DESARROLLO Y EVALUACIÓN DE TÉCNICAS DE VISIÓN ARTIFICIAL APLICADAS A UN PROBLEMA DE ROCONOCIMIENTO FACIAL

Raúl Trapiella Pino Autor

Dr. Carlos Cerrada Somolinos Director

Curso 2017-2018 – Convocatoria Septiembre

Máster Universitario en Investigación en Ingeniería de Software y Sistemas Informáticos

Itinerario: Ingeniería de Software. Trabajo de Fin de Máster Tipo A

DECLARACIÓN JURADA DE AUTORÍA DEL TRABAJO CIENTÍFICO, PARA LA DEFENSA DEL TRABAJO FIN DE MASTER

Fecha: 12/09/2018

Quién suscribe:

Autor: Raúl Trapiella Pino
D.N.I: 10906458L

Hace constar que es el autor del trabajo:

Título completo del trabajo.
DESARROLLO Y EVALUACIÓN DE TÉCNICAS DE VISIÓN ARTIFICIAL APLICADAS PROBLEMA DE ROCONOCIMIENTO FACIAL

En tal sentido, manifiesto la originalidad de la conceptualización del trabajo, interpretación de datos y la elaboración de las conclusiones, dejando establecido que aquellos aportes intelectuales de otros autores se han referenciado debidamente en el texto de dicho trabajo.

DECLARACIÓN:

- ✓ Garantizo que el trabajo que remito es un documento original y no ha sido publicado, total ni parcialmente por otros autores, en soporte papel ni en formato digital.
- ✓ Certifico que he contribuido directamente al contenido intelectual de este manuscrito, a la génesis y análisis de sus datos, por lo cual estoy en condiciones de hacerme públicamente responsable de él.
- ✓ No he incurrido en fraude científico, plagio o vicios de autoría; en caso contrario, aceptaré las medidas disciplinarias sancionadoras que correspondan.

Fdo.

Impreso TFdM05_AutorPbl. Autorización de publicación
y difusión del TFM para fines académicos

Autorización

Autorizo/amos a la Universidad Nacional de Educación a Distancia a difundir y utilizar, con fines académicos, no comerciales y mencionando expresamente a sus autores, tanto la memoria de este Trabajo Fin de Máster, como el código, la documentación y/o el prototipo desarrollado.

Firma del/los Autor/es

Resumen

El reconocimiento facial es una habilidad humana muy difícilmente simulada por sistemas informáticos. El reconocimiento consiste tanto en la detección de rostros como en la determinación de la persona.

El reconocimiento facial ofrece la posibilidad de aplicar mecanismos de seguridad en acceso a instalaciones o aplicaciones, el control de personas en áreas o lugares muy concurridos con sistemas de vigilancia, el reconocimiento de expresiones y estados de ánimo para sistemas de teleasistencia, y un largo etcétera.

***Palabras clave**: Visión, Reconocimiento Facial, Redes neuronales*

Contenido

Índice de imágenes

Índice de tablas

Índice de Códigos

Reconocimiento Facial

1 Introducción

Aunque hace años que existen mecanismos capaces de dotar de visión a las computadoras, esta ha estado limitada a la simple grabación de imágenes o al procesamiento de las mismas mediante operaciones costosas y por lo general lentas.

Con el aumento de la potencia de procesamiento, la incorporación de un mayor número de núcleos tanto a nivel de CPUs[1] como en las GPUs[2], y al uso de sistemas distribuidos se ha incrementado las capacidades de tratamiento de imágenes, lo que está posibilitando la aparición de servicios de detección y reconocimiento facial.

La capacidad de reconocer los rostros es un mecanismo muy potente en el campo de la seguridad, permitiendo su utilización como sensores biométricos para la autenticación de las personas en el uso de aplicaciones o el acceso a zonas restringidas. Como buen ejemplo funcional tenemos el sistema instalado en el aeropuerto londinense Stansted, llamado ABC[3] y que personalmente he podido probar como viajero, en el que el paso por aduana se puede realizar mediante el pasaporte electrónico. Unas cámaras detectan tu rostro y calculan y comprueban los valores biométricos incluidos en el pasaporte, permitiendo un acceso mucho más rápido y cómodo que el tradicional en el que un agente de seguridad a de validar individualmente a cada viajero.

El campo de la vigilancia y seguridad también se benefician de estos sistemas permitiendo la detección de personas y el registro de las mismas en edificios, zonas públicas o eventos multitudinarios. Esta capacidad de reconocimiento en tiempo real puede dotar a los sistemas de vigilancia de una capacidad nunca vista.

1.1 El problema del reconocimiento facial

Aunque cabe pensar que el reconocimiento de rostros es un problema sencillo debido a que es una acción cotidiana para el ser humano, la simulación de esta tarea en máquinas es un problema complejo. Las máquinas trabajan con dos estados o niveles de corriente y por tanto son sistemas binarios, aunque esto no implica que no se puedan representar tanto, valores enteros como decimales. El hecho de que los sistemas sean binarios hace que la obtención y por tanto representación de los valores del mundo real sea una interpretación o reducción, lo que implica otro nivel mayor de complejidad.

En este sentido, los sistemas informáticos almacenan las imágenes como matrices o colecciones de filas y columnas de números binarios divididos en varias capas o colores primarios.

[1] CPU: Unidad Central de procesamiento (Central Processing Unit)
[2] GPU Unidad de Procesamiento Gráfico (Graphics Processing Unit)
[3] ABC: Automation-Assisted Border Control

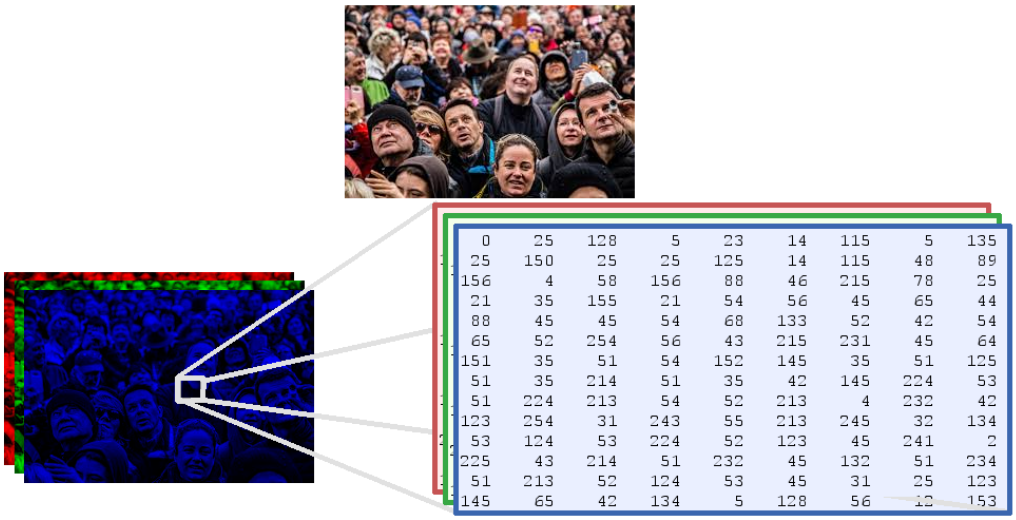


Imagen 1: Representación de imágenes en formato RGB

El reconocimiento de caras parte de estas imágenes representadas como matrices bidimensionales, e intentan extraer la información necesaria para determinar si los posibles rostros corresponden a caras conocidas o no.

El análisis de imágenes se reduce a la realización de un conjunto a veces elevado de operaciones matemáticas en los valores representados en matrices de colores, aunque por suerte, los sistemas informáticos son capaces de procesar operaciones numéricas de forma rápida y eficiente. El problema se encuentra en averiguar el cómo determinar que unos conjuntos de valores agrupados de cierta manera representan objetos que puedan ser de interés para su estudio posterior.

En el caso del procesamiento de vídeos nos encontramos con el problema adicional de la velocidad o flujo con el que las imágenes son tomadas. La tasa de frames[4] por segundos o FPS para videos normales supera los 25 FPS lo que implica la necesidad de procesar la imagen en un tiempo máximo de 0.04 segundos[5] para realizarla en tiempo real. En este sentido, y previendo FPS más altos, se han de encontrar técnicas que minimicen el área a calcular.

La detección de objetos por otro lado es un proceso complicado que requiere de mucha capacidad de procesamiento. No solo es necesario localizar regiones de interés, sino que se requiere de algún sistema de clasificación que sepa distinguir el tipo de objeto que acabamos de encontrar. En este sentido, la detección de imágenes es un caso particular de la detección de objetos, por lo que las técnicas existentes son posibles candidatos de uso para nuestra solución.

Y finalmente, una vez localizada y extraída la cara de nuestra imagen, deberemos encontrar un mecanismo de clasificación que nos permita asociar el rostro con alguno de los existentes en nuestra base de datos.

En rasgos generales, el problema se reduce a:

- Detectar caras en nuestra imagen.
- Determinar la identidad de la cara detectada.

[4] Frame: Imagen o fotograma obtenido de una secuencia de video.

[5] El tiempo se calcula con la inversa de la frecuencia. Al ser 25 veces por segundo la inversa 1/25 será de 0.04.

Y aunque el problema parezca sencillo debido a que consiste en solo dos pasos, cada uno de ellos es en particular un problema complejo.

1.2 Detección de rostros

Los métodos de detección de objetos, en nuestro caso particular detección de caras, tiene como objetivo dos tareas diferentes:

- Detección de objetos: esto consiste en determinar si en una imagen está presente algún objeto de los que el sistema sepa identificar.
- Localización: consiste en enmarcar el objeto dentro de un área para su procesamiento. Con esto obtendremos una zona de interés (ROI - Region of interest).

Tradicionalmente la detección de objetos se realiza en base a la obtención de características de la imagen, lo que permite centrar el interés en regiones concretas y de esta forma se puede procesar un menor volumen de información. Existen varios métodos que se basan en la obtención de características de la imagen:

- En base al color
- A la Forma
- Movimiento en vídeos
- O una combinación de todas las anteriores

Sin embargo, aunque los mecanismos anteriores ofrecen soluciones factibles, en la actualidad el reconocimiento de objetos se realiza mediante redes neuronales, que, aunque también se basen en la detección de características, estas no están prefijadas en la red, siendo estas aprendidas por la misma y permitiendo un continuo refinamiento del algoritmo interno.

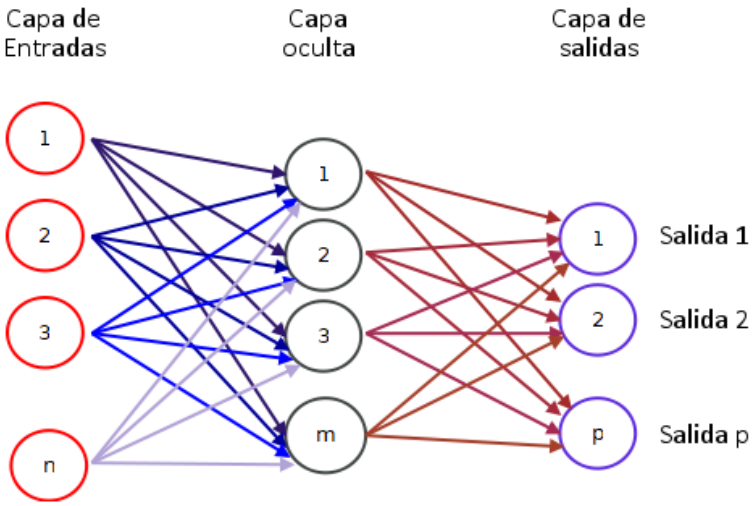


Imagen 2: Red Neuronal

A la hora de detectar un rostro, deberemos tener en cuenta que existen muchos detalles que pueden afectar negativamente a su detección, como por ejemplo la iluminación tanto en exceso como en defecto, la inclinación de la cabeza de las personas, objetos que pueden tapar parte de la imagen o generar sombras como gafas o sombreros e incluso expresiones como sonrisas, guiños, etc.

En las siguientes imágenes de Alan Turing obtenidas de Internet se pueden ver diferentes posiciones, expresiones y etapas de su vida, demostrando la complejidad de la identificación con imágenes históricas o antiguas.

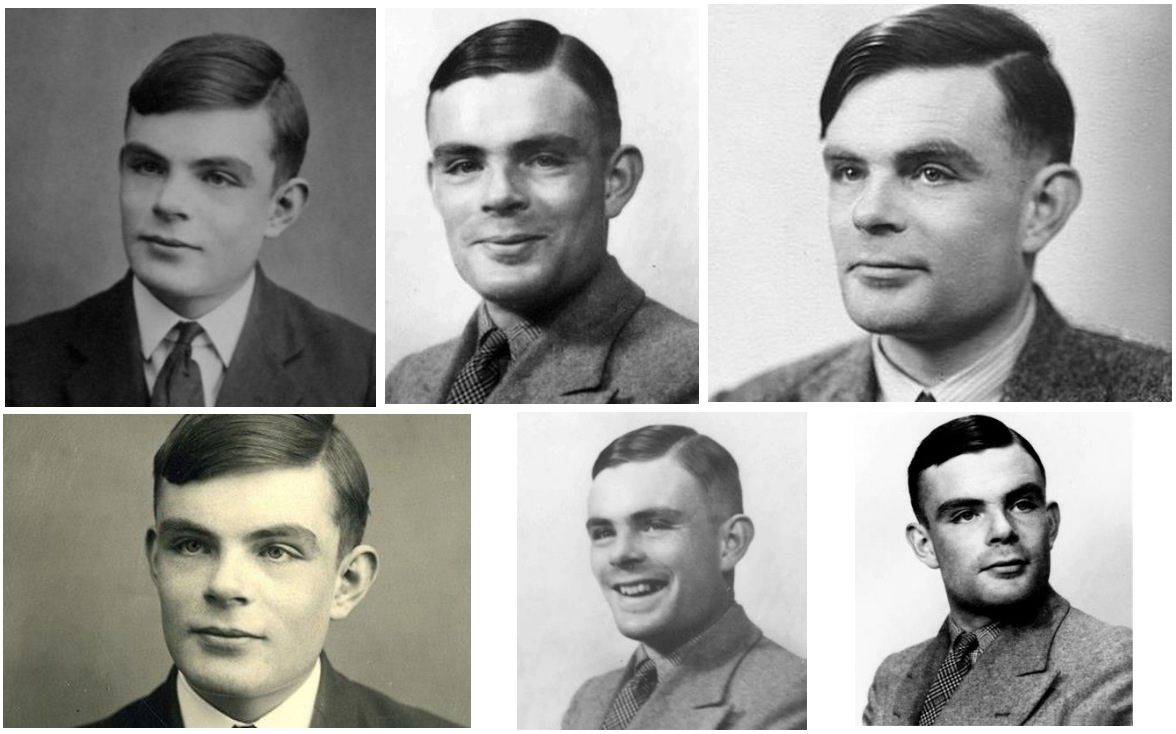


Imagen 3: Fotografías de Alan Turing en diferentes momentos y con diferentes poses

Otro ejemplo que demuestra las posibles complejidades es la de un rostro que puede estar parcialmente tapado por el cabello, lo que reduce la obtención de características y por tanto puede dificultar su clasificación.

Imagen 4: Fotografía en la que se ve un rostro parcialmente tapado por el pelo

1.3 Reconocimiento de rostros

El reconocimiento facial tiene como objetivo identificar de forma univoca a las personas presentes en una imagen.

Algunos de los problemas que tienen que hacer frente los métodos o algoritmos de reconocimiento facial son similares a los de la detección:

- Cambios de iluminación que pueden producir que algunos puntos de interés se vean alterados y por tanto generen falsas coincidencias.
- Cambios en el rostro debido a enfermedades, accidentes, tratamientos o cirugías estéticas o simples cambios en el maquillaje, barba o bigote o complementos también pueden afectar al reconocimiento.
- Reconocimiento sobre fotografías antiguas en las que las personas a identificar tienen una complexión distinta.
- Además de lo anterior, también se tiene que tener en cuenta el parecido entre padres/madres y sus hijas/hijos, hermanos, o gemelos.

Imagen 5: Gemelos idénticos

2 Técnicas existentes en la detección y el reconocimiento de rostros

2.1 Detección de rostros

El primer sistema de detección de rostros automático fue desarrollado en 1973 y desde entonces se han desarrollado nuevos y más eficientes sistema de detección. En la siguiente tabla se pueden ver los más importantes:

Año	Autor/es	Evento
1973	Kanade	First automated system
1987	Sirovich & Kirby	Principal Component Analysis
1991	Turk & Pentland	Eigenface
1996	Etemad & Chellapa	Fisherface
2001	Viola & Jones	Adaboost + Haar cascade
2005	Dalal & Triggs	HOG
2007	Naruniec & Skarbek	Gabor Jets

Tabla 1: Desarrollos de métodos de detección

- **Sistema de procesamiento de imágenes por computador y reconocimiento de rostros humanos (Kanade, 1974)**: Fue el primer sistema automático. Trabajaba sobre imágenes en tonalidades grises con 5 bits de nivel. El método extrae características de elementos como los ojos, la nariz, la boca, etc. El sistema está preparado únicamente para la detección de rostros que se encuentre mirando de forma frontal, no pudiendo detectar caras que se encuentren de perfil o que tengan objetos que tapen parte del rostro. (Kanade, Computer recognition of human faces, 1977)
- **Procedimiento de baja dimensión para caracterización de rostros humanos (Lawrence, y otros, 1987)**: Se basa en el teorema de Karhunen Loeve (Karhunen, y otros) y representa el rostro usando autovalores (valores propios) de la imagen haciendo uso de la simetría de los rostros. Solo es capaz de reconocer rostros frontales.
- **Similitud facial para reconocimiento facial (Turk, y otros, 1991)**: Se basa en los autovalores faciales y requiere poco tiempo para completar la detección realizando degradados en la iluminación, haciendo cambios de orientación y escalados en la imagen.
- **Análisis discretos para el reconocimiento de rostros (Etemad & Chellappa, 1997)**: Se basa en un análisis lineal sobre imágenes en escala de grises y solo se aplica a imágenes frontales.
- **Redes Neuronales para la detección de rostros (Rowley, y otros, 1998)**: Trabaja con imágenes en escala de grises y es capaz de detectar entre prácticamente el 77% y el 90% de los rostros en imágenes frontales. Estas redes pueden ser entrenadas para detectar rostros que se encuentran giradas.
- **Detección rápida de objetos usando características simples (Viola & Jones, 2001)**: Trabaja con imágenes grises y únicamente detecta imágenes frontales.
- **Histograms of Oriented Gradients for Human Detection (Dalal & Triggs, 2005):** Este método obtiene características en los histogramas del gradiente de una

imagen basada en rejillas.

- **Detección de rostros usando inyecciones Gabor discretos (Hoffmann, Naruniec, Yazdani, & Ebrahimi, 2008)**: Se basa en el filtro de Gabor. Este método detecta los posibles cambios de la imagen al cambiar los puntos de vista para extraer información. Cada inyección es un cambio de ángulo de la función sinusoidal, lo que implica una mayor complejidad computacional respecto a métodos como el de Viola-Jones.

Como se puede apreciar, la mayoría de los sistemas de detección de rostros utilizan el concepto de autovalores de la cara o eigenfaces.

2.1.1 Clasificación en cascada

Dentro de las técnicas de detección de rostros basados en características, la descrita por (Viola & Jones, 2001) consigue dar solución al problema mediante la búsqueda parcial en la imagen, y realizando pequeñas clasificaciones cada vez más precisas para ir descartando zonas que no sean de interés.

Con el método descrito más en detalle en el Anexo I , apartado 8.2 se obtendrán los clasificadores más adecuados para la detección de rostros. Sin embargo, tanto en el proceso de aprendizaje como en el posterior análisis de las imágenes, se requiere un considerable número de operaciones debido a que cada clasificador requiere ser pasado en la imagen por cada uno de los píxeles.

Para minimizar el número de operaciones se utiliza el método de la imagen integral, lo que consigue una reducción de las operaciones de suma.

Este método, aunque es muy eficaz a la hora de detectar objetos dentro de una imagen, tiene una serie de limitaciones que serán descritas en el apartado 4.1 “Sistema de Viola – Jones”.

2.1.2 Sistemas de autoaprendizaje

Aunque el sistema anterior tiene un componente de entrenamiento, la fuerte dependencia de las formulas no permiten un verdadero sistema de aprendizaje.

El aprendizaje automático consiste en la capacidad de nuestro sistema para aprender o auto-configurarse de tal forma que será capaz de detectar o reconocer rostros con más eficiente ante cada imagen de pruebas introducido. Esta capacidad puede centrarse tanto en los elementos de clasificación, como en los propios procesos de detección.

Dentro de estos sistemas de aprendizaje se encuentra:

- En menor medida tenemos el sistema utilizado por Jones y Viola mediante el uso de Adaboost y los clasificadores en cascada.
- Redes neuronales.
- Redes neuronales convolucionales (CNN[6]).

2.2 Reconocimiento de rostros

Como ya se ha comentado anteriormente, el reconocimiento de rostros tiene como principal objetivo el determinar la identidad de un rostro, o al menos un nivel de probabilidad o coincidencia con rostros semejantes y disponibles en una base de datos. Además de lo anterior,

[6] CNN - Convolutional neural network

hay ciertos estudios que utilizan el reconocimiento facial no solo para la identificación de la persona, sino que persigue identificar rasgos como el sexo, raza, estado de ánimo e incluso intuir la inclinación sexual de individuos.

Como ejemplo de lo anterior, se ha publicado en febrero del 2018 un estudio en el que dos ingenieros de la Universidad de Stanford aseguran haber desarrollado un sistema de reconocimiento facial que clasifica a personas homosexuales y heterosexuales. (Kosinski & Wang, s.f.). Esto viene a demostrar como la tecnología no solo se utiliza para el que se ha diseñado, sino que puede ser alterado para otros medios que pueden ser más o menos éticos.

Aunque hay varios estudios y trabajos relacionados con el reconocimiento de rostros, las técnicas más destacas son:

- PCA: Como se ha expuesto en el apartado anterior, este método utiliza los eingenface para la clasificación de rostros (apartado 8.1).
- LDA[7]: Este método consiste en la clasificación de rostros basándose en una base de datos y un sistema de entrenamiento estadístico. El sistema hace uso de la discriminante lineal de Fisher (Wikipedia, Ronald Fisher, s.f.) por lo que autores como (Etemad & Chellappa, 1997) lo han denominado fisherface.
- EBGM[8]: Este método se basa en la determinación de características basadas en cambios de iluminación, poses o expresiones dentro del rostro. (Wiskott, Fellous, Krüger, & von der Malsburg, 1999).
- RNN[9]: Son una clase de redes neuronales diseñadas para proceso de entradas secuenciales en las que se dispone de la capacidad de recordar información anterior (denominada memoria) y que pueden ser utilizadas para predecir un resultado en base a dichas entradas.

