Universidad del País Vasco Euskal Herriko Unibertsitatea
BILBOKO INGENIARITZA ESKOLA
ESCUELA DE INGENIERÍA DE BILBAO

GRADO EN INGENIERÍA EN TECNOLOGÍA DE LA TELECOMUNICACIÓN

TRABAJO FIN DE GRADO

SISTEMA DE RECONOCIMIENTO FACIAL

Alumno: Basañez, De La Rica, Diego

Director: Espinosa, Acereda, Koldo

Curso: 2017-2018

Fecha: Bilbao, 20 de Julio de 2018

Índice

Índice de figuras

Índice de tablas

Resumen

Este trabajo estudia e implementa un sistema de detección y reconocimiento facial que trabaja con imágenes fijas introducidas en una base de datos. Las bases de datos utilizadas son bases de datos de libre utilización a la comunidad científica.

En primer lugar, se realiza un estudio de diversas técnicas utilizadas hasta la actualidad reflejados en los apartados de estado del arte y análisis de alternativas. Tras dicho estudio, se seleccionará una de dichas técnicas para ser implementada en el lenguaje C++ con el fin de diseñar un proyecto capaz de identificar a distintos sujetos registrados en una base de datos. En este caso, se implementará el método Eigenfaces, que es un método basado en el principio de Análisis de Componentes Principales de I. T. Jolliffe.

Abstract

This work studies and implements a system of detection and facial recognition that works with static images introduced in a database. The databases used are freely available databases to the scientific community.

Firstly, a study is made of various techniques used to date reflected in the sections on the state of the art and analysis of alternatives. After this study, one of these techniques will be selected to be implemented in the C ++ language in order to design a project capable of identifying different subjects registered in a database. In this case, the Eigenfaces method will be implemented, which is a method based on the Principal Component Analysis principle of I. T. Jolliffe.

Laburpena

Lan honek datu base batean sartutako irudi finkoekin lan egiten duen detektatzeko eta aurpegiko aintzatespen sistema bat ikertu eta inplementatzen du. Erabilitako datu baseak komunitate zientifikorako datu-baseak dira libreki.

Lehenik eta behin, artearen egoerari buruzko ataletan islatutako egungo erabilerarako hainbat teknika eta alternatiben analisia egiten dira. Ikasketa honen ondoren, teknika horietako bat hautatuko da C ++ hizkuntzan garatzeko, datu-base batean erregistratutako gai desberdinak identifikatzeko proiektu bat diseinatzeko. Kasu honetan, Eigenfaces metodoa ezarriko da, hau da I. TJ Jolliffe osagai nagusien analisiaren oinarrian oinarritutako metodoa.

1. Introducción

El reconocimiento facial ha avanzado de la mano con la tecnología en los últimos años. Debido a la inmensa utilidad del reconocimiento facial en sectores como la seguridad, la vigilancia o la biometría, durante los últimos años se han desarrollado nuevas técnicas para lograr un método rápido, seguro y viable.

El reconocimiento facial involucra tanto a informáticos como a neurocientíficos y psicólogos. Es un apartado específico dentro del área de reconocimiento de objetos, donde la cara es el objeto tridimensional a reconocer, pudiendo sufrir variaciones de pose, iluminación…

El objetivo del reconocimiento facial es simple y común a casi todas sus aplicaciones. Dada una cara desconocida o una imagen de test, encontrar una imagen de la misma cara en un conjunto de imágenes conocidas o imágenes de entrenamiento. La dificultad de este proceso reside en la capacidad de realizar este proceso en el menor tiempo posible y con gran fiabilidad. El reconocimiento puede realizarse de dos maneras distintas:

- Verificación o autentificación de caras. Compara una imagen de la cara con otra imagen con la cara de la que queremos saber la identidad. El sistema confirmará o rechazará la identidad de esa cara.
- Identificación o reconocimiento de caras. Compara la imagen de una cara desconocida con todas las imágenes de caras conocidas que se encuentran en una base de datos para determinar su identidad.

2. Contexto

La entrada de los robots en las factorías, el desarrollo del Big Data y la inteligencia artificial enfrenta a las corporaciones al reto de afrontar rápidamente su transformación digital o ceder el puesto a los nuevos actores tecnológicos.

No hay empresa, sector o industria que en los últimos años haya escapado al impacto de la revolución digital. La tecnología ha irrumpido en la era moderna con la misma fuerza que lo hizo la máquina de vapor en la época industrial. Y es que estamos ante la llamada cuarta revolución industrial, que según cálculos del Foro Económico Mundial acabará con cinco millones de empleos en 2020 dentro de los quince países más industrializados. Pero, al igual que muchos empleos como los conocemos hoy en día desaparecerán, muchos otros aparecerán.

Muchas son las start ups que han aprovechado deficiencias tecnológicas en los empleos tradicionales para ganarse un hueco entre empresas de alta reputación como Uber, Just Eat o Airbnb.

Existen 3 motivos principales por los que se ha dado esta revolución:

- La digitalización de la información. Todo lo que hacemos deja una huella digital. Cuando accedemos a una aplicación, realizamos un pago con tarjeta, leemos la prensa online, pasamos por delante de una cámara de seguridad o compramos un billete, generamos datos que indican qué hemos hecho, dónde y cuándo. Es decir, en el mundo digital existe una representación más exhaustiva de lo que ocurre en el mundo físico.

- La ley de Moore y sus implicaciones. La democratización del acceso a infraestructuras cada vez más asequibles pone al alcance de todos la potencia de computación necesaria para tratar millones de interacciones en un instante. La capacidad de proceso y almacenamiento ha dejado de ser un privilegio de las grandes corporaciones.
- Las expectativas cada vez más exigentes de los clientes y usuarios. Los clientes y usuarios no están dispuestos a esperar por lo que quieren y buscan productos y servicios instantáneos y personalizados a sus necesidades y su contexto. Los datos dan voz a las necesidades de los clientes y las empresas que demuestren no conocerlos se encontrarán con los obstáculos propios de la navegación a ciegas.

Vistas las previsiones, parece que todo el mundo tiene claro que el futuro del mundo pertenece mayormente a la tecnología. Pero solo unas pocas personas se plantean qué hace falta para que ese futuro sea buen sitio en el que cyber-habitar.

No cabe duda de que los expertos en el sector de la Seguridad Informática deben ser los que presten mayor atención para ser capaces de proporcionar métodos asequibles que proporcionen al usuario un alto grado de seguridad. Es aquí donde entra en juego la Biometría como método de seguridad digital.

Preocuparse por la seguridad de la información en línea es cada vez más común y necesario. A los usuarios les inquieta que alguien pueda acceder a sus datos y cuentas o incluso a suplantar su identidad. En vista del problema, muchas empresas se han dedicado a ofrecer alternativas en la autenticación que den a los usuarios mayor seguridad y control sobre su información.

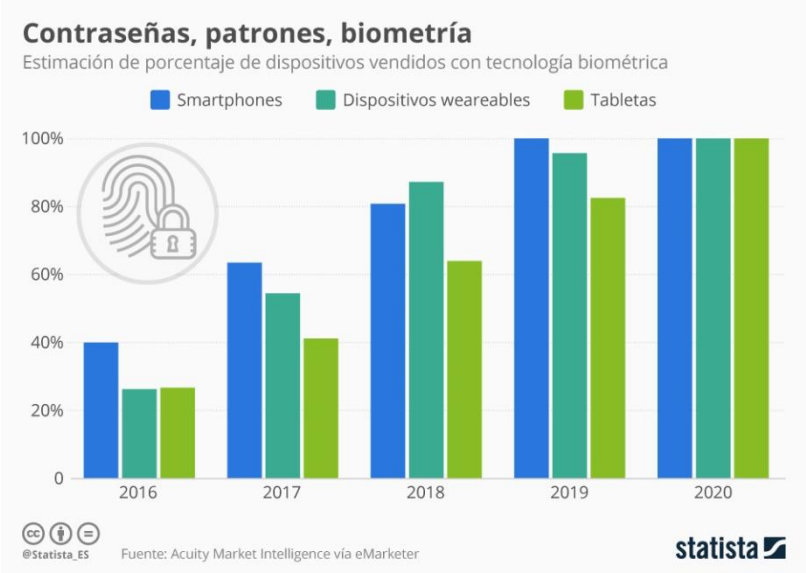

Figura 1: Estimación de porcentaje de dispositivos vendidos con tecnología biométrica (fuente: 1)

Hace tan sólo unos meses, Apple revolucionaba la industria tecnológica con el lanzamiento de Face ID, un nuevo sistema de autenticación basado en el reconocimiento facial. Que cada vez coge más fuerza en este sector.

Otro sector donde controles de acceso severos son requeridos y en el cual la biometría parece la solución definitiva son las entidades financieras. Donde las grandes empresas se están reinventando para lograrlo.

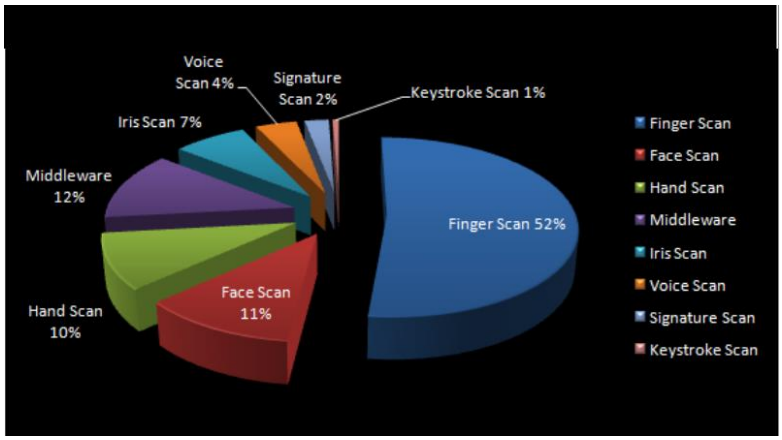

Figura 2: Porcentajes de métodos de autenticación utilizados en grandes entidades financieras (fuente: 2)

Por último, la biometría se presenta como una solución para dar fluidez a la vez que seguridad en situaciones en las que la agilidad del proceso toma un papel clave. La reglamentación antiterrorismo establecida en abril del 2017 por la comisión Europea presentó grandes obstáculos en los aeropuertos de medio mundo.

Esta reglamentación establece que los pasajeros deben enseñar los pasaportes al entrar y salir del espacio Schengen, lo que provoca grandes ralentizaciones del sistema de control.

Aeropuertos asiáticos, americanos o australianos, han introducido métodos de identificación por biometría como alternativa al control de pasaporte tradicional. El reconocimiento facial se convierte en la alternativa más significativa aunque está pendiente de mejorar en varios de ellos.

Figura 3: Sistema de acceso con reconocimiento facial (fuente: 3)

Otro sector en el que el reconocimiento facial está muy expandido es en la vigilancia en casinos, tanto para controlar a los empleados como para detectar a usuarios vetados con anterioridad. Uno de los que tienen más éxito es el casino "Venetian" donde tienen implantado un sistema automático de reconocimiento de caras para autenticar de forma segura y eficiente a los 12.000 empleados del resort.

2.1. Posibles aplicaciones del reconocimiento facial

El reconocimiento facial, en las últimas décadas, se ha convertido en un tema de investigación multidisciplinar, involucrando a investigadores de muchas áreas, de la informática y neurocientíficos, por su naturaleza poco intrusiva (solo es necesaria una fotografía del sujeto).

Este tipo de sistemas siguen siendo estudiados a pesar del uso de otros métodos muy fiables de identificación, como el análisis de huellas dactilares, pero estos últimos requieren de la colaboración del sujeto a reconocer.

Así pues, el reconocimiento facial tiene aplicaciones muy diversas, pero sobre todo relacionado con la seguridad, y el control de sujetos.

- Biometría. El control de pasaportes o DNIs en aeropuertos o edificios donde sea requerida su confirmación sería mucho más rápida, evitando así largas esperas o retrasos, así como mayor viabilidad del procesado.
- Seguridad. Ya se ha implementado en nuevos smartphones, pero el reconocimiento facial podría ir más allá, dejando atrás la introducción de contraseñas en distintas cuentas o tarjetas de crédito o incluso la necesidad de llaves en las viviendas.
- Vigilancia. Localizar sujetos, ya sean peligrosos, sospechosos, o simplemente la madre de un crio perdido en un lugar transitado.

3. Motivación

El reconocimiento facial es un proceso que nuestros ojos y nuestra mente realiza infinidad de veces a lo largo del día sin ni siquiera darnos cuenta.

La facilidad de adquisición de ordenadores y el avance tecnológico en el sector de la fotografía ha producido un enorme interés en el proceso de automatizar el tratamiento de imágenes y vídeo. Existe un sinfín de aplicaciones que demandan procesos automáticos para la identificación de individuos.

Combinar la capacidad humana de reconocer a un individuo conocido con la capacidad de almacenar infinidad de datos de dispositivos electrónicos, representaría un gran avance en el mundo de la fotografía y biometría.

Aunque los progresos en el reconocimiento de caras han sido alentadores, el reconocimiento ha resultado una tarea difícil, debido a que las imágenes varían considerablemente según el punto de vista, iluminación, expresión, pose, accesorios…

4. Objetivos

El objetivo principal del proyecto es realizar una aplicación en C++ capaz de reconocer a un sujeto dentro de una base de datos con la que la aplicación ha sido entrenada.

Para conseguirlo, la aplicación deberá cumplir los siguientes objetivos:

- Detectar caras de personas en diferentes imágenes.
- Disminuir dimensionalidades de imágenes sin perder la información más importante.
- Dar valores numéricos al parecido entre dos sujetos.

5. Beneficios

5.1. Beneficios sociales

El presente proyecto será capaz de reducir aglomeraciones de personas en cualquier recinto en el que un proceso de identificación sea requerido, como aeropuertos, estaciones de tren, exámenes etc. Una vez la persona esté registrada en una base de datos, la persona será fotografiada por una cámara móvil y posteriormente identificada por el programa, sin necesidad de personal comprobando su parecido con el documento de identidad.

Además, la falsificación de documentos no podrá ser posible debido a que el programa se fija en rasgos faciales y a pesar de operaciones faciales, la persona ha de estar registrada en la base de datos por lo que un nuevo rostro deberá volver a pasar por el control del registro.

5.2. Beneficios tecnológicos

Este proyecto supondrá un avance tecnológico en recintos de alto tránsito diario, en los que será necesaria una digitalización del establecimiento y dotará de mayor control y seguridad. La automatización del personal de control de acceso dotará de dinamismo al proceso, y motivará a distintos establecimientos a invertir en una digitalización completa.

Además, el presente trabajo será tenido en cuenta para futuros estudios sobre la viabilidad de distintas técnicas de reconocimiento facial, a la hora de decidir la utilidad de cada una de ellas.

5.3. Beneficios económicos

En lo que respecta a lo económico el bajo coste del software capaz de realizar el reconocimiento facial del individuo sería mínimo. La inversión económica que sería necesaria, es la de adquirir el material necesario para la toma de fotografías de sujetos como cámaras de seguridad y de acceso, así como de control de acceso, como barreras, tornos…

6. Estado del arte

El reconocimiento facial automatizado es un concepto relativamente nuevo, pues se introdujo en los años 60. Fue entonces cuando W. W. Bledsoe [1] desarrolló el primer sistema semiautomático para reconocimiento facial, el cual requería imagen de una persona para localizar los rasgos (ojos, nariz, boca) en las fotografías antes de que éste calculara distancias a puntos de referencia en común, que posteriormente eran comparados con datos de referencia.

En los años 70 Goldstein, Harmon & Lesk [2], utilizaron 21 marcadores subjetivos específicos tales como el color del cabello y el grosor de labios para automatizar el reconocimiento facial. El problema con estas soluciones previas era que seguían requiriendo un proceso manual. En 1988 Kirby & Sirobich [3] aplicaron análisis de componentes principales (PCA), una técnica estándar del algebra lineal, al problema del reconocimiento facial, para aumentar la exactitud de los resultados. Esto fue considerado un avance muy importante al mostrar que eran requeridos menos de 100 valores para codificar acertadamente la imagen de una cara convenientemente alineada y normalizada.

En 1991 Turk & Pentland [4], utilizando las técnicas de Eigenfaces, como se denominó al método de Kirby & Sirobich, demostraron que el error residual podía ser utilizado para detectar caras en las imágenes, un descubrimiento que permitió desarrollar sistemas automatizados fiables de reconocimiento facial en tiempo real. Si bien la aproximación era un tanto forzada por factores ambientales, creó sin embargo un interés significativo en posteriores desarrollos de estos sistemas.

A partir de estas investigaciones, el interés por técnicas de reconocimiento facial incrementó considerablemente hasta desarrollar nuevas técnicas en este ámbito.

El pixel es el elemento mínimo de una imagen, y tiene determinadas características (color, intensidad…)

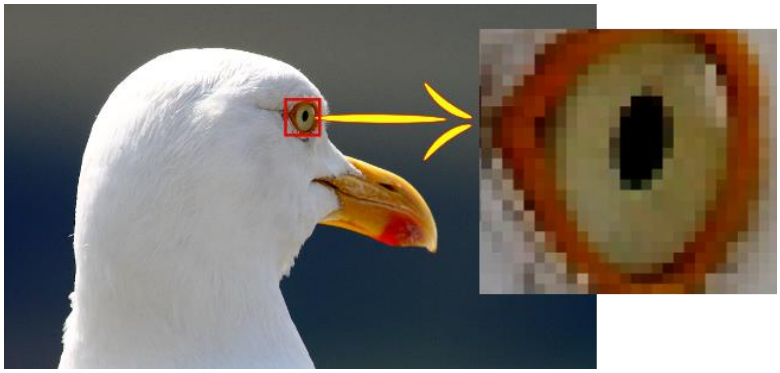

Figura 4: Pixeles del ojo de ave (fuente: 4)

Una imagen es una función de dos dimensiones que representa la intensidad de luz de la imagen en el punto (x,y).

Una imagen digital es una imagen f(x,y), donde f es el valor del brillo, que se ha discretizado en los dos ejes y cuantizado en brillo, estas imágenes se almacenan en matrices donde cada celda es un pixel.

Dependiendo de los valores que pueda tomar hay 3 tipos:

- Imágenes binarias. El pixel solo puede tomar valor 1(blanco) o 0 (negro)
- Imágenes en escala de grises. En la que el pixel puede tomar 256 valores distintos dependiendo de la intensidad del gris.
- Imágenes a color. Formadas por 3 matrices de pixeles, por lo que cada pixel tiene 3 valores distintos.

A la hora de realizar el estudio del estado del arte se han observados distintos métodos existentes para las etapas fundamentales del proyectos. La detección de rostros, la extracción de características y la clasificación de la información.

6.1. Detección de rostros

La fase de detección de rostros se basa en encontrar diversas áreas de la imagen que contengan rostros aislados, para después procesarlos. Este proceso es muy importante, debido a que una mala detección del rostro del sujeto conllevaría a no poder realizar el resto del proceso.

6.1.1. Algoritmo de Viola-Jones

El algoritmo de Viola-Jones presento en 2001 un enfoque para la detección de objetos que minimizó el tiempo de cálculo, logrando una mayor exactitud [5]. Viola y Jones presentaron un conjunto de experimentos detallados en un conjunto de datos de detección de caras difíciles de reconocer. Teniendo altos rangos de las siguientes características: iluminación, escala, pose y variación de cámara.

