Universidad Nacional del Litoral

Facultad de Ingeniería y Ciencias Hídricas

Proyecto Final de Carrera - Ingeniería en Informática

SISTEMA DE RECONOCIMIENTO FACIAL

Autor: Scarel, Germán Matías
Director: Bioing. Martínez, Cesar
Co-Director: Dra. Stegmayer, Georgina
Asesor Temático: Ing. Müller, Omar

Santa Fe - 2010

sinc(*i*) Research Institute for Signals, Systems and Computational Intelligence (fich.unl.edu.ar/sinc)
G. M. Scarel, C. E. Martínez & G. Stegmayer; "Sistema de reconocimiento facial (Undergraduate project)"
Facultad de Ingeniería y Ciencias Hídricas - Universidad Nacional del Litoral, 2010.

Resumen

El reconocimiento facial es una tarea que ha recibido recientemente una atención importante por parte de muchas áreas de investigación como procesamiento de imágenes, extracción de características y otras. Este interés se debe a que se pretende emular el proceso interno que realiza el cerebro humano al reconocer a las personas. En este trabajo se propone un sistema simple que intenta imitar este proceso interno mediante tres etapas básicas que conforman un sistema automático de reconocimiento facial: detección del rostro, extracción de características y clasificación. Primero, se lleva a cabo un análisis y documentación del sistema, identificando los requerimientos del mismo y realizando los diagramas necesarios para el desarrollo. Luego, se presenta la interfaz gráfica desarrollada para la interacción con el usuario, con el módulo inicial de captura de imágenes. Se describe la base de datos de rostros creada a partir de un conjunto de imágenes capturadas, y se explican las técnicas aplicadas en cada etapa del reconocimiento. Para la detección del rostro se utilizó un algoritmo basado en una representación global del mismo combinando segmentación de piel, aplicación de plantillas y detección de ojos. En la extracción de características se aplicó la técnica de eigenfaces para obtener una representación alternativa de las imágenes faciales, logrando reducir el espacio dimensional, implicando a su vez un bajo costo computacional. Estas representaciones fueron clasificadas por medio de una red neuronal, más precisamente un perceptrón multicapa, identificando a la persona en cuestión. Finalmente, se exponen los experimentos realizados y se muestran los resultados obtenidos donde puede verse cómo el perceptrón multicapa mejora el desempeño frente a otros clasificadores.

sinc(*i*) Research Institute for Signals, Systems and Computational Intelligence (fich.unl.edu.ar/sinc)
G. M. Scarel, C. E. Martínez & G. Stegmayer; "Sistema de reconocimiento facial (Undergraduate project)"
Facultad de Ingeniería y Ciencias Hídricas - Universidad Nacional del Litoral, 2010.

Prefacio

La robustez del cerebro humano para reconocer rostros juega un rol importante en la vida social de toda persona. Esta habilidad humana nos hace capaces de reconocer miles de rostros e incluso transcurrido el tiempo, soportando diferentes poses, expresiones faciales, puntos de vistas, oclusiones parciales y diversidad de fondos. Sin embargo, construir un sistema que emule el proceso interno del cerebro es una tarea compleja.

En este trabajo se desarrolló un sistema de reconocimiento facial completo aplicando las técnicas existentes para las distintas etapas del reconocimiento, que permita identificar una persona en un ambiente real. Para explicar detalladamente el desarrollo del sistema, el documento se organizó en seis capítulos, como se explica a continuación. El Capítulo 1 introduce al lector en los sistemas de reconocimiento facial. Explica las técnicas más destacadas en cada etapa, su evolución y realiza una descripción general de la propuesta, objetivos y alcances del trabajo. En el Capítulo 2 se desarrollan los conceptos teóricos de las técnicas de procesamiento digital de imágenes utilizados en el proyecto. Se expone la idea del Análisis de Componentes Principales como también se introducen los métodos de clasificación de patrones. En el Capítulo 3 se realiza un análisis y diseño del sistema. Se especifican los requerimientos funcionales y no funcionales, se modela la vista de diseño estática y dinámica del sistema, y se presenta la interfaz de usuario desarrollada. En el Capítulo 4 se expone la implementación del sistema llevada a cabo. Se describe la base de datos de rostros creada a partir de un conjunto de imágenes, y se comentan las técnicas aplicadas en cada paso del reconocimiento. Luego, en el Capítulo 5, se presentan diversas pruebas del sistema con la base de rostros generada. Se verifica el desempeño del clasificador empleado

y se compara con otros tipos de clasificadores, finalizando con la discusión de resultados. Finalmente, se exponen las conclusiones y desarrollos futuros en el Capítulo 6.

Índice general

sinc(*i*) Research Institute for Signals, Systems and Computational Intelligence (fich.unl.edu.ar/sinc)
G. M. Scarel, C. E. Martínez & G. Stegmayer; "Sistema de reconocimiento facial (Undergraduate project)"
Facultad de Ingeniería y Ciencias Hídricas - Universidad Nacional del Litoral, 2010.

Índice de figuras

Índice de tablas

sinc(*i*) Research Institute for Signals, Systems and Computational Intelligence (fich.unl.edu.ar/sinc)
G. M. Scarel, C. E. Martínez & G. Stegmayer; "Sistema de reconocimiento facial (Undergraduate project)"
Facultad de Ingeniería y Ciencias Hídricas - Universidad Nacional del Litoral, 2010.

CAPÍTULO 1

Introducción

1.1 Estado del arte

El reconocimiento facial (RF) automatizado por computadora es un área de investigación en pleno desarrollo. Identificar a una persona por medio de una imagen de su rostro implica emular el proceso cognitivo que realiza un ser humano al reconocer a sus pares. Aunque existen trabajos en el campo de la psicofísica y la neurociencia, aún no se sabe con certeza cómo funcionan estos procesos internamente en el cerebro humano [ZCPR03].

Independientemente de la técnica utilizada para la solución, el RF requiere de tres etapas: 1) la detección del rostro en una imagen, 2) la extracción de características y 3) la identificación y/o verificación de la cara mediante la clasificación de las características. En la identificación, el sistema informa la identidad de la persona, mientras que en la verificación confirma o rechaza ésta. Cada una de estas áreas ha sido objeto de investigaciones, dando origen a diferentes técnicas para resolver cada problema. A continuación se hará una breve descripción de las más usadas y otras recientemente propuestas.

1.1.1 Detección del rostro

La detección implica encontrar las áreas dentro de una imagen que contienen un rostro. Básicamente se trata de descartar todo lo que sea fondo, y así obtener la ubicación y tamaño exacto de la cara.

Uno de los primeros métodos para resolver este problema fue el de plantillas deformables [YHC92]. Cada característica del rostro (ojos y boca en este caso) es representada por medio de una plantilla. La idea general es tener una función de energía que relacione las intensidades de la imagen (valles y picos) con las propiedades de la plantilla. Una vez ubicada la plantilla se ajustan sus parámetros para minimizar la función de energía.

Otros enfoques se vuelcan a la clasificación de patrones por medio de ejemplos [SP98]. Como sólo hay que detectar caras (sin identificar sujetos específicamente) se crean patrones para zonas que representan una cara y zonas que no. Generalmente la solución para encontrar patrones que no son rostros, es por medio de la modalidad denominada *"bootstrap"*, en la cual se comienza con patrones que son rostros, se pone en marcha el clasificador, y todos los falsos positivos se agregan como patrones de *no-rostro*. En una siguiente pasada, si aparecen nuevos falsos positivos, éstos se siguen agregando como patrones. Cabe destacar que el método de clasificación es independiente de las imágenes utilizadas como patrones. En [SP98] se utiliza una red neuronal artificial como clasificador, específicamente un perceptrón multicapa.

Los últimos trabajos se basan en el método propuesto en [VJ04], que alcanza las tasas de reconocimiento de los mejores sistemas, pero con la ventaja de obtener una alta velocidad de detección. Una de sus contribuciones son los clasificadores en cascada, que permiten deshacerse de las regiones de fondo lo antes posible. Varios clasificadores son utilizados en serie, los primeros descartan la mayoría de regiones que no son propensas a contener un rostro. Las regiones más prometedoras pasan a los clasificadores más complejos y que requieren de mayor tiempo de cálculo.

1.1.2 Extracción de características

La extracción de características se refiere a la obtención de propiedades o parámetros particulares de cada rostro para luego poder ser clasificados. Se pueden tomar tres enfoques diferentes: 1) un enfoque holístico, basándose en la imagen del rostro como un todo, 2) un enfoque mediante características

locales, dando mayor importancia a las diferentes partes del rostro, o 3) un enfoque híbrido basado en la idea de que el sistema de percepción humana combina características locales y globales para el reconocimiento [ZCPR03].

En el enfoque holístico, la técnica más influyente es la de las *eigenfaces* [TP91]. Se utiliza un análisis de componentes principales para representar la imagen completa del rostro en un espacio de dimensiones reducido. Esta técnica no necesita de grandes cantidades de fotos por sujeto, tiene una baja sensibilidad al ruido, y aunque presenta problemas para manejar las variaciones de iluminación, pueden realizarse pequeños cambios para mejorar la efectividad en dichas condiciones [BHK97].

Por otro lado, se encuentran los modelos basados en las características locales. Uno de los métodos de mayor éxito es el de *Elastic Bunch Graph Matching* [WFKM97]. Aquí, los rostros son representados en forma de grafos. Los puntos importantes en la cara que conforman los nodos del grafo (como los ojos o la nariz), son descritos por un conjunto de onditas de Gabor de diferentes escalas y rotaciones, logrando robustez para cambios en iluminación, traslación, distorsión, rotación y escalado [ZJN07].

Los modelos híbridos son adaptaciones del método de las *eigenfaces* aplicado a características locales [PMS94]. Así, se combinan ventajas de los métodos holísticos y las características locales.

1.1.3 Reconocimiento

El reconocimiento consiste en clasificar las características extraídas de cada rostro. Esta clasificación puede ser realizada de manera *supervisada*, en la cual un patrón de entrada es identificado como miembro de una clase predefinida, o de manera *no supervisada*, donde el patrón es asignado a una clase desconocida. Aquí, cada clase es un sujeto, por lo tanto, al clasificar las características se está indicando a qué sujeto pertenecen.

Para el diseño de clasificadores se pueden distinguir tres aproximaciones basadas en: 1) concepto de similaridad, 2) aproximación probabilística y 3) optimización de un criterio de error [JDM00].

Las aproximaciones basadas en el concepto de similaridad son las más simples e intuitivas, donde los patrones similares son asignados a la misma clase. Para ello, se establece una métrica que define la similaridad y luego se clasifica por medio de plantillas o mínima distancia usando uno o varios prototipos por clase. La técnica de *eigenfaces* original [TP91] aplica la regla del vecino más cercano, utilizando como métrica la distancia Euclídea, donde

cada prototipo es la media de los patrones de entrenamiento.

En el enfoque probabilístico, se utilizan los conceptos de la teoría de la decisión estadística para establecer los bordes de decisión de las diferentes clases. Se asume que las características, que representan a un patrón, tienen una función de densidad de probabilidad condicionada a la clase. Así, un vector patrón $\mathbf{x}$ perteneciente a la clase ω_i es visto como una observación aleatoria desde la función de probabilidad $p(\mathbf{x}|\omega_i)$. Las reglas de decisión más conocidas son la *regla de decision de Bayes* y la *regla de máxima probabilidad*.

Por último, la tercer aproximación se basa en construir los bordes de decisión optimizando algún criterio de error. El objetivo es minimizar el error de clasificación entre la respuesta deseada y la salida del clasificador. Un ejemplo de este tipo de clasificadores son las *redes neuronales* [JMM96], las cuales pueden considerarse como sistemas distribuidos y paralelos que consisten de pequeñas unidades de procesamiento masivamente conectadas. Están conformadas por redes de grafos ponderados donde los nodos son las neuronas artificiales y los bordes (con pesos) son las conexiones entre las neuronas de entrada y salida de la red. Para lograr una clasificación adecuada son entrenadas con un algoritmo de entrenamiento a partir de un conjunto de datos. Los tipos de redes más comúnmente usados son las redes hacia adelante, como por ejemplo, el perceptrón multicapa.

1.2 Propuesta del proyecto

La idea central de este proyecto consiste en realizar el diseño y la implementación de un sistema integral para identificación de personas mediante el reconocimiento del rostro. El ambiente de aplicación del sistema será el Centro de Investigación y Desarrollo en Señales, Sistemas e Inteligencia Computacional (*sinc(i)*) de la FICH, UNL y el Centro de Investigación y Desarrollo de Ingeniería en Sistemas de Información (CIDISI) de la UTN-FRSF.

Para ello, se desarrolló una interfaz de usuario y se instaló una cámara web en una PC de escritorio en ambos centros de investigación para la captura de imágenes. Con el conjunto de imágenes obtenidas se realizó una clasificación manual de las mismas conformando una base de rostros para las pruebas del sistema.

Ya con la base de rostros, se desarrolló el sistema en base a las tres etapas básicas de todo sistema de reconocimiento facial:

1. Localización del rostro: el cual ubica la cara en la imagen aplicando el algoritmo propuesto en [ML06], basado en una segmentación de piel para detectar posibles regiones caras, y luego mediante combinación de plantillas decide por una región, recortándola como posible rostro.

2. Extracción de características: en esta etapa se obtienen las propiedades más relevantes del rostro mediante el método *eigenfaces*, obteniendo como resultado un vector característico que representa la imagen de entrada en un espacio dimensional reducido para su posterior clasificación.

3. Clasificación: aquí se diseñó una red neuronal para el reconocimiento de patrones, más especificamente un perceptrón multicapa. La red se entrenó mediante el algoritmo de retropropagación del error a partir de un conjunto de patrones de ejemplo. Con ésto, se logró que la red "aprenda" sobre el conjunto de potenciales usuarios.

Cabe destacar dos fases del proceso de construcción: una de entrenamiento y otra de clasificación. En la primera se ajustan varios parámetros del sistema, siendo el paso más importante el entrenamiento de la red para el conjunto de datos especificado. Una vez logrado ésto, se pasa a la fase de clasificación verificando el desempeño mediante la presentación de imágenes rostros de usuarios tanto conocidos como desconocidos.

1.3 Objetivos

Objetivos generales

- Desarrollar un prototipo de herramienta computacional de código abierto que permita realizar reconocimiento facial, basado en la utilización de técnicas del área de Procesamiento de Imágenes e Inteligencia Computacional.

- Aplicar los conocimientos adquiridos en el transcurso de la carrera a un proyecto que realice un aporte ingenieril a la comunidad en general.

- Colaborar en proyectos de investigación que requieran la participación de profesionales informáticos en el área de procesamiento de imágenes digitales e inteligencia artificial.

Objetivos específicos

- Implementar los módulos del localizador, normalizador de caras y extractor de características a partir de técnicas de procesamiento digital de imágenes.
- Diseñar e implementar el módulo del reconocedor mediante técnicas de la Inteligencia Computacional.
- Desarrollar un prototipo que permita la prueba del sistema de RF. El software ofrecerá las siguientes funcionalidades.
  - Tomar fotografías en vivo desde una cámara.
  - Registrar y almacenar individuos en el sistema, asociándole una o más imágenes capturadas.
  - Soportar el RF mediante la comparación de una fotografía actual con las registradas en el sistema. Se proveerán dos modos de funcionamiento: a) hacer corresponder la imagen tomada a la persona con la que más se ajuste de las conocidas en el sistema, y b) verificar la identidad del usuario por otro método (por ej. el ingreso del nombre y/o DNI por teclado).
  - Exportar e importar la información del individuo almacenada en el sistema. Ésto permitiría transportar su información para ser utilizada en otra ubicación.

