Universidad de las Ciencias Informáticas

Facultad 1

Título: "Sistema para la identificación de expresiones faciales"

Trabajo de Diploma para optar por el Título de Ingeniero en Ciencias Informáticas.

Autores:

Dayaris Flores Álvarez.

Luis Miguel Silva Arévalo.

Tutor:

Ing. Adrián Rivera Correa.

La Habana, 11 de Junio de 2013.

FRASE

"Mientras que nuestros pensamientos son totalmente privados, la mayoría de nuestras emociones se detectan por una señal distintiva que ayuda a los demás a comprender cómo nos sentimos."

Paul Ekman

DECLARACIÓN DE AUTORÍA

Declaramos que somos los únicos autores de este trabajo y autorizamos al Centro de Identificación y Seguridad Digital (CISED) de la Universidad de las Ciencias Informáticas (UCI) a hacer uso del mismo en su beneficio, con carácter exclusivo.

Para que así conste firmamos la presente a los __ días del mes de ___de 2013.

_____ _____

Autor: Dayaris Flores Álvarez. Autor: Luis Miguel Silva Arévalo.

_____

Tutor: Ing. Adrián Rivera Correa.

AGRADECIMIENTOS:

Queremos darle las gracias a nuestro tutor Adrián por ayudarnos en todo lo que estuvo a su alcance y brindarnos su apoyo así como su experiencia. Las gracias también a nuestros compañeros del departamento de Biometría, ya sea estudiante o profesor que de una forma u otra nos ayudaron en la realización de este trabajo de diploma. A nuestros familiares y demás amistades.

RESUMEN

Esta investigación está enmarcada en el proceso de desarrollo de un sistema F.E.R. (siglas de Facial Expression Recognition) capaz de reconocer expresiones faciales tales como felicidad, ira, tristeza, disgusto, sorpresa, miedo y estado neutral. Entre sus aplicaciones podemos encontrar el desarrollo de interfaces avanzadas entre las personas y las máquinas y la potenciación de las relaciones con el consumidor en el comercio electrónico ya que el ordenador sería capaz de identificar el nivel de interés de un cliente por un producto determinado e incluso su satisfacción después de realizada la compra.

Con el resultado de esta investigación la UCI contará con un sistema FER que puede ser adaptado a diversos proyectos y aplicaciones que pudieran enriquecerse con el mismo, tales como: sistemas de seguridad o de índole policial, videojuegos y softwares de aplicación comercial.

Palabras claves:

FER, facial expression recognition, identificación de expresiones faciales, extracción de características y redes neuronales.

ÍNDICE DE CONTENIDO:

ÍNDICE

ÍNDICE DE CONTENIDO:

ÍNDICE DE FIGURAS

ÍNDICE DE TABLAS

INTRODUCCIÓN

La interfaz de usuario para los sistemas de computadoras está evolucionando hacia una interfaz multimodal inteligente, es decir, desde las instrucciones vía-teclado a una forma más natural de interacción, usando los sentidos: visual y auditivo. Este es el primer paso en aras de lograr una comunicación más amigable entre el hombre y la máquina.

La comunicación humana tiene dos aspectos principales: verbal (auditivo) y no verbal (visual). Las palabras son las unidades atómicas de información de la comunicación verbal mientras que las expresiones faciales, los movimientos del cuerpo y las reacciones fisiológicas son las unidades atómicas de la comunicación no verbal.

Aunque está bastante claro que los gestos no verbales no son necesarios para una interacción humana exitosa (esto se evidencia en las llamadas telefónicas), abundante investigación en la psicología social ha mostrado que los gestos no verbales pueden usarse para sincronizar el diálogo, indicar comprensión o desacuerdo, suavizar el diálogo y disminuir las interrupciones en el mismo.

Como se indica en **(13)**, en la comunicación humana cara a cara sólo el 7% de los mensajes comunicativos es transmitido mediante el lenguaje lingüístico, el 38% es transmitido mediante el para lenguaje, o sea, el “cómo” se dice, mientras que el restante 55% es transmitido por expresiones faciales.

La meta de nuestra investigación es el desarrollo de un sistema automatizado inteligente para el análisis de las expresiones faciales. Las aplicaciones del mismo son diversas: desarrollo de interfaces avanzadas entre las personas y las máquinas y potenciación de las relaciones con el consumidor que compra a través de Internet. Las interfaces avanzadas hombre-máquina pueden enriquecerse con este software porque se podrían construir avatares que simularían realmente las expresiones del rostro de una persona, posibilidad que es realmente atractiva para sectores específicos como los videojuegos y los chat.

INTRODUCCIÓN:

El comercio electrónico también se podría beneficiar con esta tecnología, ya que en el momento de comprar por Internet, el ordenador sería capaz de identificar los gestos de la persona que se interesa por un producto, determinar su intención real de compra e incluso medir su nivel de satisfacción por el producto o servicio **(13).**

Debido a las diversas aplicaciones que posee este tipo de software y debido a que en la UCI no existe hasta el momento una herramienta como esta, el departamento de Biometría se ha dado a la tarea de desarrollar un sistema FER del cual sea propietario para así poder adaptarlo a diversos proyectos integrándoles interfaces hombre-máquina más inteligente. Después de un estudio de la situación antes descrita se plantea como **problema de investigación**: ¿Cómo identificar, en imágenes, expresiones faciales que permitan reconocer el estado de ánimo en una persona?

Definiéndose como **objeto de estudio**: la identificación de expresiones faciales en imágenes.

Para dar solución a la problemática planteada se ha definido como **objetivo general de la investigación** desarrollar un sistema que dada la imagen facial de una persona estime la posición de las características faciales más relevantes para determinar su expresión. Como **objetivos específicos:**

1. Realizar un estudio de los sistemas de identificación de expresiones faciales existentes en el mundo.
2. Identificar y aplicar los procedimientos asociados a la identificación de expresiones faciales.
3. Analizar y seleccionar las herramientas a utilizar en el desarrollo del sistema.
4. Realizar el diseño e implementación del sistema.
5. Realizar pruebas para verificar el correcto funcionamiento del sistema.

Tareas investigativas:

1. Identificación y análisis de las características de algunos sistemas de identificación de expresiones faciales que sirvan de base para el desarrollo de un sistema propio. [Responsable: Dayaris Flores Álvarez].
2. Definición y descripción de las características del sistema para establecer la base teórica

sobre la que se desarrollará este. [Responsable: Dayaris Flores Álvarez].

3. Análisis y caracterización de las herramientas, tecnologías y metodología a utilizar en el desarrollo del sistema. [Responsable: Luis Miguel Silva Arévalo].
4. Definición de la arquitectura con el fin de encontrar la mejor y más eficiente vía de solución para el sistema. [Responsable: Dayaris Flores Álvarez].
5. Diseño del sistema para obtener un modelo lógico del mismo. [Responsable: Luis Miguel Silva Arévalo].
6. Implementación del sistema para automatizar las funcionalidades requeridas. [Responsables: Luis Miguel Silva Arévalo].
7. Realización de las pruebas al sistema para validar su correcto funcionamiento. [Responsable: Luis Miguel Silva Arévalo].

Para desarrollar estas tareas se utilizarán los siguientes **métodos científicos**:

Métodos teóricos:

Analítico–sintético: con el objetivo de analizar y utilizar la información y la documentación más relevante para el desarrollo y la realización de este sistema.

Histórico–lógico: para constatar teóricamente y realizar un estudio de todo lo referente a la Identificación de Expresiones Faciales, la evolución y el nivel de desarrollo que tienen los sistemas.

Modelación: para representar por medios de diagramas los conceptos asociados a los procesos de Identificación de Expresiones Faciales, obteniendo como resultado un mejor entendimiento de la posible solución a implementar.

Métodos empíricos:

Entrevista: para obtener información interactuando con personas que tienen un mayor conocimiento sobre Identificación de Expresiones Faciales.

INTRODUCCIÓN:

La presente investigación se realiza debido a la necesidad de la universidad y del país de contar con un sistema FER, del cual sean propietarios y puedan así adaptarlo a diversos proyectos y aplicaciones que pudieran enriquecerse con el mismo. Con el resultado de esta investigación la UCI contará con un sistema FER que podrá integrar a proyectos que tengan un objetivo tanto de índole policial o de seguridad, como para software de aplicación comercial y de videojuegos.

Estructuración del contenido

Capítulo 1: Fundamentación teórica: En este capítulo se presenta una fundamentación teórica del sistema FER, se mencionan características de los sistemas ya existentes a nivel internacional y se analizan las tecnologías, herramientas y la metodología que se usarán para la solución del problema.

Capítulo 2: Características del sistema: En este capítulo se presenta un modelo general del sistema y una lista de las características a desarrollar en la aplicación con un plan de desarrollo para la realización de las mismas. Además se presentan diagramas de secuencias de algunas de las funcionalidades críticas y el diagrama de clases.

Capítulo 3: Implementación y prueba del sistema: En este capítulo se detalla el proceso de implementación y prueba. Se describe el estilo de codificación utilizado, se presenta el diagrama de componente que permite comprender la estructura del sistema y se muestran los resultados de las pruebas realizadas.

CAPÍTULO 1: Fundamentación teórica

Introducción

En este capítulo se presentan algunos conceptos básicos necesarios para el posterior entendimiento de los temas tratados en este trabajo. Además se abordan aspectos como: aplicación FER y proceso a seguir para la implementación del sistema FER; los cuales forman parte de los principios o cimientos en los que se apoya y se desarrolla la tesis. También se realiza un estudio de algunos sistemas FER existentes en el mundo y se define la metodología de desarrollo y herramientas a utilizar.

A continuación se abordarán algunos conceptos que son necesarios comprender para poder profundizar en la investigación.

Conceptos básicos:

Red Neuronal: Técnica de inteligencia artificial, paradigma de aprendizaje y procesamiento automático.

Análisis de FER

El análisis de expresiones faciales es un campo aún en desarrollo, a partir de la investigación realizada, no se tiene referencia de que en Cuba exista una aplicación así y en el mundo se encuentran pocos trabajos. Los que existen tienen aún un amplio margen de error y sólo son capaces de detectar emociones “puras”: enfado, disgusto, miedo, alegría, tristeza y sorpresa, además del estado neutral.

En este epígrafe se caracteriza brevemente algunos sistemas FER que han propuesto diferentes técnicas para la identificación de expresiones faciales.

El sistema *Facial Expression Recognition System using Statistical Feature and Neural Network* expone una técnica para reconocimiento de expresiones faciales que usa características estáticas de la imagen de la cara y clasifica la expresión usando una red neuronal con propagación hacia

atrás. Para demostrar su efectividad se ha usado la base de datos de rostros JAFFE y el programa ha sido implementado en MATLAB 7.0.

Expert system for automatic analysis of facial expressions realiza reconocimiento y clasificación emocional de expresiones faciales convirtiendo la geometría facial de bajo nivel en “acciones” faciales de alto nivel y luego estas en etiquetas emocionales ponderadas.

En *Analysis of Facial Expression and Recognition Based On Statistical Approach*se usan parámetros estadísticos, que posteriormente son llevados a un vector de características y finalmente traducidos a expresiones faciales mediante redes neuronales artificiales con un 92.2% de efectividad.

A partir del estudio realizado de soluciones FER, se llega a la conclusión de que ninguna de las soluciones estudiadas cumple totalmente con las necesidades de la UCI, ya que no se puede acceder a estas aplicaciones ni al código fuente, sin embargo sirvieron de guía para la obtención de características e idear una posible solución de la cual la universidad pueda hacer uso, para adaptarla a diversos proyectos y soluciones que genera.

Proceso de FER

EL proceso a seguir para la implementación del sistema FER consta de tres pasos básicos: el pre-procesamiento de la imagen, la extracción de características y la clasificación de expresiones faciales mediante redes neuronales, como se detalla en la **figura 1.1.**

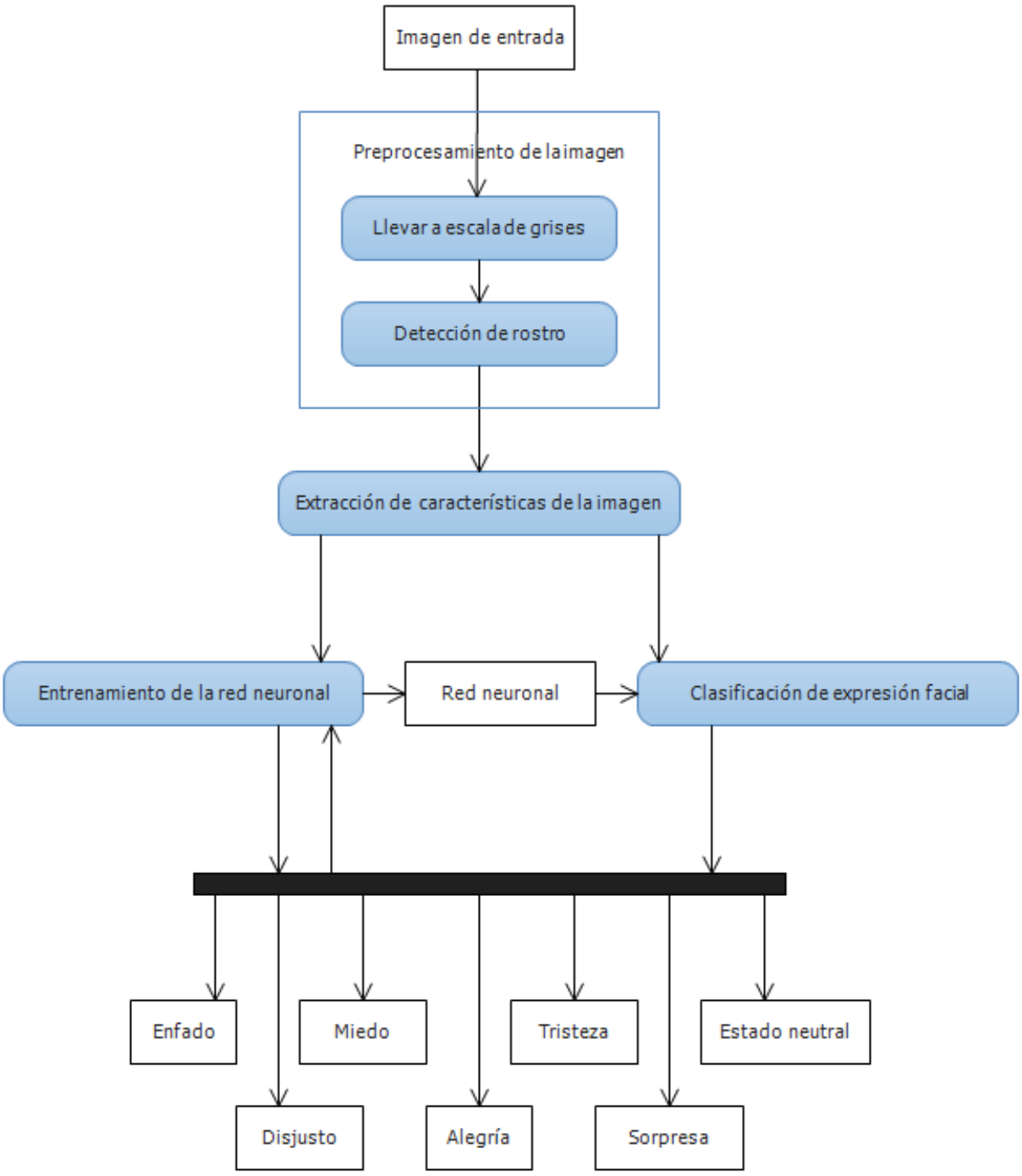


Figura 1.1 Proceso a seguir para la implementación del sistema FER

CAPÍTULO 1. FUNDAMENTACIÓN TEÓRICA

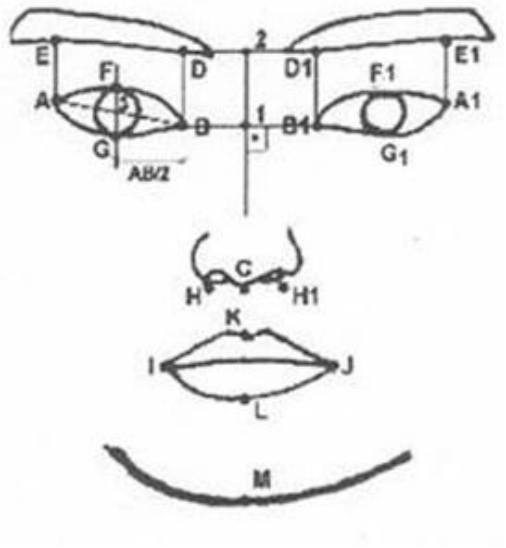
1. Pre-procesamiento de la imagen

En el pre-procesado de la imagen primeramente se lleva la imagen a escala de grises; imágenes de este tipo, son también conocidas como negro y blanco, están compuestas exclusivamente de tonos de gris, que varían de negro la intensidad más débil a la más fuerte en blanco. Luego se ecualiza el histograma de la imagen para subir el contraste de la misma, es decir, la ecualización del histograma hace que los píxeles oscuros aparezcan más oscuros y los píxeles de luz aparezcan más claros. La ecualización del histograma utiliza los resultados del histograma de la imagen original para obtener una nueva imagen con el histograma igualado. Luego de estos procesos se realiza la detección de rostro **(17)**.

