal de la institución o empresa, y con ello se puedan tomar las medidas necesarias para resolver la situación.

Este software pretende ser la punta de lanza de proyectos en la Universidad del Mar, que permitan el uso de técnicas de identificación de personas en algunas de las actividades del ser humano, como por ejemplo: en los supermercados (autentificación de clientes), en las cárceles para identificación de reos o visitantes, en las escuelas para identificar alumnos, personal, maestros, etc., ya que el sistema desarrollado necesita imágenes en las cuales el ambiente sea controlado.

2.3. Planteamiento del problema

En la Universidad del Mar se han realizado proyectos como seguimiento de objetos (Toscano 2009), restauración de imágenes (Franco 2011) e incluso simulación de dispersión de contaminantes (Sosa 2010), pero hasta el momento no existen proyectos, al interior de esta casa de estudios en el área de procesamiento digital de imágenes (PDI), que se enfoquen en el reconocimiento o identificación de personas por medio de imágenes.

Partiendo de lo anterior surge la inquitud de desarrollar un sistema que permita el reconocimiento de rostros de personas, como base para futuros proyectos enfocados a la identificación de individuos.

En el presente trabajo se utilizó como metodología de solución, los siguientes pasos:

- Adquisición de imágenes.
- Preprocesamiento de las imágenes.
- Creación de archivo de distancias.
- Entrenamiento de una RNA tipo Perceptron.
- Reconocimiento de rostros con la RNA.
- Escribir los resultados.

Para dar solución a este problema se tuvo la necesidad de apoyarse en áreas del PDI, las cuales se describen a continuación.

Áreas involucradas

Procesamiento digital de imágenes

El PDI comprende un amplio rango de hardware, software y recursos teóricos. Todos estos elementos necesitan de una serie de pasos o tiempos en que se debe implementar tal cosa, para lo cual se muestra un diagrama de etapas necesarias a considerar (Fig. 2.12).

Figura 2.12. Diagrama de las etapas fundamentales del PDI (González & Woods 1992).

El dominio del problema implica un conocimiento general y detallado de lo que se quiere, eliminando datos irrelevantes al problema, delimitando pero abarcando los objetivos. Este dominio engloba las posibles formas de resolver el problema habiendo elegido la más idónea.

La primera etapa del proceso es la **adquisición de la imagen**, esto es, digitalizarla, capturar el objeto de estudio para tener una imagen digitalizada. Una vez que se tiene la imagen, la siguiente etapa es el **preprocesamiento**, su función es mejorar la imagen para tener éxito en las próximas etapas. El preprocesamiento primordialmente implica técnicas de realzado y eliminación de ruido en las imágenes, sin perder las características principales de los objetos de la escena.

La siguiente etapa es la **segmentación**, es decir, particionar una imagen de entrada en sus partes homogéneas u objetos. Generalmente la segmentación automática es una etapa difícil para el procesamiento de imágenes digitales. Referente al reconocimiento de rostros es el de extraer áreas homogéneas del rostro resaltando las características físicas. Al final de esta etapa se tiene un conjunto de pixeles de las regiones segmentadas; decidir cómo se deben representar o tomar los datos de las regiones, decidir que puede servir del contorno o si es necesaria toda la región, esto depende del interés y problema a resolver. Esta decisión es sólo una parte de la solución para transformar los datos en bruto a datos

que pueda procesar una computadora, también se debe especificar un método para describir los datos de forma que resalten aquellos que sean de interés.

La **representación y descripción**, consiste en extraer información de interés que ayude a diferenciar una clase de objetos de otra. El último estado comprende el **reconocimiento e interpretación**.

El **reconocimiento** es el proceso que asigna un nombre o etiqueta a los objetos basándose en la información relevante de las anteriores etapas. La **interpretación** se encarga de asignar un significado a un conjunto de objetos etiquetados y así diferenciar las diversas clases de objetos que se lleguen a tener.

El módulo central o **base de conocimiento** es donde se tiene toda la información acerca del problema a resolver. Este conocimiento puede ser tan simple, como tener sólo la información de interés de las regiones de la imagen que limitan la solución. También se puede tener una base de conocimientos amplia como todas las regiones segmentadas de la imagen del rostro y detallando más aun en las interacciones de las regiones de la escena.

La **base de conocimiento** es la que interacciona proporcionando información relevante a las demás etapas, es de este módulo de información de donde toman los datos necesarios todas las etapas, para adquirir el conocimiento previo de cómo debe ser el resultado. Este conocimiento además de guiar la operación de cada módulo, ayuda a las operaciones de retroalimentación entre módulos.

En muchas aplicaciones prácticas, todas las etapas de PDI no son utilizadas implícitamente, es decir, en ocasiones no son necesarias todas las etapas para lograr los objetivos deseados del problema, por ejemplo, el realce de bordes de una imagen digital para la interpretación humana no avanzará sino hasta la etapa de preprocesamiento (Fig. 2.12).

Elementos de los sistemas de procesamiento de imágenes digitales

En la figura 2.12 se muestran generalizados los elementos que tiene un sistema para realizar el procesamiento de imágenes digitales.

Como se puede ver en la figura 2.13, los elementos de hardware, software y de conocimiento teórico, están organizados en los siguientes módulos: adquisición, presentación, almacenamiento, comunicación y procesamiento, para brindar un óptimo

funcionamiento al sistema. Por lo tanto, un sistema de procesamiento digital de imágenes lleva a cabo las siguientes etapas (González & Woods 1992).

Figura 2.13. Elementos de los sistemas de PDI (González & Woods 1992).

Adquisición de imágenes. Para obtener las imágenes deben existir dos elementos principales, el primero es un dispositivo físico que ayude a captar la radiación electromagnética (es decir captar alguna longitud de onda como son: rayos cósmicos, rayos Gamma, rayos X, rayos ultravioletas, espectro visible, infrarrojos, microondas, radio o frecuencias extremadamente bajas). Actualmente, la mayoría de los dispositivos utilizados para este fin captan una longitud de onda que va desde 380 nm hasta los 780 nm (1nm = 1 nanómetro = 0.000001 mm), que es el espectro visible que puede percibir el ojo humano e

interpretar el cerebro. Este dispositivo convierte esa información en señal eléctrica, que es proporcional a la longitud de onda percibida. El digitalizador es el segundo dispositivo que convierte esa señal eléctrica en información digital, es el encargado de crear un formato digital para que sea utilizado posteriormente por una computadora.

Los dispositivos que entran en esta categoría, y que ayudan a la adquisición de imágenes son: las cámaras CCDs, las cámaras de video, las cámaras web, las cámaras IP (Internet Protocol o cámara de red) y los escáner.

La elección de la cámara dependerá primordialmente de problema a resolver y de los resultados que se desea obtener. Las cámaras IP y las cámaras web actualmente son utilizadas para monitoreo y vigilancia, de igual forma los demás dispositivos tienen sus aplicaciones en diversas áreas científicas.

Almacenamiento. Los dispositivos de almacenamiento son una prioridad para el almacenamiento de la información que el sistema va a utilizar, una imagen BMP (archivo en mapa de bits) de 384 x 512 pixeles necesita 380 Kb de espacio en disco duro, la capacidad de estos dispositivos tiene que ser amplia para albergar toda una base de datos, las imágenes capturadas para el procesamiento de las mismas y toda la información adicional al sistema. En este módulo se analiza el proceso y las diferentes formas de almacenar o guardar la información en los dispositivos, como pueden ser: 1) almacenamiento a corto plazo en el momento de realizar el procesamiento o bien se le puede llamar almacenamiento en memoria principal, 2) almacenamiento a mediano plazo, para ser utilizada en un tiempo relativamente rápido o almacenamiento en disco duro y 3) almacenamiento en algún dispositivo de acceso poco frecuente.

El almacenamiento a corto plazo se refiere a guardar información en la memoria RAM de la computadora, para que ésta sea fluida y rápida de accesar al momento de realizar operaciones con las imágenes o con dicha información. Este tipo de memoria implica también la memoria caché del procesador que es la más cercana y la más rápida, y se ayuda de la memoria RAM. Es por ello que es importante que la velocidad de comunicación entre estos dispositivos sea igual, es decir, que la velocidad del bus sea igual en ambas para que la comunicación esté estandarizada de la misma forma, esto ayudará a la visualización posterior de los resultados. Para el almacenamiento a mediano plazo están los discos magnéticos, donde se guarda la información por más tiempo; en éste el sistema

puede tomar información después de realizar varios procesos en los diferentes módulos; por lo cual se debe tomar en cuenta este tipo de almacenamiento. Por último está el almacenamiento final o almacenamiento masivo, se puede almacenar la información en dispositivos como discos ópticos o cintas magnéticas, con el fin de respaldar cierta información del sistema.

Procesamiento. Para el PDI una de las prioridades radica en aplicar algoritmos necesarios para discriminar datos ambiguos, para después analizarlos y decidir acciones sobre ellos. Las técnicas de detección de bordes, eliminación de ruido, realce de bordes, realce de características importantes en la imagen, entre otros, son los algoritmos que se implementan por medio de las computadoras, más específicamente la ALU del procesador que se haya adquirido, para el resolver el problema.

Así como se muestra en la imagen de la figura 2.13, el procesamiento es quien maneja y decide el ritmo del sistema, decide capturar la imagen, guardar la información en memoria para procesarla y a la orden del software que se haya instalado en la computadora mostrar los resultados. Es por ello que los cinco elementos están íntimamente relacionados y son de principal importancia para lograr un sistema compacto que ayude a resolver problemas específicos, sin embargo, hay veces que se requieren aplicar los mismos algoritmos a otros problemas pero que no tendrán resultados favorables, es por eso que es importante seleccionar cada uno de los elementos para optimizar el rendimiento de los dispositivos (adquisición, presentación, almacenamiento y procesamiento, cámara para la adquisición de las imágenes, monitor o proyector para visualizar los procesos, la memoria RAM que debe tener la computadora a realizar el procesamiento y almacenamiento de los datos).

En la actualidad hay procesadores que pueden generar hasta 12 hilos, esto ayudará enormemente a las nuevas aplicaciones que necesiten realizar procesos en paralelo, es decir, tener 12 hilos corriendo y atendiendo diversos procesos, hoy día se necesita de gran poder de computacional, porque la interacción que existe entre la computadora y otras tecnologías es muy diversa, por ejemplo: sistemas inalámbricos, infrarrojos, distribuidos, bluetooth, adquisión de datos en tiempo real, análisis de video, etc. Se debe considerar que ahora la información ya no es local sino distribuida y esos nuevos procesadores ayudarán a resolver problemas que involucran todas las formas de comunicación de almacenamiento.

Comunicación. La comunicación en un modelo de PDI se ve reflejada en la rapidez de la adquisición de la imagen, en procesarla y en mandar a visualizar la información. Tener una estandarización de la comunicación con los dispositivos es prioridad, es decir, a la hora de adquirir el procesador, la memoria RAM, discos magnéticos, dispositivo para adquisición de la imagen o el dispositivo para visualizar los resultados, se debe considerar que tengan la misma o superior velocidad de frecuencia. La comunicación para un sistema local no se ve tan afectada al adquirir dispositivos con diferentes velocidades del bus. Pero si se enfoca a la transmisión de datos, video o voz en tiempo real y a grandes distancias, esto es prioridad para el sistema.

Presentación. En la presentación es donde se muestran los resultados o los procesos anteriores para ir visualizando el seguimiento del resultado. Los dispositivos más utilizados para visualizar o mostrar los resultados, enviando la salida de video de la computadora, es un monitor o por un proyector. Actualmente existen diversos métodos para enviar o presentar la información en un dispositivo de salida, como: enviarla a un proyector, a un monitor de televisión, a una impresora, por bluetooth a un celular, por infrarrojo a un PDA u otros dispositivos por medio de nuevas tecnologías (*Surface*: otras comunicaciones inalámbricas).

Las formas para visualizar los resultados dependen de las necesidades y requerimientos del sistema, pero influye más la velocidad de comunicación entre los dispositivos del sistema.

Color

El color es de gran importancia en la escena de estudio, aún más cada uno de los modelos de colores (RGB, YIQ, HSV, HSI, XYZ, YUV, etc.), ya que estos modelos tienen aplicaciones específicas en el procesamiento de imágenes digitales o en aplicaciones orientadas al hardware. Por ejemplo, los modelos de colores utilizados en el procesamiento de imágenes digitales son: RGB, YIQ, HSV y HSI; el modelo CMY se utiliza para impresión. Existen también otras aplicaciones de los modelos de colores que están enfocadas hacia el hardware y que son: el RGB para monitores de color y para una gran variedad de cámaras de video, el CMY para impresoras a color y el YIQ para television a color.

El modelo RGB. Este modelo se basa en la combinación de tres señales de iluminación cromática distintas: rojo, verde y azul (Red, Green, Blue). La forma más idónea de inferir o crear un color es determinar en qué proporciones combinar dichos colores, la cual se puede representar en un tetraedro (De la Escalera 2009), que se muestra en la figura 2.14; una forma para lograrlo es realizando la suma aritmética de las componentes (Ec. 4).

$$X = R + G + B \tag{4}$$

Como se puede observar, los valores RGB están en los tres vértices principales, el magenta, amarillo y cyan se ubican en los otros tres vértices, en el origen se encuentra el negro y en su opuesto el blanco, y es por esta diagonal desde el negro al blanco por donde se ubican los distintos niveles de gris existentes. En general, los colores de este modelo se sitúan en alguna parte de este tetraedro, que por conveniencia, se define que es un tetraedro unitario normalizado.

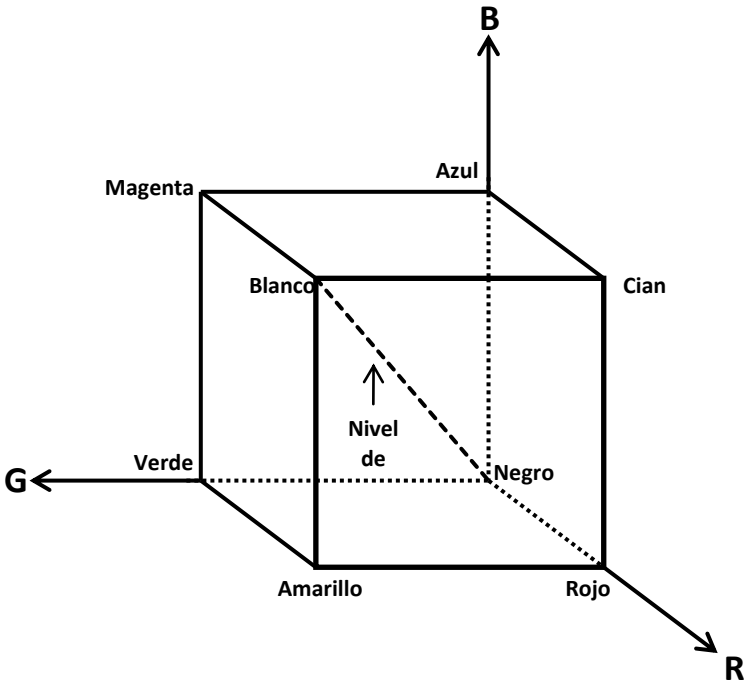


Figura 2.14. Representación del modelo de color RGB (Pajares & De la Cruz 2008).

Cuando una cámara adquiere una imagen en color, en cada pixel se tienen en realidad tres colores, uno para cada componente. La principal ventaja de este modelo es que es el más intuitivo de todos y de hecho es en el que se basan las cámaras para adquirir imágenes en color, lo que lo hace un modelo importante en el procesamiento digital de

imágenes; motivo por el cual se eligió como el modelo de color para las imágenes utilizadas en el presente trabajo de tesis. Un ejemplo de ello es el procesamiento de datos de imágenes multiespectrales aéreas o de satélite. Sin embargo, presenta inconvenientes ya que en sus tres componentes mezclan la información del color (tono y saturación) y la intensidad, por lo que para algunos problemas no es adecuado utilizar este modelo de color.

Un ejemplo de éste es que si se quiere mejorar una imagen en color de un rostro humano, pero parte del rostro está oculto por una sombra, la ecualización del histograma es una técnica sugerida para este tipo de problema, sabiendo que existen tres imágenes y a la ecualización del histograma sólo le interesan los valores de la intensidad. Esta técnica mejoraría la parte del rostro que está oculta en la sombra, pero se perderían tonos de colores del rostro por lo que a la hora de mostrarlo en algún monitor no se vería tan natural la imagen.

2.4 Objetivos

Objetivo general

Diseñar y desarrollar un sistema informático que permita la identificación de rostros en un ambiente de iluminación controlado con imágenes de rostros no invariantes a expresiones faciales, utilizando técnicas de inteligencia artificial y de procesamiento digital de imágenes.

Objetivos específicos

- Realizar un estudio de los proyectos relacionados con el tema de detección de rostros en ambientes controlados y con ello seleccionar los de mayor relevancia con el fin de analizar las técnicas que se han utilizado en la solución de problemas similares y con esas series de técnicas y herramientas respaldar teórica y prácticamente los posibles métodos utilizados en la solución del problema.
- Acondicionar el lugar y los dispositivos que se utilizaron en la captura de las imágenes.
- Tomar una serie de imágenes a rostros de personas las cuáles se utilizaron para realizar las pruebas.

- Investigar, analizar y describir las técnicas de procesamiento digital de imágenes y de redes neuronales con la finalidad de seleccionar los algoritmos adecuados que ayuden a resolver el problema.
- Editar las imágenes capturadas para estandarizarlas en cuanto a tamaño, formato, iluminación y ruido, dejándolas preparadas para las etapas siguientes.
- Escribir los resultados obtenidos en la etapa de pruebas y experimentación, así como comentarios y conclusiones generadas a partir de esta etapa.
- Implementar un módulo de preprocesamiento de las imágenes.
- Implementar un módulo de entrenamiento para la RNA.
- Implementar un módulo de reconocimiento para la RNA.
- Implementar una GUI para los módulos anteriores.

2.5 Alcances y límites del estudio

El objetivo principal del desarrollo de este proyecto es el reconocimiento de personas en imágenes de rostros, implementando una red neuronal para su reconocimiento. El sistema cuenta con tres módulos principales (preprocesamiento, entrenamiento y reconocimiento): preprocesamiento, se encarga de eliminar el ruido a las imágenes, en este mismo módulo se realiza el procesamiento de las imágenes que es cuando se aplican los algoritmos de eliminación de ruido, realce, detección de bordes, y se calculan las distancias euclideanas para formar el archivo de distancias principales; entrenamiento, se realiza la visualización de los datos de cada imagen, se muestran las distancias euclideanas, se entrena la red neuronal y se guardan en un archivo los datos resultantes (pesos y ganancias) de las neuronas; reconocimiento, en este módulo se visualiza una imagen y se realiza el reconocimiento con los datos resultantes del entrenamiento.

Alcances

El software desarrollado en este trabajo de tesis cuenta con las siguientes características:

- En la etapa de preprocesamiento se cuenta con los siguientes procesos.
    - Escala de grises.
    - Ecualización.

    - Umbralización.
    - Filtro pasa alto.
    - Filtro pasa bajo.
- En la etapa de procesamiento se tienen opciones para ejecutar diferentes algoritmos, como:
    - Operadores de Sobel.
    - Filtro de Roberts.
    - Filtro de Prewitt.
    - Filtro Frei-Chen.
    - Filtro Laplaciano.
- El sistema también cuenta con opciones para abrir una imagen con las siguientes características:
    - Formato BMP.
    - Tamaño de 384 X 512 pixeles.
- El sistema cuenta con una opción para abrir y guardar datos (distancias euclideanas de las partes del rosto) en un archivo de texto.
- El sistema cuenta con un apartado (Grid o rejilla) para mostrar los datos con los cuales se entrenará la RNA.
- El sistema muestra en el módulo de preprocesamiento la imagen original a la izquierda y la derecha muestra el resultado de aplicar los filtros o procesos.
- El sistema envía un mensaje, “La red neuronal artificial terminó de entrenar” para indicar el fin del entrenamiento.
- En el módulo de reconocimiento el sistema informa al usuario el nombre de la persona que fue reconocida.
- Se realizó la adquisición de fotografías a diferentes personas en un ambiente controlado sobre todo en iluminación.
- Se generó una base de imágenes para la fase de pruebas y experimentación.
- Se implementó un sistema con tres módulos: el primer módulo es para el preprocesamiento y procesamiento de las imágenes y el segundo es para el

entrenamiento de la red neuronal artificial y el último para reconocimiento de rostros.

- Se implementó un módulo que utiliza algoritmos de procesamiento digital de imágenes, para resaltar detalles importantes de la imagen o para eliminar ruido.
- Se implementaron métodos de cálculo de distancias, entre las partes de la cara, que se almacenan en un archivo para su posterior acceso y obtención de los datos, esto se realiza manualmente dando clic en la imagen para indicar la distancia, como son la distancia del alto de la boca, el ancho de la nariz, la distancia entre los ojos, etc.
- Se implementó la generación de una base de datos con diferente información que representa diversas medidas del rostro, como son: métricas de la nariz, ojos, boca, etc.
- Se implementaron algoritmos de inteligencia artificial para las etapas de aprendizaje y reconocimiento.
- El software se entrena con una muestra de imágenes y utilizará el resto de las imágenes para la fase de reconocimiento.

Límites

Las imágenes que se utilicen en el software, producto de este trabajo de tesis, deben tener las siguientes características:

- El tamaño de las imágenes debe ser de 384 pixeles de ancho por 512 pixeles de alto, por la limitante de la interfaz.
- El formato debe ser BMP con el modo de color RGB.
- Las imágenes se deben tomar en ambientes controlados, sobre todo iluminación.
- Las imágenes se deben tomar a la misma distancia entre la cámara y el individuo.
- Si se comete un error al calcular alguna de las distancias, se tendrá que iniciar el proceso.

Para cumplir con el objetivo del trabajo de tesis en la parte de adquisición de imágenes se tomó en consideración lo siguiente :

- Un área semicerrada con interiores en color blanco.
- Iluminación artificial en tonalidad blanca.

- Cámara digital canon powershot A560.
- Distancia entre la cámara y el rostro de 50 cm.
- Cámara montada sobre un tripie para evitar movimientos.
- Elevación horizontal de la cámara respecto al objetivo.

Para que el software sea funcional, se debe cumplir con lo siguiente:

- El límite de clases que el sistema puede llegar a entrenar es 16.
- Se debe tener por lo menos un objeto por clase.
- El total de las características que se obtengan en las imágenes deben ser 17.
- Al archivo de caracteristicas se le deben agregar los valores de salida esperados por la RNA.
- La ruta para los archivos de configuración es absoluta "C:\Archivos de programa\Reconocimiento\".

CAPÍTULO 3. MARCO TEÓRICO

Los métodos de PDI intensificaron su interés en dos áreas principales de aplicación: 1) mejoramiento de la calidad de las imágenes para la interpretación humana y 2) procesamiento de los datos de la imagen para la percepción en las máquinas de forma autónoma.

Un hito principal en la historia del PDI se ubica en 1920, cuando se querían mejorar las fotografías digitalizadas de periódico enviadas por cable submarino entre Londres y Nueva York. El sistema *Bartlane* de transmisión de imágenes redujo el tiempo necesario para enviar una fotografía a través del Atlántico de una semana a menos de tres horas. Este sistema era capaz de codificar imágenes con cinco niveles de grises de brillos distintos. Esta cantidad aumentó a 15 niveles en 1929.

Las mejoras en los métodos de procesamiento para las imágenes digitales transmitidas continuaron durante los siguientes treinta y cinco años. Sin embargo, fue el advenimiento combinado de las computadoras digitales de imágenes de gran potencia y del

programa espacial lo que puso de manifiesto el potencial de los conceptos de tratamiento digital de imágenes. La tarea de usar técnicas computacionales para mejorar las imágenes recibidas de una sonda espacial se inició en el laboratorio de Propulsión Espacial (en Pasadena, California) en 1964, cuando las imágenes de la Luna transmitidas por el *Ranger 7* fueron procesadas por una computadora para corregir diversos tipos de distorsión de la imagen, inherentes a la cámara de televisión de a bordo. Estas técnicas sirvieron como base en la mejora de los métodos utilizados para realzar y restaurar las imágenes de la Luna enviadas por las misiones *Surveyor*, la serie de misiones *Mariner* dirigidas a Marte, los vuelos tripulados *Apolo* a la Luna y otros (González & Woods 1992).

En el intento por dotar a las máquinas de un sistema de visión aparece el concepto de visión artificial (Pajares & De la Cruz 2008).

Se pueden establecer varias analogías entre la visión humana y por computadora, ya que ambas tienen un elemento sensor (el ojo y la cámara) y un procesador de la información (el cerebro y la computadora) aunque nunca perdiendo de vista que no se trata de imitar o replicar el sentido de la vista del hombre (Fig. 3.1).

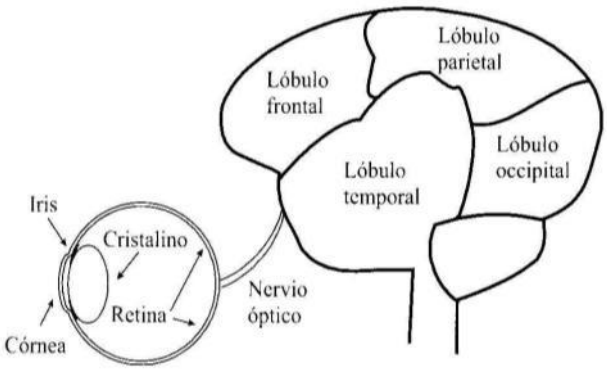


Figura 3.1. Representación de la visión humana (De la Escalera 2009).

El elemento sensor lo forma la retina, que se encuentra en la parte posterior del glóbulo ocular. Se pueden distinguir dos partes: la fóvea y la mácula (Fig. 3.2).

La primera es la zona central donde se encuentra el mayor número de elementos sensibles a la luz y que darán una zona de la imagen con una resolución alta.

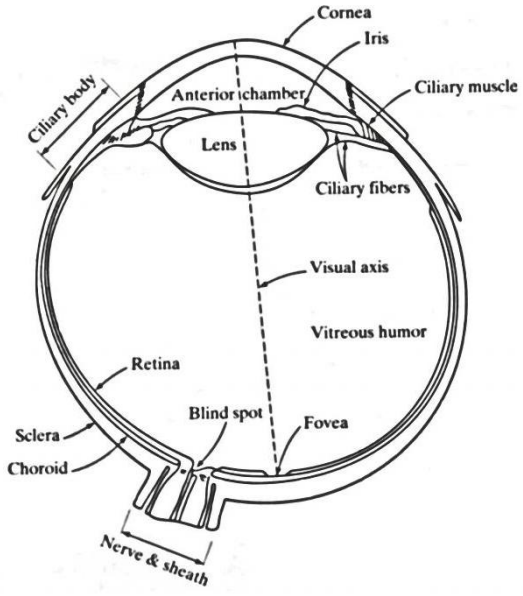


Figura 3.2. Partes principales del ojo humano (González & Woods 1992).

El área de la fóvea es muy pequeña, de 1.5 mm de diámetro y de unos dos grados de ángulo visual (aproximadamente el dedo pulgar con el brazo extendido). La mácula constituye la mayor parte de la retina y es donde la resolución es peor. Además del tamaño y resolución, ambas zonas se distinguen por las longitudes de onda de la luz a las que presentan una mayor actividad. Esto se debe a que los elementos sensores pueden ser de dos tipos: los conos y los bastones. Los primeros están distribuidos sobre todo en la fóvea y los segundos en la mácula. Los conos presentan una respuesta distinta según sea el color que perciban ya que hay tres tipos distintos, para su activación necesitan que haya una luz brillante; su número se cifra entre seis y siete millones. Cada uno de ellos está conectado a varias neuronas, por lo que la misma información espacial se transmite por distintas vías del cerebro. Los bastones no distinguen entre colores; sólo con ellos se obtendrá una imagen en escala de grises, pero en contrapartida presentan respuesta aun con poca luz. Son mucho más abundantes, entre 75 y 105 millones, pero varios de ellos se conectan a la misma neurona; por ello su misión no es tanto aumentar la resolución espacial de las imágenes como que llegue más información al cerebro. Una consecuencia es que por la noche se distinguen mejor los objetos si no se pone atención (visión foveal) directamente en ellos (visión periférica). Hay una zona del ojo, a la que llega el nervio óptico, donde no existen ni bastones ni conos por lo que, en principio, se debería ver siempre un agujero

negro en las imágenes que se observan. El cerebro "rellena" esa falta de información con la que tiene en los puntos adyacentes.

Para que la retina transmita la señal eléctrica correspondiente a la imagen, los rayos luminosos deben concentrarse en ella. La función de lente, la realizan varios elementos. El más exterior de todos es la córnea. Es un material transparente y funciona como una lente fija. El iris o pupila controla la apertura dejando pasar más o menos luz a la retina. Después del iris se encuentra otra lente (el cristalino) que a través de los músculos ciliares puede cambiar de forma, por lo que el ojo puede ir enfocando sucesivamente objetos que se encuentran en diferentes distancias.

Una vez que la señal luminosa ha sido transformada en eléctrica es procesada por el cerebro. Los nervios que conectan el ojo con éste llegan al córtex visual primario (situado en el lóbulo occipital) y se distinguen en dos grandes grupos según su mayor o menor sensibilidad al movimiento presente en las imágenes. Estos nervios tienden a mantener el contraste (la relación de los valores más altos, bajos y el valor medio) de la imagen constante, por lo que aumentan o disminuyen su ganancia. El córtex visual primario es una zona de unos 24 cm^2 y tiene $1.5x10^8$ neuronas. Al hemisferio derecho del córtex visual llega sobre todo información de la parte izquierda del campo visual de los ojos y viceversa. Aunque su función para la vista está clara para todos los mamíferos, su importancia es distinta. Así, un daño en esa zona no afecta al comportamiento de los gatos, en los monos produce una ceguera inicial, aunque con el tiempo recupera parte del sentido de la vista y en el hombre causa una ceguera total y permanente.

En el cerebro se realiza una labor de extracción de características de la imagen. Para ello existen zonas especializadas que responden mejor a un tipo de características que a otras. Así, las zonas parietales responden mejor a modificaciones espaciales de los objetos: dónde se encuentran en la imagen, con qué tamaño, si presentan diferentes orientaciones y que zonas de la imagen son estáticas y en cuales existe movimiento. El lóbulo inferior temporal responde a los diferentes colores que hay en la imagen, las texturas, formas de los objetos y tiene una zona especialmente dedicada al reconocimiento de caras. El lóbulo occipital realiza una separación de los objetos del fondo (similar a la separación que se hace en música entre melodía y acompañamiento) bien por agrupación de regiones de valor uniforme o, por lo contrario, zonas con una fuerte discontinuidad.

Con la fusión de esta información se tiene la capacidad de reconocer e identificar objetos a pesar de las siguientes limitaciones:

- Los objetos se encuentran a distintas distancias.
- Los objetos están a la misma distancia pero distinto tamaño.
- Los objetos están rotados.
- Se encuentran en distintas posiciones dentro de la imagen.
- Los objetos están ocultos parcialmente.
- Las imágenes están parcialmente degradadas.
- Generalizar y particularizar al mismo tiempo.

Debido a la importancia del sentido de la vista humana, era lógico que, con la aparición de las primeras computadoras, una de las primeras aplicaciones en las que se investigara fuera la visión artificial: el análisis de imágenes a través de computadoras para obtener una descripción de los objetos físicos que son captados por una cámara.

Los primeros trabajos relacionados con la visión artificial datan de los años cincuenta. Se pensaba al principio que el desarrollo de un sentido artificial de la vista sería una tarea fácil y alcanzable en pocos años. Esta idea se reafirmó con los primeros trabajos en los que se utilizaron cámaras para la percepción del entorno. Así puede mencionarse el importante trabajo de Roberts (González & Woods 1992) que demostraba la posibilidad de procesar una imagen digitalizada para obtener una descripción matemática de los objetos incluidos en la escena, expresando su posición y orientación mediante transformaciones homogéneas; o el trabajo de Wichman (1967) que presentó en Stanford, un equipo con cámara de televisión conectada a una computadora, que podía identificar objetos y sus posiciones en tiempo real. Sin embargo, en ambos casos, se trataba de imágenes muy simples, con fuertes restricciones. Como ejemplo del entusiasmo inicial puede citarse a Marvin Minsky, uno de los pioneros de la Inteligencia Artificial (Nilsson 2001), que propuso a un alumno en el verano de 1966 como proyecto el que una computadora describiera lo que viese. Tal proyecto nunca se terminó.

Este entusiasmo fue debido por un lado a una gran confianza en el poder de las computadoras y por otro a la consideración de que, si el "ver" constituye para los hombres una tarea fácil, igual debería suceder con las computadoras. Sin embargo, pocos años

después la nota predominante en los laboratorios era de frustración ante lo limitado de los avances obtenidos y las pocas aplicaciones existentes. Aunque se hubieran desarrollado algoritmos que son utilizados hoy día, como los detectores de bordes de Roberts en 1965, Sobel en 1970 o Prewitt en 1970, su funcionamiento estaba limitado a un reducido número de imágenes y casos. Es por ello que los años setenta presentó un abandono progresivo en la investigación (De la Escalera 2009).

Otro motivo del desánimo tras las primeras investigaciones fue la constatación de que la facilidad del proceso visual para los hombres sólo quiere decir que no se es consciente de todo el proceso que se realiza, desde la captación de la imagen hasta la obtención de la información útil. A diferencia de la resolución de ecuaciones diferenciales, donde sí se tiene conciencia de los pasos que hay que realizar, el análisis de imágenes se realiza de forma subconsciente por lo que se nos antoja una tarea fácil, cuando en realidad desconocemos sus etapas. Y de ahí estriba la complejidad del problema: hacer de forma explícita el proceso de la percepción.

"A partir de la década de los años ochenta se empieza a hacer hincapié en la extracción de características. Así se tiene la detección de texturas (Haralik 1979), y la obtención de formas a través de ellas (Witkin 1981); y en ese mismo año se publicaron artículos sobre: visión estéreo (Mayhew & Frisby), detección de movimiento (Horn), interpretación de formas (Steven) y líneas (Kanade); o los detectores de esquinas de Kitchen y Rosendfeld (1982). Quizá el trabajo más importante de esa época es el libro de David Marr (1982), donde se aborda por primera vez una metodología completa de análisis de imágenes a través de una computadora" (De la Escalera 2009).

Por ello, a partir de esa época la visión artificial despunta de nuevo como una de las principales líneas de investigación en muchas universidades. Manifestación de ello es la aparición continua de revistas especializadas, o el número cada vez mayor de congresos internacionales, los fondos dedicados a su investigación y desarrollo, la aparición de asignaturas en los planes de estudio universitarios, etc. La causa de dicho crecimiento se debe en gran parte al enfoque más realista del problema a resolver: por ejemplo empieza a denominarse con el nombre menos rimbombante **visión por computadora** en lugar de visión artificial; y además el desarrollo de las computadoras: el aumento de su capacidad de cálculo y la disminución de su precio, y la aparición de "hardware" específico para el

procesamiento y tratamiento de imágenes. Con ello empiezan a ser utilizables aplicaciones ya resueltas con anterioridad, pero con un tiempo de cálculo inviable o un precio prohibitivo.

Representación de imágenes digitales

El término de imagen monocroma o simplemente imagen se refiere a una función bidimensional de intensidad de luz *f(x,y)*, donde *x* e *y* representan las coordenadas espaciales y el valor de *f* en un punto cualquiera *(x,y)* es proporcional al brillo (o nivel de gris) de la imagen en ese punto. La figura 3.3 ilustra el convenio de ejes que se utiliza.

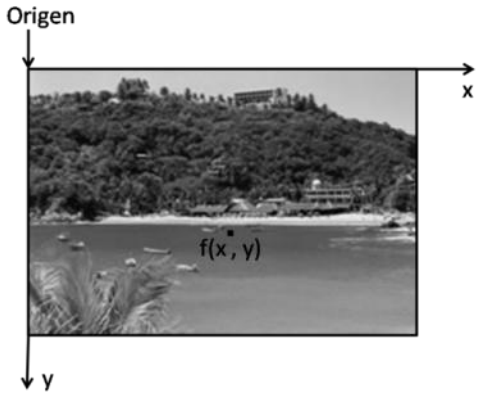


Figura 3.3. Representación de una coordenada *(x, y)* en una imagen (Pajares & De la Cruz 2008).

Una imagen digital puede considerarse como una matriz cuyos índices de filas y columnas identifican un punto de la imagen y el valor del correspondiente elemento de la matriz indica el nivel de gris en ese punto. Los elementos de una distribución digital de este tipo se denominan elementos de la imagen o más comúnmente pixel o pels, abreviaturas por su denominación inglesa “picture elements” (González & Woods 1992).

3.1. Escala de grises

Una imagen en escala de grises es el resultado de promediar las tres matrices de colores (rojo, verde y azul) de la imagen, en consecuencia cada pixel en la imagen resultante de esta operación está en el rango de 0 a 255, es decir, estará sobre la diagonal del origen (color negro) hasta el color máximo (color blanco) que se encuentra en el vértice opuesto del origen del tetraedro mostrado en la figura 2.17. Las imágenes originales ya sean en JPG

o BMP son convertidas a escala de grises para que sean utilizadas en el tratamiento de análisis de imágenes, esta conversión compacta o elimina los colores de la imagen pero este proceso no modifica su iluminación, es decir, no se altera la cantidad de luz que incide sobre los objetos en la escena o imagen, esto además disminuye a un solo canal de color, sin descartar el canal alfa de cada imagen, que da información de la transparencia de la imagen. Por otro lado, esta imagen es más fácil de procesar para la computadora y para aplicar los algoritmos de PDI.

En la figura 3.4 se muestran los canales que tiene una imagen en color RGB y una en escala de grises.

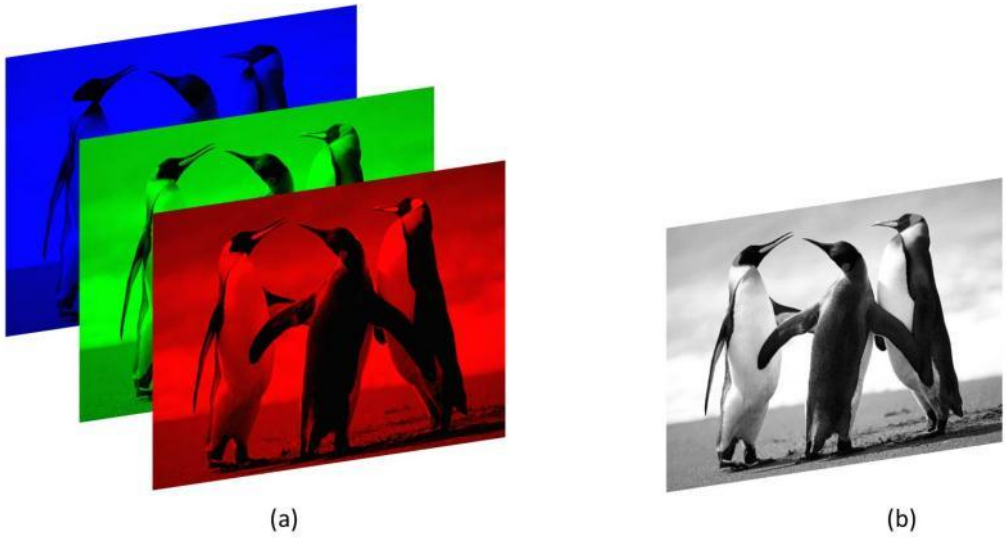


Figura 3.4. Representación de los canales de colores RGB y de escala de grises. (a) canales de color de una imagen de color, (b) canales de color de una imagen en escala de grises.

3.2. Ecualización del histograma

El histograma de una imagen digital con nivel de gris en el rango $[0, L-1]$ es una función discreta (variables con una cantidad de valores determinados) $p(r_k) = \frac{n_k}{n}$, donde r_k es el $k-ésimo$ nivel de gris, n_k es el número de pixeles de la imagen con ese nivel de gris, n es el número total de pixeles de la imagen y $k = 0,1,2,,\ldots\ldots\ldots, L-1$.

En general se puede decir que $p(r_k)$ da una idea del valor de la probabilidad de que aparezca el nivel de gris r_k. La representación gráfica de esta función para todos los valores de k proporcionan una descripción global de la apariencia de una imagen. Por ejemplo, la

figura 3.5 muestra los histogramas de cuatro tipos básicos de imágenes. El de la figura 3.5a muestra que los niveles de grises están concentrados hacia el extremo oscuro del rango de la escala de grises. Así, este histograma corresponde a una imagen con una apariencia global oscura. Sucede justo lo contrario en la figura 3.5b. El histograma mostrado en la figura 3.5c tiene un perfil estrecho, lo que significa que el rango dinámico es pequeño y que por tanto la imagen tiene bajo contraste. Como todos los niveles de gris caen en la zona central de la escala de grises, la imagen aparecería como un gris turbio.

Finalmente, la figura 3.5d muestra un histograma con una dispersión considerable que corresponde a una imagen de alto contraste (González & Woods 1992).

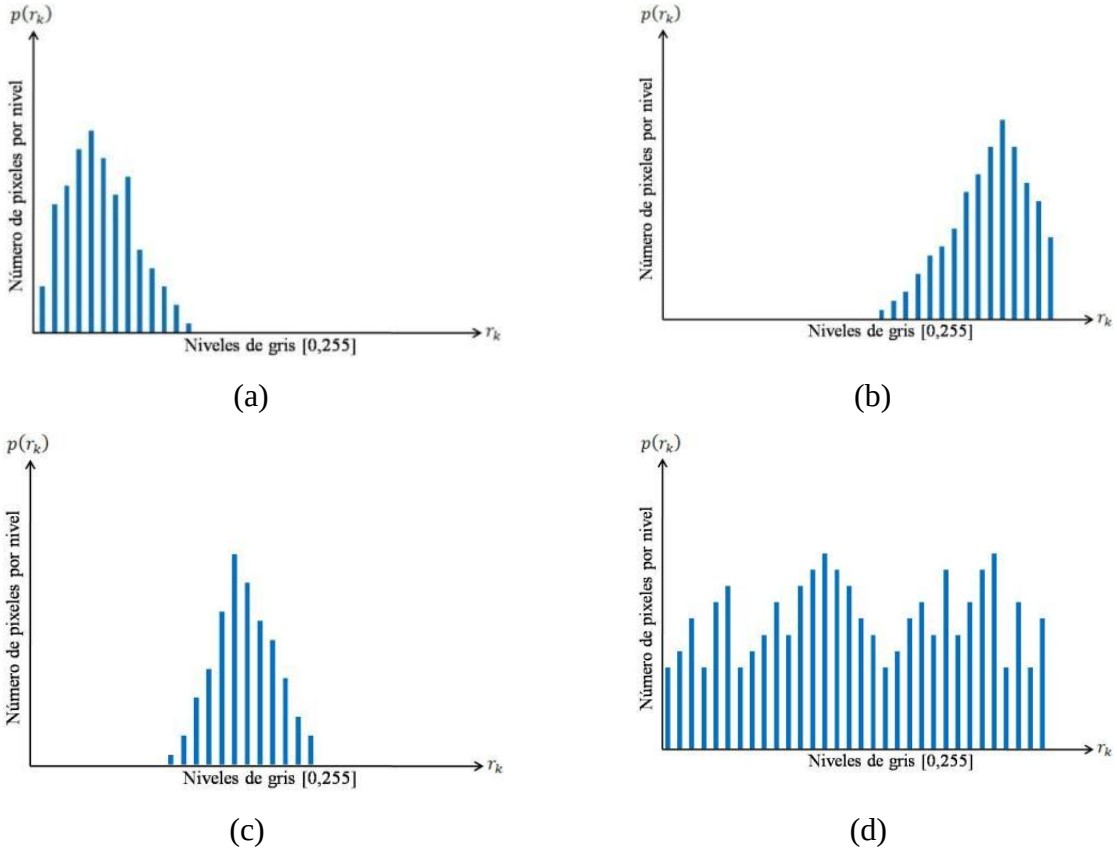


Figura 3.5. Histograma de una imagen en escala de grises. (a) Imagen oscura, (b) Brillante, (c) Bajo contraste, (d) Alto contraste (Pajares & De la Cruz 2008).

La ecualización del histograma de una imagen en escala de grises trata de normalizar o nivelar los altos (colores cercanos al blanco) y bajos (colores aproximados al negro) en el histograma con el objetivo de mejorar el contraste y el brillo en la imagen,

eliminando pequeños tonos que causan ruido en la imagen, esta técnica es una opción que trata de nivelar uniformemente en el histograma los colores en escala de grises, sin embargo, otra opción es que a partir de otra imagen que tenga su histograma linealmente uniforme dotarle de esa característica a la imagen, en otras palabra forzar la salida del histograma a partir de otra.

3.3. Umbralización

Esta técnica convierte una imagen con varios niveles de gris a una nueva con sólo dos, de manera que los objetos quedan separados del fondo. La Umbralización se basa en que los pixeles de un determinado objeto tienen el mismo nivel de gris; como esto no será del todo cierto, se buscan intervalos de gris, cada uno de ellos pertenecientes a un objeto. Esta técnica será útil si los objetos tienen una superficie parecida y el fondo es uniforme. Un caso típico es un texto, donde las letras destacan del entorno que es blanco o una imagen tomada con iluminación posterior.

Otros ejemplos pueden ser imágenes infrarrojas donde los objetos destacan del entorno por su calor. Por tanto se aplicarán las fórmulas mostradas en las ecuaciones 5, 6 y 7.

$$g(x,y) = \begin{cases} 1 & si\ T \leq f(x,y) \\ 0 & en\ otro\ caso \end{cases}, \tag{5}$$

cuando el valor buscado sea mayor que uno dado,

$$g(x,y) = \begin{cases} 1 & si\ \ T \geq f(x,y) \\ 0 & en\ otro\ caso \end{cases}, \tag{6}$$

si es menor que el umbral,

$$g(x,y) = \begin{cases} 1 & si\ \ T_a \leq f(x,y) \leq T_b \\ 0 & en\ otro\ caso \end{cases} \tag{7}$$

o si se conocen que los objetos tienen un cierto intervalo.

Donde *g(x,y)* es un pixel en la imagen, *x es* igual al número de renglones y *y* es igual al número de columnas, por lo tanto en la posición *g(x,y)* existe un pixel que contiene el valor de interés para la función. El valor *T* es una constante (valor de umbral) que definirá el límite de los pixeles que serán aceptados en la función.

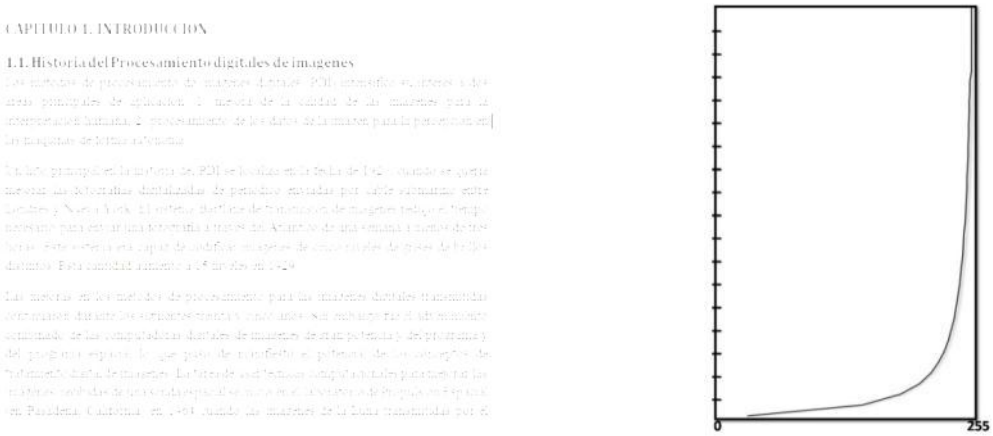
El problema está en encontrar los valores de gris a tomar como umbrales entre los objetos, ya que, debido al ruido el objeto y el fondo no tienen un único valor de gris sino un intervalo, se solapan en algunos valores. Así en la figura 3.6a aunque se distingue bien el texto de la imagen, si se observa su histograma (Fig. 3.6b) surge la duda de cuál es el valor adecuado. Así se observa si se toma un valor alto (fig. 3.6c), donde puntos del fondo se han tomado por letras; lo contrario ocurre si se toma un valor demasiado bajo (Fig. 3.6d), por último se tiene el mejor umbral (Fig. 3.6e).

Aunque hay varias técnicas para obtener los valores para realizar la umbralización, es un problema que depende del caso concreto que se quiera analizar, la experiencia y el conocimiento sobre el problema, determinan en gran manera los valores que se aplicarán (Pajares & De la Cruz 2008).

Esta técnica es utilizada en imágenes donde el fondo es oscuro y los objetos claros, esto hace que se tengan aproximadamente dos valores de nivel de gris para los pixeles en general, un grupo que identifica al fondo y el otro grupo que describe a los objetos en la imagen. En consecuencia, se tiene un histograma que diferencía a esos objetos, el cual ayuda a determinar el umbral que permitirá diferenciar a las dos clases de niveles de gris. Sin embargo, existen técnicas que con la ayuda del histograma de la imagen pueden definir el umbral óptimo, pero la elección dependerá siempre del problema a analizar.

3.4. Filtro pasa bajo

Los bordes y otras transiciones bruscas (como el ruido) en los niveles de gris de una imagen contribuyen significativamente al contenido en altas frecuencias de su transformada de Fourier. Por lo tanto, el difuminado (o suavizado) se consigue, en el dominio de la frecuencia, a base de atenuar un rango específico de componentes de alta frecuencia en la transformada de una imagen dada.



(a) (b)

CAPÍTULO 1. INTRODUCCIÓN

1.1. Historia del Procesamiento digitales de imágenes

Los métodos de procesamiento de imágenes digitales (PDI) intensifico su interés a dos áreas principales de aplicación: 1) mejora de la calidad de las imágenes para la interpretación humana; 2) procesamiento de los datos de la imagen para la percepción en las maquinas de forma autónoma.

Un hito principal en la historia del PDI se localiza en la fecha de 1920, cuando se quería mejorar las fotografías digitalizadas de periódico enviadas por cable submarino entre Londres y Nueva York. El sistema Bartlane de transmisión de imágenes redujo el tiempo necesario para enviar una fotografía a través del Atlántico de una semana a menos de tres horas. Este sistema era capaz de codificar imágenes de cinco niveles de grises de brillos distintos. Esta cantidad aumento a 15 niveles en 1929.

Las mejoras en los métodos de procesamiento para las imágenes digitales transmitidas continuaron durante los siguientes treinta y cinco años. Sin embargo fue el advenimiento combinado de las computadoras digitales de imágenes de gran potencia y del programa y del programa espacial lo que puso de manifiesto el potencial de los conceptos de tratamiento digital de imágenes. La tarea de usar técnicas computacionales para mejorar las imágenes recibidas de una sonda espacial se inicio en el laboratorio de Propulsión Espacial (en Pasadena, California) en 1964 cuando las imágenes de la Luna transmitidas por el

(c) (d)

(e)

Figura 3.6. Imágenes con umbrales diferentes. (a) Imagen original, (b) Histograma, (c) Umbral alto, (d) Umbral bajo, (e) Umbral intermedio (De la Escalera 2009).

A partir de la ecuación 8, donde *F(u,v)* es la transformada de Fourier de una imagen para ser suavizada, el problema consiste en seleccionar una función de transferencia *H(u, v)*

que proporcione *G(u,v)* atenuando las componentes de alta frecuencia de *F(u,v).* La transformada inversa proporcionará la imagen suavizada deseada *g(x,y)*. Hay funciones de transferencia del filtro que actúan sobre las partes real e imaginaria de *F(u,v)* exactamente de la misma forma. Estos filtros se llaman filtros de cambio de fase nulo, pues no alteran la fase de la transformada.

$$G(u,v) = H(u,v)F(u,v) \tag{8}$$

Filtro ideal pasa bajo

Un filtro pasa bajo ideal 2D está caracterizado por una función de transferencia que satisface la relación mostrada en la ecuación 9.