1.4 Alcances

- La cantidad de usuarios se limitará a 11 en esta etapa de desarrollo del prototipo.
- No se hará rechazo de impostores, sino que se supondrá que todos los usuarios del sistema se encuentran registrados.
- No incluirá el desarrollo del módulo de administración de usuarios.

CAPÍTULO 2

Marco teórico

2.1 Procesamiento digital de imágenes

2.1.1 Imagen digital

Una imagen en escala de grises puede ser definida como una función bidimensional $f(x, y)$, donde x e y son coordenadas *espaciales*, y el valor de f en un par de coordenadas dado es denominado *intensidad* o *nivel de gris*. Cuando x, y y f son cantidades discretas, la imagen es llamada *imagen digital* [GW02]. De este modo, ésta puede ser reprensentada como una matriz de $M \times N$ elementos llamados *píxeles*.

$$f(x,y) = \begin{bmatrix} f(0,0) & f(0,1) & \dots & f(0,N-1) \\ f(1,0) & f(1,1) & \dots & f(1,N-1) \\ \vdots & \vdots & \ddots & \vdots \\ f(M-1,0) & f(M-1,1) & \dots & f(M-1,N-1) \end{bmatrix} \tag{2.1}$$

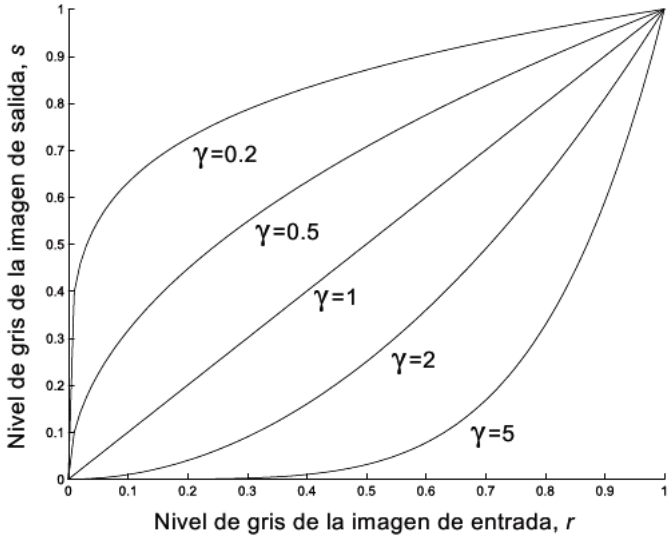


Figura 2.1: Gráfica de la transformación $s = cr^{\gamma}$ con $c = 1$ en todos los casos.

2.1.2 Corrección gamma

En el área de procesamiento de imágenes, uno se encuentra con la necesidad de mejorar una imagen con el objetivo de obtener otra más aceptable para una aplicación específica. Dependiendo del problema en sí, existen diversas técnicas para lograrlo, entre las cuales nos encontramos con las *transformaciones de potencia*. Estas transformaciones son útiles para la manipulación de contraste y tienen la forma básica

$$s = cr^{\gamma}, \tag{2.2}$$

donde r es el valor de entrada, s es el de salida y c y γ son constantes positivas. En la Figura 2.1 se muestran las transformaciones para diferentes valores de γ. Para $\gamma < 1$, ésta amplifica los valores oscuros de entrada y contrae los claros, mientras que lo opuesto ocurre para $\gamma > 1$.

Una variedad de dispositivos usados para la captura de imágenes responden de acuerdo a la ley de potencia. Por convención, el exponente en la ecuación es referido como *gamma*, y el proceso usado para la corrección de este fenómeno es llamado *corrección gamma*.

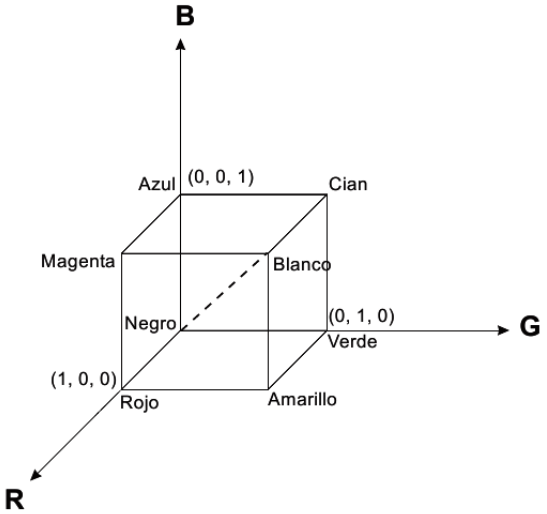


Figura 2.2: Modelo de color RGB.

2.1.3 Modelos de color

El uso del color en el procesamiento de imágenes es de importancia ya que suele simplificar la identificación y extracción de objetos en una escena. Un modelo de color especifica un sistema de coordenadas y un subespacio en el cual un color es definido por un punto. Muchos modelos están orientados al hardware como lo son el RGB (rojo, verde y azul), CMY (cian, magenta y amarillo) y el CMYK (cian, magenta, amarillo y negro); otros están orientados hacia aplicaciones donde se pretende manipular el color, tales como HSI (tono, saturación e intensidad), HSV (tono, saturación y valor) y el YCbCr (luminancia, crominancia azul y crominancia roja).

A continuación se dan las bases de los modelos empleados en este trabajo.

Modelo de color RGB

Este modelo se basa sobre un sistema de coordenadas cartesianas y el subespacio en el cual se define el color es el cubo normalizado entre [0,1] (Ver Figura 2.2). Los valores rojo (R), verde (G) y azul (B) están sobre los vértices (1,0,0), (0,1,0) y (0,0,1) respectivamente; el origen corresponde al negro, en el punto más lejano a éste se ubica el blanco, y el resto de los vértices corresponden a los colores secundarios: cian, magenta y amarillo. El

resto de colores está compuesto por la combinación de los colores primarios R, G y B, y son puntos dentro del cubo definido por un vector extendido desde el origen.

Las imágenes representadas en el modelo RGB consisten de tres planos, uno por cada color primario. El número de bits usados para representar cada píxel es llamado *profundidad*.

Modelo de color YCbCr

Éste es un modelo de color usado en sistemas de video y fotografía digital, donde Y es la componente de luminancia y Cb y Cr son las componentes de crominancia azul y roja.

YCbCr es el equivalente digital al modelo YUV (Sistema de TV). Concentra la mayor parte de la información de la imagen en la luminancia y menos en la crominancia. El resultado es que los elementos de YCbCr están menos correlacionados y pueden ser codificados por separado.

El paso de un sistema RGB a un YCbCr es obtenido mediante las ecuaciones:

$$\begin{array}{rcl} Y & = & 0{,}3R + 0{,}6G + 0{,}1B \\ Cb & = & \frac{B-Y}{2} + 0{,}5 \\ Cr & = & \frac{R-Y}{2} + 0{,}5. \end{array} \tag{2.3}$$

2.1.4 Procesamiento morfológico

El procesamiento morfológico es una herramienta que nos permite extraer componentes de una imagen que son útiles para la representación y descripción de regiones. Dos operaciones de interés en esta área son la dilatación y erosión.

Dilatación

Consideremos a las imágenes binarias como conjuntos de elementos en Z^2, donde cada elemento de un conjunto corresponde a un par de coordenadas (x, y) en la imagen. Sean A y B dos conjuntos en Z^2, la dilatación de A por B es denotada por $A \oplus B$ y se define como

$$A \oplus B = \left\{ z | (\hat{B})_z \cap A \neq \oslash \right\}, \tag{2.4}$$

donde $\hat{B}$ es la reflección de B alrededor de su origen. A la imagen B se la suele llamar *elemento estructurante*. Esta ecuación nos dice que la dilatación de A por B es el conjunto de todos los desplazamientos z, tal que $\hat{B}$ y A estén solapadas al menos por un elemento. Los desplazamientos son realizados en base al origen de $\hat{B}$.

Una de las aplicaciones de la dilatación es para "relleno de huecos" de tamaño igual o menor que el elemento estructurante.

Erosión

Al igual que en la dilatación, para conjuntos A y B en Z^2, la erosión de A por B se denota por $A \ominus B$ y es definida como

$$A \ominus B = \{z|(B)_z \subseteq A\}\,. \tag{2.5}$$

Ésto nos indica que la erosión corresponde a todos los puntos z tales que B, trasladado por z, es un subconjunto de A. Esta operación es la opuesta a la dilatación y su mayor uso es para eliminar detalles irrelevantes de tamaño menor o igual que B.

Extensión a imágenes en escala de grises

Aquí, las imágenes son representadas como $f(x, y)$ y $b(x, y)$, imagen de entrada y elemento estructurante respectivamente, donde se asume que ambas son funciones discretas.

La dilatación de f por b se define como

$$f \oplus b(s,t) = max\left\{f(s-x, t-y) + b(x,y) | (s-x), (t-y) \in D_f; (x,y) \in D_b\right\}, \tag{2.6}$$

donde D_f y D_b son el dominio de f y b, respectivamente. Las condiciones de que $(s - x)$ y $(t - y)$ deben estar en el dominio de f, y que x e y tienen que estar en el dominio de b, indican solapamiento de las dos funciones en al menos un elemento, al igual que en el caso binario.

El efecto logrado es una imagen de salida que tiende a ser más brillante que la de entrada, si todos los valores de los elementos estructurantes son positivos, y la eliminación o reducción de detalles oscuros dependiendo de los valores y forma de b.

La erosión es definida como

$$f \ominus b(s,t) = min\left\{f(s+x, t+y) - b(x,y) | (s-x), (t+y) \in D_f; (x,y) \in D_b\right\}, \tag{2.7}$$

donde D_f y D_b son el dominio de f y b, respectivamente. Las condiciones de que $(s+x)$ y $(t+y)$ tienen que estar en el dominio de f, y x e y en el dominio de b, expresan que el elemento estructurante tiene que estar completamente contenido en la imagen erosionada, es decir, f.

Como en el caso binario, la imagen de salida tiende a ser más oscura que la imagen de entrada, si todos los valores del elemento estructurante son positivos. Los detalles brillosos en la imagen de entrada son reducidos si éstos son más pequeños en área que el elemento estructurante, con un grado de reducción dependiendo de los valores de grises circundantes a los detalles brillosos y por la forma y amplitud de los valores de b.

2.2 Análisis de las componentes principales aplicado a imágenes faciales

El Análisis de las Componentes Principales (PCA, del inglés *Principal Component Analysis*) es una técnica estadística que realiza una transformación lineal y ortogonal de los datos a un nuevo sistema de coordenadas tal que la máxima varianza queda proyectada sobre la primer coordenada, la segunda mayor varianza sobre la segunda coordenada, y así sucesivamente. Transforma un conjunto de variables correlacionadas en un conjunto de variables no correlacionadas, y simplifica la transformación encontrando las componentes más cercanas a las variables originales pero ordenadas en forma decreciente al orden de su varianza. Puede ser utilizada para reducir la dimensionalidad del conjunto de datos reteniendo aquellas características del conjunto que mayor contribución hacen a su varianza, manteniendo las primeras componentes principales de más bajo orden. Tales componentes frecuentemente contienen la mayor información del conjunto de datos [Jac91].

Para el caso de imágenes faciales, se pretende extraer las características más relevantes en una imagen facial, las cuales pueden o no estar relacionadas a las características faciales como boca, nariz, ojos, etc.. En términos matemáticos, se quiere encontrar las componentes principales de la distribución de caras, o lo que es lo mismo, los eigenvectores de la matriz de covarianza del conjunto de imágenes caras [TP91]. Estos eigenvectores pueden ser vistos como el conjunto de vectores que caracterizan la variación entre las imágenes faciales. A partir de ésto, cada cara puede ser representada exactamente como una combinación lineal de los eigenvectores. Además, cada rostro se puede

aproximar usando aquellos que tienen asociados los eigenvalores más altos, es decir, los que responden a la mayor varianza del conjunto de imágenes faciales.

2.2.1 Cálculo de eigenfaces

Sea $I(x, y)$ una imagen facial en escala de grises de $N \times N$. Una imagen puede ser representada como un vector de dimensión N^2, o equivalentemente, un punto en el espacio N^2-dimensional. Por lo tanto, un conjunto de imágenes es mapeado a una colección de puntos en un espacio inmenso, considerando el tamaño de las imágenes con las que se suele trabajar. De esta manera, un conjunto de imágenes faciales, dado que todas tienen un aspecto similar, se mapean en un subespacio del espacio de imágenes y así pueden ser descritas por un subespacio de menor dimensión. La idea es encontrar los vectores que mejor describan la distribución de imágenes cara dentro del espacio completo de imágenes. Estos vectores de longitud N^2, llamados *eigenfaces*, definen el subespacio de imágenes rostro, el cual se denomina *espacio de caras*.

Sea $\mathbf{\Gamma_1}, \mathbf{\Gamma_2}, \mathbf{\Gamma_3}, \ldots, \mathbf{\Gamma_M}$ un conjunto de imágenes faciales. La cara media del conjunto es definida por

$$\mathbf{\Psi} = \frac{1}{M} \sum_{n=1}^{M} \mathbf{\Gamma_n}. \tag{2.8}$$

Cada cara difiere de ésta por el vector $\mathbf{\Phi}_i = \mathbf{\Gamma}_i - \mathbf{\Psi}$. A partir de este conjunto de vectores, se busca un conjunto de M vectores ortonormales, $\mathbf{u}_n$, los cuales mejor describan los datos. El k-ésimo vector, $\mathbf{u}_k$, es seleccionado tal que

$$\lambda_k = \frac{1}{M} \sum_{n=1}^{M} \left({\mathbf{u}_k}^T \mathbf{\Phi}_n \right)^2 \tag{2.9}$$

es un máximo, sujeto a que los vectores $\mathbf{u}_k$ sean ortonormales. Los vectores $\mathbf{u}_k$ y los escalares λ_k son los eigenvectores y eigenvalores, respectivamente, de la matriz de covarianza

$$\begin{aligned} C &= \tfrac{1}{M} \textstyle\sum_{n=1}^{M} \mathbf{\Phi}_n {\mathbf{\Phi}_n}^T \\ &= AA^T, \end{aligned} \tag{2.10}$$

donde $A = [\mathbf{\Phi}_1\ \mathbf{\Phi}_2\ \mathbf{\Phi}_3\ \ldots\ \mathbf{\Phi}_M]$. La matriz C resultante es de $N^2 \times N^2$ y obtener N^2 eigenvectores y eigenvalores es un tarea costosa para el tamaño de imágenes comúnmente procesadas.