Actualmente es uno de los algoritmos más utilizados en detección de caras. El método de Viola-Jones es un método de aproximación basado en la apariencia. Está dividido en dos etapas: una primera etapa de aprendizaje del clasificador basada en un gran número de ejemplos positivos (los objetos de interés, como por ejemplo las caras) y de ejemplos negativos, y una fase de detección mediante la aplicación de este clasificador a las imágenes no conocidas.

En vez de utilizar el valor de cada pixel, Viola y Jones deciden basar el algoritmo en características más simples dentro de la imagen. La principal razón es que de esta manera las características se pueden utilizar para codificar datos que con una cantidad finita de datos de entrenamiento sería mucho más complejo.

6.1.1.1. Haar-like features

Estas características son calculadas como la diferencia de la suma de los píxeles de dos o más zonas rectangulares adyacentes, basándose en intensidades luminosas. Viola y Jones utilizan distintas características como las que aparecen representadas en la figura 5:

- La característica-dos-rectángulos. Diferencia entre la suma de los píxeles de dos rectángulos. Estos rectángulos tienen la misma forma y son adyacentes vertical u horizontalmente.
- La característica-tres-rectángulos. Calcula la suma de los píxeles dentro de dos rectángulos exteriores, a su vez restados de la suma de un tercer rectángulo interior.

Figura 5: Algoritmos Haar utilizados en OpenCV (fuente: 5)

6.1.1.2. Imagen integral

Los autores proponen el uso de una imagen intermedia para calcular con rapidez las características rectangulares. Esta imagen se denomina imagen integralgen integral, que es una representación en forma de imagen del mismo tamaño que la imagen original, donde cada uno de los puntos se calcula como la suma de los píxeles situados por encima de él y a su izquierda.

$$ii(x,y) = \sum_{x \leq x', y' \leq y'} i(x', y')$$

En la ecuación, ii(x,y) es la imagen integral e i(x,y) es la imagen original.

Para implementar (1), se utilizan el siguiente par de ecuaciones:

$$s(x,y) = s(x,y-1) + i(x,y)$$

$$ii(x,y) = ii(x-1,y) + s(x,y)$$

Donde s(x,y) es la suma acumulativa de las filas, siendo s(x,-1=0) e ii(-1,y)=0. Con este par de ecuaciones se puede calcular la imagen integral en una sola pasada de la imagen original.

Con la imagen integral ya calculada, cualquier suma rectangular puede ser calculada utilizando cuatro puntos de la matriz. En el siguiente ejemplo, el valor del punto 1 es el de la suma de todos los píxeles de la zona A, el de 2 es A + B, el de 3 es A + C, y el de 4 es A + B + C + D, por lo que si queremos conocer la suma de los píxeles dentro del rectángulo D, se puede calcular cómo 4+1-(2+3).

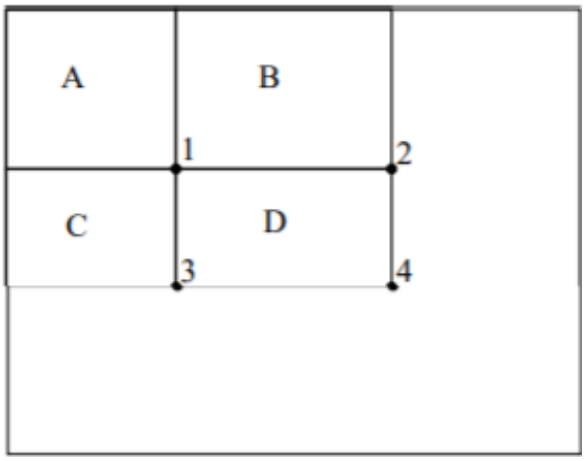

Figura 6: Ejemplo imagen integral (fuente: articulo referencia [5])

6.1.1.3. Entrenamiento del clasificador

Para el entrenamiento del clasificador Viola y Jones utilizan una variante de AdaBoost para seleccionar un pequeño conjunto de características [6]. La idea original de este algoritmo era mejorar el rendimiento de clasificación de un algoritmo de aprendizaje simple. Este algoritmo, combina clasificadores simples uno detrás de otro y selecciona las características de entrenamiento de cada uno de ellos, para después combinarlas y lograr un clasificador mucho más elaborado y preciso.

En las características elegidas por AdaBoost se miden distancias, como entre ojos y mejillas e intensidades, como la de la región del ojo.

6.1.1.4. Cascada de clasificadores

Para finalizar, Viola y Jones proponen un esquema basado en una cascada de clasificadores que logra un mayor rendimiento de detección a la vez que reduce radicalmente el tiempo de cálculo. La idea clave es que pueden construirse clasificadores más pequeños, y por lo tanto más eficientes, que rechazan muchas de las ventanas secundarias negativas y detectan casi todas las instancias positivas (es decir, el umbral de un clasificador potenciado puede ajustarse para que la tasa de falsos negativos sea cerca de cero). Los clasificadores más simples se usan para rechazar la mayoría de las subventanas antes de recurrir a clasificadores más complejos para lograr tasas bajas de falsos positivos

Cada etapa de la cascada corresponde a un clasificador, y se entrena agregando características nuevas hasta que se cumplan la detección de objetivos y los índices de falsos positivos (estas tasas se determinan al probar el detector en un conjunto de validación). Es decir, cada etapa es capaz de detectar objetos con mayor precisión que la anterior. Por ejemplo si deseamos detectar una cara las primeras etapas buscaran por la forma e iluminación mientras que las últimas serán capaces de descartar globos, pelotas etc.

6.1.2. Algoritmo de Kanade-Lucas-Tomasi

El algoritmo completo fue explicado en el documento escrito en 1994 por Jianbo Shi y Carlo Tomasi [7] es una implementación en el lenguaje C++, de un rastreador de funciones para la

comunidad “computer vision”. El código fuente está en el dominio público, disponible para uso comercial y no comercial.

KLT hace uso de la información de la intensidad espacial para dirigir la búsqueda de la posición que produce la mejor coincidencia. Es más rápido que las técnicas tradicionales para examinar muchas coincidencias potenciales entre imágenes.

El rastreador de características KLT se basa en dos documentos: en el primer documento, Lucas y Kanade [8] desarrollaron la idea de una búsqueda local utilizando gradientes ponderados por una aproximación a la segunda derivada de la imagen.

En el segundo documento, Tomasi y Kanade [9] utilizaron el mismo método básico para encontrar el registro debido a la traducción, pero mejoraron la técnica al rastrear características que son adecuadas para el algoritmo de seguimiento. Las características propuestas se seleccionarían si los valores propios de la matriz de gradiente fueran mayores que algún umbral.

En un tercer artículo, Shi y Tomasi propusieron una etapa adicional para verificar que las características se rastrearan correctamente. Este tercer y último artículo fue la recopilación de los mencionados anteriormente sumándole esa etapa adicional.

Se ajusta una transformación afín entre la imagen de la función rastreada actualmente y su imagen desde un fotograma anterior no consecutivo. Si la imagen afín compensada es muy diferente, la característica se descarta.

El razonamiento es que, entre fotogramas consecutivos, una traducción es un modelo suficiente para el seguimiento, pero debido a movimientos más complejos, efectos de perspectiva, etc., se requiere un modelo más complejo cuando los fotogramas están más separados.

6.2. Extracción de características

Es utilizada para obtener la información más importante de la cara. Dicha información es la que será posteriormente utilizada para realizar el reconocimiento del sujeto.

Existen dos tipos de técnicas de reconocimiento. Las basadas en la apariencia y las basadas en los modelos.

6.2.1. Métodos basados en la apariencia (holísticos)

6.2.1.1. PCA (Principal component Analysis)

El método denominado Eigenfaces, desarrollado por Matthew A. Turk y Alex P. Pentland basado en el principio de análisis de componentes principales de I.T.Jolliffe [10] se basa en la creación de un subespacio en la que las imágenes son representadas basándose solamente en las características más relevantes.

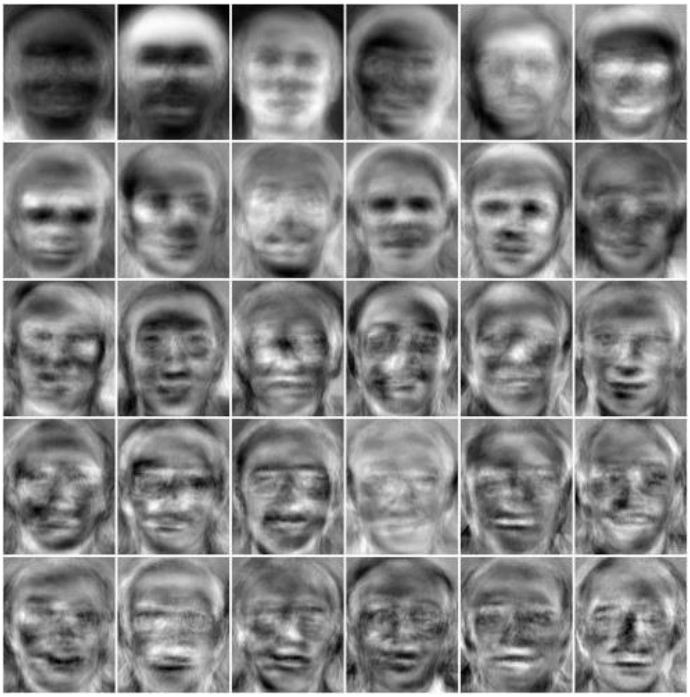

Figura 7: Colección de eigenfaces (fuente: 6)

Es una técnica usada para reducir la dimensionalidad de un conjunto de datos. La técnica sirve para hallar las causas de la variabilidad de un conjunto de datos y ordena estos por su importancia, busca la proyección para que los datos queden mejor representados en términos de mínimos cuadrados. Comporta el cálculo de la descomposición en autovalores de la matriz de la covarianza, tras centrar los datos en la media de cada atributo.

El algoritmo funciona de la siguiente manera:

- Obtener un conjunto de datos de dimensión n.
- Calcular la media de todos los datos y restarla a cada uno de ellos, así se obtienen unos datos de media cero.
- Calcular la matriz de la covarianza.
- Calcular los eigenvectores (vectores propios) y eigenvalores (valores propios) de la matriz de covarianza.
- Elegir los componentes más importantes según sus pesos y formar una matriz característica. Se ordenan de mayor a menor los eigenvalores, y se eligen los p eigenvectores ($p<n$) correspondientes a los mayores eigenvalores. De esta forma se reduce la dimensión del espacio vectorial
- Por último, obtener el nuevo conjunto de datos. Los datos introducidos se multiplican por la matriz característica, y así se representarán los datos en función de los eigenvectores elegidos.

PCA es un método estadístico de análisis de datos y caracterización de los mismos, que al aplicarse al reconocimiento facial, desarrolla alguna variación, esta puede llamarse Eigenfaces.

6.2.1.2. LDA (Linear discriminant Analysis)

Derivada del PCA, desarrollado por P. Belhumeur, J. Hespanha y D. Kriegman (1996) [11] dota de mayor fiabilidad al reconocimiento pero su computación es más costosa. Se basa en el discriminante lineal de Fisher (FLD).

Figura 8: Colección de fisherfaces (fuente: 7)

Este método es una técnica de aprendizaje supervisado para clasificar datos. Permite encontrar combinaciones lineales de características que caracterizan o separan dos o más clases de objetos o eventos. La combinación resultante puede ser usada como clasificador lineal, o más comúnmente, para la reducción del tamaño del problema antes de la clasificación. Es una técnica supervisada ya que para poder buscar esa proyección se debe entrenar el sistema con patrones etiquetados.

Se basa en que se dispone un conjunto de caras de entrenamiento compuesto por un grupo de personas con distintas expresiones faciales y con diferentes vistas. Todas las caras de la misma persona estarán en una clase, por lo que habrá c clases igual al número de personas a reconocer, separando así el espacio de entrenamiento por grupos. Además todas las instancias en el conjunto de entrenamiento deben estar etiquetadas.

6.2.1.3. ICA (Independent Component Analysis)

Desarrollado por Marian Stewart Bartlett, Javier R. Movellan y Terrence J. Sejnowski (2002) [12] es la generalización del PCA que separa pixeles y variables aleatorias en dos arquitecturas. Emplean un algoritmo desarrollado por Bell y Sejnowski [13] desde el punto de vista de la transferencia de información optima en redes neuronales con funciones de transferencia sigmoidal.

Es una generalización del método PCA. Este método trata de descomponer una señal en una combinación lineal de fuentes independientes. ICA minimiza los órdenes de dependencia. Se tiene una matriz de variables independientes y una matriz de observaciones. En esta, cada columna es el resultado de un experimento aleatorio, y en cada fila se tiene el valor de una prueba de ese experimento.

6.2.1.4. SVM (Support Machine Vector)

Son un tipo de clasificadores de patrones basados en técnicas estadísticas de aprendizaje propuestas por Vapnik [14] y sus colaboradores en 1992. Permitieron construir fronteras de decisión flexibles y tienen una buena capacidad de generalización.

Es un método genérico para resolver problemas de reconocimiento de patrones. Dado un conjunto de puntos de un determinado espacio que pertenecen a dos clases distintas, SVM encuentra el hiperplano que separa la mayor cantidad de puntos de la misma clase del mismo lado. Esto se realiza maximizando la distancia de cada clase al hiperplano de decisión, denominado OSH (Optimum Separating Hyperplane). Los puntos más cercanos al hiperplano, de cada conjunto evaluado, son los llamados vectores de soporte (support vectors). Es un

método que debe ser usado en combinación con otro, ya que este método es únicamente discriminatorio, pero 23 incapaz de obtener de las imágenes sus propias características, por lo que, por ejemplo, se puede usar con el método Eigenfaces (PCA) para que este extraiga características propias de las imágenes y a continuación se usará SVM para la diferenciación entre las clases, utilizando como valores característicos los obtenidos por el método PCA.

6.2.1.5. KPCA (Kernel Principal Component Analysis)

Desarrollado por Zhang Yankun y Liu Chongqing [15] en 2002 propone un enfoque de reconocimiento facial basado en el uso de un kernel en PCA. También implementa SVM.

El método kernel PCA utilizado para el reconocimiento facial tiene mejores resultados que el PCA convencional partiendo de las mismas condiciones.

La obtención de modelos estocásticos para la selección del kernel mejora los porcentajes de acierto, que se logran al obtener la matriz kernel basada en modelos estadísticos y optimización

6.2.2. Métodos basados en modelos

6.2.2.1. EBGM (Elastic Bunch Graph Matching)

Metodo propuesto por L.Wiskott, J.M. Fellous, N. Krüger y C. Von der Malsburg [16] en 1997 en el que utilizaban una base de datos de una sola imagen por individuo para reconocer al sujeto. Se basaba en vectores y nodos.

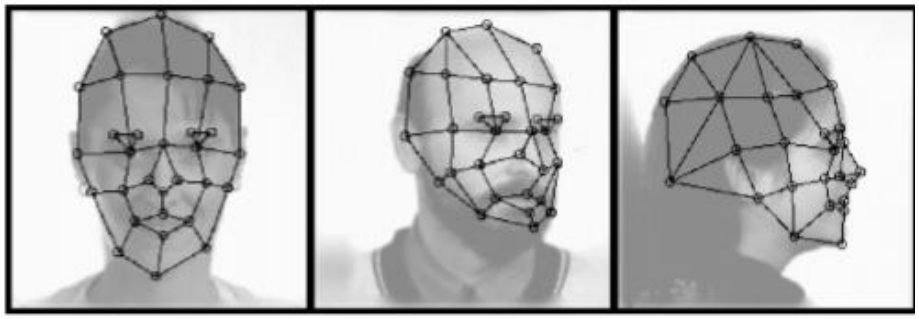

Figura 9: Celdas adaptadas a objetos para diferentes poses (fuente: 8)

La representación de la cara toma la forma de grafos etiquetados. Los grafos están formados por vectores y nodos, los vectores se etiquetan con información geométrica y los nodos se etiquetan con un conjunto de características locales. Estas características locales se basan en transformaciones de Gabor, lo cual se podría tomar como un procedimiento de preprocesamiento de imágenes basado en fenómenos biológicos. Para un sistema de reconocimiento de caras que use este método, las imágenes de las caras deberán ser normalizadas para obtener una imagen media nula, y una desviación estándar igual a uno. Además se suavizan los bordes de la imagen.

6.2.2.2. *LBP (Linear Binary Patterns)*

Desarrollado por T. Ahonen, A.Hadid y M.Pietikainen [17][18] en 2004 y 2006 en el que se basan tanto en la información de forma como en la de textura para el reconocimiento facial.

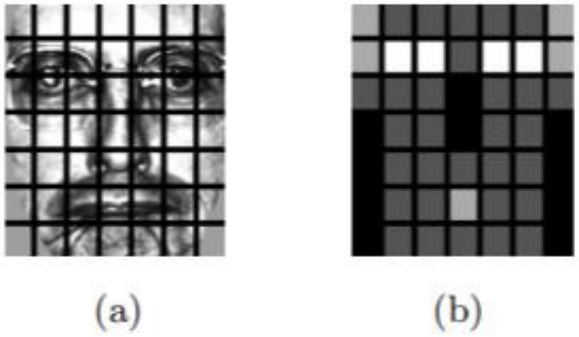

Figura 10: (a) Ejemplo de imagen facial dividido en ventanas. (b) Los pesos de cada pixel negro (0) blanco (4) (fuente: articulo referencia [18])

Es más un descriptor de textura. Este algoritmo recorre la imagen etiquetando los píxeles mediante un umbral de la diferencia entre el píxel central y sus vecinos, considerando el resultado como un número binario (1 o 0).

La concatenación de las etiquetas de los vecinos puede usarse como descriptor. LBP utiliza varios descriptores locales que se combinarán en un descriptor global, es importante mantener información sobre la relación espacial.

6.2.2.3. *HMM (Hidden Markov Models)*

Estos modelos propuestos por F. Samaria y S. Young [19] en 1994 utilizan regiones horizontales de pixeles que albergan a la frente, ojos, nariz, boca y barbilla sin obtener la posición exacta de cada rasgo.

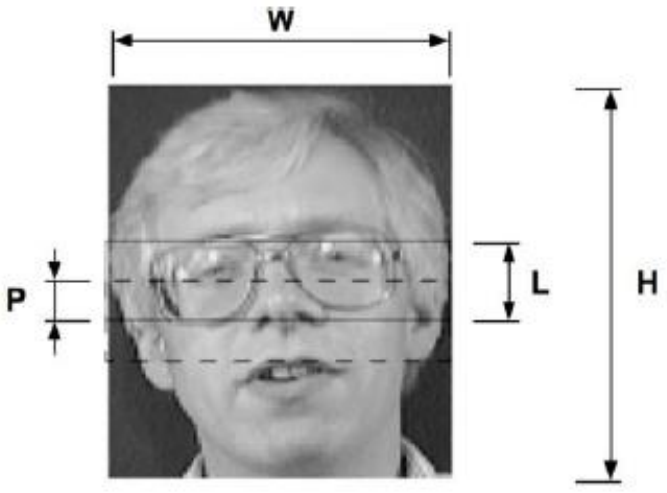

Figura 11: Extracción de bloques mediante HMM (fuente: 9)

Son un conjunto de modelos estadísticos utilizados para caracterizar las propiedades estadísticas de una señal. En este modelo se asume que el sistema a modelar es un proceso de Markov de parámetros desconocidos, el objetivo, pues, será 24 determinar los parámetros desconocidos (u ocultos, de ahí el nombre) de la cadena a partir de los parámetros que se conocen.

