



UNIVERSIDAD DE BURGOS
ESCUELA POLITÉCNICA SUPERIOR
Grado en Ingeniería Informática



**TFG del Grado en Ingeniería
Informática**

**Diseño de un sistema de
identificación de personas
Documentación Técnica**



Presentado por Víctor de Castro Hurtado
en Universidad de Burgos — 24 de junio
de 2018

Tutor: César Represa Pérez

Índice general

Índice general	I
Índice de figuras	III
Índice de tablas	vii
Apéndice A Plan de Proyecto Software	1
A.1. Introducción	1
A.2. Planificación temporal	2
A.3. Estudio de viabilidad	16
Apéndice B Especificación de Requisitos	17
B.1. Introducción	17
B.2. Objetivos generales	18
B.3. Catalogo de requisitos	19
B.4. Especificación de requisitos	22
Apéndice C Especificación de diseño	31
C.1. Introducción	31
C.2. Diseño de datos	31
C.3. Diseño procedimental	37
C.4. Diseño arquitectónico	42
Apéndice D Documentación técnica de programación	47
D.1. Introducción	47
D.2. Estructura de directorios	48
D.3. Manual del programador	52

D.4. Compilación, instalación y ejecución del proyecto	55
D.5. Pruebas del sistema	56
Apéndice E Documentación de usuario	61
E.1. Introducción	61
E.2. Requisitos de usuarios	61
E.3. Instalación	62
E.4. Manual del usuario	62
Bibliografía	75

Índice de figuras

A.1. Primer commit	3
A.2. Commit con el que finaliza la parte del planteamiento del proyecto.	3
A.3. Commit con los recursos instalados en el proyecto.	7
A.4. Commit con las primeras imágenes de prueba.	8
A.5. Commit con el que finaliza la fase de abrir, capturar, guardar y modificar una imagen.	8
A.6. Commit que solucionaba el problema de recursos no compatibles con las nuevas versiones de python.	8
A.7. Commit con la implementación de la captura de la cámara.	9
A.8. Commit con la localización del rostro en una imagen implementada.	9
A.9. Commit con la normalización de la iluminación de la imagen.	9
A.10. Reconocimiento facial utilizando el método LBPH.	10
A.11. Commit con cambios en la implementación de los métodos que permiten el reconocimiento facial (parte I).	10
A.12. Commit con cambios en la implementación de los métodos que permiten el reconocimiento facial (parte II).	10
A.13. Commit con cambios en la implementación de los métodos que permiten el reconocimiento facial (parte III).	11
A.14. Commit con cambios en la implementación de los métodos que permiten el reconocimiento facial (parte I).	11
A.15. Commit con la implementación de ventana de la nueva interfaz.	11
A.16. Commit con el menú final de cara al usuario final.	12
A.17. Commit con la implementación de la barra de progreso.	12
A.18. Commit con el <i>popup</i> y la información extra.	12
A.19. Commit solucionando el problema de las ventanas <i>popup</i>	13
A.20. Commit con el ejecutable base y su mejora.	13
A.21. Commit con la primera versión de la memoria en LaTex.	13

A.22.Commit con una de las actualizaciones del fichero README.md.	14
A.23.Grafo con el número de commits por semana.	15
B.1. Diagrama con los principales casos de uso de nuestra aplicación.	22
C.1. Estructura del fichero principal: <i>Main.py</i>	32
C.2. Estructura del fichero de utilidad: <i>Util.py</i>	32
C.3. Estructura del fichero encargado de interaccionar con ficheros: <i>Files.py</i>	33
C.4. Estructura del fichero encargado de interaccionar con la cámara: <i>Camera.py</i>	33
C.5. Estructura del fichero encargado de entrenar la red: <i>Trainer.py</i> .	34
C.6. Estructura del fichero que se encarga de la interfaz en modo texto: <i>TextInterface.py</i>	34
C.7. Estructura de la clase que se encarga de la interfaz visual: <i>GUIClass</i> .	35
C.8. Estructura del fichero que se encarga de predecir el resultado: <i>CompareImages.py</i>	36
C.9. Estructura del fichero que reconoce las imágenes en tiempo real: <i>RecognizerRealTime.py</i>	36
C.10.Relación entre entidades que representan la gestión de la base de datos.	37
C.11.Relación entre entidades que representan el entrenamiento de la red.	38
C.12.Relación entre entidades que representan la obtención de una imagen para reconocerla.	39
C.13.Relación entre entidades que representan el reconocimiento de caras.	40
C.14.Relación entre entidades que representan el muestreo de resultados.	41
C.15.Arquitectura final simplificada de nuestra aplicación.	42
C.16.Ejemplo de patrón <i>Singleton</i> utilizado en la clase <i>GUIClass</i> . . .	43
C.17.Implementación del patrón <i>Singleton</i>	43
C.18.Ejemplo de patrón <i>Abstract Factory</i> que se tenía pensado utilizar para crear las diferentes interfaces.	44
C.19.Ejemplo de patrón <i>Adapter</i> que se utilizaba al principio para transformar las imágenes al tipo adecuado.	45
C.20.Implementación final sustituyendo al patrón <i>Adapter</i>	45
C.21.Ejemplo de patrón <i>Builder</i> que se utilizaba al principio para ir modificando la interfaz.	46
C.22.Implementación final sustituyendo al patrón <i>Builder</i>	46

D.1. Entorno virtual de <i>Anaconda</i> utilizado para la instalación de <i>python 2.7</i>	52
D.2. Recursos utilizados por nuestra aplicación cuando se está realizando el reconocimiento en tiempo real al ejecutarla desde el fichero ejecutable en el sistema <i>run.bat</i>	56
D.3. Recursos utilizados por nuestra aplicación cuando se está realizando el reconocimiento en tiempo real al ejecutarla desde el editor <i>pycharm</i>	56
D.4. Comparación entre una imagen guardada en la base de datos, y otra desde la cámara en tiempo real (rasgo distintivo: muy mala iluminación - 48%).	57
D.5. Comparación entre una imagen guardada en la base de datos, y otra desde la cámara en tiempo real (rasgo distintivo: solo visibles los ojos - 79 %).	58
D.6. Comparación entre una imagen guardada en la base de datos, y otra desde la cámara en tiempo real (rasgo distintivo: foto de casi perfil - 56.5 %).	59
E.1. Programa preguntándonos si queremos introducir una nueva imagen en la base de datos.	64
E.2. Imagen que se muestra por pantalla para que elijamos cuando capturarla.	64
E.3. Programa preguntándonos si queremos sobre-escribir la imagen.	65
E.4. Programa preguntándonos los datos relacionados con la imagen.	65
E.5. Información añadida al fichero <i>info.txt</i>	66
E.6. Programa preguntándonos si queremos entrenar o no la red. .	66
E.7. Programa cargando los recursos almacenados previamente. . . .	66
E.8. Programa entrenando la red.	67
E.9. Programa cargando los nuevos recursos creados.	67
E.10. Programa preguntándonos cómo queremos continuar la ejecución.	68
E.11. Parte de la interfaz que nos permite filtrar el tipo de archivos. .	68
E.12. Interfaz que nos permite buscar un archivo en nuestro sistema. .	69
E.13. Inicialización de la cámara con la resolución que se va a usar. .	69
E.14. La cámara no se pudo encender y se cargó una imagen por defecto.	70
E.15. Programa preguntándonos si queremos salir de la aplicación. .	71
E.16. Programa comparando imágenes internamente y obteniendo un resultado.	71
E.17. Interfaz final con ambas imágenes, la barra y el porcentaje de coincidencia (en este caso la interfaz muestra el proceso de reconocimiento en tiempo real, por eso no se encuentra el botón <i>More information...</i> explicado a continuación).	72

E.18. <i>Pop-up</i> con la información de la persona reconocida.	73
E.19. Imagen original de la base de datos // Imagen recortada y reescalada para dejarla en un formato tratable para la red.	73
E.20. Menú principal de la cámara.	74
E.21. Mensajes que se muestran cuando se pausa la cámara.	74
E.22. Información que se muestra cuando se pulsa [I].	74

Índice de tablas

A.1.	Distribución del desarrollo del proyecto según Milestones e Issues.	6
B.1.	<i>CU 1.- Gestionar la base de datos</i>	23
B.2.	<i>CU 2.- Entrenar la red</i>	24
B.3.	<i>CU 3.- Obtener imagen a identificar</i>	25
B.4.	<i>CU 3.1.- Obtener imagen desde una captura</i>	26
B.5.	<i>CU 3.2.- Obtener imagen desde un fichero</i>	27
B.6.	<i>CU 3.3.- Obtener imagen en tiempo real</i>	28
B.7.	<i>CU 4.- Reconocer caras</i>	29
B.8.	<i>CU 5.- Mostrar resultados</i>	30

Apéndice A

Plan de Proyecto Software

A.1. Introducción

Aunque nos hemos basado en un método de desarrollo en cascada [67], hemos incluido algunos elementos de la **metodología scrum** [70] como las reuniones mensuales/semanales.

En cuanto al desarrollo en cascada, hemos tomado la base de este método, ya que teníamos unos requisitos iniciales, que eran los objetivos primarios que teníamos que cumplir (se pueden encontrar en la sección *2_Objetivos_del_proyecto* de la memoria principal), a partir de los cuales se hizo un primer diseño del proyecto, tras lo cual se pasó al desarrollo e implementación.

En cuanto a la *metodología scrum*, incluimos reuniones con el tutor. Las primeras para ver que estructura tenía el proyecto y que posibles cambios iniciales se podrían realizar, y el resto de ellas para ver cómo iba avanzando el proyecto, lo que necesitaba ser mejorado o cambiado, o asignar nuevas tareas hasta la próxima reunión (en nuestro caso la mayoría fueron semanales o quincenales, en vez de mensuales).

A.2. Planificación temporal

Las fases que se han seguido a la hora de desarrollar el proyecto han sido [58]:

- **Definir el proyecto**

Al principio se tuvo que escoger el tipo de proyecto que se iba a realizar. En nuestro caso un proyecto medio, ya que no se tenían ni el tiempo ni los recursos necesarios para que fuera un proyecto demasiado grande, y tampoco podía ser un proyecto demasiado pequeño dado el objetivo del mismo.

También se tuvo que definir la idea principal, que en nuestro caso iba a ser reconocer gente.

- **Identificar información, recursos y requisitos**

A continuación, se definió qué tipo de información iba a tratar: íbamos a trabajar con imágenes en sus diferentes formatos, y con redes neuronales, las cuales íbamos a entrenar para que reconocieran a la gente almacenada en nuestra base de datos.

Los recursos de los que íbamos a disponer eran:

1. La imagen que obtuviéramos con la cámara.
2. Otras imágenes de gente a la que necesitáramos reconocer (en nuestro caso gente desaparecida o en busca y captura). Dichas imágenes se obtendrían de una base de datos oficial de la policía/gobierno.
3. Información básica sobre cada una de las personas que tuviéramos almacenadas en la base de datos de nuestro programa.

En cuanto a los requisitos que necesitábamos cumplir, se establecieron los siguientes (se entrará en más detalle en la sección *B_Requisitos B*):

1. Obtener imágenes en tiempo real.
2. Entrenar una red neuronal con *machine-learning* o derivados (*deep learning, computer vision, etc*).
3. Tener una base de datos con imágenes que usaríamos para entrenar dicha red.
4. Realizar una comparación satisfactoria de la persona identificada en la imagen en tiempo real.
5. Mostrar resultados e información en una interfaz al usuario.

Estas primeras fases del proyecto (identificación de recursos y requisitos, planteamiento del proyecto, definición y diseño del proyecto) duró aproximadamente dos semanas, desde la primera de Febrero que se creó el proyecto en *Github A.1*, hasta mediados del mismo mes, que se pasó a la fase de desarrollo al empezar a instalar el material necesario *A.2*.



Figura A.1: Primer commit.

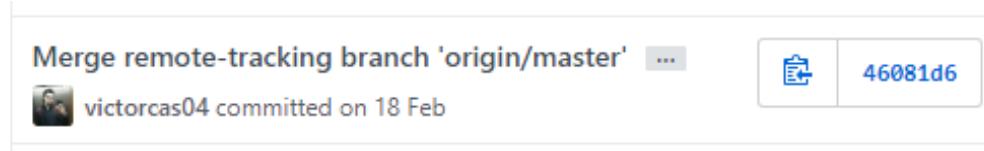


Figura A.2: Commit con el que finaliza la parte del planteamiento del proyecto.

- **Fase de planificación**

Cuando se empezó a desarrollar el proyecto, se sabía cual era el objetivo general del proyecto, aunque no el camino exacto que iba a seguir, de manera que se optó por seguir una serie de hitos principales, enumerados en el apartado anterior, y a partir de esos hitos ir creando otros más pequeños y asequibles a la hora del desarrollo.

No se pretendía ir completando los hitos uno por uno y que no se pudiera empezar a desarrollar uno hasta que se acabara el anterior, de manera que el progreso se ha ido realizando sobre todos ellos de manera más o menos equivalente.

Los hitos grandes no tenían fecha prevista de ser completados, sino que se completarían cuando no quedaran tareas pendientes relacionadas con dicho hito, de manera que era complicado planificar el desarrollo para que coincidiera con unos plazos concretos, así que se planificaron el resto de tareas más pequeñas, completando varias para cada reunión.

Dichas reuniones se llevaban a cabo en función del trabajo planteado de una a otra, así que no eran todas las semanas, sino que se adaptaban un poco al trabajo pendiente, aunque si que es cierto que se han tenido reuniones cada (como máximo) dos semanas, para llevar un seguimiento más o menos regular del desarrollo y que no se quede el mismo estancado en el mismo punto demasiado tiempo.

En las herramientas que se han barajado utilizar respecto a la planificación, se barajó utilizar *Trello* [6], aunque se descartó ya que, sin otros miembros que aporten contenido, no tenía sentido tener una lista de tareas *to do* o *doing* para una sola persona.

También se pensó en utilizar *Zenhub* [74] para la planificación de hitos, tareas y sprints, aunque se descartó la idea al no tener unos plazos fijos de entrega para cada una de ellas.

- **Fase de investigación**

A la vez que se realizaba la planificación de las tareas semanales, se iba realizando un proceso de investigación e información, tanto de conceptos y mecánicas como de seguimiento de tutoriales, documentación y diferentes referencias sobre los temas tratados en el proyecto (para más información consultar el apartado *3_Conceptos_teóricos* de la memoria principal).

Aunque fue durante esta fase cuando se realizó la mayor parte de investigación, ha sido un proceso continuo y constante de aprendizaje, realizando nuevos descubrimientos hasta la última semana de desarrollo.

- **Desarrollo**

En este apartado hemos ido realizando el desarrollo del propio proyecto, tanto la estructura de la fase de planificación como el propio código. Para ello se ha seguido un orden lógico de desarrollo, centrándonos primero en la funcionalidad básica de tomar una imagen de la cámara y compararla con las de la base de datos, para pasar más tarde a ampliar el número de muestras de nuestra red para mejorar el ratio de acierto de las predicciones, crear un interfaz gráfico para mostrar los resultados y que al usuario le resulte más sencilla su interpretación y crear un ejecutable para evitar tener que ejecutar nuestro código completo en un editor.

Estas últimas modificaciones se toman para facilidad del usuario, intentando automatizar todo lo posible el proceso, aunque es cierto que se necesitan ciertos datos por parte del usuario, además de permitirle cierta libertad (por ejemplo, podríamos entrenar la red a cada ejecución, incluso si nuestras muestras no han variado y la red está correctamente entrenada, pero supondría una pérdida de tiempo y recursos en muchos casos, de manera que se opta por darle opción al usuario).

En la tabla A.1 se pueden observar tanto los milestones principales del desarrollo del proyecto como las issues o tareas asociadas a cada uno.

Tabla A.1: Distribución del desarrollo del proyecto según Milestones e Issues.

Milestones	Issues	Type of Issue
1.- Install and Configure	Install environment Create class diagram Basic readme	Develop Develop Develop
2.- Image manipulation	Open image from file Save images Tensorflow Tutorials Meeting	Develop Develop Develop Meeting
3.- Basic image recognition	Import OpenCV Capture image from camera Normalize illumination Create basic parameters Facial recognition (I) Meeting	Develop Develop Enhancement Develop Develop Meeting
4.- Apply computer vision	Improve camera recording Change face location display Facial recognition (II) Meeting Acquire camera	Enhancement Enhancement Develop Meeting Enhancement
5.- Create database	Create local storage Get more images	Develop Enhancement
6.- Interface	Basic interface Dynamic window Facial recognition (III) Camera view Result images Compare bar/percentage Meeting	Develop Enhancement Enhancement Enhancement Develop Develop Meeting

Continúa en la siguiente página...

Tabla A.1 – Continúa desde la página anterior...

Milestones	Issues	Type of Issue
	Result information Executable	Develop Develop
7.- Report	Basic report Meeting Clean and order code Update readme Final report Video	Develop Meeting Enhancement Enhancement Enhancement Develop

El tiempo que nos ha llevado esta fase ha sido el restante desde finales de Febrero hasta mediados de Junio (algo más de tres meses), siendo la parte más importante del proyecto, a continuación se detalla el tiempo requerido en cada uno de los apartados (milestones en *Github*) en los que hemos dividido el proyecto:

- Instalación y configuración

En este apartado se han incluido pequeñas tareas como pueden ser: instalar el entorno, crear diagrama de clases, crear algunos ficheros básicos del proyecto, etc. En esta fase se instalaron todos los requisitos técnicos, librerías y dependencias.

El tiempo medio empleado en esta fase fue de una semana, como se puede ver en el primer A.3 commit de esta fase.

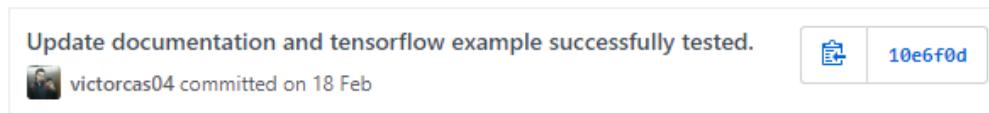


Figura A.3: Commit con los recursos instalados en el proyecto.

- Captura y manipulación de imágenes

En este apartado se hacen diversos commits de tareas en las que teníamos que implementar una forma de abrir y guardar imágenes, así como completar algunos tutoriales on-line sobre el funcionamiento de aplicaciones de *machine-learning*.

El tiempo medio empleado en esta fase fue de casi un mes, desde finales de Febrero [A.4](#) hasta mediados de Marzo [A.5](#).

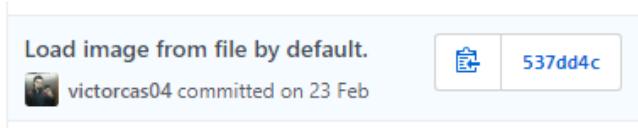


Figura A.4: Commit con las primeras imágenes de prueba.

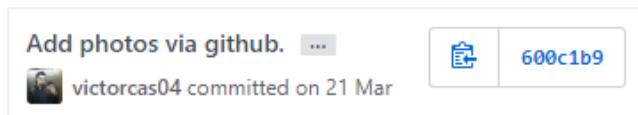


Figura A.5: Commit con el que finaliza la fase de abrir, capturar, guardar y modificar una imagen.

