PONTIFICIA UNIVERSIDAD CATÓLICA DE VALPARAÍSO
FACULTAD DE INGENIERÍA
ESCUELA DE INGENIERÍA INFORMÁTICA

Reconocimiento Facial

David Eduardo Espinoza Olguín
Peter Ignacio Jorquera Guillen

Profesor Guía: **Claudio Cubillos Figueroa**

Profesor Co-referente: Rafael Mellado

Carrera: **Ingeniería de Ejecución en Informática**

Junio 2015

Dedicatorias

Dedico este proyecto especialmente a mi familia, ya que con su apoyo en todo ámbito pude concluir la carrera de la mejor manera. A mi padre José Jorquera, por facilitarme los recursos necesarios para que mi vida universitaria fuera lo más amena posible depositando toda su confianza en mí y en mis capacidades. A mi madre Gloria Guillén, por aconsejarme en los momentos difíciles durante los primeros años de Universidad, y a que con ellos pude seguir en la misma carrera. A mi pareja Nicole Oyarzun, la cual fue el apoyo fundamental durante este largo camino, mediante su apoyo incondicional, su ánimo día a día y toda su disposición para ayudarme en todo momento. Y en general a toda las personas que apoyaron en todo momento ya sea familiares, como también los profesores que guiaron nuestra tesis hasta el final. Gracias a todos.

Peter Jorquera Guillen

Dedico este proyecto a todas las personas que estuvieron presente en mi vida. Primero a mi padre David Espinoza y a mi madre Angela Olguín por darme las herramientas en este proceso y dejarme decidir quien quería ser. A mis hermanos por las experiencias vividas y confiar en mí. A mis amigos de la infancia, sin duda son el mejor recuerdo. A mis amigos del liceo por ser personas admirables e incondicionales conmigo. A los profesores que me enseñaron cosas útiles para la vida y el ámbito profesional. A mis amigos de Universidad y a los que me acompañaron en mi primera experiencia laboral, aprendí mucho de ustedes. Por ultimo a mi pareja Natalia Retamales, por su apoyo en los momentos más difíciles, y hacer de este proceso mucho más divertido, eres la mejor, por otro lado a mi hija Ailén Espinoza que no quiso quedar afuera de esta dedicatoria, alegras mi vida todos los días y espero ser mejor persona para ti. Agradecido de todos que hicieron posible este momento.

David Espinoza Olguín

Índice

Resumen

En el siguiente proyecto realizado por estudiantes de la Pontificia Universidad Católica de Valparaíso, se estudiara, analizara e implementará un sistema de reconocimiento facial el cual será integrado con librerías de OpenCV/EmguCV para integrar un trabajo más eficiente y exacto. Esto se realizara con el fin de dar un comienzo hacia un proyecto más avanzado y que más adelante sea en línea.

El sistema dispondrá de una base de datos local la cual permitirá almacenar las diferentes fotografías ya entrenadas sacadas al usuario para que en el paso siguiente se pueda reconocer ya el rostro de dicha persona, con esto podremos analizar que algoritmo es más eficiente en trabajar con el reconocimiento y así realizar estudios sobre ellos.

Palabras claves: Reconocimiento facial, librerías OpenCV/EmguCV.

Abstract

In the next project accomplished by students of de Pontifica Universidad Catolica de Valparaiso, it will study, analyze and implement a system of face recognition which will be integrated with libraries of OpenCV/EmguCV to integrate a more efficient and accurate work. This will be perform in order to give a start to a more advanced project and later maybe will be online.

The system will have a local database that will allow to store the different images already trained taken from the user so that in the next step can recognize the face of that person, with this we will analyze which algorithm will be more efficient to work with the Recognition and thus do different studies about them.

Keywords: Face recognition, OpenCV/EmguCV libraries.

Lista de figuras

Lista de Tablas

1 Presentación del tema

1.1 Introducción

Los seres humanos desde tiempos remotos hasta la actualidad se han visto en la obligación de reconocerse unos a otros mediante nombres, apodos, etc., pero es el rostro el que nos da una propia identidad a cada persona ya que por medio de estudios la cara es una de las cosas que imposible de olvidar, es por esto que, por medio de las nuevas tecnologías y algoritmos, se han implementado varias funcionalidades propias en respecto a esta identidad.

En los inicios de esta tecnología llamada "Reconocimiento facial" se usaba algoritmos de reconocimientos muy simples el cual daba mayor oportunidad a que los errores se produjeran, ya que al ser así el mismo reconocimiento se podría dar para 2 personas diferentes. En la actualidad y con los avances logrados, además de los algoritmos que han sido exponencialmente mejorados, los errores son mínimos ya que se han afinado la forma en cómo se reconoce cada rostro.

A continuación se explica cómo funciona cada uno de estos algoritmos además del funcionamiento y cada etapa en un reconocimiento facial, como se muestra en la figura 1.1 consta de varias secciones las cuales en complemento darán un buen funcionamiento del software que se implementará.

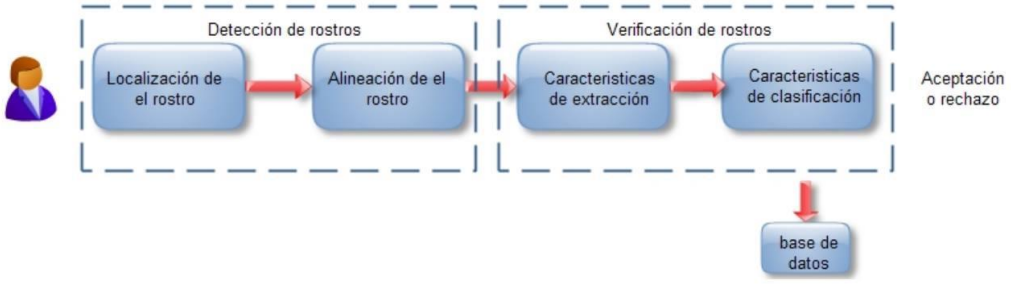

Figura 1.1 Etapas de un reconocimiento facial **[1]**.

1.1.1 Problema

En la actualidad el problema más grande referido al Reconocimiento Facial es la disponibilidad de este software como también el costo de implementarlo y que funcione de manera óptima, confiable y segura. Esto no es un impedimento para países desarrollados como Estados Unidos o Rusia los cuales invierten millones de dólares en implementar esta herramienta en sus empresas para que así la autenticación del usuario en algún sistema sea lo más segura posible.

En Chile la herramienta biométrica más utilizada es la huella dactilar ya que es menos costosa de implementar, fácil de utilizar y con un marco de error mínimo, pero a su vez se han encontrado métodos para clonar la mano completa de una persona cosa que con el Reconocimiento Facial no sucedería ya que clonar un rostro es mucho más complicado que una mano. Lo que limita implementar un software de Reconocimiento Facial en nuestro país es su alto valor.

Deben ser pocas las empresas en este país que utilicen el Reconocimiento Facial para autenticar a algún usuario siendo que esta herramienta es una de las más segura y con los años se ha ido disminuyendo la taza de error a través de nuevos métodos implementados en las técnicas y algoritmos.

1.1.2 Solución

Son por estos problemas que se decidió dar el pie para empezar a desarrollar un Software de reconocimiento facial, el cual en su primera versión tendrá las funcionalidades de detectar el rostro y verificarlo con la base de datos para así confirmar la identidad de la persona.

Todo esto se realizara en una interfaz sencilla en un comienzo para que así facilitar su uso a los usuarios que lo utilicen por primera vez e integrando los algoritmos más utilizados y eficientes que se conozcan, y que se hayan implementado en otros proyectos de la misma naturaleza.

La solución más significativa que se podrá dar en un comienzo es referida a la seguridad ya que a partir de este programa se podrá ir perfeccionando una y otra vez hasta que se complemente del tal manera que sea seguro de usar, además de ser eficiente en lo que respecta a la identificación y reconocimiento de rostro y con esto hacer que más empresas la introduzcan en sus sistemas ya sea para un control de acceso o solo para hacer pruebas de algún tipo.

1.2 Objetivo General

Implementación de un software de Reconocimiento Facial por medio de un lenguaje de programación utilizando librerías de Open CV.

1.2.1 Objetivos específicos

- Conocer y comprender las ventajas de utilizar Open CV en el ámbito de reconocimiento de patrones y entender sus librerías para aplicarlas en el uso del reconocimiento facial.
- Realizar estudios en torno a los distintos algoritmos de reconocimiento facial e indicar cuál de éstos funcionar de mejor manera.
- Validar la detección de rostro, el guardado de imágenes y el reconocimiento de la imagen de entrada desarrollar un software de reconocimiento facial.

1.3 Plan de trabajo

1.3.1 Carta Gantt

A continuación se muestra como se organizó el tiempo de cada una de las etapas del proyecto correspondientes al segundo semestre del año actual, mediante dos tipos de carta Gantt.

1	Reconocimiento Facial	02-03-2015	19-06-2015	109
1.1	**Analisis de los Algoritmos**	02-03-2015	12-03-2015	10
1.1.1	Construccion del informe	02-03-2015	12-03-2015	10
1.1.2	Analisis del codigo	05-03-2015	08-03-2015	3
1.1.3	Estudio de Algoritmos	06-03-2015	10-03-2015	4
1.1.4	Comparacion de algoritmos	10-03-2015	12-03-2015	2
1.2	**Diseño del Proyecto**	13-03-2015	25-03-2015	12
1.2.1	Diseño de Interfaz	13-03-2015	16-03-2015	3
1.2.2	Diseño de menú y configuraciones	15-03-2015	18-03-2015	3
1.2.3	Diseño de prototipo	14-03-2015	20-03-2015	6
1.3	**Diseño de la Solucion**	26-03-2015	10-04-2015	15
1.3.1	Casos de Uso	26-03-2015	30-03-2015	4
1.3.2	Diagrama de Secuencia	27-03-2015	29-03-2015	2
1.3.3	Estudio de Bases de Datos (Imagenes)	30-03-2015	05-04-2015	6
1.3.5	Entrenamiento de imágenes	08-04-2015	10-04-2015	2
1.3.4	Diseño de Pruebas	04-04-2015	10-04-2015	6
1.4	**Desarrollo de Software Final**	11-04-2015	12-06-2015	62
1.4.1	Definicion de las Tecnicas	11-04-2015	13-04-2015	2
1.4.2	Definicion de Bases de Datos	14-04-2015	18-04-2015	4
1.4.3	Implementación de algoritmos faltantes	20-04-2015	31-05-2015	41
1.4.4	Mejoras de la Interfaz	31-05-2015	05-06-2015	5
1.4.5	Entrega de código fuente	12-06-2015	13-06-2015	1
1.4.6	Pruebas de la Aplicación	30-04-2015	11-06-2015	42
1.5	**Conclusion y Resultados**	08-06-2015	19-06-2015	11

Figura 1.2 Carta Gantt

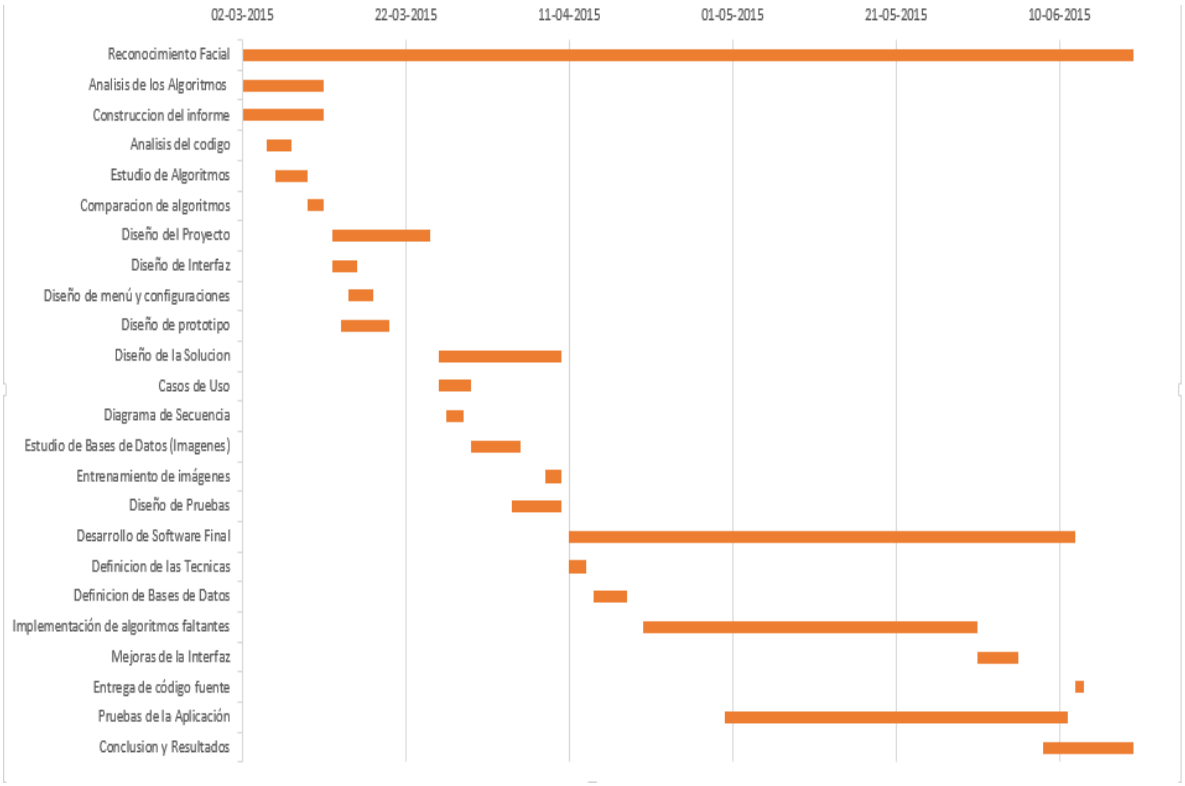

Figura 1.3 Grafico del plan de trabajo

1.4 Metodología

La metodología que se utilizará para el desarrollo del software será la del modelo iterativo incremental (figura 1.4), debido a que utiliza las ventajas del modelo en cascada y el modelo de desarrollo evolutivo, este modelo se adapta de forma más eficiente para el desarrollo de este software en comparación con otros modelos ya que está basado en la filosofía de desarrollo en incrementos, en la que cada uno (incremento) proporciona un subconjunto de funcionalidades requeridas por el cliente.

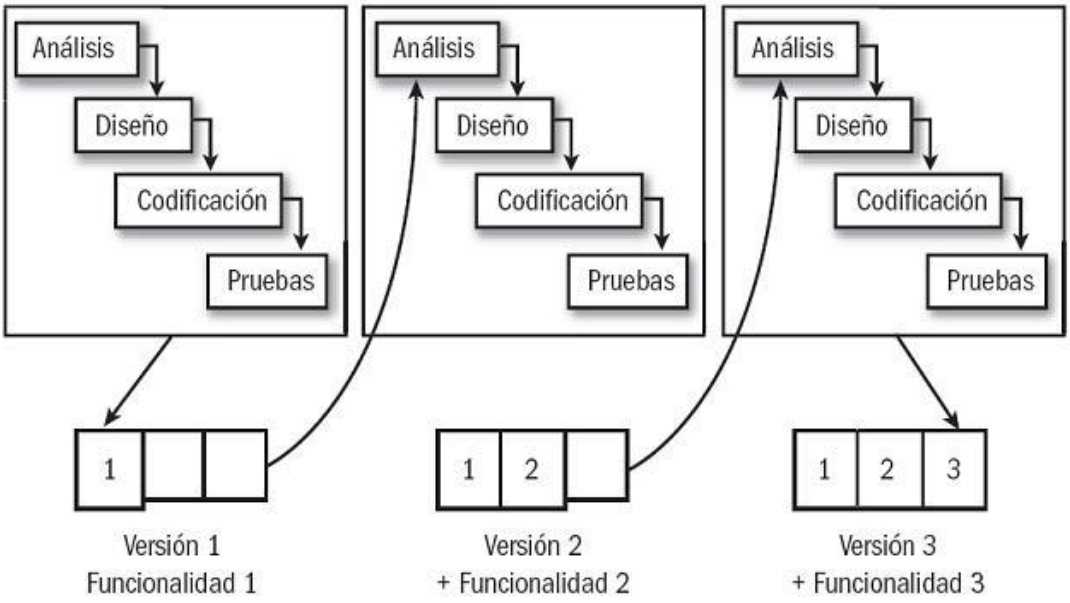

Figura 1.4 Metodología utilizada para el desarrollo del software **[2]**.

