UNIVERSIDAD NACIONAL DEL ALTIPLANO

FACULTAD DE INGENIERÍA ESTADÍSTICA E INFORMÁTICA

ESCUELA PROFESIONAL DE INGENIERÍA ESTADÍSTICA E INFORMÁTICA

SISTEMA PARA DETECCIÓN Y RECONOCIMIENTO FACIAL UTILIZANDO TÉCNICAS HÍBRIDAS EN IMÁGENES Y SECUENCIAS DE VIDEO PUNO 2013

TESIS

PRESENTADA POR:

Bach. RENZO APAZA CUTIPA

Bach. GINA FIORELLA CHARAJA SANCHEZ

PARA OPTAR EL TÍTULO PROFESIONAL DE:

INGENIERO ESTADÍSTICO E INFORMÁTICO

PUNO – PERÚ

2013

UNIVERSIDAD NACIONAL DEL ALTIPLANO – PUNO

FACULTAD DE INGENIERÍA ESTADÍSTICA E INFORMÁTICA

ESCUELA PROFESIONAL DE INGENIERÍA ESTADÍSTICA E INFORMÁTICA

TESIS

SISTEMA PARA DETECCIÓN Y RECONOCIMIENTO FACIAL UTILIZANDO TÉCNICAS HÍBRIDAS EN IMÁGENES Y SECUENCIAS DE VIDEO PUNO 2013

Presentada por:

Bach. RENZO APAZA CUTIPA

Bach. GINA FIORELLA CHARAJA SANCHEZ

A la Coordinación de Investigación de la Facultad de Ingeniería Estadística e Informática de la Universidad Nacional del Altiplano - Puno, para optar el Título Profesional de:

INGENIERO ESTADÍSTICO E INFORMÁTICO

APROBADA POR:

PRESIDENTE : Dra. María Maura SALAS PILCO

PRIMER MIEMBRO : M. Sc. Edgar Eloy CARPIO VARGAS

SEGUNDO MIEMBRO : M. Sc. Leonel COYLA IDME

DIRECTOR : M. Sc. Ernesto Nayer TUMI FIGUEROA

ASESOR : M. Sc. Alejandro APAZA TARQUI

Área : Informática
Tema : Inteligencia artificial
Fecha de Sustentación : 27/12/2013

DEDICATORIAS

A Dios por brindarme la vida, permitirme compartirla con las personas que más quiero y guiar mi camino.

A mi madre Cristina su amor, fortaleza y confianza hacen la diferencia siempre exiges más de mi porque sabes que puedo hacerlo, a mi Padre Hugo por sus palabras y aliento, su persona es el mejor ejemplo que un hijo podría tener.

A mi esposa Fiorella por su amor y apoyo incondicional, a mis hermanos Roger, Hugo, Leli y Brian por su permanente apoyo y aliento al desarrollo personal y profesional.

A todos mis amigos especialmente a Joel, Andrés y Fredy con quienes aprendimos experimentando y su compañía hizo esta experiencia amena y agradable.

Renzo Apaza Cutipa.

Dedico el presente trabajo a Dios, quien siempre guía mis pasos e ilumina mi camino con su inmenso amor y su luz.

A mi esposo Renzo quien siempre me brindo su amor incondicional y creyó firmemente en mí, a mi madre Marilia por el don de la vida y su lucha constante, y a mi padre Ernesto por enseñarme que nunca es tarde para ser grande.

A mi abuelita Dina por sus cuidados, paciencia y amor brindados, a mis tíos: Consuelo, Martha y Oscar quienes inculcaron valores que formaron mi ética personal.

A mis enanos Toddy, Winny, Prizzila y Mateo quienes siempre inyectaron felicidad a mi vida, siendo una parte esencial y vital de ella.

Y a mis amigos Joel, Andrés (Chino), Teresa, quienes fueron y son al apoyo y la alegría más grande a lo largo de los años que nos conocemos.

Gina Fiorella Charaja Sanchez.

AGRADECIMIENTOS

A la Facultad de Ingeniería Estadística e Informática de la Universidad Nacional del Altiplano.

A todos los docentes de la Facultad de Ingeniería Estadística e Informática por sus enseñanzas durante todos los años de formación universitaria.

A los investigadores del área de ciencias de la computación del mundo, a los miembros de Stackoverflow la información desarrollada constantemente en su site http://stackoverflow.com/ nos ayudaron a pasar de la teoría al código fuente

.

ÍNDICE

CAPITULO IV RESULTADOS

ÍNDICE DE FIGURAS

RESUMEN

Reconocer la identidad de un individuo de forma automática es aún, una tarea que no logra alcanzar una tasa de éxito del 100%, por lo que el presente trabajo buscó mejorar la tasa de reconocimiento; dando énfasis a los métodos de extracción y clasificación de características.

En este sentido, se propuso mejorar el ratio para el reconocimiento de rostros mediante la representación de las imágenes, utilizando la transformada Wavelets de Gabor sobre las imágenes en escala de grises obtenida luego de normalizar las imágenes originales, posteriormente a la nueva representación obtenida se le aplica la técnica de Análisis de Componentes Principales (PCA) para obtener y constituir luego el vector de características de las imágenes de rostros. A continuación se aplica un clasificador basado en Maquinas de Soporte Vectorial (SVM). El método fue probado sobre una base de datos de imágenes de rostros constituida entre los bancos de rostros FERET, ORL e imágenes obtenidas por los responsables de la investigación.

Se concluye que la combinación de las técnicas Transformada Wavelet de Gabor y Análisis de Componentes Principales en el proceso de extracción de características y la clasificación de imágenes basada en Maquinas de Soporte Vectorial, logran una tasa de reconocimiento superior al 95%.

Palabras clave: Reconocimiento de Rostros, Wavelet de Gabor, Análisis de Componentes Principales (PCA), Maquina de soporte Vectorial (SVM), Vector de Características.

ABSTRACT

The Automatic Face Recognition is still, a task that does not achieve a success rate of 100%, so this study sought to improve the recognition rate; emphasizing methods features extraction and classification.

Thus it was proposed to improve the ratio for face recognition by representing images using the Gabor Wavelet Transform on images in grayscale obtained after normalizing the original images, then the new representation obtained is applied the technique of Principal Component Analysis (PCA) to obtain and then constitute the feature vector of the face images. Then based on Support Vector Machines (SVM) classifier is applied. The method was tested on a database consists of face images from FERET and ORL face banks and images obtained by those responsible for the investigation.

We conclude that the combination of the techniques of Gabor Wavelet Transform and Principal Component Analysis in the process of feature extraction and image classification based on Support Vector Machines, achieve a recognition rate above 95%.

Keywords: Face Recognition, Gabor Wavelet, Principal Component Analysis (PCA), Support Vector Machine (SVM), Feature vector.

INTRODUCCIÓN

Con el progreso de las tecnologías asociadas a la información, labores que tradicionalmente eran realizadas por seres humanos son, gracias a las mejoras tecnológicas, realizadas por sistemas automatizados. Dentro de la amplia gama de posibles actividades que pueden automatizarse, aquella relacionada con la capacidad para establecer la identidad de los individuos ha cobrado una gran importancia y como consecuencia directa, la biometría se ha transformado en un área resaltante. Las técnicas de identificación biométrica permiten el reconocimiento de la identidad de un individuo basado en las características conductuales o en los rasgos físicos propios intrínsecos de los individuos, garantizando de esta manera que un individuo no pueda usurpar la identidad de otro. El reconocimiento de rostros es considerado por los investigadores como una técnica importante de la biometría, siendo esta la habilidad de identificar a los sujetos basados en sus características faciales.

La seguridad de las personas, bienes o información es una de las mayores preocupaciones de nuestra sociedad local, nacional e internacional hoy en día. En tal sentido son necesarios los medios de verificación de identidad para salvaguardar la seguridad e integridad personal, material e intelectual. Así mismo los actuales medios de verificación de identidad están relacionados a las pertenencias de los individuos como los documentos de identidad, tarjetas de crédito, pasaporte, etc. Sin embargo, las pertenencias personales pueden ser sustraídas sin autorización, dañadas físicamente o ser sometidas a procedimientos para revelar las contraseñas de estas mediante algoritmos de fuerza bruta que conlleva a la usurpación de la identidad.

En tal sentido la presente investigación aborda la detección y el reconocimiento automático de rostros haciendo énfasis en el proceso de extracción de características mediante la combinación de las técnicas Wavelets de Gabor y el Análisis de Componentes Principales para posteriormente realizar la identificación mediante un clasificador basado en Maquinas de soporte Vectorial.

El contenido del presente trabajo está organizado en seis capítulos y para cada uno de ellos se da a continuación una breve explicación de los puntos que en ellos se tratan.

En el Capítulo I se presenta una descripción sobre el problema que aborda este trabajo, el objetivo general, los objetivos específicos, así como también la hipótesis a justificar.

En el Capítulo II se despliega el Marco Teórico, mediante los Antecedentes hallados tanto a nivel Internacional así como también a Nivel Local; también mediante la literatura encontrada y estudiada se muestra la Base Teórica y la Definición de Términos Básicos y la Operacionalización de Variables.

En el Capítulo III se describe los Métodos utilizados para el tratamiento de las imágenes y Materiales en el cual se aprecia la Población, Muestra, los Métodos de recopilación y Tratamiento de datos, así como el Material Experimental. Se describe también el método de reconocimiento de rostros propuesto donde se detallan los procedimientos para la extracción de características utilizando la Wavelet de Gabor y el Análisis de Componentes Principales

El Capítulo IV muestra los Resultados obtenidos de aplicar los algoritmos propuestos a la implementación del software, se detallan el análisis de los

requisitos, el diseño de los módulos, la arquitectura e implementación del software. Y a continuación se realiza la contratación de la hipótesis del presente trabajo de investigación.

Por último se presentan las recomendaciones y sugerencias para la realización de trabajos futuros.

CAPÍTULO I

PLAN DE INVESTIGACIÓN

1.1. PROBLEMA

1.1.1. Formulación del Problema

El problema frecuente es la identificación de un individuo debido a que este puede manipular o adulterar su identidad y comprometer la seguridad e integridad de algún tipo de información o bien.

La aproximación para el reconocimiento y la identificación de un individuo mediante el reconocimiento automático de rostros ha conllevado al estudio de diversas técnicas y métodos, en general todos estos buscan superar las dificultades para la detección y reconocimiento de rostros mediante el uso de un computador y el uso de bancos de datos para inferir sobre la identidad de un individuo.

Considerando las condiciones expuestas y la importancia de la evolución de las técnicas de reconocimiento facial, la presente investigación busca resolver en primera instancia dada una imagen la detección de rostros para luego realizar el reconocimiento e identificación de un individuo mediante el uso de las técnicas híbridas Wavelet de Gabor y Análisis de Componentes Principales en el proceso de extracción de características y

a continuación realizar el proceso de clasificación mediante Maquinas de Soporte Vectorial para resolver la siguiente interrogante:

¿Cuál es la eficiencia de la detección y reconocimiento automático de rostros aplicando técnicas hibridas en imágenes y secuencias de video?

1.2. OBJETIVOS

1.2.1. Objetivo General

Desarrollar un Sistema para la detección y reconocimiento facial en base a técnicas híbridas en imágenes y secuencias de video y determinar su eficiencia.

1.2.2. Objetivos Específicos

- Realizar la adquisición de imágenes y secuencias de video.
- Crear un módulo para la detección de rostros en imágenes y secuencias de video.
- Crear un módulo para la extracción de características basado en la transformada Wavelet de Gabor y Análisis de Componentes Principales.
- Crear un módulo para realizar la clasificación de rostros basado en - Maquinas de Soporte Vectorial.

1.3. HIPÓTESIS

Data la formulación del problema, el presente trabajo buscó demostrar la siguiente hipótesis:

"La inclusión de las técnicas hibridas en la detección y reconocimiento de rostros permite que el sistema sea eficiente en imágenes y secuencias de video".

CAPITULO II

MARCO TEORICO

2.1. ANTECEDENTES DE LA INVESTIGACIÓN

2.1.1. Antecedentes de Nivel Internacional

- Sirovich & M. Kirby (1987), *Low-Dimensional procedure for the characterizacion of human faces*, en Journal of the optical society of America.

  Presenta un tratamiento basado en un método conocido como Karhunen. Loeve expansión en reconocimiento de patrones y Análisis de Componentes Principales en estadística. La aplicación de este procedimiento, especialmente en el análisis de señales en el dominio del tiempo, El estudio demuestra que cualquier rostro en particular puede ser representado económicamente en términos de un mejor sistema de coordenadas que denominamos eigenpictures. Estos son los eigenfunctions del promedio de covarianzas del conjunto de rostros. Para dar una idea de la compresión de datos ganada de este procedimiento, se observa que una representación aceptable de una imagen de rostro puede ser construida de una especificación de niveles de gris en 2^{14} pixeles. Como muestra de la implementación luego de proporcionar 40 valores de combinaciones de eigenpictures se

caracteriza un rostro con un 3% de error. Así en principio, cualquier colección de rostros seria clasificado almacenando una pequeña colección de números por cada rostro y un pequeño conjunto de eigenpictures.

- Turk & Pentland (1991), Face Recognition Using Eigenfaces, en la Conferencia IEEE sobre Computer Vision and Pattern Recognition Hawaii. La investigación concluye:

  El sistema presentado puede en pocos segundos reconocer muchas vistas con pequeñas variaciones, donde al menos una de las vistas caiga cerca de una de las vistas características.

  El enfoque basado en eigenface para el reconocimiento de rostros fue motivado por la Teoría de la Información, que lleva la idea de basar el reconocimiento de rostros en función de pequeños conjuntos de características de las imágenes, sin requerir que correspondan a una noción intuitiva de partes y características del rostro. Aunque no es una solución refinada al problema general de reconocimiento de objetos, el enfoque basado en eigenface proporciona una solución práctica que está bien equipada para el problema de reconocimiento de rostros. Es rápido, relativamente simple y ha demostrado que trabaja bien en un ambiente controlado.

- Chengliang Wang, Libin Lan, Yuwei Zhang & Minjie Gu (2011), Face Recognition Based on Principle component Analysis and Support Vector Machine, en la Conferencia IEEE Intelligent Systems and Applications (ISA). La investigación indica que:

Se presta especial atención a la extracción de características y a la clasificación, con el objetivo de mejorar la tasa de reconocimiento, el Análisis de componentes Principales (PCA) es utilizado para extraer las características de las imágenes, y la Maquina de Soporte Vectorial (SVM) es utilizada para tratar con el problema de reconocimiento de rostros. Toman el PCA y SVM para experimentar sobre la base de datos de rostros Cambrige ORL, y comparado el método con PCA y Nearest Neighbor y SVM sobre la tasa de reconocimiento y tiempo de reconocimiento respectivamente. Finalmente, los resultados experimentales muestran que la tasa de reconocimiento de este método, en una muestra pequeña de distintas condiciones, es mejor que los otros dos métodos. Mostrando que, para el reconocimiento de rostros el utilizar las características obtenidas luego de aplicar PCA para proporcionarlas al clasificador con SVM es factible y correcto.

- Lades, Vorbrüggen, Buhmann, Lange, Malsburg, Wüstz y Konen (1993), Distortion invariant object recognition in the dynamic link architecture, en IEEE Transactions on Computers, indica que:

  La Dynamic Link Architecture 1 deriva su poder a partir de un formato de datos basado en estructuras enlazadas sintácticamente. Esta capacidad es explotada en tres niveles. Primero cuando una imagen es formada en el dominio de la imagen, el detector de características locales centrado en uno de sus puntos son incluidos para formar un detector de características compuestas llamado jet. Un detector de

[1] Dynamic Link Architecture es una extensión de una clásica Red Neuronal Artificial.

características compuestas puede ser transportado al dominio del modelo y puede ser comparado como un todo con otros detectores de características compuestas. Esto libera al sistema de la necesidad de entrenar nuevas neuronas individuales como detectores para características complejas antes nuevos clases de objetos podrían estado reconociendo, un mayor peso que un sistema de capas convencional. En segundo lugar los enlaces son utilizados para representar las relaciones del vecindario con el dominio de la imagen y con el dominio del modelo. De este modo los objetos neuronales adquieren estructura interna, y sus comunicaciones pueden ahora ser limitadas a combinaciones con la estructura sintáctica emparejada. Esto forma las bases para elastic graph matching[2]. Finalmente, el enlace dinámico entre el emparejamiento de grafos, que en el presente contexto es de poca importancia para el reconocimiento, pero que será útil para etiquetar la imagen con todos los patrones reconocidos y crear representaciones de composición de objetos y escenas.

Debido a que las Wavelets de Gabor están diseñadas para ser robustas con respecto a pequeñas distorsiones, incluyendo cambios en tamaño y en orientación, el sistema también generaliza sobre esos cambios. Desde que los kernel de wavelet son versiones dilatados y rotados uno de otros, debería ser posible también generalizar estas conexiones sobre tamaño y orientación. Esto llevaría sin embargo a conexiones densas entre nodos y conexiones individuales menos

[2] Elastic Graph Matching es un algoritmo inspirado biológicamente para el reconocimiento de objetos en el área de visión por computador.

específicas entre características, y elastic graph matching se convertiría en difícil o imposible. Una alternativa es introducir parámetros globales para orientación y tamaño y difundirse durante el emparejamiento, de la misma manera la posición de la grilla es difundida en el primer paso del presente procedimiento de emparejamiento. Es interesante notar que en ese contexto el tiempo de reacción de sistema visual humano crece para objetos en orientaciones no esperadas. Esto sugiere que nuestro sistema visual no es invariante a la orientación en el mismo grado que tiene la invarianza de traslación.

Una cuestión algo más compleja es la invarianza con respecto a la perspectiva del movimiento. El sistema está basado en una representación bidimensional. Esencialmente diferentes vistas tridimensionales de un objeto son reconocidas con la ayuda de múltiples vistas. Esto es sin embargo, obligatorio que el reconocimiento sea robusto con respecto a pequeños cambios en la perspectiva. Un sistema basado en Wavelets de Gabor y mapas topológicos es ideal como se ha demostrado.

2.1.2. Antecedentes a nivel local y nacional

- Cayo (2005), Prototipo Neurogenético aplicado en el reconocimiento de imágenes bidimensionales estáticas: Rostros, Código de Barras y Firmas, Tesis de Maestría en Informática. Puno – Perú. Universidad Nacional del Altiplano.

  Los sistemas de Redes Neuronales artificiales (o ANS, Artificial Neural Systems) están inspiradas en la funcionalidad de las neuronas

biológicas, aplicados al reconocimiento de patrones que las convierten aptas para modelar y efectuar predicciones en sistemas muy complejos.

Para la construcción del prototipo se ha considerado los siguientes aspectos: modelo de neurona, arquitectura o topología de conexión, y algoritmo de aprendizaje. Siendo la más óptima por su simplicidad en el aprendizaje para la red, el modelo supervisado unidireccional "backpropagation supervised-Learning", ya que hace backtracking hacia la entrada una vez que tiene los datos en la salida. Al igual que las redes neuronales, los algoritmos genéticos también cuentan con la capacidad para resolver un problema específico, pero a diferencia de los parámetros-red, cualquier presentación puede ser expresada como una cadena de palabras de longitud fija (genotipo) sobre una finita (típicamente el binario) a ser usada. La interpretación (es decir el phenotype) de cadenas (genotipo) no es relevante para el algoritmo, inclusive si estos son: los parámetros de diseño del motor de inferencia, la estrategia a utilizar o, en cuanto a este proyecto, los pesos y bias de la red neuronal.

Los algoritmos genéticos son métodos sistemáticos para la resolución de problemas de búsqueda y optimización, que aplican a estos los mismos métodos de la evolución biológica: selección basada en la población, reproducción sexual y mutación. Los algoritmos genéticos son métodos de optimización, que tratan de resolver el mismo conjunto de problemas que se ha contemplado anteriormente, es decir, el objetivo es hallar (xi,.....,xn) tales que F(xi,.....,xn) sea máximo. En un

algoritmo genético tras parametrizar el problema en una serie de variables, (xi,.....,xn) se codifican en un cromosoma. Todos los operadores utilizados por un algoritmo genético se aplicarán sobre estos cromosomas, o sobre poblaciones de ellos.

El proyecto permite analizar y decodificar mediante técnicas inteligentes los medios de seguridad gráficos y obtener así la recuperación y verificación de datos de una persona garantizando un alto grado de eficiencia y certeza e la validación de los datos. La realización del procesamiento de imágenes consume demasiado recurso de memoria y tiempo de procesamiento, motivo por el cual la ejecución de redes neuronales sobre éstas implica optimizar el proceso de aprendizaje de tal forma que se consiga una convergencia más veloz. Las búsquedas de imágenes dentro de bases de datos pueden simplificarse mediante el empleo del prototipo del Sistema Híbrido Neurogenético, no necesariamente puede ser aplicado a firmas personales sino también a otros patrones similares: reconocimiento de rostros y código de barras. La técnica híbrida que se implementó en el prototipo fue de gran utilidad demostrando una vez más las ventajas de combinar las funcionalidades de dos técnicas de la Inteligencia Artificial, que dieron como resultado sacar lo mejor de ellas en la implementación del sistema.

- Carranza y Florián (2008), Extracción de características para la recuperación de imágenes médicas por contenido utilizando las

Wavelets de Gabor, Universidad Nacional de Trujillo. Facultad de Ciencias Físicas y Matemáticas. Trujillo Perú.

Los autores concluyen que a través de de la representación discreta de una familia de wavelets de Gabor es posible definir filtros que ayuden a extraer las características más resaltantes en una imagen dada, con respecto a la comparación de resultados obtenidos por todas las técnicas como SIFT3 aplicadas directamente a imágenes tomograficas no presentan resultados adecuados, sin embargo proveen una respuesta rápida.

- Sucasaca Pacori (2007), Desarrollo y análisis de un Sistema de Análisis y Reconocimiento de Huellas Dactilares (Control Biométrico) para la verificación de ingresantes a la UNA Puno, Universidad Nacional del Altiplano. Facultad de Ingeniería Estadística e Informática. Puno Perú.

  El autor concluye que el manejo de una metodología CMMI apoya en mayor eficiencia al desarrollo del software, mediante la realización de un previo análisis de la huella dactilar, el uso de una librería dinámica universal la cual facilita el manejo de las diversas unidades lectores de huella dactilar, para lo cual en el caso de la validación (1:1) de los datos sea mayor velocidad esto debido a la participación del estudiante, y la identificación (1:N) que hace que se pueda identificar a los ingresantes que no porten documentos.

[3] Scale Invariant Feature Transform o SIFT es un algoritmo de visión por computador para detectar y describir características locales en imágenes.

2.2. BASE TEÓRICA

2.2.1. Consideraciones iniciales

El rápido incremento en los volúmenes de colecciones de imágenes digitales, demandan grandes cantidades de espacio para almacenar dichos datos. Administrar tales volúmenes de información es un proceso difícil y arduo, principalmente por el aspecto subjetivo y ambiguo e las imágenes. Lo que ha motivado el aumento de investigaciones relacionadas en esa área, como ejemplo la recuperación de imágenes por contenido (*Content-Based Image Retrieval* - CBIR) (Smeulders, 2000). Los sistemas CBIR en las imágenes puedes ser indexadas a través de contenido visual como color, textura y forma, donde las imágenes son representadas mediante una secuencia numérica llamada *vector de características*.

Un vector de características es una representación numérica sucinta de una imagen o parte de una imagen (un objeto) caracterizando medidas de aspectos representativos de un objeto. El número de características dependerá del domino de la imagen y de las propiedades que se desea caracterizar, ese número determinará la dimensión del vector de características (vector n-dimensional). Esta nueva representación de la imagen puede ser almacenada en una base de datos que permita una recuperación rápida de una imagen, sin embargo esta nueva representación debe de considerar las siguientes tres consideraciones (Loew, 2000):

- Reducir la dimensionalidad de los datos,

- Resaltar aspectos de la imagen para facilitar la percepción humana,

- Ser invariante a las transformaciones de la imagen

Una forma de abordar el problema es utilizar transformaciones de imagen, siendo la transformada wavelet la que ha obtenido alta aceptación por los investigadores del área. El proceso de transformación genera nuevos espacios de información sobre los cuales se aplican medidas estadísticas con valores numéricos, los que constituirán el vector de características.

Además de la trasformada wavelet existen otras transformadas lineales que permiten realizar la misma tarea, pero el presente trabajo utiliza la transformada wavelet (Gabor wavelet).

En esta sección se describen métodos estadísticos que soportan la creación de un conjunto de técnicas que permitan caracterizar las imágenes luego de aplicar la transformada wavelet. El concepto básico de transformar un conjunto de mediciones en un nuevo conjunto de características, o vector de características.

2.2.2. Definición de una Imagen

Una imagen digital es el resultado de un proceso de discretización (sampling) de una imagen de función continua I(x, y), y que es almacena en medios digitales como un arreglo de dos dimensiones I (m, n), donde m = 0, 1,…, Nx-1 y n = 0,1,…, Ny-1 lo que significa que es almacena como un arreglo de dimensiones Nx x Ny[4]. Cada elemento (m, n) del arreglo

[4] Esta definición es aplicable a imágenes en tono de grises.

corresponde a un pixel (elemento de una imagen) de la imagen, cuya intensidad es igual a I(m, n). Esta intensidad I(m, n) es cuantificada en Ng niveles discretos (niveles de gris), conocidos como profundidad de la imagen. Y la secuencia de niveles de gris I(m, n) puede tomar los siguiente valores, 0, 1, …, Ng-1, la profundidad de la imagen es generalmente una potencia de 2 y pueden ser valores grandes (por ejemplo 64 y 256). De cualquier forma, para el ojo humano es difícil distinguir los detalles y diferencias de las intensidades, en la práctica Ng = 32 o 16 es suficiente para representar las imágenes (Theodoridis, 1999).

La necesidad de generar vectores de características nace de la dificultad de poder trabajar con toda la información de la imagen. Por ejemplo, para una imagen de dimensión 64 x 64, el número de pixeles es de 4096. Para el caso de sistemas de recuperación de imágenes por contenido, este número es aún mayor. El proceso de generación de características es un proceso que calcula nuevas variables, a partir de la imagen original I (m, n). Este proceso procura generar características que muestren una gran cantidad de información contenidas en una imagen. Consecuentemente no será utilizado directamente el total de la información de la imagen I(m, n), las características extraídas permitirán codificar adecuadamente la información de la imagen.

2.2.3. Atributos de las Imágenes

Los sistemas de recuperación de imágenes utilizan el color, textura y forma para representar una imagen, en situaciones donde la información del color no es relevante para la discriminación, se requiere el uso de atributos de

textura y/o forma para recuperar la imagen, además los sistemas basados en un único atributo de la imagen podrían no alcanzar niveles de recuperación adecuados, por ello los sistemas procuran utilizar múltiplos atributos de las imágenes para su indexación y posterior recuperación (Vailaya, 2000).

