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 — 17 de junio
de 2018
Tutor: César Represa

Índice general

Índice de figuras

Índice de tablas

Apéndice A

Plan de Proyecto Software

A.1. Introducción

Aunque nos hemos basado en un método de desarrollo en cascada [3], hemos incluido algunos elementos de la **metodología scrum** [6] 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 B.4 que teníamos que cumplir, 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 [2]:

- **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.

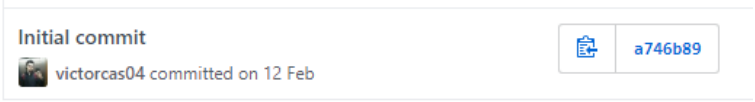


Figura A.1: Primer commit.

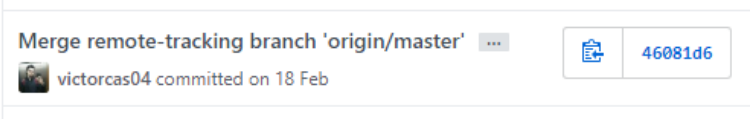


Figura A.2: Commit con el que finaliza la parte del planteamiento del proyecto.

- **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* [1], 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* [7] 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	Develop
	Create class diagram	Develop
	Basic readme	Develop
2.- Image manipulation	Open image from file	Develop
	Save images	Develop
	Tensorflow Tutorials	Develop
	Meeting	Meeting
3.- Basic image recognition	Import OpenCV	Develop
	Capture image from camera	Develop
	Normalize illumination	Enhancement
	Create basic parameters	Develop
	Facial recognition (I)	Develop
	Meeting	Meeting
4.- Apply computer vision	Improve camera recording	Enhancement
	Change face location display	Enhancement
	Facial recognition (II)	Develop
	Meeting	Meeting
	Acquire camera	Enhancement
5.- Create database	Create local storage	Develop
	Get more images	Enhancement
6.- Interface	Basic interface	Develop
	Dynamic window	Enhancement
	Facial recognition (III)	Enhancement
	Camera view	Enhancement
	Result images	Develop
	Compare bar/percentage	Develop
	Meeting	Meeting
	Result information	Develop

Continúa en la siguiente página...

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

Milestones	Issues	Type of Issue
7.- Report	Executable	Develop
	Basic report	Develop
	Meeting	Meeting
	Clean and order code	Enhancement
	Update readme	Enhancement
	Final report	Enhancement
	Video	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 y último commit de esta fase A.4.

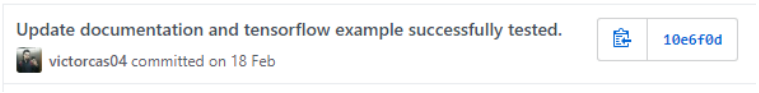


Figura A.3: Commit con los recursos instalados en el proyecto.

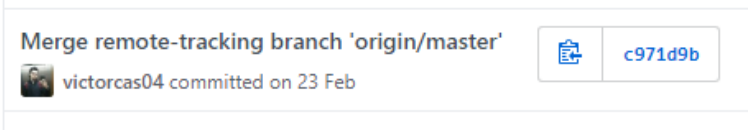


Figura A.4: Commit con el que finaliza la instalación de recursos.

- 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.5 hasta mediados de Marzo A.6.

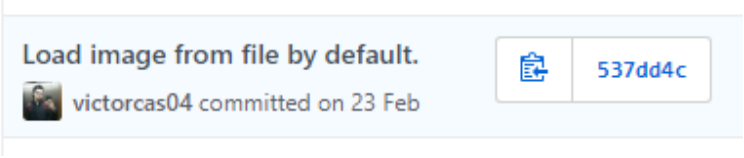


Figura A.5: Commit con las primeras imágenes de prueba.

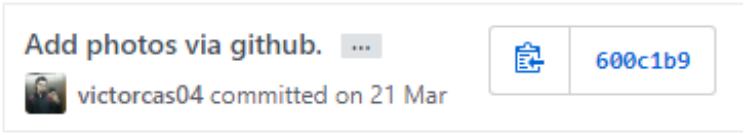


Figura A.6: Commit con el que finaliza la fase de abrir, capturar, guardar y modificar una imagen.