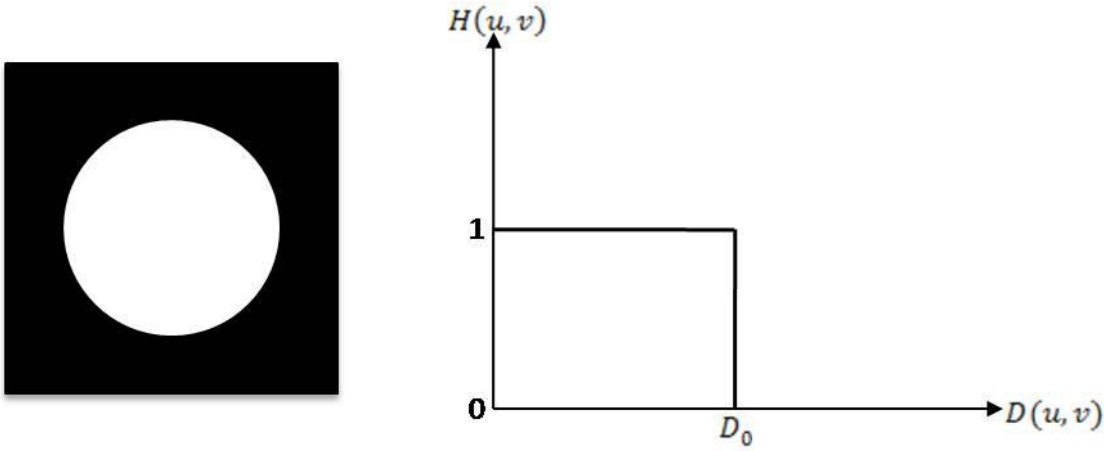
$$H(u,v) = \begin{cases} 1 \; si \;\; D(u,v) \leq D_0 \\ 0 \; si \;\; D(u,v) > D_o \end{cases}, \tag{9}$$

donde D_0 es una cantidad especificada no negativa, y *D(u, v)* es la distancia desde el punto *(u,v)* al origen del plano de frecuencias en la cual se define la ecuación 10.

$$D(u,v) = (u^2 + v^2)^{1/2} \tag{10}$$

El nombre **filtro ideal** indica que todas las frecuencias dentro del circulo de radio D_0 se pasan sin atenuación, mientras todas las frecuencias fuera de este círculo son eliminadas. La figura 3.7a muestra el filtro representado en niveles de gris. Para una mejor visualización, los valores de grises en el rango [0, 255] se han normalizado al rango [0, 1].

Los filtros considerados en esta sección son radialmente simétricos sobre el origen. Para este tipo de filtros, es suficiente especificar un corte transversal desde el origen a lo largo de una línea radial, como se muestra en la figura 3.7b. La función de transferencia completa del filtro puede generarse rotando dicha sección 360° sobre el origen. Una cuestión importante a considerar es que la especificación de filtros simétricos radiales centrados en un cuadro de frecuencia $N \; x \; N$ se basa en el hecho de que la transformada de Fourier ha sido centrada en el origen (Pajares & De la Cruz 2008).



(a) (b)

Figura 3.7. Dibujo en perspectiva de la función de transferencia de un filtro ideal pasa bajo. (a) Representación de nivel de gris del filtro, (b) Sección transversal (Pajares & De la Cruz 2008).

3.5. Filtro pasa alto

Los bordes y demás cambios bruscos de los niveles de grises están relacionados con las componentes de altas frecuencias. Puede lograrse el realce de la imagen en el dominio de las frecuencias mediante un procedimiento de filtrado de pasa alto, que atenúe las componentes de bajas frecuencias sin modificar la información de la trasformada de Fourier contenida en las componentes de alta frecuencia.

Filtro ideal pasa alto

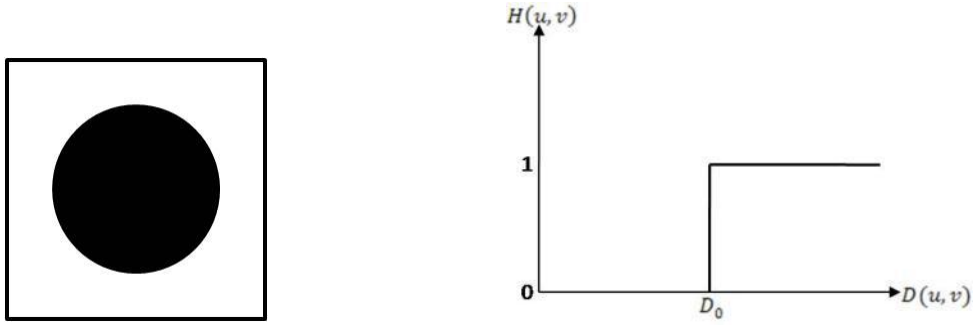
Un filtro pasa alto ideal 2D está caracterizado por la función de transferencia que satisface la relación de la ecuación 11.

$$H(u,v) = \begin{cases} 1 & si \;\; D(u,v) > D_0 \\ 0 & si \;\; D(u,v) \leq D_0 \end{cases}, \tag{11}$$

donde D_0 es la distancia de corte medida desde el origen del plano de frecuencias, y *D(u, v)* está dada por la ecuación 12.

$$D(u,v) = (u^2 + v^2)^{1/2} \tag{12}$$

Este filtro es opuesto al filtro pasa bajo ideal porque atenúa completamente todas las frecuencias dentro del círculo de radio D_0 mientras pasan sin atenuación todas las frecuencias fuera del círculo. La figura 3.8 muestra el filtro representado en niveles de gris y su sección transversal respectivamente (Pajares & De la Cruz 2008).



(a) (b)

Figura 3.8. Dibujo en perspectiva de la función de transferencia de un filtro ideal pasa alto. (a) Niveles de gris del filtro, (b) Sección transversal (Pajares & De la Cruz 2008).

Gradiente de una imagen

El gradiente de la imagen *f(x, y)* en un punto *(x, y)* se define como un vector bidimensional dado por la ecuación 13, siendo un vector perpendicular al borde,

$$\boldsymbol{G}[\boldsymbol{f}(\boldsymbol{x},\boldsymbol{y})] = \begin{bmatrix}\boldsymbol{G_x}\\ \boldsymbol{G_y}\end{bmatrix} = \begin{bmatrix}\frac{\partial}{\partial \boldsymbol{x}}\boldsymbol{f}(\boldsymbol{x},\boldsymbol{y})\\ \frac{\partial}{\partial \boldsymbol{y}}\boldsymbol{f}(\boldsymbol{x},\boldsymbol{y})\end{bmatrix}, \tag{13}$$

donde el vector *G* apunta en la dirección de variación máxima de f en el punto *(x,y)* por unidad de distancia, con la magnitud y dirección dadas por la ecuación 14.

$$|\boldsymbol{G}| = \sqrt{\boldsymbol{G_x^2} + \boldsymbol{G_y^2}}; \qquad \emptyset(\boldsymbol{x},\boldsymbol{y}) = \boldsymbol{tan^{-1}}\left(\frac{\boldsymbol{G_y}}{\boldsymbol{G_x}}\right) \tag{14}$$

Es una práctica habitual aproximar la magnitud del gradiente con valores absolutos, como se muestra en la ecuación 15,

$$|G| \approx |G_y| + |G_x| \tag{15}$$

donde:

$|G_x|$= Magnitud del gradiente de los valores verticales de una imagen.

$|G_y|$= Magnitud del gradiente de los valores horizontales de una imagen.

$|G|$ = Suma de las magnitudes $|G_x|$ y $|G_y|$ de una imagen.

Lo anterior se hace por que el valor de la magnitud del gradiente no es tan importante como la relación entre diferentes valores, sin embargo, la ecuación 15 resulta mucho más fácil de implementar, particularmente cuando se realiza en hardware.

Para calcular la derivada en la ecuación 13 se pueden utilizar las diferencias de primer orden entre dos pixeles adyacentes, como se muestra en la ecuación 16.

$$G_x = \frac{f(x+\Delta x) - f(x-\Delta x)}{2\Delta x}, \qquad G_y = \frac{f(y+\Delta y) - f(y-\Delta y)}{2\Delta y} \tag{16}$$

Ésta es la forma más elemental de obtener el gradiente en un punto. La magnitud del gradiente puede tomar cualquier valor real y el ángulo también cualquier valor real entre 0° y 360°.

La figura 3.9 muestra la imagen gradiente. En la figura 3.9a se muestra la imagen original; en la figura 3.9b se muestra la magnitud de la imagen gradiente, es decir $|G|$; en la figura 3.9c se muestra una imagen binaria utilizando la relación mostrada en la ecuación 17,

$$g(x,y) = \begin{cases} 1 & si\ G[f(x,y)] > T \\ 0 & si\ G[f(x,y)] \leq T \end{cases} \tag{17}$$

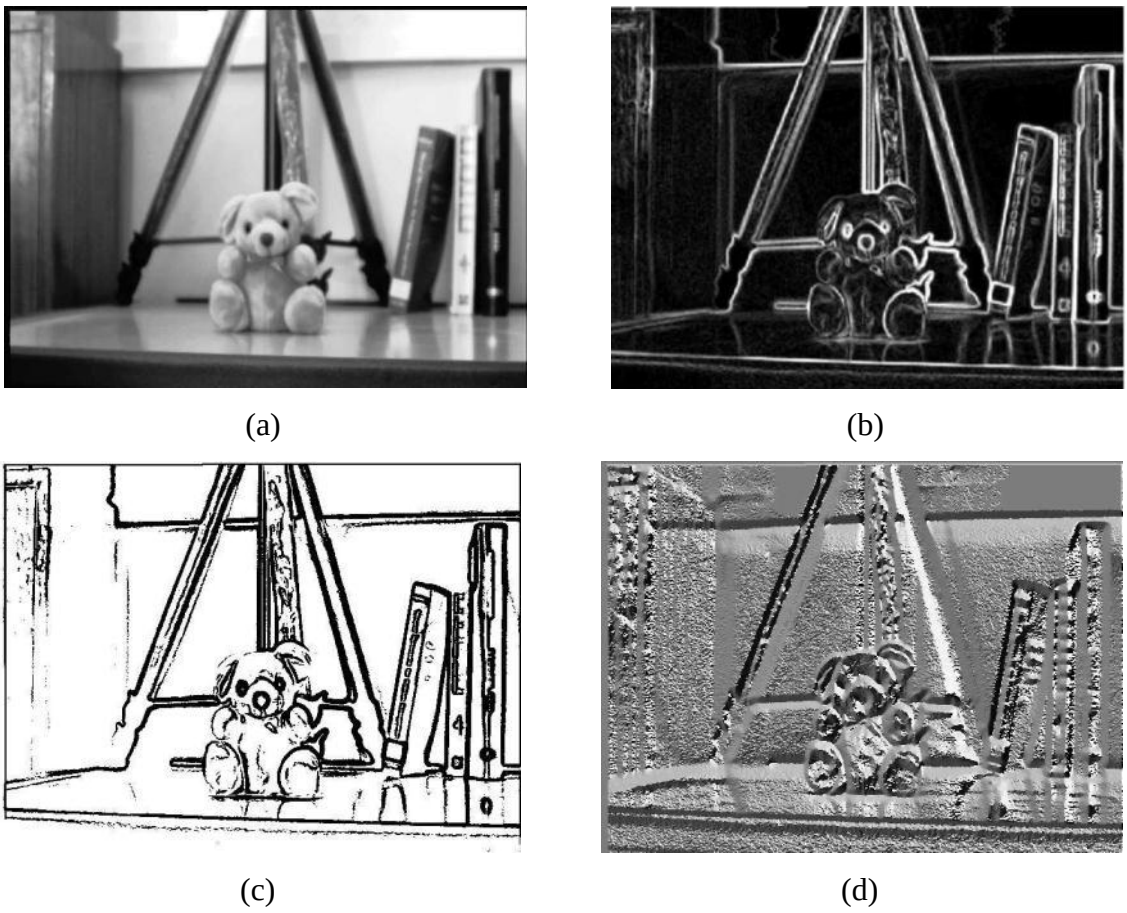
donde *T* es un valor de umbral no negativo (en este caso *T=30*). Sólo los pixeles de borde cuyo gradiente excedan el valor *T* se consideran importantes. En la figura 3.9d se muestra el ángulo del gradiente calculado con la ecuación 14.

3.6. Filtro Sobel

Los operadores gradientes en general tienen el efecto de magnificar el ruido subyacente en la imagen. Tanto los operadores de Sobel como el resto de los operadores de vecindad tienen la prioridad añadida de suavizar la imagen, eliminando parte de ruido y por

consiguiente, minimiza la aparición de los falsos bordes debido al efecto de magnificación del ruido por parte de los operadores derivada.

A partir de la ecuación 16 las derivadas basadas en los operadores de Sobel son las mostradas en la ecuación 18.



(a) (b)

(c) (d)

Figura 3.9. Gradiente de una imagen. (a) Imagen original, (b) Imagen de la magnitud del gradiente, (c) Imagen binarizada con *T* = 30, (d) Imagen del ángulo del gradiente (Pajares & De la Cruz 2008).

$$G_x = (z_3 + 2z_6 + z_9) - (z_1 + 2z_4 + z_7)$$
$$G_y = (z_7 + 2z_8 + z_9) - (z_1 + 2z_2 + z_3) \quad (18)$$

Donde los distintos valores de *z* en la región de la ecuación 19 son los niveles de gris de los pixeles solapados por las máscaras en cualquier localización de la imagen. Para obtener los valores de las componentes del vector gradiente en el punto definido por el pixel central de la región se utilizan las expresiones de las ecuaciones 20 y 21, con lo que la magnitud y el ángulo se pueden obtener a partir de la ecuación 14, es decir, se obtiene un

valor del gradiente en dicho punto. Para obtener el siguiente valor, las máscaras se mueven a la siguiente posición del nuevo pixel y se repite el proceso, después de haber barrido todas las posibles posiciones, el resultado es una imagen gradiente. Es preciso tener en cuenta que en los bordes de la imagen los valores del gradiente no se pueden calcular por sobrepasar las máscaras de la propia imagen. Una vez que se ha obtenido la magnitud del gradiente, se puede decidir si un determinado punto es de borde o no, aplicando la ecuación 17 obteniendo así una imagen binaria como resultado.

$$G[f(x,y)] = \begin{bmatrix} z_1 & z_2 & z_3 \\ z_4 & z_5 & z_6 \\ z_7 & z_8 & z_9 \end{bmatrix} \tag{19}$$

$$G_y = \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix} \tag{20}$$

$$G_x = \begin{bmatrix} -1 & 2 & -1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{bmatrix} \tag{21}$$

Se supone la siguiente imagen elemental, dada en la ecuación 22. Se pretende calcular el gradiente en el pixel marcado con el punto negro. La región de la imagen que interviene está marcada con los datos resaltados en color más obscuro.

$$\begin{bmatrix} \mathbf{0} & \mathbf{8} & \mathbf{8} & 8 & 8 & 8 \\ \mathbf{1} & \mathbf{\bullet 8} & \mathbf{9} & 7 & 6 & 8 \\ \mathbf{2} & \mathbf{3} & \mathbf{4} & 8 & 8 & 8 \\ 2 & 2 & 2 & 8 & 8 & 8 \\ 2 & 2 & 2 & 2 & 8 & 8 \end{bmatrix} \tag{22}$$

Con la ecuación 18 se calcula $G_x = 30 - 4 = 26$ y $G_y = 12 - 24 = -12$. Utilizando la ecuación 15, se tiene $|G| = 48$, si se fija un valor de umbral *T*=30, el pixel marcado con el punto negro sería un punto en el borde y si por el contrario, el umbral es mayor que 48, dicho punto no sería considerado borde. A continuación se desplazan las máscaras a la siguiente posición, al pixel con valor 9, después se mueve al pixel con valor 7, hasta llegar al valor 6, con lo que se completaría esta fila.

Al aplicar G_x en la imagen original (Fig. 3.10a), se producen los resultados en la dirección perpendicular al eje x, por lo cual se obtiene la figura 3.10b, caso contrario pasa al aplicar G_y (Fig. 3.10c) que muestra los rasgos horizontales de la imagen, en la figura

3.10d se obtiene $|G|$ que es el valor absoluto de las sumas de los resultados anteriores. En las siguientes figuras (Fig. 3.10e y 3.10f) se muestran las imágenes binarizadas aplicando la ecuación 15, para decidir qué punto es borde o no dependiendo de cierto umbral, en este caso fue de un umbral de 10 y 20 respectivamente (Pajares & De la Cruz 2008).

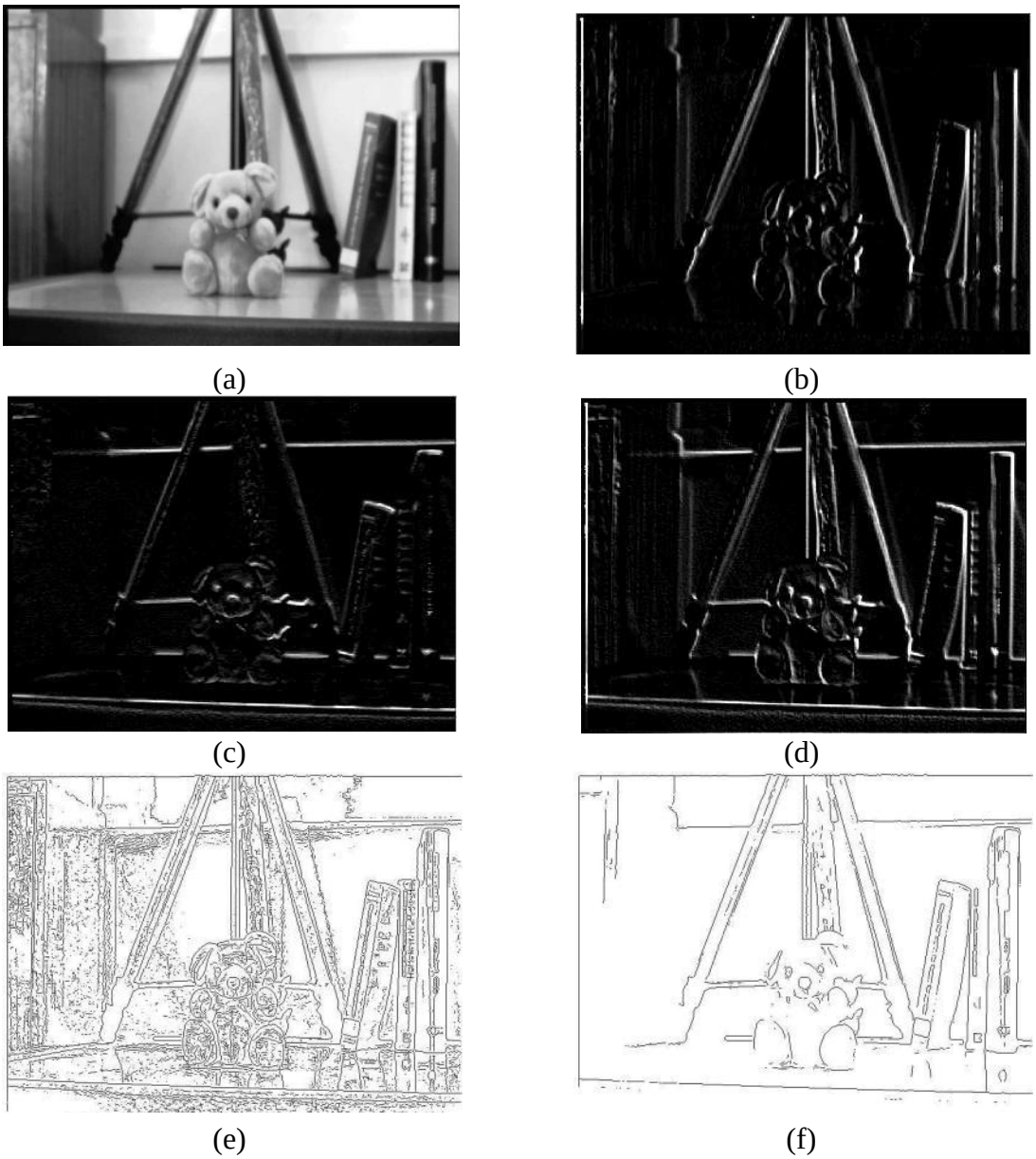


Figura 3.10. Aplicación de las máscaras de Sobel. (a) Imagen original, (b) G_x, (c) G_y, (d) Aplicando la ecuación 16, (e) T=10, (f) T=20 (Pajares & De la Cruz 2008).

3.7. Operador de Roberts

El operador de Roberts marca solamente los puntos de borde, sin información sobre la orientación de éstos. Es un operador muy simple que funciona muy bien en imágenes binarias. Opera sobre las dos diagonales perpendiculares mostradas en la ecuación 23 y definidas por la ecuación 24 y 25.

f(x-1,y-1)	*f(x,y-1)*	*f(x+1,y-1)*
f(x-1,y)	*f(x,y)*	*f(x+1,y)*
f(x-1,y+1)	*f(x,y-1)*	*f(x+1,y+1)*

(23)

Existen dos formas del operador de Roberts:

1. Raíz cuadrada de la suma de las diferencias de los vecinos diagonales al cuadrado, ecuación 24.

$$\sqrt{[f(x,y)-f(x-1,y-1)]^2+[f(x,y-1)-f(x-1,y)]^2} \quad (24)$$

2. Suma de la magnitud de las diferencias de los vecinos diagonales, ecuación 25.

$$|f(x,y)-f(x-1,y-1)|+|f(x,y-1)-f(x-1,y)| \quad (25)$$

La más usada es la ecuación 25 por su menor coste de tiempo computacional.

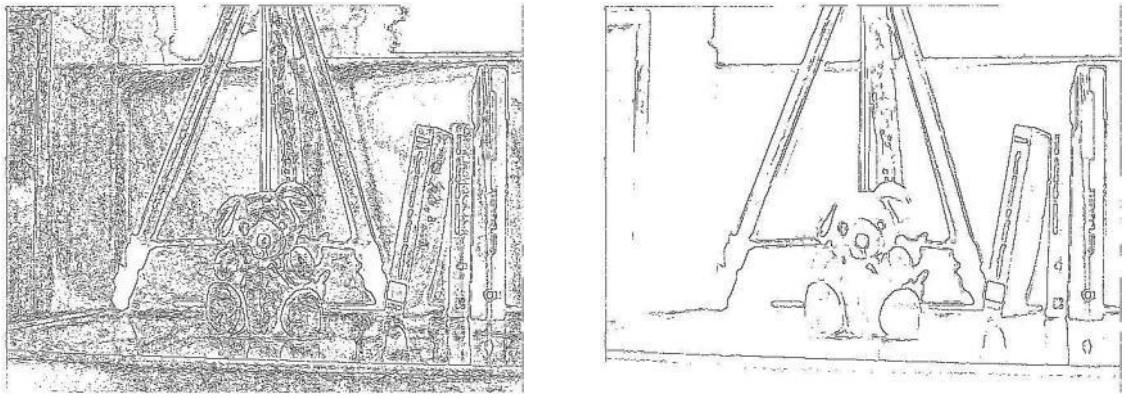
En las figuras 3.11a y 3.11b se muestran la imágenes binarizadas obtenidas al aplicar la ecuación 7 a la imagen original de la figura 3.9a utilizando los operadores de Roberts y considerando que un punto es de borde, si la magnitud del gradiente es mayor que T = 10 y 20, respectivamente (Pajares & De la Cruz 2008).

3.8. Filtro Prewitt

El operador de Prewitt es similar al operador de Sobel, diferenciándose en los coeficientes de las máscaras mostradas en las ecuaciones 26 y 27.

$$G_y = \begin{bmatrix} -1 & 0 & 1 \\ -1 & 0 & 1 \\ -1 & 0 & 1 \end{bmatrix} \quad (26)$$

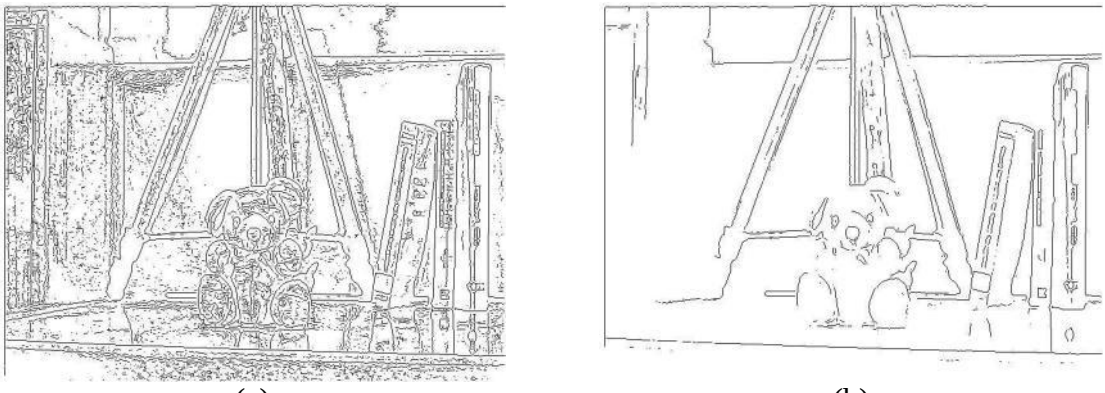
$$G_x = \begin{bmatrix} 1 & 1 & 1 \\ 0 & 0 & 0 \\ -1 & -1 & -1 \end{bmatrix} \tag{27}$$



(a) (b)

Figura 3.11. Aplicación del filtro de Roberts. (a) Imagen binarizada con *T*=10, (b) Imagen binarizada con *T*=20 (Pajares & De la Cruz 2008).

La magnitud del gradiente se obtiene de la ecuación 15. Para ejemplificar este operador, considérense las ecuaciones 26 y 27 como resultado de aplicar la ecuación 15 a la imagen original (Fig. 3.10a) utilizando los operadores de Prewitt y considerando que un punto es de borde, los resultados se muestran en las figuras 3.12a y 3.12b, si la magnitud del gradiente es mayor que *T*=10 y 20, respectivamente (Pajares & De la Cruz 2008).



(a) (b)

Figura 3.12. Filtro de Prewitt aplicándolo a la figura 3.10a. (a) Aplicando Filtro Prewitt con $T \geq 10$, (b) $T \geq 20$ (Pajares & De la Cruz 2008).

3.9. Operador de Frei-Chen (u operador isotrópico)

Mientras que el filtro de Prewitt detecta mejor los bordes verticales, los operadores de Sobel lo hacen en las diagonales. El operador isotrópico intenta un equilibrio entre ellos y las máscaras que comúnmente se usan se muestran en las ecuaciones 28 y 29. (De la Escalera 2009).

$$G_y = \begin{bmatrix} -1 & 0 & 1 \\ -\sqrt{2} & 0 & \sqrt{2} \\ -1 & 0 & 1 \end{bmatrix} \tag{28}$$

$$G_x = \begin{bmatrix} -1 & -\sqrt{2} & -1 \\ 0 & 0 & 0 \\ 1 & \sqrt{2} & 1 \end{bmatrix} \tag{29}$$

Existen 9 máscaras (ecuaciones 30-38) de Frei-Chen, que pueden agruparse en un conjunto de cuatro máscaras definiendo un subespacio de bordes, (ecuaciones 30-33), cuatro máscaras para un espacio de línea (ecuaciones 34-37) y una máscara para un subespacio medio (ecuación 38). Es decir, los subespacios de bordes y líneas sirven para detectar la presencia de puntos de bordes en general y líneas respectivamente, el subespacio medio sirve para suavizar la imagen original, ya que es realmente el operador de la media.

En la figura 3.13a se muestra la magnitud de la proyección en el subespacio de bordes correspondientes caracterizados por las máscaras definidas por las ecuaciones 30-33 y en la figura 3.13b se muestra la magnitud de la proyección en el subespacio de líneas correspondiente caracterizado por las máscaras definidas por las ecuaciones 34-37. Como se puede ver, existe una diferencia considerable entre ambos resultados.

$$G_x = \frac{1}{2\sqrt{2}} \begin{bmatrix} 1 & \sqrt{2} & 1 \\ 0 & 0 & 0 \\ -1 & -\sqrt{2} & -1 \end{bmatrix} \tag{30}$$

$$G_y = \frac{1}{2\sqrt{2}} \begin{bmatrix} 1 & 0 & -1 \\ \sqrt{2} & 0 & -\sqrt{2} \\ 1 & 0 & -1 \end{bmatrix} \tag{31}$$

$$G_x^{'} = \frac{1}{2\sqrt{2}}\begin{bmatrix} 0 & -1 & \sqrt{2} \\ 1 & 0 & -1 \\ -\sqrt{2} & 1 & 0 \end{bmatrix} \tag{32}$$

$$G_x^{''} = \frac{1}{2\sqrt{2}}\begin{bmatrix} \sqrt{2} & -1 & 0 \\ -1 & 0 & 1 \\ 0 & 1 & -\sqrt{2} \end{bmatrix} \tag{33}$$

$$G_{xy} = \frac{1}{2}\begin{bmatrix} 0 & 1 & 0 \\ -1 & 0 & -1 \\ 0 & 1 & 0 \end{bmatrix} \tag{34}$$

$$G_y^{'} = \frac{1}{2}\begin{bmatrix} -1 & 0 & 1 \\ 1 & 0 & -1 \\ 1 & 0 & -1 \end{bmatrix} \tag{35}$$

$$G_y^{''} = \frac{1}{6}\begin{bmatrix} 1 & -2 & 1 \\ -2 & 4 & -2 \\ 1 & -2 & 1 \end{bmatrix} \tag{36}$$

$$G_{xy}^{'} = \frac{1}{6}\begin{bmatrix} -2 & 1 & -2 \\ 1 & 4 & 1 \\ -2 & 1 & -2 \end{bmatrix} \tag{37}$$

$$G_{yx} = \frac{1}{3}\begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix} \tag{38}$$

(a) (b)

Figura 3.13. Aplicación de los operadores isotrópicos. (a) Imagen resultante al aplicar las ecuaciones 30-33, (b) Resultado de aplicar las ecuaciones 34-37 (Pajares & De la Cruz 2008).

3.10. Operador Laplaciano

El Laplaciano de una función 2D *f(x, y)* es un operador de segunda derivada definido como se muestra en la ecuación 39.

$$\nabla^2 f = \frac{\partial^2 f}{\partial x^2} + \frac{\partial^2 f}{\partial y^2} \tag{39}$$

Como en el caso del gradiente, la ecuación 39 se puede implementar en forma digital de varias formas, por ejemplo, como se muestra en la ecuación 40,

$$\nabla^2 f = 4z_5 - (z_2 + z_4 + z_6 + z_8) \tag{40}$$

donde los coeficientes de *z* se han definido en la ecuación 18. El requisito básico para definir el Laplaciano digital, es que los coeficientes asociados con el pixel central y los coeficientes asociados con el resto de los pixeles sean negativos, puesto que el Laplaciano se obtiene con una derivada, la suma de los coeficientes deben ser cero. Por tanto, la respuesta es cero siempre que el punto en cuestión y sus vecinos tengan el mismo valor.

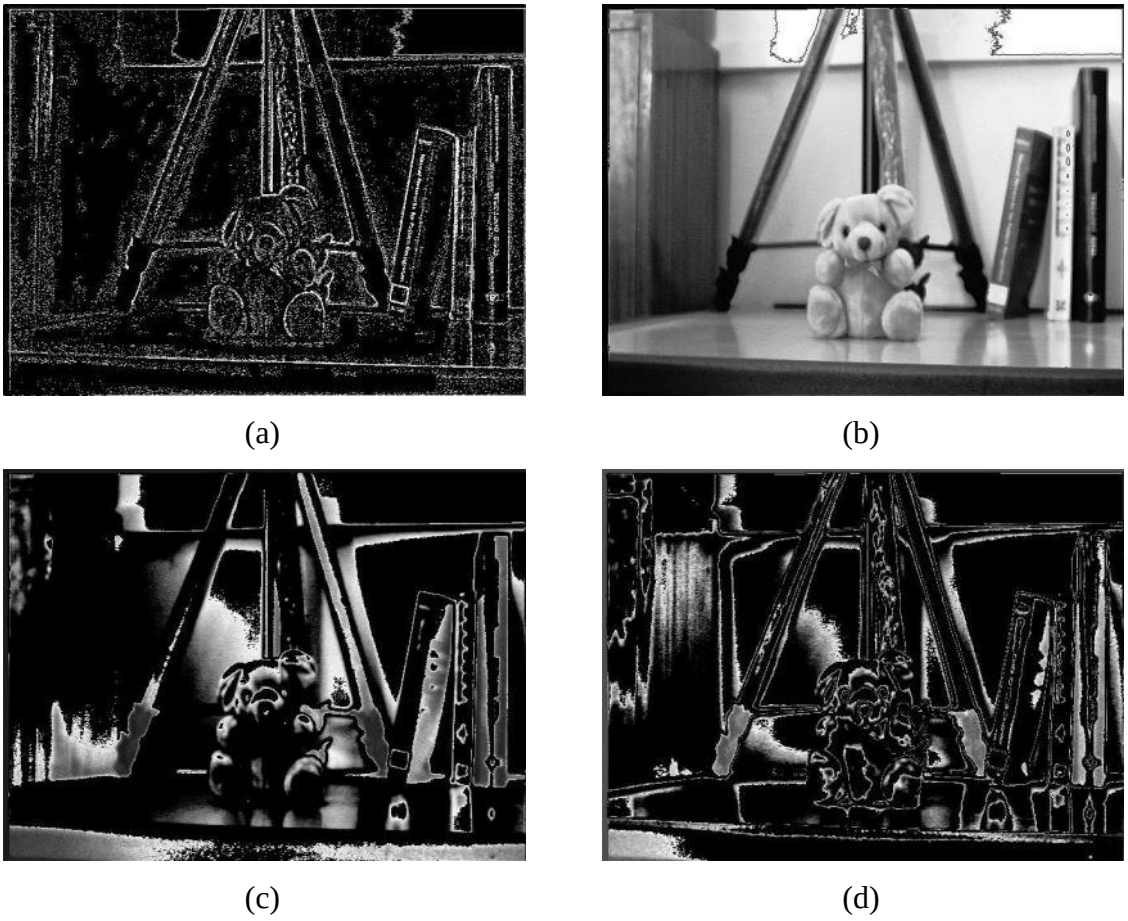
Las tres máscaras del operador Laplaciano se muestran en las ecuaciones 41-43, y representan diferentes aproximaciones del operador Laplaciano. Se aplica seleccionando una máscara y realizando una operación de convolución sobre la imagen. El signo del resultado (positivo o negativo) de los pixeles adyacentes proporciona información direccional e indica que lado del borde es más o menos obscuro.

$$G_{xy} = \begin{bmatrix} 0 & -1 & 0 \\ -1 & 4 & -1 \\ 0 & -1 & 0 \end{bmatrix} \tag{41}$$

$$G_{xy}^{'} = \begin{bmatrix} 1 & -2 & 1 \\ -2 & 4 & -2 \\ 1 & -2 & 1 \end{bmatrix} \tag{42}$$

$$G_{xy}^{''} = \begin{bmatrix} -1 & -1 & -1 \\ -1 & 8 & -1 \\ -1 & -1 & -1 \end{bmatrix} \tag{43}$$

En la figura 3.14a se muestra el resultado de aplicar el operador de la ecuación 41 a la imagen original de la figura 3.10a, en la figura 3.14b se aplica la misma máscara con un valor de 5, en las figuras 3.14c y 3.14d con un valor central en la máscara de 10 y 20 respectivamente.



(a) (b)

(c) (d)

Figura 3.14. Filtro Laplaciano. (a) Aplicación Laplaciana, (b) Laplaciana con valor central 5, (c) 10, (d) 20 (Pajares & De la Cruz 2008).

3.11. Redes neuronales

Estos modelos reciben una serie de ejemplos de entrada con el fin de identificar lo que se denomina elemento desconocido. En el caso de las redes neuronales dicho elemento se materializa en el aprendizaje, lo que se denomina pesos de conexión entre las neuronas. El tipo de inferencia realizado en este caso es de naturaleza inductiva (Pajares & Santos 2006).

Perceptron

Este modelo fue introducido por Rosenblatt a finales de los años cincuenta. La estructura del Perceptron inspira en las primeras etapas del procesamiento los sistemas sensoriales de las animales (por ejemplo el de visión), en los cuales la información va atravesando sucesivamente capas de neuronas, que realizan un procesamiento progresivo de más alto nivel (Martín & Sanz 2007).

La razón del gran interés de dichas máquinas llamadas perceptrones fue el desarrollo de las correspondientes demostraciones matemáticas llegando a la conclusión de que los perceptrones, cuando son entrenados con un conjunto de entrenamiento lineal separable, convergen a una solución en un número finito de iteraciones. La solución tomó la forma de coeficientes de hiperplanos capaces de separar correctamente las clases representadas por patrones del conjunto de entrenamiento (Pajares & De la Cruz 2008).

El Perceptron es un modelo unidireccional, compuesto por dos capas de neuronas, una sensorial o de entradas, y otra de salida. La operación de una red de este tipo, con *n* neuronas de entrada y *m* de salida, se expresa en la ecuación 44.

$$y_i(t) = f\left(\sum_{j=1}^{n} w_{ij}\, x_j - \emptyset_i\right), \forall i, 1 \leq i \leq m, \tag{44}$$

donde:

$y_i(t)$ = Resultado de la sumatoria dada por *f*.
x_j = Conjunto de entradas.
w_{ij} = Pesos sinápticos de la neurona *i*, que representa la interacción entre la neurona anterior *i* con la neurona siguiente *j*.
$\emptyset_i$ = Valor real llamado umbral, que se resta a la función principal (Ganancia).

En la figura 3.15 se muestra la estructura básica de una RNA tipo Perceptron, donde se pueden apreciar el vector de entrada *x* y el vector de pesos *w*, que entran como producto a una sumatoria, a esto se le resta la ganacia $\emptyset$ y con base en un elemento de activación se determina la salida de la RNA.

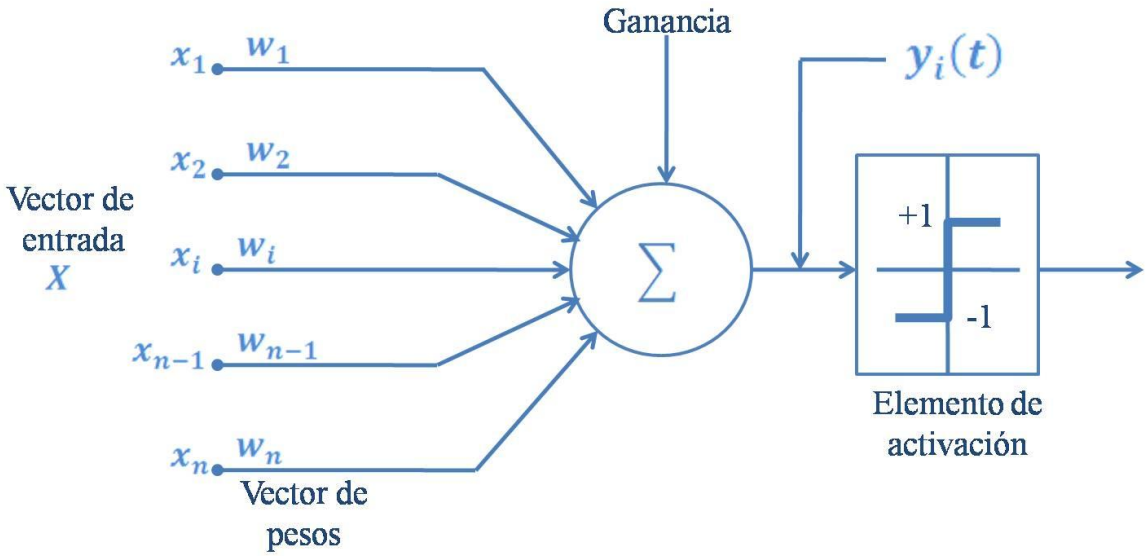


Figura 3.15. Modelo básico de la RNA tipo Perceptron.

CAPÍTULO 4. DESARROLLO DEL TEMA

En este capítulo se describe la fase de análisis, diseño, implementación y las pruebas de este proyecto de tesis con la finalidad de tener un prototipo de software.

4.1. Análisis

El objetivo principal de este proyecto de tesis es crear un software que permita identificar a una persona, a partir de una serie de características. El núcleo del software es una RNA Perceptron con 4 neuronas, la cual tiene como base de conocimiento las distancias de las características de las personas que posteriormente son utilizadas para entrenar a la red. Se consideran 3 módulos en el desarrollo del sistema.

- Preprocesamiento.
- Entrenamiento.
- Reconocimiento.

La figura 4.1 y la figura 4.2 muestran el diagrama con los procesos más representativos desde el módulo de Preprocesamiento hasta el módulo de Reconocimiento.

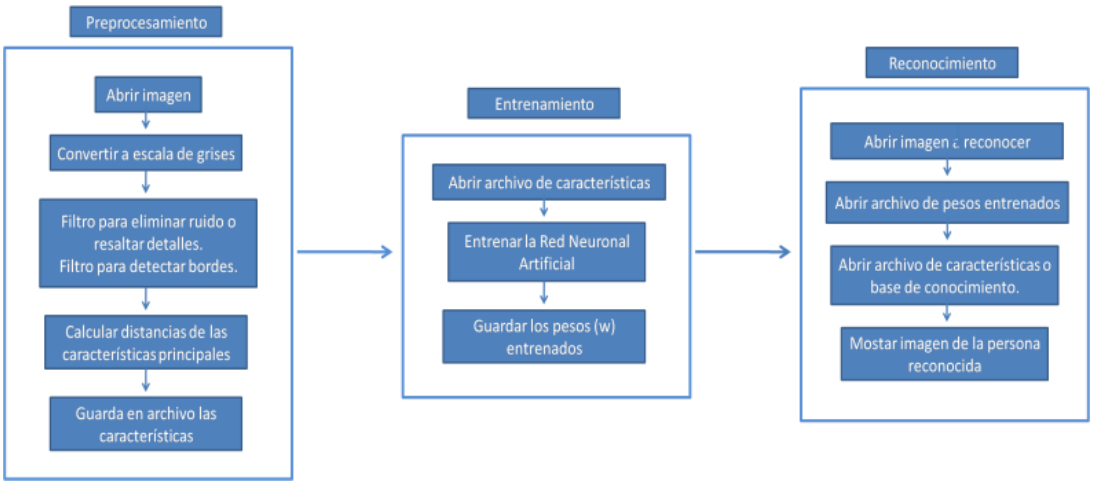


Figura 4.1. Estructura general de los módulos del sistema desarrollado.

Figura 4.2. Analogía de la figura 4.1 con la figura 2.12 (González & Woods 1992).

Tras la apertura de una imagen se procede a su conversión en escala de grises, con la finalidad de preparar la imagen para procesamientos posteriores, como son la eliminación de ruido o realce de características específicas de la imagen y la aplicación de filtros para la detección de bordes. Enseguida se procede a calcular las distancias en la imagen para tener una base de características de la imagen (Tabla VI) , que tendrá datos de salida pertenecientes a una clase *x*, al finalizar este proceso se guardan las distancias en un

archivo de texto que llegará a formar la base de conocimiento del sistema. Después de calcular las distancias de las características de la imagen del rostro el siguiente proceso es entrenar la RNA Perceptron con 4 neuronas con la base de conocimientos creada en el módulo de preprocesamiento. Entrenada la red se procede a guardar los pesos finales del entrenamiento y se pasa al módulo de reconocimiento donde se abre una imagen que se quiera reconocer (clasificar), se abre el archivo de pesos finales del entrenamiento, después de esto, se procede a abrir el archivo donde se encuentran las diferentes clases con sus respectivos objetos y el resultado final es la muestra de la imagen del rostro que fue reconocido por la RNA.

Tabla VI. Distancias principales para la base de conocimiento.

Distancia	Representación	Significado
1		Distancia del alto de la nariz.
2		Distancia del ancho de la nariz.
3		Distancia del ancho del ojo izquierdo.
4		Distancia del alto del ojo izquierdo.
5		Distancia del ancho del ojo derecho.
6		Distancia del alto del ojo derecho.

Tabla VI. Distancias principales para la base de conocimiento (continuación).

Distancia	Representación	Significado
7		Distancia del ancho de la boca.
8		Distancia del alto de la boca.
9		Distancia desde la boca de la parte derecha hasta el final del ojo derecho en su parte derecha.
10		Distancia desde la boca de la parte izquierda hasta el final del ojo izquierdo en su parte izquierda.
11		Distancia desde el ojo izquierdo de la parte izquierda hasta el ojo derecho de la parte derecha.
12		Distancia desde la nariz de la parte derecha hasta el ojo derecho de la parte derecha.
13		Distancia desde la nariz de la parte izquierda hasta el ojo izquierdo de la parte izquierda.
14		Distancia desde la boca de la parte izquierda hasta el ojo derecho de la parte derecha.
15		Distancia desde la boca de la parte izquierda hasta el ojo derecho de la parte izquierda.

Tabla VI. Distancias principales para la base de conocimiento (continuación).

Distancia	Representación	Significado
16		Distancia desde la boca, de la parte de derecha, hasta el ojo izquierdo, de la parte izquierda.
17		Distancia desde la boca, de la parte de derecha, hasta el ojo izquierdo, de la parte derecha.

4.2. Diseño

En este apartado se describen cada uno de los módulos del sistema:

Módulo de preprocesamiento

Este módulo realiza tres tareas principales: el preprocesamiento de la imagen, el procesamiento y el cálculo de las distancias euclideanas, en consecuencia el guardado de las mismas.

El proceso de preprocesamiento y procesamiento de la imagen implica abrir una imagen del rostro de una persona, convertirla a escala de grises (Sec. 3.1) con la finalidad de aplicarle filtros para eliminación de ruido como son pasa bajo (Sec. 3.4) y pasa alto (Sec. 3.5), previo a estos filtros se pueden aplicar la ecualización (Sec. 3.2) o la umbralización (Sec. 3.3) para balancear los niveles de gris. Después se aplican filtros de realce de bordes como el filtro de Sobel (Sec. 3.6), Roberts (Sec. 3.7), Prewitt (Sec. 3.8), Frei-Chen (Sec. 3.9) o Laplaciano (Sec. 3.10), es decir, se necesita tener una imagen limpia de ruido, de sombra o de otros factores (exceso de iluminación, color), donde se distinga el rostro del fondo de la imagen. Por lo tanto, el valor de cada pixel no se verá influenciado, y la imagen estaría lista para iniciar la medición de las distancias principales.

En este módulo se calculan las distancias euclideanas. Por ejemplo, si el usuario va a calcular la distancia del ancho del ojo izquierdo debe presionar en el punto A, arrastrar hasta el punto B y soltar (Fig. 1.1); de ahí automaticamente se calcula la distancia euclideana del punto A al punto B (Ec. 1), estos dos puntos delimitan el ancho del ojo

izquierdo; de la misma manera se calcularán las demás distancias. Al finalizar el cálculo de las 17 distancias principales (Tabla VI), se guardan los datos en un archivo de texto, anteponiendo el nombre de la imagen (clase) correspondiente. Con esto se creará un conjunto de características de esta imagen y así para cada una de las imágenes pertenecientes a las diferentes clases. Para el módulo de reconocimiento, este paso se realiza sólo una vez, para generar un archivo de distancias para cada imagen.

Es preciso puntualizar que para el cálculo de las distancias euclideanas se utiliza los eventos `DragOver` y `DragDrop` del componente `TImage` (componente 2 de la figura C.8), con el evento `DragOver` se captura las coordenadas iniciales de cada distancia y con el `DragDrop` se capturan sus coordenadas finales (Fig. 1.1). Después de dar click al botón `Empezar a medir`, el componente `TImage` (`ImagenProcesamiento`) está listo para que el usuario empiece a medir las distancias euclideanas, las lineas azules que indican las distancias se dibujan en el canvas del componente `ImagenProcesamiento`. A continuación se describe sus algoritmos.

Algoritmo del evento `DragOver`

El algoritmo del evento `DragOver` almacena en un vector la coordenada *x* y en otro vector la coordenada *y* de los puntos iniciales de cada distancia.

- `iVectorPuntoX` es un vector de 100 elementos que se utiliza para almacenar la coordenada *x* de cada punto inicial y final.
- `iVectorPuntoY` es un vector de 100 elementos que se utiliza para almacenar la coordenada *y* de cada punto inicial y final.
- `iPuntoInicial` es una bandera que indica que se va a calcular el punto inicial de cada distancia, el cual es inicializado en cero en el evento clic de la opción `Abrir` del menú `Imagen`.
- `ImagenProcesamiento` es un componente de tipo `TImage` de C++ Builder, sobre su canvas se dibujan las lineas azules que representan las distancias euclideanas.
- `iPosicion` contador que indica el índice en los vectores `iVectorPuntoX` y `iVectorPuntoY`.

1. Se cambia la forma del cursor para indicar visualmente al usuario que está empezando a medir el punto inicial de la distancia.
2. Se verifica se `iPuntoInicial` es igual a cero, para saber que es punto inical de la distancia euclideana, posteriormente se guardan las coordenandas en los correspondiente vectores.
3. Se cambia el valor de la variables `iPuntoInicial` y se incrementa la variable `iPosicion`, lo importante de este evento es que con el se puede obtener las posiciones iniciales de cada distancia.

Algoritmo del evento `DragDrop`

El algoritmo del evento `DragDrop` almacena en los vectores `iVectorPuntoX` y `iVectorPuntoY` las coordenadas de los puntos finales de cada distancia euclideana. Al finalizar el arastrar y soltar se guardan las coordenadas del punto final de las distancias.

- `fDistanciaX` almacena el cuadrado de la diferencia de los valores de `x` del punto inicial y final de cada distancia, es decir, $(A_x-B_x)^2$.
- `fDistanciaY` almacena el cuadrado de la diferencia de los valores de `y` del punto inicial y final de cada distancia, es decir, $(A_y-B_y)^2$.
- `fDistancia` almacena el resultado de (`fDistanciaX` + `fDistanciaY`)$^{1/2}$.
- `fDistanciasEuclideanas` vector que almacena las distancias euclideanas.
- `iPosicionDistancia` es un índice que contabiliza el número de distancias que se deben calcular.
- `nombreImagenes` es un vector donde se almacenan el nombre de las imágenes que se muestran en la parte inferior en el módulo de preprocesamiento en el momento de empezar a calcular las distancias, y las cuales indicar que distancia se debe calcular sobre la imagen.
- `nombreResultadoMedirImagen` es un vector en el que se almacena el nombre de las imágenes que se muestran en la parte inferior en el módulo de preprocesamiento en el momento de empezar a calcular las distancias, éstas van indicando qué distancias se han calculado sobre la imagen.

1. Se almacenan en los vectores `iVectorPuntoX` y `iVectorPuntoY` las coordenadas del punto final de cada distancia, la variable `iPuntoInicial` se le asigna cero.
2. Se toman los valores de las variables `iVectorPuntoX` y `iVectorPuntoY` para dibujar la recta de las distancias, posteriormente se calculan las distancias euclideanas con la ecuación 1, las cuales se almacenan en el vector `fDistanciasEuclideanas`.
3. Se verifica que la distancia dibujada y calculada anteriormente esté en el límite de la variable `DISTANCIAS`.
4. Se repiten los pasos 1, 2 y 3 hasta finalizar el cálculo de todas las distancia, en caso de que haya finalizado el cálculo de las distancia se procede a deshabilitar el componente `ImagenProcesamiento`, al realizar esto, no se invocará a los eventos `DragDrop` y `DragOver` del componente `ImagenProcesamiento`.
5. Se muestran las imágenes para indicarle al usuario qué distancia se debe calcular y qué distancias se han calculado en la imagen, dichas imágenes se observan en la parte inferior del módulo de preprocesamiento.