Además, en esta fase se encontró un problema derivado de una de las librerías (*OpenCV*), el cual hacía que dicha librería no fuera compatible con nuestra versión de python (*Python 3.6* en un principio), por lo que se cambió la configuración del proyecto para trabajar con *Python 2.7* [A.6](#).



Figura A.6: Commit que solucionaba el problema de recursos no compatibles con las nuevas versiones de python.

- Reconocimiento de imágenes: localización del rostro y extracción de características

Antes de empezar con esta fase, al final de la fase anterior se dejaron los tutoriales de TensorFlow aparte, ya que suponían una inversión enorme de tiempo, y se optó por una opción más sencilla: *OpenCV* A.6.

En este apartado se desarrolla la captura por pantalla de un frame A.7. A continuación, se implementa la primera parte del reconocimiento facial, donde se localizan los rostros de la gente en dicha imagen A.8, previamente normalizada A.9, además de extraer las características de cada uno de esos rostros (en nuestro caso hemos limitado el número de caras que puede reconocer en cada imagen a uno para evitar falsos positivos).

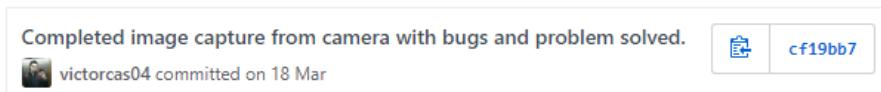


Figura A.7: Commit con la implementación de la captura de la cámara.

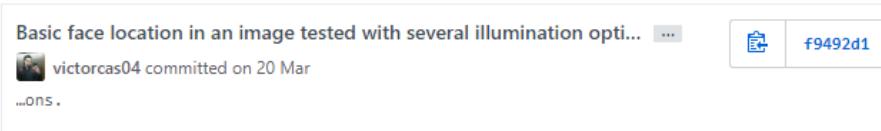


Figura A.8: Commit con la localización del rostro en una imagen implementada.

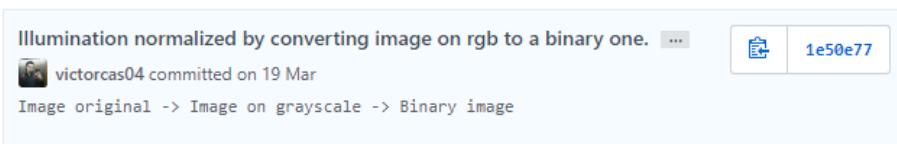


Figura A.9: Commit con la normalización de la iluminación de la imagen.

Además, se implementó una primera versión de reconocimiento facial a partir de la técnica *LBPH* A.10, explicada en la memoria principal, en la sección *3_conceptos_teóricos*.

El tiempo medio empleado en esta fase fue de casi un mes, desde mediados de Marzo hasta mediados de Abril.

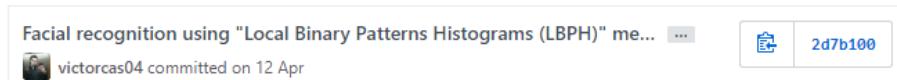


Figura A.10: Reconocimiento facial utilizando el método LBPH.

- Reconocimiento de imágenes: mejorar resultados de predicciones con *computer vision*

En esta parte se mejora el porcentaje de reconocimiento de una persona (aproximadamente, de un 40 50 % a un 60 70 %) A.11, A.12 restringiendo la zona de obtención de características del rostro y cambiando parámetros tanto de iluminación como de la predicción de la red A.13.

El tiempo medio empleado en esta fase fue de un mes, desde mediados de Abril hasta mediados de Mayo.



Figura A.11: Commit con cambios en la implementación de los métodos que permiten el reconocimiento facial (parte I).

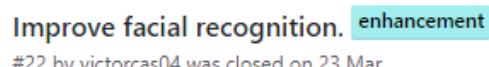


Figura A.12: Commit con cambios en la implementación de los métodos que permiten el reconocimiento facial (parte II).

Try different approaches to train. [enhancement](#)
#38 by victorcas04 was closed 15 days ago

Figura A.13: Commit con cambios en la implementación de los métodos que permiten el reconocimiento facial (parte III).

- Desarrollar una interfaz

Esta es la parte que va a ver el usuario: la interfaz visual, de manera que se han empleado algo más de dos meses, desde principios de Marzo hasta mediados de Mayo.

Una de las primeras tareas es la de ajustar el tamaño de las imágenes de salida para hacer dicha salida más estable a la hora de realizar comparativas o ejecuciones sucesivas A.14, sin mencionar el reconocimiento en real-time, en cuyo caso, de no haber considerado estandarizar el tamaño de las imágenes, cada una ocuparía una parte diferente de la interfaz, y como esta cambia cada 0.1 segundos, habría sido demasiado caótico. También se modifica el grosor del rectángulo que rodea el rostro de manera automática, de manera que sea proporcional al tamaño de la imagen, y se hace una ventana de tamaño variable en función del tamaño de la pantalla en medidas relativas, para que dicha interfaz se adapte el dispositivo en que se ejecute A.15, aunque no sea modificable por el usuario (se han inhabilitado las opciones de minimización y reescalado).

Image size display. [enhancement](#)
#21 by victorcas04 was closed on 24 Apr

Figura A.14: Commit con cambios en la implementación de los métodos que permiten el reconocimiento facial (parte I).

Dynamic window to display images. [enhancement](#)
#18 by victorcas04 was closed on 8 Mar

Figura A.15: Commit con la implementación de ventana de la nueva interfaz.

Se decide crear un pequeño pero intuitivo menú para el usuario. Las funcionalidades descritas en el menú ya estaban funcionales de etapas anteriores, pero se ha decidido incluir dicho menú para facilidad de uso por parte del usuario A.16.



Main menu. **Develop**
#29 by victorcas04 was closed on 2 Apr

Figura A.16: Commit con el menú final de cara al usuario final.

En la ventana donde se muestran los resultados, se ha añadido también en esta etapa una barra de progreso, indicando en tanto por ciento (%) la probabilidad de que sea la persona correcta identificada A.17.



Add compare bar/percentage to interface. **enhancement**
#14 by victorcas04 was closed on 12 Apr

Figura A.17: Commit con la implementación de la barra de progreso.

Como un pequeño extra, se ha incluido algo de información de cada una de las personas en la base de datos (nombre, edad, etc), de manera que si se pulsa el botón de *More information...*, nos creará una ventana emergente (*popup*) con dicha información A.18.



Add info to interface. **enhancement**
#40 by victorcas04 was closed 24 days ago

Figura A.18: Commit con el *popup* y la información extra.

Durante la implementación de esta tarea, nos encontramos con un problema, y es que no podíamos abrir dos ventanas de la misma interfaz al mismo tiempo ya que daba fallos de compatibilidad (por motivos de seguridad, para evitar que unas ventanas creen otras de forma recursiva), pero al final se consiguió solucionar este pequeño contratiempo añadiendo algunos parámetros en la creación de este *popup* para evitar que la ventana principal siguiera ejecutándose A.19.

Single info window. bug
 #41 by victorcas04 was closed 22 days ago

Figura A.19: Commit solucionando el problema de las ventanas *popup*.

Para terminar con el entorno de usuario, se ha creado un pequeño script ejecutable (fichero *.bat*) que ejecuta el proyecto desde la consola de comandos, mostrando alguna información por dicha consola (más por curiosidad que por utilidad, al usuario medio lo que haga internamente el programa no le importa, pero para este proyecto se ha decidido incluirlo para propósitos académicos), aunque la información importante se mostrará en la interfaz explicada anteriormente A.20.

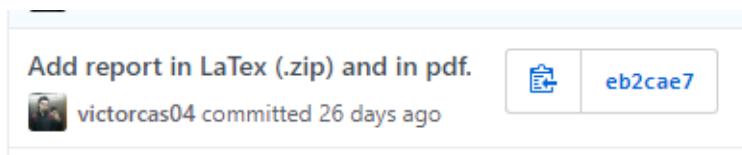
Improve executable. enhancement
 #39 by victorcas04 was closed 8 days ago

Executable. enhancement
 #37 by victorcas04 was closed 16 days ago

Figura A.20: Commit con el ejecutable base y su mejora.

- Memoria y anexos

Esta parte consiste en escribir la memoria con todos los anexos, desarrollando cada punto y dando explicación a algunas preguntas interesantes A.21.



Add report in LaTex (.zip) and in pdf.
 victorcas04 committed 26 days ago

Figura A.21: Commit con la primera versión de la memoria en LaTex.

También se ha actualizado el fichero *Readme.md* que sirve como carta de presentación de nuestro proyecto en *Github* A.22.

Esta tarea se lleva desarrollando desde principios de Mayo, y hasta la fecha (principios de Junio) se sigue desarrollando.

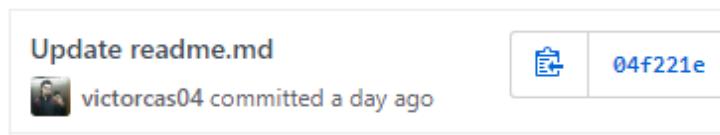


Figura A.22: Commit con una de las actualizaciones del fichero README.md.

■ Revisión y control continuo

Aunque esta parte no es una fase la cual haya que planificar como las demás, si que se han ido teniendo reuniones (semanales o bisemanales) mencionadas en la planificación, además de haber ido llevando un control de errores por cuenta propia, que se iban resolviendo inmediatamente si eran urgentes, o después de realizar las tareas asignadas para esa semana si no afectaban al uso general del programa.

Con el proyecto en *Github*, ha sido sencillo seguir el progreso y desarrollo del mismo, sabiendo en todo momento qué tareas estábamos realizando, cuales nos quedaban por realizar, cuales eran los objetivos generales, etc. Aunque no ha estado exento de problemas de todo tipo, todos ellos se consiguieron solventar. Algunos de los más frecuentes fueron: cómo remover una carpeta entera de GitHub [71], cómo deshacer un *rebase* [8] o un *commit* [53]

- **Histórico de commits**

Se puede ver un gráfico con el histórico de commits en Github en la imagen A.23. Como se puede observar, se ha mantenido un flujo más o menos constante salvo determinados días, correspondientes a épocas sin exámenes o con algo más de tiempo libre en el que se podía avanzar libremente en el proyecto.

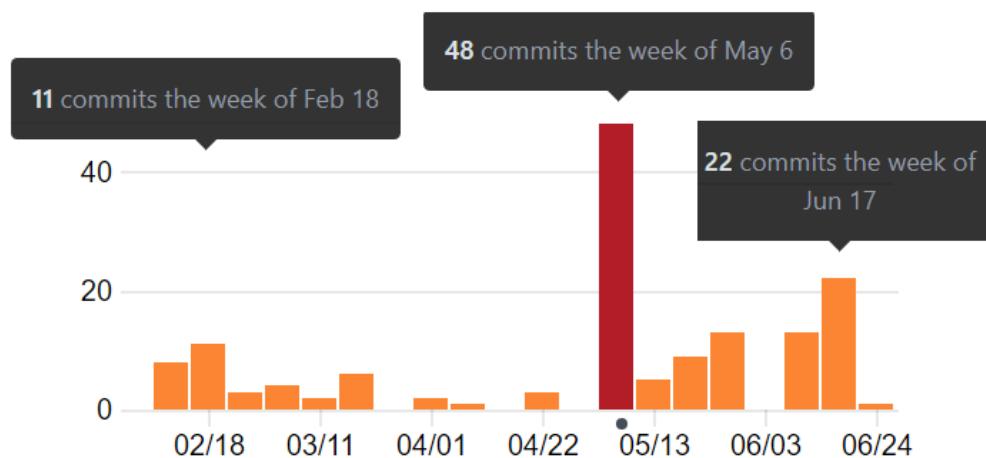


Figura A.23: Grafo con el número de commits por semana.

A.3. Estudio de viabilidad

¿Es viable?

Esta pregunta es un poco ambigua, y vamos a entrar más en detalle en cada una de sus posibles ramificaciones, pero para los impacientes, la respuesta breve es SI en el ámbito tecnológico y DEPENDE en los ámbitos moral y legal.

Viabilidad económica

Es viable tecnológicamente si se tiene un *sistema distribuido* [68] en el cual se tenga un servidor (que sea capaz de realizar procesamiento en paralelo [69]) donde se realicen todas las operaciones de comparación, predicción, muestreo de resultados, etc., y varios clientes, concretamente uno por cada cámara (o zona de cámaras), que se encargarán de obtener las imágenes de la calle y enviárselas al servidor junto con información de qué cámara ha realizado esa captura, en qué zona está, a qué hora, etc. Como podemos observar, ya se tiene una estructura similar hoy en día, de modo que se podría aprovechar ese sistema, ahorrando costes.

Viabilidad legal

En cuanto al apartado legal, es una cuestión complicada, ya que, depende de de países, unos tienen unas normas y leyes de privacidad más o menos estrictas. En el caso de España, ya se controla el acceso a determinados lugares públicos como aeropuertos, de manera que sería extender ese control y mejorarlo.

Viabilidad moral

Una cuestión que mucha gente no está dispuesta a dejar pasar es el riesgo moral: ¿merece la pena la pérdida de privacidad por un poco más de seguridad? Desde luego, es una cuestión muy importante, pero es demasiado compleja como para resolverlo aquí, ya que cada uno tendrá su opinión y tampoco es el objetivo que persigue este proyecto.

Apéndice B

Especificación de Requisitos

B.1. Introducción

Como **requisitos software**, es necesario tener instalado *python* (preferiblemente en su versión 2.7, o un entorno virtual con dicha versión ya que es el entorno en el que se ha desarrollado) y diferentes librerías: *cv2* (*machine-learning* orientado a imágenes), *tKinter* (interfaz), *numpy* (utilidad), *Pillow* (operaciones con imágenes). Todas ellas se pueden encontrar en la sección *4_Técnicas_y_herramientas* de la memoria principal). En nuestro caso se ha probado en un entorno controlado con *Windows 10*, no se asegura su funcionamiento para otras distribuciones.

En nuestro entorno de pruebas se ha utilizado tanto una cámara externa (*Logitech WEBCAM C170* con resolución [1024 X 768]) [33] como la cámara integrada del portátil (*Sony Vaio VPCF13A4E* con resolución [640 X 480]). Dicho portátil tiene las siguientes características: 6GB de memoria RAM, tarjeta gráfica *GT425M* y un procesador *Intel Core i5* de 4^a generación. De modo que se tomarán estas características como **requisitos hardware** mínimos.

B.2. Objetivos generales

El TFG consiste en una aplicación que, dadas dos imágenes, una obtenida en el momento de ejecución, ya sea mediante fichero (accediendo a la foto que tengamos almacenada en nuestro equipo), o mediante una captura que realicemos con nuestra cámara, y la otra alojada localmente en una base de datos: sea capaz de distinguir si en ambas imágenes se encuentra la misma persona utilizando técnicas de *machine-learning*.

A la hora de comparar ambas imágenes, si la persona que se pone delante de la cámara no estaba registrada en la base de datos, aparecerá la persona que esté registrada que más se parezca. En caso de que no se supere un umbral de coincidencia con ninguna de las personas registradas, mostrará una imagen por defecto avisando de que no se ha podido obtener ningún resultado satisfactorio.

En cualquier caso, junto con la imagen que se obtenga como resultado, se mostrará una barra de progreso, que indica el porcentaje de acierto que ha obtenido al encontrar dicho resultado. Además, se muestra un botón que nos permite crear otra ventana extra con información de la persona obtenida como resultado (nombre, edad, lugar de nacimiento y profesión).

Los objetivos generales derivados de la explicación anterior, así como los técnicos y los personales se pueden encontrar en la sección *2_Objetivos_del_proyecto* de la memoria principal y son:

- Obtener una imagen, ya sea almacenada o en tiempo real.
- Reconocer un rostro partir de una imagen.
- Comparar dicho rostro con otros almacenados en la base de datos.
- Mostrar el rostro almacenado con mayor coincidencia con el primero en una interfaz sencilla.

B.3. Catalogo de requisitos

El objetivo principal de este trabajo consiste en identificar el rostro de una persona presente en una imagen (ya sea grabada o capturada en tiempo real) comparándolo con una lista de rostros almacenados en una base de datos.

Requisitos funcionales

RF 1.- Gestionar la base de datos

En esta parte se le permite al usuario manipular la base de datos de la siguiente forma.

RF 1.1.- Añadir imágenes

El usuario podrá capturar nuevas imágenes y almacenarlas en la base de datos.

RF 1.2.- Personalizar datos

El usuario podrá ser capaz de personalizar los datos de las nuevas imágenes almacenadas.

RF 2.- Entrenar la red

El usuario podrá elegir entrenar o no la red.

RF 3.- Obtener imagen a identificar

Se obtiene una imagen mediante diferentes medios con la cual continuará el reconocimiento facial.

RF 3.1.- Obtener imagen desde una captura

Se le muestra al usuario una imagen en tiempo real de la cámara, y es él el que elige cuando realizar la captura que servirá para el reconocimiento.

RF 3.2.- Obtener imagen desde un fichero

El usuario puede elegir una imagen desde fichero, el cual se utilizará durante el reconocimiento.

RF 3.3.- Obtener imagen en tiempo real

Al usuario se le muestra en todo momento una imagen en tiempo real, junto con los resultados obtenidos de dicha imagen.

RF 4.- Reconocer caras

En este requisito se localizan e identifican los rostros que aparecen en una imagen.

RF 4.1.- Localizar rostros

Se localizarán automáticamente los rostros dentro de una imagen, continuando o no con la ejecución en función del número de rostros.

RF 4.2.- Identificar rostros

Se reconocen las personas que aparecen en los rostros de las imágenes utilizando sistemas de *computer-vision*.

RF 5.- Mostrar resultados

El usuario podrá observar, mediante una interfaz sencilla, los resultados obtenidos de la ejecución del programa.

Requisitos no funcionales

RNF 1.- Reducir el tiempo entre muestreos

En el reconocimiento en tiempo real, conseguir que el tiempo de muestreo entre dos frames consecutivos sea menor de un segundo para aumentar la eficacia.

RNF 2.- Usar hardware genérico

Conseguir adaptar el software para que funcione con cualquier tipo de hardware, no solamente con el propio del entorno de desarrollo.

RNF 2.1.- Usar cámaras genéricas

La aplicación podrá ejecutarse con todo tipo de cámaras. Sin establecer una resolución específica, se tomará una por defecto [640 X 480].

RNF 2.2.- Usar monitores genéricos

La aplicación podrá ejecutarse con todo tipo de monitores, de manera que se reescala la interfaz de forma automática en función de las dimensiones del mismo, siendo lo recomendado una relación de aspecto de 16:9.

RNF 3.- Facilitar el uso de la aplicación

Con estos requisitos se pretende facilitarle el trabajo al usuario.

RNF 3.1.- Ampliar opciones durante la captura

Mientras tenga que realizar una captura con la cámara, al usuario se le proporcionan varias opciones (consultar apartado *E_Manual_usuario E*) entre las que se incluye pausar la cámara, mostrar información y ayuda o salir sin tomar ninguna imagen.

RNF 3.2.- Mostrar porcentaje de coincidencia

En la interfaz final, junto con ambas imágenes, mostrar que tanto por ciento se parecen ambas según el resultado de la predicción realizada por nuestro sistema de visión artificial.