2.2.1 Eigenfaces

Los Eigenface (Wikipedia, Eigenface, s.f.) o valores propios de una cara, son los vectores no nulos que al aplicarles su operador lineal[10], dan un múltiplo escalar de sí mismo. Este escalar λ se denomina valor propio o eigenvalor[11].

La idea que hay detrás de los autovalores, es la de obtener un mínimo de características o parámetros que pudieran agrupar un conjunto de caras. Mediante el análisis de componentes principales PCA[12] se consiguen un conjunto de imágenes semejantes a caras llamadas *eigenpictures* de menor tamaño que las originales. Este término fue rebautizado por (Turk & Pentland, 1991) que lo denominaron como se conoce actualmente por *eigenface*s ya que los autovalores obtenidos se aplican a las propias caras y no solo a los píxeles de las imágenes.

[7] LDA – Linear Discriminant Analysis

[8] EBGM – Elastic Bunch Graph Matching – Coincidencias gráficas en grupos elásticos

[9] RNN - Recurrent Neural Network

[10] Un operador lineal es una función matemática que se aplica entre dos espacios vectoriales y que tiene como restricción el mantener tanto el operador suma, como el operador multiplicación por un escalar. (Wikipedia, Aplicación lineal, s.f.)

[11] Eigen es una palabra alemana que se puede traducir en español por propio. También es conocido como inherente, característico, auto-, perteneciente, etc.

[12] PCA: Principal Components Analysis

A continuación, se explicará brevemente el método de cálculo de Eigenfaces propuesto por Matthew Turk y Alex Pentland para poder extraer algunas conclusiones, el texto completo se puede obtener en la siguiente dirección http://www.face-rec.org/algorithms/pca/jcn.pdf.

2.2.1.1 Cálculo de eigenfaces

Aunque esta técnica está ideada para la clasificación de rostros y por tanto la identificación de los mismos, la posibilidad de identificar características propias de un rostro, también la habilita como técnica de detección. Eso sí, con menor precisión que otras técnicas ideadas exclusivamente para esta tarea.

El cálculo consiste en hallar el vector propio de la matriz de covarianza de la imagen de tal forma que este quede caracterizado por dicho vector.

Supongamos que la colección de imágenes de entrenamiento es de tamaño N, y que X_i es el vector o array que representa a la imagen de tamaño WxH. Cada pixel está representado por 8 bits (valores de 0 a 255) y mediante los cálculos del PCA descritos en el Anexo I apartado 8.1 podemos obtener los eigenfaces de nuestras imágenes.

El método anterior consigue reducir el conjunto de N imágenes a N-1, ya que con estos se puede obtener la cara eliminada. Este principio se puede aplicar de tal forma que la reducción del conjunto se consigue hasta un valor M<N y representado por los autovalores con mayor valor propio.

En general, con un conjunto de imágenes lo suficientemente grande se es posible generar un conjunto reducido que represente con bastante probabilidad cualquier cara que podamos analizar. En otras palabras, el conjunto obtenido en el análisis contiene las características principales que determinan que una imagen proporcionada es una cara, lo que posibilita un mecanismo de detección y reconocimiento válido.

Sin embargo, las fuertes restricciones que hay que aplicar a las imágenes para que el método sea eficiente lo convierten en un problema para el análisis en tiempo real. Estas restricciones son necesarias para la correcta sustracción de la media aplicada en la formula (2), y que consisten en preparar inicialmente nuestra imagen para que los ojos, nariz y boca coincida en tamaño y posición con las imágenes de muestra. Además, las imágenes han de tener una expresión similar para no confundir al sistema de análisis.

3 Redes Neuronales

El concepto de red neuronal artificial es un modelo conceptual basado en la biología humana. Esta red está compuesta de nodos denominados neuronas artificiales que se conectan con las neuronas adyacentes para activarlas y producir una respuesta.

Existen varios tipos de redes configuradas de diferente manera y diseñadas para realizar un determinado trabajo, aunque en general todas disponen de 3 agrupaciones con diferentes niveles o capas de neuronas:

- **Capa de entrada**: En esta capa se encuentran las neuronas (nodos) que obtienen los valores de entrada.
- **Capa oculta**: Es la capa encargada de resolver el problema y la que tiene la capacidad de aprendizaje. Esta capa se puede encontrar a su vez dividida en otros niveles o capas que consiguen componer un sistema más complejo, pero a su vez con mayor capacidad de aprendizaje y por tanto con una mayor eficacia en el resultado.
- **Capa de salida**: La capa de salida contiene las neuronas que muestran el resultado del trabajo realizado por la red. En la mayoría de las redes la salida tiene un nodo o neurona por cada uno de los elementos de clasificación disponibles y cuantificado mediante una probabilidad o nivel de activación.

El número de neuronas contenidos en cada capa no tiene porqué ser el mismo, pero por regla general el número de neuronas contenido en cada capa dentro de la capa oculta es inferior al de entrada. En el Anexo I apartado 8.4 se encuentra la formulación general aplicada en cada una de las neuronas contenidos en la red neuronal.

Basándonos en la formula (16) y suponiendo que nuestra red dispone de n capas con k nodos por capa, podemos ver que el número de cálculos definidos en la red será:

$$\sigma\left(\begin{bmatrix} w_{0,0} & w_{0,1} & \dots & w_0,n \\ w_{1,0} & w_{1,1} & \cdots & w_1,n \\ \vdots & \vdots & \ddots & \vdots \\ w_{k,0} & w_{k,1} & \cdots & w_{k,n} \end{bmatrix}\begin{bmatrix} a_{0,0} \\ a_{1,1} \\ \vdots \\ a_{k,n} \end{bmatrix} + \begin{bmatrix} b_0 \\ b_1 \\ \vdots \\ b_n \end{bmatrix}\right) \quad (1)$$

En la formula anterior se puede observar que, aunque existe un volumen elevado de operaciones, todas ellas son básicas (sumas y multiplicaciones), sin embargo, la función sigmoide es muy costosa computacionalmente y ha obliga a buscar una solución más adecuada siendo sustituida en las redes neuronales modernas por la función de *unidad lineal rectificada (ReLU[13]):*

$$ReLU(a) = \max(0, a) \quad (2)$$

3.1 Redes neuronales convolucionales (CNN)

Este tipo de redes descritas por (le Cun, 1989) están diseñadas para procesar datos que tienen una topología de rejilla definida. Los ejemplos más comunes son los datos en series temporales y los datos de imagen. Su nombre viene originado a raíz del uso de una operación matemática llamada *convolución*, descrita en el Anexo I apartado 8.5.

[13] ReLU – Rectified Linear Unit

El objetivo de cada nodo dentro de las capas de convolución consistirá en el procesado de una región concreta de la imagen lo que permitirá que el nodo o neurona pueda extraer una característica determinada.

Como se muestra en la Tabla 3, existen múltiples máscaras aplicables a nuestra imagen de las que algunas están especializadas para la obtención de bordes, otras para el enfoque, el realzado, etc., lo que puede dar a entender que un punto importante de la configuración es la elección de la máscara más adecuada en el procesamiento de la imagen en esa neurona. Sin embargo, estas máscaras se encuentran incluidos dentro de los parámetros de configuración aprendidos por la red. En otras palabras, las neuronas que procesan una convolución irán adaptando cada máscara según los valores de entrenamiento con el objetivo de extraer características de activación lo más precisas posibles para el trabajo general de la red.

Una visión más global de la red sería que cada capa, y concretamente cada neurona, extrae poco a poco pequeñas características que son acumuladas formando un todo. Por ejemplo, en el reconocimiento de caras, una posible interpretación de lo que realiza internamente la red es la de ir obteniendo pequeños bordes en las primeras capas, obteniendo formas en sucesivas capas y finalmente detectando elementos más complejos como ojos, nariz, boca, etc.

De todo esto se encargan cada una de las máscaras de convolución definidas en cada neurona, con la salvaguarda de que se han de definir los objetivos generales de cada capa. Siendo por ejemplo la detección de bordes, detección de formas, etc.

Además de las capas de convolución, las redes han de definir otros tipos de capas que permiten integrar la información obtenida en capas superiores.

En este sentido tendríamos los Pooling[14], las capas de activación (signoides) y las capas de conexión completas (fully-connected FC).

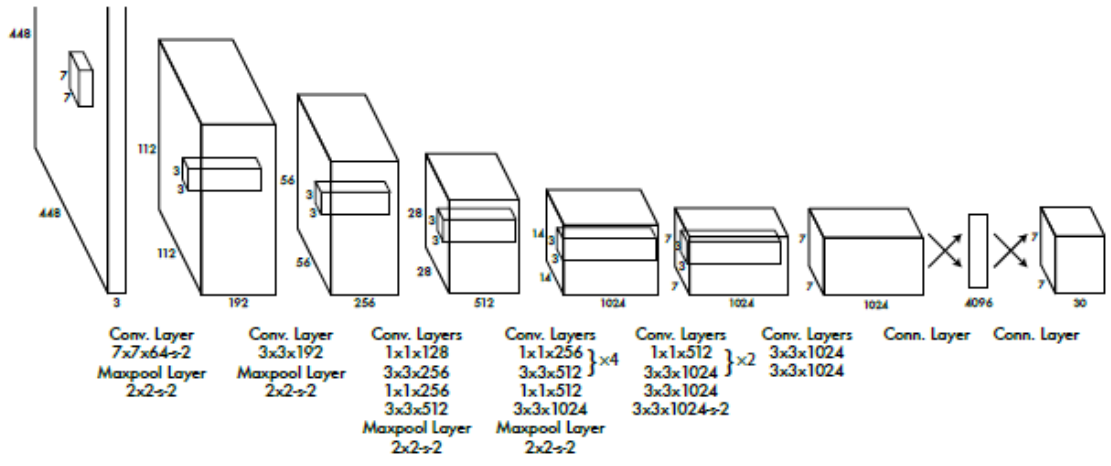


Imagen 6: Arquitectura propuesta por YOLO

3.2 Redes neuronales recurrentes (RNN)

Las RNN son redes diseñadas para trabajar con datos serializados y con el objetivo de predecir una salida en función de los datos de entrada.

Tomemos por ejemplo la siguiente secuencia como datos de entrenamiento:

[14] Pooling: Agrupamiento

1, 2, 3, 4, 3, 2, 1, 2, 3, 4, 3, 2, 1

Si configuramos la red con una salida y le introducimos la siguiente secuencia [1,2,3] el valor devuelto deberá ser 4.

Para que la red neuronal recurrente pueda funcionar, se han de realizar ciertos cambios en las neuronas para dotarlas de memoria y de algún mecanismo de detención de propagación.

Como se ha definido en el apartado de neuronas, cada una de ellas se reinicia en las etapas de cálculo, siendo el objetivo de las SRN o de las RNN la búsqueda de una relación en el tiempo y de esta manera ser capaces de anticiparse a las entradas.

4 Análisis de sistemas actuales

En los apartados anteriores se han expuesto una serie de técnicas disponibles tanto para la detección como el reconocimiento facial. El objetivo de este trabajo de investigación consiste en la evaluación y desarrollo de técnicas de reconocimiento facial, por lo que gran parte del trabajo de investigación ha consistido en el desarrollo de una serie de aplicaciones que permitan la evaluación de estas técnicas. He de destacar que todas las imágenes y parte de los conceptos expuestos en este trabajo están generadas o disponibles mediante la aplicación desarrollada lo que ha implicado un alto grado de complejidad a la solución.

El código está desarrollado en C++ y se ha tenido especial atención en que fuese multiplataforma, y debido al interés que puede tener para otros trabajos de investigación el producto es ampliable mediante el desarrollo de módulos y por tanto dotándolo de flexibilidad y escalabilidad. Todo el código y parte de los ficheros utilizados en este trabajo se encuentra en un repositorio público de github (https://github.com/raytrapi/Reconocimiento-Facial/). En el Anexo II se indica como descargar y compilar la aplicación.

Las fotografías utilizadas para las diversas pruebas han sido obtenidas de repositorios de imágenes libres como Pixabay o el propio Google mediante el filtrado de imágenes con reutilización.

En algunos casos se han obtenido recursos de terceros que, aunque son de uso libre se indicará el autor de los mismos y la dirección de descarga.

4.1 Sistema de Viola – Jones

Aunque su implementación no es compleja, para el desarrollo de la herramienta de análisis se ha utilizado la librería de terceros (OpenCV team, s.f.)[15] la cual ofrece tanto una implementación del algoritmo como una serie de clasificadores entrenados para la detección de rostros y la posibilidad de generar clasificadores específicos para intentar mejorar los disponibles por el sistema.

La clase utilizada para el reconocimiento de objetos es *cv::CascadeClassifier* incluida en la librería *objdetect*.

```
cv::CascadeClassifier faceCascade;

faceCascade.load("haarcascade_frontalface_default.xml");

int minimoW=10;

int minimoH=10;

faceCascade.detectMultiScale(*imagenGris, caras, 1.1, 3, 0 |
CV_HAAR_SCALE_IMAGE, cv::Size(minimoW, minimoH));
```

Código 1: Detección de rostros con CascadeClassifier de OpenCV

[15] OpenCV: Open Source Computer Vision Library

Como se puede apreciar en el ejemplo anterior, el código obtendrá una serie de regiones que cumplan los requisitos definidos en el clasificador de cara frontal y cuyo tamaño sea superior a 10x10px.

4.1.1 Clasificadores Haar

OpenCV incluye varios clasificadores Haar-like para la detección de ojos, caras tanto de frente como de perfil, narices, bocas, cuerpos completos, sonrisas, etc.

En nuestros ejemplos utilizaremos los siguientes clasificadores:

- haarcascade_frontalface_default.xml
- haarcascade_eye.xml
- haarcascade_mcs_mouth.xml
- haarcascade_mcs_nose.xml
- haarcascade_profileface.xml

4.1.2 Configuración

El uso de clasificadores Haar requiere de algunos parámetros de configuración que tienen como objetivo el descartar aquellas zonas de la imagen que podrían no ser relevantes para la detección de objetos.

Veremos que estos valores deberán ser definidos con cuidado ya que son relevantes a lo ahora de obtener tanto *falsos positivos* como *falsos negativos*.

4.1.3 Ejemplos

Para no extender innecesariamente la documentación solo se presentarán aquellas pruebas más significativas de la batería realizada.

La siguiente imagen muestra una fotografía de una chica en primer plano sin elementos que puedan estorbar en el reconocimiento de la cara y con un fondo desenfocado para evitar falsos negativos.

Imagen 7: Imagen de chica de tamaño 1024x683 pixeles

Tras pasar el método de detección con un tamaño fijo mínimo de imagen de 100x100 pixeles obtenemos 3 regiones de interés marcadas en violeta.

Se ha recortado la imagen para permitir una mejor presentación de los resultados.

Imagen 8: Obtención de 3 regiones de interés tras pasar el método de Viola-Jones

El resultado obtenido tras la prueba es la de 3 regiones que podrían ser consideradas como cara frontal. Como se puede deducir, solo la etiquetada como 2 es realmente una cara, pero si analizamos con detenimiento el resultado podemos intuir que uno de los problemas podría estár originado por el tamaño escogido como valor mínimo. Una posible solución sería cambiar el tamaño mínimo de 100x100 a 200x200, sin embargo, esto produciría problemas que veremos más adelante.

La siguiente prueba consiste en una fotografía en la que se muestra una chica con el pelo largo que le tapa parte del ojo izquierdo.

Imagen 9: Chica con parte del rostro tapado por el pelo

Imagen 10: Resultado válido sobre una fotografía simple con elementos perturbadores

Como se puede observar en el resultado de la detección de rostros, el método ha encontrado una posible coincidencia de tamaño 330x330 en la imagen que es de tamaño 1024x683 pixeles.

El hecho de que la fotografía se encuentre ligeramente tapada por el pelo no afecta a la detección del rostro por lo que el método parece adecuado para la resolución del problema.

Hasta ahora se han proporcionado imágenes simples en el sentido de que solo se disponía de un rostro a localizar en la imagen y de forma que la persona se encontraba en frente de la cámara.

En la siguiente prueba se ha utilizado una imagen con varios rostros que no se encuentran mirando directamente a la cámara y con el añadido de que la persona que ocupa la posición central tiene la cara pintada.

Imagen 11: Grupo de personas y con elementos que dificultan la detección

Imagen 12: Resultado de detectar caras en una fotografía grupal

En este caso el sistema solo ha podido localizar 2 rostros, pero no se ha podido encontrar la cara más cercana al observador.

Uno de los detalles que se extra es que las zonas localizadas tienen tamaños de 101x101 y 106x106 lo que puede inducir a pensar que quizás para esta imagen de 968x651 pixeles el tamaño mínimo escogido de 100x100 para la localización de rostros puede ser muy grande. Otro detalle es que los rostros encontrados están mirando al frente y las no localizadas se encuentra de perfil.

Para intentar mejorar la detección de caras en esta fotografía podemos incluir la detección de imágenes de perfil y reducir el tamaño de la imagen mínima a valores más pequeños, por ejemplo, 50x50.

Imagen 13: Detección de rostros de frente + perfil con tamaño de 50x50

Tras hacer las modificaciones el sistema ha detectado 5 posibles caras frontales y 2 de perfil (marcadas en azul celeste). Sin embargo, aunque se ha encontrado una nueva cara (en total son 3), se han generado 2 falsos negativos debido a la reducción de tamaño mínimo.

Si seguimos bajando el tamaño a 10x10 se obtienen 6 caras de frente y 4 de perfil subiendo a 5 caras localizadas y 2 falsos negativos.

Imagen 14: Detección para tamaño mínimo de 10x10

4.1.4 Restringiendo el método

Como se ha visto en las pruebas anteriores la aparición de falsos negativos aumenta si reducimos el tamaño mínimo para considerar el objeto detectado como rostro. Sin embargo, en ocasiones es necesario reducir este tamaño ya que los rostros se pueden encontrar muy alejados del observador y por tanto su tamaño será reducido.

Respecto a este tamaño no tiene porqué ser fijo, pudiendo tomarse como valor una proporción de la imagen original. Otro detalle importante es que las imágenes de la cara no suelen ser cuadraras por lo que sería interesante que el tamaño mínimo tampoco lo sea, siendo más adecuado tomar una **proporción 5 a 8**. En el caso de la imagen de 10x10 sería más correcto que fuese de 10x16

Otro detalle interesante consiste en restringir la detección de caras solo a aquellas que además incluyan en su interior la detección de otros elementos como ojos, nariz o boca.

Para evitar una sobrecarga en el proceso de detección de estos elementos se puede restringir su búsqueda solo a aquellas zonas en las que el sistema ha conseguido localizar caras. Sin embargo, y al igual que para la detección de rostros, se ha de fijar un tamaño mínimo para cada una de las regiones a localizar.

En las siguientes pruebas se aplicará tanto la localización de rostros frontales como la detección de ojos, narices y bocas.

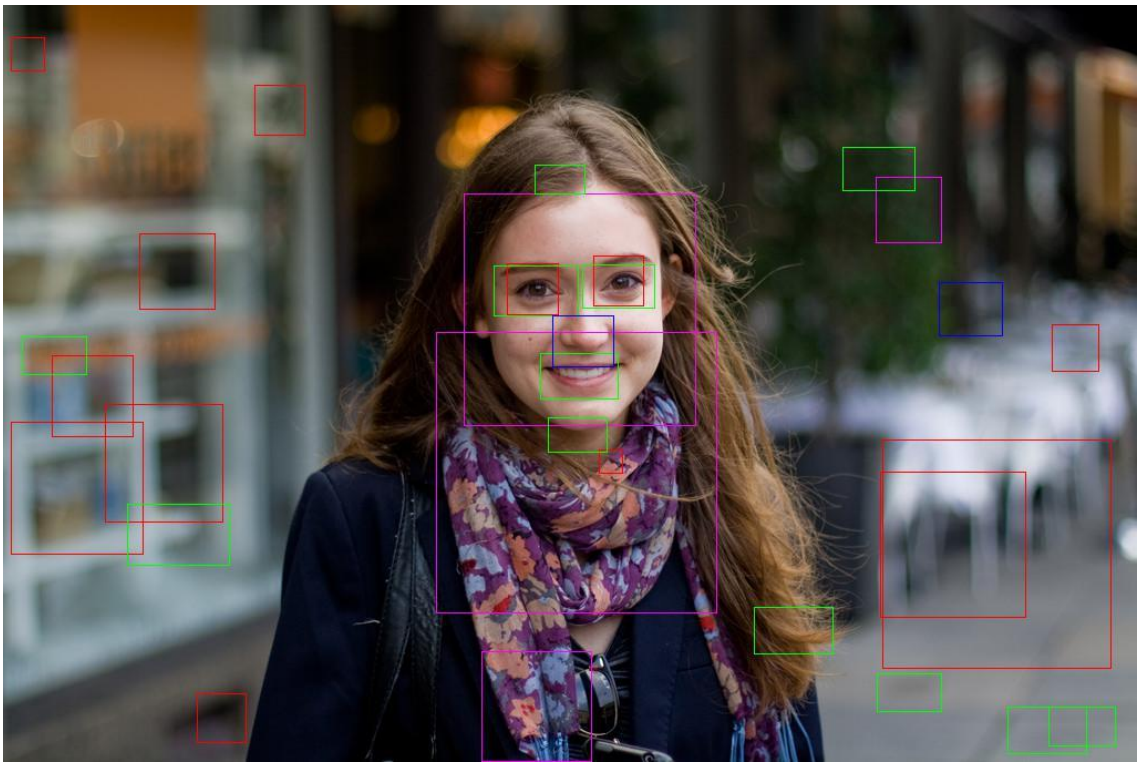


Imagen 15: Localización de rostro, ojos, nariz y boca sin limitar ni zona ni tamaño

En la prueba anterior no se ha restringido ni la zona ni el tamaño en los que se ha de localizar las diferentes partes, por lo que el resultado ha sido:

- 4 caras de frente
- 0 caras de perfil
- 13 ojos, color rojo
- 12 bocas, color verde
- 2 narices, color azul

Para comenzar a acotar las zonas y fijar el tamaño de cada una de las partes que pueden ser localizadas en un rostro, se deberá realizar un pequeño estudio de fisionomía. Obviamente no es objeto de este trabajo de investigación el estudiar en detenimiento los detalles inherentes a la morfología humana, por lo que se ha diseñado una rápida adaptación al código que permitirá mostrar una rejilla de ayuda para la obtención de dichas proporciones y zonas.

El tamaño de cada celda se ha definido en base a información obtenida tras una rápida investigación y que en promedio fijan la proporción ancho x alto del rostro humano en 5 a 8.

Una vez obtenida esa proporción tan solo ha sido necesario diseñar una rejilla que visualmente divida el rostro localizado por el método de Viola-Jones en esa proporción definida.