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.7.

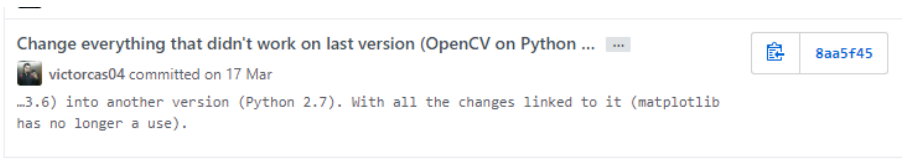


Figura A.7: Commit que solucionaba el problema de recursos no compatibles con las nuevas versiones de python.

- 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.7.

  En este apartado se desarrolla la captura por pantalla de un frame A.8. A continuación, se implementa la primera parte del reconocimiento facial, donde se localizan los rostros de la gente en dicha imagen A.9, previamente normalizada A.10, 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).

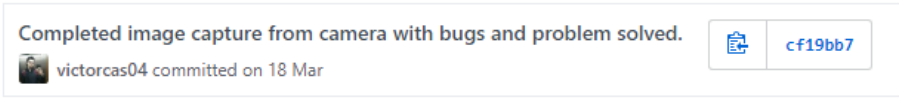


Figura A.8: Commit con la implementación de la captura de la cámara.

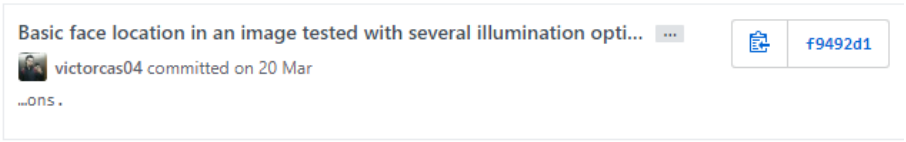


Figura A.9: Commit con la localización del rostro en una imagen implementada.

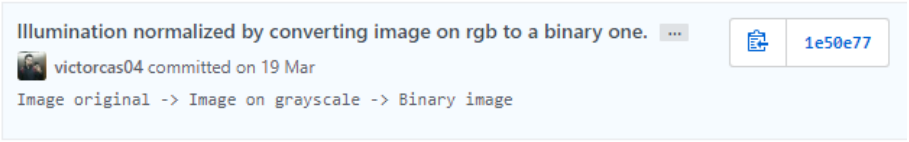


Figura A.10: Commit con la normalización de la iluminación de la imagen.

Además, se implementó una primera versión de reconocimiento facial a partir de la técnica *LBPH* A.11, explicada en la memoria principal, en la sección *3_conceptos_teoricos*.

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

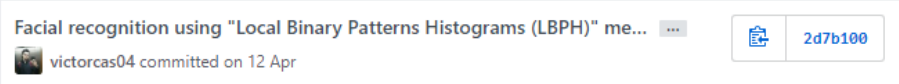


Figura A.11: Reconocimiento facial utilizando el método LBPH..

- 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.12, A.13 restringiendo la zona de obtención de características del rostro y cambiando parámetros tanto de iluminación como de la predicción de la red A.14.

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

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

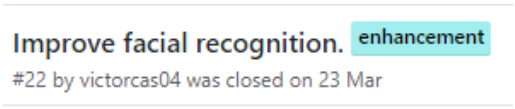


Figura A.13: Commit con cambios en la implementación de los métodos que permiten el reconocimiento facial (parte II).

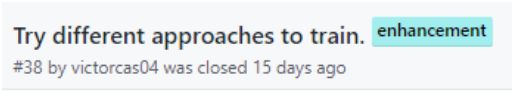


Figura A.14: Commit con cambios en la implementación de los métodos que permiten el reconocimiento facial (parte III).