RNF 3.3.- Mostrar barra de coincidencia

En la interfaz final, junto con ambas imágenes y el tanto por ciento, mostrar una barra de progreso que represente dicho % para facilitarle al usuario su comprensión.

RNF 3.4.- Mostrar más información

En la interfaz final, mostrar información acerca de la persona reconocida en la imagen.

RNF 3.5.- Filtrar archivos

Cuando el usuario elige cargar una imagen desde fichero, tiene la opción de filtrar los resultados según el tipo de archivo con una determinada extensión (.png o .jpg) para facilitar su búsqueda.

B.4. Especificación de requisitos

En esta sección se detallan todos los casos de uso de nuestra aplicación, que coinciden con los requisitos funcionales mostrados en el apartado anterior. Además se adjunta un diagrama de casos de uso [B.1](#) correspondiente de la aplicación, el cual justifica estos requisitos.

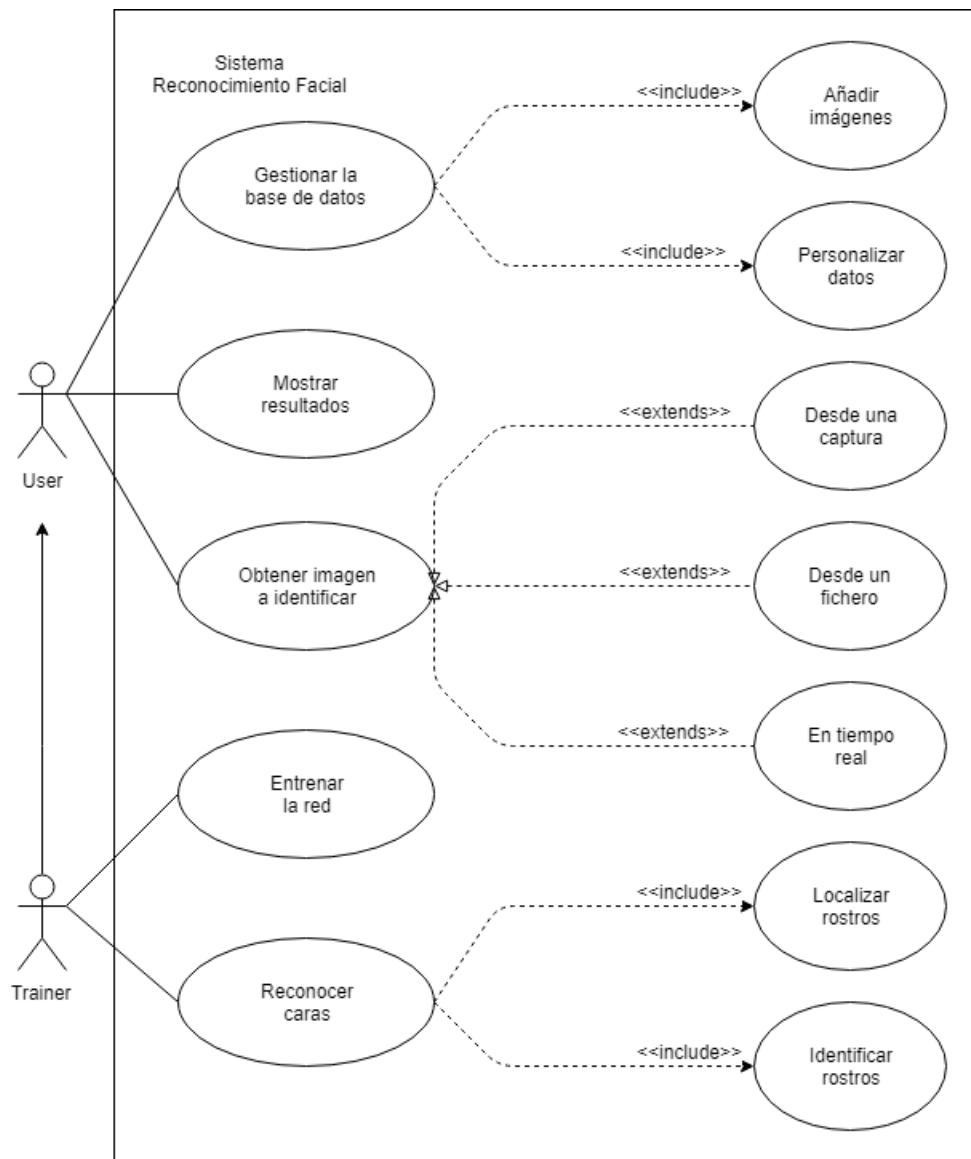


Figura B.1: Diagrama con los principales casos de uso de nuestra aplicación.

<i>CU 1.- Gestionar la base de datos</i>								
Descripción	El usuario puede elegir añadir una imagen nueva a la base de datos. En caso de que lo haga, se le pedirá que introduzca ciertos datos. El proceso se puede repetir tantas veces como el usuario desee. A continuación se pasará al <i>CU 2.- Entrenar la red</i> .							
Requisitos asociados	<i>RF 1.1.- Añadir imágenes</i> <i>RF 1.2.- Personalizar datos</i>							
Actores	User							
Precondiciones	<ul style="list-style-type: none"> - La cámara debe estar encendida. - Debe haber exactamente un rostro en la imagen. 							
Acciones	<table border="1" style="width: 100%; border-collapse: collapse;"> <thead> <tr> <th style="text-align: left;">Paso</th><th style="text-align: left;">Acción</th></tr> </thead> <tbody> <tr> <td style="text-align: center;">1</td><td>El usuario debe colocarse con el rostro mirando a la cámara.</td></tr> <tr> <td style="text-align: center;">2</td><td>Se muestran varias opciones disponibles al usuario, éste puede tomar una captura o no.</td></tr> </tbody> </table>		Paso	Acción	1	El usuario debe colocarse con el rostro mirando a la cámara.	2	Se muestran varias opciones disponibles al usuario, éste puede tomar una captura o no.
Paso	Acción							
1	El usuario debe colocarse con el rostro mirando a la cámara.							
2	Se muestran varias opciones disponibles al usuario, éste puede tomar una captura o no.							
Postcondiciones	<ul style="list-style-type: none"> - Los datos almacenados deben ser consistentes con las imágenes a las que se asocien. 							
Excepciones	<ul style="list-style-type: none"> - Imagen con número de rostros distinto de uno: se muestra mensaje al usuario y se vuelve a intentar el caso de uso hasta que toma una captura correcta o sale sin tomar ninguna. - Nombre de imagen ya existente en la base de datos: se le pregunta al usuario si desea sobre-escribir el fichero con ese nombre. - Carpeta con imágenes inexistente: se muestra mensaje al usuario y se crea vacía. 							
Frecuencia	Baja (vamos a tener una base de datos previa)							
Importancia	Baja (es opcional)							

Tabla B.1: *CU 1.- Gestionar la base de datos*

<i>CU 2.- Entrenar la red</i>								
Descripción	El usuario puede elegir entrenar o no la red si no ha modificado la base de datos. En caso contrario, este proceso se ejecutará automáticamente. A continuación se pasará al <i>CU 3.- Obtener imagen a identificar</i> .							
Actores	Trainer							
Precondiciones	<ul style="list-style-type: none"> - Debe haber como mínimo dos imágenes almacenadas con sus respectivos datos. - Cada una de las imágenes almacenadas debe tener exactamente un rostro. 							
Acciones	<table border="1" style="width: 100%; border-collapse: collapse;"> <thead> <tr> <th style="text-align: left;">Paso</th><th style="text-align: left;">Acción</th></tr> </thead> <tbody> <tr> <td style="text-align: center;">1</td><td style="text-align: left;">Se obtienen las características de cada rostro.</td></tr> <tr> <td style="text-align: center;">2</td><td style="text-align: left;">Se almacenan todas las características en el fichero <i>.yml</i>.</td></tr> </tbody> </table>		Paso	Acción	1	Se obtienen las características de cada rostro.	2	Se almacenan todas las características en el fichero <i>.yml</i> .
Paso	Acción							
1	Se obtienen las características de cada rostro.							
2	Se almacenan todas las características en el fichero <i>.yml</i> .							
Postcondiciones	<ul style="list-style-type: none"> - Los resultados de entrenar la red se guardan en un fichero <i>.yml</i> para futuras ejecuciones. - Se obtienen nuevas imágenes recortando el rostro de las originales y se guardan. 							
Excepciones	<ul style="list-style-type: none"> - Imagen con número de rostros distinto de uno: se muestra mensaje al usuario y se ignora dicha imagen. - Carpeta con imágenes inexistente: se muestra mensaje al usuario y se crea vacía. - No se ha podido entrenar la red: se muestra mensaje al usuario. 							
Frecuencia	Media (cada vez que cambie la base de datos)							
Importancia	Alta (no es esencial para la ejecución del programa, aunque es uno de los requisitos funcionales)							

Tabla B.2: *CU 2.- Entrenar la red*

<i>CU 3.- Obtener imagen a identificar</i>	
Descripción	El usuario escoge el método mediante el cual se obtendrá la imagen que se utilizará en el siguiente <i>CU 4.- Reconocer caras.</i>
Requisitos asociados	<i>RF 3.1.- Obtener imagen desde una captura</i> <i>RF 3.2.- Obtener imagen desde un fichero</i> <i>RF 3.3.- Obtener imagen en tiempo real</i>
Actores	User
Precondiciones	- La red debe estar entrenada.
Acciones	Tratadas en los <i>CU 3.1, CU 3.2</i> y <i>CU 3.3.</i>
Postcondiciones	- Se obtiene una imagen a partir de cualquiera de los tres métodos (cámara, fichero, tiempo-real). - La imagen obtenida debe tener un solo rostro.
Excepciones	Tratadas en los <i>CU 3.1, CU 3.2</i> y <i>CU 3.3.</i>
Frecuencia	Alta (es uno de los principales requisitos funcionales de la aplicación)
Importancia	Alta (es una parte fundamental)

Tabla B.3: *CU 3.- Obtener imagen a identificar*

<i>CU 3.1.- Obtener imagen desde una captura</i>								
Descripción	El usuario obtiene una imagen a partir de una captura de la que se obtiene de la cámara mediante una previsualización. Con esta imagen se pasará al <i>CU 4.- Reconocer caras</i> .							
Requisitos asociados	<i>RF 3.- Obtener imagen a identificar</i>							
Actores	User							
Precondiciones	<ul style="list-style-type: none"> - La cámara debe estar encendida. - Debe haber exactamente un rostro en la imagen. 							
Acciones	<table border="1"> <thead> <tr> <th>Paso</th><th>Acción</th></tr> </thead> <tbody> <tr> <td>1</td><td>El usuario debe colocarse con el rostro mirando a la cámara.</td></tr> <tr> <td>2</td><td>Se muestran varias opciones disponibles al usuario, éste puede tomar captura o no.</td></tr> </tbody> </table>		Paso	Acción	1	El usuario debe colocarse con el rostro mirando a la cámara.	2	Se muestran varias opciones disponibles al usuario, éste puede tomar captura o no.
Paso	Acción							
1	El usuario debe colocarse con el rostro mirando a la cámara.							
2	Se muestran varias opciones disponibles al usuario, éste puede tomar captura o no.							
Excepciones	<ul style="list-style-type: none"> - Imagen con número de rostros distinto de uno: se muestra mensaje al usuario y se vuelve a intentar el caso de uso hasta que toma una captura correcta o sale sin tomar ninguna. - Salir sin tomar imagen: si el usuario no quiere tomar una captura, se cargará una imagen por defecto. 							
Frecuencia	Media (es una alternativa a uno de los principales requisitos funcionales de la aplicación)							
Importancia	Media (no es fundamental ya que hay otras opciones, pero sigue siendo una funcionalidad importante)							

Tabla B.4: *CU 3.1.- Obtener imagen desde una captura*

<i>CU 3.2.- Obtener imagen desde un fichero</i>								
Descripción	El usuario obtiene una imagen a partir de un fichero. Con esta imagen se pasará al <i>CU 4.- Reconocer caras</i>							
Requisitos asociados	<i>RF 3.- Obtener imagen a identificar</i>							
Actores	User							
Acciones	<table border="1"> <thead> <tr> <th>Paso</th><th>Acción</th></tr> </thead> <tbody> <tr> <td>1</td><td>El usuario navega hasta el fichero donde se encuentre el fichero con la imagen.</td></tr> <tr> <td>2</td><td>El usuario puede filtrar el tipo de archivo que desee buscar (limitado a <i>.jpg</i> y <i>.png</i>).</td></tr> </tbody> </table>		Paso	Acción	1	El usuario navega hasta el fichero donde se encuentre el fichero con la imagen.	2	El usuario puede filtrar el tipo de archivo que desee buscar (limitado a <i>.jpg</i> y <i>.png</i>).
Paso	Acción							
1	El usuario navega hasta el fichero donde se encuentre el fichero con la imagen.							
2	El usuario puede filtrar el tipo de archivo que desee buscar (limitado a <i>.jpg</i> y <i>.png</i>).							
Postcondiciones	<ul style="list-style-type: none"> - El fichero debe ser de un formato reconocible (<i>.jpg</i> o <i>.png</i>). 							
Excepciones	<ul style="list-style-type: none"> - Imagen con número de rostros distinto de uno: se muestra mensaje al usuario y se carga una imagen por defecto. - Fichero con extensión desconocida: si el fichero no tiene extensión <i>.jpg</i> ni <i>.png</i>, se mostrará un mensaje al usuario y se cargará una imagen por defecto. 							
Frecuencia	Media (es una alternativa a uno de los principales requisitos funcionales de la aplicación)							
Importancia	Media (no es fundamental ya que hay otras opciones, pero sigue siendo una funcionalidad importante)							

Tabla B.5: *CU 3.2.- Obtener imagen desde un fichero*

<i>CU 3.3.- Obtener imagen en tiempo real</i>								
Descripción	El usuario obtiene una imagen en tiempo real a la vez que se realiza el <i>CU 4.- Reconocer caras</i> .							
Requisitos asociados	<i>RF 3.- Obtener imagen a identificar</i>							
Actores	User							
Precondiciones	<ul style="list-style-type: none"> - La cámara debe estar encendida. - Debe haber exactamente un rostro en las imágenes. 							
Acciones	<table border="1" style="width: 100%; border-collapse: collapse;"> <thead> <tr> <th style="text-align: left;">Paso</th><th style="text-align: left;">Acción</th></tr> </thead> <tbody> <tr> <td style="text-align: center;">1</td><td>El usuario debe colocarse con el rostro mirando hacia la cámara.</td></tr> <tr> <td style="text-align: center;">2</td><td>El usuario espera y observa los resultados.</td></tr> </tbody> </table>		Paso	Acción	1	El usuario debe colocarse con el rostro mirando hacia la cámara.	2	El usuario espera y observa los resultados.
Paso	Acción							
1	El usuario debe colocarse con el rostro mirando hacia la cámara.							
2	El usuario espera y observa los resultados.							
Excepciones	<ul style="list-style-type: none"> - Imagen con número de rostros distinto de uno: se muestra mensaje al usuario. 							
Frecuencia	Alta (es uno de los principales requisitos funcionales)							
Importancia	Alta (es fundamental)							

Tabla B.6: *CU 3.3.- Obtener imagen en tiempo real*

<i>CU 4.- Reconocer caras</i>						
Descripción	Se localizan e identifican automáticamente los rostros que aparecen en la imagen. A continuación se pasará al <i>CU 5.- Mostrar resultados</i> .					
Requisitos asociados	<i>RF 4.1.- Localizar rostros</i> <i>RF 4.2.- Identificar rostros</i>					
Actores	Trainer					
Precondiciones	<ul style="list-style-type: none"> - La red debe estar entrenada. - Debe haber exactamente un rostro en la imagen. 					
Acciones	<table border="1" style="width: 100%; border-collapse: collapse;"> <thead> <tr> <th style="text-align: left;">Paso</th><th style="text-align: left;">Acción</th></tr> </thead> <tbody> <tr> <td style="text-align: center;">1</td><td>Se utiliza un algoritmo predictivo para determinar el resultado del entrenamiento.</td></tr> </tbody> </table>	Paso	Acción	1	Se utiliza un algoritmo predictivo para determinar el resultado del entrenamiento.	
Paso	Acción					
1	Se utiliza un algoritmo predictivo para determinar el resultado del entrenamiento.					
Postcondiciones	<ul style="list-style-type: none"> - Se obtiene una imagen resultado. - La imagen resultado debe existir en la base de datos. - Se obtiene un porcentaje de coincidencia positivo. - Se obtiene la información relacionada con la persona que aparece en la imagen resultado. 					
Excepciones	<ul style="list-style-type: none"> - Imagen con número de rostros distinto de uno: se muestra mensaje al usuario y se carga una imagen por defecto. - No existe la imagen resultado: se muestra mensaje al usuario y se carga imagen por defecto. - No existe información sobre la persona: se carga información por defecto. - La red no está entrenada: se muestra mensaje al usuario. - Carpeta con imágenes inexistente: se muestra mensaje al usuario y se crea vacía. 					
Frecuencia	Alta (es uno de los principales requisitos funcionales)					
Importancia	Alta (es fundamental)					

Tabla B.7: *CU 4.- Reconocer caras*

<i>CU 5.- Mostrar resultados</i>		
Descripción	El usuario puede observar los resultados obtenidos del <i>CU 4.- Reconocer caras</i> mediante una interfaz sencilla.	
Actores	User	
Acciones	Paso	Acción
	1	El usuario observa el resultado mediante la barra de coincidencia y el porcentaje.
	2	El usuario observa la imagen original y la imagen del resultado final a la vez.
	3	Pulsar el botón <i>More information...</i> para obtener más información acerca de la persona reconocida en el <i>CU 4.- Reconocer caras</i> .
Frecuencia	Alta (es uno de los principales requisitos funcionales)	
Importancia	Alta (es fundamental)	

Tabla B.8: *CU 5.- Mostrar resultados*

Apéndice C

Especificación de diseño

C.1. Introducción

En este anexo se define (desde el punto de vista de la ingeniería del software) cómo se han resuelto los objetivos y especificaciones de los apartados anteriores. Se definen también los datos que va a manejar la aplicación, su arquitectura, el diseño de sus interfaces, sus detalles procedimentales, etc. Para ello, nos vamos a apoyar en diferentes **diagramas UML** para explicar los módulos y sus clases, así como las relaciones que hay entre ellas. Aunque cabe destacar que muchos de los métodos que utilizamos no están contenidos dentro de clases propiamente dichas, sino más bien dentro de fichero que vamos invocando según se van necesitando.

C.2. Diseño de datos

En cuanto a los datos utilizados, los únicos con los que se trabaja son las imágenes y la información asociada almacenada en el fichero de texto, así como el fichero *.yml* creado tras entrenar la red, el resto de datos, variables y parámetros no se almacenan ya que no se necesitan para nada. Las entidades que podemos encontrar en nuestra aplicación son (hay que tener en cuenta que, dado que python no permite la creación de métodos privados como tal, se han marcado todos como si fueran públicos, aunque algunos sean usados sólo en su entorno):

- Entidad principal **Main** C.1. Corresponde al fichero *Main.py* y contiene el flujo de programa principal de nuestra aplicación. Es desde donde empieza y termina nuestro programa.

