

UNIVERSIDAD POLITECNICA DE VALENCIA

ESCUELA POLITECNICA SUPERIOR DE GANDIA

Grado en Ing. Sist. de Telecom., Sonido e Imagen



UNIVERSIDAD
POLITECNICA
DE VALENCIA



ESCUELA POLITECNICA
SUPERIOR DE GANDIA

**“SISTEMA DE IDENTIFICACIÓN DE
PERSONAS MEDIANTE
RECONOCIMIENTO FACIAL APLICADO A
VIDEOVIGILANCIA”**

TRABAJO FINAL DE GRADO

Autor/a: Nerea Cañego Navio

Tutor/a: Jose Pelegrí Sebastià

Co-tutor/a: Jose Ignacio Herranz Herruzo

GANDIA, 2017

RESUMEN

El trabajo de fin de grado desarrollado consiste en el reconocimiento de personas de forma facial- El objetivo principal reside en desarrollar un software capaz de reconocer y realizar un el seguimiento de las personas que acceden a un lugar como puede ser un cajero automático, un establecimiento restringido o el lugar de trabajo.

Se busca que el software responda correctamente en tiempo real además de ser fácil de usar para el usuario. Se pretende que el usuario sea capaz de ir modificando la base de datos interna del software de forma gráfica cuando lo necesite sin necesidad de tener conocimientos técnicos sobre él.

ABSTRACT

This work consist of the recognition of people in a facial form. The main idea of the work is to be able to realize a software capable of recognizing and trcking people who access a place such as an automatic bank, restricted place or the workplace.

The idea is that the software responds correctly in real time in addition to being easy to use for the user.It is intended that the user is able to modify the internal database of the software graphically when needed without having technical knowledge about he.

PALABRAS CLAVE

Detección facial, reconocimiento facial, Fisherfaces, Eigenfaces, Vila-Jones.

KEY WORDS

Facial detection, face recognition, Fisherfaces, Eigenfaces, Viola-Jones.

ÍNDICE

1.	INTRDUCCIÓN	5
1.1	PRESENTACIÓN.....	5
1.2	OBJETIVOS.....	6
1.3	MOTIVACIÓN.....	6
1.4	ETAPAS DEL PROYECTO	7
1.4.1	PRIMERA ETAPA	7
1.4.2	SEGUNDA ETAPA	7
1.4.3	TERCERA ETAPA.....	7
1.4.4	CUARTA ETAPA	8
1.4.5	QUINTA ETAPA	8
1.4.6	SEXTA ETAPA	8
1.4.7	SEPTIMA ETAPA.....	8
2.	MATERIAL.....	9
3.	ESTADO DEL ARTE	11
3.1	DETECCIÓN Y RECONOCIMIENTO FACIAL	11
3.1.1	DETECCIÓN FACIAL.....	12
3.1.1.1	VIOLA-JONES.....	13
3.1.2	TÉCNICAS DE RECONOCIMIENTO FACIAL	15
3.1.2.1	PCA (PRINCIPAL COMPONENT ANALYSIS).....	16
3.1.2.2	LDA (LINEAR DISCRIMINANT ANALYSIS).....	17
3.1.3	SISTEMAS DE RECONOCIMIENTO FACIAL	18
3.1.3.1	EIGENFACES.....	18
3.1.3.1.1	FISHESFACES	20
4.1.1	MÉTODOS DE DECISIÓN.....	22
5.	CÓDIGO GENERADO PARA LA APLICACIÓN	23
5.1	PRIMERA FASE	23
5.2	SEGUNDA FASE.....	26
5.3	TERCERA FASE	33
5.4	CUARTA FASE.....	35
5.5	QUINTA FASE.....	36
5.6	SEXTA FASE.....	37
6.	BASE DE DATOS GENERADA	37

7.	RESULTADOS	40
7.1	RESULTADOS DOS IMÁGENES POR CLASE	41
7.2	RESULTADOS TRES IMÁGENES POR CLASE	41
7.3	RESULTADOS CUATRO IMÁGENES POR CLASE.....	42
8.	CONCLUSIONES	43
9.	TRABAJO FUTURO	43
10.	BIBLIOGRAFÍA.....	44