A continuación, observamos una comparativa entre distintos métodos de reconocimiento facial.

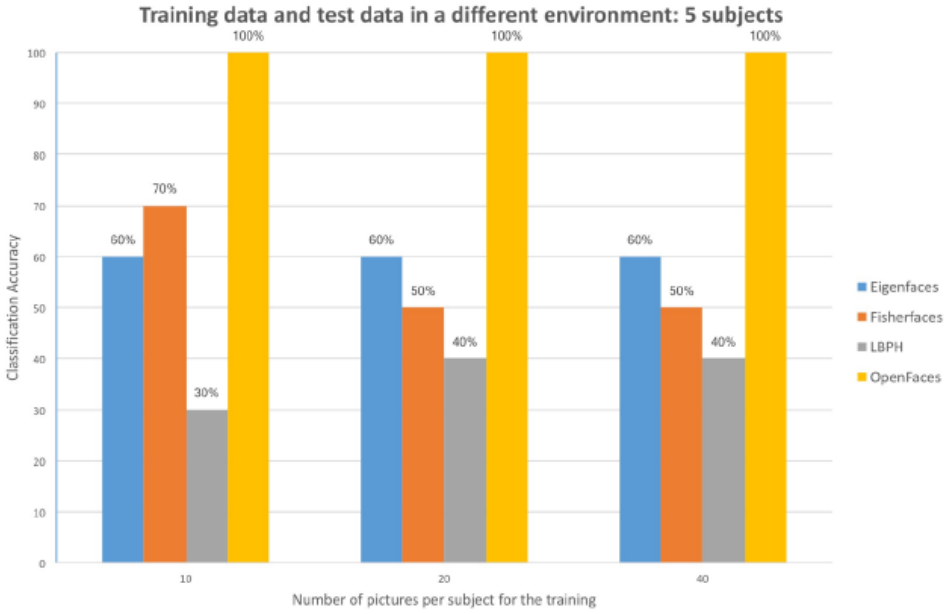

Figura 12: Comparativa distintas técnicas de reconocimiento facial (fuente: 10)

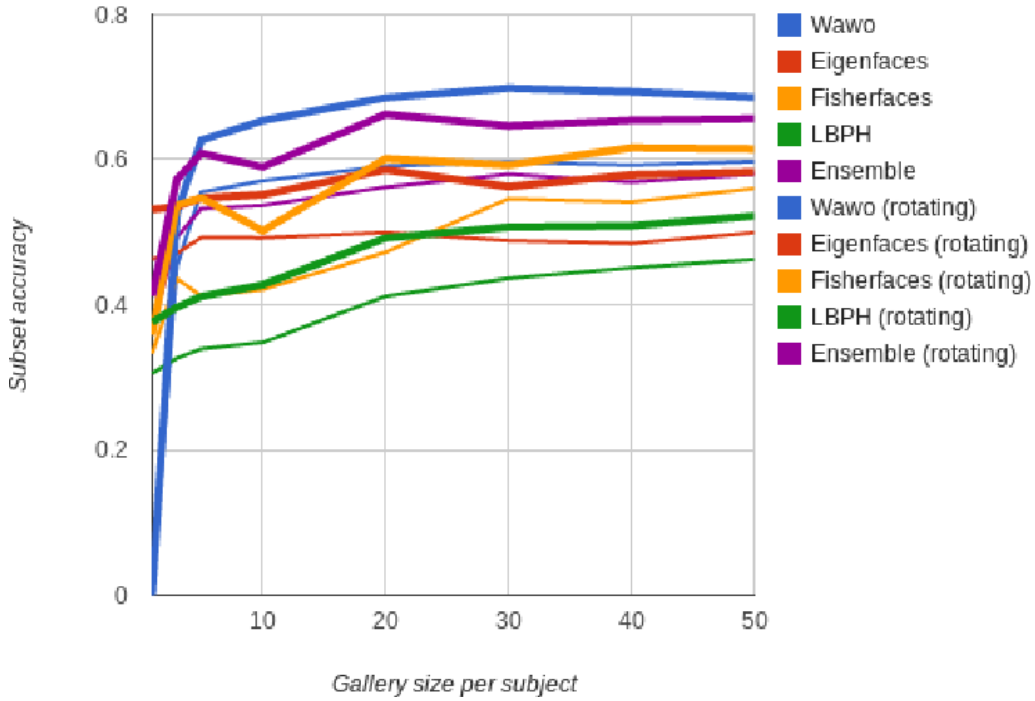

Figura 13: Comparativa distintas técnicas de reconocimiento facial (fuente: 11)

6.3. Clasificación de la información

Para realizar la comparación de imágenes, éstas han de estar en modo de vector y serán comparados a partir de las distancias entre vectores.

6.3.1. Distancias vectoriales

En la fase final de la mayoría de estos procedimientos se emplea la distancia entre vectores para encontrar la imagen más parecida a la que se está buscando.

6.3.1.1. Distancia euclídea

En matemáticas, la distancia euclídea es la distancia que se mediría con una regla entre dos puntos de un espacio euclídeo, la cual se deduce a partir del teorema de Pitágoras.

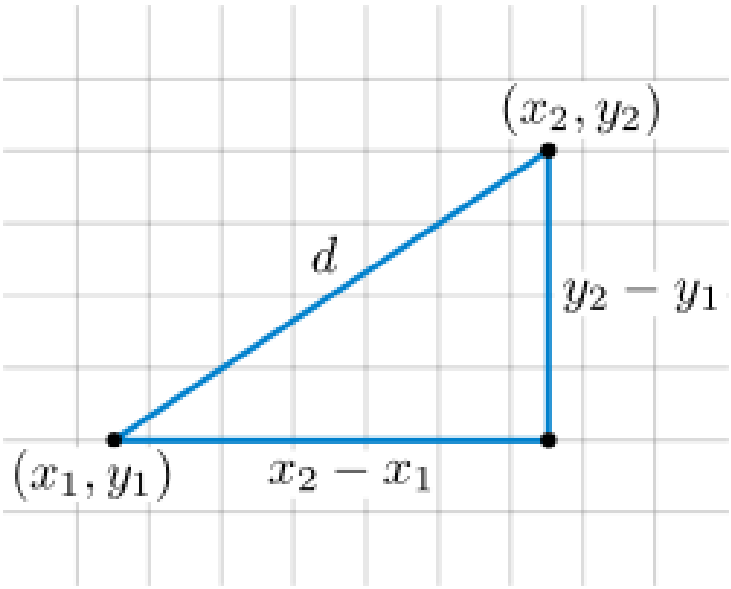

Figura 14: Distancia en un sistema de coordenadas cartesianas (fuente: 12)

$$d_E(P_1, P_2) = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}$$

Esta definición depende de la existencia de coordenadas cartesianas sobre la variedad diferenciable, aunque en un espacio euclídeo pueden definirse sistemas de coordenadas más generales, siempre se posible definir un conjunto global de coordenadas cartesianas.

6.3.1.2. Distancia de K. Pearson

Es una modificación de la distancia euclídea:

$$\delta_K^2(i,j) = \sum_{k=1}^{p} \frac{(x_{ik} - x_{jk})^2}{s_k^2} = (\mathbf{x}_i - \mathbf{x}_j)' \, \mathbf{S}_0^{-1} \, (\mathbf{x}_i - \mathbf{x}_j)$$

S_0 es la matriz diagonal que contiene varianzas de $X_1 \ldots\ldots X_p$.

Esta expresión equivale a reescalar cada variable en unidades de desviación típica. El peso que se atribuye a la diferencia entre individuos es mayor cuanto menor es la dispersión en esa variable. Pero sigue suponiendo que las variables están incorreladas.

6.3.1.3. Distancia de Canberra

Esta distancia es una medida numérica de la distancia entre pares de puntos en un espacio vectorial. Introducido en 1966 [20] y refinado en 1967 [21] por G. Lance y W. T. Williams. La

distancia de Canberra se ha utilizado como una métrica para comparar listas clasificadas y para detectar intrusiones en seguridad informática.

$$d(\mathbf{p}, \mathbf{q}) = \sum_{i=1}^{n} \frac{|p_i - q_i|}{|p_i| + |q_i|}$$

7. Análisis de alternativas

Todo algoritmo de reconocimiento facial tienes dos fases bien diferenciadas, la fase de entrenamiento y la fase de reconocimiento o test.

Durante la fase de entrenamiento se prepara la base datos que será utilizada en el reconocimiento. Para ello, se introducen una o varias imágenes del rostro de los distintos sujetos en el algoritmo de entrenamiento. Este algoritmo es el encargado de extraer las características de cada persona y almacenarlas para compararlo posteriormente.

En la fase de test se toman imágenes de un sujeto desconocido, se extraen las características por el mismo proceso que en la fase de entrenamiento, y se comparan con las de la base de datos. El proceso consta de las partes que aparecen en la figura 13.

Figura 15: Proceso de Reconocimiento Facial

7.1. Detección de rostros en imágenes

Existen varios métodos de detección de caras, de manera que cuando la cara sea detectada se puedan aplicar los procedimientos que se explicarán posteriormente.

La detección del rostro es la primera fase importante de todo programa de reconocimiento facial. Es posible que previamente haya sufrido un proceso de acondicionamiento o preparación de la imagen para mejorar el rendimiento de las fases posteriores. Previamente a la fase de reconocimiento de caras, donde ya se le asignará una identidad a las caras obtenidas en la fase de detección, se deberá realizar una localización de las caras en la imagen, pero, si la fase de localización facial no funciona correctamente, no será posible el posterior reconocimiento facial.

Uno de los algoritmos más utilizados para detectar objetos en tiempo real es el detector rápido de objetos de Viola-Jones

7.1.1. Algoritmo de Viola-Jones vs Kanade-Lucas-Tomasi

El algoritmo de Viola-Jones tiene varias ventajas sobre el de Kanade-Lucas-Tomasi como la selección de características que es muy sofisticada y un detector invariante que localiza escalas. Podemos escalar solamente las características, de esta forma ahorra mucho trabajo y errores al no tener que escalar la imagen completa.

Por otro lado, el algoritmo Kanade-Lucas-Tomasi es más rápido que la mayoría de métodos tradicionales, como es el caso del algoritmo de Viola-Jones, pero las características faciales han de ser más claras.

Tabla 1: Resultados de la detección facial con ambos algoritmos [22]

	Viola-Jones	Kanade-Lucas-Tomasi
Imagen frontal	97%	90%
Perfil izquierdo	90%	85%
Perfil derecho	88%	83%
Mirando arriba	80%	80%
Mirando abajo	80%	80%
Total	87%	84%

Basándonos en los porcentajes de acierto de cada método y en la comodidad para la implementación de ellos el algoritmo de Viola y Jones será el utilizado en el proyecto.

7.2. Extracción de características

Se han desarrollado varios métodos de reconocimiento facial durante las últimas décadas. Podemos clasificar estos métodos en dos grupos que incluyen métodos basados en la apariencia y basados en modelos. Los primeros, también son denominados métodos holísticos debido a que utilizan textura holística que se aplica a regiones de toda la cara. Mientras que el segundo método utilizan la forma y la textura de la cara.

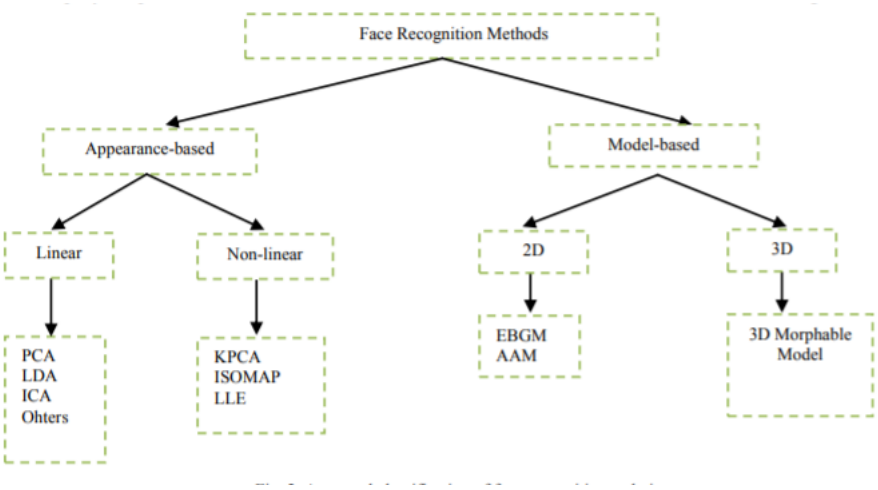

Figura 16: Esquema de los métodos de reconocimiento facial (fuente: 13)

7.2.1. Métodos holísticos

Utilizan la imagen de la cara como entrada al sistema de reconocimiento, utilizan además unos conocimientos sobre álgebra que se dan por sabidos. Son métodos basados en la correlación, se utilizan modelos de comparación para el reconocimiento.

El problema suele ser que cada píxel es una característica y este sistema tiene que compararlas todas, por lo que se suele trabajar con otros métodos que correlacionan las características entre sí para reducir el espacio facial a un número que permita aplicar el algoritmo en tiempo real.

7.2.1.1. Principal Component Analysis: PCA

Es la base de todos los sistemas de reconocimiento facial. Consigue reducir considerablemente las dimensiones de las imágenes a la vez que el porcentaje de reconocimiento no disminuye considerablemente. Aporta velocidad y fiabilidad al proceso.

7.2.1.2. Independent Component Analysis: ICA

Se utiliza para bases de datos donde las características están lo más descorreladas posibles. Es muy utilizado cuando el objetivo del proyecto es reconocer el estado de ánimo del individuo o la expresión facial que presenta. Es un método con alta probabilidad de reconocimiento a la vez necesita alta capacidad computacional y mayor tiempo de ejecución ya que la dimensionalidad de las imágenes no puede ser reducida en exceso.

7.2.1.3. Linear Discriminant Analysis: LDA

Derivado del método de análisis de componentes principales. Necesita menos tiempo para realizar el proceso pero también proporciona peores resultados, ya que intenta llevar el espacio de caras a un subespacio de baja dimensionalidad que aumente la separabilidad de las clases presentes. A la hora de elaborar la computación es un método más complejo. Necesita menos imágenes de prueba para ser capaz de reconocer a un individuo.

7.2.1.4. Support Vector Machine: SVM

Método complejo no lineal que aporta grandes porcentajes de detección de individuos. En un estudio llevado a cabo por P. Jonathon Philips [23], comparaba las técnicas de PCA con la de SVM y llego a la conclusión de que los porcentajes de reconocimiento con SVM eran mayores que los obtenidos por el método del PCA.

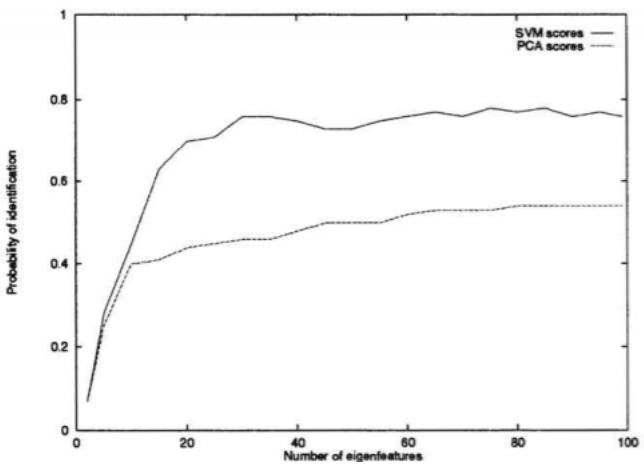

Figura 17: Porcentajes de acierto en reconocimiento facial comparando SVM Y PCA (fuente: articulo referencia [23])

7.2.1.5. KPCA (PCA con kernel)

Se ayuda de un kernel para eliminar la linealidad del proceso de PCA.

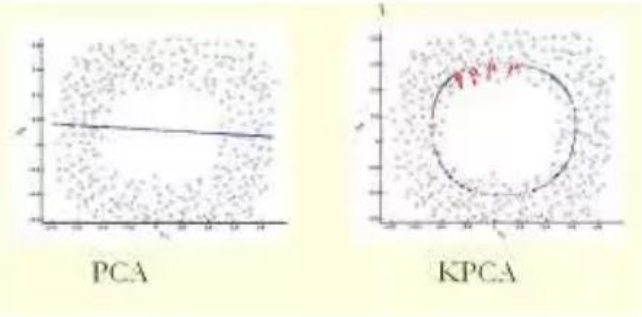

Figura 18: diferencia entre PCA y KPCA (fuente: 14)

El KPCA simplemente lleva a cabo el PCA en un nuevo espacio. PCA es capaz de reconstruir la imagen una vez tratada mientras que KPCA no realiza dicha función. Además, la complejidad computacional del KPCA para extraer componentes principales toma más tiempo en comparación con el PCA estándar.

7.2.2. Métodos basados en modelos

Se basan en la extracción de características locales, tales como ojos, nariz, etc. Las posiciones de estas características y las estadísticas locales con respecto a ellas constituyen la entrada al sistema de reconocimiento. Existen dos divisiones, la basada en los vectores característicos extraídos del perfil, y la basada en los extraídos a partir de una vista frontal, esta última utilizada mucho anteriormente pero con resultados nada correctos en la mayoría de los casos.

7.2.2.1. Elastic Bunch Graph Matching: EBGM

En comparación con los métodos lineales tradicionales EBGM funciona mejor, pero el tiempo de cálculo es relativamente largo. Además, si aumentamos el número de hitos en EBGM, el rendimiento se reduce, por lo que la elección del número de hitos y la cantidad de imágenes a utilizar en la base de datos ha de ser lo suficientemente pequeña como para realizar el proceso en tiempo real.

7.2.2.2. Local Binary Pattern: LBP

Dentro de los métodos basados en modelos es de los más básicos, aunque puede proporcionar buenos resultados. De todos modos este método sería más indicado para el proceso de autenticación de individuos y no para el de identificación ya que el número de variables que han de ser comparadas con distintas bases de datos sería demasiado grande y la lentitud del proceso sería muy considerable. Haría falta una base de datos para cada característica adquirida de la imagen y una matriz de vectores que comparar por cada base de datos. Por lo que si queremos obtener altos porcentajes de acierto el tiempo de reconocimiento sería demasiado extenso.

7.2.2.3. Hidden Markov Models: HMM

Este método es uno de los métodos menos utilizados debido a que su estudio no está demasiado elaborado. La información acerca del método es muy escasa por lo que llevarlo a cabo habría resultado muy complejo.