Color

Existen diferentes representaciones de color que incluyen desde el tradicional RGB (red, green, blue – rojo, verde y azul), a modelos más simples que mapean directamente las características físicas del dispositivo de visualización, como HSI (hue, satutation, intensidad – tonalidad, saturación, intensidad) que se refiere al modelo de colores para la percepción humana.

Muchos de los trabajos de extracción de características están basados en histogramas de distribución de colores. Los histogramas de color son invariantes a traslaciones de la imagen, debido a que, con una normalización los histogramas obtienen también una invariancia a la escala. De cualquier forma, los histogramas de color no indican la localización espacial de los pixeles de una imagen (Gonzales, 2007).

Muchas operaciones pueden ser realizadas con los histogramas de color, trasladando parámetros y umbrales, por ejemplo la figura 3 representa el histograma de la imagen original (Figura 1). Se le aplica una operación de alargamiento (stretch) sobre la imagen, el histograma lucirá como la figura 4, teniendo como resultado una imagen más definida (Figura 2).

Algunos trabajos, que utilizan la distribución de colores para ensamblar el vector de características basado en los coeficientes wavelet son presentados por (Albuz, 2001), consiguiendo disminuir el tiempo de búsqueda de las imágenes similares, pero con una eficiencia de búsqueda no muy buena con un valor máximo de 35% de eficiencia.

Figura 1: Prizzilla original.

Figura 2: Prizzilla saturada.

Figura 3: Histograma prizzilla original.

Figura 4: Histograma prizzilla saturada.

Esta propuesta muestra algunas dificultades por el propio carácter no invariante de los histogramas de colores. Además, si la imagen contiene un número alto de colores, entonces el vector de características será mayor y la indexación de los vectores con esa dimensión se volverá un problema.

Textura

No existe un definición clara de textura, pero muchos autores coinciden en definirla como los cambios en las intensidades de la imagen que forman determinados padrones repetitivos (Tucerayan, 1993). Estos patrones pueden ser el resultado de las propiedades físicas de la superficie del objeto, o ser el resultado de diferentes factores de luz sobre la superficie. La figura 5 muestra algunos ejemplos de imágenes con textura. El reconocimiento de texturas es una tarea fácil para las personas, pero en procedimientos automáticos esta tarea muchas veces requiere de técnicas computaciones complejas.

Figura 5: Diferentes tipos de textura.

Un análisis de textura tiene por objetivo establecer la relación del vecindario de los elementos de la textura y su posicionamiento en relación a los demás

(conectividad). El número de elemento por unidad espacial (densidad) y sus regularidades (homogeneidad) (Traina, 2001).

Los modelos de textura descritos en la literatura pueden ser divididos en las siguientes clases (Tucerayan, 1993):

- Métodos Estadísticos, que definen la textura en términos de la distribución espacial de los valores de tonos de gris;

- Métodos Geométricos, compuestos caracterizados como “elementos de textura” o primitivas;

Métodos de procesamiento de Señales, que utilizan el análisis de frecuencias de la imagen para clasificar una textura. Este último esquema incluye el uso del dominio espacial (Malik, 1990) y dominio de filtros de Fourier (Coggins, 1985), así como el uso de filtros de Gabor y modelos de wavelets (Jain & farrokhnia, 1991) (Ma & Manjunath, 1995). Estudios han demostrado que sistemas que utilizan Gabor y multiresolución simultáneamente se presentan como una buena técnica en sistemas de recuperación e indexación por contenido (Ma & Manjunath, 1995) (Picard, 1995).

La utilización de la textura de la imagen tiene por objetivo permitir su segmentación en determinadas regiones, que posean una misma textura. Luego de definir las regiones, se puede utilizar rectángulos que los incluyan (Minimum Bounded Rectangle - MBR) dando lugar al uso de una estructura de indexación de tipo R-tree (Guttman, 1984). Pero de la misma forma que en el caso de colores, se presentan dificultades con la invariancia de la

dimensionalidad del vector de características, propiciando una dimensionalidad alta.

Forma

Se debe de indicar que los sistemas de recuperación de imágenes por contenido, el atributo forma es una aproximación que muestra mayor dificultad, especialmente por el hecho de tener que segmentar y conocer el tamaño de los objetos contenidos en la imagen. Por ejemplo, en aplicaciones médicas, la forma y tamaño de los tumores es de mucha importancia a la hora de clasificarlos como malignos o benignos. Los tumores con bordes irregulares tienen una alta probabilidad de ser malignos y aquellos que muestran bordes regulares generalmente son benignos (Theodoridis, 1999).

Figura 6: Formas de imágenes.

Existen dos aproximaciones por las que se puede obtener la caracterización de la imagen por la forma (Theodoridis, 1999). Una es desarrollar una técnica que devuelva una descripción total del borde del objeto (Coeficiente de Fourier) y la otra es utilizar cualidades que describan

las características morfológicas de la región (cantidad de esquinas en los bordes). Este enfoque son las características basadas en momentos.

2.2.4. Características Extraídas de Imágenes 2D

A continuación se presentan algunas caracterizaciones estadísticas que describen la forma del histograma, y luego aquellas que describen la regiones (textura y forma).

Estadística Descriptiva Para Imágenes

Las características más básicas en toda imagen son las medidas que definen la amplitud de la imagen en términos de luminosidad, valor espectral y otras unidades. La forma en que se presenta el histograma de color proporciona información sobre la naturaleza de la imagen. Por ejemplo un histograma con una distribución estrecha indicaría que se trata de una imagen de bajo contraste. Un histograma bimodal indica que una imagen contiene un objeto que tiene una amplitud más estrecha en relación al fondo de la imagen. Estas medidas pueden ser cuantificadas como se describe a continuación (Theodoridis, 1999).

Sea I una variable aleatoria que representa los niveles de gris en una región de interés, el histograma P (I) está definido como:

$$P(I) = \frac{número\ de\ pixeles\ con\ el\ nivel\ de\ gris\ I}{número\ total\ de\ pixeles\ en\ la\ región} \qquad (2.1)$$

Esto significa que P (I) es una fracción de pixeles con niveles de gris I. El grafico de un histograma (donde P (I) representa el porcentaje de I para I =

0, 1,…, N_g-1 representa la intensidad) muestra la percepción de la distribución de colores.

Las características que describen la forma de un histograma son muy útiles y están proporcionados de los siguientes descriptores estadísticos, entre otros (Loew, 2000) (Theodoridis, 1999):

Momentos

$$m_i = E[I^i] = \sum_{I=0}^{N_g-1} I^i P(I), \quad i = 1, 2, \ldots \qquad (2.2)$$

Donde $m_0 = 1$ representa el momento cero. Si i = 1 entonces m_1 = E [I] que representa la media del valor I.

Momentos Centrales

$$\mu_i = E[(I - E[I])^i] = \sum_{I=0}^{N_g-1} P(I)(I - m_1)^i \qquad (2.3)$$

Donde las variables involucradas poseen el mismo significado anterior. El segundo momento central μ_2 es la varianza σ^2 que es una descripción de la uniformidad de una determinada región de la imagen.

El tercer momento central es una medida de no centralismo (skewness) que calcula el grado de asimetría de un histograma alrededor de la media, a veces denotada como σ^3. El cuarto momento central (kurtosis) mide la monotonía relativa de la imagen. Dependiendo del valor de este momento el histograma será denominado platocurtica, para valores grandes, o leptocurtica, para valores pequeños. En caso de una distribución normal es denominada como mesocurtica.

Energía

$$E = \sum_{I=0}^{N_g-1} [P(I)]^2 \qquad (2.4)$$

Entropía

$$H = -\sum_{I=0}^{N_g-1} P(I) log_2(I) \qquad (2.5)$$

Descripción de Regiones

Las regiones son típicamente definidas en base a su homogeneidad interna y a algunas características adicionales. En esta situación la escala es una importante definición de esa homogeneidad, características fractales pueden devolver información que otros tipos de características no pueden.

Textura y la Matriz de Co-Ocurrencia.

Las características resultantes de los descriptores estadísticos mostrados en 2.2.1.4.1 proporcionan la información relacionada a la distribución de los niveles de gris dentro de la imagen, pero no otorgan información acerca de la posición relativa de los niveles de gris dentro de la imagen. Si todos los valores bajos de nivel de gris se localizan juntos, o están alternados con los valores altos. Ese tipo de información puede ser extraído de los histogramas de segundo orden, donde los pixeles son considerados en pares, Dos parámetros adicionales son considerados, uno que determina una distancia relativa entre los pixeles y su orientación relativa.

Sea d la distancia relativa medida en número de pixeles (d=1 para pixeles vecinos, etc.). Una orientación ø está cuantificada en cuatro direcciones: horizontal, diagonal, vertical y anti diagonal (0°, 45°, 90°, 135°), como indica la figura 7. Para cada combinación de d e ø definen un histograma de dos dimensiones.

$$0°: P(I(m,n) = I_1, I(m \pm d, n) = I_2) \qquad (2.6)$$

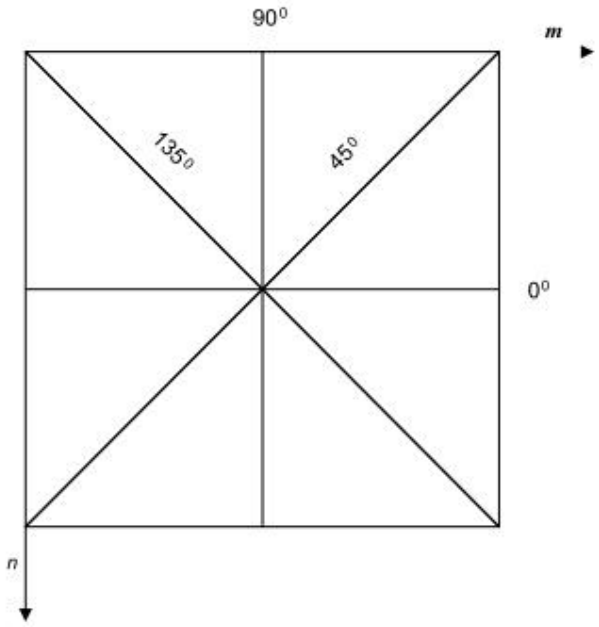


Figura 7: Las cuatro direcciones usadas para construir las matrices de co-ocurrencia (Theodoridis, 1999).

$$= \frac{número\ de\ pares\ de\ pixeles\ en\ la\ distancia\ de\ con\ valores\ (I_1\ ,\ I_2)}{número\ total\ de\ pares\ posibles}$$

En forma similar:

$$45°: P(I(m,n) = I_1, I(m \pm d, n \mp d) = I_{2)}$$

$$90°: P(I(m,n) = I_1, I(m, n \mp d) = I_{2)}$$

$$135°: P(I(m,n) = I_1, I(m \pm d, n \pm d) = I_{2)}$$

Para cada uno de estos histogramas se define un arreglo, conocido como matriz de co-ocurrencia, o matriz de dependencia espacial. Sea por ejemplo el arreglo de la imagen I (m, n)

$$I = \begin{bmatrix} 0 & 0 & 2 & 2 \\ 1 & 1 & 0 & 0 \\ 3 & 2 & 3 & 3 \\ 3 & 2 & 2 & 2 \end{bmatrix} \qquad (2.7)$$

Que corresponde a una imagen de 4 x 4 pixeles. También se asume que Ng = 4 $(I(m,n) \in \{0,1,2,3\})$. Una matriz de co-ocurrencia para un par (d, ∅) está definida como una matriz de Ng x Ng

$$A = \frac{1}{R}\begin{bmatrix} \eta(0,0) & \eta(0,1) & \eta(0,2) & \eta(0,3) \\ \eta(1,0) & \eta(1,1) & \eta(1,2) & \eta(1,3) \\ \eta(2,0) & \eta(2,1) & \eta(2,2) & \eta(2,3) \\ \eta(3,0) & \eta(3,1) & \eta(3,2) & \eta(3,3) \end{bmatrix}$$

Donde η (I1, I2) es el número de pares pixeles, en la posición relativa (d, ∅), que tiene valores de nivel de gris I1, I2, respectivamente. R es el número total de pares de pixeles. Así $\frac{1}{R}\eta(I_1,I_2) = P(I_1,I_2)$. Para la imagen de nuestro ejemplo y la posición relativa del pixel (1, 0°) tenemos:

$$A^0(d=1) = \frac{1}{24}\begin{bmatrix} 4 & 1 & 1 & 0 \\ 1 & 2 & 0 & 0 \\ 1 & 0 & 6 & 3 \\ 0 & 0 & 3 & 2 \end{bmatrix}$$

En pocas palabras, para cada par de intensidades, como el par (0, 0), contamos el número de pares de pixeles en la distancia relativa d=1 y orientación ∅=0° que toma esos valores. Para el ejemplo presentado este valor es 4. Dos de los resultados de la búsqueda in la dirección positiva y dos en la negativa. De acuerdo a la definición en 2.6, estos pares de pixeles tienes las coordenadas (m, n) y (m ±1, n) y los niveles de gris I1 = 0, I2 = 0. El número total de pares de pixeles para este caso es de 24. En efecto, para cada fila se tiene Nx - 1 pares y se tienes Ny filas. Así el número total de las direcciones (positivo y negativo) tienen 2(Nx - 1) Ny = 2(3x4)=24. Para una dirección diagonal de 45° y d =1 para cada línea se tiene 2(Nx - 1) pares, excepto para la primera, que no tiene ningún par. Así mismo, el número total es $2(Nx - 1)(Ny - 1) = 2(3x3) = 18$. Para d =1 y 90° se tiene 2(Ny - 1)Nx pares y finalmente para d = 1 y 135° 2(Nx-1)(Ny-1). Para la imagen presentada en el ejemplo y (d = 1, ∅= 45°) se obtiene:

$$A^{45}(d = 1) = \frac{1}{18}\begin{bmatrix} 0 & 1 & 2 & 1 \\ 1 & 0 & 1 & 1 \\ 2 & 1 & 0 & 3 \\ 1 & 1 & 3 & 0 \end{bmatrix}$$

A pesar del ejemplo anterior muestra matrices simétricas, es importante resaltar que este no es el caso general.

Tener definida las probabilidades de coocurrencia los niveles de gris con relación a la posición espacial relativa del pixel. En lo sucesivo, se establecen las características correspondientes a ser extraídas. Algunas de las características tienen una interpretación física directa en relación a la textura, por ejemplo, donde es posible cuantificar la rugosidad, suavidad,

etc. Por otro lado, otras características no tienen estas propiedades, más aun si codifican la información relacionada a la textura con una gran capacidad discriminatoria (Theodoridis, 1999) (Loew, 2000).

Luego de obtener la matriz de co-ocurrencia, las siguientes características pueden ser calculadas.

Segundo Momento Angular (SMA)

$$SMA = \sum_{i=0}^{N_g-1} \sum_{j=0}^{N_g-1} (P(i,j))^2 \qquad (2.8)$$

Esta característica es una medida de suavidad de la imagen. En efecto si todos los pixeles son del mismo nivel de gris I = k, entonces P(k, k) = 1 y P(i, j) = 0, $i \neq k \; o \; j \neq k$ y SMA = 1, En otro extremo, se tiene todos los pares posibles de niveles de gris con la misma probabilidad $\frac{1}{R}$, entonces $SMA = \frac{R}{R^2} = \frac{1}{R}$. Cuanto menos suave sea la región existirá una mayor distribución uniforme P(i, j) y menos SMA.

Contraste

$$CON = \sum_{n=0}^{N_g-1} n^2 \left\{ \sum_{i=0}^{N_g-1} \sum_{j=0}^{N_g-1} P(i,j) \right\}_{|i-j|=n} \qquad (2.9)$$

Esta es una medida de contraste de la imagen, mide las variaciones locales de los niveles de gris. En efecto, $\sum_i \sum_j P(i,j)$ es el porcentaje de pares de pixeles cuyas intensidades se diferencias por n. La dependencia n2

determina con el valor mayor las grandes diferencias. CON toma valores altos para imágenes de alto contraste.

Momento de diferencias Inversas (IDF)

$$IDF = \sum_{i=0}^{N_g-1} \sum_{j=0}^{n_g-1} \frac{P(i,j)}{1+(i-j)^2} \qquad (2.10)$$

Esta característica toma valores altos para imágenes de bajo contraste debido a la dependencia inversa $(i-j)^2$.

Entropía (H)

$$H_{xy} \sum_{i=0}^{N_g-1} \sum_{j=0}^{N_g-1} P(i,j)\, log_2 P(i,j) \qquad (2.11)$$

Forma

La representación y análisis morfológico cumple un papel central en diversas aplicaciones de visión computacional. De hecho, procedimientos de reconocimiento óptico de caracteres, emparejamiento de contornos para reconstrucción tridimensional de imágenes biomédicas, inspección visual y muchas otras tareas pueden ser realizadas por procesos basados en forma. De manera general, existen dos enfoques básicos para la representación y análisis morfológico (Sonka, 2007): un enfoque por contornos y uno por regiones. Ejemplos de representación de formas basadas en regiones son los descriptores simples de región escalar, momentos, convex hull, los esqueletos, ejes de simetría y la descomposición morfológica. Por otro lado la aproximación poligonal, el

código de cadena, las primitivas geométricas, las curvas paramétricas, la representación B-spline, los descriptores de Fourier y la trasformada de Hough son ejemplos de representaciones de formas basadas en contornos (Sonka, 2007).

Primitivas geométricas

- Diámetro efectivo (diámetro de la circunferencia que tiene la misma área)

$$d = 2\sqrt{\frac{A}{\pi}} \qquad (2.12)$$

Donde A es el área del objeto.

- Circularidad (C = 1 para un circulo)

$$C = \frac{4\pi A}{P^2} \qquad (2.13)$$

Donde *P* es el perímetro.

- Solidez (máximo para un circulo)

$$solidez = \frac{P^2}{A} \qquad (2.14)$$

- Proyección

A pesar de su mayor utilización en procesamiento de imágenes binarias, las proyecciones pueden servir como una base para definir los descriptores de regiones relacionadas. Estos descriptores pueden ser definidos en todas las direcciones, donde los más usados son el horizontal y el vertical.

$$proyeccion\ horizontal = p_h(i) = \sum_{j=1}^{N_g} I(i,j) \qquad (2.15)$$

$$proyeccion\ vertical = p_v(j) = \sum_{i=1}^{N_g} I(i,j) \qquad (2.16)$$

Tales proyecciones también pueden ser útiles para medir la homogeneidad en imágenes de tonos de gris y el largo y el alto de las regiones.

Momentos

Cuando dos imágenes son comparadas entre sí, o con una imagen estándar, un conjunto de momentos derivados por Hu (Hu, 1962) pueden ser útiles. Estos momentos tienen la característica de ser invariantes a la traslación, rotación y cambios de escala. Para la imagen I(x,y), se define

$$\mu_{pq} = \sum_{i=1}^{N_g} \sum_{j=1}^{N_g} (x_i - \overline{x})^p (y_j - \overline{y})^q I(x,y) \qquad (2.17)$$

Donde:

$$\overline{x} = \sum_{i=1}^{N_g} \sum_{j=1}^{N_g} x_i I(x_i, y_j) \qquad (2.18)$$

$$\overline{y} = \sum_{i=1}^{N_g} \sum_{j=1}^{N_g} y_j I(x_i, y_j) \qquad (2.19)$$

Y los momentos centrales normalizados son definidos como:

$$\eta_{pq} \frac{\mu_{pq}}{(\mu_{00})^{\gamma}} \qquad (2.20)$$

Donde:

$$\gamma = \frac{p+q}{2} + 1 \qquad (2.21)$$

Los momentos invariantes están entonces definidos como:

$$\begin{aligned}
\phi_1 &= \eta_{20} + \eta_{02} \\
\phi_2 &= (\eta_{20} - \eta_{02})^2 + 4\eta_{11}^2 \\
\phi_3 &= (\eta_{30} - 3\eta_{12})^2 + (\eta_{03} - 3\eta_{21})^2 \\
\phi_4 &= (\eta_{30} + \eta_{12})^2 + (\eta_{03} - \eta_{21})^2 \\
\phi_5 &= (\eta_{30} - 3\eta_{12})(\eta_{30} + \eta_{12})[(\eta_{30} + \eta_{12})^2 - 3(\eta_{21} + \eta_{03})^2] + \\
&\quad (\eta_{03} - 3\eta_{21})(\eta_{21} + \eta_{03})[(\eta_{21} + \eta_{03})^2 - 3(\eta_{30} + \eta_{12})^2] \\
\phi_6 &= (\eta_{20} - \eta_{02})[(\eta_{30} + \eta_{12})^2 - (\eta_{21} + \eta_{03})^2] + 4\eta_{11}(\eta_{30} + \eta_{12})(\eta_{21} + \eta_{03}) \\
\phi_7 &= (3\eta_{21} - \eta_{03})(\eta_{30} + \eta_{12})[(\eta_{30} + \eta_{12})^2 - 3(\eta_{21} + \eta_{03})^2] + \\
&\quad (\eta_{30} - 3\eta_{12})(\eta_{21} + \eta_{03})[(\eta_{21} + \eta_{03})^2 - 3(\eta_{30} + \eta_{12})^2]
\end{aligned}$$

Otros tipos de momentos son los de Zernike que están basados en funciones polinomiales complejas, formando un conjunto ortonormal completo sobre el círculo unitario $x^2+y^2 \leq 1$ y son definidos como:

$$V_{pq}(x,y) = V_{pq}(p,\theta) = R_{pq}(\rho)\exp(jq\theta) \qquad (2.22)$$

Donde:

p es un entero no negativo

q es un entero sujeto a la restricción *p*-|*q*| para, |*q*|≤*p*

$$\rho = \sqrt{x^2 + y^2}$$

$$\theta = tg^{-1}\frac{y}{x}$$

$$R_{pq}(\rho) = \sum_{s=0}^{(p-|q|)/2} \frac{(-1)^s[(p-s)!]\rho^{p-2s}}{s!\left(\frac{p+|q|}{2}-s\right)!\left(\frac{p-|q|}{2}-s\right)!}$$

Los momentos de Zernike de una función I(x, y) está dada por:

$$A_{pq} = \frac{p+1}{\pi}\iint_{x^2+y^2\leq 1} I(x,y)V^*(\rho,\theta)dxdy \qquad (2.23)$$

Donde * denota la conjugación compleja. Para una imagen digital, los respectivos momentos de Zernike son calculados como:

$$A_{pq} = \frac{p+1}{\pi}\sum I(x_i,y_i)\,V^*(\rho_i,\theta_i), x^2+y^2 \leq 1 \qquad (2.24)$$

Donde i toma todos los pixeles de la imagen. El cálculo de los momentos correspondientes de una imagen, considera el centro de la imagen como el origen y los pixeles son mapeados dentro del círculo unitario, esto es, $x^2 + y^2 \leq 1$. Los pixeles que no se encuentran dentro del círculo unitario no son considerados. La magnitud de los momentos de Zernike es invariante a la rotación.

2.2.5. Teoría de Wavelets

La transformada wavelet se presenta como una herramienta alternativa para el procesamiento de señales, cambiando el paradigma de representación de los mismos, al utilizar funciones base de wavelets (en

lugar de las senoidales de Fourier) para trasformar una señal dentro de la escala de tiempo.

En el caso de procesamiento de imágenes, uno de los puntos fuertes de la transformada wavelet viene del hecho que con una cantidad mínima de valores (en una escala menor de la imagen) es posible representar toda la imagen, Aquellos valores puedes ser aprovechados para ensamblar el vector de características para la representación de una imagen.

Las wavelets son conocidas por los matemáticos hace mucho tiempo, pero su formalización de esta teoría fue realizada recientemente en la década de los 80 (Grossmann & Morlet, 1984) (Mallat S. , 1989) (Daubechies I. , 1990).

La primera mención de lo que podría llamarse wavelet fue presentada por Alfred Haar en 1909 y que ahora se llama la base de Haar. El termino wavelets es atribuido a Norman Ricker (1940) en su trabajo de sismología, pero fue introducido por J Morlet, siendo su base matemática formalizada por el físico teórico A. Grossmann (Grossmann & Morlet, 1984). Los datos sísmicos estudiados por Morlet exhibían contenidos de frecuencia que cambian rápidamente a lo largo del tiempo, así mismo mostraran que cualquier tipo de señal puede ser analizado en términos de escala y traslaciones de una simple función wavelet madre. Yves Meyer (Meyer, 1993) y Stephane Mallat (Mallar, 1988)desarrollaron esta idea en una teoría denominada análisis multiresolucion.En 1989 Mallat (Mallat S. , 1989) mostro que el análisis multiresolución puede ser visto simplemente omo

una forma de algoritmos de pirámide usados en procesamiento de imágenes.

2.2.5.1. Transformada de Fourier

Una de las técnicas más populares en el procesamiento de señales es la transformada de Fourier, que tiene como objetivo transformar una señal (función) del dominio del espacio al dominio de la frecuencia (Gonzales, 2007). La transformación responsable de la transformación está dada por

$$F[u] = \int f[t]e^{-i2\pi ut}dt \qquad (2.25)$$

Esta ecuación corresponde a la transformada de Fourier de una señal continua f(t), la cual presenta algunas deficiencias, entre ellas el hecho que esta transformación no está localizada en el dominio del espacio, por lo que no puede representar adecuadamente los cambios que suceden en el espacio de la señal. Esto es debido al hecho que una transformada está basada en la integración de toda una función para el cálculo de cada frecuencia. Esto no sería un problema si la señal no cambiase en el tiempo (señales estacionarias) pero sucede que muchas señales interesantes contienen muchas características no estacionarias (flujos, tendencias, cambios repentinos, inicio y final de un evento). Esas características generalmente son la parte más importante de la señal.

Para solucionar ese problema Denis Gabor (1946) adapto una trasformada de Fourier para analizar una pequeña porción de la señal

en un tiempo, introduciendo la llamada trasformada por ventanas de Fourier (Windowed Fourier Transform -WFT). En este caso, una ventana de observación es localizada en el dominio del tiempo y es calculada la transformada de Fourier en la porción visible de la señal se procede igual para cada posición de la ventana. Considerando j(t) una función que cumple el papel de ventana, se puede definir una transformada por ventanas de Fourier de una señal continua f(t) como:

$$F(u,b) = \int j(t-b) f[t]\, e^{-i2\pi ut} dt \qquad (2.26)$$

El problema es que una vez que se define el tamaño para la ventana de tiempo, esta ventana permanece constante para todas las frecuencias. Puede suceder que algunas señales precisen de un enfoque mas flexible, donde el tamaño de la ventana pueda cambiar con el objetivo de detectar el contenido local de la frecuencia (Daubechies, 1992).