Al final se tiene un archivo de características (Fig. 4.3). A este archivo le falta agregar los datos de las salidas en las neuronas, es decir, a cada imagen (registro) se le tiene que introducir cuatro datos, estos datos dependen del número de clases que se desee entrenar, y se debe apoyar de las tablas VII y VIII.

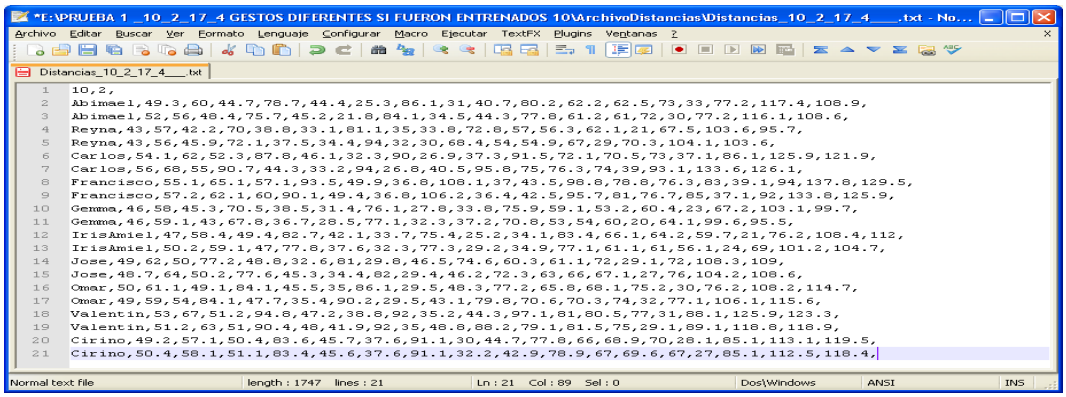


Figura 4.3. Archivo de distancias sin los datos de las salidas en las neuronas.

Tabla VII. Salidas en las neuronas.

Representación decimal	Neurona 1	Neurona 2	Neurona 3	Neurona 4
1	-1	-1	-1	-1
2	-1	-1	-1	1
3	-1	-1	1	-1
4	-1	-1	1	1
5	-1	1	-1	1
6	-1	1	1	-1
7	-1	1	1	1
8	1	-1	-1	-1
9	1	-1	-1	1
10	1	-1	1	-1
11	1	-1	1	1
12	1	1	-1	-1
13	1	1	-1	1
14	1	1	1	-1

Para agregar los últimos cuatro valores a cada registro, el usuario puede usar cualquier editor de texto tomando en cuenta la tabla VII y considerando lo siguiente:

El sistema usa 4 neuronas, $2^n\ neuronas\ =$ Total de clases que pueden clasificar. Por ejemplo si $n = 4$ se tendrá $2^4 = 16$, por tal motivo este sistema puede clasificar un máximo de 16 clases.

Tabla VIII. Salidas para la función hardlims.

Representación decimal	Neurona 1	Neurona 2	Neurona 3	Neurona 4
0	0	0	0	0
1	0	0	0	1
2	0	0	1	0
3	0	0	1	1
4	0	1	0	0
5	0	1	0	1
6	0	1	1	0
7	0	1	1	1
8	1	0	0	0
9	1	0	0	1
10	1	0	1	0
11	1	0	1	1
12	1	1	0	0
13	1	1	0	1
14	1	1	1	0

Después de agregar y guardar el archivo, se tiene un archivo de distancias completo (Fig. 4.4) y listo para usar en el módulo de entrenamiento.

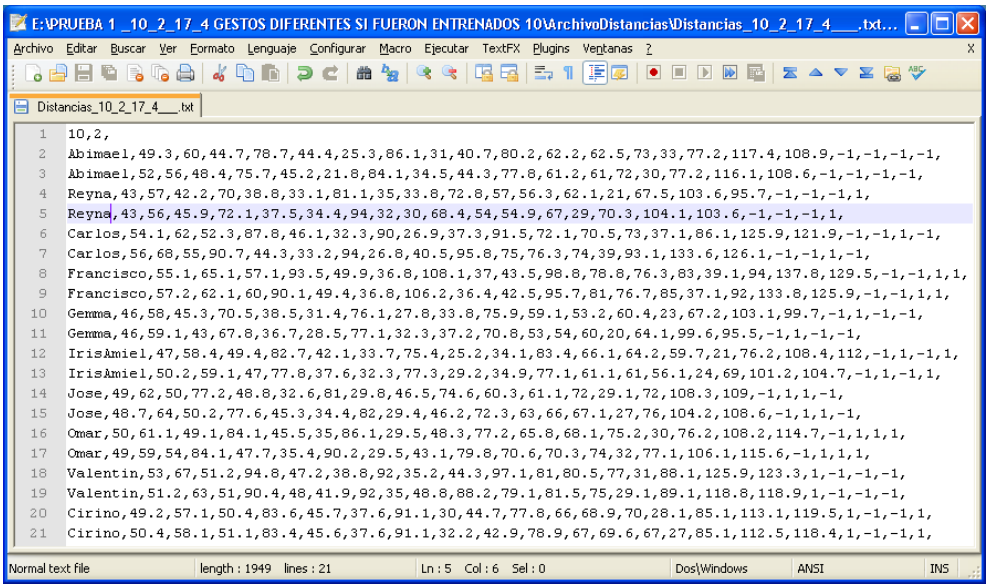


Figura 4.4. Estructura final del archivo de distancias.

A continuación se describe el algoritmo para la creación de un archivo de distancias, en consecuencia de la ejecución de los eventos `DragOver` y `DragDrop` del componente `TImage`, se tiene un vector que contiene todas las distancias euclideanas que fueron calculadas, al terminar de calcular las distacncias se habilita el botón `Guardar distancias` que invoca el algoritmo de guardar las distancias euclideanas en un archivo de texto, como se mencionó anteriormente.

Algoritmo del evento `clic` del botón `Guardar distancias`

Después de haber ejecutado los eventos `DragOver` y `DragDrop`, las distancias se almacena en el vector `fDistanciasEuclideanas` y se ejecuta el código del evento clic del botón `Guardar distancias.`

- `bGuardarArchivo` bandera que indica que las distancias fueron guardadas correctamente en el archivo de texto.
- `ARegistro` es una cadena tipo `AnsiString` donde se concatena las distancias y también se agrega una coma para separar cada distancias.
- `CuadroDialogoAperturArchivo` cuadro de diálogo para guardar archivos propio de C++ Builder.

1. Se realiza un redondeo de cada distancias euclideana, las cuales se encuentran en la variable `fDistanciasEuclideanas`, en un ciclo se itera desde 0 hasta el total de distancias, se toma cada distancia se multiplica por 10 se suma 0.5, porteriormente se divide entre 10.
2. Todas las distancias se concatena en la variables `ARegistro` también se le agrega una coma al final de cada distancias para su posterior separación.
3. Se invoca al cuadro de diálogo para guardar el archivo de distancia, se proporcionan los datos que pide dicho cuadro de diálogo.

Módulo de entrenamiento

La estructura principal de la RNA Perceptron con 4 neuronas, que se diseñó para este trabajo de tesis y que se utiliza en este módulo, se observa en la figura 4.5.

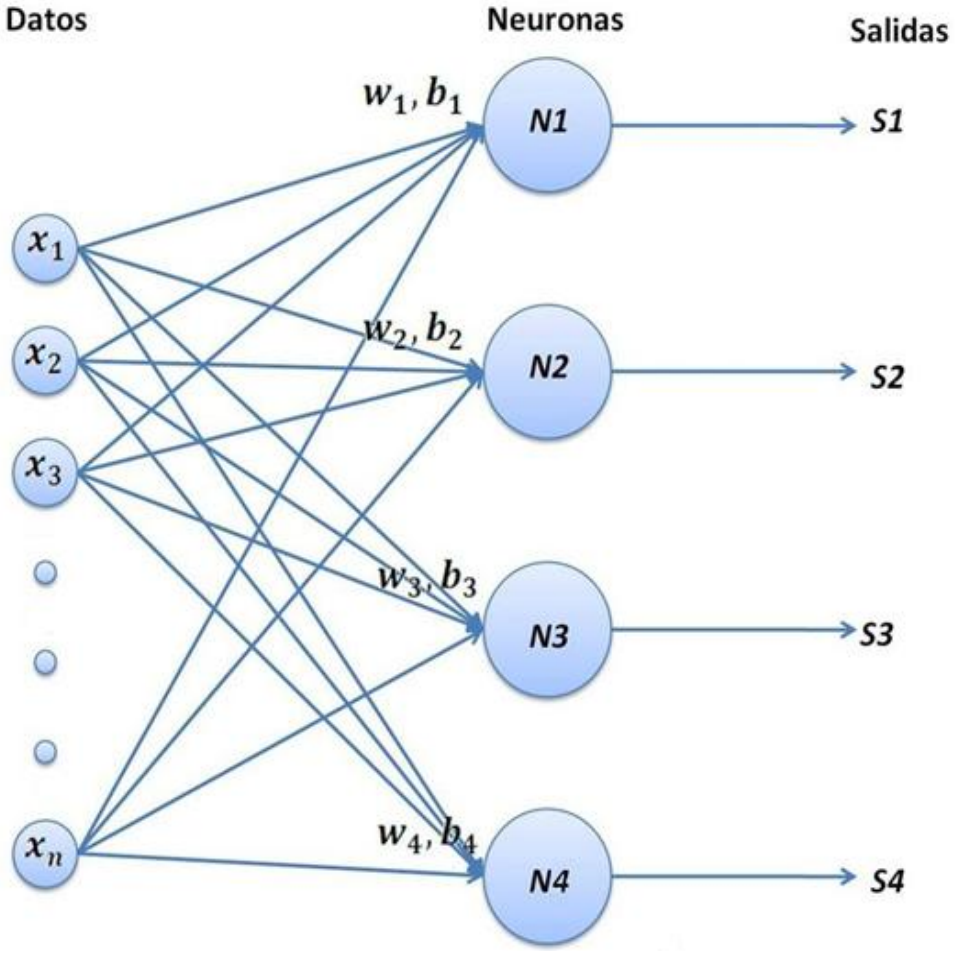


Figura 4.5. Estructura general de la RNA Perceptron con 4 neuronas.

Donde:

$x_1, x_2, x_3, \ldots, x_n$ = = Vectores de distancia de las características principales de cada rostro.

N_i = Número de neurona.

w_i, b_j = Valores de pesos y ganancias iniciales aleatorios para cada neurona, de igual manera representan los pesos y ganancias al final del entrenamiento.

S_i = Resultado de salida, obtenido de la función de transferencia de la RNA Perceptron con 4 neuronas.

El módulo de entrenamiento se basa en el archivo creado en el módulo de preprocesamiento. Se abre el archivo de distancias (Fig. 4.4), los registros del archivo son incorporados al algoritmo de entrenamiento sobre la RNA Perceptron con 4 neuronas, con pesos y ganancias inicializadas aleatoriamente (no se consideraron valores para los pesos y ganancias estáticos, con el fin de determinar la efectividad del sistema en cada caso de prueba), para cada una de las cuatro neuronas. Al culminar el algoritmo se obtienen los pesos y ganancias finales (Fig. 4.6). Básicamente es lo que hace el proceso de entrenamiento (Fig. 4.7).

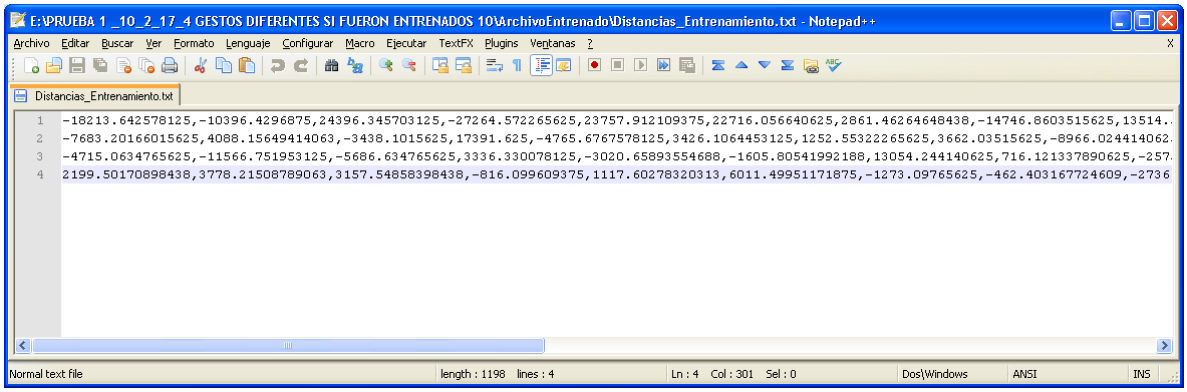


Figura 4.6. Archivo de pesos obtenidos en el entrenamiento.

Después de terminar el entrenamiento se tienen los pesos y ganancias finales en las variables `dMatrizPesosFinales` y `dVectorGanaciasFinales`, el proceso de guardar dichos datos es el mismo que se realiza en el código 3.

La estructura del archivo de pesos obtenidos en el entrenamiento consta de cuatro renglones, cada renglón tiene 17 columnas, cada columna está conformada de números reales de 4 a 8 dígitos en su parte entera y de 10 a 12 dígitos en su parte decimal, cada

número está separado por comas y al final de cada renglón existe una coma (Fig. 4.6) para indicar el fin del registro. El usuario puede elegir el nombre del archivo y la ubicación donde se almacenará el archivo.

En la figura 4.7 se representa cada una de las etapas del módulo de entrenamiento, primero se tienen los datos de entrada (registros, pesos y ganancias), después dichos datos se introducen a la RNA y finalmente se obtiene los datos finales.

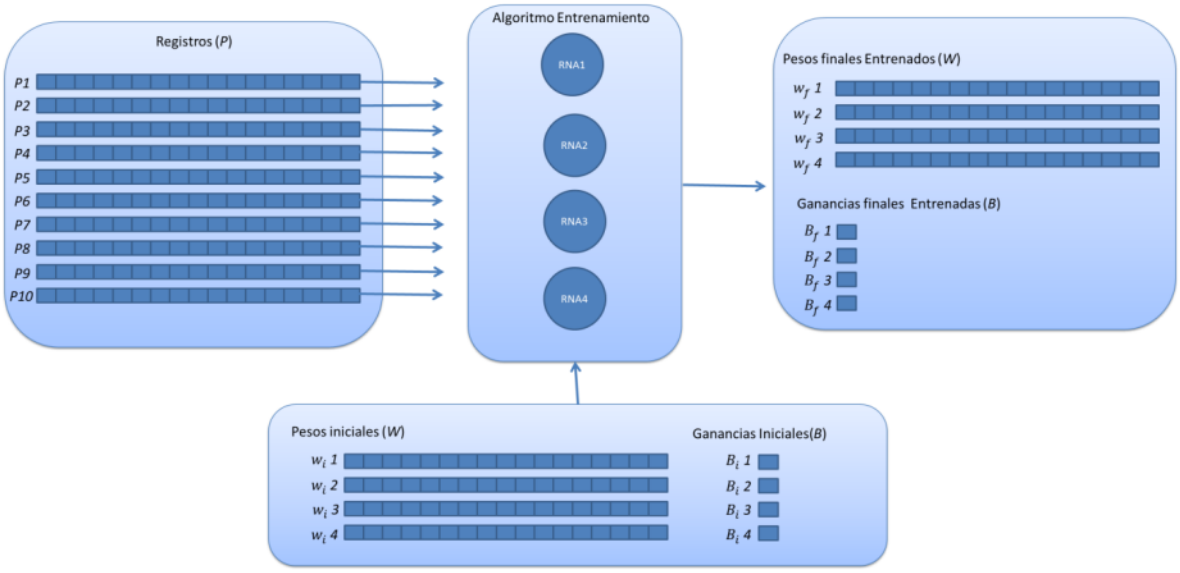


Figura 4.7. Proceso interno del entrenamiento del archivo de características.

Para complementar el proceso de la figura 4.7, se describe el algoritmo de entrenamiento de la RNA, el algoritmo de carga de datos tras la apertura del archivo de distancias, el algoritmo principal de la RNA y el algoritmo de modificación de pesos en una neurona determinada, complementando los códigos anteriores se describe el diagrama de flujo de la RNA principal a partir del paso 4.

Algoritmo de entrenamiento de la RNA

Este método se ejecuta al presionar el botón `Entrenar neurona` (Fig. C.23) del módulo de entrenamiento, dicho método tiene como función entrenar la RNA.

- `SALIDAS` esta variable expresa el número de neuronas de la RNA.
- `DISTANCIAS` variable que indica el número de distancias euclideanas de cada imagen u objeto.

- `fMatrizPesos` es una matriz de 4 filas con 17 columnas, que contiene datos reales entre [-1 1], la cual es inicializada aleatoriamente.
- `fVectorGanancias` es un vector de 4 elementos con datos reales entre [-1 1], el cual es inicializado aleatoriamente.
- `dMatrizPesosFinales` esta variable contiene los valores de los pesos al final del entrenamiento de la RNA, con las mismas propiedades de la variable `fMatrizPesos`.
- `dVectorGanaciasFinales` es una variable con las mismas caracteristicas de la variable `fVectorGanancias` al terminar el entrenamiento de la RNA.
- `iTotalRegistros` es una variable tipo entero que almacena el número de objetos que tiene un archivo de distancias (Fig. 4.4).
- `iError1`, `iError2`, `iError3` e `iError4` son variables que almacenan el valor retornado de la función `RedNeuronalArtificialPerceptron`, dependiendo del número de neurona que se envie a dicha función.
- `iContadorEpocas` almacena el número de épocas que realizó la RNA al finalizar el entrenamiento.
- `iContadorError` indica cuántos errores se obtienen en cada época, esta bandera debe ser igual a la variable `iTotalRegistro`, para que el entrenamiento finalice.

1. Se crean e inicializan las variables locales de la función: `iError1`, `iError2`, `iError3`, `iError4`, `iContadorEpocas`, `iContadorError`, `iContadorEpocas`.
2. Se invoca a la función `AbrirDistanciasImagenArchivoTexto`, en la cual se cargan los datos del archivo de distancias.
3. Se copian los datos de las variables `fMatrizPesos`, `fVectorGanancias` a las variables `dMatrizPesosFinales`, `dVectorGanaciasFinales` respectivamente.
4. Se itera hasta que `iContadoError` sea igual a `iTotalRegistros`, esto indicará el fin del entrenamiento de la RNA, en dicho ciclo se realiza lo siguiente:

4.1. Se inicializan las variables `iContadorError` y se incrementa `iContadorEpocas` y se invoca la función `AbrirDistanciasImagenArchivoTexto`, para cargar los datos.

4.2. Se realiza un ciclo con los n objetos del archivo de distancias, cada uno de estos objetos son enviados a la función `RedNeuronalArtificialPerceptron` y se capturan los datos retornados de dicha función en las variables inicializadas en el paso 4.1, es decir, que se invoca cuatro veces la función `RedNeuronalArtificialPerceptron`, para tener cuatro datos en las variables antes mencionadas, a este ciclo se le denomina época.

4.3. Se verifica cada variable, y si alguna es diferente de cero, se invoca a la función `ModificarPesos`, a la cual se le envía el número de neurona , el objeto y la salida.

4.4. Sólo en caso de que las cuatro variables sean iguales a cero, se incrementa la variable `iContadorError`.

4.5. Los pasos 4.1 al 4.4 se repiten hasta que `iContadorEpocas` sea igual al `iTotalRegistros`, es decir, hasta que ninguno de los pesos en las neuronas sean modificados. A esto se le denomina fin del entrenamiento de la RNA.

Algoritmo de la función `AbrirDistanciasImagenArchivoTexto`

Este método es invocado por el código del algoritmo de entrenamiento de la RNA en el paso 4.1 (algoritmo anterior) y se encarga de crear las variables y almacenar los datos que contiene el archivo de distancias en cada renglón del archivo, se obtiene la información para crear la matriz de nombres, matriz de distancias, matriz de salidas en cada neurona. El primer renglón es lo primero que se analiza para crear las variables principales.

- `fNumero` almacena un número de entre [0-9] generado aleatoriamente.
- `fSigno` almacena 0 ó 1, generado aleatoriamente.
- `iVectorNumeros` vector de dos elementos que almacena los datos del primer renglón, el número de clases y el número de objetos por clase.

- `iContadorComas`, `iIndiceRenglon` e `iIndiceColumnas`, contadores para comas, renglones y columnas, que se utilizan para separar los datos de cada registro.
- `ANumero` clase que almacena un `string`, que posteriormente es convertido a número.
- `cPunteroPrimeraLinea` almacena la dirección de la primera línea del archivo de características.
- `cAuliliarLinea` es un vector de caracteres con espacio para 50 elementos.
- `sLineaArchivo` es un `string` que almacena un registro del archivo cada vez que se recorre el archivo.
- `iNumeroRostros` almacena el número de clases que tiene el archivo de distancias.
- `iRostrosIguales` almacena el número de objetos por clase.
- `cNombreClase` puntero que almacena la dirección de un vector, que se utiliza para almacenar el nombre de las clases, cada registro tiene un primer dato y es el nombre de la clase.
- `dVectorElementosP` puntero a una matriz, que almacena las 17 distancias de cada objeto.
- `iVectorSalidas` vector de 4 columnas y n registros dinámicos, que almacena para salidas de cada neurona de cada objeto.

1. Se abre el archivo con opciones de lectura para leer la primera línea y se cierra, se extraen de dicha línea lo separado por comas que indican el número de clase y los objetos por clase que tiene el archivo de distancias y se almacenan en su correspondiente variable.
2. Del producto de `iNumeroRostros` con `iRostrosIguales` se obtiene el total de objetos, el cual se almacena en `iTotalRegistros`.
3. Cabe mencionar que en esta parte se utilizan las variables estáticas del código 1 (`DISTANCIAS`, `SALIDAS`), se crean punteros de las variables globales `cNombreClase`, `dVectorElementosP`, `iVectorSalidas`, `fMatrizPesos`, `fVectorGanancias`, `dMatrizPesosFinales` con sus

respectivas dimensiones, las variables `fMatrizPesos` y `fVectorGanancias` son inicializadas aleatoriamente.

4. Se abre el archivo de distancias pero ahora partiendo de la segunda línea, ya que desde ésta empiezan los datos de cada registro, cada dato esta separado por una coma, el primer dato corresponde al nombre de la clase por lo que se guarda en su correspondiente variable, los 17 datos siguientes de cada registro indican las 17 distancias principales de cada objeto y se almacenan en la matriz `dVectorElementosP`, los últimos cuatro datos se almacena en la variable `iVectorSalidas`, se va leyendo cada línea del archivo y se van separando los datos de cada objeto hasta el fin del archivo.

Algoritmo de la función `RedNeuronalArtificialPerceptron`

Este método (`RedNeuronalArtificialPerceptron`) es invocado en el código principal de la RNA en el paso 4.2, el cual realiza la operación principal de la RNA la función de activación, este método recibe dos variables tipo entero, la primera indica el número de neurona a modificar, el segundo valor es el número de objeto que se esta manipulando en el archivo (Fig. 4.4).

- `iNeurona` es una variable tipo entero, que puede tener los siguientes valores: 0, 1, 2 o 3, las cuales indican el número de nuerona (1, 2, 3 y 4 respectivamente) enviada a este método.
- `iIndice` es una variable tipo entero, que indica el registro del archivo de distancias sobre el cual se esta trabajando.

1. Se inicializan las variables locales de este método posteriormente se realiza un ciclo hasta la variable `DISTANCIAS` y se realiza lo siguiente:
    1.1 Se realiza un producto escalar de la matriz `dVectorElementosP` con `dMatrizPesosFinales` y se asigna a la variable `fHardlims`, aquí se realiza un producto de las distancias de un objeto con el peso de una neurona.
2. Al finalizar el ciclo del paso 1 se agrega a la variable `fHardlims` la ganancia de la neurona, este dato está almacenado en la variable `dVectorGanaciasFinales`

y la variable `iNeurona` sera el índice en dicho arreglo, para obtener el dato correspondiente.

3. Si `fHardlims` es mayor e igual que cero se le asigna 1, de lo contrario -1.
4. A la variable `dError` se le asigna la diferencia de la variable iVectorSalidas en los índices `iNeurona` e `iIndice` con la variable `fHardlims` obtenida en el paso 3, finalmente se retorna la variable dError.

Algoritmo de la función `ModificarPeso`

El método `ModificarPeso` es invocado en el código de la RNA pricipal en el paso 4.3, el cual recibe tres argumentos: `iNeurona`, `iIndice` y `iError`, aquí se realiza la operación de modificar el peso de una neurona, la variable `iIndice` indica la posición del objeto a modificar, `iNeurona` indica el número de neurona a modificar y `iError` es un valor que puede ser 1 o -1, dicho valor es retornado por el código 3 y enviado a esta función.

- `fAuxiliar` variables que almacena el producto de `iError` con `dVectorElementosP`.

1. Se declara la variable `fAuxiliar`, posteriormente se realiza un ciclo desde cero hasta el total de distancias, en el cual se realiza lo siguiente:
    1.1 En la variable del paso 1 se almacena el producto de `iError` con `dVectorElementosP` en los índices `iIndice` y el contador del ciclo del paso 1.
    1.2 Se asigna a `dMatrizPesosFinales` en el índice `iNeurona` la suma de `dMatrizPesosFinales` en el mismo subíndice con la variable del paso 1.1 se realiza esto hasta con todas las distancias.
2. Al finalizar el ciclo del paso 1, se asigna a la variable dVectorGananciasFinales en la posición iNeurona el contenido de ésta, más el valor de la variable iError.

Ejemplo de una iteración de la RNA principal

Para ejecutar la corrida del código de la RNA principal, se utilizaron los datos de la figura 4.8, es decir, se utilizó un archivo de distancias con 5 clases, 2 objetos por clase, 5

distancias principales y 4 salidas. Considerando que los datos ya fueron cargado (Fig. 4.9) en sus correspondientes variables (Paso 1, 2 y 3 del código de la RNA principal) se procede desde el paso 4.

CasodePrueba_5_2

```
1  5,2,
2  Abimael,49.3,60,44.7,78.7,44.4,-1,-1,-1,-1,
3  Abimael,52,56,48.4,75.7,45.2,-1,-1,-1,-1,
4  Reyna,43,57,42.2,70,38.8,-1,-1,-1,1,
5  Reyna,43,56,45.9,72.1,37.5,-1,-1,-1,1,
6  Carlos,54.1,62,52.3,87.8,46.1,-1,-1,1,-1,
7  Carlos,56,68,55,90.7,44.3,-1,-1,1,-1,
8  Francisco,55.1,65.1,57.1,93.5,49.9,-1,-1,1,1,
9  Francisco,57.2,62.1,60,90.1,49.4,-1,-1,1,1,
10 Gemma,46,58,45.3,70.5,38.5,-1,1,-1,-1,
11 Gemma,46,59.1,43,67.8,36.7,-1,1,-1,-1,
```

Figura 4.8. Archivo de distancias utilizado por el algoritmo de la RNA principal.

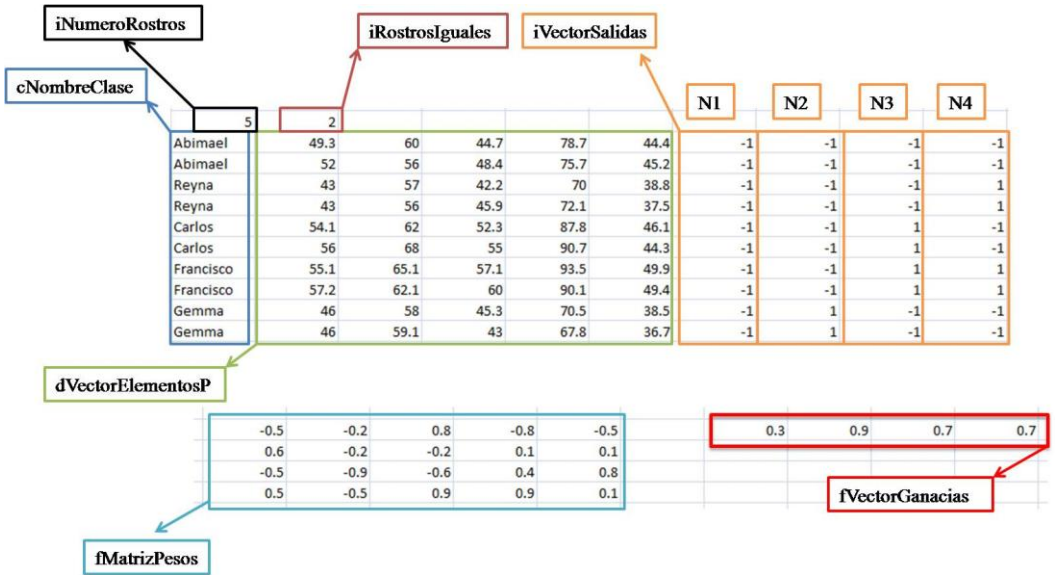


Figura 4.9. Almacenamientos de variables del archivo de distancias de la figura 4.8.

Cabe a aclarar que las variables `fMatrizPesos` y `fVectorGanancias` son inicializadas aleatoriamente, la variable `iTotalRegistros` es igual al producto de `iNumeroRostros` con `iRostrosIguales` (Fig. 4.9). En la tabla IX se muestra los resultados de código de la RNA principal.

Tabla IX. Iteración 0 a partir del paso 4 del código de la RNA principal.

Iteración	0
iContadorError	0
iContadorEpocas	1
i	0
iTotalRegistros	10
iError1 = RedNeuronalArtificialPerceptron(0,i)	0
iError2 = RedNeuronalArtificialPerceptron(1,i)	-2
iError3 = RedNeuronalArtificialPerceptron(2,i)	0
iError4 = RedNeuronalArtificialPerceptron(3,i)	-2

A continuación se invoca cuatro veces a la función `RedNeuronalArtificialPerceptron` como se muestra en la tabla IX, sus respectivas iteraciones se observan en las tablas X, XI, XII y XIII.

Tabla X. Código `RedNeuronalArtificialPerceptron`(0,0).

iNeurona	0
iIndice	0
iError	0
fHardlims	0
i	0
DISTANCIAS	5
dVectorElementosP[iIndicei][i]	[49.3,60,44.7,78.7,44.4]
dMatrizPesosFinales[iNeurona][i]	[-0.5,-0.2,0.8,-0.8,-0.5]
fHardlims+= dVectorElementosP[0][0]* dMatrizPesosFinales[0][0]	0+(-86.05)
dVectorGananciasFinales[iNeurona]	0.3
fHardlims+= dVectorGananciasFinales[0]	-86.05+0.3 = -85.75
fHardlims	-1
iVectorSalidas[iIndice][iNeurona]	-1
iError	-1 – (-1) = 0

Tabla XI. Código `RedNeuronalArtificialPerceptron`(1,0).

iNeurona	1
iIndice	0
iError	0
fHardlims	0
i	0
DISTANCIAS	5
dVectorElementosP[iIndicei][i]	[49.3,60,44.7,78.7,44.4]
dMatrizPesosFinales[iNeurona][i]	[0.6,-0.2,-0.2,0.1,0.1]
fHardlims+= dVectorElementosP[0][0]* dMatrizPesosFinales[1][0]	0+(20.95)
dVectorGananciasFinales[iNeurona]	0.9
fHardlims+= dVectorGananciasFinales[1]	20.95+0.9 = 21.85
fHardlims	1
iVectorSalidas[iIndice][iNeurona]	-1
iError	-1 – (1) = -2

Tabla XII. Código `RedNeuronalArtificialPerceptron`(2,0).

`iNeurona`	2
`iIndice`	0
`iError`	0
`fHardlims`	0
`i`	0
`DISTANCIAS`	5
`dVectorElementosP[iIndicei][i]`	[49.3,60,44.7,78.7,44.4]
`dMatrizPesosFinales[iNeurona][i]`	[-0.5,-0.9,-0.6,0.4,0.8]
`fHardlims+= dVectorElementosP[0][0]* dMatrizPesosFinales[2][0]`	0+(-38.47001)
`dVectorGananciasFinales[iNeurona]`	0.7
`fHardlims+= dVectorGananciasFinales[2]`	-38.47001+0.7 = -37.77
`fHardlims`	-1
`iVectorSalidas[iIndice][iNeurona]`	-1
`iError`	-1 – (-1) = 0

Tabla XIII. Código `RedNeuronalArtificialPerceptron`(3,0).

`iNeurona`	3
`iIndice`	0
`iError`	0
`fHardlims`	0
`i`	0
`DISTANCIAS`	5
`dVectorElementosP[iIndicei][i]`	[49.3,60,44.7,78.7,44.4]
`dMatrizPesosFinales[iNeurona][i]`	[0.5,-0.5,0.9,0.9,0.1]
`fHardlims+= dVectorElementosP[0][0]* dMatrizPesosFinales[3][0]`	0+(110.15)
`dVectorGananciasFinales[iNeurona]`	0.7
`fHardlims+= dVectorGananciasFinales[3]`	110.15+0.7 = 110.85
`fHardlims`	1
`iVectorSalidas[iIndice][iNeurona]`	-1
`iError`	-1 – (1) = -2

Como se observa en la Tabla IX las variables `iError2` e `iError4` son diferentes de 0, por tal motivo se invoca al código `ModificarPeso` para realizar la modificación de los pesos en sus respectivas neuronas, dichas modificaciones se observan en las tablas XIV y XV. Realizado lo anterior se incrementa la variable `iContadorError` sólo si las variables `iError1`, `iError2`, `iError3` y `iError4` son iguales a cero, esto se realiza para los demás registros restantes del archivo de características (Fig. 4.8), en caso de que `iContadorError` es igual a `iTotalRegistros` finaliza el entrenamiento, de lo contrario se vuelve a iterar.

Tabla XIV. Código `ModificarPeso`(1,0,-2).

`iNeurona`	1
`iIndice`	0
`iError`	-2
`fAuxiliar`	0
`i`	0
`DISTANCIAS`	5
`dVectorelementosP[iIndice][i]`	[49.3,60,44.7,78.7,44.4]
`dMatrizPesosFinales[iNeurona][i]`	[0.5,-0.5,0.9,0.9,0.1]
`fAuxiliar = iError* dVectorelementosP[iIndice][i]`	[-98.6,-120,-89.4,-157.4,-88.8]
`dMatrizPesosFinales[iNeurona][i]+ =fAuxiliar`	[-98.1,-120.5,-88.5,-156.5,-88.7]
`dVectorGananciasFinales[iNeurona]`	0.7
`dVectorGananciasFinales[iNeurona]+=iError`	-1.3

Tabla XV. Código `ModificarPeso`(3,0,-2).

`iNeurona`	3
`iIndice`	0
`iError`	-2
`fAuxiliar`	0
`i`	0
`DISTANCIAS`	5
`dVectorelementosP[iIndice][i]`	[49.3,60,44.7,78.7,44.4]
`dMatrizPesosFinales[iNeurona][i]`	[0.6,-0.2,-0.2,0.1,0.1]
`fAuxiliar = iError* dVectorelementosP[iIndice][i]`	[-98.6,-120,-89.4,-157.4,-88.8]
`dMatrizPesosFinales[iNeurona][i]+ =fAuxiliar`	[-98,-120.2,-89.6,-157.3,-88.7]
`dVectorGananciasFinales[iNeurona]`	0.9
`dVectorGananciasFinales[iNeurona]+=iError`	-1.1

El diagrama de flujo (Fig. 4.10) describe el proceso principal del código de la RNA principal, el cual inicia a partir del paso 4. Para complementar la informacion de las tablas IX, X, XI, XII, XIII, XIV y XV.

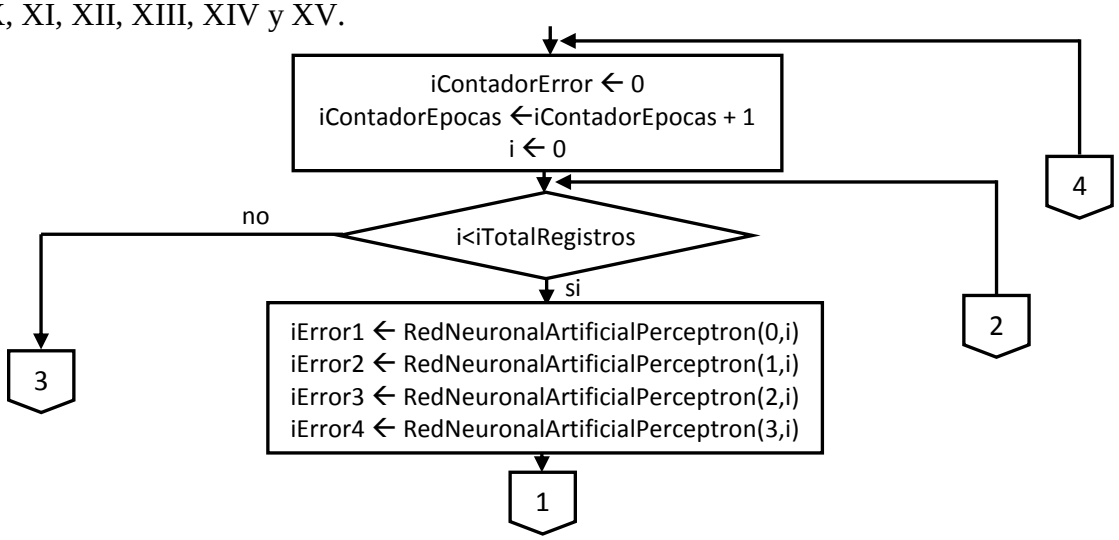


Figura 4.10. Diagrama de flujo desde el paso 4 del código de la RNA principal.

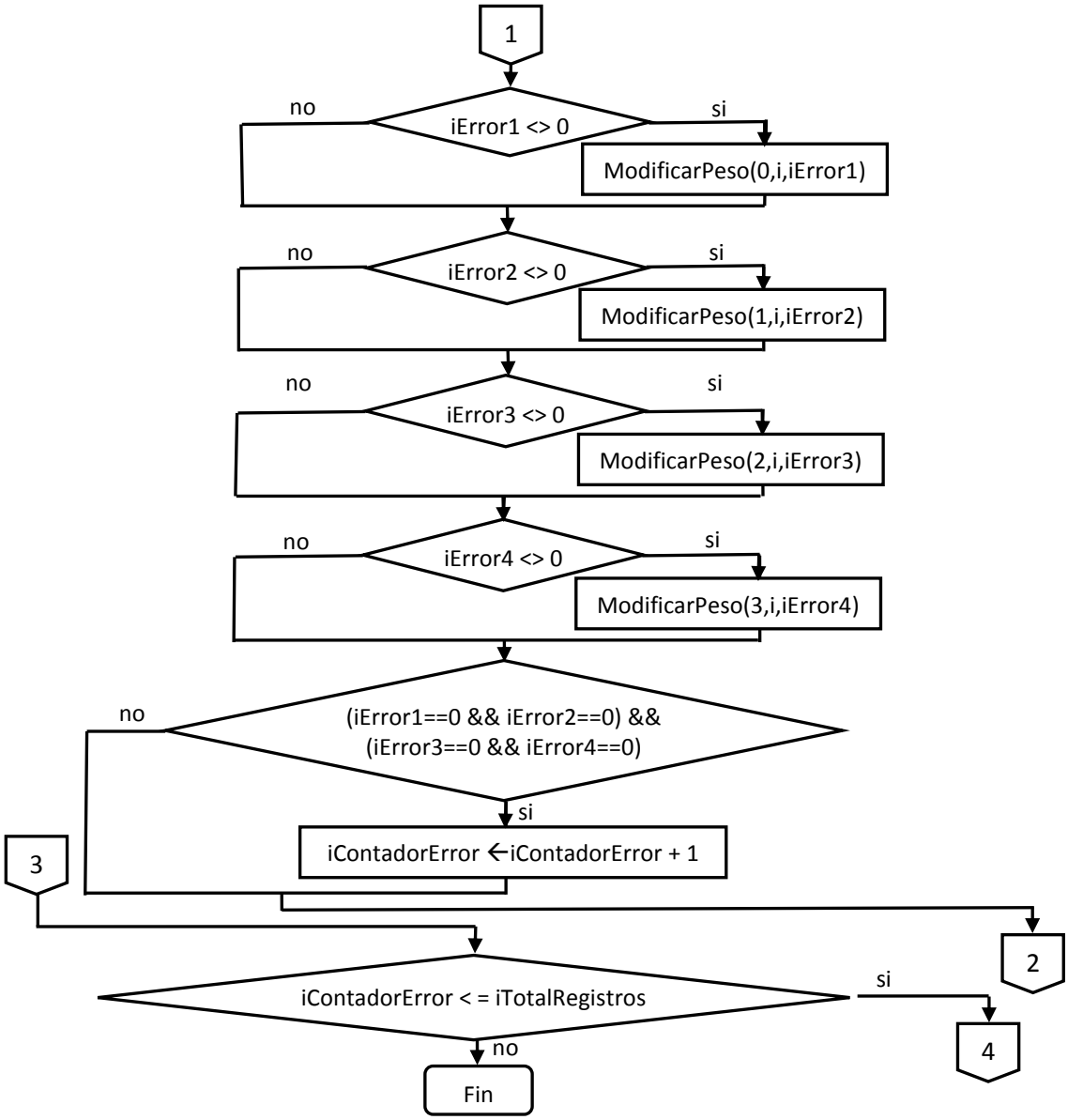


Figura 4.10. Diagrama de flujo desde el paso 4 del código de la RNA principal (continuación).

Módulo de reconocimiento

En este último módulo se utilizan los resultados del proceso de entrenamiento. Primero se abre una imagen que se desee identificar, al abrir la imagen se abre también su archivo de distancias (un archivo con un solo registro que pertenece a esa imagen), se carga el archivo donde se encuentran los pesos y ganancias que fueron obtenidas en el módulo de entrenamiento (Fig. 4.6), se realiza el proceso de reconocimiento con el registro y el algoritmo de la RNA Perceptron con 4 neuronas para obtener el resultado de las salidas

sobre cada neurona, después el usuario tendrá que abrir el archivo de distancias (Fig. 4.4) para que el programa realice la búsqueda y encuentre el registro que coincida con las salidas del reconocimiento, finalmente se muestra la imagen del rostro de la persona a la cual la RNA Perceptron con 4 neuronas asoció dicha información (Fig. 4.10). La imagen a reconocer se muestra a la izquierda y la imagen que reconoce la RNA Perceptron con 4 neuronas se muestra a la derecha con un mensaje que indica el nombre de la imagen reconocida, en caso contrario se mostrará un mensaje "No pertenece a alguna clase". El resultado esperado es que ambas imágenes sean de la misma persona. Sin embargo, el sistema puede mostrar a una persona diferente dependiendo de la similitud de sus caracteristicas. En la figura 4.11 se observa el proceso interno del módulo.

Estos tres módulos están inmersos en la GUI y es la encargada de mostrar las imágenes de entrenamiento y las imágenes de reconocimiento en sus respectivos módulos, así también mostrar los datos que se utilizaron en el entrenamiento.

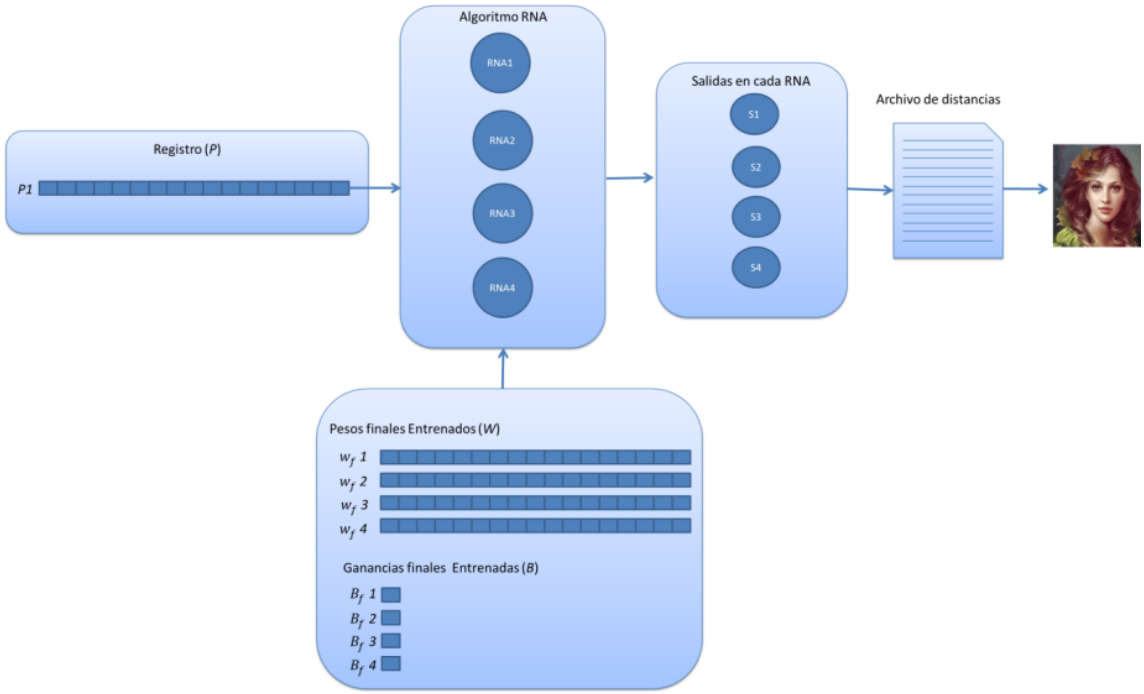


Figura 4.11. Proceso interno del módulo de reconocimiento.

Con base en los datos que se muestran en la figura 4.12 se invoca el algoritmo `FuncionOperacionReconocimiento` que se describe a continuación, el cual se ejecuta al dar clic al botón `Cargar datos de aprendizaje`. Cabe aclarar que dicho botón realiza otros procesos pero la función más importante se realiza en esta función.

Figura 4.12. Objeto utilizado en el algoritmo de la RNA principal.

Algoritmo de la función `FuncionOperacionReconocimiento`

Este algoritmo `FuncionOperacionReconocimiento` es invocado al dar clic a botón `Cargar datos de aprendizaje`, el cual recibe un `string`, que es el registro de la figura 4.12 que se va a reconocer.

- `cDatos` dirección de la variables que recibe esta función, a los datos que se muestran en la figura 4.9.
- `ARegistro` variables que almacena cada línea del archivo de distancias.
- `ADistancia` variables tipo `AnsiString` que captura cada distancias y posteriormente es convertido a su valor en `double`.
- `fHardlims` almacena el resultado de la multiplicación del peso de una neurona con el registro a reconocer.
- `dVectorDistancias` vector que almacena las distancias euclideanas, primero son almacenadas por la variables `ADistancia` y posteriormente convertidas a `double`.
- `iIndice` índice para el vector `dVectorDistancias.`
- `iSalidasReconocimiento` vector de cuatro elementos que almacena las salidas resultantes de la operación de reconocimiento.
- `dMatrizPesosEntrenados` matriz que contiene los pesos entrenados.
- `dGanaciasEntrenadas` vector que contiene las ganancias entrenadas.
- `ASalidasReconocimiento string` donde se almacena las salidas de las neuronas, se concatena cada valor agregando una coma para separar cada datos y al final se agrega otra.

1. Se asigna a la variable `ARegistro` la dirección de la variable `cDatos`.
2. Se realiza un ciclo hasta que sea fin de línea o hasta la última coma que exista en la variables `ARegistro`.

3. Se descompone la variable `ARegistro` por el delimitador ",",
   3.1. Se almacena la distancia en la variables `ADistancia` se convierte a `double` y se almacena en el vector `dVectorDistancias` en el índice `iIndice`.
4. Se realiza un ciclo hasta el total de neuronas, es decir se repiten cuatro veces el siguiente código.
   4.1. Se inicializa `fHardlims` y `iSalidasReconocimiento` a cero.
   4.2. Se realiza un ciclo hasta el total de distancias euclideanas y se realiza lo siguiente.
      4.2.1. Se realiza una multiplicación escalar de los vectores `dMatrizPesosEntrenados` con `dVectorDistancias`, el resultado se almacena en la variables `fHardlims`.
5. Al final del ciclo del paso 3.2 se le suma la ganancia entrenada de la neurona correspondiente.
6. Si el valor de `fHardlims` es mayor que cero se le asigna 1 de contrario se le asigna -1.
7. El valor de `fHardlims` es asignado al vector `iSalidasReconocimiento`, dicho valor es concatenado a la variable `ASalidasReconocimiento`.

Ejemplo de una iteración del algoritmo de reconocimiento

La figura 4.12 muestra el registro que se utilizó para esta iteración, dicho dato se muestra en el componente 2 del módulo de reconocimiento (Fig. C.25), dicha cadena se almacena en `ARegistro` del paso 1 del anterior algoritmo. En la figura 4.13 se muestra el archivo de pesos y ganancias entrenadas obtenido en el módulo de entrenamiento.

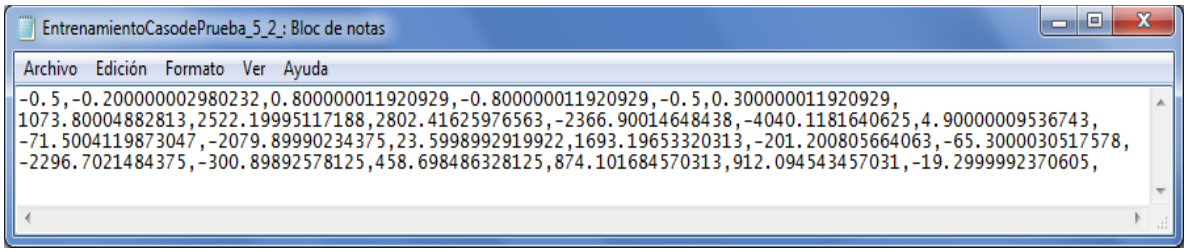


Figura 4.13. Archivo de pesos y ganancias entrenadas utilizado en el código de la RNA principal.

Considerando los datos de la figura 4.13. se descompone dicho archivo en una matriz de pesos entrenados y en un vector de ganancias entrenadas como se puede observar en la figura 4.14, estas variables son importantes en la ejecución algoritmo `FuncionOperacionReconocimiento.`

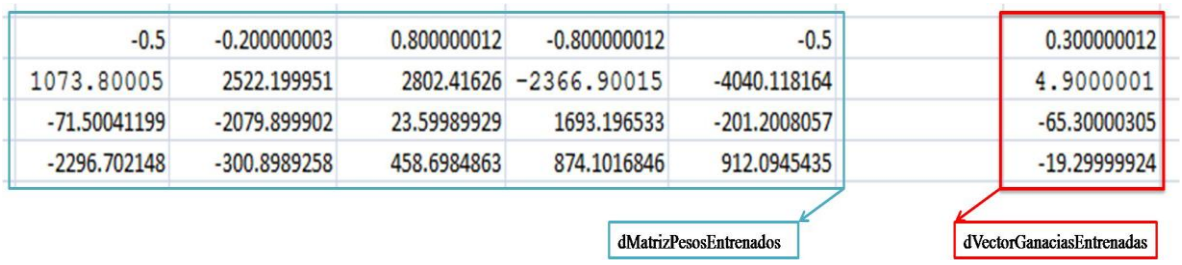


Figura 4.14. Almacenamientos de variables del archivo de la figura 4.10.