2. Extracción de características de la imagen

Una vez detectado el rostro hay que extraer una serie de características del mismo para esto se llevan a cabo una serie de mediciones entre determinados puntos claves de la cara. En la **figura 1.2** podemos observar la descripción y posición de los puntos.

Punto	Descripción
A	Esquina exterior ojo izquierdo (estable)
A1	Esquina exterior ojo derecho (estable)
B	Esquina interior ojo izquierdo (estable)
B1	Esquina interior ojo derecho (estable)
H	Centro agujero izquierdo nariz (no estable)
H1	Centro agujero derecho nariz (no estable)
D	Esquina interior ceja izquierda (no estable)
D1	Esquina interior ceja derecha (no estable)



E Esquina exterior ceja izquierda (no estable)

E1 Esquina exterior ceja derecha (no estable)

F Parte superior ojo izquierdo (no estable)

F1 Parte superior ojo derecho (no estable)

G Parte inferior ojo izquierdo (no estable)

G1 Parte inferior ojo derecho (no estable)

K Parte superior del labio superior (no estable)

L Parte inferior del labio inferior (no estable)

I Esquina izquierda de la boca (no estable)

J Esquina derecha de la boca (no estable)

M Punta de la barbilla (no estable)

Figura 1.2 Puntos característicos presentes en un rostro

Las mediciones que se realizan con los puntos son basadas en el estudio de **(3)** y estas están descritas en las siguiente **tabla 1.1**:

Tabla 1.1 Mediciones

Medición	Descripción
F1	Ángulo BAD
F2	Ángulo B1A1D1
F3	Distancia AE
F4	Distancia A1E1
F5	Distancia 3F (3 es el centro de AB)
F6	Distancia 4F (4 es el centro de A1B1)

F7	Distancia 3G
F8	Distancia 4G1
F9	Distancia FG
F10	Distancia F1G1
F11	Distancia CK (C es el centro de HH1)
F12	Distancia IB
F13	Distancia JB1
F14	Distancia CI
F15	Distancia CJ
F16	Distancia IJ
F17	Distancia KL
F18	Distancia CM
F19	Intensidad de imagen en el círculo de radio0.5*BB1 y centro en 2 sobre la línea DD1
F20	Intensidad de imagen en el círculo de radio0.5*BB1 y centro en 2 bajo la línea DD1
F21	Intensidad de imagen en el círculo de radio 0.5*AB y centro en A, a la izquierda de la línea
F22	Intensidad de imagen en el círculo de radio0.5*A1B1 y centro en A1 a la derecha de la
F23	Intensidad de la imagen en la mitad izquierda del círculo de radio 0.5*BB1 y centro en I
F24	Intensidad de la imagen en la mitad derecha del círculo de radio 0.5*BB1 y centro en J
F25[1]	Distribución del brillo a lo largo de la línea KL

[1] Esta medición fue modificada con el objetivo de obtener mejores resultados. Se calcula la intensidad de la imagen en el círculo de radio KL/2 y centro en 3 (3 es centro entre KL).

Una vez obtenidas estas características (mediciones) hay que determinar que expresión facial se refleja, con este fin se utilizó el método heurístico de las redes neuronales debido a que no existe una solución óptima con un buen tiempo de ejecución para resolver este problema, es decir, después de que la redes neuronales han sido entrenadas, el tiempo que se tarda en responder cuál es la expresión facial reflejada en una imagen de entrada es mucho menor al tiempo que tardaría un algoritmo determinista en dar una respuesta similar. Además, las redes neuronales es el método más utilizado en el reconocimiento de patrones, arrojando siempre un alto porcentaje de aciertos **(23).**

3. Clasificación de expresiones faciales mediante redes neuronales

Las redes de neuronas artificiales (denominadas habitualmente como RNA o en inglés como: "ANN") son un paradigma de aprendizaje y procesamiento automático inspirado en la forma en que funciona el sistema nervioso de los animales. Se trata de un sistema de interconexión de neuronas que colaboran entre sí para producir un estímulo de salida. En inteligencia artificial es frecuente referirse a ellas como redes de neuronas o redes neuronales.

Con un paradigma convencional de programación en ingeniería del software el objetivo es modelar matemáticamente el problema en cuestión y posteriormente formular una solución (programa) mediante un algoritmo codificado que tenga una serie de propiedades que permitan resolver dicho problema. En contraposición, la aproximación basada en las RNA parte de un conjunto de datos de entrada suficientemente significativo y el objetivo es conseguir que la red aprenda automáticamente las propiedades deseadas. En este sentido, el diseño de la red tiene menos que ver con cuestiones como los flujos de datos y la detección de condiciones, y más que ver con cuestiones tales como la selección del modelo de red, las variables a incorporar y el pre-procesamiento de la información que formará el conjunto de entrenamiento. Asimismo, el proceso por el que los parámetros de la red se adecuan a la resolución de cada problema no se denomina genéricamente programación sino que se suele denominar entrenamiento neuronal **(18).**

En este caso dicho entrenamiento neuronal estará enfocado en el reconocimiento de expresiones faciales; durante la fase de entrenamiento el sistema recibe imágenes de rostros de personas, así como las respectivas clasificaciones de dichas imágenes en cuanto a la expresión facial que

muestran. Si el entrenamiento es el adecuado, una vez concluido, el sistema podrá recibir imágenes de rostros no clasificados y obtener su clasificación con un buen grado de exactitud. Las variables de entrada son una serie de características determinadas a partir de la imagen como se explicó en la sección anterior.

Una red neuronal se compone de unidades llamadas neuronas. Cada neurona recibe una serie de entradas a través de interconexiones y emite una salida. Esta salida viene dada por dos funciones:

1. Una función de propagación (también conocida como función de excitación), que por lo general consiste en la sumatoria de cada entrada multiplicada por el peso de su interconexión (valor neto). Si el peso es positivo, la conexión se denomina excitatoria; si es negativo, se denomina inhibitoria.
2. Una función de transferencia, que se aplica al valor devuelto por la función anterior. Se utiliza para acotar la salida de la neurona y generalmente viene dada por la interpretación que queramos darle a dichas salidas. Algunas de las más utilizadas son las funciones tangente hiperbólica (para obtener valores en el intervalo [-1,1]) y sigmoidea (para obtener valores en el intervalo [0,1]), siendo esta última la seleccionada para el diseño de la red neuronal **(19).**

Las capas de una red neuronal pueden clasificarse en tres tipos:

- Capa de entrada: Constituida por aquellas neuronas que introducen los patrones de entrada en la red. En estas neuronas no se produce procesamiento.
- Capas ocultas: Formada por aquellas neuronas cuyas entradas provienen de capas anteriores y cuyas salidas pasan a neuronas de capas posteriores.
- Capa de salida: Neuronas cuyos valores de salida se corresponden con las salidas de toda la red.

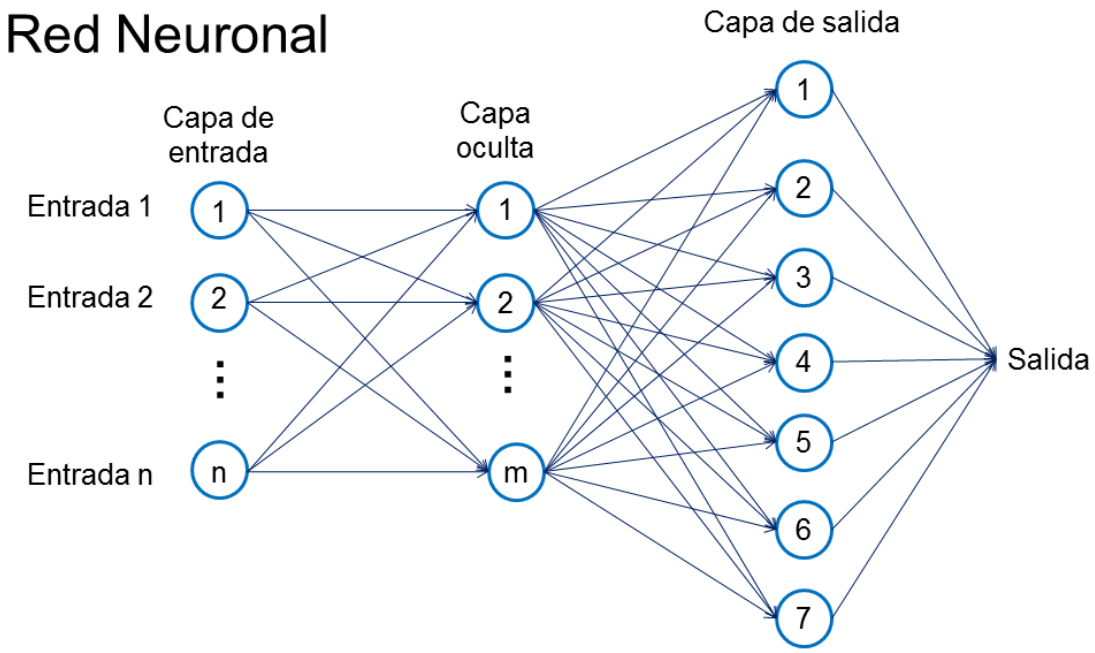


Figura 1.3 Red neuronal multicapa

En esta investigación se usa una red neuronal con una capa de entrada con 25 neuronas (una por cada característica extraída), una capa oculta con 30 neuronas y una capa de salida con siete neuronas (una por cada expresión facial: ira, miedo, felicidad, alegría, tristeza, sorpresa y neutral).

Metodologías de desarrollo de software

Para asegurar el éxito durante el desarrollo de software no es suficiente contar con las herramientas y los conocimientos acerca de estas, hace falta un elemento importante, una guía que paute los pasos a seguir para la correcta aplicación de los elementos y muestre cómo actuar ante las dificultades. Este papel lo juegan las metodologías de desarrollo de software.

Las metodologías definen Quién debe hacer, Qué, Cuándo y Cómo debe hacerlo **(16)**, engloban los procedimientos, técnicas, documentación y herramientas que se utilizan en la creación de un software. Según su filosofía de desarrollo, se clasifican en dos grupos: Metodologías Tradicionales (o Pesadas) y Metodologías Ágiles.

Las metodologías tradicionales van dirigidas a proyectos en los que por lo general ya existe un contrato prefijado, en el que los grupos de trabajo son grandes y posiblemente distribuidos, tienen una rigurosa definición de roles, con mayor énfasis en la planificación y control del proyecto, con especificación precisa de requisitos, actividades y artefactos, incluyendo modelado y documentación detallada y con cierta resistencia a los cambios. Las metodologías ágiles están más orientadas a la generación de código con ciclos cortos de desarrollo, se dirigen a equipos de desarrollo pequeños trabajando en el mismo sitio, no existe contrato tradicional o al menos es bastante flexible y hacen menos énfasis en la arquitectura del software, por lo que es esta última la escogida para el desarrollo del trabajo.

Selección de la metodología de desarrollo de software

Dada la variedad de metodologías que existen y con el objetivo de seleccionar una que se ajuste al ambiente de desarrollo y ayude a obtener los resultados esperados se analizaron las metodologías Programación Extrema, *eXtreme Programming* (XP), y Desarrollo Guiado por la Funcionalidad, *Feature Driven Development* **(5).**

XP: Es una metodología ágil centrada en potenciar las relaciones interpersonales como clave para el éxito en el desarrollo de software, promoviendo el trabajo en equipo, y propiciando un buen clima de trabajo. Se basa en la retroalimentación continua entre el cliente y el equipo de desarrollo, en la simplicidad en las soluciones implementadas y velocidad de reacción ante los cambios que puedan surgir.

Las historias de usuario es la técnica utilizada para especificar los requisitos del software. Son tarjetas de papel en las cuales el cliente describe brevemente las características que el sistema debe poseer, sean requisitos funcionales o no funcionales **(14).**

Se crea un plan de entrega entre clientes y equipo de desarrollo, tomando como base las historias de usuario y la arquitectura. En cada entrega se discuten entre las partes los objetivos y se definen las iteraciones necesarias para cumplir los objetivos. Cada iteración tiene como resultado un programa que se entrega al cliente para que lo analice, en base a su respuesta se planifican las próximas iteraciones, y si no está de acuerdo se ajusta el plan de entrega hasta que se satisfagan sus necesidades.

Una característica relevante de XP, es que el código siempre se produce en parejas, parejas que van cambiando constantemente para lograr así que todo el equipo sepa y pueda modificar según las necesidades el código generado, esto logra en el equipo que los integrantes aprendan entre sí y compartan todo el código **(20).**

FDD (Feature Driven Development): Está pensado para proyectos relativamente cortos, al igual que la metodología analizada anteriormente, está basada en iteraciones que producen un software funcional que puede ser visto, probado y monitorizado por el cliente. Estas iteraciones son decididas en base a las funcionalidades que el software debe tener y es un proceso que está dividido en cinco fases: Desarrollo de un modelo global, Construcción de la lista de funcionalidades, Planificación por funcionalidad, Diseño por funcionalidad e Implementación por funcionalidad (Ver **figura 1.4**).

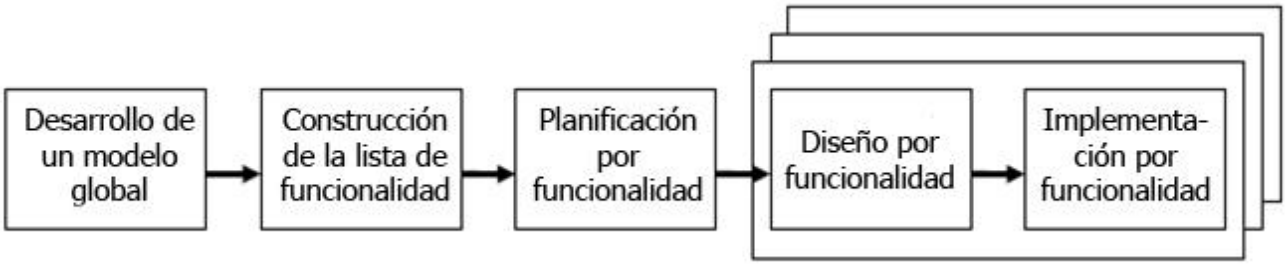


Figura 1.4 Proceso de FDD

Aquí en el equipo de trabajo si existen jerarquías, siempre debe haber un jefe de proyecto, y aunque es un proceso considerado ligero también incluye documentación (la mínima necesaria para que algún nuevo integrante pueda entender el desarrollo de inmediato) **(14).**

Se decide usar FDD debido a que la cantidad de modelos y documentación que genera, son aceptables teniendo en cuenta el tiempo de desarrollo del sistema, proporciona un software que se adapta exactamente a los requisitos esperados sin que implique tener a un cliente totalmente involucrado en el negocio y no define explícitamente la obtención de requisitos, sino el proceder a partir de que se han capturado dichos requisitos de la forma que el equipo de desarrollo haya estimado.