Una consideración a tener en cuenta es que si el número de puntos en el espacio de imágenes es menor que la dimensión del mismo ($M < N^2$), habrá solamente $M - 1$ eigenvectores significantes, el resto tendrá asociados eigenvalores iguales a cero. En este caso, se tienen M imágenes caras de dimensión N^2 y se pueden encontrar los eigenvectores N^2-dimensionales resolviendo los eigenvectores para una matriz de $M \times M$ y luego realizar una combinación lineal de las imágenes faciales $\mathbf{\Phi}_l$.

Consideremos los eigenvectores $\mathbf{v}_l$ de $A^T A$ tal que

$$A^T A \mathbf{v}_l = \mu_l \mathbf{v}_l. \tag{2.11}$$

Premultiplicando ambos lados por A, se tiene

$$A A^T A \mathbf{v}_l = \mu_l A \mathbf{v}_l. \tag{2.12}$$

De aquí se ve que $A\mathbf{v}_l$ son los eigenvectores de la matriz de covarianza C. Entonces, se construye la matriz de $M \times M$, $L = A^T A$, donde $L_{mn} = \mathbf{\Phi}_m^T \mathbf{\Phi}_n$, y se encuentran los M eigenvectores $\mathbf{v}_l$, de L. Estos vectores determinan la combinación lineal de las M imágenes faciales para obtener los eigenvectores

$$\begin{aligned} \mathbf{u}_l &= A\mathbf{v}_l \\ &= \textstyle\sum_{k=1}^{M} \mathbf{v}_{lk} \mathbf{\Phi}_k. \end{aligned} \tag{2.13}$$

De esta forma se reducen enormemente los cálculos del orden del número de píxeles en las imágenes (N^2) al orden del número de imágenes en el conjunto (M). Los eigenvalores asociados permiten ordenar los eigenvectores de acuerdo al aporte que hacen a la variación entre imágenes.

Proyección y reconstrucción de una imagen facial

Una imagen facial $\mathbf{\Gamma}$ es proyectada sobre el espacio de caras por medio de la operación

$$x_k = \mathbf{u}_k^T \left(\mathbf{\Gamma} - \mathbf{\Psi} \right), \qquad \text{para} \;\; k = 1, \ldots, M. \tag{2.14}$$

Los pesos x_k forman un vector $\mathbf{x}^T = [x_1 \; x_2 \; x_3 \; \ldots \; x_M]$ que describen el aporte de cada eigenface en representar la imagen cara, tratando a las eigenfaces como un conjunto base para las imágenes faciales.

La reconstrucción exacta de una cara se realiza a través de una combinación lineal de las eigenfaces ponderadas con los respectivos pesos x_k

$$\mathbf{\Gamma} = \sum_{k=1}^{M} x_k \mathbf{u}_k. \tag{2.15}$$

2.3 Reconocimiento de patrones

Un *patrón* es un *arreglo de características* que describen una señal, y es representado por un vector multidimensional. Una *clase* es un conjunto de patrones que comparten una propiedad en común. Las clases son denotadas por $\omega_1, \omega_2, \ldots, \omega_W$, donde W es la cantidad de clases. Las técnicas de reconocimiento de patrones son aquellas que asignan los patrones a sus respectivas clases teniendo en cuenta las características de éstos [GW02].

El reconocimiento de patrones se divide en dos áreas: *reconocimiento basado en la teoría de la decisión* y *reconocimiento estructural*. El primero trata con patrones representados por características cuantitativas, y el segundo con las relaciones estructurales de las características de los patrones.

Reconocimiento basado en la teoría de la decisión

Estos métodos se basan en *funciones de decisión*. Siendo $\mathbf{x} = (x_1, x_2, \ldots, x_n)^T$ un vector patrón y existiendo W clases $\omega_1, \omega_2, \ldots, \omega_W$, la idea es encontrar W funciones $d_1(\mathbf{x}), d_2(\mathbf{x}), \ldots, d_W(\mathbf{x})$ tal que si un patrón $\mathbf{x}$ pertenece a la clase ω_i, entonces

$$d_i(\mathbf{x}) > d_j(\mathbf{x}), \qquad j = 1, 2, \ldots, W; \qquad j \neq i \tag{2.16}$$

Es decir, un patrón $\mathbf{x}$ desconocido es evaluado en las W funciones de decisión, si el valor más alto es obtenido en la función d_i, entonces se dice que $\mathbf{x}$ pertenece a la clase ω_i.

2.3.1 Redes neuronales

Uno de los métodos basados en la teoría de la decisión es la red neuronal. *Una red neuronal es un procesador masivamente distribuido que tiene una propensión natural para almacenar el conocimiento experimental y ponerlo a disposición para su uso* [Hay99].

La idea de una red neuronal surge con el objetivo de modelar el comportamiento del cerebro humano al realizar una tarea, como es el reconocimiento de las personas, empleando una interconexión masiva de pequeñas unidades de procesamiento las cuales se denominan *neuronas* o *nodos*. Las redes neuronales se asemejan al cerebro en dos aspectos: 1) el conocimiento es adquirido

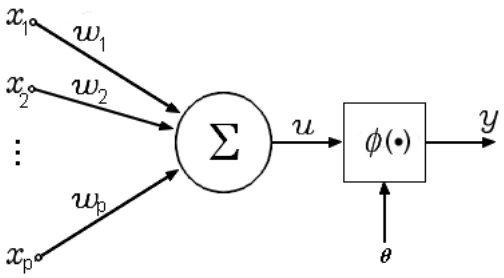


Figura 2.3: Arquitectura de un perceptrón simple.

por un proceso de aprendizaje y 2) los pesos de conexión entre neuronas, conocidos como *pesos sinápticos*, se usan para almacenar el conocimiento. El algoritmo de aprendizaje modifica los pesos sinápticos hasta lograr el conocimiento requerido.

La manera en la cual las neuronas de una red están estructuradas está ligada al algoritmo de aprendizaje usado para el entrenamiento de la red. Entre las arquitecturas posibles para modelos neuronales están las denominadas *redes hacia adelante*, como lo son el perceptrón simple, perceptrón multicapa y redes de función de base radial, las *redes recurrentes* y otros *modelos híbridos*.

A continuación se explicará el funcionamiento de un perceptrón simple, la arquitectura utilizada en este trabajo, el perceptrón multicapa, y se hará una introducción a las redes de función de base radial.

Perceptrón simple

El perceptrón es la forma más simple de red neuronal, consistiendo de una única neurona. Su arquitectura se ilustra en la Figura 2.3. Está constituida por un combinador lineal que suma las señales de entrada x_j ponderadas por los pesos sinápticos w_j correspondientes. La señal de entrada es denotada por

$$\mathbf{x} = (x_1, x_2, \ldots, x_p)^T$$

y el conjunto de pesos sinápticos está dado por el vector

$$\mathbf{w} = (w_1, w_2, \ldots, w_p)^T .$$

La salida del combinador lineal, representada por u, es la entrada a una *función de activación* que limita la amplitud de la salida de la neurona. Típicamente, el rango de amplitud normalizado de la salida es entre [0, 1] o [-1, 1].

La arquitectura también incluye un umbral θ que tiene como objetivo fijar un valor mínimo que debe superar la salida del combinador lineal para activar la neurona. Este umbral puede ser agregado al vector $\mathbf{w}$ como $w_0 = \theta$, considerando una entrada adicional $x_0 = 1$.

En términos matemáticos, una neurona está representada por medio de las ecuaciones

$$u = \sum_{j=0}^{p} w_j x_j \tag{2.17}$$

y

$$y = \phi(u), \tag{2.18}$$

donde ϕ representa la función de activación e y es la salida de la neurona.

Existen tres tipos básicos de funciones de activación para determinar la salida:

a) *Función umbral.* Aplica la siguiente ecuación

$$y = \begin{cases} 1 & \text{si } u \geq 0 \\ 0 & \text{si } u < 0 \end{cases} \tag{2.19}$$

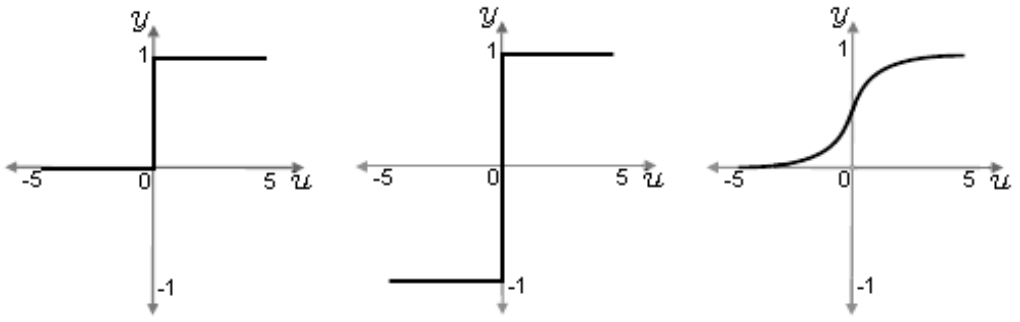
b) *Función bipolar.* Se la define como

$$y = \begin{cases} 1 & \text{si } u \geq 0 \\ -1 & \text{si } u < 0 \end{cases} \tag{2.20}$$

c) *Función sigmoidal.* Es la más comúnmente usada en el modelado de redes neuronales, también llamada *función logística*. Se define como una función estrictamente incremental de la siguiente manera

$$\phi(u) = \frac{1}{1 + e^{-u}} \tag{2.21}$$

En la Figura 2.4 se muestran las tres funciones descritas. Cuando la salida de la función de activación es cercana o igual a su valor máximo, el perceptrón es activado; mientras que cuando está cercano a su valor mínimo, se dice que está desactivado. Por lo tanto, se puede ver al perceptrón simple como un clasificador de dos clases.



(a) (b) (c)

Figura 2.4: Funciones de activación. a) Umbral, b) Bipolar, c) Sigmoidal.

Como se mencionó anteriormente, los pesos sinápticos se utilizan para almacenar el conocimiento de la neurona, siendo fijos o adaptados mediante el aprendizaje. Este aprendizaje implica que los parámetros de la red sean adaptados a través de un proceso continuo de estimulación, por el ambiente en el cual se encuentra. La estimulación es llevada a cabo por medio de los patrones de entrenamiento, y cuando algún criterio de finalización es satisfecho, el algoritmo es detenido y se dice que la red está entrenada y lista para su funcionamiento.

Hay una variedad de algoritmos y reglas de aprendizaje para los diferentes diseños de redes. Para el caso del perceptrón, una de ellas es la denominada *regla delta*. Ésta se basa en ajustar los pesos con el objetivo de minimizar la diferencia entre la respuesta deseada y la obtenida por el perceptrón. La respuesta deseada es denotada por $d(n)$ y la salida producida por un estímulo $\mathbf{x}(n)$, por $y(n)$. Entonces, el error medido en el paso n del entrenamiento está dado por

$$e(n) = d(n) - y(n). \tag{2.22}$$

El ajuste de los pesos se realiza minimizando el criterio de error cuadrático instantáneo

$$E(\mathbf{w}) = \frac{1}{2}e^2(n), \tag{2.23}$$

cuyo gradiente es

$$\nabla E(\mathbf{w}) = -e(n)\phi'(n)\mathbf{x}(n). \tag{2.24}$$

De esta manera, el valor actualizado está dado por

$$w_j(n+1) = w_j(n) + \eta e(n)\phi'(n)x_j(n), \tag{2.25}$$

donde η es una constante positiva que determina la tasa de aprendizaje. Este parámetro tiene un impacto importante en el desempeño del algoritmo. Si es pequeño, el proceso de aprendizaje avanza lentamente pero la convergencia a una solución estable puede tardar demasiado tiempo; en cambio si η es grande el aprendizaje es más acelerado pero hay peligro de que el proceso de aprendizaje diverja y se obtenga una solución inestable. El perceptrón es capaz de lograr un ajuste de los pesos con error cero solamente cuando los patrones son *separables linealmente*, es decir, los patrones pertenecen a dos clases diferentes que pueden ser separados geométricamente por un hiperplano.

Perceptrón multicapa

El perceptrón multicapa (MLP, del inglés *Multi-Layer Perceptron*) representa una generalización del perceptrón simple. Se constituye por una capa de entrada, una o varias capas ocultas y una capa de salida. Un esquema de una red con una capa oculta se muestra en la Figura 2.5. La capa de entrada simplemente copia el patrón de entrada, las capas ocultas y de salida la conforman un conjunto de neuronas. La señal de entrada se propaga por la red de capa en capa, de izquierda a derecha. Además, se puede observar que cada neurona en alguna capa está conectada a todas las neuronas de la capa previa y la siguiente.

El MLP es entrenado en una manera supervisada con algoritmo de *retropropagación del error*, el cual está basado en regla delta, mencionado anteriormente para el caso del perceptrón simple [Hay99]. Este algoritmo comienza con una fase de propagación hacia adelante, donde un patrón de entrenamiento es aplicado en la capa de entrada obteniendo la salida de cada neurona que constituye la red. Específicamente, la salida de la neurona j es calculada de la misma manera que para el caso del perceptrón simple,

$$y_j(n) = \phi_j(u_j(n)), \tag{2.26}$$

donde ϕ_j es la función de activación y $u_j(n)$ la salida del combinador lineal definida por

$$u_j(n) = \sum_{i=0}^{p} w_{ij}(n) y_i(n), \tag{2.27}$$

donde p es el número de entradas aplicado a la neurona j, $w_{ij}(n)$ es el peso sináptico que conecta la neurona i a la neurona j, y $y_i(n)$ es la entrada de la neurona j, o equivalentemente, la salida de la neurona i.

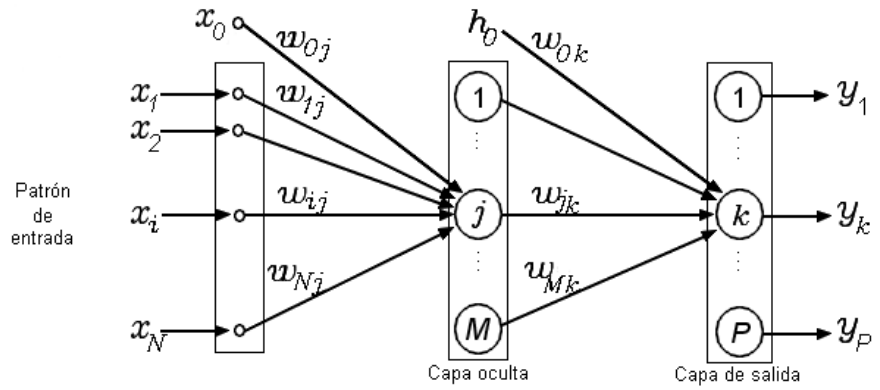


Figura 2.5: Arquitectura de un perceptrón multicapa de una capa oculta.

Siguiendo con el esquema de la Figura 2.5, consideremos los nodos de la capa de salida. El error medido en la neurona k en el paso n es definido por

$$e_k(n) = d_k(n) - y_k(n). \tag{2.28}$$

Entonces, partiendo del error cuadrático instantáneo en la neurona k definido como $\frac{1}{2}e_k^2(n)$, la suma instantánea de errores cuadráticos es obtenida por sumar $\frac{1}{2}e_k^2(n)$ sobre todas las neuronas, expresado matemáticamente como

$$E(n) = \frac{1}{2}\sum_{k=1}^{P} e_k^2(n), \tag{2.29}$$

donde P es la cantidad de neuronas en la capa de salida. La suma instantánea de errores cuadráticos $E(n)$ es una función de todos los pesos sinápticos de la red, por lo tanto, el objetivo del proceso de aprendizaje es ajustar los pesos minimizando $E(n)$.