- 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 ajusta el tamaño de las imágenes de salida para hacer dicha salida más estable a la hora de realizar comparativas o ejecuciones sucesivas A.15, 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 0.5 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.16, aunque no sea modificable por el usuario (se han inhabilitado las opciones de minimización y reescalado).

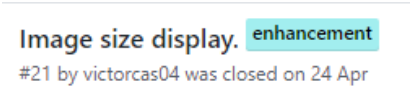


Figura A.15: Commit con cambios en la implementación de los métodos que permiten el reconocimiento facial (parte I).

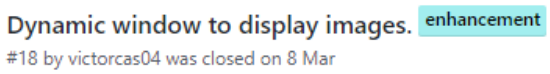


Figura A.16: Commit con la implementación de ventana de la nueva interfaz.

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.17.

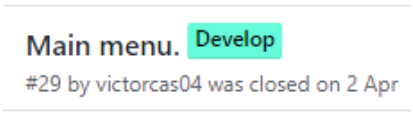


Figura A.17: Commit con el menú final de cara al usuario final.

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.18.

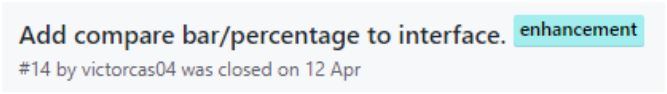


Figura A.18: Commit con la implementación de la barra de progreso.

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.19.

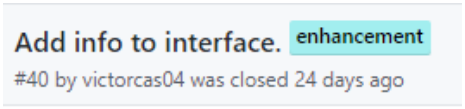


Figura A.19: Commit con el 'popup' y la información extra.

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.20.

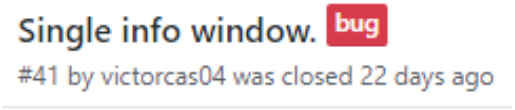


Figura A.20: Commit solucionando el problema de las ventanas 'popup'.

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 uqe 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.21.

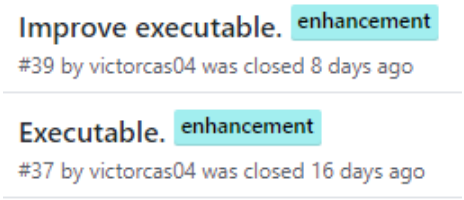


Figura A.21: Commit con el ejecutable base y su mejora.

- 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.22.

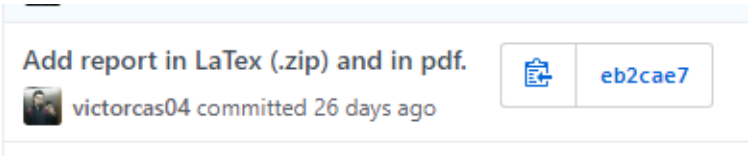


Figura A.22: Commit con la primera versión de la memoria en LaTex.

También se ha actualizado el fichero *Readme.md* que sirve como carta de presentación de nuestro proyecto en *Github* A.23.

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

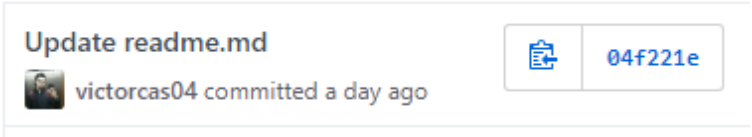


Figura A.23: Commit con una de las actualizaciones del fichero README.md.

- **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.

- **Histórico de commits**

  Se puede ver un gráfico con el histórico de commits en Github en la imágen A.24. 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.