CAPÍTULO 1. FUNDAMENTACIÓN TEÓRICA

Análisis y selección de herramientas a utilizar en el proceso de desarrollo

OpenCV

Para el desarrollo del sistema es necesario realizar varias tareas con imágenes que pueden llegar a ser algo complejas, por lo que se ha propuesto utilizar la biblioteca de clases OpenCV, la cual tiene incorporadas estas funciones y muchas más, lo que permitirá procesar las imágenes de una forma más fácil y eficiente.

OpenCV es una biblioteca libre de visión artificial originalmente desarrollada por Intel. Se ha utilizado en infinidad de aplicaciones. Esto se debe a que permite que sea usada libremente para propósitos comerciales y de investigación bajo una licencia BSD **(2).**

OpenCV es multiplataforma y contiene más de 500 funciones que abarcan una gran gama de áreas en el proceso de visión, como reconocimiento de objetos (reconocimiento facial), calibración de cámaras, visión estéreo y visión robótica. Proporciona un entorno de desarrollo fácil de utilizar y altamente eficiente **(6).**

Lenguaje de programación

La mayoría de los sistemas que utilizan procesamiento de imágenes requieren algún grado de eficiencia, de modo que C++ es el lenguaje ideal en este tipo de aplicaciones, ya que algunas características de este intentan facilitar el afinado del rendimiento cuando el código generado no es lo suficientemente eficiente. No sólo se puede conseguir el mismo bajo nivel de C (y la capacidad de escribir directamente lenguaje ensamblador dentro de un programa C++), además la experiencia práctica sugiere que la velocidad para un programa C++ orientado a objetos tiende a ser ± 10 % de un programa escrito en C, y a menudo mucho menos **(9)**. Sin embargo el uso de C++ compromete la agilidad en el desarrollo del sistema debido a la complejidad de la sintaxis del lenguaje y además los marcos de trabajo con los que cuenta para crear sistemas con interfaz gráfica de usuario (GUI por sus siglas en inglés) dificultan crear aplicaciones de gran envergadura. A veces, es apropiado intercambiar velocidad de ejecución por productividad de programación; por lo que se seleccionó el lenguaje C#.

C# elimina muchos elementos que otros lenguajes incluyen y que son innecesarios en .NET. Por ejemplo **(7)**:

- El código escrito en C# es auto-contenido, lo que significa que no necesita de ficheros adicionales al propio fuente tales como ficheros de cabecera o ficheros IDL.
- El tamaño de los tipos de datos básicos es fijo e independiente del compilador, sistema operativo o máquina para quienes se compile (no como en C++), lo que facilita la portabilidad del código.

Permite la gestión automática de memoria pues tiene a su disposición el recolector de basura de .Net. Esto tiene el efecto en el lenguaje de que no es necesario incluir instrucciones de destrucción de objetos. Sin embargo, dado que la destrucción de los objetos a través del recolector de basura es indeterminista y sólo se realiza cuando este se active, ya sea por falta de memoria, finalización de la aplicación o solicitud explícita en el fuente, C# también proporciona un mecanismo de liberación de recursos determinista a través de la instrucción using.

Además C# incluye mecanismos que permiten asegurar que los accesos a tipos de datos siempre se realicen correctamente, lo que evita que se produzcan errores difíciles de detectar por acceso a memoria no perteneciente a ningún objeto y es especialmente necesario en un entorno gestionado por un recolector de basura. Para ello se toman medidas del tipo:

- Sólo se admiten **conversiones entre tipos compatibles**. Esto es, entre un tipo y antecesores suyos, entre tipos para los que explícitamente se haya definido un operador de conversión, y entre un tipo y un tipo hijo suyo del que un objeto del primero almacenase una referencia del segundo (downcasting) Obviamente, lo último sólo puede comprobarlo en tiempo de ejecución el CLR y no el compilador, por lo que en realidad el CLR y el compilador colaboran para asegurar la corrección de las conversiones.
- No se pueden usar **variables no inicializadas**. El compilador da a los campos un valor por defecto consistente en ponerlos a cero y controla mediante análisis del flujo de control de la fuente que no se lea ninguna variable local sin que se le haya asignado previamente algún valor.

- Se comprueba que todo **acceso a los elementos de una tabla** se realice con índices que se encuentren dentro del rango de la misma.
- Se puede **controlar la producción de desbordamientos en operaciones aritméticas**, informándose de ello con una excepción cuando ocurra. Sin embargo, para conseguirse un mayor rendimiento en la aritmética estas comprobaciones no se hacen por defecto al operar con variables sino sólo con constantes (se pueden detectar en tiempo de compilación).
- Pueden definirse métodos que admitan un número indefinido de parámetros de un cierto tipo, y a diferencia lenguajes como C/C++, en C# siempre se comprueba que los valores que se les pasen en cada llamada sean de los tipos apropiados.

En principio, en C# todo el código incluye numerosas restricciones para garantizar su seguridad y no permite el uso de punteros. Sin embargo, 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 cómo se hace en C++ **(7)**, lo que puede resultar vital en la implementación de algoritmos de procesamiento de imágenes en los cuales se necesita gran eficiencia y velocidad de procesamiento.

EmguCV

Como OpenCV sólo es compatible con C++, C, y Python y el lenguaje a utilizar es C#, se debe utilizar un contenedor que proporcione una interfaz entre OpenCV y C#. Uno fácilmente disponible y fácil de usar se conoce como EmguCV, que tiene además la ventaja de poder ser compilado en Mono y por tanto poder ejecutarse en distribuciones Linux sin mayores complicaciones, puede también ejecutarse en Windows, Mac OS X, iPhone, iPad y dispositivos Android.

Otras ventajas:

- Clase Imagen con el color y la profundidad genérica.
- Documentación XML y soporte de auto completamiento.
- La decisión de utilizar la clase Imagen o invocar de forma directa las funciones de OpenCV.

- Operaciones genéricas sobre los píxeles de la imagen **(8).**

Entorno de desarrollo integrado

Los entornos de desarrollo integrado (IDEs, por sus siglas en inglés) son herramientas pensadas para escribir, compilar, depurar y ejecutar programas. El más popular para escribir programas C# es Microsoft Visual Studio.

Visual Studio permite a los desarrolladores crear aplicaciones, sitios web y servicios web en cualquier entorno que soporte la plataforma .NET (a partir de la versión net 2002).

Microsoft Visual Studio 2010 Ultimate es un IDE para sistemas operativos Windows. Soporta varios lenguajes de programación tales como Visual C++, Visual C#, Visual J#, ASP.NET y Visual Basic .NET, aunque actualmente se han desarrollado las extensiones necesarias para muchos otros.

Aparte de presentar multitud de novedades para el entorno integrado de desarrollo (IDE), trae consigo importantes mejoras para fomentar la colaboración de los equipos multidisciplinares implicados en los proyectos. En Team Foundation Server (TFS) se integra toda la información, convirtiéndose en un repositorio no sólo del código, sino también de requisitos, casos de uso, pruebas, incidencias y planes de proyecto, entre otros documentos.

Incluye soporte, por primera vez en Visual Studio, para diagramas UML 2.0 y diagrama N-Capas, con el que se representan las capas de una arquitectura y se puede validar la misma contra el código fuente real, incluso en procesos centralizados en TFS.

Incorpora la posibilidad de programar desde la misma herramienta aplicaciones para Windows 7, Windows Server, Windows Azure, Sharepoint, SQL Server, Windows Phone 7 y Xbox, gracias a .NET Framework **(4).**

UML como lenguaje de modelado

Para el futuro desarrollo del sistema se pretende utilizar como lenguaje de modelado el Lenguaje Unificado de Modelado UML debido a que está consolidado como el lenguaje estándar en el

análisis y diseño de sistemas de cómputo. Se escoge UML porque permite modelar sistemas utilizando técnicas orientadas a objetos. Permite especificar todas las decisiones de análisis y diseño, construyéndose así modelos, precisos, no ambiguos y completos. Puede conectarse con lenguajes de programación (ingeniería directa e inversa). Permite documentar todos los artefactos de un proceso de desarrollo (requisitos, arquitectura, pruebas, versiones). Existe un equilibrio entre expresividad y simplicidad, pues no es difícil de aprender ni de utilizar.

Herramienta CASE

Una herramienta CASE (Ingeniería de Software Asistida por Ordenador, de la traducción del inglés *Computer Aided Software Engineering*) es una aplicación informática que permite la realización de un buen diseño para el proyecto, implementando a partir de él, parte del código automáticamente.

Como herramienta CASE se utiliza Visual Studio 2010 Ultimate pues además de ser el IDE propuesto para el desarrollo de la aplicación es ideal para desarrollar en equipo y gestionar todo el ciclo de vida del software. Proporciona plantillas para cinco de los diagramas UML que se usan con más frecuencia: diagramas de actividades, diagramas de clases, diagramas de componentes, diagramas de secuencia y diagramas de casos de uso. Puede crear también diagramas de capas, que ayudan a definir la estructura del sistema. Presentando así las cualidades necesarias para llevar a cabo el proceso de desarrollo de software según la metodología FDD **(21)**.

Conclusiones

A partir del estudio realizado de los FER se llega a la conclusión de que ninguno cumple con las necesidades de la UCI o del país debido a que se necesita un software que pueda adaptarse a diversos ambientes de proyectos según se requiera y que pueda ser modificado fácilmente sin tener que incurrir en gastos monetarios, sin embargo dicho estudio sirvió de guía para la obtención de características e idear nuestra solución, para la cual se decidió utilizar el método de las redes neuronales ya que es el más utilizado en el reconocimiento de patrones, arrojando siempre un alto porcentaje de aciertos en un tiempo relativamente pequeño y tiene la capacidad de aprender tareas basadas en un entrenamiento o una experiencia inicial.

CAPÍTULO 1. FUNDAMENTACIÓN TEÓRICA

Para el desarrollo de la propuesta que dará solución al problema científico se utilizará C# como lenguaje de programación, Microsoft Visual Studio 2010 como IDE y la biblioteca de clase ASM para la detección de los puntos característicos y EmguCV para facilitar el trabajo con imágenes. Para crear los diagramas que documentan el software se utilizará Visual Studio 2010 Ultimate. Además el desarrollo estará guiado por la metodología FDD.

CAPÍTULO 2: Propuesta de solución

Introducción

A partir de este capítulo se comenzará a desarrollar la propuesta que dará solución al problema científico planteado utilizando la metodología de desarrollo FDD. Esta metodología se divide en cinco fases y en este capítulo se abordarán las cuatro primeras fases: desarrollo de un modelo general, construcción de la lista de funcionalidades, planificación por funcionalidad a implementar y diseño por funcionalidad, las cuales ocupan gran parte del tiempo en las primeras iteraciones.

Modelo General

El modelo general es un diagrama de clases del sistema que representa los conceptos más importantes dentro del dominio del problema y las relaciones entre ellos. Este modelo se realiza con el fin de proveer un marco general dentro del cual evoluciona el proyecto. (Ver **figura 2.1**)

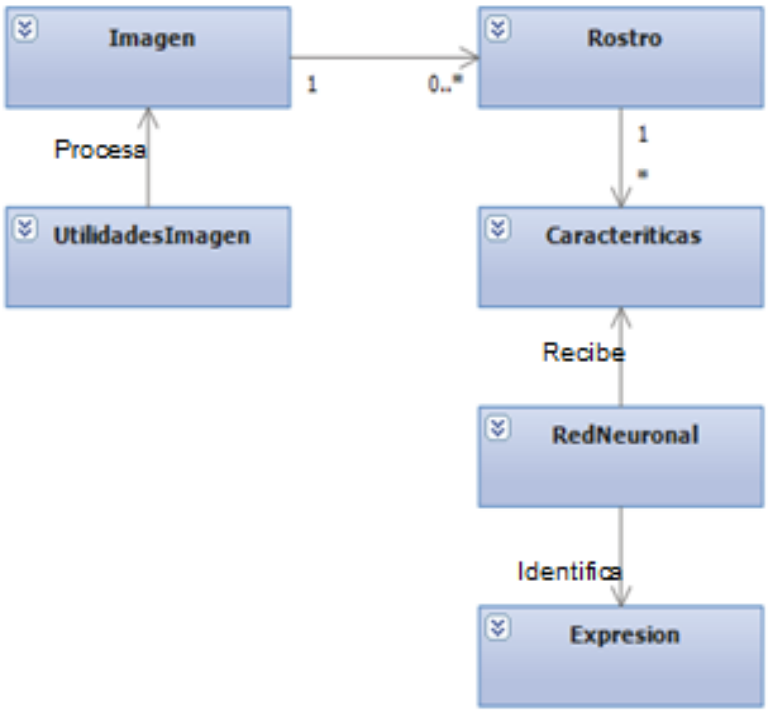


Figura 2.1 Modelo general

Conceptos tratados en el modelo general

- Imagen: Plantilla que define las características de una imagen, como: tipo de color, tamaño, altura, ancho, etc.
- Rostro: Plantilla que define las características de un rostro, como: ojos, boca, nariz, cejas.
- Características: Plantilla que define los atributos de una imagen de un rostro como mediciones entre puntos característicos.
- Red Neuronal: Se encarga de analizar las características del rostro e identificar la expresión facial.
- Expresión: Plantilla que define los tipos de expresiones faciales: tristeza, ira, disgusto, felicidad, sorpresa y miedo además del estado neutral.
- UtilidadesImagen: Se encarga de procesar la imagen, llevar a escala de grises y detectar rostro.

Lista de características

A continuación en la **tabla 2.1** se listan las características del software que se propone realizar.

Tabla 2.2 Listado de características

Conjunto de características	**Características**
Gestionar conjunto de entrenamientos	C1. Entrenar la red neuronal con un conjunto de entrenamientos dado C2. Salvar una red neuronal entrenada C3. Cargar una red neuronal entrenada

Reconocer expresiones faciales	C4. Cargar Imagen C5. Detectar rostros C6. Extraer rostro C7. Extraer características C8. Identificar expresión facial

Una vez definido el listado de características, se procede a describir las restricciones del sistema.

Restricciones del sistema

- **Restricciones de Software**

  El sistema se podrá ejecutar sobre los sistemas operativos: Windows XP/Vista/7.

  Para que el sistema se ejecute correctamente debe estar instalado .NET framework 4 ó superior.

- **Restricciones de Hardware**

  El sistema podrá ser ejecutado sobre ordenadores con las siguientes propiedades: 512 MB de RAM o más.

- **Restricciones en el diseño y la implementación**

  El sistema será implementado usando el lenguaje de programación C# sobre Microsoft .NET framework 4. Se utilizará la biblioteca de clases ASM que permite obtener los puntos ASM en el rostro y EmguCV la cual sirve de intermediaria entre la biblioteca de clases OpenCV y el lenguaje de programación C#.

Clasificación de las características

Con el fin de realizar una planificación de la construcción de la solución propuesta, en la **tabla 2.3** se agrupan y priorizan las características identificadas anteriormente, en críticas y secundarias.

Las características críticas son las que tienen mayor importancia para los clientes porque son las

que deben estar para que el sistema satisfaga las funciones principales, por lo que deben implementarse en las primeras iteraciones. Las características secundarias aunque tienen un impacto más modesto sobre la aplicación, no deben dejar de implementarse.

Tabla 2.3 Clasificación de las características según su impacto

Módulo	Características	Clasificación
Entrenamiento	C1. Entrenar la red neuronal con un conjunto de entrenamientos dado	Crítica
	C2. Salvar una red neuronal entrenada	Secundaria
	C3. Cargar una red neuronal entrenada	Secundaria
Identificación	C4. Cargar Imagen	Crítica
	C5. Detectar rostro	Crítica
	C6. Extraer rostro	Secundaria
	C7. Extraer características	Crítica
	C8. Identificar expresión facial	Crítica

Cronograma de diseño y construcción

En la **tabla 2.4** se determina en qué orden serán desarrolladas las características teniendo en cuenta su clasificación y se establecen los plazos de inicio y fin de ejecución en las fases de diseño y construcción.