La regla delta establece que la corrección aplicada a los pesos sinápticos en los nodos de la capa de salida es

$$\begin{aligned} \triangle\omega_{jk}(n) &= -\eta\frac{\partial E(n)}{\partial \omega_{jk}(n)} \\ &= \eta e_k(n)\phi_k'(u_k(n))y_j(n), \end{aligned} \tag{2.30}$$

siendo η la tasa de aprendizaje. Ésto es posible dado que se conocen las respuestas deseadas de cada neurona, mientras que no ocurre los mismo para los nodos de las capas ocultas. De todos modos, es posible inferir una regla de aprendizaje basada en modificar estos pesos para tratar de disminuir el error en la capa de salida, mediante la propagación hacia atrás de las cantidades e_k. Así, la actualización de los pesos en las capas ocultas se realiza mediante el gradiente descendiente de la función criterio error cuadrático, pero calculado con respecto a los pesos w_{ij} como

$$\triangle\omega_{ij}(n) = -\eta\frac{\partial E(n)}{\partial\omega_{ij}(n)}. \tag{2.31}$$

A través de la regla de la cadena, se puede calcular la derivada parcial de (2.31) según

$$\frac{\partial E(n)}{\partial\omega_{ij}(n)} = \frac{\partial E(n)}{\partial y_j(n)}\frac{\partial y_j(n)}{\partial u_j(n)}\frac{\partial u_j(n)}{\partial\omega_{ij}(n)}, \tag{2.32}$$

donde las últimas dos derivadas parciales son calculadas como

$$\frac{\partial y_j(n)}{\partial u_j(n)} = \phi_j'(u_j(n)) \tag{2.33}$$

y

$$\frac{\partial u_j(n)}{\partial\omega_{ij}(n)} = y_i; \tag{2.34}$$

y por último, se calcula la derivada parcial $\partial E(n)/\partial y_j(n)$ como

$$\frac{\partial E(n)}{\partial y_j(n)} = \sum_{k=1}^{P} e_k(n)\frac{\partial e_k(n)}{\partial y_j(n)}. \tag{2.35}$$

Usando la regla de la cadena para la derivada parcial $\partial e_k(n)/\partial y_j(n)$ se reescribe la ecuación y se obtiene

$$\begin{aligned}\tfrac{\partial E(n)}{\partial y_j(n)} &= \textstyle\sum_{k=1}^{P} e_k(n)\tfrac{\partial e_k(n)}{\partial u_k(n)}\tfrac{\partial u_k(n)}{\partial y_j(n)} \\ &= -\textstyle\sum_{k=1}^{P} e_k(n)\phi_k'(u_k(n))\omega_{jk}(n).\end{aligned} \tag{2.36}$$

Reemplazando en (2.31) se obtiene la regla de actualización de pesos en las capas ocultas

$$\triangle\omega_{ij}(n) = \eta\left[\sum_{k=1}^{P} e_k(n)\phi_k'(u_k(n))\omega_{jk}(n)\right]\phi_j'(u_j(n))y_i. \tag{2.37}$$

La actualización de los pesos es realizada por cada patrón de entrenamiento que se presente a la red. La convergencia del algoritmo de aprendizaje, a una solución adecuada, se comprueba al finalizar la presentación del conjunto completo de entrenamiento, verificando el valor de alguna función error prefijada o mediante alguna técnica de estimación del error.

Con respecto a la tasa de aprendizaje η, aquí se presenta el mismo caso que para el perceptrón simple. Un método para incrementar la tasa de aprendizaje y evitar una solución inestable es modificar la regla delta incluyendo un término denominado *momento*

$$\triangle\omega_{ij}(n) = \alpha\triangle\omega_{ij}(n-1) - \eta\frac{\partial E(n)}{\partial\omega_{ij}(n)}, \tag{2.38}$$

donde α es frecuentemente un número positivo llamado *constante de momento*. Cabe aclarar que los subíndices en la ecuación hacen referencia tanto a pesos de neuronas ocultas como de salidas.

Red de función de base radial

Una red de función de base radial (RBF, del inglés *Radial Basis Function*) está constituida por tres capas cuando es usada para clasificación, una capa de entrada, una oculta y una de salida [HH93]. Un esquema de una red RBF se observa en la Figura 2.6. Al igual que en una red MLP, la capa de entrada copia el patrón de entrada. La capa oculta y de salida la conforman un conjunto de nodos. Las salidas de los nodos en la capa de salida son una combinación lineal ponderada de las salidas de los nodos de la capa oculta.

En términos matemáticos, la salida del nodo k en la capa de salida está dada por

$$y_k = \sum_{j=0}^{M} w_{jk}\varphi_j, \tag{2.39}$$

donde ω_{jk} es el peso sináptico que conecta el nodo j de la capa oculta al nodo k de la capa de salida, y φ_j para $j = 1, 2, \ldots, M$ es un conjunto de funciones conocidas como *funciones de base radial*. Estas funciones son las salidas de los nodos de la capa oculta, y comúnmente tienen una forma Gaussiana dada por

$$\varphi_j = e^{\frac{-u_j}{2\sigma_j^2}}, \tag{2.40}$$

donde

$$u_j = \sum_{i=1}^{N}(x_i - w_{ij})(x_i - w_{ij}). \tag{2.41}$$

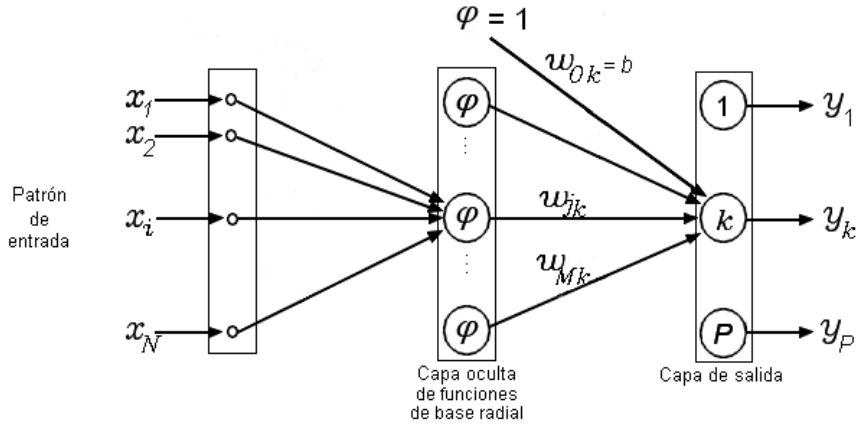


Figura 2.6: Arquitectura de una red de función de base radial.

Aquí, φ_j es la salida del nodo j en la capa oculta, x_i es la entrada a la red y w_{ij} corresponde a los pesos sinápticos del nodo j. Además, se deben notar que los pesos w_{ij} son el centro de la función Gaussiana, y que σ_j^2 define el ancho de la misma. La salida del nodo j está en el rango de 0 a 1, de modo que cuanto más cerca está la entrada de la red al centro de la función Gaussiana, mayor será la respuesta del nodo j. El nombre *función de base radial* se debe a que φ_j es radialmente simétrica.

Existen varias aproximaciones para el entrenamiento de una red RBF. Algunas de éstas, suelen dividir el aprendizaje en dos etapas. En la primer etapa, el aprendizaje se realiza en la capa oculta mediante un método no supervisado, como por ejemplo, un algoritmo de clustering; en la segunda etapa, el aprendizaje se realiza en la capa de salida usando un método supervisado. Una vez obtenida una solución inicial, se puede aplicar un algoritmo de entrenamiento supervisado en ambas capas, para optimizar los parámetros de la red.

Hay una variedad de algoritmos de clustering que se pueden aplicar en la capa oculta. Uno de los más conocidos por su simplicidad y producir buenos resultados, es el algoritmo *K-means*. Por otro lado, para el caso de la capa de salida, se suele usar la regla delta; mientras que para la optimización de parámetros se puede aplicar un método de gradiente.

Comparación de redes RBF y perceptrones multicapa

Al igual que las redes MLP, las redes RBF son redes hacia adelante y pueden ser usadas para clasificación. Siempre es posible encontrar una red RBF que realice la misma tarea que un MLP, y viceversa. De cualquier modo, ambas difieren en varios aspectos. Una de las diferencias es el número de capas ocultas que las constituyen. Mientras que una red MLP puede tener varias capas ocultas, una red RBF tiene una única capa oculta. Otra de las diferencias es que los modelos de neuronas en una red MLP son similares tanto en las capas ocultas como de salida; en tanto que en una red RBF, los nodos en la capa oculta son absolutamente distintos y tienen un propósito diferente al de los de la capa de salida.

CAPÍTULO 3

Desarrollo del software

En este proyecto se desarrolló un sistema que permite identificar a una persona por medio del reconocimiento de una imagen de su rostro. Para ello, se instaló una cámara web en una PC de escritorio para la captura de imágenes. Por medio de ésta, el sistema toma una foto de la persona, la procesa e informa la identidad de la misma.

A la hora de modelar sistemas, una de las herramientas más utilizadas es el Lenguaje Unificado de Modelado (UML, del inglés *Unified Modeling Language*) [BRJ98]. Este estándar es un lenguage gráfico para visualizar, especificar, construir y documentar un sistema. Provee un vocabulario y reglas que permiten realizar modelos conceptuales de varios aspectos del sistema, con el objetivo de facilitar la comunicación entre el equipo de desarrollo y los usuarios finales, quienes son los que especifican las funciones y el comportamiento deseado del mismo.

A fin de realizar un análisis y documentación del sistema, en las siguientes secciones se explican las funcionalidades necesarias por medio de UML. Se especifican los requerimientos identificados, y se realizan los diagramas de casos de uso, clases y secuencia respectivamente.

3.1 Análisis de requerimientos

Un requerimiento puede ser definido como:

- Una condición o capacidad que un usuario necesita para resolver un problema o lograr un objetivo.
- Una condición o capacidad que debe tener un sistema para satisfacer un contrato, norma, especificación, u otros documentos formales.
- Una representación documentada de una condición o capacidad como las mencionadas anteriormente.

A su vez, estos requerimientos pueden ser diferenciados en *requerimientos funcionales* y *requerimientos no funcionales*. Los primeros, son declaraciones de las funciones que el sistema debe ser capaz de realizar, de como debe responder ante entradas particulares; mientras que los segundos, son restricciones del sistema, tales como disponibilidad, mantenimiento, seguridad, capacidad de los dispositivos de entrada/salida, rendimiento (como velocidad y tiempo de respuesta), etc..

En base a estas definiciones, se realiza un análisis de requerimientos con el objetivo de identificar tanto las funcionalidades que se esperan del software como sus limitaciones.

3.1.1 Requerimientos funcionales

El sistema debe ser capaz de:

1. Capturar una imagen a través de una cámara web, con la posibilidad de guardarla.
2. Identificar a una persona por medio de una imagen de su rostro.

3.1.2 Requerimientos no funcionales

Como requerimientos no funcionales se especifican los siguientes:

1. El sistema debe funcionar con una cámara web estándar y luz artificial.

2. El sistema debe ejecutarse en tiempo real, lo que implica que las operaciones computacionales deben procesarse en un tiempo aceptable.

3.2 Diagrama de casos de uso

Un caso de uso captura el comportamiento deseado de un sistema en desarrollo, sin tener que detallar como se implementa éste. Es una descripción de un conjunto de secuencias de acciones que el sistema lleva a cabo para producir un resultado de interés para un usuario, reflejando como deberían interactuar ambas partes.

Un diagrama de casos de uso muestra la relación entre los usuarios del sistema y los casos de uso, modelando los aspectos dinámicos del mismo.

Para el caso analizado aquí, se pretende visualizar como interactúa el usuario con el sistema haciendo uso de las funcionalidades ofrecidas por el mismo. En la Figura 3.1 se puede observar un diagrama de alto nivel del sistema, y en la Figura 3.2 un diagrama más detallado.

A continuación se describen los casos de uso correspondientes al diagrama de la Figura 3.1.

Caso de uso: Capturar imagen
Actor: Usuario
Descripción: Captura una imagen de la persona con la cámara web.
Curso normal:
1) Comienza cuando el usuario presiona el botón 'Capturar' o 'Identificar'. 2) Se toma una foto de la persona.
Cursos alternativos:
2.A) La cámara no está inicializada. 2.A.1) El sistema inicializa la cámara web previsualizando lo que ésta captura.

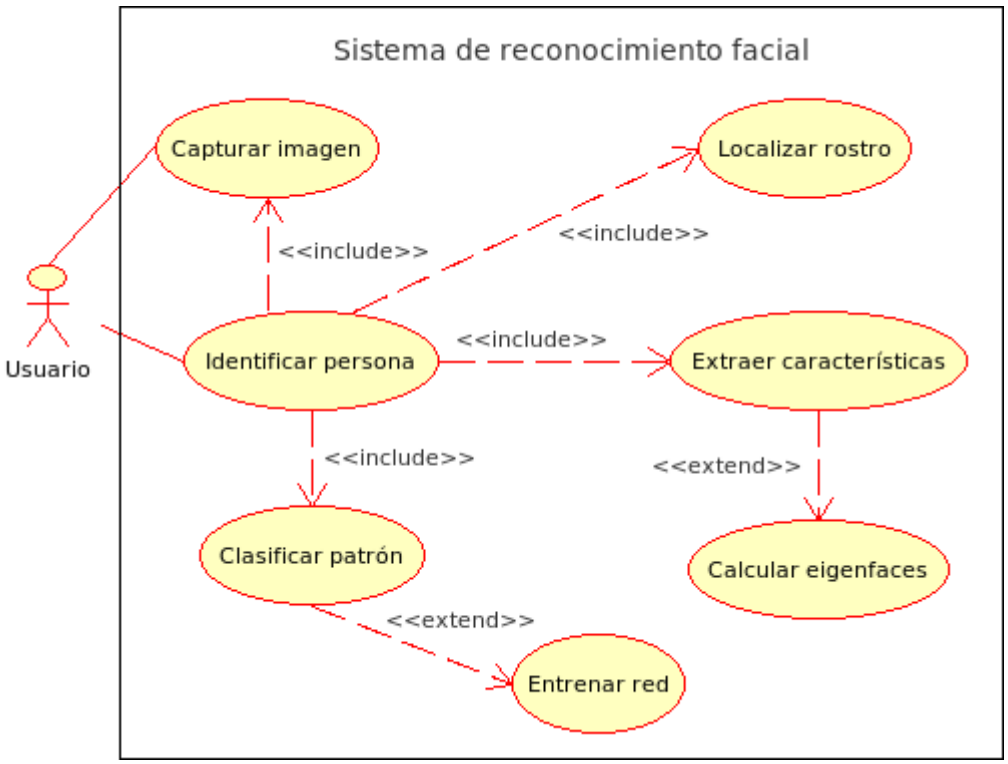


Figura 3.1: Diagrama de casos de uso de alto nivel.