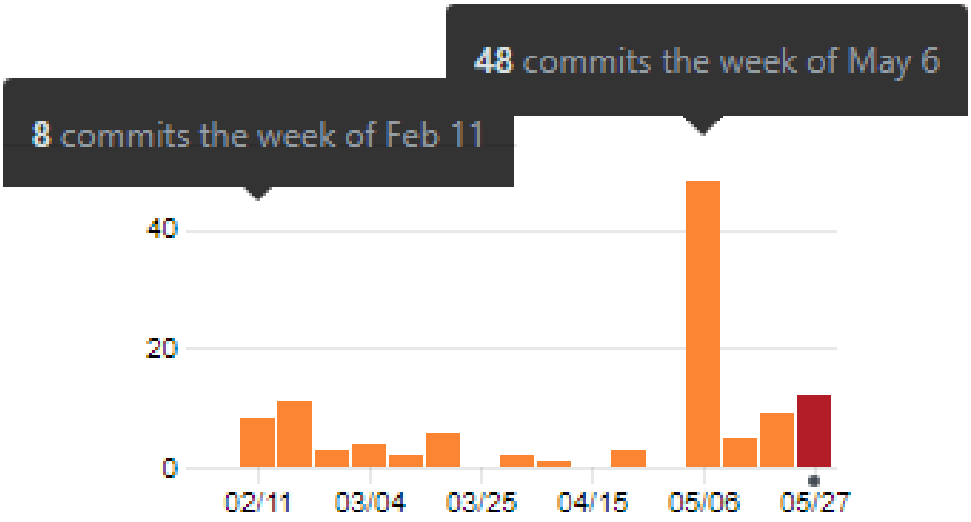


Figura A.24: Grafo con el número de commits por semana.

Flujogramas y diagramas de clases iniciales

A continuación se dejan algunos de los flujogramas realizados al comienzo del proyecto. La mayor parte de ellos se han mantenido hasta el final salvo por pequeños detalles que se han tenido que modificar.

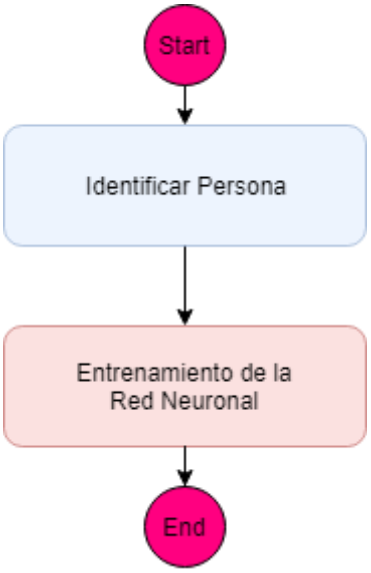


Figura A.25: Flujograma inicial de primer nivel.

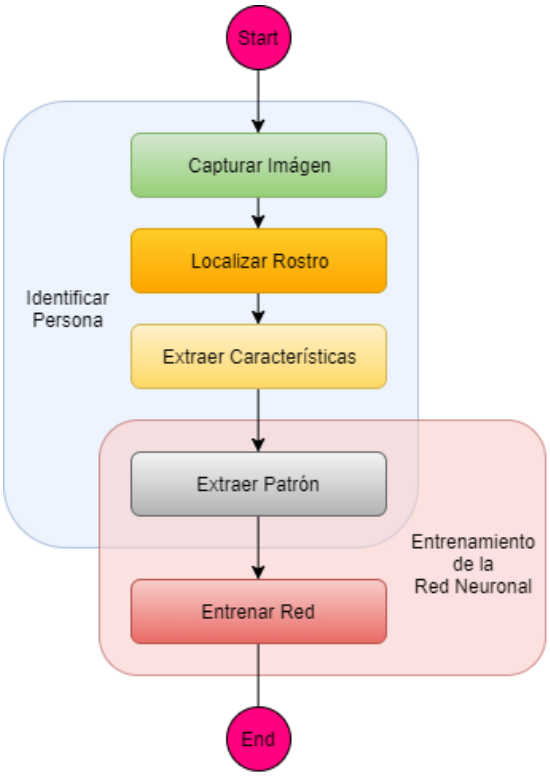


Figura A.26: Flujograma inicial de 2º nivel.