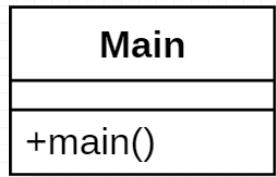


Figura C.1: Estructura del fichero principal: *Main.py*.

- Entidad de utilidad **Util** C.2. Corresponde al fichero *Util.py* y contiene funciones adicionales y de utilidad para nuestro programa como transformar imágenes en diferentes formatos.

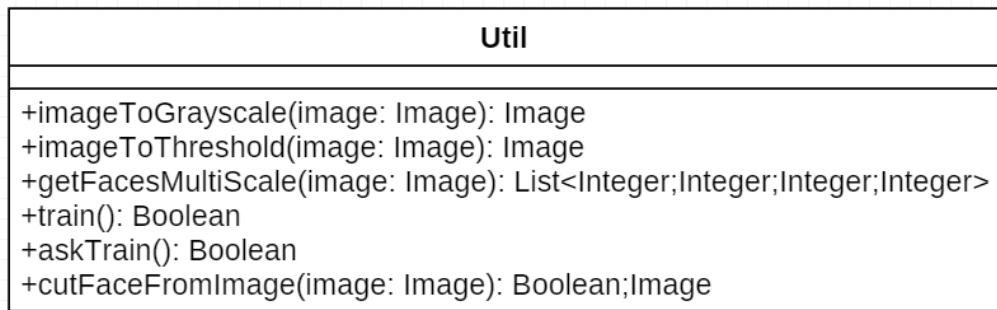


Figura C.2: Estructura del fichero de utilidad: *Util.py*.

- Entidad encargada de manejar los ficheros **Files** C.3. Corresponde al fichero *Files.py* y contiene métodos que interaccionan con ficheros como cargar o guardar imágenes, o gestionar la información de la base de datos. Además, contiene todas las rutas y nombres de ficheros importantes en formato *String*.

Files
+multiple_string_paths_names: String
+getFileName(file: String, folder: String): String
+loadImage(nameImage: String): Image
+getLoadedXml(): CascadeClassifier
+getReco(): FaceRecognizer
+loadInfo(nameImage: String): Dict<String>
+filesOnDir(path: String): List<String>
+saveImage(image: Image, path: String)
+writeInfoNewImage(stringInfo: String)
+overwriteFile(nameImage: String): Boolean
+doWhenNewImage(image: Image): Boolean
+checkFolderExists(path: String)

Figura C.3: Estructura del fichero encargado de interaccionar con ficheros: *Files.py*.

- Entidad encargada de manejar la cámara **Camera** C.4. Corresponde al fichero *Camera.py* y contiene los métodos que inicializan y obtienen capturas de las imágenes de la cámara.

Camera
+resolutionWidth: Integer
+resolutionHeight: Integer
+cameraIndex: Integer
+initializeCamera(): VideoCapture
+captureImage(captureToCompare: Boolean): Boolean;Image

Figura C.4: Estructura del fichero encargado de interaccionar con la cámara: *Camera.py*.

- Entidad encargada del entrenamiento de la red **Trainer** C.5. Corresponde al fichero *Trainer.py* y contiene los métodos que guardan las imágenes en su formato correcto (solo el rostro) y entrena nuestra red.

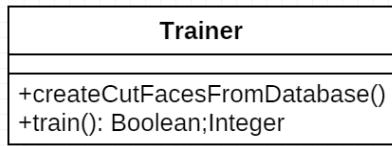


Figura C.5: Estructura del fichero encargado de entrenar la red: *Trainer.py*.

- Entidad que se encarga de la interfaz en modo texto ***TextInterface*** **C.6**. Corresponde al fichero *TextInterface.py* y contiene dos subclases con enumeraciones de *MESSAGES* y *ERRORS*, que se podrán llamar como si de excepciones se tratara para mostrar mensajes varios por pantalla cuando la situación lo requiera. Además contiene otros métodos que interaccionan con el usuario en modo texto como son mostrar menús o preguntarle información (cuando se introduce una imagen nueva).

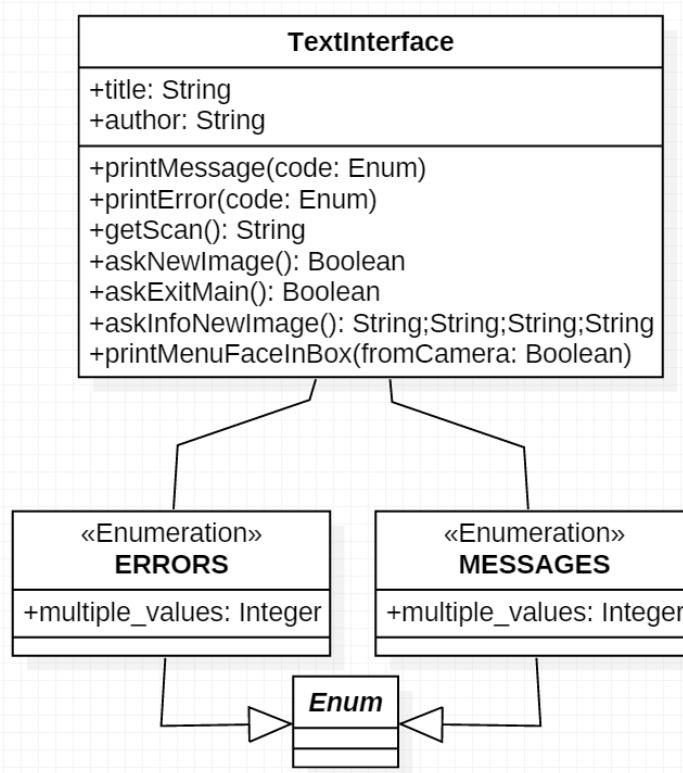


Figura C.6: Estructura del fichero que se encarga de la interfaz en modo texto: *TextInterface.py*.

- Entidad que se encarga de la interfaz en modo gráfico ***GUIClass*** C.7. Corresponde al fichero *GUI.py* y a la clase *GUIClass*, y contiene una subclase con una enumeración de *COLORS*, que se encargan de establecer el color de la barra de progreso cuando cambia el porcentaje de coincidencia.

Además contiene todos los métodos necesarios para crear la ventana principal de la interfaz, así como el *pop-up* con la información de la persona reconocida, y otros métodos para establecer una nueva imagen. Es el único fichero en el cual se ha necesitado crear una clase, ya que necesitábamos que la interfaz fuera única, de manera que para utilizar el patrón *Singleton* necesitábamos una clase.

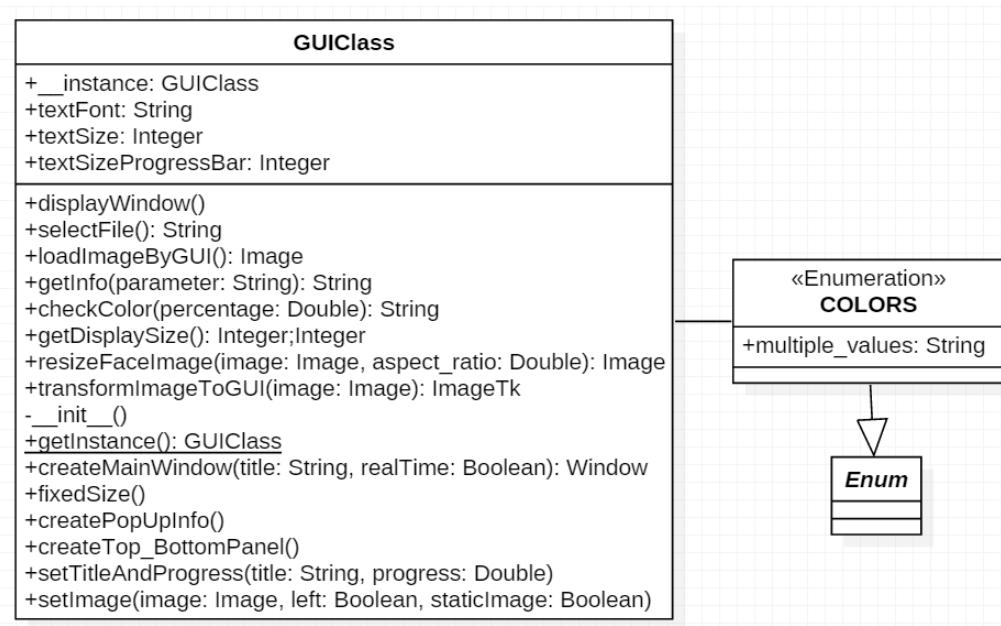


Figura C.7: Estructura de la clase que se encarga de la interfaz visual: *GUIClass*.

- Entidad encargada de realizar el reconocimiento propiamente dicho ***CompareImages*** C.8. Corresponde al fichero *CompareImages.py* y contiene los métodos que comparan la imagen que queremos reconocer con las de la base de datos.

CompareImages
+getLoadedYml(): Recognizer
+compareSingle(image: Image): Integer;Double
+compare(image: Image, path: String): Image;Double;String

Figura C.8: Estructura del fichero que se encarga de predecir el resultado: *CompareImages.py*.

- Entidad que se encarga de reconocer imágenes en tiempo real **RecognizerRealTime** C.9. Corresponde al fichero *RecognizerRealTime.py* y contiene un solo método que utiliza otras entidades (*Camera*, *GUI*, *CompareImages*) para realizar el reconocimiento en un solo método. Además tiene un atributo que define la velocidad a la que se actualiza la interfaz.

RecognizerRealTime
+segBetweenFrames: Double
+compareInRealTime()

Figura C.9: Estructura del fichero que reconoce las imágenes en tiempo real: *RecognizerRealTime.py*.

C.3. Diseño procedimental

En este apartado se recogen los detalles más relevantes respecto a la ejecución de nuestro programa en términos de diseño.

En el siguiente diagrama C.10 se puede observar la **gestión de la base de datos** B.1.

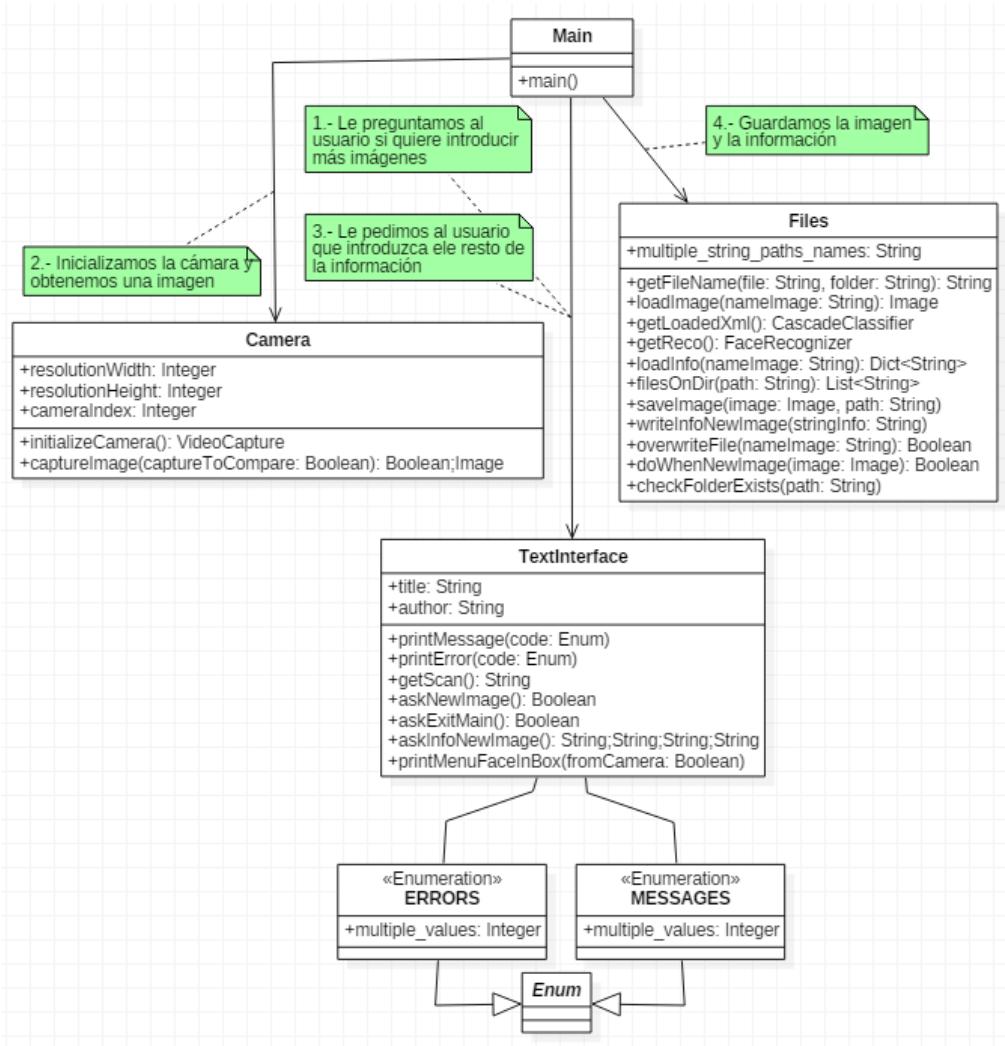


Figura C.10: Relación entre entidades que representan la gestión de la base de datos.

En el siguiente diagrama C.11 se puede observar el proceso que se sigue para **entrena la red** B.2.

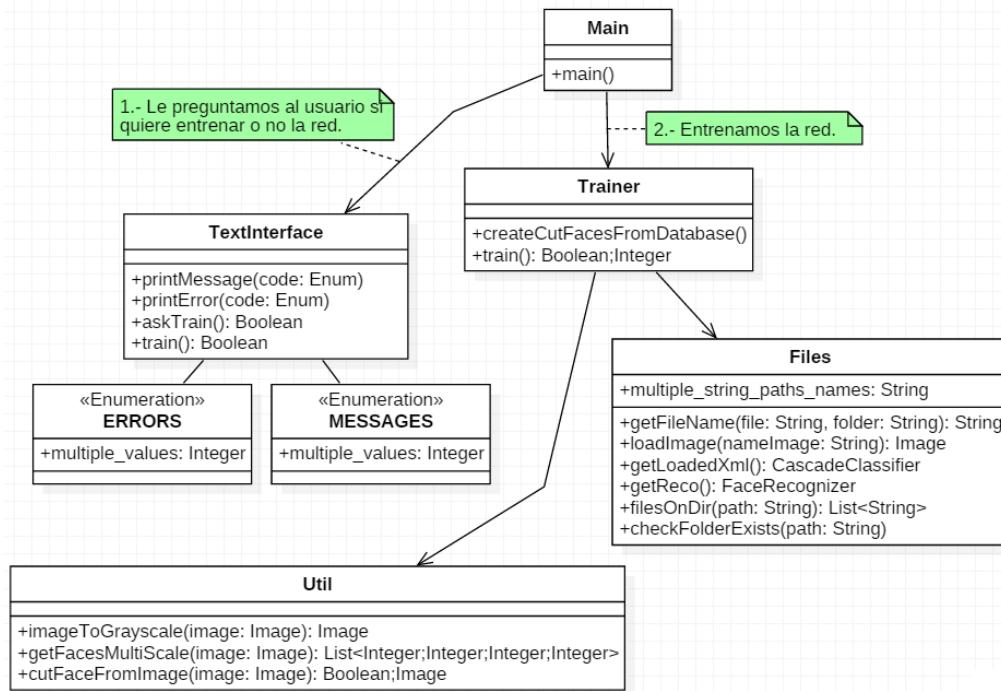


Figura C.11: Relación entre entidades que representan el entrenamiento de la red.

En el siguiente diagrama C.12 se puede observar cómo es el proceso encargado de **obtiene una imagen a identificar B.3**.

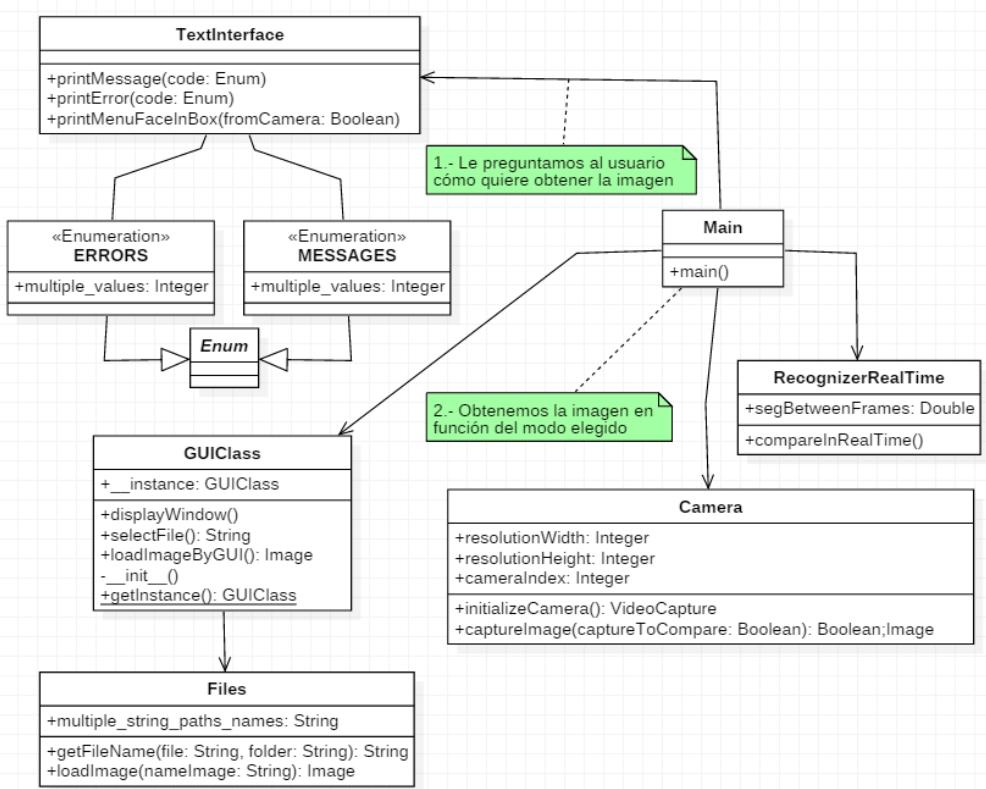


Figura C.12: Relación entre entidades que representan la obtención de una imagen para reconocerla.

En el siguiente diagrama C.13 se puede observar cómo se **reconocen caras** B.7.