Caso de uso: Identificar persona
Actor: Usuario
Descripción: Identifica a una persona a partir de una imagen de su rostro.
Curso normal:
1) Comienza cuando el usuario presiona el botón 'Identificar'. 2) El sistema captura una imagen del usuario. 3) Se localiza el rostro en la imagen. 4) Se proyecta la imagen facial mediante las eigenfaces y se obtiene un patrón representativo de ésta. 5) Se clasifica el patrón mediante una red neuronal. 6) El sistema informa la identidad de la persona.

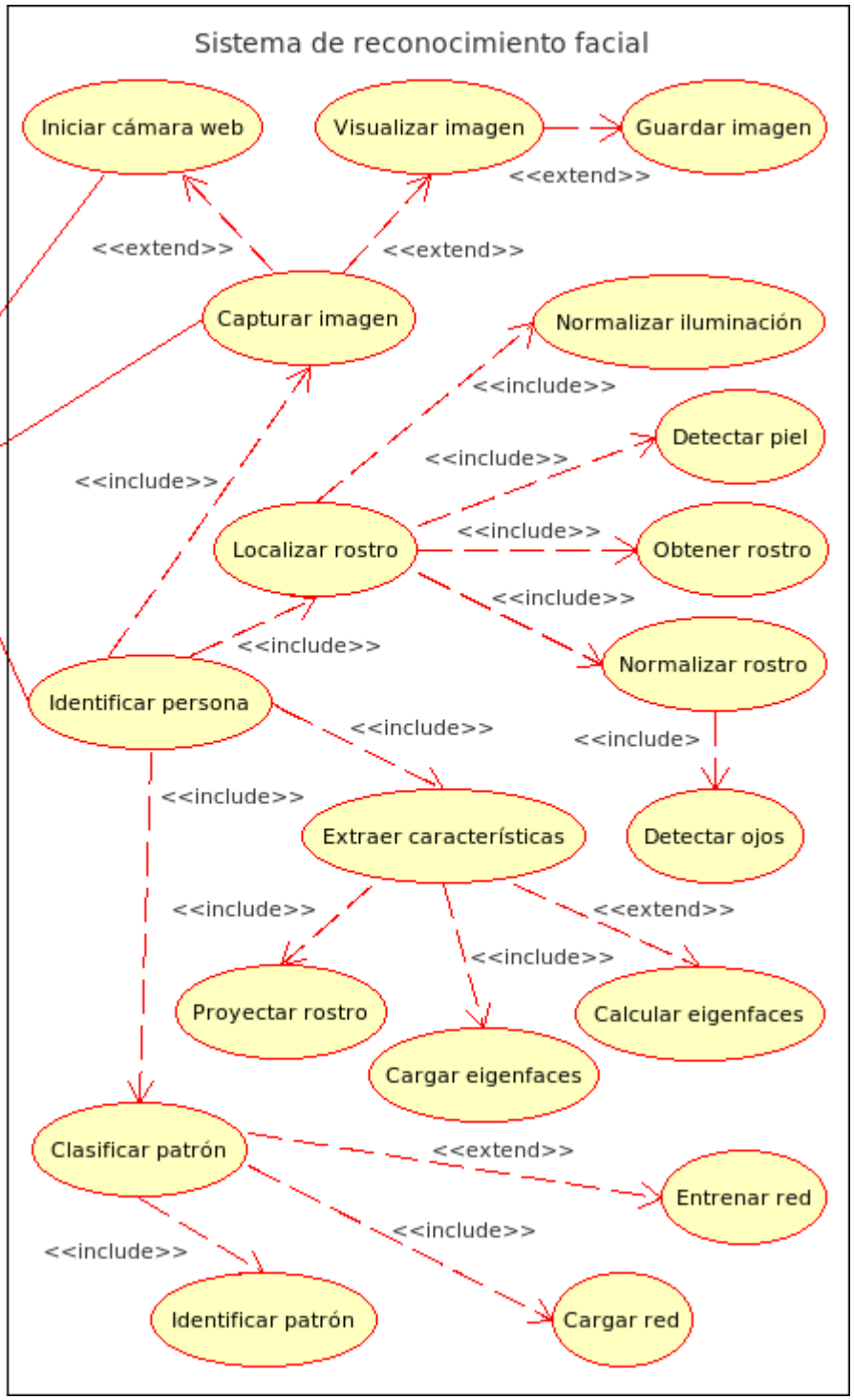


Figura 3.2: Diagrama de casos de uso detallado.

Caso de uso: Localizar rostro
Actor: -
Descripción: Obtiene una subimagen facial normalizada.
Curso normal:
1) Se normaliza la iluminación en la imagen. 2) Se detectan las regiones de piel. 3) Se aplica una plantilla elíptica para ubicar la cara y se recorta una subimagen de tamaño definido. 4) Se detectan los ojos a partir de la imagen obtenida. 5) Se normaliza el rostro a partir de los ojos y se recorta la imagen original nuevamente.

Caso de uso: Extraer características
Actor: -
Descripción: Obtiene un vector representativo de la imagen facial para clasificarlo.
Curso normal:
1) Se cargan las eigenfaces y la imagen media. 2) Se centra la imagen facial sustrayéndole la imagen media. 3) Se proyecta la imagen centrada multiplicándola por cada una de las eigenfaces, generando un punto en el espacio de caras.
Cursos alternativos:
1.A) Las eigenfaces y la imagen media no están calculadas. 1.A.1) Se calculan las eigenfaces y la imagen media. 1.A.2) Se guardan las eigenfaces y la imagen media.

Caso de uso: Calcular eigenfaces
Actor: -
Descripción: Calcula las eigenfaces de todas las imágenes en la base de datos.
Curso normal:
1) Se calcula la imagen media de todas las imágenes en la base de rostros. 2) Se calculan las eigenfaces. 3) Se guardan las eigenfaces y la imagen media.

Caso de uso: Clasificar patrón
Actor: -
Descripción: Clasifica un patrón para identificar a la persona.
Curso normal:
1) Se carga la red. 2) Se aplica el patrón a la red devolviendo como resultado la identificación de la persona.
Cursos alternativos
1.A) La red no está entrenada. 1.A.1) Se entrena la red. 1.A.2) Se guarda la red.

Caso de uso: Entrenar red
Actor: -
Descripción: Entrena la red neuronal para realizar la clasificación de patrones.
Curso normal:
1) Se proyectan todas las imágenes de la base de rostro al espacio de caras obteniendo los patrones correspondientes. 2) Se entrena la red mediante los patrones obtenidos. 3) Se guarda la red entrenada.

3.3 Diagrama de clases

Los diagramas de clases son los más usados en el modelado de sistemas orientados a objetos, los cuales muestran un conjunto de clases y sus relaciones. Se utilizan para modelar la vista de diseño estática de un sistema, lo que implica modelar el vocabulario del mismo, modelar colaboraciones o modelar esquemas.

Así como son importantes para visualizar, especificar y documentar modelos estructurales, también lo son para construir sistemas ejecutables, aplicando ingeniería directa e inversa.

En la Figura 3.3 se puede observar el diagrama correspondiente al sistema de reconocimiento facial. Como se puede notar, se definieron distintas clases

que permiten realizar tanto la captura de imágenes como el reconocimiento de una persona.

Las clases *Application*, *MainFrame*, *WebCam* y *SnapshotDialog* son extensiones de clases contenidas en la librería *wxWidgets*[1], las cuales no son mostradas en el diagrama a modo de simplificar el mismo. Esta librería permite crear interfaces de usuario haciendo uso de las bibliotecas ya existentes en el sistema.

La clase *Application* (extensión de wxApp) representa la aplicación misma. Es usada para definir y obtener propiedades de ésta, como también para iniciarla.

La clase *MainFrame* (extensión de wxFrame) corresponde a la ventana de la aplicación. Gestiona las funcionalidades del menú y los botones incluidos en ella. Inicializa la cámara web, captura una imagen o identifica un usuario, según la acción ejecutada por el usuario. Además, visualiza lo que la cámara web captura.

La clase *WebCam* (extensión de wxWindow) se encarga de manejar el dispositivo de captura de imágenes. Detecta si se encuentra conectada una cámara a la PC, inicializa y detiene su controlador, como también captura imágenes a ser visualizadas en la vista previa por la clase MainFrame.

La clase *SnapshotDialog* (extensión de wxDialog) se utiliza para visualizar la imagen capturada por el usuario en una ventana de diálogo. Permite guardar la misma en el disco rígido o descartarla.

Para el procesamiento básico de las imágenes, se utiliza la clase *CImg*[2], siendo eficiente y simple de usar. Es libre, de código abierto y es distribuida bajo las licencias CeCILL-C (similar a la LGPL) o CeCILL (similar a la GPL). Permite cargar y guardar imágenes en formatos ppm, jpg y otros, manipular imágenes como matrices, aplicar operaciones de dilatación y erosión, rotar y cortar imágenes, efectuar transformaciones a diferentes espacios de color, etc..

La clase *Images* hereda todas las operaciones de CImg, y además incluye funciones para normalizar la iluminación, segmentar la piel, localizar el rostro en una imagen y normalizar la geometría del mismo.

La clase *SubjectsList* se encarga de cargar las imágenes de la base de rostros.

La clase *Eigenfaces* contiene las funciones necesarias para generar el espacio de caras. Calcula, guarda y carga la imagen media y las eigenfaces del

[1]http://www.wxwidgets.org/
[2]http://cimg.sourceforge.net

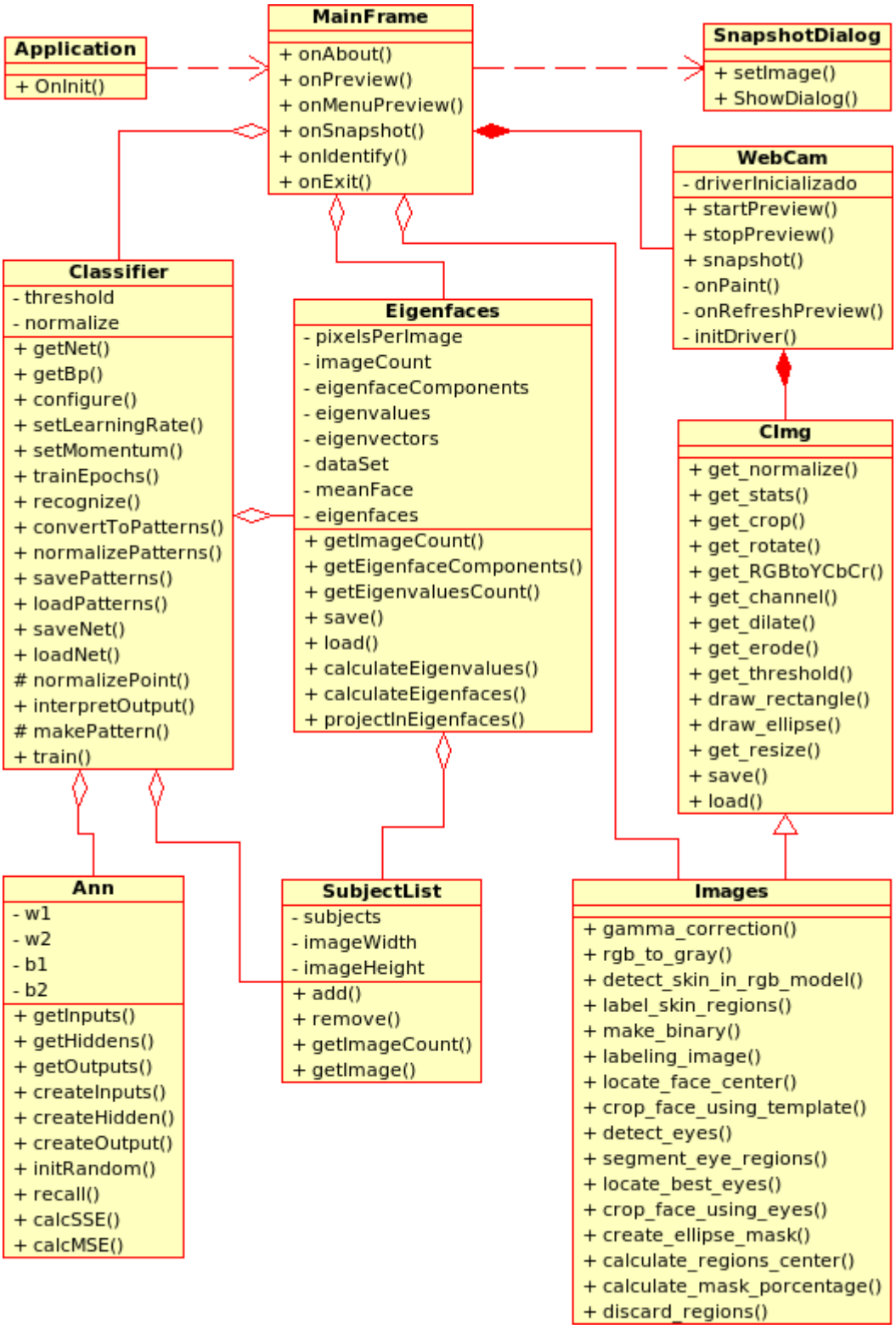


Figura 3.3: Diagrama de clases.

conjunto completo de rostros, y proyecta una imagen facial a este espacio.

La clase *Ann* implementa un perceptrón multicapa de tres capas: entrada, oculta y salida. Construye la red según el número de entradas y la cantidad de nodos en la capa oculta y salida e identifica un patrón aplicado a la entrada de la misma.

Por último, la clase *Classifier* se encarga de generar los patrones de entrenamiento haciendo uso de la clase Eigenfaces, proyectando cada imagen facial al espacio de caras. Con estos patrones, entrena la red neuronal mediante el algoritmo de retropropagación del error, y permite guardar y cargar tanto la red entrenada como los patrones mismos.

3.4 Diagrama de secuencia

Los diagramas de secuencia son usados para modelar los aspectos dinámicos del sistema. Muestran las interacciones entre los distintos objetos participantes y los mensajes que intercambian. Estos diagramas se forman colocando los objetos y actores que interactúan en el eje horizontal, y en el eje vertical se colocan los mensajes intercambiados en orden temporal, desde arriba hacia abajo. El objeto que inicia la interacción suele ubicarse a la izquierda y los objetos subordinados a la derecha.

En la Figura 3.4 se puede observar el diagrama correspondiente al momento en que un usuario intenta identificarse ante el sistema. El usuario inicia el reconocimiento al solicitarle a la clase MainFrame que lo identifique. Ésta instancia a la clase WebCam para que tome una foto del usuario. Con la imagen capturada, la clase Images la carga, la realza y, a partir de la detección de la piel, ubica el rostro obteniendo como resultado una subimagen que corresponde a la cara normalizada.

Una vez localizado el rostro, se instancia a la clase Eigenfaces que carga las eigenfaces y la imagen media. Toma la cara normalizada, la transforma en un vector y a este último, lo proyecta al espacio de caras obteniendo un patrón a clasificar. Luego, con la clase Classifier se carga la red neuronal entrenada y se aplica el patrón obtenido a la entrada de la misma. Finalmente, se informa el resultado de la identificación al usuario.

