Universidad de las Ciencias Informáticas

Facultad 1

Trabajo de diploma para optar por el título de
Ingeniero en Ciencias Informáticas

***Título**: Componente para el reconocimiento de rostros mediante el uso de Eigenfaces.*

Autores: Herbert Tamayo Valero
Sairenys Barrios Pérez

Tutores: MSc. Martha Ambruster Crespo
Ing. Ramón Santana Fernández

La Habana, 2013

Declaración de Autoría

Declaramos ser las autoras del trabajo titulado “Componente para el reconocimiento de rostros mediante el uso de *Eigenfaces*” de la Universidad de las Ciencias Informáticas y autorizo a la Universidad de las Ciencias Informáticas los derechos patrimoniales de la misma, con carácter exclusivo.

Para que así conste firmo el presente a los ____ días del mes de ____ del año 2013.

Sairenys Barrios Pérez	Herbert Tamayo Valero
Autor	**Autor**
Ing. Ramón Santana Fernández	MSc. Martha Ambruster Crespo
Tutor	**Tutor**

Agradecimientos

Común

A todas las personas del proyecto Biometría que nos ayudaron y nos dieron aliento para seguir adelante. En especial a Melvis, que trabajó con nosotros hasta altas horas de la noche en la revisión del documento, a Royli por estar siempre dispuesto a responder cualquier pregunta e inquietud, a Adrián (el primo), dispuesto a ayudarnos en todo momento sin decir nunca que no.

Sairenys

A todos los que de una forma u otra hicieron posible que este día se hiciera realidad.

A mis padres Ana (mami), Giraldo (papi), a mis hermanas Sailanys y Erquita y a mi querida abuela Clara, que son las personas que me han guiado por el buen camino de la vida y han sabido hacer de mí una persona correcta.

A la Revolución Cubana por haber formado esta Universidad y haberme dado la oportunidad de realizarme como profesional, de crecer como mujer y de conocer a las personas más maravillosas del mundo, mis amigos, que sin ellos nada de esto hubiera sido posible.

A mi novio Rubén y a mi suegra Senia por su apoyo incondicional y por haber depositado su confianza en mí.

A todas las personas que durante los cinco años de la carrera han estado siempre a mi lado haciendo que mi estancia en la Universidad estuviera llena de buenos recuerdos. A Lieny, Mabel, Adysmarys, Jose Miguel, Daniel, Aynel, Yam, Omar, Nilberto, Herbert, Yindra y en especial a Lieny por ser mi amiga incondicional y soportar muchas de mis majaderías, a Yindra mi amiga de primer año y mi hermana para toda la vida, a Herbert mi compañero de tesis al cual le agradezco por su paciencia y su amistad.

Herbert

A mis padres, por darme la oportunidad de vivir, por su gran cariño, por demostrarme lo mucho que me quieren, por sus consejos y apoyo infinito.

A mi novia por todo su amor, por ser mi amiga, por obligarme a ser cada día mejor, por saber comprenderme y apoyarme en todo momento.

A mis hermanos por quererme tanto y ayudarme siempre en todo.

A mi familia, porque de una forma u otra, siempre están a mi lado.

A mi suegra, por su cariño, por aceptarme en su familia incondicionalmente, por tenerme presente en todos estos años.

A mi padrastro, por querer tanto a mi madre, por apoyarme siempre sin decir no.

A mis amigos de toda la vida Geyler, Francisco y Yunior por no defraudarme nunca, por creer siempre de que esto era posible.

A mis amigos y hermanos de universidad, y que será así para toda la vida, Nilberto y Royli, por enseñarme y demostrarme el concepto de ser buen compañero, por compartir conmigo los buenos y malos momentos en todos estos años de universidad, por estar siempre disponibles cuando los necesité.

A mis amigos, Daniel, Yoevis, Jose, Aynel, Juan, Isidro, Tayler por su apoyo constante, sincero y desinteresado. Por todos los buenos y malos momentos que hemos pasado.

A la amiga más especial del mundo, Lissy, gracias por en tan poquito tiempo llegar y quedar para siempre, gracias por tu comprensión, por tu cariño y tu apoyo incondicional.

A Yuniel, por tan buenos consejos, por toda la ayuda depositada para la realizacion de este trabajo.

A mi compañera de tesis Sairenys, por confiar en mí, por su compromiso y dedicación, fue un placer hacer este trabajo contigo.

A todos los que confiaron en mí y a los que de una forma u otra ayudaron a la realización de este trabajo.

Dedicatoria

Le dedico el presente trabajo a la persona más importante de mi vida, mi querida madre: Ana Aurora Pérez Venega, por estar siempre a mi lado apoyándome en mis decisiones y aconsejarme cada vez que lo necesito, sabiendo ser madre, padre, hermana y amiga.

A mi hermana Sailanys, por darme todo su apoyo, su amor y dedicación, convirtiéndose en una segunda madre para mí.

A mi hermana Erquita y a mi abuela Clara que me han acompañado la vida entera, y han sabido hacer de mí una persona alegre en todo momento.

Por último a mi querido abuelo Yeyo que está siempre en mi pensamiento y con el que me hubiera gustado compartir este momento.

Sairenys

A mi mamá, tu que fuiste la principal artífice de este sueño, a ti te regalo este triunfo.

A mi novia que es lo más lindo que me ha pasado en la vida. Gracias por confiar en mí.

Son lo más grande que tengo, por eso les dedico este momento de mi vida.

Herbert.

Resumen

Un sistema de reconocimiento facial permite identificar unívocamente a una persona, mediante la comparación de determinadas características faciales extraídas de una imagen o una fuente de vídeo. Existen diversas técnicas para la implementación de estos sistemas, como el emparejamiento de plantillas, las aplicaciones de transformadas y la geometría de la cara.

En el Centro de Identificación y Seguridad Digital (CISED) de la Universidad de las Ciencias Informáticas (UCI), se desarrolla un sistema de reconocimiento facial, que tiene como objetivo dar la posibilidad de realizar la identificación de un individuo a partir de la selección de una de las técnicas antes mencionadas. En el presente trabajo se muestra el desarrollo de un componente para el sistema de identificación del CISED, que permite extraer las características distintivas de las personas a través de la técnica aplicación de transformadas, para la representación de imágenes mediante *Eigenfaces*. Una de las principales ventajas que presentan estos tipos de sistemas, es que tienen el mayor compromiso entre complejidad, rapidez de ejecución y resultados.

El documento recoge los resultados de la investigación realizada, haciendo un estudio de las principales características de los sistemas de reconocimiento facial. Se explica la arquitectura y el diseño del componente propuesto. Se describen las herramientas, tecnologías utilizadas y los artefactos generados en el proceso de desarrollo.

Índice

Índice de Figuras

Índice de Tablas

Introducción

El reconocimiento automático de rostros humanos es uno de los desafíos más importantes en el ámbito de la informática. Los seres humanos están acostumbrados a reconocerse usando los rasgos físicos, en especial, los faciales que posee cada individuo de manera particular. Los avances en el campo de la computación en las últimas décadas, permiten realizar reconocimientos similares de forma automática.

El desarrollo del reconocimiento facial comienza en los años 60 con el primer sistema semiautomático implementado para este fin. El mismo requería del administrador para localizar rasgos (ojos, orejas, nariz y boca) en las fotografías. En los años 70, los señores Goldstein, Harmon, Lesk (1), usaron 21 marcadores subjetivos específicos tales como: el color del cabello y grosor de los labios, para automatizar el reconocimiento facial. En 1988 Kirby y Sirobich aplicaron el Análisis de Componentes Principales (PCA), una técnica estándar del álgebra lineal, al problema del reconocimiento facial (2). En 1991 Matthew Turk y Alex Pentland, publican "*Eigenfaces for Recognition*" en "*Journal Cognitive Neuroscience*", donde se planteaba que el reconocimiento facial en tiempo real, era posible utilizando las técnicas *Eigenfaces*, un descubrimiento que permitió implementar sistemas automatizados de reconocimiento facial en tiempo real (3). Para el transcurso de los años 90 se produce una gran explosión en este campo, creando tecnologías masivas, más económicas y al alcance de la mayor cantidad de usuarios, lo que introduce a los sistemas biométricos en un sinfín de aplicaciones que se utilizan día a día.

En la actualidad, la tecnología de reconocimiento facial está siendo utilizada para combatir el fraude de pasaportes, soporte al orden público, en la identificación de niños extraviados, control de acceso, autentificación de usuarios, identificación de usuarios, ingreso a páginas web, control de asistencia, sistemas de seguridad, etc.

Las aplicaciones biométricas en su mayor escala, son producidas por países del primer mundo, donde generalmente existe un gran desarrollo del *software*. Cuba, a pesar de ser un país subdesarrollado y bloqueado, no está ajena al progresivo auge de las aplicaciones biométricas. En el año 2004 fue creado en el país el Centro de Aplicaciones de Tecnologías de Avanzada (CENATAV), dando paso a un grupo de investigaciones teóricas y aplicadas que tienen como misión fundamental: asimilar, desarrollar e introducir en la práctica social, los aspectos más novedosos de la teoría y la práctica del Reconocimiento de Patrones (RP) y Minería de Datos (MD) (4). Desde su surgimiento han realizado investigaciones relacionadas con la biometría, donde se han hecho una serie de

publicaciones de diferentes temas, tales como: Reconocimiento automático de rostros usando FPGAs (del inglés *Field Programmable Gate Array*) (49), Estado del arte sobre los métodos de detección de características locales, Estado actual de los algoritmos de reconocimiento de rostro usando tecnología GPU (del inglés *Graphics Processing Unit*) y muchas más (5).

El desarrollo de *software* en Cuba ha evolucionado en gran medida con el surgimiento de la Universidad de las Ciencias Informáticas, centro de altos estudios creado por idea del Comandante en Jefe Fidel Castro Ruz, en el año 2002. Dentro de la UCI, los múltiples proyectos que se desarrollan han sido organizados en diferentes centros; uno de ellos es el Centro de Identificación y Seguridad Digital (CISED), que cuenta con el Departamento de Biometría, en el que se realizan soluciones informáticas para identificar o verificar la identidad de una persona a partir de sus rasgos físicos y de comportamiento.

Dicho departamento desarrollará un sistema basado en el reconocimiento facial. Entre las técnicas analizadas para su implementación se encuentran la extracción de características geométricas de la cara, los procedimientos de *template matching* (emparejamiento de plantillas) y la aplicación de transformadas. Sin embargo, cada una de estas técnicas es sensible con mayor o menor intensidad a varios factores que atentan contra la calidad de las imágenes a identificar, lo cual provoca un bajo porcentaje de aciertos en el sistema. Algunos de estos factores son las variaciones en las expresiones del rostro, las variaciones en las condiciones de iluminación y las rotaciones en profundidad. Por lo que se hace necesaria la implementación de varias técnicas de identificación de rostros, con el objetivo de formar un sistema robusto que de la posibilidad de escoger que técnica utilizar para la identificación de un individuo.

De acuerdo a la situación problemática planteada anteriormente, el **problema científico** de la investigación queda formulado de la siguiente manera: ¿Cómo implementar un componente para el reconocimiento de rostro que permita mejorar el desempeño del sistema frente a las diferentes variaciones presentadas en las imágenes faciales?

Con el objetivo de dar solución al problema planteado, se define como **objetivo general** de la investigación**:** Desarrollar un componente que permita realizar el reconocimiento de rostros mediante la técnica *Eigenfaces* para el Sistema de Identificación Facial del Departamento de Biometría.

Se derivan así los siguientes **objetivos específicos:**

- Elaborar el marco teórico de la investigación.
- Analizar el algoritmo *Principal Component Analysis* (PCA) para la obtención de las *Eigenfaces*.
- Realizar el análisis y diseño del componente de reconocimiento facial.

- Implementar el componente de reconocimiento facial.
- Validar el componente implementado.

Constituye el **objeto de estudio** el proceso de reconocimiento facial haciendo uso de la técnica *Eigenfaces* y como **campo de acción** el proceso de reconocimiento facial.

Con el propósito de guiar y perfilar el trabajo hacia el alcance de los objetivos trazados, se definieron las siguientes **tareas de investigación**:

- Análisis del estado del arte.
    - Revisión de la bibliografía relacionada con el tema.
    - Análisis de la existencia de otras aplicaciones o soluciones similares.
- Elaboración del diseño teórico de la investigación.
- Elaboración de la propuesta de solución.
    - Descripción de los requisitos funcionales.
    - Análisis de las herramientas necesarias para el análisis, diseño e implementación del componente.
    - Elaboración del análisis y diseño del componente.
- Análisis de las técnicas necesarias para implementar el proceso de detección del rostro.
- Análisis de las técnicas necesarias para implementar el pre-procesado de las imágenes.
- Implementación del proceso de extracción de características utilizando el algoritmo *Principal Component Analysis* (PCA) para la obtención de las *Eigenfaces*.
- Análisis de los algoritmos existentes para implementar el proceso de clasificación.
- Implementación del proceso de reconocimiento de la “cara desconocida”.
- Análisis de las bases de datos de rostros públicas para realizar el entrenamiento y evaluar la efectividad del componente.
- Realización de las pruebas de evaluación para determinar la efectividad del componente implementado.
- Conformación del documento de tesis.

Luego de realizar las tareas y dar cumplimiento a los objetivos que se han planteado, los **aportes prácticos esperados del trabajo** serán los siguientes:

Un componente que permita comparar la imagen de una cara desconocida con todas las imágenes de caras conocidas que se encuentran en una base de datos, con el objetivo de determinar su identidad.

Seguidamente se relacionan los métodos de investigación utilizados para el desarrollo del trabajo de diploma.

Métodos Teóricos:

Histórico-lógico: se utilizó para la compresión lógica del objeto de estudio haciendo un análisis riguroso de sus antecedentes y el proceso evolutivo por el cual han transitado todas las tecnologías relacionadas con el reconocimiento facial.

Analítico-sintético: se utilizó para descomponer el problema de investigación en elementos por separado y profundizar en el estudio de cada uno de ellos, para luego sintetizarlos en la solución de la propuesta.

Inductivo-deductivo: se utilizó para llegar a un grupo de conclusiones sobre lo que se quiere lograr.

Métodos Empíricos:

Entrevista: se utilizó para realizar entrevistas a especialistas y a personal capacitado en el tema con el objetivo de profundizar los conocimientos sobre la identificación de individuos utilizando la técnica *Eigenfaces*.

Estructuración del contenido y una breve explicación de sus partes:

Capítulo I: Fundamentación Teórica.

Es el respaldo teórico de los temas tratados en el informe, está dedicado a exponer los fundamentos teóricos relacionados con el objeto de estudio e incluye un estudio del estado del arte del tema tratado. Se hace un análisis de las técnicas, tecnologías, herramientas y metodologías a emplear durante la investigación.

Capítulo II: Características del sistema.

Se describe el objeto de estudio en términos de negocio, se plantean las principales funcionalidades y se definen los requisitos no funcionales, se realiza una propuesta de la solución de acuerdo a las características que deben cumplir el componente, se define la arquitectura del componente implementado. Se confeccionan las Historias de Usuario (HU).

Capítulo III: Implementación y Prueba.

Se cumplen los elementos de implementación trazados mediante la codificación de la solución propuesta, describiéndose los elementos relacionados con ella. Finalmente se plasman los resultados de las pruebas realizadas al componente.

Capítulo I: Fundamentación teórica

1.1 Introducción

En el presente capítulo se realiza el estudio del estado del arte acerca de los sistemas de identificación de rostros. Se analizan las metodologías, herramientas y tecnologías que intervienen en el proceso de desarrollo del componente.

1.2 Sistemas biométricos

En la actualidad existen sistemas biométricos que basan su acción en el reconocimiento de diversas características, como puede apreciarse en la Figura 1.

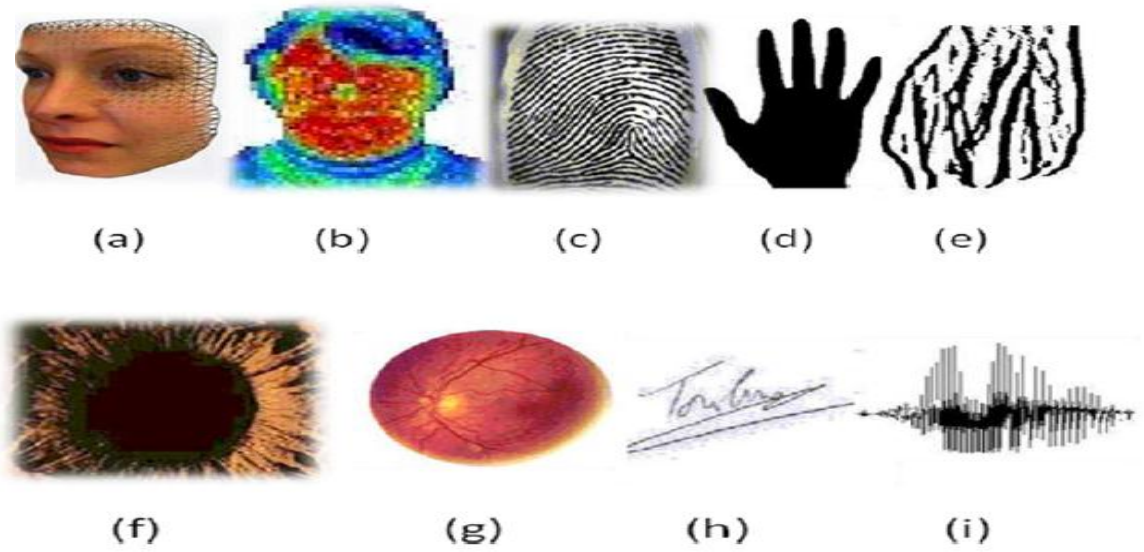

Figura 1. Técnicas biométricas actuales. (a) Rostro, (b) Termograma facial, (c) Huella dactilar, (d) Geometría de la mano, (e) Venas de la mano, (f) Iris, (g) Patrones de la retina, (h) Firma, (i) Voz.

1.2.1 Características de los sistemas biométricos

Los sistemas de más persistencia en el mercado actual son los sistemas biométricos basados en huella dactilar y reconocimiento facial, esto se debe a la presencia de ciertas características (Tabla 1) que le han dado una gran aceptación, como su gran fiabilidad y facilidad de uso.

Tabla 1. Comparación de características de sistemas biométricos. (6)

	Ojo (iris)	Ojo (retina)	Huellas dactilares	Geometría de la mano	Escritura y firma	Voz	Cara
Fiabilidad	Muy alta	Muy alta	Alta	Alta	Media	Alta	Alta
Facilidad de uso	Madia	Baja	Alta	Alta	Alta	Alta	Alta
Prevención de ataques	Muy alta	Muy alta	Alta	Alta	Media	Media	Media
Aceptación	Media	Media	Alta	Alta	Muy alta	Alta	Muy alta
Estabilidad	Alta	Alta	Alta	Media	Baja	Media	Media

1.2.2 Clasificación de las características biométricas

Los sistemas biométricos se clasifican en anatómicos y de comportamiento. En la Figura 2 se muestra la distribución de dichos sistemas.

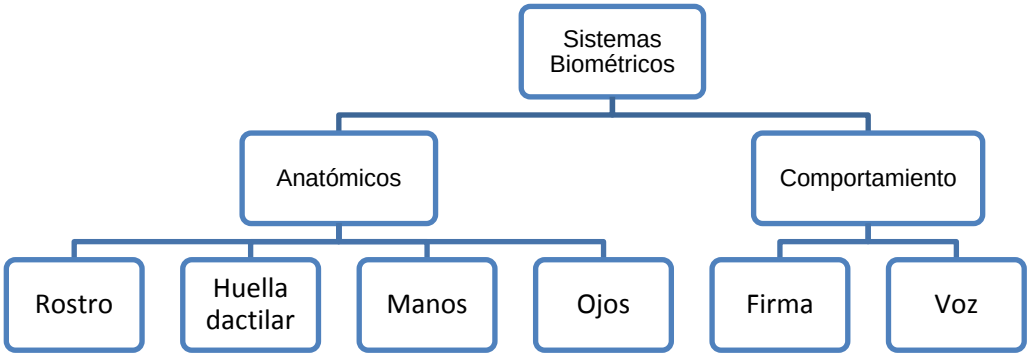

Figura 2. División de las características biométricas.

Al hacer un estudio de la Figura 2, se puede apreciar que los sistemas biométricos anatómicos están determinados por aquellos rasgos que caracterizan físicamente a la persona; los de comportamiento se definen por los rasgos con los que las personas manifiestan su presencia a través de su conducta. La investigación del presente trabajo se sustenta en el sistema biométrico basado en el reconocimiento facial.

1.3 Reconocimiento facial

Un sistema de reconocimiento facial es una aplicación dirigida por ordenador para identificar automáticamente a una persona, mediante la comparación de determinadas características faciales a partir de una imagen o un fotograma de una fuente de vídeo (7).

Los sistemas de reconocimiento facial comparan las características del rostro de una persona tomadas mediante una foto o video, con imágenes de rostros previamente tomadas y almacenadas en una base de datos. Dichos sistemas han repercutido en la sociedad debido a la amplia gama de ventajas que presentan (8):

- ✓ Es un método no intrusivo.
- ✓ El sujeto no siente invadida su intimidad.
- ✓ La persona no tiene que realizar ninguna acción para identificarse.
- ✓ Un sistema de este tipo evitaría la necesidad de llevar documentación (tarjeta, pasaporte, DNI, etc.) para la identificación, bastaría solamente con acercarse a una cámara fotográfica.

1.3.1 Etapas del reconocimiento facial

Los sistemas de reconocimiento facial están compuestos por dos etapas (10):

En una primera etapa que se denomina etapa de **entrenamiento**, se toman de una base de datos un conjunto de imágenes, a las cuales se le aplican las tres primeras fases (Figura 3) del proceso de reconocimiento, con el objetivo de extraer las características que caracterizan a las imágenes de este conjunto.

Una segunda etapa es la denominada, etapa de **prueba**, donde la imagen de la persona que se quiere reconocer pasa por todas las fases (Figura 3) del proceso de reconocimiento, siendo la parte de clasificación la que determina finalmente si la persona es conocida o no, comparando los valores obtenidos en el entrenamiento con la valores extraídos de la imagen de la persona (imagen de prueba).

Un sistema de reconocimiento facial está compuesto por las siguientes fases (9):

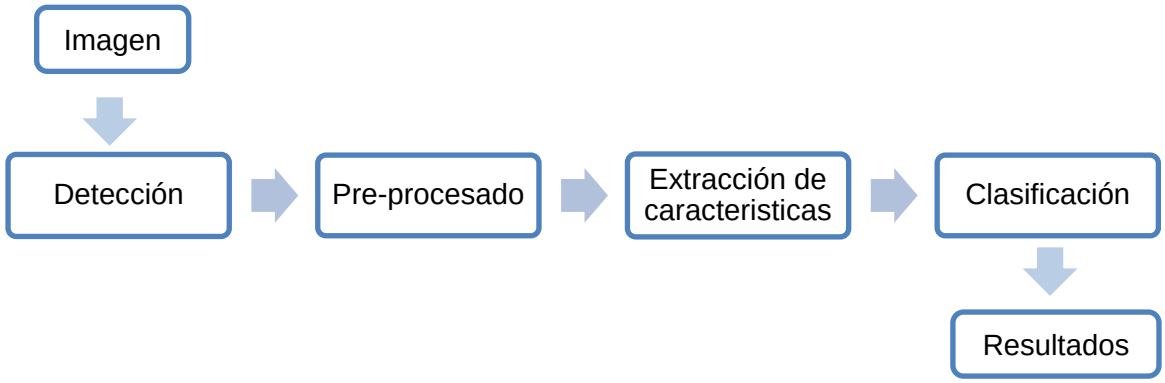

Figura 3. Fases de un sistema de Reconocimiento Facial.