1.5 Análisis de Riesgo

1.5.1 Riesgos

- **Riesgos de costo**
  - Sobrepasar los costos de desarrollo previstos.
  - Cambios en el alcance y los requerimientos de la parte del cliente.
  - Mala estimación de los costos durante la fase de inicialización.
  - Sobre-estimación de los costos de desarrollos previstos.

- **Riesgos de calendario**
  - Mala estimación del tiempo necesario.
  - Mala asignación de recursos.
  - Pérdida de recursos humanos no prevista.
- **Riesgos tecnológicos**
  - Usar herramientas mal adaptadas.
  - Usar herramientas no aprobadas o con fallas.
  - Problemas de hardware/software.
  - Problemas de integración de las diferentes partes del proyecto desarrolladas en paralelo.
- **Riesgos operacionales**
  - Falta de liderazgo en el equipo.
  - Falta de comunicación.
  - Falta de motivación del equipo.
- **Riesgos externos**
  - Desastres naturales (fuego, inundación, terremoto, entre otros).
  - Enfermedades.
  - Accidentes de los integrantes del grupo.

1.5.2 Plan de Mitigación

- **Riesgos de calendario**
  - Al sobrepasar el tiempo estimado, se buscara reajustar el calendario para las nuevas tareas a realizar; se buscara aumentar la rapidez de producción, para así recuperar el tiempo perdido.
  - Realizar un buen análisis previo.
  - En el caso de atrasos con tareas específicas, se debería darle prioridades a las tareas más importantes en el desarrollo del software.
- **Riesgos tecnológicos**
  - Llevar a cabo un buen análisis de los requerimientos técnicos de lo que se necesita hacer y realizar una buena investigación de las tecnologías a utilizar.
  - Procurar mantener una buena documentación y buen orden mientras se avanza en el proyecto.

- **Riesgos externos**
  - Manejar dentro del presupuesto el uso de respaldo de información en caso de cualquier accidente del lugar físico.
  - Contratar seguros.
  - Cuidarse de las bajas temperaturas.

1.5.3 Plan de Mitigación

- **Riesgos de calendario**
  - Desarrollar versiones betas del programa.
  - Cambiar la asignación delos recursos según prioridad.
  - Investigar e ir adquiriendo conocimiento para mayor rapidez de las tareas planteadas.
- **Riesgos tecnológicos**
  - Corrección en el trabajo existente y cambio a una tecnología más adecuada que este dentro del presupuesto.
  - Pago de multas en caso de que la herramienta haya sido usada de una mala manera o corrección en las aplicaciones existentes.
  - Estudio y corrección de fallas en la sinergia del proyecto. Hacer uso de documentación.
- **Riesgos externos**
  - Hacer uso de respaldos, protecciones existentes y manejar la opción de ir cambiando el tiempo de las tareas según se vaya avanzando en el proyecto.

2 Reconocimiento Facial

2.1 Descripción del proyecto

El reconocimiento facial es una herramienta que nos permite identificar a una persona automáticamente por medio de una imagen digital. Es una forma de seguridad biométrica que se ha desarrollado crecientemente desde las primeras pruebas en 1995 , en la actualidad los algoritmos poseen hasta 100% más de exactitud que en aquella época en un comienzo solo se utilizaban imágenes en 2D las cuales solos permite ver el rostro de la persona a la cual queremos identificar , pero ha habido actualizaciones de esta herramienta que permiten , en este momento y aun en desarrollo , verlas en 3D , esta dará una mayor precisión y eficacia al momento de que la imagen de la persona coincida con la imagen que esta almacenada en la base de datos.

Para realizar un reconocimiento facial se deben analizar las características faciales de la persona, las cuales se pueden extraer ya sea de una fotografía o desde un fotograma en una fuente de video, esta se convierte en una plantilla y luego se compara con las imágenes en una base de datos para verificar la identidad de la persona.

El objetivo de este software es el de encontrar, dada una cara desconocida, una conocida que posea las mismas características.

Existen dos formas de operar:

1. Verificación o autentificación de caras: En donde se compara una imagen del rostro de una persona con otra imagen a la cual se le quiere hacer coincidir. El software aceptara si las imágenes coinciden o las rechazara si no.
2. Identificación o reconocimiento: Se compara una imagen de una persona desconocida con la base de datos para encontrar la identidad de la persona.

2.2 Detección de rostros

La detección facial es una tecnología de visión computarizada que determina el lugar y el tamaño de rostros humanos en imágenes o videos. Es uno de los subtipos de detección de objeto/clase, cuya tarea es encontrar la localización y el tamaño de los objetos en una imagen perteneciente a la clase dada **[3]**.

Mientras que para el ser humano es trivial esta tarea, una computadora presenta dificultades debido a diversos factores como: variabilidad en la posición del rostro, presencia o ausencia de componentes estructurales (ejemplo: bigote), expresión facial, oclusión (uso de lentes, gorro, etc.) y condiciones del ambiente.

Detectar el rostro humano es el primer paso en un sistema de reconocimiento facial, sin embargo influye de manera significante en el resultado del proceso, ya sea dado un conjunto de imágenes o video en tiempo real. Por lo tanto, debe ser capaz de identificar los rostros independientemente de los factores que anteriormente se mencionaron.

Los métodos de detección facial se dividen en cuatro categorías (no excluyentes):

- Métodos basados en conocimiento: Codifican el conocimiento humano mediante distancias y posiciones entre las características humanas (ojos, nariz, labios).
- Métodos basados en características invariantes: Las características invariantes son aquellas que no se modifican a eventuales cambio de luz, pose o ubicación de la cámara, tales como la ceja, nariz, textura de la piel y línea de pelo. Este método

funciona detectando uno de estos componentes, construyendo un modelo estadístico y con los resultados, verificar la existencia de un rostro.

- Métodos basados en moldes (patrones): Es la relación entre una imagen de entrada y un patrón o molde previamente definido, cuyo objetivo es capturar características del rostro.
- Métodos basados en apariencia: Utilizan modelos obtenidos mediante entrenamiento de imágenes, tomando la imagen como un vector de características, es decir, es visto como una variable aleatoria. A diferencia de los métodos basados en moldes, donde el patrón es definido por un "experto", los patrones en este modelo son determinados por el aprendizaje obtenido en el entrenamiento de imágenes.

Existen varios métodos para detectar rostros, sin embargo el algoritmo de Viola-Jones, es el más eficaz, obteniendo un mayor porcentaje de aciertos respecto a sus pares, además de más rapidez. Este algoritmo integra un nuevo concepto, la imagen integral, que junto con el algoritmo de boost como método de entrenamiento, forman un clasificador complejo y preciso **[4]**.

2.3 Detector de caras: Algoritmo de Viola Jones

Paul Viola y Michael Jones desarrollaron este algoritmo en 2001. Este sistema de detección de rostros representa un gran avance debido a su rapidez para identificar caras humanas, ya que realiza la clasificación mediante características extraídas en una escala de grises, a diferencia de sus predecesores que la realizaban pixel a pixel y en imágenes de color **[5]**.

En el método de Viola Jones se definen los siguientes conceptos:

- Integral de la Imagen, usada para la rápida detección de las características.
- Características de rectángulos, llamados características de Haar.
- AdaBoost, un método machine-learning, que consiste en reconocimiento de patrones y el estudio y construcción de algoritmos que aprenden y predicen el comportamiento en un conjunto de datos **[6]**.
- Una cascada de decisión para combinar características de manera eficiente.

2.3.1 Integral de la Imagen

Es una representación de la imagen original, permitiendo extraer las características de Haar. Esta imagen es obtenida a partir de un algoritmo que genera la suma de los valores de un rectángulo dentro de una imagen.

La integral de una imagen es una matriz de igual tamaño que la matriz de la imagen original, que en la posición x, y contiene la suma de los pixeles contenidos arriba y a la izquierda del punto x, y, tal como se define en la fórmula:

$$II(x, y) = \sum_{x' \leq x,\, y' \leq y} I(x', y') \quad \textbf{[4]}$$

Dónde:

- II(x, y) es la integral de la imagen.
- I (x', y') es la imagen original.

La integral de la imagen, en forma recursiva, es representada por:

$$S(x, y)=S(x, y-1)+I(x, y)$$

$$II(x, y)= II(x-1, y) + S(x, y)$$ **[4]**

Dónde:

- S(x, y) es la suma acumulada en fila.
- S(x, -1) = 0 y II (-1, y) =0.
-

2.3.2 Características de Haar

Son descriptores que permiten obtener información de una zona en particular mediante operaciones aritméticas, la principal razón para usar esto es que permite gran eficiencia de cálculo.

La extracción de características es realizada aplicando filtros con bases Haar. En el algoritmo de Viola-Jones se usan tres características de Haar:

- Característica de dos rectángulos: es la diferencia entre la suma de los pixeles de ambas regiones rectangulares.
- Característica de tres rectángulos: Es la suma de los pixeles de los rectángulos exteriores menos la suma del rectángulo central.
- Característica de cuatro rectángulos: Es la diferencia entre los pares diagonales de los rectángulos.

Un descriptor es representado por un rectángulo definido por su vértice, su altura, su longitud y sus pesos (negativo o positivo).

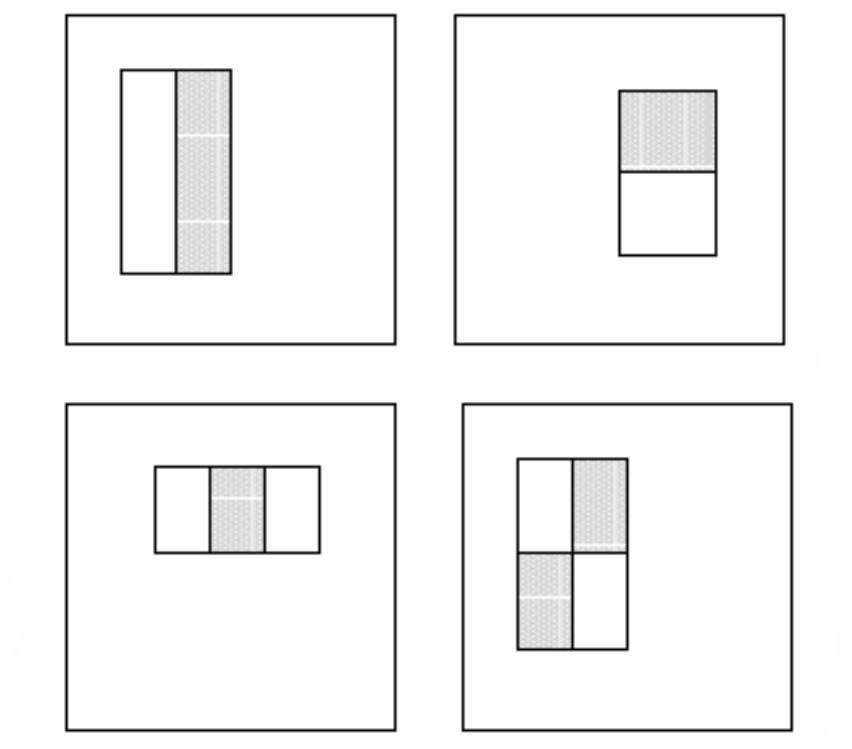

Figura 2.1 Características de Haar

2.3.3 AdaBoost

AdaBoost (AdaptiveBoost) es un algoritmo de aprendizaje máquina que consiste en la extracción de características por medio de clasificadores. Basado en la mejor forma de ejecutar los clasificadores para detectar de forma favorable el rostro humano.

En el algoritmo de Viola-Jones, *AdaBoost* elige un gran número de características de Haar, para seleccionar cuál de ellas se ajusta mejor para clasificar (clasificador) los distintos elementos, en este caso si es rostro o no. Para seleccionar las características y entrenar al clasificador, se combina una selección de funciones débiles de clasificación para formar un clasificador fuerte. Las funciones débiles están diseñadas para seleccionar una característica de Haar que separa de mejor forma los ejemplos positivos y negativos **[7]**.

Los pasos del algoritmo son:

- Se tienen las imágenes de ejemplo (X1, Y1),….., (Xn, Yn) donde Yi = 0,1 para ejemplos negativos y positivos respectivamente.
- Inicializar los pesos, dados por 1/2m y 1/2l donde m y l son el número de ejemplos positivos y negativos, respectivamente.
- Normalizar los pesos.
- Para cada característica, se evalúa el error con respecto a una ventana de 24x24 pixeles y ejemplos positivos.
- De esta forma se escoge un clasificador, con el menor error.
- Se actualizan los pesos.

- Después de algunas iteraciones, se obtiene un clasificador fuerte, que separa de manera eficiente ejemplos positivos de ejemplos negativos.

Para detectar bien el rostro, las características elegidas por *AdaBoost* son significativas y de fácil interpretación. La elección de la primera característica se basa en la propiedad basada en que la región de los ojos es más oscura que la región de la nariz y las mejillas. La segunda característica se basa en que los ojos son más oscuros que el puente de la nariz (comienzo de la nariz).

Figura 2.2 Las dos características de Haar más significativas seleccionadas por *AdaBoost*

2.3.4 Cascada de decisión

Un clasificador no es suficiente para detectar rostros de manera eficiente, por eso se implementa una cascada de clasificadores entrenados y ajustados **[8]**.

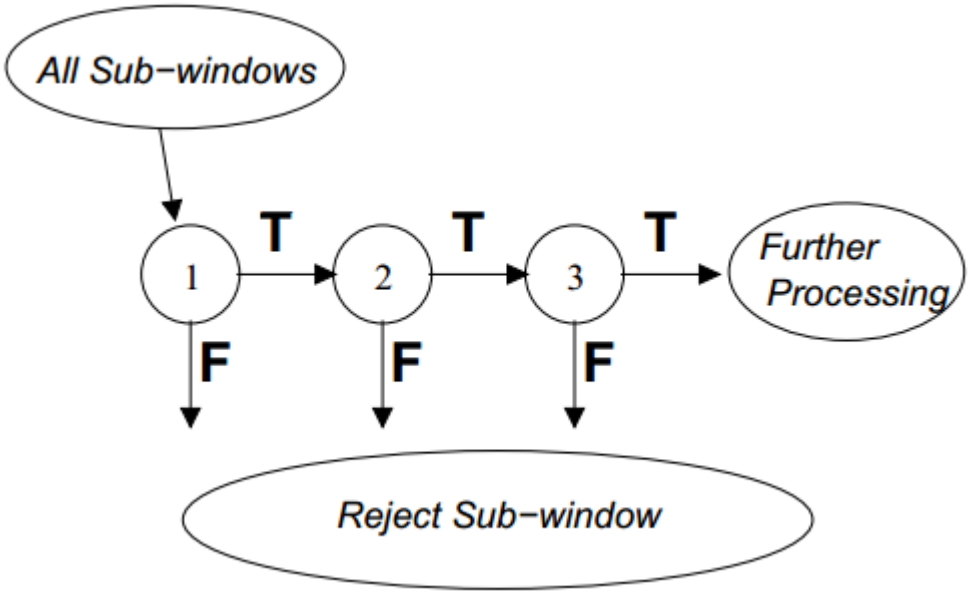

Figura 2.3 Esquema de Cascada de decisión, donde T= true (cara) y F = false (no cara) **[3]**

Esta cascada se compone de etapas donde cada una contiene un clasificador fuerte. Cada etapa determina si una sub-ventana es una cara o no. Cuando no es cara esa sub-ventana se descarta inmediatamente. Por otro lado, si es cara pasa a la siguiente etapa donde se realiza una tarea más compleja que la anterior. De esta forma, se deduce que mientras una sub-ventana pase por más etapas, mayor será la probabilidad que la sub-ventana contenga un rostro. Se añaden etapas hasta cumplir con la tasa de detección, determinada por un conjunto de validaciones.

2.4 Reconocimiento Facial

Este sistema es usado cuando la identidad del individuo es desconocida, y es necesario identificarlo para diversas tareas, tales como: inicio de sesión y acceso a base de datos. Se realiza una búsqueda en la base de datos con imágenes y si existe coincidencia, el individuo es identificado.

El objetivo de un sistema de reconocimiento facial es identificar automáticamente a una persona en una imagen o video en tiempo real, discriminando las señales de entrada (imágenes) en varias clases (personas).

Existen 2 tipos de técnicas de reconocimiento facial: técnicas basadas en apariencia y técnicas basadas en modelos **[4]**. En este proyecto, se tratarán métodos basados en apariencia lineales, cuyos principales algoritmos son: Eigenfaces (basado en PCA), Fisherfaces (basado en FLD) y Local Binary Patterns Histograms.