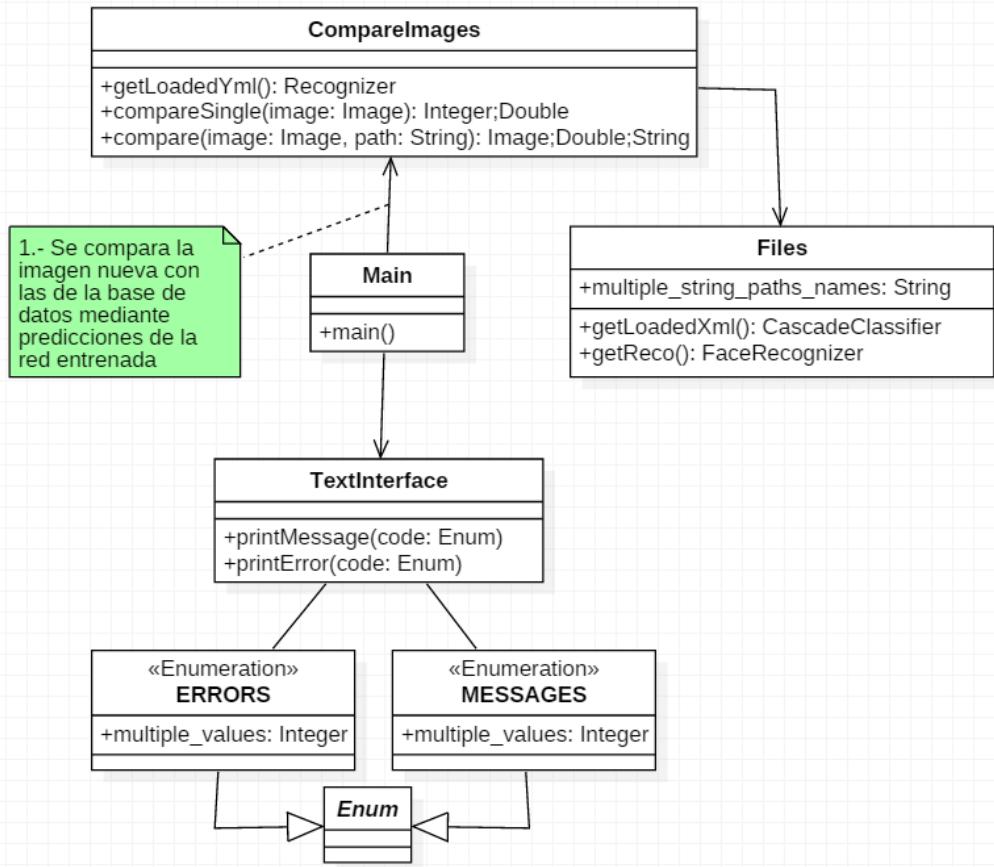


Figura C.13: Relación entre entidades que representan el reconocimiento de caras.

En el siguiente diagrama C.14 se puede observar el proceso mediante el cual se muestran los resultados B.8.

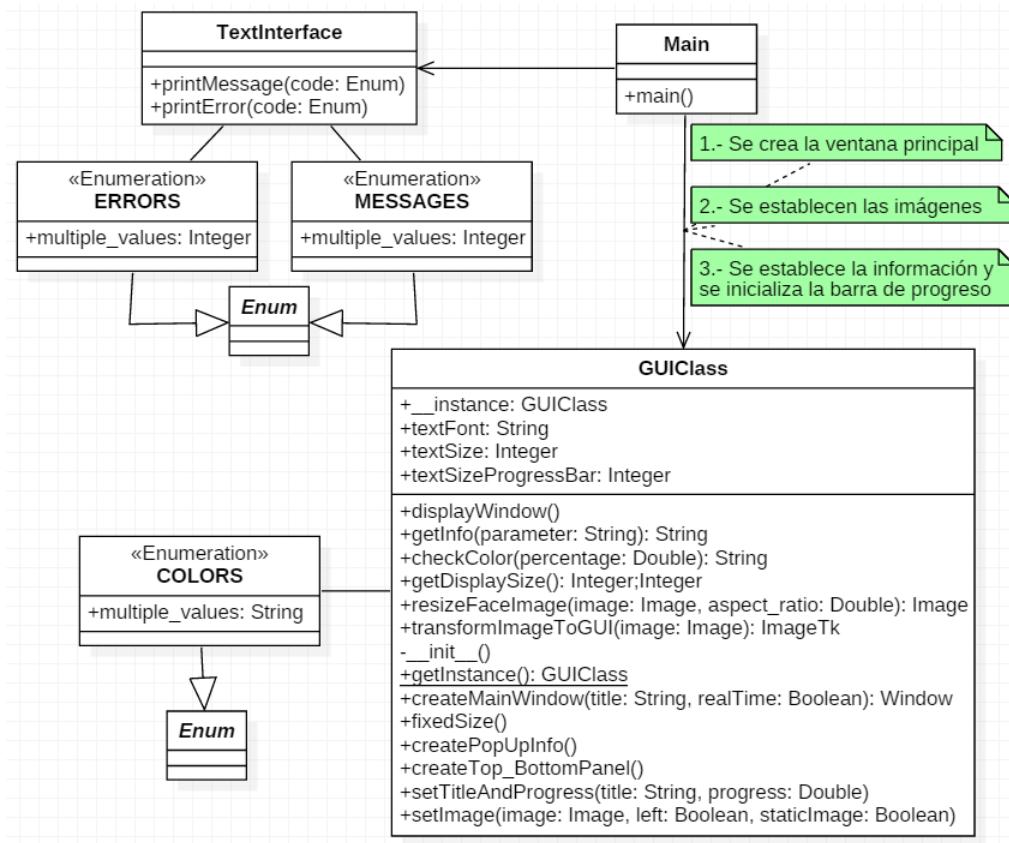


Figura C.14: Relación entre entidades que representan el muestreo de resultados.

NOTA: *TextInterface* aparece en todas ellas aunque no tenga funcionalidad aparente ya que es la encargada de mostrar tanto mensajes de ayuda al usuario como mensajes de error, y como los errores hay que controlarlos en todas las demás entidades, aparece en todos los diagramas.

C.4. Diseño arquitectónico

En esta sección vamos a definir las principales relaciones entre entidades, aunque de manera más general que en la sección anterior. Para simplificar la vista se han eliminado ciertas relaciones que se pueden observar en los diagramas específicos de cada funcionalidad, así como las subclases *Enum*, para evitar tener excesivos elementos en el diagrama. También se han suprimido métodos y atributos de cada una de ellas para más claridad [C.15](#).

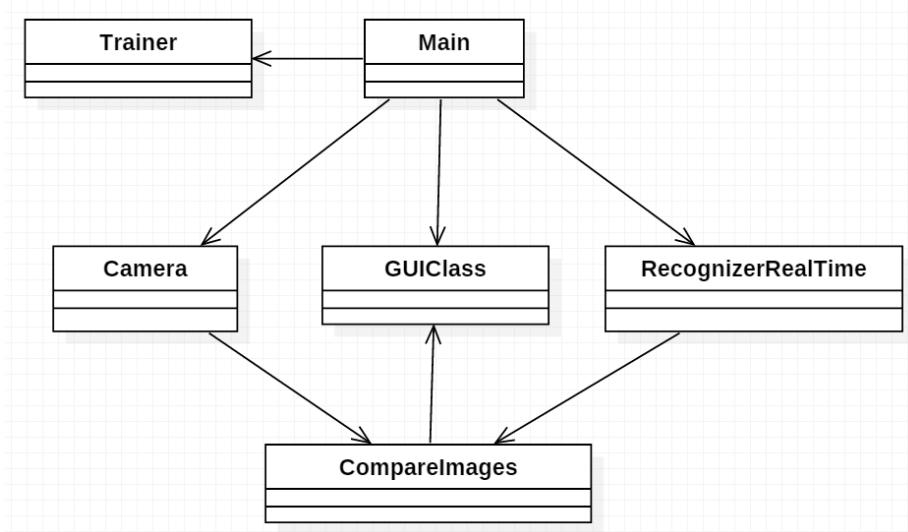


Figura C.15: Arquitectura final simplificada de nuestra aplicación.

El sistema de clases y privacidad en python no es tan claro como en otros lenguajes, y a pesar de ser una implementación experimental de una tecnología con la que no habíamos trabajado nunca (tanto los sistemas de *machine-learning* como el sistema completo de la interfaz con *tKinter* eran nuevos para mí), se han aplicado una serie de patrones (al principio no de manera intencionada, sino que fueron surgiendo a medida que surgían nuevas necesidades) para intentar mejorar todo lo posible la calidad de nuestro código, para que sea más sencillo trabajar con él el día de mañana.

Aunque al principio se utilizaron más patrones, posteriormente se hizo refactorización de muchos de los métodos y ficheros, de modo que algunos de estos patrones ya no eran necesarios.

Al final, uno de los patrones más útiles es el de **Singleton** C.16, especialmente cuando queremos instanciar una nueva interfaz gráfica, que nos ayuda a asegurar que no se creen y sobreesciban varias interfaces (ya que la utilizamos de varias maneras diferentes, no nos interesa crear una interfaz para cada uno de estos usos, sino una común que vaya cambiando en función del uso que le queramos dar) C.17.

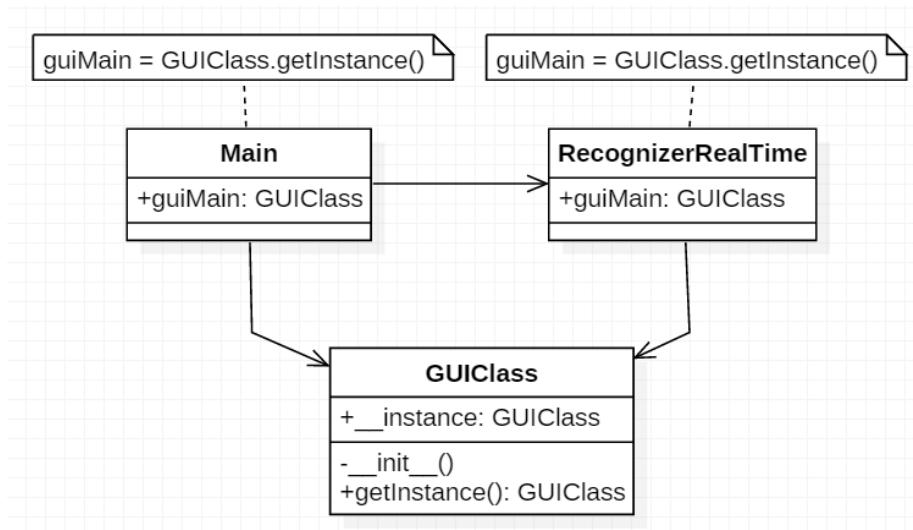


Figura C.16: Ejemplo de patrón *Singleton* utilizado en la clase *GUIClass*.

```

def __init__(self):
    if GUIClass.__instance != None:
        raise Exception("This is a Singleton class. "
                        "Access it through getInstance()")
    else:
        GUIClass.__instance = self

    @staticmethod
    def getInstance():
        if GUIClass.__instance == None:
            GUIClass()
        return GUIClass.__instance

```

Figura C.17: Implementación del patrón *Singleton*.

Otro de los patrones que se tenía pensado aplicar era el de **Abstract Factory** [C.18](#) para instanciar los diferentes tipos de interfaz (gráfica y en modo texto). Al final esto no se ha llevado a cabo ya que se siguen mostrando mensajes en modo texto incluso cuando estamos usando la interfaz gráfica, además de que necesitábamos una interfaz de todas maneras para mostrar las imágenes y los resultados, así que el modo texto no era autosuficiente, por lo que se ha optado por una opción intermedia: mostrar la información relevante en la interfaz y el resto de información en modo texto.

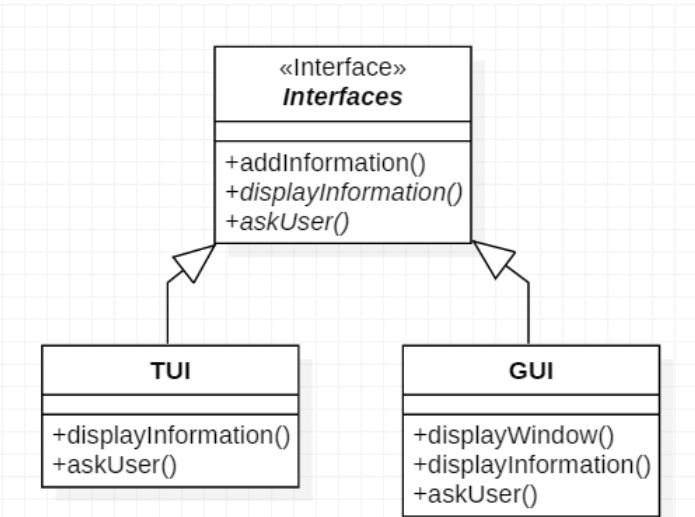


Figura C.18: Ejemplo de patrón *Abstract Factory* que se tenía pensado utilizar para crear las diferentes interfaces.

Por último, tenemos los patrones **Adapter** y **Builder**.

El primero de ellos se utilizó en un comienzo como solución a un problema: las imágenes que obteníamos y modificábamos al principio no se podían mostrar en la interfaz, de manera que teníamos que adaptarlas al tipo de dato adecuado C.19. Pero era una solución poco práctica ya que se utilizaba en contadas ocasiones y no era excesivamente útil, de modo que se pasó esa adaptación a un método dentro de *GUIClass* llamado *transformImageToGUI* C.20.

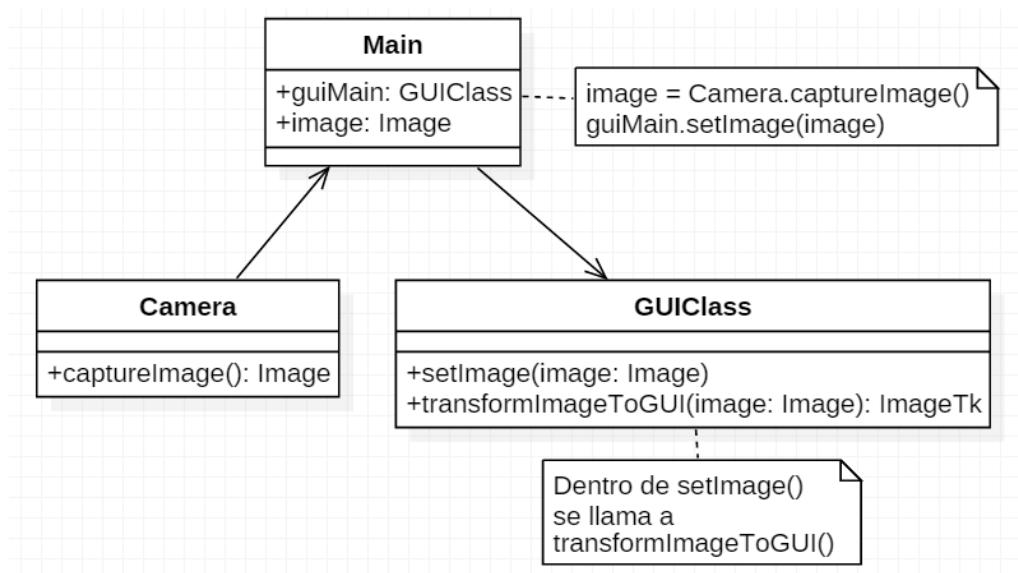


Figura C.19: Ejemplo de patrón *Adapter* que se utilizaba al principio para transformar las imágenes al tipo adecuado.

```

def transformImageToGUI(self, img):
    imgRecolor = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)
    imgFromArray = Image.fromarray(imgRecolor)
    return ImageTk.PhotoImage(imgFromArray)

```

Figura C.20: Implementación final sustituyendo al patrón *Adapter*.

En cuanto al patrón constructor C.21, se optó por la solución del apartado anterior: realizar la implementación de un par de métodos en la clase *GUIClass* C.22 (*setTitleAndProgress()* y *setImage()*) ya que era el único sitio en el que se modificaba dicha interfaz, de modo que no tenía sentido hacer una clase externa exclusiva para implementar este patrón.

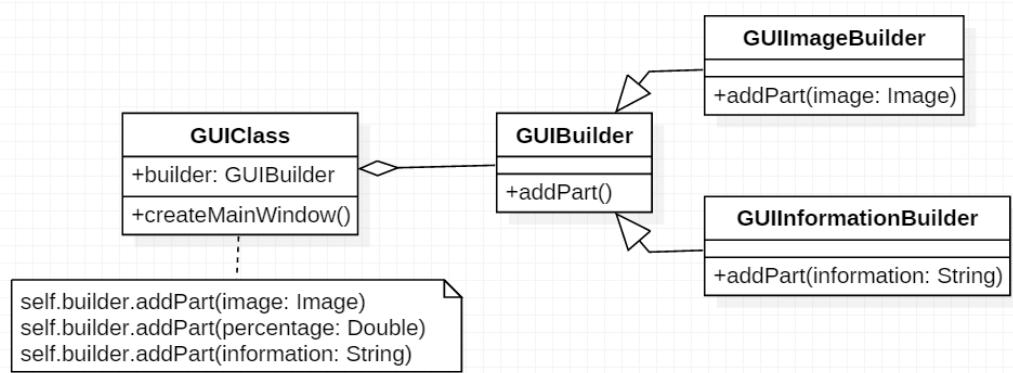


Figura C.21: Ejemplo de patrón *Builder* que se utilizaba al principio para ir modificando la interfaz.

```

def setImage(self, image=None, left=True, staticImage=True):
    image = image if (image is not None) else files.loadImage()
    if staticImage:
        image = self.resizeFaceImage(image)
        img = self.transformImageToGUI(image)
        panel = self.myLabelC if left else self.myLabelD
        panel.image = img
        panel.configure(image=img)
    if not left:
        panel.configure(text=files.loadInfo(self.namePhoto)[ "name" ])
  
```

Figura C.22: Implementación final sustituyendo al patrón *Builder*.

Apéndice D

Documentación técnica de programación

D.1. Introducción

Este proyecto se ejecuta en una máquina destino distinta del entorno de desarrollo, por lo que la configuración utilizada en los dos entornos tiene que estar sincronizada. En el apartado *Manual del programador D.3* se entrará en más detalle de cómo instalar dicho entorno, aunque lo primero que vamos a necesitar va a ser *Windows 10*, *python 2.7*, y diversas librerías de python.

APÉNDICE D. DOCUMENTACIÓN TÉCNICA DE PROGRAMACIÓN

D.2. Estructura de directorios

En la carpeta `src` se encuentra el código de nuestro programa. Si se desea modificar algún comportamiento concreto o estudiar cómo funciona alguna parte específica, es en esta carpeta en la que deberemos mirar. Está dividido en diferentes ficheros que vemos a continuación:

- Fichero ***Camera.py***

Esta clase tiene métodos relacionados con la cámara, ya sea obtener una imagen o inicializar la propia cámara. Dentro de esta clase se especifica si la imagen que estamos obteniendo la vamos a usar para una cosa (compararla) o para otra (guardarla en la base de datos).

- Fichero ***CompareImages.py***

Realiza la comparación entre la imagen que acabamos de obtener (ya sea mediante captura con la cámara o desde fichero), con todas las de la base de datos. Se puede ver el funcionamiento exacto de esta función en el anexo *D_Manual_programador*. Se ha reutilizado esta clase tanto para el reconocimiento de una imagen 'estática' (un solo frame) como de la imagen en tiempo real (vídeo o varios frames).

- Fichero ***Files.py***

En esta clase están todos los métodos relacionados con ficheros, desde obtener una simple ruta hasta cargar todos los datos almacenados en él (`loadInfo()`).

- Fichero ***GUI.py***

Contiene todo lo relacionado con la interfaz gráfica, desde el panel central, donde se muestra toda la información y las imágenes obtenidas, hasta el menú que nos permite seleccionar una imagen desde fichero en vez de desde la cámara. Dicha interfaz ha sido realizada con *tKinter*. Se puede ver cada uno de los métodos y su funcionalidad en el anexo *D_Manual_programador*.