2.2.5.2. Transformada Wavelet

Después de la transformada por ventanas de Fourier. La utilización de las wavelets es el paso lógico siguiente. La que se puede interpretar como una técnica por ventana con regiones de dimensión variable, donde a diferencia de Fourier tiene como base una función de duración limitada, esto es de soporte compacto, que es una propiedad en la que su dominio es diferente de cero en una extensión finita e igual a cero en todo el resto. Esto hace interesante la utilización de wavelets en el

caso específico de análisis de imágenes. Pues los cambios de regiones o bordes pueden ser detectados mas fácilmente.

Una definición de una transformada wavelets considerando una señal continua está dada por:

$$F\ (a,b) = \int f(t)\psi_{a,b}\ (t)dt \qquad (2.27)$$

Los parámetros a y b varían continuamente en R, y las funciones $\psi_{a,b}$ son denominadas wavelets y definidas de la siguiente forma:

$$\psi_{a,b}(t) = \frac{1}{\sqrt{a}}\psi\left(\frac{t-b}{a}\right) \qquad (2.28)$$

Una transformada wavelet para señales discretas es definida como:

$$F_{m,n}(a,b) = a_0^{-m/2}\int f(t)\psi(a_0^{-m}t - nb_0) \qquad (2.29)$$

Se puede ver claramente que el comportamiento de esta función está basada en dilataciones y traslaciones a partir de una wavelet madre ψ. En ambos casos, esta wavelet madre, debe satisfacer la propiedad:

$$\int \psi(t)dt = 0 \qquad (2.30)$$

Observando la ecuación 2.27 se percibe que la transformada wavelet depende de dos parámetros a y b, que corresponde a las informaciones de escala y tiempo respectivamente (Daubechies, 1992).

Obtener los coeficientes de wavelet en todas las escalas posibles requiere de muchos cálculos, convirtiéndose en un trabajo tedioso (transformada continua de wavelets). Debido a este hecho es que la transformada discreta de wavelets elige un subconjunto de escalas y ubicaciones sobre los cuales se realiza los cálculos.

Según (Daubechies, 1992) dentro de la transformada discreta de wavelets se distinguen dos enfoques: sistemas redundantes discretos (frames) y bases ortonormales de wavelets. Un segundo enfoque considera la estrategia de análisis multiresolución, desarrollada por (Mallat S. , 1989).

2.2.5.2.1. Métodos Multiresolución

Las primeras construcciones de bases ortonormales de *wavelets* parecían un poco misteriosas. La situación cambio con la llegada del análisis multiresolución, formulada en 1986 por Mallat e Meyer. La multiresolución proporciona un punto de referencia donde las bases de *wavelets* son naturalmente comprendidas y permite la construcción de nuevas bases. Cuando Mallat trabajo con las wavelets de Meyer por primera vez, estaba trabajando con análisis de imágenes, donde la idea de estudiar imágenes en varias escalas simultáneamente era popular. Esto estímulo a ver bases ortonormales de wavelets como una herramienta para describir matemáticamente el “incremento de la información” necesario para pasar de una aproximación tosca a una aproximación con mayor resolución (Daubechies, 1992).

Una interpretación multiresolución permite obtener una interpretación invariante de la escala de la imagen. Una escala de una imagen cambia en razón de la distancia entre la escena y el centro óptico de la cámara. Cuando la escala de la imagen es modificada, la interpreación de la imagen no debería cambiar. Una representación multiresolución puede ser parcialmente invariante de escala si la secuencia de parámetros de la resolución $(V_j)_{j\in z}$ varia exponencialmente. (Mallat S. , 1989).

Para un mejor entendimiento se utiliza el concepto de “espacio vectorial” del algebra lineal. Un espacio vectorial V es básicamente una colección de “objetos” (nombrados vectores, en este contexto) para los cuales está definidos la adición y producto escalar. Así es posible sumar dos vectores, escalar un vector por alguna constante (Stollnitz, 1996).

Las funciones base para un espacio Vj son denominadas funciones escalares y son usualmente denotadas por el símbolo ϕ Una base simple para Vj está dada por el conjunto de funciones caja escaladas y trasladadas:

$$\phi_i^j(x) \coloneqq \phi\left(2^j x - i\right) i = 0, \dots, 2^j - 1$$

Donde

$$\phi(x) := \begin{cases} 1 & \text{se} \quad 0 \leq x < 1 \\ 0 & \text{se} \quad \text{caso contrário} \end{cases}$$

El paso siguiente en la construcción del análisis multiresolución consiste en escoger un producto interno definido sobre los espacios de vectores Vj . Para el ejemplo mostrado, el producto interno "estandar" esta dado por:

$$\langle f|g\rangle \coloneqq \int_0^1 f(x)g(x)dx \qquad (2.31)$$

Dos vectores u y v se dicen ortogonales si su producto interno es <u|v> = 0. Entonces, ahora es posible definir un nuevo espacio vectorial Wj como el complemento ortogonal de Vj en Vj+1. En otras palabras, Wj es un espacio de todas las funciones en Vj+1 que son ortogonales en todas las funciones en Vj en el producto interno escogido.

Una colección de funciones $\psi_i^j(x)$ que genera Wj, son nombradas wavelets. Estas funciones bases tienes dos propiedades importantes:

- Las funciones base $\psi_i^j(x)$ de W_j , juntan las funciones base ϕ_i^j de V_j, que forman una base para V_{j+1}.
- Cada funcion base $\psi_i^j(x)$ de Wj es ortogonal a cada función base ϕ_i^j de Vj del producto interno escogido.

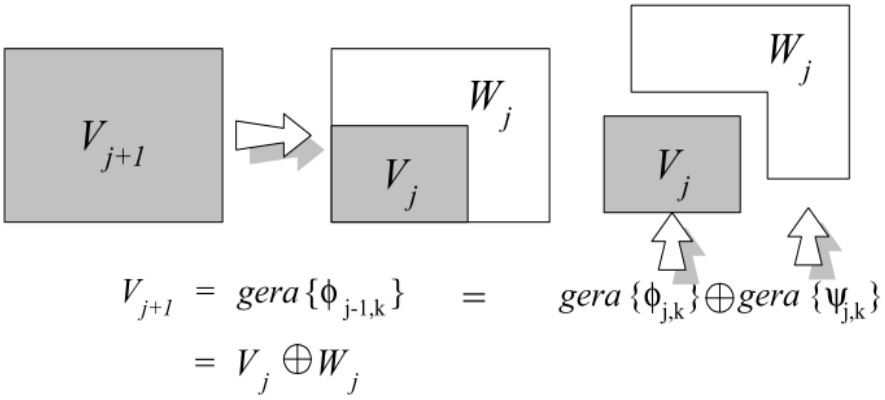


Figura 8: Generación del espacio superior en funcion de $\phi(x)$ y $\psi(x)$.

En resumen (Daubechies, 1992), un análisis multiresolución consiste de una secuencia de espacios de aproximaciones sucesivas Vj, Exactamente, de subespacios cerrados Vj que satisfacen

$$...V_{-2} \subset V_{-1} \subset V_0 \subset V_1 \subset V_2... \quad (2.32)$$

$$\overline{\bigcup_{j\in\mathbb{Z}} V_j} = L^2(\mathbb{R}) \quad (2.33)$$

$$\bigcap_{j\in\mathbb{Z}} V_j = \{0\} \quad (2.34)$$

Todos los espacios son una versión escalada del espacio central V0.

$$f(x) \in V_j \Leftrightarrow f(2^j x) \in V_0 \quad (2.35)$$

e

$$f(x) \in V_0 \Rightarrow f(x-n) \in V_0, para\ todo\ n \in \mathbb{Z} \quad (2.36)$$

Indica la invariancia de V0 por traslaciones enteras.

Sabiendo que ϕ ∈ V_0

$$\phi_{0,n}(x) = \phi(x-n), n \in \mathbb{Z}$$

es una base ortonormal para V0 (2.37)

La propiedad indicada en la ecuación 2.35 expresa que todos los espacios están relacionados por la escala en un mismo espacio V0 (este es un aspecto de la "multiresolución"). Sin embargo debido a esta propiedad, si $f(x) \in V_j$, entonces $f(x-2^j n) \in V_j$, para todo $n \in Z$, Las condiciones indicadas por las ecuaciones 2.37 y 2.35 implican que $\{\phi_{j,n}\}_{j,n\in Z}$ es una base ortonormal para V_j, para todo $j \in Z$. Si definimos Pj como un operador de proyección ortogonal sobre V_j, la condición 2.34 asegura que $lim_{j\to\infty} P_j f = f$ para todo$f \in L^2(\mathbf{R})$.

Cada V_j puede ser interpretado como un espacio de aproximación sucesiva: La aproximación de $f \in L^2(\mathbf{R})$ en resolución 2^j que está definida como la proyección de f sobre Vj y cuanto j sea mayor más fina es la resolución obtenida. La condición 2.35 quiere decir que ninguna escala es privilegiada. Los detalles adicionales necesarios para aumentar la resolución de 2^j a 2^{j+1} están dadas por la proyección de f sobre el complemento ortogonal de Vj en relación a Vj+1 que denotamos por W_j: $V_j \oplus W_j = V_{j+1}$ (Daubechies, 1992).

2.2.6. Transformada Wavelet de Gabor

Según Manjunath (Manjunath, 1996), la transformada wavelet de Gabor (Gabor Wavelets Transform – GWT) originalmente propuesta como funciones de Gabor por (Gabor, 1946), que consiguió resultados prometedores cuando se aplicó en aplicaciones de reconocimiento de textura y objetos (Tucerayan, 1993). Las *wavelets* de Gabor son especialmente apropiadas para la representación de características locales por el hecho de presentar las siguientes propiedades (He, Dong, & Sheng, 2001): (a) Las *wavelets* de Gabor son las mejores *wavelets* localizadas en tiempo y frecuencia, (b) Las *wavelets* de Gabor contienen un mayor número de parámetros, y (c) las investigaciones en psicología muestran que las reacciones de las células simples en la corteza visual pueden ser modeladas mediante las funciones de Gabor (Daugman, Complete Discrete 2-D Gabor Transforms by Neural Networks for Image Analysis and Compession, 1988).

La utilización de las wavelets de Gabor en extracción de características está motivada por varios factores. La representación de Gabor ha sido demostrada como óptima en el sentido de minimizar la incertidumbre de las articulaciones bi-dimensionales en el espacio y frecuencia (Daugman, Complete Discrete 2-D Gabor Transforms by Neural Networks for Image Analysis and Compession, 1988). Estos filtros pueden ser considerados localizables en orientación y en escala. De esta forma, detectores de líneas y bordes, y cálculos estadísticos sobre estas micro – características en una determinada región son usualmente efectuados para caracterizar la

información de textura subyacente (Manjunath, 1996) (Jain & farrokhnia, 1991).

Las funciones de Gabor han sido utilizadas en el procesamiento de imágenes en diversas formas. Como la representación total de la imagen mediante la definición de un conjunto de wavelets de Gabor 2D, que proporcionan una representación completa de una imagen cualquiera (Lee, 1996). En (Manjunath, 1996) y (Jain A. &., 1998) la desviación estándar de los coeficientes de GWT fueron usados con éxito en la recuperación de imágenes basadas en textura. Utilizando esta técnica, se puede efectuar el reconocimiento de iris con gran éxito, como es demostrado en el trabajo de Daugman (Daugman, How Iris Recognition Works, 2002), así como en firmas digitales (Jain, Ross, & Prabhakar, 2001), En el área de reconocimiento de rostros, (Feris & Krueger, 2001) desarrollan una técnica que analiza los subespacios de wavelets generados por la GWN (Gabor Wavelets Network), para el reconocimiento de rostros en secuencias de video.

2.2.6.1. Funciones de Gabor y Wavelet

Una función bidimensional de Gabor $\psi(x,y)$, usado como una *wavelet* madre, es definida como (Manjunath, 1996):

$$\psi(x,y) = \left(\frac{1}{2\pi\sigma_x\sigma_y}\right) exp\left[-\frac{1}{2}\left(\frac{x^2}{\sigma_x^2}+\frac{y^2}{\sigma_y^2}\right)+2\pi jWx\right] \quad (2.38)$$

Donde σ_x y σ_y son las desviaciones estándar de $\psi(x,y)$ a lo largo de los ejes x e y , respectivamente. La constante W determina la longitud

de banda de la frecuencia de los filtros. Las partes real e imaginaria de $\psi(x, y)$ son mostrados en la figura 9 (Rubner, 2000)

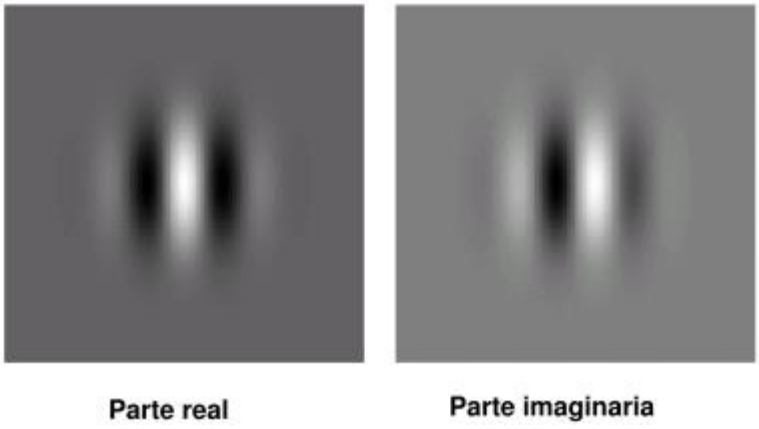


Figura 9: Componentes real e imaginario de $\psi(x, y)$.

Las Funciones de Gabor forman un completo, pero no ortogonal, conjunto de bases. Ejemplo de una clase particular de wavelets de Gabor 2D son presentadas en la figura 10. Usando estas bases, se expande la señal que proporciona una descripción localizada. Esto hace que las funciones Gabor sean conocidas por la óptima localización de tiempo-frecuencia (Mallat S. G., 1999).

Una clase de funciones auto-similares, nombradas como Gabor wavelets son generadas a partir de la función $\psi(x, y)$, la wavelet de Gabor madre. Entonces un diccionario de filtros auto-similares pueden ser obtenidos mediante dilataciones y rotaciones apropiadas de $\psi(x, y)$ através de la función generatriz:

$$\psi_{mn}(x, y) = a^{-m}\psi(x', y'), a > 1, m = 1 : M.n = 1 : N. \quad (2.39)$$

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = a^{-m} \begin{bmatrix} cos\theta_n & sen\theta_n \\ -sen\theta_n & cos\theta_n \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix}, \theta_n = n\pi/N. \quad (2.40)$$

Donde a^{-m} es un parámetro de escala que tiene como objetivo asegurar que la energía sea independiente de m.

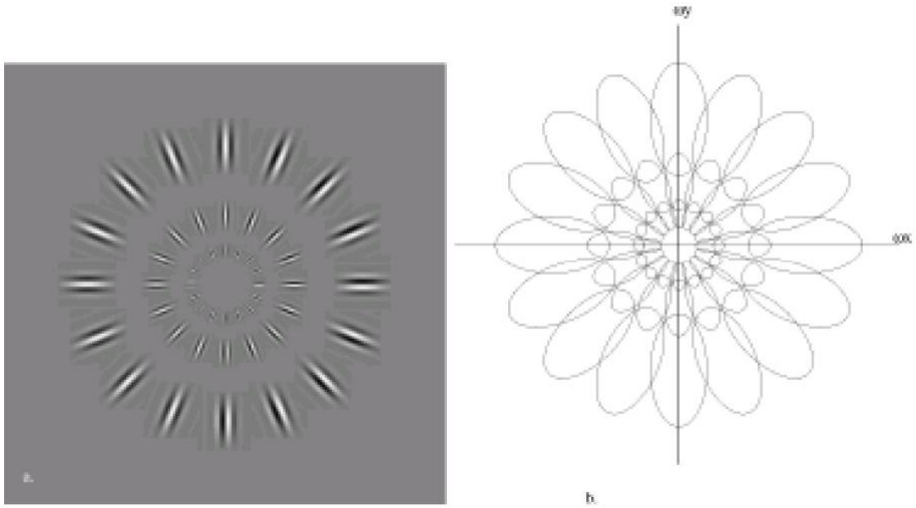


Figura 10: (a) Conjunto de wavelets Gabor (longitud de banda octal 1.5) (b) cobertura en el plano de frecuencia espacial (Lee, 1996).

Las wavelets de Gabor pueden ser interpretadas como un conjunto de funciones Gabor con distintos centro de frecuencia y orientación la figura 10. El tamaño o la longitud de banda de las wavelets de Gabor es también controlado por θ. Debido que las wavelets de Gabor se extienden para el tiempo, frecuencia y orientación.

La no ortogonalidad de la wavelets de Gabor implica que existe información redundante en las imágenes filtradas, Esto se puede ver en la figura 10 donde la elipses correspondientes a cada filtro se intersectan entre si. Manjunath (Manjunath, 1996) define una estrategia para reducir eta redundancia, que sigue los siguientes principios (Rubner, 2000):

- *Separación uniforme en la orientación*. Asumiendo la simetría rotacional, todos los filtros en una escala especifica deberían

tener la misma desviación estándar angular (σ_v) y deberían estar espaciados en el eje de orientación.

- *Separación exponencial en escala. Las longitudes de los filtros deberián incrementar exponencialmente con distancia a partir del centro del plano (u, v). Esto quiere decir que la diferencia entre una escala y otra está dada por la multiplicación por un parámetro escalar.*

- *Cobertura continua de espacio de frecuencia. El contorno de los filtros vecinos en contacto entre si en los ejes de escala y orientación.*

Esta estrategia para reducir redundancia está definida por $(\mathrm{U_l}\ \mathrm{e}\ \mathrm{U_h})$ que denotan los centros de frecuencia de interés superior e inferior. Sea K el número de orientaciones y S el número de escalas la descomposición multiresolución, entonces la estrategia consiste en asegurar que el soporte de magnitud de pico medio (half-peak) de la respuesta de los filtros en el espectro de frecuencia se estén en contacto unos con otros, como es mostrado en la figura 11. El proceso anteriormente descrito es producto de las siguientes fórmulas para el cálculo de los parámetros de los filtros σ_u y σ_v (y asi σ_x y σ_y):

$$a = (U_h/U_l)^{\frac{1}{S-1}}, \qquad (2.41)$$

$$\sigma_u = \frac{(a-1)U_h}{(a+1)\sqrt{2ln2}}, \qquad (2.42)$$

$$\sigma_v = tg\left(\frac{\pi}{2K}\right)\left[U_h - 2ln\left(\frac{2\sigma_u^2}{U_h}\right)\right]\left[2ln2 - \frac{(2ln2)^2\sigma_u^2}{U_h^2}\right]^{-\frac{1}{2}}, \quad (2.43)$$

Donde $W = U_h$ y $m = 0, 1, \ldots, S-1$.

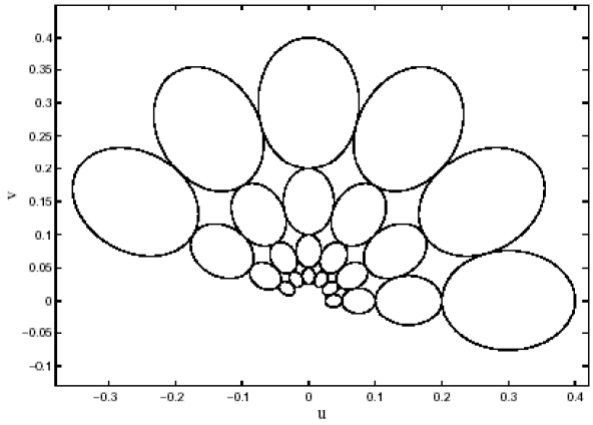


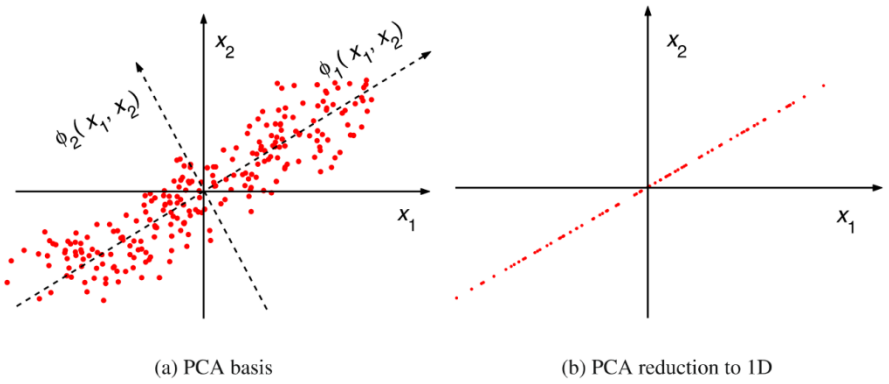
Figura 11: Filtros de respuesta ortogonal, luego de aplicar la técnica de eliminación de redundancia (Manjunath, 1996).

2.2.7. Análisis de componentes Principales (PCA)

El Análisis de Componentes Principales (Principal Component Analisys - PCA) es una técnica de reducción de dimensionalidad basado en la extracción de un número deseado de *componentes principales* de los datos multideimensinales. Los primeros componentes principales son la combinación lineal de la dimensión original que tiene la máxima varianza; el enésimo componente principal es la combinación lineal con la varianza más alta, sujeto a ser ortogonales a los primeros n-1 componentes principales.

La idea de PCA está ilustrada en la figura 12 a; los ejes con las etiquetas ϕ_1 corresponde a la dirección de la máxima varianza y es escogido como el primer componente principal. En un caso de dos dimensiones, el segundo componente principal esta entonces determinado únicamente por las restricciones de ortogonalidad; en un espacio de alta dimensionalidad

el proceso de selección continuaría, guiado por las varianzas de las proyecciones.



(a) PCA basis (b) PCA reduction to 1D

Figura 12: El concepto de PCA/KLT, a *Lineas Solidas*, las bases originales; líneas punteadas, las bases KLT. Los puntos son seleccionados regularmente en ubicaciones espaciadas sobre una línea recta rotada en 30° y entonces perturbada por un ruido Gauseano isotrópico en 2D. b La proyección (ID reconstrucción) de los datos usando solo el primer componente principal.

El PCA está cercanamente relacionado a la transformada Karhunen.Loeve (KLT) (Loeve, 1955), Que fue derivado en el contexto de procesamiento de señales como la trasformada ortogonal con las bases $\phi = [\phi_1, \dots, \phi_N]^T$ que para cualquier $k \leq N$ minimiza el error de reconstrucción promedio L_2 para los puntos de datos x.

$$\varepsilon(\boldsymbol{x}) = \left\| \boldsymbol{x} - \sum_{i=1}^{k} (\boldsymbol{\phi}_i^{\mathrm{T}} \boldsymbol{x}) \phi_i \right\|. \qquad (2.44)$$

Uno puede mostrar (Gerbrands, 1981) que, bajo la presunción que los datos tienen media cero, la formulación de PCA y KLT son idénticas. Sin pérdida de generalidad, a continuación asume que los datos son en efecto de media cero; esto es, el rostro promedio $\bar{x}$ esta siempre derivada de los datos.

Los vectores base en KLT pueden ser calculadas de la siguiente manera. Sea X la matriz de datos de N x M con columnas $x_1, \dots, x_M$, que son

observaciones de una señal que pertenece a $\mathbb{R}^N$; en el contexto de reconocimiento de rostros, M es el número de imágenes de rostros disponible, y N = m n es el número de pixeles en una imagen. Las bases KLT ϕ es obtenida de resolver el problema de la obtención de los valores propios (eigenvalue) $\Lambda = \phi^T \Sigma \phi$, donde Σ es la matriz de covarianza de los datos

$$\boldsymbol{\Sigma} = \frac{1}{M} \sum_{i=1}^{M} \boldsymbol{x}_i \boldsymbol{x}_i^{\mathrm{T}} \qquad (2.45)$$

$\phi = [\phi_1, \dots, \phi_m]^T$ es el eigenvector de Σ, y Λ es la matriz diagonal con eigenvalues $\lambda_1 \geq \cdots \geq \lambda_N$ de Σ sobre su diagonal principal, así ϕ_j es el eigenvector correspondiente al j-taesimo eigenvalue más grande. Entonces puede estar mostrando que los eigenvalues λ_i es la variancia de los datos proyectados sobre ϕ_i.

Así, para realizar PCA y extraer los k componentes principales de los datos, se debe proyectar los datos sobre ϕ_k, las primeras k columnas de las bases KLT ϕ, la cual corresponde a los k eigenvalues más altos de Σ Esto puede ser visto como una proyección lineal $\mathbb{R}^N \rightarrow \mathbb{R}^k$, la cual retiene la máxima energía (por ejemplo variancia) de la señal. Otra propiedad importante de PCA es que decorrelaciona los datos: la matriz de covarianza de $\phi_K^T X$ es siempre diagonal.

Las principales propiedades de PCA son resumidos a continuación

$$\boldsymbol{x} \approx \boldsymbol{\Phi}_k \boldsymbol{y}, \qquad \boldsymbol{\Phi}_k^{\mathrm{T}} \boldsymbol{\Phi}_k = \boldsymbol{I}, \qquad E\{y_i y_j\}_{i \neq j} = 0 \qquad (2.46)$$

A saber, reconstrucción aproximada, ortonormalidad de las bases ϕ_k, y componentes principales decorrelacionados $y_i = \phi_i^T x$, respectivamente, Estas propiedades son ilustradas en la figura 12, donde PCA encuentra de forma satisfactoria los colectores principales, y en la figura 13, donde es menos satisfactorio, debido a la clara no linealidad de los principales colectores.

El PCA puede ser implementado via la descomposición de valor singular (SVD). LA SVD de una matriz X de dimensión M x N (M >= N) está dada por

$$\boldsymbol{X} = \boldsymbol{U}\boldsymbol{D}\boldsymbol{V}^{\mathrm{T}} \qquad (2.47)$$

Donde la matriz $U\ de\ M \times N$ y la matriz $V\ de\ N \times N$ tiene columnas ortonormales y la matriz $\mathrm{D}\ de\ N \times N$ tiene los valores singulares de X sobre su diagonal principal y cero en cualquier otro caso.

Se puede demostar que $U = \phi$, asi SVD permite un cálculo eficiente y robustode PCA sin la necesidad de estimar la matriz de covarianza Σ. Cuando el número de muestras M es más pequeño que la dimensión N, esto es una ventaja crucial.