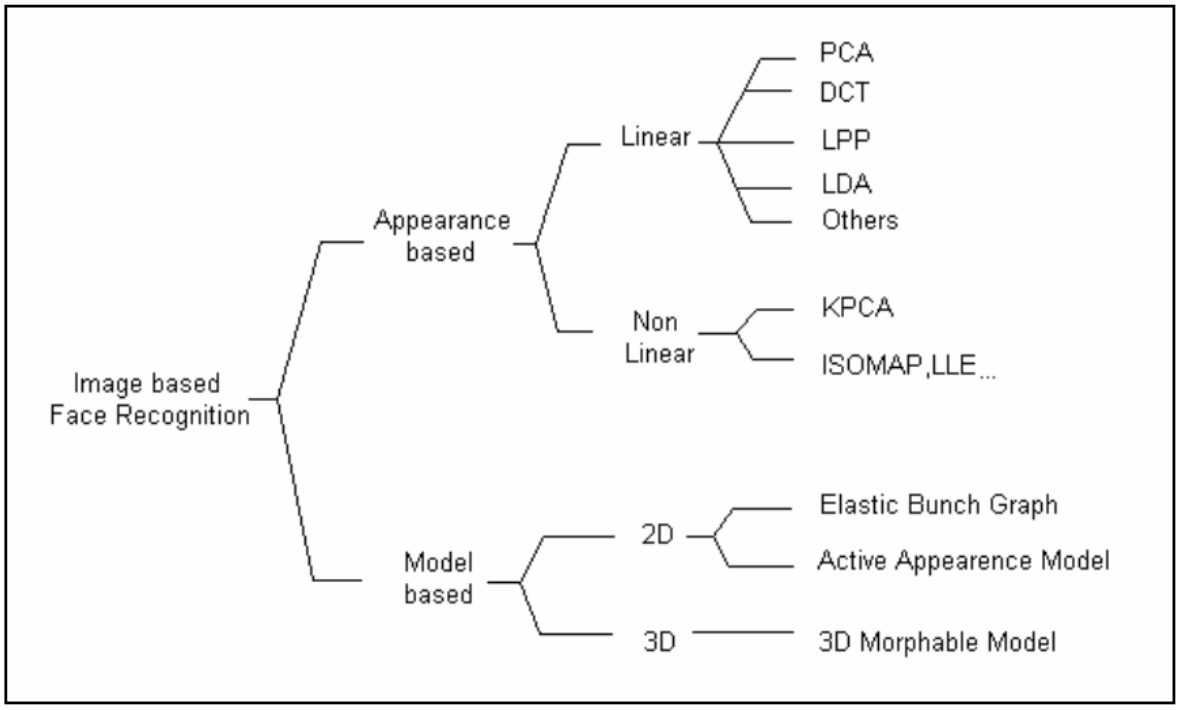

Figura 2.4 Métodos de Reconocimiento Facial **[9]**.

2.4.1 Eigenfaces

En una imagen de entrada existen *Principal Components* (Componentes Principales) o características comunes como: ojos, labio, nariz y distancias entre estos componentes, esos componentes principales son llamados eigenfaces **[10]**.

El algoritmo de reconocimiento de rostro Eigenfaces sigue los siguientes pasos:

El primer paso es poseer un conjunto de imágenes de entrenamiento de diferentes personas, compuesto en lo posible de subconjuntos de imágenes para cada persona que contengan diferentes posturas, condiciones de iluminación, etc. Este proceso es conocido como etapa de entrenamiento, donde las imágenes poseen el mismo tamaño.

Figura 2.5 Imágenes de Entrenamiento

Eigenfaces no trabaja directamente en las imágenes, primero las convierte en una matriz (vector). Es decir, una imagen de n x n pixeles (cada pixel posee un valor entre 0 y 255) es transformada en un vector de n^2 x 1. Se calcula un promedio con todos los vectores, llamado "vector promedio del rostro". Se resta cada vector con este promedio, obteniendo los vectores normalizados. Se calcula la matriz de covarianza para calcular los *eigenvectors* cuya matriz posee una dimensión de n^2 x n^2. Este cálculo requiere una gran cantidad de memoria para almacenar la matriz. La solución es aplicar una SVD (descomposición en valores singulares o *singular value descomposition*), reduciendo la dimensión de la matriz de n^2 x n^2 a M x M, donde M es el número de imágenes de entrenamiento. El SVD descompone una matriz en un producto de tres matrices, donde una de ellas posee los vectores propios (*eigenvectors*) y otra posee los valores propios (*eigenvalues*) **[11]**.

En este proceso se seleccionan los n vectores con los valores propios de mayor valor. Luego se calculan los vectores propios de la matriz de covarianza para encontrar las *eigenfaces*. Un nuevo rostro es formado con las *eigenfaces*. Se resta el vector promedio del rostro con la nueva imagen y se multiplica con cada vector donde fueron encontradas las *eigenfaces.* El objetivo es determinar qué imagen del conjunto de entrenamiento se parece más a la imagen de entrada, mediante una formula denominada la distancia Euclidiana. Aquella imagen que posea la menor distancia es considerada como el rostro de la imagen de entrada.

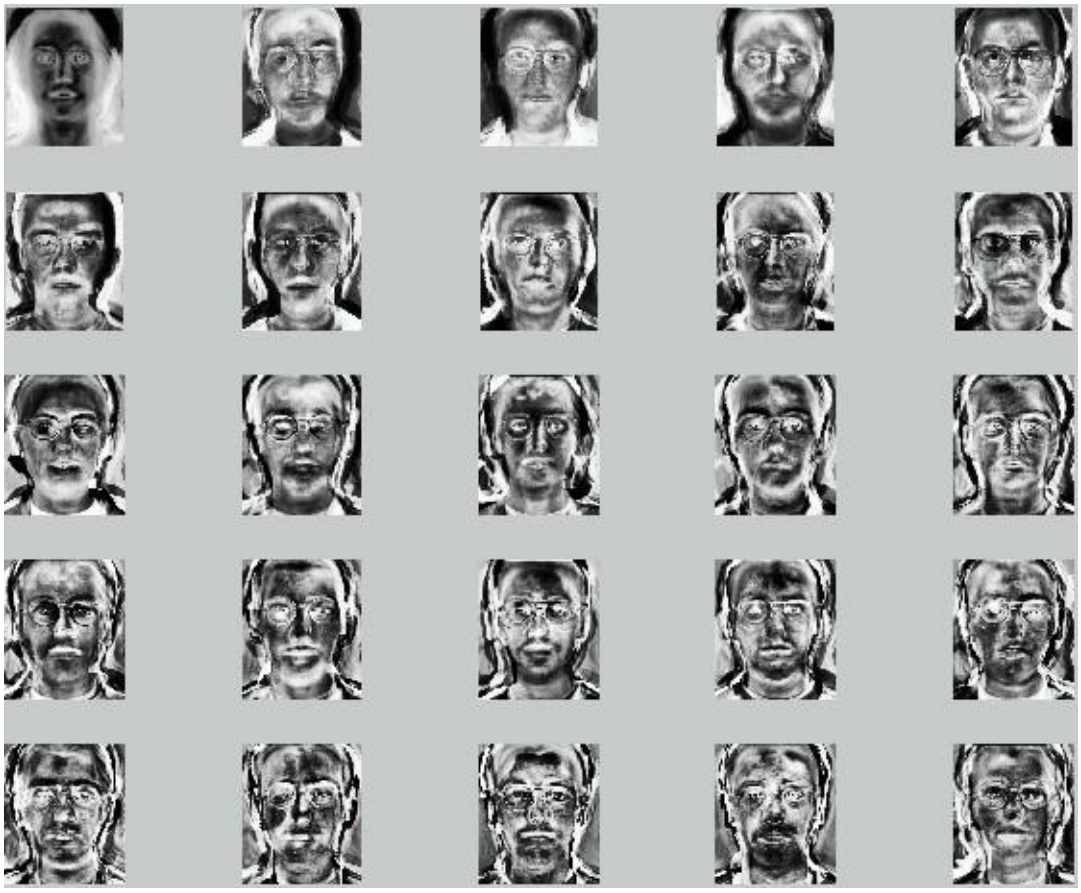

Figura 2.6 Conjunto de Eigenfaces

De aquí se desprende que los valores de las distancias dependen en cierta medida del tamaño de la base de datos, puesto que la matriz de covarianza y los vectores propios son calculados a partir de la matriz formada por la imagen de entrada y las ya almacenadas.

2.4.2 FisherFaces

Esta técnica considera las imágenes de entrenamiento de un mismo individuo como clases, por lo tanto existen el mismo número de clases que personas. Una vez definida las clases se procede a calcular dos matrices: la matriz de dispersión entre clases y la matriz de dispersión dentro de clases. Una vez calculada estas matrices se obtiene una matriz de proyección donde cada columna será la base del nuevo sub-espacio, denominada *Fisherfaces* **[7]**.

Figura 2.7 Cuatro FisherFaces de una imagen de entrada

Para realizar el cálculo de la matriz de proyección, requiere que las matrices de dispersión sean no-singulares (que posean inversa), esto no siempre es posible debido a que el número de imágenes casi siempre es menos al número de pixeles de cada imagen. Para solucionar esto, se utilizan Componentes Principales en las matrices de dispersión para reducir su dimensión.

Las FisherFaces también permiten una reconstrucción de la imagen, por lo tanto también se utiliza la comparación de imágenes mediante la distancia euclidiana.

2.4.3 Local Binary Pattern

Local BinaryPattern o Patrón Binario Local **[12]** es un operador de textura que etiqueta los pixeles de una imagen por thresholding o umbral , método de segmentación de imagen que a partir de una en escala de grises se crea una imagen binaria, cada vecino del pixel y el resultado de la operación se considera como un numero binario. Una de las propiedades más importantes es su indiscriminación frente a los cambios en la escala de grises como por ejemplo la luminosidad.

En este algoritmo se dice que para cada pixel de la imagen a procesar se examina su 8-vecindad alrededor del pixel central como se muestra en la figura X-X de dimensiones 3x3. Para cada pixel en la vecindad se determinara la intensidad de ella por medio de dos valores: 1 si el valor es mayor al del pixel central o 0 si el valor es menor al del pixel, como se puede ver en la figura 2.8 el valor del píxel central es 5 por lo tanto 8 es mayor que él y en la matriz resultante se agregara un 1 en ese mismo lugar y los valores se codificaran en una cadena binaria formada los 8 números, a veces se transforma a número decimal para el mejor funcionamiento del algoritmo **[13]**.

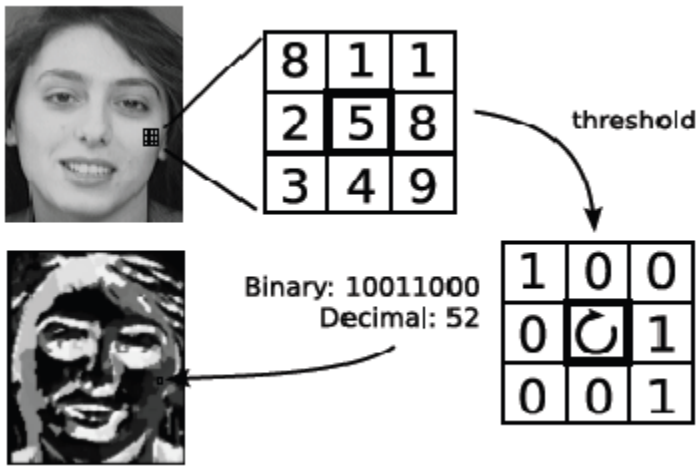

Figura 2.8 Creación de vector de algoritmo LBP

Al ya tener calculada la matriz de número con valores 1 y 0 se tienen 2 elevado a 8 posibles patrones , como podemos ver en la figura 2.9 cada valor del pixel se multiplica con la matriz para así calcular la distribución de textura de forma similar a los histogramas de escala de grises.

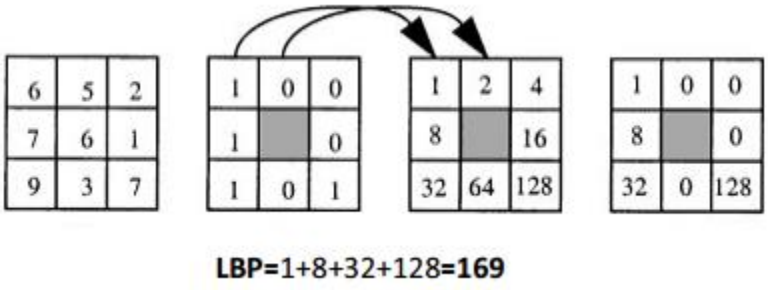

Figura 2.9 Calculo de distribución de textura **[13]**.

Para comparar dos imágenes se deben comparar los histogramas de cada una de ellas a través de la siguiente formula:

$$f(G,H) = \frac{\sum_{i=1}^{256} min(G_i, H_i)}{n}$$

En donde G, H son los histogramas de dos imágenes creadas por la técnica de la vecindad, Gi, Hi es el número de valores iguales en los histogramas respectivamente y n es el número de pixeles **[14]**.

Como se puede ver en la figura 2.10 la fotografía del rostro de la persona se ha dividido en bloques de igual tamaño , aunque no es necesario que sean iguales, y a cada uno de ellos se le ha calculado un histograma único el cual se ira complementando con los demás para así crear uno para toda la imagen. Cabe destacar que cada histograma se ve referenciado por los tonos grises de la imagen y es por esto que posee varios niveles de altura en su gráfico **[12]**.

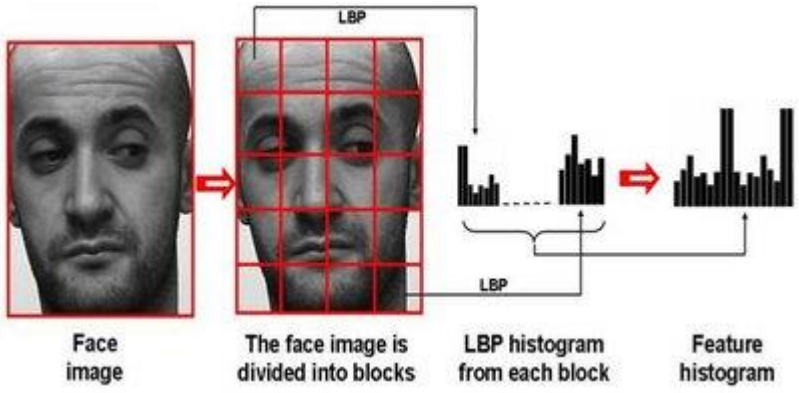

Figura 2.10 Reconocimiento facial con los patrones binarios locales **[10]**.

3 Modelo de datos

Como en un principio se mencionó el sistema de Reconocimiento Facial no está sujeto a servidores , ni requiere una conexión a Internet , solo se gestiona a través de una base de datos local situada en el Disco Duro del computador o en la dirección de escritorio que uno escriba en el código de programación.

Una Base de datos Local trabaja de la misma manera que una distribuida, o sea que se puede acceder a ella por medio de la red , solo que la primera se ve más restringida a los usuarios que puedan utilizarla , sin embargo sus datos se pueden visualizar , actualizar , ingresar o eliminar dependiendo de la función que se quiera realizar.