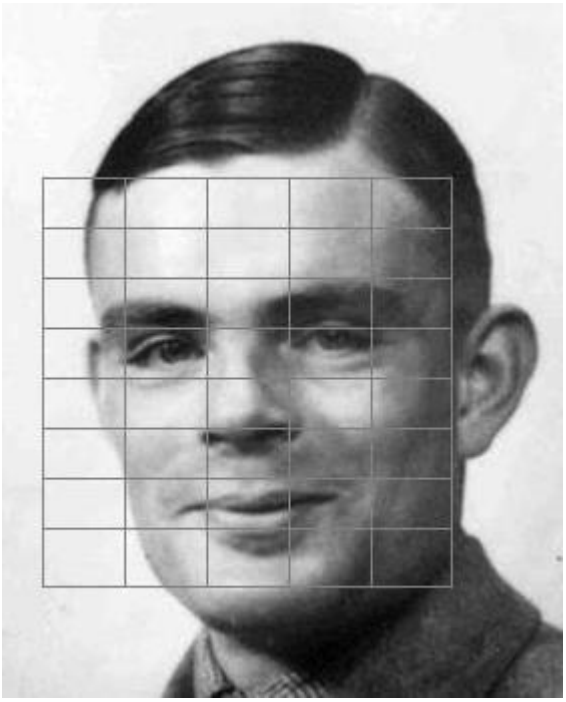


Imagen 16: Proporción de rostro

En la imagen anterior se puede observar que las zonas de interés en los ojos ocupan una zona de 2x2 en la zona superior media del rostro (marcado en rojo). De igual forma la nariz ocupa una zona de 1x2 marcado en azul y por último la boca se ha definido como una matriz de 3x3 coloreado en verde.

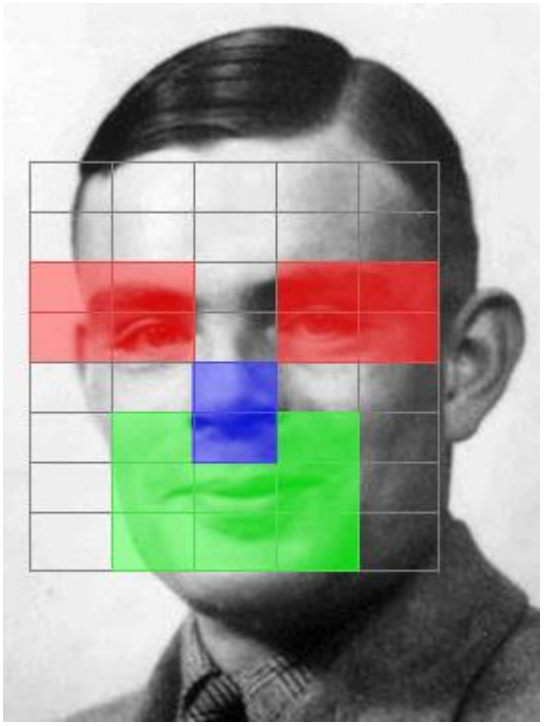


Imagen 17: Zonas de interés para localización de ojos, nariz y boca

En la siguiente imagen podemos comprobar que, tras realizar la detección de objetos mediante los clasificadores de ojos, nariz y boca en la imagen anterior, se han localizado los

elementos en las zonas fijadas como posibles ubicaciones. Esto demuestra que las zonas elegidas pueden ser unos buenos candidatos.

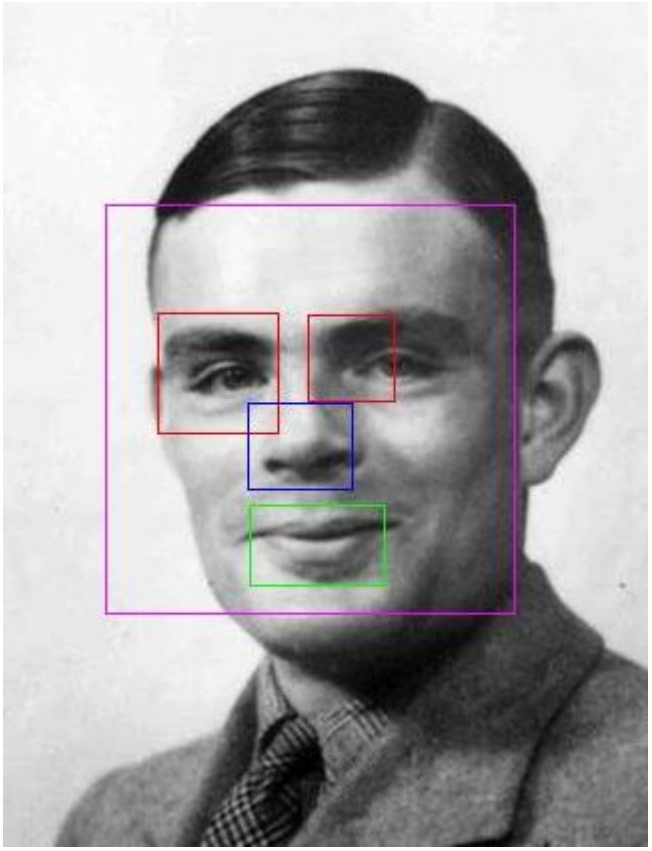


Imagen 18: Detección dentro de la zona seleccionada

Una vez definida la zona de detección, podremos definir un sistema que solo seleccione aquellas caras que además tengan al menos un ojo y o bien una nariz o una boca, pero no más de cada una de ellas.

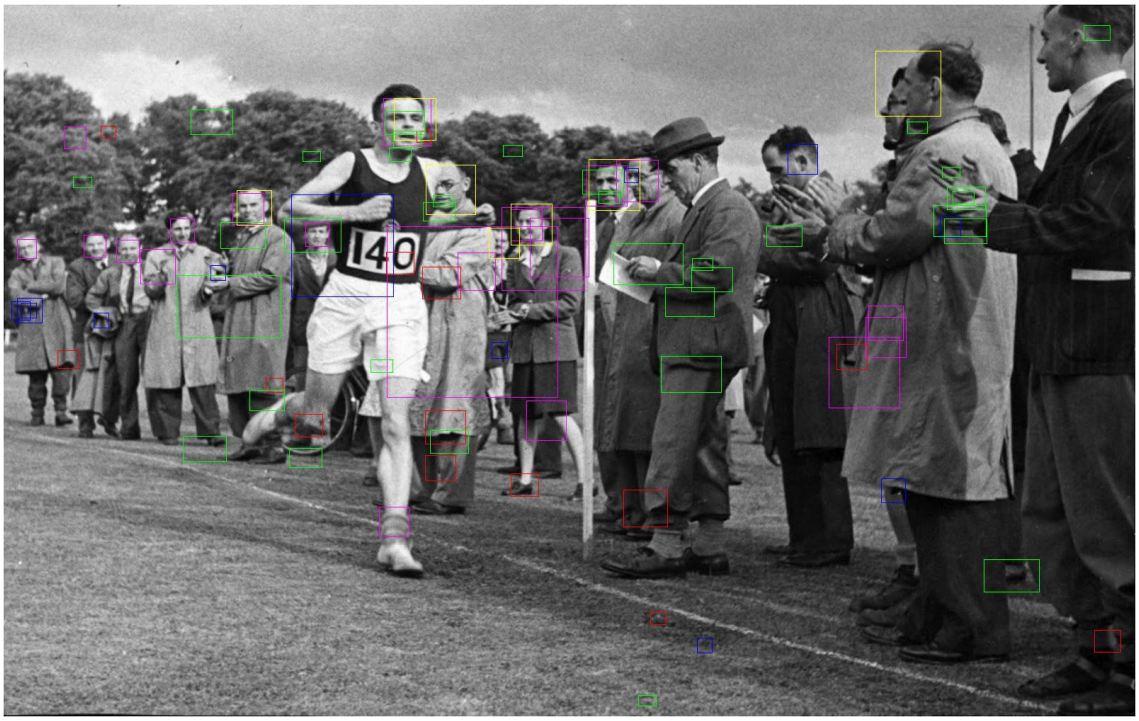


Imagen 19: Detección de objetos sueltos

En la imagen anterior se han detectado:

- 21 caras frontales.
- 7 caras de perfil (amarillo).
- 12 ojos.
- 14 narices.
- 34 bocas.

Sin embargo, si acotamos la detección de parte del rostro, se reducen a:

- 21 caras frontales.
- 7 caras de perfil (amarillo).
- 0 ojos.
- 1 narices (rojo).
- 3 bocas (verde).

Imagen 20: acotación de partes del rostro

Sin embargo, si decidimos seleccionar solo aquellas que cumplan la regla de contener al menos un ojo y no más de dos, una única nariz y una única voca, no se cumple ninguno de los criterios.

Como veremos en la imagen siguiente, para una fotografía con mejór calidad el resultado es más positivo.

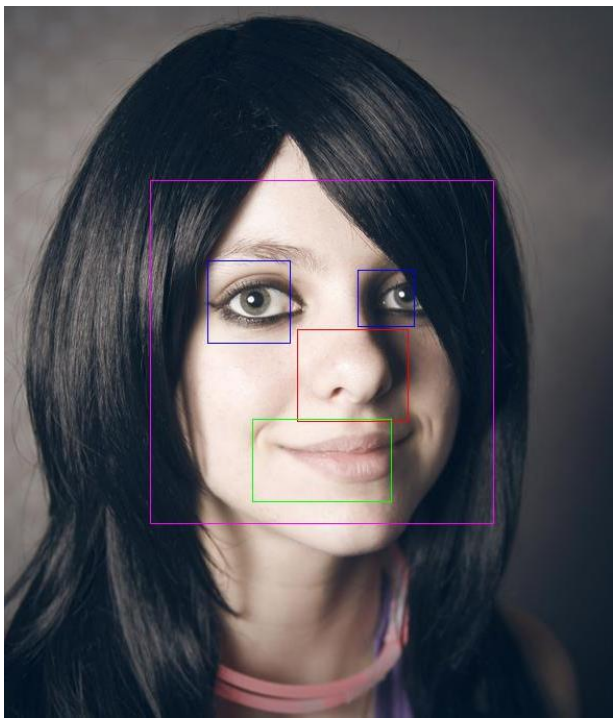


Imagen 21: Detección efectiva del rostro

Sin embargo, la incorporación de detección de objetos dentro de un área demarcado produce una serie de problemas originados con el clasificador y el tamaño mínimo y máximo de los objetos detectados. En la siguiente imagen se puede apreciar como el número de ojos y bocas detectados son superiores a los reales.

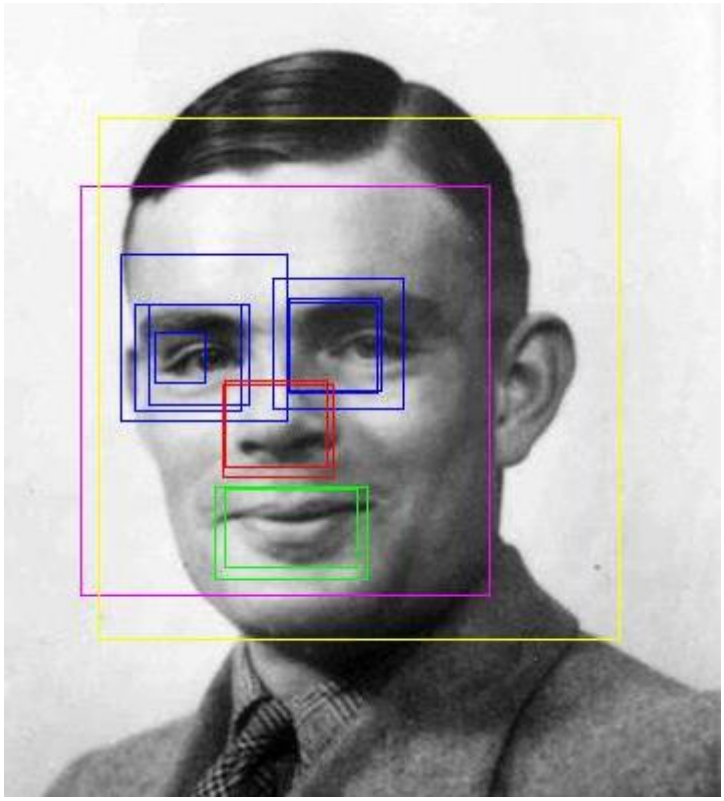


Imagen 22: Detección de partes con tamaño 0

En esta ocasión el problema estriba en que el tamaño mínimo escogido para las partes es 0, lo que implica la localización de múltiples partes que además son incluidas unas dentro de las otras. En este caso surge un nuevo problema que consiste en la intersección sobre uniones (IoU[16]) explicada en el punto 8.10 del Anexo I.

En este caso en concreto se realizarán las uniones de caras tanto frontales como de perfil y posteriormente de cada una de las partes. Como se puede apreciar en la Imagen 22, algunas partes (como los ojos) disponen de zonas incluidas unas dentro de otras, lo que se resolverá eliminando la zona exterior si esta obtiene un IoU inferior a 0,5.

[16] IoU: Intersection over Union

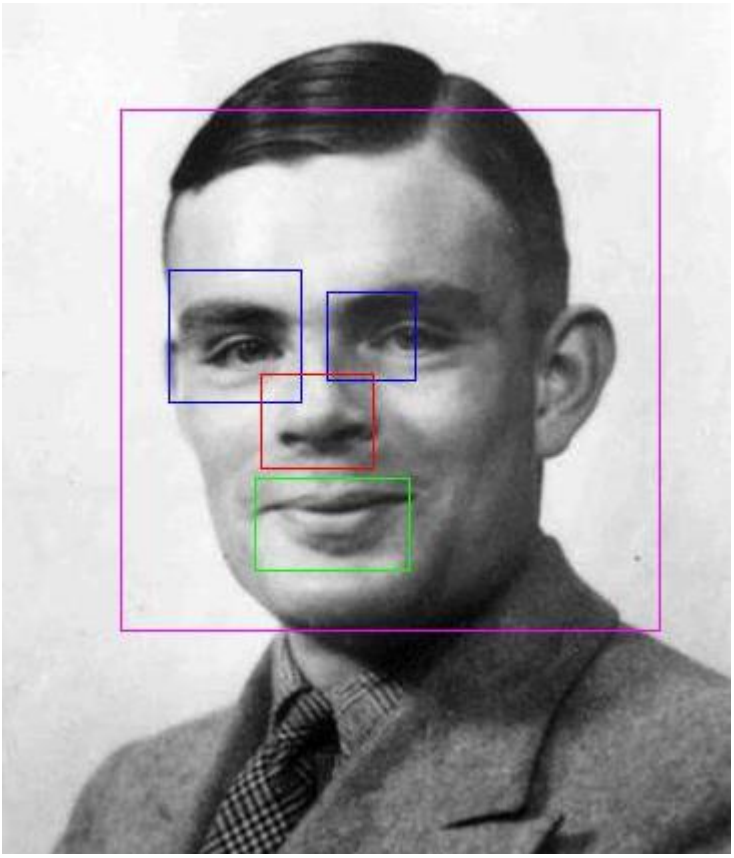


Imagen 23: Filtrado de rostros mediante IoU

4.2 HOG

Tal y como se indica en el artículo publicado en el 2005 (Dalal & Triggs, 2005), los autores del método se dieron cuenta que tras la obtención de bordes y gradientes la realización de un histograma de orientaciones de estos gradientes permitía la obtención de características que ayudaban a la detección rápida de objetos. En el Anexo I, apartado 8.7 se muestra un detalle más amplio sobre la obtención de histogramas.

Una vez obtenido el histograma se podrá realizar la obtención de objetos mediante la comparación de patrones obtenidos en un proceso de aprendizaje. La obtención de estos patrones se puede realizar con cualquier sistema de clasificación, como SVM[17], Redes neuronales, etc.

En la siguiente imagen se puede ver el proceso completo para la detección de objetos:

[17] SVM – Support Vector Machines – Máquinas de soporte vectorial

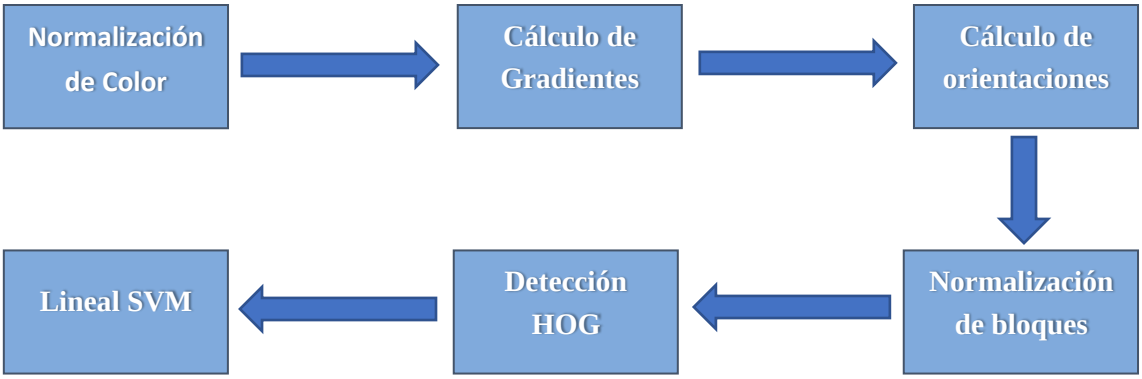


Imagen 24: Método HOG

A la hora de implementar el método HOG, se ha de tener en cuenta varios parámetros y procesos intermedios que pueden afectar a las características de la imagen. Como ejemplo se muestra la representación de dos procesos de HOG, el primero con una normalización L2, y otra con F-HOG.

Imagen 25: Foto original prueba HOG 1

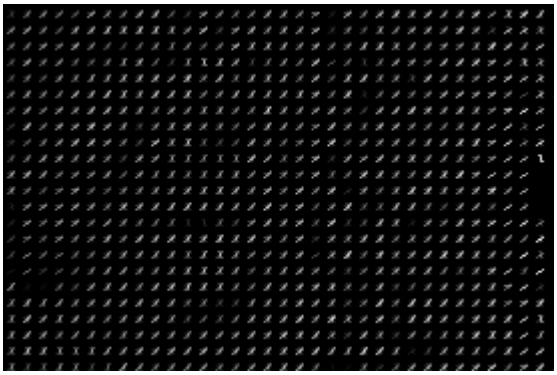


Imagen 26: HOG prueba 1 Norma-L2

Imagen 27: HOG prueba 1 en color Norma-L2

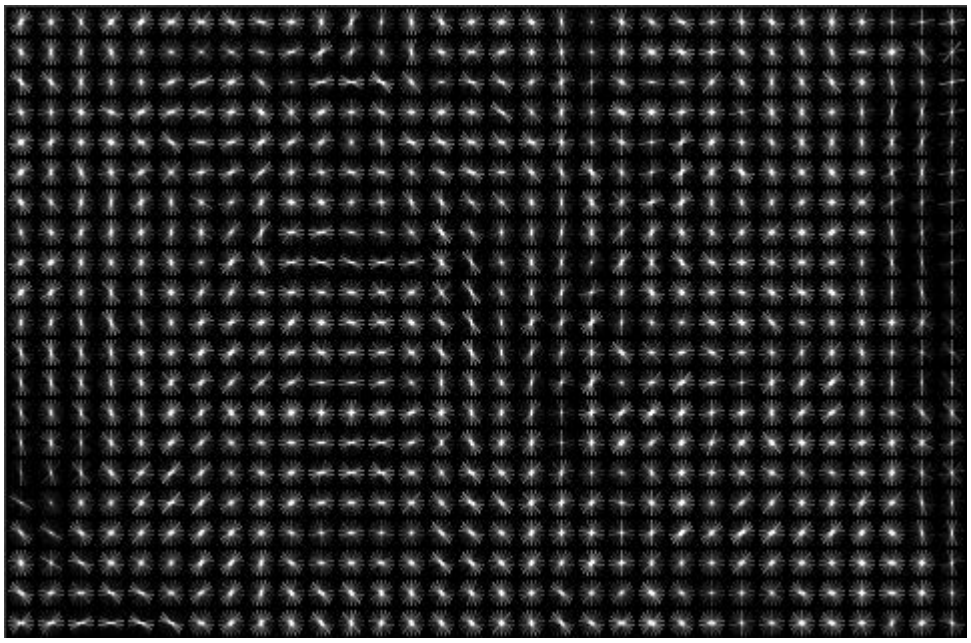


Imagen 28: HOG prueba 1 F-HOG

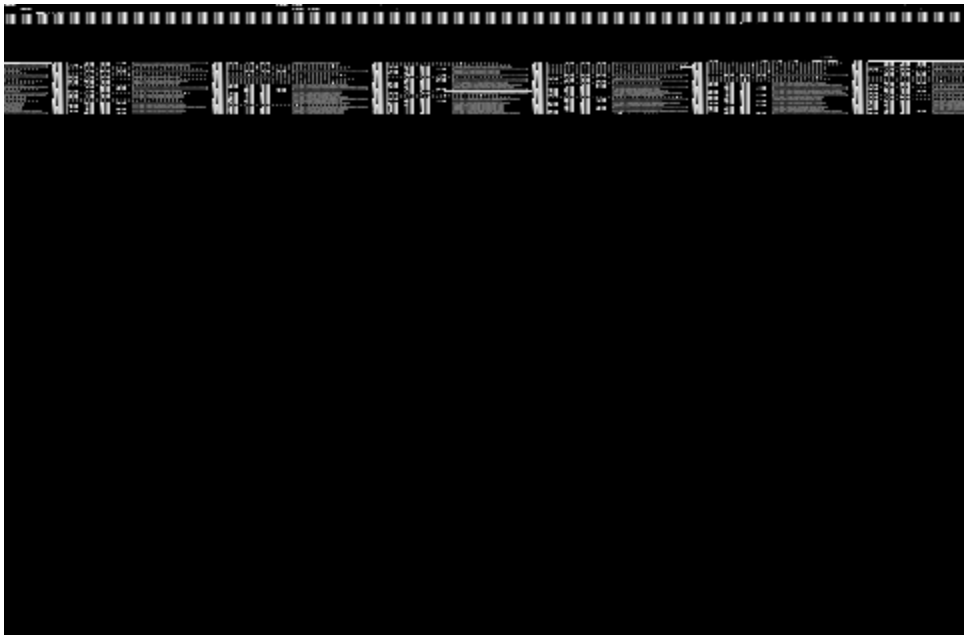


Imagen 29: Representación de autovalores para la prueba 1

4.2.1 Detección de rostros

Al igual que el método de Viola&Jones, con los HOG podemos realizar detecciones de rostros en base al entrenamiento de, por ejemplo, una SVM. La librería Dlib nos ofrece tanto la posibilidad de entrenar nuestra propia máquina de vectores como utilizar los clasificadores que tiene incluidos por defecto.

Como se puede ver en las siguientes imágenes, la localización de rostros de frente es algo mejor que la obtenida por Viola&Jones.