TABLA DE FIGURAS

Figura 1:	Cámara AXIS P1534 conectada a la fuente de alimentación	10
Figura 2:	Detección de la cámara una vez se ha reseteado para proceder a su configuración	10
Figura 3:	Muestra de la captura de la cámara AXIS P1534	10
Figura 4:	Configuración de la cámara AXIS P1534	11
Figura 5:	Esquema pasos a seguir para la detección y el reconocimiento facial	11
Figura 6:	Tipo de descriptores utilizador por Haar	13
Figura 7:	Algoritmo de Haar modificado para ser utilizado en Viola-Jones	14
Figura 8:	Esquema algoritmo Viola-Jones	15
Figura 9:	Esquema técnicas de reconocimiento facial	15
Figura 10:	Proyección LDA	18
Figura 11:	Método K-Vecinos	22
Figura 12:	.VI adquisición de imagen	23
Figura 13:	Icono subvi adquisición de imagen	24
Figura 14:	.VI procesado de imagen	24
Figura 15:	Icono subvi procesado de imagen	24
Figura 16:	Subvi que extrae el nombre de la persona.	24
Figura 17:	.VI principal	25
Figura 18:	Parte de la adquisición de la imagen en el .vi principal	25
Figura 19:	Parte del procesado de imagen en el .vi principal	25
Figura 20:	Método para incorporar el subvi que extraerá el nombre y la hora	26
Figura 21:	Formato en el que han de estar las imágenes de la Base de Datos para tratar con ellas	27
Figura 22:	Código Matlab realizado para leer las imágenes que el usuario almacene en la Base de Datos	28
Figura 23:	Código Matlab para la extracción de las Fisherfaces de las imagenes de la Base de Datos	29
Figura 24:	Función PCA aplicada para la extracción de las Fisherfaces	30
Figura 25:	Función LDA aplicada para extracción de las Fisherfaces	31
Figura 26:	Fisherfaces de la base de datos	33

Figura 27: Código Matlab para la extracción de las imágenes captadas por la imagen y detección facial.	34
Figura 28: Extracción de las características Fisherfaces de las imágenes de entrada	35
Figura 29: Código Matlab del cálculo de la distancia Euclídea y clasificador K-Vecinos	36
Figura 30: .Vi de lectura de la clase para mostrarlo en la interfaz	36
Figura 31: Interfaz programa	37
Figura 32: Base de Datos generada en la aplicación	39
Figura 33: Imágenes de testeo utilizadas para las pruebas	41
Figura 34: Tabla de resultado con dos imágenes por clase	41
Figura 35: Tabla de resultado con tres imágenes por clase	42
Figura 36: Tabla de resultado con cuatro imágenes por clase	42

1. INTRODUCCIÓN

1.1 PRESENTACIÓN

El reconocimiento facial es una técnica muy usada en la actualidad para poder tener un control de acceso a diversas áreas restringidas. Así como la búsqueda de personas.

Para poder llegar al reconocimiento facial, previamente se realiza una detección facial que también es muy utilizada en diversas aplicaciones, como pueden ser videojuegos controlados por movimiento, animaciones, realidad aumentada o cámaras fotográficas.

Con este proyecto se realizará dicho reconocimiento facial, de uno o varios usuarios de forma simultánea, a través de un cámara de video vigilancia en tiempo real. Posteriormente se aplicará en control de acceso, seguridad o control de asistencia laboral.

El proyecto se centrará primero en la detección facial de la secuencia de imágenes grabadas por la cámara de video vigilancia. Se detectará si existe una o más caras en cada una de las imágenes que se captará por la cámara.

Una vez se ha producido la detección facial se procederá al reconocimiento facial. Primero se creará una Base de Datos con varias personas. Si la persona que pasa por delante de la cámara de vídeo vigilancia se encuentra en esta base de datos se dará una descripción de esta persona, es decir, nombre de la persona y hora a la que ha pasado por delante de ella.

Por otro lado, si la persona que pasa por delante de la cámara no se encuentra en esta Base de Datos se hará referencia a que una persona ha pasado por delante de la cámara pero que se tratará de una persona desconocida.

Además de realizar esta parte de reconocimiento de personas de forma facial, con este proyecto se quiere crear una interfaz de forma interactiva y sencilla para el usuario.