- Fichero **Main.py**

Contiene el método principal de la ejecución del programa. Es donde se le pregunta al usuario si quiere entrenar la red, seleccionar una imagen desde fichero, desde la cámara o realizar el reconocimiento en tiempo real, llama a todos los demás métodos de otras clases para comparar las imágenes, generar la interfaz y mostrar resultados. Se pueden ver la estructura en el anexo *D_Manual_programador*, y el flujo de programa en el anexo *C_Diseno*.

- Fichero **RecognizerRealTime.py**

Esta clase contiene el flujo de programa encargado sólo de la captura y comparación de las imágenes en tiempo real, así como su respectiva interfaz.

- Fichero **TextInterface.py**

Esta clase contiene todo lo relacionado con la interfaz en modo texto (por consola de comandos) de nuestro programa, así como un par de clases internas que se encargan de almacenar variables de tipo *Enum*, las cuales vamos a usar para mostrar los mensajes por pantalla, como si de los mensajes de excepciones se tratara.

- Fichero **Trainer.py**

Esta clase se encarga exclusivamente de entrenar la red a partir de unas imágenes fuente, que se encuentran en la carpeta *faces-Dataset*. La funcionalidad completa se puede observar en el anexo *D_Manual_programador*.

- Fichero **Util.py**

Contiene algunos de los métodos de utilidad que no se encuentran ya en otros ficheros (*Files.py*), como pasar una imagen de *RGB* a escala de grises o recortar la parte de la imagen correspondiente al rostro. Se pueden ver todos los métodos en el anexo *D_Manual_programador*.

APÉNDICE D. DOCUMENTACIÓN TÉCNICA DE PROGRAMACIÓN

En la carpeta ***sources*** se puede encontrar material que afecta a la ejecución del programa, pero que no es código ejecutable como tal. En dicha carpeta encontramos:

- Carpeta ***dataset***

En esta carpeta podemos encontrar las imágenes originales que tenemos almacenadas, y hace las veces de base de datos (ya que no se dispone de una base de datos en línea, tenemos que utilizar un directorio local).

NOTA: se pueden añadir imágenes manualmente a este directorio, pero hay que asegurarse de añadir una línea con la información asociada al fichero *info.txt* que explicaremos a continuación antes de ejecutar de nuevo el programa. Por ello se recomienda añadir imágenes con la opción habilitada dentro de la aplicación.

Como se puede observar, son imágenes de cuerpo entero la mayoría de ellas (aunque con una calidad, tamaño y características diferentes, ver apartado *7_Conclusiones_Líneas_de_trabajo_futuras_*: *Posibilidades de mejora del proyecto: Base de datos* de la memoria principal), por lo tanto no están ajustadas a la posición de la cara de la persona, por lo que necesitamos crear (interna y automáticamente al entrenar la red) la siguiente carpeta:

- Carpeta ***facesDataset***

Contiene las imágenes anteriores, pero exclusivamente con la cara que haya podido encontrar en la imagen, recortando dicha imagen para eliminar elementos ajenos al rostro que puedan dificultar el reconocimiento. Esta carpeta se actualiza automáticamente cuando entrenamos la red.

- Carpeta ***xml***

Contiene dos ficheros *.xml* que son los encargados de encontrar las partes identificables de un rostro en la imagen. Dichos ficheros se proporcionan gratuitamente con fines educativos en un repositorio [44].

Ahora mismo está en uso exclusivamente el fichero *haarcascade_frontalface_default.xml*, que es el encargado de extraer las características de un rostro de frente, pero se podría cambiar el fichero utilizado dentro del código (en el fichero *Files.py*) para que utilizara el otro fichero *haarcascade_profileface.xml* para reconocer personas de perfil.

- Carpeta ***recognizer***

Contiene ficheros que utilizamos durante la ejecución del programa para visualizar datos sobre la persona reconocida o cargar la imagen correcta de la base de datos a partir de un diccionario.

El fichero *dictionary_ID_labels.txt* se crea automáticamente tras entrenar la red, y contiene el diccionario que relaciona una ID automática a cada imagen de la que ha conseguido obtener un rostro. Se podría evitar usar este fichero si se utilizara directamente el diccionario en el programa, pero eso supondría tener que entrenar la red todas las ejecuciones, y si no se han añadido imágenes nuevas es tiempo y recursos perdidos, de modo que se ha optado por crear un fichero para que el diccionario persista a cada ejecución.

El fichero *info.txt* se ha creado sólo para mostrar información de la persona que se haya reconocido durante la ejecución. Para mostrar esta información consultar el anexo *E_Manual_usuario*. Esta información se puede actualizar a mano actualmente, aunque conviene no hacerlo, ya que es el propio programa el encargado de actualizarla en caso de añadir nuevas imágenes.

El fichero *trainedData.yml* es el que se crea automáticamente al entrenar la red. Por dentro está dividido en tantas secciones como imágenes tuviéramos en nuestra base de datos, y cada una de ellas tiene las características de los histogramas extraídos con la técnica *LBPH* (explicada en la sección *3_Conceptos_teóricos* de la memoria principal).

- Fichero ***default.png***

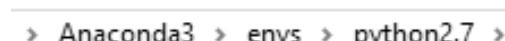
Esta imagen es la que carga el programa por defecto cuando no se ha conseguido superar el umbral de reconocimiento de un rostro en una imagen. Normalmente se conseguirá superar dicho umbral, aunque el reconocimiento falle, pero en caso de que no se hayan podido encontrar rostros, no se haya entrenado la red, o se le pase un fichero que no sea una imagen, el programa cargará esta imagen por defecto.

APÉNDICE D. DOCUMENTACIÓN TÉCNICA DE PROGRAMACIÓN

D.3. Manual del programador

Para que la aplicación funcione correctamente, es necesario que lo haga en el mismo entorno en el que fue desarrollada, para ello se utiliza un sistema operativo *Windows 10* (en otras distribuciones de *Windows* no se asegura su funcionamiento, aunque probablemente funcione, al contrario que con distribuciones de *Linux*, ya que los delimitadores que utilizamos en nuestras rutas son los propios de *Windows*). Si se desea utilizar en otro sistema operativo, deberemos cambiar el *String* que define el delimitador de las rutas (en nuestro caso ”\” por ”/”) en el fichero *Files.py*.

Debemos tener instalado también *Python* (preferiblemente en su versión 2.7, o un entorno virtual con dicha versión ya que es el entorno en el que se ha desarrollado, en nuestro caso se utilizó un entorno virtual de *Anaconda* [D.1](#)). Si se desea adaptar el código, se puede utilizando las librerías *__future__* y *six*.



► Anaconda3 ► envs ► python2.7 ►

Figura D.1: Entorno virtual de *Anaconda* utilizado para la instalación de *python 2.7*.

También es necesario instalar diferentes librerías, las principales son la de *OpenCV*: *cv* (*machine-learning* orientado a imágenes) y *tKinter* (necesario para la interfaz interfaz). Aunque hay otras librerías que no son el núcleo de la aplicación pero aún así necesitan ser instaladas: *numpy* (utilidad, viene por defecto cuando instalamos *python*), *Pillow* (operaciones con imágenes).

Como editor se puede usar el que más se adapte a los gustos de cada uno, aunque en nuestro caso hemos utilizado *Pycharm*.

Todas estas herramientas y librerías se pueden encontrar en la sección *4_Técnicas_y_herramientas* de la memoria principal. En nuestro caso se ha probado en un entorno controlado con y no se asegura su funcionamiento para otras distribuciones.

Algunas dudas o problemas que nos pueden surgir a nosotros como desarrolladores pueden ser las siguientes, adjuntando varias referencias sobre posibles soluciones.

- Algunos problemas que pueden surgir relacionados de la instalación de módulos y paquetes en python.
 - "How do I install pip on Windows?" [46].
 - "Installing packages with pip" [22].
 - "Installing modules with python" [20].
 - "Using wheels to install python dependences" [27].
 - Se puede consultar las siguientes páginas para solucionar alguno de los problemas que pueden surgir de la instalación de *OpenCV*: [14], [28], [59], [65].
- Como utilizamos una versión de python un poco desfasada, puede que tengamos problemas con versiones más modernas, de modo que se ha intentado minimizar al máximo las diferencias entre ambas.
 - Solve compatibility problems between python 2.7 and 3.6 [49].
 - ERROR: "problema a la hora de usar input()/raw_input() con las diferentes versiones de python" [29].
- Como desarrolladores, nos hemos encontrado con algunos de los siguientes problemas generales, con sus respectivas soluciones, además de otros enlaces de interés.
 - Clases y paquetes en python [23].
 - ERROR: "missing 1 required positional argument" (Problema relacionado con el uso de clases y objetos en python [35]).
 - Algunos problema con las llamadas a funciones desde el *main* [24] y desde otras clases [32].
 - Clases abstractas e interfaces en python [73], [31].
 - Tratamiento de excepciones en python [47].
 - Operadores ternarios [30].
 - Operaciones con listas [15], [5].
 - Contador de tiempo en python [19].
 - *Singleton* simple en python [48].

APÉNDICE D. DOCUMENTACIÓN TÉCNICA DE PROGRAMACIÓN

- Operaciones con ficheros [61], [10], [18], [64], [72].
- Enumeraciones en python [21].
- Artículo [7] bastante interesante sobre cómo trabaja python con los redondeos y como solucionar posibles problemas que surgen de ello.
- Otros problemas más concretos de este proyecto que se han ido resolviendo han sido los siguientes.
 - Cargar, modificar y guardar imágenes (con librería *OpenCV*) [1].
 - Abrir imágenes desde fichero [41].
 - Activar cámara [54].
 - Cómo funciona realmente el método *predict()* [39] de *OpenCV*.
 - Documentación [42] sobre *FaceRecognition* de *OpenCV*.
 - ERROR: "problema a la hora de convertir las imágenes de unos tipos a otros" [9], [34].
 - Pausar ejecución para ver resultados [56], [36].
 - ERROR: cámara muestra imagen en negro-ajustar parámetros [66].
 - Cambiar tamaño de las imágenes obtenidas [12], [3].
 - Recortar imágenes para guardar en la base de datos [40].
 - Imagen en b/n utilizando funciones de *Threshold* (para normalizar iluminación y elementos innecesarios) [43].
 - Capturar imágenes con la cámara [38], [13], [26].
 - Resolución de pantalla con python (en nuestro caso con las librerías de *tKinter* y *wx*) [37].
 - Redimensionar automáticamente las ventanas de la interfaz en función de la resolución de nuestra pantalla [63].
 - Mostrar múltiples imágenes en una sola ventana [50] (con la librería de *OpenCV*), [60] (con la librería de *matplotlib*).
 - Redimensionar imágenes de salida (con la librería de *OpenCV*) [17].
 - Mostrar imagen capturada con la cámara en sentido correcto [4].
 - Sencillo tutorial para aprender a usar *tKinter* [11].
 - Juntando los resultados de *OpenCV* con la interfaz de *tKinter* [55].
 - Personalizando la interfaz con *tKinter* [2].
 - ERROR: "botón para mostrar información adicional no responde" [52].
 - ERROR: "abrir dos ventanas con *tKinter*" [25], [57], [51], [62].
 - Barra de progreso (comparación) [16] en *tKinter*.

D.4. COMPILACIÓN, INSTALACIÓN Y EJECUCIÓN DEL PROYECTO⁴⁵

D.4. Compilación, instalación y ejecución del proyecto

El proyecto al estar desarrollado en *python* no hace falta compilarlo, con tener descargado el proyecto de *github* (croquetamente las carpetas *src* y *sources*), y tener instaladas las librerías y recursos especificadas en el apartado anterior es suficiente.

Para ejecutar el proyecto debemos ejecutar el fichero *Main.py* mediante la consola de comandos utilizando *python Main.py*.

IMPORTANTE: debemos estar con la consola de comandos abierta dentro de la carpeta *src*.

Si no queremos utilizar comandos, simplemente nos descargamos el fichero *run.bat* que se encuentra en la carpeta principal del proyecto y damos doble click sobre él.

IMPORTANTE: el fichero debe estar en la misma carpeta que *src* y *sources*.

A la hora de la ejecución, pueden surgir diferentes **errores**, pero todos ellos están tratados y se le mostrará un **mensaje al usuario** avisando del error. En ningún caso se detiene la aplicación hasta que el usuario pulsa el botón de salir, simplemente se restringen ciertas áreas del programa que el usuario no puede utilizar e función del error (por ejemplo, si la red no está entrenada se le avisa al usuario y se le impide realizar el reconocimiento hasta que la entrena).

NOTA: Cuando se cierra la ventana de la interfaz de tiempo real al ejecutar la aplicación como usuario (mediante *run.bat* o *python Main.py*), aparece el siguiente mensaje:

```
invalid command name "143620856show_frame" while executing  
"143620856show_frame" ("after" script).
```

Al ejecutarlo en un editor como *PyCharm* no sucede esto. Se ha seguido la pista pero no parece haber forma de solucionarlo [45], ya que es un problema que afecta a la forma que tiene de tratar *tKinter* los eventos *after*, y es que, al cerrar la ventana principal, hay uno (el ligado al método *show_frame*), que no se cierra correctamente. De todas formas es un mensaje de aviso y no de error, además de que el funcionamiento de nuestra aplicación no se ve afectado, por lo que no se le da mayor importancia.

APÉNDICE D. DOCUMENTACIÓN TÉCNICA DE PROGRAMACIÓN

D.5. Pruebas del sistema

Consumo de recursos

Como se puede observar [D.2](#), se ha conseguido reducir bastante el consumo de recursos, siendo el más acusado el consumo de CPU por parte de *tKinter*, al realizar el reconocimiento en tiempo real, ya que la interfaz se actualiza constantemente. En las pruebas realizadas nunca ha superado el 35 % de consumo de CPU, aún así, sigue siendo un consumo bastante elevado en comparación con los recursos utilizados cuando se entrena la red (especialmente porque al entrenar la red esos recursos se ocupan durante menos tiempo, en el reconocimiento se ocupan hasta que el usuario decide salir).

Nombre	39% CPU	58% Memoria	13% Disco	0% Red	1% GPU
> python.exe (32 bits)	31,0%	27,1 MB	0 MB/s	0 Mbps	0%
> Procesador de comandos de Windows (3)	0%	9,6 MB	0 MB/s	0 Mbps	0%

Figura D.2: Recursos utilizados por nuestra aplicación cuando se está realizando el reconocimiento en tiempo real al ejecutarla desde el fichero ejecutable en el sistema *run.bat*.

Si se decide ejecutar la aplicación con el editor [D.3](#), el consumo de memoria aumenta bastante, ya que *pycharm* utiliza otras rutinas internas, tiene más referencias y herramientas trabajando en segundo plano, sin embargo, el consumo de CPU sigue manteniéndose estable por debajo del 35 %.

Nombre	49% CPU	61% Memoria	8% Disco	0% Red	2% GPU
> Python (2)	31,9%	48,5 MB	0 MB/s	0 Mbps	0%
> PyCharm (5)	7,2%	518,9 MB	0,1 MB/s	0 Mbps	0%

Figura D.3: Recursos utilizados por nuestra aplicación cuando se está realizando el reconocimiento en tiempo real al ejecutarla desde el editor *pycharm*.

Fiabilidad

Al ser una aplicación tan dependiente de factores externos, la fiabilidad varía mucho de unos entornos a otros, por lo que, para obtener unos resultados más adecuados, es conveniente usarlo en un entorno controlado y con factores externos limitados (por ejemplo, en una sala sin ventanas, con iluminación artificial y con la cámara y las personas siempre en la misma posición). Como se mencionaba en la sección *7_Conclusiones_Líneas_de_trabajo_futuras: Conclusiones* de la memoria principal, los elementos más influyentes son:

1. La iluminación

Con un mismo rostro con las mismas características (expresión facial, sin otros elementos como gafas o barba), en dos ambientes con diferente iluminación se obtenían resultados diferentes, llegando a verse diferencias de un 20 % aproximadamente.

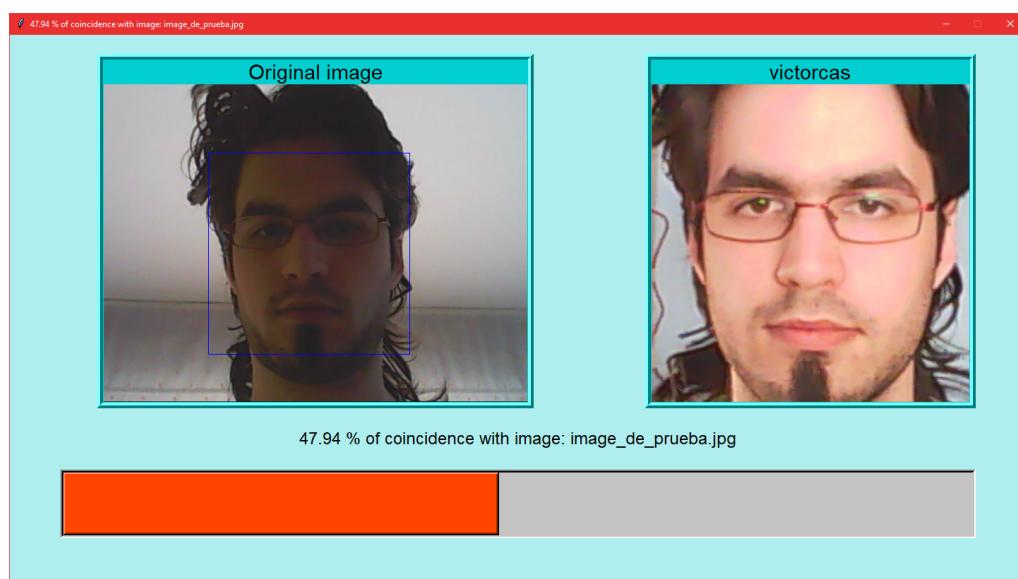


Figura D.4: Comparación entre una imagen guardada en la base de datos, y otra desde la cámara en tiempo real (rasgo distintivo: muy mala iluminación - 48 %).

APÉNDICE D. DOCUMENTACIÓN TÉCNICA DE PROGRAMACIÓN

2. Características del rostro

En este apartado destacan la importancia de los ojos sobre el resto de elementos, llegando a no identificar ningún rostro en la imagen si nos tapamos completamente los ojos, sin embargo, si nos tapamos la boca y la nariz, sigue detectando los ojos como elemento relevante, de manera que seguirá realizando el reconocimiento.

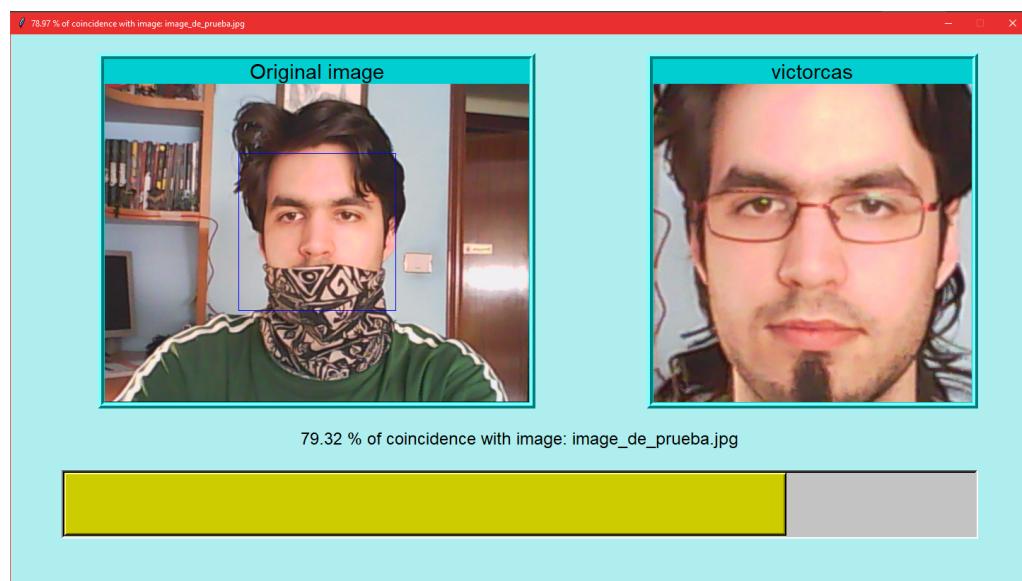


Figura D.5: Comparación entre una imagen guardada en la base de datos, y otra desde la cámara en tiempo real (rasgo distintivo: solo visibles los ojos - 79 %).