1.3.2 Fases de un sistema de reconocimiento facial

Detección

Esta etapa consiste en la ubicación del rostro de la persona dentro de la imagen capturada, esto se hace con el objetivo de que en la etapa siguiente sólo se incluyan pixeles que pertenezcan a la cara del sujeto. Para realizar la detección existen muchos métodos propuestos en la literatura, de los cuales el más ampliamente usado es el método de Viola-Jones, que consiste en varios detectores en cascada. En cada nivel se analiza una ventana y se decide si la ventana califica o no para pasar al siguiente nivel, si la ventana supera todos los niveles, entonces se determina que la ventana posee una cara, también existen otros métodos que se basan en encontrar el patrón de color de la piel, o en encontrar los ojos y la boca en la imagen. (9).

Figura 4. Detección del rostro.

Técnicas basadas en rasgos

Estas técnicas están basadas en propiedades aparentes del rostro, como el color de la piel y la geometría facial. La detección de la cara se resuelve manipulando medidas de distancia, ángulos y áreas de los rasgos visuales en la imagen. Lo más importante en este tipo de técnicas es decidir que rasgos de la cara interesan para su estudio (11) (12).

Dentro de las técnicas basadas en rasgos se pueden hacer diferentes tipos de análisis:

- **Análisis de bajo nivel**

Son técnicas que trabajan a nivel de pixel. El problema del análisis a bajo nivel es que puede proporcionar información ambigua, por ejemplo, pueden aparecer en la imagen objetos que tengan un color similar al del modelo de color de piel utilizado (9). A continuación se mencionan algunos ejemplos:

- Bordes: analizan las líneas que componen los bordes de la cara y se utilizan para detectar los rasgos faciales.
- Color: minimizan los efectos que tienen las variaciones de iluminación en los resultados del proceso de detección de rostros. Sin embargo, poseen dificultad al localizar ciertos rasgos faciales por presencia de ruido en la imagen o en imágenes con un fondo complejo.
- Video: en una secuencia de video, la localización de objetos es más fácil que en una imagen. Existen técnicas que miden variaciones verticales y horizontales para encontrar los ojos. También existen técnicas que trabajan con la velocidad a la que se mueven ciertas regiones de la escena. En estas técnicas se completa la localización de la cara con un algoritmo que construye una elipse.

- **Análisis de rasgos**

El análisis de rasgos se basa en la geometría de la cara para caracterizar y posteriormente verificar rasgos a fin de evitar la ambigüedad que puede proporcionar el análisis de bajo nivel (9). A continuación se muestran algunos ejemplos:

- Búsqueda de la parte superior de la cabeza: en este análisis se llega a una hipótesis sobre lo que puede ser una posible línea del pelo en lo alto de la frente. Puede resultar muy difícil si la persona tiene pelo cubriendo zonas de la frente.
- Búsqueda de los ojos: se trata de buscar, a partir de la línea del pelo buscada anteriormente, las zonas donde la densidad de gris aumente y disminuya bruscamente en el plano horizontal. Dichas zonas corresponden con las pupilas.

- Uso de plantillas flexibles: se basa en la distancia entre la línea del pelo y el plano de los ojos, para usarlo como medida de referencia para inicializar una plantilla flexible que cubre el resto de los rasgos, como la nariz y la boca. La plantilla trata entonces de ajustarse a dichos rasgos usando una función de costes basada en bordes.

- **Análisis de formas activas**

Se basan en representar la imagen en rasgos de alto nivel para posteriormente interactuar con rasgos locales de la imagen (ojos, brillo) y gradualmente deformarla hasta adaptarla a la forma de los rasgos (9). Algunos de los ejemplos son los siguientes:

- *Snakes* (Serpientes): son empleados para localizar el contorno de la cara. Se basan en la minimización de una función de energía para adaptar el modelo. Se inicializa una *snake* ante una cara inminente, y posteriormente se va cerrando a los contornos de la imagen asumiendo la forma del rostro.
- Plantillas deformables: se usan las *snakes* para encontrar más rasgos faciales además del contorno de la cara. Por ejemplo, se pueden encontrar los ojos usando para las *snakes* un mecanismo de deformación que incluye el cálculo de gradientes de una función de energía.

Técnicas basadas en la imagen

En estas técnicas el objeto de estudio es la imagen misma. Se trabaja directamente con la representación de la imagen a la que se le aplican algoritmos de entrenamiento y análisis (9). Dentro de esta técnica se encuentran los siguientes métodos:

- **Métodos basados en sub-espacio**

Consideran las imágenes de caras humanas como un sub-espacio lineal de un espacio mayor (de todas las imágenes) (13) (14).

Redes neuronales: es uno de los métodos más utilizado en la detección de caras en una imagen, dado el alto porcentaje de aciertos que produce, siendo en algunos casos superiores al 95%. Las redes neuronales se usan generalmente para la clasificación de imágenes según patrones establecidos, por ejemplo, conseguir averiguar qué representa una imagen borrosa o incompleta. La red neuronal se entrena usando un conjunto de imágenes que representan caras de todo tipo (de varias razas, tonos de piel, con y sin gafas, pendientes, posiciones de los labios y ojos, ligeras rotaciones) y otro conjunto de imágenes que no representan caras, de forma que la red neuronal pueda establecer el criterio adecuado

acerca de lo que es una cara y lo que no lo es. La respuesta de la red neuronal ante una imagen de entrada es la de decidir si dicha imagen corresponde o no a una cara, es decir, una respuesta binaria.

- **Métodos estadísticos**

Este tipo de algoritmo no asume ningún tipo de información previa de la tipología de una cara; sino, que a partir de un conjunto de muestras (imágenes de caras y de no caras) de entrenamiento, extraen la información relevante que diferencia un objeto cara de un objeto no cara. Este grupo incluye dos de los métodos más referenciados y con mayor avance en el campo de la detección: *Boosting y Adaboost.*

- *Boosting*: los métodos de *boosting* son algoritmos que en el enfoque supervisado conocen la solución a priori y utilizarán esto para adaptar su comportamiento. La mayoría de los algoritmos de *boosting* consiste en procesos iterativos que aprenden de clasificadores débiles con respecto a una distribución, para finalmente agregarlos a un clasificador final el cual es más fuerte. Cuando se agregan, son almacenados de manera que se relaciona comúnmente con la precisión del clasificador débil. Los ejemplos que son clasificados correctamente pierden peso y los que son clasificados de manera errónea aumentan su peso. Así, los siguientes clasificadores se centran más en los ejemplos que los anteriores clasificaron erróneamente. Existen una variedad de algoritmos de *boosting*; los originales propuestos por Schapire (15) y Freund (16) no eran adaptativos y no podían tomar ventaja completa de los clasificadores débiles. Por tal motivo, Freund y Schapire (17) proponen posteriormente a *Adaboost*, el cual es un algoritmo adaptativo que toma ventaja de los clasificadores débiles para formar un clasificador fuerte.
- *Adaboost*: uno de los trabajos de detección de rostros en tiempo real es el esquema algorítmico desarrollado por Viola y Jones que se basa en el aprendizaje de AdaBoost (*Boosting* adaptativo), para construir un clasificador no lineal. *Adaboost* está diseñado para detectar cualquier objeto y posee además una buena efectividad. El algoritmo clasificador consiste en un árbol de decisión compuesto por una serie de clasificadores débiles, los cuales están en forma de cascada formando una sucesión, lo cual trae como resultado un clasificador fuerte con alto grado de efectividad (18).
- Viola y Jones: fue el primero en ofrecer detección de rostros robusta en tiempo real. Posee dos etapas principales: una de entrenamiento del detector con mayor procesamiento de demanda, y otra de detección, donde se emplea el detector entrenado en la primera etapa sobre cada imagen a analizar. Esta segunda etapa es muy rápida y permite realizar la detección en tiempo real. Partiendo de la relación detección y falsos positivos, este método es comparable al de la red

neuronal, pero se realiza varias veces más rápido. El algoritmo que se utilizará basado en este método es AdaBoost, el cual utiliza una ventana que recorre la imagen comprobando en cada posición la existencia del objeto con el cual fue entrenado, en este caso es un rostro. El tamaño de la ventana está dado por el tamaño mínimo de los rostros que se desean detectar y, una vez finalizado el recorrido de la imagen, es aumentado sucesivamente hasta cubrir el total de la imagen. Finalmente, el clasificador devuelve una secuencia de rectángulos que indican las posiciones de los rostros detectados en la imagen (19) (20) (21).

Pre-procesado

El pre-procesado consiste en intentar compensar todo lo que puede provocar que dos imágenes de la misma cara sean diferentes. Esto incluye normalizar el tamaño, el contraste de la imagen y la rotación. A continuación se detallan los elementos del pre-procesado implementados en el componente (9).

- Tamaño: se normaliza el tamaño de la imagen a 100 x 100 pixeles.
- Rotación: utilizando las posiciones de los dos ojos, se efectúa una rotación de la imagen de manera que los dos ojos queden en un eje horizontal.
- Zona de la cara utilizada: la cara se recorta de manera que no aparezcan zonas del fondo de la imagen ni zonas del pelo, cuyo aspecto es muy variable y perjudicaría la robustez del componente.
- Brillo y contraste: se normaliza el brillo y se incrementa el contraste (se normaliza el histograma).

Extracción

En esta fase se extraen una serie de valores característicos de cada imagen, como pueden ser los coeficientes de algún desarrollo, la salida de un filtro, etc. Independientemente de su origen, estos valores deben intentar caracterizar con la mayor exactitud cada cara (lo que se considera eficiente) y, al mismo tiempo, deben tener capacidad de discriminación. Esto significa que los valores extraídos de las imágenes de una cara y los de las imágenes de otras caras deben formar dos grupos lo más compactos y separados posible (9).

Métodos para la fase de extracción

En esta fase los algoritmos se pueden clasificar en dos tipos como se encuentra a continuación.

- **Métodos holísticos**

Los métodos holísticos forman parte del proceso de detección de la cara de tal forma que utilizan como unidad básica de procesamiento para la entrada al sistema la imagen facial de un individuo. A continuación se describen los métodos más utilizados de esta categoría.

- Análisis de Componentes Principales (PCA): el objetivo de este método es transformar las imágenes de rostros en un grupo pequeño de características, denominadas *Eigenfaces*, que son los componentes principales del entrenamiento inicial de un conjunto de imágenes. El reconocimiento se lleva a cabo proyectando una nueva imagen dentro del sub-espacio formado por las *Eigenfaces*, para su clasificación el rostro se compara por la posición en el espacio característico con las posiciones de individuos conocidos (22) (23) (24).
- Análisis Lineal Discriminante (LDA): la idea central de LDA es obtener una proyección de los datos en un espacio de menor (o incluso igual) dimensión que los datos entrantes, con el fin de que la separación de las clases sea la mayor posible. Es una técnica supervisada, ya que para poder buscar esa proyección, se debe entrenar el sistema con patrones etiquetados. Aplicando el Análisis Discriminante Lineal (*Lineal Discriminant Analysis*) de Fisher, es posible construir una matriz de proyección en la cual la razón entre la dispersión intra-clase y la inter-clase sea máxima. Los resultados muestran que tanto PCA como LDA (a veces referida en la bibliografía como *Fisherfaces*) obtienen un buen rendimiento si las imágenes de prueba son similares a las imágenes de entrenamiento (25) (26).
- Análisis de Componentes Independientes (ICA): es una generalización del método PCA. Consiste en transformar un conjunto de señales observadas en un conjunto de señales estadísticamente independientes llamadas fuentes o componentes independientes del primer conjunto. El principal objetivo de ICA es identificar a partir de las señales observadas, un nuevo grupo de señales subyacentes con significado. Esta representación puede ser usada en distintos ámbitos como la extracción de características y el reconocimiento de patrones. Esta técnica difiere de los análisis en componentes principales en que ICA impone independencia de alto orden (27) (28).

- **Métodos basados en características locales**

Se extraen características locales, como ojos, nariz, boca, etc. Sus posiciones y estadísticas locales constituyen la entrada al sistema. A continuación se describen los métodos más utilizados de esta categoría.

- Patrón Binario Local (LBP): consiste en recorrer la imagen y etiquetar los pixeles mediante un umbral de diferencia entre el pixel central y sus vecinos, considerando el resultado como un número binario. LBP se utiliza para la descripción de la imagen de una cara. Se construyen varios descriptores locales que se combinan en un descriptor global (29).
- Correspondencia entre agrupaciones de grafos elásticos: *Elastic bunch graph matching* (EBGM): el proceso fundamental en el sistema, consiste en la correspondencia elástica entre grafos. Mediante este planteamiento, un grafo modelo (un grafo derivado de una imagen facial con posiciones de nodos adecuadas) se compara con la imagen de prueba. Aquí los nodos del grafo modelo se colocan de forma aproximada en la imagen, se extraen los jets (conjunto de características locales) de estos puntos y se calcula la similaridad entre el grafo modelo y el grafo imagen así construido. Esta similaridad se optimiza variando las posiciones de los nodos en la imagen (30)

Comparación de los métodos principales de reconocimiento facial

Tabla 2. Comparación de los métodos de reconocimiento facial.

Métodos	Ventajas	Desventajas
Análisis de Componentes Principales (PCA)	Reduce datos necesarios a 1/1000 de los datos presentados. Buen rendimiento si las imágenes de prueba son similares a las imágenes de entrenamiento. Su porciento de aciertos está por encima del 95%.	Requiere el rostro de frente y la no existencia de variaciones en la iluminación.

Análisis Lineal Discriminante (LDA). *Fisherfaces*	Proyección de datos en espacio de menor o igual dimensión que los entrantes. Por lo que presenta buen rendimiento si las imágenes de prueba son similares a las imágenes de entrenamiento	No busca minimizar error de representación cometido. Requiere de imágenes de prueba similares a las imágenes de entrenamiento para tener un buen rendimiento.
Patrón Binario Local (LBP)	Ofrece mejores resultados en las imágenes con problemas de iluminación. Porciento de aciertos del 99%.	Solo puede considerar pequeñas y fijas áreas espaciales (3x3), quedando imposibilitado así de capturar texturas mayores.
Correspondencia entre agrupaciones de grafos elásticos (EBGM).	Menos sensible frente a cambios de iluminación, orientación y gestos que los demás antes vistos. Se obtienen buenos resultados utilizando esta técnica y los tiempos de cómputo no son elevados.	La dificultada de este método es el requerimiento de la precisa localización del punto de referencia.

Luego de esta comparación se ha decidido a nivel de proyecto utilizar el método PCA para la implementación del componente de identificación facial, ya que, está basado en propiedades estadísticas de representaciones vectoriales y ha encontrado aplicación en los campos del reconocimiento de rostros y en la compresión de imágenes. Es una técnica común para encontrar patrones o rasgos principales finitos en problemas de dimensión infinita. Está basado además, en el análisis de datos multivariantes con el objetivo fundamental de convertirlos en un espacio de dimensionalidad reducida. PCA es un método general de análisis de datos y se aplica en el reconocimiento de rostros con alguna variación en el método, llamándolo *Eigenfaces* (31).

El método consiste en los siguientes pasos (32):

1. Recopilar el conjunto de datos.
2. Obtener el conjunto promedio de los datos.
3. Calcular la matriz de covarianza.

4. Seleccionar los componentes, formar un vector característico y calcular los vectores propios (*eigenvectors)* y valores propios (*eigenvalues*) de la matriz de covarianza.

Clasificación

En esta etapa se comparan los valores característicos de la imagen de prueba (la que se quiere reconocer) con los de las imágenes de entrenamiento y se calcula una medida de semejanza. Los métodos van desde la distancia euclidiana (considerando que el conjunto de valores característicos forman un vector como se muestra en la Figura 5) a otros mucho más sofisticados. La imagen de entrenamiento que más semejante sea a la de prueba se considerará que es de la misma persona (34).

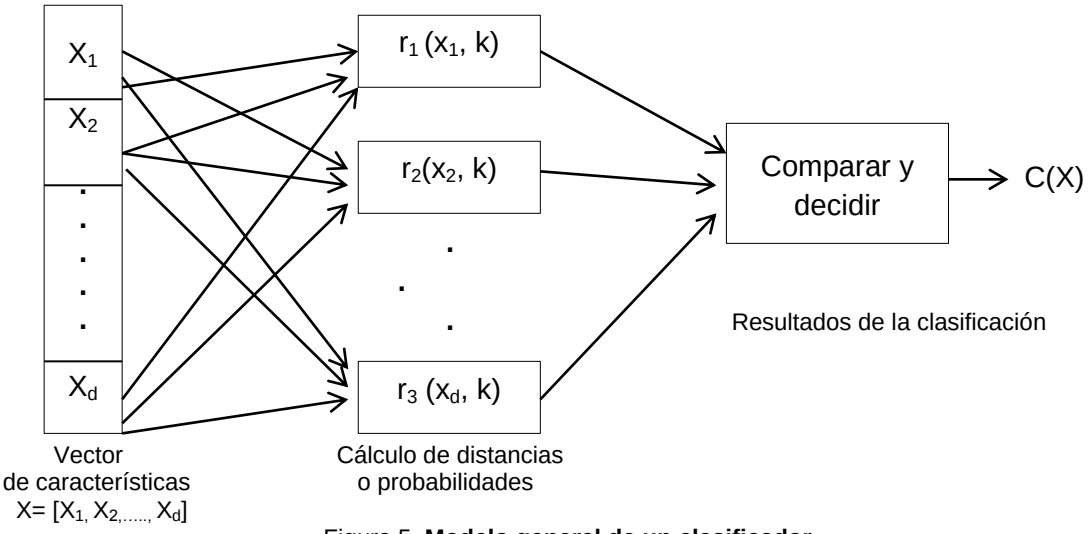

Figura 5. **Modelo general de un clasificador.**

Clasificadores basados en la distancia

Entre los diferentes tipos de clasificadores, se encuentran los clasificadores basados en la distancia entre los vectores característicos de la persona. A continuación, se exponen diferentes clasificadores basados en distancias, que permiten por su simplicidad y variedad una aproximación didáctica y adecuada al problema de la clasificación (34).

- Clasificador de distancia euclidiana.

El clasificador de distancia euclidiana conocido también como clasificador de distancia mínima, ha sido el más usado hasta el momento en tareas de reconocimiento. Supone que las clases de objetos son de naturaleza determinística, es decir, los elementos que componen una clase se representan por un único

vector llamado prototipo de clase. Este clasificador emplea la distancia euclidiana como medida de similitud entre un vector de entrada y un conjunto de vectores prototipo. Sobre la base de esta distancia, el clasificador determina la clase a la cual el patrón de entrada será asignado.

El funcionamiento de un clasificador euclidiano consiste en las siguientes etapas:

1. Dado un vector Ω (el vector característico de una imagen de prueba), calcular las distancias euclidianas del Ω a cada uno de los vectores característicos de las M imágenes de entrenamiento $\Omega_1, \Omega_2, \Omega_3, \ldots\ldots, \Omega_M$.
2. Asignar Ω a la imagen Γ, tal que la distancia $d_E(\Omega,\Omega_M)$ es mínima (34).

   La Figura 6 muestra el diagrama de bloques de este proceso.

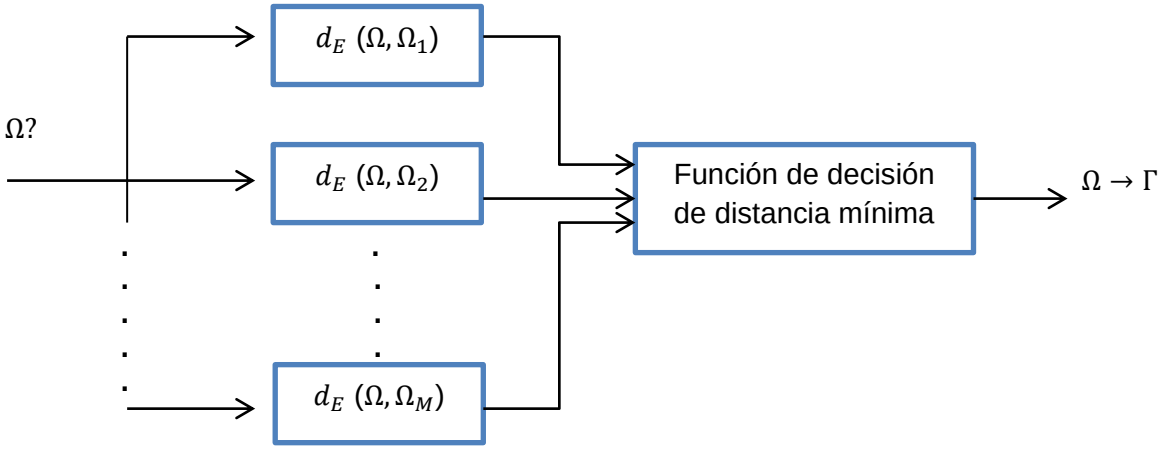

Figura 6. Esquema del clasificador euclidiano.

- Clasificador de k-vecinos más cercanos.

En aquellos casos en los que la región de clasificación no es lineal ni sigue una curva cónica puede usarse el clasificador de los k-vecinos más cercanos. Intuitivamente este clasificador supervisado apriorístico es uno euclidiano con múltiples centroides por cada clase. Un patrón se clasifica como perteneciente a la clase a la que pertenezca la mayoría de los k centroides más próximos. La principal ventaja de este clasificador reside en su potencia. Su principal desventaja estriba en la dificultad para la elección de k, para la determinación del número de centroides por clase y su disposición, tareas que en

general son eminentemente empíricas. Además, el tiempo de cálculo puede ser significativamente mayor si el número de centroides llega a ser muy elevado (34).

- Support Vector Machines (SVM)

Las Support Vector Machines (SVM) son un tipo de clasificadores de patrones basados en técnicas estadísticas de aprendizaje propuestas por Vapnik y sus colaboradores en 1992. Las SVM están a la cabeza de los métodos de clasificación por permitir construir fronteras de decisión flexibles y por su buena capacidad de generalización. El método de las SVM permite abordar de manera general la resolución de problemas de clasificación y de regresión. La idea consiste en transformar el conjunto de vectores de entrada $X = \{X_i | X_i = (X_{i1}, X_{i2}, \ldots\ldots, X_{in})\}$ patrones n-dimensionales) en otro conjunto de vectores Y de una dimensión más alta (incluso de dimensión infinita), en los que el problema pueda solucionarse linealmente (34).

1.4 Factores negativos para el sistema de reconocimiento facial

Existen cinco factores principales que pueden afectar la eficiencia del sistema al momento de tomar las imágenes ellos son (35):

- La iluminación (Figura 7): este factor puede considerarse como el más problemático de todos. Solo con una pequeña variación en la iluminación de las imágenes de una misma persona, el sistema podría confundirse a tal grado de considerar que es otra persona.

Figura 7. Cambios de iluminación.