La tabla XVI muestra una iteración del algoritmo `FuncionOperacionReconocimieto`

Tabla XVI. Código `FuncionOperacionReconocimiento("43,56,45.9,72.1,37.5,").`

cDatos	"43,56,45.9,72.1,37.5,"
ARegistro	""
ADistancia	""
dVectorDistancias[200]	0
fHardlims	12434E-12
iIndice	0
ARegistro	"43,56,45.9,72.1,37.5,"
i	1,2,**3**,4,5,**6**,7,8,9,10,**11**,12,13,14,15,**16,**17,18, 19,20,**21**
j , j=i +1	1, 4,7,12,17,22
ADistancia	43,56,45.9,72.1,37.5
dVectorDistancias[iIndice]	[43 56 45.9 72.1 37.5]
iIndice	1,2,3,4,5
i	0
j	0
fHardlims	0
SALIDAS	4
iSalidasReconocimiento[i]	0
dVectorDistancias[j]	[43 56 45.9 72.1 37.5]
dMatrizPesosEntrenados[i][j]	[-0.5 0.2 0.8 -0.8 -0.5]
fHardlims	-21.5,-32.7,4.02,-53.66,-72.41
dGanaciasEntrenadas[i]	0.3
fHardlims=fHardlims+dGanaciasEntrenadas[i]	-72.11
fHardlims	-1
iSalidasReconocimiento[i]	-1
ASalidasReconocimiento+= AnsiString(fHardlims)+AnsiString(",")	"-1,"
i	1
fHardlims	0
iSalidasReconocimiento[i]	0

Tabla XVI. Código `FuncionOperacionReconocimiento("43,56,45.9,72.1,37.5,")` (continuación).

j	0
dVectorDistancias[j]	[43 56 45.9 72.1 37.5]
dMatrizPesosEntrenados[i][j]	[1073.8 2522.2 2802.416 -2366.9 -4040.118]
fHardlims	46173.4,187416.6,316047.5,145394,-6110.431
dGanaciasEntrenadas[i]	4.9
fHardlims=fHardlims+ dGanaciasEntrenadas[i]	-6105.531
fHardlims	-1
iSalidasReconocimiento[i]	-1
ASalidasReconocimiento+= AnsiString(fHardlims)+AnsiString(",")	"-1,-1,"
i	2
fHardlims	0
iSalidasReconocimiento[i]	0
j	0
dVectorDistancias[j]	[43 56 45.9 72.1 37.5]
dMatrizPesosEntrenados[i][j]	[-71.50041 -2079.9 23.5999 1693.197 -201.2008]
fHardlims	-3074.518,-119548.9,-118465.7,3613.79,-3931.24
dGanaciasEntrenadas[i]	-65.3
fHardlims	-1
iSalidasReconocimiento[i]	-1
ASalidasReconocimiento+= AnsiString(fHardlims)+AnsiString(",")	"-1,-1,-1"
i	3
fHardlims	0
iSalidasReconocimiento[i]	0
j	0
dVectorDistancias[j]	[43 56 45.9 72.1 37.5]
dMatrizPesosEntrenados[i][j]	[-2296.702 -300.8989 458.6985 874.1017 912.0945]
fHardlims	-98758.2,-115608.5,-94554.27,-31531.54,2672.004
dGanaciasEntrenadas[i]	-19.3
fHardlims=fHardlims+dGanaciasEntrenadas[i]	2652.704
fHardlims	1
iSalidasReconocimiento[i]	1
ASalidasReconocimiento+= AnsiString(fHarlims)+AnsiString(",")	"-1,-1,-1,1,"
fHardlims	-98758.2,-115608.5,-94554.27,-31531.54,2672.004

Para quitar las dudas que pudiera tener el algoritmo `FuncionOperacionReconocimiento`, en la figura 4.15 se describe el diagrama de flujo, esperando dar una idea más amplia en los proceso que se realizaron en este sistema.

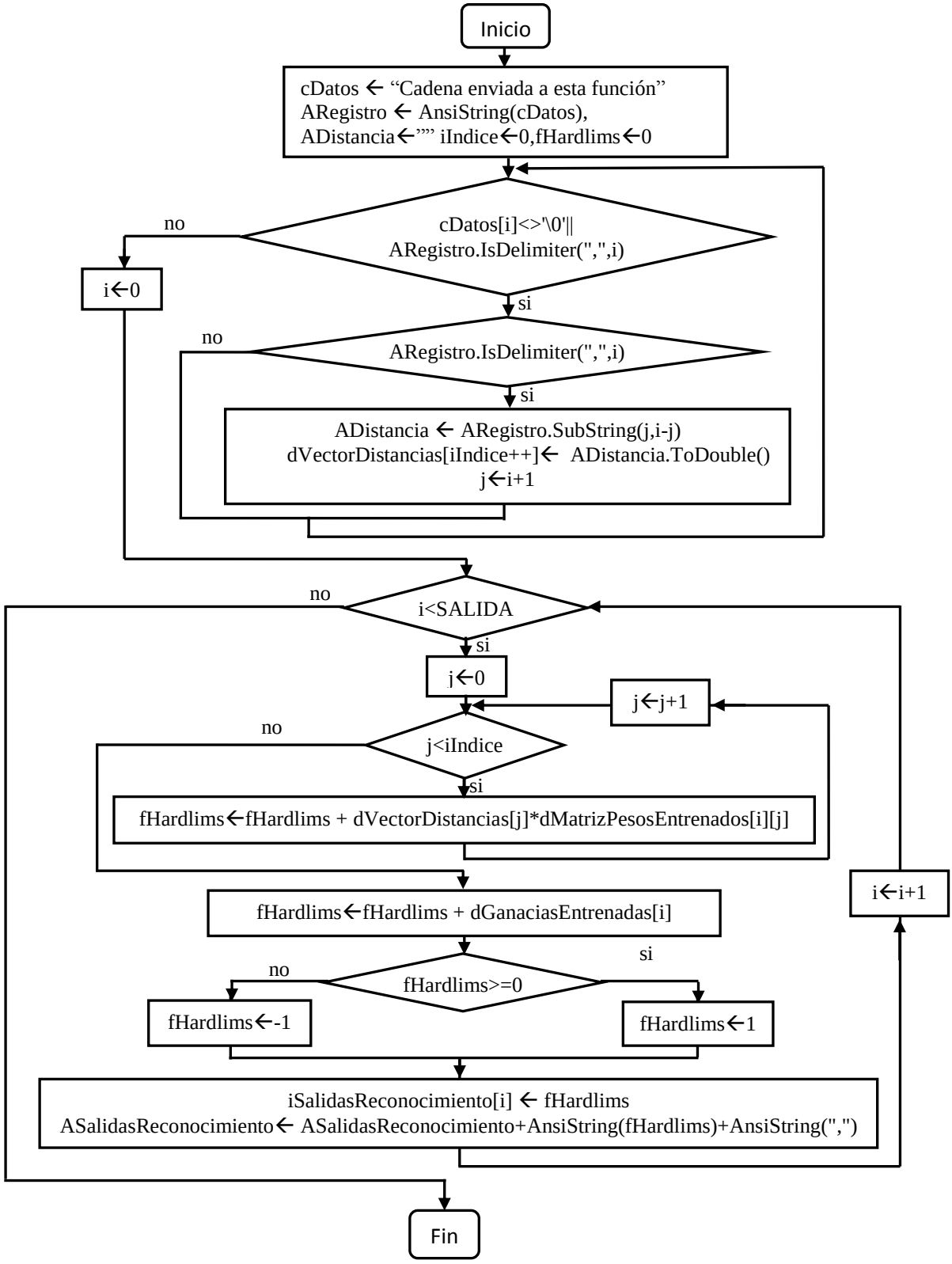


Figura 4.15. Diagrama de flujo del algoritmo `FuncionOperacionReconocimiento`.

Al finalizar el algoritmo `FuncionOperacionReconocimiento` se tiene las salidas resultantes en la variables `ASalidasReconocimiento` esta cadena se utiliza en el algoritmo `btnReconocerImagenRClick` para realizar la búsqueda en el archivo de distancias, para obtener la coincidencia y mostrar la imagen que reconoció la RNA.

Algoritmo de la función `btnReconocerImagenRClick`

El algoritmo `btnReconocerImagenRClick` busca la coincidencia de la variables `ASalidasRecononocimiento` en el archivo de distancias.

- `sLineaArchivoDatos` variables que almacena una línea leída del archivo de distancias.
- `ALinea` variable auxiliar de la variables `sLineaArchivoDistancia`.
- `bImagenEncontrada` es una variables que se vuelve verdadera cuando se encuentra la coincidencia de la variable `ASalidasRecono` en el archivo de distancias.
- `iIndice` almacena el entero que regresa la función buscar (`find`, propia de C++) en `sLineaArchivoDistancia` la cadena `ASalidasReconocimiento`.
- `ASalidasReconocimiento` es la cadena a buscar en el archivo de distancias, es decir, son las salidas en las neuronas, del registro a reconocer.
- `ANombreReconocido` es el nombre de la clase que se extrae de la línea leída.
- `AbrirArchivoDatosAprendizaje` variable `TopenDialog` que muestra una cuadro de diálogo para abrir un archivo, en este caso txt.
- `NombreArchivoEntrenamiento` variables `ifstream` que guarda el buffer del archivo de distancias.

1. Se inicializan las variables `sLineaArchivoDatos`, `ALinea`, `bImagenEncontrada`, `iIndice` a valores nulos.
2. Al dar clic al botón `Reconocer` imagen se abre un cuadro de diálogo para abrir el archivo de distancias, se busca, selecciona el archivo y se da clic en abrir.
3. Se lee el archivo mientras no sea fin de archivo o hasta que `bImagenEncontrada` sea verdadero, nos posicionamos en la segunda línea, que

es donde inicia la información que necesitamos en este módulo, se realiza lo siguiente:

3.1 Cada vez que se lee una línea se busca la cadena `ASalidasReconocimiento`, para encontrar la clase a la que pertenecen.

3.2 Cuando se encuentre dicha coincidencia del paso 3.1, a dicha línea se le extrae el nombre de la clase, dicha cadena se almacena en `ANombreReconocido` que será el nombre de la imagen reconocida, este nombre se le agrega a la ruta C:\Reconocimiento\ImagenesReconocer\ que es donde se almacenan las imágenes que se muestran para reconocer, además de que se le agrega la cadena "1.bmp" y se muestra un mensaje "Se reconoció a" + nombre de clase.

4. Se carga la imagen encontrada en el paso 3.2, que es la reconoció la RNA.
5. Al finalizar el ciclo se verifica si `bImagenEncontrada` es `false`, si no se encontró ninguna coincidencia se envía un mensaje "No pertenece a alguna clase".

Ejemplo de una iteración del algoritmo `btnReconocerImagenRClick`

Al final de la tabla XVI se observa `ASalidasReconocimiento="-1,-1,-1,1,"`, dicha cadena es buscada en el archivo de distancias, cuando sea localizada se obtienen el nombre de la clase en el renglón que se encuentre la coincidencia. Para mostrar en el módulo de reconocimiento la imagen que reconoció la RNA. En la tabla XVII se realiza una iteración de dicho algoritmo.

Tabla XVII. Iteracion 0 del algoritmo `btnReconocerImagenRClick()`.

`ANombreReconocido`	NULL
`ASalidasReconocimiento`	"-1,-1,-1,1,"
`sLineArchivoDatos`	""
`ALinea`	""
`iIndice`	-1
`bImagenEncontrada`	`false`
`sLineArchivoDatos`	"5,2,"
`sLineArchivoDatos`	"Abimael,49.3,60,44.7,78.7,44.4,-1,-1,-1,-1,"
`iIndice`	-1
`sLineArchivoDatos`	"Abimael,52,56,48.4,75.7,45.2,-1,-1,-1,-1,"
`iIndice`	-1
`sLineArchivoDatos`	"Reyna,43,57,42.2,70,38.8,-1,-1,-1,1,"
`iIndice`	25
`ALinea`	"Reyna,43,57,42.2,70,38.8,-1,-1,-1,1,"
`ANombreReconocido`	Reyna
`Ruta`	"C:\\Reconocimiento\\ImagenesReconocer\\Reyna1.bmp"

Tabla XVII. Iteracion 0 del algoritmo `btnReconocerImagenRClick()` (continuación).

`bImagenEncontrada`	`True`
`ASalidasReconocimiento`	“”

Para un mayor entendimiento en la figura 4.16 se muestra una representación en diagrama de flujo del algoritmo `btnReconocerImagenRClick`.

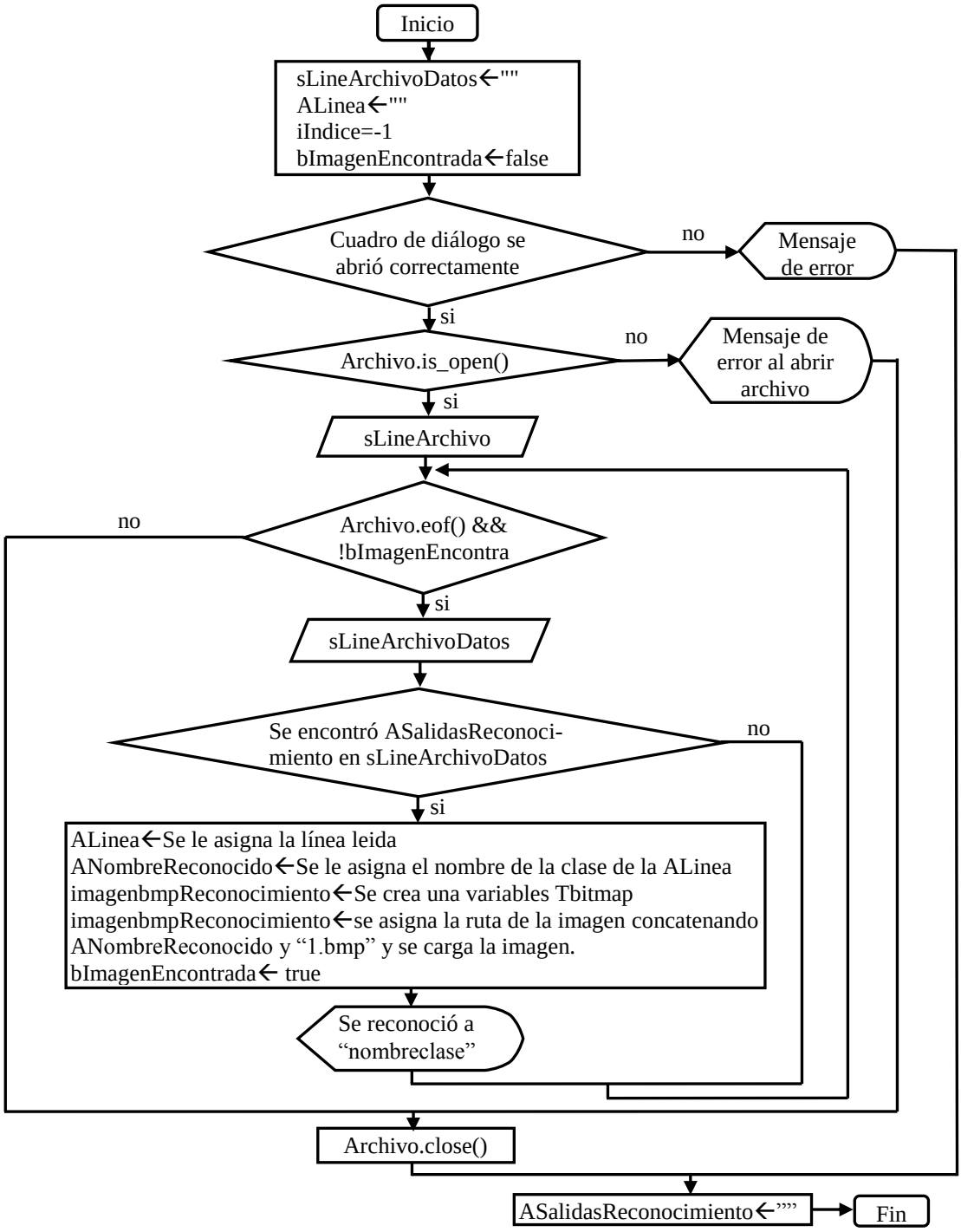


Figura 4.16. Diagrama de flujo del algoritmo `btnReconocerImagenRClick`.

Estructura de la base de conocimiento

Para la base de conocimiento que se utiliza en la fase de entrenamiento de la RNA Perceptron con 4 neuronas se tiene un archivo de texto (Fig. 4.4) con la siguiente estructura: en la primera línea se indican las características generales del archivo, existen 2 números separados por comas agregando también una coma al final de cada renglón; el primer valor indica el número de clases (personas) a entrenar, el cual varia de 3 a 16 clases; el segundo número indica la cantidad de objetos por cada clase (imágenes de rostros de la misma persona a entrenar), este número varía de 1 a *n* objetos por clase (*n* esta limitado a la memoria y capacidad de procesamiento de la computadora), para esta investigación el valor máximo para *n* es 4; existen dos datos implícitos que se consideran en el programa para estos archivos: 17 y 4, cantidad de distancias que se calculan en cada imagen y el número de neuronas para el entrenamiento respectivamente, cabe mencionar que estos valores son fijos ya que el algoritmo está diseñado para esta cantidad de distancias.

El archivo de las distancias de cada uno de los objetos debe estar en la ruta "C:\Archivos de programa\Reconocimiento\ArchivosDistancias\". En la primera línea, como se mencionó anteriormente, deberá especificar los datos contenidos en este archivo, es decir, deberá contener; número de clases y número de objetos por clase. En el módulo de procesamiento se termina de crear el archivo de distancias, cada vez que se abre una imagen y se calculan sus distancias, el usuario guardará cada registro al archivo (dando clic en el componente 6 del módulo de preprocesamiento, figura C.8).

A partir del segundo renglón se tiene la descripción de cada uno de los objetos de cada clase. La primera columna indica el nombre de la imagen o clase, la cual se obtiene cuando se abre dicha imagen, las siguientes 17 columnas pertenecen a las distancias euclideanas de la imagen, son números reales de dos a tres dígitos en su parte entera con un número en su parte decimal, las últimas 4 columnas de cada registro son las salidas que identifican a esa clase, estos 4 datos sólo pueden ser 1 o -1. Si realiza la multiplicación del número de clases por el número de objetos de cada clase, se obtiene el número de objetos del archivo. Esta estructura se muestra en la tabla XVIII, teniendo al final un archivo como el que se muestra en la figura 4.4.

Tabla XVIII. Estructura del archivo de distancias.

	Distancias					**Salidas**			
Clase	1	2	3	……..…….	17	1	2	3	4
Abimael	49.3	60	44.7	……..…….	108.9	-1	-1	-1	1
Abimael	52.2	59	45.4	……..…….	108.1	-1	-1	-1	1
Reyna	43	57	70	……..…….	95.7	-1	-1	1	1
Reyna	43	56	72.1	……..…….	103.6	-1	-1	1	1

4.3. Implementación

Los requerimientos de hardware y software donde se albergó y fue desarrollado el producto de este trabajo de tesis son:

- Dell Optiplex 745.
- Procesador Pentium D 3.0 GHz.
- 1 GB de RAM.
- Sistema Operativo Windows XP SP3.
- Resolución de pantalla de 1280 por 1024 pixeles.

El lenguaje y entorno de programación que se utilizó para el desarrollo del presente proyecto de tesis fue Borland C++ Builder 5.0 *enterprise edition*, dicho entorno cumple con las cualidades necesarias para el manejo de procesamiento digital de imágenes, gráficos, vectores, matrices, librerías para el manejo de caracteres, funciones matemáticas, etc. y facilidad para crear interfaces gráficas.

4.4. Pruebas y resultados

En este apartado se describen y muestran ejemplos de las pruebas que se realizaron para fundamentar los resultados escritos en las conclusiones del capítulo 5. Cabe señalar que se implementaron filtros como Sobel, Roberts, Prewitt, Frei-Chen y Lapalciano, sin embargo, se eligió el filtro de Sobel (Anexo E) debido a que los resultados obtenidos en algunas pruebas realizadas durante el desarrollo de este proyecto entregó mejores resultados que los otros filtros.

Caso de prueba 1

El objetivo de esta prueba es determinar si el entrenamiento de la RNA es correcto, con base en el resultado que se obtenga en el módulo de reconocimiento, debido a que las imágenes utilizadas para el reconocimiento también se consideraron en el entrenamiento.

Para esta prueba se utilizaron las imágenes de las figuras A.1 y A.2, para el módulo de entrenamiento.

Lo que se espera como resultado de esta prueba es que se asocie cada una de las imágenes de la figura A.2 con las imágenes utilizadas en el módulo de entrenamiento.

Para esta primera prueba se muestra un ejemplo con los pasos que se realizan desde el módulo de preprocesamiento hasta el módulo de reconocimiento, el ejemplo práctico es de una alumna (Reyna) de la UMAR campus Puerto Escondido. El archivo de la figura 4.4 es el que se utilizó para esta prueba.

La primera línea del archivo es: "10,2,", por lo tanto hay $10 \text{ x } 2 = 20$ renglones u objetos, más 1 renglón de la primera línea del archivo. Cada objeto se compone en su primera columna por su nombre de clase, de la columna 2 a la 18, son las características de cada objeto (Tabla VI), las últimas 4 columnas de cada objeto, es el valor de salida por cada una de las cuatro neuronas respectivamente. Por lo anterior los *n* objetos que pertenecen a cada una de las clases tendrán el mismo valor en sus cuatro neuronas, así con cada uno de los siguientes objetos o registros del archivo. El ejemplo fue hecho con los datos del registro 3 o renglón cuatro de la figura 4.4.

Módulo de preprocesamiento

Paso 1. Estando la aplicación abierta, se selecciona el menú `Imagen`, se da clic a la opción `Preprocesamiento`, por lo tanto se mostrará la pestaña de preprocesamiento (Fig. 4.17), después se selecciona la opción `Abrir` (Fig. 4.18a) del menú `Imagen`, con lo que se abrirá la ventana de la figura 4.19, en la cual se selecciona la imagen a la que se le van a calcular las distancias del rostro y representarán las características. Después se elige el botón `Abrir` en dicha ventana, a continuación se mostrará la imagen en el módulo de preprocesamiento, como se muestra en la figura 4.20.

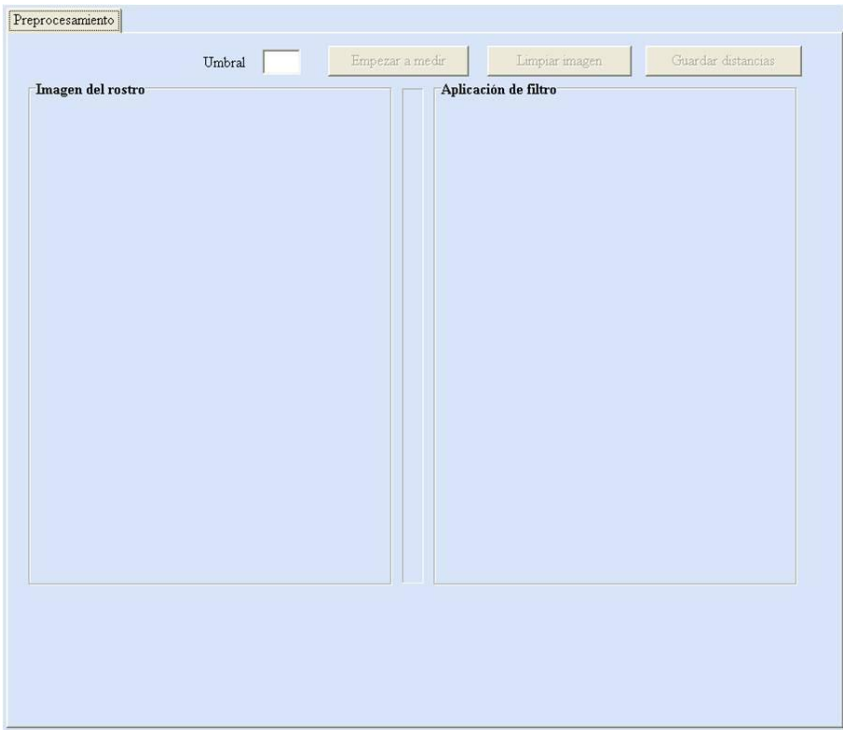
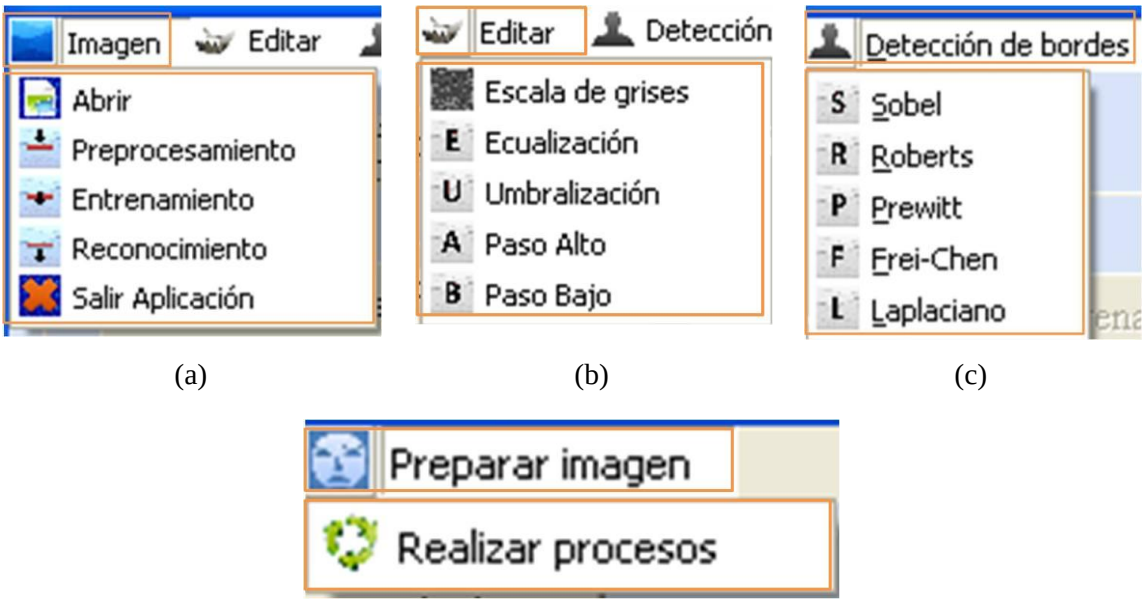


Figura 4.17. Módulo de preprocesamiento del sistema.



(a) (b) (c)

(d)

Figura 4.18. Menús del sistema. (a) Menú *`Imagen`*, (b) Menú *`Editar`*, (c) Menú *`Detección de bordes y`* (d) Menú *`Preparar imagen`*.

Figura 4.19. Ventana de abrir imagen.

Paso 2. En este paso se aplica primero la escala de grises a la imagen mostrada en la figura 4.20 cuyo resultado se muestra en la figura 4.21, a continuación se aplica el filtro de Sobel del menú `Detección de bordes` (Fig. 4.18c), el resultado al aplicar el filtro Sobel se muestra en la imagen de la figura 4.22.

Figura 4.20. Imagen cargada en el módulo de preprocesamiento.

Figura 4.21. Aplicación de escala de grises a la imagen de la figura 4.20.

Figura 4.22. Aplicación del filtro de Sobel a la imagen de la figura 4.21.

Paso 3. Después de hacer el paso 2, se da clic al botón `Empezar a medir` para empezar a marcar las 17 distancias. Este proceso mostrará la imagen que se puede apreciar en la figura 4.23 y muestra qué distancias se van a calcular, así como el resultado de las distancias calculadas.

Figura 4.23. Imagen después de accionar el botón `Empezar a medir`.

El proceso para calcular las distancias es presionar el botón izquierdo del mouse y arrastrar hasta donde se desea calcular la distancia (Fig. 1.1), el resultado de haber calculado las primeras 10 distancias se muestra en la figura 4.24.

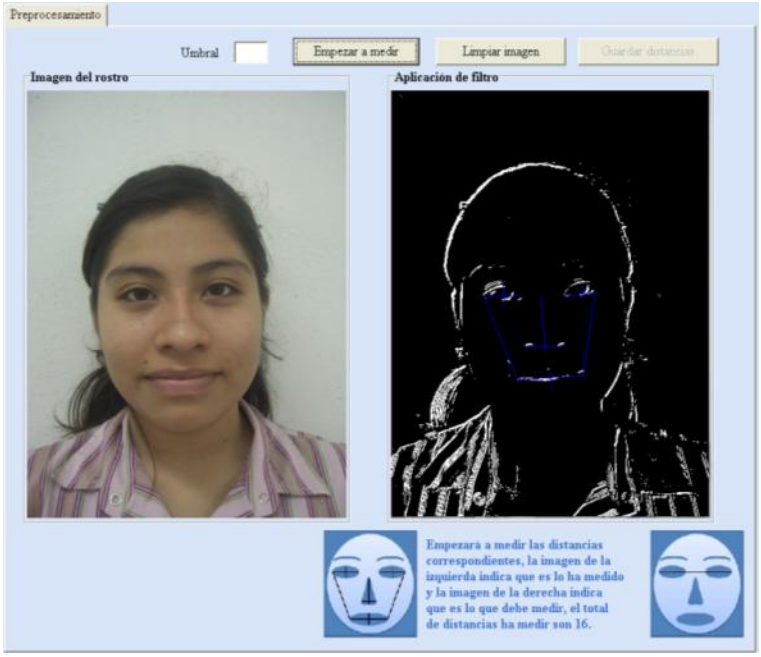


Figura 4.24. Imagen después de haber calculado las 10 primeras distancias.

Al finalizar el cálculo de las 17 distancias se activará el botón `Guardar distancias`, por lo tanto se procede a guardar el registro (C:\Archivos de programa\ Reconocimiento\ArchivosDistancias\) de la persona en el archivo de distancias (Fig. 4.25). Estos tres pasos se repiten dependiendo del número de imágenes a las que se deseen calcular sus distancias de características o dependiendo el número de imágenes por persona, en este caso este paso se realizaría 20 veces, ya que el archivo de texto indica "10,2," en su primera línea (Fig. 4.4).

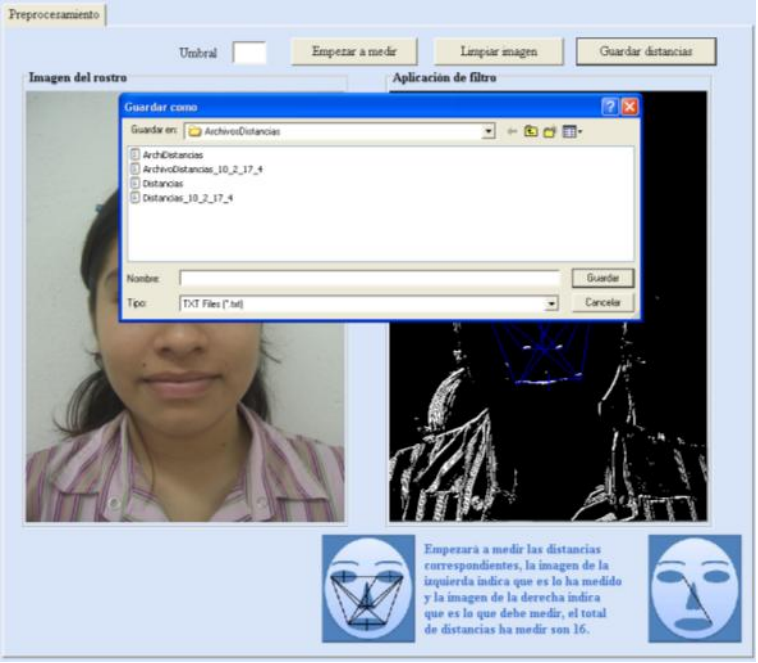


Figura 4.25. Imagen después de calcular todas las distancias y proceder a guardar los datos.

Módulo de entrenamiento

En este apartado se muestran los datos que se calcularon en el módulo anterior y se entrena la red neuronal. A continuación se muestran las imágenes como ejemplo de los procesos que se realizan en este módulo.

Paso 1. Después de terminar con el módulo de preprocesamiento, lo siguiente que hay que hacer es abrir el módulo de entrenamiento, para esto se va al menú `Imagen` se da clic en la opción `entrenamiento` (Fig. 4.18a) y el sistema muestra la pestaña que se aprecia en la figura 4.26.

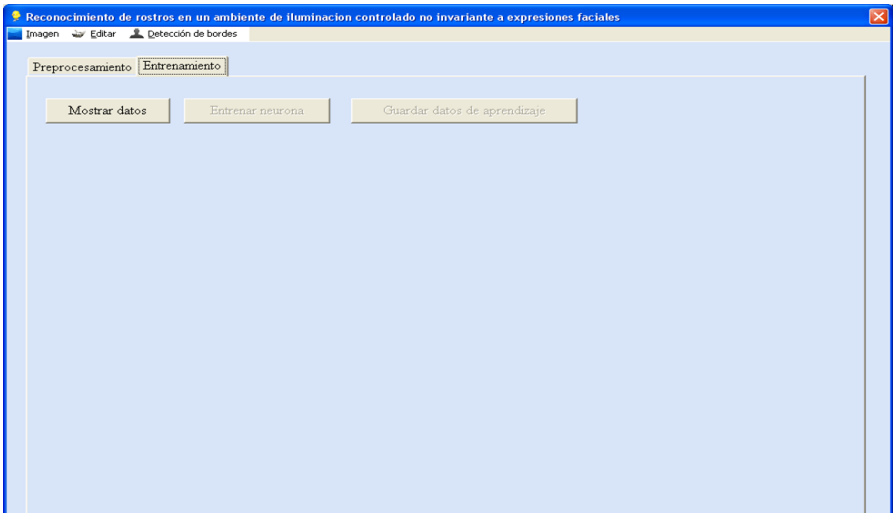


Figura 4.26. Módulo de entrenamiento del sistema.

Teniendo activo este módulo se procede a abrir el archivo de distancias, se da clic al botón `Mostrar datos` para buscar el archivo de distancia como se muestra en la ventana de abrir archivo (Fig. 4.27).

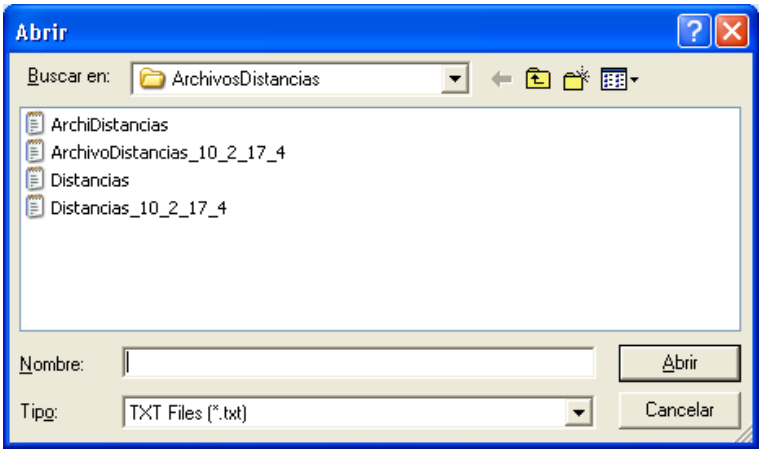
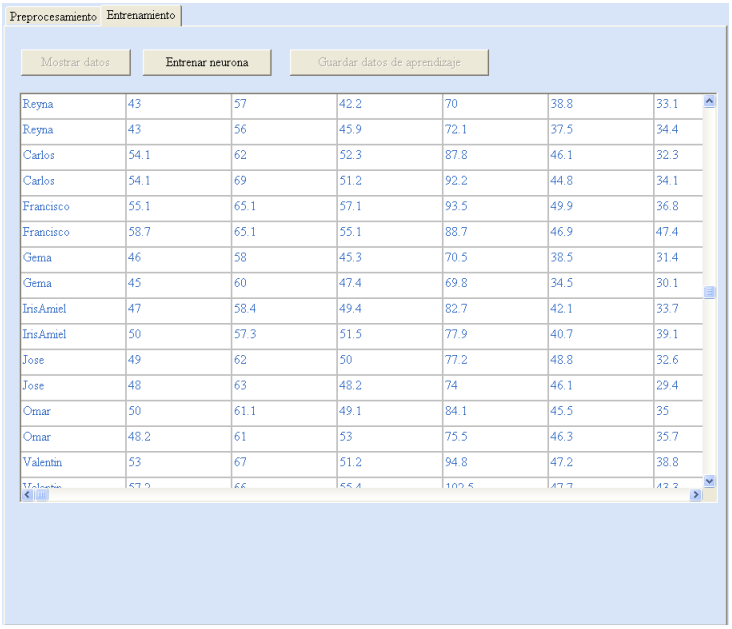


Figura 4.27. Ventana para abrir el archivo de datos.

Después de buscar el archivo, seleccionarlo y dar clic en abrir, los datos se mostrarán en un componente (Fig. 4.28) y se activa el botón `Entrenar neurona`, se da clic a dicho botón y después el sistema avisa cuando ya terminó de entrenar (Fig. 4.29) y se procede a guardar los pesos resultantes del entrenamiento (Fig. 4.30).



Reyna	43	57	42.2	70	38.8	33.1
Reyna	43	56	45.9	72.1	37.5	34.4
Carlos	54.1	62	52.3	87.8	46.1	32.3
Carlos	54.1	69	51.2	92.2	44.8	34.1
Francisco	55.1	65.1	57.1	93.5	49.9	36.8
Francisco	58.7	65.1	55.1	88.7	46.9	47.4
Gema	46	58	45.3	70.5	38.5	31.4
Gema	45	60	47.4	69.8	34.5	30.1
IrisAmiel	47	58.4	49.4	82.7	42.1	33.7
IrisAmiel	50	57.3	51.5	77.9	40.7	39.1
Jose	49	62	50	77.2	48.8	32.6
Jose	48	63	48.2	74	46.1	29.4
Omar	50	61.1	49.1	84.1	45.5	35
Omar	48.2	61	53	75.5	46.3	35.7
Valentin	53	67	51.2	94.8	47.2	38.8

Figura 4.28. Imagen que muestra los datos del archivo de características.

Figura 4.29. Imagen que muestra el término del entrenamiento.

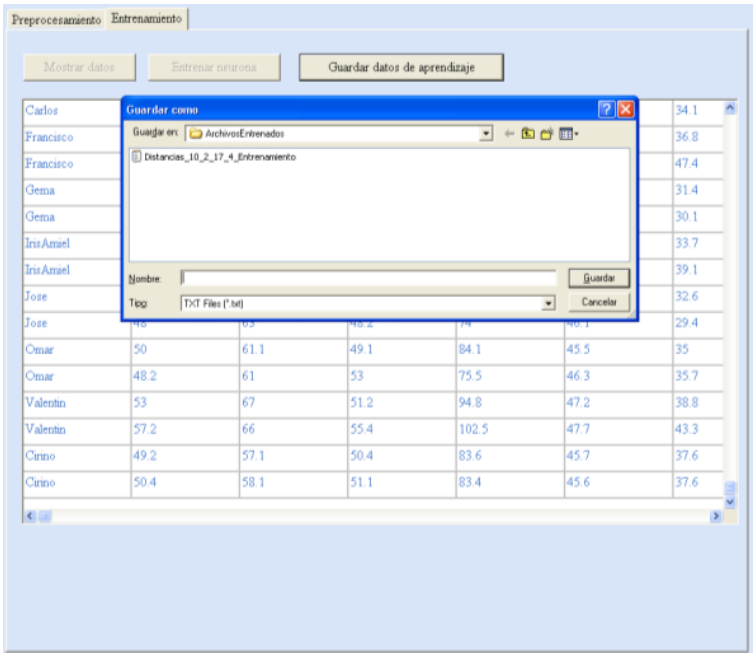


Figura 4.30. Ventana para guardar los pesos entrenados.

En la figura 4.31 se puede apreciar el contenido del archivo con los datos de los pesos obtenidos en el entrenamiento.

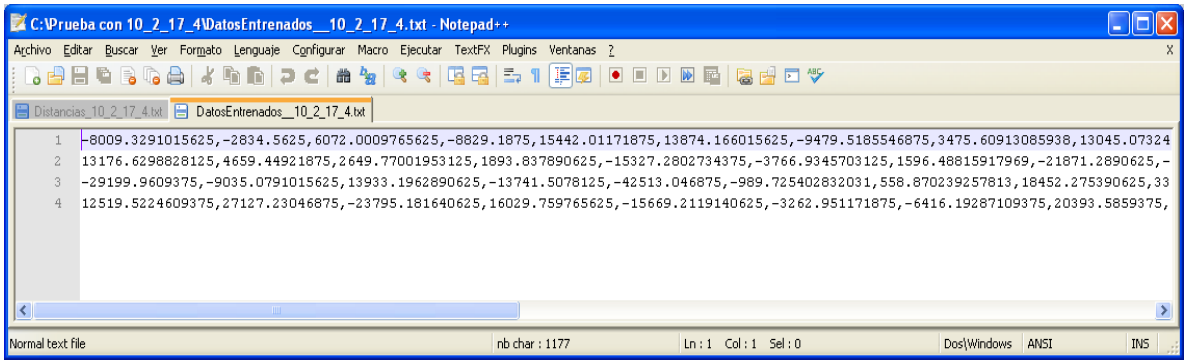


Figura 4.31. Archivo de pesos obtenidos en el entrenamiento.

Este archivo contiene los pesos y ganancias resultantes (Fig. 4.31), de las cuatro neuronas de la RNA, consta de 4 renglones con 18 números reales separados por comas (vease la estructura en el ejemplo de las figuras 4.18 y 4.19), de los cuales 17 corresponden a los pesos obtenidos en el entrenamiento y el último valor representa la ganancia (al final de cada renglón también existe una coma), este archivo sera utilizado para el reconocimiento de objetos en el módulo de reconocimiento.

Módulo de reconocimiento

Paso 1. En este último módulo se procede a reconocer una persona, se abre la pestaña `Reconocimiento` (Fig. 4.17) del menú `Imagen`, se abre la imagen a reconocer dando clic al botón `Abrir imagen` (Fig. 4.18a) del módulo reconocimiento y se activará el botón `Cargar datos de aprendizaje`, mostrará en el módulo dicha imagen con sus distancias como se ve en la figura 4.32.

Enseguida se abre el archivo de entrenamiento dando clic al botón `Cargar datos de aprendizaje`, dónde aparecerá una ventana para buscar el archivo (Fig. 4.27) de pesos entrenados y se activará el botón `Reconocer imagen` (Fig. 4.33). Finalmente, se da clic a dicho botón y el sistema muestra una ventana para buscar el archivo de características (Fig. 4.27), mismo que fue abierto en el módulo de entrenamiento, esto con la finalidad de que el sistema busque y encuentre la imagen resultante del reconocimiento (Fig. 4.34).

Figura 4.32. Imagen que muestra la imagen que se va a reconocer.

Figura 4.33. Imagen que muestra el botón `Reconocer imagen` activo.

Figura 4.34. Imagen que muestra la persona reconocida.

Como se puede observar en la figura 4.34, la persona en ambas imágenes es la misma, por lo tanto, se puede concluir que el reconocimiento de la persona se ha realizado con éxito, asi como un mensaje indicando el nombre de la imagen reconocida (para los casos de prueba reportados en este trabajo se contaba con los nombres de cada persona, en caso contrario mostraría el nombre genérico de la imagen). Cabe resaltar que la persona tiene diferentes expresiones en las imágenes.

En esta prueba se utilizaron las imágenes de la figura A.2 para el módulo de reconocimiento, de las cuales todas fueron consideradas en el proceso de entrenamiento, dando como resultado todas las imágenes reconocidas, en consecuencia se obtuvo un 100 % de certeza (Fig. B.16).

Caso de prueba 2

Esta prueba permitirá determinar si la RNA permite reconocer imágenes que no fueron consideradas en el módulo de entrenamiento, pero que sí pertenecen a las clases utilizadas en dicho módulo.

Para esta prueba se utilizaron las imágenes de las figuras A.1 y A.2, para el módulo de entrenamiento.

Lo que se espera como resultado de esta prueba es que se asocie cada una de las imágenes de la figura A.3 con las imágenes utilizadas en el módulo de entrenamiento.

En esta prueba se utilizó el archivo que muestra la figura 4.4 en el módulo de preprocesamiento y en la figura 4.26 se muestra la imagen del archivo de pesos y ganancias obtenido en el entrenamiento, es decir, en la prueba 2 se tienen 10 clases, dos objetos por clase que dan un total de 20 objetos para esta prueba. En el módulo de preprocesamiento se aplicaron las operaciones de escala de grises y el filtro de Sobel, posteriormente se calcularon las 17 distancias de cada objeto, para el módulo de entrenamiento se utiliza el archivo de distancias para obtener el archivo de pesos y ganancias, finalmente en el módulo de reconocimiento se tomaron las imágenes de la figura A.3. Cabe mencionar que los datos de estas 10 imágenes no fueron considerados en el entrenamiento, es decir, son otras imágenes de esas 10 clases, dando como resultado 9 imágenes reconocidas, con lo cual se obtiene un 90 % de certeza (Fig. B.16).

Caso de prueba 3

En esta prueba se determinará si la RNA no permite asociar imagenes que no fueron consideradas en el módulo de entrenamiento y que no pertenecen a las clases utilizadas en este módulo.

Para esta prueba se utilizaron las imágenes de las figuras A.1 y A.2, para el módulo de entrenamiento.

Lo que se espera como resultado de esta prueba es que no se clasifique ninguna de las imágenes de la figura A.5 con alguna de las clases que se utilizaron en el módulo de entrenamiento.

En esta prueba se utilizó el archivo de distancias de la figura 4.4 y el archivo de pesos y ganancias obtenido en el entrenamiento (Fig. 4.31). Sin embargo, en el módulo de reconocimiento se utilizaron las imágenes de la figura A.5, que no fueron utilizadas en el entrenamiento ni tampoco pertenecían a alguna clase, dando como resultado ninguna reconocida, por lo tanto, se tiene una certeza del 100 % debido a que no se les clasificó en ninguna clase, en caso contrario, esto se habría considerado como un error (Fig. B.16).

Caso de prueba 4

Esta prueba permitirá determinar si la RNA logra reconocer imágenes que fueron consideradas en el módulo de entrenamiento.

Para esta prueba se utilizaron las imágenes de las figuras A.1, A.2 y A.3, para el módulo de entrenamiento.

Lo que se espera como resultado de esta prueba es que se asocie cada una de las imágenes de la figura A.1 con las imágenes utilizadas en el módulo de entrenamiento.

En esta prueba se tiene un archivo de 30 rostros de 10 clases, a estas 30 imágenes se le aplicó la operación de escala de grises y el filtro Sobel en el módulo de preprocesamiento, se procedió a calcular sus 17 distancias teniendo al final un archivo con dicha información. Para el módulo de entrenamiento se utilizaron estos datos para obtener el archivo de pesos y ganancias resultantes en el módulo de entrenamiento. Finalmente, para el módulo de reconocimiento se utilizaron las imágenes de la figura A.1, dando como resultado 10 imágenes reconocidas, con la cual se obtuvo un 100 % de certeza para esta prueba (Fig. B.16).

Caso de prueba 5

Esta prueba permitirá determinar si la RNA puede reconocer imágenes que no fueron consideradas en el módulo de entrenamiento.

Para esta prueba se utilizaron las imágenes de las figuras A.1, A.2 y A.3, para el módulo de entrenamiento.

Lo que se espera como resultado de esta prueba es que se asocie cada una de las imágenes de la figura A.4 con las imágenes utilizadas en el módulo de entrenamiento.

Para esta prueba se utilizaron los mismos datos de la prueba 4 del módulo de reconocimiento y entrenamiento. Sin embargo, para el módulo de reconocimiento se utilizaron las imágenes de la figura A.4 que no se consideraron en el entrenamiento. De estas 10 imágenes sólo 7 fueron reconocidas por sus respectivas clases, en consecuencia se obtuvo un 70 % de certeza (Fig. B.16).

Caso de prueba 6

Esta prueba ayudará a determinar si la RNA permite reconocer imágenes que fueron consideradas en el módulo de entrenamiento.

Para esta prueba se utilizaron las imágenes de las figuras A.1, A.2, A.3 y A.4, para el módulo de entrenamiento.

Lo que se espera como resultado de esta prueba es que se asocie cada una de las imágenes de la figura A.2 con las imágenes utilizadas en el módulo de entrenamiento.

En esta prueba se tiene un archivo de 40 objetos, es decir, 4 objetos por cada una de las 10 clases. A estas 40 imágenes se les aplicaron las operaciones de escala de grises y el filtro Sobel. Posteriormente, en el mismo módulo de preprocesamiento se calcularon sus 17 distancias, con lo cual se obtuvo un archivo de distancias. Para el módulo de entrenamiento se utilizó el archivo de distancias para obtener el archivo de pesos y ganancias finales, del mismo modo, en el módulo de reconocimiento se utilizaron las imágenes de la figura A.2 para ser reconocidas, dando como resultado las 10 imágenes clasificadas correctamente con lo cual se obtuvo un 100 % de certeza (Fig. B.16).

Las pruebas mencionadas anteriormente tienen ciertas características gestuales, las pruebas 1, 2, 4, 5, y 6 son con las cuales el sistema explota toda su funcionalidad y se obtienen buenos resultados. Es importante recordar que el sistema inicialmente fue desarrollado con el objetivo de identificar rostros no invariantes a expresiones faciales, como lo describe el título de este proyecto de tesis, por tanto, son de estas 5 pruebas que se obtienen el porcentaje de efectividad y de error (Tabla XIX).

Tabla XIX. Resultado de las 5 pruebas principales.

Prueba	Imágenes en el módulo de Entrenamiento		Reconocimiento	Entrenadas *	% de certeza	% de error
	Número de personas diferentes	Número de rostros por persona				
1	10	2	10	10	100 %	0 %
2	10	2	10	0	90 %	10 %
4	10	3	10	10	100 %	0 %
5	10	3	10	0	70 %	30 %
6	10	4	10	10	100 %	0 %

* Indica el número de imágenes que fueron utilizadas para el módulo de reconocimiento y también en el módulo de preprocesamiento. El valor 0, indica que las imágenes no fueron utilizadas para el entrenamiento de la RNA.

El porcentaje de certeza promedio es de 92 %, el cual se obtuvo de sumar la columna 5 (% de certeza) y dividirla entre el total de pruebas realizadas, que son 5, y su diferencia refleja el porcentaje de error, que es de 8 %.

En la prueba 3 se utilizaron imágenes que no fueron consideradas en el módulo de entrenamiento y son imágenes de otras personas, por tanto, es preciso mencionarla y describir el resultado obtenido (Tabla XX), para conocer el comportamiento del sistema ante dicha situación.

Tabla XX. Prueba extra.

	Imágenes en el módulo de Entrenamiento					
Prueba	Número de personas diferentes	Número de rostros por persona	Reconocimiento	Entrenadas *	% de certeza	% de error
3	10	2	5	0	100 %	0 %

En caso de considerar el resultado del caso de prueba 3 como parte del porcentaje de certeza general para el sistema, éste aumentaría en 0.7 %, es decir, tendría un 92.7 %, con un error de 7.3 %.

CAPÍTULO 5. CONCLUSIONES Y TRABAJOS FUTUROS

Conclusiones

A lo largo de este proyecto de tesis denominado "Reconocimiento de rostros en un ambiente de iluminación controlado no invariante a expresiones faciales", se abordaron diversos capítulos como son: **Introducción**, **Antecedentes, Marco teórico, Desarrollo del tema,** por último **Conclusiones** y **trabajos futuros**. Dichos capítulos fueron descritos detalladamente en este trabajo de tesis cuyo principal objetivo consistió en desarrollar un software para el reconocimiento de rostros implementando una RNA Perceptron con 4 neuronas, con la finalidad de identificar rostros de personas, con la ayuda de conocimientos informáticos, inteligencia artificial y procesamiento digital de imágenes.

El software generado está estructurado en tres módulos principales:

- Preprocesamiento.
- Entrenamiento.
- Reconocimiento.

Estos tres módulos implican una interfaz gráfica de usuario, que se ha estructurado de tal forma que sea fácil de utilizar por el usuario. Dicho sistema está enfocado al reconocimiento de personas, para los casos de prueba se eligieron algunos alumnos de la Universidad del Mar.