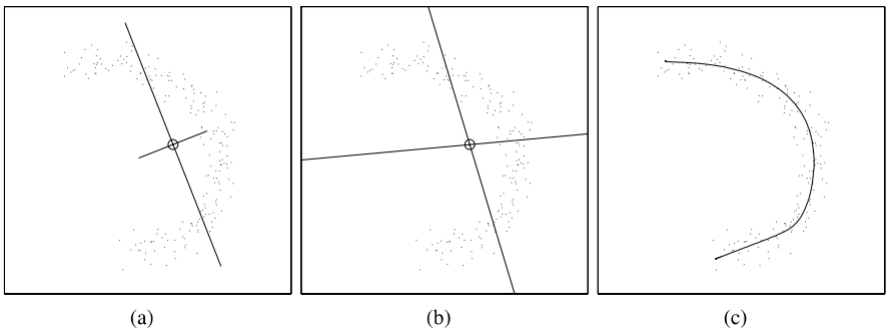


Figura 13: (a) bases del PCA (lineal, ordenado, ortogonal). (b) bases ICA y (c) Curva principal.

2.2.8. Máquina de Soporte Vectorial (Support Vector Machine - SVM)

Según (Team, 2013), support vector machine (SVM) es un clasificador discriminativo formalmente definido por un hiperplano que separa. En otras palabras, dados datos de entrenamiento etiquetados (aprendizaje supervisado), la respuesta del algoritmo es un hiperplano óptimo capaz de clasificar nuevas muestras.

Dado un conjunto de puntos en dos dimensiones (Figura 14) que pertenecen a una de dos clases, donde el problema consiste en encontrar una línea recta que las separe.

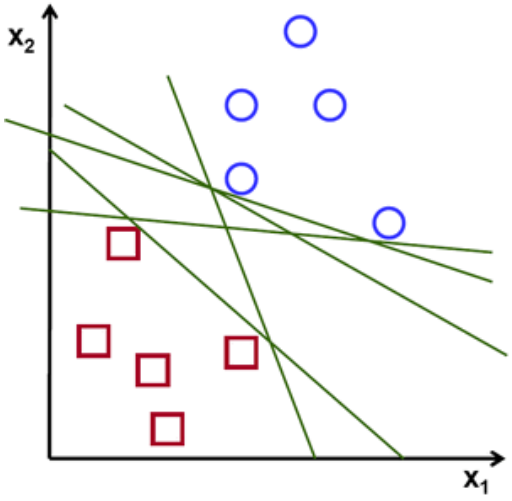


Figura 14: Puntos en el plano, representando dos clases.

En la figura 14 se puede ver que existen múltiples líneas que ofrecen una solución al problema. De forma intuitiva se podría definir un criterio para estimar el valor de las líneas:

Una línea no es útil si pasa muy cerca de los puntos porque será sensible al ruido y no generalizara de la forma adecuada. Así mismo el objetivo sería encontrar la línea que pase lo más lejos posible de todos los puntos.

Entonces la operación del algoritmo SVM se basa en encontrar el hiperplano que da la distancia mínima de los datos de entrenamiento. Dos veces esta distancia recibe el nombre de margen en la teoría de SVM. Asimismo el hiperplano óptimo separador maximiza el margen de los datos de entrenamiento.

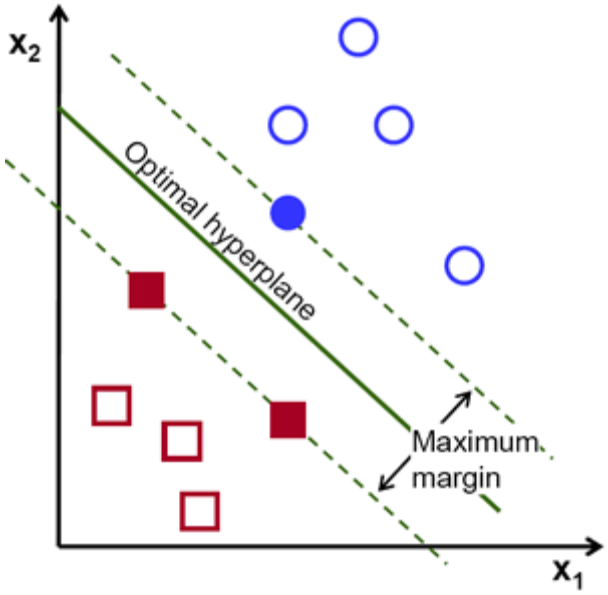


Figura 15: Hiperplano óptimo.

2.2.8.1. Calculo del hiperplano

Definido el hiperplano:

$$f(x) = \beta_0 + \beta^T x, \qquad (2.48)$$

Donde β es conocido como el vector de pesos y β_0 como el bias.

El hiperplano óptimo puede ser representado en un número infinito de diferentes maneras, escalando β y β_0 . Como una importante convención, entre todas las posibles representaciones del hiperplano, el elegido es:

$$|\beta_0 + \beta^T x| = 1 \quad (2.49)$$

Donde x simboliza los datos de entrenamiento más cercanos al hiperplano. En general, las muestras de entrenamiento que están cercanas al hiperplano son llamadas supaport vector. Está representación es conocida como el hiperpalno canónico.

Ahora, necesitamos usar el resultado de geometría que da la distancia entre un punto x y un hiperplano (β , β_0):

$$\text{distance} = \frac{|\beta_0 + \beta^T x|}{\|\beta\|}. \quad (2.50)$$

En particular para el hiperplano canónico, el numerador es igual a uno y la distancia al support vector es

$$\text{distance}_{\text{support vectors}} = \frac{|\beta_0 + \beta^T x|}{\|\beta\|} = \frac{1}{\|\beta\|}. \quad (2.51)$$

Recuerda que el margen mencionado previamente, aquí denotado como M, es dos veces la distancia a las muestras cercanas:

$$M = \frac{2}{\|\beta\|} \quad (2.52)$$

Finalmente, el problema de maximizar M es equivalente a el problema de minimizar la función L(β) sujeto a algunas restricciones. El modelo de restricciones el requerimiento para el hiperplano para clasificar correctamente todos las muestras de entrenamiento x_i. Formalmente,

$$\min_{\beta,\beta_0} L(\beta) = \frac{1}{2}\|\beta\|^2 \text{ subject to } y_i(\beta^T x_i + \beta_0) \geq 1 \ \forall i, \quad (2.53)$$

Donde y_i representa cada una de las etiquetas de las muestras de entrenamiento.

Este es un problema de optimización de Lagrangian que puede ser resuelta usando multiplicadores de Lagrange para obtener el vector de pesos β y la bias β_0 del hiperplano óptimo.

2.3. Definición de términos básicos

2.3.1. Imagen digital

Una imagen digital es una imagen f(x,y) que se ha discretizado tanto en las coordenadas espaciales como en el brillo. Una imagen digital puede considerarse como un matriz cuyos índices de fila y columna identifican un punto de la imagen y el valor del correspondiente elemento de la matriz indica el nivel de brillo en ese punto (Gonzales, 2007).

2.3.2. Pixel

Los elementos de una distribución digital (imagen digital) de este tipo se denominan elementos de la imagen, o más comúnmente pixeles o pels abreviaturas de su denominación inglesa picture elements (Gonzales, 2007).

2.3.3. Reconocimiento de patrones

El reconocimiento de patrones es la disciplina científica cuyo objetivo es la clasificación de objetos en un número determinado de categorías o clases.

Dependiendo de la aplicación estos objetos pueden ser imágenes o señales de onda o cualquier tipo de medida que necesite ser clasificada (Theodoridis, 1999).

2.3.4. Detección de Rostros

La detección de rostros (face detection) es el primer paso en el reconocimiento automático de rostros. Su confiabilidad tiene mayor influencia en el rendimiento y usabilidad de todo el sistema de reconocimiento de rostros. Dada una imagen o video. Un detector de rostros ideal deberá estar en la capacidad de identificar y localizar todos los rostros presentes independientemente de su posición, escala, orientación, edad y expresión (Li, 2011).

2.3.5. Reconocimiento de Rostros

El reconocimiento de rostros es una tarea que los seres humanos realizan de forma rutinaria y sin esfuerzo en su diario vivir (Li, 2011), que consiste en asignar una identidad a determinado individuo por medio de su rostro.

2.3.6. Reconocimiento Automático de Rostros

Es el proceso para determinar la identificación de una persona que significa clasificar a la persona como conocida o no conocida, conlleva a la realización de tres procedimientos: enrolamiento (enrollment), comparación (matching - emparejamiento) y clasificación (Mou, 2010).

2.3.7. Señal

Una señal es modelada como una función de valores reales de una variable real e independiente t, usualmente el tiempo. Específicamente, considere un proceso físico que es dependiente del tiempo. Supóngase que en cada tiempo t dentro de un intervalo $a \leq t \leq b$ se realiza la medida de algún aspecto de este proceso, y su medida de rendimiento un número que asume cualquier valor en el rango dado. En este caso nuestra medida representa de forma natural por una función real x(t) con dominio $a \leq t \leq b$. Nos referiremos a x(t) como una señal análoga. La función x(t) podría representar la intensidad del sonido en una ubicación dada (señal de audio), la intensidad de luz sobre una superficie u objeto (señal de luz - imagen), la velocidad de un objeto, entre otros (Broughton & Bryan, 2009).

2.3.8. Características

Las características elementales están explícitamente presentes en los datos adquiridos y pueden ser pasados directamente a la etapa de clasificación. Las características de alto orden son derivadas de las elementales y son generadas por manipulaciones o transformaciones en los datos.

2.3.9. Extracción de características

Es el proceso de generar características que puedan ser usadas en el proceso de clasificación de los datos. En ocasiones viene precedido por un pre proceso de la señal, necesario para corregir posibles deficiencias en los datos debido a errores del sensor, o bien para preparar los datos de

cara a posteriores procesos en las etapas de extracción de características o clasificación.

2.3.10. Longitud de onda

En ondas armónicas, se define la longitud de onda como la separación especial existen entre dos puntos cuyo estado de movimiento es idéntico.

2.4. Operacionalización de Variables

VARIABLES	INDICADOR	MEDICION
Detección Automático de Rostros	Adquisición de imágenes y secuencias de video	• Buena • Mala
	Pre-procesamiento de la imagen	
	Extracción de características	
	Clasificación	• Efectiva • No efectiva
Reconocimiento Automático de Rostros	Adquisición de imágenes y secuencias de video	• Buena • Mala
	Pre-procesamiento de la imagen	
	Extracción de características	
	Clasificación	• Efectiva • No efectiva

CAPÍTULO III

MATERIALES Y METODOS

3.1. Población

Se consideró como población la base de datos de rostros constituido por la unión de los bancos de imágenes de rostros FERET (2413 imágenes faciales, representando a 856 personas), ORL (400 imágenes faciales, representando a 40 personas 10 por persona) y la colección realizada por los investigadores (50 imágenes de rostros representando a 10 personas) representando a un total de 906 personas.

3.2. Muestra

El tamaño de la muestra se realizó utilizando el muestreo aleatorio simple para la proporción con la siguiente formula:

$$n = \frac{Z_{\alpha/2}^2 P * Q}{e^2}$$

Dónde:

n = Tamaño de muestra.

$Z_{(\alpha/2)}$ = Nivel de confianza elegido (5%).

P = Casos a favor (se considera un 90%)

Q = Casos en contra (se considera un 10%)

e = Error deseado (6%)

De operar se obtuvo el tamaño de muestra de 96 personas, de las que se elegirán las imágenes para evaluar el sistema de reconocimiento de rostros.

3.3. Métodos de recopilación de datos

Los datos de análisis son proporcionados por las imágenes digitales del rostro de las personas que fueron obtenidas mediante la captura de imágenes en ambientes controlados estas imágenes son seleccionadas de forma aleatoria para ser evaluadas por el sistema construido para evaluar la efectividad de las técnicas propuestas en la extracción de características.

3.4. Métodos de tratamiento de datos

3.4.1. Método general de reconocimiento de rostros

Para la elaboración del sistema y posterior tratamiento de los datos obtenidos se tomó en consideración los procesos de un sistema general de reconocimiento de rostros, que contempla cuatro etapas la adquisición de una imagen o video, la detección de rostros, localización y extracción de características y la clasificación tal como se aprecia en la figura 16.

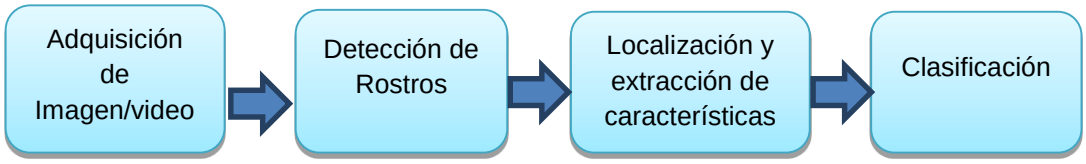


Figura 16: Diagrama de bloques de un esquema general para el reconocimiento de rostros.

3.4.2. Método de reconocimiento de rostros propuesto

El método propuesto para el reconocimiento de rostros contempla dos etapas:

- Entrenamiento y
- Reconocimiento.

Entrenamiento, el entrenamiento prepara al sistema con datos etiquetados que nos permiten obtener los vectores de características que son almacenados para poder realizar posteriormente la recuperación de estos. En esta etapa se siguen los procesos de: *adquisición de la imagen desde una base de datos, detección de rostro en la imagen, representación de la imagen mediante la wavelet de Gabor y extracción de características mediante el análisis de componentes principales PCA*.

Reconocimiento, el reconocimiento consiste en la operación de adquirir una imagen nueva no incluida en la etapa de entrenamiento y posteriormente asignar esta imagen a una clase almacenada en la base de datos, los procesos seguidos en esta etapa son: *adquisición de una imagen de prueba, derección de rostro en la imagen*, *representación de la imagen mediante la wavelet de Gabor, extracción de características mediante el análisis de componentes principales PCA y clasificación de la imagen mediante máquina de soporte vectorial SVM.*

La figura 17 resume los dos procesos propuestos para la tarea de reconocimiento de rostros

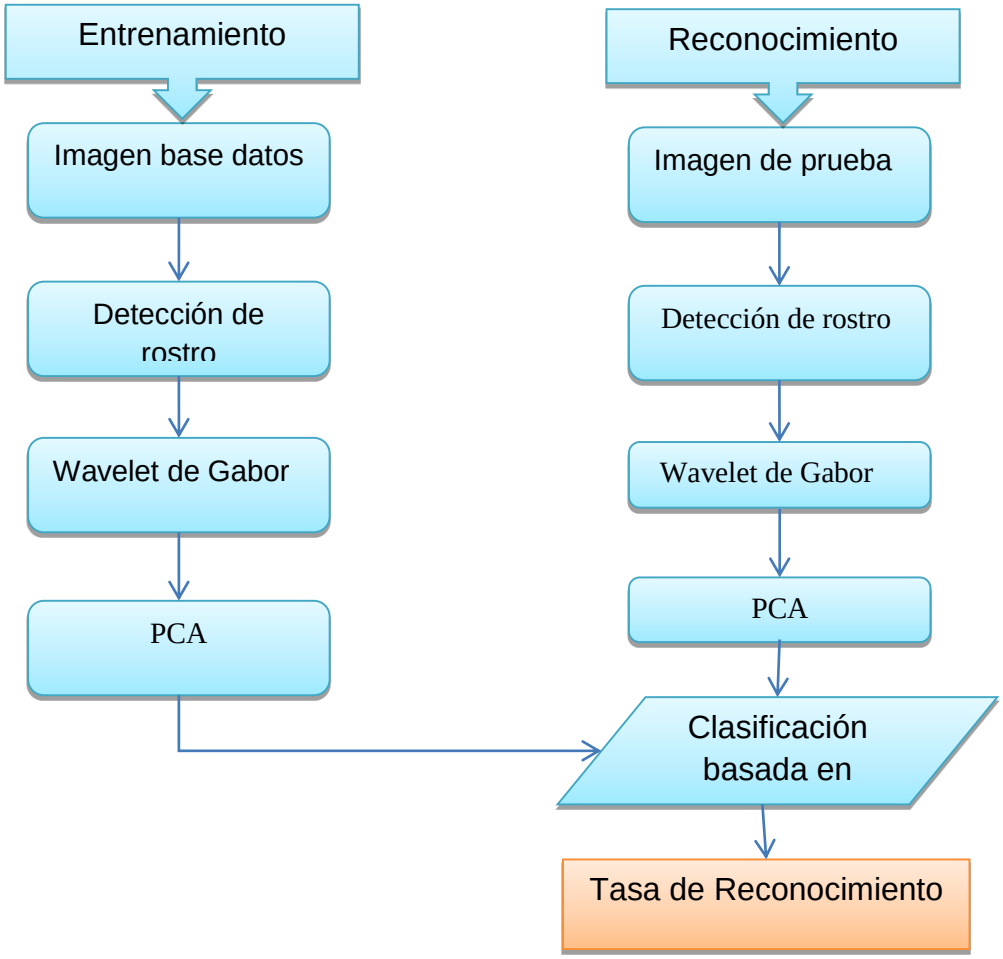


Figura 17: Diagrama de procesos de reconocimiento de rostros propuesto.

3.4.3. Método para la detección de rostros

Para lograr el propósito de detectar rostros se utiliza el clasificador en cascada que usa las características basadas en Haar este es un método efectivo para la detección de objetos propuesto por Paul Viola y Michael Jones en el paper, "Rapid Object Detection using a Boosted Cascade of Simple Features" en el año 2001. Este es un método basado en aprendizaje de la maquina donde una función en cascada es entrenada con imágenes positivas e imágenes negativas, es usada para detectar objetos en diferentes tipos de objetos.

Figura 18: Detección de rostro utilizando clasificador en cascada.

3.4.4. Método para la extracción de características.

3.4.4.1. Wavelet de Gabor

Se utiliza el filtro de gabor que está definido como:

$$\Psi_{\mu,v}(\boldsymbol{z}) = \frac{||\mathbf{k}_{\mu,v}||^2}{\sigma^2} \mathrm{e}^{(-||\mathbf{k}_{\mu,v}||^2 ||\mathbf{z}||^2 / 2\sigma^2)} \left[\mathrm{e}^{\mathrm{i}\mathbf{k}_{\mu,v}\mathbf{z}} - \mathrm{e}^{-\frac{\sigma^2}{2}}\right] \qquad (3.1)$$

Donde u y v definen las escalas y orientaciones de los filtros de gabor, z = (x,y) y Ku,v está definida como:

$$k_{u,v} = k_v e^{i\varphi u} \qquad (3.2)$$

$k_v = kmax/f^v$ y $\varphi u = \pi u/8$, kmax es la frecuencia máxima, y f es el factor de espacio entre los kernels en el dominio de la frecuencia. En el presente $\sigma = 2\pi$, $kmax = \frac{\pi}{2}$, $f = \sqrt{2}$, $u \in \{0,1,\dots,7\}$ y $v \in \{1,2,3,4\}$.

La representación de la imagen de rostro mediante la wavelet de gabor es obtenida mediante la convolución entre la imagen y la familia de filtros de gabor descritas en la ecuación 3.3

$$F_{u,v}(Z) = I(Z) * \Psi_{u,v}(Z) \qquad (3.3)$$

Donde z = (x,y), * denota el operador de convolución y Fu,v es la respuesta del filtro de gabor de la imagen con orientación u y escala v.

En el presente se convoluciona la imagen con 40 kernels de Gabor para generar las caracteristicas de Gabor. La imagen de entrada es una imagen de rostro que es geométricamente normalizado y de 64 x 64 pixeles. Así el tamaño del vector es (64 x 64 x 40 x 2).

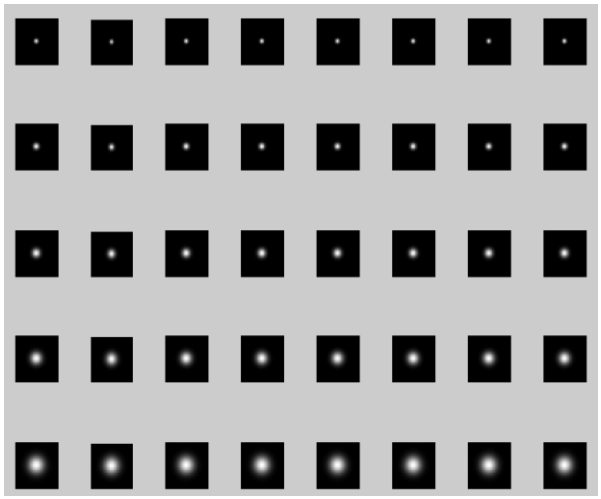


Figura 19: Filtros de Gabor con diferentes orientaciones y escala.

3.4.4.2. Convolución de la imagen

La convolución es realizada entre todas las partes de la imagen y un operador (Kernel), el kernel para el presente trabajo son el

banco de filtros de Gabor, se siguió el siguiente procedimiento para efectuar esta operación:

- Ubicar el ancla del kernel (centro de la imagen) en lo alto de un determinado pixel, con el resto del kernel sobrepuesto en los pixeles locales de la imagen.

- Multiplicar los coeficientes del kernel por el valor del pixel correspondiente y sumar el resultado.

- Localizar el resultado en la ubicación del ancla en la imagen de entrada.

- Repetir el proceso para todos los pixeles desplazando el kernel sobre la totalidad de la imagen.

La ecuación (3.4) expresa el procedimiento descrito anteriormente:

$$H(x,y) = \sum_{i=0}^{M_i-1} \sum_{j=0}^{M_j-1} I(x+i-a_i, y+j-a_j)K(i,j) \qquad (3.4)$$

3.4.4.3. Análisis de Componentes de Principales

En esta fase, cargamos la base de datos. En general, aplicamos muchas transformaciones antes de cargarlas. En efecto, la señal contiene información útil para el reconocimiento y solo los parámetros relevantes son extraídos. El modelo es una representación compacta de la señal que facilita la fase de reconocimiento, pero también reduce la cantidad de datos a ser almacenados

$$\begin{pmatrix} a_{1,1} & \cdots & a_{1,m} \\ \vdots & \ddots & \vdots \\ a_{n,1} & \cdots & a_{n,m} \end{pmatrix} \rightarrow \begin{pmatrix} a_{1,1} \\ \vdots \\ a_{n,1} \\ \vdots \\ a_{1,m} \\ \vdots \\ a_{n,m} \end{pmatrix} \quad (3.5)$$

Entonces, el conjunto de entrenamiento de imágenes de rostros será $\Gamma_1, \Gamma_2, \Gamma_3, \ldots, \Gamma_M$. El rostro promedio del conjunto es definido por la ecuación (3.6):

$$\Psi = \frac{1}{M}\sum_{i=1}^{M} \Gamma_i \quad (3.6)$$

Cada rostro se diferencia del promedio por el vector de la ecuación (3.7):

$$\Phi_i = \Gamma_i - \Psi, \mathrm{i} = 1, \ldots, \mathrm{M} \quad (3.7)$$

A continuación calculamos la matriz de covarianza C, según la ecuación (3.8):

$$C = \sum_{i=1}^{N} \Phi_i {\Phi_i}^T = AA^T \quad (3.8)$$

La matriz C es N2 por N2, y determinar los N2 vectores y valores propios se vuelve una tarea intratable para el tamaño de una imagen típica. Necesitamos un método factible para encontrar los eigen vectores ecuación (3.9).

$$\begin{pmatrix} \mathrm{e_i} = & \mathrm{Av_i} \\ \lambda_\mathrm{i} = & \mu_\mathrm{i} \end{pmatrix} \quad (3.9)$$

De M eigen vectores (eigen faces) ei, solo escogemos M1 que tiene el valor propio (eigen value) más alto. Cuanto más alto sea el valor propio, describe la característica más representativa del rostro de un particular eigenvector. Los eigen face con bajos eigen values pueden ser omitidos, porque ellos solo explican una pequeña parte de los rasgos característicos del rostro. Luego M1 eigen faces ei son determinados, la fase de entrenamiento del algoritmo es finalizado.

El proceso de clasificación de un nuevo rostro Γ_{new} a una de las clases procede en dos pasos.

Primero, la nueva imagen es transformada en sus componentes eigen face, Los pesos resultantes forman el vector de pesos Ω_T ecuación 3.10

$$W_k = e_k^T (\Gamma_{new} - \Psi),\ k = 1 \ldots\ldots\ldots\ldots M',\ \Omega_T = [w_1, w_2, \ldots, w_{M'}] \qquad (3.10)$$

Los pesos forman un vector $\Omega_T = [w_1, w_2, \ldots, w_M]$ que describen la contribución de cada eigenface de la representación de la imagen de rostro de entrada, tratando los eigen faces como un conjunto de bases para las imágenes de rostro. El vector puede entonces ser usado en un algoritmo de reconocimiento con patrón estándar para encontrar a que número de clases predefinidas pertenece. El método utilizado para determinar que clase de rostro provee la mejor descripción de una imagen de rostro de entrada es la máquina de soporte vectorial SVM.

3.4.5. Eficiencia del sistema

Para determinar la eficiencia del sistema se estima la tasa de falsa aceptación (rate of false acceptance - FAR) ecuación 3.11 la tasa de falso rechazo (rate of false rejection - FRR) ecuación 3.12 ya la tasa de igual error (equal error rate -ERR) ecuación 3.13:

$$FAR = \frac{número\ de\ falsas\ aceptaciones}{numero\ de\ impostores} \qquad (3.11)$$

$$FRR = \frac{número\ de\ falsos\ rechazos}{numero\ de\ rostros\ evaluados} \qquad (3.12)$$

$$EER = \frac{número\ de\ falsas\ aceptaciones + número\ falsos\ rechazos}{número\ total\ de\ entradas} \qquad (3.13)$$

3.4.6. Evaluación del sistema

Para determinar si la inclusión de las técnicas hibridas en la detección y reconocimiento de rostros permite que el sistema sea eficiente en imágenes y secuencias de video se aplicara una Prueba z para la proporción a los datos que serán resultados de la evaluación del prototipo.

1) H_0: $\pi = p$

   Ha: $\pi <> p$

2) Calcular

$$Z = \frac{p - \pi_H}{\sigma_p}$$

3) Compara Z calulada con Zt..

4) Si Z mayor o menor que Zt Rechazar H_0;

Si Z estas entre < -Zt, Zt > Aceptar H_0.

3.4.7. Metodología para el desarrollo del Software

El presente proyecto se desarrolló basado en la metodología Extreme Programming XP (Programación Extrema), que es el más destacado de los procesos ágiles de desarrollo de software que a diferencia de las metodologías tradicionales pone más énfasis en la adaptabilidad que en la previsibilidad, incluye al cliente en el proceso de desarrollo de software. Sus características fundamentales son:

- Desarrollo iterativo e incremental
- Pruebas unitarias continuas
- Programación en parejas
- Integración del equipo de programación con el cliente
- Corrección de todos los errores
- Refactorización del código
- Propiedad del código compartida
- Simplicidad del código

3.5. Material experimental

Durante el desarrollo y prueba de la presente investigación se utilizaron:

Datos

- imágenes y videos conteniendo rostros en escala de grises y a colores (RGB), en formatos jpg, pgm y avi.