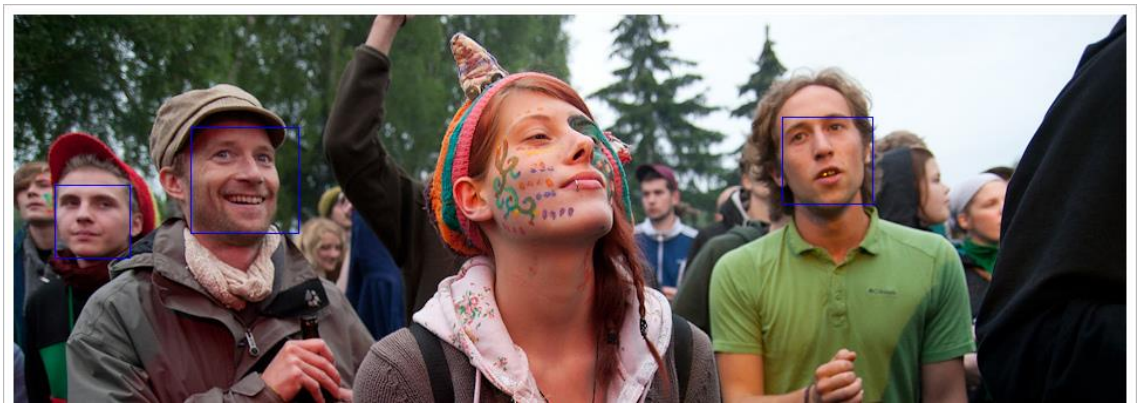


Imagen 30: Localización de 3 rostros frontales con el método HOG

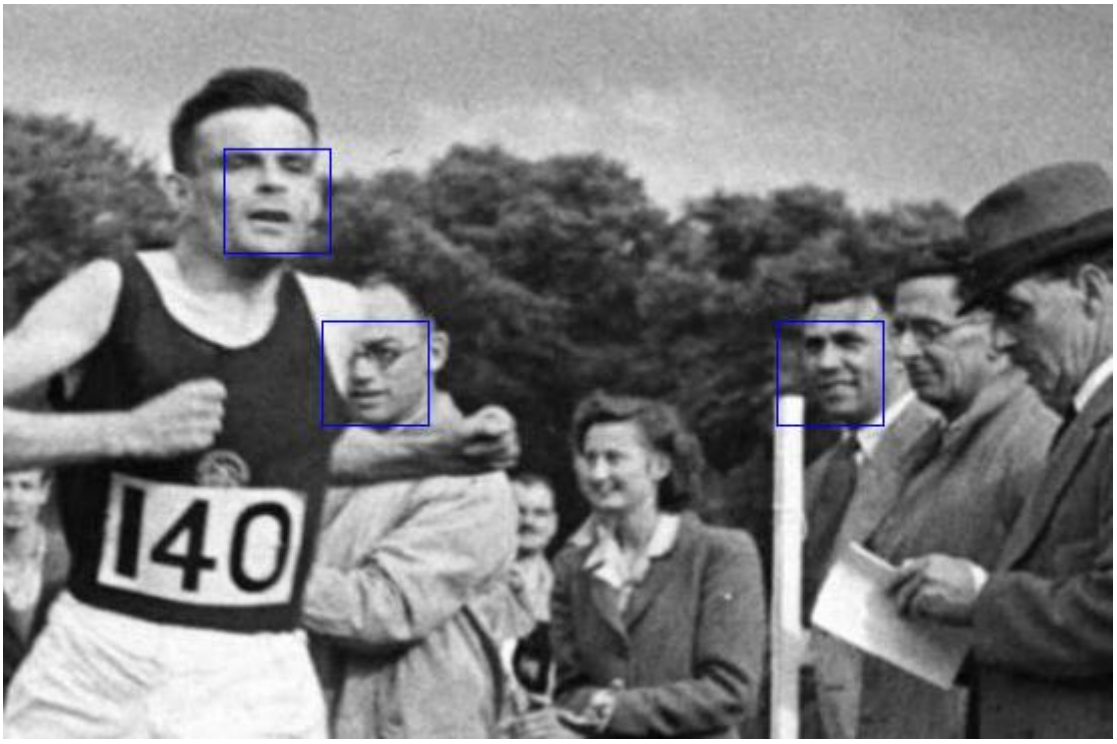


Imagen 31: Localización de 3 rostros frontales con HOG

4.3 Estimación de puntos de referencia

La estimación de puntos de referencia es una técnica para la obtención de zonas o puntos dentro de nuestra imagen que permiten precisar características de esta. Existen muchos algoritmos para realizar esta estimación, como por ejemplo el descrito en “One Millisecond Face Alignment with an Ensemble of Regresion Trees” (Kazemi & Sullivan, 2014), que utiliza árboles de regresión para realizar dicha estimación. Otro algoritmo de estimación es el utilizado en el framework CLM y desarrollado por la universidad de Cambridge.

Como se puede ver en la siguiente imagen, la estimación de puntos permite definir las cejas, ojos, boca, nariz y contorno.

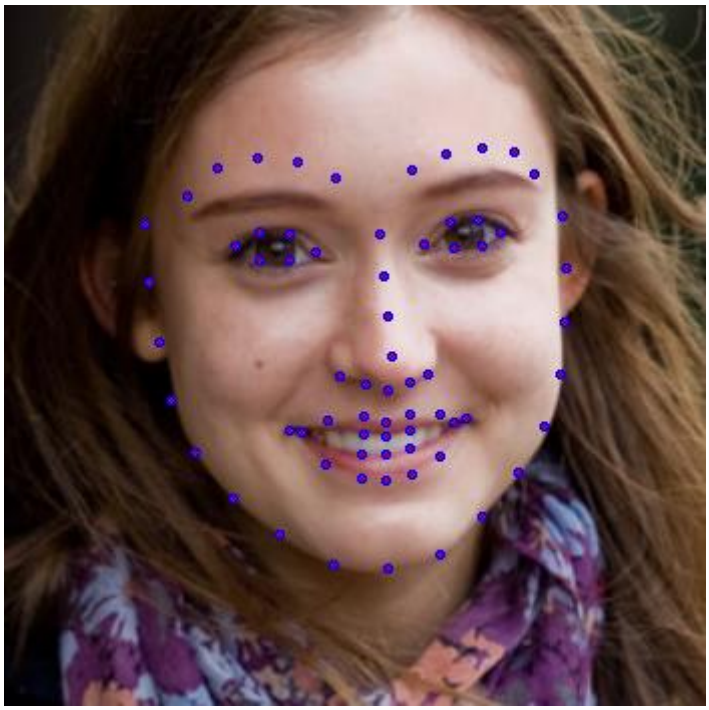


Imagen 32: Estimación de Puntos de Referencia

Esta estimación nos ofrece muchas posibilidades de las que se destacan 2:

- Determinación de rostros: Al existir unos valores que permiten la identificación de puntos concretos del rostro, se posibilita una mayor precisión a la hora de determinar si el ROI es un verdadero rostro.
- Normalización de rostros: A la hora de capturar rostros, se puede comprobar que la posición y punto de vista del observador varia, lo que genera una dificultad a la hora de identificar a la persono. Con la estimación de puntos de referencia y el uso de una máscara genérica, es posible realizar una transposición, escalado y rotación a la imagen para adaptarla a un formato normalizado que permita la identificación de la persona.

4.3.1 Normalización

Como se ha indicado en el punto anterior, la normalización persigue la obtención de una imagen que se encuentre orientada y rotada de una forma concreta.

Para esto se debería generar una representación general de los puntos de información de todos los rostros mediante una media obtenida en una buena colección de ejemplos. Para la realización de este trabajo de investigación he utilizado el mapa o máscara calculado en el proyecto OpenFace en su versión 2.0.4, el cual representa 68 valores tal y como se muestran en la Imagen 33.

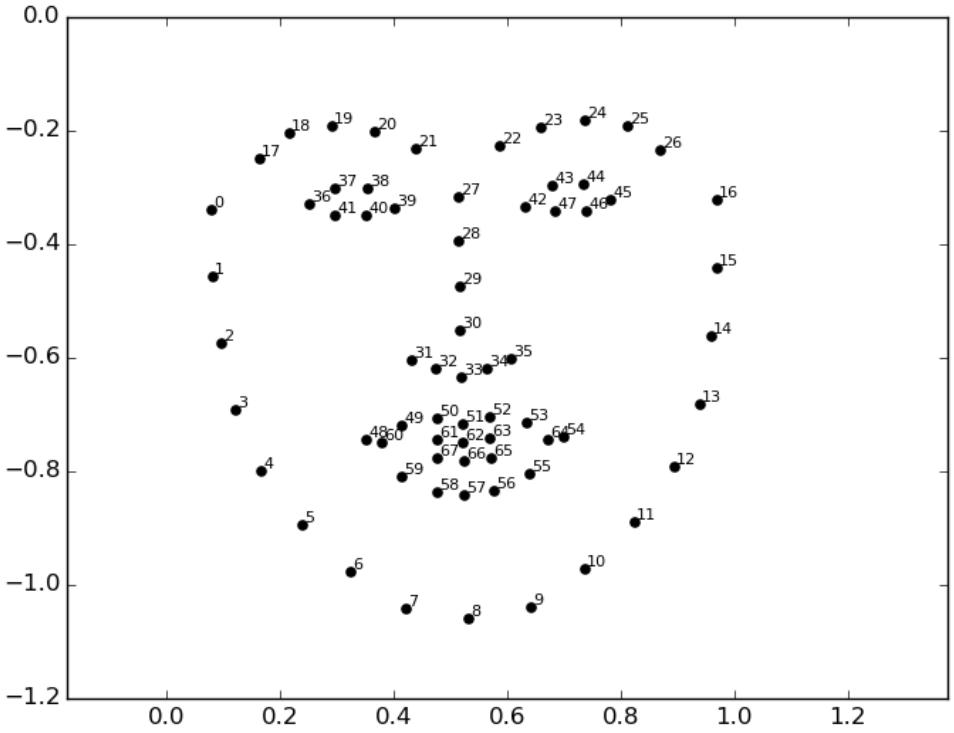


Imagen 33: Máscara obtenida en el proyecto OpenFace[18]

Para realizar la transformación del rostro demarcado por los puntos estimados se realizará una transformación afín.

4.4 Red Neuronal Convolucionales para el reconocimiento de rostros

Las redes neuronales se han vuelto desde hace unos años una poderosa herramienta para la resolución de problemas relacionados con la clasificación. El uso extendido de las GPUs como tecnología de procesamiento de los cálculos necesarios para la evaluación de la red y los recientes dispositivos especializados para inteligencia artificial (como por ejemplo el JETSON[19] de NVIDIA), han generalizado el uso de estas redes permitiendo su estudio y viabilidad en tareas como la detección y la predicción de acciones.

En general, el cálculo necesario para la toma de decisión no es demasiado elevado, el problema radica en el aprendizaje de estas redes ya que suelen requerir de un gran volumen de pruebas sobre los datos hasta hallar una combinación cuya probabilidad esté dentro de los limites requeridos para la correcta clasificación de esos mismos datos. Sin el uso de la potencia de cálculo de las GPU, el uso de estas redes no sería tan común ya que el tiempo requerido hasta que el sistema funcione adecuadamente es elevado.

[18] OpenFace: http://cmusatyalab.github.io/openface/

[19] NVIDIA JETSON: https://www.nvidia.co.uk/autonomous-machines/embedded-systems-dev-kits-modules/

La implementación de una red neuronal no es excesivamente compleja y existen una multitud de librerías y sistemas de código abierto que pueden ser utilizadas para llevar a cabo este tipo de redes, como el comúnmente utilizado TensorFlow[20] de Google.

Tal y como se recoge en la información de su página WEB, TensorFlow es una librería de software libre para máquinas de aprendizaje, siendo sus principales características la Rapidez ya que utiliza tanto las CPUs como las GPUs, Flexibilidad al disponer de un API para facilitar el desarrollo y Multiplataforma lo que le permite adaptarse a cualquier arquitectura.

Sin embargo, aunque TensorFlow parece ser un buen candidato para el reconocimiento y detección de rostros ya que a fin de cuantas las redes neuronales son una particularización de las máquinas de aprendizaje, para el desarrollo de la investigación de este trabajo se ha utilizado la librería YOLO[21] puesto es una implementación específica de una CNN.

4.5 YOLO

Como se ha comentado en el párrafo anterior, YOLO (Redmon & Farhadi, 2017) implementa una Red Neuronal Convolucional que tiene como objetivo la detección de objetos. Las pruebas realizadas por su autor en el reconocimiento de varios tipos de objetos reflejan una tasa de procesamiento de 30 imágenes por segundo, lo que la convierte en un buen candidato para el procesamiento de videos.

Sin entrar en detalles puesto que se dispone de toda la información en las publicaciones realizadas por los autores (https://pjreddie.com/publications/), el método consiste en la división en rejillas de tamaño SxS. Cada rejilla es la responsable de detectar los objetos que se encuentran dentro, generando B limites (bounding boxes) y puntuando la precisión o nivel de confianza de cada objeto encontrado. Este nivel de confianza es definido como la probabilidad del objeto multiplicado por el IoU de la predicción real.

4.5.1 Configuración

La configuración de nuestra red neuronal es una parte fundamental a la hora de construirla y utilizarla. De nada sirve tener una red compuesta de muchos nodos si estos no se encuentran bien distribuidos y configurados.

En una red neuronal de convolución, es precisos no solo definir los niveles que conforman cada capa, sino que se han de definir las matices o máscaras que colocaremos en cada capa y los procesos de unión entre ellas. Esta configuración es un trabajo de investigación en sí solo, y debido a que sobrepasa los límites fijados para el trabajo actual, se ha optado por utilizar una configuración por defecto incluida en Yolo. Esta configuración está especialmente diseñada para la detección de objetos entre los que se incluyen personas.

El fichero de configuración llamado “yolov3.cfg” incluido tanto en el paquete de descarga de YOLO, indica entre otras cosas que se dispone de 75 mascaras de convolución de diferentes dimensiones.

layer	filters	size	input		output	
0	conv	32 3 x 3 / 1	416 x 416 x 3	->	416 x 416 x 32	0.299 BFLOPs
1	conv	64 3 x 3 / 2	416 x 416 x 32	->	208 x 208 x 64	1.595 BFLOPs
2	conv	32 1 x 1 / 1	208 x 208 x 64	->	208 x 208 x 32	0.177 BFLOPs
3	conv	64 3 x 3 / 1	208 x 208 x 32	->	208 x 208 x 64	1.595 BFLOPs

[20] TensorFlow: https://www.tensorflow.org/
[21] YOLO: https://pjreddie.com/darknet/yolo/

```
  4   res    1                 208 x 208 x  64   ->   208 x 208 x  64
  5   conv  128  3 x 3 / 2     208 x 208 x  64   ->   104 x 104 x 128   1.595 BFLOPs
  6   conv   64  1 x 1 / 1     104 x 104 x 128   ->   104 x 104 x  64   0.177 BFLOPs
  7   conv  128  3 x 3 / 1     104 x 104 x  64   ->   104 x 104 x 128   1.595 BFLOPs
  8   res    5                 104 x 104 x 128   ->   104 x 104 x 128
  9   conv   64  1 x 1 / 1     104 x 104 x 128   ->   104 x 104 x  64   0.177 BFLOPs
                                                 ........
                                                 ........
 93   conv  255  1 x 1 / 1     26 x  26 x 512    ->   26 x  26 x 255    0.177 BFLOPs
 94   detection
 95   route  91
 96   conv  128  1 x 1 / 1     26 x  26 x 256    ->   26 x  26 x 128    0.044 BFLOPs
 97   upsample          2x     26 x  26 x 128    ->   52 x  52 x 128
 98   route  97                36
 99   conv  128  1 x 1 / 1     52 x  52 x 384    ->   52 x  52 x 128    0.266 BFLOPs
100   conv  256  3 x 3 / 1     52 x  52 x 128    ->   52 x  52 x 256    1.595 BFLOPs
101   conv  128  1 x 1 / 1     52 x  52 x 256    ->   52 x  52 x 128    0.177 BFLOPs
102   conv  256  3 x 3 / 1     52 x  52 x 128    ->   52 x  52 x 256    1.595 BFLOPs
103   conv  128  1 x 1 / 1     52 x  52 x 256    ->   52 x  52 x 128    0.177 BFLOPs
104   conv  256  3 x 3 / 1     52 x  52 x 128    ->   52 x  52 x 256    1.595 BFLOPs
105   conv  255  1 x 1 / 1     52 x  52 x 256    ->   52 x  52 x 255    0.353 BFLOPs
106   detection
```

Código 2: Parte de la configuración de la Red neuronal

Esta configuración incluye además 3 pruebas de detección a distintos niveles.

4.5.2 Fase de aprendizaje

Antes de poder utilizar Yolo u otra red neuronal, es preciso entrenarla para que las bías o valores asignados a cada nodo tengan unos pesos adecuados para poder clasificar correctamente las entradas y generar una salida valida.

Cuando trabajamos con redes neuronales y más concretamente con redes de convolución, se requerirá mucho tiempo de procesamiento, el cual se dispara si carecemos de un sistema con GPU. Lamentablemente en el desarrollo de este trabajo de investigación, carezco de dicho sistema lo que me ha generado un grave problema a la hora de entrenar la red. De hecho, aunque he lanzado una tarea en varios meses, esta no me ha generado valores correctos debido a que para cada tarea de entrenamiento el tiempo rondaba la media hora, siendo necesarios entre 2000 o 3000 de estas tareas para que el sistema sea capaz de clasificar adecuadamente nuestros datos.

La causa por la que es necesario una gran cantidad de pasos a la hora de realizar nuestro entrenamiento es que todos los parámetros de la red comienzan con valores aleatorios, y concretamente en las redes de convolución hay un conjunto muy elevado de estos parámetros ya que corresponden a los valores dentro de cada matiz, siendo la mayoría de 3x3.

Para intentar acelerar el aprendizaje de nuestra red y por tanto reducir los tiempos, he utilizado el fichero de pesos o valores que vienen por defecto en Yolo con la idea de que el sistema al estar entrenado fuese capaz de encontrar nuestros rostros de forma más rápida. Sin embargo, y como veremos más adelante, el uso de esta técnica generó un problema adicional y algo complicado de solucionar.

Concretamente el fichero utilizado para entrenar la red ha sido “yolov3.weights” que ha sido procesado en 500.200 iteraciones por parte de sus creadores y es capaz de clasificar hasta 80 objetos.

Imagen 34: Imagen de prueba para las pruebas YOLO

Imagen 35: Localización de personas con YOLO

En la imagen anterior se muestran como YOLO con el fichero de pesos original es capaz de localizar 39 personas con un nivel de confianza de al menos 0.5.

```
..\people-4.jpg: Predicted in 20.585000 seconds.
10 person: 100%
6 person: 99%
```

```
2 person: 98%
1 person: 97%
2 person: 96%
1 person: 94%
1 person: 91%
1 person: 89%
1 person: 88%
1 person: 87%
1 person: 86%
1 person: 85%
1 person: 80%
1 person: 79%
2 person: 75%
2 person: 74%
1 person: 73%
1 person: 65%
1 person: 59%
1 person: 58%
1 person: 55%
```

Tras lanzar el proceso de aprendizaje en 100 iteraciones con nuestras imágenes de prueba y que corresponden a tratar 6400 imágenes, el número de personas encontradas aumenta a 41, pero aparecen 3 caballos.

Imagen 36: Detección YOLO tras sobrecargar con 6400 imágenes

```
..\people-4.jpg: Predicted in 21.869000 seconds.
horse: 90%
horse: 77%
horse: 52%
13 person: 100%
1 person: 99%
3 person: 97%
1 person: 95%
2 person: 94%
2 person: 93%
1 person: 92%
1 person: 87%
1 person: 84%
1 person: 81%
1 person: 79%
1 person: 77%
2 person: 76%
1 person: 73%
1 person: 69%
1 person: 66%
1 person: 65%
2 person: 63%
```

```
1 person: 61%
1 person: 54%
3 person: 52%
```

La causa de que aparezcan 3 detecciones de objetos que nos son personas se debe al sobre-entrenamiento. En este caso estamos entrenando la red aportándole imágenes de caras pero sin proporcionarle el resto de elementos para los que el sistema se encuentra ya entrenado. Esto produce un desajuste de los valores de las matrices de convolución lo que a su vez genera los falsos positivos.

Aun así, se puede apreciar como los elementos "persona" nuevos se ajustan realmente a personas, por lo que en este sentido se ha ganado en calidad del conjunto de datos.

Tras 440 iteraciones, lo que implica un total de 28.160+500.200 imágenes, el sistema es capaz de localizar 41 personas con niveles de confianza altos.

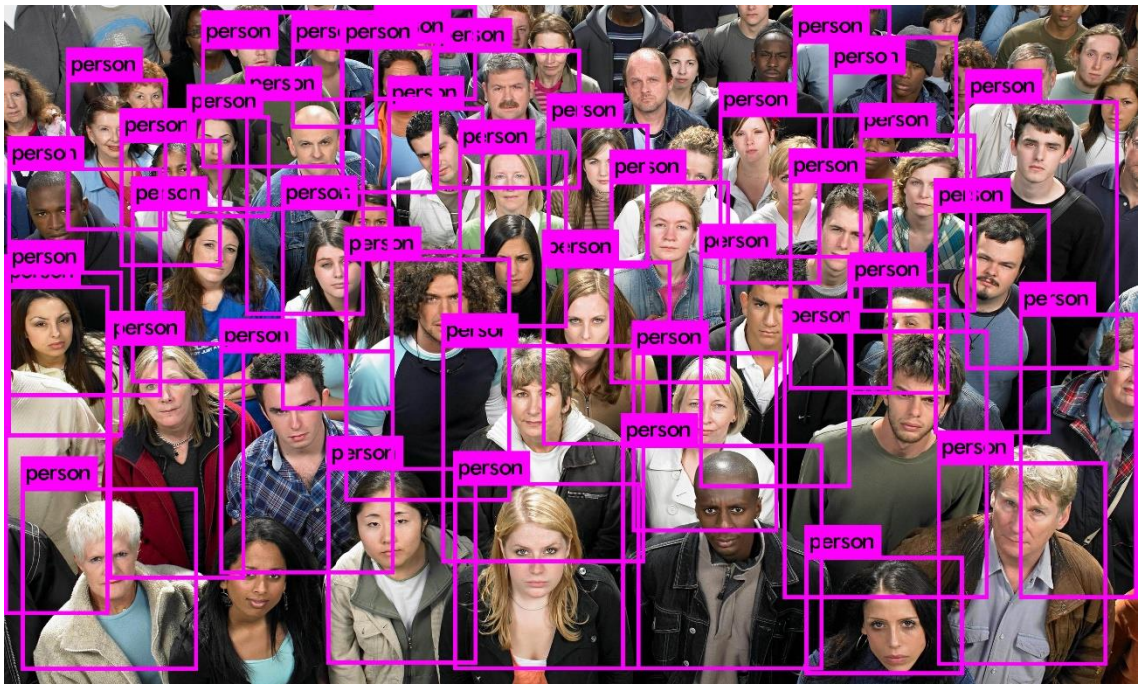


Imagen 37: YOLO tras 527080 imágenes procesadas

```
..\people-4.jpg: Predicted in 22.043000 seconds.
11 person: 100%
5 person: 99%
3 person: 98%
1 person: 96%
2 person: 95%
1 person: 94%
1 person: 93%
```

```
3 person: 92%
2 person: 91%
2 person: 90%
1 person: 87%
1 person: 86%
1 person: 85%
1 person: 82%
1 person: 68%
1 person: 63%
2 person: 58%
2 person: 53%
```

En el apartado 9.3 “Entrenar a YOLO” del Anexo II se expone como realizar el entrenamiento de nuestra red.