En el presente proyecto se ejecutará el funcionamiento de una Base de Datos local. Como se puede ver en la línea de código la dirección en donde se almacenaran las imágenes y nombre es: /TrainedFaces en donde se recibe la imagen de entrada y se obtiene el nombre escrito en el TextBox “NOMBRE_PERSONA”, se valida si esta repetido, y si la carpeta de guardado no existe se crea una con el mismo nombre.

```
Random rand = new Random();
bool esta_creado = true;
string nombrerostro = "face_" + NOMBRE_PERSONA.Text + "_" + rand.Next().ToString() + ".jpg";
while (esta_creado)
{

    if (!File.Exists(Application.StartupPath + "/TrainedFaces/" + nombrerostro))
    {
        esta_creado = false;
    }
    else
    {
        nombrerostro = "face_" + NOMBRE_PERSONA.Text + "_" + rand.Next().ToString() + ".jpg";
    }
}


if (Directory.Exists(Application.StartupPath + "/TrainedFaces/"))
{
    imagen_rostro.Save(Application.StartupPath + "/TrainedFaces/" + nombrerostro, ImageFormat.Jpeg);
}
```

Figura 3.1Función guardar imagen

Figura 3.2 Guardado de imágenes

Por otra parte dentro de la carpeta se irá almacenando cada imagen entrenada de la persona a la que se le capturó el rostro, además de esto se guarda un archivo llamado TrainedLabels que es un .xml generado donde se irán almacenando todos los nombres de las personas de sus respectivas imágenes. Como se muestra en la Figura 3.3 se puede ver que cada nombre tiene asociada su respectiva imagen.

```
<?xml version="1.0" encoding="utf-8"?>
<Faces_For_Training>
  <FACE>
    <NAME>Persona1</NAME>
    <FILE>face_Persona1_1899578231.jpg</FILE>
  </FACE>
  <FACE>
    <NAME>Persona1</NAME>
    <FILE>face_Persona1_1605338993.jpg</FILE>
  </FACE>
  <FACE>
    <NAME>Persona1</NAME>
    <FILE>face_Persona1_1803593042.jpg</FILE>
  </FACE>
  <FACE>
    <NAME>Persona1</NAME>
    <FILE>face_Persona1_1991942437.jpg</FILE>
  </FACE>
```

Figura 3.3 Nombres almacenados en xml

3.1 Requerimientos del proyecto

La ingeniería de requerimientos trata de establecer lo que el sistema debe hacer, sus propiedades emergentes deseadas y esenciales, y las restricciones en el funcionamiento del sistema y los procesos de desarrollo de software. Por lo tanto se debe considerar a la ingeniería de requerimientos como el proceso de comunicación entre los clientes, usuarios del software y los desarrolladores del mismo. "La captura de requisitos es la actividad mediante la que el equipo de desarrollo de un sistema de software extrae, de cualquier fuente de información disponible, las necesidades que debe cubrir dicho sistema".

3.1.1 Requerimiento General

La finalidad del proyecto a realizar es el de la implementación de un software que permita ,a través de algoritmos y librerías , poder reconocer el rostro de una persona por medio de una fotografía tomada por una cámara web , hacerle modificaciones a dicha imagen y poder compararla con una base de datos de para verificar la identidad de la persona.

3.1.2 Requerimientos funcionales

- Implementación de algoritmo para la detección de rostro.
- Implementación de algoritmo para el reconocimiento facial.
- Añadir y eliminar imágenes desde la base de datos.
- Mostrar en una pantalla de Windows la cámara web para la prueba de software.
- Un administrador del sistema podrá ingresar a él por medio de una contraseña única.
- Añadir opción para elegir el algoritmo de Reconocimiento facial que se desee ocupar.
- Mostrar imagen ya entrenada (blanco y negro).
- Crear base de datos local en la cual se almacenara cada imagen.
- Añadir figura geométrica la cual se posicionara en el rostro de la persona.
- Implementar segundo algoritmo de reconocimiento facial.
- Generar un manual de usuario para el uso posterior del sistema.
- Realizar más de un entrenamiento de imágenes por toma, por ejemplo tomar 10 fotografías, almacenarlas y entrenarlas con un solo click en el botón de entrenar.
- Realizar informe sobre los aciertos o no del reconocimiento facial, tanto en el reconocimiento como en la detección, abordando los 2 algoritmos implementados en el sistema.

3.1.3 Requerimientos no funcionales

- Diseño de una interfaz sencilla para el buen uso de un usuario principiante.
- Completa documentación del código.
- Rendimiento de software será optimo en donde tomara muy poco tiempo en realizar sus funciones.

- La Base de datos como se implementara de forma local siempre estará disponible.
- Integridad de las imágenes en la base de datos.
- Fácil interacción don el usuario.

3.2 Casos de uso

Un caso de uso es una descripción de la secuencia de interacciones que se producen entre un actor y el sistema, cuando el actor usa el sistema para llevar a cabo una tarea específica. Expresa una unidad coherente de funcionalidad, y se representa en el diagrama de casos de uso mediante una elipse con el nombre del caso de uso en su interior. El nombre del caso de uso debe reflejar la tarea específica que el actor desea llevar a cabo usando el sistema.

Los diagramas de casos de uso que se presentan a continuación tienen como finalidad modelar la interacción existente entre los distintos usuarios del sistema con éste, en conjunto con los actores involucrados para la ejecución de las funcionalidades diseñadas.

3.2.1 Caso de uso general

El siguiente diagrama nos muestra el caso de uso general del sistema y como los actores se desenvuelven en él.

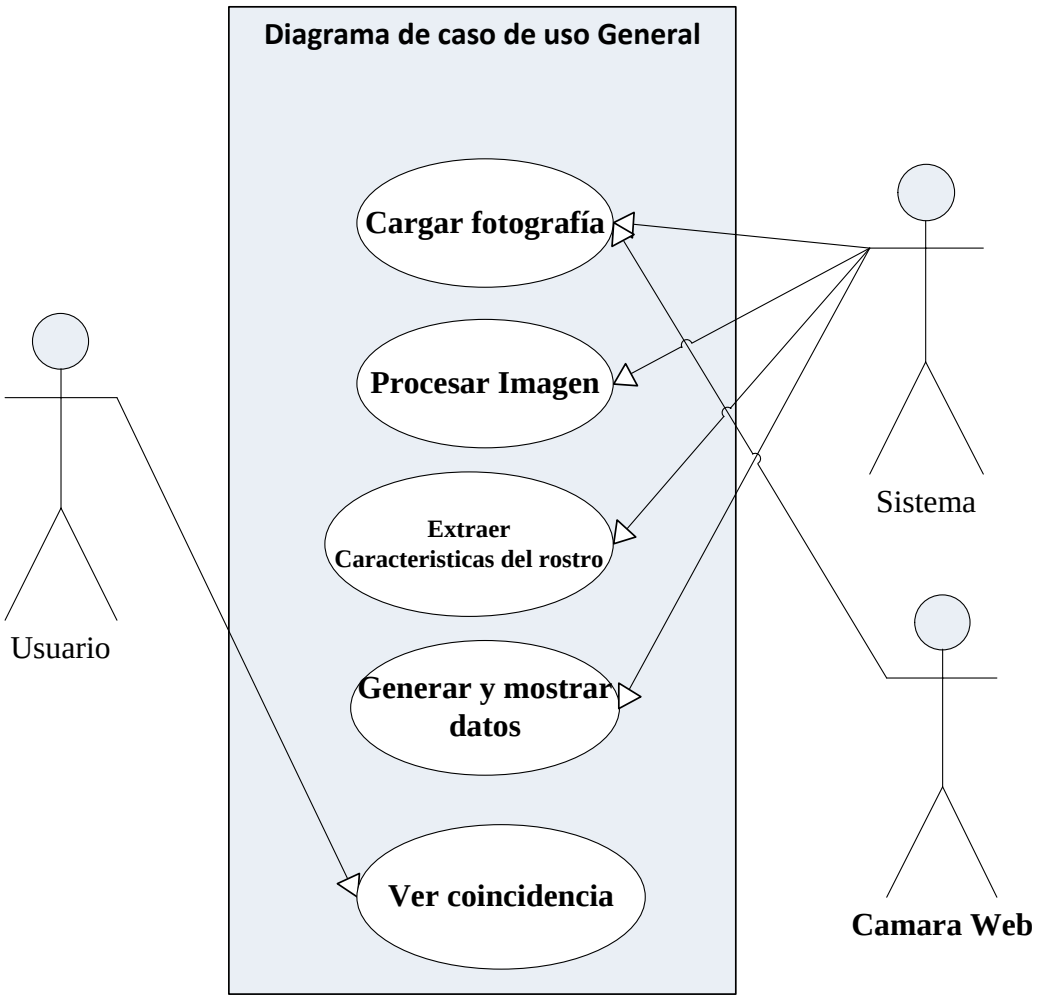

Figura 3.4 Diagrama de caso de uso general

3.2.2 Caso de uso Cargar fotografía

El siguiente caso de uso permitirá ingresar un usuario al sistema por medio de una imagen tomada por la cámara web.

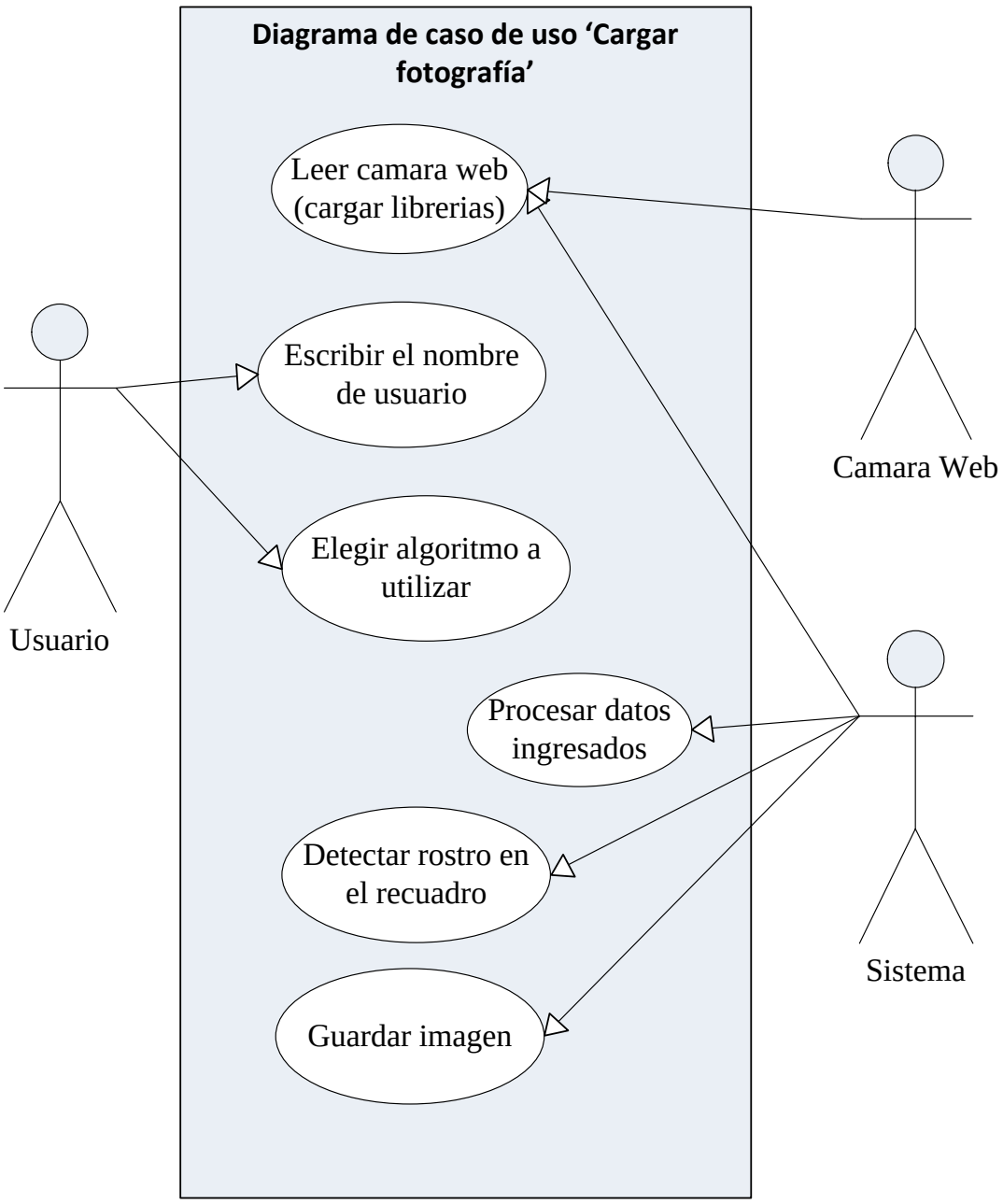

Figura 3.5 Diagrama de caso de uso Cargar fotografía

3.3 Diagrama de Secuencia

Los diagramas de secuencia que se presentan a continuación tienen por finalidad la de modelar la interacción entre objetos en un sistema.