- Distancia (Figura 8): la distancia puede cambiar al tomar fotos, es por eso que se corre el riesgo de meter ruido, como por ejemplo, que no solo aparezca el rostro sino también los hombros, brazos, etc. Haciendo más difícil la comparación.

Figura 8. Cambios en la distancia.

- Fondo de la imagen (Figura 9): si el fondo no fuera uniforme, el sistema puede confundir el rostro y tomar la imagen completa con todo lo que tenga alrededor.

Figura 9. Cambios en el fondo de la imagen.

- Cambios faciales (Figura 10): si las fotos fueron tomadas en distintos días, puede ser que la persona cuando se tomó las imágenes de entrenamiento utilizaba un tipo de peinado, y cuando se realizó la prueba tenía otro e incluso puede ser que llevara gorra.

Figura 10. Cambios faciales.

- Orientación del rostro (Figura 11): esto se debe a que una persona puede moverse al ser tomada la foto, por ejemplo por alguna distracción ocasionada en el momento.

Figura 11. Cambios en la orientación del rostro.

1.5 Métodos y técnicas empleadas en el componente

En el presente epígrafe se describen las diferentes técnicas y métodos utilizados para el desarrollo del componente, para ello se realizó un profundo estudio que conllevó a la posterior implementación, buscando la forma de obtener un producto lo más robusto posible con la calidad requerida. A continuación se muestra la valoración correspondiente a cada técnica o método de las fases del proceso de reconocimiento facial.

1.5.1 Fase de detección

Para esta fase se determinó la utilización del método de Viola y Jones por ser el más utilizado y con más nivel de aceptación en la literatura, además de la posibilidad que brinda las bibliotecas de clases EMGU CV de contar con un clasificador de objetos basado en el método *AdaBoost*, el cual se utilizó para llevar a cabo el proceso de detección del rostro del sistema de identificación, siendo así la función que permitirá desarrollar dicho paso: *CascadeClassifier(srting dirAchiXml)*, donde el parámetro *dirAchiXml* es la dirección del archivo XML (Lenguaje de Marcas Extensible) por el cual se va a entrenar. Esta función retorna un rectángulo que representa la posición y el área que ocupa el rostro detectado dentro de la imagen. A partir de este proceso, el rostro es extraído y almacenado como una imagen para continuar a la siguiente etapa del reconocimiento. A continuación se muestra un conjunto de imágenes antes y después de aplicar la fase de detección.

Figura 12. Conjunto de imágenes antes de la detección.

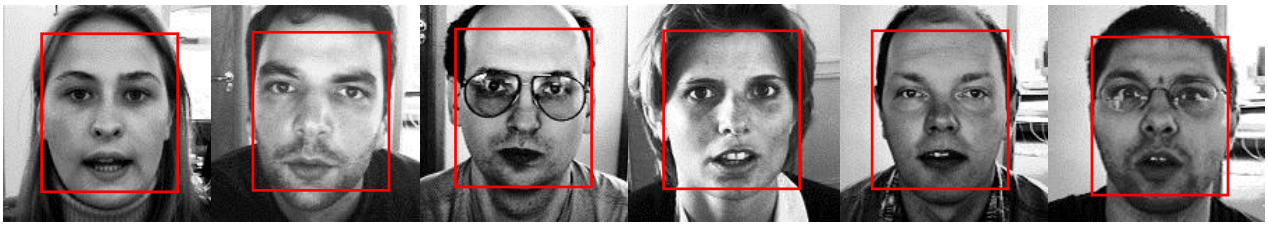

Figura 13. Conjunto de imágenes después de la detección.

1.5.2 Fase de extracción

Para esta etapa haciendo uso de las bibliotecas de clases EMGU CV, se utilizó la técnica de *Eigenfaces* basada en el método PCA. Su principal ventaja se basa en la facilidad de implementación y en su bajo costo computacional. Permite además la creación de sistemas en tiempo real. A continuación se muestran las primeras seis *Eigenfaces* más significativas obtenidas por el componente al aplicar el método PCA para un grupo de 230 imágenes de entrenamiento.

Figura 14. *Eigenfaces*.

1.5.3 Fase de clasificación

En esta etapa se utilizó un clasificador basado en la distancia, específicamente el clasificador de distancia euclidiana. Este clasificador recibe como entrada un vector característico que es comparado con los vectores característicos de las imágenes de entrenamiento (proyecciones). Una vez concluido este proceso, se indica cual es la persona correspondiente al vector de menor distancia euclidiana, siempre que esa distancia sea menor que un umbral definido.

1.5.4 Reconocimiento de rostros mediante Eigenfaces (PCA)

Por lo engorroso que suele ser el entendimiento de la técnica *Eigenfaces* (PCA) en la utilización de sistemas o componentes de reconocimiento facial, en este caso un componente, se fundamenta a continuación todo el proceso matemático que se oculta detrás de dicha técnica.

La técnica *Eigenfaces* es el PCA aplicado en imágenes faciales sobre un espacio representado por un conjunto de rostros grande. Los vectores característicos significativos son conocidos como *Eigenfaces*

porque son los vectores propios (componentes principales) del conjunto de rostros. En términos matemáticos se requiere encontrar los componentes principales de la distribución de rostros, o los vectores propios de la matriz de covarianza del conjunto de imágenes faciales, tratando una imagen como un punto (vector) en un espacio dimensional infinito. Estos vectores propios se pueden considerar como un conjunto de vectores característicos que juntos caracterizan la variación entre las imágenes faciales. Se puede ver, a cada uno de los vectores propios, como un pequeño espectro o "fantasma", del rostro original, al cual se le llama *Eigenface* (31)(33).

Para todo el proceso matemático que sigue a continuación se hace referencia a las bibliografías (9) (10) (36) (37)

Toda imagen digital puede ser representada por un vector dentro de un espacio vectorial (denominado "espacio cara") cuya dimensionalidad depende de la imagen en cuestión, es decir, si la imagen tiene $\boldsymbol{m}$ pixeles de ancho y n pixeles de alto, la dimensionalidad del espacio cara será de $m\ x\ n.$

Cuando se trata de caracterizar rostros el "espacio cara" es muy redundante, esto se debe a que cada pixel en una imagen digital está muy correlacionado con los demás pixeles. Sin embargo cuando un grupo de imágenes de rostros son convertidas a una forma vectorial, todas se agrupan en un sub-espacio del "espacio cara", esto se debe a que las imágenes de rostro comparten ciertas características globales (la posición del rostro dentro de la imagen, el tamaño que ocupa el rostro dentro de la imagen, etc.).

Conjunto de imágenes de entrenamiento

Considere el conjunto de imágenes de entrenamiento $\Gamma_{F=\{\Gamma_1,\Gamma_2,\ldots,\Gamma_M\}}$, donde M es el número de imágenes en el conjunto, considere que el tamaño de las imágenes es de $N = m \times n$ pixeles.

Cada imagen es reorganizada como un vector de dimensión N cuyo valor es construido como la concatenación de cada una de las filas de la imagen, formando así una matriz de $m\ x\ n$.

El vector que define la media del conjunto de entrenamiento está dado por:

$$\Psi = \frac{1}{M}\sum_{k=1}^{M}\Gamma_k$$

Posteriormente se obtiene un nuevo conjunto de vectores al calcular la diferencia entre la i-ésima imagen o fotografía y el rostro promedio:

$$\Phi_i = \Gamma_i - \Psi \quad (i = 1,2,\ldots, M)$$

Denotemos una matriz A como aquella matriz que captura cada vector Φ_i, de acuerdo a lo anterior sus dimensiones serán $N\ x\ M$.

$$A = [\Phi_1, \Phi_2, \ldots\ldots, \Phi_M]$$

Matriz de covarianza

Una vez obtenida la matriz A, se procede a calcular la matriz de covarianza C del conjunto de datos, que está dada por la fórmula:

$$C = AA^T$$

Las dimensiones de esta matriz serán de $N\ x\ N$

Cálculo de los vectores propios y valores propios

Dado que la matriz de covarianza C es muy compleja debido a sus dimensiones, es conveniente considerar la matriz L de dimensiones $M\ \times\ M$ dada por:

$$L = A^T A$$

Los vectores propios v_i y los valores propios μ_i de L están definidos por la siguiente ecuación:

$$A^T A v_i = \mu_i v_i$$

Al multiplicar la ecuación anterior por A:

$$AA^T A v_i = A\mu_i v_i$$

Dado que μ_i es un escalar y $C = AA^T$, la ecuación anterior puede escribirse como:

$$CAv_i = \mu_i A v_i$$

De donde podemos concluir que $v_i = Av_i$ es uno de los vectores propios de C, sin embargo solo se pueden obtener los primeros M valores propios y vectores propios de C (debido a las dimensiones de L),

pero, si $M < N^2$ solo habrá $M - 1$ vectores propios significativos (los vectores propios restantes tendrán valores propios asociados igual a cero).

Dado que los vectores propios de C constituyen la base vectorial del "espacio cara" son denominados *Eigenfaces.*

Eigenfaces

Una característica interesante de los valores propios asociados a los *Eigenfaces* de C, es que estos siempre tendrán una distribución exponencial.

Aunque es posible caracterizar el "espacio cara" a partir de los $M - 1$ *Eigenfaces de* C, no todos ellos aportan la misma cantidad de varianza, debido a esto, es posible eliminar los *Eigenfaces* menos significativos, sin que esto signifique una gran pérdida de información. Los *Eigenfaces* son vectores que apuntan en la dirección de máxima varianza, el valor de varianza que el *Eigenface* representa es directamente proporcional al valor de su valor propio asociado, en consecuencia, entre más grande sea el valor propio más grande será la varianza que representa la *Eigenface.*

Podemos concluir que solo es necesario un reducido número (M') del total de *Eigenfaces* para caracterizar adecuadamente el "espacio cara".

Proyección de las imágenes de entrenamiento sobre el "espacio cara"

Una vez que el "espacio cara" ha sido caracterizado por los M' *Eigenfaces,* el conjunto de imágenes de entrenamiento debe ser proyectado sobre este espacio vectorial, con el objetivo de obtener la contribución de cada *Eigenface* para representar cada una de las imágenes del conjunto de entrenamiento. La contribución de un *Eigenface* para representar una imagen se define como:

$$\omega_i = v_i^T \cdot \Phi = v_i^T \cdot (\Gamma - \Psi), \; i = 1,2,\ldots, M'$$

Nótese que esta operación está definida como el producto punto entre 2 vectores (un *Eigenface* y el vector que define la desviación de la media de una imagen de entrenamiento).

A partir de la contribución de cada *Eigenface* para la representación de una imagen, es posible construir los vectores característicos Ω (cuyas dimensiones serán $1 \times M'$), que son la representación de una imagen dentro del "espacio cara".

$$\Omega_k = (\omega_1, \omega_2, \dots, \omega_{M'}),\ k = 1,2,\dots, M$$

Vemos entonces que a través de PCA es posible representar una imagen como un simple vector de dimensiones $1 \times M'$ en el "espacio cara", al contrario de una representación $N = m \times n$.

Proceso de reconocimiento

Considere la imagen de prueba Γ_T **,** para efectuar el reconocimiento de esta imagen es necesario comparala con las imágenes en el conjunto de entrenamiento, debe ser proyectada sobre el "espacio cara". La desviación de la media de Γ_T se define como:

$$\Phi_T = \Gamma_T - \Psi$$

La contribución de cada *Eigenface* para representar $\boldsymbol{\Gamma_T}$ esta dada por:

$$\omega_{Ti} = v_i^T \cdot \Phi_T = v_i^T \cdot (\Gamma_T - \Psi),\quad i = 1,2,\dots, M'$$

El vector característico que representa a Γ_T en el "espacio cara" está definido por:

$$\Omega_T = (\omega_{T1}, \omega_{T2}, \dots, \omega_{TM'})$$

Una vez que se ha obtenido el vector característico Ω_T, este puede ser comparado con los vectores característicos del conjunto de entrenamiento utilizando clasificadores como los descritos en el epígrafe 1.3 (en la Clasificación).

1.6 Ejemplos de *software* para el reconocimiento facial

1.6.1 Internacional

Los sistemas de reconocimiento facial a nivel internacional han alcanzado un gran auge dentro de la sociedad actual. Debido a esto se han creado en el mundo disímiles empresas dedicadas al desarrollo de aplicaciones para la seguridad social. A continuación se muestran algunas aplicaciones:

- Avatar Street Meeting: la Universidad Nacional de Colombia, Sede Medellín, desarrolló el *software* que pretende implementar los avances del reconocimiento facial pertenecientes al área de visión artificial. El proceso inicial de manera resumido es simple: se pretende que una persona con Avatar instalado en su dispositivo móvil pueda, "escanear a alguien" sea por vídeo en tiempo real o por una foto tomada segundos antes, y este muestre información aumentada de la persona. Para ello

se debe implementar una técnica biométrica de reconocimiento facial adecuada. En este caso optaron por trabajar con la técnica *Eigenfaces*, ya que es trivial su implementación en lenguajes destinados a los dispositivos móviles (35).

- Batur, Flinchbaugh y Hayes en el 2003 realizaron la implementación de un sistema DSP TMS 320C6416 de la técnica de PCA que podía reconocer un rostro en alrededor de 3.5 segundos (36).
- Google desarrolló un producto llamado Picasa y una herramienta de búsqueda de reconocimiento de rostros. El *software* cuenta con una interfaz que permite subir álbumes de fotos en Internet para incorporarla a un banco de imágenes. La herramienta busca e identifica automáticamente el rostro de cualquier persona que se le indique, y muestra todas las imágenes donde crea que ésta se encuentre (37) .

1.6.2 Nacional

En Cuba, se han creado empresas e instituciones educacionales con la visión futurista de la informatización de la sociedad. A continuación se presentan dos grandes empresas dedicadas al desarrollo de productos para la identificación biométrica.

- **CENATAV[1]:** empresa dedicada fundamentalmente a asimilar, desarrollar, e introducir en la práctica social, los aspectos más novedosos de la teoría y la práctica del Reconocimiento de Patrones (RP) y Minería de Datos (MD). Desde su surgimiento han realizado investigaciones relacionadas con la biometría desarrollando sistemas de reconocimiento facial basado en diferentes técnicas, uno de ellos está basado en la técnica LBP[2]. Este algoritmo está siendo usado en diferentes aplicaciones como la verificación de identidad de las personas en las fronteras del país y en la emisión de documentos de identidad (pasaportes, carnet de identidad, etc.) para evitar suplantaciones. Desarrolló además un método que permite a partir de una secuencia de imágenes de una persona, obtenidas con o sin su conocimiento, determinar en tiempo real la mejor imagen del conjunto que cumpla con los requisitos para realizar un buen reconocimiento automático de rostros. La mayor utilidad del método propuesto es en el reconocimiento de rostros encubierto, ya que en este caso los individuos no tienden a permanecer en una posición frontal (4).

[1] CENATAV: Centro de Aplicaciones de Tecnologías de Avanzada.
[2] LBP: Local Binary Pattern.

- **DATYS[3]:** empresa dedicada al desarrollo de aplicaciones informáticas en diversas líneas de negocios, a las que aporta soluciones con sus productos en diferentes áreas y una de ellas es la identificación biométrica, destacando en ella la rama del reconocimiento facial. Hoy en día uno de sus productos más significativos, es el sistema llamado Biomesys Face**,** dedicado a la identificación de rostros (38).
- **UCI:** institución educacional dedicada a la formación de miles de jóvenes universitarios, formada por diferentes facultades, todas con la labor de desarrollar diferentes proyectos con carácter nacional e internacional. Cada facultad está organizada en centros y cada centro por diferentes departamentos. En la facultad 1 se encuentra el centro de Identificación y Seguridad Digital (CISED) donde se encuentra el Departamento de Biometría en el que se desarrollan soluciones informáticas para el reconocimiento de un individuo según sus rasgos físicos o de comportamiento.

1.7 Metodologías de desarrollo

La determinación de una metodología de desarrollo del *software*, es un factor fundamental en el éxito de un proyecto. Surge ante la necesidad de utilizar una serie de procedimientos, técnicas, herramientas y soporte documental a la hora de desarrollar un producto de *software*. Su selección incluye un marco de trabajo usado para estructurar, planificar y controlar el proceso de desarrollo en sistemas de información (39). Existen dos grandes grupos de metodologías:

1.7.1 Las metodologías tradicionales

Este tipo de metodología requiere de una extensa documentación durante todo el ciclo de vida, ya que pretende prever todo de antemano. Suelen ser eficaces y necesarias cuando se trata de proyectos que requieren de grandes equipos de desarrollo. Una de las metodologías tradicionales más utilizadas es la Metodología RUP (Rational Unified Process), la cual divide el desarrollo en cuatro fases que definen su ciclo de vida y en nueve flujos de trabajo, seis de ingeniería y tres de soporte (40).

Las principales características de RUP son (41):

[3] DATYS : (Tecnologías y Sistemas): Es una empresa de capital 100 % Cubano dedicada al desarrollo de aplicaciones informáticas en diversas líneas de negocios, a las que aporta soluciones con sus productos en las áreas de la identificación biométrica, la gestión de contenidos, la minería en textos, la seguridad técnica integral, la gestión de sistemas empresariales, hoteleros y de agencias de viajes.

- ✓ Centrado en la arquitectura: la arquitectura involucra los elementos más significativos del sistema y está influenciada, entre otros, por plataformas de *software*, sistemas operativos, administradores de bases de datos, protocolos para la comunicación de red, consideraciones de desarrollo como sistemas heredados y requerimientos no funcionales.
- ✓ Dirigido por casos de uso: tiene a los casos de uso como el hilo conductor del proceso de desarrollo, los desarrolladores crean una serie de modelos de diseño e implementación que los llevan a cabo. Los cuales se validan para que sean conformes a los casos de uso. Finalmente, los casos de uso también sirven para realizar las pruebas sobre los componentes desarrollados.
- ✓ Iterativo e incremental: esta característica propone dividir el proceso de desarrollo en partes, cada una de las cuales incluya las fases de: Requerimientos, Análisis, Diseño, Implementación y Pruebas, con el objetivo de acelerar el ritmo de desarrollo para que el producto sea liberado en el menor tiempo posible y con mayor calidad.

Fases (42)

- ✓ Inicio: el objetivo es determinar la visión del proyecto y definir lo que se desea realizar.
- ✓ Elaboración: etapa en la que se determina la arquitectura óptima del proyecto.
- ✓ Construcción: se obtiene la capacidad operacional inicial.
- ✓ Transición: obtener el producto acabado y definido.

Flujos de trabajo

Flujos de ingeniería

- ✓ Modelación del negocio: describe los procesos de negocio, identificando quiénes participan y las actividades que requieren automatización.
- ✓ Requerimientos: define qué es lo que el sistema debe hacer, para lo cual se identifican las funcionalidades requeridas y las restricciones que se imponen.
- ✓ Análisis y diseño: describe cómo el sistema será realizado a partir de la funcionalidad prevista y las restricciones impuestas (requerimientos), por lo que indica con precisión lo que se debe programar.
- ✓ Implementación: define cómo se organizan las clases y objetos en componentes, cuáles nodos se utilizarán y la ubicación en ellos de los componentes y la estructura de capas de la aplicación.
- ✓ Prueba: busca los defectos a lo largo del ciclo de vida.
- ✓ Despliegue: produce un entregable del producto y realiza actividades como empaque, instalación, asistencia a usuarios, para entregar el *software* a los usuarios finales (43).

Flujos de soporte:

- ✓ Gestión de configuración y cambios: describe cómo controlar los elementos producidos por todos los integrantes del equipo de proyecto en cuanto a: utilización/actualización concurrente de elementos y control de versiones.
- ✓ Gestión del proyecto: involucra actividades con las que se busca producir un producto que satisfaga las necesidades de los clientes.
- ✓ Ambiente (*Environment*): contiene actividades que describen los procesos y herramientas que soportarán el equipo de trabajo del proyecto, así como el procedimiento para implementar el proceso en una organización (43).

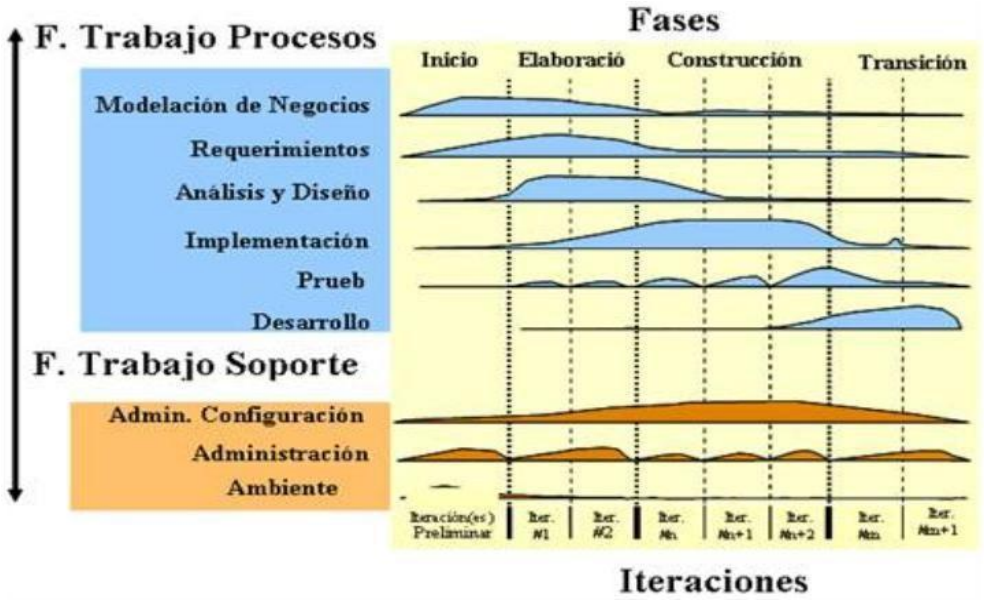

Figura 15. Fases y flujos de trabajo de RUP.

1.7.2 Las metodologías ágiles (44)

Para este tipo de metodologías es más importante la capacidad de respuesta ante los cambios realizados que el seguimiento estricto de un plan. Se enfatiza en la satisfacción del cliente y promueve el trabajo en equipo. Una de las metodologías más utilizadas es la metodología XP (Extreme Programming), la cual se basa en una serie de valores, principios y prácticas que brindan una satisfactoria productividad y un gran alcance en el proceso de desarrollo del *software*. Durante el ciclo de vida, en dicha metodología aparecen cambios frecuentes, por lo que a veces el equipo que lo integra no se encuentra preparado para

enfrentarlos, ante tal situación los desarrolladores enfrenta un conjunto de valores que son importantes para el logro del producto.

XP está basada en los siguientes principios (44):

- ✓ Pruebas unitarias: se basa en las pruebas realizadas a los principales procesos, de tal manera que se adelanta en algo hacia el futuro, se puede hacer pruebas de las fallas que pudieran ocurrir. Es como si se adelantara a obtener los posibles errores.
- ✓ Re-fabricación: se desarrolla en la reutilización de código, para lo cual se crean patrones o modelos estándares, siendo más flexible al cambio.
- ✓ Programación en pareja: una particularidad de esta metodología es que propone la programación en pares, la cual consiste en que dos desarrolladores participen en un proyecto en una misma estación de trabajo. Cada miembro lleva a cabo la acción que el otro no está haciendo en ese momento.