El sistema se validó con 6 casos de prueba, con el objetivo de conocer el comportamiento y la efectividad del programa en cada uno de éstos. En los casos de prueba 1, 4 y 6 se utilizaron imágenes que fueron consideradas en el módulo de entrenamiento, con las cuales se obtuvo un porcentaje de certeza del 100 % en las tres pruebas, sin embargo, en los casos de prueba 2 y 5 se obtuvo un porcentaje de certeza del 90 % y 70 % respectivamente, y en el caso de la prueba 3 se utilizaron imágenes que no se consideraron en la fase de entrenamiento, es decir, eran otras personas, en la cual se obtuvo un 100 % de certeza, lo cual se determinó considerando que en esta prueba no se reconoció a ninguna persona, por lo tanto, el sistema no asoció ninguna de estas imágenes a las clases existentes, lo cual es correcto.

Si se considera el reconocimiento de rostros de personas previamente utilizadas en el entrenamiento, se puede determinar que el sistema tiene una efectividad de 90 % y un porcentaje de error de 10 %. Cabe mencionar que las imágenes utilizadas en los casos de prueba tienen ligeras variaciones gestuales (rostro serio, sonriendo, ojos cerrados y aleatorio) y el proyecto de tesis inicialmente propuesto se enfocaba al reconocimiento de rostros no invariantes a expresiones faciales, con lo cual se puede considerar que los resultados obtenidos por el sistema van más allá de las expectativas iniciales.

Cabe mencionar que las pruebas realizadas permiten concluir que si se quisiera trabajar con imágenes de rostros con gestos más pronunciados, sería necesario implementar otro tipo de RNA.

Trabajos futuros

Es preciso mencionar que este proyecto es el primer trabajo realizado en este recinto universitario enfocado al área de reconocimiento de rostros aplicando inteligencia artificial, RNA´s y procesamiento digital de imágenes, y se espera que este proyecto de investigación sea la base para la realización de nuevos proyectos enfocados a estas áreas.

Por lo tanto se sugieren algunas modificaciones que se le podrían hacer al sistema:

- Hacer dinámico el número de distancias calculadas sobre cada objeto.
- Implementar un módulo para que las distancias se obtengan de forma automática y la cantidad de éstas no sea estática.
- Implementar un módulo para agregar de manera automática los valores de salida esperados por la RNA.
- Manejar rutas relativas para la apertura y el almacenamiento de archivos, asi como para la instalación del sistema.
- Implementar un módulo automático de eliminación de ruido en las imágenes.
- Implementar un módulo donde se apliquen algoritmos automáticos que realcen los bordes y aplicar algoritmos automáticos para la detección de los bordes.
- Implementar algún algoritmo para adelgazar los bordes, ya que existen bordes que tienen más de un pixel de grosor, si se tiene un borde de ancho mínimo al momento de calcular alguna distancia o área, ésta será más exacta.
- Aumentar el número de neuronas, con lo cual se obtendría un aumento en el número de clases que puede clasificar la red.

A continuación se sugiere realizar una serie de actividades por módulos, para mejorar la funcionalidad del sistema:

Módulo de preprocesamiento

- Implementar una barra de desplazamiento al componente 2 del módulo de preprocesamiento (Fig. C.8) para poder cargar y visualizar una imagen de un tamaño más grande.
- Implementar un módulo para pedir al usuario el número de distancias que quiera medir al rostro de la imagen, con la finalidad de eliminar que sean fijas las 17 distancias que se calculan a cada imagen.
- Implementar un módulo para solicitar al usuario los datos necesarios para cambiar el número de neuronas a utilizar y con esto hacerlo dinámico.
- Implementar algún algoritmo que permita identificar y calcular las distancias de las imágenes de forma automática.

- Implementar una solución que permita cargar otro tipo de formato de imágenes y realice la conversión a BMP para trabajar internamente con este tipo de imágenes.

Módulo de entrenamiento

- Implementar una función para insertar nuevos registros en los datos que se muestran en el componente 1 del módulo de entrenamiento (Fig. C.23) para verlo reflejado en el archivo de distancias.
- Implementar una función para cargar datos de archivos (entrenamiento, distancias, reconocimiento) que se hubieran utilizado anteriormente en otros módulos, para que sea automático el acceso a los datos de los archivos necesarios en cada módulo.

Módulo de reconocimiento

- Implementar una barra de desplazamiento a los componentes 1 y 6 del módulo de reconocimiento (Fig. C.25) para poder visualizar y cargar una imagen de un tamaño más grande.

Enseguida se sugieren proyectos que se puede realizar a partir de este sistema:

- Desarrollo de sistemas para el control de acceso de personas a determinadas instalaciones.
- Desarrollar un sistema para el Instituto Federal Electoral (IFE), con la finalidad de saber si la persona que está tramitando su credencial cuenta con el trámite previo y así evitar fraudes.
- Desarrollar un sistema que ayude en supermercados donde den membresía a sus clientes, para evitar duplicados u otras actividades ilícitas.
- Desarrollar un sistema para el control vehicular estatal, con el objetivo de controlar la emisión de licencias de conducir, para evitar que los automovilistas evadan el pago de multas u otros cargos.

Actualmente, un buen sistema de seguridad y detección es aquel que tiene un módulo de adquisión de imágenes en tiempo real y distribuido, que el dispositivo físico sea de alta calidad para la captura de las imágenes, el procesamiento y algoritmo sean eficientes y eficaces, y que los resultados obtenidos se pueda mandar a cualquier dispositivo de nueva tecnología, por ejemplo: celulares, PDA´s, Palm´s, etc. Esto teóricamente hace un buen sistema de seguridad y este proyecto de tesis puede ser el inicio de la formación de un sistema en esta área y con estas características.

ANEXO A. IMÁGENES ORIGINALES PARA LAS PRUEBAS

En este apartado se muestran las imágenes en original utilizadas en las pruebas desarrolladas en este proyecto.

En la figura A.1 se muestran las imágenes con gesto serio, en la figura A.2 se presentan las imágenes con gesto sonriente, en la figura A.3 se pueden observar las imágenes con ojos cerrados, en la figura A.4 se muestran las imágenes con gesto aleatorio y en la figura A.5 se presentan imágenes que no fueron consideradas en el módulo de entrenamiento y que no pertenecen a ninguna clase de la base de conocimiento.

Abimael1.bmp

Reyna1.bmp

Carlos1.bmp

Francisco1.bmp

Gemma1.bmp

IrisAmiel1.bmp

Jose1.bmp

Omar1.bmp

Valentin1.bmp

Cirino1.bmp

Figura A.1. Imágenes de las 10 personas utilizadas en las pruebas con gesto serio.


Abimael2.bmp


Reyna2.bmp


Carlos2.bmp


Francisco2.bmp


Gemma2.bmp


IrisAmiel2.bmp


Jose2.bmp


Omar2.bmp


Valentin2.bmp


Cirino2.bmp

Figura A.2. Imágenes de las 10 personas utilizadas en las pruebas con gesto sonriente.


Abimael3.bmp


Reyna3.bmp


Carlos3.bmp


Francisco3.bmp


Gemma3.bmp


IrisAmiel3.bmp


Jose3.bmp


Omar3.bmp


Valentin3.bmp


Cirino3.bmp

Figura A.3. Imágenes de las 10 personas utilizadas en las pruebas con ojos cerrados.

Abimael4.bmp

Reyna4.bmp

Carlos4.bmp

Francisco4.bmp

Gemma4.bmp

IrisAmiel4.bmp

Jose4.bmp

Omar4.bmp

Valentin4.bmp

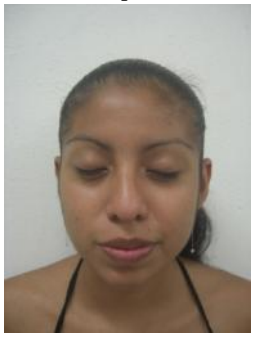
Cirino4.bmp

Figura A.4. Imágenes de las 10 personas utilizadas en las pruebas con gesto aleatorio.

Armando1.bmp

JuanJose1.bmp

Ricardo1.bmp



Victoria1.bmp

Ochoa1.bmp

Figura A.5. Imágenes de 5 personas utilizadas para una prueba extra.

ANEXO B. IMÁGENES PARA EL PREPROCESAMIENTO Y RECONOCIMIENTO

B.1. Preprocesamiento de las imágenes utilizadas en las pruebas

En las figuras B.1 a la B.5 se muestran imágenes en el siguiente orden: en la columna uno las imágenes en escala de grises, en la columna dos el resultado de aplicar el filtro de Sobel y en la tercera columna se muestra el resultado de calcular las 17 características principales de la imagen; este proceso se realiza en el módulo de preprocesamiento.

La figura 4.4 muestra el archivo de distancias que contiene todos los objetos de la prueba 1 y la figura 4.5 muestra el archivo de pesos obtenidos en la misma prueba.

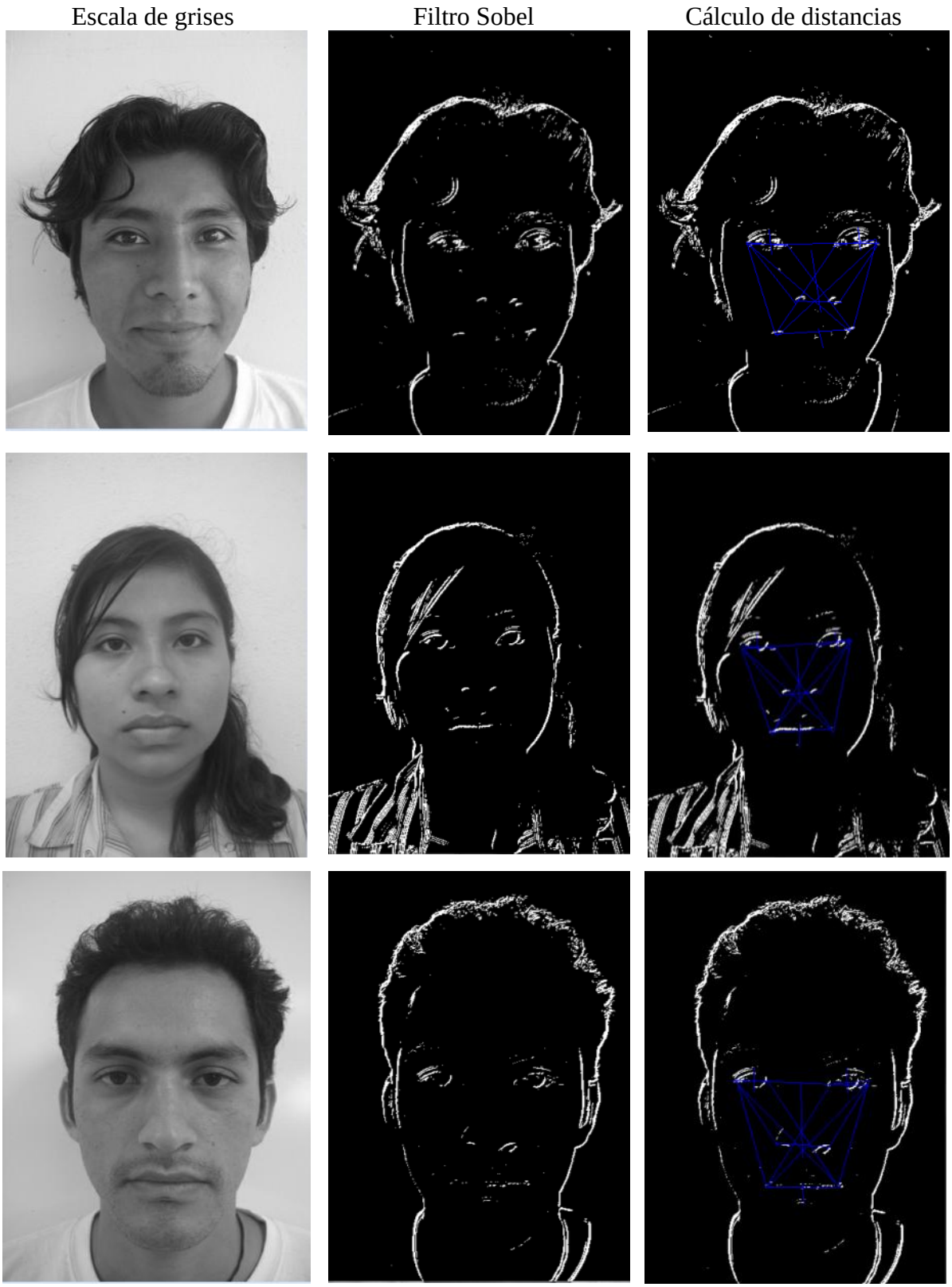


Figura B.1. Preprocesamiento de las imágenes de la figura A.1.

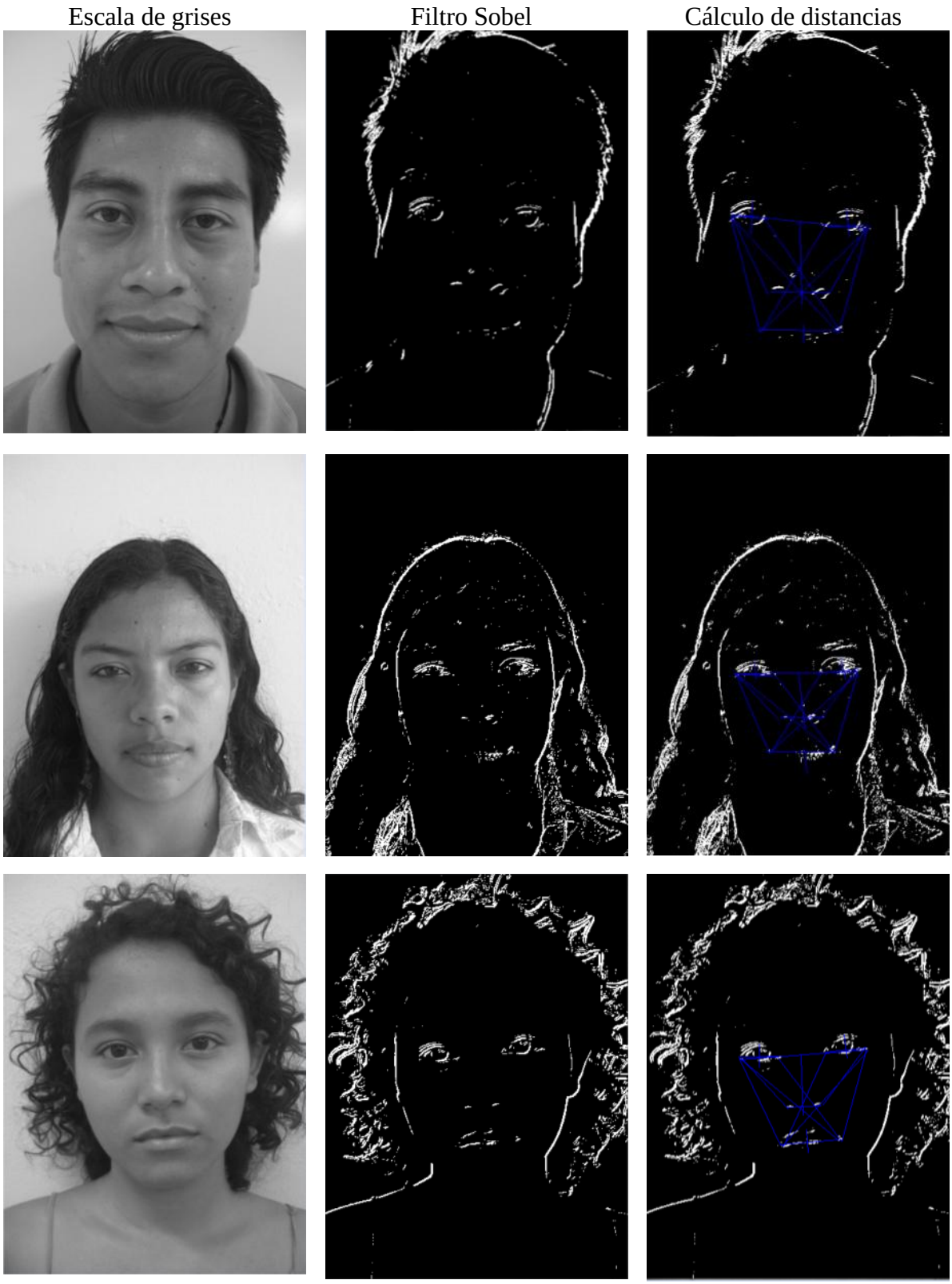


Figura B.1. Preprocesamiento de las imágenes de la figura A.1 (continuación).

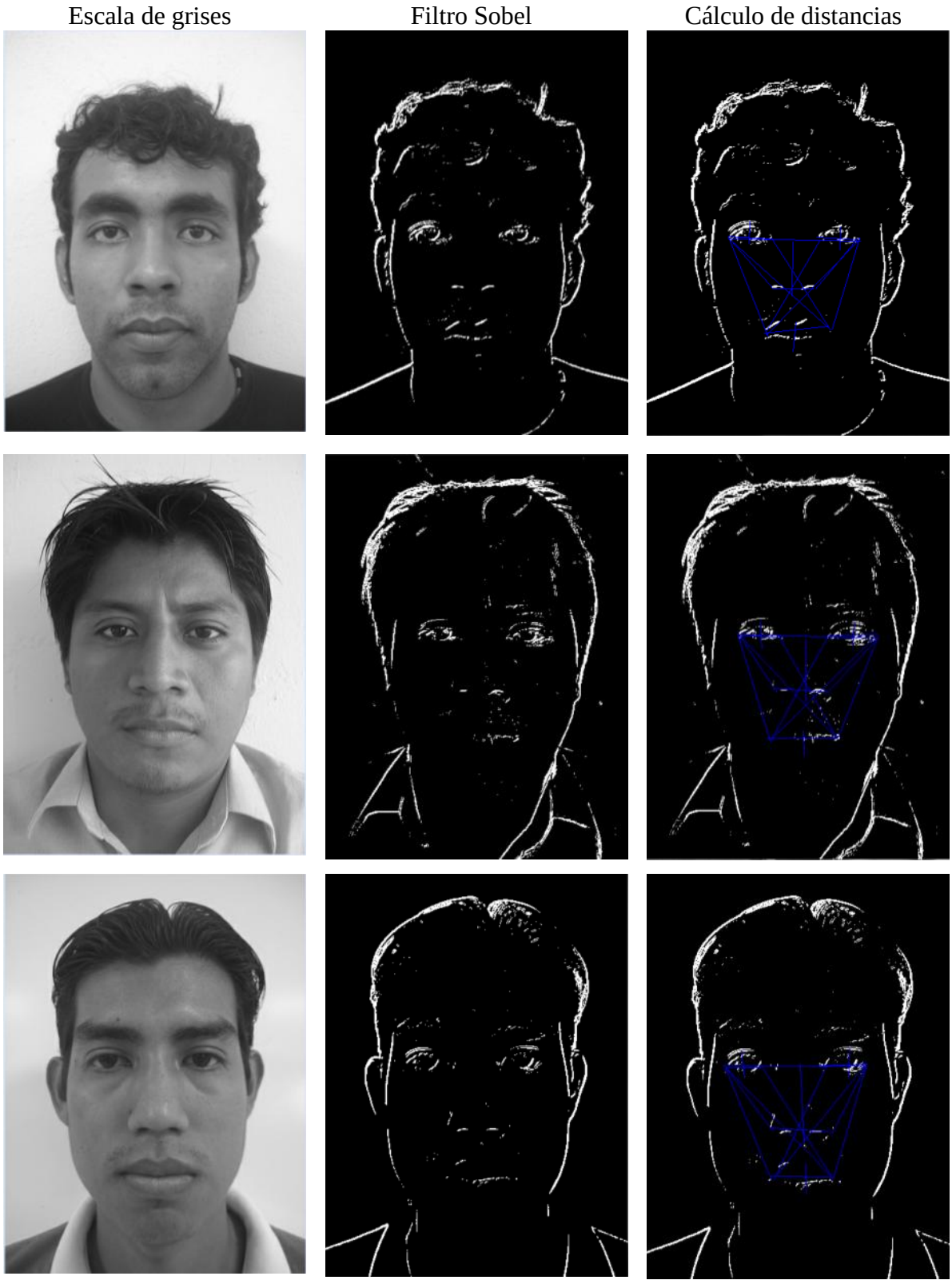


Figura B.1. Preprocesamiento de las imágenes de la figura A.1 (continuación).

Figura B.1. Preprocesamiento de las imágenes de la figura A.1 (continuación).

Figura B.2. Preprocesamiento de las imágenes de la figura A.2.

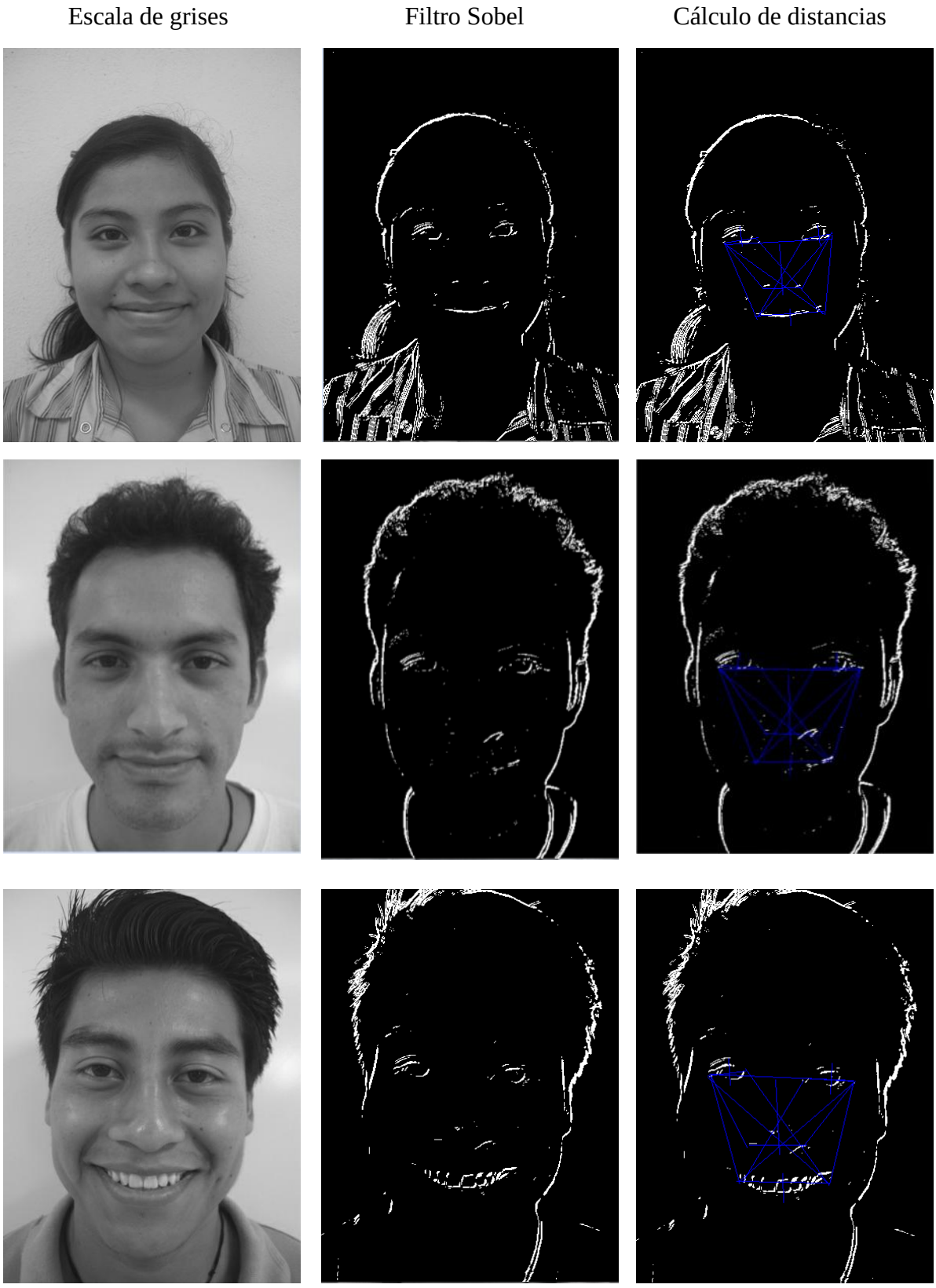


Figura B.2. Preprocesamiento de las imágenes de la figura A.2 (continuación).

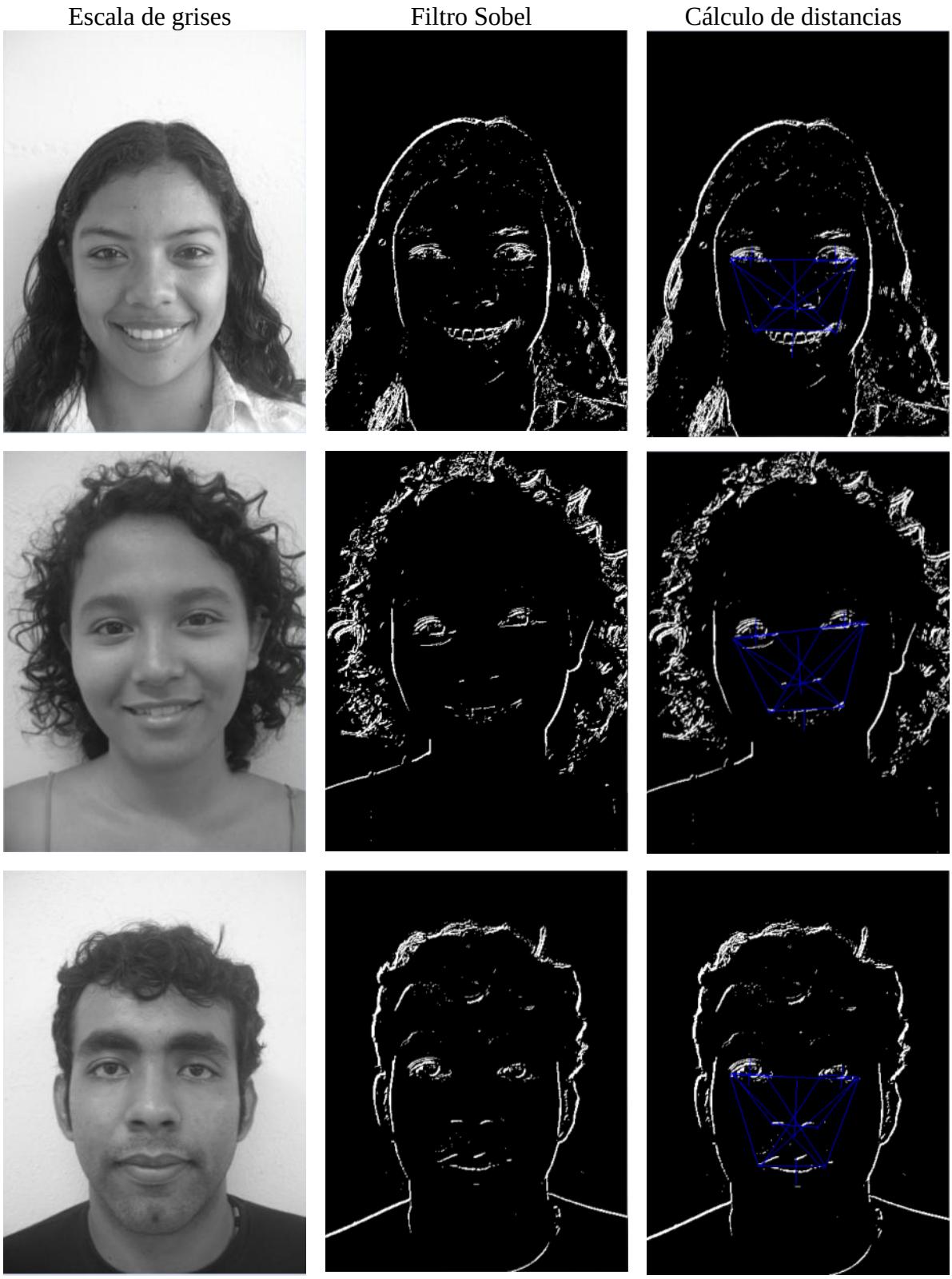


Figura B.2. Preprocesamiento de las imágenes de la figura A.2 (continuación).

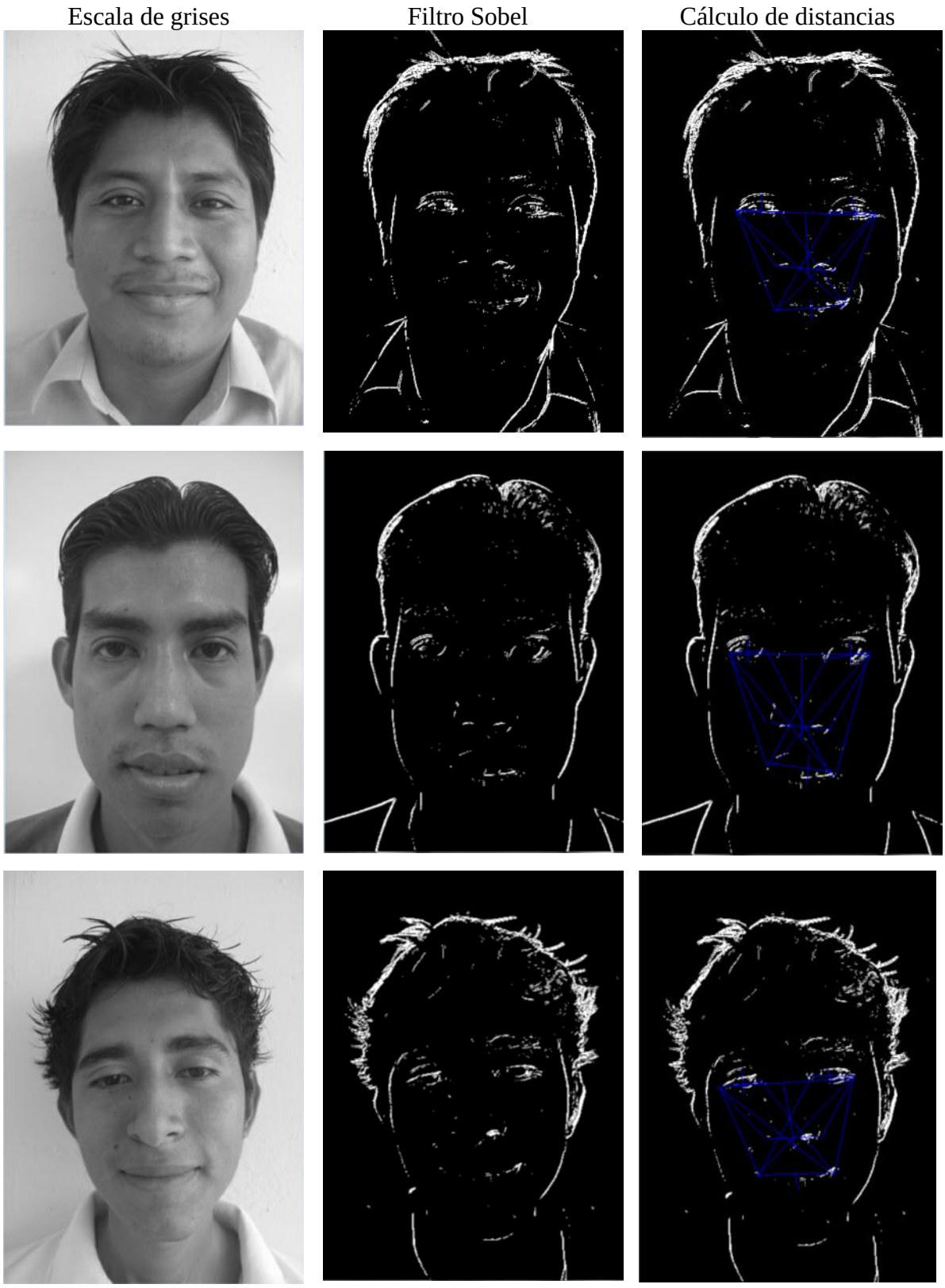


Figura B.2. Preprocesamiento de las imágenes de la figura A.2 (continuación).

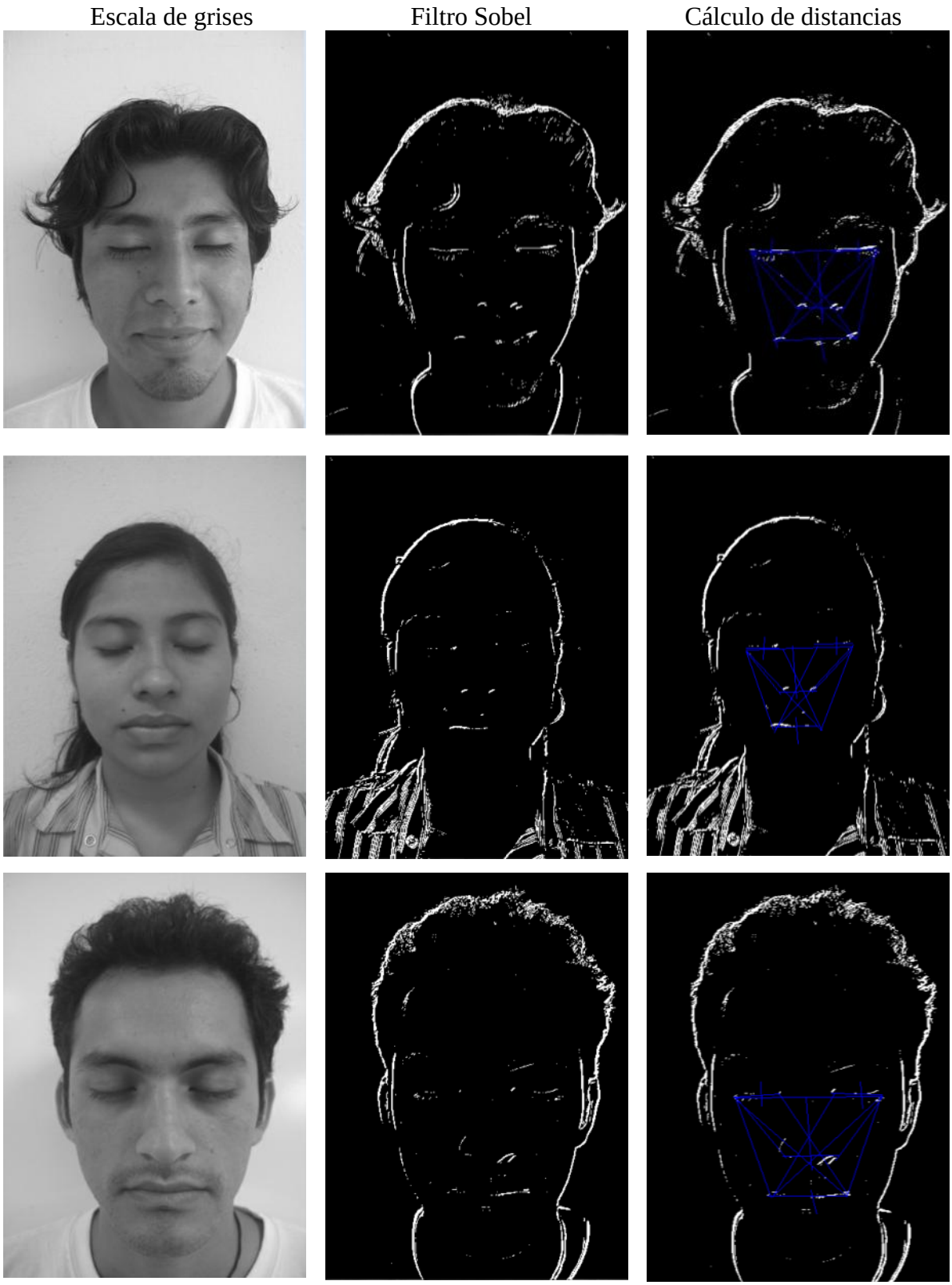


Figura B.3. Preprocesamiento de las imágenes de la figura A.3.

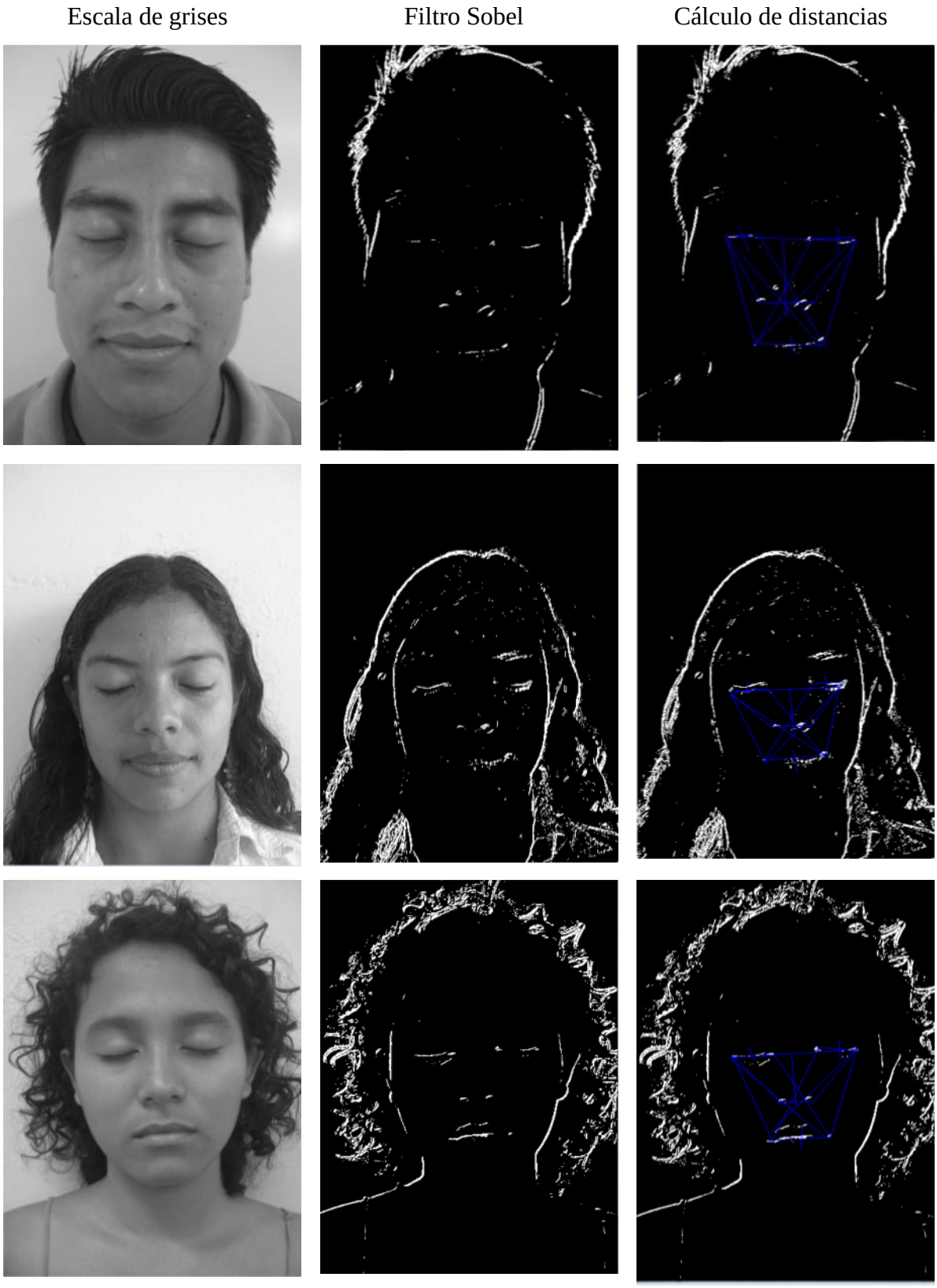


Figura B.3. Preprocesamiento de las imágenes de la figura A.3 (continuación).

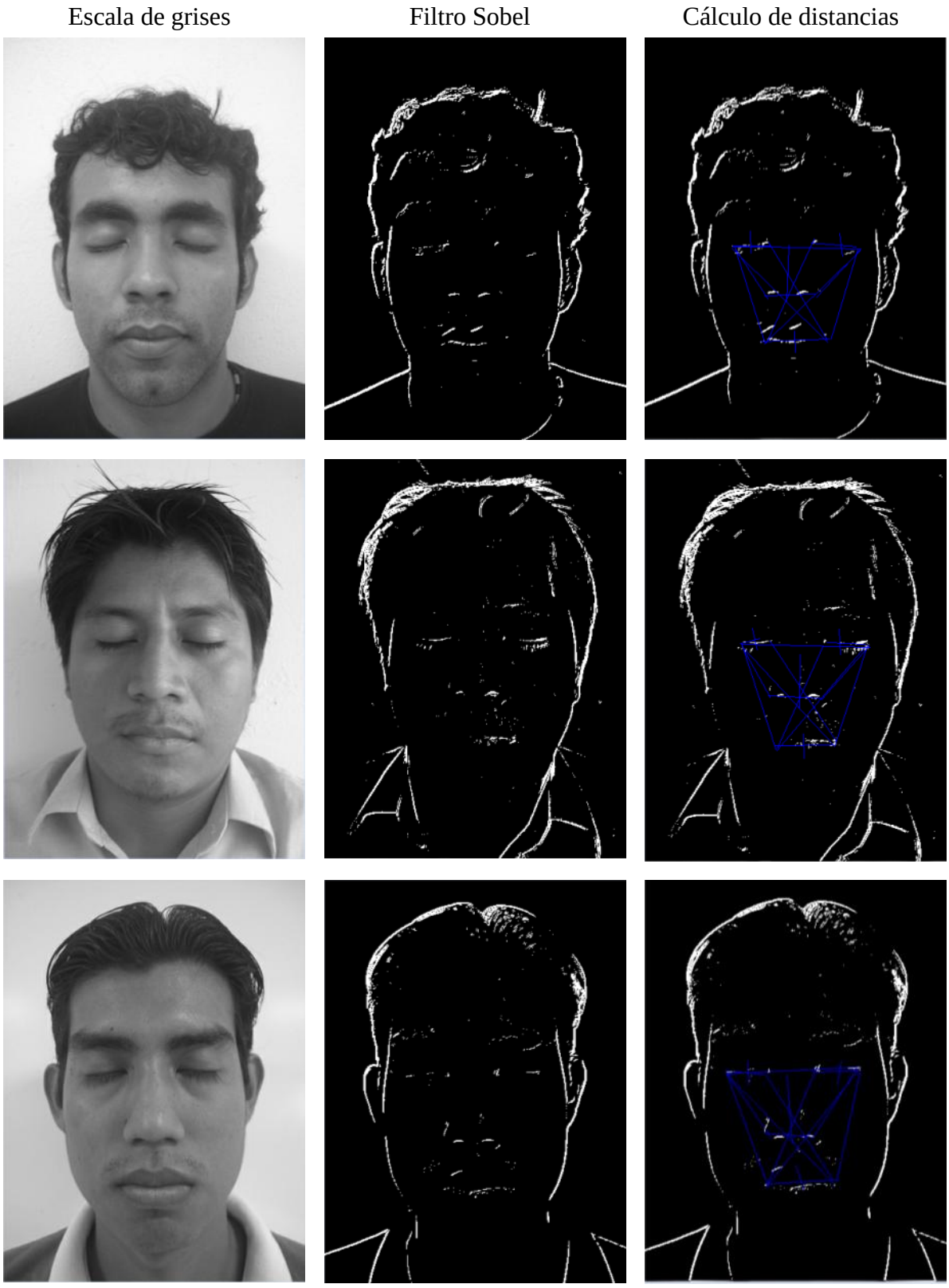


Figura B.3. Preprocesamiento de las imágenes de la figura A.3 (continuación).

Figura B.3. Preprocesamiento de las imágenes de la figura A.3 (continuación).

Figura B.4. Preprocesamiento de las imágenes de la figura A.4.

Escala de grises | Filtro Sobel | Cálculo de distancias

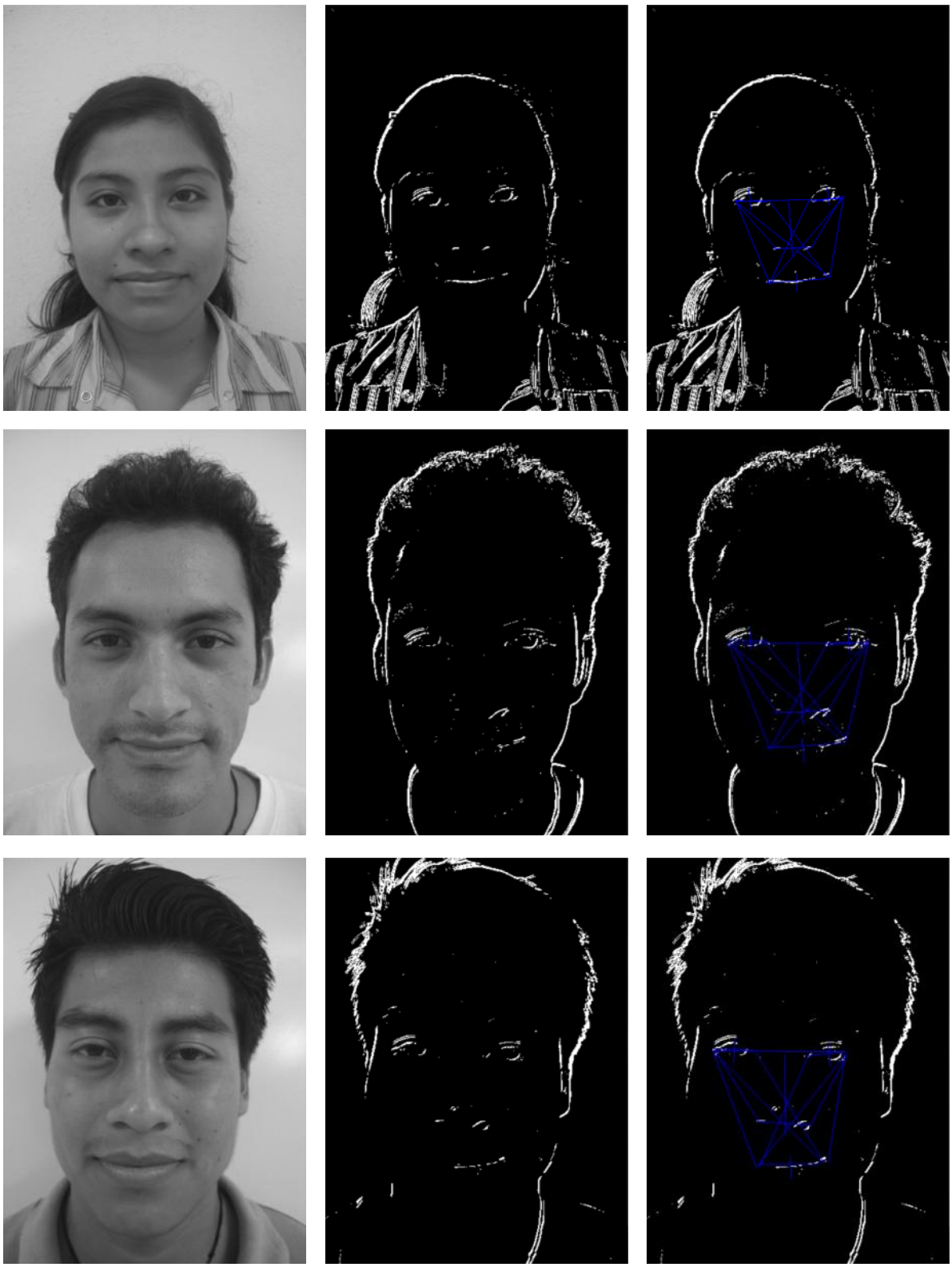


Figura B.4. Preprocesamiento de las imágenes de la figura A.4 (continuación).

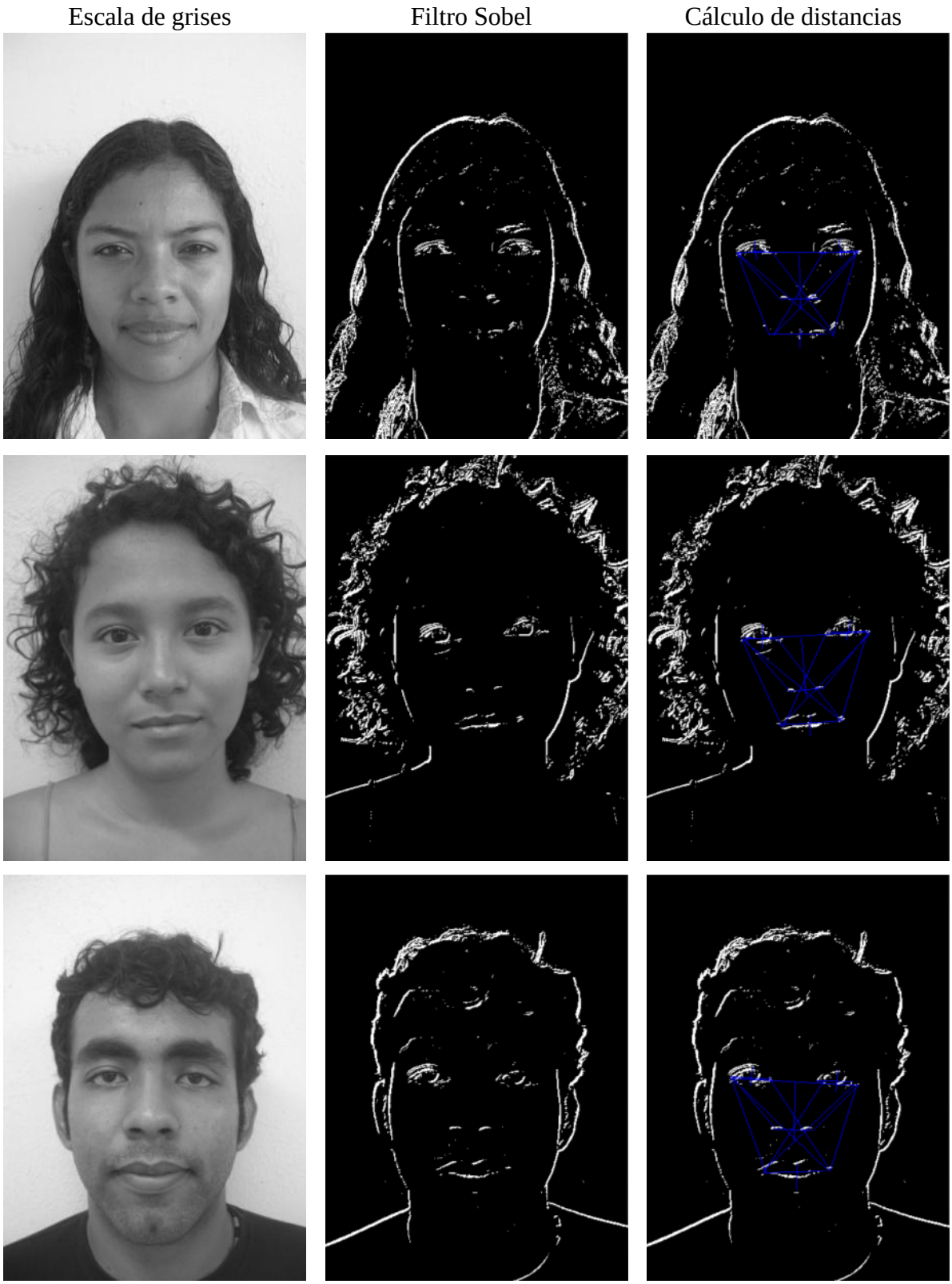


Figura B.4. Preprocesamiento de las imágenes de la figura A.4 (continuación).