Por último, al usuario se le dará la posibilidad de modificar la Base de Datos para tener capacidad de elección sobre las personas a reconocer. El usuario podrá almacenar fotografías de nuevos miembros o eliminar aquellas que ya no desee que sean reconocidas.

1.2 OBJETIVOS

El objetivo principal del proyecto a nivel personal es poder aumentar el conocimiento sobre la Inteligencia Artificial, más concreto sobre el reconocimiento aplicado a personas..

Por otro lado, se pretende también asentar conocimientos sobre Labview y Matlab dirigidos a este ámbito y aportar conocimiento tanto teórico como práctico sobre las formas de entrenar algoritmos de detección.

Además se pretende ampliar conocimiento sobre bases de datos, desde conocer bases de datos existentes hasta su propia creación e implementación.

Otro objetivo planteado en este proyecto es conocer y aplicar formas de entrenamiento de diferentes algoritmos.

En cuanto al proyecto, el objetivo principal es conseguir un error aceptable, al ser posible mayor del 70% en cuanto al reconocimiento facial.

Otro objetivo es adquirir un software con una interfaz orientada y fácil hacia el usuario y la posibilidad de que la base de datos puede ser modificada por el usuario.

1.3 MOTIVACIÓN

La motivación principal de este proyecto vino a partir de una asignatura cursada en tercero del Grado en Ingeniería de Sistemas de Telecomunicación, Imagen y Sonido, llamada Instrumentación Avanzada donde después de aprender a lo largo de la asignatura el uso de Labview se decidió hacer un proyecto final para dicha asignatura.

La idea del reconocimiento vino aportada por una 'Brain Storming' que hicimos todos los alumnos de la asignatura, de ahí se decidió realizar un reconocimiento de género y emociones que llevé a cabo en el último periodo de la asignatura junto a dos compañeras más de la asignatura.

A partir de ahí, se nos despertó la curiosidad sobre la detección y el reconocimiento facial, con la idea de este trabajo salió también la opción de este proyecto que es la detección de personas para introducir en el área de la seguridad. Decidimos repartir entre dos los diferentes trabajos posibles, que a pesar de ser diferentes tipos de detección sí podrían complementarse para uso futuro la unión de ambos proyectos.

1.4 ETAPAS DEL PROYECTO

Para este proyecto se han necesitado ocho etapas diferentes. En cada una de estas etapas se ha hecho uso de unos conocimientos, materiales y técnicas distintas.

1.4.1 PRIMERA ETAPA

En la primera etapa se ha hecho una búsqueda amplia sobre diferentes técnicas de detección facial, técnicas de reconocimiento facial, descriptores, formas de entrenar códigos, bases de datos, tratamiento de imagen y cámaras posibles a utilizar. También se ha ampliado el conocimientos que se tenían sobre los softwares que se van a utilizar en el proyecto..

A esta etapa se le dedicó una duración de un mes, pero aunque haya sido la primera etapa, se puede considerar que ha sido una tarea transversal dado que se ha ido ampliando dicha información a lo largo de todo el trabajo para mejorar las decisiones a tomar.

1.4.2 SEGUNDA ETAPA

En la segunda etapa se ha procedido al control de la cámara de video vigilancia que se ha utilizado. Se buscó información sobre cómo utilizarla, resetearla, conectar la a la red y acceder a sus configuraciones.

Esta fase se ha llevado a cabo en un periodo de tres días desde que se ha tenido la posesión de la cámara.

1.4.3 TERCERA ETAPA

La tercera etapa del proyecto se ha centrado en la conexión de la cámara y la captación de las imágenes a través de Labview. Para ello se ha dedicado un día en la configuración de la cámara para Labview, y unas tres semanas en conseguir el control de la cámara con Labview así como la captación de las imágenes y el almacenamiento de las mismas mediante dicho Software.

1.4.4 CUARTA ETAPA

Para la cuarta etapa se ha hecho uso de aproximadamente una semana. Este tiempo se ha empleado en el procesamiento de las imágenes que han sido captadas para la detección facial.

Haciendo uso de Matlab se ha determinado si existe una cara en la imagen y si es así la imagen se almacena mientras que por el contrario la imagen capturada se descarta.