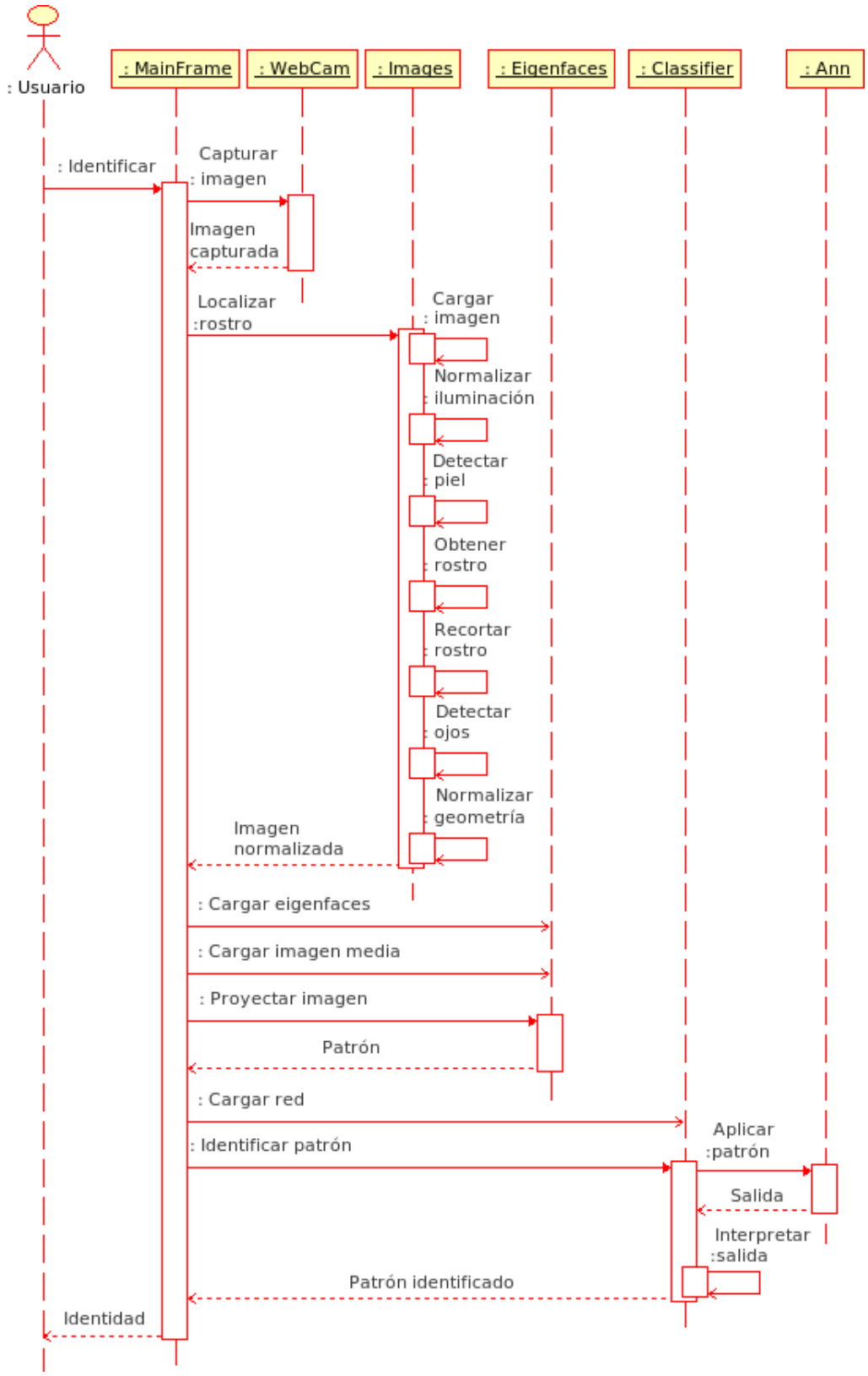


Figura 3.4: Diagrama de secuencia de identificación de persona.

3.5 Interfaz de usuario

A partir de los requerimientos especificados y los distintos diagramas expuestos, se desarrolló la interfaz de usuario implementándose tres funcionalidades: 1) iniciar la cámara web, 2) capturar una imagen y 3) identificar un usuario. Ésta se desarrolló en lenguaje C++ utilizando la librería wxWidgets. Como se indicó anteriormente, esta librería permite crear interfaces de usuario basadas en las bibliotecas ya existentes en el sistema, con lo que se obtiene una apariencia muy similar a una aplicación nativa. De esta manera, resulta muy portable entre distintos sistemas operativos, estando disponibles para Windows, OS X, Linux y Unix. Es libre, de código abierto y distribuida bajo una licencia LGPL (similar a la GPL). En la Figura 3.5 se puede observar la interfaz desarrollada. Aquí se pueden notar tres botones en la parte inferior: Vista Previa, Capturar e Identificar. Cada botón realiza una acción como se describe a continuación:

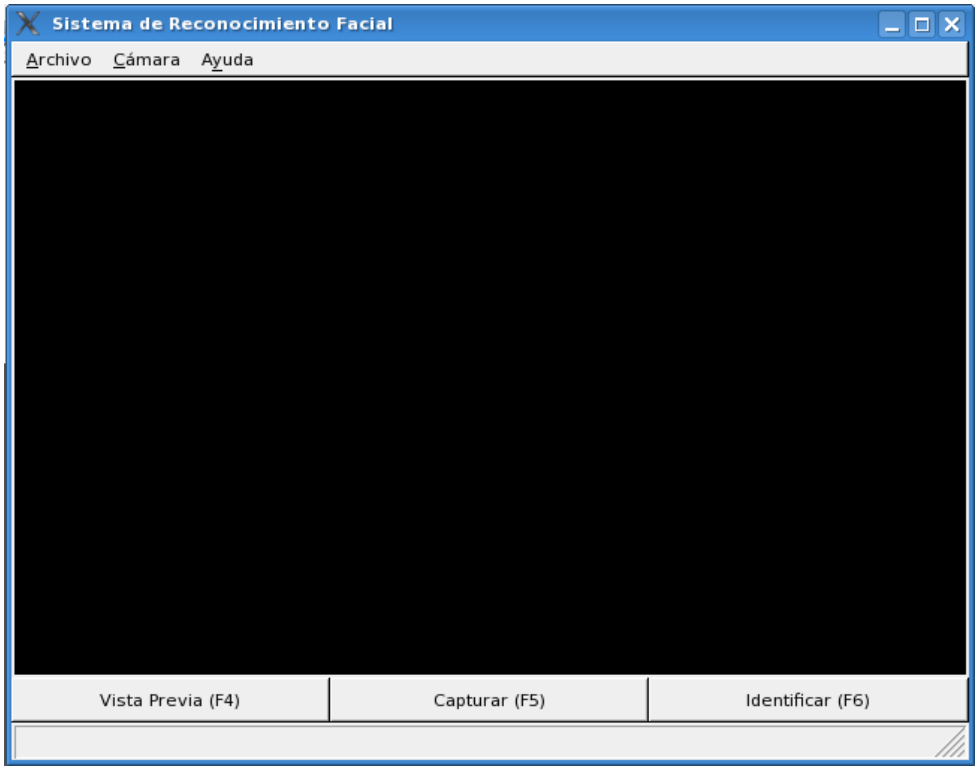


Figura 3.5: Interfaz de usuario.

- Vista Previa: inicializa la cámara web, y reproduce en el recuadro negro lo que ésta captura.
- Capturar: captura una imagen de la vista previa visualizándola en otra ventana, en la cual se puede optar por guardar la imagen tomada (botón Guardar) o descartarla (botón Cancelar). Un ejemplo se muestra en la Figura 3.6.
- Identificar: captura una imagen de la vista previa, aplica las tres etapas para el reconocimiento e indica quién es el usuario que se encuentra frente a la cámara (Ver Figura 3.7).

Figura 3.6: Captura de imagen.

Estas mismas opciones también pueden ser accedidas desde el menú, en la opción Archivo. Además, se debe mencionar que la interfaz funciona sobre los sistemas Windows y Linux.

Figura 3.7: Identificación de usuario.

Las imágenes son capturadas utilizando la cámara web en una resolución de 320 x 240 píxeles. Sobre Linux, se usó el controlador no oficial *spca*[3], el cual soporta una variedad de dispositivos. Una lista de éstos puede ser encontrada en su página web. Bajo Windows, se requirió del controlador de la cámara web utilizada.

[3]http://mxhaard.free.fr/spca5xx.html

CAPÍTULO 4

Implementación del sistema

Un sistema de RF consta de tres etapas, como se mencionó anteriormente: (1) detección del rostro, (2) extracción de características y (3) clasificación. En la primera etapa se localiza el rostro en una imagen eliminando todo lo que sea fondo. Una vez obtenida la cara, el segundo paso se encarga de extraer características representativas de ésta, formando así, un vector patrón que se utiliza en la etapa de clasificación para la identificación o verificación de la identidad de una persona con respecto a los usuarios registrados en el sistema. En la Figura 4.1 se puede observar una configuración genérica de un sistema de RF.

A partir de la interfaz de usuario desarrollada, se capturaron un conjunto de imágenes conformando una base de rostros denominada *SINCIDISI*, con la cual se llevaron a cabo las pruebas del sistema. Luego, se desarrollaron

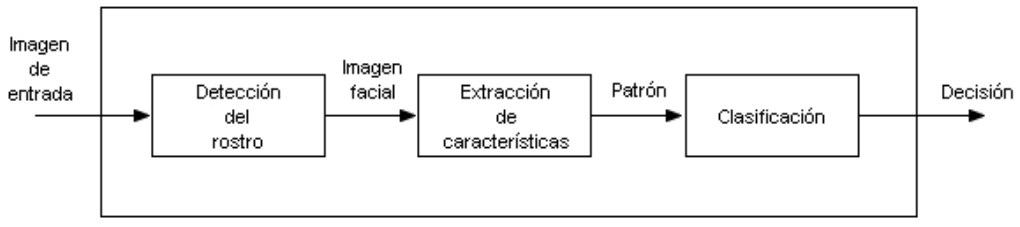


Figura 4.1: Configuración genérica de un sistema de RF.

las tres etapas para el reconocimiento. Previo a la detección del rostro, las imágenes fueron sometidas a un proceso de normalización de la iluminación con el objetivo de realzar las imágenes capturadas. La detección del rostro se basó en el algoritmo propuesto en [ML06], el cual combina detección de piel, aplicación de plantillas y detección de características faciales. Para la extracción de características se proyecta la imagen facial a un espacio de caras, obteniendo un vector patrón que refleja las propiedades más relevantes del rostro. Para la clasificación se diseñó una estructura de red neuronal MLP, constituida por una capa de entrada, una oculta y otra de salida, entrenándola mediante el algoritmo de retropropagación del error. Una vez entrenada, se le presentan los diferentes patrones y devuelve las identidades de los usuarios.

Por otro lado, se debe mencionar que el proceso de construcción se divide en dos fases, una de entrenamiento y una de clasificación. En la primer fase, se captura un conjunto de N imágenes a procesar, denominado *conjunto de imágenes de entrenamiento* $I = \{I_1, ..., I_N\}$, de un grupo de usuarios. En cada imagen se localiza el rostro obteniendo las imágenes faciales F_i correspondientes, y se calculan los eigenvectores que mejor describen al conjunto facial resultante. Cada rostro F_i es proyectado al nuevo espacio obteniendo un vector característico $\mathbf{x}_i$. De esta manera, se genera un conjunto de patrones que es utilizado para entrenar la red neuronal. La fase de clasificación toma una imagen de entrada I_q, localiza el rostro y proyecta éste al nuevo espacio, en base a los eigenvectores calculados en la fase de entrenamiento, obteniendo el vector característico $\mathbf{x}_q$. Luego, este vector es aplicado a la entrada de la red neuronal obteniendo como resultado la identidad del usuario.

4.1 Base de datos de rostros SINCIDISI

Con la interfaz de usuario ya implementada se capturaron varias fotos en un ambiente real, creando así la base de rostros *SINCIDISI*. Para ello, se instaló una cámara web en una PC del centro *sinc(i)*[1] de la FICH y en una PC del *CIDISI*[2] de la UTN FRSF (de allí su nombre SINCIDISI), colocándose una fuente de luz artificial para lograr una mejor iluminación en las muestras. Las fotos fueron tomadas con fondo, pose, expresión facial y distancia a la cámara totalmente natural. De esta manera, se obtuvo una cantidad de 448

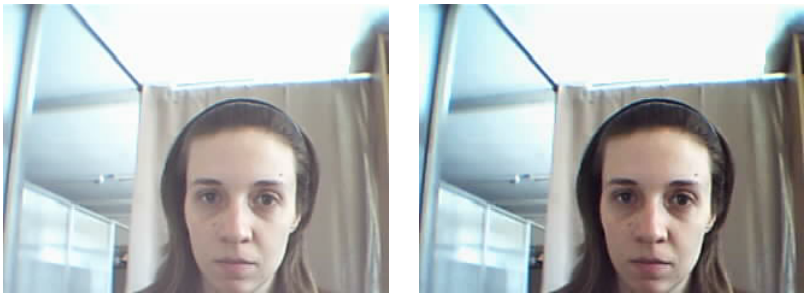
[1] http://fich.unl.edu.ar/sinc/
[2] http://cidisi.frsf.utn.edu.ar/

Figura 4.2: Ejemplos de todos los usuarios de la base de rostros SINCIDISI.

imágenes de 21 personas.

A partir del conjunto de imágenes capturadas, se realizó una selección manual de las mismas a fin de descartar aquellas en las que no se tuvieran variaciones aceptables de pose e iluminación, como también aquellas con oclusiones parciales del rostro. Además, se eliminaron imágenes en las que se obtuvieron falsas localizaciones del rostro por la presencia de puertas, armarios u otros objetos con un color similar a la piel.

Por otro lado, se notó que la cantidad de fotos para algunos usuarios era escasa y se decidió descartar éstos. A partir de allí, se seleccionaron las imágenes de mejor calidad, en una cantidad igual por cada usuario. De este modo, se obtuvo un conjunto final de 110 imágenes, de un total de 11 personas (10 imágenes por persona). Las fotos fueron almacenadas en formato PPM, con un tamaño de 320 x 240 píxeles. En la Figura 4.2 se exhibe un subconjunto de la base, con ejemplos de todos los usuarios.



(a) (b)

Figura 4.3: Normalización de la iluminación. a) Imagen original, b) Imagen con iluminación normalizada.

4.2 Metodología

4.2.1 Detección del rostro

Un cambio en la distribución de la fuente de luz y en el nivel de iluminación produce un cambio en el color de la piel en la imagen, degradando el rendimiento de los sistemas de detección de piel [KMB07]. Debido a que el algoritmo aplicado utiliza información sobre el color de piel para ubicar el rostro, es necesario aplicar un proceso de normalización que compense la luminosidad en las imágenes capturadas por la cámara web, ya sean imágenes con poca o mucha iluminación.