Tabla 2.4 Cronograma de diseño y construcción

Iteración	Módulo	Característica	Fecha de ejecución	
			Diseño	Construcción
1	Identificación	Cargar imagen	10/12/12→11/12/12	11/12/12→12/12/12

2	Identificación	Detectar rostro	12/12/12→14/12/12	14/12/12→21/12/12
3	Identificación	Extraer rostro	10/1/13→15/1/13	15/1/13→22/1/13
4	Identificación	Extraer características	23/1/13→1/2/13	1/2/13→14/2/13
5	Entrenamiento	Entrenar la red neuronal con un conjunto de entrenamientos dado	14/2/13→28/2/13	28/2/13→14/3/13
6	Identificación	Identificar expresión facial (ira, disgusto, miedo, alegría, tristeza, sorpresa o estado neutral)	15/3/13→20/3/13	20/3/13→3/4/13
7	Entrenamiento	Salvar una red neuronal entrenada	3/4/13→8/4/13	8/4/13→12/4/13
8	Entrenamiento	Cargar una red neuronal entrenada	12/4/13→15/4/13	15/4/13→19/4/13

Descripción de características del sistema

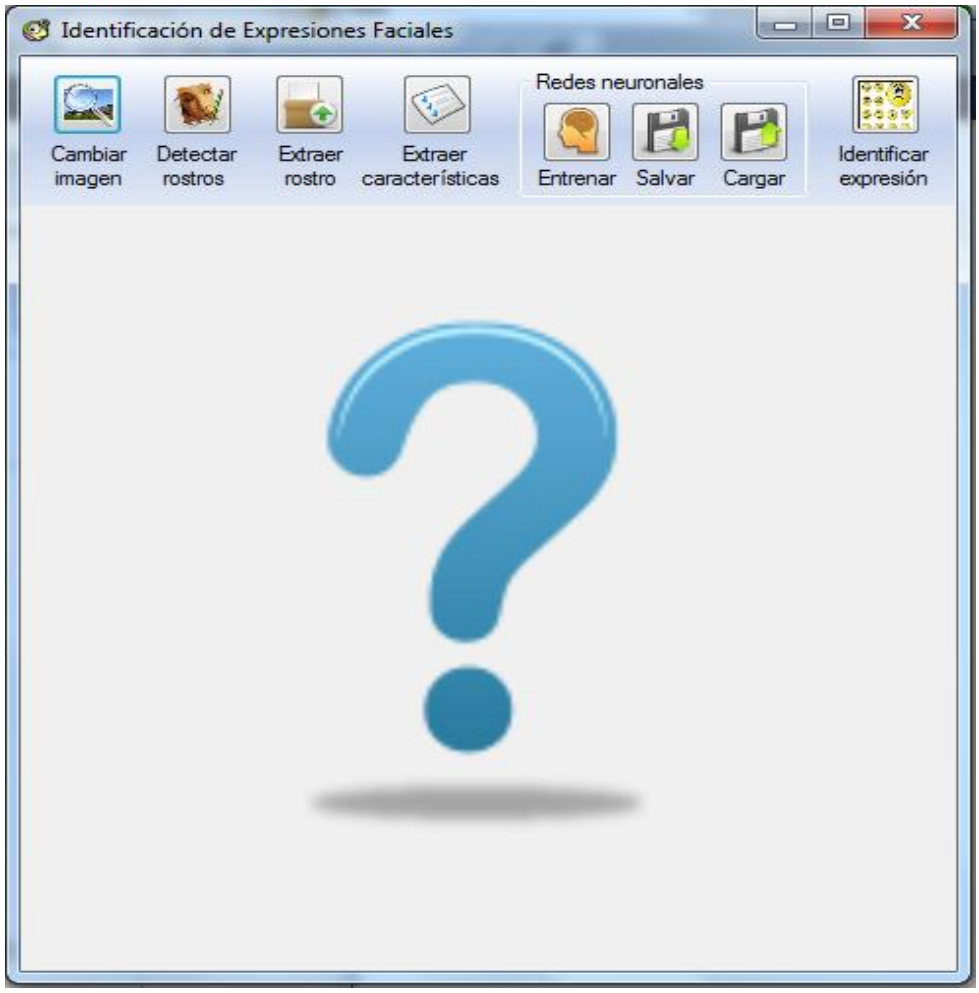
A continuación se describen las características identificadas como críticas para una mejor comprensión de las mismas. La metodología FDD no contempla en sus fases la descripción del listado de características, por lo que se creó una plantilla para la descripción de estas.

Tabla 2.5 Descripción de característica: Cargar Imagen

Nombre de la característica	Cargar Imagen
Precondiciones	No tiene
Características asociadas	C5, C6, C7,C8
Conceptos tratados	Imagen

Descripción básica	1. El sistema muestra la interfaz principal que contiene: • Menú horizontal en el borde superior de la ventana con 8 botones: 1. Cargar imagen 2. Detectar rostro (Deshabilitado) 3. Extraer rostro(Deshabilitado) 4. Extraer características(Deshabilitado) 5. Entrenar red neuronal 6. Salvar red neuronal(Deshabilitado) 7. Cargar red neuronal 8. Identificar expresión(Deshabilitado) 2. Si el usuario selecciona la opción del menú **Cargar imagen**, el sistema muestra una ventana para cargar la imagen deseada. 2.1. Si el usuario selecciona una imagen, el sistema carga esta correctamente. 2.2. El sistema habilita la opción del menú “Detectar rostro”. 3. Si el usuario selecciona la opción del menú **Detectar rostros**, (ver tabla 2.4 Descripción de característica. Detectar Rostros). 4. Si el usuario selecciona la opción del menú **Entrenar red neuronal**, (ver tabla 2.6 Descripción de característica. Entrenar red neuronal). 5. Si el usuario selecciona la opción del menú **Cargar red neuronal**, (ver Anexo 3 Descripción de característica. Cargar red neuronal).

CAPÍTULO 2. PROPUESTA DE SOLUCIÓN



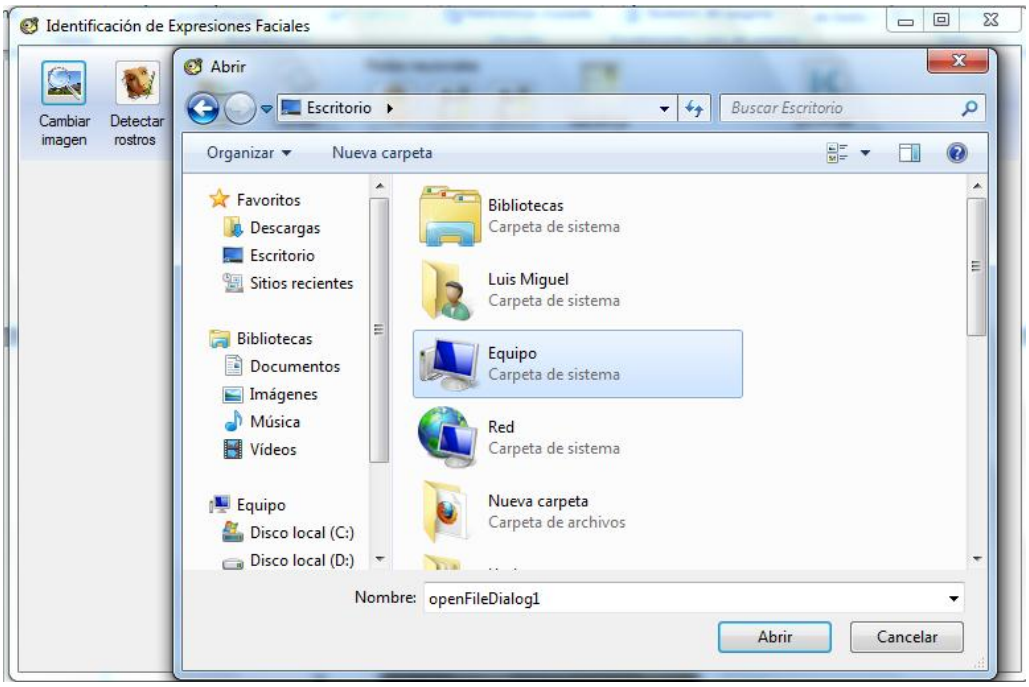
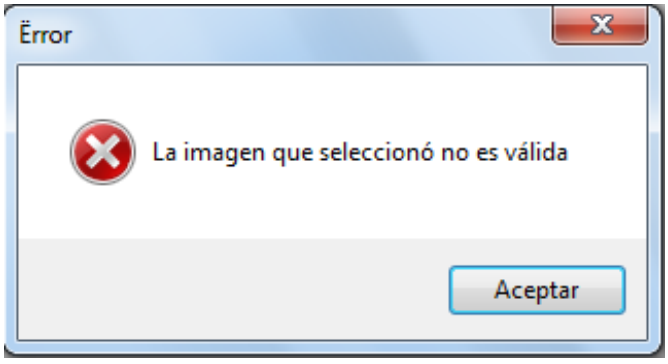
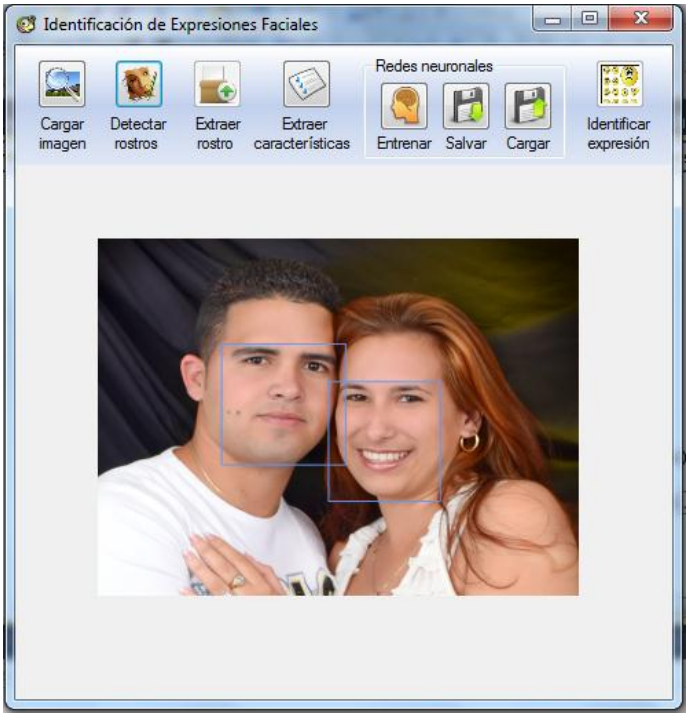
<table>
<tr><td colspan="2"> 
</td></tr>
<tr><td>Descripción alterna</td><td>Si la imagen que el usuario seleccionó no es válida, el sistema muestra un mensaje “La imagen que seleccionó no es válida”.</td></tr>
<tr><td colspan="2"> 
</td></tr>
<tr><td>Poscondiciones</td><td>Queda cargada la imagen en la ventana principal del sistema.</td></tr>
</table>

Tabla 2.6 Descripción de característica: Detectar Rostros

Nombre de la característica	Detectar rostro.
Precondiciones	El usuario debe haber cargado una imagen
Características asociadas	C4, C6, C7,C8
Conceptos tratados	UtilidadesImagen, Imagen, Rostro.
Descripción básica	1. El usuario selecciona la opción del menú **Detectar rostro**. 2. El sistema muestra en la imagen los rostros detectados y habilita la opción del menú “Extraer rostro”. 3. Si el usuario selecciona la opción del menú **Extraer rostro**, (ver Anexo 1 Descripción de característica. Extraer Rostro).
	
Descripción alterna	Si en la imagen no se detectan rostros, se muestra

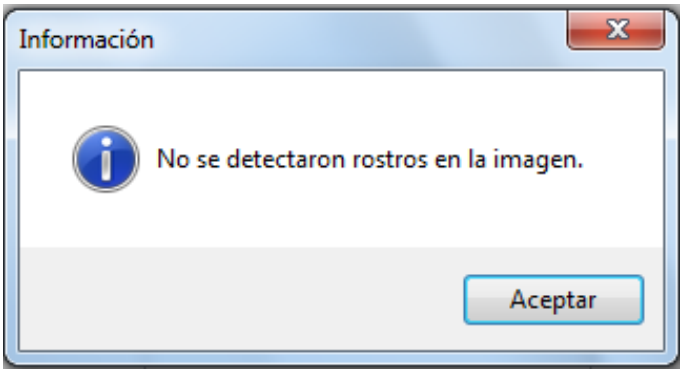
	un mensaje informándolo.
	
Poscondiciones	Se enmarcan en rectángulos los rostros detectados en la imagen.

Tabla 2.7 Descripción de característica: Extraer Características

Nombre de la característica	Extraer características
Precondiciones	El usuario debe haber extraído el rostro de la imagen anteriormente cargada.
Características asociadas	C4, C5, C7,C8
Conceptos tratados	UtilidadesImagen, Rostro, Características
Descripción básica	1. El usuario selecciona la opción del menú **Extraer características**. 2. El sistema muestra la interfaz principal que contiene además: • En la imagen, los puntos característicos del rostro. • Una tabla con la descripción de las características y sus respectivos valores. • Un botón en el menú horizontal “Esconder características”.

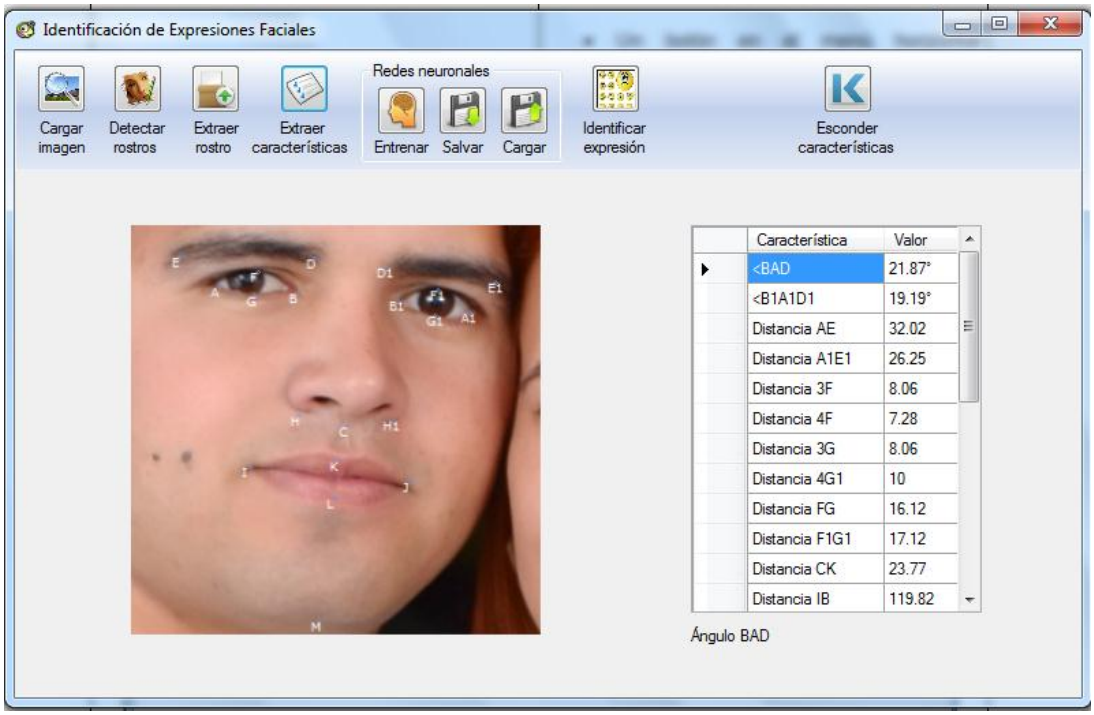
	2.1. Si el usuario selecciona la opción “Esconder características”, el sistema oculta la tabla con la descripción de las características y sus valores. 3. El sistema habilita la opción del menú “Identificar expresión facial”. 4. Si el usuario selecciona la opción del menú **Identificar expresión facial**, (ver tabla 2.7 Descripción de característica. Identificar expresión facial)
	
Descripción alterna	No tiene
Poscondiciones	Quedan mostradas las características extraídas del rostro.