Artefactos esenciales en XP:

1. Historias de Usuario
2. Pruebas de aceptación
3. Plan de iteraciones
4. Metáfora
5. Tareas de ingeniería
6. Pruebas unitarias y de integración
7. Código

Fases de XP (43):

- ✓ Exploración: se planean las Historias de Usuarios (HU). Se realiza un estudio de las herramientas, tecnologías y prácticas que se utilizarán en el proyecto. Se prueban las tecnologías y se construye un prototipo del sistema.
- ✓ Planeamiento: el cliente establece la prioridad de cada HU y sobre la base de esto, los programadores realizan una estimación del esfuerzo necesario para cada una de ellas. Se determina el contenido y el cronograma de la primera entrega, en conjunto con el cliente. Se generan los artefactos: Plan de iteraciones, que se basa en la elaboración de HU no abordadas, Pruebas de aceptación no superadas en la iteración anterior y tareas no terminadas en la iteración anterior y, el Plan de entrega compuesto por iteraciones de no más de tres semanas.

- ✓ Iteraciones: se realizan varias iteraciones sobre el producto antes de ser entregado. Para cada una de estas iteraciones se confeccionan las unidades de prueba y se aprueban, antes de empezar a codificar la solución. Luego se aplican las pruebas de aceptación diseñadas por el cliente, destinadas a evaluar si al final de una iteración se consiguió la funcionalidad requerida. El objetivo de estas pruebas es verificar el cumplimiento de los requisitos.
- ✓ Producción: requiere de pruebas adicionales y revisiones de rendimiento antes de que el sistema sea trasladado al entorno del cliente.
- ✓ Mantenimiento: mientras la primera versión se encuentra en producción, el proyecto XP debe mantener el sistema en funcionamiento al mismo tiempo que desarrolla nuevas iteraciones. La fase de mantenimiento puede requerir nuevo personal dentro del equipo y cambios en su estructura.
- ✓ Muerte del proyecto: se genera la documentación final del sistema y no se realizan más cambios en la arquitectura. La muerte del proyecto ocurre cuando el sistema no genera los beneficios esperados por el cliente o cuando no hay presupuesto para mantenerlo.

Figura 16. Ciclo de vida de un proyecto XP.

Luego de haber realizado un profundo estudio de las metodologías, se ha decidido basar el desarrollo del producto sobre una de las metodologías ágiles, Extreme Programming (XP), a continuación se fundamentará el por qué.

1.7.3 Fundamentación de XP como metodología a utilizar

Se ha utilizado la metodología XP (Extreme Programming) porque en este caso el grupo de desarrollo encargado del producto de *software* es de apenas dos personas, por lo que no es recomendable utilizar una de las metodologías tradicionales, debido a que la cantidad de documentación y roles que requiere es excesiva para un par de personas. En el grupo de desarrollo persiste la presencia del cliente y es de vital importancia hacer entregas de los resultados del producto en corto plazo.

1.8 Herramientas y tecnologías

Con el propósito de desarrollar un componente con la mayor calidad posible se definieron las siguientes tecnologías y herramientas, las cuales serán detalladas más adelante.

- Entorno de desarrollo integrado (IDE) Visual Studio 2010
- Lenguaje de programación, C Sharp (C#)
- Lenguaje Unificado de Modelado, UML
- Herramienta de modelado, Visual Paradigm
- Bibliotecas de clases: EMGU CV y OPEN CV
- Sistema gestor de base de datos, PostgreSQL
- .NET Framework 4.0
- ORM Telerik OpenAccess

1.8.1 Entorno de desarrollo integrado (IDE)

Un entorno de desarrollo integrado, llamado también IDE (sigla en inglés de Integrated Development Environment), es un programa informático compuesto por un conjunto de herramientas de programación que proveen un marco de trabajo amigable para la mayoría de los lenguajes de programación (45).

IDE Visual Studio 2010

Visual Studio es un entorno de desarrollo solamente para los sistemas operativos Windows. Con este IDE es posible crear aplicaciones de escritorio, sitios y aplicaciones web, así como servicios web en cualquier

entorno que soporte la plataforma .NET y aplicaciones que se intercomuniquen entre estaciones de trabajo, páginas web y dispositivos móviles. Soporta varios lenguajes de programación tales como C++, C#, J#, y Visual Basic .NET, aunque actualmente se han desarrollado las extensiones necesarias para muchos otros (46).

Su interfaz es limpia y personalizable. Brinda herramientas que facilitan todo el proceso de desarrollo de software simplificando procesos tales como el diseño visual de interfaces, la implementación, depuración y prueba. Incluye mecanismos de colaboración para la administración del ciclo de vida de las aplicaciones, chequeando en tiempo real el estado de los proyectos, para cada miembro del equipo, con potentes herramientas para la elaboración de informes y reportes (47).

1.8.2 Lenguajes utilizados

Lenguaje de programación C#

Es un lenguaje de programación orientado a objetos de alto nivel, diseñado para la interacción entre hombre-máquina, lo que lo hace tener una infraestructura de lenguaje común. Es desarrollado y estandarizado por Microsoft como parte de su plataforma .NET. Dicho lenguaje ha sido una fusión de características originadas por dos lenguajes de programación anteriores: el C y el C++ e intentando solucionar algunos problemas de Java, por lo que programar usando C# es mucho más sencillo e intuitivo. C# ha sido diseñado para la plataforma Microsoft .NET, en la que los servicios son ofrecidos en forma de componentes (48).

Algunas de las características del lenguaje de programación C# son:

- Sencillez de uso: C# suprime elementos que otros lenguajes incluyen y que son innecesarios en .NET. El código escrito en C# es auto-contenido, lo que significa que no necesita de ficheros adicionales tales como ficheros de cabecera. El tamaño de los tipos de datos básicos es fijo e independiente del compilador.
- Orientación a componentes: la propia sintaxis de C# incluye elementos propios del diseño de componentes que otros lenguajes tienen que simular mediante construcciones más o menos complejas. Es decir, la sintaxis de C# permite definir plácidamente propiedades (similares a campos de acceso controlado), eventos (asociación controlada de funciones de respuesta a notificaciones) o atributos (información sobre un tipo o sus miembros).

- Eficiente: en principio, en C# todo el código incluye numerosas restricciones para asegurar su seguridad y no permite el uso de punteros. Sin embargo, a diferencia de Java, en C# es posible saltarse dichas restricciones manipulando objetos a través de punteros. Para ello basta marcar regiones de código como inseguras (modificador *unsafe*) y podrán usarse en ellas punteros de forma similar a como se hace en C++, lo que puede resultar vital para situaciones donde se necesite velocidad de procesamiento grande.
- Orientado a objetos: C# como lenguaje de última generación, y de propósito general, es orientado a objetos. No permite la inclusión de funciones ni variables globales que no estén incluidos en una definición de tipos, por lo que la orientación a objetos es más pura y clara que en otros lenguajes como C++. Además, C# soporta todas las características del paradigma de la programación orientada a objetos, como son la encapsulación, la herencia y el polimorfismo (48).

Lenguaje Unificado de Modelado (UML)

El modelado es esencial para la construcción de *software* para comunicar la estructura de un sistema complejo, especificar el comportamiento deseado del sistema, comprender mejor lo que se está construyendo, descubrir oportunidades de simplificación y reutilización.

El Lenguaje Unificado de Modelado es un lenguaje de modelado visual que se usa para especificar, visualizar, construir y documentar artefactos de un sistema de *software.* Captura decisiones y conocimiento sobre los sistemas que se deben construir. Pretende dar apoyo a la mayoría de los procesos de desarrollo orientados a objetos, métodos de desarrollo, etapas del ciclo de vida, dominios de aplicación y medios. El lenguaje de modelado pretende unificar la experiencia pasada sobre técnicas de modelado e incorporar las mejores prácticas actuales en un acercamiento estándar. UML está pensado para ser utilizado en herramientas interactivas de modelado visual que tengan generadores de código así como generadores de informes. Este lenguaje UML tiene como objetivo permitir visualizar de una forma gráfica un sistema de tal manera que pueda ser entendido por todos, especificando cuáles son las características antes de que comience a ser construido (49).

1.8.3 Herramienta de modelado (Visual Paradigm)

Visual Paradigm es una herramienta UML profesional que ayuda a los equipos de desarrollo en la confección de los distintos modelos que van desde la construcción hasta el despliegue. El *software* de

modelado UML ayuda a una rápida construcción de aplicaciones de calidad. Permite graficar varios tipos de diagramas de clases. Esta herramienta también proporciona una abundante documentación UML (50). Está diseñado para una amplia gama de usuarios, incluidos analistas, arquitectos, que estén interesados en la construcción de sistemas de *software*.

1.8.4 Open CV

Open CV es un potente conjunto de librerías utilizadas para la manipulación de imágenes, desarrollada por Intel. Se da bajo la licencia BSD[4], que permite que sea usada libremente para propósitos comerciales y de investigación con las condiciones en ella expresadas. Open CV es multiplataforma, existiendo versiones para GNU/Linux, Mac OS X y Windows. OpenCV sólo es compatible con C++, C, y los lenguajes de programación Python, pero esto no quiere decir que no se pueda utilizar con el lenguaje C#, solo que para esto se debe de utilizar una envoltura llamada Emgu CV, la cual proporciona una interfaz entre Open CV y C # (51). La OpenCV proporciona numerosos elementos de alto nivel que facilitan grandemente el trabajo al usuario. La principal característica de esta biblioteca, además de su funcionalidad y calidad, es su rendimiento. Los algoritmos están basados en estructuras de datos altamente flexibles, y más de la mitad de las funciones han sido optimizadas aprovechando la arquitectura de Intel. La disponibilidad del código de la biblioteca permite añadir nuevas funcionalidades o simplemente modificar las existentes (respetando siempre los requisitos exigidos por la licencia BSD, por la que se rige la misma) lo que proporciona otra ventaja para su uso.

1.8.5 Emgu CV

La biblioteca de clases Emgu CV es una plataforma para .Net de las librerías de procesado de imagen de Intel Open CV. Permite llamar las funciones de Open CV desde lenguajes .Net compatibles como C#, VB, VC++.

En la Figura 17 se muestra un esquema de la arquitectura de Emgu CV. Donde la primera capa (layer 1) contiene las funciones, estructuras y enumeraciones que se reflejan directamente en Open CV. La segunda capa (layer 2) contiene las clases que combinan las ventajas del entorno .NET (52).

[4] **BSD:** (Berkeley *Software* Distribution): Licencia de *software* otorgada principalmente a los sistemas.

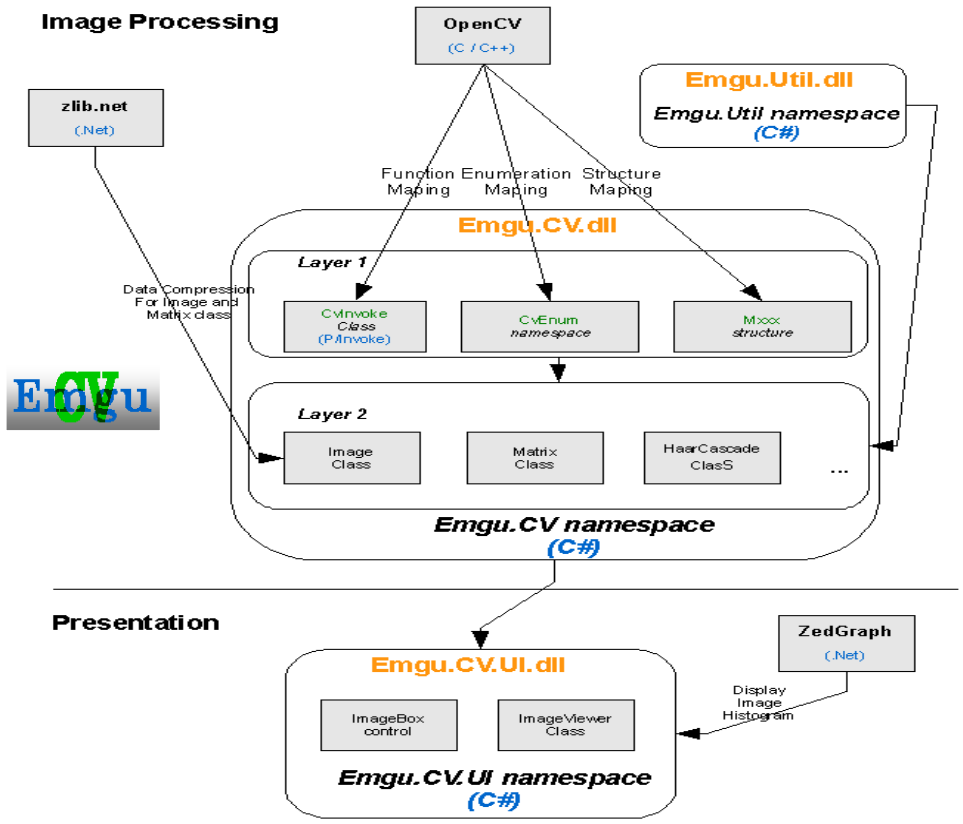

Figura 17. Estructura del EMGU (52).

1.8.6 Sistema gestor de base de datos

Un Sistema Gestor de Bases de Datos (SGBD) es un conjunto de programas que permite definir, construir y manipular la base de datos, garantizando la seguridad e integridad de los datos. Entre las características que deben cumplir los SGBD se encuentran: abstracción de la información, la independencia de los datos, redundancia mínima, consistencia en los datos, seguridad, integridad, respaldo y recuperación, tiempo de respuesta y control de concurrencia. Dentro de los SGBD se encuentran los comerciales (ejemplo: SQL Server y Oracle) y los libres (ejemplo: PostgreSQL y MySQL) (53).

PostgreSQL

PostgreSQL es un sistema de gestión de base de datos relacional, orientado a objetos de *software* libre, publicado bajo la licencia BSD. El código fuente de PostgreSQL está disponible para cualquier persona

libre de cargos directos, permitiendo a cualquiera colaborar con el desarrollo del proyecto o modificar el sistema para ajustarlo a sus necesidades (54).

Entre sus principales características se encuentran:

- Alta concurrencia: permite que mientras un proceso escribe en una tabla, otros accedan a la misma tabla sin necesidad de bloqueos.
- Consistencia: es la propiedad que asegura que solo se empieza aquello que se puede acabar. Por lo tanto se ejecutan aquellas operaciones que no van a romper las reglas y directrices de integridad de la base de datos.
- Aislamiento: es la propiedad que asegura que una operación no puede afectar a otras. Esto garantiza que dos transacciones sobre la misma información nunca generarán ningún tipo de error.
- Durabilidad: es la propiedad que asegura que una vez realizada la operación, esta persistirá y no se podrá deshacer aunque falle el sistema (55).

1.8.7 .Net Framework 4.0

Se trata de un conjunto de herramientas de trabajo, que evoluciona junto con las versiones de los sistemas operativos de Microsoft. .NET Framework 4.0 es perfectamente compatible con los sistemas operativos Windows; se compone por el motor Common Language Runtime (CLR) y por la biblioteca de clase base (BCL). El CRL se encarga de la administración de la memoria, la ejecución de subprocesos, la seguridad del código o la compilación y la BCL, es orientado a objetos, con lo que se pretende una rápida familiarización con las prestaciones del *software*. .NET Framework 4.0 requiere como mínimo Windows XP SP3 (56).

1.8.8 Telerik OpenAccess ORM

Telerik OpenAccess ORM[5], es una herramienta de mapeo que ha sido diseñada para generar códigos de acceso a datos. Utiliza poco espacio de memoria en el cliente, es muy veloz cuando se obtienen los datos. Está organizado por herramientas y técnicas que optimizan el rendimiento, logrando así un buen acceso a los datos. Esta herramienta funciona para todas las plataformas .NET, ofrece soporte integrado para más de 12 bases de datos, incluyendo SQL Server, Oracle y MySQL; proporciona una perfecta integración con Visual Studio permitiendo crear bases de datos independientes del código. Permite realizar el proceso de

[5] ORM -> Glosario de términos.

mapeado y genera modelos con el potente diseñador visual para OpenAccess ORM y las características avanzadas de optimización de código (57).

1.9 Conclusiones del capítulo

En el presente capítulo se realizó el estudio del estado del arte sobre los diferentes sistemas biométricos de reconocimiento facial, que basan su acción en diferentes técnicas de identificación como: PCA, LBP, EBGM, entre otros. Algunas de estas técnicas han sido utilizadas para el desarrollado de sistemas en empresas como: CENATAV y DATYS, las cuales están orientadas hacia el área del reconocimiento facial. Se analizaron las metodologías existentes llegando a la conclusión de utilizar la metodología XP para el desarrollo del componente. Como entorno de desarrollo integrado, Visual Studio, acompañado de la biblioteca Emgu CV para utilizar Open CV desde el lenguaje de programación C Sharp. Como gestor de base de datos se utilizó PostgreSQL y como base para la plataforma .Net el *framework* 4.0 además del Telerik OpenAccess para el mapeo de la base de datos.

Capítulo II: Características del sistema

El presente capítulo hace referencia a las características que presenta el componente de reconocimiento facial basado en la técnica de *Eigenfaces*. Se definen las principales funcionalidades por las cuales se conforman las historias de usuarios, y se definen además los requisitos no funcionales. Se presenta la propuesta de solución, la arquitectura a utilizar por el componente y los artefactos que define la metodología XP.

2.1 Propuesta de solución

Con la necesidad de implementar un sistema de reconocimiento facial para el CISED, se ha determinado desarrollar varios componentes basados cada uno de ellos en diferentes métodos y técnicas. La propuesta de solución que se presenta a continuación describe las características que presentará el componente para el reconocimiento de rostro mediante el uso de *Eigenfaces*, dividiendo el proceso de reconocimiento en dos etapas fundamentales: la primera es la etapa de entrenamiento, donde a partir de un conjunto de imágenes seleccionadas de la base de datos, se realiza el entrenamiento previo pasando dichas imágenes por los procesos de detección, pre-procesado y extracción, dando como salida los coeficientes de las imágenes de entrenamiento como se muestra en la Figura 18, y en una segunda etapa denominada etapa de prueba, donde se realiza el proceso de reconocimiento a través de una imagen facial, la cual pasa por los procesos de detección, pre-procesado, extracción y clasificación, hasta llegar a la respuesta de conocido o no conocido como se muestra en la Figura 19.

2.1.1 Etapa de entrenamiento

Para que el sistema esté preparado para realizar la parte de comparación es necesario disponer de las características de las imágenes de entrenamiento.

Esta etapa se realiza a partir de un conjunto de imágenes conocidas, seleccionadas de la base de datos (Figura 18), con el objetivo de poder disponer de las características de las imágenes en la etapa de prueba (Figura 19). Los siguientes conceptos de dominio forman parte del esquema de solución para esta etapa.

Conjunto de imágenes de entrenamiento: conjunto de imágenes por cada persona de la base de datos para las cuales se realizará el entrenamiento.

Detección: se detecta si la imagen contiene la presencia de un rostro.

Pre-procesado: se normaliza el tamaño, el brillo y contraste de la imagen además de corregir la rotación.
PCA: método utilizado para el cálculo de las *Eigenfaces*.
***Eigenfaces*:** resultados obtenidos una vez aplicado PCA para las imágenes de entrenamiento.
Proyección: proyectar cada una de las imágenes del conjunto de entrenamiento con las *Eigenfaces*.
Coeficientes de las imágenes de entrenamiento: resultado obtenido de la proyección.

Una vez realizado todo el proceso, los coeficientes de las imágenes de entrenamiento y las *Eigenfaces* son guardados en la base de datos.

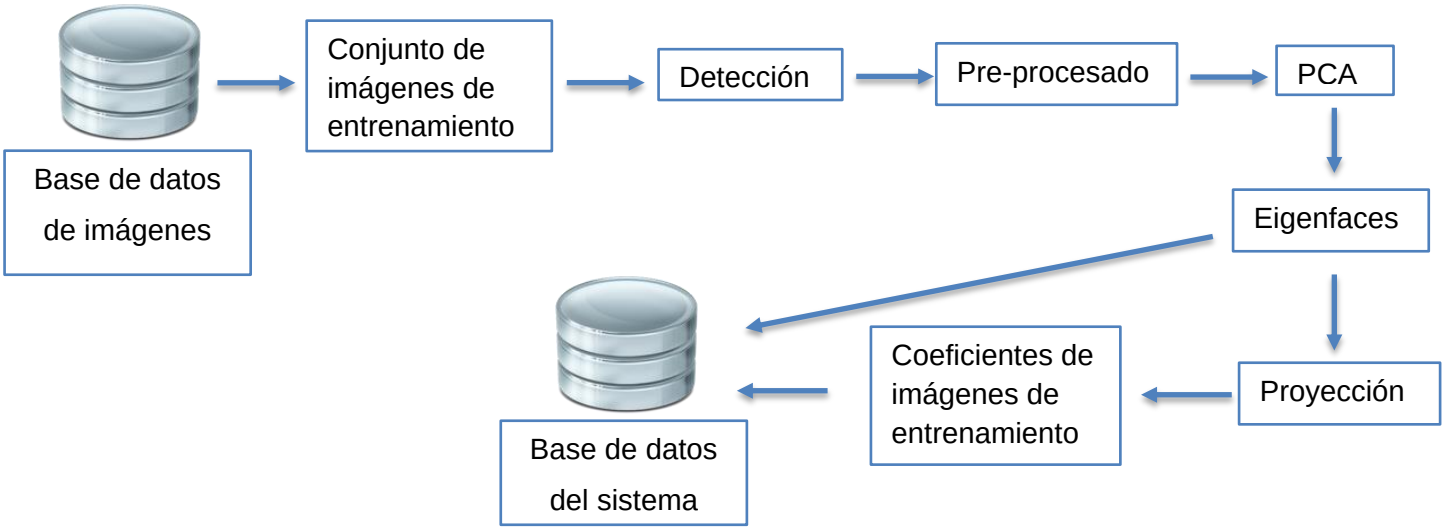

Figura 18. Esquema de la etapa de entrenamiento.

2.1.2 Etapa de prueba

La etapa de prueba se realiza a partir de la entrada de una imagen con el objetivo de realizar la identificación de la persona, posteriormente se realizan una serie de procesos como se muestra en la Figura 19. Los siguientes conceptos de dominio forman parte del esquema de solución para esta etapa.

Imagen facial: imagen facial de la persona que se identificará.
Detección: se detecta si la imagen contiene la presencia de un rostro.
Pre-procesado: se normaliza el tamaño, el brillo y contraste de la imagen además de corregir la rotación.
Proyección: proyecta la imagen con todas las *Eigenfaces* guardadas en la base de datos producto de la etapa de entrenamiento, obteniendo los coeficientes de la imagen de prueba.

Comparación: se comparan los coeficientes de la imagen con los coeficientes almacenados en la base de datos producto de la etapa de entrenamiento, donde se decide si el rostro es conocidos o no.