Para compensar la luminosidad se aplicó una transformación de potencia (a cada canal del modelo RGB) dada por

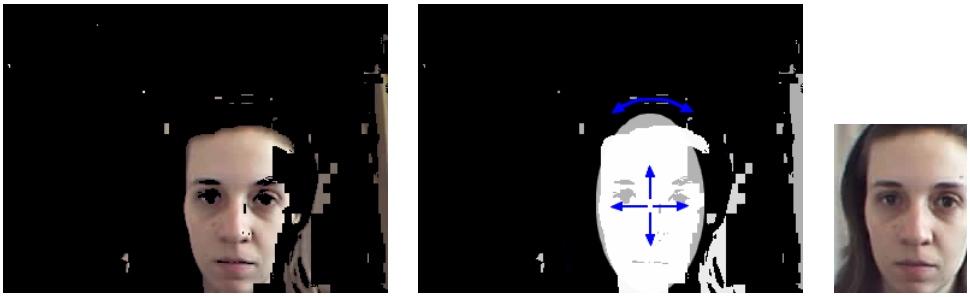
$$f(x, y) = f(x, y)^{\gamma}, \tag{4.1}$$

donde γ es adaptada según la cantidad de luz en cada imagen por

$$\gamma = c_1 \mu_f + c_2. \tag{4.2}$$

Aquí, c_1 y c_2 son constantes y μ_f es la media de la imagen a normalizar en escala de grises. Con ésto se logra que γ sea directamente proporcional al brillo. En la Figura 4.3 se observa el resultado de aplicar esta transformación.

A partir de la imagen normalizada, como primer paso, el algoritmo localiza el rostro en la imagen. Para ésto, realiza una segmentación de piel aplicando



(a) (b) (c)

Figura 4.4: Localización del rostro. a) Segmentación de piel, b) Combinación de plantilla sobre las regiones, c) Subimagen con el rostro localizado.

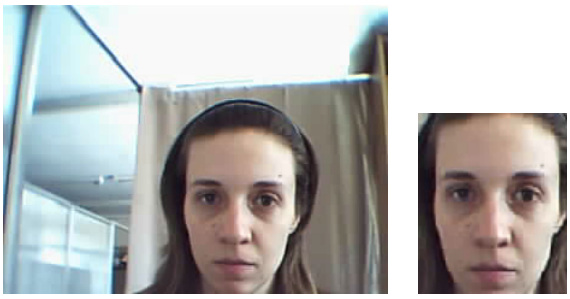
ciertas reglas definidas en [KPS97], basadas en que el color de piel se mantiene en un cierto rango del espacio de color RGB. De esta manera, logra localizar las posibles regiones caras en la imagen. La Figura 4.4(a) muestra el resultado de la segmentación.

Una vez ubicadas las posibles regiones de piel, éstas son etiquetadas mediante un algoritmo recursivo. Luego, se elige aquella región que mejor se asemeje geométricamente a un rostro. Para ésto, se aplican plantillas elípticas a las diferentes regiones, haciendo ampliaciones, traslaciones y rotaciones de las mismas para lograr un mejor rendimiento ante variación de pose y distancia a la cámara (Figura 4.4(b)), y se elige aquella región que contiene la mayor cantidad de píxeles que caen dentro de la plantilla. Posteriormente se recorta una subimagen que contiene la región seleccionada, o dicho de otro modo, el rostro del usuario (Figura 4.4(c)).

En segundo lugar, en base a la imagen recortada, el algoritmo normaliza el tamaño y orientación del rostro teniendo en cuenta la distancia entre ojos, posibilitando la comparación de patrones estandarizados geométricamente. Primero detecta las posibles regiones de ojos mediante operaciones morfológicas de dilatación y erosión aplicadas a la componente de luminancia Y del modelo YCbCr, filtrando aquellas que tengan demasiados o muy escasos píxeles como para representar un ojo. En la Figura 4.5(a) y (b) se observa la componente de luminancia Y y la segmentación resultante respectivamente. Luego, busca el par de ojos candidatos tomando cada región de la mitad izquierda, buscando su par a la derecha que caiga dentro de un rectángulo predefinido (Figura 4.5(c)). Los pares de ojos candidatos resultantes son validados mediante una ponderación de propiedades como posición en la imagen y distancia entre ellos, descartando aquellos que no superan un umbral preestablecido.

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

Figura 4.5: Normalización de tamaño. a) Componente Y, b) Posibles regiones ojos, c) Búsqueda del par de ojos, d) Ojos detectados.



(a) (b)

Figura 4.6: Resultado de la detección del rostro. a) Imagen con iluminación normalizada, b) Imagen normalizada.

Con el par de ojos ganador (Figura 4.5(d)), se calcula la distancia ocular horizontal Dx y vertical Dy y el ángulo de inclinación del rostro $\alpha = arctan(Dy/Dx)$. Se corrige la inclinación del rostro en base al ángulo calculado y luego, en base a Dx y Dy, se recorta la subimagen facial normalizada a partir de la imagen realzada como se observa en la Figura 4.6.

4.2.2 Extracción de características

En esta etapa se extraen las características más relevantes de las imágenes faciales aplicando el método *eigenfaces* explicado en la Sección 2.2.

Como se explicó anteriormente, el primer paso es calcular las componentes eigenfaces que describen el conjunto de imágenes faciales aplicando la

ecuación (2.13), escrita nuevamente para conveniencia del lector como

$$\begin{aligned} \mathbf{u}_l &= A\mathbf{v}_l \\ &= \sum_{k=1}^{M} \mathbf{v}_{lk}\mathbf{\Phi}_k, \end{aligned} \tag{4.3}$$

donde M es la cantidad de imágenes en el conjunto de entrenamiento. De esta manera, se obtienen M eigenvectores $\mathbf{u}_l$ que conforman el conjunto base de las imágenes faciales.

Dado que no se requiere la reconstrucción de las imágenes, se utilizan únicamente $M' < M$ eigenfaces, las cuales corresponden a las más significativas según el orden decreciente de los eigenvalores asociados. Estos M' eigenvectores contienen la mayor información del conjunto de datos.

Una vez calculadas las eigenfaces, tanto en el entrenamiento como en la clasificación, las imágenes faciales son proyectadas al eigenespacio mediante

$$x_k = \mathbf{u}_k^T (\mathbf{I} - \mathbf{\Psi}) \qquad \text{para} \quad k = 1, \ldots, M', \tag{4.4}$$

donde I representa una imagen, obteniendo el vector característico $\mathbf{x}$ correspondiente.

4.2.3 Clasificación

Para el reconocimiento de patrones se diseñó una red neuronal MLP de tres capas (entrada, oculta y salida). Un esquema similar de esta estructura se puede observar en la Figura 2.5.

La entrada a la red se definió según las características extraídas de las imágenes faciales en la etapa anterior, dependiendo de la cantidad de eigenfaces utilizadas para la proyección en el espacio de caras. El número de nodos en la capa oculta se determinó mediante experimentación, donde se observó que cantidad era requerida para obtener un desempeño aceptable. La cantidad de salidas es el número de usuarios a reconocer por el sistema, cada salida representa a uno de éstos. La salida cuyo valor es más alto corresponde al usuario reconocido.

Una vez establecido el diseño, la red se entrenó mediante el algoritmo de retropropagación del error, siguiendo las ecuaciones de la Sección 2.3.1. Ésto se realizó en la fase de entrenamiento, en la cual se crearon patrones definiendo la relación entrada-salida para la red. Con dichos patrones, se estimuló la red a fin de minimizar el error entre salida y respuesta deseada. En la fase de

clasificación, la red entrenada es estimulada con el vector característico del usuario que intenta identificarse ante el sistema, y ésta retorna la clase a la que pertenece o indica que no pertenece a ninguna de las clases conocidas.

CAPÍTULO 5

Experimentos y resultados

A partir de la base SINCIDISI se realizaron varias pruebas con el fin de ajustar los parámetros del sistema y obtener un rendimiento aceptable para el conjunto de datos dado.

En primera instancia se determinó la cantidad de eigenfaces a utilizar. Para ésto, se aplicó la técnica del Scree Test. Luego, se evaluó el clasificador MLP. Se estimó el error de clasificación variando la cantidad de entradas en la red y el número de nodos en la capa oculta. Se realizaron diversas gráficas que permiten visualizar el error para las diferentes variaciones, donde se puede ver cuáles son los valores de ambos parámetros para los que se obtiene un error mínimo. Una vez fijada la configuración del MLP se compara el desempeño del mismo con otros métodos de clasificación: la técnica del *vecino más cercano*, utilizada en [ML06], y otra arquitectura de red neuronal, las *redes de función de base radial*.

La estimación del error de clasificación consiste en calcular la proporción de patrones clasificados tanto correcta como erróneamente. Un estimador adecuado cuando el conjunto de entrenamiento es pequeño es el *estimador por validación cruzada con V conjuntos*. El conjunto de entrenamiento T se distribuye en V conjuntos disjuntos $T_1, T_2, \ldots, T_V$ de tamaño similar. De esta manera, se construyen V clasificadores usando $T - T_v$ como conjunto de aprendizaje y T_v como conjunto de prueba, donde $v = 1, 2, \ldots, V$. Por cada clasificador se obtiene el error correspondiente, y finalmente, la estimación se calcula mediante el promediado de éstos. La suposición básica de este estimador es que este procedimiento es "estable", en el sentido de que cada

clasificador construido tiene un error aproximadamente igual al clasificador construido con el conjunto de entrenamiento completo.

En cada método de clasificación, se utilizó validación cruzada con cinco particiones para estimar el error. La base cuenta con 110 imágenes y cada partición se conformó con el 80 % (88) de los datos para entrenamiento y el resto (22) para prueba. Cabe aclarar que los conjuntos de prueba son disjuntos.

Para realizar los experimentos se implementó una herramienta que permite particionar el conjunto de entrenamiento de manera disjunta haciendo uso de la clase SubjectList. Ésta se utilizó tanto para los experimentos del MLP como para la técnica del vecino más cercano. En el caso del MLP, se utilizó la clase Classifier para entrenar la red por cada partición, y de esa manera obtener el error correspondiente; mientras que para la técnica del vecino más cercano se definió una clase denominada *NearestNeighbor* encargada de aplicar el método, el cual se explica posteriormente. Para diseñar la red RBF se utilizó un software denominado *Weka*[1] (Waikato Environment for Knowledge Analysis - Entorno para Análisis del Conocimiento de la Universidad de Waikato), el cual es una herramienta para aprendizaje automático y minería de datos.

5.1 Experimentación

5.1.1 Scree Test

El Scree Test es una técnica gráfica que se aplica para determinar el número de componentes principales (de una matriz de covarianza) a utilizar según el porcentaje de variabilidad de los datos que se desea representar. El término Scree se define como escombros en el fondo de un acantilado, es decir, las componentes principales usadas son el acantilado y el resto los escombros [Jac91].

Para el caso analizado aquí, se trata de determinar la cantidad de eigenfaces a utilizar para representar aproximadamente la totalidad de la información, la gráfica resultante se aprecia en la Figura 5.1. Como se puede observar, con alrededor de 50 o 60 eigenfaces se puede representar alrededor

[1] http://www.cs.waikato.ac.nz/ml/weka/

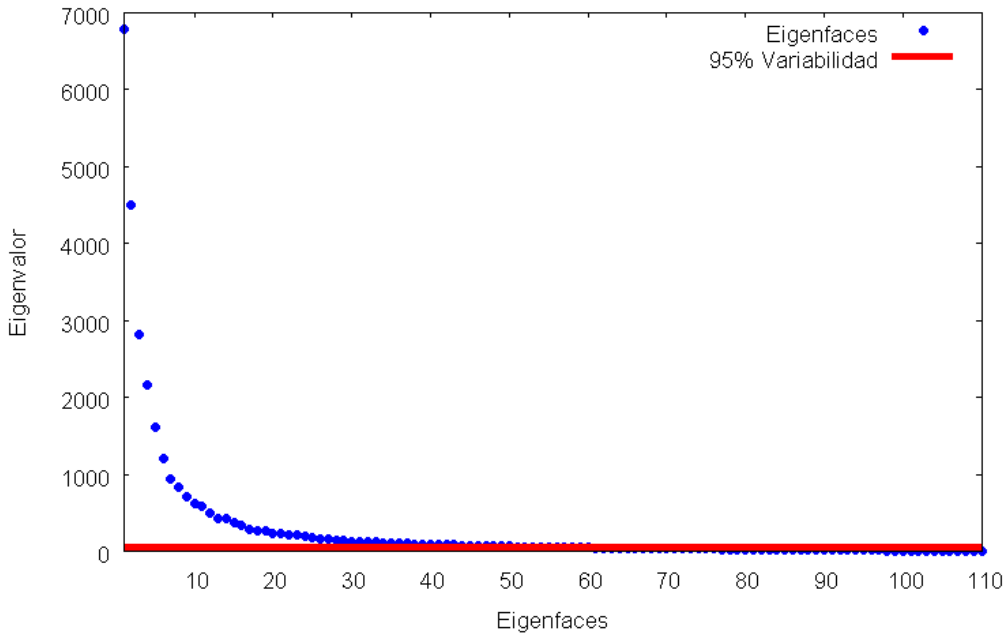


Figura 5.1: Scree Test.

Tabla 5.1: Detalle de eigenfaces.

Nº Eigenfaces	Eigenvalor	Varianza (%)	Varianza acumulada (%)
1	6792,40	21,42	21,42
2	4495,68	14,18	35,61
3	2817,37	8,88	44,50
⋮	⋮	⋮	⋮
50	72,25	0,22	93,77
51	69,78	0,22	94,00
⋮	⋮	⋮	⋮
59	54,21	0,17	95,51
60	53,16	0,16	95,68
⋮	⋮	⋮	⋮
108	8,68	0,03	99,97
109	7,14	0,02	99,99
110	1,30e-13	4,09e-16	100,00

del 95 % de la varianza acumulada. Además, se debe notar que, a partir de 60 eigenfaces, éstas son menores y están casi en linea recta, lo que significa que carecen de relevancia. En la Tabla 5.1 se detallan los eigenvalores, el porcentaje de la varianza y de la varianza acumulada por cada eigenface.

5.1.2 Evaluación del clasificador MLP

Con el objetivo de fijar el número de entradas y la cantidad de nodos en la capa oculta de la red se llevaron a cabo distintas pruebas variando ambas cantidades, estimando el error de clasificación en cada una. Como se indicó anteriormente, se utilizó validación cruzada con cinco particiones para estimar el error. Además, se debe recordar que la cantidad de nodos en la capa de salida es el número de usuarios a reconocer por el sistema.