1.4.5 QUINTA ETAPA

En la quinta etapa se ha procedido al reconocimiento facial mediante Matlab. Se ha intentado adaptar diversos scripts de código abierto. Una vez probados hasta ocho códigos diferentes, se ha obtenido un código viable modificado personalmente. Con esta fase se ha empleado aproximadamente dos meses.

En esta fase también se ha incluido, a parte del procesamiento de la imagen para sacar las características pertinentes para el reconocimiento facial, las técnicas empleadas para reconocer si la persona se encuentra en la base de datos así como la creación de la propia base de datos.

1.4.6 SEXTA ETAPA

En la sexta etapa, que se han utilizado tres días para realizar pruebas con el código de Matlab generado para calcular la tasa de error y poder rectificar el código inicial para corregir estos errores. Con esta fase se ha hecho uso de cuatro días.

1.4.7 SEPTIMA ETAPA

En la séptima etapa, utilizando de nuevo Labview, se ha juntado la tercera, cuarta y quinta fase.

En esta etapa también se ha hecho pruebas sobre el conjunto total de la aplicación y se ha generado la interfaz final. Con esta etapa se ha dedicado un periodo de tres días.

2. MATERIAL

Para poder realizar el siguiente proyecto es necesario hacer uso de material de Software y Hardware.

En cuanto al material de Software se hará uso de Labview y Matlab de con licencia de forma académica proporcionado por la Universidad Politécnica de Valencia.

Se ha decidido utilizar estos dos sistemas de software por el conocimiento adquirido en diversas asignaturas a lo largo del Grado cursado y por la facilidad de adquisición de las licencias.

En cuanto al sistema de Hardware es necesario utilizar una cámara de video vigilancia. En este caso no se ha realizado ningún estudio comparativo sobre cámaras dado que se ha conseguido poder utilizar una cámara AXIS P1354 proporcionada por el Departamento de Comunicaciones de la Universidad Politécnica de Valencia, Campus de Gandia.

La cámara utilizada es una cámara de red fija diurna y nocturna, con la que se obtiene una calidad de vídeo con exploración progresiva en varias secuencias H.264.

Esta cámara proporcionada incluye tecnología Lightfinder que ayuda a mantener los colores en condiciones bajas en cuanto a iluminación. Algunas de las aplicaciones que se le han dado a este modelo de cámara es el de reconocimiento de matrículas utilizado por la policía. [1]

Previamente, al inicio del proyecto, ha sido necesario configurar dicha cámara. Para poder configurar la cámara, primero se ha procedido a resetearla, para ello se presiona el botón de reset que se encuentra en la parte de debajo de la cámara durante unos segundos hasta que la luz naranja de la cámara parpadea y después se mantiene encendida durante unos segundos hasta que finalmente se apaga.

Una vez se ha reiniciado, se conecta la cámara a una fuente de alimentación con aproximadamente 12-23V y a un cable de red conectado al ordenador del que se va hacer uso, se introduce en el explorador de internet el siguiente enlace: [URL:http://axis.die.gnd.upv.es](http://axis.die.gnd.upv.es) proporcionada por el técnico que corresponde a la cámara que se va a utilizar. En este enlace se introduce el usuario ROOT y nos aparecerá las configuraciones técnicas y de seguridad de la cámara.



Figura 1: Cámara AXIS P1534 conectada a la fuente de alimentación

En nuestro caso, en cuanto a seguridad se configura la cámara con usuario 'ROOT' y contraseña 'camara'. Por otro lado, para las especificaciones técnicas de la cámara sólo se ha hecho uso de los frames/s que deseamos que capture la cámara. Para este proyecto se va hacer uso de 2 frames/s. No es necesario capturar más fotogramas debido a que se estima que es lo que tardara una persona en moverse. Es conveniente no cargar imágenes innecesarias para reducir el coste computacional, con lo que se consigue un software más ágil.