Software

- Metodologías para el desarrollo de software (XP)
- Lenguaje de Programación C++
- Librería OpenCV
- Sistema operativo Ubuntu 12.04

Hardware

- Cámara Fotográfica
- Webcam
- WorkStation móvil (estación de trabajo movil).

CAPITULO IV

RESULTADOS

A continuación se realiza la exposición de los resultados, según el planteamiento de los objetivos específicos donde se describe el desarrollo de los módulos construidos y la contrastación de la hipótesis propuesta para el presente estudio.

4.1. Adquisición de Imágenes y Secuencias de Video.

4.1.1. Requisitos del Módulo

Los requisitos de software se registraron mediante las historias de los usuarios y las historias de los desarrolladores. Las historias 01, 02, 03 y 04 resumen los requisitos necesarios para este Módulo.

4.1.1.1. Requisitos de Usuario

Numero: 01	**Historias de Usuario**
Nombre Historia	Iniciar la Aplicación.
Quiero: Iniciar la aplicación de forma local sin usar usuario y contraseña.	
Para: Poder acceder a las funciones de la aplicación de procesamiento, detección y reconocimiento de rostro.	

Numero: 02	**Historias de Usuario**
Nombre Historia	Captura de Imagen.
Quiero: Imagen desde el programa.	
Para: almacenar imágenes y poderlas analizar.	

Numero: 03	**Historias de Usuario**
Nombre Historia	Captura de Video.
Quiero: Capturar segmentos de video desde el programa.	
Para: Almacenar segmentos de videos y poderlas analizar..	

Numero: 04	**Historias de Usuario**
Nombre Historia	Selección de imagen/video desde archivo
Quiero: seleccionar imagen/video desde archivo	
Para: poderlas analizar desde el programa.	

4.1.1.2. Tareas de Usuario

	Tarea de Ingeniería
Número Tarea: 01	**Número Historia: 01**
Nombre Tarea: Ejecutar la aplicación	
Tipo de Tarea: Desarrollo	**Puntos Estimados:** 3
Programador: Programador 1	**Iteración:** 3
Descripción: El acceso a la aplicación se ejecuta mediante el administrador de aplicaciones del Sistema Operativo (El usuario lanzara la aplicación con doble clic, enter o mediante el Shell del sistema).	

	Tarea de Ingeniería
Número Tarea: 02	**Número Historia: 02, 03**
Nombre Tarea: Crear menús e iconos de acceso	
Tipo de Tarea: Desarrollo	**Puntos Estimados:** 3
Programador: Programador 1	**Iteración:** 3
Descripción: El usuario podar acceder a las funciones de captura de pantalla a través del menú principal de la aplicación y mediante la barra de herramientas de la aplicación.	

	Tarea de Ingeniería
Número Tarea: 03	**Número Historia: 02, 03 y 04**
Nombre Tarea: Cargar archivos (imagen/video) desde archivo al programa.	
Tipo de Tarea: Desarrollo	**Puntos Estimados:** 3
Programador: Programador 1	**Iteración:** 3
Descripción: - El usuario inicia la carga del archivo desde el menú principal o la barra de herramientas de aplicación la imagen se carga en una ventana auxiliar donde se proyecta la imagen y/o video. - Se crean funciones de Carga de archivo en memoria para operar con ella.	

4.1.2. Análisis del Módulo

4.1.2.1. Diagrama de Casos de Uso

En la Figura 20 se observa el diagrama de casos de uso que resumen las funciones del módulo, estas funciones están orientadas a la adquisición de recursos en el presente trabajo imágenes, video o streamming para posteriormente ser analizadas y procesadas, los recursos son extraídos de imágenes (en formatos *.jpg, *.png, *.ppm), secuencias de video de diversa duración y la adquisición realizada en tiempo real mediante una cámara de video.

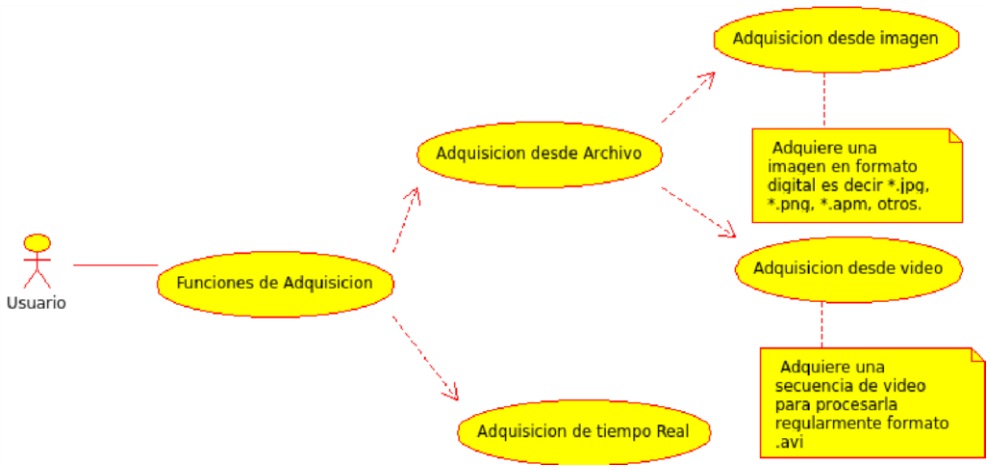


Figura 20: caso de uso función de adquisición de recursos para análisis.

4.1.3. Diseño del Módulo

4.1.3.1. Diagrama de Secuencias

El Diagrama de Secuencias mostrado en Figura 21 muestra las secuencias del Módulo realizado frente a la petición de apertura de una imagen, video o apertura de streaming.

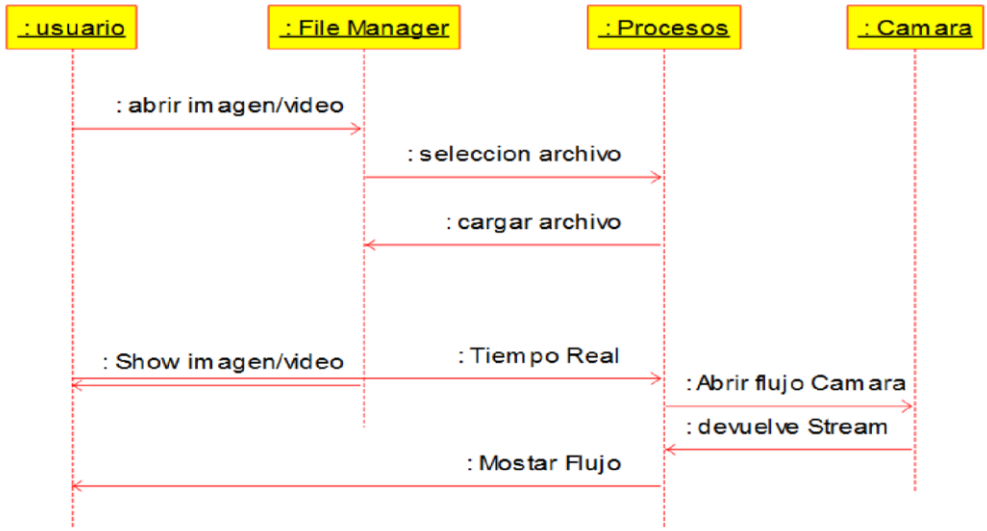


Figura 21: Diagrama de secuencia módulo adquisición de Imágenes y Secuencias de video.

4.1.3.2. Arquitectura del Módulo

La arquitectura diseñada para el desarrollo del Módulo resume la interacción con el usuario, el componente administrador de archivos y el uso de la cámara web. Se resalta que para el uso de las imágenes adquiridas por la Webcam es necesaria la interacción con los drivers del Sistema Operativo.

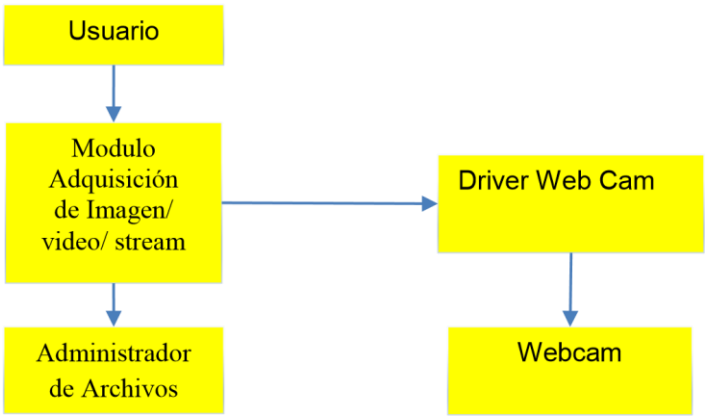


Figura 22: Arquitectura del Módulo adquisición de Imágenes y Secuencias de video.

4.1.4. Implementación del Módulo

La implementación del presente modulo se resume en la Tabla 01. Donde se muestra fragmento del código fuente desarrollado para la apertura de archivos y la adquisición de flujo de video desde la Webcam.

Tabla 1: Extracto código fuente Módulo adquisición de Imágenes y Secuencias de video.

```
void MainWindow::on_actionDesde_Archivo_triggered()
{
    fileName = QFileDialog::getOpenFileName(this, tr("Abrir
imagen"), ".", tr("Archivos de imagen (*.jpg *.png *.jpeg *.bmp
*.ppm *.ljpeg)"));
    image = cv::imread(fileName.toAscii().data());
    cv::namedWindow("Imagen de Entrada");
    cv::imshow("Imagen de Entrada", image);
    winImagen = true;
}
void MainWindow::on_actionTiempo_Real_triggered()
{
    tRealVideo.open(0);

    if(!tRealVideo.isOpened()){
        msgBox.setText("No se pudo abrir el video, Intentelo
nuevamente");
        msgBox.exec();
    }
    else
    {
        winTReal = true;
        ui->actionTomar_Imagen->setEnabled(true);
        cv::namedWindow("Tiempo Real",1);
        for(;;)
        {
            tRealVideo>> frame;
            imgTReal = frame;
            cv::imshow("Tiempo Real", imgTReal);
            if(cv::waitKey(30)>=0)
            {
                ui->actionTomar_Imagen->setEnabled(false);
                cv::destroyWindow("Tiempo Real");
                break;
            }
        }
        winTReal = false;
    }
}
```

Para la adquisición de imágenes y secuencias de video se utilizaron cuadros de dialogo del Sistema Operativo, y para la adquisición de secuencias de video en tiempo real se utilizaron funciones de apertura de video desde la cámara web. La figura 23 muestra una imagen capturada con el sistema realizado en tiempo real.

Figura 23: Adquisición de imagen en tiempo real realizada con el sistema.

4.2. Detección de Rostros en Imágenes y Secuencias de Video

4.2.1. Requisitos del Módulo

Las historias de usuario 05 y 06 describen los requisitos contemplados para la tarea de detección de Rostros en Imágenes y Secuencias de Video. Los detalles Técnicos para la implementación del módulo se resumen en las tareas de Ingeniería 04, 05 y 06. La Tarea 05 es la tarea principal contempla las especificaciones del algoritmo Boosted Cascade of Simple Features.

4.2.1.1. Requisitos de Usuario

Numero: 05	Historias de Usuario
Nombre Historia	Detección de Rostros en imágenes/videos
Quiero: Que el programa seleccione los rostros que existen en una imagen.	
Para: Poder trabajar solamente con los rostros de las personas que existan dentro de las imágenes omitiendo información no relevante.	

Numero: 06	Historias de Usuario
Nombre Historia	Diferenciar rostros de otros objetos en las imágenes
Quiero: Que el programa seleccione solamente las imágenes de rostros.	
Para: Almacenar las imágenes de rostros en la base datos para posteriormente analizarlas.	

4.2.1.2. Tareas de Usuario

	Tarea de Ingeniería
Número Tarea: 04	Numero Historia: 05 y 06
Nombre Tarea: Ejecutar la detección de rostr os en imágenes/video.	
Tarea Asociada: Tarea 05, 03	
Tipo de Tarea: Desarrollo	Puntos Estimados: 3
Programador: Programador 1	Iteración: 3
Descripción: Lanza la función de apertura de archivo de imagen/video para ejecutar sobre esta la detección de rostros.	

	Tarea de Ingeniería
Nùmero Tarea: 05	Numero Historia: 05, 06
Nombre Tarea: Detección de Rostros.	
Tipo de Tarea: Desarrollo	Puntos Estimados: 3
Programador: Programador 1	Iteración: 3

Descripción:
El desarrollo de la detección de rostros usa “Boosted Cascade of Simple Features”, modelo machine learning. Donde el modelo en cascada es entrenada con imágenes donde hay rostros y donde no las hay. Sigue el siguiente procedimiento.

- Dada una imagen de muestra $(x_1, y_1), \ldots, (x_n, y_n)$ donde yi = 0,1 para muestras negativas y positivas respectivamente.
- Inicializar ponderaciones $w_{1,i} = \frac{1}{2m}, \frac{1}{2l}$ para yi = 0,1 respectivamente, donde m y l son el número de negativos y positivos.
- Para *t* = 1,…, T:
- Normalizar las ponderaciones,

$$w_{t,i} \leftarrow \frac{w_{t,i}}{\sum_{j=1}^{n} w_{t,j}}$$

Asi que w_t es una distribución de probabilidad.

- Para cada característica, j, entrenar un clasificador h_j que eta restringida a usar una característica simple. El erro es evaluado con respecto a w_t

$$\epsilon_j = \sum_i w_i \left| h_j(x_i) - y_i \right|.$$

- Escoger el casificador, h_t, con el error menor . ϵ_j
- Actualizar las ponderaciones: $w_{t+1,i} = w_{t,i}\beta_t^{1-e_i}$, donde ei = 0 si la muestra, es clasificada correctamente, ei = 1 de otra forma, y $\beta_t = \frac{\epsilon_t}{1-\epsilon_t}$.
- El clasificador fuerte final es:

$$h(x) = \begin{cases} 1 & \sum_{t=1}^{T} \alpha_t h_t(x) \geq \frac{1}{2} \sum_{t=1}^{T} \alpha_t \\ 0 & \text{otherwise} \end{cases}$$

Donde:

$$\alpha_t = \log \frac{1}{\beta_t}$$

Tarea de Ingeniería	
Numero Tarea: 06	Numero Historia: 05 y 06
Nombre Tarea: Almacenar los rostros detectados.	
Tarea Asociada: Tarea 05.	
Tipo de Tarea: Desarrollo	Puntos Estimados: 3
Programador: Programador 1	Iteración: 3
Descripción: Almacenar los rostros detectados en el directorio imágenes, las imágenes son ahora solo los rostros recortados de la imagen original..	

4.2.2. Análisis del Módulo

4.2.2.1. Diagrama de Casos de Uso

La detección de rostros se realiza con el módulo de adquisición de imágenes, efectuando un pre procesamiento (conversión a escala de gris, ecualización mediante histogramas) y finalmente aplicar el clasificador en cascada. En esta etapa la clasificación consiste en determinar la existencia de un rostro en una determinada escena. En la Figura 24 se aprecia el diagrama de Casos de uso para la realización de este módulo.

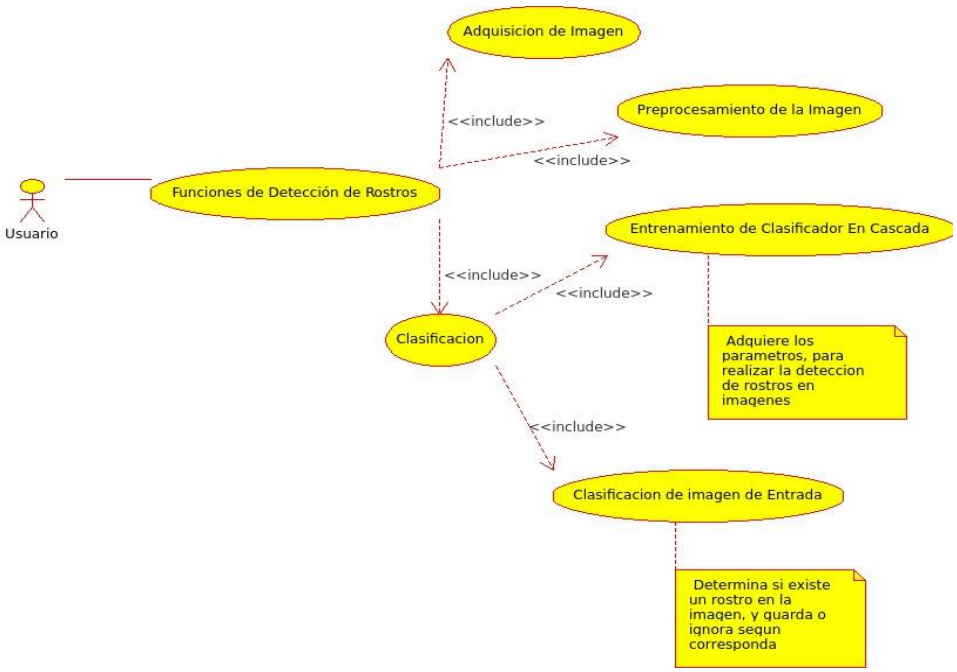


Figura 24: Diagrama de casos de uso para la módulo de detección de Rostros.

4.2.3. Diseño del Módulo

4.2.3.1. Diagrama de Secuencias

En el diagrama de secuencias de la figura 25 se aprecia la realimentación del pre procesamiento de las imágenes de entrada.

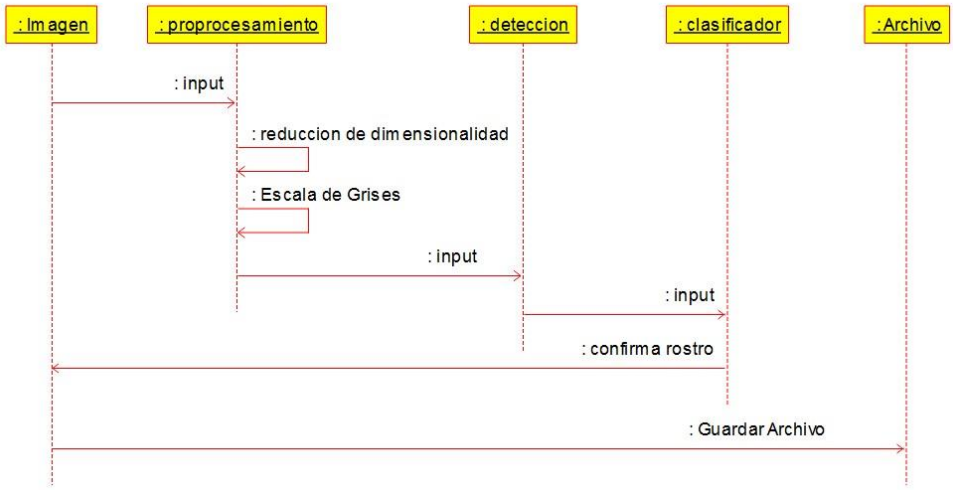


Figura 25 : Diagrama de secuencia para el módulo de detección de rostros.

4.2.3.2. Arquitectura del Módulo

En la figura 26 se muestra el modelo de la Arquitectura del módulo.

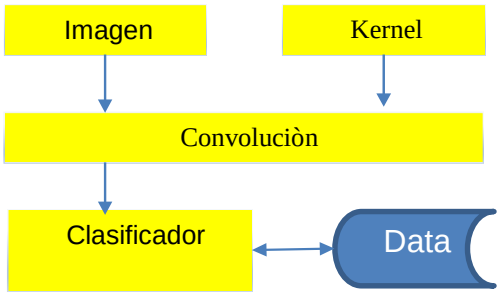


Figura 26: Arquitectura del módulo de detección de rostros.

4.2.4. Implementación del Módulo

La implementación del presente modulo se resume en el Tabla 02, donde se muestra fragmento del código fuente desarrollado.

Tabla 02: Extracto código fuente Módulo para la detección de rostros.

```
try{faceDetector.load("haar_xml/haarcascade_frontalface_default.xml");     }
        catch(cv::Exception e)  {            }
        if ( faceDetector.empty() ) {
          en cascada, intentelo nuevamente"); msgBox.exec(); }
             fileName = QFileDialog::getOpenFileName(this, tr("Abrir imagen"), ".",
             tr("Archivos de imagen (*.jpg *.png *.jpeg *.bmp *.ppm *.pgm)"));
        image = cv::imread(fileName.toAscii().data());
        //--------------- preprocesado escala gris y equalizacion
        cv::cvtColor(image,gray ,CV_BGR2GRAY);
        cv::equalizeHist(gray, imgTReal);
        //----------------detecta rostros en la imagen
         std::vector<cv::Rect> faces;
         faceDetector.detectMultiScale(imgTReal,faces,searchScaleFactor,minNeighbors,
         flags, minFeatureSize);
        if(faces.size()>0){
            //--------escala los recuadros con las coordenadas de rostros halladas
            for(int i = 0; i<(int)faces.size();i++){
                //--------------dibuja un rectangulo en el rostro
                cv::rectangle(image,faces[i],CV_RGB(0,255,0), 1);
                //--------------- guarda los rostros detectados
                QString nombreImagen = QString("img_test/rostro_%1.bmp").arg(i);
                faceDetected= image(faces[i]);
                cv::imwrite(nombreImagen.toAscii().data(), faceDetected);
                //---------------- guarda en escala de grises guarda cada segundo
                nombreImagen = "img_test/" + QString("g_rostro_%1.bmp").arg(i);
                cv::cvtColor(faceDetected, faceDetected, CV_RGB2GRAY);
                cv::imwrite(nombreImagen.toAscii().data(), faceDetected);
            }}
        cv::imshow("Original", image);  //show la imagen reducida a color
```

La captura de la imagen a ser analizada es realizada en el espacio de colores RGB (red, green y blue), si la imagen tiene dimensiones muy altas los procesos para la detección de rostros son lentos es por ello que las imágenes son reducidas a un espacio de 240 x 320 pixeles y la profundidad de la imagen es reducida a un solo canal es decir escala de Grises. La figura 27 muestra la reducción de dimensión y color de la imagen original capturada por el software.

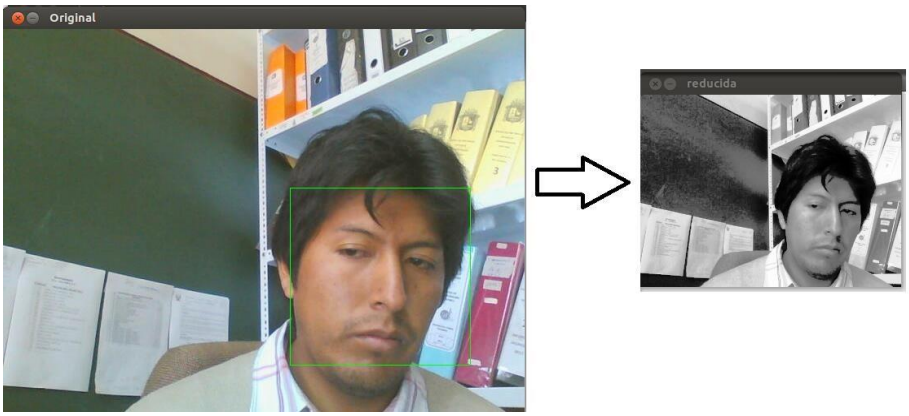


Figura 27: Reducción de dimensiones de la imagen en tamaño y color.

A continuación se realiza la clasificación de la imagen mediante el clasificador en cascada basado en las características de Haar iniciando un kernel de 20 x 20 pixeles, la figura 28 muestra las características de Haar y la figura 29 el diagrama de flujo del procedimiento.

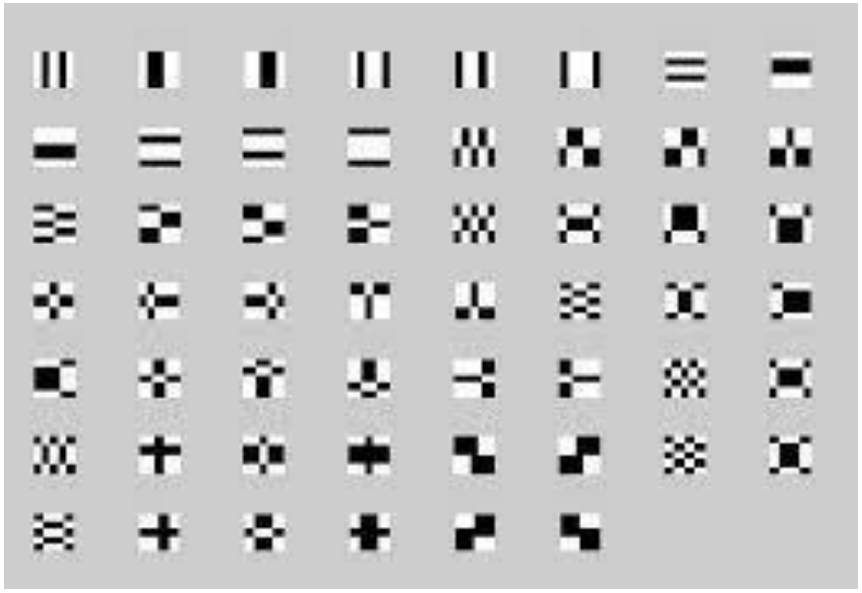


Figura 28: Características Haar para un clasificador en cascada.

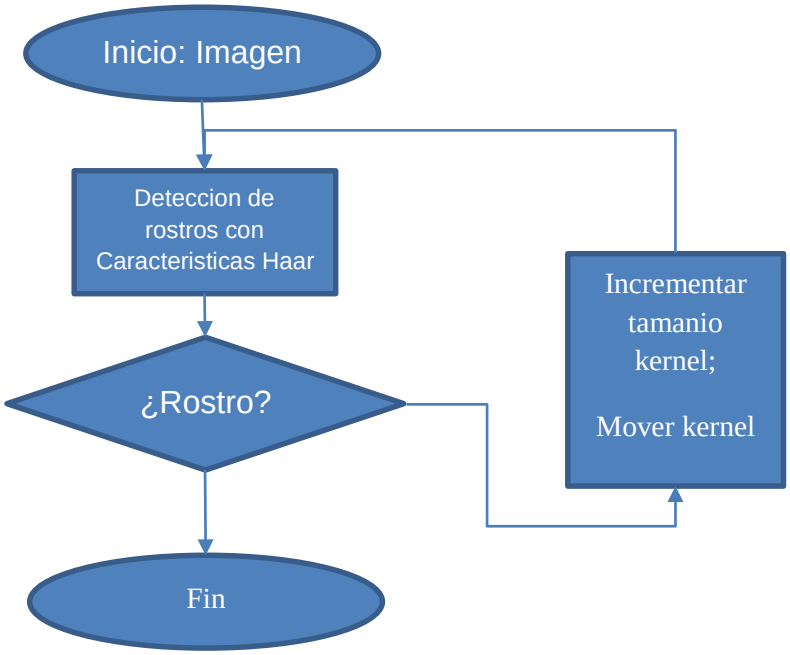


Figura 29: Diagrama de flujo proceso de detección de rostro.

