



UNIVERSIDAD DE BURGOS
ESCUELA POLITÉCNICA SUPERIOR
Grado en Ingeniería Informática



**TFG del Grado en Ingeniería
Informática**

**Diseño de un sistema de
identificación de personas**



Presentado por Víctor de Castro Hurtado
en Universidad de Burgos — 13 de mayo
de 2018

Tutor: César Represa



UNIVERSIDAD DE BURGOS
ESCUELA POLITÉCNICA SUPERIOR
Grado en Ingeniería Informática



D. César Represa, profesor del departamento de Ingeniería Electromecánica, área de Tecnología Electrónica.

Expone:

Que el alumno D. Víctor de Castro Hurtado, con DNI 71289378G, ha realizado el Trabajo final de Grado en Ingeniería Informática titulado Diseño de un sistema de identificación de personas.

Y que dicho trabajo ha sido realizado por el alumno bajo la dirección del que suscribe, en virtud de lo cual se autoriza su presentación y defensa.

En Burgos, 13 de mayo de 2018

Vº. Bº. del Tutor:

D. César Represa

Resumen

Se pretende diseñar un sistema de **identificación de personas** a través de un programa de **reconocimiento facial** con rostros previamente incluidos en una base de datos. El sistema se desarrollará en **Python** y utilizando técnicas de **Machine Learning**.

Descriptores

python, inteligencia artificial, IA, reconocimiento facial, cámara, tiempo real, identificación, aprendizaje automático, aprendizaje profundo, entrenando de redes, visión artificial

Abstract

The main goal is to develop a **people identification** system with a **facial recognition** program and faces included on a database. This system will be developed in **Python** using **Machine Learning** techniques.

Keywords

python, artificial intelligence, AI, facial recognition, camera, real time, identification, machine learning, deep learning, network training, computer vision

Índice general

Índice general	III
Índice de figuras	IV
Índice de tablas	V
Introducción	1
Objetivos del proyecto	7
Conceptos teóricos	9
3.1. Conceptos	9
Técnicas y herramientas	13
Aspectos relevantes del desarrollo del proyecto	15
Trabajos relacionados	17
Conclusiones y Líneas de trabajo futuras	19
Referencias bibliográficas	21

Índice de figuras

Índice de tablas

Introducción

El proyecto consiste en una aplicación que, dadas dos imágenes (una obtenida desde una cámara y la otra alojada localmente), sea capaz de distinguir si en ambas imágenes se encuentra la misma persona utilizando técnicas de machine-learning.

El repositorio del proyecto se puede encontrar en el siguiente [repositorio](#).

La estructura que sigue nuestro proyecto y que se puede encontrar en Github es la siguiente:

- Carpeta **Other**

En esta carpeta se encuentra el material ajeno al proyecto en python, y que, por lo tanto, no afecta a la ejecución del programa. En dicha carpeta se encuentra:

- Carpeta **ClassDiagram**

Esta carpeta contiene los diagramas de clases, tanto los que representaban la estructura inicial del proyecto como el resultado final. En ellos se incluye también la estructura de paquetes, cómo se relacionan entre ellas, etc. La explicación sobre ello se puede encontrar en el Anexo C-Diseno del documento anexos.pdf.

- Carpeta **flowCharts**

Esta carpeta contiene los diagramas de flujo del programa, representando cuales son los pasos que sigue, posibles ramas y

desenlaces que puede tener. Como en el caso de los diagramas de clases, se incluyen los diagramas con la idea original y el resultado final. La explicación sobre ello se puede encontrar en el Anexo C-Diseno del documento anexos.pdf.

- Carpeta **results**

Esta carpeta contiene imágenes sobre la ejecución del programa, para poder observar cómo sería el resultado tanto si se ejecuta sin problemas, pero no identificando a la persona que tiene delante de la cámara, como si se ejecuta sin problemas identificándola, incluso posibles fallos.

- Fichero **TFG-FacialRecognition_{Memoria}.pdf**

Contiene esta memoria en formato .pdf.

- Fichero **TFG-FacialRecognition_{Anexos}.pdf**

Contiene los anexos de la memoria, en formato .pdf.

- Carpeta comprimida **TFG-FacialRecognition.zip**

Contiene esta memoria y los anexos, divididos en diferentes ficheros .tex.