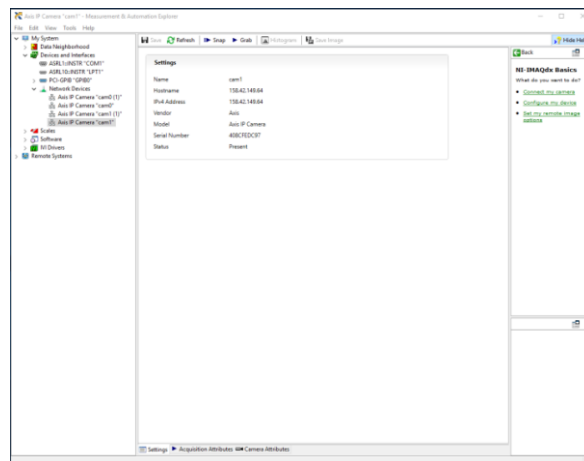


Figura 2: Detección de la cámara una vez se ha reseteado para proceder a su configuración

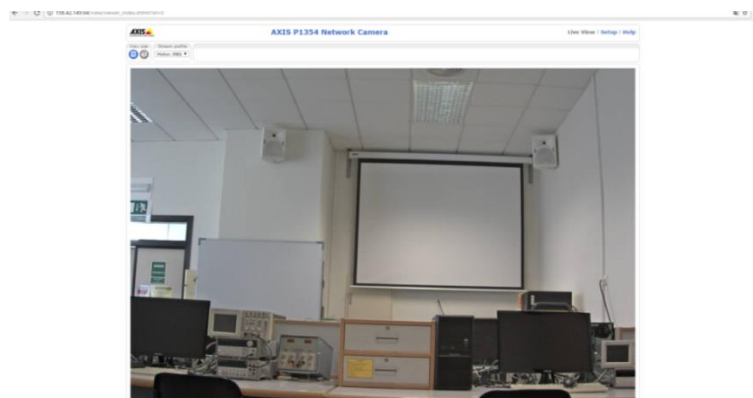


Figura 3: Muestra de la captura de la cámara AXIS P1534

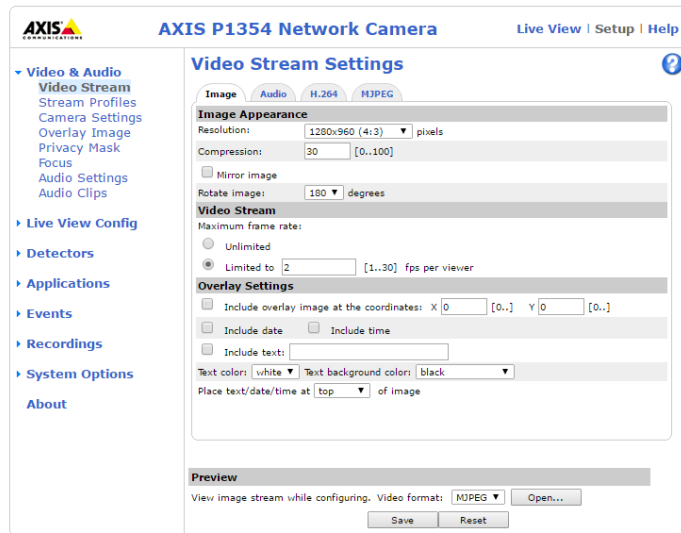


Figura 4: Configuración de la cámara AXIS P1534

Posteriormente se configura la cámara para poder utilizarla desde Labivew. Se hace uso de la aplicación IMAX utilizada para configurar el uso de equipos externos con Labview.

Normalmente en este paso, haciendo uso de IMAX la cámara será detectada automáticamente por el programa sin necesidad de configurar ningún extra si previamente se configuran los parámetros necesarios directamente en la propia cámara.

3. ESTADO DEL ARTE

3.1 DETECCIÓN Y RECONOCIMIENTO FACIAL

Para poder conseguir identificar al individuo previamente es necesario poder identificar la cara de una persona, es decir, es necesario realizar la detección facial, a través de la imagen recibida, de todos los individuos que se encuentren en la imagen.

Para proceder a la detección y el reconocimiento facial es necesario realizar los siguientes pasos:

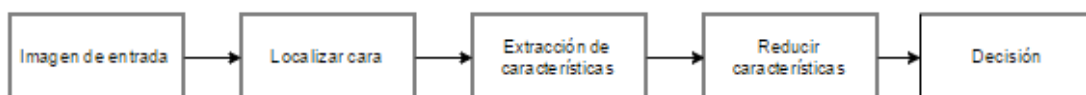


Figura 5: Esquema pasos a seguir para la detección y el reconocimiento facial

Como podemos ver en la figura 5, tenemos la imagen de entrada que en nuestro caso será cada uno de los frames capturados por la cámara de video-vigilancia.