4.5.3 Detección de rostros

Una vez entrenada nuestra red, Yolo será capaz de encontrar objetos que estén dentro de los clasificadores obtenidos de forma relativamente rápida. El hecho de basarse en clasificaciones y niveles de confianza, pude producir falsos resultados como la clasificación de objetos de forma incorrecta o la no clasificación de objetos correctos, y como se ha visto en el apartado anterior, esto se debe en gran medida tanto a las imágenes de entrenamiento, como al número de clasificadores a determinar.

Debido a que nuestra red se basa en otra que tiene clasificada 80 tipos de objetos y que particularmente el referido a las caras en origen estaba diseñado para detectar personas, las pruebas realizadas demuestran un alto grado validez llegando incluso a detectar personas donde otros métodos no son capaces.

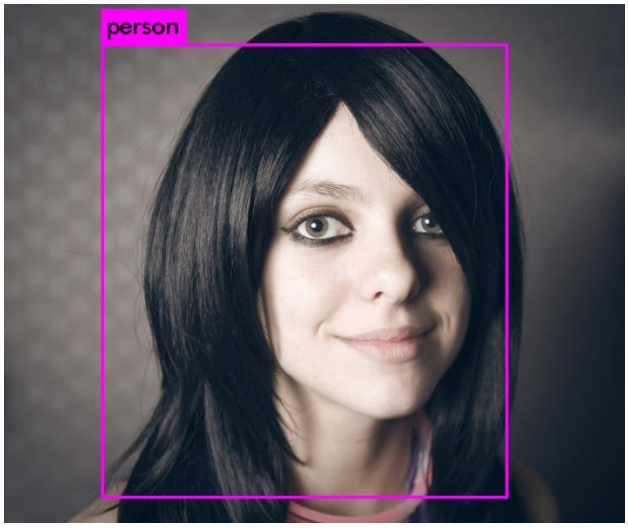


Imagen 38: Resultados de pasar YOLO sobre una imagen simple

En la siguiente imagen tanto para el método de Viola&Jones como para el de HOG, el resultado era de 3 detecciones, sin embargo, utilizando Yolo el sistema encuentra 10 personas.

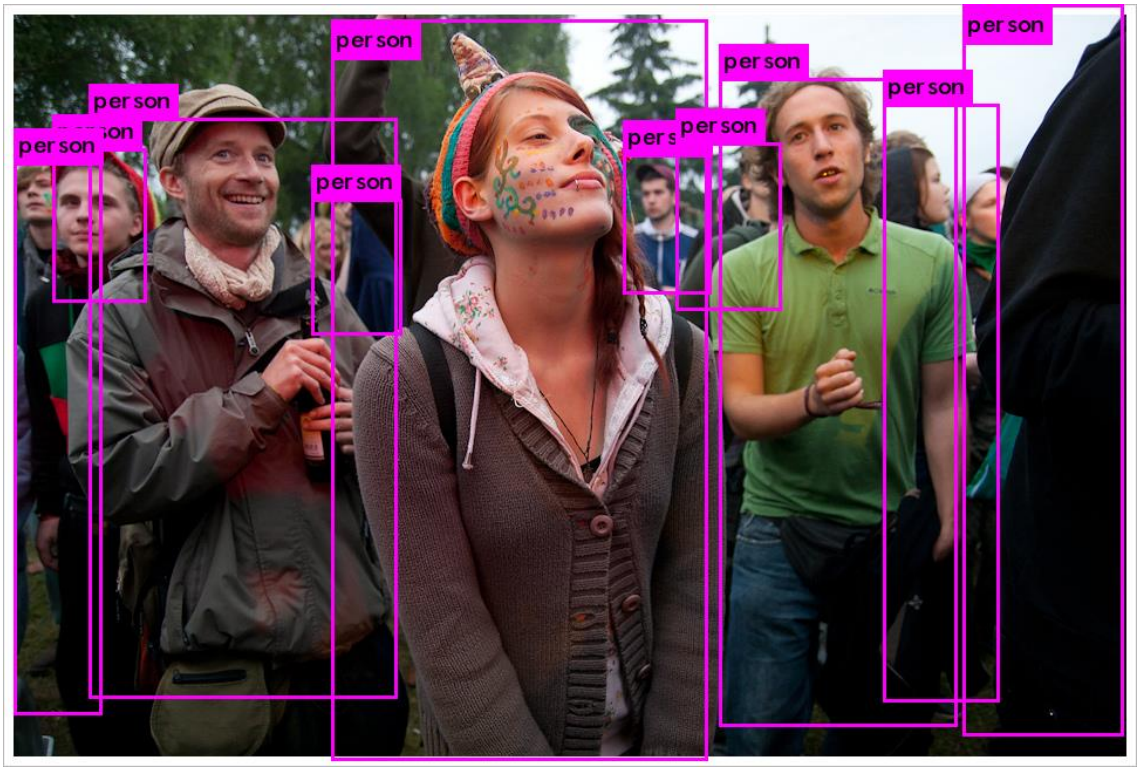


Imagen 39: Prueba de Yolo con una imagen de complejidad intermedia

Añadiendo un poco más de complejidad, si realizamos la búsqueda sobre una imagen en la que HOG era capaz de localizar 3 caras, vemos que Yolo es capaz de localizar 12 personas y además 2 corbatas.

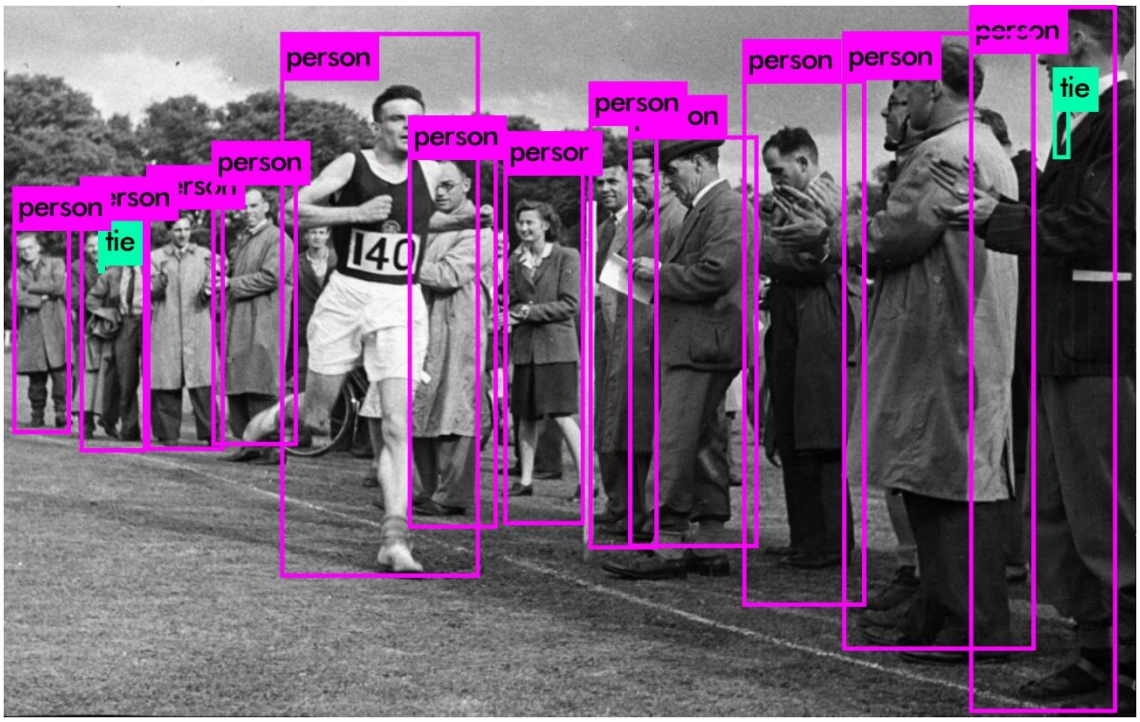


Imagen 40: Prueba de Yolo con una imagen más compleja

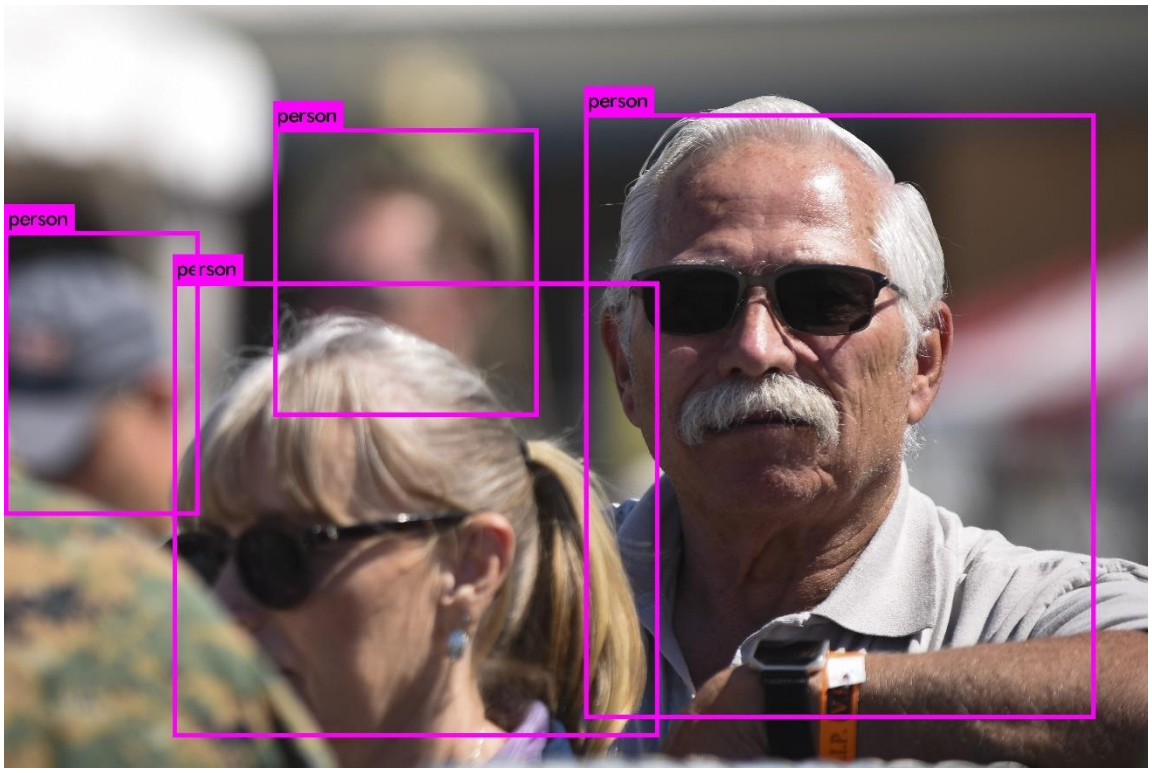


Imagen 41: Prueba de Yolo con una imagen compleja

Por último, y como se puede apreciar en la imagen anterior, cuando procesaos una imagen que contiene tanto personas con gafas de sol como personas en la lejanía y borrosas, el sistema es capaz de localizarlas.

4.6 Eigenfaces

Como ya se ha expuesto en el punto 2.2.1, las auto-caras nos permitirán obtener los valores propios de una colección de rostros. Estos valores propios se pueden utilizar de varias formas, entre las que se encuentran la detección de caras mediante la extracción de características en la colección de rostros similares, o el reconocimiento si hacemos que la colección de rostros corresponda con una misma persona.

En el primer caso, el de la detección, el problema viene por la limitación en la obtención de los autovalores, y que radica en que las caras han de estar dispuestas en la misma proporción y orientación, lo que no siempre es sencillo. Una solución a lo anterior consistiría en la generación de valores propios para frontales y perfil al igual que realizamos para el resto de técnicas.

Aunque la detección es posible, en este caso utilizaremos los autovalores para realizar la identificación. Esto lo conseguiremos mediante el cálculo de diferentes rostros de la misma persona.

Nuevamente nos surge el problema de la orientación y escalado de los rostros en la fotografía, el cual resolveremos mediante la extracción de los rostros de la imagen y el normalizado de las caras (ver 4.3.1)

En la propuesta que se hace en OpenFace, existen dos formas de alinear la imagen, mediante la conexión ojos-labio o mediante la conexión ojos-nariz.

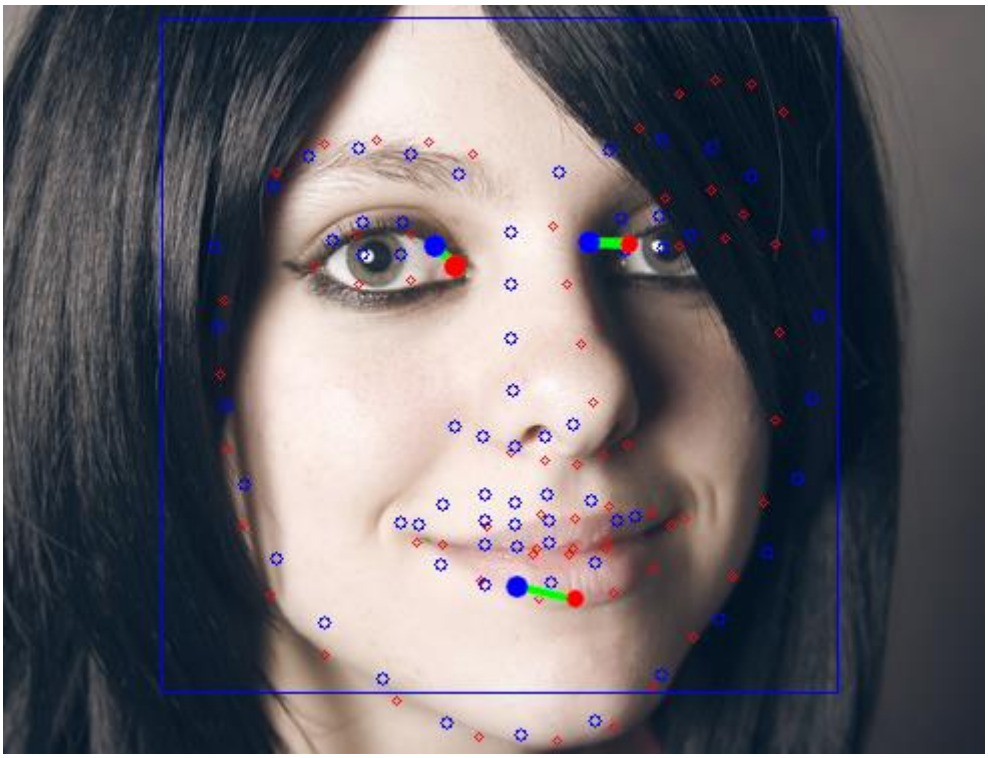


Imagen 42: Adaptación de Rostro a Marcas OJOS-LABIO

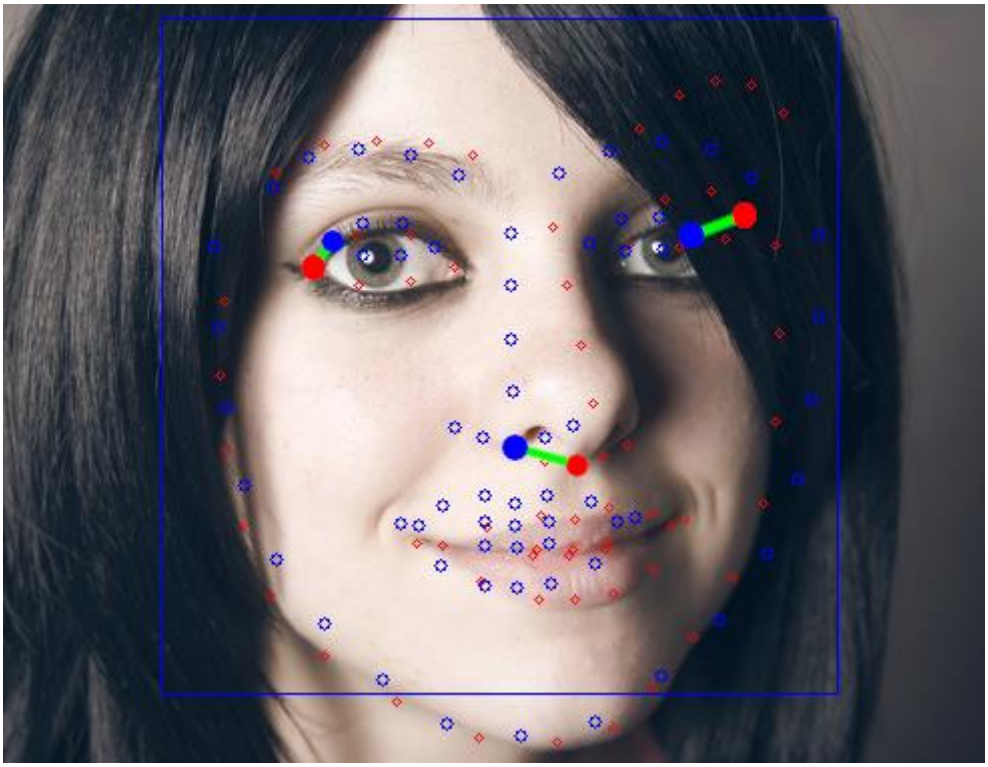


Imagen 43: Adaptación de marcas a OJOS-NARIZ

En las imágenes anteriores los puntos azules corresponden con la media de puntos de interés (ver Imagen 33) y los puntos rojos los estimados por el sistema.

Una vez que tenemos detectada la imagen, localizados los puntos de interés y transformada la imagen para que todas sean similares, podremos obtener los autovalores según se explica en el apartado 8.1 del Anexo I.

Imagen 44: Normalización de imágenes

Imagen 45: EigenFace sobre 11 imágenes

Como se puede apreciar en el eigenface anterior, la media de 11 imágenes de Fernando Alonso sigue mostrando, al ojo humano, la imagen de Fernando, aunque hemos escogido algunas imágenes de perfil.

4.7 Reconocimiento

Muchas de las técnicas anteriores nos permiten realizar tanto la detección como el reconocimiento de personas.

El método de EigenFace es uno de los mejores candidatos para realizar esta tarea. OpenCV incluye un entrenador "FaceRecognizer" que nos posibilita entrenar nuestra red con los eigenface obtenidos en el punto anterior encontrando un clasificador adecuado.

OpenCV utiliza los patrones binarios locales de histogramas (LBPH[22]). Los patrones binarios locales son un operador simple pero eficiente que etiqueta los pixeles de una imagen limitándolo con sus vecinos y considerando un número binario. LBPH utiliza como descriptor de la imagen los histogramas de orientación de gradientes (HOG)

Otra opción sería el de utilizar las redes neuronales para generar un clasificador de personas, para lo cual es suficiente con definir clasificadores para cada persona y entrenar la red convolucionada para que clasifique a cada persona. Esto puede tener un problema si el número de personas a clasificar es muy elevado ya que la complejidad del entrenamiento aumenta de forma elevada. Además, corremos el riesgo de que al añadir una nueva persona se desajuste el resto de clasificadores.

[22] LBPH – Local Binary Patterns Histograms

5 Sistemas actuales

Son muchos los servicios disponibles en la actualidad capaces de procesar nuestras imágenes para obtener no solo los rostros, sino cuestiones como estados de ánimo o acciones realizadas por las personas de la foto.

5.1 API de Google

Google dispone de un API diseñado para reconocimiento visual y que se puede utilizar para la detección de imágenes.

https://cloud.google.com/vision/

Este API tiene un coste por uso, aunque se permite su utilización si no se supera un número determinado de usos.

Para el caso de la Imagen 34 el resultado obtenido al realizar la llama al API para las 10 primeras caras es:

```
POST
https://vision.googleapis.com/v1/images:annotate?fields=responses&key={YOUR
_API_KEY}
 "requests": [ {
   "image": {
    "source": {
     "imageUri": "https://www.youmethod.com/assets/um/people.jpg"
    }
   },
   "features": [
    {
     "type": "FACE_DETECTION"
    }
   ]
  }
 ]
}
```

Código 3: Llamada al API de Google

```
200
{
 "responses": [
  {
   "faceAnnotations": [
    {"boundingPoly": {
      "vertices": [
```

```
      {"x": 1394, "y": 106},
      {"x": 1517, "y": 106},
      {"x": 1517, "y": 249},
      {"x": 1394, "y": 249}
     ] },
    "fdBoundingPoly": {
     "vertices": [
      {"x": 1404,"y": 142},
      {"x": 1500,"y": 142},
      {"x": 1500,"y": 238},
      {"x": 1404,"y": 238}
     ]},
    "landmarks": [
     {"type": "LEFT_EYE",
      "position":
        {"x": 1427.9059, "y": 174.7213, "z": -0.0012884678}
     }, {"type": "RIGHT_EYE",
      "position":
        {"x": 1465.2277, "y": 169.21655, "z": -7.312589}
     },
     …
     …
    ],
    "rollAngle": -7.253332,
    "panAngle": -11.044502,
    "tiltAngle": -2.3316476,
    "detectionConfidence": 0.9995425,
    "landmarkingConfidence": 0.67137027,
    "joyLikelihood": "VERY_UNLIKELY",
    "sorrowLikelihood": "VERY_UNLIKELY",
    "angerLikelihood": "VERY_UNLIKELY",
    "surpriseLikelihood": "VERY_UNLIKELY",
    "underExposedLikelihood": "VERY_UNLIKELY",
    "blurredLikelihood": "VERY_UNLIKELY",
    "headwearLikelihood": "VERY_UNLIKELY"
   },
  …
```

```
      ...
      ...
  ]
}
```

Código 4: Parte del resultado de la llamada al API de Google

Como se puede apreciar, no solo encuentra la cara, sino que encuentra ojos, nariz, boca, etc, e incluso es capaz de indicar el estado de ánimo de la persona.

5.2 Amazon Rekognition

Amazon también nos ofrece una alternativa capaz de localizar rostros mediante servicios WEB.

https://aws.amazon.com/es/rekognition/

El sistema no solo es capaz de localizar objetos, también es capaz de localizar rostros, etiquetarlos y seguirlos a través del vídeo e incluso es capaz de reconocer personas.