Los tres factores más influyentes en la elección de método a emplear en la extracción de características son la tasa de error, el tiempo y la complejidad de la computación. En base a

estos tres factores, el método utilizado será el de Análisis de Componentes Principales (PCA) ya que es capaz de dar unos resultados fiables en un tiempo real sin ser un programa que requiera una complejidad de computación demasiado alta.

7.3. Clasificación

7.3.1. Distancia euclídea

Distancia básica utilizada en numerosos métodos de reconocimiento facial. Es la distancia más sencilla de calcular, pero a la vez muy efectiva si el fin del cálculo es simplemente hallar la distancia natural entre dos vectores.

7.3.2. Distancia de K. Pearsons

La distancia de Pearson es la relación lineal entre dos variables aleatorias cuantitativas.

Como ventajas, tiene que el valor del coeficiente de correlación es independiente de cualquier unidad usada para medir variables y que mientras más grande sea la muestra más exacta será la estimación.

Por otro lado, requiere conocimiento acerca de la naturaleza de los datos analizados y que ambas variables hayan ido medidas hasta un nivel cuantitativo continuo.

7.3.3. Distancia de Canberra

Centrada en encontrar errores en listas de datos clasificados, tiene en cuenta infinidad de datos que no son de interés para nuestro proyecto. Además, su computación es mucho más complicada.

DISTANCIA	AUTOR	METRICA	EUCLIDEA
$\left(\sum_{k=1}^{n} (x_{ik} - x_{jk})^2\right)^{1/2}$	*Euclides*	SI	SI
$\left(\sum_{k=1}^{n} (x_{ik} - x_{jk})^q\right)^{1/q}$	*Minkowski*	SI	NO
$\left(\sum_{k=1}^{n} \left(\frac{x_{ik} - x_{jk}}{s_k}\right)^2\right)^{1/2}$	*K. Pearson*	SI	SI
$\sum_{k=1}^{n} \frlac{\lvert x_{ik} - x_{jk} \rvert}{\lvert x_{ik} \rvert + \lvert x_{jk} \rvert}$	*Canberra*	SI	NO

Figura 19: Características de varias distancias vectoriales (fuente: 15)

Debido a que el único dato que deseamos saber al calcular la distancia entre dos vectores es el de la distancia real entre ellos, se utilizará la distancia euclídea ya que el resto son variantes más complejas que ella porque dan datos irrelevantes para nuestro proyecto.

8. Descripción de la solución

En el proyecto se implementa una aplicación que se encarga del reconocimiento facial en lenguaje C++. Para realizar este proyecto nos hemos apoyado en las librerías de libre utilización que se emplean en el procesado de imágenes.

A continuación, se detallarán las fases que llevará a cabo el presente proyecto. En primer lugar la adquisición de las imágenes de entrenamiento. Después, el algoritmo a utilizar para la detección de rostros, seguido por la técnica que se empleará para el reconocimiento. Por último, se decidirá la distancia que se utilizará para comparar distintos vectores.

Para las pruebas a realizar sobre el sistema se ha decidido tomar de distintas bases de datos explicadas a continuación.

8.1. Bases de datos

8.1.1. FEI

Creada por el Laboratorio de Inteligencia Artificial de FEI en Sao Bernardo do Campo entre junio de 2005 y marzo de 2006. Dicha base de datos consta de 14 fotos de 200 sujetos (100 hombres y 100 mujeres) distintos entre 19 y 40 años que forman un total de 2800 imágenes. Todas las imágenes son a color sobre un fondo blanco homogéneo en una posición frontal vertical, con una rotación de aproximadamente 180 grados. El tamaño de las imágenes es de 640x480 pixeles. Todos los rostros están representados principalmente por alumnos y personal del FEI.

Figura 20: 12 imágenes del primer sujeto de la base de datos FEI (fuente: 16)

8.1.2. Yale face

La base de datos Yale face contiene 165 imágenes, ya en escala de grises en formato GIF de 15 personas. Hay 11 imágenes por sujeto, una expresión facial o configuración diferente: luz central, con gafas, feliz, luz izquierda, sin gafas, normal, luz derecha, triste, soñoliento, sorprendido y guiño.

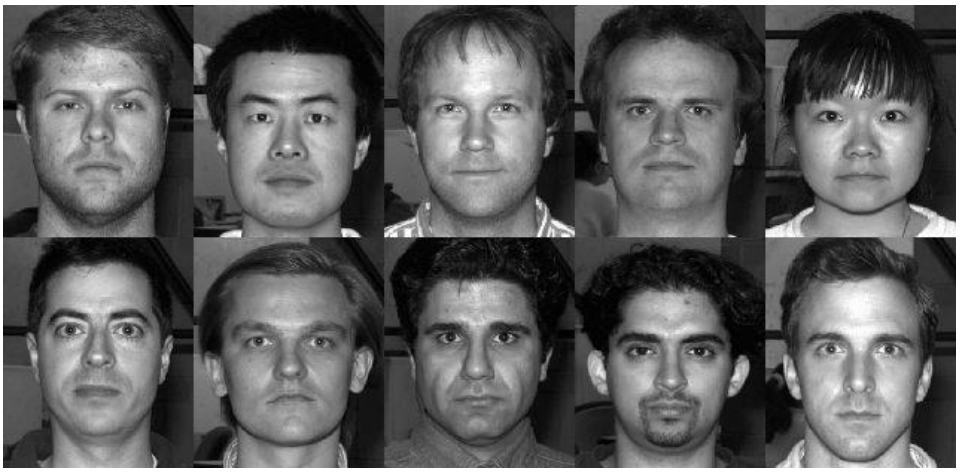

Figura 21: Distintos rostros de Yale face database (fuente: 17)

8.1.3. ORL

Contiene un conjunto de imágenes faciales tomadas entre abril de 1992 y abril de 1994. La base de datos se utilizó en el contexto de un proyecto de reconocimiento facial llevado a cabo por AT&T laboratorios en colaboración con "Speech, Vison and Robotics Group" del departamento de ingeniería de la universidad de Cambridge.

Hay diez imágenes diferentes para cada uno de los 40 sujetos. Para algunos sujetos, las imágenes se tomaron en diferentes momentos variando la iluminación, las expresiones faciales (ojos abiertos, cerrados, sonriendo) y los detalles faciales (gafas, sin gafas). Todas las imágenes se tomaron sobre un fondo oscuro y homogéneo con los sujetos en posición vertical y frontal (con tolerancia para algunos movimientos laterales).

Los archivos están en formato pgm y se pueden ver cómodamente en sistemas UNIX. El tamaño de cada imagen es 92x112 pixeles con 256 niveles de gris por pixel.

Figura 22: Diferentes imágenes de la base de datos ORL (fuente: 18)

8.1.4. FERET

El programa antidrogas del DOD patrocinó el programa de Tecnología de reconocimiento facial y el desarrollo de la base de datos FERET. El Instituto Nacional de Estandares y Tecnología (NIST) se desempeña como agente técnico para la distribución de la base de datos FERET.

La base de datos contiene imágenes obtenidas entre 1993 y 1996. La base de datos contien 1564 conjuntos de imágenes para un total de 14126 imágenes que incluyen a 1199 individuos y 365 conjuntos duplicados de imágenes. Los conjuntos duplicados son conjuntos de imágenes de un mismo individuo pero tomadas en distintos periodos de tiempo.

Figura 23: Imágenes correspondientes a la base de datos FERET (fuente: 19)

A continuación, se realiza un pequeño resumen de todas ellas.

Tabla 2: Características de las distintas bases de datos utilizadas

Base de datos	FERET	ORL	YALE	FEI
Año	1993	1994	1997	2005
Número de personas	1.199	40	16	200
Imágenes	14.051	400	160	2800
Poses	9-20	3	1	14
Iluminación	2	3	3	2
Expresiones	2	2	6	2
Sesiones	2	2	1	1

8.2. Algoritmo Viola-Jones

Tras el estudio de diferentes técnicas de detección de rostros y el análisis de las diferentes alternativas, se ha decidido utilizar el algoritmo de Viola-Jones utilizando una cascada de clasificadores Haar por dos sencillas razones.

- El algoritmo de Viola-Jones está incluido en las funciones descargadas de las librerías OpenCV, lo que facilita la utilización de dicho algoritmo.
- Dentro de los métodos estudiados el algoritmo de Viola-Jones es el que tiene menor porcentaje de error en las detecciones.

8.3. PCA

La técnica de reconocimiento que utilizaremos será la denominada Analisis de Componentes principales (PCA). Aunque no es el método de reconocimiento con mayor porcentaje de acierto, es la técnica base de todos los procesos lineales basados en la apariencia del individuo, ya que los métodos no lineales tenían demasiadas variables complejas que analizar hemos decidido utilizar el método PCA en el que profundizaremos a continuación.

8.3.1. Concepto

El análisis de componentes principales es una técnica estadística de síntesis de la información, o reducción de la dimensión. Ante un conjunto de datos con multitud de variables, PCA las reduce a un número menor pero intentando perder la menor información posible. Es decir, desecha aquellos datos donde la información es más insignificante.

Cuando tenemos una infinidad de datos en los que ver las diferencias principales entre ellos, PCA es una herramienta muy útil ya que permite analizarlos sin la necesidad de tener que tratar con las enormes dimensionalidades de las variables.

EL PCA recoge el conjunto de datos original y crea un subespacio con unos nuevos ejes de coordenadas, donde la varianza de mayor tamaño es el primer eje, también llamado primer componente principal, la segunda varianza más grande es el segunde eje y así sucesivamente.

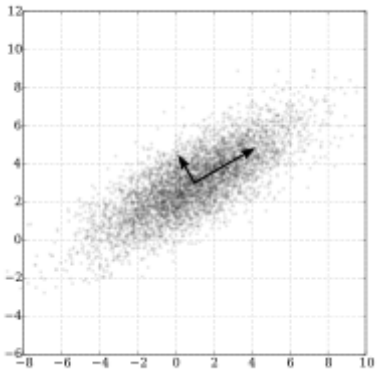

Figura 24: Nuevos ejes creados por PCA (fuente: 20)

La otra ventaja principal del PCA es que una vez reconstruidos dichos ejes, permite la compresión de los datos reduciendo el número de dimensiones, perdiendo poca información.

La forma de emplear el PCA en un conjunto de datos de dos dimensiones (x, y) es la siguiente:

1. Para obtener un correcto funcionamiento del PCA, a cada valor de los datos hay que restarle la media de cada dimensión. Es decir a cada dato x se le resta $\bar{x}$ y a cada dato y se le resta $\bar{y}$. El resultado final de esta operación es un conjunto de datos que tienen media 0.
2. A continuación, se calcula la matriz de covarianza de dichos datos.

$$cov(X,Y) = \frac{\sum_{i=1}^{n}(X_i - \bar{X})(Y_i - \bar{Y})}{(n-1)}$$

$$C = \begin{pmatrix} cov(x,x) & cov(x,y) \\ con(y,x) & cov(y,y) \end{pmatrix}$$

3. Una vez calculada la matriz de covarianza, se calculan los vectores propios (eigenvectors) y los valores propios (eigenvalues) de dicha matriz. Los vectores propios son los vectores no nulos que si son trnasformados, tienen como resultado un múltiplo escalar de sí mismos, por lo que su dirección sigue siendo la misma. El escalar por el que son multiplicados es el valor propio (eigenvalue). Cada valor propio tiene su propio vector propio.

$$\begin{pmatrix} 1 & 0 \\ 1 & -2 \end{pmatrix} \times \begin{pmatrix} 3 \\ 1 \end{pmatrix} = \begin{pmatrix} 3 \\ 1 \end{pmatrix} = 1 \times \begin{pmatrix} 3 \\ 1 \end{pmatrix}$$

Donde 1 sería el autovalor y $\begin{pmatrix} 3 \\ 1 \end{pmatrix}$ el autovector.

4. Para saber cuál es el primer componente principal del conjunto de datos debemos encontrar el eigenvector con el eigenvalue asociado más alto. Si se ordenan los eigenvectors de mayor a menor en función de sus eigenvalues obtendremos los componentes principales ordenados de mayor a menor importancia. Es aquí donde PCA tiene la capacidad de reducir la dimensionalidad desechando los vectores propios con menor importancia (los eigenvectors con valores de eigenvalue mas pequeños) siendo la información perdida muy pequeña. Una vez realizado el proceso de eliminación se forma una nueva matriz con los eigenvectors deseados en columnas. Esta matriz se denomina matriz de Características.
5. Para terminar, el nuevo conjunto de datos es creado multiplicando la traspuesta del vector de características por la traspuesta de la matriz de datos original, con el valor medio sustraído como observamos a continuación:

$$Datos\ Finales = CararcteristicasT \times dataT$$

6. Si se desea reconstruir los datos originales, se ha de multiplicar la inversa del vector de características traspuesto por los datos finales y sumarle el valor medio original. AL haber quitado los vectores propios menos importantes la información reconstruida habrá perdido datos poco relevantes por el camino.

$$Datos\ Originales = \big((CaracteristicasT) - 1 \times Datos\ Finales\big) + Valor\ Medio$$

8.3.2. PCA en reconocimiento facial

Una imagen facial de dimensiones MxN puede ser considerada un vector de dimensión NxM en el que las filas de la imagen original van colocadas una detrás de otra en el nuevo vector para conseguir que la imagen sea una sola fila.

$$X = (x_1\ x_2\ x_3\ \ldots . x_{n2})$$

Para aplicar el PCA a las imágenes faciales de la fase de entrenamiento, primero se transforman las imágenes en vectores como se indica en el apartado anterior. Estos valores de los vectores han de ser valores en escala de grises, por lo que las imágenes han de ser previamente transformadas a escala de grises si fuera necesario. A continuación, se creará una matriz con tantas filas como imágenes se deseen tratar en la que cada fila sea la imagen facialque se desea introducir en la base de datos. A esta matriz es a la que se le aplicará el PCA, consiguiendo el

nuevo subespacio y sus nuevos ejes adquiridos con el conjunto total de todos los eigenvectors que deseamos utilizar.

$$Matriz\ imagenes = \begin{pmatrix} vector\ imagen\ 1 \\ vector\ imagen\ 2 \\ . \\ . \\ . \\ vector\ imagen\ M \end{pmatrix}$$

Al realizar el PCA sobre esta matriz obtendremos tres variables:

1. La media de las imágenes que se corresponde a un vector de dimension MxN que podrá ser transformado en una imagen se modifican sus dimensiones.
2. Un vector con los valores propios ordenados de mayor a menor en un vector
3. Una matriz con los vectores propios ordenados en concordancia con el orden de los valores propios

Esta última matriz será la correspondiente a los nuevos ejes de coordenadas. A continuación se deberán proyectar todas las imágenes que deseamos tener en la base de datos en estos nuevos ejes y ser almacenados en una nueva matriz.

8.4. Comparación de vectores

Una vez tenemos almacenados en una matriz todas las proyecciones de las imágenes se debe pasar a la fase de reconocimiento final. Para ello, se ha de proyectar en los mismos ejes que el resto de imágenes, la imagen el individuo que se desea reconocer. Una vez tengamos dicho vector proyectado debemos compararlo con el resto de vectores resultantes de las imágenes proyectadas.

Para este apartado, tras realizar un estudio de los diferentes métodos de comparación de vectores hemos decidido utilizar la distancia euclídea de Euclides ya que es una comparación simple en la que únicamente queremos averiguar la distancia real entre dos vectores. Esta distancia se calcula con la distancia valor por valor entre cada imagen de entrenamiento y la imagen a reconocer. Tras realizar el método con todas las imágenes de entrenamiento, el valor más pequeño, debería ser el correspondiente a la imagen más parecida a la que se desea reconocer.

$$dist = \sum_{i=1}^{N2} \sqrt{(Valor\ imagen\ reconocimiento_i - Valor\ imagen\ entrenamiento_i)^2}$$

Para evitar falsos positivos, hay que definir un umbral a partir del cual la distancia euclídea sea demasiado grande como para considerar que la persona ha sido reconocida.

8.5 OpenCV

OpenCV es una librería de visión por computador de código abierto, disponible en http://sourceforge.net/projects/opencvlibrary/.

La librería está escrita en los lenguajes C y C++ y es compatible con Linux, Windows y Mac OS X. Cuenta con un desarrollo activo en interfaces para Python, Ruby, Matlab y otros lenguajes.

OpenCV ha sido diseñado para ser eficiente en cuanto a gasto de recursos computacionales y con un enfoque hacia las aplicaciones de tiempo real.

La librería OpenCV contiene aproximadamente 500 funciones que abarcan muchas áreas de CV, incluyendo inspección de productos de fábricas, escaneo médico, seguridad, interfaces de usuario, calibración de cámaras, robótica…etc, porque la visión por computador y el aprendizaje automático van de la mano.

OpenCV también tiene una completa librería de uso general de aprendizaje automático (MLL o Machine Learning Library), la cual es muy útil para cualquier problema de aprendizaje automático. Esta sublibrería está especializada en el reconocimiento estadístico de patrones y clustering.

OpenCV tiene una estructura modular. Sus módulos principales son los siguientes:

- core: Este es el módulo básico de OpenCV. Incluye las estructuras de datos básicas y las funciones básicas de procesamiento de imágenes. Este módulo también es usado por otros módulos como highgui.
- highhui: Este módulo provee interfaz de usuario, códecs de imagen y vídeo y capacidad para capturar imágenes y vídeo, además de otras capacidades como la de capturar eventos del ratón…etc.
- imgproc: Este módulo incluye algoritmos básicos de procesado de imágenes, incluyendo filtrado de imágenes, transformado de imágenes…etc.
- video: Este módulo de análisis de vídeo incluye algoritmos de seguimiento de objetos, entre otros.
- objdetect: Incluye algoritmos de detección y reconocimiento de objetos para objetos estándar.

Estas librerías han sido muy útiles a la hora de elaborar el proyecto debido a las funciones que proporciona a la hora de procesar imágenes. Las funciones que han sido utilizadas serán detalladas en el apartado Proyecto.

9. Metodología

Una vez decidido el enfoque dado a la aplicación, este apartado analizará las funciones que deben ser realizadas por la aplicación y como han sido implementadas.

9.1 Lectura y procesamiento de la imagen facial

La aplicación leerá una por una el número de imágenes indicado por el programador de la base de datos. A continuación pasará la imagen a escala de grises si procede, y le aplicará el clasificador con el fin de detectar el rostro en la imagen. El número de pixeles será establecido por el programador para que todas las imágenes de rostros tengan el mismo tamaño. Finalmente, se ecualizará el histograma de cada imagen para intentar reducir el impacto de la iluminación a la hora de capturar las imágenes.