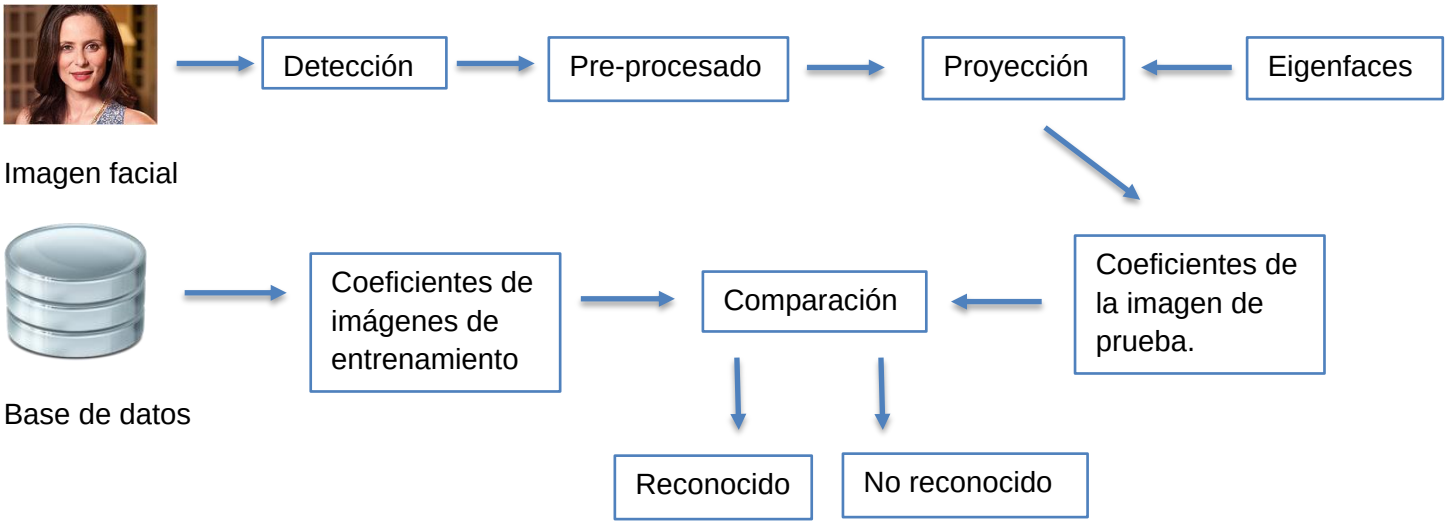

Figura 19. Esquema de la etapa de prueba.

2.2 Metáfora

Una metáfora es un pequeño resumen sobre cómo debe funcionar el componente con el objetivo de que el dominio del problema sea fácilmente comprendido. La selección de una metáfora permite mantener la coherencia e integridad conceptual de todos los elementos a implementar. A continuación se define la metáfora global que guía el desarrollo del presente proyecto.

El componente de reconocimiento facial mediante el uso de *Eigenfaces* será integrado a un sistema general de reconocimiento de rostros que brindará la posibilidad de escoger entre diferentes técnicas de identificación, logrando así que sea más robusto y seguro. Básicamente se divide en dos etapas fundamentales: entrenamiento y prueba.

Para la etapa de entrenamiento se seleccionan un conjunto de imágenes por cada persona de la base de datos, a las que se les realiza un pre-procesado para normalizar una serie de características como son el brillo, contraste, rotación del rostro y tamaño. Seguidamente se les aplica el método PCA para la obtención de las *Eigenfaces* y el rostro promedio, para calcular los coeficientes o vectores propios de estas imágenes se proyectan las *Eigenfaces* obtenidas con cada una de ellas. Finalmente, los datos

obtenidos (*Eigenfaces*, vectores propios, rostro promedio) son almacenados en la base de datos para su posterior reutilización.

Una vez culminado el proceso de entrenamiento, el componente estará listo para realizar el reconocimiento de una persona, lo cual se realiza a través de la etapa de prueba, que consiste en la identificación de un individuo. Para ello es pre-procesada la imagen a comparar y calculado su vector característico a partir de las *Eigenfaces* y el rostro promedio obtenido en la etapa de entrenamiento. Posteriormente son calculadas las distancias euclidianas entre este vector y los vectores almacenados. Los datos de la persona asociados al vector de menor distancia es el resultado retornado por el componente.

2.3 Requisitos del componente

Este epígrafe cumple un papel esencial en el proceso de desarrollo de un producto de *software*, se orienta a la definición de lo que se desea producir, describiendo con claridad, sin imprecisiones, en forma estable y compacta. Se define el comportamiento que deberá tener el componente, minimizando la posibilidad de errores que puedan ocurrir relacionados al desarrollo del mismo, pues se tiene especificado de forma clara lo que el cliente desea. Los requisitos se dividen en dos grupos: principales funcionalidades y requisitos no funcionales.

2.7.3 Principales funcionalidades

Son capacidades o condiciones que el componente debe ser capaz de cumplir. Establecen los comportamientos y describen las transformaciones que el mismo debe realizar para obtener los resultados deseados por los programadores.

1. Pre-procesar imágenes de entrenamiento.
    - 1.1. Verificar que sea una imagen de rostro.
    - 1.2. Centrar la imagen.
    - 1.3. Corregir el brillo y contraste.
    - 1.4. Normalizar el tamaño.
    - 1.5. Corregir la rotación.
2. Realizar el entrenamiento.
    - 2.1. Calcular las *Eigenfaces*.

2.2. Calcular los coeficientes de todas las imágenes almacenadas a partir de las *Eigenfaces* obtenidas.

3. Identificar persona.
   3.1. Cargar la imagen del rostro a identificar.
   3.2. Pre-procesar imagen (RF-1)
   3.3. Calcular los coeficientes.
   3.4. Comparar coeficientes con los coeficientes de las imágenes de entrenamiento.
   3.5. Mostrar si la persona es identificada o no.

2.8.3 Requisitos no funcionales

Los requisitos no funcionales son propiedades o cualidades que el producto debe tener. Son las características que hacen al *software* atractivo, usable, rápido o confiable. En muchos casos los requisitos no funcionales son fundamentales en el éxito del producto. Normalmente están vinculados a las principales funcionalidades.

Requisitos de *software*.

- La aplicación deberá estar instalada en plataforma .NET.
- Instalación del SGBD PostgreSQL.

Requisitos mínimos de *hardware*.

- Periféricos: mouse y teclado.
- 1GB de RAM.
- Procesador Intel Pentium IV o superior.
- 2 GB de espacio en disco.

Restricciones en el diseño y la implementación.

- El componente debe implementarse usando el lenguaje C#.
- El sistema gestor de bases de datos será PostgreSQL 8.4.
- El componente debe desarrollarse usando el IDE Visual Studio 2010.
- Visual Paradigm Enterprise Edition como herramienta UML.

Requisitos de seguridad.

- El componente garantizará el tratamiento de excepciones.

- Integridad: la información manejada por el componente será objeto de cuidadosa protección contra la corrupción de datos.

Soporte.

- Se requiere que el producto reciba mantenimiento y configuración ante cualquier fallo que ocurra durante el período de prueba.
- Es necesaria una correcta documentación para el uso o desarrollo de las funcionalidades de la herramienta.

2.4 Historias de Usuario

Las Historias de Usuario (HU), es el artefacto generado por la metodología XP para especificar los requisitos que debe cumplir el software desde el punto de vista del cliente. A continuación se muestra la HU "Pre-procesar imágenes de entrenamiento" (Ver Anexo 1 para las demás HU).

Historia de usuario 1

Tabla 3. HU_1 Pre-procesar imágenes de entrenamiento**.**

Historia de usuario	
Numero: HU_1	Nombre de historia de usuario: Pre-procesar imágenes de entrenamiento.
Usuario : Sairenys Barrios Pérez y Herbert Tamayo Valero	
Prioridad de negocio: Alta	Puntos estimados: 3
Riesgo en desarrollo: Alta	Iteraciones : 1
Descripción: se intenta compensar todo lo que puede provocar que dos imágenes de la misma cara sean diferentes. Esto incluye normalizar el tamaño, brillo y el contraste, corregir la rotación.	

Observaciones: en ocasiones, al tratar de corregir el contraste y/o brillo con determinado filtro, puede quedar muy oscura la imagen.

2.5 Arquitectura

La arquitectura es la representación de más alto nivel de la estructura de un sistema informático. Define los componentes funcionales que la integran y las interacciones entre ellos. Además incluye un conjunto de patrones y restricciones que supervisan su composición como estándares, convenciones, reglas y procesos (58).

La propuesta de la arquitectura para el componente de reconocimiento facial mediante el uso de *Eigenfaces* basa su implementación en la arquitectura n capas como se muestra en la Figura 20. El principal objetivo que persigue la arquitectura n capas es reducir dependencias entre artefactos, situándolos en capas lógicas, donde cada capa depende del servicio prestado por la inferior y presta un servicio a la superior, proporcionando a los desarrolladores ventajas en cuanto al mantenimiento y reutilización de componentes, de manera tal que una capa puede ser totalmente reemplazada sin afectar el resto del sistema. En este caso la arquitectura es basada en 2 capas, las cuales se describen a continuación:

Capa de negocio

Es la capa encargada de modelar la lógica de negocio que dará solución a las funcionalidades del componente. Involucra cálculos basados en la información dada por el usuario, datos almacenados y validaciones. Controla la ejecución de la capa de acceso a datos y servicios externos.

Capa de acceso a datos

La capa de acceso a datos se encarga de acceder y manipular los datos persistentes de un sistema, considerándose la capa más crítica y sensible de la arquitectura. Recibe solicitudes de almacenamiento o recuperación de información desde la capa de negocio.

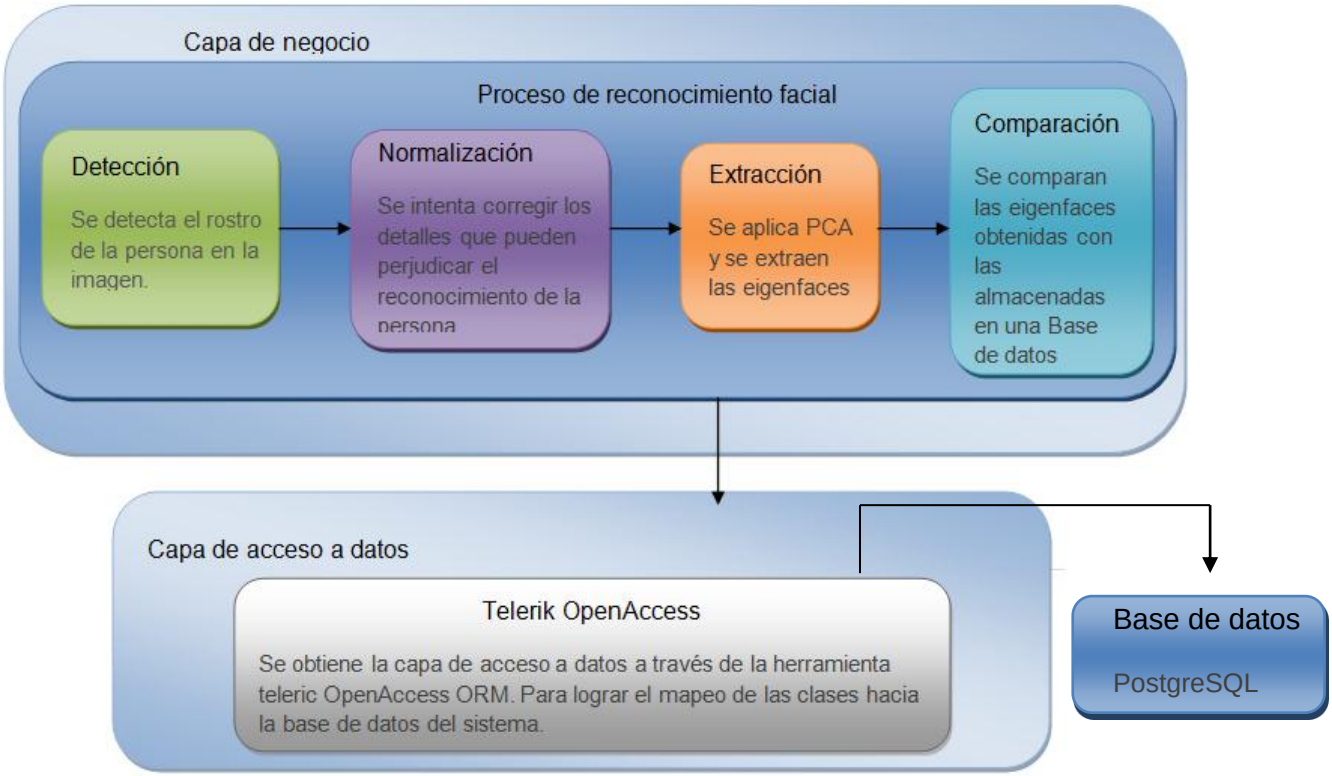

Figura 20. Diagrama de arquitectura

2.8.6 Patrones arquitectónicos

El patrón arquitectónico que se utiliza es Inyección de dependencia e Inversión de control, el cual permite delegar al sistema de reconocimiento de rostros, la función de seleccionar un tipo de implementación concreta de las dependencias de nuestras clases. En definitiva, este patrón describe técnicas para soportar una arquitectura de tipo plug-in donde los objetos pueden buscar instancias de otros objetos que requieren y de los cuales dependen (59). En este caso se le brinda la posibilidad al sistema general de reconocimiento de rostros de utilizar el componente implementado a través de la creación de una interfaz, en la que se hacen reiteradas llamadas a los distintos métodos u objetos de las clases del componente.

2.6 Patrones de diseño

Los patrones de diseño ofrecen esquemas para definir estructuras de diseño con las que construye sistemas de *software* orientado a objetos. Permite formalizar un vocabulario común entre diseñadores y

estandarizar el modo en que se realiza el diseño, así como la reiteración en la búsqueda de soluciones (60). Los patrones del diseño utilizados en el desarrollo del componente son los Patrones de *Software* para la Asignación General de Responsabilidad (GRASP) y patrones *Gang of Four* (GOF).

Patrones GRASP

- Experto: cada responsabilidad es asignada a la clase contenedora de la información necesaria para cumplirla. El uso de este patrón permitirá a los objetos poseer su propia información para hacer lo que se les pide, también garantiza el mínimo de relaciones entre las clases, para así obtener un componente sólido y fácil de mantener. Un ejemplo es la clase “Detección.cs”, la cual es responsable de realizar todas las operaciones relacionadas con la detección del rostro.
- Creador: responsable de generar nuevas instancias de alguna clase. Este patrón es empleado para la asignación de responsabilidades a las clases relacionadas con la creación de objetos, de forma tal que una instancia de un objeto solo pueda ser creada por la clase que contiene la información necesaria para ello. Su uso permite crear las dependencias mínimas necesarias entre las clases, beneficiando el mantenimiento del sistema y brindando mejores oportunidades de reutilización. Un ejemplo sería la clase controladora “IdentificadorEigenfaces.cs”, donde se crean instancias de las diferentes clases como “Normalización.cs, Detección.cs”, entre otras.
- Controlador: responsable de gestionar un evento de entrada al sistema. Es un intermediario entre la capa de presentación y el núcleo de las clases donde reside la lógica del sistema. El controlador coordina la actividad de otros objetos. Por ejemplo, en el componente se hace uso del patrón controlador definiendo la clase “IdentificadorEigenfaces.cs” que es donde se manejan varias instancias de objetos.
- Alta Cohesión: basado en la asignación de responsabilidades teniendo en cuenta que la cohesión permanezca alta. Su utilización facilita la comprensión del diseño e incrementa las capacidades de reutilización.
- Bajo Acoplamiento: plantea que debe existir una alta reutilización entre las funcionalidades de las clases con una mínima dependencia, contribuyendo así al mantenimiento de las mismas. El empleo de los patrones Experto y Creador favorecen al bajo acoplamiento entre las clases del componente. Este patrón es fundamental siempre que se desee realizar un diseño de clases independientes que puedan soportar los cambios de una manera fácil y permitan la reutilización.

Patrones GoF

- *Facade* (Fachada): proporciona una interfaz unificada de alto nivel que, representando a todo un subsistema, facilite su uso. La "fachada" satisface a la mayoría de los clientes, sin ocultar las funciones de menor nivel a aquellos que necesiten acceder a ellas.

2.7 Plan de iteraciones

Durante el ciclo de vida de la metodología XP se desarrolla el artefacto Plan de iteraciones con el objetivo de mostrar la duración y el orden en que serán implementadas las historias de usuario dentro de cada iteración. Para la solución se han definido 3 historias de usuario divididas en 3 iteraciones, de acuerdo a los intereses del cliente, para una duración total del proyecto de 17 semanas. (Ver Anexo 2)

2.8 Tarjetas CRC

Las tarjetas CRC son una técnica utilizada en XP para diseñar la solución informática según el paradigma orientado a objetos. Las siglas CRC se refieren a *Class*, *Responsabilities* y *Collaborators* que son precisamente los elementos fundamentales de estas tarjetas, o sea, para su confección se identifican las responsabilidades del *software*, los objetos que las ejecutan, las clases que los definen, así como aquellos objetos que colaboran en una determinada responsabilidad. A continuación se muestra una de las principales tarjetas CRC (Ver Anexo 3 para el resto de las tarjetas CRC).

Tabla 4. Tarjeta CRC IdentificadorEigenfaces.

IdentificadorEigenfaces	
Descripción	Colaboradores
Clase controladora donde se encuentran los objetos de las clases necesarias para acceder a las funcionalidades implementadas por el componente.	✓ Normalización.cs ✓ Detección.cs ✓ Clasificación.cs ✓ Entrenamiento.cs ✓ Conexión.cs

2.9 Modelo de clases del diseño

Después de definidas las tarjetas CRC, donde se identificaron las clases del sistema y sus principales responsabilidades, se está en condiciones de diseñar el diagrama de clases que aunque no es un artefacto propio de XP, permitirá describir la estructura del componente mostrando sus clases, atributos y las relaciones entre ellos.

La implementación del componente se sustenta en nueve clases que basan su composición en diferentes métodos, que tienen el objetivo de llevar a cabo el correcto funcionamiento del componente de identificación facial (Ver Anexo 4)

2.10 Modelo de la base de datos

La persistencia es la capacidad de un objeto de mantener su valor en el espacio y en el tiempo. Las clases entidad son las que contienen toda la información persistente manejada por el sistema, cuya estructura física y lógica describe el modelo de datos. A continuación se muestra el modelo de datos del componente a desarrollar (Figura 22), compuesto por cuatro tablas en la que se guardan los datos relacionados con cada una de ellas (Ver el Anexo 5 para la especificación de las tablas).

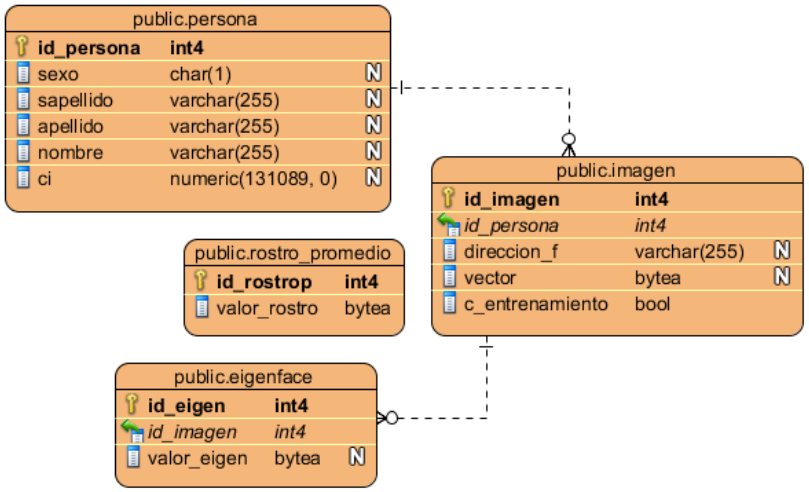

Figura 21. Diagrama de la base de datos.

2.11 Conclusiones del capítulo

Durante el desarrollo del presente capítulo se han presentado una serie de características por las que se rige el componente implementado a través de la presentación de la propuesta de solución. Se definieron las principales funcionalidades y los requisitos no funcionales, así como las historias de usuario como parte del ciclo de vida de la metodología XP, las cuales se corresponden a las principales funcionalidades del sistema. Se desarrolló además el artefacto Metáfora en la que se narra una historia que todo el mundo puede contar, o sea, se basa en una breve descripción acerca de las funcionalidades que presenta el componente y, por último, el Plan de iteraciones, encargado de mostrar el orden en el que se implementa y se da cumplimiento a las historias de usuario anteriormente definidas. Como base para la implementación del componente, se definió la arquitectura n capas, estructuradas por las capas de negocio y acceso a datos. Se obtuvo además, el modelo de clases del diseño y el modelo de la base de datos.

Capítulo III: Implementación y Prueba

3.1 Introducción

En el presente capítulo se recoge información sobre los estándares de codificación, las tareas de ingeniería y los casos de prueba realizados para validar la calidad y el correcto funcionamiento del componente implementado.

3.2 Estándar de codificación

El cumplimiento de los estándares de codificación hace que todo el código lleve el sello personal del programador y, en caso de ser varios los programadores, se busca que el código parezca que ha sido implementado por la misma persona. De esta manera, se consigue que la aplicación se convierta en un sistema fácil de comprender y de mantener.

Con el objetivo de facilitar el mantenimiento del componente se emplearon estándares de codificación basados en los siguientes términos:

- Convención **Pascal**– el primer carácter de cada palabra es en mayúscula y el resto en minúscula.

  Ejemplo: *BackColor*
- Convención **Camel**– el primer carácter de cada palabra es en mayúscula (excepto la primera palabra) y el resto en minúscula.

  Ejemplo: *backColor*

1. Convenciones para los nombres.
   a. La/cada palabra del nombre de la clase en mayúscula (convención Pascal)

      public class HolaMundo {…}

   b. Prefijo “I” para las interfaces (convención Camel)

      (Ejemplo*: IEntity*)

   c. El nombre de los métodos en mayúsculas, cada inicio de una palabra (convención Pascal)

      void SayHello (string name) {…}

   d. Nombre de variables y parámetros (convención Camel)

      ✓ El primer carácter de cada palabra es en mayúscula (excepto la primera palabra) y el resto en minúscula.

      int totalCount = 0;

- ✓ Todos los parámetros en minúsculas.

void SayHello (string name) {…}

- ✓ Los nombres de las variables son palabras con significados, no abreviaturas.

var vector;

- ✓ Solo nombres de variables como caracteres simples para los ciclos

i, n, s, f

- ✓ No se usan underscores (_) para los nombres de las variables locales.
- ✓ Todas las variables miembros de las clases son prefijadas con underscore (_) para ser diferenciadas de otras variables.
- ✓ No se usan nombres de variables que coincidan con palabras reservadas.

2. Espaciado.

- ✓ Se usa TAB para la indentación.
- ✓ No se usa SPACES.

3. Comentarios.

- ✓ Los comentarios están en el mismo nivel del código.
- ✓ Se usa **//** o **///** para los comentarios en vez de usar**/* … */**

4. Acerca de las llaves y líneas en blanco.

- ✓ Las llaves están puestas al mismo nivel del código que las contiene.
- ✓ Se usan líneas en blanco para separar agrupaciones lógicas del código.
- ✓ Se deja una línea en blanco entre cada método dentro de las clases.
- ✓ Las llaves se usan sobre líneas separadas y no sobre la misma línea como en if, for etc.
- ✓ Se usa un espacio simple antes y después de cada operador y llave.
- ✓ Se usa #región para agrupar las partes del código relacionadas.
- ✓ Se tienen los miembros privados: variables, propiedades y métodos en la parte superior de los ficheros y, clases, y los públicos en la parte de abajo.