Tabla 2.8 Descripción de característica: Entrenar Red Neuronal

Nombre de la característica	Entrenar red neuronal
Precondiciones	Las imágenes con el resultado que debería dar para cada imagen deben estar copiadas en la carpeta SampleFaces del software.
Características asociadas	C2, C3
Conceptos tratados	RedNeuronal, UtilidadesImagen, Rostro, Características, Expresión.
Descripción básica	1. El usuario selecciona la opción del menú Entrenar Red Neuronal. 2. El sistema realiza el entrenamiento de la red neuronal. 3. El sistema habilita la opción del menú “Salvar red neuronal”. 4. Si el usuario selecciona la opción del menú **Salvar red neuronal**, (ver Anexo 2 Descripción de característica. Salvar red neuronal).
Descripción alterna	No tiene
Poscondiciones	Se obtiene la red neuronal entrenada.

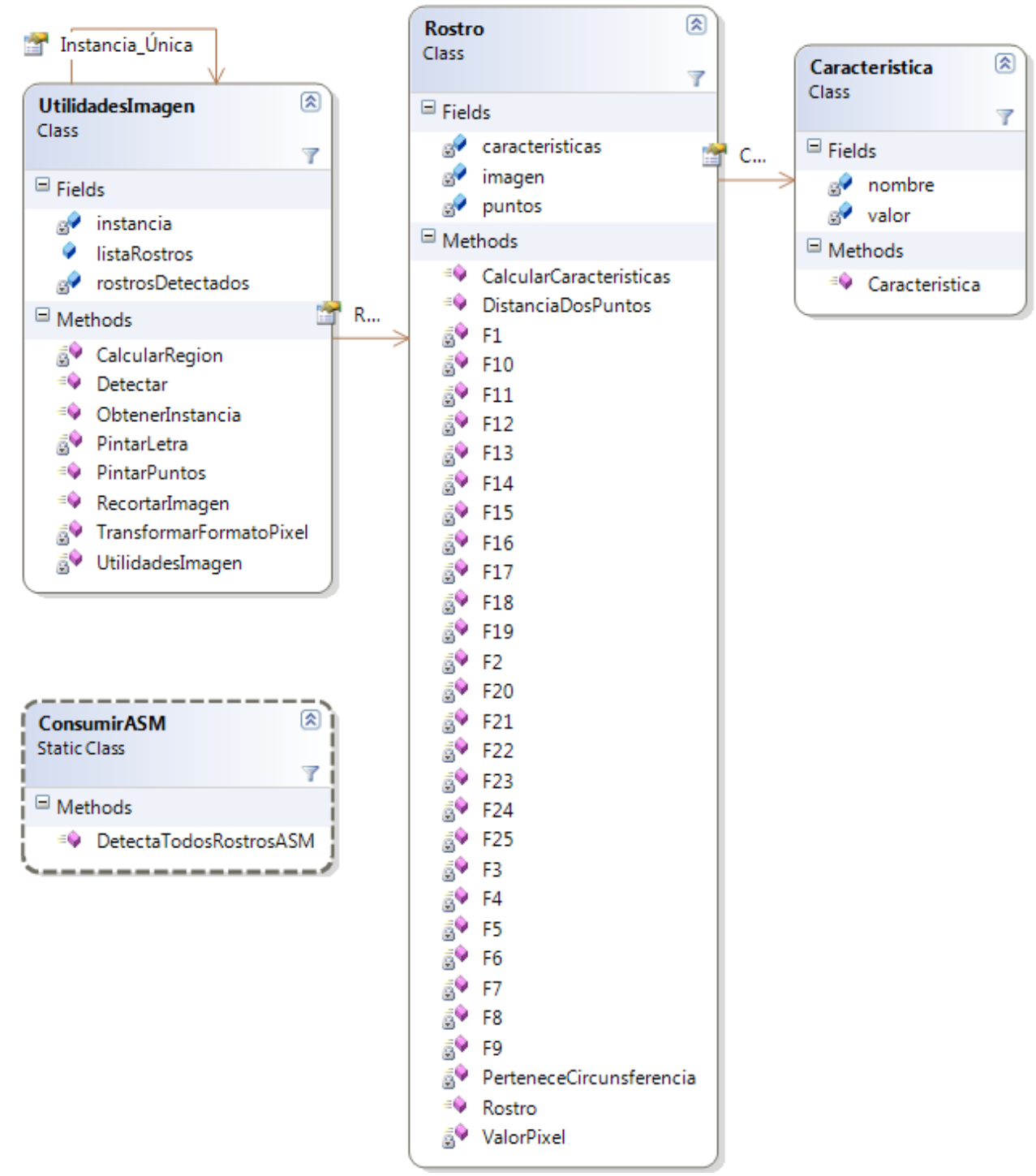
Tabla 2.9 Descripción de característica: Identificar expresión facial

Nombre de la característica	Identificar expresión facial.
Precondiciones	Deben de estar extraídas las características y la red neuronal entrenada.
Características asociadas	C4, C5, C6, C7.
Conceptos tratados	RedNeuronal, UtilidadesImagen, Rostro,

	Características, Expresión.
Descripción básica	1. El usuario selecciona la opción del menú **Identificar expresión facial**. 2. El sistema muestra un mensaje con la expresión facial correspondiente.
Descripción alterna	No tiene
Poscondiciones	Se identificó la expresión facial resultante de la imagen.

Diagrama de clases

A continuación se muestra el diagrama de clases del sistema el cual describe la estructura del sistema mostrando sus clases, atributos y relaciones entre ellos.


Instancia_Única
UtilidadesImagen
Class
Fields
instancia
listaRostros
rostrosDetectados
Methods
CalcularRegion
Detectar
ObtenerInstancia
PintarLetra
PintarPuntos
RecortarImagen
TransformarFormatoPixel
UtilidadesImagen
R...
ConsumirASM
Static Class
Methods
DetectaTodosRostrosASM
Rostro
Class
Fields
caracteristicas
imagen
puntos
Methods
CalcularCaracteristicas
DistanciaDosPuntos
F1
F10
F11
F12
F13
F14
F15
F16
F17
F18
F19
F2
F20
F21
F22
F23
F24
F25
F3
F4
F5
F6
F7
F8
F9
PerteneceCircunsferencia
Rostro
ValorPixel
C...
Caracteristica
Class
Fields
nombre
valor
Methods
Caracteristica

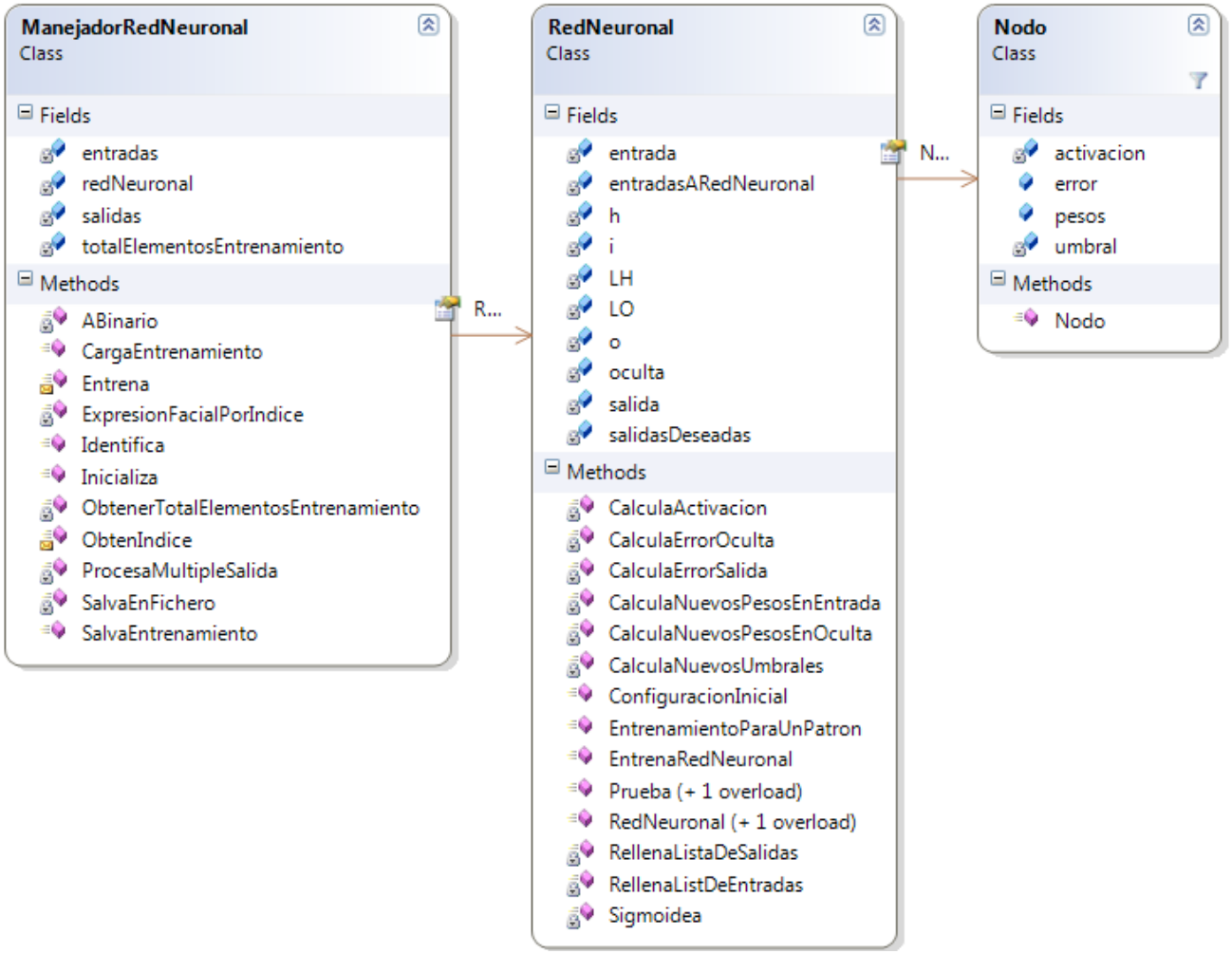


Figura 2.2 Diagrama de clases

Patrones de diseños

Para el diseño del diagrama de clases (ver **figura 2.2**) se aplicaron diferentes patrones de diseño pues estos ayudan a conseguir un diseño correcto rápidamente, ayudan a elegir diseños alternativos que hacen a un sistema reutilizable y pueden incluso mejorar la documentación y mantenimiento de sistemas existentes **(15).**

Para la aplicación de estos se tuvieron en cuenta los patrones del grupo de los cuatro(GOF) utilizando en la clase UtilidadesImagen el Instancia Única (Singleton) para garantizar que solamente se cree una instancia de la clase y proveer un punto de acceso global a esta.

Se aplicaron además patrones generales de software para asignar responsabilidades (GRASP), por ejemplo el patrón Experto el cual consiste en asignar responsabilidad a individuos que disponen de la información necesaria para llevar a cabo una tarea, se evidencia en las clases Rostro, Características, RedNeuronal y Nodo. El patrón Creador se utilizó en UtilidadesImagen, Rostro, ManejadorRedNeuronal y RedNeuronal pues estas clases tienen la responsabilidad de crear objetos de otras clases. El patrón Controlador se aplica en ManejadorRedNeuronal y UtilidadesImagen ya que son responsables de controlar el flujo de eventos del sistema a clases específicas. Además se observa Bajo Acoplamiento entre las clases Rostro-Características y RedNeuronal-Nodo ya que hay pocas dependencias entre estas, existiendo a su vez Alta Cohesión pues tienen responsabilidades moderadas en su área y colaboran con las otras para llevar a cabo las tareas.

Descripción de las clases y métodos

Tabla 2.10 Descripción de la clase UtilidadesImagen

Nombre: UtilidadesImagen	
Tipo de clase: controladora. Está diseñada para manejar las funcionalidades de mayor envergadura de la aplicación.	
Atributo	**Tipo**
rostroDetectados	Rectangle
instancia	UtilidadesImagen
listaRostros.	Rostro
Para cada responsabilidad:	

Nombre:	Descripción:
ObtenerInstancia()	Obtiene la instancia de la clase UtilidadesImagen, de no existir la crea.
DetectarRostro(Bitmapimg, stringdireccion)	Detecta todos los rostros en una imagen.
CalcularRegion(List<Point> puntos)	Es utilizado por DetectarRostro para marcar la región del los rostros detectados.
RecortarImagen(Bitmap nueva, List<Point>listaPtos)	Recorta la imagen pasada por parámetro a partir de la lista de puntos que también le pasan por parámetro.
TransformarFormatoPixel(Bitmapbmp)	Cambia los pixeles de una imagen pasada por parámetro, si son de tipo Format8bppIndexed.
PintarPuntos(Bitmap imagen)	Pinta los puntos del rostro anteriormente detectado en la imagen pasada por parámetro.
PintarLetra(Graphics g, string letra, int x, int y, List<Point> l, int i)	Pinta una letra pasada por parámetro en la posición especificada de la lista de puntos.

Tabla 2.11 Descripción de la clase Rostro

Nombre: Rostro	
Tipo de clase: entidad	
Atributo	**Tipo**
caracteristicas	List<Caracteristica>
puntos	List<Point>
imagen	Bitmap

Para cada responsabilidad:	
Nombre:	**Descripción:**
CalcularCaracteristicas()	Calcula los valores de las características *F1* hasta *F25*.
DistanciaDosPuntos(Point p1, Point p2)	Calcula la distancia entre los puntos pasados por parámetros.
PerteneceCircunsferencia(Point p, Point c, double r)	Devuelve verdadero si el punto p pertenece a la circunferencia de centro *c* y radio *r* y falso si no pertenece.
ValorPixel(Point p)	Devuelve el valor del pixel del punto *p*.
F1()[2]	Ver en la **Tabla 1.2**

Tabla 2.12 Descripción de la clase Caracteristica

Nombre: Caracteristica	
Tipo de clase: entidad	
Atributo	**Tipo**
nombre	string
valor	double

Tabla 2.13 Descripción de la clase ConsumirASM

[2] La descripción de los otros métodos se muestran en la **tabla 1.2.**

Nombre: ConsumirASM	
Tipo de clase: interfaz Diseñada para la detección de los rostros de la imagen, utiliza la dll ASM para la obtención de la lista de características encontradas en un rostro.	
Para cada responsabilidad:	
Nombre:	**Descripción:**
DetectaTodosRostrosASM	Detecta los rostros presentes en una imagen.

Tabla 2.14 Descripción de la clase ManejadorRedNeuronal

Nombre: ManejadorRedNeuronal	
Tipo de clase: controladora	
Atributo	**Tipo**
entradas	double[,]
salidas	double[,]
redNeuronal	RedNeuronal
totalElementosEntrenamiento	int
Para cada responsabilidad:	
Nombre:	**Descripción:**
ABinario	Interpreta el nombre de la imagen a entrenar, informando la expresión facial reflejada.
CargaEntrenamiento	Carga entrenamiento salvado previamente.

Entrena	Entrena la Red Neuronal con las imágenes del conjunto de entrenamiento.
ExpresionFacialPorIndice	Devuelve la expresión facial correspondiente a un índice determinado.
Identifica	Identifica la expresión facial dada las características de la imagen a identificar.
Inicializa	Inicializa las variables de instancia de la Red Neuronal según la cantidad de características a analizar.
ObtenerTotalElementosEntrenamiento	Obtiene el total de elementos del conjunto de entrenamiento.
ObtenIndice	Obtiene índice general de la imagen en el conjunto de entrenamiento.
ProcesaMultipleSalida	Se usa para obtener una lista de las expresiones faciales ordenada por la probabilidad de correctitud.
SalvaEnFichero	Salva en fichero las características de una imagen determinada.
SalvaEntrenamiento	Salva entrenamiento en fichero.