- Carpeta **sources**

En esta carpeta 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 guardamos al principio en nuestra base de datos. Como se puede observar, son imágenes de cuerpo entero la mayoría de ellas, por lo tanto no están ajustadas a la posición de la cara de la persona, por lo que necesitamos crear (interna y automáticamente) 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.

- 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 el siguiente [enlace](#).

- 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.

- Fichero **dictionary-ID-labels.txt**

Este fichero 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.

- Fichero **info.txt**

Este fichero 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 en un futuro se podría incluir en la base de datos para evitar accesos no permitidos.

- Fichero **trainedData.yml**

Este fichero contiene las principales características de los rostros detectados cuando entrenamos la red. Es el resultado de dicho entrenamiento, y por lo tanto se crea automáticamente. Las características que obtiene son las mismas que las

que se pueden encontrar en los ficheros .xml de dicha carpeta.

- 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.

- Carpeta **src**

En esta carpeta se encuentra el código de nuestro programa. Está dividido en diferentes clases que vemos a continuación:

- 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 o desde la cámara, 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 **util.py**

Contiene todos los métodos de utilidad, desde pedir una simple cadena de texto por teclado hasta inicializar la cámara y guardar la imagen que queramos para compararla, pasando por cargar los diferentes ficheros .xml y .txt que usamos o mostrar los menús por pantalla. Se pueden ver todos los métodos en el anexo D-Manual-programador.

- 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 de ha

realizado con tkinter. Se puede ver cada uno de los métodos y su funcionalidad en el anexo D-Manual-programador.

- 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.

- 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 **README.txt**

Fichero con información general sobre el proyecto, instrucciones básicas y modo de empleo.

Objetivos del proyecto

El proyecto pretende servir como forma de identificación de personas que, teniendo una imagen de la persona en cuestión previamente obtenida en la base de datos, y su información básica (nombre, edad, lugar de nacimiento y profesión), se puede identificar a dicha persona cuando se pone delante de la cámara utilizando técnicas de machine-learning y computer vision.

Conceptos teóricos

3.1. Conceptos

Algunos de los conceptos que vamos a necesitar comprender para entender el proyecto son los siguientes:

- **Inteligencia Artificial**

La inteligencia artificial (IA) según [searchdatacenter](#), es 'la simulación de procesos de inteligencia humana por parte de máquinas, especialmente sistemas informáticos. Estos procesos incluyen el aprendizaje (la adquisición de información y reglas para el uso de la información), el razonamiento (usando las reglas para llegar a conclusiones aproximadas o definitivas) y la autocorrección.'

- **Aprendizaje Automático**

El aprendizaje automático (o más conocido por su nombre en inglés: Machine Learning). es la parte de la inteligencia artificial que consiste en conseguir que **un ordenador actúe de manera automática**, sin necesidad de programarlo para ello.

- **Analítica predictiva**

La analítica predictiva es la ciencia que analiza los datos para intentar predecir datos futuros relacionados. En nuestro caso, a partir de unos datos originales, se analizan mediante el aprendizaje automático para crear nuevos modelos predictivos.

■ Aprendizaje profundo

El aprendizaje profundo (o más conocido por su nombre en inglés: Deep Learning), es una rama del aprendizaje automático que automatiza la analítica predictiva.

■ Red neuronal artificial

Las **redes neuronales artificiales** son un modelo computacional basado en nodos y enlaces, donde **cada nodo corresponde con una neurona**, y los enlaces corresponden a las relaciones entre dichos nodos. Cada nodo almacena información independiente del resto de nodos, de manera que nuestro sistema se divide en función de la información independiente que tenga.

Dados unos **datos de entrada** (nodos 'input'), se forman relaciones entre ellos para establecer los aspectos comunes y las diferencias.

De estas relaciones nos quedamos con los **nodos intermedios** de la red (nodos 'hidden'), que son los que no ve el usuario (programador), y que puede haber tantas capas de nodos 'hidden' como hagan falta.

Por último, la última de estas capas se conforma de los **nodos de salida** que vamos a obtener como **resultado del entrenamiento** de la red (nodos 'output').

Este proceso se puede observar en la **siguiente imagen**.