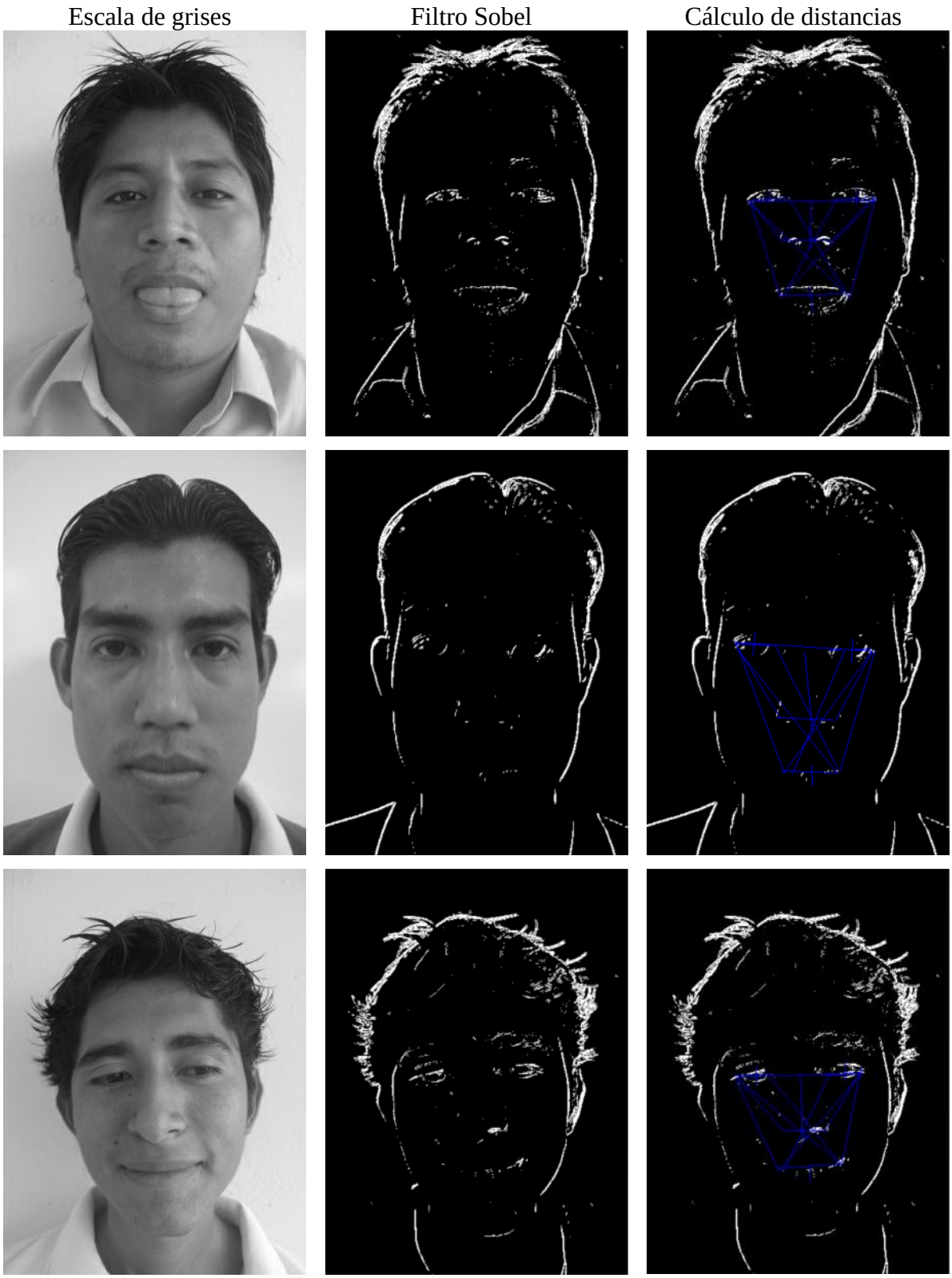


Figura B.4. Preprocesamiento de las imágenes de la figura A.4 (continuación).

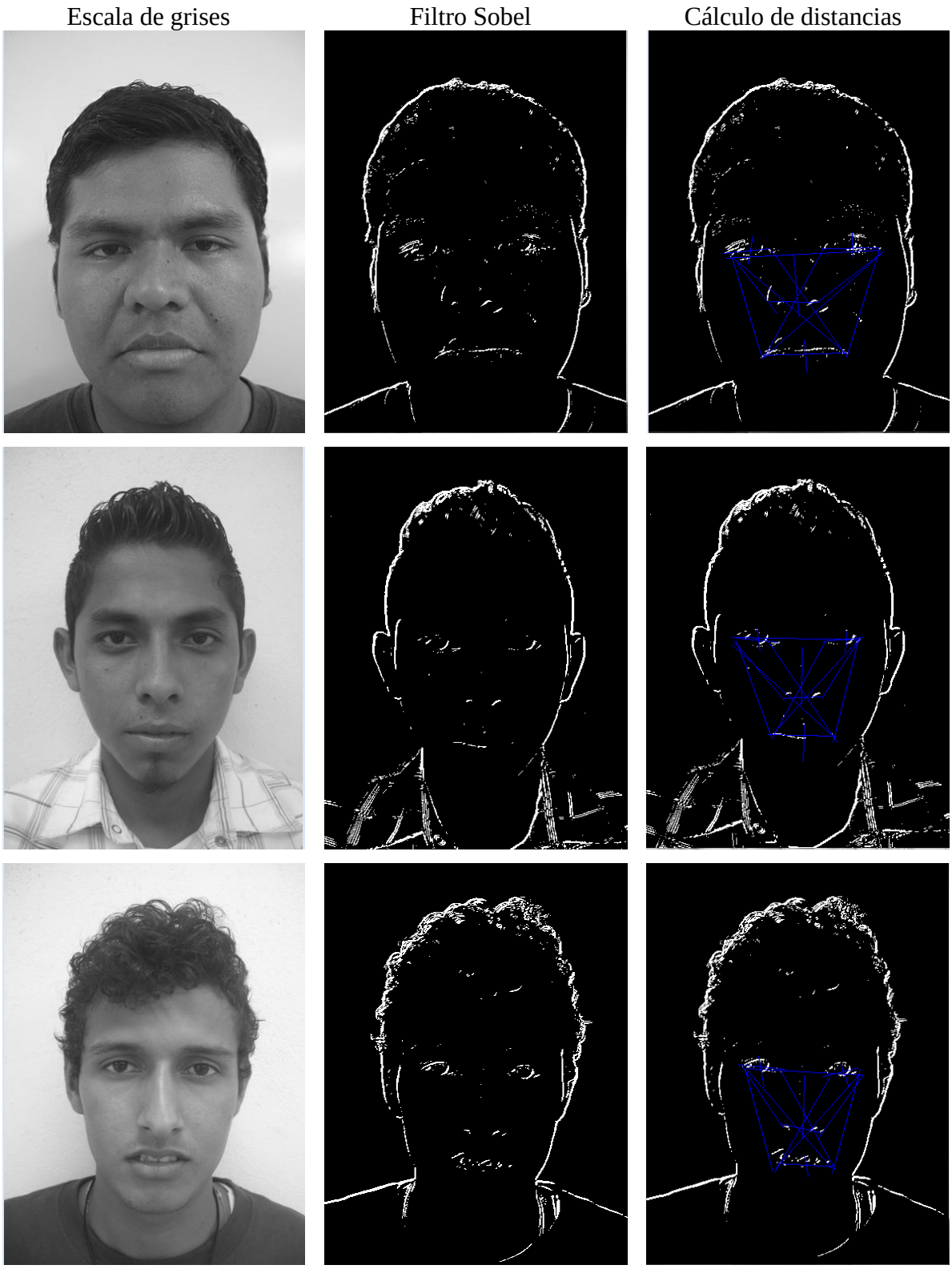


Figura B.5. Preprocesamiento de las imágenes de la figura A.5.

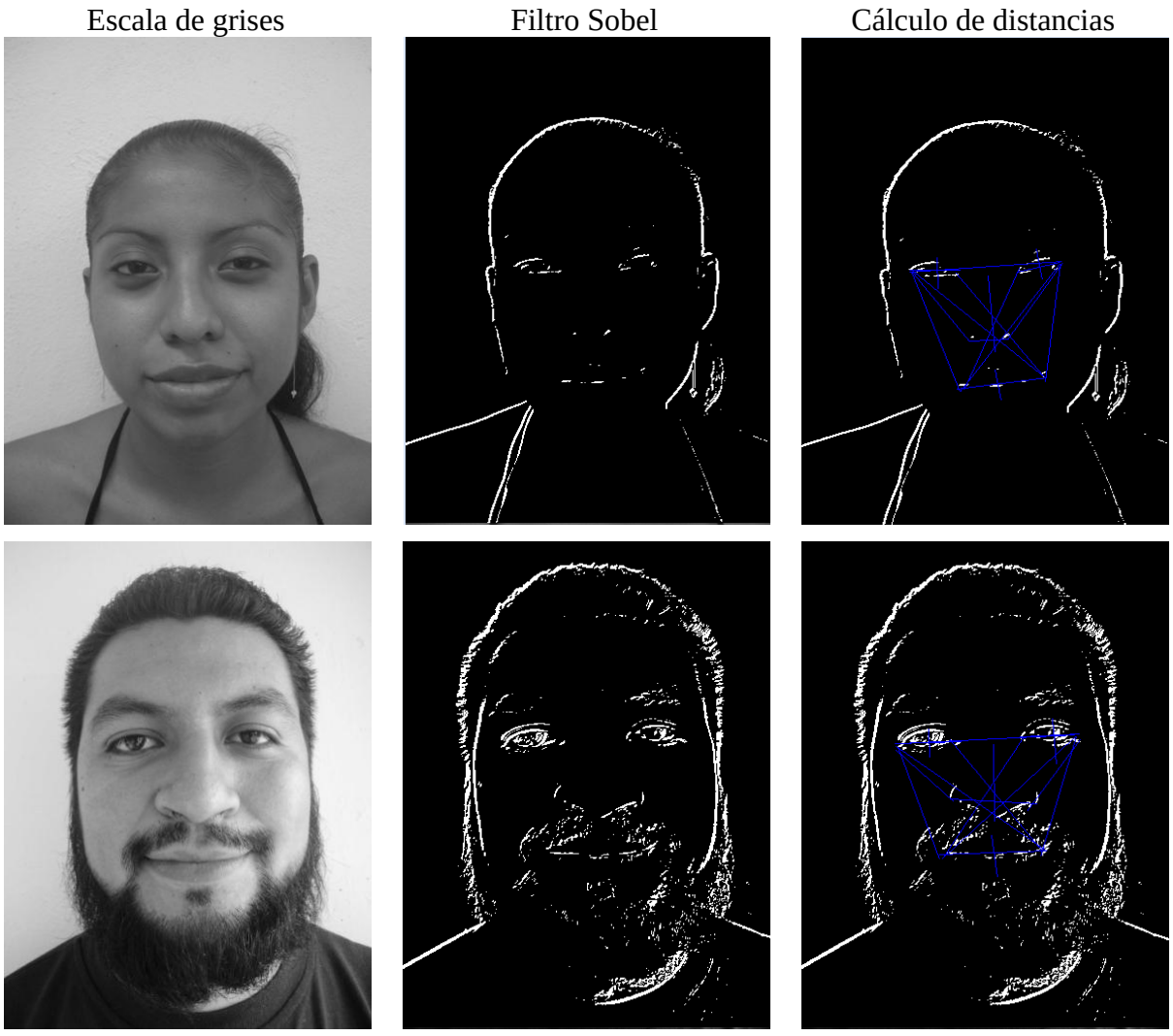


Figura B.5. Preprocesamiento de las imágenes de la figura A.5 (continuación).

B.2. Imágenes utilizadas en el módulo de reconocimiento en las pruebas

Las imágenes originales utilizadas en el módulo de reconocimiento de las pruebas 1, 2, 3, 4, 5 y 6 fueron las figuras A.2, A.3, A.5, A.1, A.4 y A.2 respectivamente, dichos resultados se observan en ese mismo orden, en las figuras B.6, B.7, B.8, B.9, B.10 y B.11. Cabe mencionar que debajo de cada objeto a reconocer se muestran sus distancias y el nombre de la clase a la que pertenece. El nombre se utiliza exclusivamente como un identificador para la imagen, y para ser utilizado en el mensaje de identificación “Se reconoció a ”, tal como se muestra en las imágenes de la figura B.6.

Abimael,50.2,60,44.6,78.3,46.3,28.4,90,37.1,47,78.2,60.2,63.6,74.1,26,77.3,118.9,107.4,

Reyna,44,56.1,43.3,69.8,37.6,32.4,91,33.7,32.8,68.4,53,56.7,67,26,71.1,103.3,102.1,

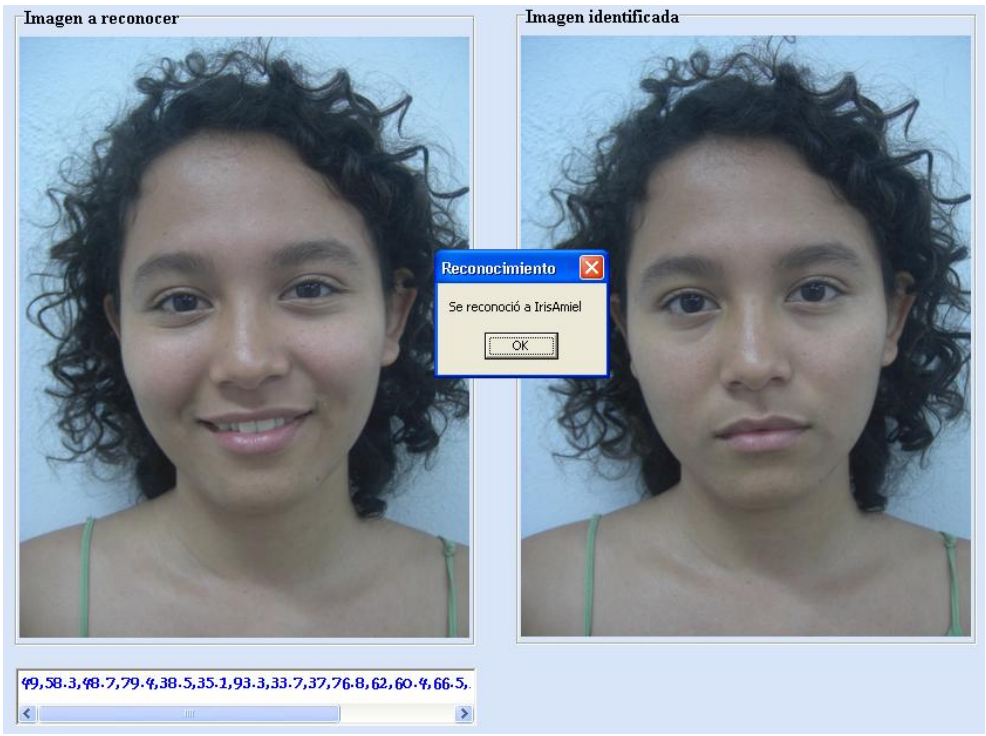
Figura B.6. Reconocimiento de las imágenes de la figura A.2 utilizadas en la prueba 1.

Carlos,51.1,70,54.1,93.2,44.8,32.2,103,30.1,42.7,95.7,78,77.4,78,41,97.1,135.3,131.2,

Francisco,56.6,65.1,57,89.7,45.5,43.4,132.1,45.2,42.2,90,76.4,76.8,95.1,44.2,96,137.6,130.2,

Figura B.6. Reconocimiento de las imágenes de la figura A.2 utilizadas en la prueba 1 (continuación).

Gemma,46,61,45,71.2,32.2,34.1,99,38.5,33.6,71.1,55,54,63,27,67.1,103.6,100.4,



IrisAmiel,49,58.3,48.7,79.4,38.5,35.1,93.3,33.7,37,76.8,62,60.4,66.5,25.2,74.1,106.4,112.7
,

Figura B.6. Reconocimiento de las imágenes de la figura A.2 utilizadas en la prueba 1 (continuación).

Jose,50,63,46,73.8,48.8,33,92,34.5,48.3,71.8,58.2,60.1,71.1,28.2,71,107.8,104.1,

Omar,49.1,62,52,76,46.6,40.8,104,36.7,47,74.1,60.8,62.5,80.1,30,74.1,108.9,112.8,

Figura B.6. Reconocimiento de las imágenes de la figura A.2 utilizadas en la prueba 1 (continuación).

Valentin,56,67,52.2,99.7,48.8,42.7,100.1,39.4,46.5,100.8,85.1,84.3,79,29,95,131.6,131.1,

Cirino,49.5,58.1,51.4,84.6,47.4,40.3,97.1,34.2,48,74.8,66.1,68.7,71,24,87.1,112.4,121.5,

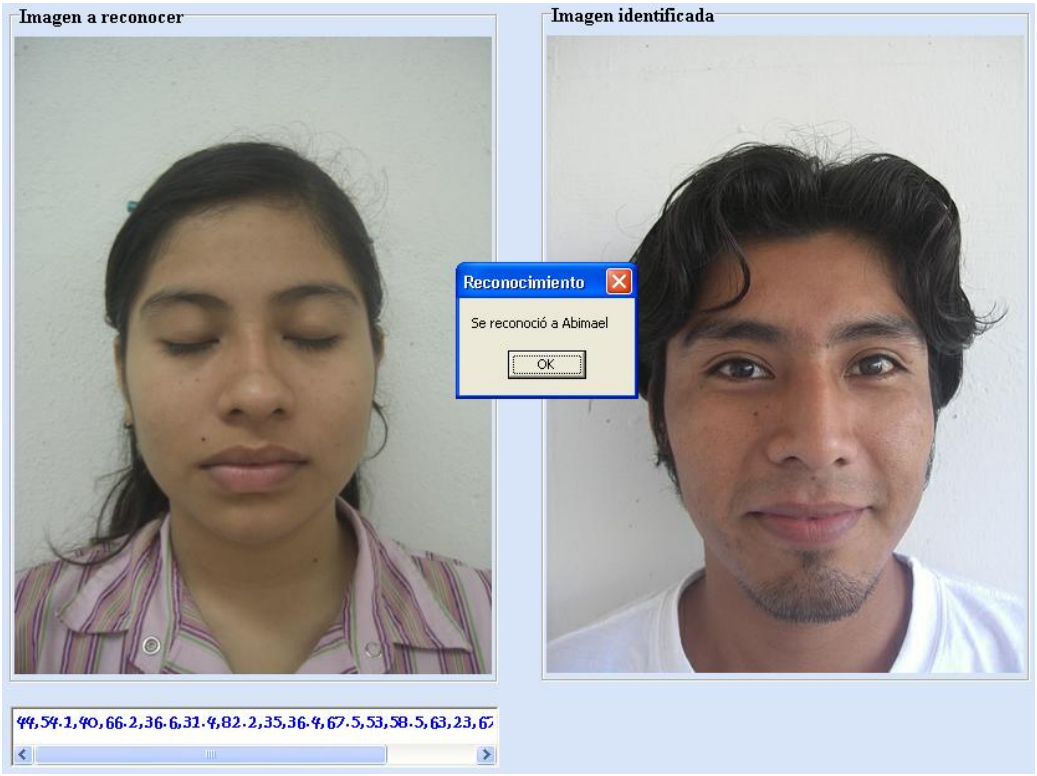
Figura B.6. Reconocimiento de las imágenes de la figura A.2 utilizadas en la prueba 1 (continuación).

Como se observa en la figura B.6 las personas reconocidas fueron Abimael, Reyna, Carlos, Francisco, Gemma, Iris Amiel, José, Omar, Valentín y Cirino, es decir, se reconocieron las imágenes de la figura A.2, dando como resultado un 100 % de certeza para esta prueba.

La figura B.7 muestra el resultado del módulo de reconocimiento obtenido en la prueba 2.

Abimael,52,57,46.1,76.1,42.9,24.3,88.3,30.4,43,78.8,63.3,64.8,73,35.1,77.4,116.9,105.1,

Figura B.7. Reconocimiento de las imágenes de la figura A.3 utilizadas en la prueba 2.



Reyna,44,54.1,40,66.2,36.6,31.4,82.2,35,36.4,67.5,53,58.5,63,23,67.3,100.1,91.4,

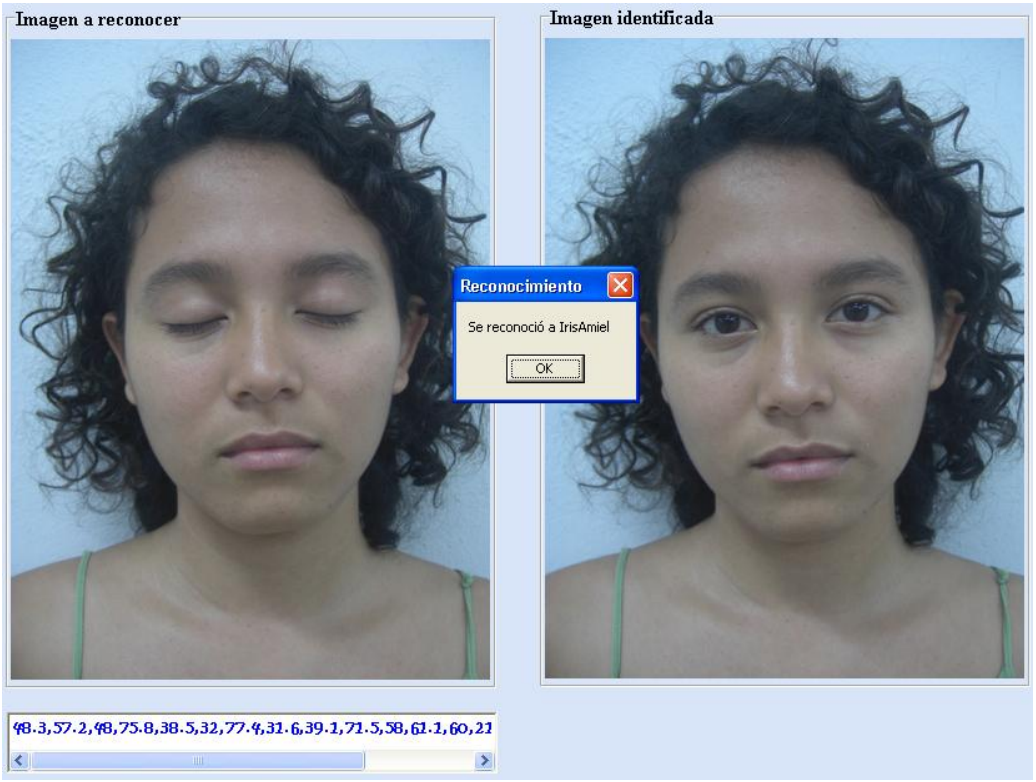
Carlos,54,69,53,87.7,43.9,32.9,100,29.2,38.6,96.3,78,76.2,74.1,42,94.2,132.9,123.2,

Figura B.7. Reconocimiento de las imágenes de la figura A.3 utilizadas en la prueba 2. (continuación).

Francisco,57.2,64,57.1,88.2,49.8,39,105.1,34.2,44.8,91.3,76.1,77.5,86.1,37,92,131.6,125.2,

Gemma,47.2,58,41,66.6,34.4,27.5,81.1,30.7,37.7,67.2,53,53,61,26.2,64.1,99.4,93.3,

Figura B.7. Reconocimiento de las imágenes de la figura A.3 utilizadas en la prueba 2. (continuación).



IrisAmiel,48.3,57.2,48,75.8,38.5,32,77.4,31.6,39.1,71.5,58,61.1,60,21,70.1,101.2,102.5,

Jose,49.4,63,49.2,75.4,47.4,34.9,86.1,28.8,47,72.2,64.3,63,71,30,74,104.2,107.2,

Figura B.7. Reconocimiento de las imágenes de la figura A.3 utilizadas en la prueba 2. (continuación).

Omar,47,58,51,73.2,45.8,44.4,90,36.4,49.6,63.4,52.6,59.7,75.2,25.1,60,95,101.2,

Valentin,56,67,52.2,99.7,48.8,42.7,100.1,39.4,46.5,100.8,85.1,84.3,79,29,95,131.6,131.1,

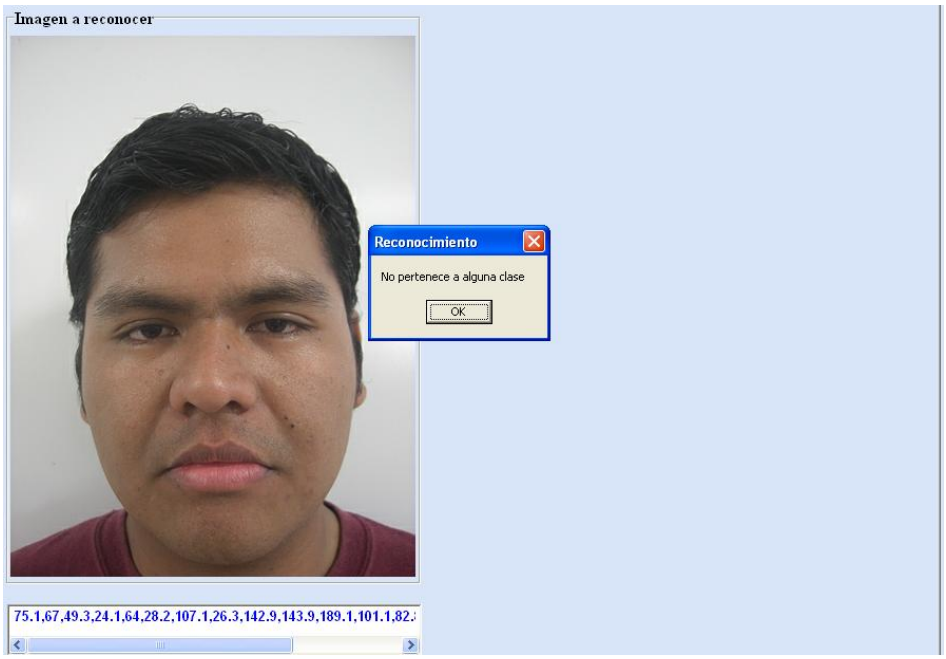
Figura B.7. Reconocimiento de las imágenes de la figura A.3 utilizadas en la prueba 2. (continuación).

Cirino,47.5,60.3,48,83.2,44.7,38.6,93.1,34,47.9,73.8,63,72.4,67.1,26,85.1,109.7,116.4,

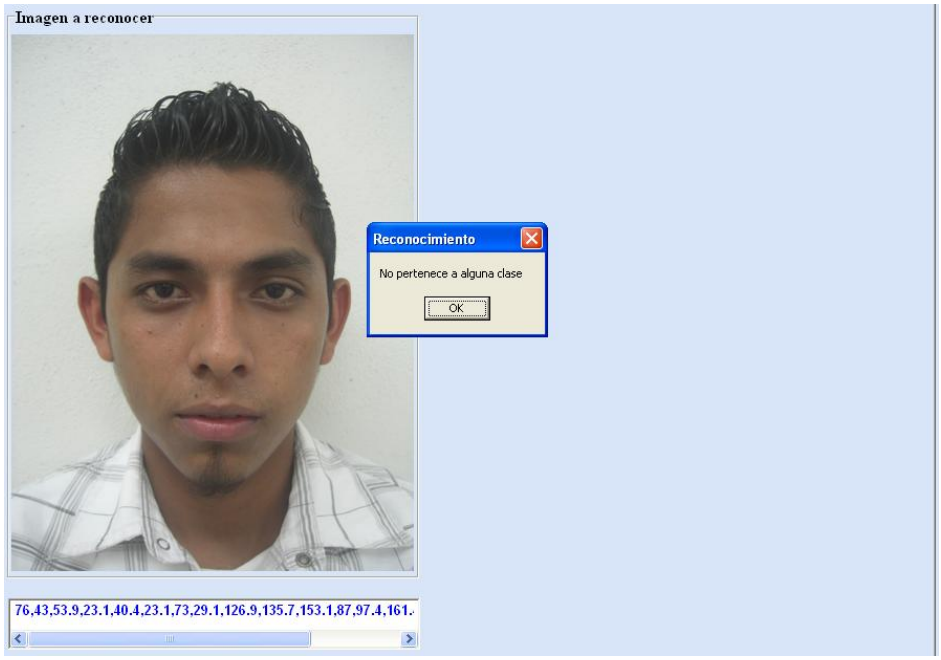
Figura B.7. Reconocimiento de las imágenes de la figura A.3 utilizadas en la prueba 2. (continuación).

Como se observa en la figura B.7 las personas reconocidas fueron Abimael, Carlos, Francisco, Gemma, Iris Amiel, José, Omar, Valentín y Cirino, es decir, se reconocieron imágenes de la figura A.3, dando como resultado un 90 % de certeza en dicha prueba.

La figura B.8 muestra el resultado del módulo de reconocimiento obtenido en la prueba 3.



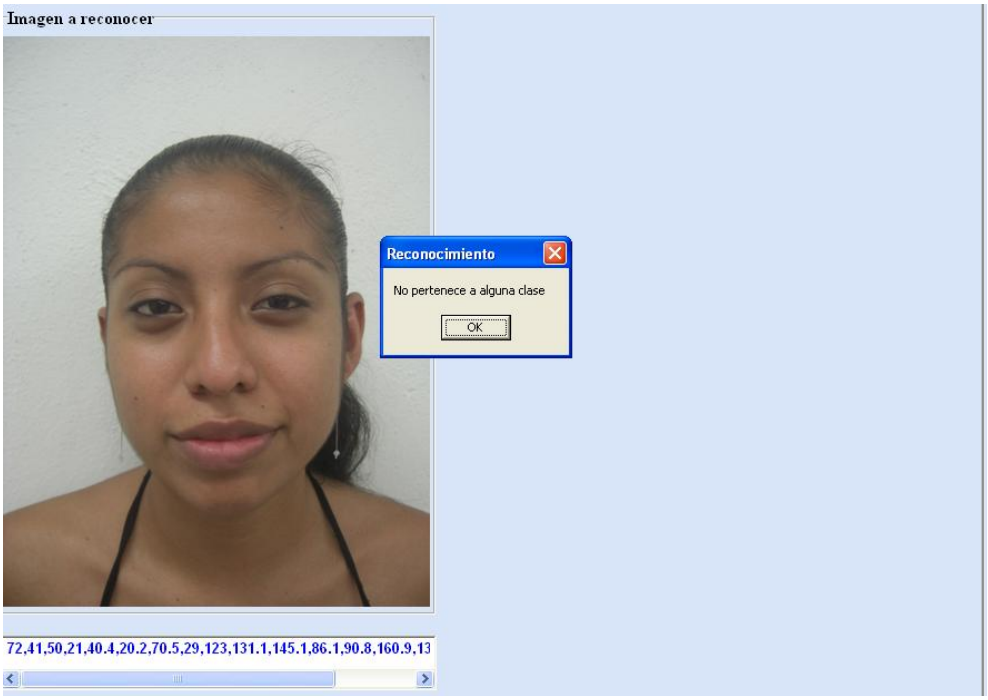
Armando,75.1,67,49.3,24.1,64,28.2,107.1,26.3,142.9,143.9,189.1,101.1,82.8,200.8,158.5,196.5,170.8,



JuanJose,76,43,53.9,23.1,40.4,23.1,73,29.1,126.9,135.7,153.1,87,97.4,161.4,133.8,171.1,135.8,

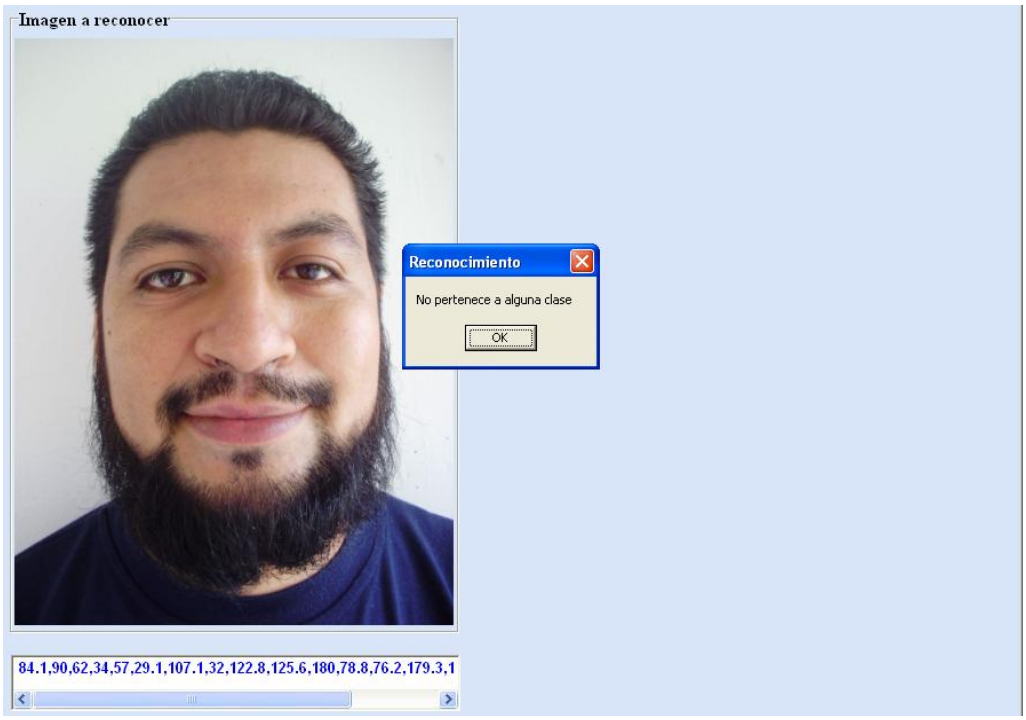
Figura B.8. Reconocimiento de las imágenes de la figura A.5 utilizadas en la prueba 3.

Ricardo,77,46,47.5,21.6,47,23,75.2,29,121.8,124,145.5,88,91.8,154.9,128.6,161,133.3,



Victoria,72,41,50,21,40.4,20.2,70.5,29,123,131.1,145.1,86.1,90.8,160.9,136.1,167.8,135.1,

Figura B.8. Reconocimiento de las imágenes de la figura A.5 utilizadas en la prueba 3 (continuación).



Ochoa,84.1,90,62,34,57,29.1,107.1,32,122.8,125.6,180,78.8,76.2,179.3,141.2,182.6,143.4,

Figura B.8. Reconocimiento de las imágenes de la figura A.5 utilizadas en la prueba 3 (continuación).

Como se observa en la figura B.8 la RNA no asoció ninguna imagen de la figura A.5 por lo que el resultado del reconocimiento es correcto.

La figura B.9 muestra el archivo final que contiene todas las distancias de las imágenes, las cuales se obtuvieron a partir de las imágenes de la figura B.1, B.2 y B.3. La figura B.10 muestra el archivo que contiene los pesos y ganancias finales obtenidas en el módulo de entrenamiento, estos archivos son utilizados en las pruebas 4 y 5.

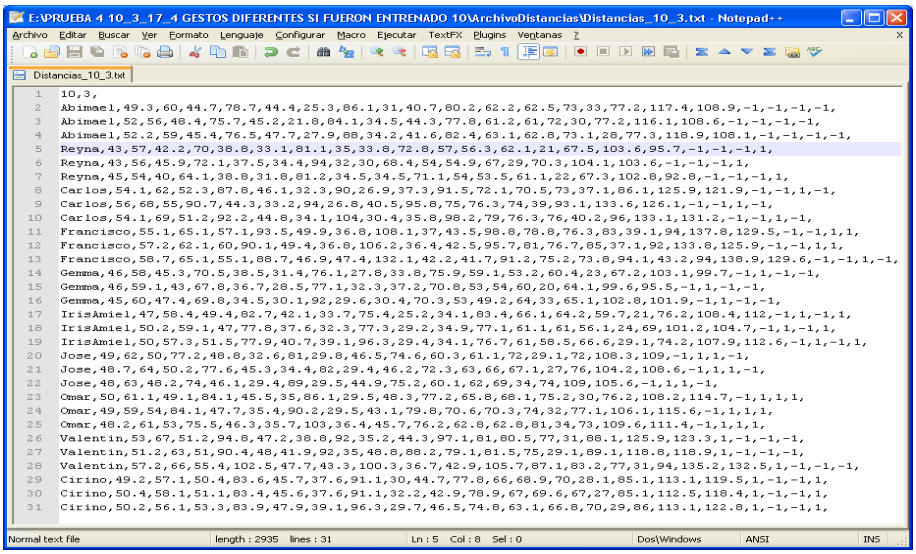


Figura B.9. Archivo de distancias de las pruebas 4 y 5.

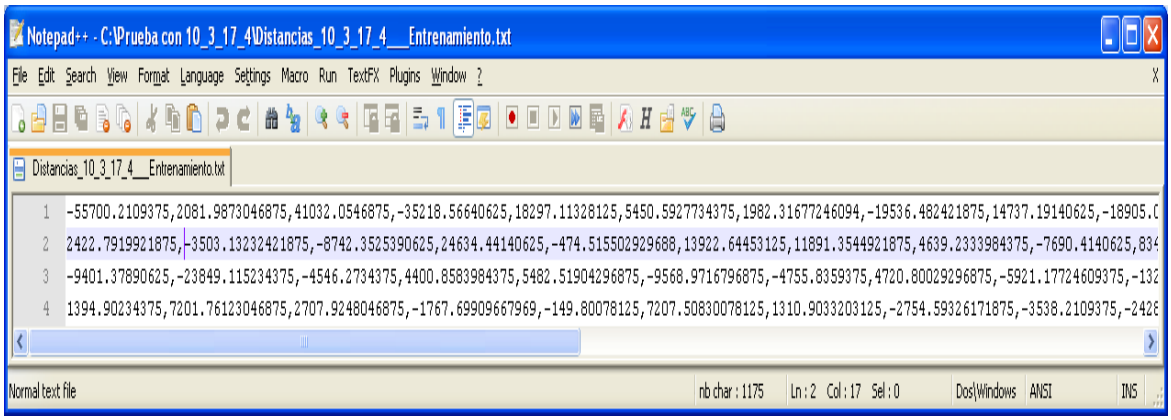


Figura B.10. Archivo de pesos entrenados de las pruebas 4 y 5.

La figura B.11 muestra el resultado del módulo de reconocimiento obtenido en la prueba 4.

Abimael,52,56,48.4,75.7,45.2,21.8,84.1,34.5,44.3,77.8,61.2,61,72,30,77.2,116.1,108.6,

Reyna,43,56,45.9,72.1,37.5,34.4,94,32,30,68.4,54,54.9,67,29,70.3,104.1,103.6,

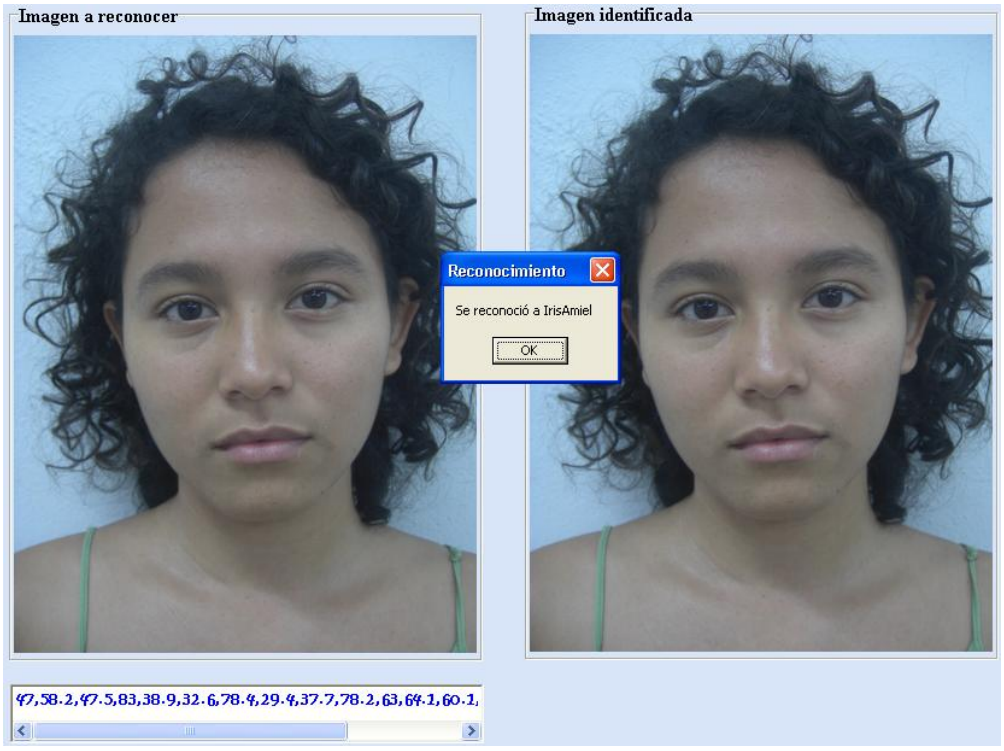
Figura B.11. Reconocimiento de las imágenes de la figura A.1 utilizadas en la prueba 4.

Carlos,56,68,55,90.7,44.3,33.2,94,26.8,40.5,95.8,75,76.3,74,39,93.1,133.6,126.1,

Francisco,57.2,62.1,60,90.1,49.4,36.8,106.2,36.4,42.5,95.7,81,76.7,85,37.1,92,133.8,125.9,

Figura B.11. Reconocimiento de las imágenes de la figura A.1 utilizadas en la prueba 4 (continuación).

Gemma,46,59.1,43,67.8,36.7,28.5,77.1,32.3,37.2,70.8,53,54,60,20,64.1,99.6,95.5,



IrisAmiel,50.2,59.1,47,77.8,37.6,32.3,77.3,29.2,34.9,77.1,61.1,61,56.1,24,69,101.2,104.7,

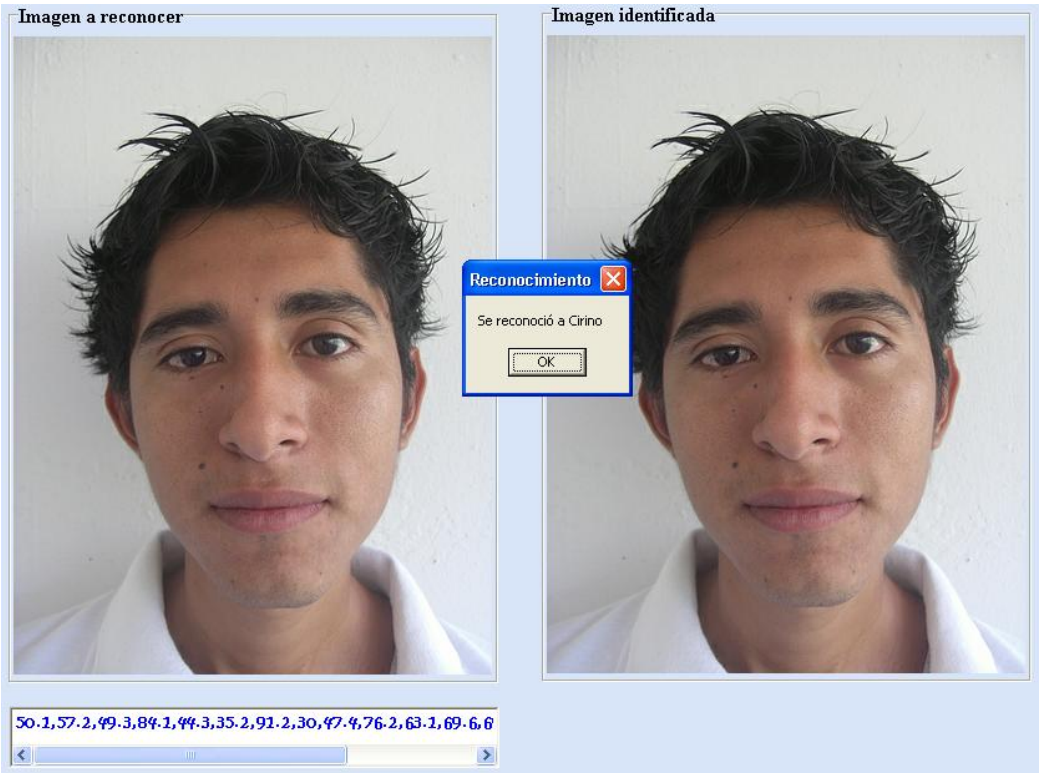
Figura B.11. Reconocimiento de las imágenes de la figura A.1 utilizadas en la prueba 4 (continuación).

Jose,48.7,64,50.2,77.6,45.3,34.4,82,29.4,46.2,72.3,63,66,67.1,27,76,104.2,108.6,

Omar,49,59,54,84.1,47.7,35.4,90.2,29.5,43.1,79.8,70.6,70.3,74,32,77.1,106.1,115.6,

Figura B.11. Reconocimiento de las imágenes de la figura A.1 utilizadas en la prueba 4 (continuación).

Valentin,51.2,63,51,90.4,48,41.9,92,35,48.8,88.2,79.1,81.5,75,29.1,89.1,118.8,118.9,



Cirino,49.2,57.1,50.4,83.6,45.7,37.6,91.1,30,44.7,77.8,66,68.9,70,28.1,85.1,113.1,119.5,

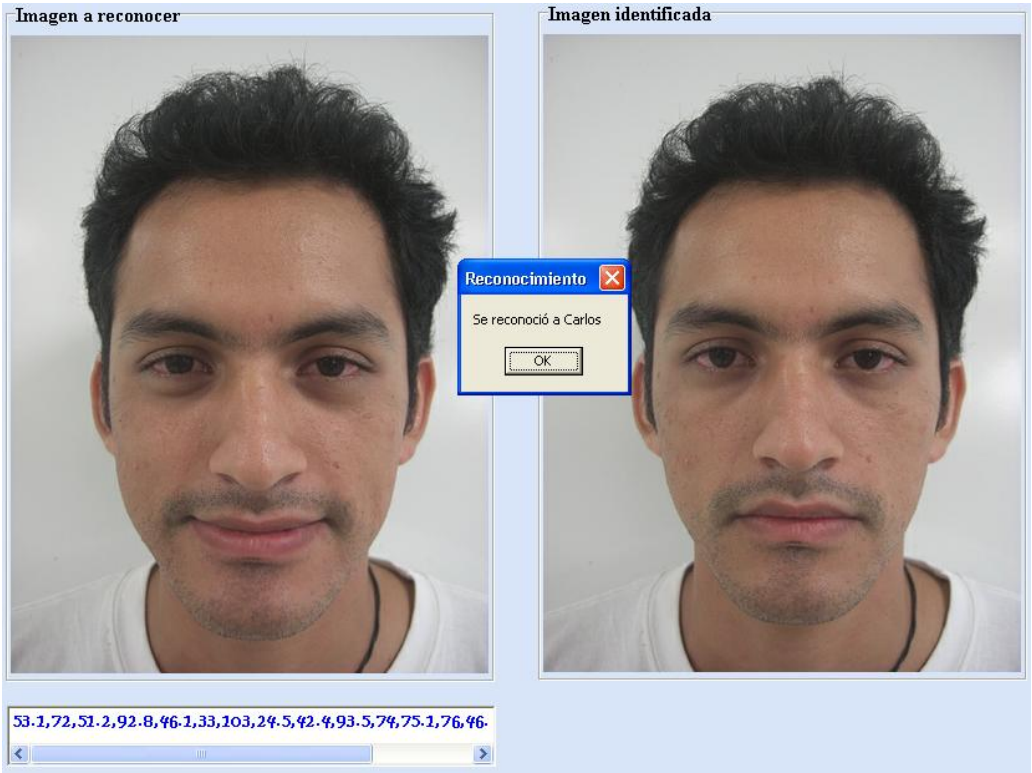
Figura B.11. Reconocimiento de las imágenes de la figura A.1 utilizadas en la prueba 4 (continuación).

Como se muestra en la figura B.11 las personas reconocidas fueron Abimael, Reyna Carlos, Francisco, Gemma, Iris Amiel, José, Omar, Valentín y Cirino, es decir, se reconocieron las imágenes de la figura A.1, dando como resultado un 100 % de certeza en dicha prueba.

En la figura B.12 se observa el resultado del módulo de reconocimiento obtenido en la prueba 5.

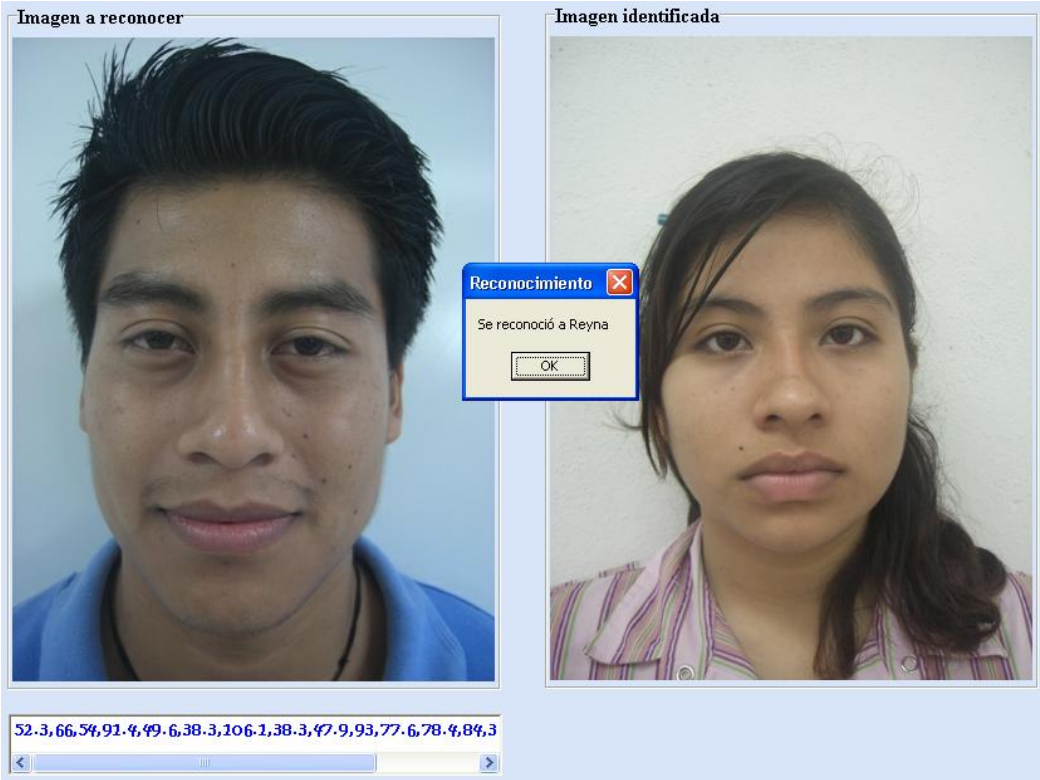
Figura B.12. Reconocimiento de las imágenes de la figura A.4 utilizadas en la prueba 5.

Reyna,44,55.1,43.2,72.5,36.4,31.4,86.1,30.7,35.8,69.2,54,60.7,66,26,69.3,102.7,99.2,



Carlos,53.1,72,51.2,92.8,46.1,33,103,24.5,42.4,93.5,74,75.1,76,46.1,95,133.7,130.5,

Figura B.12. Reconocimiento de las imágenes de la figura A.4 utilizadas en la prueba 5 (continuación).



Francisco,52.3,66,54,91.4,49.6,38.3,106.1,38.3,47.9,93,77.6,78.4,84,33,94,132.5,126.7,

Gemma,43,61,43,71,33.6,29.8,83.1,29.7,34.7,70.4,56,56,62,26,65.3,101.7,95.5,

Figura B.12. Reconocimiento de las imágenes de la figura A.4 utilizadas en la prueba 5 (continuación).

IrisAmiel,46,61.2,47.4,81.8,39.4,32,79.5,26.1,37.6,78.4,63,63.1,62.2,26.2,76.2,108.4,111.3
,

Jose,48,63,45,74.2,43.9,36.1,90,30.5,47.9,70.8,59.3,61,69.1,28.1,70,104.1,102.8,

Figura B.12. Reconocimiento de las imágenes de la figura A.4 utilizadas en la prueba 5 (continuación).