Tabla 2.15 Descripción de la clase RedNeuronal

Nombre: RedNeuronal	
Tipo de clase: entidad	
Atributo	**Tipo**
i	int
o	int
h	int

LH	double
LO	double
entradasARedNeuronal	double[]
salidasDeseadas	double[]
entrada	Nodo[]
salida	Nodo[]
oculta	Nodo[]
Para cada responsabilidad:	
Nombre:	**Descripción:**
CalculaActivacion	Calcula las activaciones de la capa oculta y la capa de salida.
CalculaErrorOculta	Calcula error de cada neurona de la capa oculta.
CalculaErrorSalida	Calcula error de cada neurona de la capa de salida.
CalculaNuevosPesosEnEntrada	Calcula los nuevos pesos entre la capa de entrada y la oculta.
CalculaNuevosPesosEnOculta	Calcula los nuevos pesos entre la capa de oculta y la de salida.
CalculaNuevosUmbrales	Calcula los nuevos umbrales para cada neurona.
ConfiguracionInicial	Proporciona los valores iniciales de los umbrales de las neuronas de la capa oculta y la de salida.
EntrenamientoParaUnPatron	Entrena la Red Neuronal.
EntrenaRedNeuronal	Pasa los datos de entrenamiento a la Red Neuronal, entradas y salidas deseadas.

Prueba	Realiza la identificación después de entrenada la Red Neuronal.
RedNeuronal	Constructor de la clase.
RellenaListDeEntradas	Se usa para rellenar las entradas en el entrenamiento.
RellenaListDeSalidas	Se usa para rellenar las salidas deseadas en el entrenamiento.
Sigmoidea	Acota el valor de entrada en el intervalo (0,1).

Tabla 2.16 Descripción de la clase Nodo

Nombre: Nodo	
Tipo de clase: entidad	
Atributo	**Tipo**
activacion	double
umbral	double
pesos	double []
error	double
Para cada responsabilidad:	
Nombre:	**Descripción:**
Nodo	Constructor de la clase.

Diagramas de secuencias

A continuación se muestran los diagramas de secuencia de algunas de las funcionalidades críticas, los cuales permiten comprender la interacción entre los objetos que integran la solución.

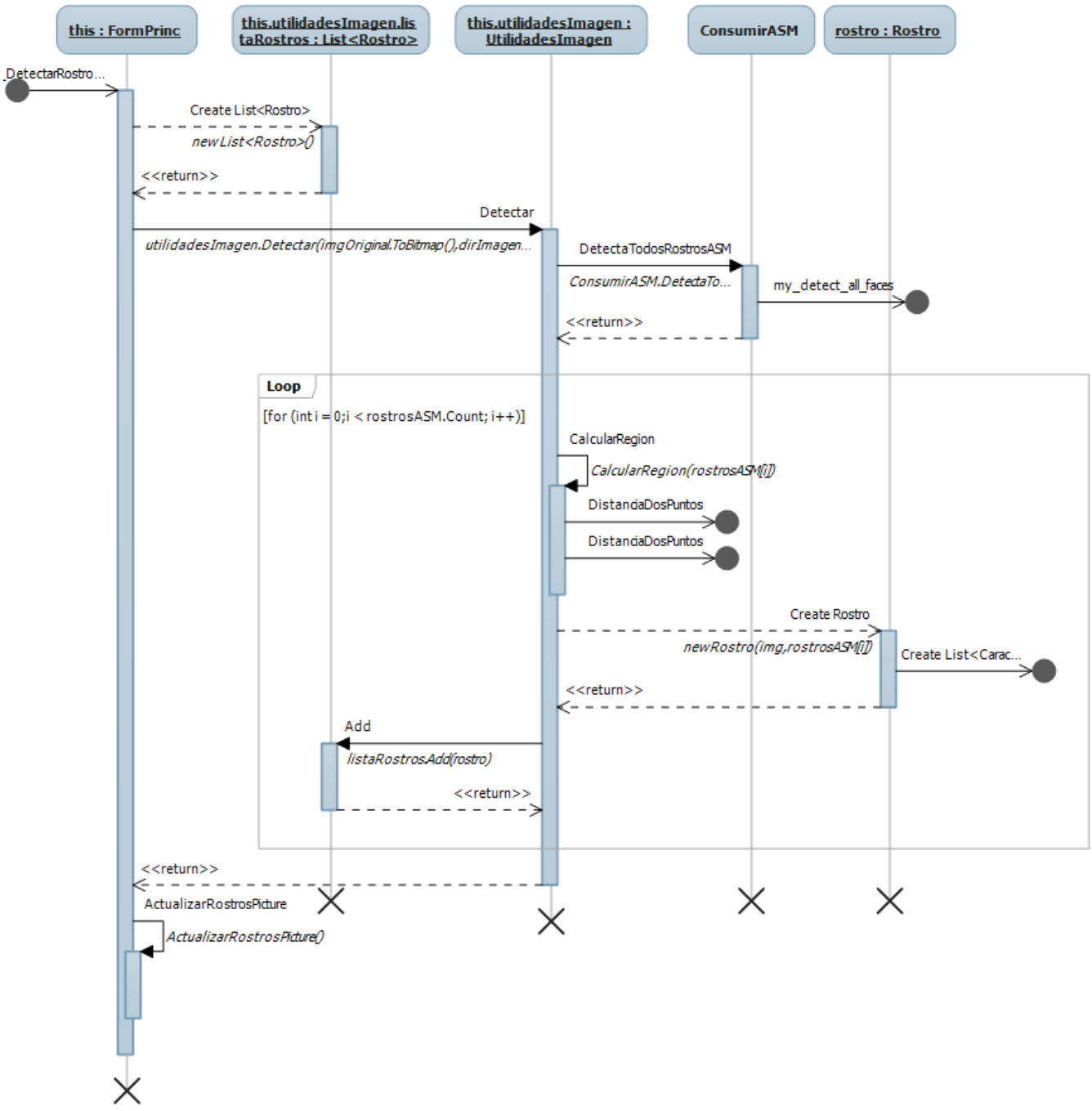


Figura 2.3 Diagrama de secuencia “Detectar rostro”

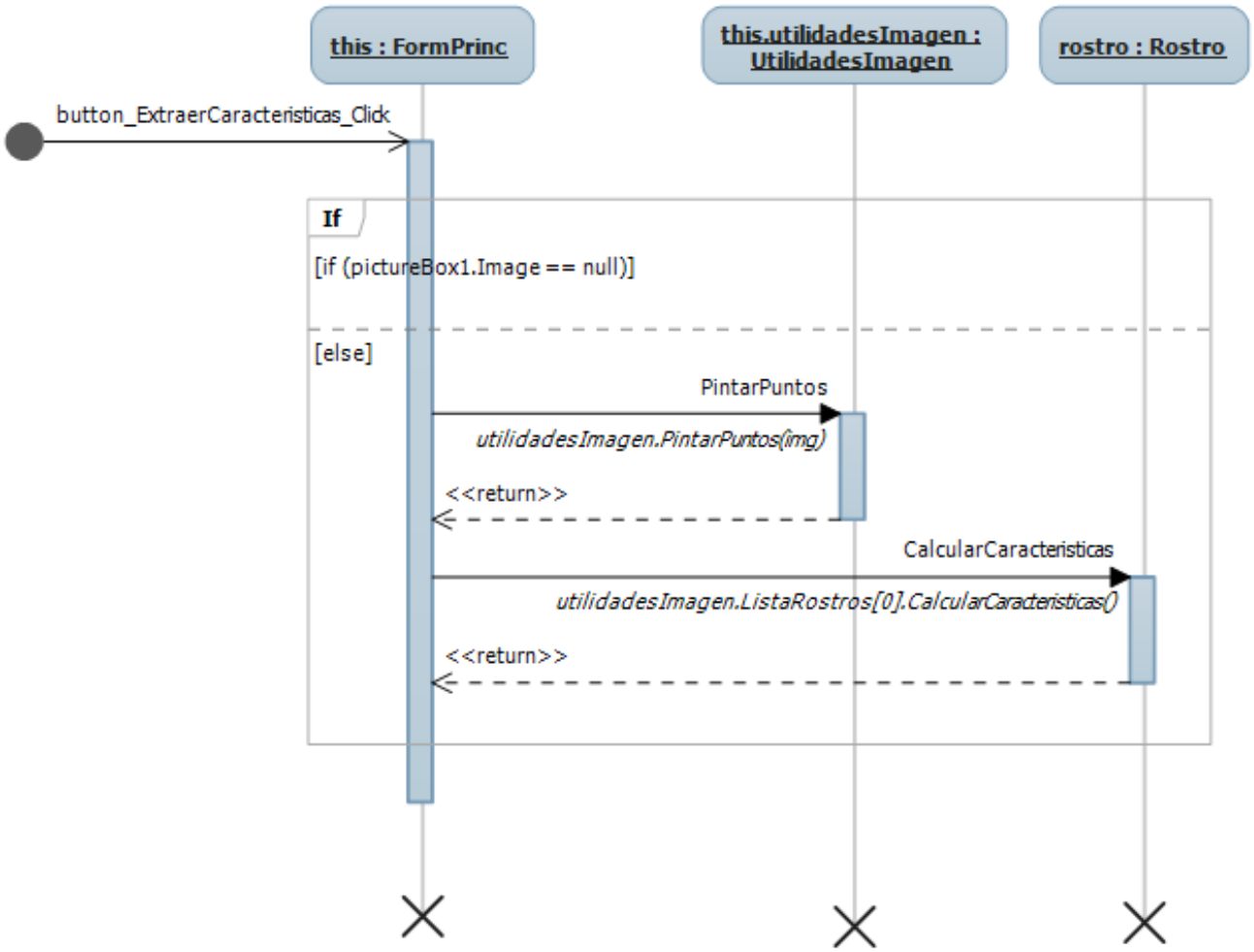


Figura 2.4 Diagrama de secuencia “Extraer características”

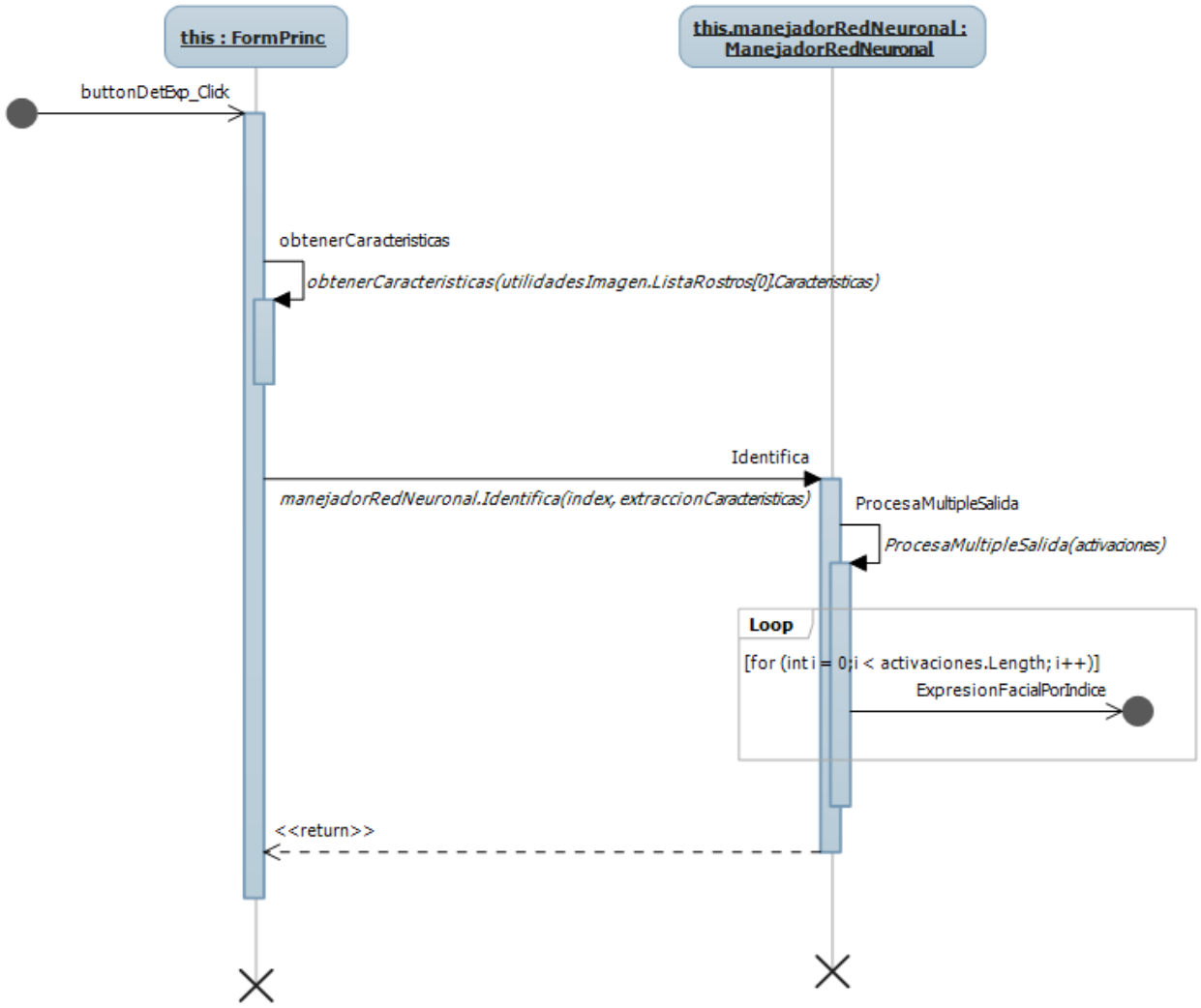


Figura 2.5 Diagrama de secuencia “Identificar expresión facial”

Conclusiones

Con la culminación de este capítulo, quedó definido el modelo general del sistema y una lista de características representando así los conceptos más importantes dentro del dominio del problema y las funcionalidades a desarrollar. Quedó establecido un plan que garantiza la organización del trabajo y permitirá definir el estado en que se encuentra el desarrollo de la solución. Se presentó la propuesta de solución a partir de la realización de forma detallada de los diagramas de clases con

sus descripciones y los diagramas de secuencia. Además fueron descritos los principios de diseño en los que se basa la solución propuesta.

CAPÍTULO 3: Implementación y pruebas

Introducción

En este capítulo se detallan las fases de implementación y prueba según lo define la metodología FDD. Se describe el estilo de codificación utilizado y se muestran los resultados de los casos de prueba realizados.

Estándar de codificación

Es aconsejable seguir un estándar de codificación con el objetivo de mejorar la calidad de nuestro código aumentando considerablemente su legibilidad. A continuación se describe el estándar utilizado en la implementación de la solución.

Reglas de codificación

- Se debe evitar las líneas de más de 80 caracteres, ya que no son bien manejadas por muchas herramientas.
- Los estilos de capitalización para identificadores usados serán:
    - PascalCase: se pone en mayúscula el primer carácter de cada palabra.
    - camelCase: se pone en mayúscula el primer carácter de cada palabra, exceptuando la primera.
- Los miembros públicos usarán el estilo de capitalización PascalCase.
- Los miembros no públicos usarán el estilo de capitalización camelCase.
- Los nombres de los métodos deben reflejar la acción a realizar.

Diagrama de componentes

Con el fin de agrupar los elementos del software en estructuras lógicas, se modela el diagrama de componentes que representa cómo el sistema de software es dividido en componentes y muestra las dependencias entre estos.