3.3.1 Diagrama de secuencia Detectar Rostro

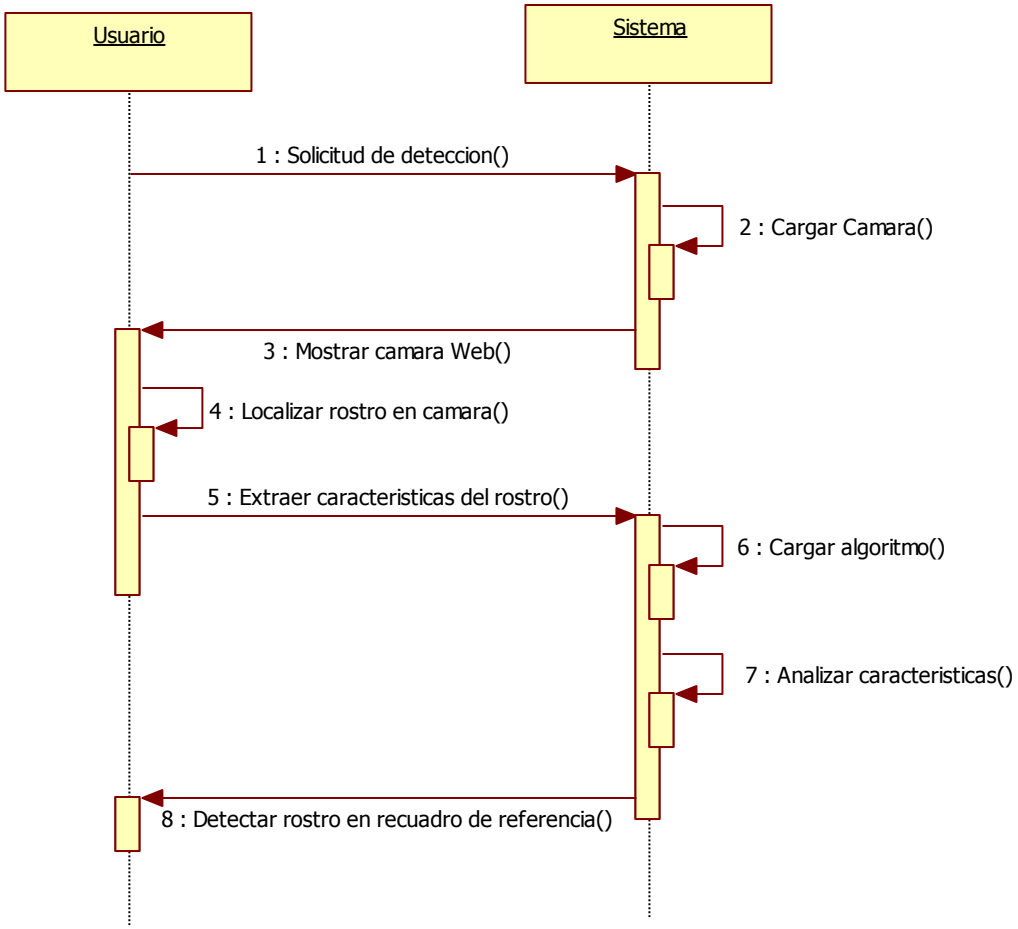

Figura 3.6 Diagrama de Secuencia Detectar Rostro

3.3.2 Diagrama de secuencia Reconocer Rostro

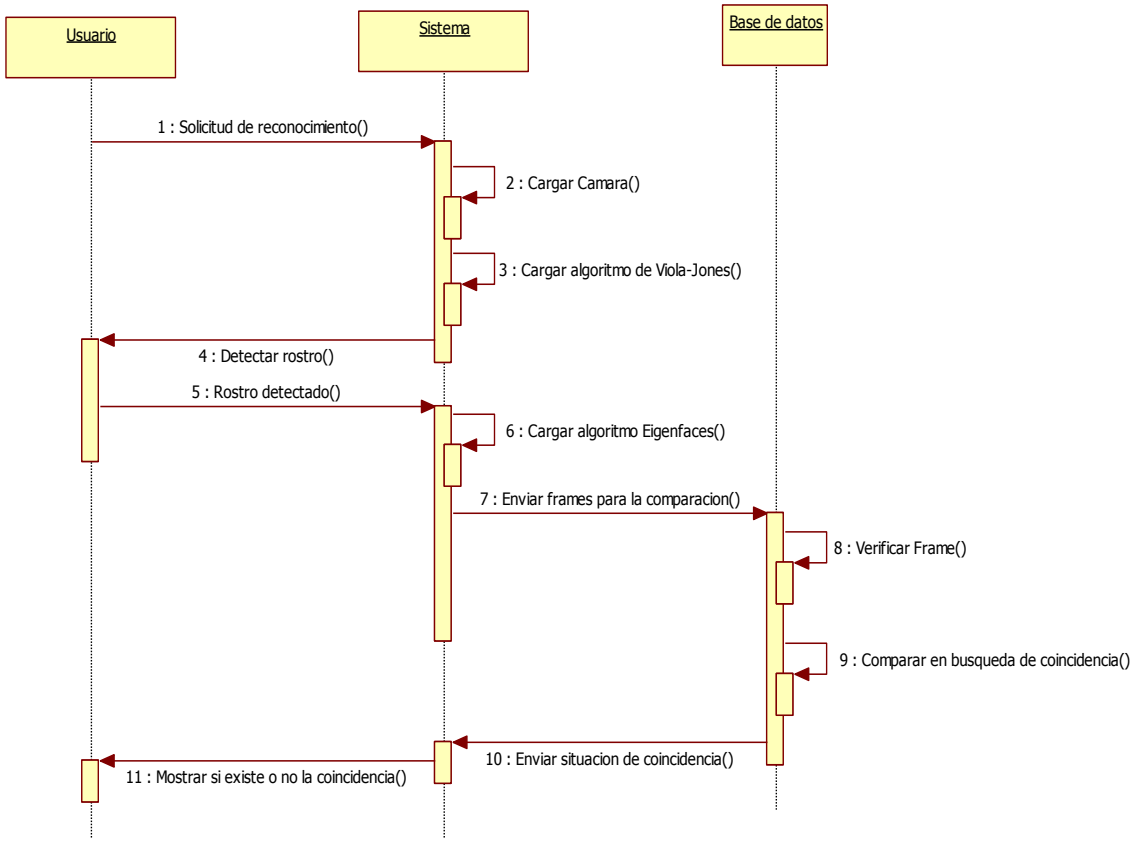

Figura 3.7 Diagrama de Secuencia Reconocer Rostro

3.3.3 Diagrama de secuencia Convertir imagen a grises

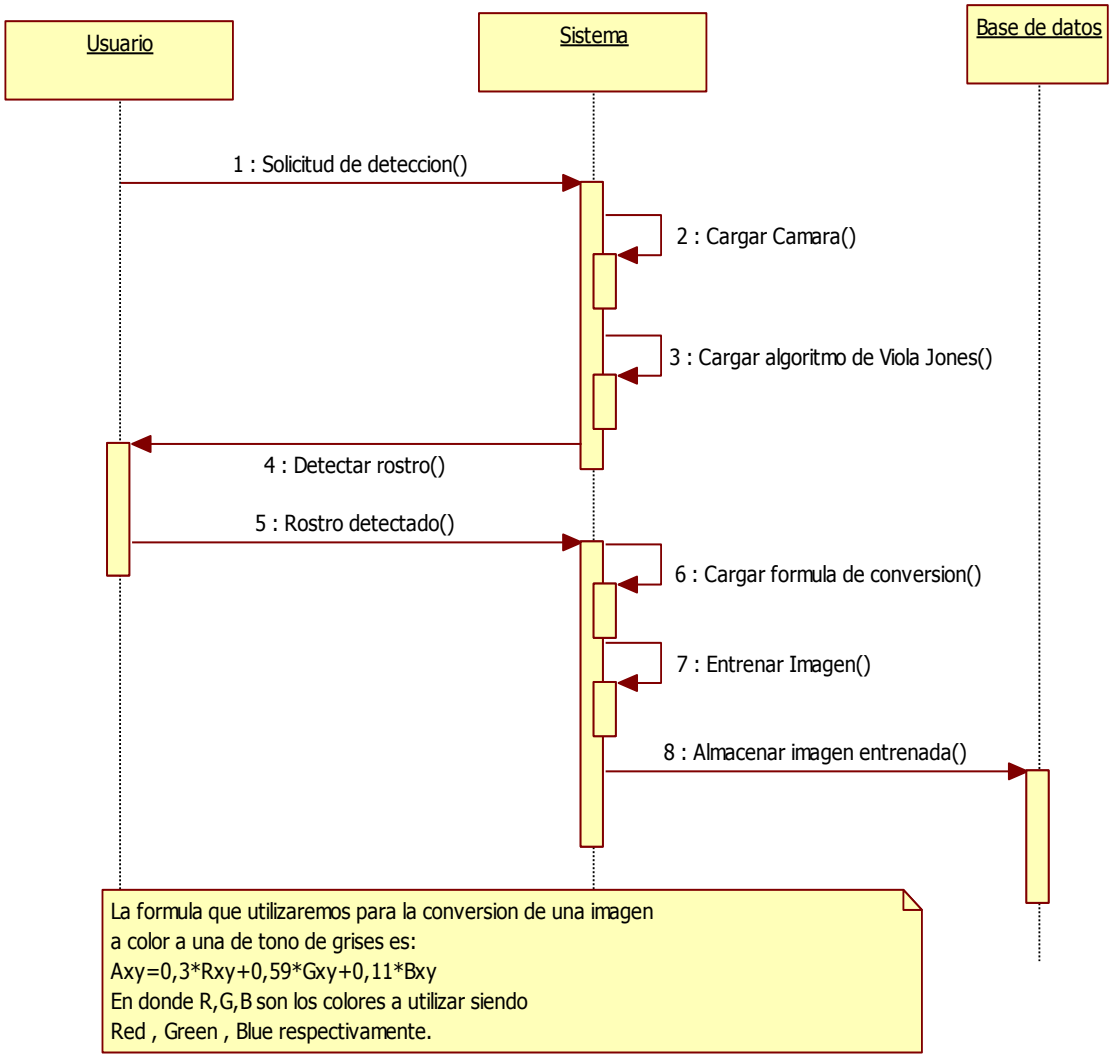

Figura 3.8 Diagrama de Secuencia Convertir imagen a grises

3.4 Diagramas BPMN

El principal objetivo de BPMN es proporcionar una notación estándar que sea fácilmente legible y entendible por parte de todos los involucrados e interesados del negocio. Entre estos interesados están los analistas de negocio, los desarrolladores técnicos y los gerentes y administradores del negocio. En síntesis BPMN tiene la finalidad de servir como lenguaje común para cerrar la brecha de comunicación que frecuentemente se presenta entre el diseño de los procesos de negocio y su implementación.

3.5 Diagrama BPMN Guardar el nombre de la persona

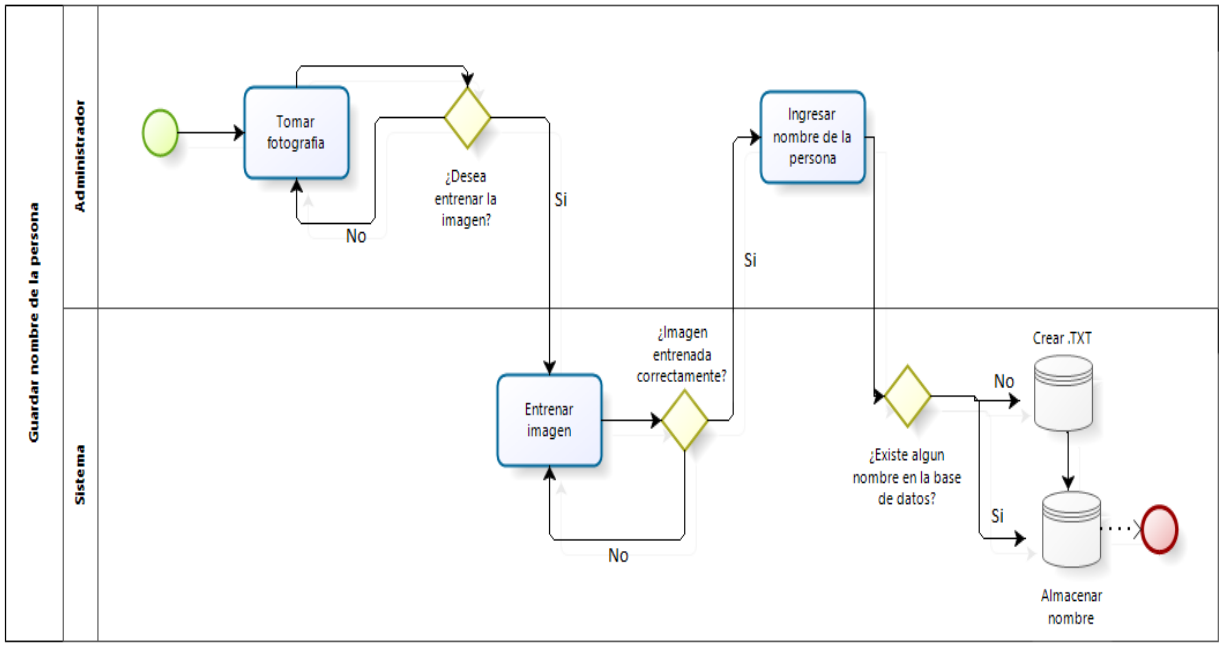

Figura 3.9 Diagrama BPMN Guardar nombre de la persona

3.5.1 Diagrama BPMN Mostrar Coincidencia

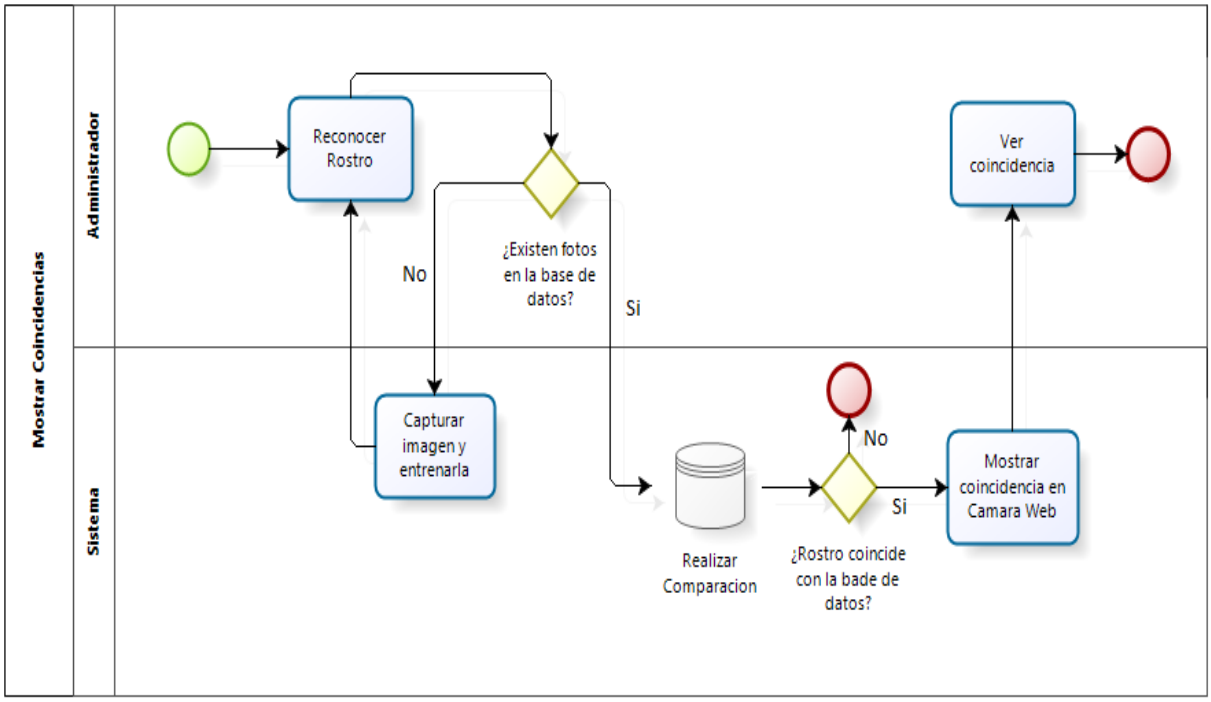

Figura 3.10 Diagrama BPMN Mostrar Coincidencia

4 Implementación del Algoritmo

4.1 Desarrollo del Sistema

Para el desarrollo del sistema de reconocimiento facial, se han estado estudiando los lenguajes de programación y las herramientas necesarias para el reconocimiento de rostro. A continuación se irán nombrando los diferentes elementos con los que se estará desarrollando el software de reconocimiento facial.

4.1.1 Microsoft Visual Studio

Microsoft Visual Studio es un entorno de desarrollo integrado (IDE, por sus siglas en inglés) para sistemas operativos Windows. Soporta múltiples lenguajes de programación tales como C++, C#, Visual Basic .NET, F#, Java, Python, Ruby, PHP; al igual que entornos de desarrollo web como ASP.NET MVC, Django, etc., a lo cual sumarle las nuevas capacidades online bajo Windows Azure en forma del editor Monaco. También es compatible con XML / XSLT, HTML / XHTML, JavaScript y CSS **[15]**.

Visual Studio permite a los desarrolladores crear aplicaciones, sitios y aplicaciones web, así como servicios web en cualquier entorno que soporte la plataforma .NET (a partir de la versión .NET 2002). Así se pueden crear aplicaciones que se comuniquen entre estaciones de trabajo, páginas web, dispositivos móviles, dispositivos embebidos, consolas, etc.

Figura 4.1 Logo de programa Visual Studio

4.1.2 OpenCV

OpenCV viene de las siglas Open Source Computer Vision Library **[16]**, es una librería abierta desarrollada por Intel en el año 1999, contiene alrededor de 500 funciones. Esta librería proporciona un alto nivel de funciones para el procesado de imágenes. Algunas de las características que permite OpenCV son operaciones básicas, procesamiento de imágenes, análisis estructural, análisis de movimiento, reconocimiento del modelo, reconstrucción 3D, calibración de cámara, etc.

Figura 4.2 Logo de la librería OpenCV

4.1.3 EmguCV

EmguCV es una plataforma cruzada .Net ligada a la librería de Intel OpenCV de procesamiento de imágenes, permitiendo que las funciones de OpenCV sean llamadas desde .Net, compatible con lenguajes como C#, VB, VC ++, etc. EmguCV está escrito en C#, puede ser compilado en forma Mono (Monodevelop) por lo cual puede ejecutarse en cualquier plataforma que contenga la forma Mono, incluyendo Linux/Solaris y Mac.

Es necesario descargar todos los dlls que vienen incluidos en EmguCV para el uso de esta plataforma.

Figura 4.3 Logo de la plataforma EmguCV **[17]**

4.1.4 Entrenamiento de rostros

Existen bases de datos universales en las cuales están almacenado los rostros de personas comunes y corrientes grabadas en condiciones naturales y con distintos tipos de gestos faciales como por ejemplo, como se muestra en la figura 4.4. Estas bases de datos están liberadas para poder ser descargadas desde la Web para fines de investigación **[18]**.

Figura 4.4 Gestos faciales de un sujeto en una base de datos YALE

Estas imágenes permiten ahorrarnos el paso del cambio de color a gris por medio de un algoritmo en el software y a su vez poder entrenarlas inmediatamente.

Como se muestra en la figura .5 a cada imagen original se le ha aplicado un algoritmo en particular ya sea:

- Eigenfaces.
- FisherFaces.
- Laplacianfaces.

Esto permite saber qué imágenes son aptas para poder reconocerlas y cuales se deben rechazar por no ser aptas para entrenarlas y así se podrán realizar estudios sobre qué algoritmo posee menor taza de error y cuál es el que permite mayor tolerancia a la luz o a los movimientos de la cabeza.

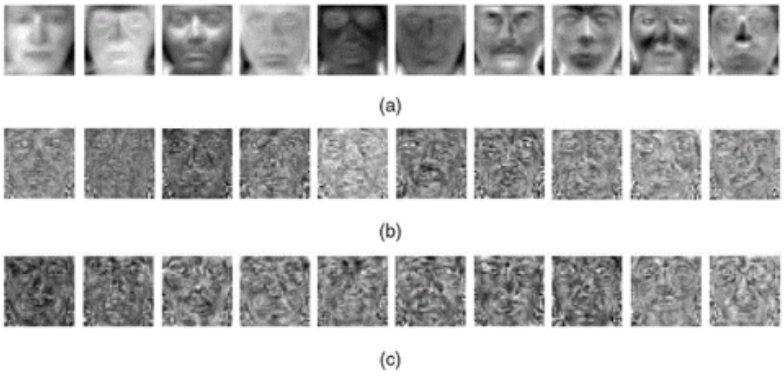

Figura 4.5 Imágenes entrenadas a partir de base de datos YALE

Las bases de datos más conocidas en torno a imágenes para reconocimiento facial son YALE, BioId **[19]**, The AR-FACE, ORL, entre otras. Todas se descargan de manera gratuita y poseen características diferentes unas de otras como la iluminación, incluyen puntos en la cara etc. **[20]**.