4.3. Extracción de Características Basadas en la Wavelet de Gabor y Análisis de Componentes Principales.

4.3.1. Requisitos del Módulo

Para el módulo de extracción de características se reunieron los requisitos contemplados en las historias de usuario 07 extracción de características basadas en la Wavelet de Gabor y 08 extracción de características usando el PCA, los requisitos 07 y 08 contemplan la realización de las Tareas de Ingeniería 07, 08, 09, 10, 11, 12 y 13.

4.3.1.1. Requisitos de Usuario

Numero: 07	Historias de Usuario
Nombre Historia	Extracción de características usando Wavelets de Gabor
Quiero: el programa extraiga características utilizando wavelets de Gabor.	
Para: construir una colección de imágenes filtradas, que constituyen las características más relevantes de las imágenes de rostros..	

Numero: 08	Historias de Usuario
Nombre Historia	Extracción de características usando Análisis de componentes Principales
Quiero: el programa extraiga características con PCA..	
Para: Extraer características utilizando el banco de datos generado por la Wavelet de Gabor.	

4.3.1.2. Tareas de Usuario

	Tarea de Ingeniería
Número Tarea: 07	Numero Historia: 07
Nombre Tarea: Ejecutar extracción de características con Wavelet de Gabor.	
Tarea Asociada: Tarea 03.	
Tipo de Tarea: Desarrollo	Puntos Estimados: 3
Programador: Programador 1	Iteración: 3
Descripción: Inicia la ejecución del módulo encargado de la extracción de características basadas en la Wavelet de Gabor, esta tarea proporciona a la función la imagen a tratar.	

	Tarea de Ingeniería
Número Tarea: 08	Numero Historia: 07
Nombre Tarea: Construir kernels de Gabor variando las escalas y Orientaciones.	
Tarea Asociada: Tarea 07.	
Tipo de Tarea: Desarrollo	Puntos Estimados: 3
Programador: Programador 1	Iteración: 3
Descripción: Desarrollar Kernels de tamaño 10 por 10 con: - Ecuacion basada en : $\Psi_{\mu,v}(z) = \frac{\|k_{\mu,v}\|^2}{\sigma^2} e^{(-\|k_{\mu,v}\|^2 \|z\|^2 / 2\sigma^2)} \left[e^{ik_{\mu,v}z} - e^{-\frac{\sigma^2}{2}} \right]$ - $z = (x, y)$ - $k_{\mu,v} = k_v e^{i\varphi u}$ - - - $k_v = kmax/f^v$ - $\varphi u = \pi\mu/8\sigma = 2\pi$, $kmax = \pi/2$ y $f = \sqrt{2}$ - Orientaciones u ∈ { 0, 1, 2, 3, 4, 5, 6, 7} - Escala v ∈ { 1, 2, 3, 4}	

	Tarea de Ingeniería
Número Tarea: 09	Numero Historia: 07
Nombre Tarea: Convolucionar los kernels con la imagen de entrada.	
Tarea Asociada: Tarea 08.	
Tipo de Tarea: Desarrollo	Puntos Estimados: 3
Programador: Programador 1	Iteración: 3
Descripción: Obtener la representación Wavelet de Gabor del rostro haciendo convolución entre la imagen y una familia de filtros de gabor, como se define en: - . $F\mu,v(z) = I(z) \times \Psi\mu,v(z)$	
- $I(z)$ es la imagen. - $\Psi_{\mu,v}(z)$ es el filtro de Gabor. - $z = (x, y)$ - X el operador de convolución.	

	Tarea de Ingeniería
Número Tarea: 10	Numero Historia: 07
Nombre Tarea: Almacenar las imágenes resultantes de la convolución (características extraidas) en archivos bmp..	
Tarea Asociada: Tarea 09, 11.	
Tipo de Tarea: Desarrollo	Puntos Estimados: 3
Programador: Programador 1	Iteración: 3
Descripción: .Almacenar las imágenes resultados de la convolución, estas imágenes son las represetaciones de wavelets de Gabor de los rostros almacenarlos para luego construir la matriz que características de Gabor.	

	Tarea de Ingeniería
Número Tarea: 11	Numero Historia: 08
Nombre Tarea: Construcción de la matriz de características de gabor.	
Tarea Asociada: Tarea 10.	
Tipo de Tarea: Desarrollo	Puntos Estimados: 3
Programador: Programador 1	Iteración: 3
Descripción: Concatenar las características de las imágenes por clase para crear la matriz de características de Gabor	

	Tarea de Ingeniería
Número Tarea: 12	Numero Historia: 08
Nombre Tarea: Ejecución de la extracción de características utilizando PCA..	
Tarea Asociada: Tarea 10.	
Tipo de Tarea: Desarrollo	Puntos Estimados: 3
Programador: Programador 1	Iteración: 3

Descripción:

Con la matriz construida de características de Gabor se calcular el Análisis de Conponentes Principal y prosigue como:

- Entonces, el conjunto de entrenamiento de imágenes de rostros será $\Gamma_1, \Gamma_2, \Gamma_3, \dots, \Gamma_M$. El rostro promedio del conjunto es definido por la ecuación

$$\Psi = \frac{1}{M}\sum_{i=1}^{M}\Gamma_i \qquad (3.6)$$

- Cada rostro se diferencia del promedio por el vector de la ecuación (3.7):

$$\Phi_i = \Gamma_i - \Psi, \mathrm{i} = 1, \dots, \mathrm{M} \qquad (3.7)$$

- A continuación calculamos la matriz de covarianza C, según la ecuación (3.8):

$$C = \sum_{i=1}^{N}\Phi_i{\Phi_i}^T = AA^T \qquad (3.8)$$

- La matriz C es N2 por N2, y determinar los N2 vectores y valores propios se vuelve una tarea intratable para el tamaño de una imagen típica. Necesitamos un método factible para encontrar los eigen vectores ecuación (3.9).

$$\begin{pmatrix} e_i = & Av_i \\ \lambda_i = & \mu_i \end{pmatrix} \qquad (3.9)$$

- De M eigen vectores (eigen faces) ei, solo escogemos M1 que tiene el valor propio (eigen value) más alto. Cuanto más alto sea el valor propio, describe la característica más representativa del rostro de un particular eigenvector. Los eigen face con bajos eigen values pueden ser omitidos, porque ellos solo explican una pequeña parte de los rasgos característicos del rostro. Luego M1 eigen faces ei son determinados, la fase de entrenamiento del algoritmo es finalizado.

	Tarea de Ingeniería
Número Tarea: 13	Numero Historia: 07
Nombre Tarea: Almacenar características extr aídas luego de aplicar la Wavelets de Gabor.	
Tarea Asociada: Tarea 12.	
Tipo de Tarea: Desarrollo	Puntos Estimados: 3
Programador: Programador 1	Iteración: 3

Descripción:

.Almacenar el vector de características en una estructura de datos.

4.3.2. Análisis del Módulo

4.3.2.1. Diagrama de Casos de Uso

El análisis de los requisitos para este módulo se resume en el diagrama de casos de uso mostrado en la Figura 30, además de los requisitos mostrados en el diagrama se implementó una etapa de pre procesamiento de las imágenes de entrada a la función encargada de la extracción de características basadas en la Wavelet de Gabor logrando mejorar los resultados, los detalles de la implementación de esta etapa son mostrados en la sección de implementación del módulo.

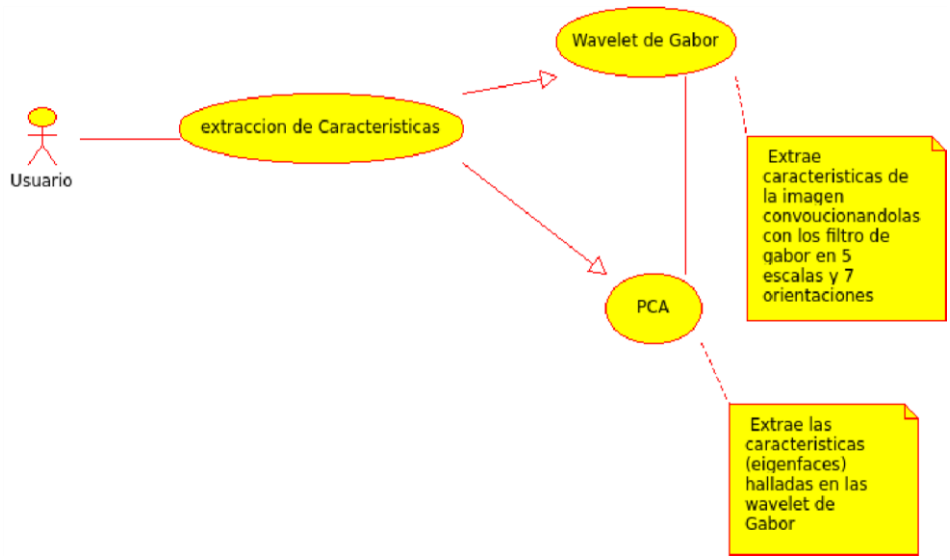


Figura 30: casos de uso para el módulo de extracción de Características.

4.3.3. Diseño del Módulo

4.3.3.1. Diagrama de Secuencias

La figura 31 detalla la secuencia de los procesos necesarios para la extracción de características.

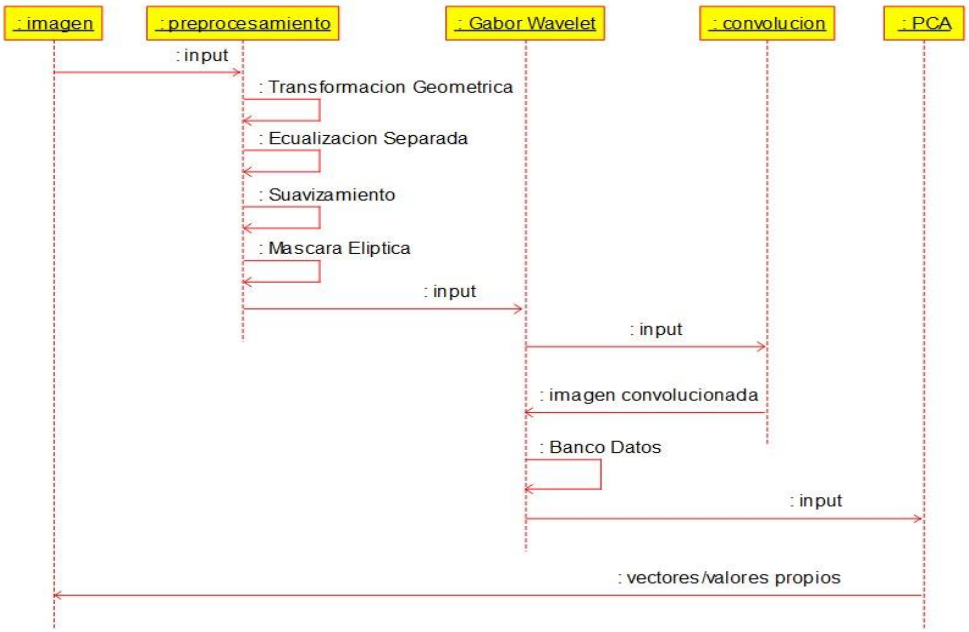


Figura 31: Diagrama de Secuencias para el módulo de extracción de características.

4.3.3.2. Arquitectura del Módulo

El modelo de la Arquitectura del módulo de extracción de características es mostrado en la Figura 32, se detallan los componentes de pre procesamiento, wavelets de Gabor, convolución PCA y los vectores de características.

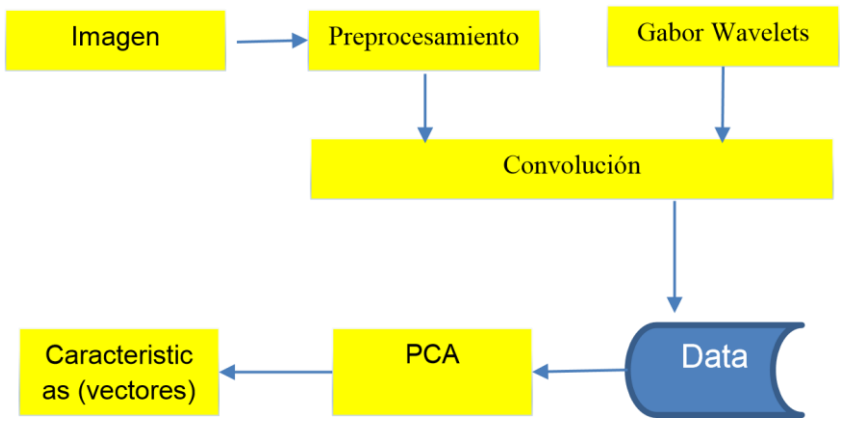


Figura 32: Arquitectura del módulo de extracción de Características.

4.3.4. Implementación del Módulo

La implementación del presente modulo se resumen en las Tablas 03 y 04 donde se muestra fragmento del código fuente desarrollado para la extracción de características basadas en las Wavelets de Gabor y el Análisis de Componentes Principales respectivamente. A continuación se exponen los resultados de la aplicación del pre procesamiento y los resultados Hallados para la extracción de características.

Tabla 03: Extracto código fuente para la extracción de Características Basadas en la Wavelet de Gabor.

```
void MainWindow::on_actionGabor_Wavelet_triggered()
{
    cv::Size ksize(10,10);
    cv::Mat gabor;
    cv::Mat inputfaces;
    float pi = 3.141592654;

    QString nombre;

    //obtener las imagenes de rostros
    //leemos las imagenesdesde archivo y almacenamos

    for(int i = 1; i<20; i++)
    {
        for(int j = 1; j<5; j++)
        {
            QString filename =
            "img_training/train/"+QString().number(i)+"/"+QString().number(j)+
            ".bmp";

            inputfaces =
                        cv::imread(filename.toAscii().data(),CV_LOAD_IMAGE_GRAY
                        SCALE);

            //calculamos las caracteristicas waveelts de las imagenes
            for(int k=0; k<4; k++)
            {
                for(int l=1; l<=8; l++)
                {
                     nombre = "img_training/train/" +QString().number(i)+"/"+
                     QString().number(k) + QString().number(l)
                     +QString().number(j)+ ".bmp";

                    gabor = cv::getGaborKernel(ksize, 2*pi, k, sqrt(2), l, pi/2,
CV_32F );
                    cv::filter2D(inputfaces, inputfaces, CV_32F, gabor);
                    cv::imwrite(nombre.toAscii().data(),inputfaces);
                }
            }
        }
    }
}
```

El módulo para la extracción de características se construyó en función de las técnicas propuestas es decir la Transformada Wavelet de Gabor y el Análisis de Componentes Principales. Primero se realizan la extracción de características aplicando la transformada Wavelet de Gabor a las imágenes de entrada obteniéndose una representación luego estas imágenes son analizadas mediante el Análisis de Componentes Principales, de este modo se hallan los vectores y valores propios que constituyen los vectores de características de la imagen.

Tabla 04: Extracto código fuente para la extracción de Características Basadas en Análisis de Componentes Principales.

```
void MainWindow::trainingPCA()
{
    //cv::Ptr<cv::FaceRecognizer> model = cv::createEigenFaceRecognizer();
    model = cv::createEigenFaceRecognizer();
    if(model.empty()){
        msgBox.setText("El algoritmo eigen face no esta en tu version");
        msgBox.exec();
    }else{
        std::vector<cv::Mat> images;
        std::vector<int> labels;
        cv::Mat inputfaces;
        //leemos las imágenes de archivo y almacenamos en un arreglo de imagenes
        for(int i = 1; i<20; i++){
            for(int j = 1; j<5; j++){
                QString filename =
"img_training/train/"+QString().number(i)+"/"+QString().number(j)+".bmp";
                inputfaces =
cv::imread(filename.toAscii().data(),CV_LOAD_IMAGE_GRAYSCALE);
                images.push_back(inputfaces);
        }}
        //colocamos las etiquetas
        for(int i = 1; i<20; i++){
            for(int j = 1; j<5; j++){
                labels.push_back(i);
            }
        }
        model->train(images, labels);
        // rostro promedio
        cv::Mat averageFace = model->get<cv::Mat>("mean");
        averageFace = getImageFrom1DFloatMat(averageFace,70);
        cv::namedWindow("averageFace");
        cv::imshow("averageFace",averageFace);
        // muestra los eigen values
        QString valor;
        cv::Mat eigenValues = model->get<cv::Mat>("eigenvalues");
        for(int i=0; i< eigenValues.rows; i++){
            valor = valor + QString().number(eigenValues.at<double>(i,0))+"\n";
        }
        msgBox.setText(valor);
        msgBox.exec();
}}
```

Además para el proceso de extracción de características se aplica un pre procesamiento a la imagen. Este pre procesamiento incluye:

- **Transformación geométrica**: incluye la detección de ojos basada en el clasificador Haar, rotación de la imagen basada en la posición de los ojos, y redimensión de la imagen obtenida.

Figura 33: Detección de Ojos.

Figura 34: Imagen redimensionada.

- **Ecualización separada:** Se secciona la imagen del rostro en dos, se aplica la ecualización basada en histograma a cada lado de la imagen tiene el objetivo de reducir el ruido producido por la luz.

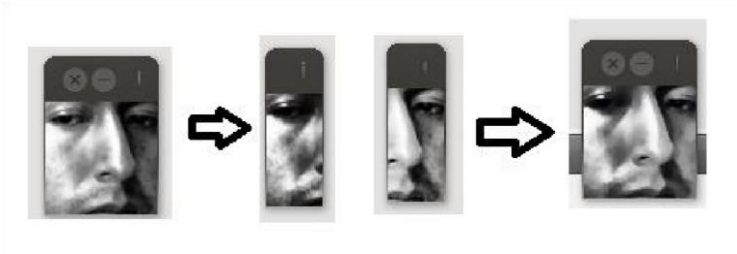


Figura 35: Ecualización Separada de la Imagen.

- **Suavizamiento de la imagen:** A la imagen ecualizada se aplica un filtro basado en la media aritmética.

- **Mascara Elíptica:** Finalmente se aplica una máscara elíptica a la imagen para quedarnos con la imagen de centrada en los rostros de las personas.

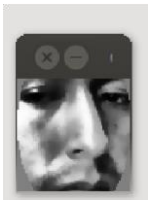


Figura 36: Respuesta Luego de Aplicar una máscara Elíptica.

4.3.5. Wavelet de Gabor

Se aplicaron 40 filtros de gabor a la imagen con 8 escalas diferentes y 5 direcciones diferentes, estas imágenes constituyen los vectores de características basadas en la wavelet de Gabor.

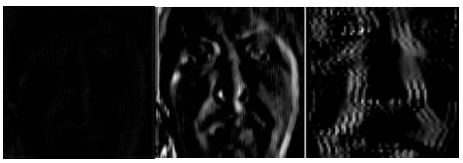


Figura 37: Imágenes filtradas con la Wavelet de Gabor.

4.3.6. Análisis de Componentes Principales PCA

Finalmente aplicamos el Análisis de Componentes Principales a las imágenes concatenadas de cada rostro que se aplicó la wavelet de Gabor. De esta manera reducimos la dimensionalidad de la imagen, y se constituye el vector de características de la imagen.

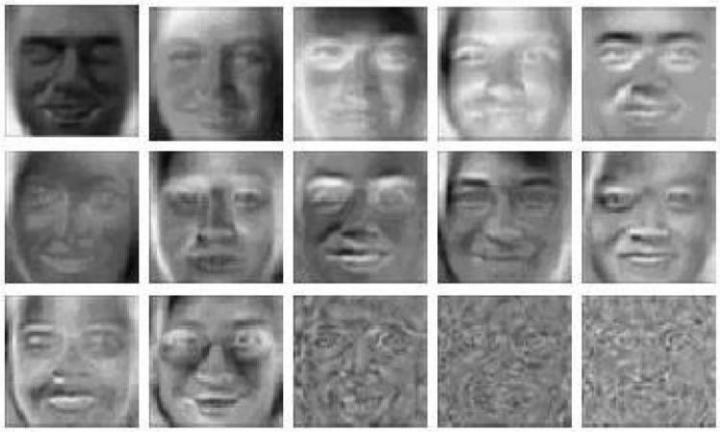


Figura 38 imágenes luego de aplicar PCA.

4.4. Clasificación de rostros basado en Maquinas de Soporte Vectorial.

4.4.1. Requisitos del Módulo

Se muestran a continuación los requisitos de usuario y las tareas de ingeniería para la realización de este módulo.

4.4.1.1. Requisitos de Usuario

Numero: 09	Historias de Usuario
Nombre Historia	Clasificación de rostros.
Quiero: Que le programa utilice SVM para clasificar los rostros.	
Para: Obtener resultados confiables de la identidad del rostro evaluado.	

4.4.1.2. Tareas de Usuario

	Tarea de Ingeniería
Número Tarea: 14	Numero Historia: 09
Nombre Tarea: Clasificar las características al macenadas por rostro.	
Tarea Asociada: Tarea 13.	
Tipo de Tarea: Desarrollo	Puntos Estimados: 3
Programador: Programador 1	Iteración: 3
Descripción: Dada una imagen de entrada nueva (imagen de prueba) el clasificador basado en SVM es capaz de identificar: - Si el Usuario Existe en la base de datos. - De existir el usuario identificar cuál de los almacenados es.	

	Tarea de Ingeniería
Número Tarea: 15	Numero Historia: 09
Nombre Tarea: Desarrollo del clasificador.	
Tarea Asociada: Tarea 13.	
Tipo de Tarea: Desarrollo	Puntos Estimados: 3
Programador: Programador 1	Iteración: 3

Descripción:

El clasificador esta basado en:

$$f(x) = \beta_0 + \beta^T x,$$

donde β es el vector ponderado y β_0 como bias.

$$|\beta_0 + \beta^T x| = 1$$

x Muestras de entrenamiento más cercano al hiperplano.

Usar la distancia de un punto x y el hiperplano (β, β_0):

$$\text{distance} = \frac{|\beta_0 + \beta^T x|}{\|\beta\|}.$$

Para un hiperplano canonico

$$\text{distance}_{\text{support vectors}} = \frac{|\beta_0 + \beta^T x|}{\|\beta\|} = \frac{1}{\|\beta\|}.$$

El margen M , es dos veces la distancia al plano escogido:

$$M = \frac{2}{\|\beta\|}$$

Finalmente

$$\min_{\beta,\beta_0} L(\beta) = \frac{1}{2}\|\beta\|^2 \text{ subject to } y_i(\beta^T x_i + \beta_0) \geq 1 \ \forall i,$$

donde y_i representa cada una de las etiquetas de entrenamiento.

4.4.2. Análisis del Módulo

4.4.2.1. Diagrama de Casos de Uso

La clasificación se efectúa, basado en un clasificador que utiliza las Máquinas De Soporte Vectorial, recibe como entrada los vectores hallados en la etapa de extracción de características luego realiza la operación con los valores almacenado.

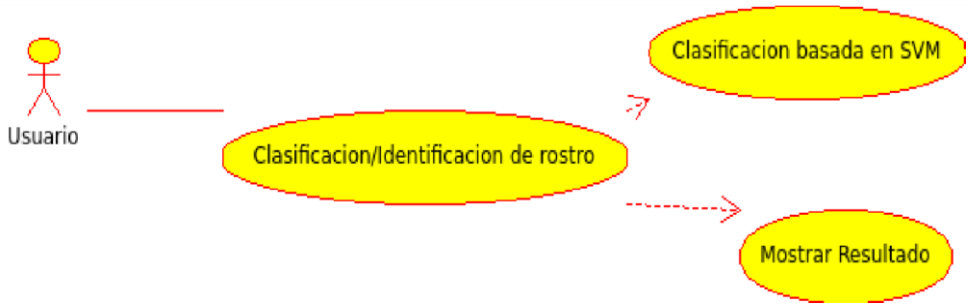


Figura 39: Diagrama de casos de uso para la clasificación de rostros.

4.4.3. Diseño del Módulo

4.4.3.1. Diagrama de Secuencias

El diagrama de secuencias de la figura 40 muestra el orden de ejecución de los procesos.

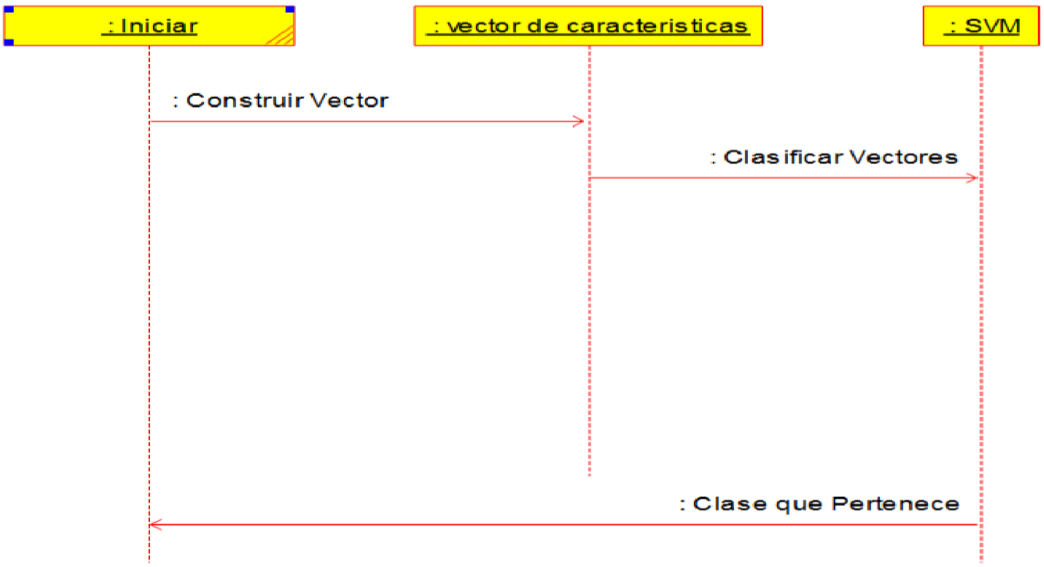


Figura 40: Diagrama de secuencia para el módulo de clasificación de rostros.