El siguiente paso a realizar será la localización de la cara, es decir la detección facial. Será necesario procesar la imagen de entrada para saber si en ese fragmento de imagen recibido existe persona o no. En caso de que se localice una cara la imagen será almacenada y en caso contrario la imagen será descartada.

Si la imagen ha sido almacenada querrá decir que existe una cara en dicha imagen y se realizará una extracción de características de dicha cara. Estas características se pueden extraer mediante diferentes descriptores o algoritmos de los que se hablará más tarde.

Después de haber obtenido las diferentes características de la cara, si es necesario se realiza una reducción de estas para facilitar la clasificación

Por último, se realiza una decisión sobre la cara. Se decidirá si la cara pertenece o no a alguna persona existente en la base de datos. Esta decisión se puede tomar haciendo uso de diferentes técnicas que serán explicadas posteriormente.

3.1.1 DETECCIÓN FACIAL

La detección facial, como se ha explicado en el apartado anterior consiste en localizar si existe la cara en una imagen de entrada. Para poder localizar la cara se pueden utilizar diversos métodos con diferentes características.[2]

Para realizar esta tarea se puede utilizar métodos basados en las características geométricas de la cara. Estos métodos consisten en utilizar el cálculo de distancias geométricas y otras características que se combinan en un vector de características y se reduce quedándose con las más determinantes.

Existen también los métodos de plantilla que mejoran respecto al anterior dado que guarda en el vector todas las características de una cara en lugar de las más relevantes por lo que a la hora de testear se tiene más información.

Por otro lado, actualmente podemos encontrar diferentes algoritmos ya testeados previamente como puede ser el Algoritmo 'top-down'. Este algoritmo se basa en reglas que detectan caras, por ejemplo, ubicando los ojos por simetría con respecto al centro de la cara o considerando que la cara ha de ser de color uniforme.

También encontramos, en contraste con estos algoritmos, los procedimientos 'up-down'. Estos buscan encontrar características que indiquen ciertas regiones de la cara como podrían ser colores, bordes o texturas. Sin embargo estos algoritmos son sensibles a ruido u oclusión en las imágenes.

Los algoritmos denominados 'Template-matching' comienzan construyendo una plantilla predefinida a partir de una imagen de una cara establecida o el borde de una cara. Posteriormente aplica la correlación de esta plantilla a las imágenes de entrada y en las regiones que se obtenga una mayor respuesta se considerará que existe una cara, el problema de este tipo de algoritmos es que no es útil cuando nos

encontramos antes variaciones de escala u orientaciones diferentes.[3]

Uno de los algoritmos más nombrados en este ámbito es el algoritmo de Viola-Jones, del que se va hacer uso en este proyecto ya que es un algoritmo muy popular implementado en diversos lenguajes de programación con una alta tasa de acierto y con un coste computacional bajo.

3.1.1.1 VIOLA-JONES

Viola-Jones es un algoritmo capaz de detectar objetos y caras en tiempo real. Este algoritmo fue propuesto por Paul Viola y Michael Jones en 2001 [4][5][6].

La característica principal que consigue hacer que este algoritmo sea uno de los más usados son la robustez dado que tiene un tasa de detección alta, tasa de verdaderos positivos muy alta y tasa de falsos positivos muy baja.

Además, con el uso de este algoritmo se pueden realizar reconocimientos faciales en tiempo real para aplicaciones con un mínimo de 2 fotogramas por segundo.

Este algoritmo aplica un clasificador en cascada, entrenado mediante un método de Machine Learning, con concreto Adaboost. Los descriptores empleados son las características Haar, muy adecuadas para la detección facial.

Las características Haar utilizadas por este algoritmo son cuatro tipos de características diferentes que podemos observar en la figura 6.