Para este apartado, será vital el apoyo proporcionado por las librerías OpenCV ya que contiene las funciones necesarias para realizar operaciones fundamentales

El diagrama utilizado para la lectura de la imagen de procesamiento, así como para el reconocimiento será el siguiente:

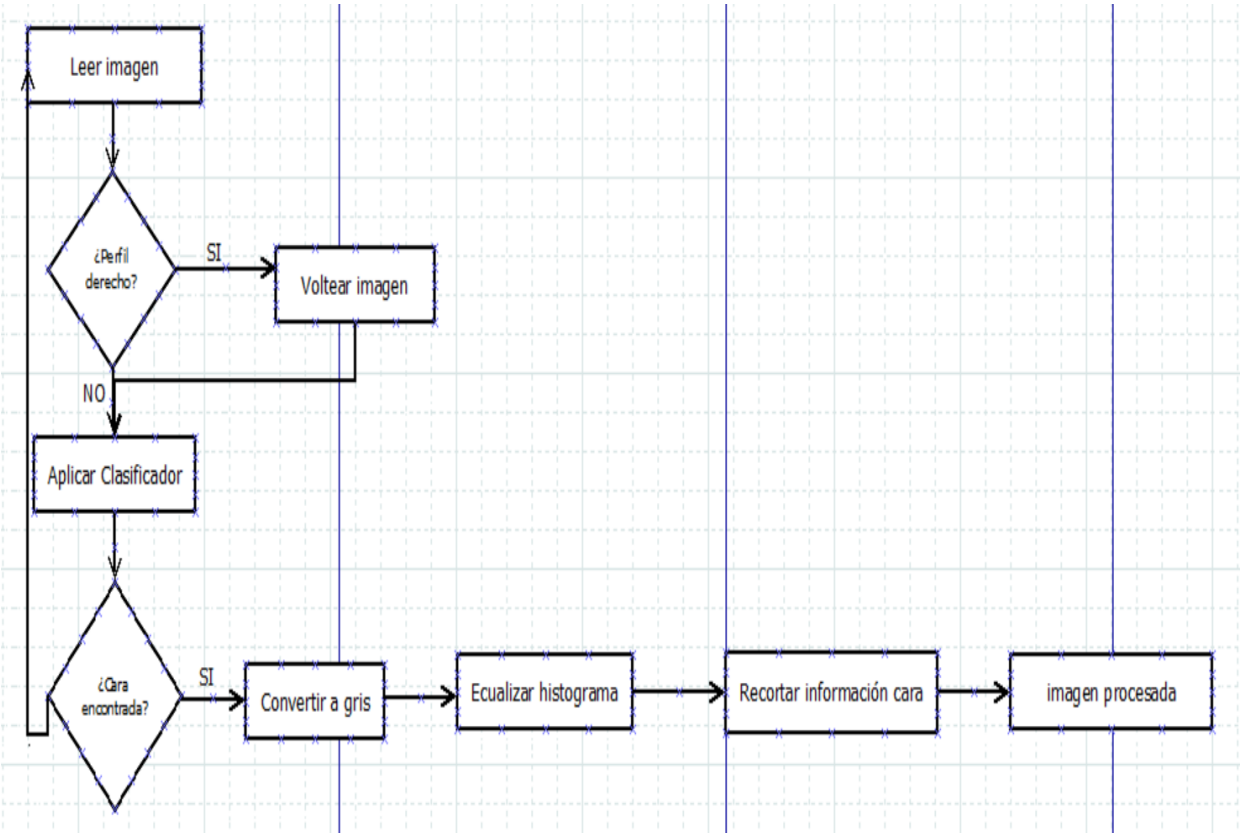

Figura 25: Diagrama de Flujo detección facial.

Para definir el tamaño de la imagen, se ha consultado el ISO/IEC 19794 parte 5[24] en el que se detallan las normas a seguir en las imágenes faciales.

En este apartado lo que se ha de tener en cuenta es el ancho y el alto de la imagen facial, por lo que se observa la proporción del alto con respecto al ancho de una imagen facial, que es el siguiente:

$$Alto = 0{,}75 \times Ancho$$

Por ejemplo, para la base de datos FEI se ha decidido tomar un ancho de 240cm y un alto de 320cm. De esta forma, se obtiene suficiente información para poder realizar el reconocimiento adecuadamente sin tener que procesar las imágenes completas que tienen mayor número de valores. Además, se evitan errores en el proceso de reconocimiento ya que se anularían factores como la iluminación en áreas ajenas a la de la cara que podrían dificultar dicho proceso.

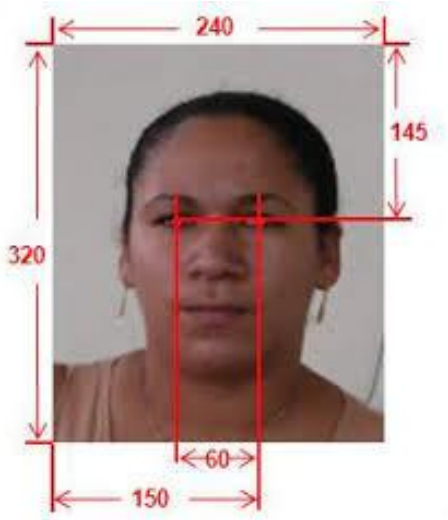

Figura 26: Medidas imagen facial (fuente: 21)

9.2. Entrenamiento

La fase de entrenamiento ha sido implementada para leer imágenes directamente del disco duro. El programa creará el nombre de las imágenes automáticamente y las leerá de la carpeta en la que han sido guardadas. Todas las imágenes han sido tomadas en el mismo entorno, con las mismas sombras, condiciones ambientales y resolución.

El problema que presenta la aplicación es que el método de PCA es realizado sobre todas las imágenes de una base de datos. Por lo que si se desea introducir una nueva persona en dicha base de datos. El proceso deberá ser realizado de nuevo para calcular la nueva matriz de covarianza y los nuevos vectores propios. Este proceso es un proceso que lleva un tiempo de computación por lo que el tamaño y número de imágenes juegan un papel fundamental en el tiempo de computación de la aplicación.

De todas formas, este problema solo se encuentra en la fase de entrenamiento, ya que en la fase de reconocimiento la imagen desconocida tendrá que ser pasada por la PCA que debe estar guardada y simplemente ha de ser cargada al proyecto para ser utilizada.

El último paso de la fase de entrenamiento supone transformar una a una cada imagen que se ha utilizado para crear el PCA, con los nuevos ejes disponibles. El vector resultante será almacenado para poder ser comparado con posterioridad por las imágenes de reconocimiento.

9.3. Reconocimiento

Para realizar el reconocimiento de un sujeto, en primer lugar se ha de procesar la imagen como se ha explicado en el apartado de lectura y procesamiento de la imagen, a continuación la imagen se proyectara en los nuevos ejes de la PCA elaborada en el apartado de entrenamiento, y el vector resultante será comparado con los vectores almacenados en la fase anterior. Si a la

hora de comparar hay un valor por debajo del valor umbral, el reconocimiento será dado por válido.

El valor umbral es necesario para evitar falsos positivos, ya que el valor identifica la distancia menor de muchos valores, lo que no significa que sea suficiente para haber reconocido al individuo. La elección del valor del umbral debe ser tomada con cautela ya que con un valor muy alto podría la imagen más parecida sería dada por buena pero podría no ser la correcta, o uno muy bajo podría dar por no reconocido a un individuo que realmente sí lo ha sido.

Para reducir carga al proceso, no se comparara cada valor obtenido en por la distancia euclidea con el valor umbral sino que se obtendrá el valor mínimo del programa y se comparará con el valor del umbral. Si este valor es menor, significa que la persona ha sido reconocida, si es mayor, la persona a reconocer no ha sido identificada.

El valor umbral se obtiene de forma experimental, utilizando distintos sujetos hasta alcanzar un valor orientativo que nos proporcione seguridad a la hora de reconocer a un sujeto.

10. Software

Para la implementación de este proyecto se ha utilizado un entorno Ubuntu que ha utilizado el apoyo del programa NetBeans en el que ha sido necesario realizar una serie de ajustes para poder instalar las librerías OpenCV en el programa.

Las funciones que se detallaran a lo largo de este apartado serán las necesarias para el reconocimiento para la base de datos FEI ya que es la única base de datos que está a color, por lo que hay que pasarla a escala de grises y es la base de datos en la que las imágenes tienen mayores dimensiones.

Figura 28: Logo NetBeans (fuente: 22)

10.1. Clases de OpenCV

Para la implementación de este proyecto se ha tenido que dar uso de unas clases y funciones ya definidas en las librerías OpenCV. Las utilizadas en el proyecto son las siguientes:

- Clase Mat. Se trata de una estructura que representa una matriz de datos, pudiendo ser monocanal o multicanal. En esta aplicación se utiliza para almacenar las imágenes normales, asi como las editadas y para almacenar vectores, matrices e histogramas. Las funciones utilizadas en el proyecto son:
  - Size: Devuelve el tamaño de la matriz
  - Rows/cols: Devuelven el número de filas o de columnas y permiten acceder a datos concretos de la matriz.
  - Reshape: Cambia las dimensiones de la matriz.
  - convertTo. Convierte el tipo de datos de una matriz a otro.
  - Release. Anula la matriz y libera memoria.
  - At: Devuelve el valor de la posición exacta que se le indique
- Clase Rect. Almacena las coordenadas de un rectángulo.
- Ventanas. Las librerías OpenCV incluyen unas sencillas ventanas que serán utilizadas para representar imágenes para probar el correcto funcionamiento del proyecto.
  - namedWindow: Crea una ventana
  - Imshow: muestra una imagen en una ventana ya abierta.
  - destroyWindow. Cierra la ventana abierta.
- waitKey. Utilizado a la hora de representar las imágenes, para que la imagen esté disponible el tiempo deseado waitKey(tiempo deseado). Si por el contrario queremos que esté disponible hasta que se pulse una tecla, bastar con utilizar la función waitKey(0).

- PCA. OpenCV incluye unas funciones que son las encargadas de calcular los eigenvalues, eigenvector y media de un subespacio. Así como de proyectar nuevos vectores sobre esos ejes calculados. Esta clase viene programada para recibir una matriz de datos. De esta matriz calcula la matriz de covarianza, y a continuación calcula los eigenvectors. El número de eignevector a calcular lo puede elegir el usuario para poder descartar los de menor importancia. En la aplicación se ha decidido elegir 100 eigenvectors ya que tras indicar de forma experimental el porcentaje de información deseada era un valor suficientemente fiable. Se realiza en el siguiente comando.

```
PCA pca(data,Mat(),PCA::DATA_AS_ROW,EIGEN);
```

  Donde data es la matriz donde se encuentran todas las imagenes transformadas en filas y por ello el valor de DATA_AS_ROW.

  Por último la clase PCA dispone de otras funciones que se utilizarán en el presente proyecto.

    - project: Esta función nos permite cambiar los ejes de una imagen y ponerla en función de los eigenvectors del nuevo subespacio.
    - eigenvectors: devuelve una matriz donde se encuentran todos los eigenvectors
    - eigenvalues: devuelves una matriz de una fila donde se encuentran todos los eigenvalues
    - mean: devuelve la imagen media del subespacio.
- Imread. Función de openCV encargada de leer archivos de imagen y cargarlos en el programa en una estructura Mat.

10.2. Variables y constantes

A continuación, se definirán las variables globales que se utilizarán en el proyecto.

10.2.1. Constantes

- EIGEN. Numero de Eigenvectors que se utilizarán a la hora de crear la PCA.
- FOTOS. Número de imágenes que tiene la base de datos a utilizar.

10.2.2. Variables

- Data. Variable global de estructura Mat empleada para almacenar todas las imágenes de la base de datos a modo de vectores.
- BD. Array bidimensional de tipo int empleado para almacenar la proyección de cada imagen de la base de datos por filas.

10.3. Información de la imagen facial

Para realizar el proyecto se ha de procesar la imagen del sujeto a reconocer de manera que solamente se obtenga la información facial del sujeto. Para obtener el resultado deseado se hará uso de las siguientes funciones.

- Funcion mostrarinfoCara: A esta función se le pasa el nombre de la imagen en la cual se encuentra el sujeto del que se desee encontrar la imagen facial. La función lee la imagen

y le aplica un clasificador que ya viene definido en las librerías openCV. Este clasificador se denomina **haarcascade clasifier.**

En la función se utilizarán dos clasificadores de este tipo en primer lugar se comprobará que existe el clasificador para imágenes frontales "haarcascade_frontalface_alt.xml" y en el caso de no encontrar ninguna imagen frontal, se comprobará el clasificador para imágenes de perfil "haarcascade_profileface.xml" como vemos a continuación:

CascadeClassifier detector;

```
if(!detector.load("haarcascade_frontalface_alt.xml"))
if(!detector.load("haarcascade_profileface.xml"))
cout << "No se puede abrir clasificador." << endl;
```

Después de esto la función pasa la imagen a escala de grises y ecualiza el histograma de la imagen. Esta última imagen (dest) es a la que se le aplica el clasificador en la siguiente línea:

```
vector<Rect> rect;
detector.detectMultiScale(dest, rect);
```

Esta función nos devuelve los valores del cuadrado en el que se encuentra la cara, en el caso de no devolver ningún valor se hará lo propio con el clasificador de imágenes de perfil, ya que por defecto empieza con el clasificador de imágenes frontales.

Por último, se ajustan los valores del rectángulo a la altura y anchura deseada para el proyecto y obtenemos el siguiente resultado.

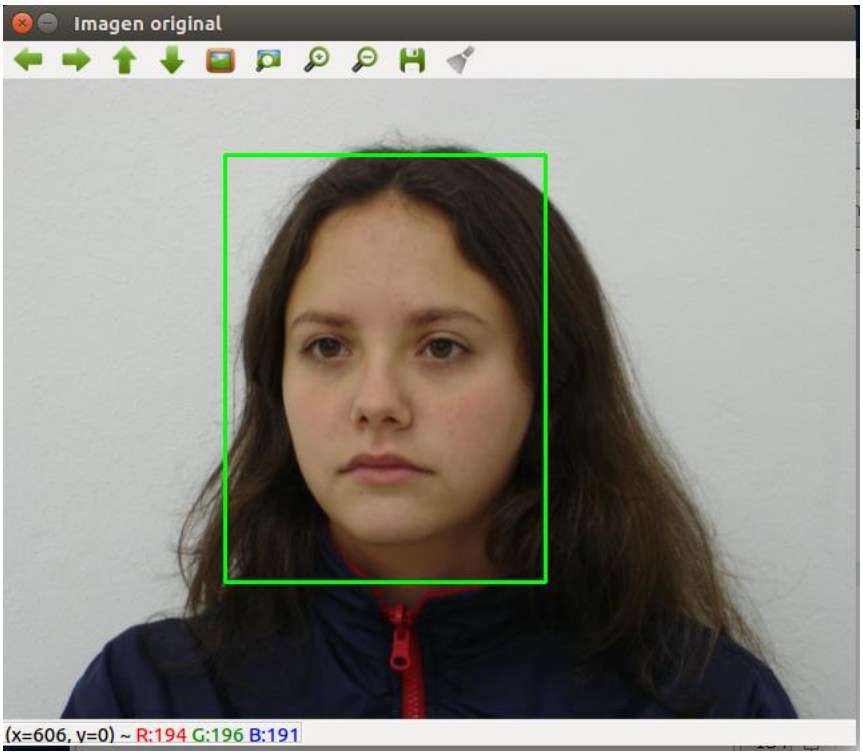

Figura 29: Imagen facial detectada

- Función infoCara: Esta función crea una nueva estructura Mat en la que se guarda la información de la imagen una vez ha sido ya detectado el valor del rectángulo donde se encuentra la cara, como hemos podido observar en la función anterior. Esta imagen será una imagen en escala de grises y con el histograma realizado.

  El guardado de la imagen facial se realiza con los siguientes comandos:

```
ancho=(rect[0].x+rect[0].width)/2+ajusteAncho;

alto=(rect[0].y+rect[0].height)/2+ajusteAlto;

Rect limites((ancho-anchoTotal/2),(alto-altoTotal/2),anchoTotal,altoTotal);

info=imagen(limites);
```

10.4. Entrenamiento

Una vez realizada la detección facial de los sujetos, se entrenará al proyecto para obtener el subespacio PCA, así como la proyección de todas las imágenes de los sujetos. Para ello, se utilizaran las siguientes funciones.

El funcionamiento de este apartado se consigue leyendo las imágenes de cada individuo que van a formar parte de la base de datos que queremos utilizar, una vez leídas, pasan por el proceso de ser aplanadas para posteriormente ser introducidas en una matriz en la que cada fila será una imagen diferente. Esta matriz es en la que se basa la función PCA para realizar el nuevo subespacio. Por ultimo cada imagen que ha creado ese subespacio será proyectada en los nuevos ejes y el resultado en forma de vector será almacenado en una nueva matriz para que posteriormente puedan ser comparados con las imágenes que se desean reconocer.

En primer lugar hay que colocar todas las imágenes de la base de datos en una solo matriz teniendo cada imagen en una fila de la matriz. Utilizaremos las funciones:

- crearNombre: Esta función recibe una variable de tipo int a la que le corresponde el número del sujeto a crear y una cadena de caracteres la cual devuelve con el valor del int sin '0' por delante para después concatenarla con otras cadenas para sacar el nombre completo de la imagen.

- cuatroFotos: Recibe una cadena de caracteres a la cual le devuelve 4 distintas cadenas en las que tiene el nombre completo de las cadenas de caracteres de las 4 imágenes de cada persona que se quiere reconocer. Lo realiza mediante el siguiente comando.

```
strcpy(foto1,aux);

strcat(foto1,"-02.jpg\0");
```

Estas imágenes tienen el formato "Numerosujeto-numFoto.jpg" siendo el número de foto 02,05,11,12. Dependiendo de la posición de la cara del sujeto.

- Funcion aplanaMatriz: Le es indiferente que la imagen que recibe sea facial o no. Simplemente coge una matriz de N dimensiones y la convierte en una de una sola fila poniendo las filas una detrás de otra mediante el comando reshape.

```
array=imagen.reshape(1,1);
```

- Funcion aplanaCara: Esta función es la encargada de realizar todo el proceso que se encarga de pasar cada imagen a la matriz a la que se le realizará el PCA. El objetivo de la función es devolver un vector en el que se encuentra la información de una imagen facial, ya en escala de grises para más adelante introducirlo en una matriz en la que se encuentren todas. Para ello, lee la imagen donde se encuentra el sujeto a reconocer, llama a la función infoCara que le devuelve la matriz cara y por último, llama a la función aplanaMatriz para que le devuelva el esa misma imagen pero en una sola fila.

```
imagen=imread(nomfich);
```

```
cara=infoCara(imagen);
```

```
caraplan=aplanarMatriz(cara);

return (caraplan);
```

- insertarFila: Recibe una matriz de una sola fila y el número de imagen correspondiente y la introduce en la posición que indica dicho número en la matriz data. Además incrementa el valor del número de imágenes insertadas en la matriz data (num).

```
foto.copyTo(data.row(*num));
(*num)++;
```

- funcionAplanaBD: Es la función encargada de completar la matriz data con todas las imágenes de la base de datos. Lo hace apoyándose en las funciones anteriores.

```
Mat fila;
  for(i=1;i<FOTOS/4+1;i++){
     crearNombre(i,aux);
     cuatroFotos(aux,foto1,foto2,foto4,foto5);
     fila=aplanaCara(foto1);
     insertarFila(&num,fila);
```

Una vez completada la matriz data con la información de todas las imágenes, se procede a crear el subespacio PCA. Para ello utilizaremos estas funciones.