Omar,47,58,51,73.2,45.8,44.4,90,36.4,49.6,63.4,52.6,59.7,75.2,25.1,60,95,101.2,

Valentin,55.1,69,53.5,98.9,52.2,40,96,36.6,44.7,100.3,83.2,80.2,80,32.1,92,133,130.4,

Figura B.12. Reconocimiento de las imágenes de la figura A.4 utilizadas en la prueba 5 (continuación).

Cirino,48.4,60.1,50.2,82,47.9,39.1,95.3,30.4,47.9,74,64,66.6,71,27,84.1,111.1,119.2,

Figura B.12. Reconocimiento de las imágenes de la figura A.4 utilizadas en la prueba 5 (continuación).

Como se muestra en la figura B.12 las personas reconocidas fueron Abimael, Reyna Carlos, Francisco, Gemma, José y Omar, es decir, se reconocieron imágenes de la figura A.4, dando como resultado un 70 % de certeza en esta prueba.

En la figura B.13 se muestra el archivo de distancias de la prueba 6 y en la figura B.14 el archivo de entrenamiento.

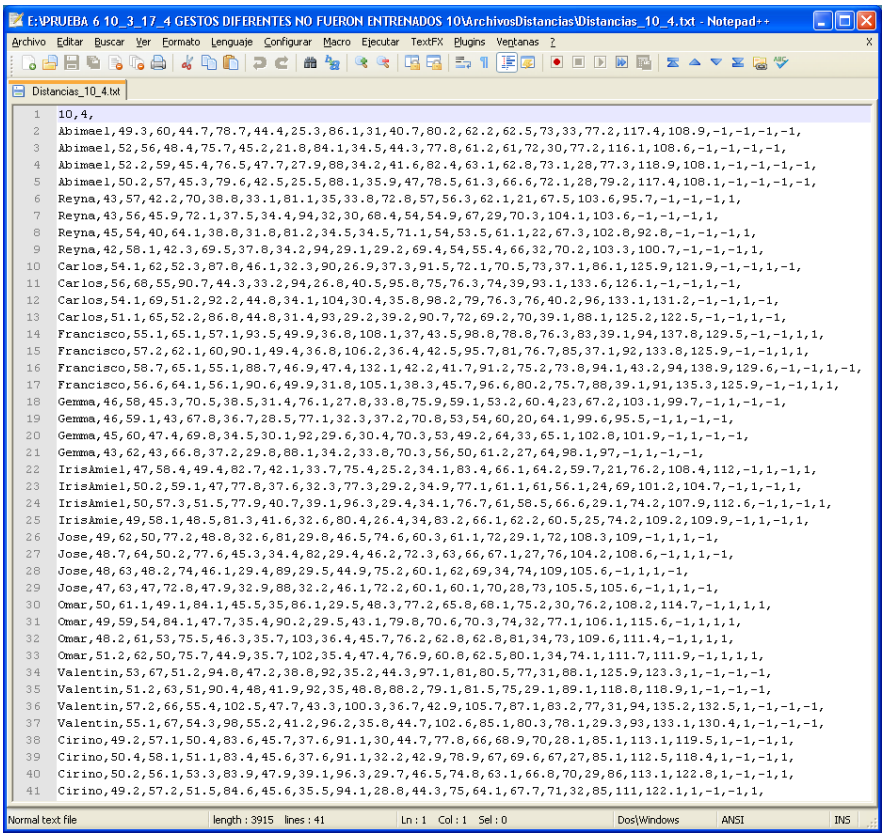


Figura B.13. Archivo de distancias de la prueba 6.

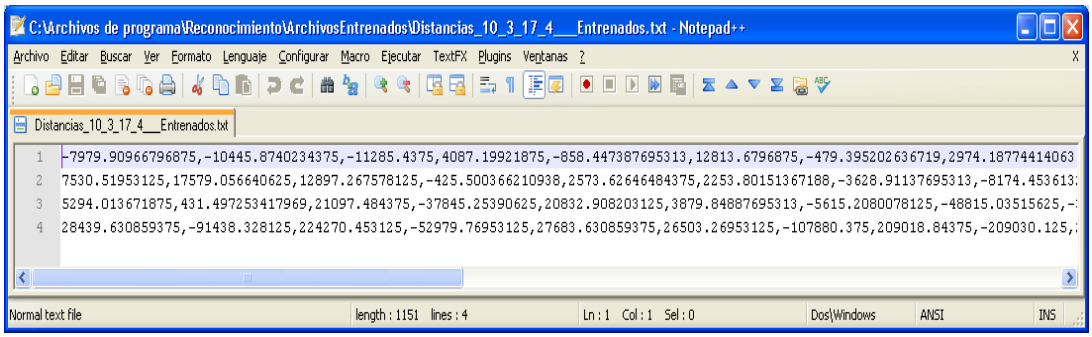


Figura B.14. Archivo de entrenamiento de la prueba 6.

En la figura B.15 se observa el resultado del módulo de reconocimiento obtenido en la prueba 6.

Abimael,52,56,48.4,75.7,45.2,21.8,84.1,34.5,44.3,77.8,61.2,61,72,30,77.2,116.1,108.6,

Reyna,43,56,45.9,72.1,37.5,34.4,94,32,30,68.4,54,54.9,67,29,70.3,104.1,103.6,

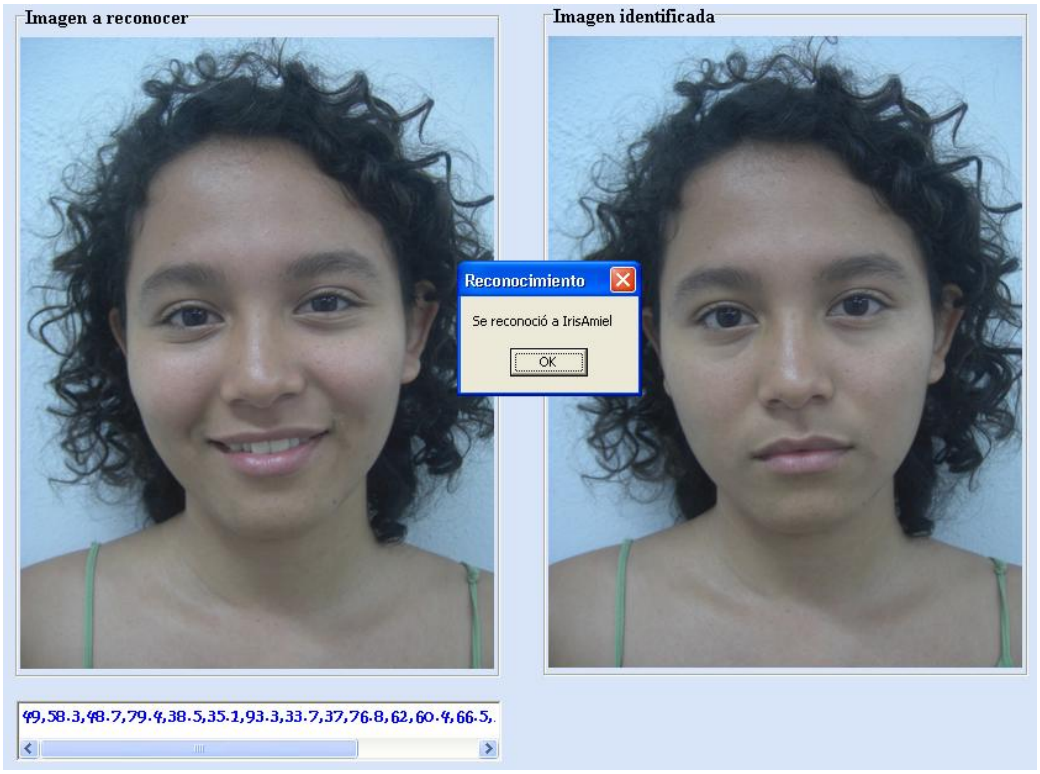
Figura B.15. Reconocimiento de las imágenes de la figura A.2 utilizadas en la prueba 6.

Carlos,56,68,55,90.7,44.3,33.2,94,26.8,40.5,95.8,75,76.3,74,39,93.1,133.6,126.1,

Francisco,57.2,62.1,60,90.1,49.4,36.8,106.2,36.4,42.5,95.7,81,76.7,85,37.1,92,133.8,125.9,

Figura B.15. Reconocimiento de las imágenes de la figura A.2 utilizadas en la prueba 6 (continuación).

Gemma,46,59.1,43,67.8,36.7,28.5,77.1,32.3,37.2,70.8,53,54,60,20,64.1,99.6,95.5,



IrisAmiel,50.2,59.1,47,77.8,37.6,32.3,77.3,29.2,34.9,77.1,61.1,61,56.1,24,69,101.2,104.7,

Figura B.15. Reconocimiento de las imágenes de la figura A.2 utilizadas en la prueba 6 (continuación).

Jose,48.7,64,50.2,77.6,45.3,34.4,82,29.4,46.2,72.3,63,66,67.1,27,76,104.2,108.6,

Omar,49,59,54,84.1,47.7,35.4,90.2,29.5,43.1,79.8,70.6,70.3,74,32,77.1,106.1,115.6,

Figura B.15. Reconocimiento de las imágenes de la figura A.2 utilizadas en la prueba 6 (continuación).

Valentin,51.2,63,51,90.4,48,41.9,92,35,48.8,88.2,79.1,81.5,75,29.1,89.1,118.8,118.9,

Cirino,49.2,57.1,50.4,83.6,45.7,37.6,91.1,30,44.7,77.8,66,68.9,70,28.1,85.1,113.1,119.5,

Figura B.15. Reconocimiento de las imágenes de la figura A.2 utilizadas en la prueba 6 (continuación).

Como se observa en la figura B.15 las personas reconocidas fueron Abimael, Reyna Carlos, Francisco, Gemma, José, Iris Amiel, Omar, Valentín y Cirino, es decir, se reconocieron las imágenes de la figura A.2, dando como resultado un 100 % de certeza en esta prueba.

A continuación se muestra un resumen de las imágenes utilizadas en las pruebas (Fig. B.16).

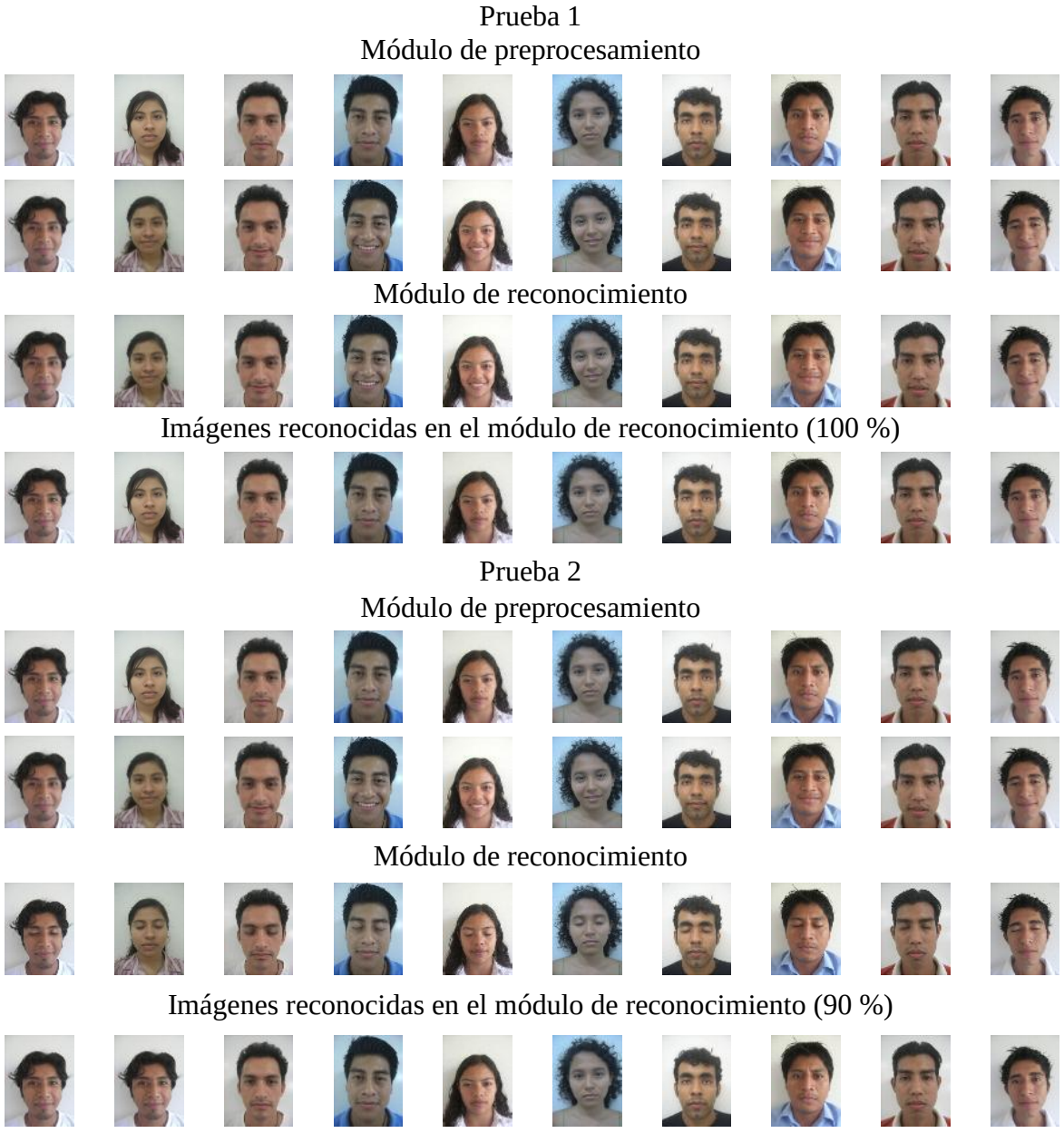


Figura B.16. Imágenes utilizadas en los módulos de las pruebas.

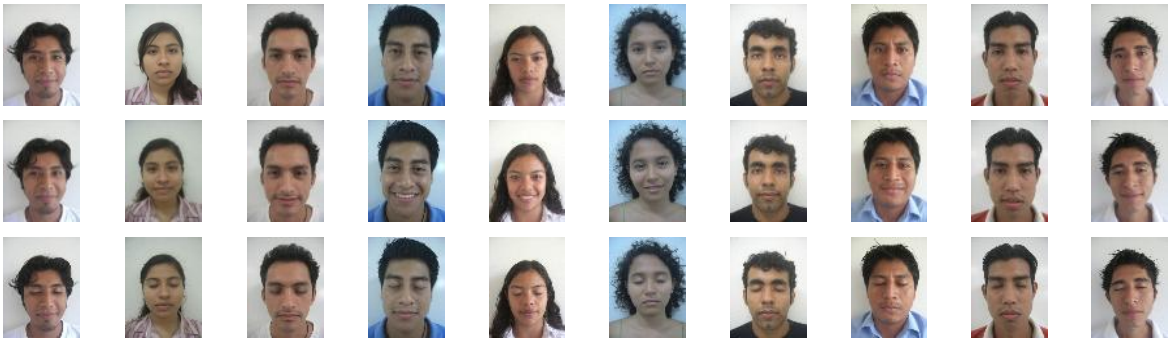
Prueba 3

Módulo de preprocesamiento

Módulo de reconocimiento (100 %)

Prueba 4

Módulo de preprocesamiento



Módulo de reconocimiento

Imágenes reconocidas en el módulo de reconocimiento (100 %)

Prueba 5

Módulo de preprocesamiento

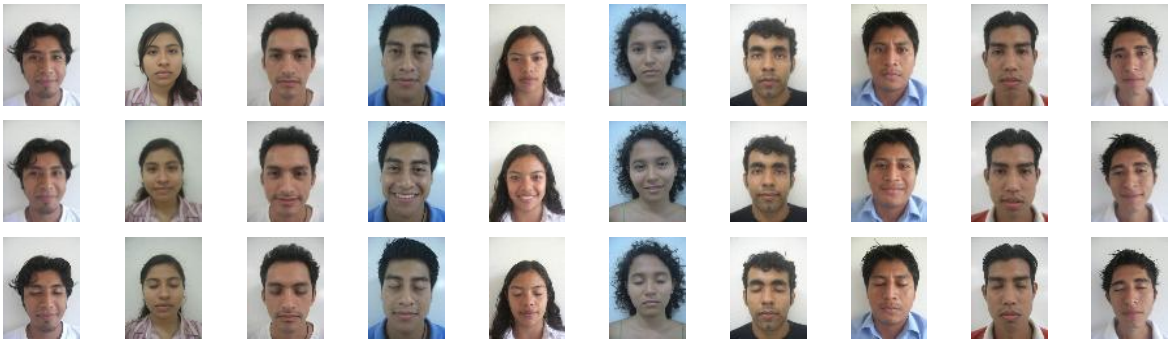


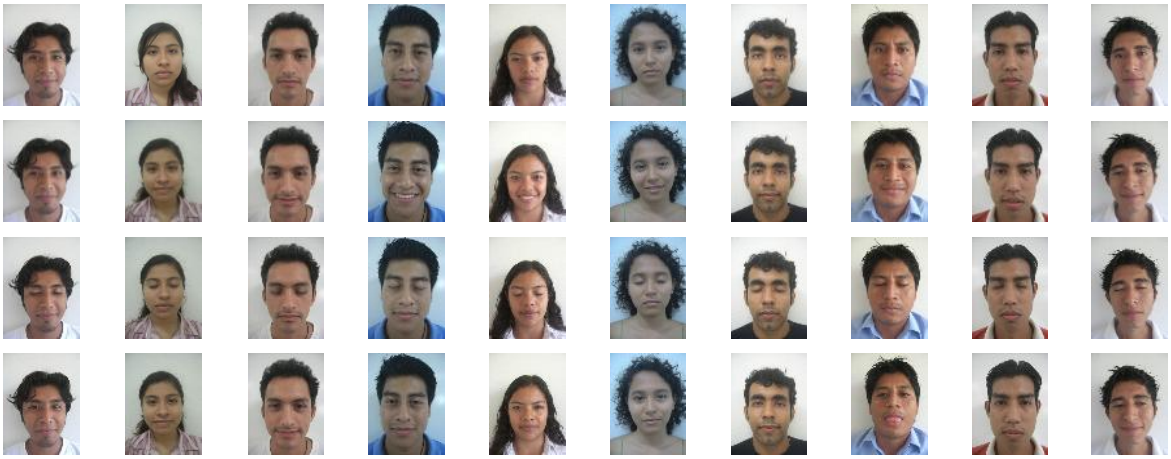
Figura B.16. Imágenes utilizadas en los módulos de las pruebas (continuación).

Módulo de reconocimiento

Imágenes reconocidas en el módulo de reconocimiento (70 %)

Prueba 6

Módulo de preprocesamiento



Módulo de reconocimiento

Imágenes reconocidas en el módulo de reconocimiento (100 %)

Figura B.16. Imágenes utilizadas en los módulos de las pruebas (continuación).

ANEXO C. FUNCIONAMIENTO DEL SISTEMA

A continuación se describe cada uno de los módulos del sistema para su mejor comprensión. Se sugiere instalar la aplicación que se encuentra en el CD adjunto a esta tesis, para ubicar cada una de las imágenes que se muestran en este apartado.

En la figura C.1 se muestra la pantalla de presentación del sistema, después de unos cuantos segundos se abre la ventana principal de la aplicación (Fig. C.2).

En la figura C.3, se puede ver el menú principal (menú `Imagen`). Es en éste donde se puede abrir cada uno de los módulos: (a) Preprocesamiento, (b) Entrenamiento y (c) Reconocimiento, seleccionando estas opciones se muestran dichos módulos y existe una opción para salir de la aplicación.

Figura C.1. Ventana de presentación del sistema.

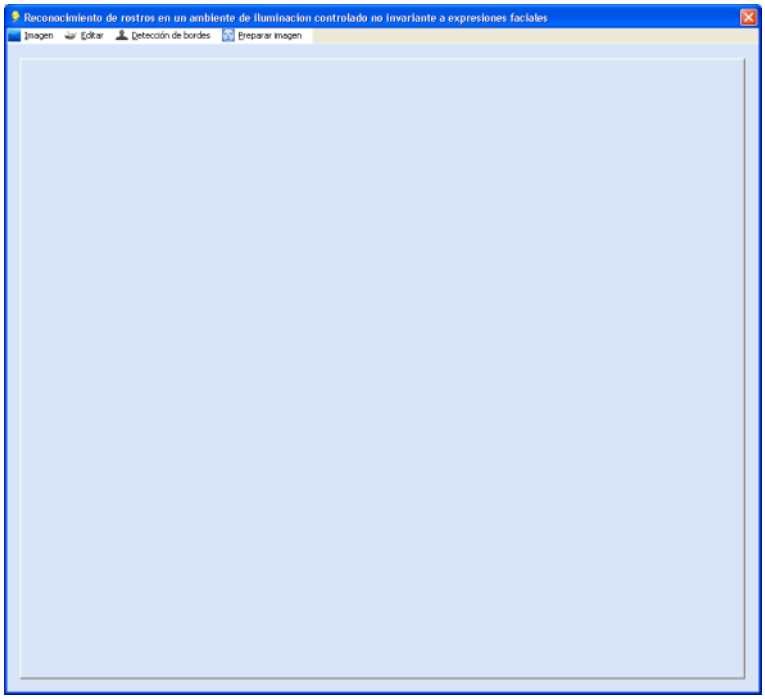


Figura C.2. Ventana principal de la aplicación.

En el menú de la figura C.3 está la opción `Abrir`, la cual abre una imagen y la muestra en el módulo de preprocesamiento, al dar clic a esta opción se muestra un cuadro de diálogo (Fig. C.4) para buscar la imagen y llevar a cabo su preprocesamiento.

Figura C.3. Menú `Imagen`.

Figura C.4. Ventana para abrir una imagen.

El menú `Editar` es el segundo menú, este menú es importante para el módulo de preprocesamiento por que la opción `escala de grises` es parte fundamental de la aplicación para los siguientes algoritmos, a toda imagen se le aplica este primer proceso. Existen también opciones de `Ecualización` y `Umbralización`, en esta última opción se debe indicar un valor de umbral, de lo contrario se toma el 127 por defecto; las opciones `Pasa Alto` o `Pasa Bajo` son para eliminar altas o bajas frecuencias en la imagen, son buenas para eliminar ruido de este tipo (Fig. C.5).

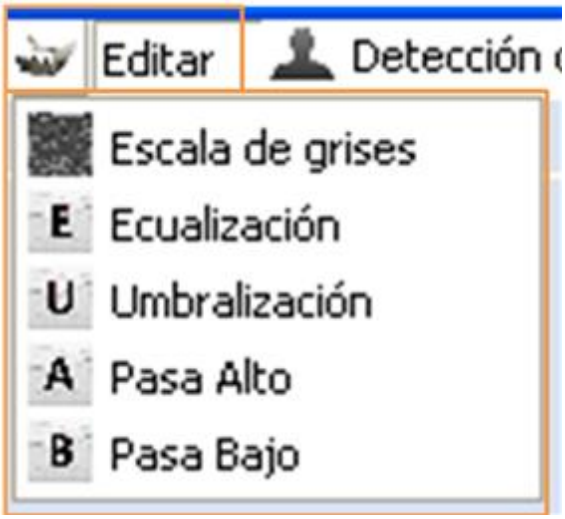


Figura C.5. Menú `Editar`.

El menú `Detección de bordes` tiene 5 algoritmos para aplicar: Sobel, Roberts, Prewitt, Frei-Chen y Laplaciano, los cuales son utilizados para detectar bordes en el contenido de las imágenes (Fig. C.6).

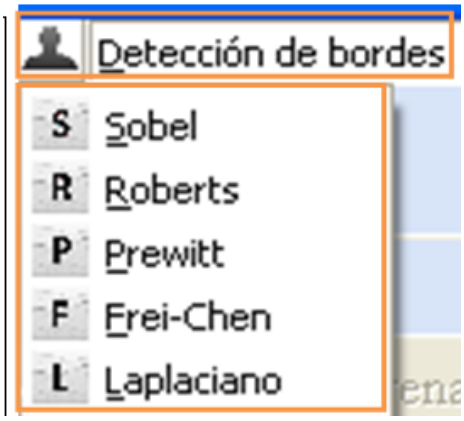


Figura C.6. Menú `Detección de bordes`.

El menú `Preparar imagen` (Fig. C.7) tiene una sola opción `Realizar procesos`. Esta opción realiza las operaciones básicas de escala de grises y aplicación del filtro de Sobel, que son las operaciones que se aplican para el cálculo de las distancias de los rasgos principales de la imagen.

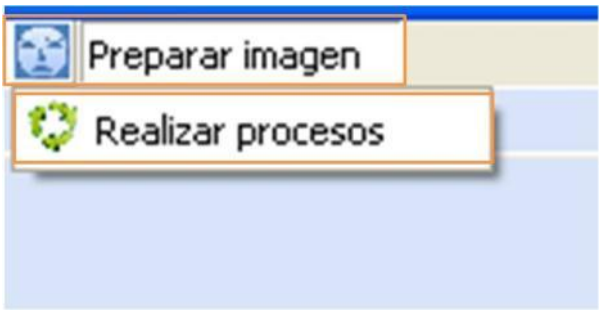


Figura C.7. Menú `Preparar imagen`.

Módulo de preprocesamiento

Para mostrar este módulo, en la ventana principal hay que abrir el menú `Imagen` (Fig. C.3) y seleccionar la opción `Preprocesamiento`. Enseguida se mostrará el módulo en la ventana principal de la aplicación (Fig. C.8).

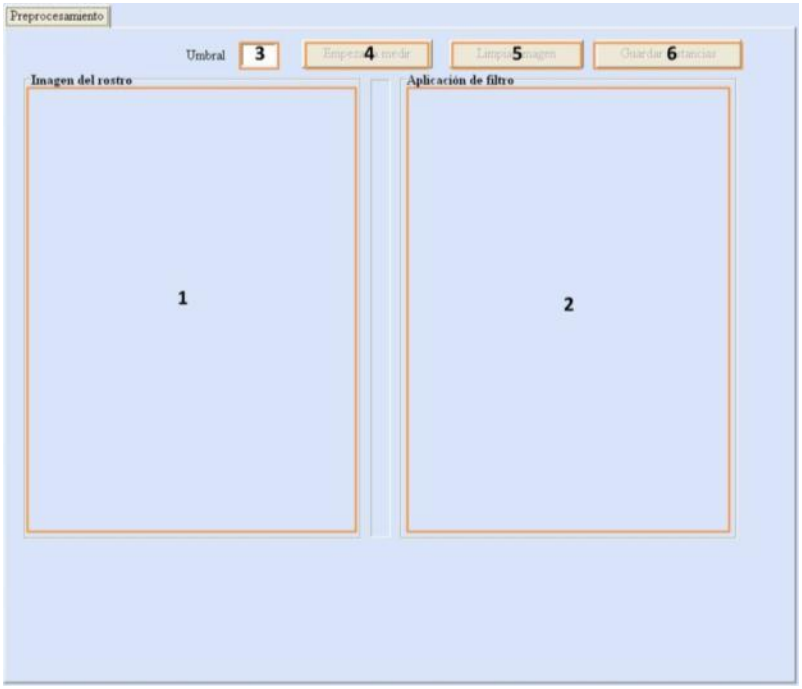


Figura C.8. Componentes funcionales del módulo de preprocesamiento.

Este módulo tiene 6 componentes principales:

Componente 1: es un panel que se utiliza para mostrar la imagen a la que se le quiere calcular sus distancias, después de haber aplicado los filtros correspondientes.

Componente 2: es donde se carga una copia de la imagen abierta en el componente 1, en este componente se muestran los resultados de la aplicación de los filtros.

Componente 3: en éste el usuario introduce un valor entero utilizado como valor de umbral, y es utilizado cuando el usuario aplica el filtro de umbralización. Si el usuario no introduce ningún dato, el sistema toma el valor por defecto de 127 para el umbral.

Componente 4: es un botón que avisa al sistema que se van a empezar a medir las distancias sobre la imagen, al presionar el botón se muestran tres elementos iniciales en la parte de abajo del componente 2, la primera es una imagen que muestra que es lo que se va midiendo, el segundo elemento es un mensaje y el tercer elemento es una imagen que muestra que distancia sobre la imagen se va a medir (Fig. C.9).

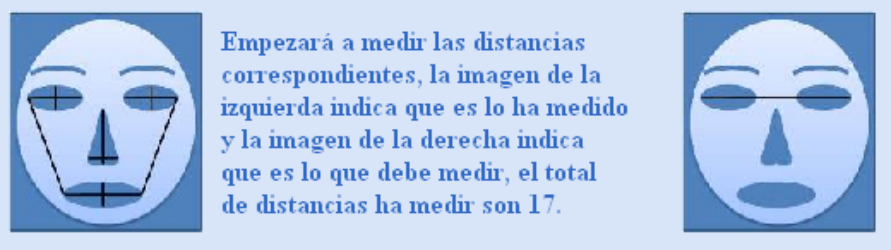


Figura C.9. Tres elementos que se muestran debajo del componente 2.

Componente 5: este botón sirve para limpiar las distancias (líneas) de la imagen, por si el usuario se equivocó. Por ejemplo, si el usuario se equivocó en el cálculo de la tercera distancia, al dar clic a este componente el sistema borrará las tres distancias y tendría que volver a empezar el cálculo de las distancias.

Componente 6: este último botón guarda las distancias (registro) en un archivo de texto, por lo tanto, al presionar este botón, el sistema muestra un cuadro de diálogo para guardar los datos (Fig. C.10). Cada vez que se abre una imagen y se calculan sus distancias se deben guardar los datos de esta imagen, al hacer esto el sistema concatena el nombre de la imagen abierta.

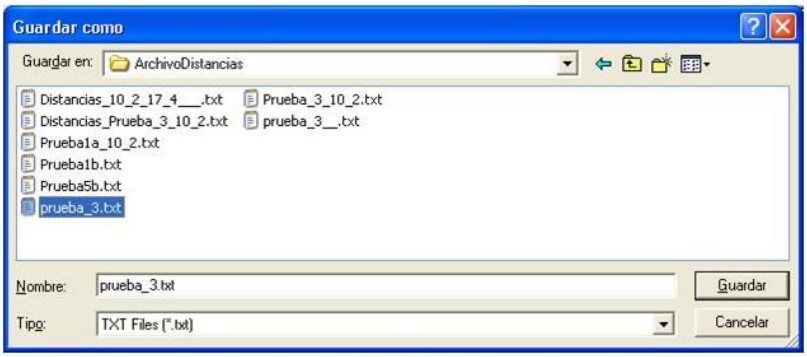


Figura C.10. Ventana para guardar un archivo de texto.

Para abrir una imagen y mostrarla en el módulo de preprocesamiento se debe estar en el menú `Imagen`, seleccionar `Abrir` (Fig. C.3) y mostrará un cuadro de diálogo (Fig. C.4), en el cual se tiene que localizar la imagen, seleccionarla y dar clic en `Abrir`, para que se muestre dicha imagen en los componentes 1 y 2 del módulo de preprocesamiento (Fig. C.8) y se activa el botón `Empezar a medir`, como se puede ver en la figura C.11.

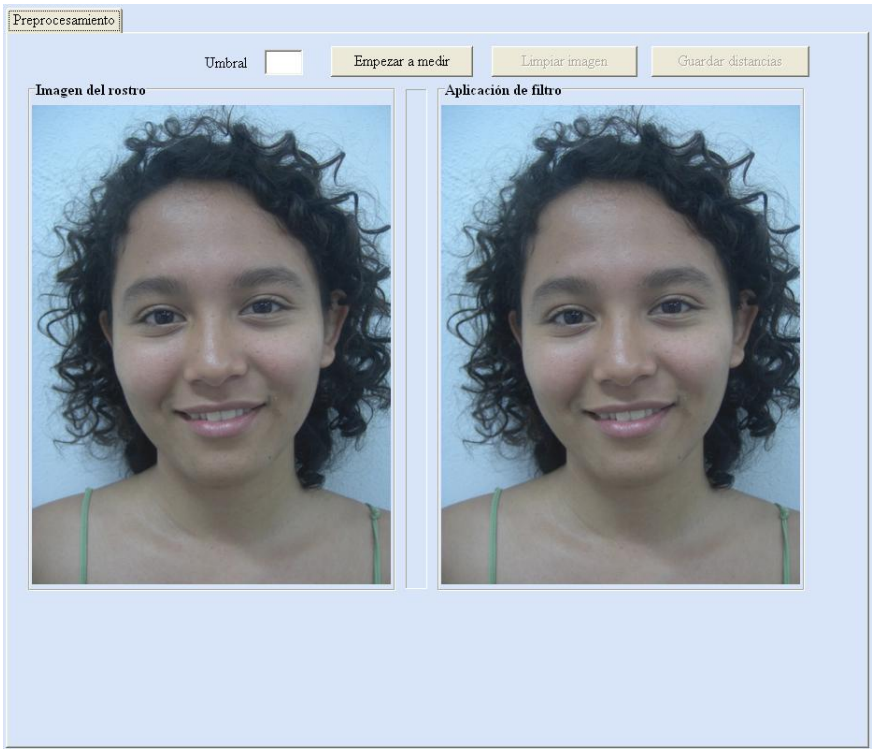


Figura C.11. Después de abrir una imagen se muestra en los componentes 1 y 2.

Después de hacer lo anterior se debe pasar la imagen a escala de grises. Siempre será la escala de grises el proceso que se aplique primero a la imagen para que después se pueda aplicar alguna otra técnica, como puede ser la aplicación de un filtro.

Aplicar escala de grises

Para aplicar este algoritmo a la imagen se debe estar en el menú `Editar` (Fig. C.5), seleccionar la opción `Escala de grises` y el resultado de dicho algoritmo se verá reflejado en el componente 2 del módulo de preprocesamiento (Fig. C.8.). El resultado de aplicar la escala de grises a la imagen de la figura C.11, se muestra en la figura C.12.

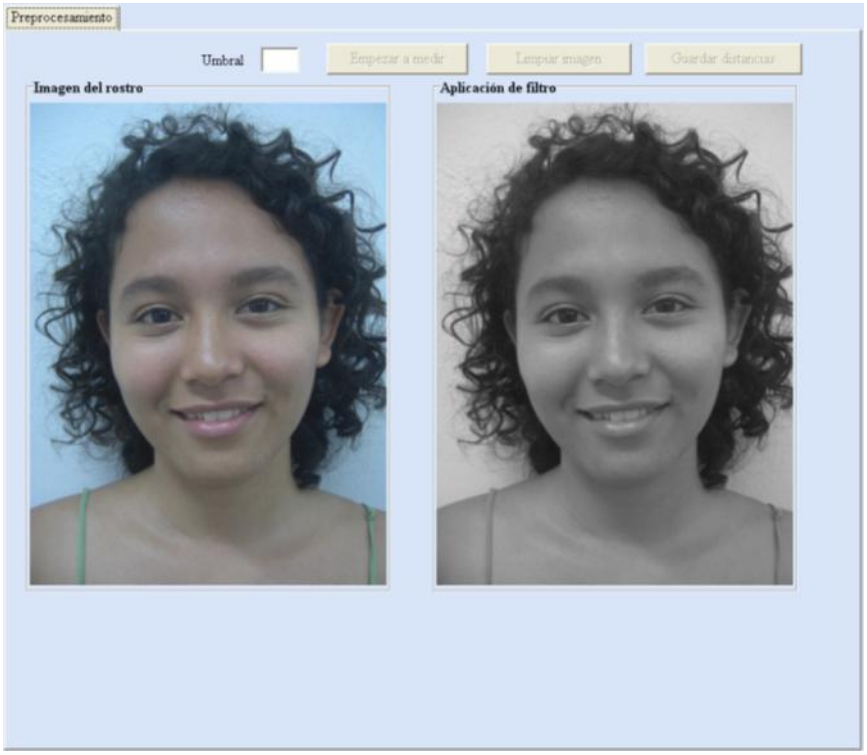


Figura C.12. Aplicación del algoritmo de escala de grises.

Aplicar ecualización

Para ejecutar el algoritmo de ecualización primero debe estar la imagen en escala de grises, posteriormente del menú `Editar` (Fig. C.4) se selecciona la opción `Ecualización` y el resultado de dicho algoritmo se verá reflejado en el componente 2 del módulo de preprocesamiento (Fig. C.8). El resultado de aplicar la ecualización a la imagen en escala de grises de la figura C.12, se muestra en la figura C.13.

Aplicar umbralización

Para ejecutar el algoritmo de umbralización primero debe estar la imagen en escala de grises, posteriormente en el componente etiquetado con el número 3 en la figura C.8 (cuadro de texto) se pone un valor entero entre 0 y 255 (umbral utilizado), este valor es el que tomará el algoritmo para realizar la operación de umbralización. Si no se introduce ningún dato, tomaría el valor por defecto de 127, después de hacer esto, ir al menú `Editar` (Fig. C.4), seleccionar la opción `Umbralización` y el resultado de dicho algoritmo se vería reflejado en el componente 2 del módulo de preprocesamiento (Fig. C.8).

El resultado de aplicar la umbralización a la imagen ecualizada de la figura C.13, se muestra en la figura C.14.

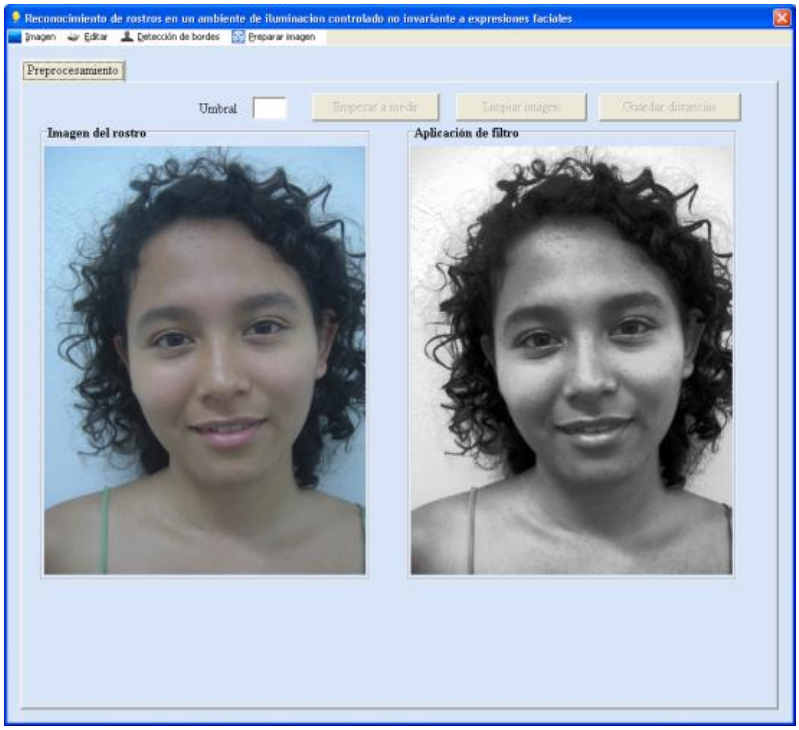


Figura C.13. Aplicación del algoritmo de ecualización.

Figura C.14. Aplicación del algoritmo de umbralización con un valor de 180.

Aplicar pasa alto

Para ejecutar el filtro pasa alto primero debe estar la imagen en escala de grises, después de hacer esto, ir al menú `Editar` (Fig. C.4), seleccionar la opción `Pasa Alto` y el resultado de dicho algoritmo se verá reflejado en el componente 2 del módulo de preprocesamiento (Fig. C.8). El resultado de aplicar pasa alto a la imagen en escala de grises de la figura C.12, se muestra en la figura C.15.

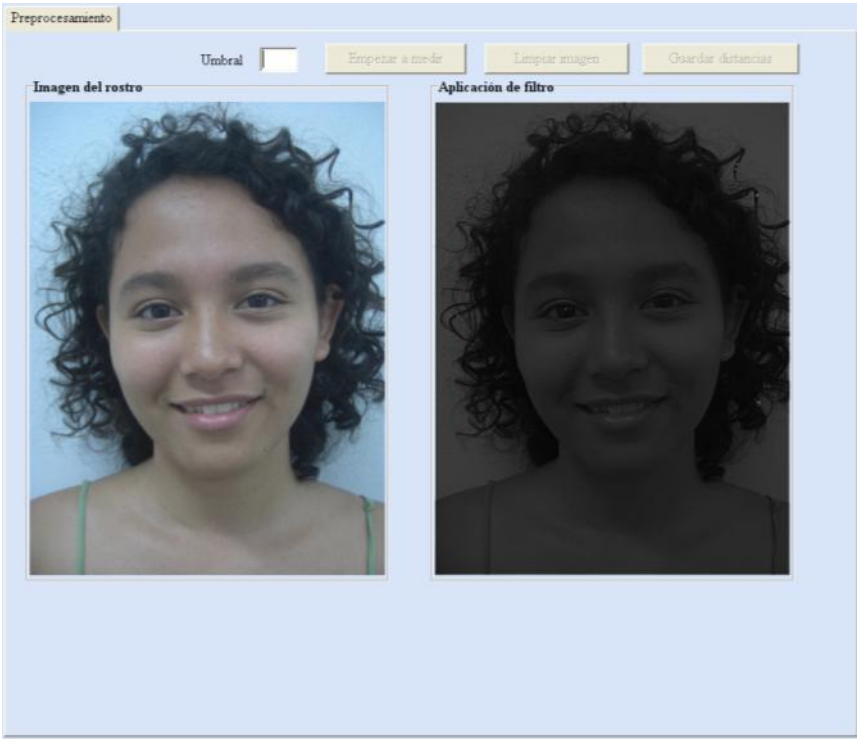


Figura C.15. Aplicación del algoritmo de pasa alto.

Aplicar pasa bajo

Para ejecutar el filtro pasa bajo primero debe estar la imagen en escala de grises, posteriormente se va al menú `Editar` (Fig. C.4), se selecciona la opción `Pasa bajo` y el resultado de dicho algoritmo se verá reflejado en el componente 2 del módulo de preprocesamiento (Fig. C.8). El resultado de aplicar pasa bajo a la imagen en escala de grises de la figura C.12, se muestra en la figura C.16.

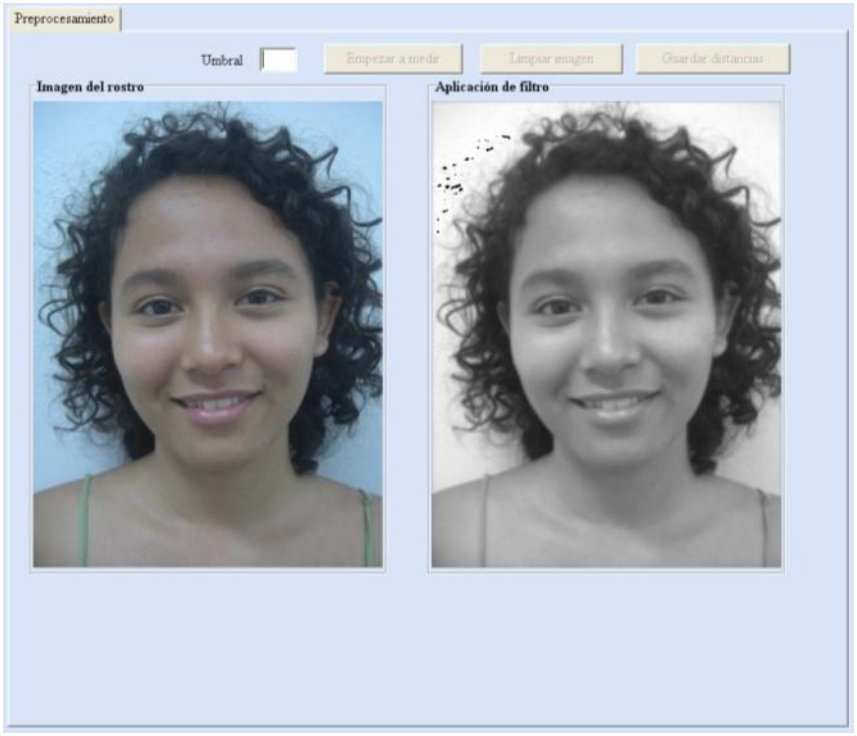


Figura C.16. Aplicación del algoritmo de pasa bajo.

Para ejecutar cada uno de los filtros del menú `Detección de bordes` (Fig. C.6) la imagen primero debe estar en escala de grises, por lo tanto, en el ejemplo se omitió este paso. Aclarado esto se procede a mostrar los ejemplos de la detección de bordes.

Aplicar filtro de Sobel

En el menú `Detección de bordes` (Fig. C.6), seleccionar la opción `Sobel` y el resultado de dicho algoritmo se verá reflejado en el componente 2 del módulo de preprocesamiento (Fig. C.8). El resultado de aplicar el filtro de Sobel a la imagen en escala de grises de la figura C.12, se muestra en la figura C.17.

Aplicar filtro de Roberts

Ir al menú `Detección de bordes` (Fig. C.6), seleccionar la opción `Roberts` y el resultado de dicho algoritmo se verá reflejado en el componente 2 del módulo de preprocesamiento (Fig. C.8). El resultado de aplicar el filtro de Roberts a la imagen en escala de grises abierta en la figura C.12, se muestra en la figura C.18.

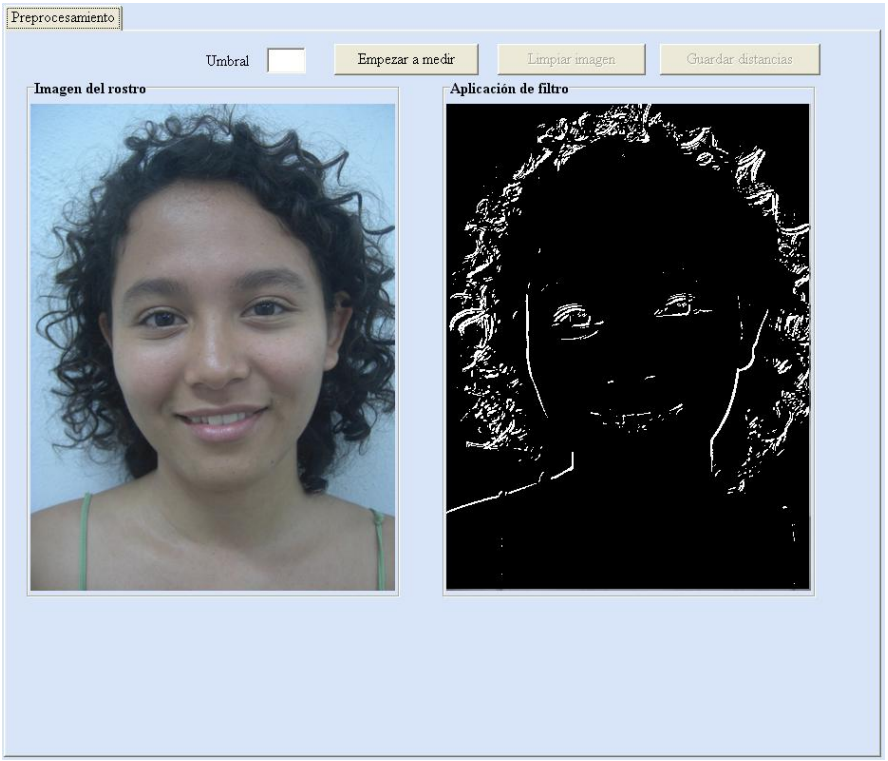


Figura C.17. Aplicación del filtro Sobel.

Figura C.18. Aplicación del filtro de Roberts.

Aplicar filtro de Prewitt

Para aplicar este filtro se debe ir al menú `Detección de bordes` (Fig. C.6), seleccionar la opción `Prewitt` y el resultado de dicho algoritmo se verá reflejado en el componente 2 del módulo de preprocesamiento (Fig. C.8). El resultado de aplicar el filtro de Prewitt a la imagen en escala de grises abierta en la figura C.12, se muestra en la figura C.19.

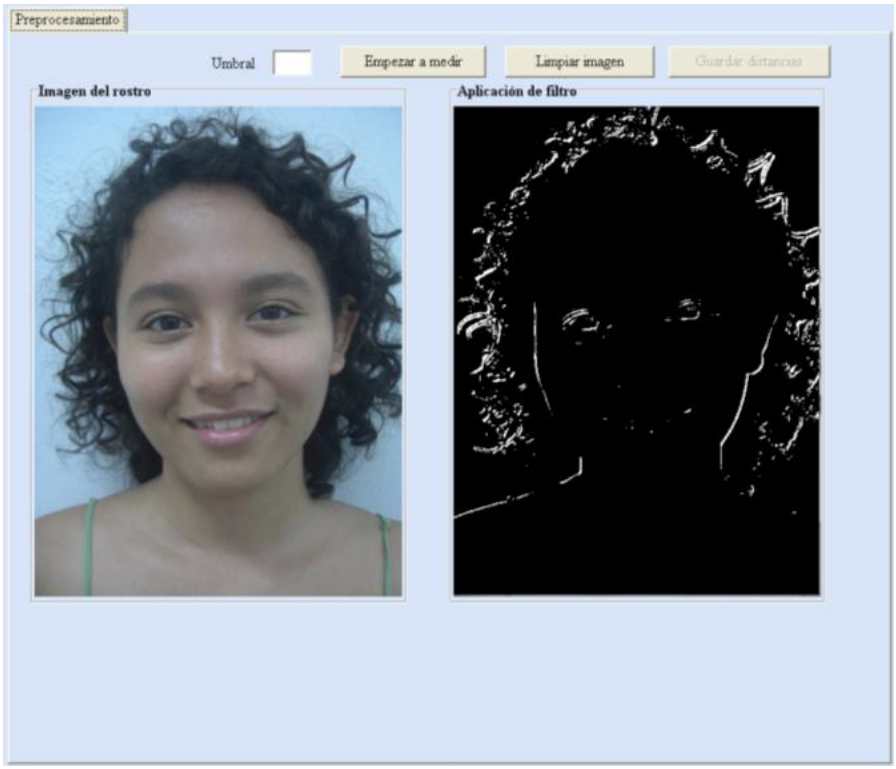


Figura C.19. Aplicación del filtro de Prewitt.

Aplicar filtro Frei-Chen

Para aplicar este filtro se tiene que ir al menú `Detección de bordes` (Fig. C.6), seleccionar la opción `Frei-Chen` y el resultado de dicho algoritmo se verá reflejado en el componente 2 del módulo de preprocesamiento (Fig. C.8). El resultado de aplicar el filtro Frei-Chen a la imagen en escala de grises de la figura C.12, se muestra en la figura C.20.

Aplicar filtro Laplaciano

Para ejecutar este filtro se debe de ir al menú `Detección de bordes` (Fig. C.6), seleccionar la opción `Laplaciano` y el resultado de dicho algoritmo se verá reflejado en

el componente 2 del módulo de preprocesamiento (Fig. C.8). El resultado de aplicar el filtro Laplaciano a la imagen en escala de grises abierta en la figura C.12, se muestra en la figura C.21.

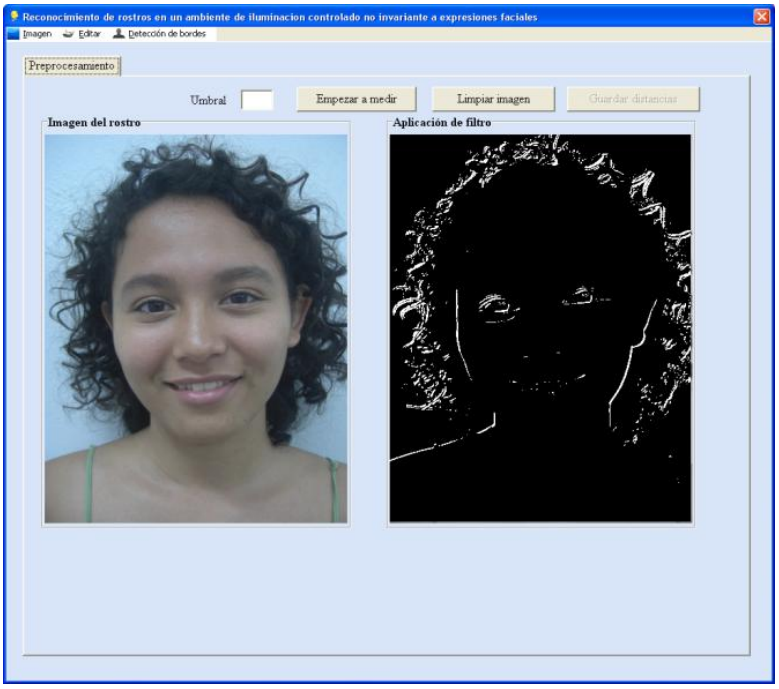


Figura C.20. Aplicación del filtro de Frei-Chen.