Dispone, al igual que el sistema de Google, un apartado de aprendizaje que aporta mayor capacidad al sistema.

5.3 Sistemas en funcionamiento

Además de los servicios anteriores que nos posibilitan realizar nuestros sistemas de detección, otras entidades como Facebook o Apple disponen de sus propios algoritmos de reconocimiento para etiquetar imágenes la primera, e identificar a los usuarios la segunda (FaceID).

Pero no solo se quedan en sistemas de etiquetado y validación. Países como China o Paypal, incorporan el reconocimiento de caras como sistema de pagos.

6 Conclusiones

Como se ha visto a lo largo del trabajo de investigación, el campo de la visión por computadora y más concretamente la relacionada con el reconocimiento de rostros tiene una amplia literatura y está repleta de algoritmos y soluciones que tienen buenos resultados. Sin embargo, todas las técnicas se encuentran centradas en la clasificación de puntos de interés dentro de la imagen y la predicción de rostros en base a una estadística diseñada. Quizás el cerebro humano se base en principios similares para la captación de patrones y la consecuente detección de rostros humanos, pero las técnicas actuales se alejan mucho de las capacidades del cerebro humano.

En relación con esto el sistema implantado por Google para etiquetar las fotografías subidas por los usuarios genero un gran revuelo cuando etiqueto, de forma equivocada a una pareja africana como gorilas. Obviamente el sistema realizó el etiquetado de forma equivocada no por el supuesto racismo del algoritmo, sino debido a que quizás las imágenes de muestra suministradas y utilizadas para entrenar la red no fueron lo suficientemente adecuadas para clasificar de forma correcta las imágenes. En este caso Google reaccionó eliminando la etiqueta de "Gorila" lo que resolvió temporalmente el problema, pero fue nuevamente criticada.

https://www.forbes.com/sites/mzhang/2015/07/01/google-photos-tags-two-african-americans-as-gorillas-through-facial-recognition-software/#65066e69713d

Recientemente el sistema de Amazon generó nuevamente un problema de seguridad al reconocer de forma equivocada a congresistas como delincuentes. No es más que una simple anécdota que demuestra la inmadurez del sistema.

https://www.xataka.com/privacidad/sistema-reconocimiento-facial-amazon-falla-confunde-28-congresistas-delincuentes

Con la conclusión anterior no digo que los sistemas actuales sean inútiles, lo contrario, ya que la capacidad que demuestran es sobradamente funcional para las necesidades a las que están aplicadas, sin embargo, queda mucho trabajo en este campo si queremos disponer de un sistema que realmente sea capaz de localizar rostros sin ninguna duda, estén o no registradas en el sistema.

Durante el transcurso de la investigación que inicialmente tenía un aspecto más práctico, el volumen de información a analizar me ha obligado a basarme solo en aspectos más teóricos y basados en técnicas ya disponibles desde hace años. Sin embargo, no he descartado completamente el desarrollo de herramientas que me han permitido estudiar con más profundidad las técnicas analizadas.

Respecto a las noticias y sistemas que han ido apareciendo a lo largo de estos meses, son muchas las opiniones que tiene la comunidad creada alrededor de estas tecnologías, creciendo cada vez más la idea de la necesidad de regular el uso de estos sistemas. Por ejemplo, la reciente red de cámaras de vigilancia e identificación de china está teniendo ciertos problemas al identificar personas asociándolas a clanes de delincuencia de forma equivocada lo que produce un mal estar general ante este tipo de tecnología. En este sentido Londres tampoco se libra de problemas de protección de la intimidad con sus sistemas de vigilancia.

https://www.eliberico.com/reconocimiento-facial-automatico-en-reino-unido/

7 Trabajos futuros

Existen muchas líneas de trabajo que se pueden abrir a partir del trabajo actual y que más allá de seguir analizando sistemas existentes se pueden centrar en nuevas técnicas que se deriven de las vistas.

Un ejemplo podría ser la adaptación de las redes neuronales para utilizar histogramas de orientación de gradientes. Como se ha visto en el apartado 4.2, el uso de HOG permite la obtención de rostros de forma muy rápida y mejorando otras técnicas. Obviamente el uso de redes neuronales de convolución supera con creces el uso de HOG, sin embargo, estas redes neuronales se basan en la obtención de bordes para configurar la red y quizás el uso de histogramas permitan que la red se comporte de forma más eficiente.

También se ha visto que la estimación de puntos de referencia es muy eficiente con el uso de máquinas de soporte vectorial o mediante árboles de decisión, quizás el uso nuevamente de redes neuronales sea una buena estrategia para mejorar estas localizaciones.

Como puede parecer, las redes neuronales se están convirtiendo en una herramienta muy poderosa en el reconocimiento por computadora, pero no podemos perder de vista otras técnicas de inteligencia artificial que últimamente están llamando la atención y mejorando algunos resultados generados por redes neuronales.

Particularmente seguiré desarrollando el sistema que he creado para adaptarlo a un resultado concreto y centrándome en:

- Detección de movimiento: Esto permitirá centrar la búsqueda solo en aquellas zonas de interés lo que pueden reducir el tiempo de detección.
- Aprendizaje en la detección de puntos de referencia.
- Investigación de otras técnicas de inteligencia artificial para el reconocimiento de rostros.

Todos los trabajos los iré publicando en el repositorio de github y en mis redes sociales (twitter y youtube).

8 Anexo I

8.1 PCA

En este cálculo consideramos X_i como un vector siendo i ∈ (1..N).

El vector medio se puede representar por:

$$\Psi = \frac{1}{N}\sum_{i=1}^{N} X_i \tag{3}$$

Una vez obtenido el valor medio, definimos la diferencia de cada vector como:

$$\Phi_i = X_i - \Psi \tag{4}$$

Los valores obtenidos en la Ecuación anterior serán los utilizados para el análisis de vectores, los cuales consisten en N vectores ortonormales u_n. El vector k (u_k) se escoge de tal manera que:

$$\lambda_k = \frac{1}{N}\sum_{i=1}^{N} (u_k^T \Phi_i)^2 \tag{5}$$

$$u_i^T u_k = \delta_{ik} = \begin{cases} 1, & si\ i = k \\ 0, & en\ otro\ caso \end{cases} \tag{6}$$

El vector u_k y el escalar λ_k son loe autovectores y el autovalor de la matriz de covarianza:

$$C = \frac{1}{N}\sum_{i=1}^{N} \Phi_i \Phi_i^T = AA^T \tag{7}$$

donde $A=[\varphi_1, \varphi_2 \dots \varphi_n]$.

8.2 Clasificador en cascada (Adaboost + Haar cascade)

Esta técnica tiene dos partes diferenciadas.

- Por un lado, se deberá entrenar al sistema para que sea capaz de encontrar los objetos, en nuestro caso caras, ojos, nariz, boca, etc.
- Por otro, y una vez dispongamos de los clasificadores, se encuentra la fase de clasificación y detección.

Como para otras técnicas expuestas en este trabajo, no se describirá en detalle su funcionamiento ya que este se encuentra disponible en los trabajos de los propios autores, aunque para poder facilitar la lectura y comprensión de este documento, se indicarán las ideas y formulaciones principales sin sus desarrollos o comprobaciones.

8.2.1 Adaboost

El algoritmo de AdaBoost[23] fue definido por primera vez en la publicación de la "Journal of Computer and System Sciences" en su edición 55 por (Freund & Schapire, 1997) [http://www.face-rec.org/algorithms/Boosting-Ensemble/decision-theoretic_generalization.pdf].

[23] AdaBoost - Adaptive boosting

Sin entrar en detalles, el algoritmo tiene como objetivo encontrar de forma iterativa una serie de características simples que clasifiquen con mayor acierto las muestras de entrenamiento.

Tomando como ejemplos $x_1, x_2 \dots x_n$, salidas deseadas $y_1, y_2 \dots y_n$ donde $y \in \{-1,1\}$.

Los pesos iniciales para $w_{1,1}, w_{2,1} \dots w_{n,1}$ valen 1/n.

La función error es $E(f(x), y, i) = e^{-y_i f(x_i)}$

Clasificador débil $h: x \to [-1,1]$

Para t en 1 … T

- Escogemos $h_t(x)$:
  - Encontramos un clasificador débil $h_t(x)$ que minimice e_t, $e_t = \sum^{n}_{\substack{i=1 \\ h_{t(x_i)} \neq y_i}} w_{i,t}$
  - Escogemos $\alpha_t = \frac{1}{2}\ln(\frac{1-e_t}{e_t})$
- Añadimos al conjunto:
  - $F_{t(x)} = F_{t-1}(x) + \alpha_t h_t(x)$
- Actualizamos los pesos_
  - $w_{i,t+1} = w_{i,t} e^{-y_i \alpha_t h_t(x_i)}$ para todos los i
  - Re-normalizamos $w_{i,t+1}$ de tal forma que $\sum_i w_{i,t+1} = 1$

Tabla 2: Algoritmo de AdaBoost

8.2.2 Haar feature

Las características Haar son similares a las ondas Haar (Haar, 1909) las cuales son una secuencia de funciones “cuadradas” reescaladas que juntas forman una base de ondas (wavelets).

A la hora de realizar tratamiento de imágenes es habitual la aplicación de transformaciones sobre el dominio de las señales. Una de las transformadas más utilizada en el tratamiento es la transformada de Fourier la cual nos permite descomponer y extraer información de la señal. Las transformadas de wavelets, al igual que la transformada de Fourier consisten en la descomposición de la señal en varias versiones escaladas y trasladadas de la onda original.

Las características Haar, se basan en la captura de una ventana o área de una imagen y a la división de esta en varias zonas. La operación a realizar es la diferencia de la suma de las áreas en las que se dividida dicha ventana.

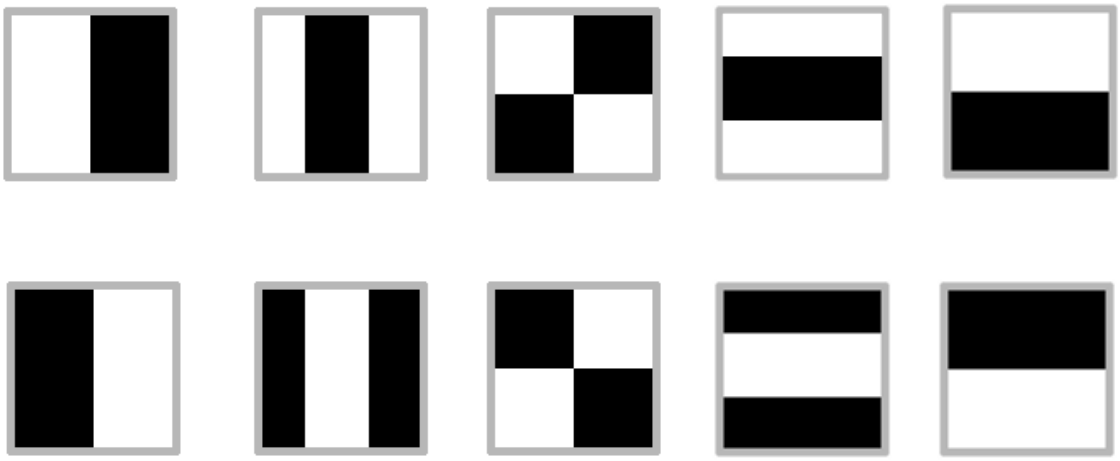


Imagen 46: Ejemplos de clasificadores Haar

8.2.3 Imagen integral

El método de la imagen integral consiste en calcular por cada pixel de la imagen la suma de todos los pixeles superiores y hacia la izquierda.

$$p(x,y) = \sum_{x' \le x, y' \le y} p(x',y') \tag{8}$$

Con el cálculo anterior se puede obtener el valor de cualquier área de la imagen mediante la suma y resta de los valores situados en cada esquina.

En el ejemplo de la Imagen 47 se muestra esquemáticamente cómo se comporta la suma en cada pixel. En este caso el pixel etiquetado con 1 realiza la suma de todos los pixeles marcados en amarillo (Zona A). El etiquetado como 2 realiza la suma tanto de la zona azul como de la amarilla (Zona A + B). El etiquetado como 3 tiene la suma de la zona verde y la amarilla (Zona A + C) y por último el etiquetado con 4 tiene la suma de todas las zonas (Zona A + B + C + D).

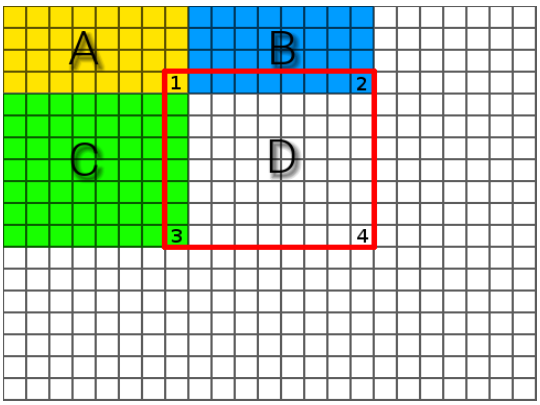


Imagen 47: Esquema para el cálculo de áreas en imágenes integrales

Por tanto, para obtener únicamente la zona D tendríamos que realizar la siguiente operación:

$$Suma(D) = 4 + 1 - (2 + 3) \tag{9}$$

8.3 LDA

El método LDA a diferencia del PCA permite la división de vectores en clases con una probabilidad asociada. El cálculo es similar al PCA con la diferencia de que se calcula una media por cada grupo de vectores perteneciente a la misma clase y se obtiene por tanto un conjunto de clasificadores.

Los pasos a seguir en este método, y partiendo de por ejemplo C*N ejemplos en los que cada clase tiene C vectores sería:

Calculamos la media de todos los vectores:

$$\Psi = \frac{1}{CxN}\sum_{i=1}^{CxN} X_i \tag{10}$$

Calculamos la media para cada clase, p_k:

$$\mathrm{p}_k = \frac{1}{C}\sum_{i=1}^{C} X_i \tag{11}$$

Calculamos el valor de entrenamiento $t_{k,j}$, donde k es el índice de clase y j el índice del vector a entrenar:

$$\mathrm{t}_{k,j} = X_j - p_k \tag{12}$$

Construimos las matrices de dispersión S_c:

$$S_c = \sum_{i=1}^{C} t_{c,i} t_{c,i}^T \tag{13}$$

Y obtenemos la matriz de dispersión de clases S_W:

$$S_W = \sum_{i=1}^{N} S_i \tag{14}$$

8.4 Neurona artificial

Las neuronas artificiales están basadas en las neuronas biológicas. Como se muestra en la Imagen 48, las neuronas biológicas se componen de un centro o núcleo que es la parte que realiza el trabajo, una serie de conexiones a modo de sensores llamadas dentritas y unas terminales denominadas axones que tienen como misión ofrecer el estímulo generado por la neurona.

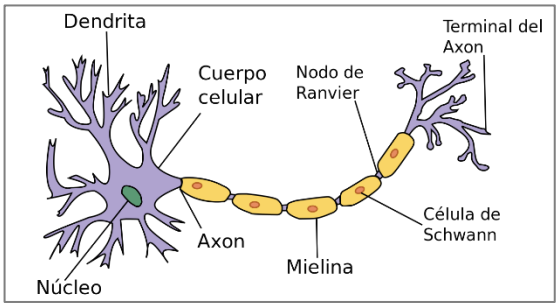


Imagen 48: Neurona biológica

Gracias a la definición sobre el trabajo o cálculo realizado por las neuronas biológicas publicado en el artículo "The bulletin of Mathematical Biophysics" por (McCulloch & Pitts, 1943), se ha podido extender el uso de este tipo de entidades en la actualidad. En el artículo, los autores parten de varias consideraciones acerca del comportamiento de las neuronas:

- La actividad de una neurona tiene solo dos estados, activada-desactivada.
- Un cierto número de sinapsis[24] deben ser excitados dentro de un tiempo determinado para que la neurona sea excitada. Esta activación es independiente de actividades previas o del estado de la neurona.
- El único retardo que puede afectar al sistema es el retardo sináptico.
- La activación de cualquier sinapsis con la capacidad de inhibir previene absolutamente la activación de las neuronas en ese tiempo.
- La estructura de la red no cambia en el tiempo.
- Cada entrada o sinapsis que excitan a la neurona tiene el mismo peso.

[24]Sinapsis: Región de comunicación entre la dendrita y el núcleo de la neurona.

El modo en el que operan las neuronas expuestas en este artículo es simple. Durante el tiempo de activación, la neurona responde a las entradas o actividad de su sinapsis. Si no hay sinapsis que inhiban a la neurona, suma las entradas de todas las sinapsis y verifica si el resultado alcanza o excede su umbral de excitación. En caso afirmativo la neurona se activa.

En 1958 (RosenBlatt, 1958) define un hipotético sistema nervioso o máquina llamado **perceptrón**, el cual fue diseñado para simular el principio fundamental de inteligencia. Basándose en la neurona y en el hecho de que por sí sola no tiene razón de ser ya que su única labor es la de activarse ante una serie de estímulos y producir excitación en otras neuronas conectadas, el perceptrón se basa en el concepto de función que recibe una serie de entradas, representadas en un vector y devuelve una salida booleana única.

$$f(\vec{x}) = \begin{cases} 1, & si\ \vec{w} \cdot \vec{x} - u > 0 \\ 0, & en\ otro\ caso \end{cases}$$

Donde $\vec{w}$ es un vector de pesos (weights) y $\vec{w} \cdot \vec{x}$ es el producto escalar. u es el umbral de inhibición de la neurona.

La salida obtenida por el perceptrón es binaria y por tanto separa o clasifica las entradas en dos zonas o clasificaciones. El sistema es capaz de aprender o adaptar la línea de separación de las entradas para ajustarse a una mejor clasificación en base a los datos de entrenamiento suministrados.

El aprendizaje del perceptrón consiste en la modificación de los pesos para adaptarse a una salida más adecuada y dan pie a la formula general del perceptrón.

$$w_i' = w_i + \sigma(y - y')x_i$$

Siendo "σ" una función que normaliza el error en el rango [0,1], "y" la salida esperada e "y'" la salida obtenida.

Sin embargo, el concepto de perceptrón introducido por RosenBlatt tenía problemas a la hora de realizar ciertas tareas y requería un uso computacional no disponible hasta la fecha. No fue hasta que en 1986 cuando (Rumelhart, Hinton, & Williams, 1986) introdujeron la propagación hacia atrás (backpropagation) ofreciendo una solución para el sistema de aprendizaje automático de redes con múltiples capas.

La ventaja de disponer de múltiples capas o niveles de neuronas permite limitar o diseñar zonas de clasificación más complejas, lo que a su vez produce un resultado más fiel al esperado.

Además de la mayor precisión en la definición de las zonas de clasificación, al consistir en una formulación basada únicamente en sumas y multiplicaciones, se posibilita la utilización de unidades de procesamiento menos complejas que las disponibles en una CPU, habilitando de esta manera el uso masivo de GPUs y por tanto aumentando el uso de las redes neuronales.

Aunque existen varias adaptaciones en base al tipo de red neuronal que estemos diseñando, la formula general que define una neurona es la siguiente:

$$a_{i+1} = \sigma(W_i \cdot a_i + b_i) \qquad (15)$$

Siendo a_{i+1} la salida de la capa i, W_i la matriz de pesos de la capa y b_i un valor de capa llamada *bías* que nos proporciona un grado de libertad adicional a la capa.

La función sigmoide “σ” suele utilizar la curva logística para normalizar su valor.

$$\sigma(x) = \frac{1}{1 + e^{-x}} \tag{16}$$

8.5 Convolución

La convolución es una operación de dos funciones en el dominio de los números reales que tiene como misión obtener la variación de la primera sobre la segunda.

$$s(t) = \int_{a=-\infty}^{+\infty} f(a)g(t-a)da \tag{17}$$

Típicamente es denotada como:

$$s(t) = (f * g)(t) \tag{18}$$

Para funciones discretas tenemos que:

$$s(t) = f(t) * g(t) = \sum_{n} f(n)g(t-n) \tag{19}$$

8.5.1 Convolución de matrices

Es una técnica o procedimiento utilizado para el filtrado de imágenes mediante la utilización de una matriz.

La definición matemática de la convolución de matrices dada una matriz I_{nxm} y una matriz $M_{(2N+1)x(2N+1)}$ con 2N+1<n,m será:

$$I' = I * M \tag{20}$$

$$i'_{x,y} = \frac{1}{c} \sum_{r=1}^{2N+1} \sum_{s=1}^{2N+1} i_{x-N+r-1,y-N+s-1} m_{r,s} \tag{21}$$

Como se puede apreciar, la dimensión de la matriz de convolución siempre tendrá un valor impar.

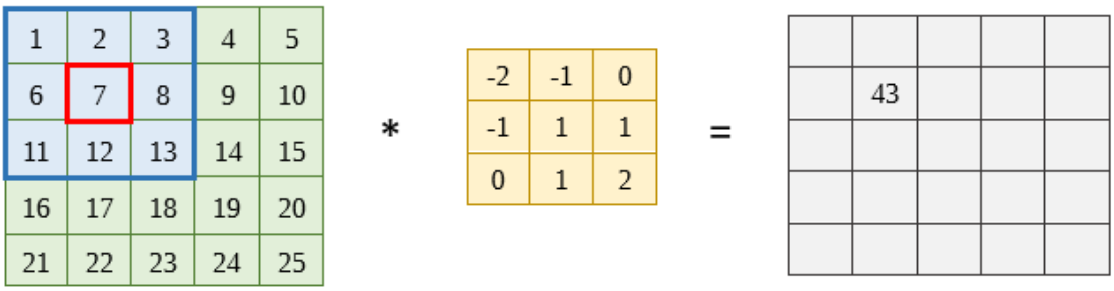


Imagen 49: Calculo de un pixel con la matriz de convolución

Como se observa en la imagen anterior, el cálculo de convolución se realiza moviendo la matriz de convolución sobre la imagen original. Debido a esto, estas matrices también reciben el nombre de máscaras.

Por norma general estas máscaras suelen tener una dimensión de 3x3 o de 5x5. En la siguiente tabla se muestra algunas de las máscaras más utilizadas en el reconocimiento visual.