3. Posición del rostro

En función de la posición del rostro también varía el porcentaje de coincidencia obtenido, de manera que si nos colocamos de perfil, llega a variar hasta un 10-15 % de coincidencia con la misma persona. En caso de colocarnos completamente de lateral, puede llegar a perderse la detección de rostro, de manera que tomará la imagen como que no hay nadie en ella, de modo que no realizará el reconocimiento.



Figura D.6: Comparación entre una imagen guardada en la base de datos, y otra desde la cámara en tiempo real (rasgo distintivo: foto de casi perfil - 56.5 %).

Tests

No se han desarrollado ya que todos los módulos de nuestra aplicación están muy correlacionados entre ellos, de manera que no se podrían probar la gran mayoría de las funcionalidades de esta manera. La forma de testear nuestra aplicación ha sido mediante prueba y error, probando diferentes entradas, ya sean mediante texto o mediante imágenes, y en diferentes combinaciones entre ellas. Además, se proporcionó una copia a alguno de los compañeros para que la probaran y dieran su opinión, feedback, etc.

Apéndice E

Documentación de usuario

E.1. Introducción

Las tecnologías avanzan continuamente, al igual que hacen las amenazas que acompañan a dichas tecnologías. Por ello, y para mantener una sociedad estable, la seguridad debe ir ligada a estos avances.

Con este proyecto se pretende abordar la temática de la seguridad en lugares públicos (como pueden ser aeropuertos o centros comerciales) desde el punto de vista del reconocimiento facial.

Para ello, se ha diseñado un prototipo de un sistema de identificación de personas a través de un programa de reconocimiento facial, desarrollado en *Python* y utilizando técnicas de *Machine-Learning*. En dicho sistema se parte de una base de datos que contiene imágenes de personas en una lista de busca y captura. De este modo, a partir de una imagen capturada en el lugar de interés, el sistema reconocerá el rostro y lo identificará si se encuentra en la base de datos.

E.2. Requisitos de usuarios

Es necesario tener instalado *python* (preferiblemente en su versión 2.7, o un entorno virtual con dicha versión ya que es el entorno en el que se ha desarrollado) y diferentes librerías imprescindibles como son *cv2* (librería de *OpenCV* orientada a *computer vision*), *tKinter* (necesaria para mostrar la interfaz gráfica), *numpy* (diversas operaciones de utilidad), *Pillow* (para realizar cambios sobre imágenes).

El proyecto ha sido desarrollado y probado en *Windows 10*.

NOTAS

- En el resto de versiones de *Windows* no se asegura su correcto funcionamiento.
- En otros sistemas operativos como *Linux* no funciona debido a las librerías exclusivas de *Windows* y a las rutas utilizadas en los ficheros.

E.3. Instalación

Para ejecutar el programa, simplemente hay que descargarse el proyecto (podemos hacer un *clone* a una carpeta local desde *Github* o simplemente descargarlo como *.zip*) y dar doble click sobre el fichero llamado *run.bat* que encontramos en la carpeta principal.

Esto nos abrirá una consola de comandos donde nos irán saliendo mensajes informativos sobre el progreso del programa (orientados más al uso por parte del desarrollador, aunque también pueden resultar útiles a todo tipo de usuarios), aunque la parte principal creará ventanas para mostrar la interfaz y que la información aparezca de forma más visual y entendible para el usuario medio.

NOTAS

- Es necesario tener los requisitos de usuario instalados [E.2](#) antes de comenzar la ejecución.

E.4. Manual del usuario

Poniéndonos técnicos...

El repositorio de *Github* donde se encuentran los ficheros del TFG de reconocimiento facial basado en *machine-learning* se puede encontrar en el siguiente enlace: <https://github.com/victorcas04/TFG-FacialRecognition>.

El TFG consiste en una aplicación que, dadas dos imágenes, una obtenida en el momento de ejecución, ya sea mediante fichero (accediendo a la foto que tengamos almacenada en nuestro equipo), o mediante una captura que realicemos con nuestra cámara, y la otra alojada localmente en una base de datos: sea capaz de distinguir si en ambas imágenes se encuentra la misma persona utilizando técnicas de *machine-learning*.

A la hora de comparar ambas imágenes se utiliza la extracción de características mediante el algoritmo *LBP-H* y la herramienta *OpenCV*. Si la persona que se intenta identificar no estaba registrada en la base de datos, aparecerá la persona que esté registrada que más se parezca.

En caso de que no se supere un **umbral de coincidencia** con ninguna de las personas registradas, mostrará una imagen por defecto avisando de que no se ha podido obtener ningún resultado satisfactorio. Además, esta misma imagen por defecto se mostrará si no se reconoce exactamente un sólo rostro (se toma esta decisión para ahorrarnos problemas a la hora de extraer características).

En cualquier caso, junto con la imagen que se obtenga como resultado, se mostrará una barra de progreso, que indica el porcentaje de acierto que ha obtenido al encontrar dicho resultado. Además, se muestra un botón que nos permite crear un pequeño *pop-up* con información de la persona obtenida como resultado (nombre, edad, lugar de nacimiento y profesión).

Como desarrollo adicional, se ha implementado un sistema de reconocimiento totalmente en tiempo-real, en el cual se muestra un vídeo de la cámara que esté grabando, el cual se analiza cada determinados frames para obtener las comparaciones automáticamente (en lugar de una comparación por ejecución como en el modo anterior). La interfaz utilizada en este último caso es similar a la anterior, pero eliminando el botón que proporcionaba esa ventana con más información. Esto se ha hecho ya que, al poder cambiar el resultado en poco tiempo, era un elemento contraproducente.

Instrucciones

En esta guía se seguirá el curso del programa durante las principales fases por las que pasa nuestro programa son, indicándole al usuario las opciones que tiene:

1.- Añadir nuevas imágenes a la base de datos

- Nos preguntará si queremos añadir una imagen nueva a la base de datos (ver imagen E.1).

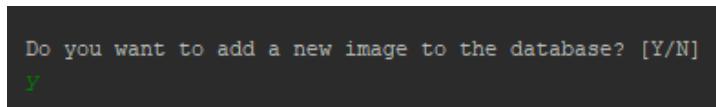


Figura E.1: Programa preguntándonos si queremos introducir una nueva imagen en la base de datos.

- En caso de responder sí [Y], se inicializará la cámara por defecto del equipo (si tenemos una externa conectada, utilizará esa, en caso contrario usará la webcam integrada) E.2.

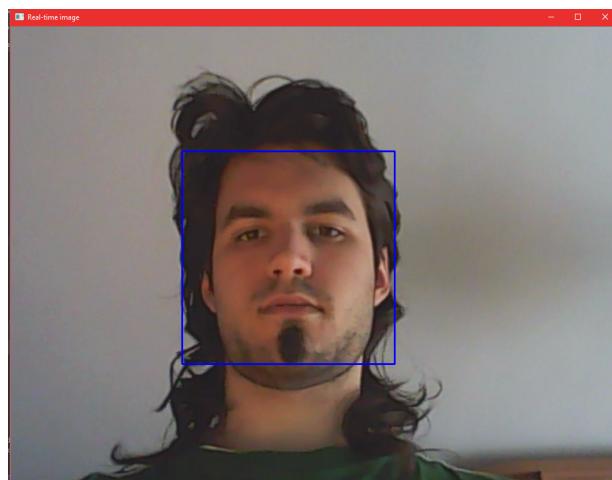


Figura E.2: Imagen que se muestra por pantalla para que elijamos cuando capturarla.

- Se pueden seguir las instrucciones del apartado *Instrucciones adicionales: Menú de utilización general de la cámara* E.4.

- En caso de obtener una imagen (podemos elegir no hacerlo), nos pedirá introducir una serie de datos sobre la persona que deberíamos encontrar en esa imagen: nombre, edad, ciudad de nacimiento y profesión (ver imagen E.4). En caso de no querer llenar alguno de esos campos se tomarán valores por defecto. Además, se pedirá introducir el nombre de la imagen que acabamos de obtener para almacenarla en nuestra base de datos. Estos datos se almacenarán en un fichero local llamado *info.txt*.

- Si ya existía una imagen con el mismo nombre, nos preguntará si queremos sobre-escribirla, como se puede observar en la imagen E.3.

```
WARNING: An image with the same name already exists.
Do you want to overwrite it? [Y/N]
n
```

Figura E.3: Programa preguntándonos si queremos sobre-escribir la imagen.

- IMPORTANTE: Los datos de este fichero se pueden modificar manualmente, aunque conviene no hacerlo ya que, de cambiar el nombre de la imagen por error, podemos encontrarnos con datos inconsistentes y tener errores en la ejecución. En caso de hacer cambios sobre este fichero o sobre las imágenes de la base de datos de forma manual (antes de su ejecución), asegurarse de que quedan datos consistentes.

```
Introduce the name of the new file (to save on database):
If no name is provided, 'new_image_name' will be used.

Introduce the name of the person who appears on that image:
If no name is provided, 'Default Name' will be used.
nombre

Introduce the age of that person:
If no age is provided, 'Default Age' will be used.
edad

Introduce the city name where he/she was born:
If no city name is provided, 'Default City' will be used.
ciudad

Introduce the actual profession of that person:
If no occupation is provided, 'Default Job' will be used.
profesion

Saving image in: ..\sources\dataset\new_image_name.jpg...
```

Figura E.4: Programa preguntándonos los datos relacionados con la imagen.

Como podemos observar [E.5](#), la nueva información se ha añadido a nuestro fichero con toda la información relacionada con las imágenes de nuestra base de datos.

```
11  natalie_dormer, Natalie Dormer, 36, Reading, Actress
12  new_image_name, nombre, edad, ciudad, profesión
13  robert_downey_ir, Robert Downey Jr, 53, New York, Ac
```

Figura E.5: Información añadida al fichero *info.txt*.

2.- Entrenamiento de la red

- A continuación, si hemos guardado una imagen nueva en la base de datos en el apartado *1.- Añadir nuevas imágenes a la base de datos* [E.4](#), pasaremos directamente al punto en que se entrena la red, en caso contrario nos preguntará si queremos entrenarla o no (ver imagen [E.6](#)).

```
Do you want to train the network? [Y/N]
y
```

Figura E.6: Programa preguntándonos si queremos entrenar o no la red.

- En caso de decir que NO, se utilizarán los recursos almacenados del último entrenamiento [E.7](#) (la primera vez que se ejecute el programa, a pesar de tener un fichero por defecto, es probable que tengamos que entrenar la red debido a registros internos de la librería que utilizamos).

```
The file trainedData.yml existing from before will be used.
Loading file ..\sources\xml\haarcascade_frontalface_default.xml...
```

Figura E.7: Programa cargando los recursos almacenados previamente.

- En caso de decir que SI, se crearán las imágenes en el formato adecuado y se entrenará la red [E.8](#), tras lo cual se cargarán los recursos recién creados [E.9](#).

```
Creating face-focused images from original-database images...
Training network...

Loading file ..\sources\dataset\alexandra_daddario.jpg...
Loading file ..\sources\dataset\cas.jpg...
Loading file ..\sources\dataset\cas_abii.jpg...
Loading file ..\sources\dataset\cesar_sala_juntas.jpg...
Loading file ..\sources\dataset\chris_pratt.jpg...
Loading file ..\sources\dataset\dwayne_the_rock_johnson.jpg...
Loading file ..\sources\dataset\elizabeth_olsen.jpg...
Loading file ..\sources\dataset\gaben.jpg...
Loading file ..\sources\dataset\gal_gadot.jpg...
Loading file ..\sources\dataset\john_cena.jpg...
Loading file ..\sources\dataset\katherlyn_winnick.jpg...
Loading file ..\sources\dataset\madds_mikkelsen.jpg...
Loading file ..\sources\dataset\natalie_dormer.jpg...
Loading file ..\sources\dataset\new_image_name.jpg...
Loading file ..\sources\dataset\robert_downey_jr.jpg...
Loading file ..\sources\dataset\shohreh_aghdashloo.jpg...
Loading file ..\sources\dataset\stan_lee.jpg...
Loading file ..\sources\dataset\will_smith.jpg...

Training time with 18 images: 6.33 seconds.
```

Figura E.8: Programa entrenando la red.

```
The file trainedData.yml just created will be used.
Loading file ..\sources\xml\haarcascade_frontalface_default.xml...
```

Figura E.9: Programa cargando los nuevos recursos creados.

- **IMPORTANTE:** si se añaden o eliminan imágenes manualmente, de forma previa a la ejecución del programa a la base de datos (antes del apartado 1.- *Añadir nuevas imágenes a la base de datos* E.4), entrenar la red para evitar problemas sobre identificaciones erróneas.

3.- Carga de recursos e inicialización

- Se cargan los recursos necesarios: *trainerData.yml* (los datos de nuestra red entrenada en el apartado 2.- *Entrenamiento de la red* E.4) y *haarcascade_frontalface_default.xml* (fichero que nos permite identificar un rostro dentro de una imagen a partir de sus características).

- A continuación, se le pregunta al usuario cómo desea obtener la imagen que va a contrastar con la base de datos: desde fichero (apartado 4.1.- *Obtener la imagen desde fichero* E.4), desde la cámara (apartado 4.2.- *Obtener la imagen mediante una captura* E.4) o en tiempo real (apartado 4.3.- *Realizar la identificación en tiempo-real* E.4) E.10.

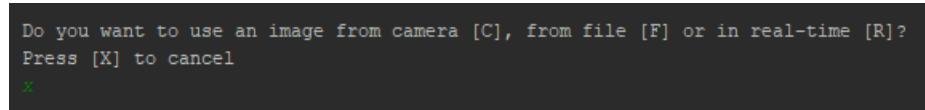


Figura E.10: Programa preguntándonos cómo queremos continuar la ejecución.

4.1.- Obtener la imagen desde fichero

- Si se selecciona esta opción, se abrirá una pequeña interfaz en la que el usuario puede buscar la imagen que quiera en el sistema E.12.

- Una vez que se encuentra la imagen, basta con dar doble click sobre ella o seleccionarla y pulsar aceptar.

- A continuación, y si la imagen es correcta (sólo tiene un rostro en ella y ocupa al menos el 10 % de la imagen), se mostrará en la interfaz explicada en el apartado 6.- *Resultados finales: Interfaz visual* E.4.

NOTAS

- Se puede filtrar el tipo de ficheros que se pueden ver para facilitar la búsqueda (restringido a .png y .jpg) E.11.

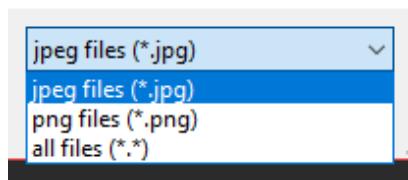


Figura E.11: Parte de la interfaz que nos permite filtrar el tipo de archivos.

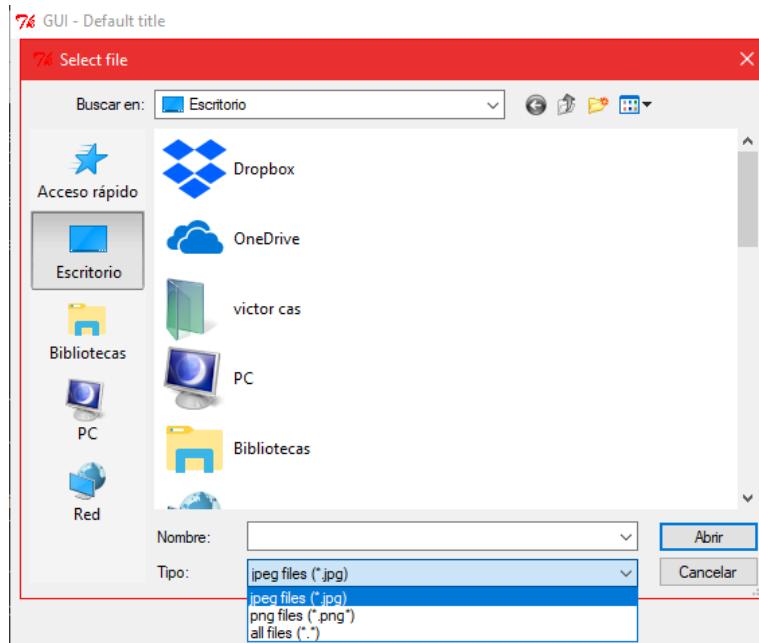


Figura E.12: Interfaz que nos permite buscar un archivo en nuestro sistema.

4.2.- Obtener la imagen mediante una captura

- Al seleccionar esta opción, se inicializará la cámara por defecto (en función del modelo de la cámara, tendremos que especificar la máxima resolución posible, en caso de no conocer este dato, se puede dejar el valor por defecto *[640 X 480]*) [E.13](#).

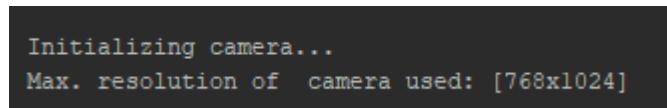


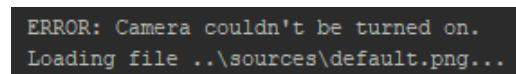
Figura E.13: Inicialización de la cámara con la resolución que se va a usar.

- Con la cámara inicializada correctamente, aparecerá una ventana con la imagen en tiempo-real capturada con la cámara [E.2](#).

- Se pueden seguir las instrucciones del apartado *Instrucciones adicionales: Menú de utilización general de la cámara* [E.4](#) para capturar la imagen que queremos analizar. No nos dejará tomar una imagen hasta que haya exactamente una sola persona en dicha imagen.

NOTAS

- La cámara por defecto es la primera cámara externa que tengamos conectada. En caso de no tener ninguna externa, se utilizará la webcam integrada.
- En caso de no poder inicializar la cámara con los parámetros por defecto, mostrará un mensaje avisándonos y pasará al apartado 6.- *Resultados finales: Interfaz visual* [E.4](#) con una imagen por defecto [E.14](#).



```
ERROR: Camera couldn't be turned on.
Loading file ..\sources\default.png...
```

Figura E.14: La cámara no se pudo encender y se cargó una imagen por defecto.

- En caso de no querer tomar una captura (opción [Q] del menú), se pasará al apartado 6.- *Resultados finales: Interfaz visual* [E.4](#) con una imagen por defecto.