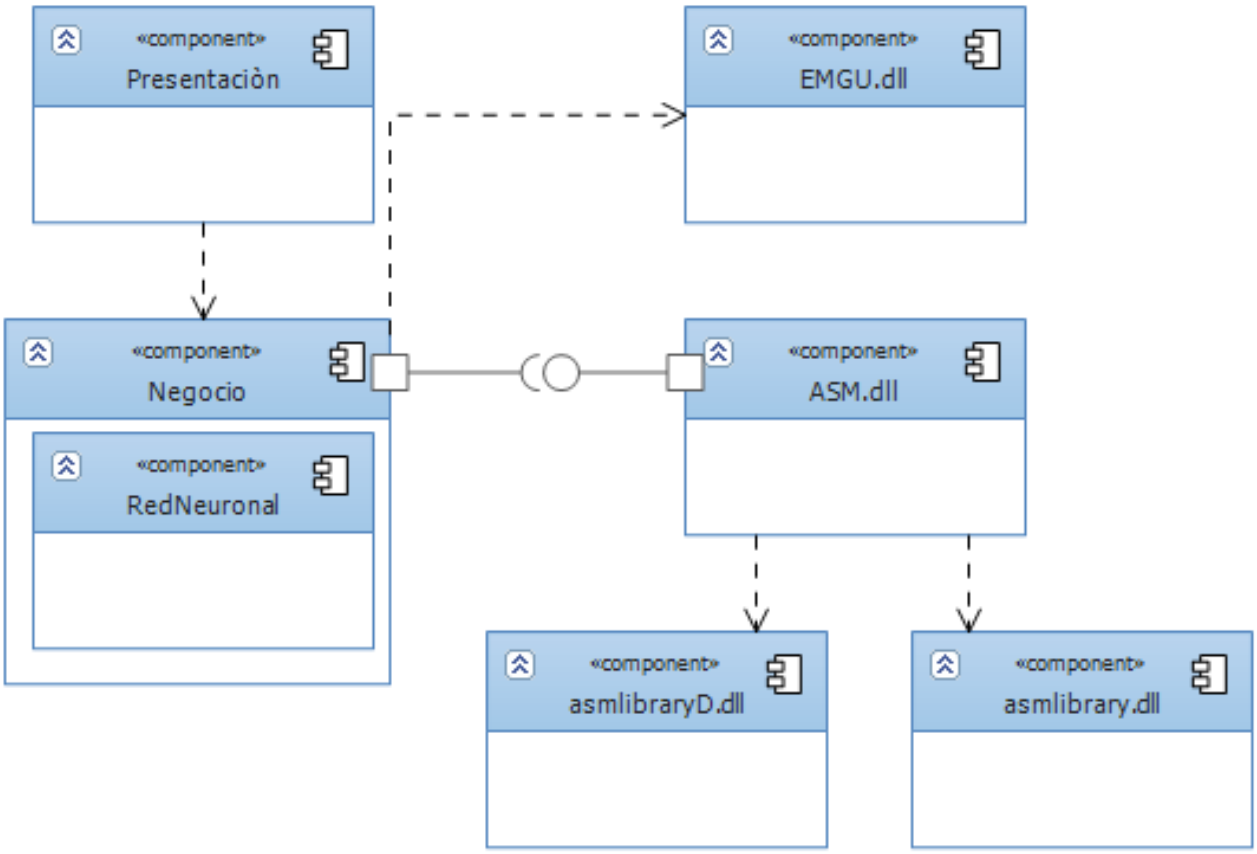


Figura 3.1 Diagrama de componentes

Tabla 3.17 Descripción de componentes

Componente	Propósito
Presentación	Componente que contiene la interfaz de usuario.

Negocio	Componente que contiene la implementación de las funcionalidades que se realizan con imágenes y el componente RedNeuronal.
RedNeuronal	Componente que contiene la implementación y el trabajo con las redes neuronales.
EMGU	Componente que se utiliza como interfaz para la utilización de las funcionalidades de la biblioteca de clases OpenCV.
ASM	Componente que se utiliza para la detección de los puntos característicos de la imagen.
asmlibrary	Componente auxiliar que es usado por ASM.dll para la detección de los puntos característicos.
asmlibraryD	Componente auxiliar que es usado por ASM.dll para la detección de los puntos característicos.

Pruebas

Con el fin de detectar errores y verificar la calidad del software se efectúan a los sistemas biométricos un conjunto de pruebas en función del tipo de reconocimiento que realizan.

En este caso se está en presencia de sistemas de reconocimiento de identificación en un conjunto cerrado ya que el objetivo es clasificar una expresión facial en un conjunto de siete posibles, por lo que se procede a realizar las siguientes pruebas basadas en el estudio de **(24)**:

Errores de clasificación

En la identificación en conjunto cerrado, todas las realizaciones de prueba corresponden a una de las expresiones faciales predefinidas. Por tanto, se producirá un error de clasificación para la realización *k* de la expresión *Xi* cuando

$$\delta\left[\hat{\imath}\left(x_i^k\right), i\right] = 0$$

donde δ denota la función delta de Kronecker, que vale 1 si los dos argumentos son iguales, y 0 en el resto de los casos, $\hat{\imath}(x_i^k)$ denota el resultado de la k-ésima prueba de la expresión *i*, i denota la expresión esperada.

El rendimiento de un sistema de identificación se mide calculando el número relativo de veces (%) que el sistema falla en identificar correctamente la expresión de entrada, o lo que es lo mismo, con qué frecuencia una realización de prueba es asignada a una expresión errónea. Este valor se calcula por cada expresión facial. Veamos, sin embargo, cómo se obtiene este valor para un sistema global.

- error de clasificación por expresión *Xi*

$$\gamma \mathrm{i} = 1 - \frac{1}{c}\sum_{k=1}^{ci} \delta\left[\hat{\imath}\left(x_i^k\right), i\right]$$

- error de clasificación promedio

$$\bar{\gamma} = \frac{1}{m}\sum_{i=1}^{m} \gamma i$$

  Donde *m* es el total de expresiones faciales, $\boldsymbol{ci}$ es la cantidad de pruebas realizadas a la expresión *i* y *c* es la suma de los $\boldsymbol{ci}$.

Tasas de desconfianza

Abordado desde otro punto de vista, podemos diseñar las tasas de rendimiento de los sistemas en función del grado de fiabilidad cuando el sistema asigna una determinada identidad, o en otras palabras, disponer de un estimador de la probabilidad de que la expresión facial no sea realmente *Xi* cuando el sistema da a *Xi* como la expresión más verosímil. Para ello, llamaremos:

$$\widehat{c_\imath} = \sum_{k=1}^{c} \delta\left[\hat{\imath}\left(x^k\right), i\right]$$

$$\widehat{m}^* = \sum_{i=1}^{m} \widehat{m}_i^*$$

donde $\widehat{m}_i^*$=1 si $\widehat{\boldsymbol{c}_\iota} \neq 0$ y $\widehat{m}_i^*$=0 si $\widehat{\boldsymbol{c}_\iota} = 0$

Donde $\widehat{\boldsymbol{c}_\iota}$ es la cantidad de veces que se obtuvo como resultado la expresión *i* en el total de pruebas, $\widehat{m}^*$ es el número de expresiones identificadas al menos una vez en el total de pruebas.

- La **tasa de desconfianza** para la expresión *Xi*

$$\widehat{\gamma_\iota} = 1 - (1 - \gamma i)\frac{ci}{\hat{c}_i}$$

Tasa de desconfianza promedio

$$\bar{\hat{\gamma}} = \frac{1}{\widehat{m}^*}\sum_{i=1}^{m} \widehat{\gamma_\iota}$$

Resultado de las pruebas

Para la realización de las pruebas al software se utilizó la base de datos de expresiones faciales femeninas japonesas (jaffe por sus siglas en inglés) constituida por 213 imágenes de 7 expresiones faciales (6 expresiones faciales básicas más el estado neutral). De esta se extrajeron 24 imágenes para realizar las pruebas y 189 se utilizaron para el entrenamiento de la red neuronal.

- error de clasificación para la expresión ira:

  cantidad de imágenes: 4

  cantidad de aciertos: 3

  resultado: $\gamma ira = 0.25$

- error de clasificación para la expresión disgusto:

  cantidad de imágenes: 3

  cantidad de aciertos: 3

  resultado: $\gamma disgusto = 0$

- error de clasificación para la expresión miedo:

  cantidad de imágenes: 4

  cantidad de aciertos: 1

  resultado: $\gamma miedo = 0.75$

- error de clasificación para la expresión felicidad:

  cantidad de imágenes: 4

  cantidad de aciertos: 2

  resultado: $\gamma felicidad = 0.5$

- error de clasificación para la expresión tristeza:

  cantidad de imágenes: 3

  cantidad de aciertos: 1

  resultado: $\gamma tristeza = 0.67$

- error de clasificación para la expresión neutral:

  cantidad de imágenes: 3

  cantidad de aciertos: 3

resultado: $\gamma neutral = 0$

- error de clasificación para la expresión sorpresa:

  cantidad de imágenes: 3

  cantidad de aciertos: 2

  resultado: $\gamma sorpresa = 0.33$

- error de clasificación promedio

  $\bar{\gamma} = \frac{1}{7}(\gamma ira + \gamma disgusto + \gamma miedo + \gamma felicidad + \gamma tristeza + \gamma neutral + \gamma sorpresa)$

  $\hat{\gamma} = 0.36$

Tasa de desconfianza

- Tasa de desconfianza para la expresión ira:

  $\gamma ira = 0.25$

  $\bar{\hat{\gamma}}ira = 1 - (1 - 0.25)\frac{3}{3}$

  $\bar{\hat{\gamma}}ira = 0.4$

- Tasa de desconfianza para la expresión disgusto:

  $\gamma disgusto = 0$

  $\bar{\hat{\gamma}}disgusto = 1 - (1 - 0)\frac{3}{10}$

  $\bar{\hat{\gamma}}disgusto = 0.4$

- Tasa de desconfianza para la expresión miedo:

$\gamma miedo = 0.75$

$\bar{\hat{\gamma}} miedo = 1 - (1 - 0.75)\frac{4}{1}$

$\bar{\hat{\gamma}} miedo = 0$

- Tasa de desconfianza para la expresión felicidad:

$\gamma felicidad = 0.5$

$\bar{\hat{\gamma}} felicidad = 1 - (1 - 0.5)\frac{4}{2}$

$\bar{\hat{\gamma}} felicidad = 0$

- Tasa de desconfianza para la expresión tristeza:

$\gamma tristeza = 0.67$

$\bar{\hat{\gamma}} tristeza = 1 - (1 - 0.67)\frac{3}{3}$

$\bar{\hat{\gamma}} tristeza = 0.67$

- Tasa de desconfianza para la expresión neutral:

$\gamma neutral = 0$

$\bar{\hat{\gamma}} neutral = 1 - (1 - 0)\frac{3}{4}$

$\bar{\hat{\gamma}} neutral = 0.25$

- Tasa de desconfianza para la expresión sorpresa:

$\gamma sorpresa = 0.33$

$$\bar{\hat{\gamma}}sorpresa = 1 - (1 - 0.33)\frac{3}{3}$$

$$\bar{\hat{\gamma}}sorpresa = 0.33$$

- Tasa de desconfianza promedio:

$$\bar{\hat{\gamma}} = \frac{1}{7}\left(\bar{\hat{\gamma}}ira + \bar{\hat{\gamma}}disgusto + \bar{\hat{\gamma}}miedo + \bar{\hat{\gamma}}felicidad + \bar{\hat{\gamma}}tristeza + \bar{\hat{\gamma}}neutral + \bar{\hat{\gamma}}sorpresa\right)$$

$$\bar{\hat{\gamma}} = 0.29$$

Como se explica en **(22)** cuanto más complejo sea el problema a modelar, más grande deberá ser la red y, por lo tanto, más ejemplos se necesitarán para entrenarla, ejemplos que deberán cubrir todo el espacio de entrada, contemplando todas las situaciones posibles. Debido a la evidente complejidad de este estudio se puede afirmar que el número de patrones-ejemplo disponibles es limitado, ya que, tomando en consideración la topología y características de la red neuronal usada en esta investigación, para obtener un error de un 10% serían necesarias 920 imágenes en el conjunto de entrenamiento y solo se cuenta con 189.

Teniendo en cuenta el estudio realizado del estado de arte de “Facial Expression Recognition System using Statical Feature and Neural Network” donde se aprecian trabajos similares al nuestro con porcientos de efectividad de 61.11%, 85.7% y 84.7% y que además, la cantidad de imágenes que contiene el conjunto de entrenamiento es relativamente pequeño; se consideran como favorables los resultados obtenidos del error de clasificación (36%) y la tasa de desconfianza (29%).

Conclusiones

En este capítulo se ha definido el estilo de codificación utilizado en la implementación del sistema, lo que permitió mantener una uniformidad en la codificación y mayor claridad a la hora de leer o agregar código. Se confeccionó el diagrama de componentes a través del cual se observó las dependencias entre los elementos físicos que componen al sistema. Se realizaron pruebas al

software con el objetivo de verificar la calidad del sistema obteniéndose un grado de efectividad de 64%.

CONCLUSIONES GENERALES

A partir de la necesidad de la universidad de contar con un sistema FER que pudiera integrar a diversos proyectos para enriquecerlos creando interfaces hombre-máquina más inteligente se diseñó e implementó un FER capaz de reconocer el estado de ánimo de una persona a partir de sus expresiones faciales (ira, tristeza, miedo, disgusto, felicidad y sorpresa además del estado neutral).

Para esto se realizó el estudio de tres soluciones FER desarrolladas a nivel internacional, lo que permitió analizar e identificar las principales características, vías de solución y grado de efectividad para clasificar estas expresiones.

Se analizaron y escogieron las herramientas, tecnologías y metodología a utilizar para el desarrollo del sistema de acuerdo a las características requeridas lo que permitió crear una buena solución con el menor costo de construcción posible.

Se siguieron los principios básicos de diseño e implementación descritos para el desarrollo del sistema obteniéndose un software que cumple con los requisitos esperados.

Con el fin de verificar la calidad del software se realizaron pruebas de errores de clasificación y tasas de desconfianza que arrojaron un resultado de 36% y 29% respectivamente. En comparación con los software estudiados y teniendo en cuenta que se contaba con una base de datos que no contenía la cantidad de imágenes necesaria para el correcto entrenamiento de la red neuronal se concluye que los resultados obtenidos fueron satisfactorios.

RECOMENDACIONES

Con el fin de mejorar el sistema FER se recomienda:

Desarrollar una base de datos robusta respecto a la cantidad de imágenes para lograr un mejor entrenamiento de la red neuronal y obtener mayor éxito en la identificación de expresiones faciales.

Realizar la identificación no solo a imágenes previamente almacenadas sino también a imágenes en tiempo real a través de la interacción del sistema FER con una cámara.

GLOSARIO DE TÉRMINOS

Acoplamiento: Es una medida de la fuerza con que una clase está conectada a otras clases, con que las conoce y con que recurre a ellas.

Cohesión: En la perspectiva del diseño orientado a objetos, la cohesión (o, más exactamente, la cohesión funcional) es una medida de cuán relacionadas y enfocadas están las responsabilidades de una clase.

GOF: El grupo de los cuatro (Gang of Four, por sus siglas en inglés), constituido por Erich Gamma, Richard Helm, Ralph Johnson y John Vlissides, realizaron el primer catálogo de patrones de diseño.

Instancia: Miembro individual de un tipo o de una clase.

Interfaz: Conjunto de representaciones de operaciones públicas.

Patrón: Es la descripción etiquetada de un problema, de la solución, de cuándo aplicar la solución y la manera de hacerlo dentro de otros contextos.

REFERENCIAS BIBLIOGRÁFICAS

1. Proyecto Ing. Software. [En línea] 6 de septiembre de 2007. http://ingsoftware8.blogspot.com/2007/09/metodologias-rup-y-xp.html.

2. **OpenCV.** OpenCV. [En línea] http://ubaa.net/shared/processing/opencv/.

3. **Rothkrantz, M. Pantic y L. J. M.** *Expert system for automatic analysis of facial expressions.*

4. **Microsoft.** Microsoft. [En línea] http://www.microsoft.com/spain/enterprise/perspectivas/contenido-detalle.aspx?ContenidoTipo=lo-ultimo&ContenidoID=20100421001.

5. **Calabria, Luis.** *Metodología FDD.* 2003.

6. **Mateos Ortiz, Luis Gilberto.** *OpenCV. Guía de Instalación para Windows.*

7. **Seco, José Antonio González.** *El lenguaje de programación C#.*

8. **EmguCV.** EmguCV. [En línea] http://www.emgu.com.