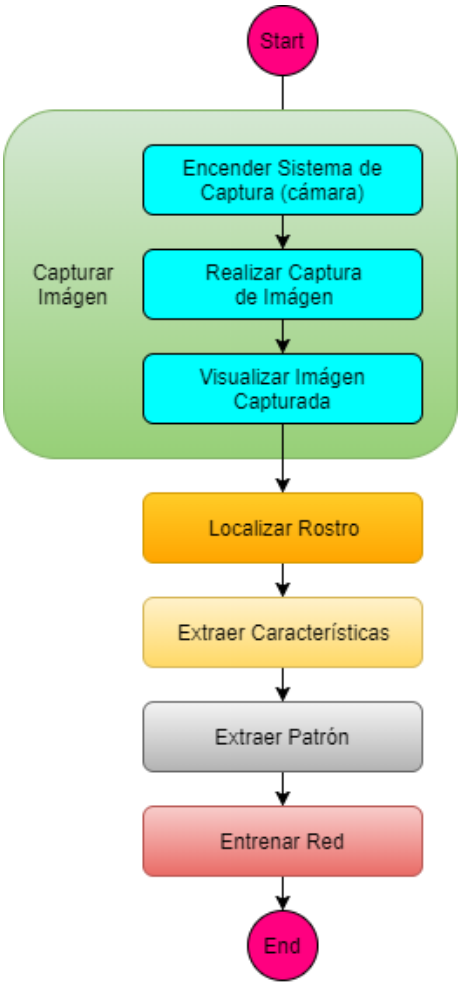


Figura A.27: Flujograma inicial de tercer nivel: Capturar imagen.

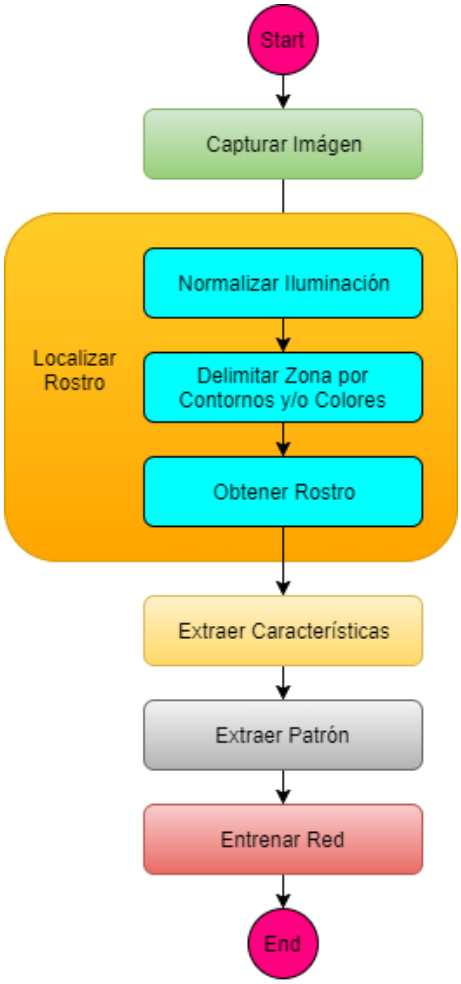


Figura A.28: Flujograma inicial de tercer nivel: Localizar rostro.