- guardarPCA: Recibe el nombre del archivo donde se guardara y la PCA creada y utilizando la función FileStorage guarda los valores de la media, los eigenvalues y los eigenvectors en el sistema.

```
FileStorage fs(file_name,FileStorage::WRITE);
  fs << "mean" << pca_.mean;
  fs << "e_vectors" << pca_.eigenvectors;
  fs << "e_values" << pca_.eigenvalues;
  fs.release();
```

- crearPCA: Es la encargada de realizar el proceso entero. Primero llama a la función funcionAplanaBD, después mediante la función pca de las librerías openCV crea el subespacio PCA, y por último guarda dicho subespacio mediante la función guardarPCA.

```
funcionAplanaBD();
  PCA pca(data,Mat(),PCA::DATA_AS_ROW,EIGEN);
  guardarPCA("ACP",pca);
  cout << " DONE"<< endl;
```

Para terminar con la fase de entrenamiento se debe proyectar cada imagen de la base de datos una por una y guardarlas en una nueva matriz. Para ello se utilizarán las siguientes funciones.

- cargarPCA: Proceso opuesto a la función guardarPCA. Se emplea esta función para ahorrarnos tiempo y no tener que realizar el proceso anterior cada vez que se quiera reconocer una imagen.

```
FileStorage fs(file_name,FileStorage::READ);
  fs["mean"] >> pca_.mean ;
  fs["e_vectors"] >> pca_.eigenvectors ;
  fs["e_values"] >> pca_.eigenvalues ;
  fs.release();
```

- insertarFila2: función parecida a insertarFila solo que esta función inserta filas de mucha menor longitud en la matriz BD.
- volcarMatrizaFichero: Esta función es la encargada de guardar la matriz BD en un archivo de datos para poder cargarla cuando se necesite.

- entrenamiento: realiza el proceso completo del entrenamiento. En primer lugar reserva memoria para el array bidimensional, llama a la función cargarPCA, y mediante el mismo proceso que las imágenes fueron insertadas en la matriz data inserta la fila resultante de la proyección en la matriz BD. Por último guarda la matriz en un fichero.

```
int **BD=ReservaMemoria1(FOTOS,EIGEN);
cargarPCA("ACP",pca_);
for(i=1;i<201;i++){
        crearNombre(i,aux);
        cuatroFotos(aux,foto1,foto2,foto4,foto5);
        fila=aplanaCara(foto1);
        fila=pca_.project(fila);
        insertarFila2(&num,fila,BD);
            ……
volcarMatrizAFichero("datosBD.dat",BD,dimor);
```

Donde dimor, son las dimensiones de la matriz.

10.5. Test

La última fase del proyecto es la encargada de reconocer al sujeto de una imagen dada. Para ello, además de algunas utilizadas anteriormente, las funciones a utilizar son las siguientes.

- volcarFicheroMatriz: función encargada de cargar al proyecto la matriz BD para utilizarla posteriormente.
- calcularDistanciaEuclidea: recibe la matriz BD, el número de fila y la cara proyectada a reconocer y devuelve un int con la distancia euclídea entre los dos vectores.

```
for(int i=0;i<EIGEN;i++){
        array+=sqrt((nuevo[i]-BD[col][i])*(nuevo[i]-BD[col][i]));
```

- encontrarMin: Recibe un array y devuelve la posición del valor mínimo del mediante una búsqueda secuencial.
- comparaMin: Recibe el valor minimo del array y lo compara con el valor determinado como el valor umbral.
- reconocerPersona: encargada de realizar el proceso completo de reconocimiento. En primer lugar llama a la función cargarPCA, a continuación carga la matriz BD mediante la función volvarFicheroArray, aplana la imagen a reconocer y la proyecta en los nuevos ejes. Después compara dicho array con cada fila de la matriz BD y encuentra el mínimo. Por último, divide la posición en la que se encuentra el minimo, lo divide entre 4 y sumándole la unidad al valor asignado a la persona indicada y lee su nombre del fichero de nombres.

```
PCA pca_;
cargarPCA("ACP",pca_);
imagen=aplanaCara(nomfich);
imagen=pca_.project(imagen);
for(int i=0;i<dim.numFilas;i++){
        array[i]=calcularDistanciaEuclidea(BD,i,imagen);
persona=encontrarMin(array,FOTOS);
persona=persona/4+1;
```

```
fichPersonas.open("nombres.txt");
while(getline(fichPersonas,linea)){
        if(contador==persona)
                cout<<linea<<endl;
        contador++;
}
```

11. Resultados

A continuación se explicaran los distintos procesos utilizados para el método de reconocimiento facial mediante PCA alterando características como número de autovalores, número de fotos, bases de datos y numero de fotos por individuo en la fase de datos.

11.1. Resultados según número de autovalores

Las imágenes de test han sido probadas con 20, 50, 100 y 200 autovectores. No se han querido utilizar más autovectores ya que el proceso con 200 autovectores ya sobrepasaba el tiempo aceptable para un reconocimiento lo suficientemente dinámico.

Este experimento se ha realizado con 16 individuos para todas las bases de datos e incluyendo 4 imágenes de cada individuo en la base de datos.

Para obtener el resultado final, se han realizado 25 procesos de identificación con cada base de datos.

Los resultados son los siguientes:

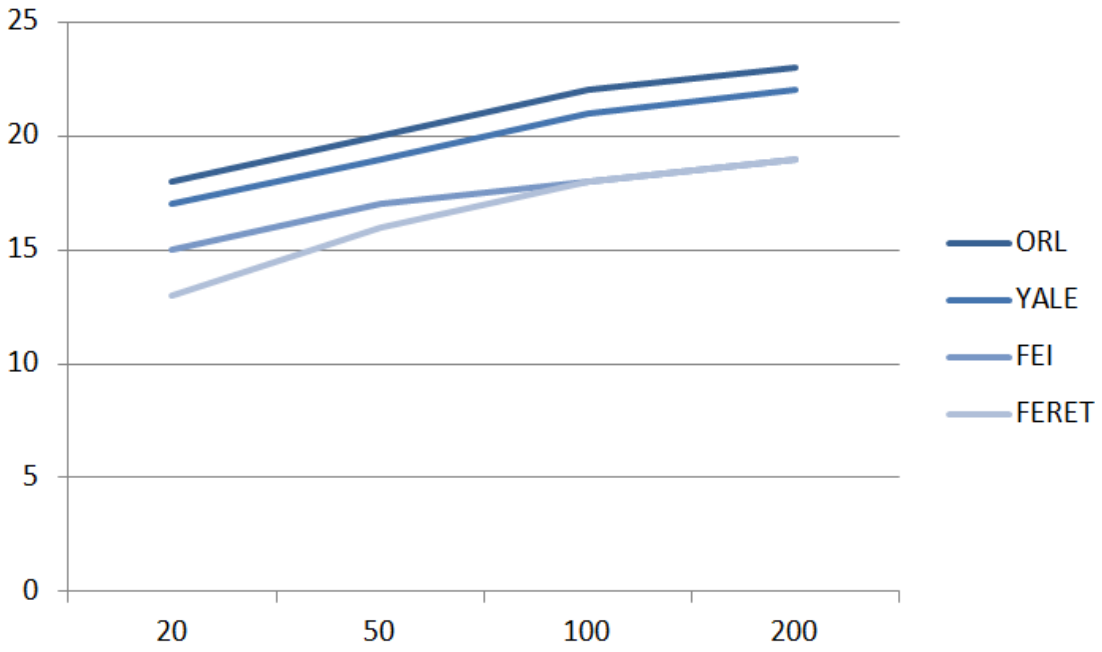

Figura 30: Reconocimiento según nº autovalores

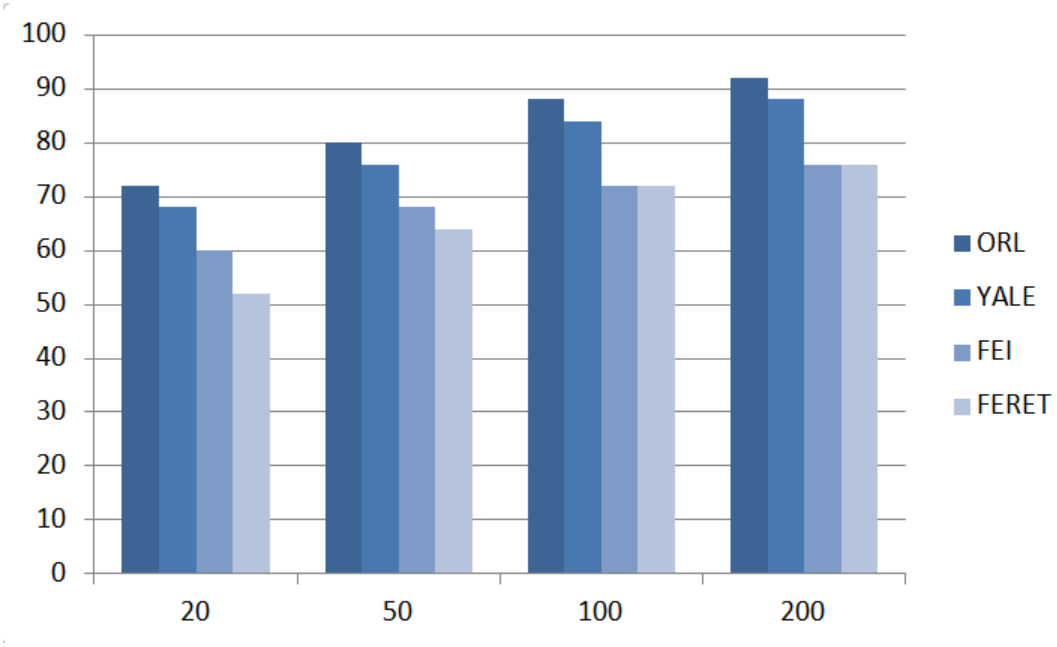

Figura 31: Porcentaje reconocimiento según nº autovalores

Como podemos observar el número de autovalores elegidos es un dato crítico ya que los valores en los que los resultados son lo suficientemente fiables empiezan a coincidir con los del tiempo máximo posible. Ya que con 50 autovalores los resultados son (ORL-80,9%,YALE-75,6%,FEI-69,8% Y FERET-63,4%) que depende de la forma de tomar las imágenes será fiable o no. Por lo que en este caso solamente se darán por fiables los resultados a partir de 100 autovectores.

11.2. Resultados según número de individuos totales

Otro dato de interés que se ha decidido analizar es el número de imágenes utilizadas en el proceso, ya que se ha notado que un gran número de imágenes totales no solo afecta a la velocidad de computación sino que también al porcentaje de éxito en el reconocimiento.

Para este apartado se ha decidido analizar los resultados con 5 personas, 16, 40 y 200. Ya que 16 es el límite de imágenes de la base de datos de Yale, 40 el de ORL y 200 el de la base de datos FEI. Para la creación de la base de datos propia se han incluido 4 imágenes de cada individuo utilizando las restantes para el reconocimiento de usuarios desconocidos.

Para tener un nivel de porcentajes aceptable y un tiempo de computación aceptable hemos decidido utilizar 100 autovectores.

Para obtener los resultados se han realizado 25 procesos de identificación con cada base de datos.

Los resultados son los siguientes:

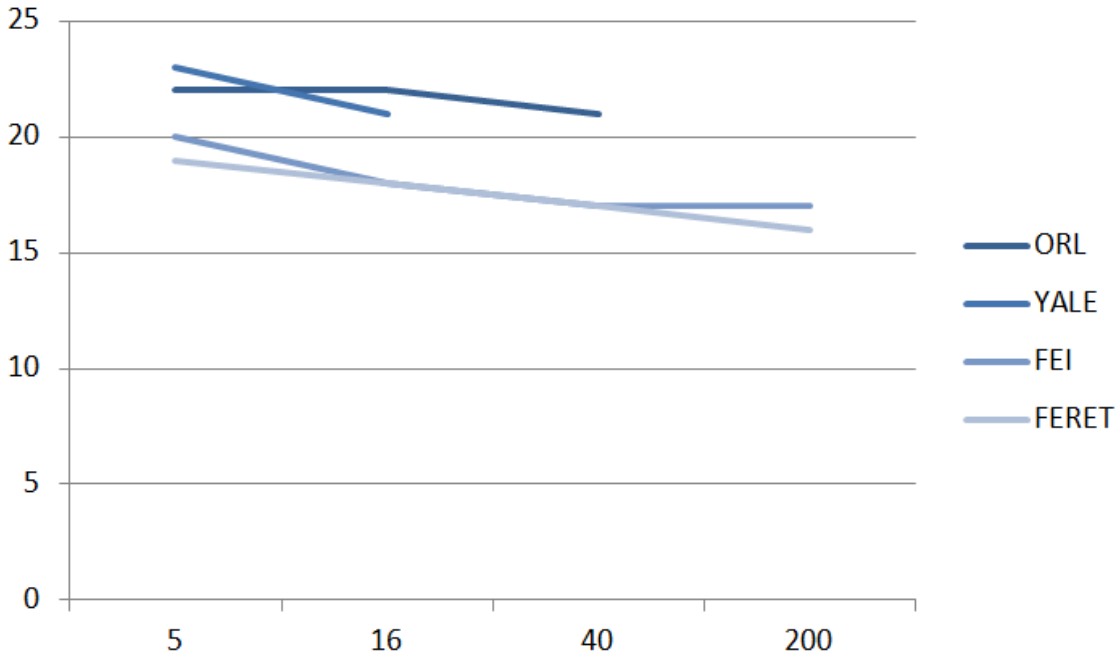

Figura 32: Resultados según el número de individuos

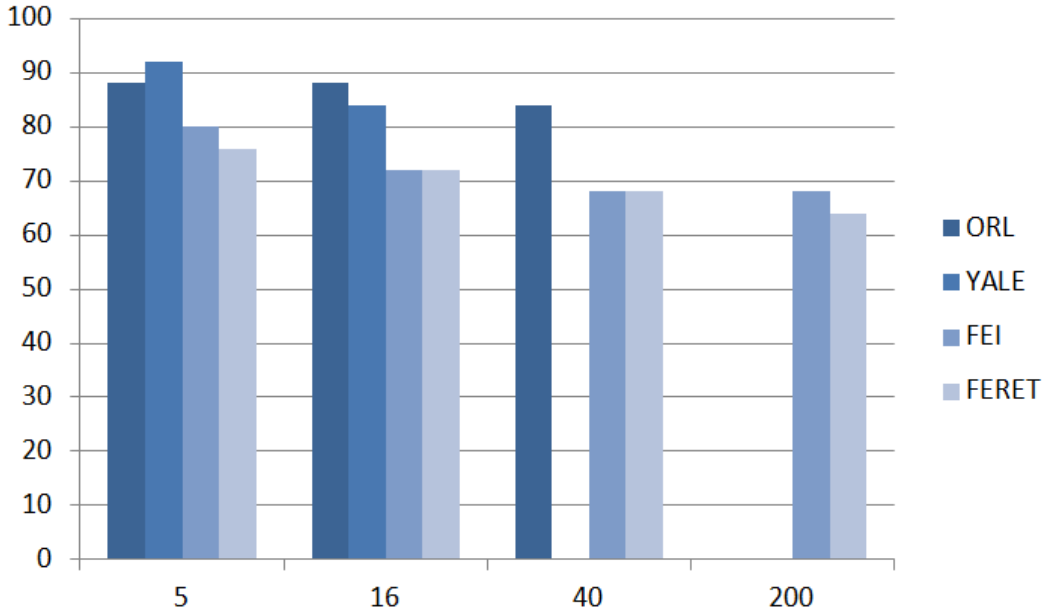

Figura 33: Resultados según el número de individuos

Como se aprecia en los resultados el número de individuos es un dato a tratar ya que el proyecto se desea utilizar con grandes bases de datos de personas, para evitar aglomeraciones. En los resultados se aprecia como el porcentaje de aciertos disminuye al aumentar el número de individuos pero, podemos concluir que si la toma de imágenes es adecuada al proceso como en las bases de datos ORL y YALE podrán utilizarse un gran número de individuos para que en un futuro puedan ser reconocidos.

11.3. Resultados según el número de imágenes almacenadas

El último dato que se analiza en el presente proyecto es el del número de imágenes de cada individuo que se desean incluir dentro de la base de datos.

En este caso estudiaremos los casos de incluir de 8 a 2 imágenes dentro de la base de datos del proyecto y utilizar las restantes para el reconocimiento de individuos desconocidos.

Una vez más, se realizaran 25 reconocimientos con cada base de datos para obtener un porcentaje fiable del estudio realizado.

En esta ocasión, se utilizaran 16 imágenes de cada base de datos en las que serán utilizados 100 autovectores como en el apartado anterior.

Los resultados son los siguientes:

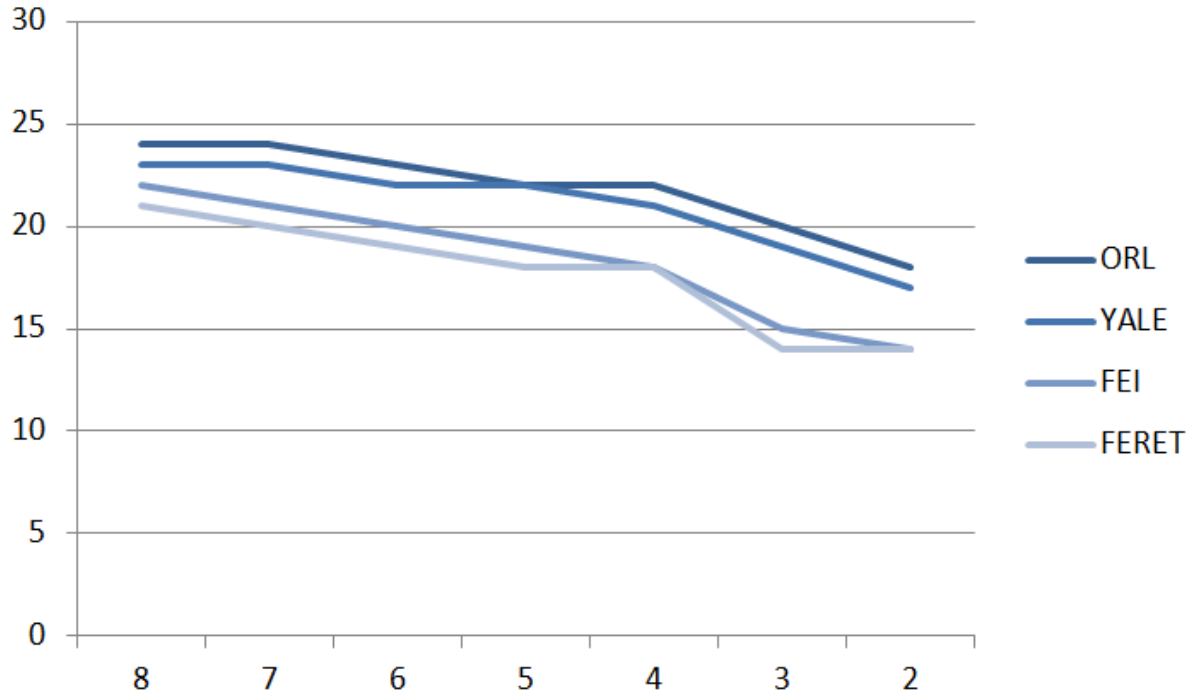

Figura 34: Porcentaje de acierto según número de imágenes almacenadas

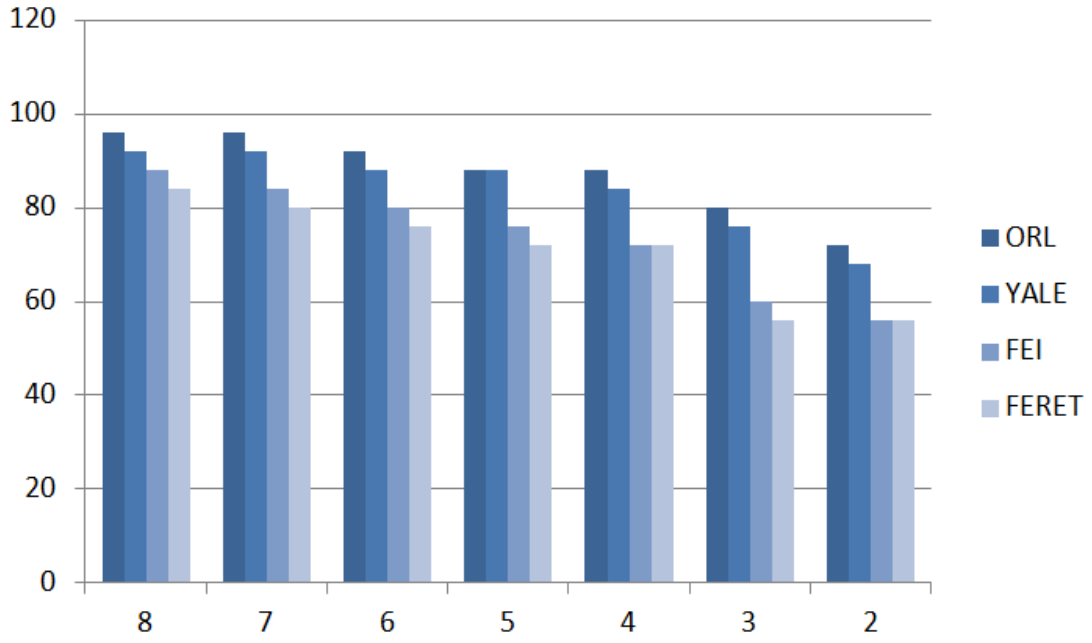

Figura 35: Porcentaje de acierto según número de imágenes almacenadas

En base a los resultados, comprobamos que cuantas más imágenes de cada individuo estén incluidas en la base de datos, la probabilidad de que el sujeto sea identificado aumenta considerablemente. Además, podemos observar que hace falta un número mínimo de imágenes necesarias de cada individuo que rondará el 4.