5. Buenas prácticas de programación.

- ✓ No se escriben métodos tan extensos.
- ✓ Los nombres de los métodos sugieren su función. No se utilizan abreviaturas.
- ✓ Cada método cumple solamente una función. No se combinan más de una función por método.

- ✓ No se usan directamente números o cadenas como constantes en el código, se declaran en la parte superior.
- ✓ Se usa String.Empty en lugar de “”
- ✓ Las variables no se hacen miembros públicos o protected, se mantienen privadas y expone public/protected las Properties.

3.3 Tareas de ingeniería

Las tareas de ingeniería son un conjunto de acciones a desarrollar para resolver las HU. Permiten organizar el proceso de implementación además de posibilitar que sea conocido el grado de complejidad de cada una de ellas (HU), teniendo en cuenta la cantidad de tareas asociadas (Ver Anexo 6 para las tareas de ingenierías asociadas a cada HU), (Ver Anexo 7 para las especificaciones de cada una de las tareas).

3.4 Diagrama de despliegue

El diagrama de despliegue muestra las relaciones físicas de los distintos nodos que componen un sistema y el reparto de los componentes sobre dichos nodos. La vista de despliegue representa la disposición de las instancias de componentes de ejecución en instancias de nodos conectados por enlaces de comunicación. El siguiente diagrama describe la arquitectura física del componente durante su ejecución.

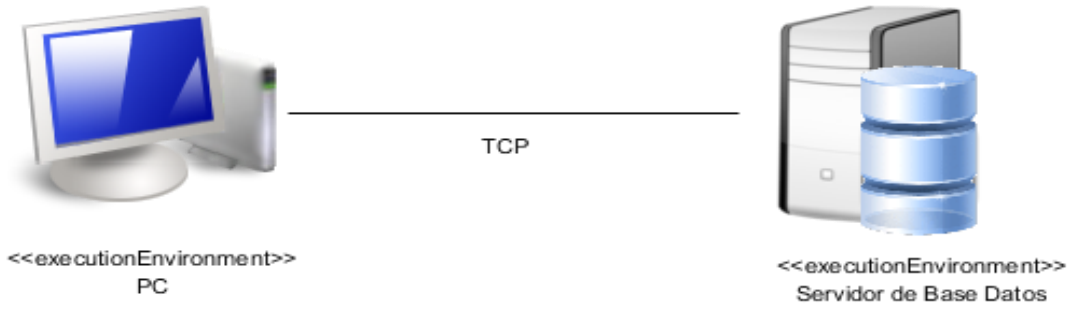

Figura 22. Diagrama de despliegue.

3.5 Diagrama de componentes

Los componentes son partes modulares del sistema, que pueden desplegarse y reemplazarse. Por lo general contienen clases y pueden ser implementados por uno o más artefactos. El diagrama de componentes muestra un conjunto de componentes relacionados entre sí. Su uso fundamental es estructurar el modelo de implementación y mostrar las relaciones de los elementos de implementación. El diagrama de componentes que se muestra en la Figura 24, representa las partes modulares en las que se

divide el componente implementado:

FaceIdentification: representa la interfaz de entrada, o sea, por donde el sistema general de identificación de personas entrará los datos que el componente necesita para su ejecución.

IdentificationService: representa el puerto por donde se conecta el sistema general de identificación de personas y el componente de reconocimiento implementado.

EigenFaceIdentification<library>: representa el nombre del componente, el cual se exporta como una dll.

EigenFaceDAL<library>: representa el componente de acceso a datos para la comunicación con la base de datos del sistema.

EMGU_CV<library>: bibliotecas de clases utilizadas por el componente para poder utilizar las librerías OPEN_CV en C#.

OPEN_CV<library>: bibliotecas de clases utilizadas para el procesamiento de imágenes.

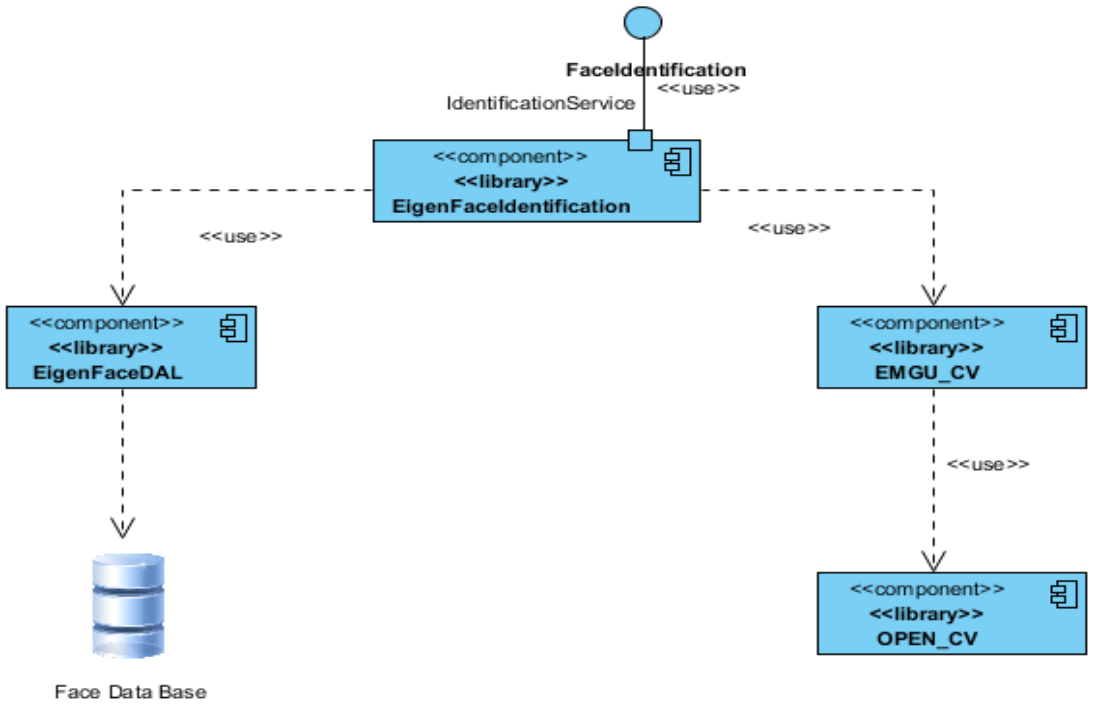

Figura 23. Diagrama de componentes.

3.6 Pruebas

XP divide las pruebas en dos grupos: pruebas unitarias, encargadas de verificar el código y diseñada por

los programadores, y pruebas de aceptación, destinadas a evaluar si al final de una iteración se consiguió la funcionalidad requerida diseñada por el cliente.

3.6.1 Pruebas unitarias

Se realizaron pruebas unitarias a las principales funcionalidades del componente, con el objetivo de aislar las diferentes partes del código y demostrar que no contenían errores, logrando como resultado que disminuyeran en gran porciento el número de errores existentes. Dichas pruebas fueron realizadas por los programadores en el transcurso del proceso de codificación, lo que facilitó que se realizaran pruebas finales más sencillas. Para ello se utilizó el propio Visual Studio, que permitió realizar dichas pruebas posibilitando resultados satisfactorios en todos los casos aplicados (Ver Anexo 8).

3.6.2 Pruebas de aceptación

Las pruebas de aceptación son más importantes que las pruebas unitarias dado que significan la satisfacción del cliente con el producto.

Dichas pruebas son creadas a partir de las historias de usuario, las cuales pueden ser sometidas a varias pruebas de aceptación, para garantizar su correcto funcionamiento. Una historia de usuario no se considera completa hasta que no pasa todas las pruebas de aceptación, ya que constituyen el final de una iteración y el comienzo de la otra.

A continuación se muestra el caso de prueba para la historia de usuario: Pre-procesar imágenes de entrenamiento. Para los demás casos de pruebas (Ver Anexo 9).

Tabla 5. HU1_CP1 Prueba de funcionalidad para el pre-procesado de las imágenes de entrenamiento.

Caso de prueba de aceptación	
Código de caso de prueba: HU1_CP1	Nombre de historia de usuario: Pre-procesar imágenes de entrenamiento
Responsable de la prueba: Sairenys Barrios Pérez.	
Descripción de la prueba: prueba de funcionalidad para el pre-procesado de las imágenes de entrenamiento donde se cargan un conjunto de imágenes de la base de datos, para luego realizarle el pre-procesado a cada una de ellas.	

Condiciones de ejecución: Debe existir un rostro frontal en cada imagen cargada de la base de datos.
Entrada/Pasos de ejecución: a partir del conjunto de imágenes tomadas de la base de datos se les realiza la detección verificando que cada imagen sea portadora del rostro frontal de una persona, para posteriormente realizar el pre-procesado: -Normalizar brillo y contraste. -Normalizar el tamaño. -Corregir la rotación.
Resultado esperado: obtener satisfactoriamente el conjunto de imágenes pre-procesadas.
Evaluación de la prueba: Prueba satisfactoria.

3.6.3 Pruebas de rendimiento

Las pruebas de rendimiento son las pruebas que se realizan para determinar lo rápido que funciona la tarea de un sistema. Dichas pruebas fueron empleadas con el objetivo de medir la velocidad de respuesta del componente en dependencia de condiciones variables como el crecimiento de los datos. (Ver Anexo 10)

3.6.4 Determinación del umbral de funcionamiento (Falso rechazo y Falsa aceptación)

Para establecer el nivel de seguridad y el correcto funcionamiento del componente, se realizan diferentes pruebas para conocer sus diferentes parámetros. Las pruebas se emplean para establecer el umbral de funcionamiento, determinar los falsos rechazos, las falsas aceptaciones y la operación tanto en *hardware* como en *software*.

Una prueba de falso rechazo se mide a partir de la respuesta del componente al indicar que está rechazando a una persona cuando en realidad no debería, y la falsa aceptación es cuando el componente acepta una persona que en realidad no es. Para la realización de dichas pruebas se definió un rango de umbrales de 5000 a 12000 para una cantidad de 1286 imágenes tomada de una base de datos de 23 personas. A continuación se muestran los valores de falsos rechazos y falsas aceptaciones obtenidos por el componente para el rango de umbrales antes mencionado.

Tabla 6. Falsos rechazos y Falsas aceptaciones obtenidas por el componente.

Umbral	Falso rechazo	Falsa aceptación
5000	297	0
6000	149	0
7000	81	2
8000	34	8
9000	10	20
10000	3	24
11000	1	26
12000	0	27

Luego de haber obtenido los valores de los falsos rechazos y las falsas aceptaciones se procede a calcular el umbral de decisión, donde el componente se encuentra equilibrado. Si observamos la gráfica de la Figura 25 podemos notar que existe un punto en el que las curvas de falsa aceptación (FA) y falso rechazo (FR) se cruzan en función del umbral. Este punto es llamado, punto de igual error (EER (*Equals Error Rate*)) que permite caracterizar de forma directa el funcionamiento del componente.

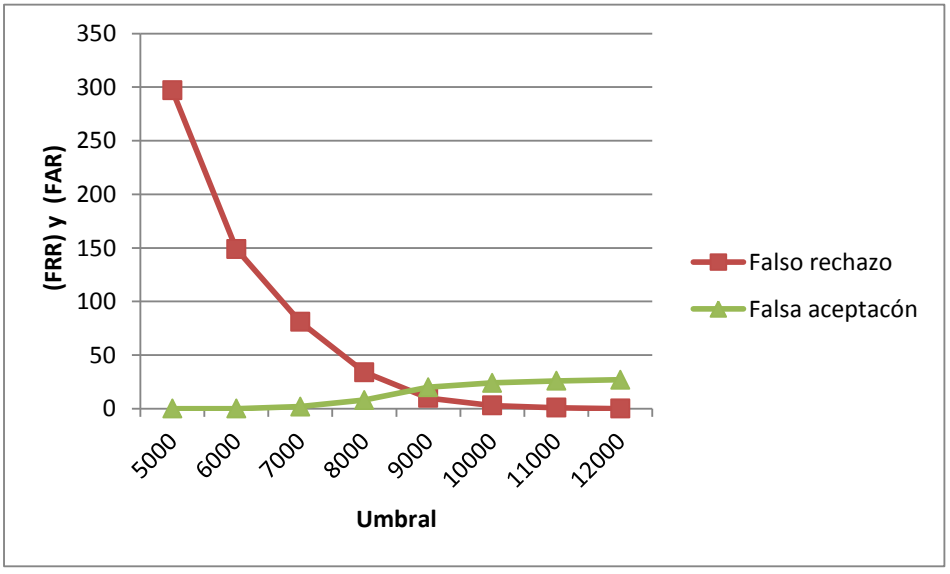

Figura 24. Gráfico del FRR y FAR

Los resultados fueron obtenidos de la siguiente manera:

Se tomaron dos puntos donde los umbrales de decisión marcaran el punto donde se cortan las rectas de los falsos rechazos y las falsas aceptaciones, los puntos fueron:

Tabla 7. Puntos para el cálculo del FR y FA.

Umbral	Falso rechazo	Falsa aceptación
5000	297	0
6000	149	0
7000	81	2
8000	34	8
9000	10	20
10000	3	24
11000	1	26
12000	0	27

Para calcular la recta del FR:

P1= (9000; 10)

P2= (8000; 34)

Y para la recta de las FA:

P1= (9000; 20)

P2= (8000; 8)

Ambos puntos permitieron realizar los cálculos a partir de la ecuación para la recta, y= mx+n, dando como resultado un punto de intersección igual a: P (8722.2; 16.6) donde el umbral de decisión es de 8722.2 con un valor de 16.6 donde se igualan los FR y FA, lo que representa una tasa de FR y FA del 1,29%.

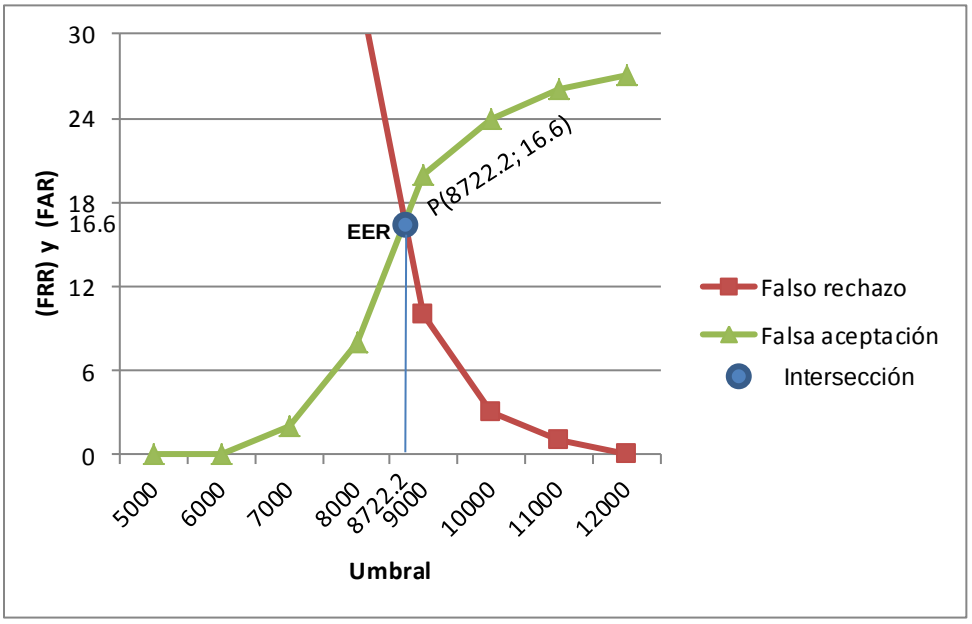

Figura 25. Gráfico del EER

3.7 Conclusiones del capítulo

A través del presente capítulo, se tomaron las decisiones de implementación del componente como son: los estándares de codificación y las tareas de ingeniería en las que se basa el desarrollo del producto final. Se desarrollaron los diagramas de componentes y despliegue para el mejor entendimiento del proceso y levantamiento del componente. Se realizaron las pruebas unitarias, pruebas de aceptación, de rendimiento, falso rechazo y falsa aceptación con el objetivo de resolver los posibles fallos y lograr la satisfacción del cliente.

Conclusiones

Una vez terminado todo el proceso de trabajo se ha podido demostrar, que la creación de un componente de reconocimiento de rostros mediante el uso de *Eigenfaces* para el sistema general de identificación facial, podrá brindarle una mayor robustez, con la afirmación de que este componente da solución a la situación problemática que lo originó y se logra el cumplimiento del objetivo general trazado inicialmente así como los objetivos específicos.

- Se fundamentaron las metodologías y las tecnologías utilizadas en el desarrollo del componente, empleando como metodología de desarrollo XP, para la consulta y actualización de la base de datos ORM Telerik OpenAccess. Como lenguajes se utilizaron C# en el entorno de desarrollo Visual Studio, como lenguaje de modelado UML en la herramienta Visual Paradigm y como SGBD PostgreSQL.
- Se fundamentó acerca de los diferentes algoritmos existentes para la obtención de las características del rostro logrando enfatizar las particularidades del método PCA para su utilización en la etapa de extracción de características del componente.
- Se realizó el estudio del estado del arte a nivel nacional e internacional para obtener información acerca de sistemas ya implementados con las condiciones requeridas por el componente.
- Se definieron las principales funcionalidades y los requisitos no funcionales que el componente debe cumplir.
- Se describió la arquitectura n capas y los patrones de diseño GRASP y GoF.
- La implementación del componente cumplió con los requerimientos definidos.
- Las pruebas (unitarias, aceptación, rendimiento, falso rechazo y falsa aceptación) realizadas a la solución permitieron corregir errores en la implantación, logrando así que los resultados finales fueran satisfactorios.

Recomendaciones

Durante la realización de la investigación surgieron ideas que pueden servir como recomendación para el perfeccionamiento del componente, ellas son:

- Continuar el estudio de los diferentes métodos de reconocimiento facial expuestos en el capítulo 1, para el desarrollo de otros componentes que integren el sistema general de identificación facial.
- Integrar este componente a futuros sistemas de identificación que se desarrollen en el Departamento de Biometría del CISED.

Referencias Bibliográficas

1. *"Identification of Human faces".* **A.J. Goldestein, L.D. Harmon, and A.B. Lesk.** No.5, May 1971, Vols. 59. 748-760.

2. *"A Low-Dimensional Procedure for the Chatecterization of Human Faces".* **Kirby, L. Sirovich and M.** 1987.

3. **M. A Turkand, A.P Pentland.** *"Face Recognition Using Eigenfaces".* s.l. : Proc IEEE., 1991. 586-591.

4. **CENATAV.** Sitio oficial de CENATAV. [Online] 2004. [Cited: Diciembre 3, 2013.] www.cenatav.co.cu.

5. **CENATAV.** [Online] [Cited: Diciembre 3, 2013.] http://www.cenatav.co.cu/index.php/publications-cenatav/blue-series-publications .

6. J.C.P. & ASSOCIATES SOFTWARE, S.A DE C.V. [Online] [Cited: Enero 20, 2013.] http://www.jcpsoftinc.com/Biometricos.html.

7. Biometría. [Online] [Cited: Enero 15, 2013.] http://www.biometria.gov.ar/metodos-biometricos/facial.aspx.

8. **Duró, Virginia Espinosa.** *'Special Issue on Automated Biometric Systems, Proceedings.* s.l. : September 1997. Vol 85, No 9.

9. **Jiménez, Carmen Virginia Gámez.** *Diseño y Desarrollo de un Sistema de Reconocimiento de Caras.* Madrid : s.n., Abril, 2009.

10. **Giménez, Luis Lorente.** REPRESENTACIÓN DE CARAS MEDIANTE EIGENFACES.

11. **Marcelo, J.** *"Análisis comparativo de métodos basados en subespacios aplicados al reconocimiento de caras".* 2006.

12. **Renaud, Seguie.** A Very Fast Adaptative Face Detection System . *International Conference on Visualization, Imaging and Image Processing.* 2004.

13. **L., Sirovich y M., Kirby.** *"Low dimensional procedure for the characterization of human faces.Journal of Optical Society of America".* 1997. págs. 519-524..

14. **H., Rowley, S., Baluja and T., Kanade.** Neural network-based face detection . s.l. : IEEE Transactions on Pattern Analysis and Machine Intelligence. , 1998. Vol. Vol. 20, No.1. págs. 23–38..

15. **R., Schapire.** "Strength of weak learnability", Journal of Machine Learning. Vol. 5.

16. **Y., Freund,.** "Boosting a weak learning algorithm by majority", In Annual Workshop on Computational Learning Theory. 1990. págs. 202 – 216..

17. **Y., Freund, y R. E., Schapire,.** “A decision-theoretic generalization of on-line learning and an application to boosting”, Journal of Computer and System Sciences. . 1997. Vol. 55. págs. 119 – 139.

18. **Schapire., Yoav Freund and Robert E.** A decision-theoretic generalization of on-line learning and an application to boosting. Journal of Computer and System Sciences. 1997. 119_139 .

19. **P, Viola y M, Jones.** "Rapid object detection using boosted cascade of simple features," . s.l. : IEEE Computer Society Conference on Computer Vision and Pattern Recognition., 2001. Vol. 1. págs 511- 518.

20. **P., Viola y M., Jones.** “Robust real time object detection” . s.l. : In IEEE ICCV Workshop on Statistical and Computational Theories of Vision, Vancouver., 2001.

21. **P., Viola y M., Jones,.** “Robust Real-Time Face Detection”. International Journal of Computer Vision. 2004. Vol. 57, No.2. págs. 137-154..

22. **D., Valentin, H., Abdi, A., O’Toole y G., Cottrell.** “Connectionist models of face processing: a survey”, Pattern Recogn. 1994. Vol. 27. págs 1208–1230..

23. **M., Turk y A., Pentland.** “Eigenfaces for Recognition”, Journal of Cognitive Neurosicenc. 1991. Vol. 3, No.1. págs. 71-86.

24. **H., Moon y P.J., Phillips.** “Computational and Performance aspects of PCA-based Face Recognition Algorithms”, Perception. . 2001. Vol. 30. págs. 303-321..

25. **K., Etemad y R., Chellappa.** “Discriminant Analysis for Recognition of Human Face Images”, Journal of the Optical Society of America. . August 1997. Vol. 14, No. 8. págs. 1724-1733..

26. **W., Zhao, R., Chellappa y A., Krishnaswamy.** “Discriminant Analysis of Principal Components for Face Recognition”. s.l. : in Proc. of the 3rd IEEE International Conference on Face and Gesture Recognition. , April 1998. págs. 336-341.

27. **M.S., Bartlett, J.R., Movellan y T.J., Sejnowski.** “Face Recognition by Independent Component Analysis”. s.l. : IEEE Trans. on Neural Networks., November 2002. Vol. 13, No.6. págs. 1450-1464..

28. **C., Liu y H., Wechsler.** “Comparative Assessment of Independent Component Analysis (ICA) for Face Recognition”, in Proc. of the Second International Conference on Audio- and Video-based Biometric Person Authentication. March 1999. págs. 211-216..

29. *INTERIOR, INSTITUTO UNIVERSITARIO DE INVESTIGACIÓN SOBRE SEGURIDAD. EMPLEO DE SISTEMAS BIOMÉTRICOS PARA EL ECONOCIMIENTODE PERSONAS EN AEROPUERTOS .* Mayo 2006.