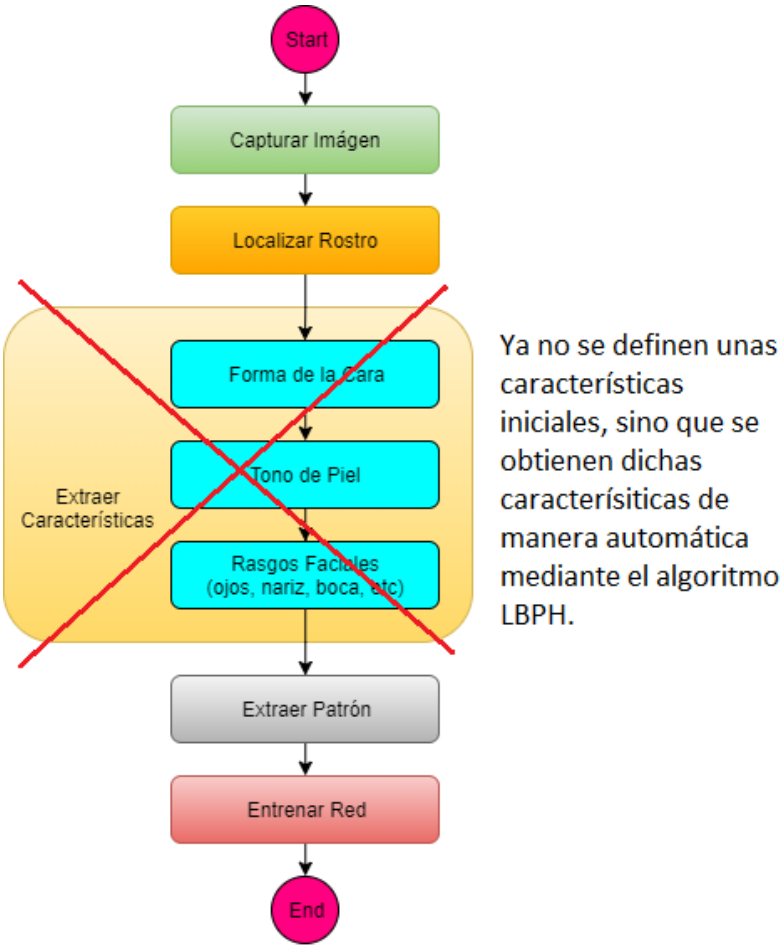


Figura A.29: Flujograma inicial de tercer nivel: Extraer características.

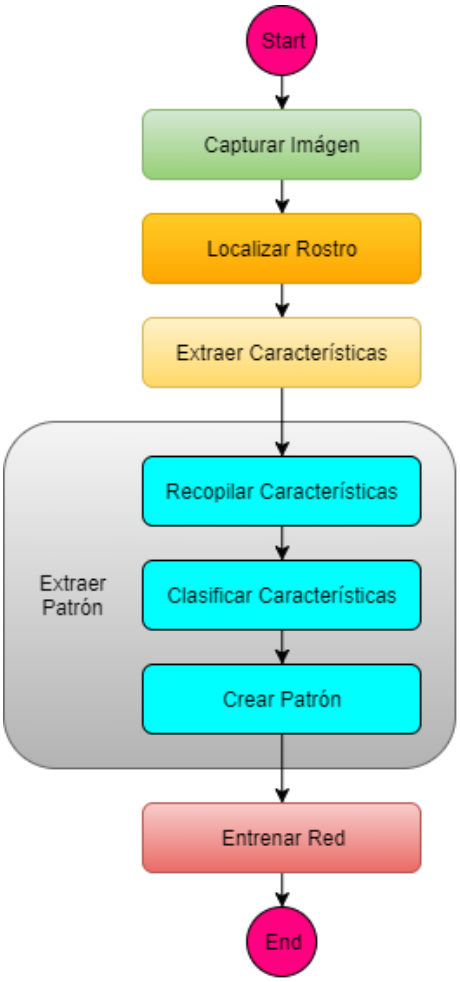


Figura A.30: Flujograma inicial de tercer nivel: Extraer patrón.