4.4.3.2. Arquitectura del Módulo

La figura 41 resume la arquitectura del módulo de clasificación de rostros

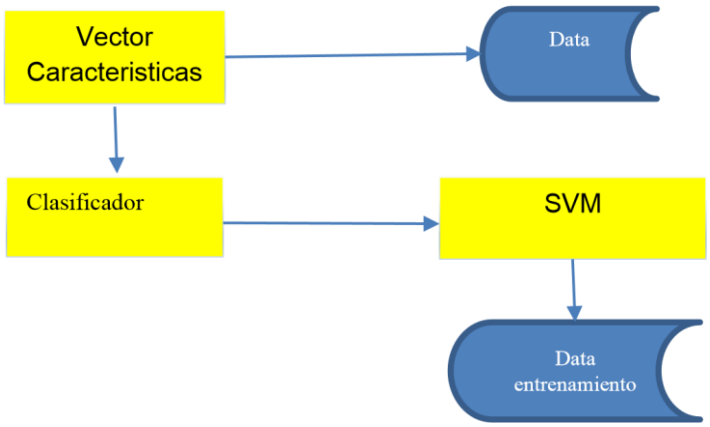


Figura 41: Arquitectura del módulo de clasificación de rostros.

4.4.4. Implementación del Módulo

La implementación del presente modulo se resume en las Tablas 05 y 06. Donde se muestra fragmento del código fuente desarrollada.

Tabla 05: Extracto código fuente para clasificación de rostros basado en SVM.

```
void MainWindow::matToLine(cv::Mat &img)
{
    cv::SVMParams params;
    params.svm_type = CvSVM::C_SVC;
    params.kernel_type = CvSVM::POLY;
    params.gamma = 3;

    int columnas = 70*70;
    int filas = 19*4;
    cv::Mat training_mat(filas,columnas,CV_32FC1);
    int columna =0;
    cv::Mat labels;

    cv::Mat inputfaces;
    //obtener las imagenes de rostros
    //leemos las imagenesdesde archivo y almacenamos el vector fila en
training_mat
    for(int i = 1; i<20; i++)
    {
        for(int j = 1; j<5; j++)
        {
            QString filename =
"img_training/train/"+QString().number(i)+"/"+QString().number(j)+".bmp";
            inputfaces =
cv::imread(filename.toAscii().data(),CV_LOAD_IMAGE_GRAYSCALE);
            //convertimos la imagen en filas
            for(int k =0; k<70; k++)
            {
                for(int l=0;l<70; l++)
                {

training_mat.at<float>(j,columna++)=inputfaces.at<uchar>(k,l);
                }
            }
        }
    }
    int fila = 0;
    for(int i = 1; i<20;i++)
    {
        for(int j=1;j<5;j++)
        {
            labels.at<uchar>(fila, 0)=i;
            fila++;
        }
    }

    cv::SVM(training_mat, labels, cv::Mat(), cv::Mat(), params);
}
```

Tabla 06: Extracto código fuente predicción de la Clase del rostro.

```
void MainWindow::PredicSVM(cv::Mat &img)
{
    int clase = svm.predict(img_mat_1d);
msgBox.setText(clase);     msgBox.exec();


}
```

4.5. Contrastación de Hipótesis

Se exponen a continuación los resultados obtenidos de aplicar el software construido a la tarea de reconocimiento automático de rostros.

Concretamente la evaluación de la combinación de las Wavelets de Gabor y Análisis de Componentes Principales para la extracción de características. Y la Maquina de Soporte Vectorial para la clasificación de rostros.

Para la contrastación de la hipótesis:

“El uso de las técnicas hibridas en la detección y reconocimiento automático de rostros en imágenes y secuencias de video permite que el sistema sea eficiente”.

El presente trabajo evalúa la efectividad del sistema con un nivel de significancia del 5% ($\alpha = 0.05$). Además se calculó la proporción de los aciertos mediante las siguientes tasas:

- Falsa aceptación
- Falso rechazo
- Error Igual.

Luego se realizó la contrastación de Hipótesis mediante la prueba Z para la proporción.

4.5.1. Tasa de falsa aceptación (FAR)

Datos:

- Número de falsas aceptaciones = 1

- Numero de impostores = 48

Calculo de FAR

$$FAR = \frac{número\ de\ falsas\ aceptaciones}{número\ de\ impostores} = \frac{1}{48} = 0.02 = 2\%$$

La tasa de falsa aceptación FAR es igual al 0.02 que representa el 2%, que nos indica que para la prueba realizada existe un 2 % de posibilidad de aceptar un usuario no registrado en la base de datos.

4.5.2. Tasa de falso rechazo (FRR)

Datos:

- Numero de falsos rechazos = 1

- Numero de rostros evaluados = 48

$$FRR = \frac{número\ de\ falsos\ rechazos}{número\ de\ rostros\ evaluados} = \frac{1}{48} = 0.02 = 2\%$$

La tasa de falso rechazo FRR es igual a 0.02 que representa el 2%, que nos indica que para la prueba realizada existe un 2 % de posibilidad de rechazar un usuario registrado en la base de datos

4.5.3. Tasa de Error Igual

Datos:

- Número de falsas aceptaciones = 1

- Número de falsos rechazos = 1

- Número total de entradas = 96

$$EER = \frac{número\ de\ falsas\ aceptaciones + número\ falsos\ rechazos}{número\ total\ de\ entradas}$$

$$EER = \frac{1+1}{96} = 0.02 = 2\%$$

La tasa de error igual es 0.02 que representa el 2 % que nos indica que para la prueba realizada existe un 2 % de cometer un error (tipo I o tipo II), también podemos decir que el sistema proporcionó resultados exactos en un 98%.

4.5.4. Contrastación de Hipótesis

Para determinar si la inclusión de las técnicas hibridas en la detección y reconocimiento de rostros permite que el sistema sea eficiente en imágenes y secuencias de video se aplicara una Prueba Z para la proporción a los datos que serán resultados de la evaluación del sistema.

Datos:

$\alpha = 0.05$

$n = 96$

$p = 98\%$

$\pi = 95\%$

Planteamiento de Hipótesis

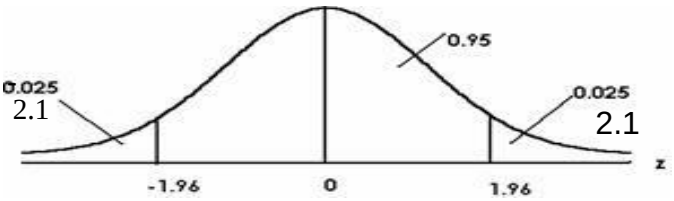
$H_0: \pi = 95\%$

$H_a: \pi \neq 95\%$

Cálculo del valor Z

$$Z = \frac{p - \pi_H}{\sigma_p} = \frac{0.98 - 0.95}{\sqrt{\frac{0.98 * 0.02}{96}}} = 2.1$$

Gráfica de región de aceptación



Contrastación con Z de tabla

Como el valor de Z calculada es igual a 2.1 mayor a Z de Tabla al 95% de probabilidad debemos de rechazar la hipótesis nula y aceptar la alterna, la proporción obtenida es mayor al 95%.

Conclusión

Podemos concluir que según la prueba de de hipótesis para la proporción el porcentaje de respuestas correctas del sistema es mayor al 95%.

CONCLUSIONES

1. Se verificó que el desarrollo del sistema para la detección y reconocimiento de rostros basadas en técnicas híbridas logró alcanzar un porcentaje del 98% superando al porcentaje esperado que era del 95%, también se obtuvo una tasa de falsos aceptados del 2 % y una tasa de falsos rechazados del 2 %.

2. Se concluye que, la base de datos de imágenes utilizadas constituida por imágenes de tres colecciones diferentes (FERET, ORL, y la de los investigadores) no presentaron diferencias significativas para el desarrollo y prueba del sistema.

3. Se concluye que, la implementación del módulo de detección de rostros basada en un clasificador a cascada fue útil para poder realizar la posterior tarea de reconocimiento de rostros. El módulo implementado para la detección de rostros realizó la detección de rostros satisfactoriamente,

4. Se concluye que, la combinación de las técnicas Wavelet de Gabor y Análisis de Componentes Principales en las tareas de extracción de características logro alcanzar una tasa de falsos aceptados del 2 % y una tasa de falsos rechazados del 2 %. Esto demuestra un alto grado de confiabilidad para la constitución de vectores característicos. De esta manera las wavelets de Gabor demostraron ser adecuadas para la extracción de característica, debido a sus propiedades de orientación, localización espacial y optima localización en el espacio y dominio de la frecuencia. El algoritmo Análisis de Componentes Principales PCA es un

método global que usa principalmente los niveles de gris de los pixeles de una imagen. La simplicidad para la implementación del algoritmo contrasta con una fuerte sensibilidad a cambios en la luz, posición y expresiones faciales. Es por eso que aumentamos el número de posturas para cada persona. Sin embargo, el ACP no requiere ningún conocimiento a priori sobre la imagen El principio de que se puede construir un espacio de sub-vectores conservando sólo los mejores vectores propios, al tiempo que conserva una gran cantidad de información útil, hace que el PCA sea un algoritmo efectivo y de uso común en la reducción de dimensional dad, donde puede entonces ser utilizado con otros algoritmos para mejorar los resultados de la aplicación.

5. Se concluye que, la ejecución del clasificador basado en máquinas de Soporte Vectorial, realiza de manera efectiva el reconocimiento de rostros, alcanzando un porcentaje del 98% de efectividad luego de la evaluación. Así, afirmamos que el reconocimiento de individuos mantiene un problema complejo y que a pesar de ser un área activa de investigación aún existen muchas dificultades en su aplicación en condiciones reales, dado que la presente investigación logro una tasa de reconocimiento efectivo de 98%, no logrando resolver el problema en un 100% de los casos.

RECOMENDACIONES Y SUGERENCIAS

- Para lograr una tasa más elevada de aserción en la tarea de reconocimiento automático de rostros se recomienda ampliar la extracción de características a través de técnicas geométricas tales como la distancia entre los puntos característicos de los rostros de los individuos y desarrollar una combinación de métodos efectivo.

- Para resolver el problema de aplicación del sistema en ambientes no controlados se recomienda a los investigadores la inclusión de técnicas basadas en color y forma así como evaluar la aplicación de técnicas de computación paralela aprovechando el número de núcleos de los procesadores actuales existentes en el mercado.

- Debido a que las imágenes digitales pueden ser representados a través de matrices se recomienda el uso del algebra lineal y su teoría para proponer nuevos modelos que mejoren las tareas de detección y reconocimiento de rostros.

- Debido a la amplia aplicación de la estadística en la teoría de decisión se recomienda a los estudiantes realizar investigaciones en biometría, visión por computador, interacción humano computador y otros.

- Finalmente se recomienda utilizar la documentación, métodos, conclusiones y resultados del presente trabajo solo como una referencia, debido a que el sistema desarrollado no alcanzo a resolver el problema de reconocimiento de rostros en un 100%.

BIBLIOGRAFÌA

Albuz. (2001). Scalable Color Image Indexign and Retrieval usign Vector Wavelets. *IEEE Transaction on knowledge and Data Engineering*, 851-861.

Broughton, s. A., & Bryan, K. (2009). *Discrete Fourier Analysis and Wavelets.* Canada: John Wiley.

Coggins. (1985). A Spatial Filtering Approacj to Texture Analysis. *Pattern Recognition Letter*, 195-203.

Daubechies. (1992). Ten lectures on Wavelets. *CBMS-NFS Regional Conference Series in Applieed Mathematics*, .

Daubechies, I. (1990). The Wavelet Transform, Time Frequency Localization, and Signal Analysis. *IEEE Transactions on Information Theory*, 961-1005.

Daugman, D. (1988). Complete Discrete 2-D Gabor Transforms by Neural Networks for Image Analysis and Compession. *IEEE Transactions on Acoustic, Speech, and Signal Processing*, 1169-1179.

Daugman, D. (2002). How Iris Recognition Works. *In Proceedings of 2002 International Congerence on Image Processing*, 33-36.

Feris, R. S., & Krueger, V. (2001). Efficient Real Time Face Tracking in Wavelets Subspace. *Proceedings of ICCV 2001 International Congerene on Computer Vision, Workshop on Recognition, Analysis and Tracking of Faces and Gestures in Real-Time Systems*, 113-118.

Gabor, D. (1946). *Theory of Communication.* London: J.IEE.

Gerbrands, J. (1981). On the relationships between SVD, KLT and PCA. *Pattern Recognition*, 375-381.

Gonzales. (2007). *Digital Image Processing.* Pearson Education International.

Grossmann, A., & Morlet, J. (1984). Decomposition of hardy Function into Square Integrable Waveles of Constant Shape. *SIAM, Journal of Mathematical Analysis*, 723-736.

Guttman. (1984). A dynamic Index Structure for Spatial Searching. *Proceeding of the International Conference on Data Management (ACM-SIGMOD)*, 47-57.

He, C., Dong, J., & Sheng, Y. (2001). Object Tracking Using the Gabor Wavelet Transform and the Golden Section Algorithm. *Proceeding of the 2001 IEEE International Conference on Robotic and Automation* (pp. 1671-1676). Seoul, Korea: 1671-1676.

Hu. (1962). Visual Pattern Recognition by Moment Invariants. *IRE Transaction on Information Theory*, 179-187.

Jain, A. &. (1998). A Multiscale Representation Including Opponent Color Features for Texture Recognition. *IEEE Transactions in Inage Processing*, 124.128.

Jain, A., & farrokhnia, F. (1991). Unsupervised Texture Segmentation using Gaborr Filters. *Pattern Recognition*, 1167-1186.

Jain, A., Ross, A., & Prabhakar, S. (2001). Fingerprint Marching using Minutiae and Texture Features. *in Proc. of Int'l Conference on Image Processing (ICIP)*, 282-285.

Lee, T. S. (1996). Image Representation Using 2D Gabor Wavelets. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 959-971.

Li, S. (2011). *Handbook of Face Recognition.* London: Springer-Verlag.

Loeve, M. (1955). *Probability Theory.* Princeton: Van Nostrand.

Loew. (2000). Feature extraction. In M. H. Loew, *Medical Image Processing and Analysis* (pp. 273-341). SPIE Press.

Ma, W., & Manjunath, B. (1995). Image Indexing Using a Texture Dictionary. *SPIE Conference on Image Storage and Archiving System*, 188-198.

Malik. (1990). Preattentive Texture Discrimination with early Vision Mechanisms. *Journal of Optical Society of America*, 923-932.

Mallar, S. G. (1988). *Multiresolution Representation and Wavelet.* Pennsylvania: University of Pennsylvania.

Mallat, S. (1989). A Theory for Multiresolution Signal Decomposition: The Wavelet Representation. *IEEE Transaction on Pattern analysis and Machine intelligence*, 674-693.

Mallat, S. G. (1999). *A Wavelet Tour of Signal Processing.* Academic Press, 2nd. edition.

Manjunath, B. a. (1996). Texture Features for Browsin and Retrieval of Image Data. *IEEE Transaction on Pattern Analysis and Machine Intelligence*, 837-842.

Meyer, Y. (1993). *Algorithms and Applications.* Philadelphia: Society for industrial and Applied Mathematics.

Mou, D. (2010). Machine-based Intelligent Face Recognition. In D. Mou, *Automatic Face Recognition* (pp. 91-106). London: Springer.

Picard, R. (1995). Vision Texture for Anotation. *Multimedia Systems: Special Issue on Content-Based Retrieval*, 3-14.

Rubner, Y. a. (2000). Perceptual Metrics for Image Database Navigation. *Series in Enginering and Comuter Science*.

Smeulders, W. S. (2000). Content-Based Image Retrieval at the End of the Early Years. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 1349-1380.

Sonka. (2007). *Image Processing, Analysis, and Machine Vision.* Pacific Grove.

Stollnitz. (1996). *Wavelets for Computer Graphics, Theory and Applications.* San Francisco: Morgan Kaufmann Publisher.

Team, o. d. (2013, 11 08). *http://docs.opencv.org.* Retrieved 11 23, 2013, from http://docs.opencv.org: http://docs.opencv.org/doc/tutorials/ml/introduction_to_svm/introduction_to_svm.html

Theodoridis. (1999). *Pattern Recognition.* Greece: Academic Press.

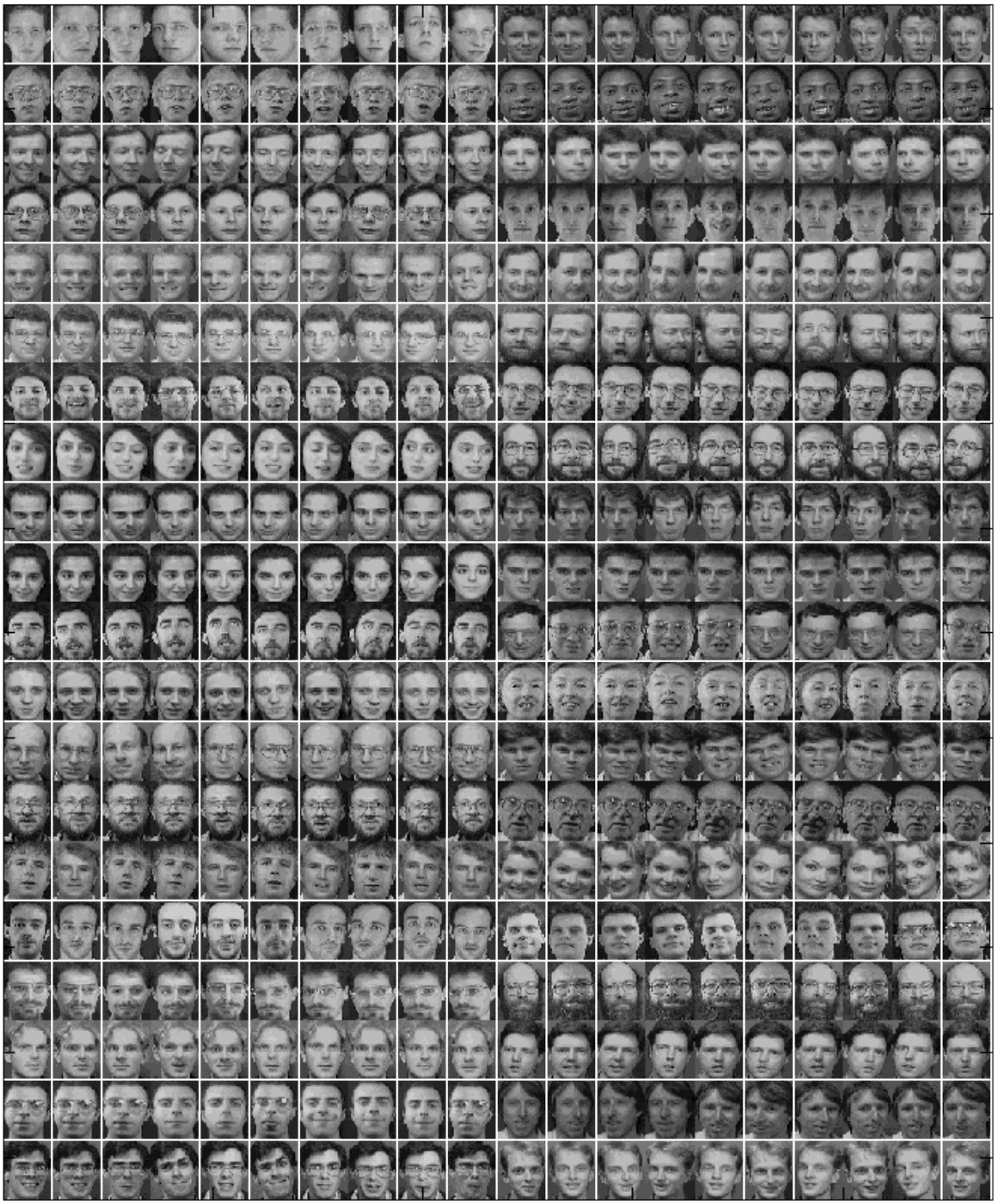
Traina. (2001). *Suporte à Visualizacao de Consultas por Dimilaridade em Imagens Médicas Atravpes de Estructuras de Indexacao Métrica.* Sao Paulo - Brasil: Instituto de Ciencias Matemáticas y Computación - Universidad Sao Paulo.

Tucerayan. (1993). Texture Analysis. In Tucerayan, *Handbook of Pattern Recongnition and Computer Vision* (pp. 235-276). Worl Scientific Publishing Company.

Vailaya. (2000). *Semantic Classification in Image Databases.* Michigan: Michigan State University.