4.2 Plan y diseño de pruebas

Las pruebas en un sistema sirven para verificar que el sistema de información cumple con las necesidades establecidas por el usuario, con las debidas garantías de calidad, para ello se realiza un sistema de pruebas que implican la operación o aplicación del mismo a través de condiciones controladas y por tanto la consiguiente evaluación. Las condiciones controladas deben incluir tanto situaciones normales como anormales. El objetivo de las pruebas de un sistema es encontrar un error para determinar qué hacer en el momento que ocurre algo que no debe pasar y viceversa, es decir, un sistema de pruebas está orientado a detectar errores del sistema de información.

Para poder realizar el sistema que deseamos veremos a continuación los distintos tipos de prueba que existen.

- Pruebas de Caja Negra: El sistema de pruebas de caja negra no considera la codificación dentro de sus parámetros a evaluar, es decir, que no están basadas en el conocimiento del diseño interno del programa. Estas pruebas se enfocan en los requerimientos establecidos y en la funcionalidad del sistema.
- Pruebas de Caja Blanca: Al contrario de las pruebas de caja negra, éstas se basan en el conocimiento de la lógica interna del código del sistema. Las pruebas contemplan los distintos caminos que se pueden generar a través de las estructuras condicionales, a los distintos estados del mismo y otros.
- Pruebas de Integración. Las pruebas de integración buscan probar la combinación de las distintas partes de la aplicación para determinar si funcionan correctamente en conjunto.
- Pruebas del sistema. Son similares a las pruebas de caja negra, solo que éstas buscan probar al sistema como un todo. Están basadas en los requerimientos generales y abarca todas las partes combinadas del sistema.
- Pruebas de implantación. Incluyen las verificaciones necesarias para asegurar que el sistema funcionará correctamente en el entorno de operación al responder satisfactoriamente a los requisitos de rendimiento, seguridad, operación y coexistencia con el resto de los sistemas de la instalación, y conseguir la aceptación del sistema por parte del usuario de operación.
- Pruebas de aceptación. Van dirigidas a validar que el sistema cumple con los requisitos de funcionamiento esperados.

4.3 Planificación de las pruebas

Para realizar la planificación de pruebas en nuestro sistema se utilizó las pruebas de caja negra y los puntos a evaluar se definen a continuación.

- Pruebas de funcionalidad: Este tipo de pruebas examina si el sistema cubre sus necesidades de funcionamiento, acorde a las especificaciones de diseño. En ellas se debe verificar si el sistema lleva a cabo correctamente todas las funciones requeridas, se debe verificar la validación de los datos y se debe realizar pruebas de comportamiento ante distintos escenarios.

  Estas pruebas deben estar enfocadas a tareas, a límites del sistema, a condiciones planeadas de error y de exploración. Para estas pruebas usamos los esquemas de pruebas de caja negra ya que nos interesa saber si funciona correctamente independientemente de la forma en que se haga.

- Pruebas de usabilidad: Las pruebas realizadas en este rubro tienen la finalidad de verificar que tan fácil de usar es el software. Las pruebas de usabilidad deben verificar aprendizaje. Eficiencia, manejo de errores y grado de satisfacción.

4.4 Pruebas y Resultados

En este capítulo, se entregan los resultados de las pruebas realizadas para evaluar el rendimiento de los métodos de reconocimiento facial que se han implementado en este proyecto. Cada una de estas pruebas utilizó una o más de una base de datos, con características diferentes que serán útiles para tener una idea más clara del funcionamiento de los algoritmos frente a diferentes condiciones.

4.4.1 Estudio N°1

El primer set de pruebas está compuesto por 76 clases (76 individuos distintos), con una foto 100x100 entrenada desde una foto tamaño carnet 100x125.

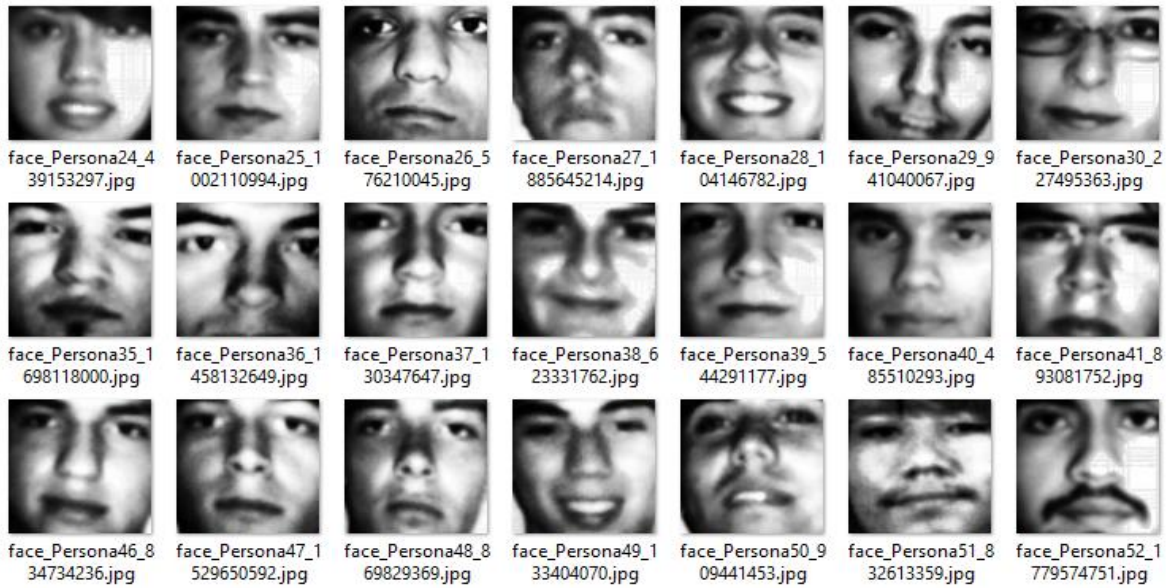

Figura 4.6 Figura Set de datos Prueba N°1

En este estudio todos los rostros tienen la misma expresión, y están situados de manera frontal, además no hay cambio de entorno. Los resultados son:

Generación 2012: 76 individuos	Aciertos	Falsos Positivos	Falsos Negativos
EigenFaces	80,2%	11,8%	8%
FisherFaces	80,2%	4%	15.8%
LBP	92,1%	4%	3,9%
EigenFaces2	80,2%	14,4	5,4%

Tabla 1.1 Resultados Estudio N°1

Los resultados son óptimos debido a las condiciones ideales de las fotos entrenadas y de las imágenes de entrada, puesto que ambas imágenes corresponden a la misma, una foto extraída de un sitio externo, por lo tanto son independientes de la iluminación, de la posición del individuo y de la distancia con respecto a la cámara.

4.4.2 Estudio N°2

El segundo set de pruebas está compuesto por 33 clases, que fueron entrenadas de fotos tomadas por una cámara digital. Cada clase tiene tres imágenes. La imagen de entrada en la mayoría de los casos fue una cuarta foto, distinta a la almacenada en la carpeta de entrenamiento.

Figura 4.7 Figura Set de datos Prueba N°2

Resultados del estudio:

Generación 2013: 33 individuos	Aciertos	Falsos Positivos	Falsos Negativos
EigenFaces	93,3%	0%	6,7%
FisherFaces	96,9%	0%	3,1%
LBP	100%	0%	0%
EigenFaces2	96,9%	0%	3,1%

Tabla 1.2 Resultados Estudio N°2

El éxito de este estudio se debe a la cantidad de fotos por clase y al nulo cambio de luz.

4.4.3 Estudio N° 3

Para la prueba número 3 se registraron los rostros de 11 individuos, para cada uno de éstos se les tomó 3 fotos, 1 de rostro frontal y 2 con leves inclinaciones hacia los costados, las cuales posteriormente fueron entrenadas y almacenadas en la base de datos local. El registro de cada persona se realizó por medio del nombre: Persona 1, Persona 2 y así sucesivamente hasta la Persona número 11. El algoritmo utilizado en esta prueba fue Local Binary Pattern, se decidió usar éste por los buenos resultados obtenidos en las pruebas anteriores. En la siguiente imagen se visualizará de mejor manera lo explicado.

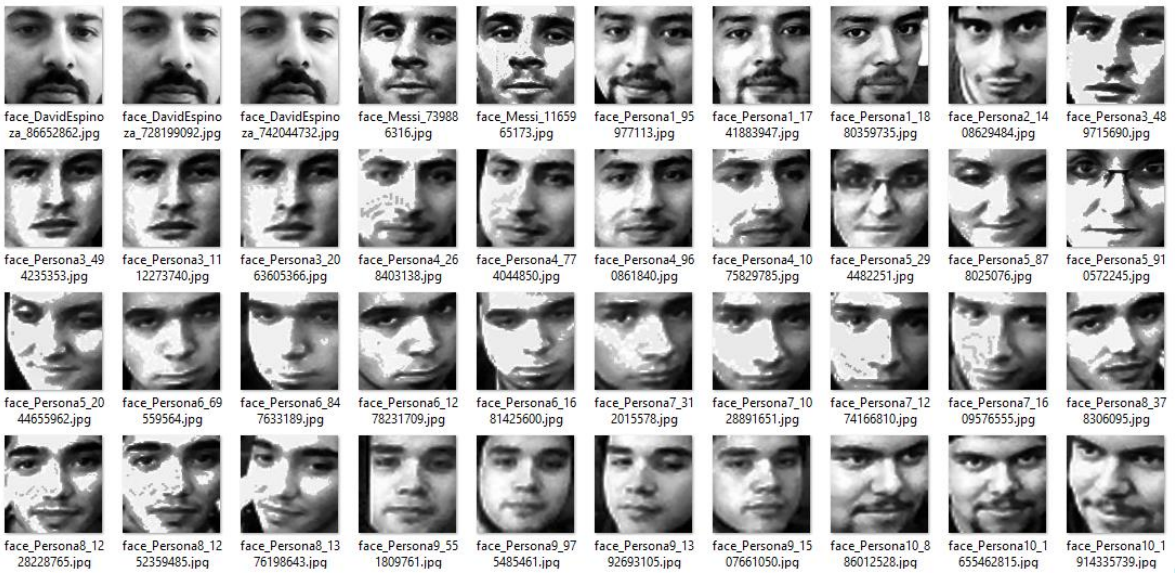

Figura 4.8 Figura Set de datos Prueba N°3

Los resultados obtenidos durante la prueba número 3 fueron de cierta manera los esperados dado el algoritmo utilizado. Se situó al individuo frente a la cámara web para poder realizar el reconocimiento de su rostro, postura ideal para poder realizar el estudio. Los resultados fueron los siguientes:

- De 11 Personas 7 fueron reconocidas de manera correcta. (63.63%)
- 2 personas nunca fueron reconocidas por el nombre que se le había asignado. (18.18%)
- 1 Persona no fue reconocida ni detectada por la toma de fotografías realizada anteriormente. (9.09%)
- 1 Persona falto al estudio. (9.09%)

Como conclusión a la prueba realizada podemos referirnos a que el algoritmo Local Binary Pattern es el algoritmo que mejor funcionó en el ámbito de reconocimiento durante las 3 pruebas realizadas en el proceso de estudio. Éste funciona de gran manera generando un grado de acierto de aproximadamente mayor a 80%.

5 Conclusión y trabajos futuros

5.1 Conclusión

El ser humano en su quehacer diario siempre ha buscado aumentar la seguridad en lo referente a bienes que poseen y que pudieran ser robados por medio de una simple contraseña o una clonación de la tarjeta. El reconocimiento facial le da al usuario lo que realmente necesita, más seguridad y sentir la confianza de que no les pasará nada a sus cuentas o los bienes con que interactúa.

Cada sistema biométrico posee características que lo llevan a crear quiebres en su seguridad ya sea por ejemplo la clonación de una mano para la huella dactilar o un simple lente de contacto que refleje el iris de otra persona. En la actualidad no existe una técnica de reconocimiento que en realidad pueda ser considerada como la mejor, todo depende de las condiciones en la que se quiera implementar el sistema, sin embargo la herramienta biométrica de reconocimiento facial es una casi impenetrable.

El reconocimiento facial no requiere un contacto físico con la persona solo una cámara web y un software eficiente y es lo que se intentará implementar en este proyecto.

Al estudiar cada una de las técnicas para reconocer un rostro se pudo observar la complejidad que posee cada uno de ellos por medio de fórmulas matemáticas complejas combinadas con matrices en relación con el rostro humano. Cada una de ellas posee sus ventajas y desventajas, como también su taza de error lo cual se estudió para utilizar la técnica correcta y poner mayor énfasis en ella. A su vez se comparó cada una de ellas con imágenes ya entrenadas en bases de datos descargadas desde la Web lo cual permite obtener resultados más exactos y de forma transparente, al ser imágenes utilizadas por todos.

Por concluir en este proyecto se evaluara cada uno de los algoritmos ya mencionados para que más adelante personas externas puedan sacar sus propias conclusiones sobre cuál de ellos implementarán en su software.

5.2 Mejoras en el sistema

Para que un sistema sea perfecto y funcione de forma correcta se debe ir mejorando y actualizando hasta quedar un prototipo dispuesto a realizar sus funcionalidades de manera correcta y con pequeñas tazas de errores.

Es por esto que el sistema no queda exento de estos problemas los cuales se explicarán detalladamente a continuación:

1. Primero que todo se debe optimizar el sistema para que cuando hayan muchas imágenes entrenadas en la base de datos local, el programa no sufra de lentitud al momento de reconocer al individuo, esto se produce posiblemente por la cantidad de veces que recorre las imágenes entrenadas, por ejemplo en un segundo.
2. Mejoras en el apartado de interfaz para que ésta sea más acorde a la tecnología que es usada en la actualidad, por ejemplo colores o ventanas más llamativas
3. Se debe mejorar el ámbito de almacenamiento de fotografías entrenadas en la base de datos, esto para optimizar el reconocimiento facial, a fin de mejorar el recorrido del algoritmo por las imágenes, permitiendo recorrer por imágenes candidatas y no por el número de imágenes totales. Algunas soluciones para este problema serían:
    - Agrupar imágenes por medio del promedio de umbral que posee cada una para así generar una comparación más fluida.
    - Agrupar las imágenes por carpeta, siendo ésta una por persona para poder realizar la búsqueda de manera más rápida.
4. Mejorar el momento de generar el reconocimiento del rostro de los individuos para que éste reconozca con una menor tasa de error y así poseer un software más confiable dentro de lo que se requiera en su uso.

6 Referencias

[1] García, M. (2009). Diseño e implementación de una herramienta de detección facial. [Fecha de consulta: 23 de marzo 2015]. Disponible en: http://tesis.ipn.mx/bitstream/handle/123456789/6111/DISENOIMPLEMFACIAL.pdf

[2] van Vliet, H. (2007). Software Engineering: Principles and Practice. [Fecha de consulta: 28 de marzo 2015]. Disponible en: http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.128.2614&rep=rep1&type=pdf

[3] Viola, P., Jones M. (2001). Rapid Object Detection using a Boosted Cascade as Simple Features. *Computer vision and pattern recognition*, 16(1), 511-518.

[4] Yang, M., Kriegman D., Ahuja N. (2002). Detecting Faces in Images: A Survey. *Transactions on pattern analysis and Machine Intelligence*, 24(1), 34-58.

[5] Viola, P., Jones M. (2001). Robust Real-Time Face Detection. *International Journal of Computer Vision*, 57(2), 137-154.

[6] Horsh W. (2005). Machine Learning. [Fecha de consulta: 22 de septiembre 2015]. Disponible en: http://global.britannica.com/technology/machine-learning

[7] Belhumeur, P., Hespanha J., Kriegman D. (1997). Eigenfaces vs. Fisherfaces: Recognition Using Class Specic Linear Projection. *Transactions on pattern analysis and Machine Intelligence*, 19(7), 711-720.

[8] Deparment of Economic Informatics and Cybernetics, Bucharest (2011). A Real-Time Face Recognition System Using EigenFaces. www.jmeds.eu.

[9] Xiaoguang L. (2005). Image Analysis for Face Recognition. [Fecha de consulta: 23 de marzo 2015]. Disponible en: http://face-rec.org/interesting-papers/General/ImAna4FacRcg_lu.pdf

[10] Saha, R., Bhattacharjee B. (2013). Faces Recognition Using EigenFaces.