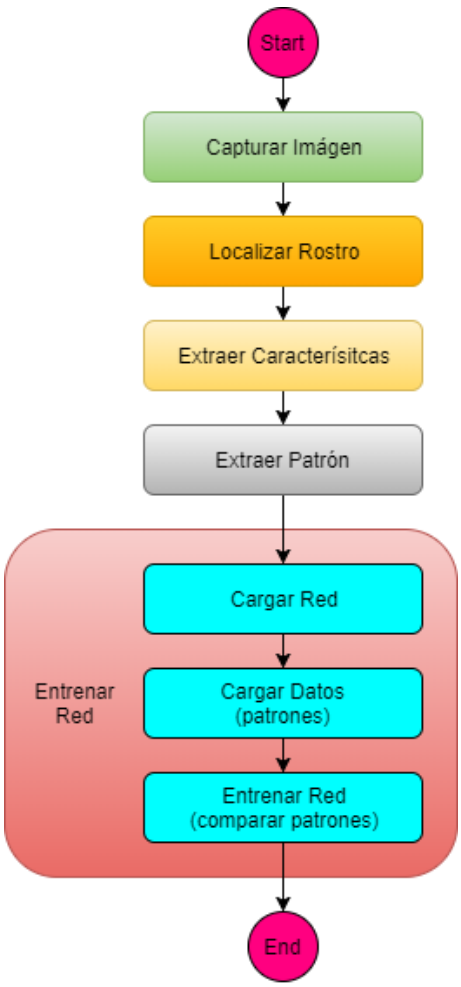


Figura A.31: Flujograma inicial de tercer nivel: Entrenar red.

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* [4] en el cual se tenga un servidor (que sea capaz de realizar procesamiento en paralelo [5]) 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 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

B.2. Objetivos generales

B.3. Catalogo de requisitos

B.4. Especificación de requisitos

Apéndice C

Especificación de diseño

C.1. Introducción

C.2. Diseño de datos

C.3. Diseño procedimental

C.4. Diseño arquitectónico

Apéndice D

Documentación técnica de programación

D.1. Introducción

D.2. Estructura de directorios

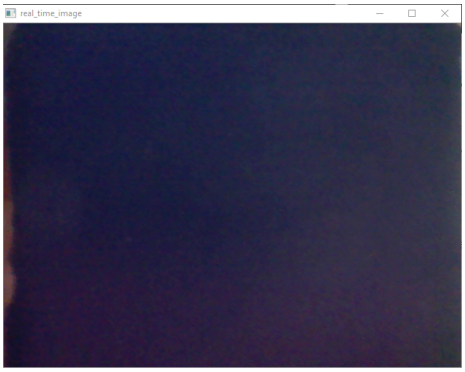
D.3. Manual del programador

D.4. Compilación, instalación y ejecución del proyecto

D.5. Pruebas del sistema

Si nuestra base de datos está vacía o tiene menos de dos imágenes, utilizará el fichero *.yml* de antes de entrenar la red... imágenes de la interfaz con los diferentes errores y warnings

Texto explicando que es una imagen de prueba en la que no hay rostros.



ESTRUCTURA DE CADA FICHERO; LO QUE HACE CADA UNO; ASPECTOS RELEVANTES ETC INTERFACE; ENLACES; MÉTODOS ETC

Imágen de referencia cuando la base de datos no tiene imágenes (o menos de dos) D.1.

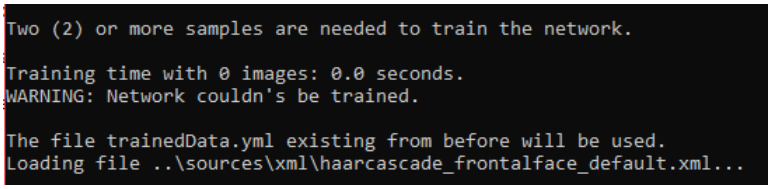


Figura D.1: Fallo al entrenar por haber 0 imágenes en la base de datos.

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*.

NOTA: En el resto de versiones de windows no se asegura su correcto funcionamiento.

NOTA: 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.

NOTA: Es necesario tener los requisitos de usuario instalados antes de continuar E.2.

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.

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 *LBPH* y la herramiento *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 le 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 contra-producente.

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.

- 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).

- 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. En caso de no querer rellenar 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*.

- **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.

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.

- En caso de decir que NO, se utilizarán los recursos almacenados del último entrenamiento (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).

- En caso de decir que SI, se crearán las imágenes en el formato adecuado y se entrenará la red, utilizando los recursos recién 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).

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.

- 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.

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

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]*).

- *NOTA*: La cámara por defecto es la primera cámara externa que tengamos conectada. En caso de no tener ninguna externa, se utilizará la webcam integrada.

- *NOTA*: en caso de no poder inicializar la cámara con estos parámetros, mostrará un mensaje avisándonos y pasará al apartado *6.- Resultados finales: Interfaz visual* E.4 con una imagen por defecto.