Enfoque	Desenfoque	Realce de bordes	Repujado
$\begin{bmatrix} 0 & -1 & 0 \\ -1 & 5 & -1 \\ 0 & -1 & 0 \end{bmatrix}$	$\begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}$	$\begin{bmatrix} 0 & 0 & 0 \\ -1 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix}$	$\begin{bmatrix} -2 & -1 & 0 \\ -1 & 1 & 1 \\ 0 & 1 & 2 \end{bmatrix}$
Detección de bordes	Filtro de tipo Sobel	Filtro de tipo Prewitt	Filtro de tipo Sharpen
$\begin{bmatrix} 0 & 1 & 0 \\ 1 & -4 & 1 \\ 0 & 1 & 0 \end{bmatrix}$	$\begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix}$	$\begin{bmatrix} -1 & 0 & 1 \\ -1 & 0 & 1 \\ -1 & 0 & 1 \end{bmatrix}$	$\begin{bmatrix} 1 & -2 & 1 \\ -2 & 5 & -2 \\ 1 & -2 & 1 \end{bmatrix}$
Filtro Norte	Filtro Este	Filtro de tipo Gauss	
$\begin{bmatrix} 1 & 1 & 1 \\ 1 & -2 & 1 \\ -1 & -1 & -1 \end{bmatrix}$	$\begin{bmatrix} -1 & 1 & 1 \\ -1 & -2 & 1 \\ -1 & 1 & 1 \end{bmatrix}$	$\begin{bmatrix} 1 & 2 & 3 & 1 & 1 \\ 2 & 7 & 11 & 7 & 2 \\ 3 & 11 & 17 & 11 & 3 \\ 2 & 7 & 11 & 7 & 1 \\ 1 & 2 & 3 & 2 & 1 \end{bmatrix}$	

Tabla 3: Ejemplo de máscaras más utilizadas

Algunos de las máscaras expuestas en la tabla anterior, como las del filtro Sobel, Prewitt, Sharpen, … solo se presentan en una de las dimensiones de la imagen. Será necesario aplicar tanto la convolución sobre el eje de las X, como la correspondiente en el eje de las Y utilizando posteriormente la operación gradiente.

8.6 Gradiente

Aunque no entra en el ámbito de este trabajo, se definirá brevemente que es y cómo se calcula el gradiente de una imagen.

Básicamente el gradiente de una función de dos variables *f(x,y)* es un vector bidimensional perpendicular al borde en ese punto. Como se puede ver en la ecuación (22), el gradiente se calcula mediante la primera derivada, lo que nos permite calcular la cantidad de variación existente entre dos puntos cercanos.

$$G[f(x,y)] = \begin{bmatrix} G_x \\ G_y \end{bmatrix} = \begin{bmatrix} \frac{\partial}{\partial x} f(x,y) \\ \frac{\partial}{\partial y} f(x,y) \end{bmatrix} \tag{22}$$

De la ecuación anterior se puede obtener tanto la magnitud (el cuanto) como la dirección (hacia donde) se produce la máxima variación:

$$|G| = \sqrt{G_x^2 + G_y^2} \qquad |G| \approx |G_x| + |G_y| \qquad \phi(x,y) = \tan^{-1}\frac{G_y}{G_x} \tag{23}$$

8.7 Cálculo de Histograma de orientaciones de gradientes

En apartados anteriores se ha visto como obtener el gradiente de una imagen y como utilizar máscaras para su cálculo.

El estudio llevado a cabo por *Navneet Dalal* y *Bill Triggs* demuestra que la mejor máscara para la obtención del gradiente en el cálculo HOG en la detección de humanos, es la de una dimensión (1-D) del tipo:

$$g_x = [-1 \quad 0 \quad 1] \qquad g_y = \begin{bmatrix} -1 \\ 0 \\ 1 \end{bmatrix}$$

Imagen 50: Imagen de ejemplo para el cálculo de HOG

Imagen 51: Gx para sobel 3-D

Imagen 52: Sobel 3D

Imagen 53: Gy para sobel 3-D

Imagen 54: Gx para sobel 1-D

Imagen 55: Sobel 1-D

Imagen 56: Gy para sobel 1-D

Una vez obtenida tanto la magnitud como la dirección en cada punto, se procederá al cálculo del histograma de orientaciones. Para ello se discretizará los grados en un rango que maximice las características de la imagen. Nuevamente los autores del método, tras la realización de varias pruebas determinaron que el mejor rango es el de 9 valores con un intervalo de 20 grados.

Grados	0	20	40	60	80	100	120	140	160
Radianes	0	π/9	2π/9	π/3	4π/9	5π/9	2π/3	7π/9	8π/9

Tabla 4: Rango de orientaciones para el método HOG

El cálculo de orientación en el gradiente se basa en la tangente del gradiente en y sobre el gradiente en x, lo que devuelve un rango de valores está comprendido ente 0 y 360º [0-2π radianes].

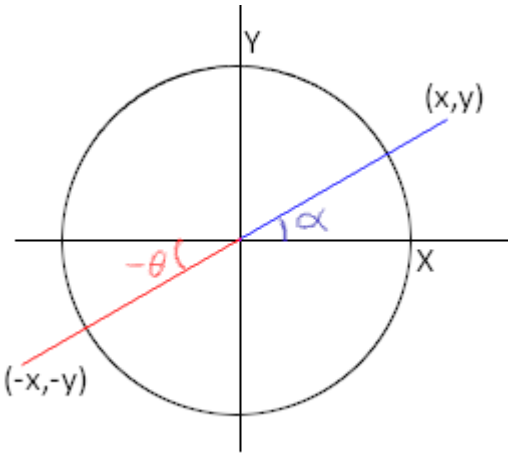


Imagen 57: Arco-tangente

Como se puede comprobar en la imagen anterior, el resultado de la inversa de la tangente para (x,y)=α y para (-x,-y)=-θ deberá devolver la misma pendiente que se asemeja a la de grado 45º.

A la hora de asignar cada uno de los valores al rango propuesto, se realizará una división de la magnitud en ese punto ponderado la distancia que media el grado real y ambos extremos. En el caso de que el ángulo supere 160º, el reparto se realizará entre el 160 y el 180, asignando los correspondientes valores del 180 al 0.

50	0	0	23	54	88	20	32
45	54	65	10	1	3	12	12
11	11	12	123	23	55	45	12
11	11	31	12	21	12	61	32
21	32	23	54	68	10	51	1
15	54	78	54	84	31	13	122
51	65	66	0	78	10	18	13
54	0	33	13	0	89	11	0

Imagen 60: Magnitudes para el Gradiente

5	0	0	3	54	88	20	32
12	1	6	100	1	3	12	5
6	11	12	123	23	55	45	5
5	3	31	12	21	12	136	32
21	32	23	22	160	10	51	1
35	54	90	180	84	31	5	122
5	108	11	0	5	10	18	13
19	0	33	13	0	175	4	0

Imagen 59: Orientaciones para el gradiente

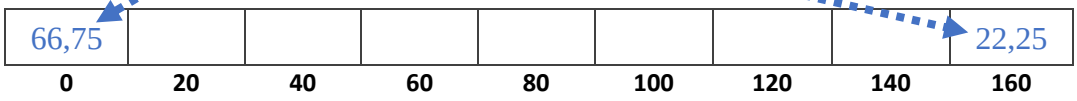


Imagen 58: Histograma de gradiente

A la hora de hacer el histograma no se realizará sobre el total de la imagen, sino que se agrupará en bloques. Nuevamente los autores determinaros que el tamaño de etas agrupaciones deberá ser de 8x8 para obtener una correcta colección de características en la imagen.

Una vez obtenidos los correspondientes histogramas, estos contendrán valores globales que puede llegar a producir problemas en la obtención de características debido por ejemplo a cambios de iluminación. Para solucionar este problema se realiza un paso de normalización por bloques con 3 vecinos tal y como se muestra en la Imagen 61. Como se puede ver, los vecinos serán el inmediatamente situado a la derecha, el situado debajo y el de la diagonal.

La normalización se puede realizar de cualquiera de las formas propuestas por los autores del método, siendo la más utilizada la normalización L2 o norma euclídea.

$$\|v\| = \sqrt{v_1^2 + v_2^2 + \cdots + v_n^2} = \sqrt{\sum_{i=1}^{n} v_i^2} \qquad (24)$$

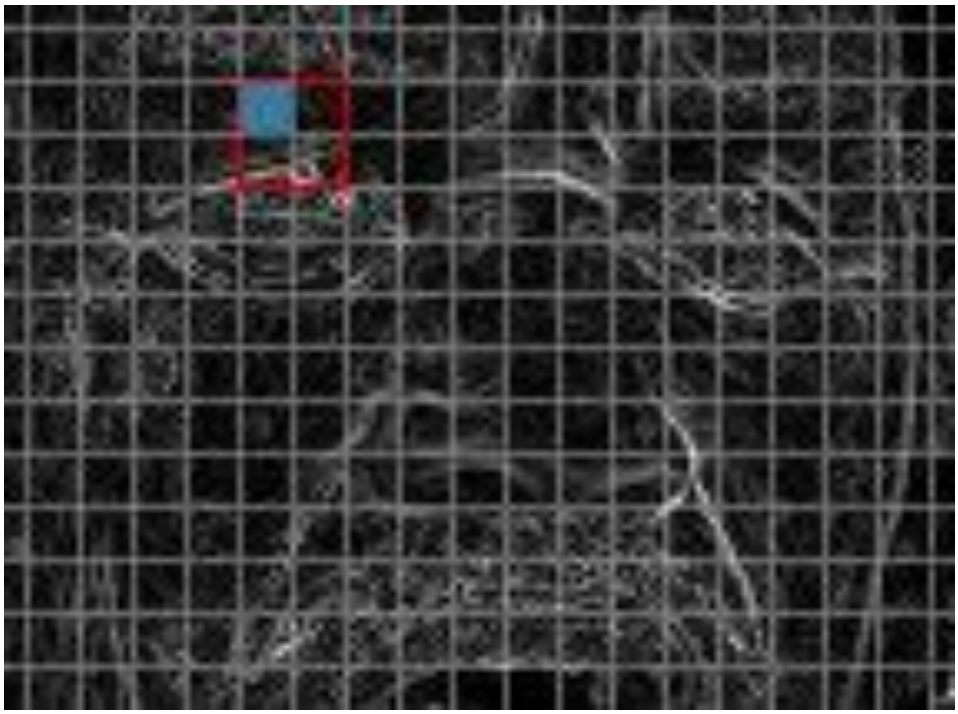


Imagen 61: Normalización de Histograma

Una vez normalizado el histograma, los valores de orientación dentro de cada rango estará comprendida entre [0-1].

8.7.1 HOG Features

En el trabajo "Object Detection with Discriminatively Trained Part Based Models" (Felzenszwalb, Girshick, McAllester, & Ramana, 2010) se presenta esta normalización para el cálculo de histograma basado en características o auto-vectores.

Las características son descritas en un histograma de 36 dimensiones, aunque se puede utilizar una alternativa con solo 13 dimensiones ya que éstas son capaces de obtener la información esencial.

Para discretizar el gradiente en cada punto se utilizará bien el contraste (25) o la intensidad (26)

$$B_1(x,y) = redondeo\left(\frac{p \cdot \theta(x,y)}{2\pi}\right) módulo\ p \tag{25}$$

$$B_2(x,y) = redondeo\left(\frac{p \cdot \theta(x,y)}{\pi}\right) módulo\ p \tag{26}$$

Siendo p el número de valores discretos o pertenecientes al rango del histograma, en nuestro caso 9.

El siguiente paso consiste en la normalización por bloques, el cual se realiza con un tamaño (k) de 8x8 y que incluye un valor de corte (α). La normalización utiliza un factor $N_{\delta,\gamma}(i,j)$ donde $\delta,\gamma \in \{-1, 1\}$ y se define $T_\alpha(v)$ como el corte con límite α para el vector "v".

$$N_{\delta,\gamma}(i,j) = (\|C(i,j)\|^2 + \|C(i+\delta,j)\|^2 + \|C(i,j+\gamma)\|^2 + \|C(i+\delta,j+\gamma)\|^2)^{\frac{1}{2}} \quad (27)$$

$$H(i,j) = \begin{pmatrix} T_\alpha(\frac{C(i,j)}{N_{-1,-1}(i,j)}) \\ T_\alpha(\frac{C(i,j)}{N_{+1,-1}(i,j)}) \\ T_\alpha(\frac{C(i,j)}{N_{+1,+1}(i,j)}) \\ T_\alpha(\frac{C(i,j)}{N_{-1,+1}(i,j)}) \end{pmatrix} \quad (28)$$

De lo anterior se deduce que para cada valor del histograma tendremos 4 normalizaciones por 9 valores discretos, en total 36 valores para cada normalización.

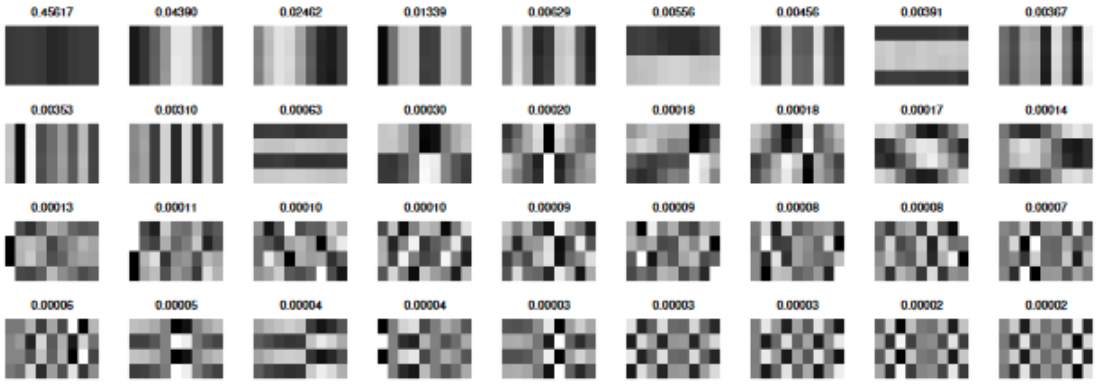
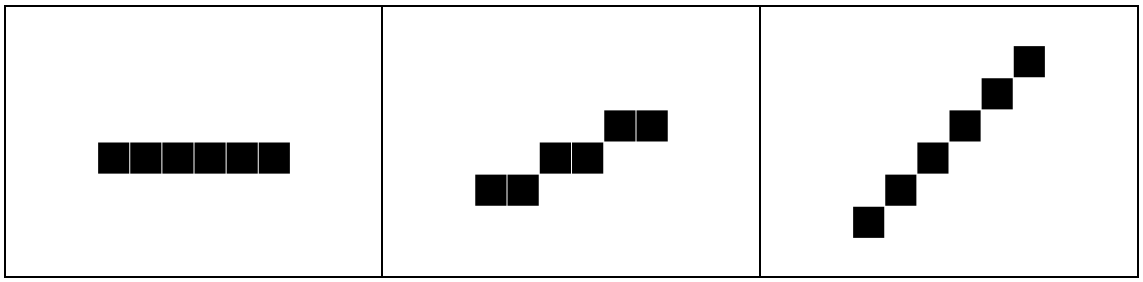


Imagen 62: Autovalores para 4 normalizaciones y 9 valores discretos

8.7.2 Representación de valores

La representación es lo más complejo de llevar a cabo en el método de HOG. En la aplicación desarrollada para el estudio del método se ha utilizado una representación gráfica para cada uno de los ángulos que conforman nuestro histograma. En la siguiente tabla se puede ver cada uno de los trazos utilizado.

Sin embargo, la representación tal cual de estos trazos sobre la imagen o sobre un fondo negro no representa con claridad la dirección dominante del histograma. Para dar una solución a este problema se utilizará el valor de la magnitud como valor de opacidad de cada trazo.



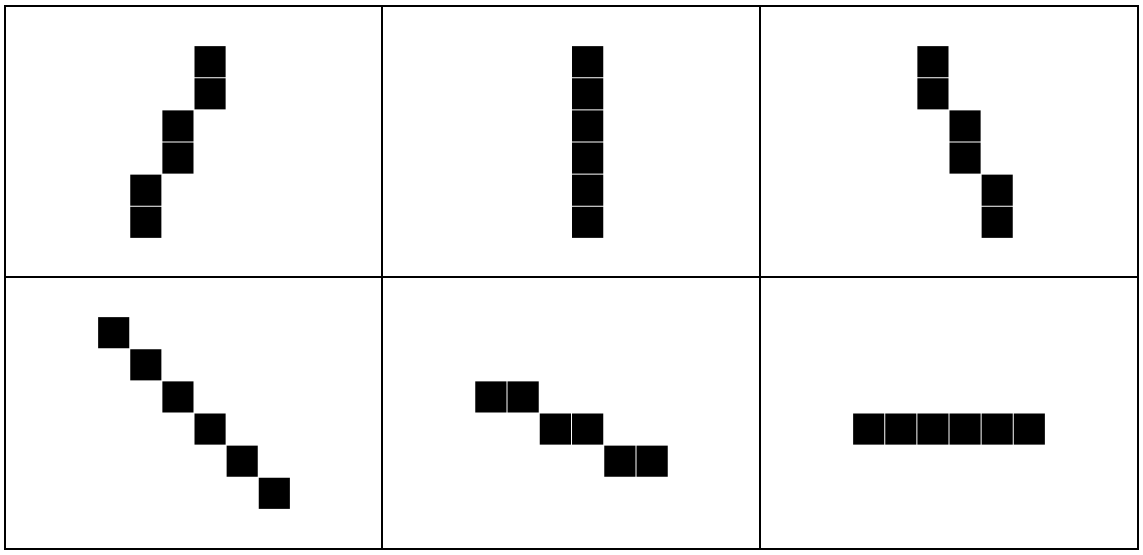


Tabla 5: Trazos para la representación gráfica del método HOG

8.8 Pooling

Las funciones de agrupamiento reemplazan la salida de nuestras capas superiores o de nuestra red con un resumen de salidas vecinas.

Existen diferentes funciones de agrupamiento como “max pooling” definida por (Zhou & Chellappa, 1988). Esta operación selecciona la máxima salida en una vecindad rectangular.

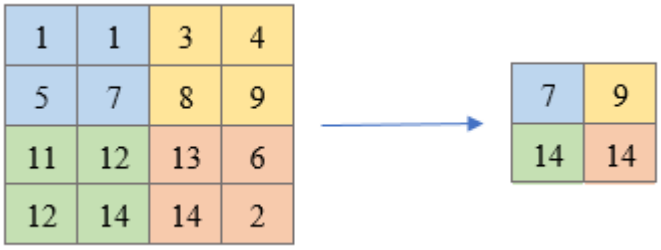


Imagen 63: max polling con filtro de 2x2

Average pooling, es otra técnica de agrupación que se basa en la obtención de la media de los valores existentes en la vecindad.

8.9 Red neuronal recurrente simple (SRN)

Las redes recurrentes simples son aquellas que usan contexto en las neuronas. Estas neuronas son un tipo de neurona que recuerda sus entradas y además inyecta su propia salida como entrada de sí misma para la siguiente etapa de cálculo de la red.

Las SRN más conocidas son las de Elman (Elman, 1990) y Jordan (Jordan, 1997) la cuales utilizan una matriz de parámetros adicional para gestionar la memoria.

Partiendo de la ecuación (15) se hacen unos pequeños ajustes de la siguiente forma:

Red Elman

$$h_t = \sigma_h(W_h x_t + U_h h_{t-1} + b_h) \quad (29)$$

$$y_t = \sigma_y(W_y h_t + b_y) \quad (30)$$

Red Jordan

$$h_t = \sigma_h(W_h x_t + U_h y_{t-1} + b_h) \quad (31)$$

$$y_t = \sigma_y(W_y h_t + b_y) \quad (32)$$

Donde

- x_t es cada uno de los nodos en la capa de entrada.
- h_t es cada uno de los nodos en la capa oculta.
- y_t es cada uno de los nodos en la capa de salida.
- W, U son los pesos (Weight) tanto de la capa como de las inyecciones.
- b son las bías.

Como se puede apreciar en las ecuaciones (29) y (31), las diferencias entre la red de Jordan y la de Elman es que el coeficiente aplicado al peso inyectado en las capas ocultas son en un caso la salida en esa capa en el instante anterior y en otro la salida final en el tiempo anterior.

8.10 Intersección sobre uniones

La IoU por sus siglas en inglés, es la operación que permite ampliar o definir un área más amplia ante la intersección de dos objetos. La regla aplicada consiste en la evaluación de la expresión matemática de la ecuación (33) y la validación de sus datos en base a un valor mínimo indicado como límite de aceptación. Por norma general, un IoU mayor a 0.5 es considerada como buena.

$$IoU = \frac{Área\ de\ solapamiento}{Área\ de\ la\ unión} \quad (33)$$

Imagen 64: Área de solapamiento

Imagen 65: Área de la unión

8.10.1 Cálculo de áreas

Para realizar el cálculo será necesario tanto el área de intersección como el de la unión. El primero de ellos, en de la intersección se calcula obteniendo los siguientes parámetros:

- X_1: Sera el máximo entre la coordenada «x» izquierda de la caja 1 y la caja 2.
- X_2: Sera el mínimo entre la coordenada «x» derecha de la caja 1 y la caja 2.
- Y_1: Sera el máximo entre la coordenada «y» superior de la caja 1 y la caja 2.
- Y_2: Sera el mínimo entre la coordenada «y» inferior de la caja 1 y la caja 2.

Una vez hallados estos valores se comprobará si existe solapamiento entre las cajas mediante las siguientes comprobaciones ($X_2>X_1$ e $Y_2>Y_1$). Si es correcto podremos calcular:

$$Área\ de\ solapamiento = A_S = (X_2 - X_1) * (Y_2 - Y_1) \quad (34)$$

$$Área\ de\ la\ unión = (X_A^2 * X_A^1 - Y_A^2 * Y_A^1) + (X_B^2 * X_B^1 - Y_B^2 * Y_B^1) - A_S \quad (35)$$

Siendo X e Y las coordenadas de cada caja, A y B cada una de las cajas y finalmente 1 y 2 el menor y mayor valor de cada coordenada.

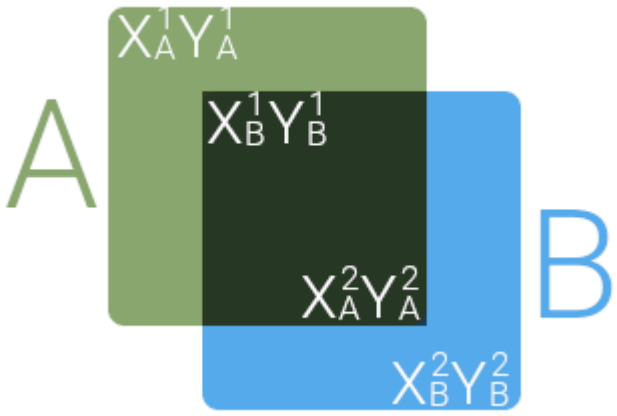


Imagen 66: Cálculo del IoU

Como se puede ver en la imagen de más atrás, en el cálculo del área de la unión se ha de restar el área de solapamiento para evitar contar dos veces esa zona.

8.10.2 Analizando resultados

Si analizamos más detenidamente la formula, veremos que los valores más próximos a 1 serán aquellos que dispongan de una mayor área de solapamiento, lo que obtendrá una proporción más elevada.