International Journal of Emerging Technology and Advanced Engineering, 3(3), 90-93.

[11] Lorente, L. (1998). Representación de Caras mediante EigenFaces. *Escola Técnica Superior d'Enginyers de Telecomunicación de Barcelona*, 11(14), 13-20.

[12] PIETIKÄINEN M. Local Binary Patterns. Actualizada: 20 septiembre 2014. [Fecha de consulta: 27 agosto 2014. Disponible en: http://www.scholarpedia.org/article/Local_Binary_Patterns

[13] Maturana D., Mery D., & Soto A. (2010). Face Recognition with Local Binary Patterns, Spatial Pyramid Histograms and Naive Bayes Nearest Neighbor classification. *Chilean Computer Science Society (SCCC)*, 25(1), 125-132.

[14] Trefný J. Matas J (2010). Extended Set of Local Binary Patterns for Rapid Object Detection. *Computer Vision Winter Worshop*, 15(1), 37-43.

[15] Microsoft (2015). [Fecha de consulta: 01 de septiembre 2015]. Disponible en: https://www.visualstudio.com/features/universal-windows-platform-vs

[16] G. R. Bradski, A. Kaehler, Learning OpenCV Vision with the OpenCV Library. [Fecha de consulta: 23 de Octubre 2014]). Disponible en: http://www.cse.iitk.ac.in/users/vision/dipakmj/papers/OReilly%20Learning%20OpenCV.pdf

[17] EmguCV (2015). [Fecha de consulta: 11 de septiembre 2015]. Disponible en: http://www.emgu.com/wiki/index.php/Main_Page

[18] Vincent RABAUD. UCSD ComputerVision. Actualizada: 27 Agosto 2014. [Fecha de consulta: 23 de septiembre 2014]. Disponible en: http://vision.ucsd.edu/content/yale-face-database

[19] BioIDGMBH.BioID Face Database. Actualizada: 23 de septiembre 2014. [Fecha de consulta: 23 de septiembre 2014]). Disponible en: https://www.bioid.com/About/BioID-Face-Database

[20] Robert FRISCHHOLZ. Datasets. Actualizada: 13 de septiembre 2012. [Fecha de consulta: 23 de septiembre 2014]. Disponible en: http://www.facedetection.com/facedetection/datasets.htm

[21] Vision Artificial con OpenCV: un Enfoque Práctico, Curso EmguCV – Sesión Practica Nivel 2: Ejercicios Avanzados.

7 Anexos

7.1 Diseño final del sistema

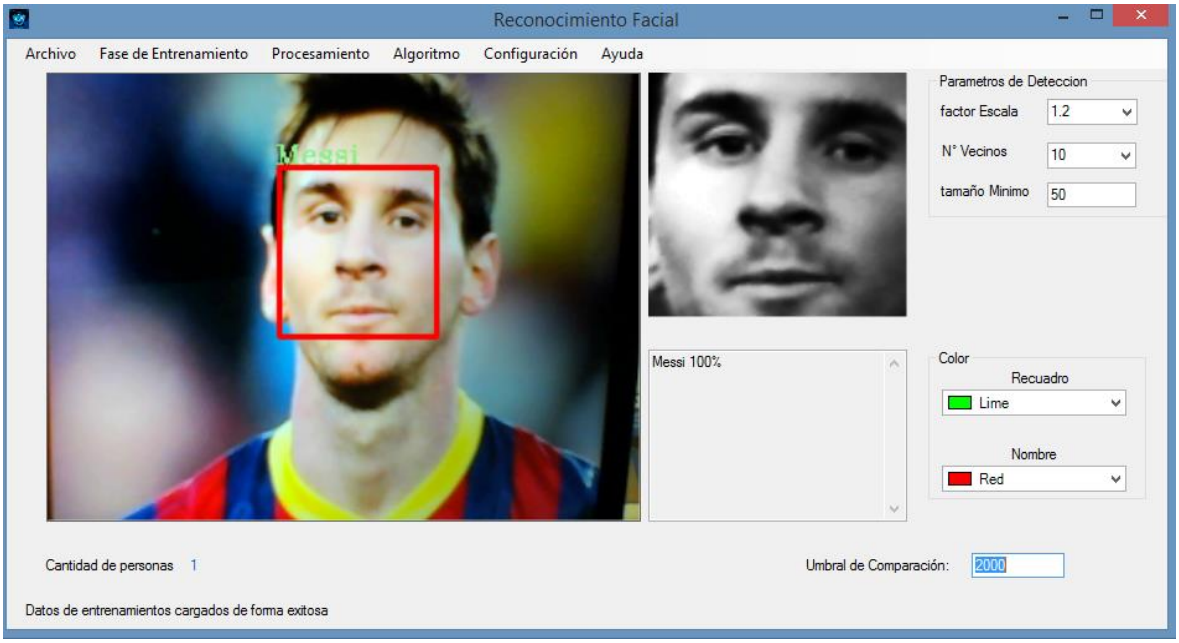

Figura 5.1 Diseño en ejecución

7.1.1 Parametros de detección

En el código se encuentra la función Rectangle:

```
Rectangle[] rostrosDetectados = rostro.DetectMultiScale(gris, FactorEscala, MinVecinos, newSize (WindowsSize, WindowsSize), Size.Empty);
```

Donde:

- gris: Es una imagen en escala de grises.
- FactorEscala: número real que indica el factor de escala utilizado a la hora de recorrer la imagen en busca de patrones, en las diversas iteraciones. Un factor de 1.1 implica que tras la primera iteración en busca de patrones, aplicará un offset de un 10% para buscar en una segunda iteración, y así sucesivamente. Este parámetro varía de 1.1 a 1.5 (+10% - +50%, respectivamente), siendo mucho más lento (pero más eficaz) el valor más pequeño.
- MinVecinos: número mínimo de vecinos antes de considerar un positivo. En cada iteración se obtiene un conjunto de patrones positivos y aquellos que se encuentran solapados se unifican en uno solo. Este número entero especifica cuántos solapamientos han de considerarse para denotar ese patrón como positivo. Un número de vecinos de 2 a 5 es lo más utilizado (usando 2 tendremos resultados más rápidos pero menos eficientes).
- WindowsSize: tamaño mínimo del rostro a buscar **[21]**.

7.1.2 Llamada a EigenFaces para reconocimiento de rostros

Mediante un for, donde previamente con el algoritmo de viola Jones se identifican la cantidad de rostros detectados, se llama a la función EigenFacesAlgoritmo, donde los parámetros ocupados corresponden a:

- ImágenesEntrenadas.ToArray(): Imágenes usadas para el entrenamiento, donde cada imagen debe tener el mismo tamaño, y es recomendable que las imágenes posean el histograma normalizado.
- labels,ToArray(): Array de string correspondiente a las imágenes.
- Umbral: El umbral de la eigen-distancia, se recomienda su uso entre [0,1000]. Mientras menor sea el número la imagen será tratada como un objeto desconocido. Si el umbral es < 0, el algoritmo siempre tratara la imagen como un objeto conocido.
- termCrit: Los criterios para el reconocimiento.

Para determinar si los rostros detectados coinciden con la base de datos de imágenes y desplegar el nombre que devuelve la función en el frame correspondiente y en las coordenadas que coinciden con el rostro de la persona identificada.

```
for (int i = 0; i < rostrosDetectados.Length; i++) {
    t = t + 1;
    resultado = ocupadoFrame.Copy(rostrosDetectados[i]).Convert<Gray, byte>().Resize(100, 100, Emgu.CV.CvEnum.INTER.CV_INTER_CUBIC);
    //Dibuja el cuadrado indicando rostro con color determinado
    ocupadoFrame.Draw(rostrosDetectados[i], new Bgr(colorPickerRecuadro.SelectedValue), 2);

    if (ImagenesEntrenadas.ToArray().Length != 0)
    {

        MCvTermCriteria termCrit = new MCvTermCriteria(ContEnt, 0.001);

        //Eigen face funcion
        EigenFacesAlgoritmo recognizer = new EigenFacesAlgoritmo(
          ImagenesEntrenadas.ToArray(),
          labels.ToArray(),
          Umbral,
          ref termCrit);

        nombre = recognizer.Reconocimiento(resultado);

        //Obtener el nombre y mostrarlo en el label
        ocupadoFrame.Draw(nombre, ref fuente, new Point(rostrosDetectados[i].X - 2, rostrosDetectados[i].Y - 2), new Bgr(colorPickerNombre.SelectedValue));
    }
    //Mostrar el numero de rostros identificados en la camara
    labelcantidadrostros.Text = rostrosDetectados.Length.ToString();
}
```

Figura 5.2 Llamada a función EigenFacesAlgoritmo

7.1.3 Momento del reconocimiento o rechazo

Luego de obtener la imagen promedio de las imágenes previamente entrenadas y los eigenvalores que determinan la eigendistancia se procede a reconocer al individuo, donde si la eigendistancia es menor al umbral ingresado por el usuario el rostro es reconocido por el algoritmo, de lo contrario despliega un string indicando que es desconocido.

```
public String Reconocimiento(Image<Gray, Byte> imagenes)
{
    int indice;
    float eigenDistancia;
    String label;
    String desconocido;

    desconocido = "Desconocido";
    EncontrarObjetosSimilares(imagenes, out indice, out eigenDistancia, out label);

    //Si no se cumple una de las 2 condiciones no reconoce al individuo, de lo contrario retorna el string _labels
    return (_eigenDistanciaUmbral <= 0 || eigenDistancia < _eigenDistanciaUmbral) ? _labels[indice] : desconocido;
}
```

Figura 5.3 Estructura de la Función Reconocimiento

7.1.4 Calculo de porcentaje de posibles candidatos

Para este procedimiento se ocupa la función EigenObjectRecognizer que recibe como parámetros el array de las imágenes entrenadas, el array de nombres extraidos del xml, el umbral que es el mismo utilizado para la detección de rostro, y MCvTermCriteria, un criterio de iteración utilizado tanto para la detección como en esta función. Si el número de labels (cantidad de imágenes guardadas) es mayor a 10, procede a realizar los cálculos mediante las eigendistancias obtenidas con la llamada a la función GetEigenDistancias. Se define un array de string del largo de los labels y se copia el array de nombres. Se crea un diccionario con las 10 eigendistancias más cercanas al umbral asociadas a su correspondiente imagen y nombre, y se le asigna un peso de 10%. El orden de las probabilidades es de mayor a menor.

```
//PORCENTAJEEEE
if (NumLabels >= 10)
{
    MCvTermCriteria termCrit = new MCvTermCriteria(ContTrain, 0.001);
    objRec = new EigenObjectRecognizer(imagenesEntrenadas.ToArray(),
             names2.ToArray(), Umbral, ref termCrit);

    //obtener las distancias
    float[] distances = objRec.GetEigenDistances(resultado);
    //copiar los labels en un array de string
    string[] labels2 = new string[objRec.Labels.Length];
    names2.CopyTo(labels2, 0);
    Array.Sort(distances, labels2); //Ordena el arreglo

    Dictionary<string, int> votedNames = new Dictionary<string, int>();

    //Agregar un nombre o incrementar su instancia (peso, %)
    for (int t = 0; t < 10; t++)
    {
        string names = labels2[t];

        if (!votedNames.ContainsKey(names))
        {
            //agregar nuevo
            votedNames.Add(names, 1);
        }
        else
        {
            //modificar peso, %
            votedNames[names] += 1;
        }
    }
```

Figura 5.4 Estructura de la Función Porcentaje (Parte 1)

```
//Mostrar probabilidades en orden de mayor a menor
List<string> results = new List<string>();
string[] names3 = votedNames.Keys.ToArray();
int[] weights = votedNames.Values.ToArray();
Array.Sort(weights, names3);
Array.Reverse(weights);
Array.Reverse(names3);
for (int j = 0; j < names3.Length; j++)
{
    results.Add(names3[j] + " " + ((int)(((double)weights[j] / (double)10) * 100)).ToString() + "%");
}

//tbResults.Lines = results.ToArray();
MethodInvoker del = delegate { tbResults.Lines = results.ToArray(); };
try
{
    tbResults.Invoke(del);
}
catch
{
    //nada
}
//PORCENTAJE
```

Figura 5.5 Estructura de la Función Porcentaje (Parte 2)