9. **Eckel, Bruce.** *Pensar en C++.* s.l. : volumen 1 , 2012.

10. **Bram Adrams, Krzysztof Cwalina.** *Framework Design Guidelines.* s.l. : segunda edición.

11. **Troelsen, Andrew.** *C# and the .NET Framework.* 2001.

12. **Anders Hejlsberg, Scott Wiltamuth y Peter Golde.** *C# Language Specification.*

13. **Mehrabian, A.** *Communication without words.* s.l. : Psychology Today 2, 1968.

14. **Amaro Calderón, Sarah Dámaris, Valverde Rebaza. Jorge Carlos.** *Metodologías Ágiles.* Perú : s.n., 2007. 1.

15. **Juan, Francisco Javier Martínez.** *Guía de construcción de software en java con patrones de diseño.*

REFERENCIAS BIBLIOGRÁFICAS

16. **Larman, Craig.** *UML y patrones.*

17. **Phillips, Dwayne.** *Image Processing in C .* s.l. : Second Edition, 2000.

18. **http://www.ecured.cu/index.php/Red_neuronal ecured Miércoles, 8 de mayo de 2013.** ecured. [En línea] [Citado el: 8 de abril de 2013.] http://www.ecured.cu/index.php/Red_neuronal .

19. magazine. [En línea] http://msdn.microsoft.com/es-es/magazine/hh975375.aspxmsdn .

20. revista tino. [En línea] http://archivo.revista.jovenclub.cu/index.php?option=com_content&task=view&id=461&Itemid=80&limit=1&limitstart=2.

21. msdn. [En línea] http://msdn.microsoft.com/es-es/library/dd409445.aspx.

22. **Bonifacio Martín del Brío, Alfredo Sanz Molina.** *Redes Neuronales y Sistemas Difusos.* s.l. : segunda edición, 2001.

23. *Detección de caras y análisis de expresiones faciales.* s.l. : 3.2.2. Redes neuronales.

24. **EscuelaPolitécnicaSuperior.** *Ampliación de Señales Aleatorias/Reconocimiento Biométrico. Sistemas biométricos e identificación personal.*

BIBLIOGRAFÍA

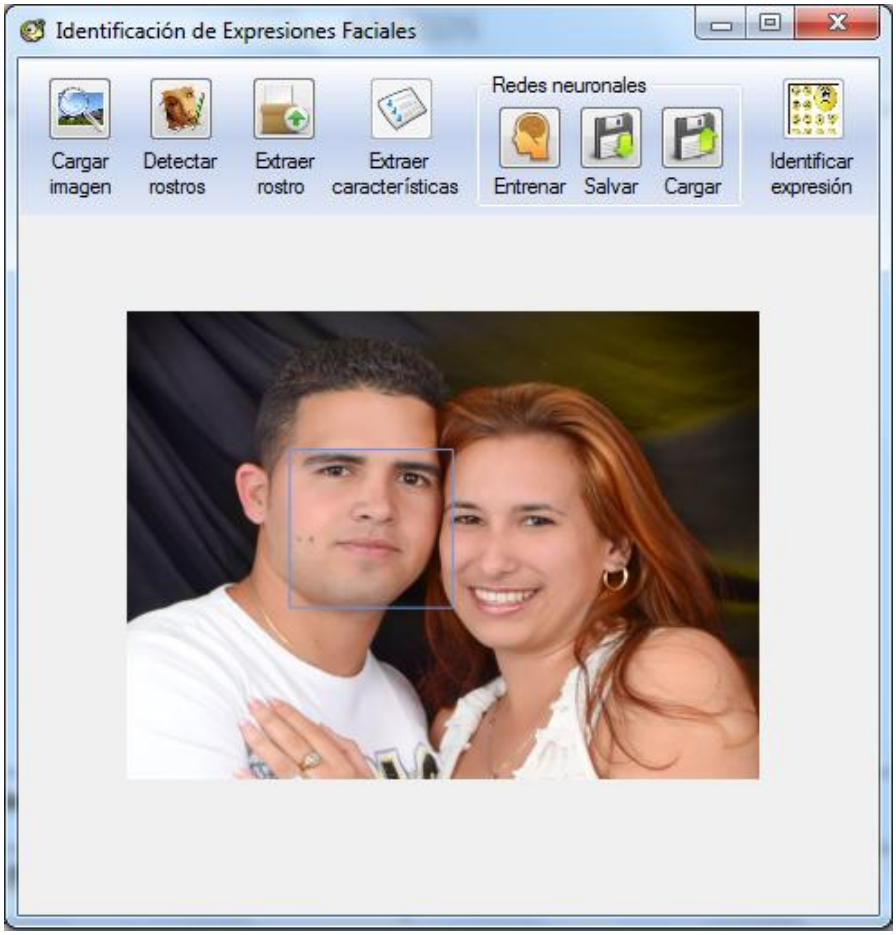
- Renuka R. Londhe, Dr. Vrushshen P. Pawar**.***Analysis of facial expression on recognition based on statistical approach.* 2012
- M. Kato, I. So, Y. Hishinuma, O. Nakamura, T. Minami.*Description and synthesis of facial expressions based on isodensity maps.* Tokyo : s.n., 1991.
- M.J. Black, Y. Yacoob.*Recognising Facial Expressions in Image Sequences using Local Parameterised Models of Image Motion.* 1998.
- *Lenguaje Unificado de Modelado Unified Modeling Language.* 2011.
- Stan Z. Li, Anil K. Jain. *Handbook of face recognition.* 2004.
- *Facial Expression Recognition System using Statistical Feature and Neural Network.* 2012.

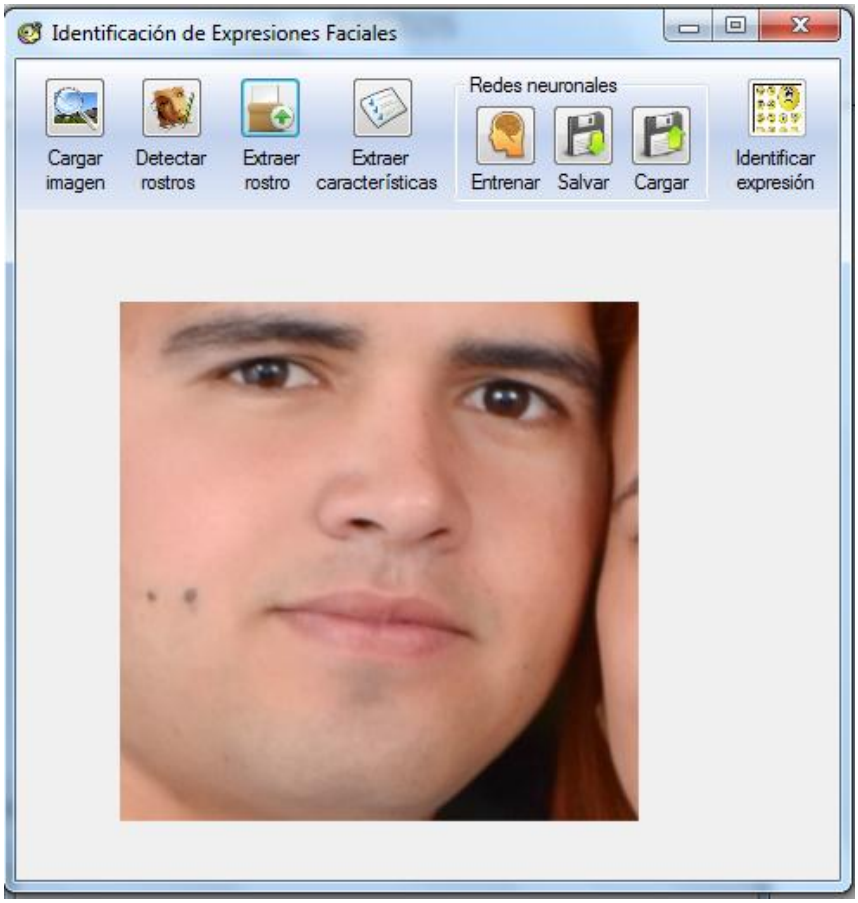
ANEXOS

Anexo 1: Descripción de característica. Extraer rostro.

Nombre de la característica	Extraer rostro.
Precondiciones	Se debe haber detectado el rostro
Características asociadas	C4, C5, C7,C8
Conceptos tratados	UtilidadesImagen, Imagen, Rostro.
Descripción básica	1. El usuario deja seleccionado el rostro con el que desea trabajar y selecciona la opción del menú **Extraer rostro**. 2. El sistema muestra en la imagen solo el rostro extraído y habilita la opción del menú “Extraer características”. 3. Si el usuario selecciona la opción del menú **Extraer características**, (ver tabla 2.5 Descripción de característica. Extraer característica).

ANEXOS

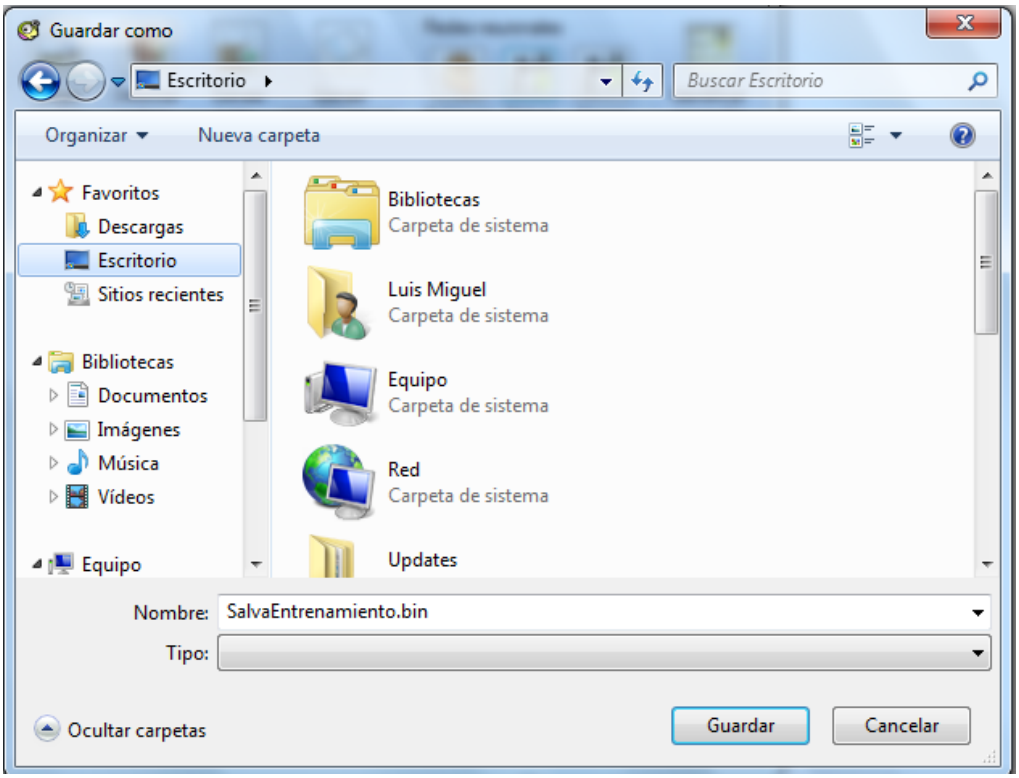

Identificación de Expresiones Faciales
Cargar imagen
Detectar rostros
Extraer rostro
Extraer características
Redes neuronales
Entrenar
Salvar
Cargar
Identificar expresión



Descripción alterna	No tiene
Poscondiciones	Queda mostrado solo el rostro extraído

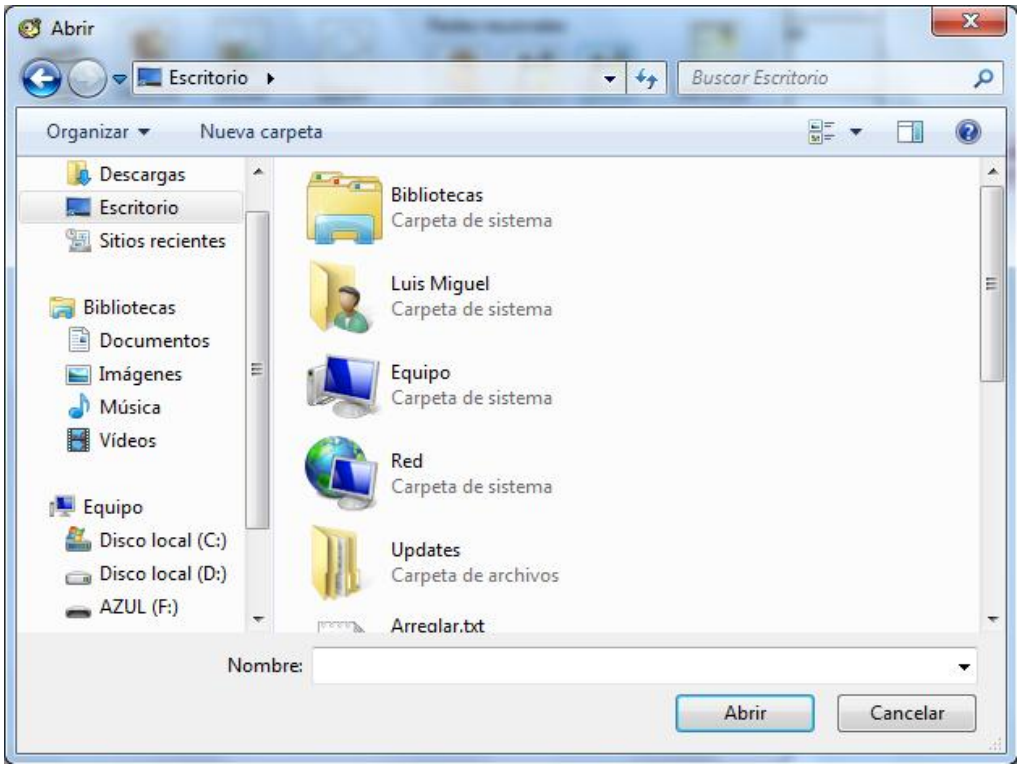
Anexo 2: Descripción de característica. Salvar entrenamiento.

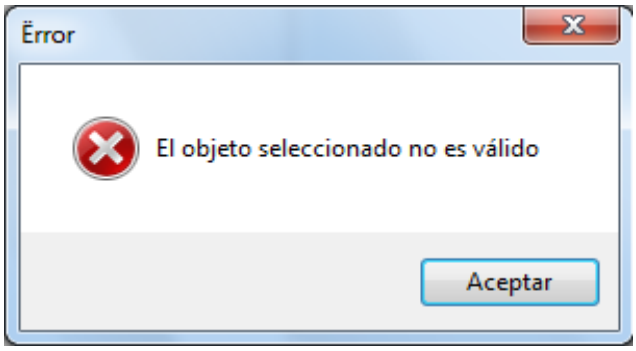
Nombre de la característica	Salvar entrenamiento.
Precondiciones	Debe haber sido entrenada la red neuronal.
Características asociadas	C1, C3.
Conceptos tratados	Red Neuronal.

Descripción básica	1. El usuario selecciona la opción del menú **Salvar Red Neuronal**. 2. El sistema muestra una ventana para que el usuario escoja la ubicación donde quiere salvar.
	
Descripción alterna	No tiene.
Poscondiciones	Queda salvado el entrenamiento realizado.

Anexo 3: Descripción de característica. Cagar entrenamiento.

Nombre de la característica	Cargar entrenamiento.
Precondiciones	No tiene.
Características asociadas	C1, C2.

Conceptos tratados	Red Neuronal.
Descripción básica	1. El usuario selecciona la opción del menú **Cargar entrenamiento**. 2. El sistema muestra una ventana para que el usuario seleccione el entrenamiento a cargar.
	
Descripción alterna	Si el objeto que el usuario selecciona no es un entrenamiento el sistema muestra un mensaje “El objeto seleccionado no es válido”.

<table>
<tr><td colspan="2"> 
</td></tr>
<tr><td>Poscondiciones</td><td>Se carga correctamente el entrenamiento.</td></tr>
</table>