30. **Bolme, D, y otros.** The CSU Face Identification Evaluation System: Its Purpose, Features and Structure. 2003. . 304-311..

31. **Adrian Soto-Giróna, Vidal García-Martíneza, Jorge Servín-Pereza, Carlos Barrón-Romero y Rafael Felipe Monroy Pérez.** Imagen promedio de un conjunto de rostros. [Online] Disponible en: http://scma.cua.uam.mx/Documents/book/sotorost.

32. **Pentland., M. Turk y A.** *Eigenfaces for Recognition J. Cognitive Neuroscience.* 1991. 71-86.

33. **José Francisco Vélez Serrano, Ana Belén Moreno Díaz, Ángel Sánchez Calle, José Luis Esteban Sánchez-Marín.** "Introducción a clasificadores". 2002-2003.

34. **Tesisall, Proyecto.** Análisis de Reconocimiento de Patrones Biométricos .

35. **Franco Chichizola, Armando De Giusti , Marcelo Naiouf.** *"Eigenfaces de Imagen Reducida' para el Reconocimiento Automático de Rostros".* 2003.

36. **Edwin Omar Ortiz G, Henry Argüello Fuentes.** *Reconocimiento de rostros utilizando PCA en dispositivo DSP.* Bucaramanga, Santander,COLOMBIA : s.n.

37. **Angel Gil, Maria Benavides, Yessika Guilarte, Miguel Márquez.** *Sistema para el reconocimiento e identificación de rostros a través de fotografías.* 2008.

38. **Batur, Flinchbaugh, Hayes.** A DSP-based approach for the implementation of face recognition algorithms. s.l. : Proceedings of Seconds Canadian Conference : International Conference, 2003. . 330-338.

39. **. Mikaela Bauzá, Joaquin Vanschoren,Marcelo P. Funes, Gabriel M. Barrera, Daniela López De Luise.** Sistema de Autentificación Facial.

40. DATYS. [Online] [Cited: Febrero 5, 2013.] www.datys.cu.

41. **Figueroa, Roberth G., Solís, Camilo J. y Cabrera, Armando A.** METODOLOGIAS TRADICIONALES VS. METODOLOGIAS AGILES. . [Online] [Cited: Enero 17 , 2013.]

42. **Kruchten, Philippe.** The Rational UnifiedProcess An Introduction. [Online] Addison WesleY, 2001.

43. **Kroll, Per y Kruchten, Philippe.** The Rational Unified Process Made Easy :A Practitioner's Guide to the Rup. [Online] Addison-Wesley Professional, 2003.

44. **Letelier, Patricio.** Rational Unified Process (RUP). [Online] 17 de Enero de 2013.]. [Cited: Enero 17, 2013.]

45. **Jacobson, Ivar, Booch, Grady y Rumbaugh, James.** "El Proceso Unificado de Desarrollo de Software". . 2000. .

46. Canos, Jose H., Letelier, Patricio y Penades, Marıa Carmen. Metodologıas Agiles en el Desarrollo de Software. . [Online] [Cited: Enero 17 , 2013.] http://www.funtec.org.ar/pdf/Pdf_semin_completo_MetodologiasAgiles.pdf..

47. **Ramos, Isidro Salavert y Lozano, María Dolores Pérez.** Entornos de Desarrollo Integrado. Ingeniería Del Software Y Bases de Datos. Tendencias Actuales . s.l. : Univ de Castilla La Mancha, 2000.

48. **Avery, James.** Visual Studio Hacks Tips & Tools for Turbocharging the IDE. s.l. : O'Reilly Media, 2005.

49. **Dunaway, Robert B.** The Book of Visual Studio.Net: Aguide for Developers . s.l. : No Starch.

50. **Cerezo, Yolanda López, y otros.** Iniciación a la programación en C#: un enfoque práctico . s.l. : Delta Publicaciones, 2006.

51. **Rumbauhg, James, Jacobson, Ivar y Booch, Grady.** El lenguaje Unificado de Modelado.Manual de referencia. s.l. : Addison Wesley , 2007.

52. Visual Paradigm for UML. [Online]

53. **Bradski, Gary &Kaehler, Adrian.** "Learning OpenCV". s.l. : O'Reilly, 1ª edición, ., 2008.

54. EMGU CV. [Online] Enero 15, 2013.

55. CAVSI. Sistema Gestor de base de datos. [Online] Enero 6, 2013. http://www.cavsi.com/preguntasrespuestas/que-es-un-sistema-gestor-de-bases-de-datos-o-sgbd/..

56. **Denzer, Patricio.** PsotgreSQL] . [Online] Octubre 23, 2013. http://profesores.elo.utfsm.cl/~agv/elo330/2s02/projects/denzer/informe.pdf.

57. **Quiñones, Ernesto.** Introducción a Postrgesql. [Online] [Cited: Enero 13, 2013.]

58. **4., .NET Framework.** [Online]

59. Telerik OpenAccess. [Online] http://www.telerik.com/products/orm.aspx.

60. **Alonso, Fernando Amo, Normand, Martínez A. and Segovia, Francisco Javier Pérez.** El Proceso Unificado está centrado en la arquitectura. Introducción a la ingeniería del software. s.l. : Delta Publicaciones, 2005 . 337-338.

61. **César de la Torre Llorente, Unai Zorrilla Castro,Miguel Angel Ramos Barros, Javier Calvarro Nelson.** *Guía de arquitectura N-Capas orientada al dominio con .Net 4.0.* Marzo, 2010.

62. **Larman, Craig. UML y Patrones. Introducción al análisis y diseño orientado a objetos.** s.l. : México : Prentice Hall, 1999.

Bibliografía

". **A.J. Goldestein, L.D. Harmon, and A.B. Lesk.** *"Identification of Human faces*No.5, May 1971, Vol. 59.

Kirby, L. Sirovich and M. *"A Low-Dimensional Procedure for the Chatecterization of Human Faces".* 1987.

M. A Turkand, A.P Pentland. *"Face Recognition Using Eigenfaces".* s.l. : Proc IEEE., 1991.

H., Rowley, S., Baluja and T., Kanade. Neural network-based face detection . s.l. : IEEE Transactions on Pattern Analysis and Machine Intelligence. , 1998. Vol. Vol. 20.

R., Schapire. "Strength of weak learnability", Journal of Machine Learning. Vol. 5.

Y., Freund,. "Boosting a weak learning algorithm by majority", In Annual Workshop on Computational Learning Theory. 1990.

P., Viola y M., Jones,. "Robust Real-Time Face Detection". International Journal of Computer Vision. 2004. Vol. 57.

D., Valentin, H., Abdi, A., O'Toole y G., Cottrell. "Connectionist models of face processing: a survey", Pattern Recogn. 1994.

M., Turk y A., Pentland. "Eigenfaces for Recognition", Journal of Cognitive Neurosicenc. 1991. Vol. 3.

H., Moon y P.J., Phillips. "Computational and Performance aspects of PCA-based Face Recognition Algorithms", Perception. . 2001. Vol. 30.

K., Etemad y R., Chellappa. "Discriminant Analysis for Recognition of Human Face Images", Journal of the Optical Society of America. . August 1997. Vol. 14.

M.S., Bartlett, J.R., Movellan y T.J., Sejnowski. "Face Recognition by Independent Component Analysis". s.l. : IEEE Trans. on Neural Networks., November 2002. Vol. 13.

Jacobson, Ivar, Booch, Grady y Rumbaugh, James. *"El Proceso Unificado de Desarrollo de Software".* 2000.

Don Wells. 2009. Extreme Programming. [En línea] 2009. http://www.extremeprogramming.org/.

FILIBERTO. Implementación de un componente de software para la visualización tridimensional de Neuro- imágenes 2009.

Jacobson, Ivar, Booch, Grady y Rumbaugh, James. El Lenguaje Unificado de Modelado. 2000.

Glosario de Términos

Ado .Net: Es un conjunto de componentes de *software* que forman parte de la biblioteca de clases bases del Framework .Net de Microsoft y permiten acceder y modificar los datos almacenados en un Sistema Gestor de Base de Datos Relacional.

Eigenfaces: Término en inglés por el cual se conoce al algoritmo de reconocimiento facial basado en el análisis de componentes principales (PCA).

ORM: El mapeo objeto-relacional es una técnica de programación para convertir datos entre el sistema de tipos utilizado en un lenguaje de programación orientado a objetos y el utilizado en una base de datos relacional, utilizando un motor de persistencia.

Pixel: es la menor unidad homogénea que forma parte de una imagen digital, ya sea esta una fotografía, un fotograma de video o un gráfico.

PCA: Análisis de Componentes Principales o Eigenfaces: es la técnica basada en el aspecto, usada extensamente para la reducción de dimensionalidad y ha registrado gran interpretación en reconocimiento de cara.

Software: Es el conjunto de los programas de cómputo, procedimientos, reglas, documentación y datos asociados que forman parte de las operaciones de un sistema de computación.

XML (Extensible Markup Language, 'lenguaje de marcas extensible'): Metalenguaje extensible de etiquetas desarrollado por el World Wide Web Consortium (W3C). Permite definir la gramática de lenguajes específicos, por lo tanto XML no es realmente un lenguaje en particular, sino una manera de definir lenguajes para diferentes necesidades. Algunos de estos lenguajes que usan XML para su definición son XHTML, SVG y MathML.

Eigenvector: Los vectores propios o auto-vectores de un operador lineal son los vectores no nulos que, cuando son transformados por el operador, dan lugar a un múltiplo escalar de sí mismos, con lo que no cambian su dirección.

Eigenvalue: Valor asociado a cada eigenvector.

Anexos

Anexo 1. Historias de Usuarios

Historia de usuario 2

Tabla 8.HU_2 Realizar el entrenamiento.

Historia de usuario	
Numero: HU_2	Nombre de historia de usuario: Realizar el entrenamiento.
Usuario : Sairenys Barrios Pérez y Herbert Tamayo Valero	
Prioridad de negocio: Alta	Puntos estimados: 3
Riesgo en desarrollo: Alta	Iteraciones : 2
Descripción: se le aplica el método PCA al conjunto de imágenes para obtener la *Eigenfaces*. Se calcula el rostro promedio. Todas las imágenes de la BD se proyectan contras las *Eigenfaces* para determinar el vector de peso, el cual se guardará en la BD del sistema.	
Observaciones: el número de *Eigenfaces* no puede ser mayor que el número de imágenes del grupo de entrenamiento.	

Historia de usuario 3

Tabla 9. HU_3 Identificar persona.

Historia de usuario	
Numero: HU_3	Nombre de historia de usuario: Identificar persona.
Usuario : Sairenys Barrios Pérez y Herbert Tamayo Valero	
Prioridad de negocio: Alta	Puntos estimados: 3
Riesgo en desarrollo: Alta	Iteraciones : 3

Descripción: a la imagen de prueba se le realiza el pre-procesado, se calculan los coeficientes (con el rostro promedio y aplicando el método PCA para las *Eigenfaces*) y se compara con los coeficientes de entrenamiento guardados en la base de datos. Esta comparación se hace a través de la distancia euclidiana.
Observaciones: en el pre-procesado se verifica que sea un rostro válido.

Anexo 2. Plan de iteraciones

Tabla 10. Plan de iteraciones

Iteración	No.HU	Historia de Usuario	Duración estimada (semanas)
Iteración 1	HU_1	Pre-procesar imágenes de entrenamiento	2
Iteración 2	HU_2	Realizar el entrenamiento	7
Iteración 3	HU_3	Identificar persona	8

Anexo 3. Tarjetas CRC

Tabla 11. Tarjeta CRC PersonaIdentificada.

PersonaIdentificada	
Descripción	Colaboradores
Es la clase para almacenar los datos de la persona identificada, esta clase hereda de la clase interfaz IPersonaIdentificada.	✓ IpersonaIdentificada.cs ✓ EMGU_CV.dll

Tabla 12. Tarjeta CRC Serialización.

Serialización	
Descripción	Colaboradores
En esta clase estarán las funcionalidades necesarias para serializar los datos obtenidos productos a las proyecciones en la etapa de entrenamiento, con el objetivo de almacenarlos en la base de datos.	

Tabla 13. Tarjeta CRC Conexión.

Conexión	
Descripción	Colaboradores
En esta clase estarán todas las funcionalidades necesarias para la consulta y actualización de datos, haciendo uso de ORM Telerik OpenAccess.	Serializacion Telerik OpenAccess EMGU_CV.dll

Tabla 14. Tarjeta CRC Detección.

Detección	
Descripción	Colaboradores
Es la clase encargada de detectar que la imagen de la persona a identificar, sea una imagen de un rostro.	✓ Entrenamiento.cs ✓ EMCU_CV.dll

Tabla 15. Tarjeta CRC Normalización.

Normalización	
Descripción	Colaboradores
Clase que contiene las funcionalidades para lograr que la imagen del individuo a identificar se encuentre normalizada en cuanto a: • Brillo y contraste. • Tamaño de la imagen. • Rotación de la imagen.	✓ EMGU_CV.dll

Tabla 16. Tarjeta CRC Entrenamiento

Entrenamiento	
Descripción	Colaboradores
Clase encargada de realizar el entrenamiento del componente con el conjunto de imágenes de las personas, haciendo uso del método PCA, para poder realizar la identificación.	✓ Serialización.cs ✓ EMGU_CV.dll ✓ Conexión.cs

Tabla 17. Tarjeta CRC Clasificación

Clasificación	
Descripción	Colaboradores
Es la clase que tiene todas aquellas funcionalidades para lograr la identificación de un individuo, a partir de los datos obtenidos de la imagen de la persona a identificar y los obtenidos en el entrenamiento.	✓ PersonaIdentificada.cs ✓ EMCU_CV.dll ✓ Conexión.cs

Anexo 4. Modelo de clases del diseño

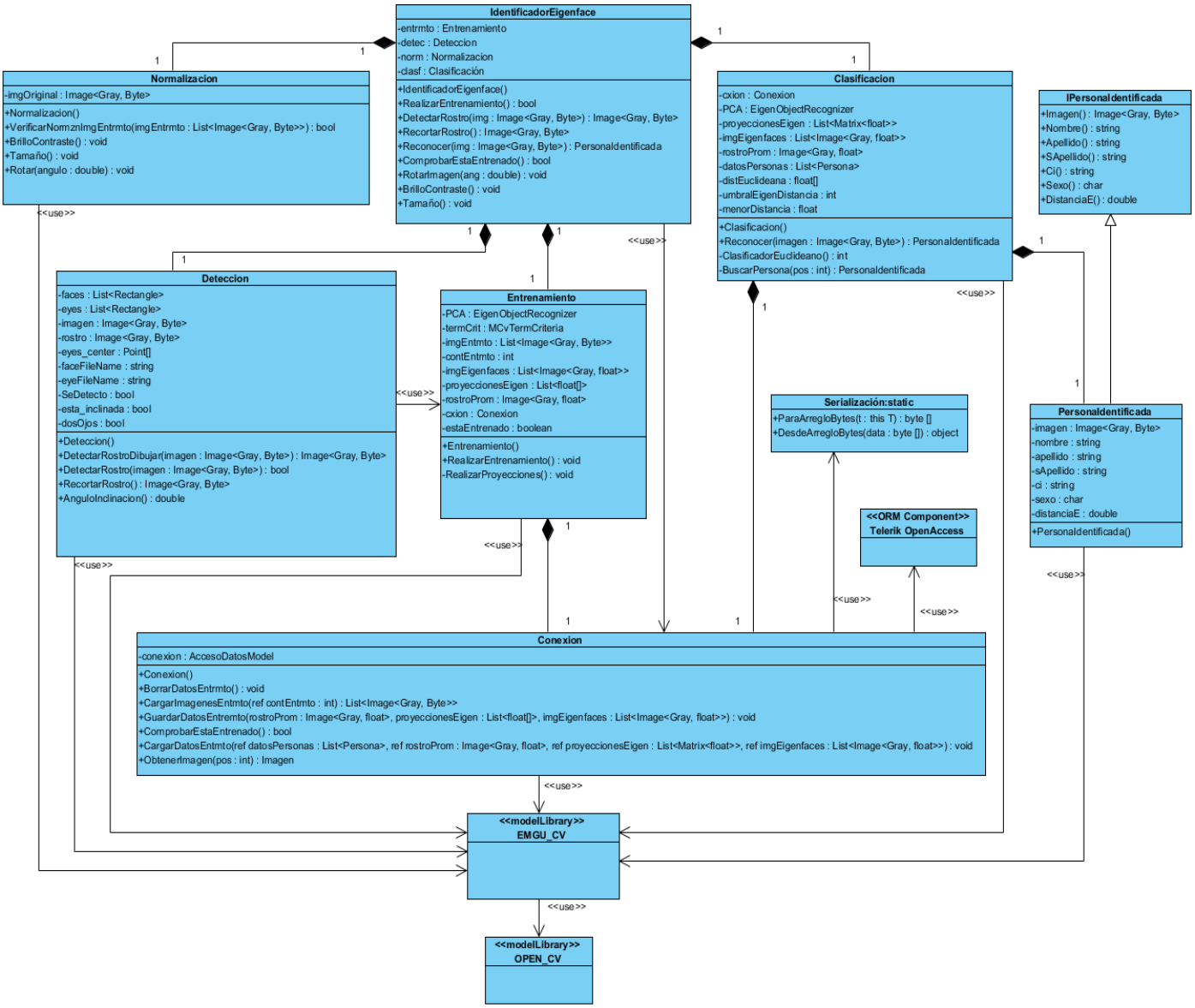

Figura 26. Diagrama de clases.

Anexo 5. Descripción de las tablas (BD)

Tabla 18. Descripción de la tabla persona (Base de datos)

Nombre	Persona	
Descripción	Contiene los datos correspondientes a todas las personas.	
Atributo	Tipo	Descripción
id_persona	int	Identificador de la persona
sexo	char	Sexo de la persona “Masculino o Femenino”
sapellido	varchar	Segundo apellido de la persona
apellido	varchar	Primer apellido de la persona
nombre	varchar	Nombre de la persona
ci	numeric	Carnet de identidad de la persona

Tabla 19. Descripción de la tabla imagen (Base de datos)

Nombre	Imagen	
Descripción	Contiene los datos correspondientes a todas las imágenes.	
Atributo	Tipo	Descripción
id_imagen	int	Identificador de la imagen
id_persona	int	Llave foránea obtenida de la relación de 1 a mucho que tiene con la tabla persona.
dirección_f	varchar	Dirección física de donde son cargadas las imágenes.
vector	bytea	Coeficientes de entrenamiento para la imagen, producto de la proyección con las *Eigenfaces*.

c_entrenamiento	bool	Representa si la persona está o no en el conjunto de entrenamiento.

Tabla 20. Descripción de la tabla Eigenface (Base de datos)

Nombre	Eigenface	
Descripción	Contiene todas las *Eigenfaces* obtenidas en el entrenamiento.	
Atributo	Tipo	Descripción
id_eigen	int	Identificador de la *Eigenface.*
Id_imagen	int	Llave foránea obtenida de la relación de 1 a mucho que tiene con la tabla imagen.
valor_eigen	bytea	Valor de la *Eigenface.*

Tabla 21.Descripción de la tabla rostro promedio (Base de datos).

Nombre	Rostro promedio	
Descripción	Contiene todo lo relacionado con el rostro promedio obtenido en el entrenamiento.	
Atributo	Tipo	Descripción
id_rostro	int	Identificador del rostro promedio.
valor_rostro	int	Valor del rostro promedio.

Anexo 6. Tareas de ingeniería

Tabla 22. Tareas de ingeniería.

Iteración	Historia de usuario	Tareas
1	Pre-procesar imágenes de entrenamiento	-Verificar que sea una imagen de rostro. -Normalizar brillo y contraste. -Normalizar el tamaño. -Corregir la rotación.

2	Realizar el entrenamiento	-Calcular las *Eigenfaces*. -Calcular los coeficientes de todas las imágenes almacenadas a partir de las *Eigenfaces* obtenidas.
3	Identificar persona	-Pre-procesar la imagen de prueba. -Calcular los coeficientes. -Comparar los coeficientes con los coeficientes de las imágenes de entrenamiento. -Verificar si la persona es identificada o no.

Anexo 7. Especificación tareas de ingeniería

Tabla 23. HU1_T1. Verificar que sea una imagen de rostro.

Tarea	
No. de tarea: HU1_T1	**Historia de usuario:** Pre-procesar imágenes de entrenamiento
Nombre de la tarea: Verificar que sea una imagen de rostro.	
Tipo de tarea: Desarrollar	**Puntos estimados:** 2
Fecha inicio: 15/ 1 /2013	**Fecha fin: 30/1/2013**
Programador responsable: Herbert Tamayo Valero.	
Descripción: se utilizó le método de Viola y Jones, el cual utiliza la clase CascadeClassifier y se entrena con un archivo XML para que reconozca un rostro. Devuelve una región rectangular donde probablemente se encuentre un rostro.	

Tabla 24. HU1_T2.Normalizar el brillo y contraste.

Tarea	
No. de tarea: HU1_T2	**Historia de usuario:** Pre-procesar imágenes de entrenamiento
Nombre de la tarea: Normalizar el brillo y contraste.	

Tipo de tarea: Desarrollar	**Puntos estimados:** 2
Fecha inicio: 15/1/2013	**Fecha fin: 30/1/2013**
Programador responsable: Herbert Tamayo Valero.	
Descripción: después de llevar la imagen a escala de grises, se le aplica la función _EqualizeHist () la cual normaliza el histograma.	

Tabla 25.HU1_T3. Normalizar el tamaño.

Tarea	
No. de tarea: HU1_T3	**Historia de usuario:** Pre-procesar imágenes de entrenamiento
Nombre de la tarea: Normalizar el tamaño.	
Tipo de tarea: Desarrollar	**Puntos estimados:** 2
Fecha inicio: 15/1/2013	**Fecha fin: 30/1/2013**
Programador responsable: Herbert Tamayo Valero.	
Descripción: se normaliza el tamaño a 150x150 usando el método Resize ().	

Tabla 26. HU1_T4. Corregir la rotación.

Tarea	
No. de tarea: HU1_T4	**Historia de usuario:** Pre-procesar imágenes de entrenamiento
Nombre de la tarea: Corregir la rotación.	
Tipo de tarea: Desarrollar	**Puntos estimados:** 2
Fecha inicio: 15/1/2013	**Fecha fin: 30/1/2013**
Programador responsable: Herbert Tamayo Valero.	
Descripción: se detecta el ángulo de inclinación con los puntos medios del rectángulo de cada ojo, y se utiliza el método Rotate () para rotar la imagen de acuerdo al ángulo de inclinación.	

Tabla 27. HU2_T1.Calcular las *Eigenfaces*.

Tarea	
No. de tarea: HU2_T1	**Historia de usuario:** Realizar el entrenamiento.
Nombre de la tarea: Calcular las *Eigenfaces*.	
Tipo de tarea: Desarrollar	**Puntos estimados:** 2
Fecha inicio: 30/1/2013	**Fecha fin: 18/3/2013**
Programador responsable: Herbert Tamayo Valero.	
Descripción: se realiza el PCA con el que se extraen las *Eigenfaces*, mediante los siguientes pasos: • Se calcula de la rostro promedio y se resta a todos los rostros normalizados. • Se forma la matriz Mx cuyas columnas son las imágenes del conjunto de entrenamiento normalizadas. • Se calcula la matriz de los coeficientes (las *Eigenfaces*).	