ANEXOS

BASE DE DATOS ORL



BASE DE DATOS FERET

CÓDIGO FUENTE

```
#include "mainwindow.h"
#include "ui_mainwindow.h"

MainWindow::MainWindow(QWidget *parent) :
    QMainWindow(parent),
    ui(new Ui::MainWindow)
{
    winImagen = winVideo = winTReal = false;
    ui->setupUi(this);
}


MainWindow::~MainWindow()
{
    delete ui;
}


void MainWindow::on_actionSalir_triggered()
{
    this->close();
}


void MainWindow::on_actionDesde_Archivo_triggered()
{
    fileName = QFileDialog::getOpenFileName(this, tr("Abrir imagen"), ".", tr("Archivos
de imagen (*.jpg *.png *.jpeg *.bmp *.ppm *.ljpeg)"));
    image = cv::imread(fileName.toAscii().data());
    cv::namedWindow("Imagen de Entrada");
    cv::imshow("Imagen de Entrada", image);
    winImagen = true;
}


void MainWindow::on_actionVideo_triggered()
{
    videoFileName = QFileDialog::getOpenFileName(this, tr("Abrir imagen"), ".",
tr("Archivos de imagen (*.avi *.mov *.mpeg)"));
    video.open(videoFileName.toAscii().data());
    if(!video.isOpened()){
        msgBox.setText("No se pudo abrir el video, Intentelo nuevamente");
        msgBox.exec();
    }
    else
    {
        ui->actionTomar_Imagen->setDisabled(false);

        winVideo = true;
        cv::namedWindow("Video",1);
        for(;;)
        {
            video>> frame;
            //cv::cvtColor(frame, imgVideo, CV_BGR2GRAY);
            imgVideo = frame;
            cv::imshow("Video", imgVideo);
            if(cv::waitKey(30)>=0)
            {
                video.release();
                cv::destroyWindow("Video");
                break;
            }
        }
        winVideo = false;
    }
}
```

```
void MainWindow::on_actionTiempo_Real_triggered()
{
    tRealVideo.open(0);

    if(!tRealVideo.isOpened()){
        msgBox.setText("No se pudo abrir el video, Intentelo nuevamente");
        msgBox.exec();
    }
    else
    {
        winTReal = true;
        ui->actionTomar_Imagen->setEnabled(true);
        cv::namedWindow("Tiempo Real",1);
        for(;;)
        {
            tRealVideo>> frame;
            /*cv::cvtColor(frame, edges, CV_BGR2BGRA);
            GaussianBlur(edges, edges, cv::Size(7,7), 1.5, 1.5);
            Canny(edges, edges, 0, 30, 3);*/
            imgTReal = frame;
            cv::imshow("Tiempo Real", imgTReal);
            if(cv::waitKey(30)>=0)
            {
                ui->actionTomar_Imagen->setEnabled(false);
                cv::destroyWindow("Tiempo Real");
                break;
            }
        }
        winTReal = false;
    }
}


void MainWindow::on_actionTomar_Imagen_triggered()
{
    QDir dirImgTest;
    int contador;

    contador = dirImgTest.count();

    QString mensaje = QString("numero de directorios: %1").arg(contador);
    msgBox.setText("Se capturo la imagen, y se guardo en el directorio principal");
    msgBox.exec();

    save_imagen = frame;
    cv::namedWindow("imagen capturada", 1);
    cv::imshow("imagen capturada",save_imagen );
    cv::imwrite("img_test/imagen1.bmp",save_imagen);
}


void MainWindow::on_actionEscala_de_Gris_2_triggered()
{
    if(winImagen)
    {
        cv::cvtColor(image,image,CV_BGR2GRAY);
        cv::imshow("Imagen de Entrada", image);
    }
    if(winVideo)
    {
        video.release();
        cv::destroyWindow("Video");

        video.open(videoFileName.toAscii().data());
        if(!video.isOpened()){
            msgBox.setText("No se pudo abrir el video, Intentelo nuevamente");
            msgBox.exec();
        }
        else
        {
            winVideo = true;
            ui->actionTomar_Imagen->setDisabled(false);
            cv::namedWindow("Video",1);
            for(;;)
            {
                video>> frame;
                cv::cvtColor(frame, imgVideo, CV_BGR2GRAY);
```

```
                cv::imshow("Video", imgVideo);
                if(cv::waitKey(30)>=0)
                {
                    cv::destroyWindow("Video");
                    break;
                }
            }
        }
        winVideo = false;
    }
    if(winTReal)
    {
        tRealVideo.release();
        cv::destroyWindow("Tiempo Real");

        tRealVideo.open(0);
        if(!tRealVideo.isOpened())
        {
            msgBox.setText("intente tiempo real nuevamente");
            msgBox.exec();
        }
        else
        {
            winTReal = true;
            ui->actionTomar_Imagen->setDisabled(false);
            cv::namedWindow("Tiempo Real",1);
            for(;;)
            {
                tRealVideo>> frame;
                cv::cvtColor(frame, imgTReal, CV_BGR2GRAY);
                cv::imshow("Tiempo Real", imgTReal);
                if(cv::waitKey(30)>=0)
                {
                    cv::destroyWindow("Tiempo Real");
                    break;
                }
            }
        }
        winTReal = false;
    }
}


void MainWindow::on_actionNormalizar_triggered()
{
    if(winImagen)
    {
        int nl= image.rows; // number of lines
        // total number of elements per line
        int nc= image.cols * image.channels();
        int div;

        bool ok;
        QString text = QInputDialog::getText(this, tr("QInputDialog::getText()"),
                                              tr("Factor de Reduccion:"),
QLineEdit::Normal,
                                          "64", &ok);
        if (ok && !text.isEmpty())
            div  = text.toInt();

        for (int j=0; j<nl; j++) {
            // get the address of row j
            uchar* data= image.ptr<uchar>(j);
            for (int i=0; i<nc; i++) {
                // process each pixel ---------------------
                data[i]=data[i]/div*div + div/2;
                // end of pixel processing ----------------
            } // end of line
        }
        cv::imshow("Imagen de Entrada", image);
        cv::imwrite(fileName.toAscii().data(), image);

        //colorReduce(image);
    }
}
```

```
void MainWindow::on_actionHSI_triggered()
{
    if(winImagen)
    {
        cv::cvtColor(image,image,CV_RGB2HSV);
        cv::imshow("Imagen de Entrada", image);
    }
}


void MainWindow::on_actionHaar_Detector_triggered()
{
    anchoDeteccion = 320;
    flags = cv::CASCADE_SCALE_IMAGE;
    cv::Size minFeatureSize(20,20);
    float searchScaleFactor = 1.1F;
    int minNeighbors = 4;
    //para escoger la fuente de la imagen
    int fuente;
    bool ok;
    QString text = QInputDialog::getText(this, tr("QInputDialog::getText()"),
                                          tr("fuente 0:imagen,1:tiempo real"),
QLineEdit::Normal,
                                       "1", &ok);
    if (ok && !text.isEmpty())
        fuente  = text.toInt();

    flags = cv::CASCADE_SCALE_IMAGE;
    minFeatureSize.height = 20;
    minFeatureSize.width = 20;

    if(fuente==1)
    {
        tRealVideo.open(0);
        if(!tRealVideo.isOpened())
        {
            msgBox.setText("Intente nuevamente");
            msgBox.exec();
        }
        else
        {
            //******* cargar el archivo xml para haar face deteccion
            try
            {
                faceDetector.load("haar_xml/haarcascade_frontalface_default.xml");
            }
            catch(cv::Exception e)
            {            }
            if ( faceDetector.empty() ) {
                msgBox.setText("No se cargo, el clasificador en cascada, intentelo
nuevamente");
                msgBox.exec();
            }
            double oldTime;
            oldTime = cv::getTickCount();
            int j=1;
            for(;;)
            {
                tRealVideo>> frame;
                filas = frame.rows;
                columnas = frame.cols;
                QString mensaje;
                mensaje = "Dimension imagen original:"+
QString().setNum(filas)+"x"+QString().setNum(columnas);
                ui->statusBar->showMessage(mensaje);
                float scale = columnas/(float) anchoDeteccion;
                if(columnas>anchoDeteccion)
                {
                    int escalaAltura = cvRound(filas/scale);
                    cv::resize(frame, smallImg, cv::Size(anchoDeteccion, escalaAltura));
                    cv::cvtColor(smallImg,gray ,CV_BGR2GRAY);
                    cv::equalizeHist(gray, imgTReal);
                    std::vector<cv::Rect> faces;

faceDetector.detectMultiScale(imgTReal,faces,searchScaleFactor,minNeighbors, flags,
minFeatureSize);
                    if(faces.size()>0)
```

```
                    {
                        for(int i = 0; i<(int)faces.size();i++)
                        {
                            faces[i].x = cvRound(faces[i].x*scale);
                            faces[i].y = cvRound(faces[i].y*scale);
                            faces[i].width = cvRound(faces[i].width*scale);
                            faces[i].height = cvRound(faces[i].height*scale);
                            //cuando el objeto este en un borde
                            if (faces[i].x < 0)
                                faces[i].x = 0;
                            if (faces[i].y < 0)
                                faces[i].y = 0;
                            if (faces[i].x + faces[i].width > columnas)
                                faces[i].x = columnas - faces[i].width;
                            if (faces[i].y + faces[i].height > filas)
                                faces[i].y = filas - faces[i].height;
                            //--------------dibuja un rectangulo en el rostro
                            cv::rectangle(frame,faces[i],CV_RGB(0,255,0), 1);
                            //--------------- guarda los rostros detectados
                            QString nombreImagen =
QString("img_test/rostro_%1.bmp").arg(i);
                            faceDetected= frame(faces[i]);
                            cv::imwrite(nombreImagen.toAscii().data(), faceDetected);
guarda solo si paso un segundo
                            if((cv::getTickCount()-oldTime)/cv::getTickFrequency() >= 1)
                            {
                                nombreImagen = "img_test/" + QString().setNum(j) +
QString("g_rostro_%1.bmp").arg(i);
                                cv::cvtColor(faceDetected, faceDetected, CV_RGB2GRAY);
                                cv::imwrite(nombreImagen.toAscii().data(),
faceDetected);
                                j++;
                                oldTime = cv::getTickCount();
                            }
                        }
                    }
                    if(cv::waitKey(30)>=0)
                    {
                        ui->actionTomar_Imagen->setEnabled(false);
                        cv::destroyWindow("reducida");
                        break;
                    }
                    cv::imshow("reducida", imgTReal);
                    cv::imshow("Original", frame);  // imagen reducida a color
                }
                else
                {
                    cv::cvtColor(frame,gray ,CV_BGR2GRAY);
                    cv::equalizeHist(gray, imgTReal);

                    QString mensaje;
                    mensaje = "Dimension: "+
QString().setNum(filas)+"x"+QString().setNum(columnas);
                    ui->statusBar->showMessage(mensaje);
                    std::vector<cv::Rect> faces;
faceDetector.detectMultiScale(imgTReal,faces,searchScaleFactor,minNeighbors, flags,
minFeatureSize);
                    if(faces.size()>0)
                    {
                        for(int i = 0; i<(int)faces.size(); i++)
                        {
                            QString ruta = QString("img_test/rostro%1.bmp").arg(i);
                            faceDetected = frame(faces[i]);
                            cv::rectangle(frame, faces[i], CV_RGB(0,255,0), 1);
                            cv::imwrite(ruta.toAscii().data(),faceDetected);
                            }
                    }
                    cv::imshow("Tiempo Real", frame);
                    if(cv::waitKey(30)>=0)
                    {
                        ui->actionTomar_Imagen->setEnabled(false);
                        cv::destroyWindow("Tiempo Real");
                        break;
                    }
                }
            }
        }
```

```
    }
    else if(fuente==0)
    {
        try
        {
            faceDetector.load("haar_xml/haarcascade_frontalface_default.xml");
        }
        catch(cv::Exception e)
        {            }
        if ( faceDetector.empty() ) {
            msgBox.setText("No se cargo, el clasificador en cascada");
            msgBox.exec();
        }
fileName = QFileDialog::getOpenFileName(this, tr("Abrir imagen"), ".", tr("Archivos de
imagen (*.jpg *.png *.jpeg *.bmp *.ppm *.pgm)"));
        image = cv::imread(fileName.toAscii().data());
        cv::cvtColor(image,gray ,CV_BGR2GRAY);
        cv::equalizeHist(gray, imgTReal);
        std::vector<cv::Rect> faces;
faceDetector.detectMultiScale(imgTReal,faces,searchScaleFactor,minNeighbors, flags,
minFeatureSize);
        if(faces.size()>0)
        {
            for(int i = 0; i<(int)faces.size();i++)
            {
                cv::rectangle(image,faces[i],CV_RGB(0,255,0), 1);
                QString nombreImagen = QString("img_test/rostro_%1.bmp").arg(i);
                faceDetected= image(faces[i]);
                cv::imwrite(nombreImagen.toAscii().data(), faceDetected);
                nombreImagen = "img_test/" + QString("g_rostro_%1.bmp").arg(i);
                cv::cvtColor(faceDetected, faceDetected, CV_RGB2GRAY);
                cv::imwrite(nombreImagen.toAscii().data(), faceDetected);
            }
        }
        cv::imshow("Original", image);  //muestra la imagen reducida a color
    }
}


void MainWindow::on_actionGabor_Wavelet_triggered()
{
    cv::Size ksize(10,10);
    cv::Mat gabor;
    cv::Mat inputfaces;
    float pi = 3.141592654;

    QString nombre;
    //obtener las imagenes de rostros
    //leemos las imagenesdesde archivo y almacenamos
    for(int i = 1; i<20; i++)
    {
        for(int j = 1; j<5; j++)
        {
            QString filename =
"img_training/train/"+QString().number(i)+"/"+QString().number(j)+".bmp";
            inputfaces = cv::imread(filename.toAscii().data(),CV_LOAD_IMAGE_GRAYSCALE);
            //calculamos las caracteristicas waveelts de las imagenes
            for(int k=0; k<4; k++)
            {
                for(int l=1; l<=8; l++)
                {
                    nombre = "img_training/train/" +QString().number(i)+"/"+
QString().number(k) + QString().number(l) +QString().number(j)+ ".bmp";
                    gabor = cv::getGaborKernel(ksize, 2*pi, k, sqrt(2), l, pi/2, CV_32F
);
                    cv::filter2D(inputfaces, inputfaces, CV_32F, gabor);
                    cv::imwrite(nombre.toAscii().data(),inputfaces);
                }
            }
        }
    }
}
```

```
void MainWindow::on_actionImagen_de_Prueba_triggered()
{
    fileName = QFileDialog::getOpenFileName(this, tr("Abrir imagen"), ".", tr("Archivos
de imagen (*.jpg *.png *.jpeg *.bmp *.ppm *.pgm)"));
    image = cv::imread(fileName.toAscii().data());
    cv::namedWindow("Imagen de Entrada");
    cv::imshow("Imagen de Entrada", image);
    winImagen = true;
}


void MainWindow::on_actionDeteccion_de_ojos_triggered()
{
    float EYE_SX = 0.16, EYE_SY=0.26, EYE_SW=0.30, EYE_SH=0.28;
    cv::Size minFeatureSize(20,20);
    int leftX = cvRound(image.cols*EYE_SX);
    int topY = cvRound(image.rows*EYE_SY);
    int widthX = cvRound(image.cols * EYE_SW);
    int heightY = cvRound(image.rows*EYE_SH);
    int rightX = cvRound(image.cols*(1.0-EYE_SX-EYE_SW));
    cv::Rect izquierdo = cv::Rect(leftX, topY, widthX, heightY);
    cv::Rect derecho = cv::Rect(rightX, topY, widthX, heightY);
    cv::Mat topLeftOfFace = image(izquierdo);
    cv::Mat topRightOfFace = image(derecho);
    cv::namedWindow("deteccion ojos");
    cv::imshow("deteccion ojos",image);
    cv::CascadeClassifier eyeDetector1("haar_xml/haarcascade_eye.xml");
    std::vector<cv::Rect> leftEyeRect;
eyeDetector1.detectMultiScale(topLeftOfFace,leftEyeRect,1.1F,4,cv::CASCADE_SCALE_IMAGE,
minFeatureSize);
    leftEye = cv::Point(-1,-1);
    if(leftEyeRect.size()>0)
    {
        leftEye.x = leftEyeRect[0].x + leftEyeRect[0].width/2 + leftX;
        leftEye.y = leftEyeRect[0].y + leftEyeRect[0].height/2 + topY;

        cv::namedWindow("deteccion ojos izquierdo");
        cv::imshow("deteccion ojos izquierdo",topLeftOfFace);
    }
    std::vector<cv::Rect> rightEyeRect;
eyeDetector1.detectMultiScale(topRightOfFace,rightEyeRect,1.1F,4,cv::CASCADE_SCALE_IMAGE
,minFeatureSize);
    rightEye = cv::Point(-1,-1);
    if(rightEyeRect.size()>0)
    {
        rightEye.x = rightEyeRect[0].x + rightEyeRect[0].width/2 + rightX;
        rightEye.y = rightEyeRect[0].y + rightEyeRect[0].height/2 + topY;
        cv::namedWindow("deteccion ojos derecho");
        cv::imshow("deteccion ojos derecho",topRightOfFace);
    }
}


void MainWindow::on_actionTransformacion_Geometrica_triggered()
{
    if (leftEye.x >= 0 && rightEye.x >= 0)
    {
        cv::Point2f eyesCenter;
        eyesCenter.x = (leftEye.x + rightEye.x) * 0.5f;
        eyesCenter.y = (leftEye.y + rightEye.y) * 0.5f;
        double dy = (rightEye.y - leftEye.y);
        double dx = (rightEye.x - leftEye.x);
        double len = sqrt(dx*dx + dy*dy);
        double angle = atan2(dy, dx) * 180.0/CV_PI;
        const double DESIRED_LEFT_EYE_X = 0.16;
        const double DESIRED_RIGHT_EYE_X = (1.0f - 0.16);
        const int DESIRED_FACE_WIDTH = 70;
        const int DESIRED_FACE_HEIGHT = 70;
        double desiredLen = (DESIRED_RIGHT_EYE_X - 0.16);
        double scale = desiredLen * DESIRED_FACE_WIDTH / len;
        cv::Mat rot_mat = getRotationMatrix2D(eyesCenter, angle, scale);
        double ex = DESIRED_FACE_WIDTH * 0.5f - eyesCenter.x;
        double ey = DESIRED_FACE_HEIGHT * DESIRED_LEFT_EYE_X - eyesCenter.y;
        rot_mat.at<double>(0, 2) += ex;
        rot_mat.at<double>(1, 2) += ey;
        warped = cv::Mat(DESIRED_FACE_HEIGHT, DESIRED_FACE_WIDTH, CV_8U,
cv::Scalar(128));
```

```
        cv::warpAffine(image, warped, rot_mat, warped.size());
        cv::namedWindow("warped");
        cv::imshow("warped", warped);
    }
}


void MainWindow::on_actionEcualizacion_Separada_triggered()
{
    int w = warped.cols;
    int h = warped.rows;
    cv::cvtColor(warped, warped,CV_RGB2GRAY);
    cv::Mat wholeFace;
    cv::equalizeHist(warped, wholeFace);
    int midX = w/2;
    cv::Mat leftSide = warped(cv::Rect(0,0, midX, h));
    cv::Mat rightSide = warped(cv::Rect(midX, 0, w-midX,h));
    cv::equalizeHist(leftSide, leftSide);
    cv::equalizeHist(rightSide, rightSide);
    cv::namedWindow("toda");
    cv::imshow("toda", wholeFace);
    cv::namedWindow("izquierda");
    cv::imshow("izquierda", leftSide);
    cv::namedWindow("derecha");
    cv::imshow("derecha", rightSide);
    for (int y=0; y<h; y++) {
        for (int x=0; x<w; x++) {
        int v;
        if (x < w/4) {
            v = leftSide.at<uchar>(y,x);
        }
        else if (x < w*2/4) {
            int lv = leftSide.at<uchar>(y,x);
            int wv = wholeFace.at<uchar>(y,x);
            float f = (x - w*1/4) / (float)(w/4);
            v = cvRound((1.0f - f) * lv + (f) * wv);
        }
        else if (x < w*3/4) {
            int rv = rightSide.at<uchar>(y,x-midX);
            int wv = wholeFace.at<uchar>(y,x);
            // mezcal el lado derecho
            float f = (x - w*2/4) / (float)(w/4);
            v = cvRound((1.0f - f) * wv + (f) * rv);
        }
        else {
            v = rightSide.at<uchar>(y,x-midX);
        }
            warped.at<uchar>(y,x) = v;
        }// end x loop
    }//end y loop
    cv::namedWindow("toda_");
    cv::imshow("toda_", warped);
}


void MainWindow::on_actionSuavizamiento_triggered()
{
    filtered = cv::Mat(warped.size(), CV_8U);
    cv::bilateralFilter(warped, filtered, 0 , 20.0, 2.0);
    cv::namedWindow("filtrada");
    cv::imshow("filtrada", filtered);
}

void MainWindow::on_actionMascara_Eliptica_triggered()
{
    cv::Mat mask = cv::Mat(filtered.size(),CV_8UC1, cv::Scalar(255));
    double dw = 70.0;
    double dh = 70.0;
    cv::Point faceCenter = cv::Point(cvRound(dw*0.5), cvRound(dh*0.4));
    cv::Size size = cv::Size(cvRound(dw*0.5), cvRound(dh*0.8));
    cv::ellipse(mask,faceCenter,size,0,0,360, cv::Scalar(0), CV_FILLED);
    filtered.setTo(cv::Scalar(128), mask);
    cv::namedWindow("con mascara");
    cv::imshow("con mascara", filtered);
    cv::imwrite("preprocesada.bmp", filtered);

}
```

```
double MainWindow::errorCuadrado(const cv::Mat A, const cv::Mat B)
{
    double errorCuadrado = cv::norm(A, B, CV_L2);
    double similaridad = errorCuadrado/(double)(A.rows*A.cols);
    return similaridad;
}


void MainWindow::detectarOjos(cv::Mat & image, cv::Point & leftEye, cv::Point &
rightEye)
{
    float EYE_SX = 0.16, EYE_SY=0.26, EYE_SW=0.30, EYE_SH=0.28;
    cv::Size minFeatureSize(20,20);
    int leftX = cvRound(image.cols*EYE_SX);
    int topY = cvRound(image.rows*EYE_SY);
    int widthX = cvRound(image.cols * EYE_SW);
    int heightY = cvRound(image.rows*EYE_SH);
    int rightX = cvRound(image.cols*(1.0-EYE_SX-EYE_SW));
    cv::Rect izquierdo = cv::Rect(leftX, topY, widthX, heightY);
    cv::Rect derecho = cv::Rect(rightX, topY, widthX, heightY);
    cv::Mat topLeftOfFace = image(izquierdo);
    cv::Mat topRightOfFace = image(derecho);
    cv::namedWindow("deteccion ojos");
    cv::imshow("deteccion ojos",image);
    cv::CascadeClassifier eyeDetector1("haar_xml/haarcascade_eye.xml");
    cv::CascadeClassifier eyeDetector2("haarcascade_eye_tree_eyeglasses.xml");
    std::vector<cv::Rect> leftEyeRect;//almacena el ojo detectado izquierdo
eyeDetector1.detectMultiScale(topLeftOfFace,leftEyeRect,1.1F,4,cv::CASCADE_SCALE_IMAGE,
minFeatureSize);
    leftEye = cv::Point(-1,-1);
    if(leftEyeRect.size()>0)    {
        leftEye.x = leftEyeRect[0].x + leftEyeRect[0].width/2 + leftX;
        leftEye.y = leftEyeRect[0].y + leftEyeRect[0].height/2 + topY;
        cv::namedWindow("deteccion ojos izquierdo");
        cv::imshow("deteccion ojos izquierdo",topLeftOfFace);
    }
    std::vector<cv::Rect> rightEyeRect;
eyeDetector1.detectMultiScale(topRightOfFace,rightEyeRect,1.1F,4,cv::CASCADE_SCALE_IMAGE
,minFeatureSize);
    rightEye = cv::Point(-1,-1);
    if(rightEyeRect.size()>0)
    {
        rightEye.x = rightEyeRect[0].x + rightEyeRect[0].width/2 + rightX;
        rightEye.y = rightEyeRect[0].y + rightEyeRect[0].height/2 + topY;
        cv::namedWindow("deteccion ojos derecho");
        cv::imshow("deteccion ojos derecho",topRightOfFace);
    }
}


void MainWindow::automaticFaceRecognition(cv::Mat & image)
{
    int prediccion;
    prediccion = model->predict(image);
    QString respuesta = QString("pertenece a la clase %").arg(prediccion);
    msgBox.setText(respuesta);
    msgBox.exec();
}


void MainWindow::trainingPCA()
{
    model = cv::createEigenFaceRecognizer();
    if(model.empty())
    {
        msgBox.setText("El algoritmo eigen face no esta ");
        msgBox.exec();
    }
    else
    {
        std::vector<cv::Mat> images;
        std::vector<int> labels;
        cv::Mat inputfaces;
        //leemos y almacenamos en un arreglo de imagenes
        for(int i = 1; i<20; i++)
```

```
        {
            for(int j = 1; j<5; j++)
            {
                QString filename =
"img_training/train/"+QString().number(i)+"/"+QString().number(j)+".bmp";
                inputfaces =
cv::imread(filename.toAscii().data(),CV_LOAD_IMAGE_GRAYSCALE);
                images.push_back(inputfaces);
            }
        }
        for(int i = 1; i<20; i++)
        {
            for(int j = 1; j<5; j++)
            {
                labels.push_back(i);
            }
        }
        model->train(images, labels);
        cv::Mat averageFace = model->get<cv::Mat>("mean");
        averageFace = getImageFrom1DFloatMat(averageFace,70);
        cv::namedWindow("averageFace");
        cv::imshow("averageFace",averageFace);
        QString valor;
        cv::Mat eigenValues = model->get<cv::Mat>("eigenvalues");
        for(int i=0; i< 20; i++)
        {
            valor = valor + QString().number(eigenValues.at<double>(i,0))+"\n";
        }
        msgBox.setText(valor);
        msgBox.exec();
    }
}


void MainWindow::on_actionReal_triggered()
{
    anchoDeteccion = 320;
    flags = cv::CASCADE_SCALE_IMAGE;
    cv::Size minFeatureSize(20,20);
    float searchScaleFactor = 1.1F;
    int minNeighbors = 4;
    flags = cv::CASCADE_SCALE_IMAGE;
    minFeatureSize.height = 20;
    minFeatureSize.width = 20;
    trainingPCA();
    tRealVideo.open(0);
    if(!tRealVideo.isOpened())
    {
        msgBox.setText("Intente nuevamente");
        msgBox.exec();
    }
    else
    {
        try
        {
            faceDetector.load("haar_xml/haarcascade_frontalface_default.xml");
        }
        catch(cv::Exception e)
        {            }
        if ( faceDetector.empty() ) {
            msgBox.setText("No se cargo, el clasificador en cascada ");
            msgBox.exec();
        }
        double oldTime;
        oldTime = cv::getTickCount();
        int j=1;
        for(;;)
        {
            tRealVideo>> frame;
            filas = frame.rows;
            columnas = frame.cols;
            QString mensaje;
            mensaje = "Dimension imagen original:"+
QString().setNum(filas)+"x"+QString().setNum(columnas);
            ui->statusBar->showMessage(mensaje);
            float scale = columnas/(float) anchoDeteccion;
            if(columnas>anchoDeteccion)
```

```
            {
                int escalaAltura = cvRound(filas/scale);
                cv::resize(frame, smallImg, cv::Size(anchoDeteccion, escalaAltura));
                cv::cvtColor(smallImg,gray ,CV_BGR2GRAY);
                cv::equalizeHist(gray, imgTReal);
                std::vector<cv::Rect> faces;
faceDetector.detectMultiScale(imgTReal,faces,searchScaleFactor,minNeighbors, flags,
minFeatureSize);
                if(faces.size()>0)
                {
                    for(int i = 0; i<(int)faces.size();i++)
                    {
                        faces[i].x = cvRound(faces[i].x*scale);
                        faces[i].y = cvRound(faces[i].y*scale);
                        faces[i].width = cvRound(faces[i].width*scale);
                        faces[i].height = cvRound(faces[i].height*scale);
                        if (faces[i].x < 0)
                            faces[i].x = 0;
                        if (faces[i].y < 0)
                            faces[i].y = 0;
                        if (faces[i].x + faces[i].width > columnas)
                            faces[i].x = columnas - faces[i].width;
                        if (faces[i].y + faces[i].height > filas)
                            faces[i].y = filas - faces[i].height;
                        cv::rectangle(frame,faces[i],CV_RGB(0,255,0), 1);
                        QString nombreImagen = QString("img_test/rostro_%1.bmp").arg(i);
                        faceDetected= frame(faces[i]);
                        cv::imwrite(nombreImagen.toAscii().data(), faceDetected);
                        if((cv::getTickCount()-oldTime)/cv::getTickFrequency() >= 1)
                        {
                            nombreImagen = "img_test/" + QString().setNum(j) +
QString("g_rostro_%1.bmp").arg(i);
                            cv::cvtColor(faceDetected, faceDetected, CV_RGB2GRAY);
                            cv::imwrite(nombreImagen.toAscii().data(), faceDetected);
                            detectarOjos(faceDetected, leftEye, rightEye);
                            if(transGeometrica(faceDetected, warped, leftEye, rightEye))
                            {
                                ecualizacionSeparada(warped,false);
                                suavizamiento(warped, filtered);
                                cv::Mat mask = cv::Mat(filtered.size(),CV_8UC1,
cv::Scalar(255));
                                double dw = 70.0, dh = 70.0;
                                mascaraEliptica(mask, filtered, dw, dh);
                                automaticFaceRecognition(filtered);
                            }
                            j++;
                            oldTime = cv::getTickCount();
                        }
                    }
                }
                if(cv::waitKey(30)>=0)
                {
                    ui->actionTomar_Imagen->setEnabled(false);
                    cv::destroyWindow("reducida");
                    break;
                }
                cv::imshow("reducida", imgTReal);
                cv::imshow("Original", frame);  //imagen reducida a color
            }
            else
            {
                cv::cvtColor(frame,gray ,CV_BGR2GRAY);
                cv::equalizeHist(gray, imgTReal);
                QString mensaje;
                mensaje = "Dimension: "+
QString().setNum(filas)+"x"+QString().setNum(columnas);
                ui->statusBar->showMessage(mensaje);
                std::vector<cv::Rect> faces;
faceDetector.detectMultiScale(imgTReal,faces,searchScaleFactor,minNeighbors, flags,
minFeatureSize);
                if(faces.size()>0)
                {
                    for(int i = 0; i<(int)faces.size(); i++)
                    {
                        QString ruta = QString("img_test/rostro%1.bmp").arg(i);
                        faceDetected = frame(faces[i]);
                        cv::rectangle(frame, faces[i], CV_RGB(0,255,0), 1);
```

```
                        cv::imwrite(ruta.toAscii().data(),faceDetected);
                    }
                }

                cv::imshow("Tiempo Real", frame);
                if(cv::waitKey(30)>=0)
                {
                    ui->actionTomar_Imagen->setEnabled(false);
                    cv::destroyWindow("Tiempo Real");
                    break;
                }
            }
        }
    }
}


void MainWindow::matToLine(cv::Mat &img)
{
    cv::SVMParams params;
    params.svm_type = CvSVM::C_SVC;
    params.kernel_type = CvSVM::POLY;
    params.gamma = 3;

    int columnas = 70*70;
    int filas = 19*4;
    cv::Mat training_mat(filas,columnas,CV_32FC1);
    int columna =0;
    cv::Mat labels;

    cv::Mat inputfaces;
    //obtener las imagenes de rostros
    //leemos las imagenesdesde archivo y almacenamos el vector fila en training_mat
    for(int i = 1; i<20; i++)
    {
        for(int j = 1; j<5; j++)
        {
            QString filename =
"img_training/train/"+QString().number(i)+"/"+QString().number(j)+".bmp";
            inputfaces = cv::imread(filename.toAscii().data(),CV_LOAD_IMAGE_GRAYSCALE);
            //convertimos la imagen en filas
            for(int k =0; k<70; k++)
            {
                for(int l=0;l<70; l++)
                {
                    training_mat.at<float>(j,columna++)=inputfaces.at<uchar>(k,l);
                }
            }
        }
    }
    int fila = 0;
    for(int i = 1; i<20;i++)
    {
        for(int j=1;j<5;j++)
        {
            labels.at<uchar>(fila, 0)=i;
            fila++;
        }
    }

    cv::SVM(training_mat, labels, cv::Mat(), cv::Mat(), params);
}

void MainWindow::PredicSVM(cv::Mat &img)
{
    int clase = svm.predict(img_mat_1d);
    msgBox.setText(clase);
    msgBox.exec();
}
```