4.3.- Realizar la identificación en tiempo-real

- En este apartado se inicializa la cámara por defecto como en los apartados anteriores, y se muestra directamente en la interfaz (imagen de la izquierda con el título *Original image*).

- Si queremos que nos reconozca (suponiendo que estemos en la base de datos), simplemente debemos ponernos delante de la cámara y esperar a que se actualice la interfaz (este tiempo se ha establecido de **0.1 segundos** para darle tiempo al programa a que extraiga las características del rostro y las compare con las de la red, en caso de disponer de un equipo más potente se puede reducir este tiempo).

- A continuación, se puede observar como los resultados van cambiando en la interfaz explicada en el apartado 6.- *Resultados finales: Interfaz visual* [E.4](#) en función de la persona que se coloque delante de la cámara, o incluso si cambiamos de posición o modificamos la iluminación.

- El proceso que sigue este apartado en cuanto a la obtención de resultados es similar al que se sigue en los otros dos casos (comprobar apartado 5.- *Comparar imágenes y obtener resultados* [E.4](#)), salvo que se repite y actualiza cada **0.1 segundos**.

- Si se cierra la ventana de la interfaz se volverá al menú principal donde se le preguntará al usuario si quiere salir o volver al comienzo E.15.

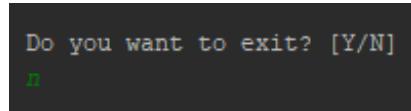


Figura E.15: Programa preguntándonos si queremos salir de la aplicación.

5.- Comparar imágenes y obtener resultados

- Este proceso es interno y no tiene efecto en la interfaz de usuario, de manera que no afecta a estos.
- Con la imagen obtenida de los apartados E.4, E.4, se analiza dicha imagen y se comparan los resultados con los obtenidos de entrenar las imágenes almacenadas en la base de datos (fichero *trainerData.yml* mencionado en el apartado 3.- *Carga de recursos e inicialización* E.4). Esta comparación se realiza mediante algunas funciones proporcionadas por la librería de *OpenCV*.

- Se obtiene una ID perteneciente a la imagen con mejor resultado de la comparación en nuestro fichero de entrenamiento y un porcentaje, que indica el éxito de dicha comparación E.16. Con esa ID, se carga la imagen correspondiente, el nombre que tenga dicha imagen y el porcentaje en la interfaz gráfica.

```
Comparing images...
Information about comparison: id= 13    ---    label= new_image_name    ---    coincidence= 90.35
Loading file ..\sources\facesDataset\face_new_image_name.jpg...
```

Figura E.16: Programa comparando imágenes internamente y obteniendo un resultado.

6.- Resultados finales: Interfaz visual

- La interfaz consiste en una ventana con ambas imágenes [E.17](#): la que se quería reconocer y la de máxima coincidencia de la base de datos obtenida en el apartado [5.- Comparar imágenes y obtener resultados E.4](#).

- Se muestra además el porcentaje de coincidencia que hayan tenido la comparación, y una barra de progreso que indica dicho porcentaje junto con el nombre de la imagen de la base de datos. Esta barra de progreso cambia de color en función del tanto por ciento que hayamos obtenido (negro <10 %, rojo <35 %, naranja <50 %, amarillo <80 %, verde <95 %, morado >= 95 %).



Figura E.17: Interfaz final con ambas imágenes, la barra y el porcentaje de coincidencia (en este caso la interfaz muestra el proceso de reconocimiento en tiempo real, por eso no se encuentra el botón *More information...* explicado a continuación).

- Además, se crea un botón *More Information...* (en los casos [E.4](#) y [E.4](#)), que al pulsarle crea un pequeño *pop-up* con información sobre la imagen resultado. Esta información se puede encontrar en el fichero *info.txt* mencionado en el apartado [1.- Añadir nuevas imágenes a la base de datos E.4](#).



Figura E.18: *Pop-up* con la información de la persona reconocida.

NOTAS

- Las imágenes se recortan para mostrar sólo el rostro y se ajustan todas al mismo tamaño [E.19](#), de esta manera se evitan los tamaños de imágenes excesivamente grandes/pequeños.



Figura E.19: Imagen original de la base de datos // Imagen recortada y reescalada para dejarla en un formato tratable para la red.

- El nombre que se muestra sobre la imagen de la derecha es el de la persona a la que corresponda dicha fotografía (extraído del fichero *info.txt*), mientras que el nombre que se muestra sobre la barra de progreso es el correspondiente al nombre del fichero de dicha foto (en el caso de la imagen [E.17](#), se muestra la información almacenada en [E.5](#)). Se muestran ambos nombres en caso de que el usuario necesite acceder a esa foto en la base de datos manualmente.

Instrucciones adicionales: Menú de utilización general de la cámara

- Cuando el usuario tiene el control sobre la cámara, puede:
- Pulsar [Q] para salir sin tomar ninguna imagen.
- Pulsar [I] para mostrar información sobre esa imagen y el menú de la cámara.
- Pulsar [P] para pausar la captura de imágenes. Mientras se esté en modo pausa el programa se queda en *stand-by*, y no se toman acciones hasta que se reanuda la ejecución.
- Pulsar [ESPACIO] mientras está en modo pausa para reanudar la ejecución.
- Pulsar [C] para tomar una captura y seguir ejecutando el programa.

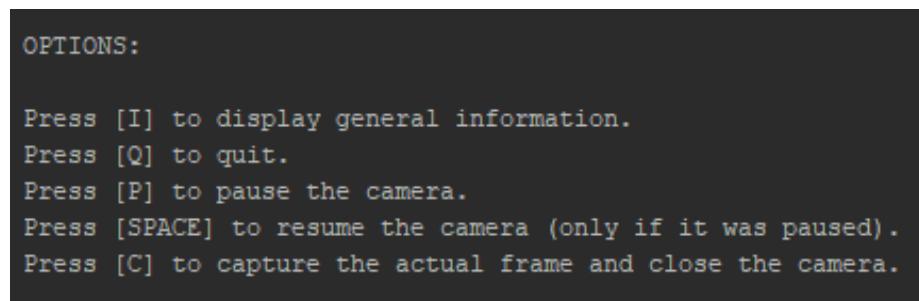


Figura E.20: Menú principal de la cámara.

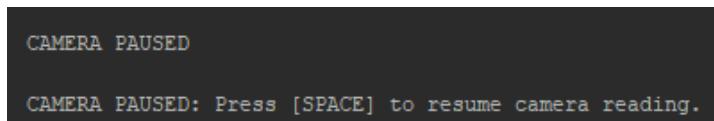


Figura E.21: Mensajes que se muestran cuando se pausa la cámara.

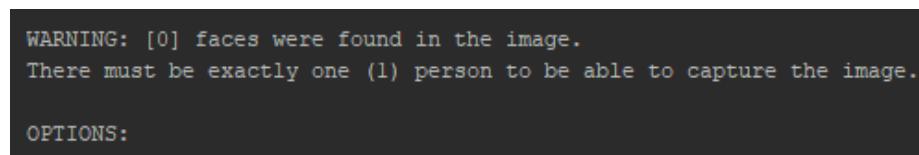


Figura E.22: Información que se muestra cuando se pulsa [I].

Bibliografía

- [1] OpenCV Version 2.4.13. Load, modify, and save an image. https://docs.opencv.org/2.4/doc/tutorials/introduction/load_save_image/load_save_image.html, 2018.
- [2] Abkb and Mark Mikofski. How to change font and size of buttons and frame in tkinter using python? <https://stackoverflow.com/questions/20588417/how-to-change-font-and-size-of-buttons-and-frame-in-tkinter-using-python>, 2013.
- [3] achalddave. image.scale vs opencv resize. <https://github.com/torch/image/issues/188>, 2016.
- [4] ADMIN. Flip image opencv python. <https://scottontechology.com/flip-image-opencv-python/>, 2016.
- [5] alvas and Burhan Khalid. Concatenate item in list to strings. <https://stackoverflow.com/questions/12453580/concatenate-item-in-list-to-strings>, 2012.
- [6] Atlassian. Trello. <https://trello.com>.
- [7] Danielle B. Rounding in python — when arithmetic isn't quite right. <https://kfolks.com/rounding-in-python-when-arithmetic-isnt-quite-right-11a79a30390a>, 2017.
- [8] Shreya Batra. How to undo a mistaken git rebase (life saver). <https://medium.com/@shreyaWhiz/>

- how-to-undo-a-mistaken-git-rebase-life-saver-2977ff0a0602, 2017.
- [9] Martin Beckett and zarthur. Convert numpy array to cvmat cv2. <https://stackoverflow.com/questions/9913392/convert-numpy-array-to-cvmat-cv2>, 2012.
 - [10] Python For Beginners. Reading and writing files in python. <http://www.pythonforbeginners.com/files/reading-and-writing-files-in-python>, 2013.
 - [11] Bodenseo Bernd Klein. Python tkinter course. https://www.python-course.eu/tkinter_labels.php, 2018.
 - [12] Tanmay Bhatnagar and thewaywewere. Resize an image without distortion opencv. <https://github.com/torch/image/issues/188>, 2017.
 - [13] Cerin, Multimedia Mike, and jsbueno. How to programmatically capture a webcam photo. <https://stackoverflow.com/questions/9711946/how-to-programmatically-capture-a-webcam-photo>, 2012.
 - [14] Andrei Cheremskoy. Opencv installation on linux and windows. <https://gettocode.com/2017/02/07/opencv-installation-on-windows-and-maybe-linux>, 2017.
 - [15] Omid CompSCI and Afloroaie Robert. How to get everything from the list except the first element using list slicing [duplicate]. <https://stackoverflow.com/questions/40443331/how-to-get-everything-from-the-list-except-the-first-element-using-list-slicing>, 2016.
 - [16] david_p, Noelkd, and BANZ111. Ttk.progressbar: How to change thickness of a horizontal bar. <https://stackoverflow.com/questions/17912624/ttk-progressbar-how-to-change-thickness-of-a-horizontal-bar>, 2013.
 - [17] Dr Dre and Guyggarty. Resizing the output window of imshow function. <http://answers.opencv.org/question/84985/resizing-the-output-window-of-imshow-function/>, 2016.
 - [18] duhhunjonn and pycrust. How do i list all files of a directory? <https://stackoverflow.com/questions/3207219/how-do-i-list-all-files-of-a-directory>, 2013.

- [19] Evan Fosmark and pobk. How can i make a time delay in python? <https://stackoverflow.com/questions/510348/how-can-i-make-a-time-delay-in-python>, 2012.
- [20] Python Software Foundation. Building and installing python modules. <https://docs.python.org/3/library/distutils.html#module-distutils>, 2018.
- [21] Python Software Foundation. enum — support for enumerations. <https://docs.python.org/3/library/enum.html>, 2018.
- [22] Python Software Foundation. Installing packages. <https://packaging.python.org/tutorials/installing-packages/#requirements-for-installing-packages>, 2018.
- [23] Python Software Foundation. Python packages. <https://docs.python.org/2/tutorial/modules.html#packages>, 2018.
- [24] franka and Amber. Python main call within class. <https://stackoverflow.com/questions/7870869/python-main-call-within-class>, 2011.
- [25] Tom Fuller, PM 2Ring, and Steven Summers. I can't seem to open two windows with python, tkinter. <https://stackoverflow.com/questions/39039123/i-cant-seem-to-open-two-windows-with-python-tkinter>, 2016.
- [26] Matthew G, thebjorn, and Froyo. Capturing a single image from my webcam in java or python. <https://stackoverflow.com/questions/11094481/capturing-a-single-image-from-my-webcam-in-java-or-python>, 2012.
- [27] Christoph Gohlke. Unofficial windows binaries for python extension packages. <https://www.lfd.uci.edu/~gohlke/pythonlibs/#pygame>.
- [28] Michael Hirsch. Install opencv 3.4 in python 3.6 / 2.7. <https://www.scivision.co/install-opencv-python-windows/>, 2018.
- [29] Jogofus and FJSevilla. Duda con raw_input(). <https://es.stackoverflow.com/questions/38288/duda-con-raw-input>, 2017.
- [30] kip. Python - ejemplo del operador ternario en python. <https://www.lawebdelprogramador.com/foros/Python/1517833-Ejemplo-del-operador-ternario-en-Python.html>, 2016.

- [31] Bernd Klein. Abstract classes. https://www.python-course.eu/python3_abstract_classes.php, 2018.
- [32] levacjeep and ShadowRanger. Python3 - typeerror: module.__init__() takes at most 2 arguments (3 given). <https://stackoverflow.com/questions/35367340/python3-typeerror-module-init-takes-at-most-2-arguments-3-given/35367871>, 2016.
- [33] Logitech. Webcam c170 - plug-and-play video calls. <https://www.logitech.com/en-gb/product/webcam-c170>, 2018.
- [34] M456 and dF. Saving a numpy array as an image. <https://stackoverflow.com/questions/902761/saving-a-numpy-array-as-an-image>, 2009.
- [35] mahasamoot. trouble with the tutorial: Typeerror: step() missing 1 required positional argument: 'model'. <https://github.com/projectmesa/mesa/issues/308>, 2016.
- [36] md1hunox and Abid Rahman K. Using other keys for the waitkey() function of opencv. <https://stackoverflow.com/questions/14494101/using-other-keys-for-the-waitkey-function-of-opencv>, 2017.
- [37] Mike. Getting your screen resolution with python. <https://www.blog.pythonlibrary.org/2015/08/18/getting-your-screen-resolution-with-python/>, 2015.
- [38] Alexander Mordvintsev and Abid K. Getting started with videos - capture video from camera. http://opencv-python-tutroals.readthedocs.io/en/latest/py_tutorials/py_gui/py_video_display/py_video_display.html, 2013.
- [39] MS23 and berak. What is the label that 'predict' sends? <http://answers.opencv.org/question/138933/what-is-the-label-that-predict-sends>, 2017.
- [40] Nolik and Froyo. How to crop an image in opencv using python. <https://stackoverflow.com/questions/15589517/how-to-crop-an-image-in-opencv-using-python>, 2013.
- [41] Casper Olsson and BrtH. Open images? python. <https://stackoverflow.com/questions/16387069/open-images-python>, 2013.

- [42] OpeCV. cv::face::facerecognizer class reference. https://docs.opencv.org/trunk/dd/d65/classcv_1_1face_1_1FaceRecognizer.html, 2018.
- [43] OpenCV. Image thresholding. https://docs.opencv.org/3.4.0/d7/d4d/tutorial_py_thresholding.html, 2017.
- [44] OpenCV. haarcascades. <https://github.com/opencv/opencv/blob/master/data/haarcascades>, 2018.
- [45] Russell Owen and Andrew Svetlov. Python bug tracker - wait_variable hangs at exit, 2004-2012. [Internet; descargado 22-junio-2018].
- [46] Colonel Panic. How do i install pip on windows? <https://stackoverflow.com/questions/4750806/how-do-i-install-pip-on-windows>, 2017.
- [47] Joey Payne. Catching python exceptions – the try/except/else keywords. <https://www.pythoncentral.io-catching-python-exceptions-the-try-except-else-keywords>, 2013.
- [48] pazdera. Singleton example in python. <https://gist.github.com/pazdera/1098129>, 2011.
- [49] Benjamin Peterson. Python 2 and 3 compatibility utilities. <https://pypi.org/project/six/#description>, 2018.
- [50] poljakov13 and eshirima. How to display multiple images in one window? <http://answers.opencv.org/question/175912/how-to-display-multiple-images-in-one-window/>, 2017.
- [51] Pythonspot. Tkinter message box. <https://pythonspot.com/tk-message-box>, 2017.
- [52] Rikg09, Noshii, and mayure098. Tkinter command for button not working. <https://stackoverflow.com/questions/45710162/tkinter-command-for-button-not-working>, 2017.
- [53] Seth Robertson and Adam McKerlie. On undoing, fixing, or removing commits in git. https://gist.github.com/silent1mezzo/1670623#discard_all_unpushed, 2011.
- [54] Rodrigo and John Montgomery. How do i access my webcam in python? <https://stackoverflow.com/questions/604749/how-do-i-access-my-webcam-in-python>, 2009.

- [55] Adrian Rosebrock. Opencv with tkinter. <https://www.pyimagesearch.com/2016/05/23/opencv-with-tkinter>, 2016.
- [56] Antonio Serrano and Martin Beckett. Make a pause between images display in opencv. <https://stackoverflow.com/questions/46671348/make-a-pause-between-images-display-in-opencv>, 2017.
- [57] Silmarillion101. Tkinter create image function error (pyimage1 does not exist). <https://stackoverflow.com/questions/23224574/tkinter-create-image-function-error-pyimage1-does-not-exist>, 2014.
- [58] sinnaps. Metodología de un proyecto. <https://www.sinnaps.com/blog-gestion-proyectos/metodologia-de-un-proyecto>, 2018.
- [59] Sol. Install opencv 3 with python 3 on windows. <https://solarianprogrammer.com/2016/09/17/install-opencv-3-with-python-3-on-windows/>, 2016.
- [60] sopl. disp_multiple_images.py. <https://gist.github.com/sopl/f3eec2e79c165e39c9d540e916142ae1>, 2018.
- [61] spence91 and rslite. How to check whether a file exists? <https://stackoverflow.com/questions/82831/how-to-check-whether-a-file-exists>, 2008.
- [62] Tom and guillegraldo. How to disable window controls when a modal dialog box is active in tkinter? <https://stackoverflow.com/questions/31892015/how-to-disable-window-controls-when-a-modal-dialog-box-is-active-in-tkinter>, 2015.
- [63] Tomha, Marcin, bodger, and scraper. Tkinter resize background image to window size (python 3.4). <https://stackoverflow.com/questions/24061099/tkinter-resize-background-image-to-window-size-python-3-4>, 2014 - 2016.
- [64] UnkwnTech and Nick Stinemates. How to delete the contents of a folder in python? <https://stackoverflow.com/questions/185936/how-to-delete-the-contents-of-a-folder-in-python>, 2008.
- [65] user2971844 and Breeze. Opencv - cannot find module cv2. <https://stackoverflow.com/questions/19876079/opencv-cannot-find-module-cv2>, 2013 - 2017.

- [66] victor1234 and elzbth. How to set camera fps in opencv? cv_cap_prop_fps is a fake. <https://stackoverflow.com/questions/7039575/how-to-set-camera-fps-in-opencv-cv-cap-prop-fps-is-a-fake>, 2011-2014.
- [67] Wikipedia. Desarrollo en cascada — wikipedia, la enciclopedia libre, 2018. [Internet; descargado 30-mayo-2018].
- [68] Wikipedia. Distributed computing, 2018. [Internet; descargado 10-junio-2018].
- [69] Wikipedia. Parallel computing, 2018. [Internet; descargado 10-junio-2018].
- [70] Wikipedia. Scrum (desarrollo de software) — wikipedia, la enciclopedia libre, 2018. [Internet; descargado 30-mayo-2018].
- [71] Sahat Yalkabov and karmakaze. How to remove a directory from git repository? <https://stackoverflow.com/questions/6313126/how-to-remove-a-directory-from-git-repository>, 2011.
- [72] Yasoob. The open function explained. <https://pythontips.com/2014/01/15/the-open-function-explained/#more-416>, 2014.
- [73] Zaiste. Abstract classes in python. https://zaiste.net/abstract_classes_in_python/, 2010.
- [74] ZenHub. Zenhub. <https://www.zenhub.com>.