Inicialmente se calculó el error variando el número de nodos en la capa oculta como se aprecia en la Figura 5.2. Aquí se observan distintas gráficas, cada una correspondiendo a una cantidad de eigenfaces en particular. Cabe recordar que las características extraídas a las imágenes faciales son las entra-

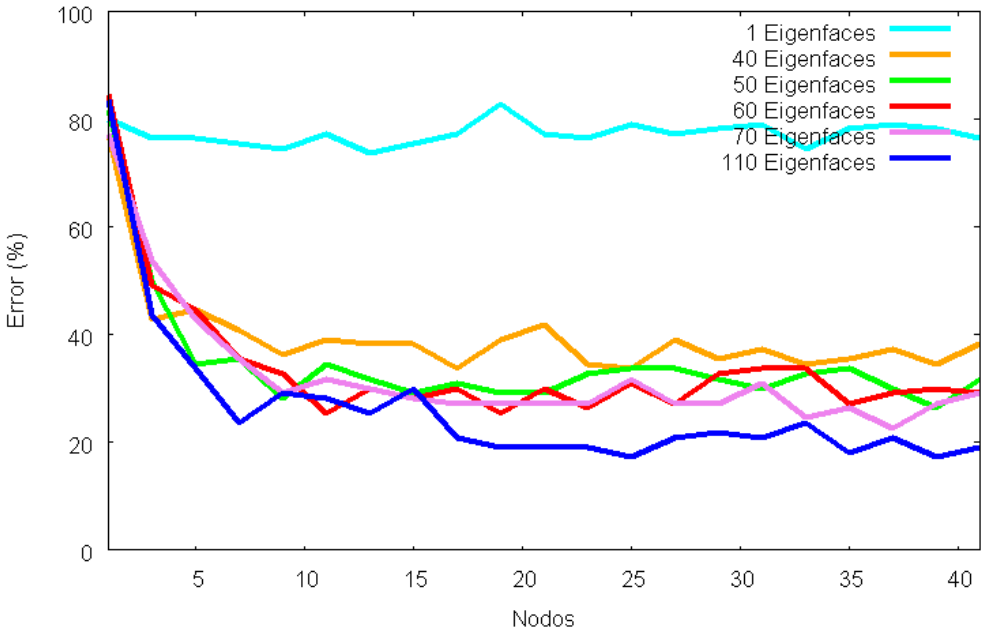


Figura 5.2: Tasa de error para cantidades específicas de eigenfaces.

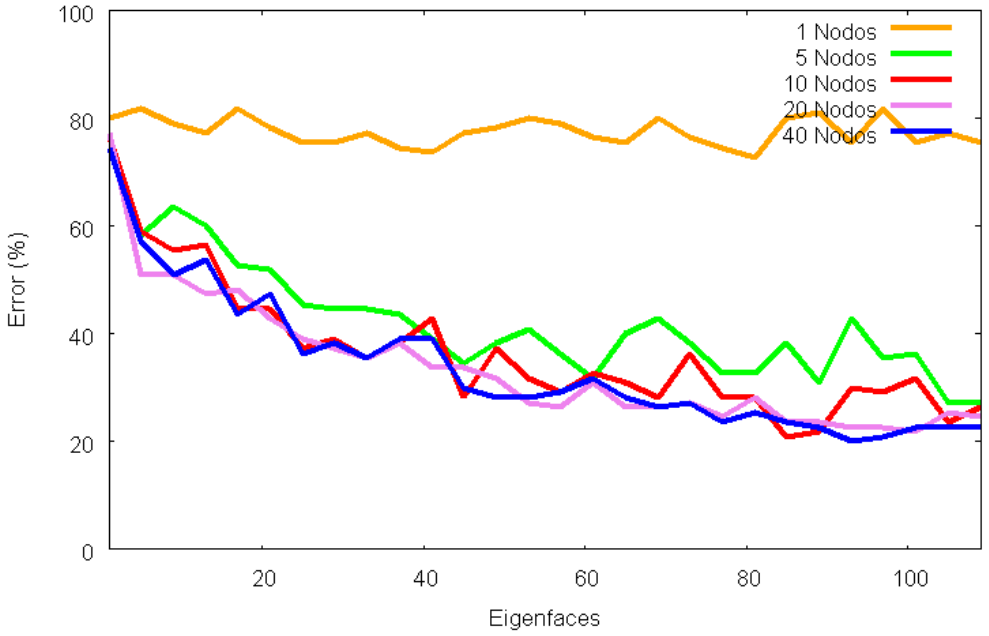


Figura 5.3: Tasa de error para cantidades específicas de nodos ocultos.

Tabla 5.2: Tasa de reconocimiento con un número de 60 entradas para MLP.

N° nodos ocultos	Reconocimiento (%)
1	15,45
2	33,63
3	50,91
⋮	⋮
9	67,27
10	69,09
11	74,54
12	74,54
13	70,00
⋮	⋮
38	70,91
39	70,00
40	69,09

das a la red. Luego se hizo el proceso inverso, es decir, variando el número de eigenfaces para cantidades específicas de nodos ocultos (Figura 5.3).

Observando ambas figuras se puede notar que, a partir de 10 nodos en la capa oculta (para cualquier cantidad de eigenfaces), el error se mantiene aproximadamente constante. La misma situación se presenta a partir de un número de 60 eigenfaces (para cualquier cantidad de nodos en la capa oculta). Además, como se vió en el Scree Test, con alrededor de 60 eigenfaces se representa casi la totalidad de la información, por lo que se decide generar el conjunto de entrenamiento a partir de esta cantidad para la comparación con otros métodos de clasificación. En la Tabla 5.2 se observan los valores de reconocimiento con un número de 60 entradas para el MLP.

5.1.3 Evaluación del vecino más cercano

El método más simple para determinar qué clase describe mejor una imagen facial es la técnica del vecino más cercano, la cual se basa en encontrar la clase k que minimize la distancia Euclídea en el espacio de caras entre la imagen facial y un prototipo Ω_k que representa la clase

$$\epsilon_k{}^2 = \|(\Omega - \Omega_k)\|^2.$$

Cada prototipo Ω_k es calculado mediante el promediado de los patrones de entrenamiento de la clase correspondiente. Finalmente, el rostro es clasificado como perteneciente a la clase k cuando el mínimo ϵ_k está por debajo de un umbral θ_ϵ, caso contrario se dice que éste es desconocido.

Entonces, a partir de esta técnica, se calculan las tasas de reconocimiento y rechazo sobre el conjunto de entrenamiento generado con 60 eigenfaces, usando validación cruzada con cinco particiones. La tasa de rechazo se calcula sobre el total de imágenes de prueba, mientras que la de reconocimiento sobre el total de imágenes no rechazadas. Los resultados obtenidos se observan en la Tabla 5.3.

Tabla 5.3: Tasa de reconocimiento para vecino más cercano.

θ_ϵ	Reconocimiento (%)	Rechazo (%)
∞	46,36	0
390000	47,02	4,54
360000	46,23	6,36
330000	46,35	9,09
300000	47,19	10,91
270000	46,96	16,36
240000	47,97	22,72
210000	50,87	30,00
180000	53,23	42,72
150000	59,32	59,09
120000	60,85	72,72

5.1.4 Evaluación de la red de función de base radial

Otro método de clasificación es la red de función de base radial. Como se indicó anteriormente, se asemejan a las redes MLP en el sentido de que son redes hacia adelante y pueden ser utilizadas para clasificación.

Con el propósito de comparar el desempeño de este tipo de redes contra el MLP implementado, se diseña una red RBF utilizando el software Weka, mencionado anteriormente. La red se diseñó con un número de 60 entradas y una cantidad de 11 nodos en la capa de salida. Luego, se la entrenó mediante el algoritmo de clustering K-means para diferentes cantidades de bases, estimándose el error de clasificación (Tabla 5.4). Cabe mencionar que se utilizó el mismo conjunto de entrenamiento que para el MLP.

Tabla 5.4: Tasa de reconocimiento para RBF.

Nº bases	Reconocimiento (%)
1	56,18
2	59,64
3	57,82
4	59,27
5	57,45
6	56,91
7	57,64
8	59,64
9	59,64
10	59,64

5.2 Análisis de resultados

De la evaluación del clasificador MLP se logra ver que a partir de 10 nodos en la capa oculta la curva de error tiende a ser constante para las diferentes gráficas presentadas en la Figura 5.2. Ésto también puede ser observado en la Figura 5.3, donde las gráficas tanto para 10, 20 y 40 nodos ocultos son parecidas.

Un caso similar se presenta para las eigenfaces. Con 60 características extraídas, por imagen facial, el porcentaje de error no varía en gran magnitud respecto a un número mayor de eigenfaces (Figura 5.3). Las gráficas correspondientes a 50, 60 y 70 eigenfaces, en la Figura 5.2, demuestran este hecho. Ésto se debe a que las componentes principales de menor orden contienen la mayor información del conjunto de imágenes faciales, careciendo de relevancia las restantes. La técnica del Scree Test evidencia que en el rango de 50 y 60 eigenfaces se representa alrededor del 95 % de la información, cantidad suficiente para el reconocimiento.

Si bien con 110 eigenfaces se obtiene el error mínimo, para no tener excesiva cantidad de pesos a entrenar en la red, se decide utilizar 60 eigenfaces para la proyección en el espacio de caras de cada rostro a identificar, y establecer una configuración de 11 nodos en la capa oculta del modelo MLP. De esta manera, se obtuvo una tasa de reconocimiento del 74,54 % (Ver Tabla 5.2).

Del experimento con la técnica del vecino más cercano, se observa en la Tabla 5.3, que a medida que θ_ϵ disminuye, la tasa de reconocimiento aumenta pero la de rechazo también lo hace. Ésto ocurre debido a que la mínima distancia no se encuentra dentro del radio de alguno de los puntos que representan las clases. De cualquier modo, considerando un rechazo igual a cero como en el MLP, se obtuvo una tasa de reconocimiento del 46,36 %.

Para el caso de la red RBF, la mayor cantidad de aciertos se logró con 2 funciones de base radial consiguiendo una tasa de reconocimiento del 59,64 %, como se aprecia en la Tabla 5.4. De cualquier modo, el rendimiento obtenido fue muy similar para las diferentes cantidades de bases analizadas.

Comparando los resultados finales se destaca el MLP ya que obtiene un mejor desempeño que los otros dos clasificadores analizados. En la Tabla 5.5 se detalla el mejor desempeño obtenido por cada uno. De aquí se observa que el MLP mejora el reconocimiento en un 14,90 % con respecto a la red RBF y en un 28,18 % al vecino más cercano, para la base de datos de fotos analizada.

Por otro lado, se compararon dos arquitecturas diferentes de redes neuronales, teniendo fijas la cantidad de características (número de entradas) y la cantidad de salidas. Se ajustó cada clasificador para lograr el mejor desempeño posible, sabiendo que probablemente la cantidad de parámetros sea diferente. El foco de análisis estuvo puesto en comparar el MLP contra la mejor RBF, y una posible prueba futura sería comparar ambas arquitecturas a igual cantidad de parámetros.

Tabla 5.5: Mejor rendimiento de cada clasificador.

Clasificador	Reconocimiento (%)
MLP	74,54
RBF	59,64
Vecino más cercano	46,36

sinc(*i*) Research Institute for Signals, Systems and Computational Intelligence (fich.unl.edu.ar/sinc)
G. M. Scarel, C. E. Martínez & G. Stegmayer; "Sistema de reconocimiento facial (Undergraduate project)"
Facultad de Ingeniería y Ciencias Hídricas - Universidad Nacional del Litoral, 2010.

CAPÍTULO 6

Conclusiones y desarrollos futuros

6.1 Conclusiones

En este trabajo se ha implementado un sistema gráfico de reconocimiento facial que contempla todas las etapas requeridas, desde la interfaz gráfica de usuario para la captura de la imagen hasta su clasificación, pasando por la localización del rostro y la extracción de características, identificando al usuario frente a la cámara.

El sistema realizado cumple con los objetivos propuestos logrando un rendimiento aceptable dado el ambiente en el cual se presentó el problema. Éste presenta una interfaz gráfica sencilla que permite operar fácilmente tanto a un administrador del sistema a la hora de tomar fotos como a un usuario que intenta identificarse. A través de los experimentos se demostró que el clasificador MLP se desempeñó mejor frente a una red RBF y a la técnica del vecino más cercano, logrando un mayor reconocimiento.

6.2 Desarrollos futuros

Con la idea de mejorar el desempeño general del sistema, una mejora estaría dada por un montaje dedicado en el lugar de instalación del mismo. Ésto implicaría colocar un fondo fijo de color blanco y un flash como fuente de luz para lograr luminosidad constante permitiendo, de esta manera, obtener una mejor calidad en las imágenes capturadas.

Por otro lado, una ampliación del sistema estaría dada por el desarrollo de un módulo de administración de usuarios a fin de automatizar el proceso de carga de los mismos. Éste incluiría funcionalidades para el registro de nuevos usuarios, dar de baja aquellos que ya no se requieran en el sistema así como también asociar las imágenes capturadas a los usuarios correspondientes.

Bibliografía

[BHK97] P. N. Belhumeur, J. P. Hespanha, and D. J. Kriegman. Eigenfaces vs. fisherfaces: Recognition using class specific linear projection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 19:711–720, 1997.

[BRJ98] G. Booch, J. Rumbaugh, and I. Jacobson. *The Unified Modeling Language User Guide*. Addison Wesley, 1998.

[GW02] R. González and R. Woods. *Digital Image Processing*. Prentice Hall, 2002.

[Hay99] S. Haykin. *Neural Networks: A Comprehensive Foundation*. Prentice Hall, 1999.

[HH93] D. R. Hush and B. G. Horne. Progress in supervised neural networks. *IEEE Signal Processing Magazine*, 10:8–39, 1993.

[Jac91] J. E. Jackson. *A User's Guide To Principal Components*. John Wiley and Sons, New York, 1991.

[JDM00] A. K. Jain, R. P. W. Duin, and J. Mao. Statistical pattern recognition: A review. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22:4–37, 2000.

[JMM96] A. K. Jain, J. Mao, and K. M. Mohiuddin. Artificial neural networks: A tutorial. *IEEE Computer*, 29:31–44, 1996.

[KMB07] P. Kakumanu, S. Makrogiannis, and N. Bourbakis. A survey of skin-color modeling and detection methods. *Pattern Recognition*, 40:1106–1122, Elsevier, 2007.

[KPS97] J. Kovac, P. Peer, and F. Solina. Human skin colour clustering for face detection. *Computer as a Tool, EUROCON 2003*, 2:721–732, 1997.

[ML06] O. Müller and S. Long. Verificación biométrica automática de identidad mediante reconocimiento facial. *Proyecto final*, Facultad de Ingeniería y Ciencias Hídricas. Universidad Nacional del Litoral, Argentina, 2006.

[PMS94] A. Pentland, B. Moghaddam, and T. Starner. View-based and modular eigenspaces for face recognition. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 84–91, 1994.

[SP98] K. K. Sung and T. Poggio. Example-based learning for view-based human face detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 20:39–51, 1998.

[TP91] M. Turk and A. Pentland. Eigenfaces for recognition. *Journal of Cognitive Neuroscience*, 3:71–86, MIT Press, 1991.

[VJ04] P. Viola and M. J. Jones. Robust real-time face detection. *International Journal of Computer Vision*, 57:137–154, Springer, 2004.

[WFKM97] L. Wiskott, J. M. Fellous, N. Kuiger, and C. von der Malsburg. Face recognition by elastic bunch graph matching. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 19(7):775–779, 1997.

[YHC92] A. L. Yuille, P. W. Hallinan, and D. S. Cohen. Feature extraction from faces using deformable templates. *International Journal of Computer Vision*, 8(2):99–111, Springer, 1992.

[ZCPR03] W. Zhao, R. Chellappa, P. J. Phillips, and A. Rosenfeld. Face recognition: A literature survey. *ACM Computing Surveys*, 35:399–458, 2003.

[ZJN07] J. Zou, Q. Ji, and G. Nagy. A comparative study of local matching approach for face recognition. *IEEE Transactions on Image Processing*, 16:2617–2628, 2007.