En el caso de que una de las zonas se encuentre incluida dentro de la otra el valor decaerá si la zona interior es muy pequeña en proporción a la zona exterior, en este caso se ha de valorar la idoneidad de unión de ambas zonas.

Imagen 67: Zona de selección interna

8.10.3 Algoritmo de cálculo

```
double  calcularIoU(int AX1,int AY1,int AX2,int AY2,int BX1,int BY1,int
BX2,int BY2) {

   int X1=(AX1>BX1)?AX1:BX1;

   int X2=(AX2<BX2)?AX2:BX2;

   int Y1=(AY1>BY1)?AY1:BY1;

   int Y2=(AY2<BY2)?AY2:BY2;

   if(X2<X1 || Y2<Y1){

      return 0;

   }

   double areaInterna=(X2-X1)*(Y2-Y1);

   double areaUnion=((AX2-AX1)*(AY2-AY1))+((BX2-BX1)*(BY2-BY1))-
areaInterna;

   return areaInterna/areaUnion;

}
```

Código 5: Método de cálculo de IoU

8.11 Transformación afín

Mediante las transformaciones se consigue adaptar objetos a un modelo determinado. En el libro "Visión por computador" (Pajares Martinsanz & de la Cruz García, 2007) en la página 76 se expone tanto el concepto de transformada afín como la formulación, aun así he creído interesante indicar aquí la utilizada por la librería OpenCV y documentada en la dirección https://docs.opencv.org/3.1.0/d4/d61/tutorial_warp_affine.html de cara a que este trabajo contenga toda la información relevante.

$$\begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = \begin{bmatrix} S_x \cos\theta & -\operatorname{sen}\theta & i_d \\ \operatorname{sen}\theta & S_y \cos\theta & j_d \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} i \\ j \\ 1 \end{bmatrix}$$

9 Anexo II

9.1 El proyecto

Para hacer frente al estudio de las diferentes técnicas descritas en este trabajo de investigación se ha desarrollado un código en C++ que incluye diferentes herramientas.

El desarrollo de este software ha sido pensado desde el punto de vista de su reutilización por otros desarrolladores que deseen ampliar el conjunto de herramientas disponibles. Para ello el sistema funciona en base a una serie de algoritmos desarrollados fuera de la propia aplicación y que pueden ser incluidos en esta tras su arranque.

No se ha incluido el código en este trabajo ya que el mismo se encuentra disponible en un repositorio público y dispondrá de la documentación necesaria para ponerlo en marcha o diseñar nuevos algoritmos.

Aunque la parte principal del software es el "visor", existen otros desarrollos incluidos dentro del paquete como "deteccionOpenCV" o "darknet" que sirven para probar Viola&Jones o Yolo.

9.1.1 Visor

El visor permite la carga y ejecución de algoritmos diseñados para el tratamiento de imágenes. Actualmente solo se dispone de la posibilidad de ejecución del algoritmo y el posterior guardado de resultados, estando limitados solo a aspectos gráficos, aunque con una previsión futura de obtención de datos numéricos y la consecuente posibilidad de análisis.

Los nuevos algoritmos cargados deberán ser construidos mediante la derivación de la clase abstracta "Algoritmo" disponible en el código.

A modo de ejemplos se incluyen los siguientes algoritmos desarrollados:

- Sobel: Implementa el operador Sobel de 3x3.
- Prewitt: Implementa el operado Prewit de 3x3.
- Viola_Jones: Este algoritmo se basa en OpenCV para detectar rostros de máximo 100x100.
- HOG: Implementación de HOG utilizando Sobel de 1D. Es una implementación totalmente propia.
- HOG_DLib: Implementa el Histograma mediante la utilización de DLib.
- HOG_Detect: Utiliza el detector de caras de DLIB con HOG.
- EPR: Obtiene la estimación de puntos de Referencia mediante CLM y realiza la transformación afín de nuestras caras. Los parámetros disponibles para esta librería serán:
    - Ampliar región: Con este número podremos ampliar el área de las imágenes localizadas.
    - Inicializar: Permite inicializar la configuración del método.
    - Algoritmo detección: Librería (uno de los algoritmos aquí descritos) que será utilizada para localizar caras.
    - Mascara: Fichero donde está definida la máscara de EPR.
    - Mostrar mascara: Indica si se ha de mostrar la máscara o no.
    - Transformar a mascara: Obtiene el recorte de la imagen y realiza la

transformación afín.
    - Modo de transformación: Indica el tipo de transformación utilizada. 0 para Ojos-Labio y 1 para Ojos-Nariz.
    - Ancho imagen afín: Ancho de la imagen afín generada.
    - Alto imagen afín: Alto de la imagen afín generada.
- PCA: Genera la media de los Eigenfaces.

9.1.1.1 Ejemplo de construcción y compilación

A continuación, se muestran los pasos necesarios para crear y compilar un nuevo algoritmo:

1. Se creará una clase que derive de Vision:: Algoritmo_plugin.
2. Se implementará necesariamente los métodos:
    a. getNombre: este devolverá el nombre del algoritmo que será mostrado en la lista de algoritmos disponibles en la aplicación "visor".
    b. procesar: este método recibe una imagen en formato cv::Mat (de OpenCV) y devuelve un cv::Mat que será mostrado por el "visor".
3. Registrar la clase.

Ejemplo.

```
namespace vision {

    class HOGDETECT: public Algoritmo_plugin {

       public:

          ~HOGDETECT(){};

          const char * getNombre()const{return "HOG Detect";};

          cv::Mat procesar(const cv::Mat & imgOriginal);

       private:

          void convertirGris(const Mat& , Mat& );

    };

}
```

Código 6: Definición de un nuevo algoritmo

```
cv::Mat HOGDETECT::procesar(const cv::Mat& imgOriginal) {

       Mat_<uchar> imgGris;

       convertirGris(imgOriginal, imgGris);


       vector<Rect_<double> > face_detections;

       dlib::frontal_face_detector face_detector_hog =
dlib::get_frontal_face_detector();


       vector<double> confidences;

       CLMTracker::DetectFacesHOG(face_detections, imgGris,
face_detector_hog, confidences);
```

```
        // Detect landmarks around detected faces
        int face_det = 0;
        cv::Mat  temp=imgOriginal.clone();

        for(size_t face=0; face < face_detections.size(); ++face){
           cv::Scalar colorCV(0,0,255);
           cv::rectangle(temp, face_detections[face], colorCV);
        }

        return temp;
    }
```

Código 7: Implementación de un nuevo algoritmo

```
REGISTRAR_PLUGIN(vision::HOGDETECT);
```

Código 8: Registro de un nuevo algoritmo

```
find_package( OpenCV REQUIRED )
include_directories(
   src
   ${OpenCV_INCLUDE_DIRS}
)
include_directories(../../herramientas/CLM-framework-
master/lib/local/CLM/include)
include_directories(../../herramientas/CLM-framework-
master/lib/local/FaceAnalyser/include)
#link_directories(${CMAKE_BINARY_DIR}/bin/)
file(GLOB codigos src/*.cpp)
add_library(hog_detect "SHARED" ${codigos})
if(WIN32)
    target_link_libraries( hog_detect ${OpenCV_LIBS} plugin)
elseif(UNIX)
    target_link_libraries( hog_detect ${OpenCV_LIBS} libplugin.so)
endif()
target_link_libraries( hog_detect dlib::dlib)
target_link_libraries(hog_detect CLM)
target_link_libraries(hog_detect FaceAnalyser)
add_custom_command(
```

```
    TARGET hog_detect POST_BUILD
    COMMAND ${CMAKE_COMMAND} -E copy_if_different
        ${CMAKE_CURRENT_BINARY_DIR}/../../../bin/hog_detect.dll
        ${CMAKE_SOURCE_DIR}/dll
)
```

Código 9: CMakeLists.txt de un nuevo algoritmo

9.1.2 deteccionOpenCV

Esta herramienta utiliza OpenCV y Viola-Jones para la detección de rostros, habilitando una serie de valores de configuración. Se ha utilizado para la generación de toda la documentación y pruebas de Viola-jones.

9.1.3 Darknet

Es el código de yolo adaptado para compilar en Windows.

9.2 Libarías y herramientas

Para la compilación del proyecto será necesaria una serie de herramientas algunas incluidas dentro del propio proyecto.

Las librerías necesarias y que se han de instalar en el sistema antes de poder compilar serán:

- Conan
- OpenCV

9.2.1 Conan

Conan es un gestor de paquetes para el desarrollo de aplicaciones en C/C++. Con este gestor podremos despreocuparnos de gestionar la descarga o compilación de los paquetes que maneje.

https://conan.io/

9.2.2 Boost

Este conjunto de librerías ofrece una gran variedad de funcionalidades con la ventaja de ser multiplataforma. Alguna de las funcionalidades que incluye son la gestión de ficheros, hilos, gestión de memoria, diversos tipos de algoritmos y más.

https://www.boost.org/

9.2.3 OpenCV

Es una librería de código abierto para la computación de imágenes (Open Source Computer Vision Library).

https://opencv.org/

9.2.4 Dlib

Esta librería contiene un conjunto de algoritmos de máquinas de aprendizaje y herramientas necesarias para realizar aplicaciones en C++.

http://dlib.net/

9.2.5 CLM-Framework

Este framework incluye las implementaciones de los algoritmos CLM, CLM-Z y CLNF, que entre otras permite la obtención de Puntos de Referencia.

Aunque este framework esté obsoleta ya que existe una versión más moderna OpenFace, he preferido utilizar esta versión por ser la primera, aunque se han tenido que hacer algunas adaptaciones para permitir su compilación con compiladores modernos.

- https://github.com/TadasBaltrusaitis/CLM-framework
- https://github.com/TadasBaltrusaitis/OpenFace

9.2.6 Yolo: Real-Time Object Detection

Es un código desarrollado en C++ con la implementación de una Red Neuronal de Convolución. Se encuentra en su versión 3 y ofrece altas prestaciones en la detección de objetos

https://pjreddie.com/darknet/yolo/

9.3 Entrenar a YOLO

Para entrenar nuestra red neuronal necesitaremos una buena colección de imágenes que permitan al sistema generar una buena clasificación.

En el apartado anterior vimos que es necesaria la creación de un fichero de configuración para la construcción de la red, punto que, aunque es crítico para el buen funcionamiento del clasificador no depende a la hora de realizar un entrenamiento.

Para obtener las imágenes de entrenamiento se puede hacer una labor de búsqueda en páginas de fotografía que o bien tengan libre uso o tengamos el derecho de los autores para su utilización. Otra alternativa es la obtención de imágenes de fuentes que ofrezca imágenes ya tratadas para el reconocimiento.

En la realización de este trabajo he utilizado las siguientes fuentes de imagen:

- FEI Face Database: Contiene 2.800 imágenes en las que cada persona (en total suman 200) tiene 14 fotografías en diferentes poses y grados de iluminación. Lo bueno de esta base de datos es que las caras de todas las imágenes se encuentran igualmente posicionadas y escaladas en su respectiva pose.
  - http://fei.edu.br/~cet/facedatabase.html
- Psychological Image Collection at Stirling (PICS): Contiene 2.677 imagenes agrupadas por ciudades escocesas.
  - http://pics.psych.stir.ac.uk/
- Labeled Faces in the Wild: Aunque contiene 26.466 en realidad solo son 13.233 ya que están divididas en imágenes originales y la misma algo alterada para poder realizar una prueba de detección. La ventaja de esta fuente es que las imágenes se encuentran clasificadas por el nombre de la persona lo que permite realizar pruebas de identificación.
  - http://vis-www.cs.umass.edu/lfw/#download

Aunque todas las fuentes son buenas para la realización del trabajo, para entrenar nuestra red neuronal he preferido utilizar la base de datos de FEI ya que al disponer de imágenes bien centradas me ha sido sencillo obtener la zona de detección del rostro.

Una vez que tenemos las imágenes que utilizaremos para realizar el entrenamiento, tenemos que o bien recortarlas para que solo contengan la información del rostro o indicarle a Yolo cual es la zona correcta.

Lo más sencillo es utilizar la segunda opción ya que de esta forma se puede mantener la imagen original, aunque indicando cual es realmente la zona de interés. Esta zona se configura mediante un fichero que ha de tener el mismo nombre que la imagen, pero con extensión "txt". El fichero contendrá una fila por cada uno de los clasificadores y con la siguiente información:

Id_clasificador x y w h [x y w h]

- El id del clasificador será el identificador comenzando por 0 de lo que representa el área delimitada.
- X, Y, W y H son las coordenadas del centro del recorte (x,y) y el ancho (w) y alto (h) de la caja. Pero deberán estar proporcionadas en función de ancho y alto de la imagen con el objetivo de que esas coordenadas sean siempre las mismas independientemente del escalado de la imagen.

Para calcular los valores proporcionados deberemos obtener el ancho de la imagen (W_i) el alto de la imagen (H_i) y las coordenadas x, y, w y h de la caja, luego aplicaremos las siguientes formulas:

$$x_f = \frac{x}{W_i} \tag{36}$$

$$y_f = \frac{y}{H_i} \tag{37}$$

$$w_f = \frac{w}{W_i} \tag{38}$$

$$h_f = \frac{h}{H_i} \tag{39}$$

Como se puede intuir de las formulas anteriores el resultado de las coordenadas será siempre entre 0 y 1.

Mediante la herramienta "Recortador" se puede ver o generar los ficheros de recorte.

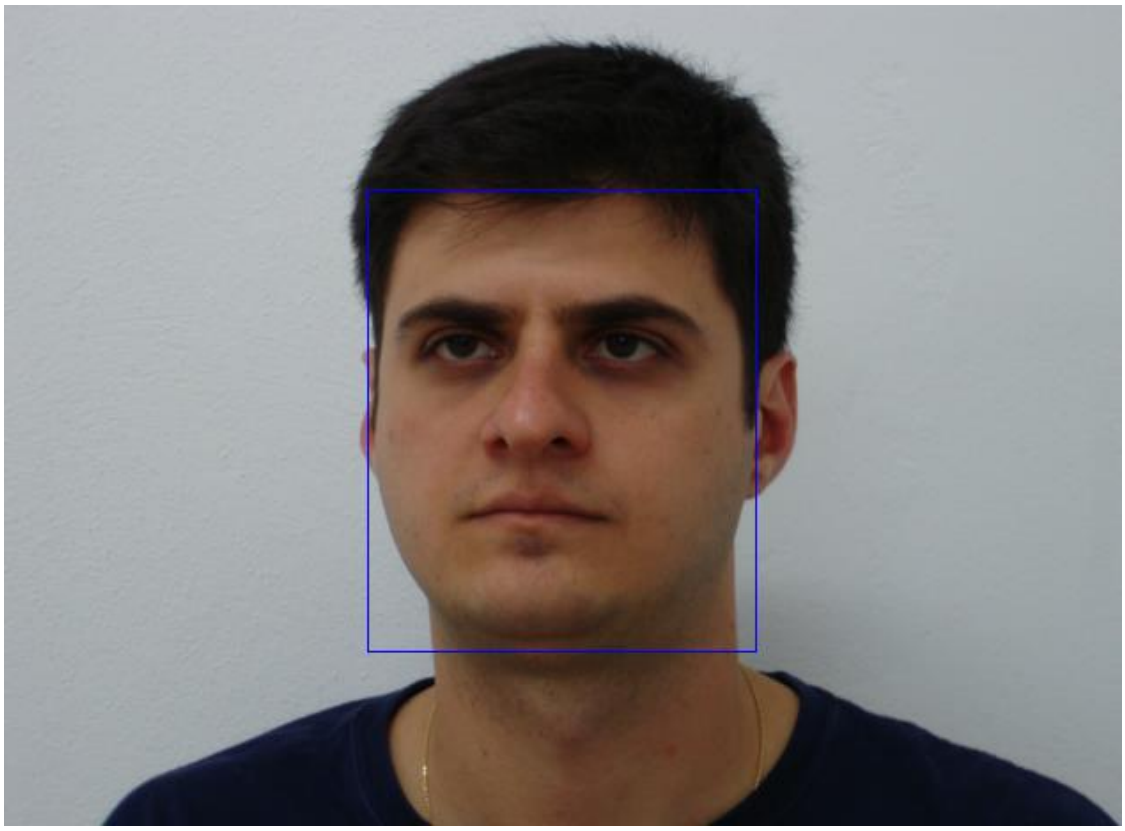


Imagen 68: Recorte del rostro. Fichero 1-05.jpg

El contenido del fichero "1-05.txt" es:

```
0 0.49296875 0.5062419 0.3421875 0.5458246
```

Lo anterior indica que se utilizará el primer clasificador. Y tras aplicar las coordenadas tenemos:

- W_i=640
- H_i=480
- X=315
- Y=243
- W=219
- H=262

Una vez tenemos los recortes, deberemos preparar las imágenes de entrenamiento y las de muestra. Estas últimas serán utilizadas por la red para comprobar la validez de la misma y adaptar los valores en caso de ser negativa. Con lo anterior estaremos realizando el aprendizaje.

Para indicar a Yolo cuales son las imágenes de entrenamiento y las de test, crearemos dos ficheros que tendrán un listado de las direcciones de las imágenes. En este punto tenemos que ser precavidos puesto que la ruta ha de ser absoluta.

Ahora configuraremos un fichero que le indicará a Yolo todos nuestros ficheros.

```
classes= 1

train  = e:/Vision/yolo/rostros/train2WIN.txt
```

```
valid  = e:/Vision/yolo/rostros/test2WIN.txt

labels = e:/Vision/yolo/rostros/cfg/obj.names

backup = e:/Vision/yolo/rostros/backup3/
```

Código 10: objWIN.data Fichero de configuración

El fichero obj.names tan solo tiene en cada fila el nombre del clasificador.

```
CARAS
```

Código 11: obj.names Fichero con la lista de nombres

La carpeta backup contendrá los resultados del entrenamiento. Respecto a esto, el entrenador por defecto almacena cada 100 iteraciones un backup, en mi caso al no disponer de GPU el proceso se demoraba tanto que realice ajustes en el código de Yolo para que el almacenamiento de la copia de seguridad se realice además cada 10 pasos.

Este backup nos permite relanzar el entrenamiento en caso de que se haya parado.

10 Bibliografía

Dalal, N., & Triggs, B. (2005). Histograms of Oriented Gradients for Human Detection. *IEEE Computer Society Coference*.

Elman, J. (1990). Finding Structure in Time. *Cognitive Science, v. 40*, 179-211.

Etemad, K., & Chellappa, R. (1997). *Discriminant analysis for recognition of human face images.* Journal of the Optical Society of America A.

Felzenszwalb, P., Girshick, R., McAllester, D., & Ramana, D. (2010). Object Detection width Discriminatively Trained Part Based Models. *IEEE Transactions on Pattern Analysis and Machine Intelligence, Vol. 32, No. 9*.

Freund, Y., & Schapire, R. (1997). A Decision-Theoretic Generalization of On-Line Learning. *Journal of computer and system sciences*, 119-139.

Haar, A. (1909). Zur Theorie der orthogonalen Funktionensysteme. *Mathematische Annalen*, 331-371.

Hoffmann, U., Naruniec, J., Yazdani, A., & Ebrahimi, T. (2008). *Face detection using discrete gabor jets and color information.* Int. Conf. on Signal Processing and Multimedia Applications.

Jordan, M. (1997). Serial Order: A Parallel Distributed Processing Approach. *Advances in Psychology, V. 121*, 471-495.

Kamran, E., & Chellappa, R. (1997). *Discriminant analysis for recognition of human face images* (Vol. 14). J. of the Optical Society of America.

Kanade, T. (1974). Picture processing system by computer complex and recognition of human faces.

Kanade, T. (1977). *Computer recognition of human faces.* Obtenido de CiteSeerX: http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.73.1766&rep=rep1&type=pdf

Karhunen, K., & Loève, M. (s.f.). *Wikipedia.* Obtenido de Teorema de Karhunen-Loève: https://es.wikipedia.org/wiki/Teorema_de_Karhunen-Lo%C3%A8ve

Kazemi, V., & Sullivan, J. (2014). One Millisecond Face Aligment with an Ensemble of Regression Trees. *Computer Vision and Active Perception Lab*.

Kosinski, M., & Wang, Y. (s.f.). *Stanford Business*. Obtenido de Deep Neural Networks Are More Accurate Than Humans at Detecting Sexual Orientation From Facial Images: https://www.gsb.stanford.edu/faculty-research/publications/deep-neural-networks-are-more-accurate-humans-detecting-sexual

Lawrence, S., & Kirby, M. (1987). *Low-dimensional procedure for the characterization of human faces* (Vol. 4). J. of the Optical Society of America.

le Cun, Y. (1989). *Generalization an Network Design Strategies.* Toronto, Ontario, M5S 1A4. CANADA: Department of Computer Science, University of Toronto.

McCulloch, W., & Pitts, W. (1943). A logical calculus of the ideas immanent in nervous activity. *Bulletin of Mathematical Biophysics, vol. 5*, 115-133.

OpenCV team. (s.f.). *OpenCV*. Obtenido de Open Source Computer Vision Library: https://opencv.org

Pajares Martinsanz, G., & de la Cruz García, J. (2007). *Visión por computadora.* Madrid: Ra-Ma.

Redmon, J., & Farhadi, A. (2017). YOLO9000: Better, Faster, Stronger. *University of Washington, Allen Institute for AI*.

RosenBlatt, F. (1958). The perceptron: A probabilistic model for information storage and organization in the brain. *Psychological Review, Vol. 65, Nº 6*.

Rowley, H., Baluja, S., & Kanade, T. (1998). *Neural network-based face detection* (Vol. 20). IEEE Patt. Anal. Match. Intell.

Rumelhart, D., Hinton, G., & Williams, R. (1986). Learning representations by back-propagating errors. *Nature, vol. 323*, 533-536.

Turk, M., & Pentland, A. (1991). *Face Recognition using Eigenfaces.* IEEE Conf. on Computer Vision an Pattern Recognition.

Viola, P., & Jones, M. (2001). *Rapid Object Detection using a Boosted Cascade of Simple Features.* Obtenido de https://www.cs.cmu.edu/~efros/courses/LBMV07/Papers/viola-cvpr-01.pdf

Wikipedia. (s.f.). *Aplicación lineal*. Obtenido de https://es.wikipedia.org/wiki/Aplicaci%C3%B3n_lineal

Wikipedia. (s.f.). *Eigenface*. Obtenido de https://en.wikipedia.org/wiki/Eigenface

Wikipedia. (s.f.). *Ronald Fisher*. Obtenido de https://es.wikipedia.org/wiki/Ronald_Fisher

Wiskott, L., Fellous, J.-M., Krüger, N., & von der Malsburg, C. (1999). Face Recognition by Elastic Bunch Graph Matching. In *Intelligent Biometric Techniques in Fingerprint and Face Recognition* (pp. 355-396). CRC Press.

Zhou, Y., & Chellappa, R. (1988). Neural network approach to computation of optical flow. *Neural Networks*, 71-78.

Índice