- Con la cámara inicializada correctamente, aparecerá una ventana con la imagen en tiempo-real capturada con la cámara.

- 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.

- *NOTA*: 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 **0.5 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 **0.5 segundos**.

- Para terminar de ejecutar el programa en este punto cerrar la ventana de la interfaz.

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. Con esa ID, se carga la imagen correspondiente, el nombre que tenga dicha imagen y el porcentaje en la interfaz gráfica.

6.- Resultados finales: Interfaz visual

- La interfaz consiste en una ventana con ambas imágenes: 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.

- *NOTA*: las imágenes se recortan para mostrar sólo el rostro y se ajustan todas al mismo tamaño, de esta manera se evitan los tamaños de imágenes excesivamente grandes/pequeños.

- 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 %).

- *NOTA*: 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. Se muestran ambos nombres en caso de que el usuario necesite acceder a esa foto en la base de datos manualmente.

- 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.

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.

A continuación se deja una imagen de ejemplo de un resultado de ejecución obteniendo la imagen mediante fichero E.1.

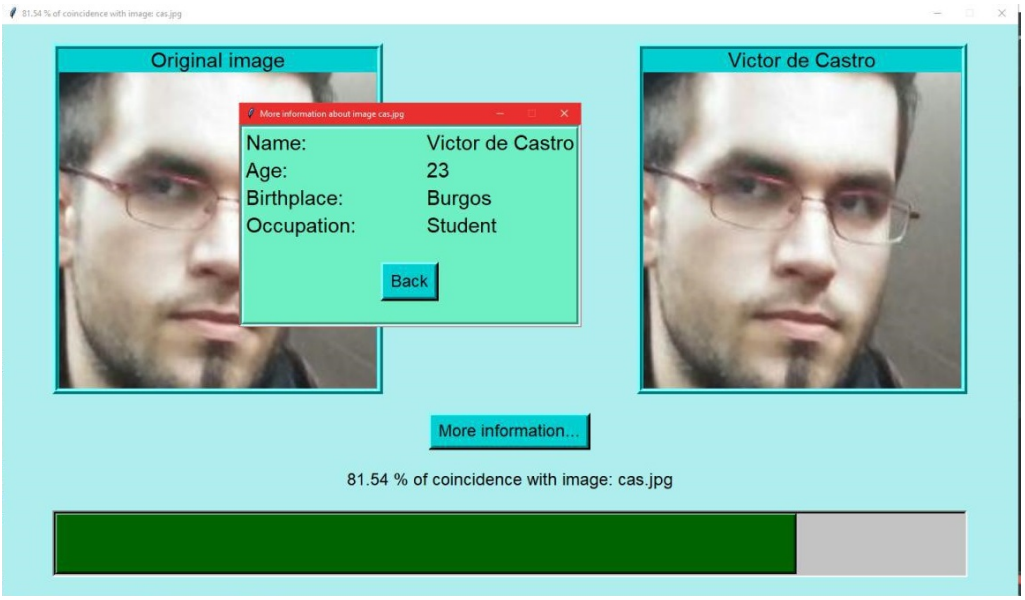


Figura E.1: Ejemplo de interfaz final con color al comparar dos imágenes (desde fichero).

Bibliografía

[1] Atlassian. Trello. https://trello.com.

[2] sinnaps. MetodologÍa de un proyecto. https://www.sinnaps.com/blog-gestion-proyectos/metodologia-de-un-proyecto, 2018.

[3] Wikipedia. Desarrollo en cascada — wikipedia, la enciclopedia libre, 2018. [Internet; descargado 30-mayo-2018].

[4] Wikipedia. Distributed computing, 2018. [Internet; descargado 10-junio-2018].

[5] Wikipedia. Parallel computing, 2018. [Internet; descargado 10-junio-2018].

[6] Wikipedia. Scrum (desarrollo de software) — wikipedia, la enciclopedia libre, 2018. [Internet; descargado 30-mayo-2018].

[7] ZenHub. Zenhub. https://www.zenhub.com.