11.4 Análisis de errores

Existen dos tipos de errores:

- Falso positivo: El programa cree haber reconocido al sujeto pero la identificación es incorrecta. El resultado está por debajo del umbral mínimo pero la identificación es incorrecta.
- Negativo: El programa no encuentra un sujeto lo suficientemente parecido al individuo a identificar. En este caso, el resultado estará por encima del umbral mínimo.

En este caso se estudian los errores para las distintas bases de datos con los mismos valores de autovectores y el mismo umbral mínimo:

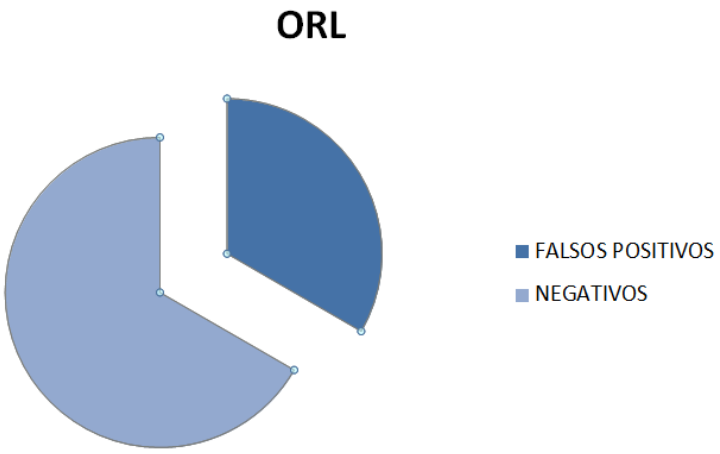

Figura 36: Porcentaje tipos de error para base de datos ORL

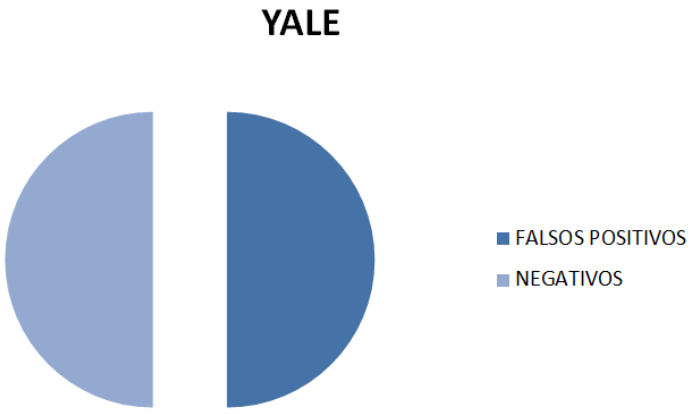

Figura 37: Porcentaje tipos de error para base de datos YALE

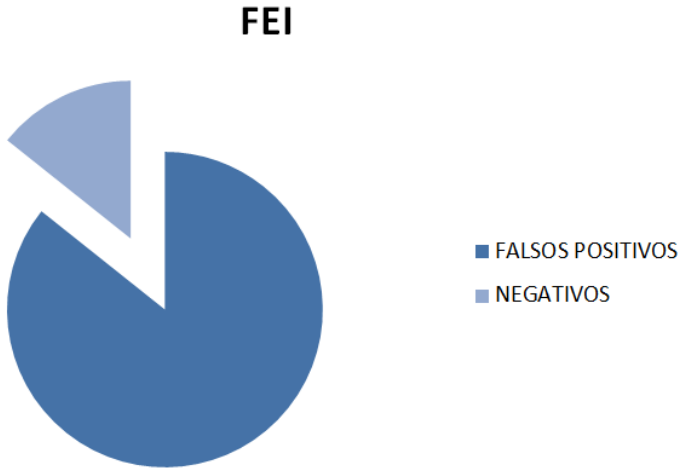

Figura 38: Porcentaje tipos de error para base de datos FEI

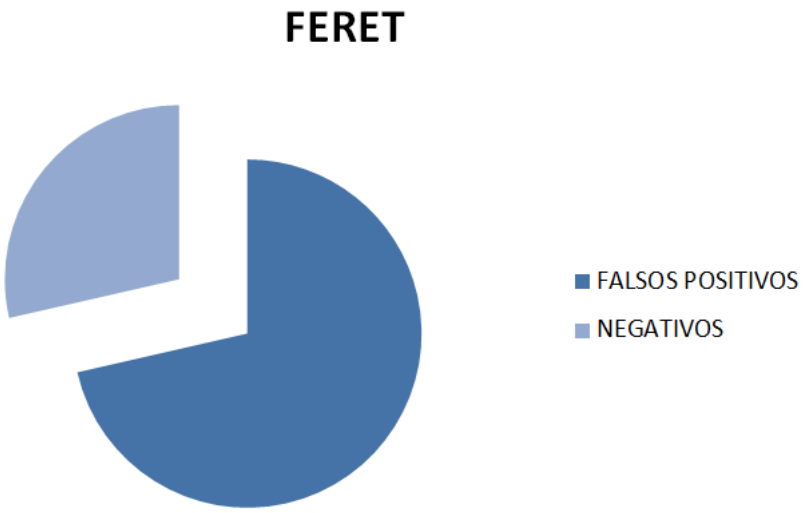

Figura 39: Porcentaje tipos de error para base de datos FERET

En base a los resultados vemos que el valor umbral juega un papel fundamental la elección del sujeto ya que como podemos comprobar funciona bien con la base de datos ORL pero en cambio el resto tienen altos porcentajes de falsos positivos. Esto implica que para cada base de datos el valor del umbral mínimo debe ser calculado minuciosamente para evitar tener que realizar comprobaciones manuales una vez el programa este en funcionamiento.

12. Conclusiones

El presente Trabajo Fin de Grado ha mostrado diferentes técnicas de reconocimiento facial y ha profundizado en el método de PCA.

Tras los resultados obtenidos y en vista de los documentos estudiados, podemos concluir que el sistema de reconocimiento PCA es un método valido para el reconocimiento, a pesar de que, como todo algoritmo de reconocimiento facial, presente tasas de error considerables debido a la iluminación.

Al tratar la imagen solamente en escala de grises, los rasgos característicos de la cara no tienen tanto peso como deberían. Características como la iluminación toman papeles demasiado importantes en el proyecto, por lo que la toma de imágenes ha de ser con la misma luz para todas las imágenes. Por lo que debe ser un recinto cerrado sin luz exterior que varíe a lo largo del día.

El fondo de la imagen, en cambio, no afecta en absoluto al proceso de reconocimiento, ya que el programa recorta por si solo el rostro del individuo y no analiza lo que lo rodea.

Aun así, en vista de los resultados el proyecto podría ser llevado a cabo teniendo en cuenta las siguientes observaciones:

- Las imágenes utilizadas deberán ser, en la medida de lo posible, frontales, ya que facilita el proceso de detección de rostro y sobre todo el de creación y utilización del nuevo subespacio.
- Se deberán evitar excesos de iluminación, como reflejos, así como deberá realizarse la adquisición de imágenes superando unos niveles mínimos de iluminación.
- Serán necesarias al menos 4 imágenes de cada individuo para introducir a la base de datos del programa.

13. Descripción de tareas. Gantt

El presente Trabajo Fin de Grado consta de 4 fases: Información, OpenCV, práctica y redacción.

1. INFORMACIÓN
   En esta fase se realiza una recopilación de información sobre un tema. Como es un tema que en un primer lugar era desconocido para el estudiante, supondrá un largo trabajo de aprendizaje, en el que también se incluyen el procesamiento y análisis de la información. Esta fase da comienzo el 10 de Febrero y termina el 15 de abril, donde se cumple el primer hito.
2. OpenCV
   Esta fase consiste en habituarse a las librerías OpenCV utilizadas en el proyecto. Estas librerías son completamente nuevas para el alumno por lo que ha de realizarse este proceso de aprendizaje y adaptación. Esta fase se inicia a la vez que la fase anterior y tiene una duración de 20 días en los que el alumno adquiere los conocimientos necesarios para emplear las nuevas bibliotecas. El segunde hito concluye con el término de esta tarea.
3. PARTE PRACTICA
   Esta fase es la fase en la que se programa el proyecto utilizando tanto el programa Netbeans como las librerías OpenCV. Para realizar esta fase los dos primeros hitos se habrán tenido que cumplir. Por lo que la fase comenzará el 15 de abril y concluirá el 12 de Junio, cumpliéndose el tercer hito del trabajo.
4. REDACCIÓN
   Esta tarea plasmará todo lo que se ha realizado en las fases anteriores. En ésta se redacta el Trabajo Fin de Grado, con la estructura y formato adecuado. Para poder iniciar esta fase deben estar terminados todos los apartados anteriores. La tarea comienza el 15 de Junio y concluye el 22 de Julio. Cumpliéndose el cuarto y último hito.

A continuación se muestra el diagrama de Grantt:

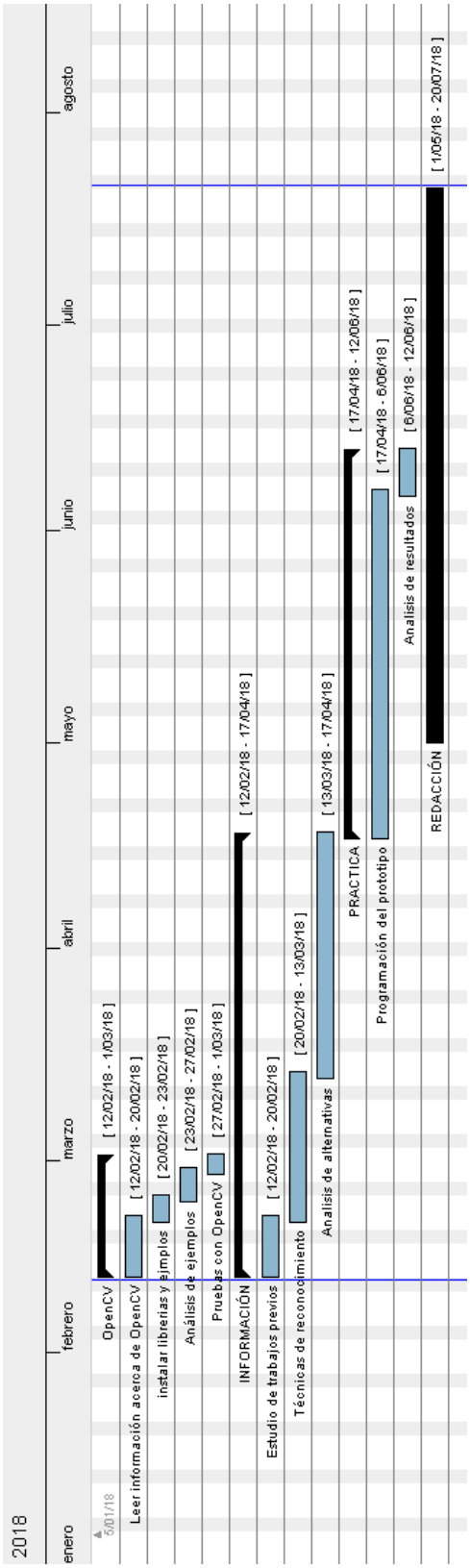

Figura 40: Diagrama de Gantt del proyecto

14. Desglose de costes

A continuación, se presenta el desglose de costes total del proyecto. Los costes serán divididos en tres apartados: horas internas, amortizaciones y gastos. También se incluirá una tabla resumen donde aparecerán los costes totales de cada partida, además de los costes indirectos.

Tabla 3: Partida de horas internas

HORAS INTERNAS			
Concepto	Nº de horas	€/hora	Total
Ingeniero graduado	250	18	4.500€
Ingeniero senior	25	50	1.250€
Total			5.750€

Tabla 4: Partida de amortizaciones

Amortizaciones				
Concepto	Precio adquisición	Vida útil	Utilización	Total
Licencia de Office	90€	1800h	30h	1,50€
Ordenador	1.300€	5000h	200h	52€
Total				53,50€

Tabla 5: Partida de costes

Costes	
Concepto	Total
Material oficina	100€
Material bibliográfico	30€
TOTAL	130€

Tabla 6: Resumen de los costes

Resumen	
Horas internas	5.750€
Amortizaciones	53,50€
Costes	130€
TOTAL	5.933,30€
Costes indirectos (5%)	296,67€
TOTAL	6229,97€

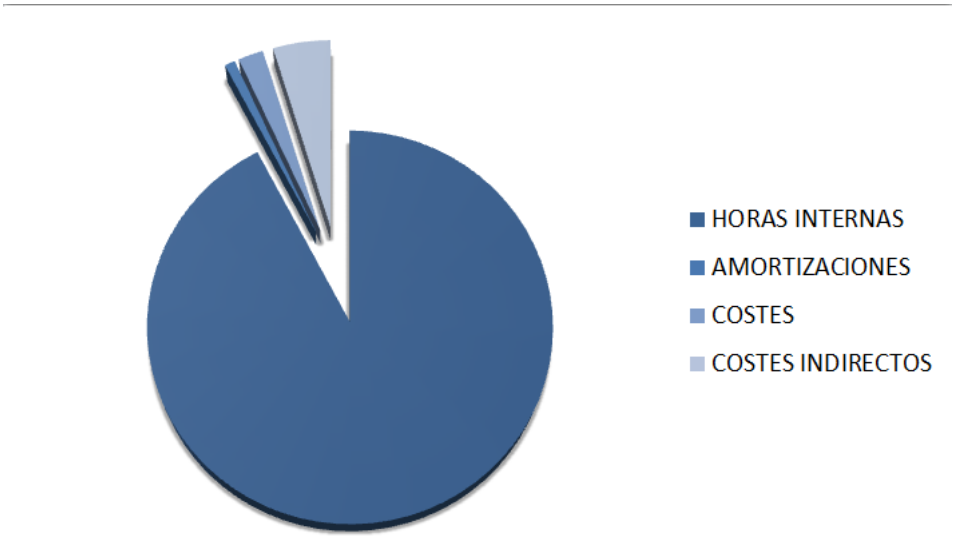

Figura 41: Desglose costes

15. Referencias

[1] W. W. Bledsoe, The model method in facial recognition, Technical report, Panoramic Research Inc, 1966.

[2] A. J. Goldstein, L. D. Harmon, and A. B. Lesk, “Identification of Human Faces”, Proc. IEEE, May 1971, Vol. 59, No. 5, 748-760.

[3] L. Sirovich and M.Kirby, “A Low-Dimensional Procedure for the Characterizacion of Human Faces”, J. Optical Soc. Am. A, 1987, Vol. 4, No. 3, 519-524.

[4] M. A. Turk and A. P. Pentland, “Face Recognition Using Eigenfaces”, Proc. IEEE, 1991, 586-591.

[5] Paul Viola y Michael Jones, Rapid Object Detection using a Boosted Cascade of Simple Features, 2001

[6] Yoav Freund and Robert E. Schapire. A decision-theoretic generalization of on-line learning and an application to boosting. In Computational Learning Theory: Eurocolt ’95, pages 23–37. Springer-Verlag, 1995.

[7] Jianbo Shi and Carlo Tomasi. Good features to track. In 9th IEEE Conference on Computer Vision and Pattern Recognition (CVPR). Springer, 1994

[8] Bruce D. Lucas and Takeo Kanade. An Iterative Image Registration Technique with an Application to Stereo Vision. International Joint Conference on Artificial Intelligence, pages 674–679, 1981.

[9] Carlo Tomasi and Takeo Kanade. Detection and Tracking of Point Features. Carnegie Mellon University Technical Report CMU-CS-91-132, April 1991.

[10] I. T. Jolliffe, Principal Component Analysis, Springer Verlag, 1986.

[11] P. N. Belhumeur, J. P. Hespanha and D. Kriegman, Eigenfaces vs. Fisherfaces: recognition using class specific linear projection, Pattern Analysis and Machine Intelligence, IEEE Transactions, 1997.

[12] Marian Stewart Bartlett, Member, IEEE, Javier R. Movellan, Member, IEEE, and Terrence J. Sejnowski, Fellow, IEEE

[13] An information-maximization approach to blind separation and blind deconvolution. *Bell AJ, Sejnowski TJ*

[14] V. Vapnik, The Nature of Statistical Learning Theory, 1995.

[15] Zhang Yankun, Liu Chongqing. Face recognition using kernel principal component analysis and genetic algorithms

[16] L. Wiskott, J. M. Fellous, N. Krüger and C. Von Der Malsburg, Face recognition by elastic bunch graph matching, IEEE Transactions on Pattern Analysis and Machine Intelligence, 1997.

[17] T. Ahonen, A. Hadid and M. Pietikainen, Face recognition with local binary patterns, In Tomás Pajdla and Jirí Matas, editors, Computer Vision, 2004.