7.2 Caso de uso

7.2.1 Caso de uso Detectar Rostro

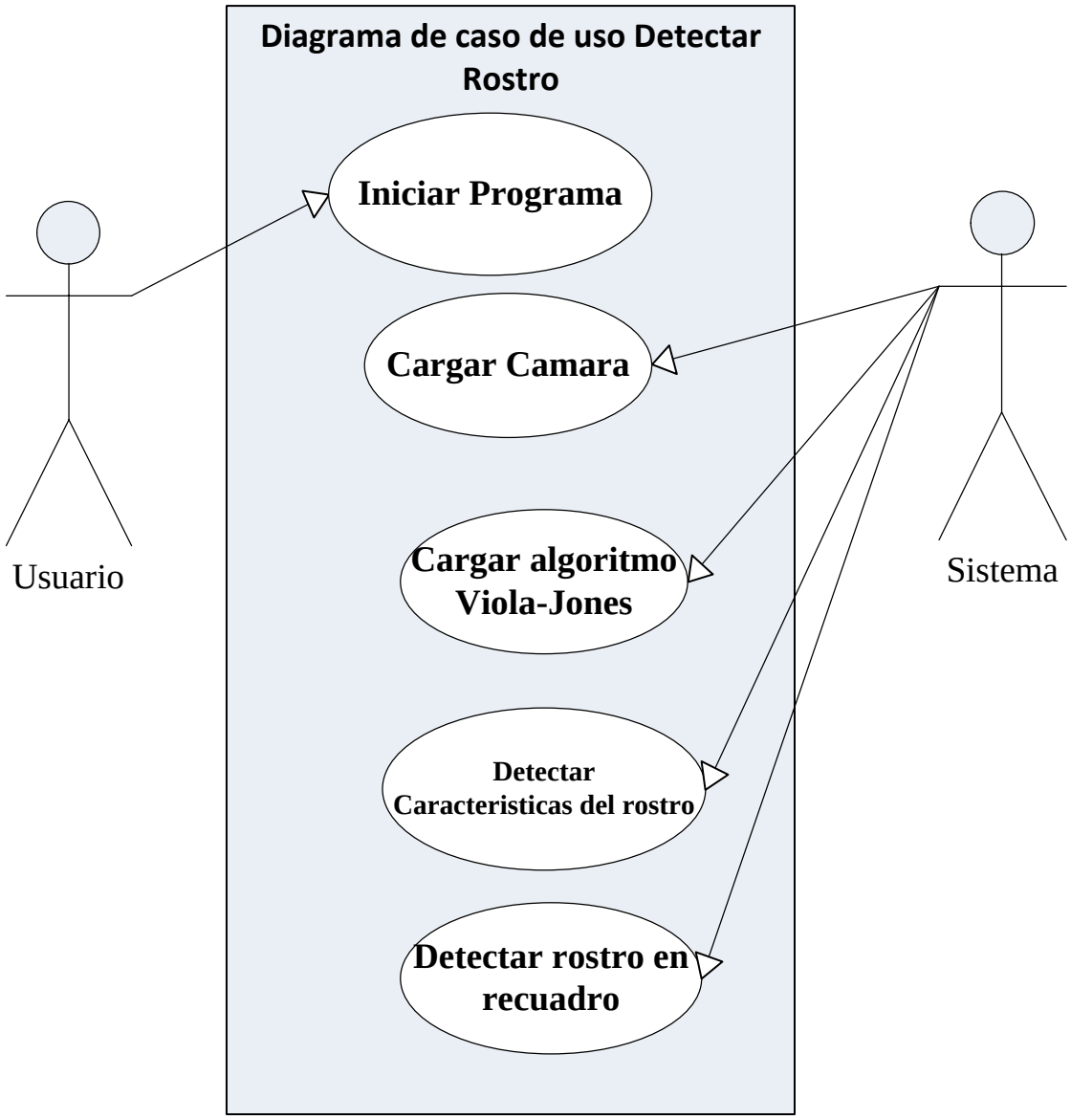

Figura 5.6 Caso de uso Detectar Rostro

7.2.2 Caso de uso Reconocer Rostro

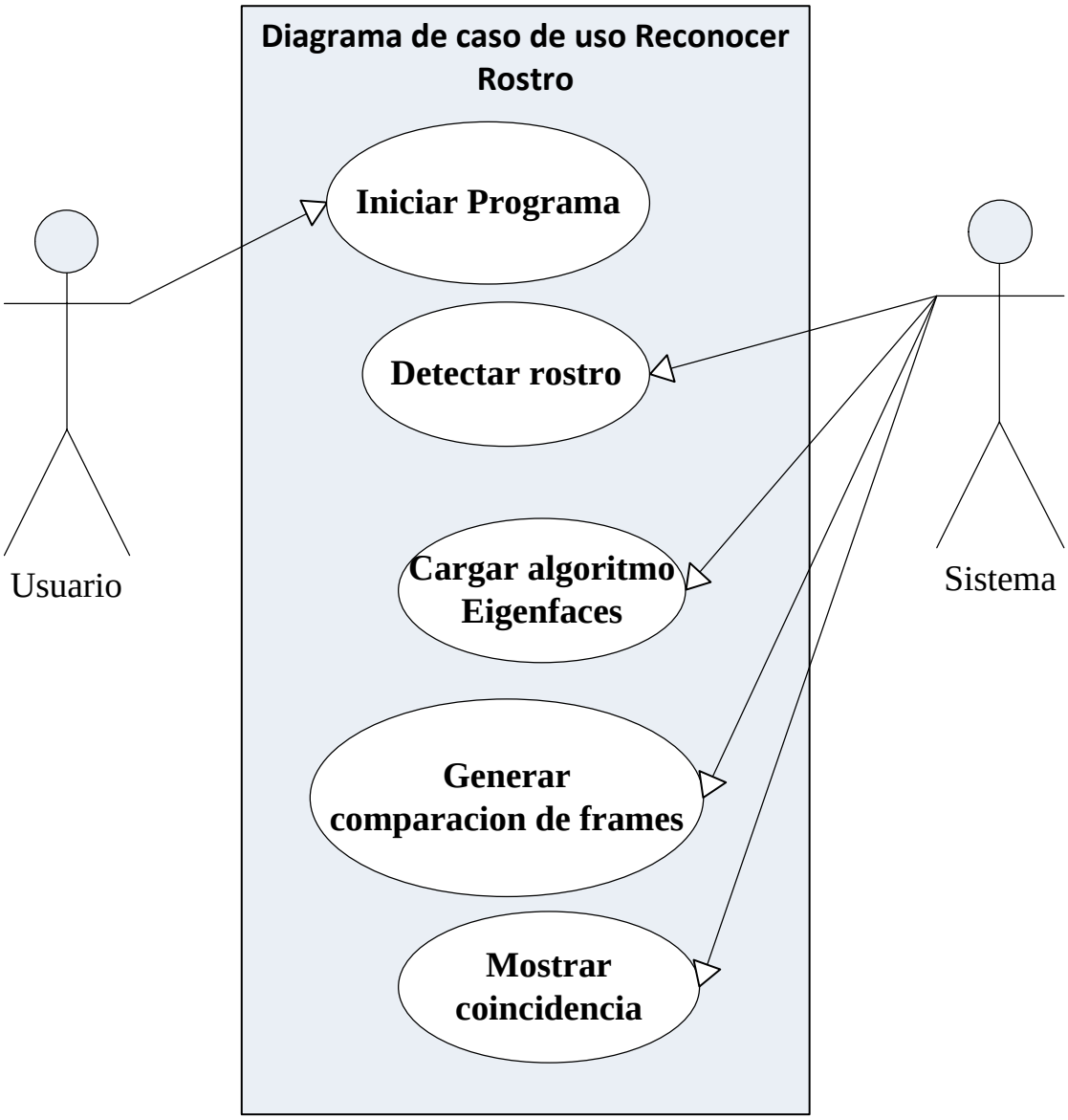

Figura 5.7 Caso de uso Reconocer Rostro

7.3 Especificaciones de caso de uso

7.3.1 Especificación de caso de uso Detectar Rostro

Caso de uso	Detectar Rostro
Actor Principal	Sistema
Participantes e Intereses	Sistema: El sistema debe generar la detección del rostro de la persona situada en la cámara web.
Precondiciones	Algoritmo Viola Jones cargado y Cámara web encendida
Post condiciones	Detección de rostro por medio de un recuadro de referencia
Escenario Principal	1.- El Administrador da inicio al programa 2.- El Sistema carga la cámara web 3.- El Sistema ejecuta el algoritmo de Viola-Jones 4.- Se detectan las características faciales de la persona Detecta el rostro de la persona
Extensiones	3.1.- Elegir color de recuadro de referencia
Requisitos Especiales	----------
Frecuencia de Ocurrencia	Muy Alta

Tabla 2.1 Especificación de uso Detectar Rostro

7.3.2 Especificación de caso de uso Reconocer Rostro

Caso de uso	Reconocer Rostro
Actor Principal	Sistema
Participantes e Intereses	El sistema luego de detectar y ejecutar la comparación del rostro de la persona con la Base de Datos, reconoce ésta y muestra si coincide o no.
Precondiciones	Rostro de la persona detectado e imágenes en la Base de Datos.
Post condiciones	Reconocimiento facial de la persona con nombre en el recuadro de referencia.
Escenario Principal	1.- El administrador da inicio al programa. 2.-Sistema detecta el rostro. 3.- Sistema carga el algoritmo de Eigenfaces. 4.- Sistema compara los frames del rostro de la imagen en la cámara web con las imágenes de la Base de Datos. 5.- Muestra la coincidencia de la persona en un nombre encima del recuadro de detección.
Extensiones	5.1.- Elegir el color del nombre de la coincidencia.
Requisitos Especiales	----------
Frecuencia de Ocurrencia	Muy alta.

Tabla 2.2 Especificación de uso Reconocer Rostro

7.3.3 Especificación de caso de uso Convertir Imagen a Grises

Caso de uso	Convertir imagen a grises.
Actor Principal	Sistema.
Participantes e Intereses	El Sistema debe ser capaz de convertir una imagen a color a una en tono de grises.
Precondiciones	Rostro detectado.
Postcondiciones	Imagen entrenada en todo de grises.
Escenario Principal	1.- El administrador da inicio al programa. 2.- Sistema detecta el rostro de la persona. 3.- El administrador debe seleccionar la opción de entrenar imagen. 4.- El sistema multiplica los pixeles de la imagen a color por una fórmula Matemática. 5.- Genera la imagen entrenada.
Extensiones	5.1.- Imagen entrenada se almacena en una Base de datos local.
Requisitos Especiales	----------
Frecuencia de Ocurrencia	Muy alta.

Tabla 2.3 Especificación de uso Convertir Imagen a Grises

8 Manual usuario

8.1 Interfaz principal

A continuación se explicará de manera completa y detallada cada función en el Sistema de Reconocimiento Facial, tales como cada ventana de éste, sus botones y configuraciones que se pueden escoger.

Primero que todo empezaremos hablando de la ventana principal, ésta contiene una pantalla de Windows en donde se enciende la cámara web, esta pantalla nos proporciona la detección y el reconocimiento del rostro de la persona, véase la figura 12.1 el punto número 1. El punto número 2 nos proporciona un frame del rostro de la persona en tiempo real por medio de una imagen ya entrenada. El siguiente punto es a caja de texto en donde se podrá visualizar los porcentajes de reconocimiento otorgados a la persona en frente a la cámara web, siendo el mayor de éstos el que se le asignará a la persona.

El punto número 4 corresponde a los Parámetros de detección, en donde encontramos configuraciones para el tamaño mínimo del rostro a buscar, el número de vecinos , o sea número mínimo de vecinos antes de considerar un positivo, en cada iteración se obtiene un conjunto de patrones positivos y aquellos que se encuentran solapados se unifican en uno solo. Este número entero especifica cuántos solapamientos han de considerarse para denotar ese patrón como positivo. Un número de vecinos de 2 a 5 es lo más utilizado (usando 2 tendremos resultados más rápidos pero menos eficientes) y el factor escala que es número real que indica el factor de escala utilizado a la hora de recorrer la imagen en busca de patrones, en las diversas iteraciones. Un factor de 1.1 implica que tras la primera iteración en busca de patrones, aplicará un offset de un 10% para buscar en una segunda iteración, y así sucesivamente. Este parámetro varía de 1.1 a 1.5 (+10% - +50%, respectivamente), siendo mucho más lento (pero más eficaz) el valor más pequeño.
En este mismo ámbito se encuentra el punto número 5 el cual se refiere a los colores de los parámetros de detección, configuración llamada Recuadro, y el color del nombre de la persona detectada y reconocida, configuración llamada Nombre.

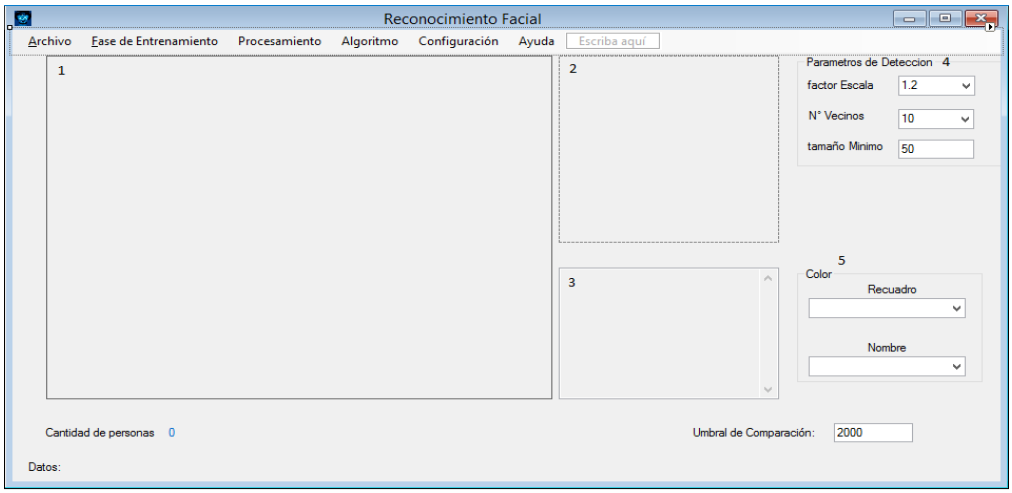

Figura 6.1 Interfaz Principal

Pasando al menú de configuraciones se poseen 6 botones cuyas funciones se explicaran a continuación:

Archivo: En este botón solo se tendrá la opción de Salir, la cual cerrara el programa.

Fase de Entrenamiento: Este botón re direccionará a la Interfaz de entrenamiento, en donde se entrenaran los rostros de las personas, intercambio de colores a escalas de grises, y así poder almacenarlas en la base de datos. En la figura 12.2 se muestra la interfaz como también sus configuraciones.

Algoritmo: En este boto se desplegaran los cuatro algoritmos implementados para el reconocimiento de rostros, 3 de éstos mediante librerías que son: Eigenfaces, FisherFaces y Local Binary Pattern y el otro algoritmo de Eigenfaces generado por los estudiantes. Cada algoritmo reconoce el rostro de las personas gracias a la base de datos local proporcionada por el sistema en donde se almacenan las imágenes y nombres de los individuos. Cabe destacar que para cada algoritmo es la misma carpeta en donde se almacenaran las imágenes entrenadas.

Configuración: Este botón posee la opción de Mostrar la cual al presionarla se generara un check el cual permitirá mostrar o no el punto número 4 y 5 de la configuración principal, o sea los parámetros de detección y Colores.

Ayuda: Este botón abrirá una nueva ventana dentro del programa en donde se podrá apreciar de qué manera funciona el entrenamiento de imágenes, ya sea a través de las funciones de colores como son el rojo, azul y blanco como también el valor de la imagen en cada pixel (véase figura 12.2)

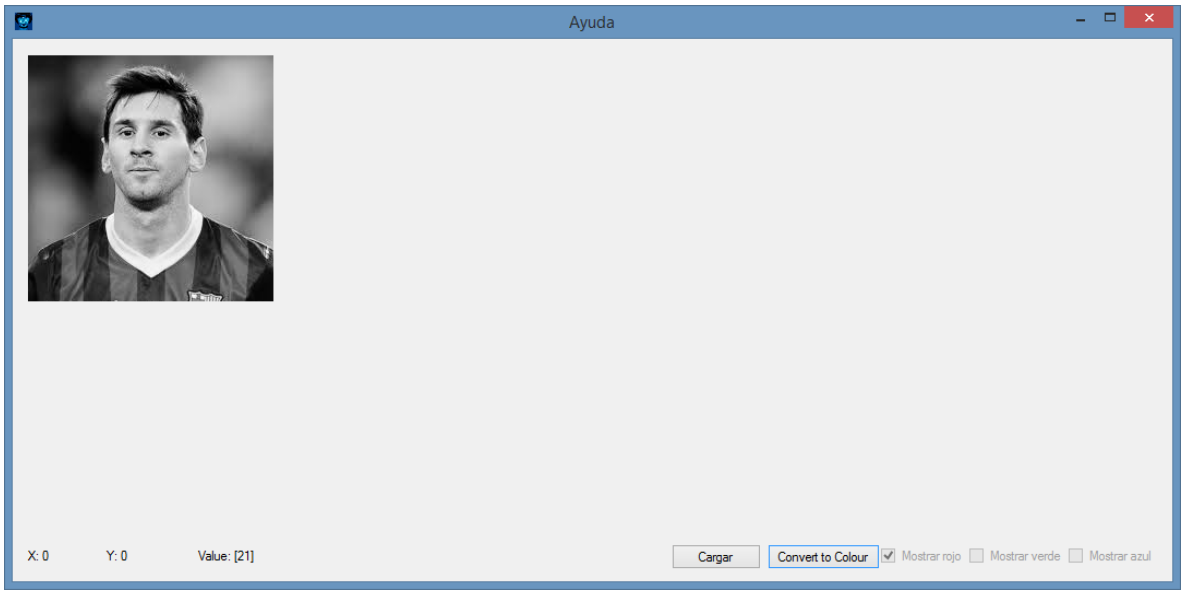

Figura 6.2 Interfaz de ayuda

8.2 Interfaz de entrenamiento

En la interfaz de entrenamiento se producen todos los pasos para entrenar el rostro de la persona. Como se muestra en la figura N° 12.3 es una interfaz muy sencilla y fácil para la manipulación del usuario.

Figura 6.3 Interfaz de entrenamiento

Esta interfaz como la desarrollada anteriormente posee una ventana en donde se cargara la cámara web para poder entrenar la imagen, el método para realizar este proceso primero que todo es detectar el rostro de la persona, una vez detectado clickear el botón Add Image el cual agregara la imagen entrenada a la base de datos local anterior a esto se debe escribir el nombre de la persona en el label Nombre, ya teniendo estos pasos realizados el sistema está en óptimas condiciones para poder realizar el reconocimiento facial.

Algunas opciones extras integradas en esta interfaz es, primero que todo, el poder entrenar hasta 10 imágenes en un solo click lo que ayudara a desarrollar el entrenamiento de forma más rápida y el Delete Data el cual eliminara todos los registros de la base de datos.

8.3 Uso del programa

Para el uso correcto del programa de sistema de reconocimiento facial se deben seguir los siguientes pasos.

1. Se debe verificar que la cámara web pueda detectar el rostro de la persona para tener una correcta posición de la cara del individuo en ésta.
2. Luego debemos entrenar las imágenes, esto se realiza en la interfaz de entrenamiento en donde primero que todo se debe escribir el nombre del individuo a entrenar para luego, teniendo el rostro d la persona detectado en la cámara web, presionar el botón de agregar imagen lo cual almacenara la imagen en tono de grises en una carpeta interna dentro del proyecto, cabe destacar que esta carpeta puede ser modificada dentro del código del proyecto para re direccionarla a cualquier sitio de nuestro computador.
3. En la interfaz de entrenamiento como se explicó anteriormente se pueden ejecutar opciones extras para aso ahorrar tiempo en cada entrenamiento.
4. Terminado el proceso de entrenamiento se procede a elegir a través del botón Algoritmo el tipo de algoritmo que se va a utilizar para el reconocimiento de rostro del individuo, como se explicó anteriormente se tienen 4 algoritmos de siendo cada uno de éstos de alguna forma aptos para el reconocimiento facial.
5. Ya eligiendo el algoritmo a utilizar se podrá visualizar el nombre del individuo, si es que fue correcto o no , encima del recuadro de detección, con esto se podrán realizar estudios relacionados con cada algoritmo de reconocimiento