El objetivo de la red neuronal es resolver los problemas de la misma manera que el cerebro humano, aunque de forma más abstracta y con variantes de entre miles a millones de neuronas (nodos).

Algo importante cuando hablamos de redes neuronales, inteligencia artificial y machine learning, es que **no se puede garantizar nunca su grado de éxito** después de realizar el auto-aprendizaje. Para obtener unos resultados más o menos homogéneos, se necesitan miles y miles de ejecuciones.

■ Tipos de Entrenamiento

Los diferentes tipos de entrenamiento de una red son:

- **Aprendizaje supervisado**

En este tipo de aprendizaje los datos tienen etiquetas, las cuales se usan para obtener nuevos patrones relacionados con los datos de dichas etiquetas.

- **Aprendizaje no supervisado**

En este tipo de aprendizaje los datos no tienen etiquetas, de manera que los nuevos patrones se obtienen y clasifican en función de similitudes y/o diferencias con el resto de datos.

- **Aprendizaje por refuerzo**

En este sistema de aprendizaje, como en el no supervisado, no hay etiquetas, pero a diferencia de este, tras realizar un modelo o patrón, el sistema recibe retroalimentación para ajustar dicho patrón.

- **Visión Artificial**

La **visión artificial** (o más conocido por su nombre en inglés: Computer Vision) pretende **simular el comportamiento del ojo humano**, permitiendo adquirir, procesar, analizar y comprender las imágenes mediante la clasificación y segmentación de las mismas utilizando técnicas de machine learning.

- **Reconocimiento Facial**

El **reconocimiento facial** es una de las principales aplicaciones de la visión artificial, y consiste en **localizar y reconocer rostros de gente** en imágenes.

Este facial recognition tiene muchas aplicaciones, cada día más utilizadas, destacando las relacionadas con la **seguridad**, como la utilizada en los aeropuertos para llevar un control de los pasajeros, o en las cámaras públicas de las ciudades, para obtener información sobre atracos, ataques terroristas, etc.

También destacan los usos en reconocimiento y seguimiento de clientes por parte de empresas y tiendas, así como el reciente "desbloqueo facial" de los teléfonos inteligentes.

El proceso de este reconocimiento se compone de dos partes:

- **Detección de rostros**

En esta primera parte del proceso se **identifican los píxeles de la imagen que pertenezcan a un rostro**. Para realizar dicha identificación, hay varios algoritmos, aunque uno de los más utilizados y extendidos es el llamado "**Haar Cascade face detection**". Todas las versiones de este algoritmo se pueden encontrar en el siguiente [repositorio](#).

- **Reconocimiento de rostros**

En esta segunda parte se utilizan diversas técnicas para **evitar los problemas derivados de la calidad de la imagen** (iluminación, posición de la persona, cambios en los rasgos faciales, etc), y se obtiene como resultado una segunda **imagen más estandarizada**", la cual **comparamos** con las otras imágenes que teníamos preparadas para ser comparadas, y como resultado final obtenemos una de estas imágenes, que será la que contenga el rostro que queríamos reconocer en la primera imagen (o la que más se acerque).

- **Algoritmos de Reconocimiento Facial**

Los diferentes algoritmos que se barajaban para realizar el reconocimiento facial fueron:

- **Eigenfaces**

aaa...

- **Fisherfaces**

bbb...

- **Local binary patterns**

ccc...

Técnicas y herramientas

Aspectos relevantes del desarrollo del proyecto

Trabajos relacionados

Conclusiones y Líneas de trabajo futuras

Referencias bibliográficas

1. Inteligencia artificial
Author = Margaret Rouse
Title = Inteligencia artificial, o AI
Year = 2017
Link = [link](#)
2. Red neuronal
Author = Wikipedia
Title = Red neuronal artificial
Year = 2018
Link = [link](#)
3. Reconocimiento facial
Author = Jan Fajfr
Title = The basics of face recognition
Year = 2011
Link = [link](#)
4. Visión artificial
Author = Wikipedia
Title = Visión artificial
Year = 2017
Link = [link](#)
5. Haar cascade face detection
Author = OpenCV
Title = Face Detection using Haar Cascades
Year = 2018
Link = [link](#)