[18] T. Ahonen, A. Hadid and M. Pietikainen, Face description with local binary patterns: Application to face recognition, IEEE Trans. Pattern Anal. Mach. Intell, 2006.

[19] F. Samaria and S. Young, Hmm-based architecture for face identification, Image and Vision Computing, 1994.

[20] Lance, G. N.; Williams, W. T. (1966). "Computer programs for hierarchical polythetic classification ("similarity analysis")". Computer Journal. 9 (1): 60–64.

[21] Lance, G. N.; Williams, W. T. (1967). "Mixed-data classificatory programs I.) Agglomerative Systems". Australian Computer Journal: 15–20.

[22] Aashish & Vijayalakshmi, Orient. J. Comp. Sci. & Technol., Vol. 10(1), 151-159 (2017)

[23] P. Jonathon Phillips. Support Vector Machines Applied to Face Recognition

[24] ISO/IEC 19794 Information technology — Biometric data interchange formats — Part 5:Face image data, ISO/IEC 19794-5, 2005

16. Bibliografía

1. Wikipedia, *OpenCV* [http://es.wikipedia.org/wiki/OpenCV], consultado en Marzo 2014

2. Servo magazine 04-2007 http://s1.nonlinear.ir/epublish/magazine/Servo/Servo%5Bnonlinear.ir%5D%202007-04.pdf

3. Alphonse Bertillon, Alphonse Bertillon's instructions for taking descriptions for the identification of criminals, and others, by the means of anthropometric indications, 1889.

4. Use Netbeans to work with OpenCV on Ubuntu https://thefreecoder.wordpress.com/2012/09/10/use-netbeans-to-work-with-opencv-on-ubuntuand-linux-mint-and-other-distros-too/

5. OpenCV en Linux. https://geekytheory.com/opencv-en-linux

6. S. G. Mallat, A theory for multiresolution signal decomposition: the wavelet representation, IEEE Transactions on Pattern Analysis and Machine Intelligence, 1989.

7. A. K. Agrawala and Y. N. Singhb, Evaluation of face recognition methods in unconstrained environments, Procedia Computer Science, International Conference on Intelligent Computing, Communication & Convergence, 2015.

8. http://www.expansion.com/economia-digital/innovacion/2017/02/14/589c679d468aeb243d8b45be.html

9. https://www.bbva.com/es/tres-pilares-revolucion-digital-datos-talento-innovacion/

17. Fuente de las imágenes

1. https://es.statista.com/grafico/11129/en-dos-anos-todos-los-smartphones-estaran-equipados-con-tecnologia-biometrica/
2. http://totalsapiens.com/total/bases-de-datos-biometricos-con-un-alto-nivel-de-peligro/
3. http://www.m2sys.com/blog/guest-blog-posts/biometric-smartgates-improve-border-security-airport-efficiency/attachment/gat_south_egates/
4. http://www.ite.educacion.es/formacion/enred/materiales_en_pruebas_2011/gimp_noviembre_11/m2/el_pxel.html
5. http://comp3204.ecs.soton.ac.uk/cw/c6223_coursework2.html
6. https://courses.cs.washington.edu/courses/cse576/13sp/projects/project3/
7. https://github.com/wihoho/FaceRecognition
8. https://www.ini.rub.de/PEOPLE/wiskott/Projects/EGMFaceRecognition.html
9. http://ebooks.bharathuniv.ac.in/gdlc1/gdlc4/Arts_and_Science_Books/science/physical_sciences/Computer%20and%20Information%20Science/Artificial%20Intelligence/Books/Reviews%20Refinements%20and%20New%20Ideas%20in%20Face%20Recognition.pdf
10. https://www.theseus.fi/bitstream/handle/10024/132808/Delbiaggio_Nicolas.pdf?sequence=1
11. https://www.semanticscholar.org/paper/Towards-a-Video-Annotation-System-using-Face-Lindstr%C3%B6m/5031d87d6e5282525def0434ba02d2df390568a2/figure/20
12. https://upload.wikimedia.org/wikipedia/commons/thumb/6/67/Distance_Formula.svg/250px-Distance_Formula.svg.png
13. https://core.ac.uk/download/pdf/82472891.pdf
14. https://stats.stackexchange.com/questions/94463/what-are-the-advantages-of-kernel-pca-over-standard-pca
15. http://www.ine.es/ss/Satellite?blobcol=urldata&blobheader=application%2Fpdf&blobheadername1=Content-Disposition&blobheadervalue1=attachment%3B+filename%3D15%2F737%2F119_1.pdf&blobkey=urldata&blobtable=MungoBlobs&blobwhere=15%2F737%2F119_1.pdf&ssbinary=true
16. https://csperson.kku.ac.th/chakchai/faceDBweb/face_dataset.html
17. https://www.semanticscholar.org/paper/From-Few-to-Many%3A-Illumination-Cone-Models-for-Face-Georghiades-Belhumeur/18c72175ddbb7d5956d180b65a96005c100f6014/figure/2
18. http://terminalcoders.blogspot.com/2017/03/at-face-database-in-png.html
19. https://www.researchgate.net/figure/Face-examples-from-FERET-database_fig11_255634364
20. https://es.wikipedia.org/wiki/An%C3%A1lisis_de_componentes_principales
21. http://www.scielo.org.mx/pdf/cys/v16n2/v16n2a3.pdf
22. https://sites.google.com/site/portafolionetbeans/que-es-netbeans

18. Anexo I: Instalar las librerias openCV en NetBeans (Ubuntu)

Este apartado se separará en dos partes. En primer lugar se explicará la instalación de las librerías openCv en el entorno Ubuntu y después, como poder utilizarlas con el programa netbeans.

Figura 42: Logo OpenCV

18.1. Instalar OpenCV en Ubuntu

Todo el proceso se ejecutará mediante comandos en el cmd del usuario. En primer lugar, para que todo funcione correctamente la versión de Ubuntu debe estar actualizada. Esto lo realizaremos a partir de los siguientes comandos.

18.1.1. Instalación con apt-get

```
sudo apt-get update

sudo apt-get upgrade
```

Una vez actualizado el sistema operativo, instalamos la biblioteca con:

```
sudo apt-get install libopencv-dev
```

Por último, podemos bajarnos la documentación y algunos ejemplos:

```
sudo apt-get install opencv-doc
```

18.2. Instalación manual de las OpenCV

Si no tenemos versiones tan avanzadas de ubuntu debemos instalarl las librerias nosotros manualmente mediante los siguientes comandos:

```
sudo apt-get install build-essential

sudo apt-get install cmake
```

```
sudo apt-get install pkg-config
```

Esto nos permite instalar las herramientas imprescindibles como son build-essential, cmake para la automatización de código y pkg-config para manejar los flags del compilador.

Ahora vamos con los paquetes que se utilizan para leer y escribir imágenes o video en distintos formatos y mostrarlos en ventanas. No es necesario instalarlos, y de no hacerlo OpenCV usará las versiones por defecto que vienen con la biblioteca, pero en el caso de que algún modulo no funcione puede deberse a que no tiene el codec para leer el video o algún problema relacionado con la falta de alguno de estos paquetes.

```
sudo apt-get install libjpeg-dev libtiff4-dev libjasper-dev libopenexr-dev li
btbb-dev libeigen2-dev yasm libfaac-dev libopencore-amrnb-dev libopencore-amr
wb-dev libtheora-dev libvorbis-dev libxvidcore-dev libx264-dev libqt4-dev lib
qt4-opengl-dev sphinx-common texlive-latex-extra libv4l-dev libdc1394-22-dev
libavcodec-dev libavformat-dev libswscale-dev
```

18.2.1. Código fuente

Una vez tengamos todo listo se procede a descargar el código de OpenCV. Se pueden descargar tanto desde Sourceforge como desde GitHub.

GitHub:

```
git clone https://github.com/Itseez/opencv.git

cd opencv
```

Sourceforge:

```
wget http://downloads.sourceforge.net/project/opencvlibrary/opencv-unix/2.4.5
/opencv-2.4.5.tar.gz

tar xzvf opencv-2.4.5.tar.gz

cd opencv-2.4.5
```

18.2.2. Compilación

A continuación, se genera el makefile que define que partes de OpenCV necesitan ser compiladas. Recuerda que cmake es:

```
cmake [<parametros opcionales>] <ruta al código fuente>;
```

Creamos el directorio build y entramos:

```
mkdir build

cd build
```

Después todo lo que se vaya a trabajar de esta biblioteca y se genera el makefile.

```
cmake -D CMAKE_BUILD_TYPE=RELEASE -D CMAKE_INSTALL_PREFIX=/usr/local ..
```

Es muy importante comprobar que este apartado no de error. En caso contrario se debe retroceder e instalar los paquetes extra y ejecutar el cmake de nuevo.

A continuación se lanza el make para compilar:

```
make
```

Por último para instalar la biblioteca:

```
sudo make install
```

18.2.3. Configuración

Una vez instaladas las librerías debemos configurar la ruta de OpenCV. Esto se especifica creando un archivo "opencv.conf" en el directorio "/etc/ld.so.conf.d/". En este archivo añadiremos en una nueva línea /usr/local/lib.

```
sudo gedit /etc/ld.so.conf.d/opencv.conf

# y añadimos /usr/local/lib
```

Y configuramos la biblioteca ejecutando:

```
sudo ldconfig -v
```

Esto crea los enlaces y el cache necesarios para las bibliotecas más recientes que se encuentren en los directorios especificados en los archivos de /etc/ld.so.conf/(en este caso OpenCV) Otra opción es exportar la ruta a la variable de entorno LD_LIBRARY_PATH y enlazar dinamicamente con ldconfig.

```
export LD_LIBRARY_PATH=~/usr/local/lib/:$LD_LIBRARY_PATH

sudo ldconfig
```

y en el caso de que no hayamos instalado OpenCV en "/usr/local/lib", que se encuentre en por ejemplo "~/OpenCV-2.4.5/build"

```
export LD_LIBRARY_PATH=~/opencv-2.4.5/build/:$LD_LIBRARY_PATH

sudo ldconfig
```

Como último paso queda establecer la variable de entorno pkg_config_path. Asi que se edita "bash.bashrc":

```
sudo gedit /etc/bash.bashrc
```

y se añade:

```
export PKG_CONFIG_PATH=$PKG_CONFIG_PATH:/usr/local/lib/pkgconfig
```

Se cierra la consola y se reinicia el ordenador. De otra maner la configuración que acabamos de añadir no se cargará.

18.3. Configuración del NetBeans

En primer lugar debemos abrir el Netbeans en Ubuntu. En caso de no tenerlo en instalado se puede realizar a través de la pagina web: http://netbeans.org/downloads/start.html?platform=linux&lang=en&option=cpp

Figura 43: Imagen al ejecutar NetBeans

A continuación, debemos crear un proyecto en C/C++ y se debe elegir C++ Application:

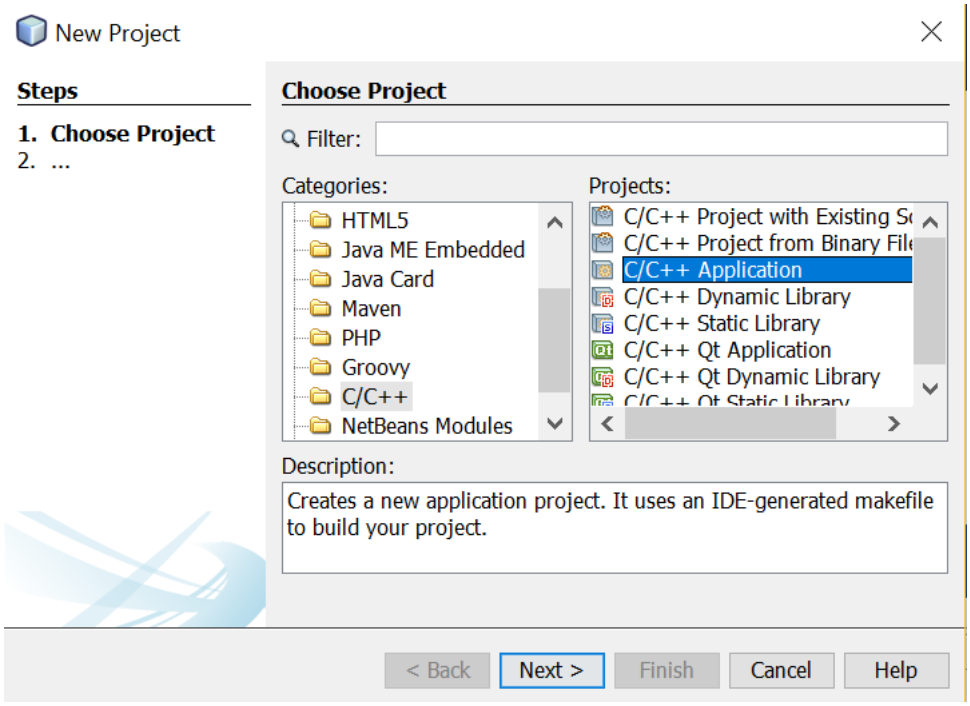

Figura 44: Crear nuevo proyecto en NetBeans

Una vez creado el proyecto, se hace click con el botón derecho en el proyecto y pulsamos propiedades/C++ Compiler. Y en el apartado de include directories añadimos el directorio "usr/local/include/opencv2"

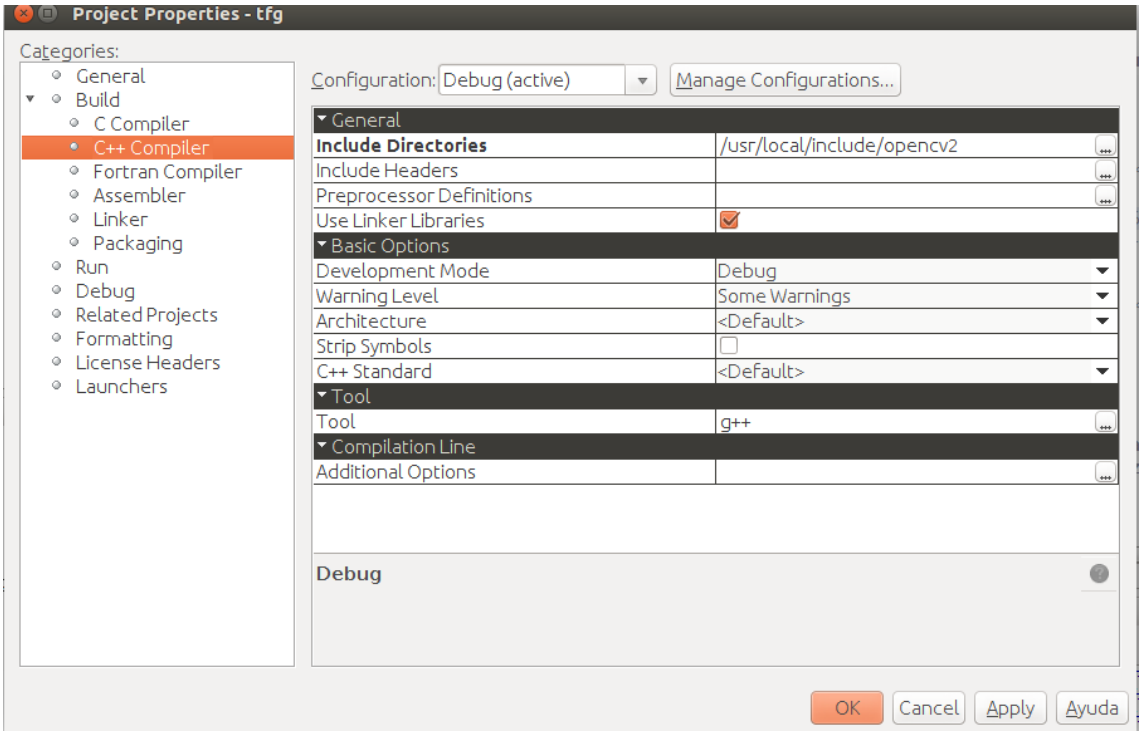

Figura 45: C++ Compiler properties

Ahora, pulsamos Linker y se añade el mismo path al apartado additional library directories y en el apartado libraries, hacemos click en add pkgConfig library. Nos saldrá una serie de librerías en la que debemos buscar opencv. Pinchamos, la añadimos y este debería ser el resultado final.

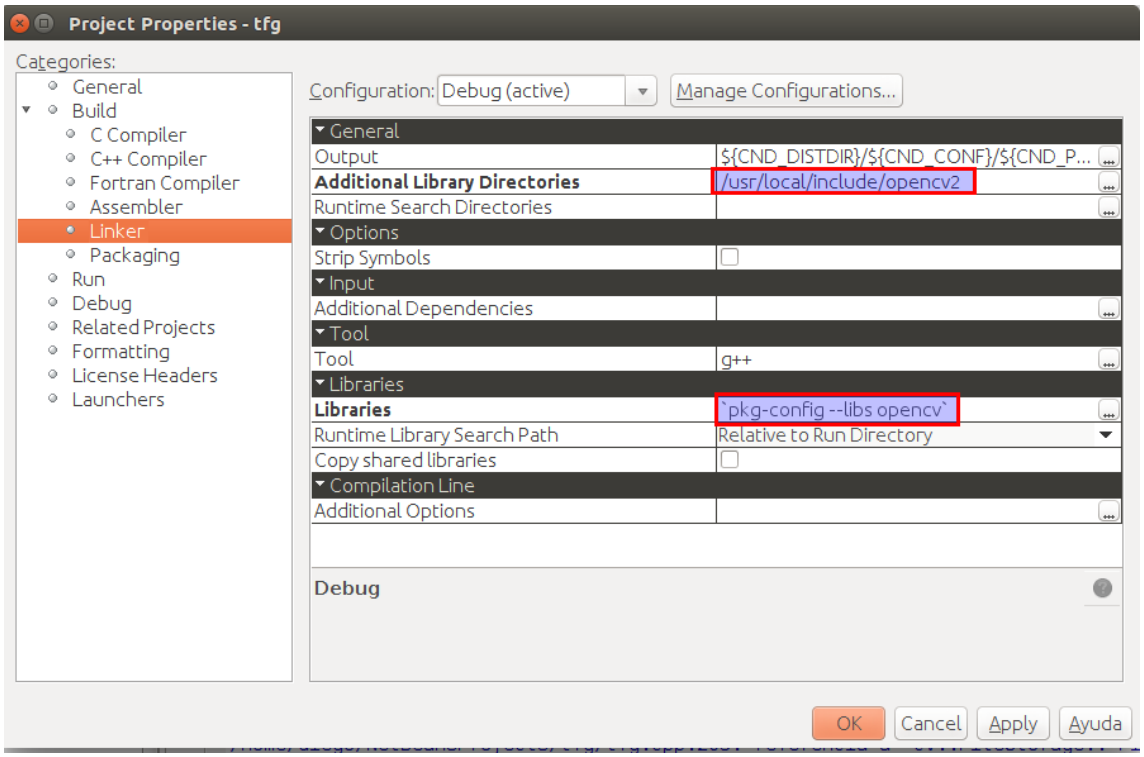

Figura 46: Configuración de las propiedades del Linker

Una vez realizado el proceso NetBeans estará configurado correctamente para utilizar las librerías OpenCV sin problema.