Figura C.21. Aplicación del filtro Laplaciano.

A continuación se muestra como ejemplo una serie de algoritmos aplicados a una imagen, con la finalidad de mostrar que se puede encontrar una secuencia de algoritmos que ayude a mejorar la detección de características. El objetivo sería su aplicación para calcular las distancias y ver las mejoras en los resultados que se pueden obtener con estos filtros.

Aplicar varios filtros secuenciales

Algoritmos utilizados:

- Escala de grises.
- Pasa bajo.
- Sobel.
- Laplaciano.

El resultado generado se muestra en la imagen de la figura C.22.

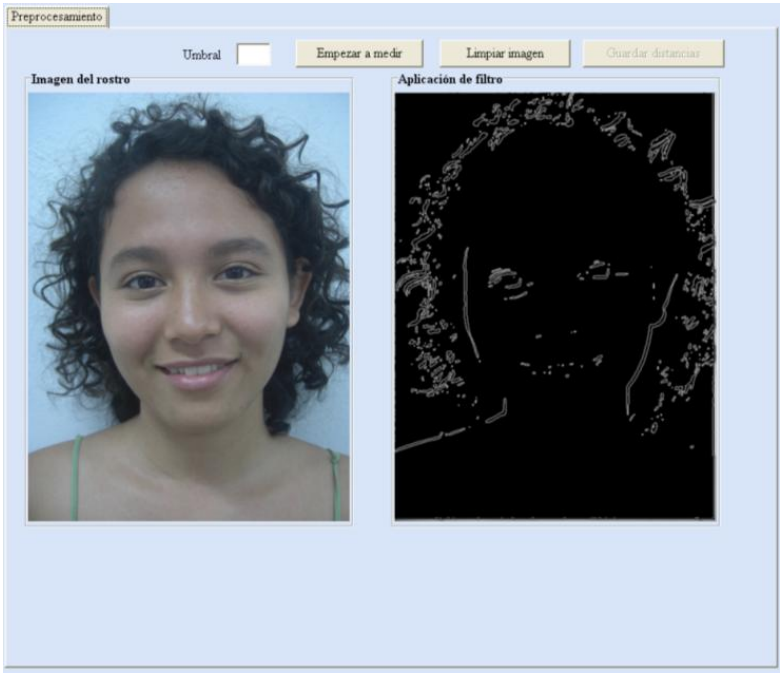


Figura C.22. Imagen resultado de aplicar escala de grises, pasa bajo, Sobel y Laplaciano.

De la misma forma se pueden ejecutar una serie de algoritmos que ayuden a identificar más detalles de la imagen para hacer el proceso automático, pero esta búsqueda dependerá del problema que se quiera resolver.

Como se puede observar en la figura C.17 en la cual se muestra el resultado del filtro de Sobel se aprecian mejor los bordes de la imagen, en contraste con las figuras C.18, C.19, C.20 y C.21, donde se observan los resultados de los filtros de Roberts, Prewitt, Frei-Chen y Laplaciano respectivamente. Motivo por el cual se decidio utilizar el filtro de Sobel como parte del desarrollo de este proyecto.

Módulo de entrenamiento

En este módulo es donde se cargan los datos, se muestran, se entrenan y se guardan los resultados. Este módulo tiene 4 componentes principales como se observa en la figura C.23. Este módulo es el más intuitivo, al dar clic al componente 2 (botón `Mostrar datos`) el sistema muestra un cuadro de diálogo (Fig. 4.19) para buscar el archivo de distancias, el cual fue creado en el módulo de procesamiento, y los datos son mostrados en el componente 1. Después de hacer esto, se activa el botón `Entrenar neurona` (componente 3), se pulsa este botón para entrenar los datos y al final del entrenamiento mostrará un mensaje de confirmación indicando dicha situación (Fig. C.24). Después se guardan los datos del entrenamiento, como son los pesos y las ganancias, presionando el botón `Guardar datos de aprendizaje` (componente 4), enseguida se muestra un cuadro de diálogo (Fig. 4.22) para guardar los datos.

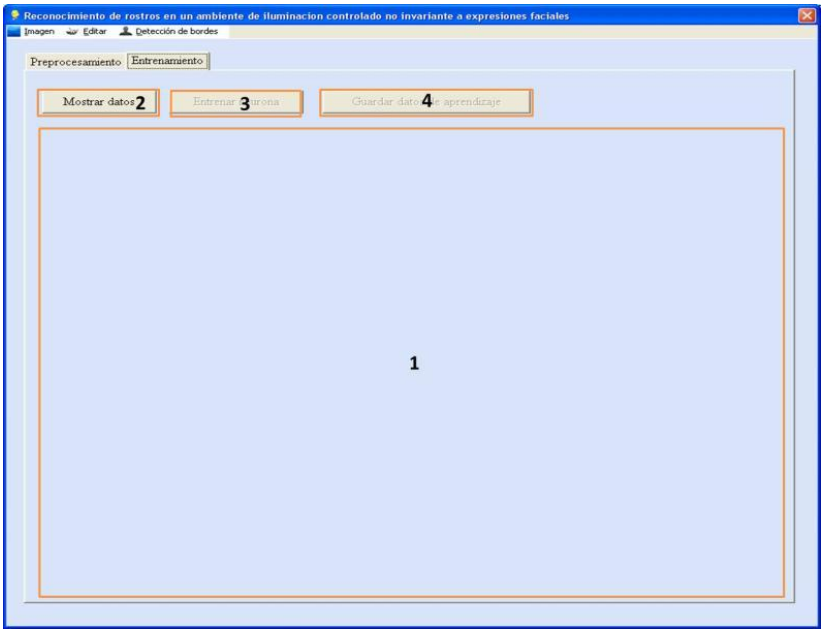


Figura C.23. Módulo de entrenamiento, componentes del módulo.

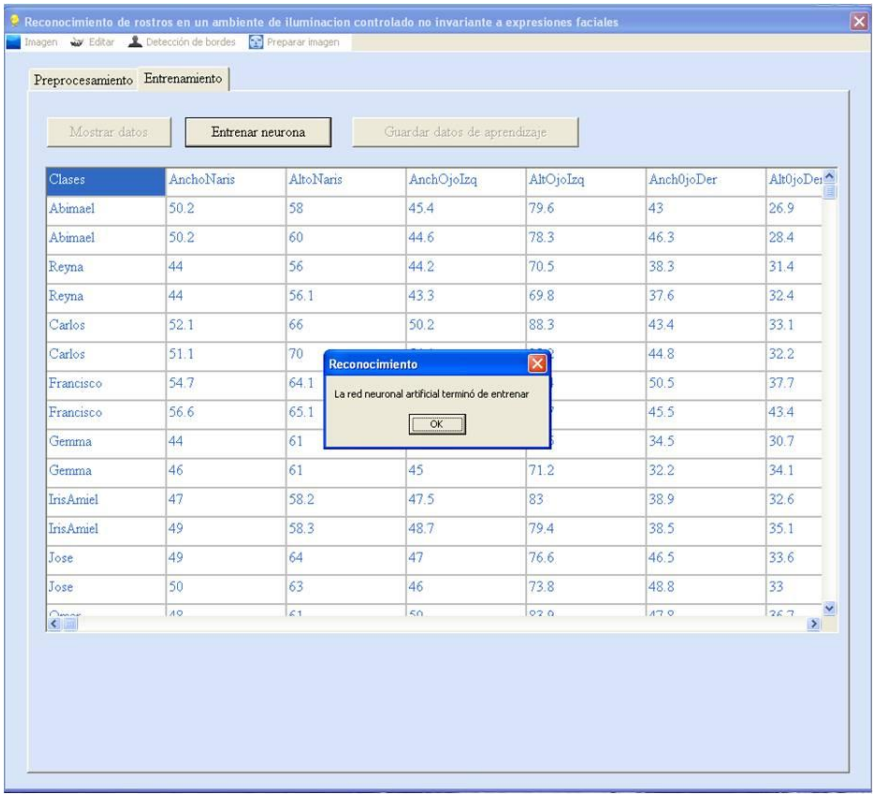


Figura C.24. Imagen que muestra el mensaje al finalizar la neurona su entrenamiento.

Módulo de reconocimiento

Este módulo (Fig. C.25) es para reconocer las imágenes de las persona. Aquí se abre una imagen y se muestra en el componente 1, los datos de la imagen se muestran en el componente 2, se abre el archivo de datos entrenados el cual se guardó en el módulo anterior, después se procede a abrir la base de conocimiento, es decir, el archivo de características creado en el módulo de preprocesamiento para que el sistema muestre el resultado de la clasificación y muestre en el componente 6 la imagen de la persona reconocida. Para que se muestre este módulo hay que dirigirse al menú `Imagen` (Fig. C.3) y elegir la opción `Reconocimiento`, el sistema mostrará la pestaña de reconocimiento.

Dos puntos importantes a tener en cuenta para este módulo:

- Deberá existir una carpeta en C:\Archivos de programa\Procesamiento\ con el nombre “ImagenesReconocer”, donde deberán estar todas las imágenes que se quieran reconocer.
- Los archivos que contienen los datos de la persona para su reconocimiento, es decir, un registro con 17 distancias principales, deberá estar en:

C:\Archivos de programa\Procesamiento\ArchivosReconocimiento\.

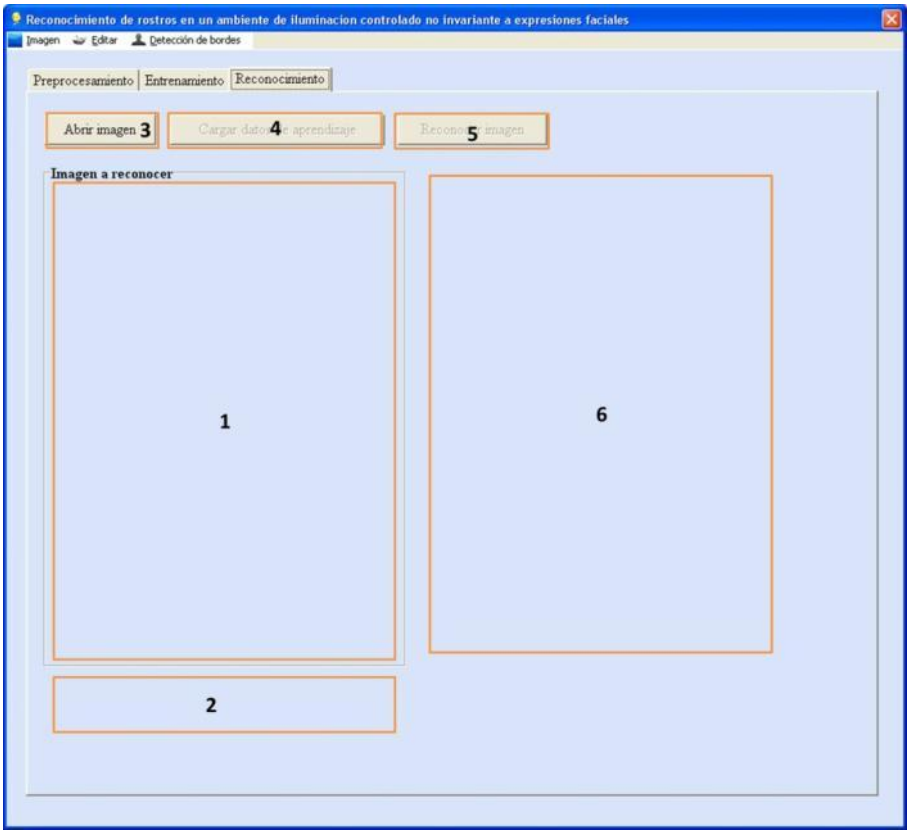


Figura C.25. Módulo de Reconocimiento con sus componentes principales.

Para abrir una imagen en este módulo es necesario dar clic al botón `Abrir Imagen`, éste mostrará un cuadro de diálogo (Fig. C.4), para buscar la imagen se selecciona y se da clic al botón `Abrir`, después se cargará la imagen en el componente 1. A continuación se abre un cuadro de diálogo como el mostrado en la figura 4.27, en el cual se busca, selecciona y abre el archivo de reconocimiento que contiene las 17 características del rostro, los datos del archivo (distancias euclideanas) se cargan en el componente 2 del módulo de reconocimiento (Fig. C.25). En caso de que no esté creado el archivo, el usuario cerrará la ventana y el sistema avisa lo que se debe hacer por medio de un mensaje (Fig. C.26). Después de dar `ok` a este mensaje el sistema envía al módulo de preprocesamiento para realizar el cálculo de las distancias, es decir, abrirá la imagen, aplicará los filtros, se calcularán sus distancias y posteriormente se guardará el archivo de reconocimiento. El nombre de este archivo no sigue ninguna regla específica, salvo que sea con extensión .txt.

Figura C.26. Imagen que indica que el archivo de datos no existe.

Enseguida se activa el botón `Cargar datos de aprendizaje` (componente 4 de la figura C.25), después se abre el archivo de pesos entrenados y posteriormente se activa el componente 5 (Fig. C.25). Se abre el archivo de distancias para que el sistema busque la imagen que reconoció y la muestre en el componente 6 (Fig. C.27).

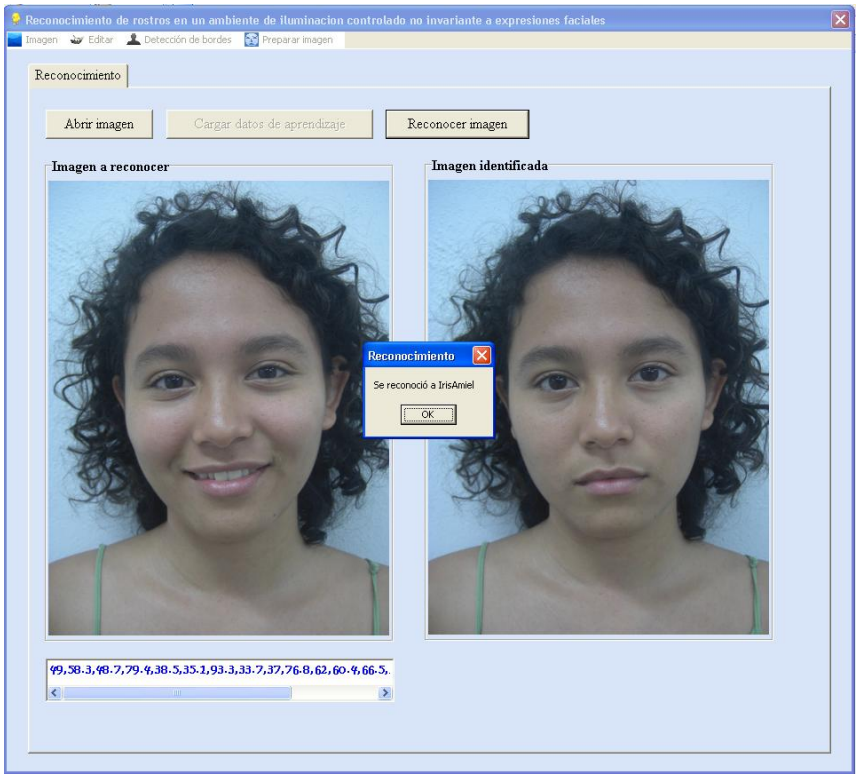


Figura C.27. Imagen para probar en el módulo de reconocimiento.

ANEXO D. CÓDIGO FUENTE

En este anexo se describe y se muestran las funciones del sistema que realizan las acciones principales en el proyecto.

MetodoEscalaGrises()

Este método realiza la media aritmética de los colores RGB de cada uno de los pixeles de la imagen abierta en el módulo de preprocesamiento, es decir, hace la suma de los tres valores de color de cada pixel y los divide entre tres, ese valor lo almacena en misma posición del pixel (Código 1).

```
void TfrmPrincipal::MetodoEscalaGrises(){
  intiColorReal,iColorRojo,iColorVerde,iColorAzul,iColorGris;
    for(int i=0;i<ANCHOIMAGEN;i++){
     BarraProceso->StepIt();
      for(int j=0;j<ALTOIMAGEN;j++){
```

Código 1. Conversión una imagen de mapa de bits a escala de grises.

```
      iColorReal=ColorToRGB(ImagenProcesamiento->Canvas-
       >Pixels[i][j]);
       iColorRojo = GetRValue(iColorReal);
       iColorVerde = GetGValue(iColorReal);
       iColorAzul = GetBValue(iColorReal);
      iColorGris = (iColorRojo+iColorVerde+iColorAzul)/3;
      ImagenProcesamiento->Canvas->Pixels[i][j]=(TColor)RGB
      (iColorGris,iColorGris,iColorGris);
      ImagenAuxiliar->Canvas->Pixels[i][j]=(TColor)RGB
      (iColorGris,iColorGris,iColorGris);
      MatrizGeneralEscalaGrises[i][j]= iColorGris;
       }
      }
    }
```

Código 1. Conversión una imagen de mapa de bits a escala de grises (continuación).

MetodoEcualizacion()

Este método realiza la operación de tener una normalización de los niveles de gris de una imagen, basándose en la probabilidad de cada pixel (Código 2).

```
void TfrmPrincipal::MetodoEcualizacion(){
  double VectorColores[256]={0},VectorFecuencia[256]={0},
  VectorSumatoria[256]={0};
  double VectoResultado[256]={0}, iSumaColores=0,
  iSumaFrecuencia=0,ValorGris=256;
  intValorPixel=0;
  for(int i=0;i<ANCHOIMAGEN;i++){
   for(int j=0;j<ALTOIMAGEN;j++){
     ValorPixel=MatrizGeneralEscalaGrises[i][j];
     VectorColores[ValorPixel]=VectorColores[ValorPixel]+1;
   }
  }
  for(int i=0;i<ValorGris;i++){
   iSumaColores +=VectorColores[i];
  }
  for(int i=0,k=0;i<ValorGris;i++){
   if(VectorColores[i]!=0){
     VectorFecuencia[i]=(VectorColores[i]/iSumaColores);
     iSumaFrecuencia+=VectorFecuencia[i];
     VectorSumatoria[i]=iSumaFrecuencia;//sumatoria frecuencia
     VectoResultado[i]=(VectorSumatoria[i]*(ValorGris-1))+0.5;//
     fórmula de la ecualización
     k++;
   }
  }
  for(int i=0;i<ANCHOIMAGEN;i++){
   BarraProceso->StepIt();
```

Código 2. Ecualización de los niveles de gris de una imagen.

```
    for(int j=0;j<ALTOIMAGEN;j++){
    ValorPixel=MatrizGeneralEscalaGrises[i][j];
    MatrizGeneralEscalaGrises[i][j] = (int)VectoResultado
    [ValorPixel;
    ImagenProcesamiento->Canvas->Pixels[i][j]=(TColor)RGB
    (MatrizGeneralEscalaGrises[i][j],
    MatrizGeneralEscalaGrises[i][j],MatrizGeneralEscalaGrises
    [i][j]);
    ImagenAuxiliar->Canvas->Pixels[i][j]=(TColor)RGB
  (MatrizGeneralEscalaGrises[i][j],MatrizGeneralEscalaGrises[i][j],
  MatrizGeneralEscalaGrises[i][j]);
  }
 }
}
```

Código 2. Ecualización de los niveles de gris de una imagen (continuación).

MetodoUmbralizacion()

Este método consiste en cambiar el valor de los pixeles dependiendo de un valor umbral *T*, es decir, hacer un filtro de datos a partir de uno sugerido o elegido (Código 3).

```
void TfrmPrincipal::MetodoUmbralizacion(){
  intValorPixel;
  for(int i=0;i<ANCHOIMAGEN;i++){
   for(int j=0;j<ALTOIMAGEN;j++){
     ValorPixel=MatrizGeneralEscalaGrises[i][j];
     FuncionValidaDatosdelUmbral();
     if(ValorPixel>=0 &&ValorPixel<=Umbral)
      MatrizGeneralEscalaGrises[i][j] = 0;
     else
      if(ValorPixel>Umbral || ValorPixel<=255)
       MatrizGeneralEscalaGrises[i][j] = 250;
   }
  }
}
```

Código 3. Umbralización de una imagen con un valor de umbral introducido por el usuario.

MetodoFiltroPasa()

Este método se encarga de atenuar los pixeles de baja frecuencia o resaltar los pixeles de alta frecuencia, según sea el caso (Código 4).

```
void TfrmPrincipal::MetodoFiltroPasa(int Mascara[3][3]){
  for(int i=0,iSumaFiltro=0;i<=(ANCHOIMAGEN-MASCARATAM);i++){
   for(int j=0;j<=(ALTOIMAGEN-MASCARATAM);j++){
    for(int k=0;k<MASCARATAM;k++){
      for(int z=0;z<MASCARATAM;z++){
```

Código 4. Filtro pasa bajo y pasa alto. La elección del filtro depende de los valores de la máscara.

```
        iSumaFiltro+=(MatrizGeneralEscalaGrises[i+k][z+j])*
        (Mascara[k][z]);
        }
       }
       MatrizGeneralEscalaGrises[i+1][j+1] = (iSumaFiltro/9);
      iSumaFiltro=0;
     }
    }
}
```

Código 4. Filtro pasa bajo y pasa alto. La elección del filtro depende de los valores de la máscara (continuación).

EscogeFiltroPaso()

En este método se elige la máscara que se envía a la función, si se ha seleccionado el filtro pasa alto su valor es 1, si se elige el filtro pasa bajo su valor es cero, dependiendo de estos valores el método `EscogeFiltroPasa` envía la máscara correspondiente a utilizar (Código 5).

```
void TfrmPrincipal::EscogeFiltroPasa(int ValorFiltro){
  int FiltroPasaAlto[3][3]={{-1,-1,-1},{-1,8,-1},{-1,-1,-1}};
  int FiltroPasaBajo[3][3]={{1,1,1},{1,2,1},{1,1,1}};
  if(ValorFiltro)
   MetodoFiltroPasa(FiltroPasoAlto);
  else
   MetodoFiltroPasa(FiltroPasoBajo);
}
```

Código 5. Método que elige qué filtro aplicar entre filtro pasa alto o filtro pasa bajo.

TecnicasBasadasGradiente()

En este método aplica un filtro de valores de gris debido a un umbral proporcionado por el usuario. Este método se utiliza para aplicar el filtro de Sobel y el filtro de Prewitt, lo que varía son las máscaras utilizadas para cada filtro (Código 6).

```
int Sobel1[3][3]={{-1,0,1},{-2,0,2},{-1,0,1}};//variables globales
int Sobel2[3][3]={{-1,-2,-1},{0,0,0},{1,2,1}};
int Prewitt1[3][3]={{-1,-1,-1},{0,0,0},{1,1,1}};
int Prewitt2[3][3]={{-1,0,1},{-1,0,1},{-1,0,1}};
void TfrmPrincipal::TecnicasBasadasGradiente(intPrimeraMascara[3]
[3],intSegundaMascara[3][3]){
int iSumaPrimeraMascara=0,iSumaSegundaMascara=0,iSumaMascara=0;
PrepararBarraProceso();
for(int i=0;i<=(ANCHOIMAGEN-MASCARATAM);i++){
```

Código 6. Método que ejecuta el filtro de Sobel o de Prewitt dependiendo de la máscara recibida.

```
   for(int j=0;j<=(ALTOIMAGEN-MASCARATAM);j++){
    for(int k=0;k<MASCARATAM;k++){
     for(int z=0;z<MASCARATAM;z++){
      SumaPrimeraMascara+=(MatrizGeneralEscalaGrises[i+k][z+j])*
      (PrimeraMascara[k][z]);
      SumaSegundaMascara+=(MatrizGeneralEscalaGrises[i+k][z+j])
      *(SegundaMascara[k][z]);
      }
     }
     SumaMascara = abs(SumaPrimeraMascara+SumaSegundaMascara);
     FuncionValidaDatosdelUmbral();
     if(SumaMascara>Umbral)
      SumaMascara=255;
     else
      SumaMascara=0;
     MatrizGeneralEscalaGrises[i][j] = SumaMascara;
     SumaPrimeraMascara=0;SumaSegundaMascara=0;
    }
   }
  }
```

Código 6. Método que ejecuta el filtro de Sobel o de Prewitt dependiendo de la máscara recibida (continuación).

MetodoRoberts()

Es uno de los métodos utilizados para la detección de bordes, el cual utiliza dos máscaras, una máscara realza las líneas horizontales y la otra máscara las líneas verticales (Código 7).

```
void TfrmPrincipal::MetodoRoberts(void){
  int Roberts1[2][2]={{1,0},{0,-1}};
  int Roberts2[2][2]={{0,1},{-1,0}};//Máscara de Roberts
  int iSumaPrimeraMascara=0,iSumaSegundaMascara=0,iSumaMascara=0;
  for(int i=0;i<=(ANCHOIMAGEN-(MASCARATAM-1));i++){
   for(int j=0;j<=(ALTOIMAGEN-(MASCARATAM-1));j++){
    for(int k=0;k<(MASCARATAM-1);k++){
     for(int z=0;z<(MASCARATAM-1);z++){
      iSumaPrimeraMascara+=(MatrizGeneralEscalaGrises[i+k][z+j])
      *(Roberts1[k][z]);
      iSumaSegundaMascara+=(MatrizGeneralEscalaGrises[i+k][z+j])
      *(Roberts2[k][z]);}
    }
    iSumaMascara = abs(iSumaPrimeraMascara+iSumaSegundaMascara);
    FuncionValidaDatosdelUmbral();
    if(iSumaMascara>Umbral) iSumaMascara=255;
    else
     iSumaMascara=0;
    MatrizGeneralEscalaGrises[i][j] = iSumaMascara;
    iSumaPrimeraMascara=0;iSumaSegundaMascara=0;}
  }
}
```

Código 7. Filtro de Roberts a partir de una imagen en escala de gris.

FiltroFreiChen()

Este método también es nombrado por otros autores como operador isotrópico, utiliza dos máscaras una para resaltar los bordes horizontales y la otra para los bordes verticales (Código 8).

```
void TfrmPrincipal::FiltroFreiChen(){
  float Freichen1[3][3]={{-1,0,1},{-1.41,0,1.41},{-1,0,1}};
  float Freichen2[3][3]={{-1,-1.41,-1},{0,0,0},{1,1.41,1}};
  float iSumaPrimeraMascara=0,iSumaSegundaMascara=0,iSumaMascara=0;
  for(int i=0;i<=(ANCHOIMAGEN-MASCARATAM);i++){
   for(int j=0;j<=(ALTOIMAGEN-MASCARATAM);j++){
    for(int k=0;k<MASCARATAM;k++){
     for(int z=0;z<MASCARATAM;z++){
      iSumaPrimeraMascara+=(MatrizGeneralEscalaGrises[i+k][z+j])
      *(Freichen1[k][z]);
      iSumaSegundaMascara+=(MatrizGeneralEscalaGrises[i+k][z+j])
      *(Freichen12[k][z]);
     }
    }
    iSumaMascara = abs(iSumaPrimeraMascara+iSumaSegundaMascara);
    FuncionValidaDatosdelUmbral();
    if(iSumaMascara>Umbral)
     iSumaMascara=255;
    else
     iSumaMascara=0;
    MatrizGeneralEscalaGrises[i][j] = iSumaMascara;
    iSumaPrimeraMascara=0;iSumaSegundaMascara=0;
   }
  }
 }
```

Código 8. Filtro de Frei-Chen o isotrópico.

MetodoLaplaciano()

Este método se utiliza con una sola máscara, para realzar los bordes en la imagen, la operación es similar al filtro de Sobel (Código 9).

```
void TfrmPrincipal::MetodoLaplaciano(void){
  int Laplaciano2[3][3]={{-1,-1,-1},{-1,8,-1},{-1,-1,-1}};
  for(int i=0,iSumaTotal=0;i<=(ANCHOIMAGEN-MASCARATAM);i++){
   for(int j=0;j<=(ALTOIMAGEN-MASCARATAM);j++){
    for(int k=0;k<MASCARATAM;k++){
     for(int z=0;z<MASCARATAM;z++){
      iSumaTotal+=(MatrizGeneralEscalaGrises[i+k][z+j])
      *(Laplaciano2[k][z]);
     }
    }    iSumaTotal = abs(iSumaTotal);
    MatrizGeneralEscalaGrises[i][j] = iSumaTotal/9;
    iSumaTotal=0;
   }
  }
}
```

Código 9. Método Laplaciano a partir de una imagen en escala de gris.

Abrir1Click()

Al dar clic al menú `Imagen` y seleccionar la opción `abrir` se muestra un cuadro de diálogo para abrir una imagen y mostrarla en el módulo de preprocesamiento (Código 10).

```
void __fastcall TfrmPrincipal::Abrir1Click(TObject *Sender){
  char* cRutaImagen;
  int l=0,k=-1,i;
  posicionDistancia=0;posicion=0;pos=0;indiceTam=0;
  ////////////////////////INICIO  abrir imágenes BMP ///////////////
  imagenbmp = new  Graphics::TBitmap;  //imagen
  if(CuadroDialogoAbrirImagen->Execute()){
   try{
    imagenbmp->LoadFromFile(CuadroDialogoAbrirImagen->FileName);
    ImagenOriginal->Picture->Graphic = imagenbmp;
    ImagenProcesamiento->Picture->Graphic = imagenbmp;
    ImagenAuxiliar->Picture->Graphic = imagenbmp;
    ASRutaImagen=CuadroDialogoAbrirImagen->FileName.c_str();
    cRutaImagen = CuadroDialogoAbrirImagen->FileName.c_str();
    for(i=0;cRutaImagen[i]!='.';i++);l=i;
     for(int j=l;cRutaImagen[j]!='\\';j--,k++);
      ASNombreClase=ASRutaImagen.SubString((l-k)+1,k);
      cRutaImagen=ASNombreClase.c_str();
      for(i=0;isalpha(cRutaImagen[i]);i++);
       ASNombreClase = ASNombreClase.SubString(1,i);
       bImagenCargada = true;
   }
   catch(const Exception & E){
    ShowMessage("error " + E.Message);bImagenCargada = false;
   }
  }
  else{
   ShowMessage("cancelado por el usuario");
   bImagenCargada = false;
  }
  ////////////////////////FIN  abrir imagenes BMP //////////////////
 if(bImagenCargada){
  ActivarMenuImagen(true,false,true,true,true,true);
  ActivarMenuEditar(true,false,false,false,false);
  ActivarMenuDeteccionBordes(false,false,false,false,false);
  ActivarComponentesPanelPreprocesamiento(true,false,
  false,false,false,true);
  Label1->Visible=false;
  Image6->Visible=false;
  Image7->Visible=false;
 }
}
```

Código 10. Método para realizar la apertura de una imagen.

Método medir distancias

El proceso de medir las distancias sobre la imagen del rostro, se ejecuta en el evento `OnDragDrop` (Código 11) y `OnDragOver` (Código 12) del componente 2 del módulo

de preprocesamiento (Fig. C.6) componente `TImage` nombrado como `ImagenProcesamiento`).

```
void __fastcall TfrmPrincipal::ImagenProcesamientoDragDrop(TObject
*Sender,TObject *Source, int X, int Y){
iVectorPuntoX[iPosicion]=X;
iVectorPuntoY[iPosicion]=Y;
iPuntoInicial=0;
ImagenProcesamiento->Canvas->MoveTo(0,0);
ImagenProcesamiento->Canvas->MoveTo(iVectorPuntoX[iPosicion-1]-
1,iVectorPuntoY[iPosicion-1]-2);
ImagenProcesamiento->Canvas->Pen->Color=clBlue;
ImagenProcesamiento->Canvas->LineTo(X-1,Y-2);
fDistanciaX = pow(fabs(iVectorPuntoX[iPosicion-1]-
iVectorPuntoX[iPosicion]),2);
fDistanciaY = pow(fabs(iVectorPuntoY[iPosicion-1]-
iVectorPuntoY[iPosicion]),2);
fDistancia = sqrt(fDistanciaXfDistanciaY);
if(iPosicionDistancia<=16){
   fDistanciasEuclidianas[iPosicionDistancia++]=fDistancia;
   if(iPosicionDistancia>16){
     ImagenProcesamiento->Enabled = false;
     btnGuardarDistanciasP->Enabled = true;
   }
}
if((++iIndiceImagenMuestra)<=16&&iIndiceImagenResultado<=16){
    ImagenMedirP->Picture-
>LoadFromFile("C:\\Rostros\\"+nombreImagenes[iIndiceImagenMuestra])
;
ImagenResultado->Picture-
>LoadFromFile("C:\\Rostros\\"+nombreResultadoMedirImagen[iIndiceIma
genResultado++]);
}
iPosicion++;
}
```

Código 11. Evento `OnDragDrop` del componente 2 del módulo de preprocesamiento**.**

```
void __fastcall TfrmPrincipal::ImagenProcesamientoDragOver(TObject
*Sender,TObject *Source, int X, int Y, TDragState State, bool
&Accept){
   ImagenProcesamiento->DragCursor=crCross;
   if(iPosicionInicial==0){
      Edit1->Text= AnsiString(X);
      Edit2->Text= AnsiString(Y);
      VectorPuntoX[iPosicion]=X;
      VectorPuntoY[iPosicion]=Y;
      iPosicionInicial=-1;
      iPosicion++;}
    }
```

Código 12. Evento `OnDragOver` del componente 2 del módulo de preprocesamiento.

btnEntrenarNeuronaEClick()

El proceso de entrenamiento de los datos se realiza en el evento `OnClick` del componente 3 en el módulo entrenamiento figura C.22 (Código 13), este evento invoca a dos métodos principales `RedNeuronalArtificialPerceptron` (Código 14) y `ModificarPeso` (Código 15) que en conjunto realizan el entrenamiento de los datos.

```
void __fastcall TfrmPrincipal::btnEntrenarNeuronaEClick(TObject
*Sender){
   int
iError1,iError2,iError3,iError4,iContadorEpocas=0,iContadorError;
   AbrirDistanciaImagenArchivoTexto();
   for(int i=0;i<SALIDAS;i++){
        dVectorGananciasFinales[i]=fVectosGanancias[i];
      for(int j=0;j<DISTANCIAS;j++){
         dMatrizPesosFinales[i][j]=fMatrizPesos[i][j];
      }
      datosOperacionesChecar+="\n";
   }
   do{
     iContadorError=0;
     iContadorEpocas++;
     for(int i=0;i<iTotalRegistros;i++){//ciclo para registros
         iError1 = RedNeuronalArtificialPerceptron(0,i);//Neurona 1
         iError2 = RedNeuronalArtificialPerceptron(1,i);//Neurona 2
         iError3 = RedNeuronalArtificialPerceptron(2,i);//Neurona 3
         iError4 = RedNeuronalArtificialPerceptron(3,i);//Neurona 4
         if(iError1!=0){
            ModificarPeso(0,i,iError1);//Modificar Pesos Neurona 1
         }
         if(iError2!=0){
            ModificarPeso(1,i,iError2);//Modificar Pesos Neurona 2
         }
         if(iError3!=0){
            ModificarPeso(2,i,iError3);//Modificar Pesos Neurona 3
         }
         if(iError4!=0){
            ModificarPeso(3,i,iError4);//Modificar Pesos Neurona 4
         }
      if( ((fError1==0)&&(fError2==0))&& ((fError3==0)
       &&(iError4==0)) ){
           iContadorError++;}
     }//ciclo para registros
   }while(iContadorError!=iTotalRegistros);
   ShowMessage("La red neuronal artificial terrminó de entrenar");
   btnEntrenarNeuronaE->Enabled = false;
   btnGuardarEntrenamientoE->Enabled = true;
   btnMostrarDatosE->Enabled = false;
}
```

Código 13. Evento `OnClick` del componente 3 del módulo entrenamiento.

```
int TfrmPrincipal::RedNeuronalArtificialPerceptron(int iNeurona,
int iIndicei){
  int iError=0;
  float fHardlims=0;
  for(int i=0;i<DISTANCIAS;i++){//Inicio
  ciclo para hacer las operaciones con el vector pesos
  fHardlims+= (dVectorElementosP[iIndicei][i]
  *dMatrizPesosFinales[iNeurona][i]);//a = W_ant * P
  }// Fin ciclo para hacer las operaciones con el vector pesos
  fHardlims+=dVectorGananciasFinales[iNeurona];//a = W_ant*P+B_ant
  (fHardlims>=0)?fHardlims=1:fHardlims=-1;
  dError = iVectorSalidas[iIndicei][iNeurona]-fHardlims;//Error =
   t_en_neurona - a_n
  return (dError);
}
```

Código 14. Método `RedNeuronalArtificialPerceptron` que es invocado en el código 13.

```
void TfrmPrincipal::ModificarPeso(int iNeurona,int iIndicex,int
iError){
float Auxiliar=0;
for(int i=0;i<DISTANCIAS;i++){//Inicio ciclo
para hacer las operaciones con el vector pesos
 Auxiliar = iError*dVectorElementosP[iIndicex][i];
 dMatrizPesosFinales[iNeurona][i]= dMatrizPesosFinales
 [iNeurona][i]  + Auxiliar;// W_sig = W_ant + Error*P
}// Fin ciclo para hacer las operaciones con el vector pesos
dVectorGananciasFinales[iNeurona]=
dVectorGananciasFinales[iNeurona]+ iError; // B_sig = B_ant + Error
}
```

Código 15. Método `ModificarPeso` que es invocado en el código 13.

Método de Reconocimiento

El método de reconocimiento se realiza con dos eventos principalmente el evento `OnClick` del componente 4 y el evento `OnClick` del componente 5 en el módulo de reconocimiento del sistema (Fig C.24), es decir, en el proceso de reconocimiento se efectúan varias acciones, abrir la imagen que se quiere reconocer, abrir el archivo donde se guardó el resultado con los datos entrenados el cual contienen los pesos y ganancias de cada neurona y finalmente ejecutar el proceso de reconocer la imagen, código 16 y 17 respectivamente.

```
void    __fastcall   TfrmPrincipal::btnReconocerImagenRClick(TObject
*Sender){
   string sLineArchivoDatos="";
   AnsiString ALinea="";
   int iIndice=-1;
   bool bImagenEncontrada=false;
   if(AbrirArchivoDatosAprendizaje->Execute()){
      try{
          ifstream NombreArchivoEntrenamiento(AbrirArchivoDatos
            Aprendizaje->FileName.c_str(),ios::in);
          if(NombreArchivoEntrenamiento.is_open()){
           getline(NombreArchivoEntrenamiento,sLineArchivoDatos);
           while(!NombreArchivoEntrenamiento.eof()&&
             !bImagenEncontrada){//inicio del ciclo while
            getline(NombreArchivoEntrenamiento,sLineArchivoDatos);
                 if( ( iIndice=sLineArchivoDatos.find(
                        ASalidasReconocimiento.c_str()) ) > 0){
                    ALinea = sLineArchivoDatos.begin();
                    nombreReconocido = ALinea.SubString(1,(sLi
                  neArchivoDatos.find(",")));
                    try{
                        imagenbmpReconocimiento = new  Graphics
                         ::TBitmap;  //variable imagen
                        imagenbmpReconocimiento->LoadFromFile("C:\\
                       Reconocimiento\\ImagenesReconocer\\
                       "+ nombreReconocido+"1.bmp");
                        ImagenReconocida->Picture->
                       Graphic = imagenbmpReconocimiento;
                        ImagenReconocida->Visible = true;
                        GroupBox5->Visible = true;
                        bImagenEncontrada=true;
                    }
                    catch( const Exception & E){
                        ShowMessage("Error " + E.Message);
                    }
                }
             }//fin del ciclo while
          }
          else ShowMessage("Error al abrir el archivo distancias");
          NombreArchivoEntrenamiento.close(); //cerrando archivo
      }
      catch( const Exception & E){
           ShowMessage("Error");
      }
   }
   else{
       ShowMessage("Cancelado por el usuario");
   }
   asSaidasRecono = "";
 }
```

Código 16. Evento `OnClick` del componente 4 en el módulo de reconocimiento.

```
void __fastcall TfrmPrincipal::btnReconocerImagenRClick(TObject
*Sender){
string sLineArchivo; string sLineArchivoDatos;
char *cPunteroPrimeraLinea="";
AnsiString prueba="",aux="";
if(AbrirArchivoDatosAprendizaje->Execute()){
 try{
  ifstream NombreArchivoEntrenamiento(AbrirArchivoDatosAprendizaje-
  >FileName.c_str(),ios::in);
  boolbandera = false;

 if(NombreArchivoEntrenamiento.is_open()){
      getline(NombreArchivoEntrenamiento,sLineArchivoDatos);
   while(!NombreArchivoEntrenamiento.eof()&&!bandera){//inicio del
   ciclo while
    getline(NombreArchivoEntrenamiento,sLineArchivoDatos);
    cPunteroPrimeraLinea = sLineArchivoDatos.begin();
    prueba = sLineArchivoDatos.begin();
    int indx=prueba.Length();
    int iIndice=indx,contuno=0;
    int iIndicesalida = (asSaidasRecono.Length())-1;
    char* auxiliarpunterolinea;
    char* auxsalidareco = asSaidasRecono.c_str();
    prueba = prueba.SubString( 1,indx-1);
    indx=prueba.Length()-1;
    iIndice=indx;
    auxiliarpunterolinea =  prueba.c_str();
    for(int iIndiceAux=indx,iIndiceDos=0;iIndice>0&&!bandera ;
    iIndice--){
     if(auxiliarpunterolinea[iIndice]==auxsalidareco[iIndicesalida
     --]){
      contuno++;
     }
     else{
      if(asSaidasRecono.Length()==contuno){
       bandera=true;
       for(int iIndice=1,iIndiceAux=1;cPunteroPrimeraLinea[iIndice]
       !='\0'||prueba.IsDelimiter(",",iIndice);iIndice++){
        if(prueba.IsDelimiter(",",iIndice)){
         nombreReconocido = prueba.SubString( iIndiceAux,iIndice-
         iIndiceAux);
         try{
          imagenbmpReconocimiento = new  Graphics::TBitmap;
          imagenbmpReconocimiento->LoadFromFile("C:\\Imagenes
          rostros \\"+nombreReconocido+"1.bmp");
          ImagenReconocida->Picture->Graphic =
          imagenbmpReconocimiento;
          ImagenReconocida->Visible = true;
          GroupBox5->Visible = true;
         }
         catch(const Exception & E){
           ShowMessage("error " + E.Message);
           bImagenCargadaReconocimiento = false;
         }
         break;
```

Código 17. Evento `OnClick` del componente 5 en el módulo de reconocimiento.

```
            }
          }//fin del for
         }
         else
          break;
         }//fin del else
        }//fin del for
       }//fin del ciclo while
      }//fin del if
      else ShowMessage(" Error al abrir el archivo Distancias.txt ");
      NombreArchivoEntrenamiento.close(); //cerrando archivo
     }
     catch( constException& E){
      ShowMessage("error ");
     }
    }
else{
ShowMessage("cancelado por el usuario");
    }
asSaidasRecono = "";
}
```

Código 17. Evento `OnClick` del componente 5 en el módulo de reconocimiento (continuación).

ANEXO E. CONTENIDO DEL CD

A continuación se describe el contenido del CD-ROM que acompaña a este trabajo de tesis, en la figura D.1 se muestra su estructura.

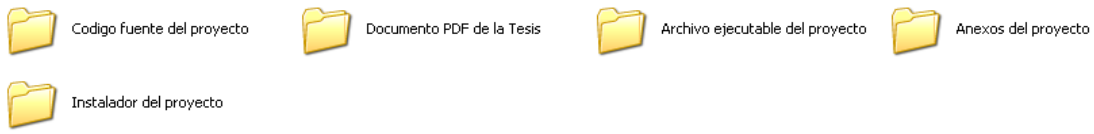


Figura D.1. Estructura del CD-ROM.

Código fuente del proyecto

Esta carpeta contiene todos los archivos del proyecto, en la tabla XXI se muestran los tipos de archivos que contiene la carpeta y en la figura D.2 se muestran los archivos almacenados en la carpeta.

Tabla XXI. Tipos de extensiones del proyecto del sistema.

EXT	DESCRIPCIÓN
BPR	Es el archivo makefile del proyecto. Define qué y cómo se debe compilar.
CPP	Archivos fuente de C++.
H	Archivos de cabecera de C++.
OBJ	Archivos objetos resultados de la compilación.
EXE	Archivo ejecutable del proyecto.
TDS	Archivos temporales para la compilación incremental.
DFM	Archivos de descripción de formulario. Contiene los valores de las propiedades de cada componente.
RES	Archivo de recursos.

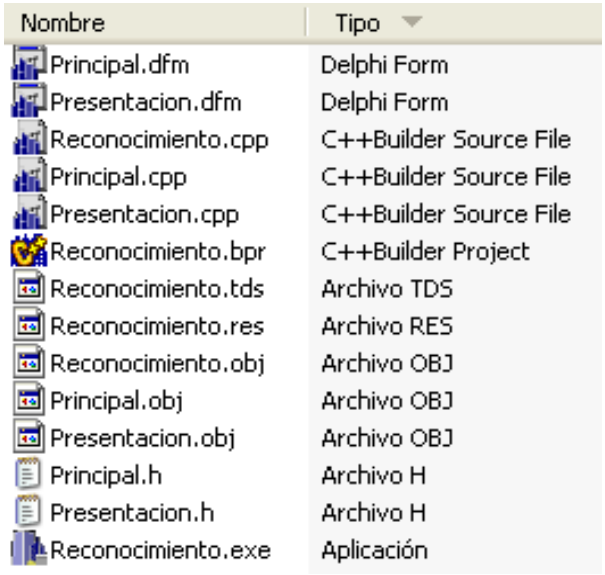


Figura D.2. Contenido de la carpeta `código fuente del proyecto`.

Documento PDF de la tesis

Esta carpeta contiene el documento en formato PDF de la tesis (Fig. D.3).

Figura D.3. Documento PDF de la tesis.

Archivo ejecutable del sistema

Figura D.4. Archivo ejecutable de la aplicación.

Anexos del proyecto

Esta carpeta contiene sub carpetas que son los anexos al proyecto, forman parte del capítulo 4 (Fig. D.5).

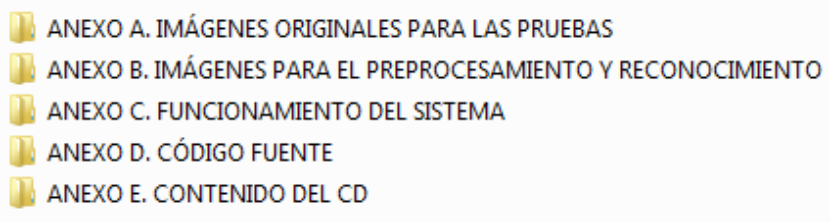


Figura D.5. Contenido de la carpeta `Anexos del proyecto`.

Instalador del proyecto

En esta carpeta se encuentran diferentes archivos, SETUP.exe es el archivo de instalación (Fig. D.6).

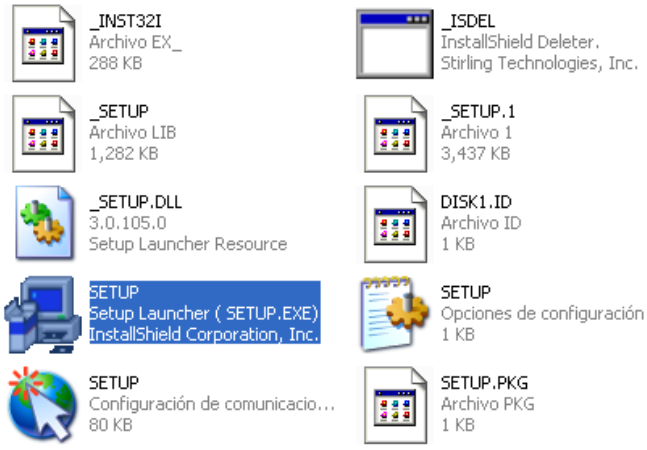


Figura D.6. Archivos de instalación.

Nota: para instalar correctamente el proyecto se deben seguir los pasos que muestra el propio instalador.

REFERENCIAS

Blauden electronics 2008, *Qué es bluetooth*, BlueZona, consultado el 10 de octubre de 2010, <*www.bluezona.com/que-es-bluetooth/*>.

De la Escalera, A 2009, *Visión por Computador: Fundamentos y Métodos*, Pearson, España.

Diccionario de la lengua española 1997, Océano, España.

Franco, P 2011, 'Restauración de imágenes con degradación por borrosidad empleando técnicas de procesamiento digital de imágenes', tesis de licenciatura, Universidad del Mar campus Puerto Escondido, México.

González, RC & Woods, RE 1992, *Tratamiento digital de imágenes*, Addison-Wesley/Diaz de Santos, Estados Unidos de América.

Kuri, AF 2002, 'Hollistic Face Recognition through Multivariate Analysis and Genetic Algorithms', *Procceeding of the 4th Asia Pacific Conference on Simulated Evolution and Learning*, Vol.2, Singapore, pp. 560-566.

Landesa, I & Alba, JL 2007, 'Detección de caras y localización de características faciales', *XXII Simposium Nacional de la Unión Científica Internacional de Radio*, URSI 2007, España.

Larousse enciclopédico universal 2000, Larousse, Colombia.

Martín, B & Sanz, A 2007, *Redes Neuronales y Sistemas Borrosos*, 3ra ed, Alfaomega RA-MA, España.

Microsoft 2010, *What is Surface*, Microsoft corporation, consultado el 10 de octubre de 2010, <http://www.microsoft.com/surface/>.

Moreno, J, Gómez, FJ, Fernández, MA & Fernández, A 1999, 'Reconocimiento de Rostros Utilizando Secuencias de Histogramas como Tramas Espacio-Temporales', *IV Simposio Iberoamericano de Reconocimiento de Patrones,* SIARP'99, Cuba, pp. 465-474.

Nilsson, N 2001, *Inteligencia Artificial: una nueva síntesis*, McGraw-Hill, España.

Norton, P 1999, *Introducción a la Computación,* 3ra ed, McGraw-Hill, México.

Pajares, G & De la Cruz, JM 2008, *Visión por computador: imágenes digitales y aplicaciones*, 2da ed, Alfaomega RA-MA, España.

Pajares, G & Santos, M 2006, *Inteligencia Artificial e Ingeniería del Conocimiento*, 1ra ed, Alfaomega RA-MA, España.

Pressman, RS 2005, *Ingeniería del software: un enfoque práctico*, 6a ed, McGraw Hill, México.

Russell, S & Norving, P 2004, *Inteligencia artificial un enfoque moderno*, 2da ed, Prentice Hall, España.

Sosa, D 2010, 'Simulación de dispersión de contaminantes a futuro en las playas de Puerto Escondido, Oaxaca', tesis de licenciatura, Universidad del Mar campus Puerto Escondido, México.

Toscano, JH 2009, 'Detección y seguimiento lineal de objetos no flexibles invarantes en color en una secuencia de imágenes', tesis de licenciatura, Universidad del Mar campus Puerto Escondido, México.

Villa, SM 2007, ‘Sistema de reconocimiento de rostros’, *XIV Congreso internacional de ingeniería electrónica, eléctrica y de sistemas,* INTERCON 2007, Perú.