Tabla 28.HU2_T2. Calcular los coeficientes de entrenamiento.

Tarea	
No. de tarea: HU2_T2	**Historia de usuario:** Realizar el entrenamiento.
Nombre de la tarea: Calcular los coeficientes de todas las imágenes almacenadas a partir de las *Eigenfaces* obtenidas.	
Tipo de tarea: Desarrollar	**Puntos estimados:** 2
Fecha inicio: 30/1/2013	**Fecha fin: 18/3/2013**
Programador responsable: Herbert Tamayo Valero.	
Descripción: se hace la proyección de cada imagen sobre las *Eigenfaces* obtenidas, o sea, se realiza el producto escalar de cada imagen sobre cada una de las *Eigenfaces*.	

Tabla 29. HU3_T1. Pre-procesar imagen de prueba.

Tarea	
No. de tarea: HU3_T1	**Historia de usuario:** Identificar persona.
Nombre de la tarea: Pre- procesar imagen de prueba.	
Tipo de tarea: Desarrollar	**Puntos estimados:** 2
Fecha inicio: 18/3/2013	**Fecha fin: 6/5/2013**

Programador responsable: Herbert Tamayo Valero.
Descripción: se le realiza a la imagen el mismo procedimiento de las tareas anteriores.

Tabla 30.HU3_T2. Calcular coeficientes de la imagen de prueba.

Tarea	
No. de tarea: HU3_T2	**Historia de usuario:** Identificar persona.
Nombre de la tarea: Calcular los coeficientes.	
Tipo de tarea: Desarrollar	**Puntos estimados:** 2
Fecha inicio: 18/3/2013	**Fecha fin: 6/5/2013**
Programador responsable: Herbert Tamayo Valero.	
Descripción: se realiza el mismo procedimiento de la tarea 2.	

Tabla 31. HU3_T3. Comparar los coeficientes.

Tarea	
No. de tarea: HU3_T3	**Historia de usuario:** Identificar persona.
Nombre de la tarea: Comparar coeficientes con todos los coeficientes de las imágenes de entrenamiento.	
Tipo de tarea: Desarrollar	**Puntos estimados:** 2
Fecha inicio: 18/3/2013	**Fecha fin: 6/5/2013**
Programador responsable: Herbert Tamayo Valero.	
Descripción: se compara el vector $\mathbf{Y_{TEST}}$ formado por las proyecciones de la imagen de prueba sobre las *Eigenfaces* con cada uno de los vectores $\mathbf{Y_{ENTi}}$. El criterio que se utiliza es el de la menor distancia euclidiana, es decir, menor $\mid \mathbf{Y_{TEST}} - \mathbf{Y_{ENTi}} \mid$.	

Tabla 32.HU3_T4. Mostrar si el rostro es conocido o no.

Tarea	
No. de tarea: HU3_T4	**Historia de usuario:** Identificar persona.
Nombre de la tarea: Verificar si la persona es identificada o no.	
Tipo de tarea: Desarrollar	**Puntos estimados:** 2

Fecha inicio: 18/3/2013	Fecha fin: 6/5/2013
Programador responsable: Herbert Tamayo Valero.	
Descripción: en caso de ser un rostro conocido se muestran los datos almacenados en la base de datos, en caso contrario se muestra la información: “Persona desconocida”.	

Anexo 8. Pruebas unitarias

```
        /// <summary>
        ///Una prueba de DetectarRostroDibujar
        ///</summary>
        [TestMethod()]
        public void DetectarRostroDibujarTest()
        {
            Detección target = new Detección(); // TODO: Inicializar en un valor adecuado
            Image<Gray, byte> imagen = new Image<Gray, byte>(@"D:\Usuarios\Sierpe\proyecto
            Image<Gray, byte> actual;
            actual = target.DetectarRostroDibujar(imagen);
            Assert.IsNotNull(actual,"No se detecto rostro en la imagen");
        }
    }
}
```

Resultados de pruebas

Herbert@PC-BIOMETRIA29 2013-05 | Ejecutar | Depurar | Agrupar por:

Ejecución de pruebas completado Resultados: 1/1 correctas; Elementos comprobados: 0

Resultado	Nombre de la prueba	Proyecto	Mensaje de error
Pasada	DetectarRostroDibujarTest	pruebaUnitaria	

Lista de errores | Resultados | Resultados de pruebas

Figura 27. Prueba unitaria para el método Detectar Rostro.

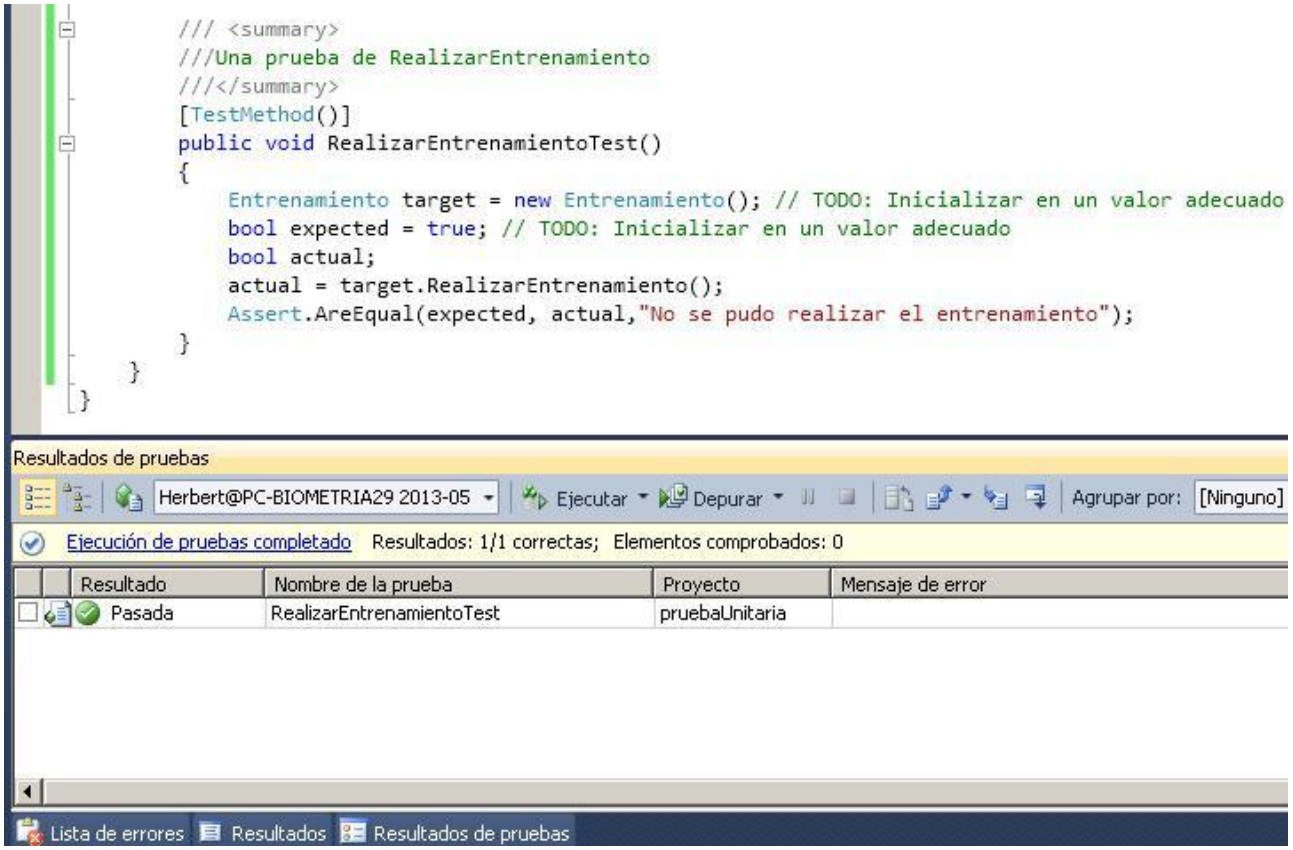

Figura 28. Prueba unitaria para el método Realizar Entrenamiento

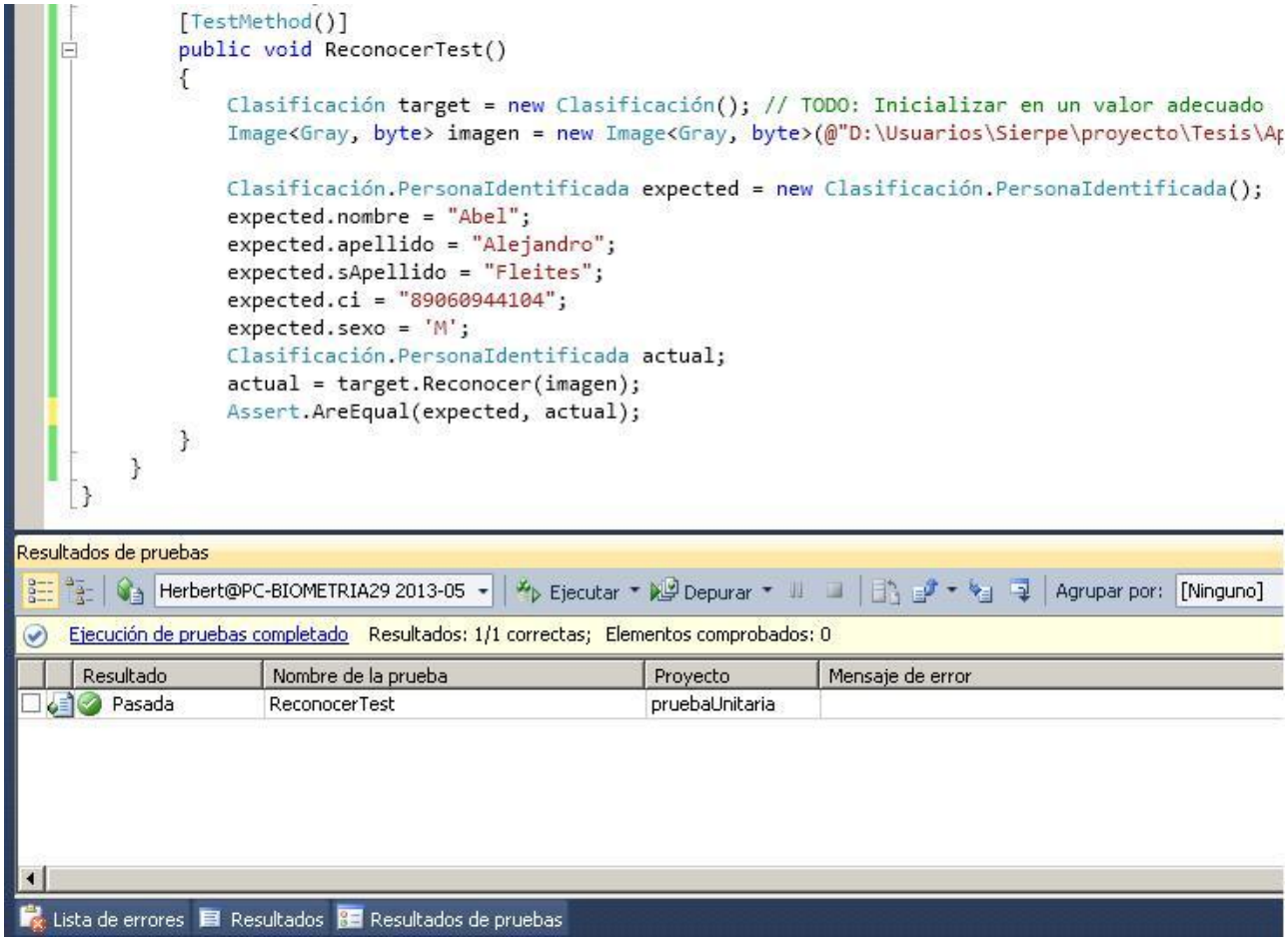

Figura 29. Prueba unitaria para el método Reconocer

Anexo 9. Pruebas de aceptación

Tabla 33. HU2_CP2 Prueba de funcionalidad para realizar el entrenamiento.

Caso de prueba de aceptación	
Código de caso de prueba: HU2_CP2	Nombre de historia de usuario: Realizar el entrenamiento.
Responsable de la prueba: Herbert Tamayo Valero.	
Descripción de la prueba: prueba de funcionalidad para realizar el entrenamiento de las imágenes cargadas	

de la base de datos pre-procesadas.
Condiciones de ejecución: se debe tener el conjunto de imágenes pre-procesadas.
Entrada/Pasos de ejecución: se tiene el conjunto de imágenes de entrenamiento pre-procesadas y a partir de ello se realizan procedimientos matemáticos para la obtención de: - *Eigenfaces*. - Vectores de todas las imágenes almacenadas a partir de las proyecciones con las *Eigenfaces* obtenidas.
Resultado esperado: se obtienen los coeficientes de cada una de las imágenes de entrenamiento como producto de la proyección de las *Eigenfaces* con cada una de las imágenes de entrenamiento.
Evaluación de la prueba: Prueba satisfactoria.

Tabla 34. HU3_CP3 Prueba de funcionalidad para identificar una persona.

Caso de prueba de aceptación	
Código de caso de prueba: HU3_CP3	Nombre de historia de usuario: Identificar persona
Responsable de la prueba: Herbert Tamayo Valero y Sairenys Barrios Pérez	
Descripción de la prueba: prueba de funcionalidad para identificar una persona que ha sido cargada.	
Condiciones de ejecución: la imagen de la persona entrada debe de tener el rostro frontal.	
Entrada/Pasos de ejecución: a partir de la imagen cargada se debe de: -Pre-procesar la imagen de prueba. -Calcular los coeficientes. -Comparar los coeficientes con los coeficientes de las imágenes de entrenamiento. -Mostrar si el rostro es identificado o no.	
Resultado esperado: se muestran los datos de la persona que fue identificada, en caso contrario un mensaje: “Persona desconocida”.	

Evaluación de la prueba: Prueba satisfactoria.

Anexo 10. Pruebas de rendimiento

Etapa de entrenamiento

Esta prueba fue realizada para una base de datos de 23 personas, de las que se tomaron diferentes cantidades de imágenes por personas para realizar el entrenamiento del componente, la siguiente tabla muestra el tiempo (en segundo), que demora el entrenamiento para una cierta cantidad de imágenes por personas. Se muestra además la gráfica que hace referencia a las pruebas con sus respectivos datos.

Tabla 35. Tiempo de entrenamiento.

Imágenes / personas	Total de imágenes	Prueba 1	Prueba 2	Prueba 3
5	115	1.8	1.7	1.6
10	230	6.5	6.5	6.4
15	345	13.9	14	14
20	460	23.7	23.5	23.6

La tabla muestra la variación existente en la etapa de entrenamiento para diferentes cantidades de imágenes por personas. A partir de ahí podemos apreciar la representación gráfica (Figura 32) de cada uno de los valores y podemos observar que para una cantidad de 115 imágenes de entrenamiento el tiempo varía entre 1.6 y 1.8 segundos según la prueba, sin embargo, al aumentar la cantidad de imágenes para el entrenamiento, aumenta el tiempo respecto a la cantidad de imágenes utilizadas, pero la respuesta sigue siendo considerables para cada una de las pruebas (poca variación en los segundos de respuesta).

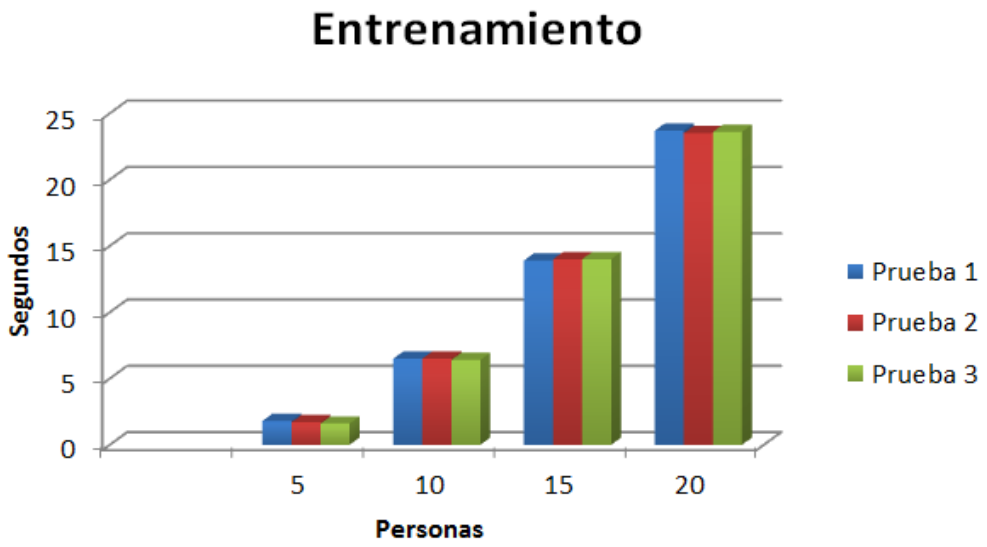

Figura 30. Gráfico de tiempo (Entrenamiento).

Etapa de prueba

A continuación se muestra la tabla con los tiempos tomados en segundos para la etapa de prueba del componente, donde a partir de la entrada al sistema de una imagen, se comparó contra una cierta cantidad de imágenes de diferentes personas almacenadas en la base de datos producto de la etapa de entrenamiento.

Tabla 36. Tiempo de prueba.

Imágenes/ personas	Total de imágenes	Prueba 1	Prueba 2	Prueba 3
5	115	0.01	0.02	0.009
10	230	0.015	0.014	0.023
15	345	0.025	0.019	0.038
20	460	0.04	0.04	0.044

Para esta etapa el tiempo de respuesta es considerable (Ver Figura 33), la variación existente entre las pruebas varía muy poco independientemente de la cantidad de imágenes que se tomaron. Si

observamos el gráfico de esta etapa podemos ver que el tiempo de respuesta más alto fue en la tercera prueba para un margen de 20 imágenes y no llega a 0.1 segundos. Esta etapa demuestra la velocidad del componente al identificar una persona.

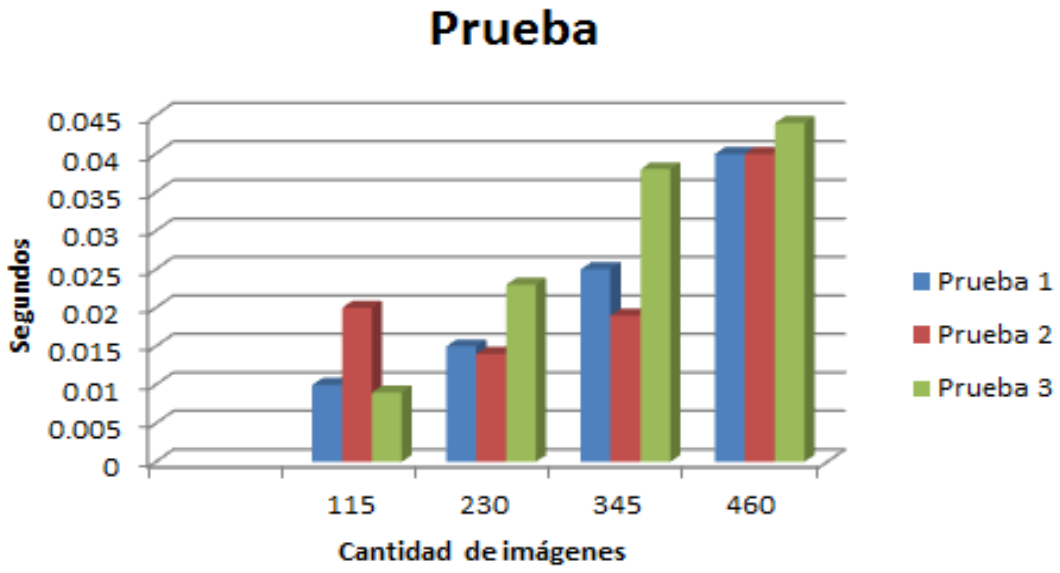

Figura 31. Gráfico de tiempo (Prueba).

Fase de detección

En esta etapa del componente se realizó la prueba utilizando imágenes de distintos tamaños para determinar el tiempo que se demora el método de Viola y Jones para la detección del rostro en la imagen (Ver Tabla 28). Los tiempos de respuesta demostraron la velocidad del método utilizado, dando diferentes tiempos para imágenes de diferentes tamaños y aun así se mantuvo con muy poca variación en los segundos como se observa en la gráfica de la Figura 34.

Tabla 37. Tiempo de detección.

Tamaño	Prueba 1	Prueba 2	Prueba 3
100x100	0.15	0.14	0.15
150x150	0.18	0.19	0.17
640x480	1.05	0.94	0.96
1024x768	2.54	2.32	2.88
1900x1200	7.14	7.05	6.22

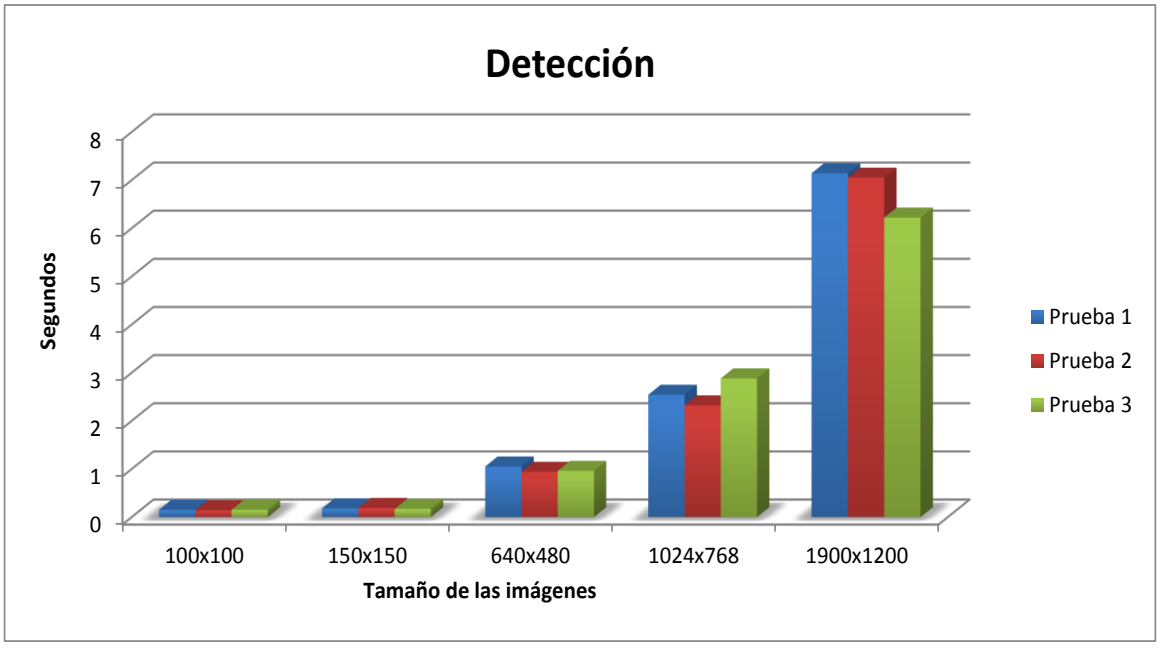

Figura 32. Gráfico de tiempo (Detección).