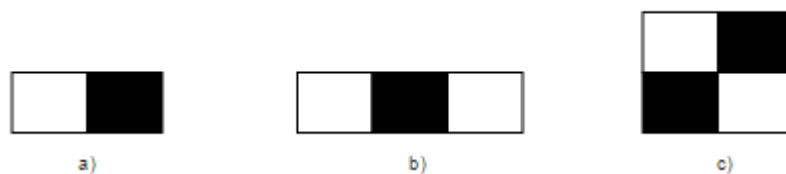


Figura 6: Tipo de descriptores utilizador por Haar

Cada una de las características se superponen sobre la imagen en todas las posiciones y tamaños posibles, el valor de la característica se obtiene restando a la suma de todos los píxeles la zona negra de la característica.

En a) se puede ver un descriptor de dos rectángulos. Con este descriptor se calcula la diferencia entre la suma de los píxeles que se encuentran en el lado negro y el lado blanco de la imagen, estos dos rectángulos tienen el mismo tamaño y forma.

En b) se encuentra un descriptor de tres rectángulos, en este caso se calcula la suma de los píxeles entre los dos rectángulos exteriores y se restan la suma de los píxeles que se encuentran en el rectángulo central.

En c) hay un descriptor de cuatro rectángulos, haciendo uso de este descriptor se

calcula la diferencia de los píxeles que se encuentran en los pares diagonales de los rectángulos.

Con estas características se obtiene una imagen integral para la posición (x,y) de la imagen original. Haciendo uso de este algoritmo se crea la imagen integral con un solo barrido de la imagen original.

La utilidad de esta imagen integral es calcular las características Haar de forma muy eficiente.

Una vez se ha obtenido esta imagen integral se hace uso del algoritmo Adaboost de Freund y Schapire, algoritmo utilizado para el aprendizaje de Vila-Jones.

En la siguiente figura, figura 7 obtenida en [5] podemos ver el algoritmo final que utiliza Viola-Jones con Adaboost.

- Dadas las imágenes de muestra $(x_1, y_1), \dots, (x_n, y_n)$ donde $y_i = 0, 1$ para muestras negativas y positivas, respectivamente.
- Inicializar los pesos $w_{1,i} = \frac{1}{2m} \cdot \frac{1}{2l}$ para $y_i = 0, 1$ respectivamente, donde m y l son el número de muestras negativas y positivas, respectivamente.
- Para $t = 1, \dots, T$:
 1. Normalizar los pesos,

$$w_{t,i} \leftarrow \frac{w_{t,i}}{\sum_{j=1}^n w_{t,j}}$$
 De tal manera que w_t sea una distribución de probabilidad.
 2. Para cada característica j , entrenar un clasificador h_j que esté restringido a utilizar únicamente una característica. El error es evaluado con respecto a w_t , $\epsilon_j = \sum_i w_t |h_j(x_i) - y_i|$.
 3. Elegir el clasificador h_t , que tenga el menor error ϵ_t .
 4. Actualizar los pesos:

$$w_{t+1,i} = w_{t,i} \beta_t^{1-\epsilon_i}$$
 Donde $\epsilon_i = 0$ Si la muestra x_i es clasificada correctamente, $\epsilon_i = 1$ en caso contrario, y $\beta_t = \frac{\epsilon_t}{1-\epsilon_t}$.
- El clasificador fuerte sería:

$$h(x) = f(x) = \begin{cases} 1, & \sum_{t=1}^T \alpha_t h_t(x) \geq \frac{1}{2} \sum_{t=1}^T \alpha_t \\ 0, & \text{En caso contrario} \end{cases}$$
 Donde $\alpha_t = \log \frac{1}{\beta_t}$

Figura 7: Algoritmo de Haar modificado para ser utilizado en Viola-Jones

En la figura 7 se puede ver como las muestras obtenidas con la parte oscura se tratan de forma independientes a las obtenidas con la parte blanca del descriptor, con lo que se podría utilizar un número diferente de muestras positivas y negativas en el set de entrenamiento.

Como se ha hecho referencia al inicio de este punto, en el último paso en el algoritmo de Viola-Jones se hace uso de varios clasificadores en cascada. Esto se realiza para aumentar la velocidad del algoritmo lo que nos proporciona una buena característica para ser usado en tiempo real.

Estos clasificadores en cascada van siendo cada vez más estrictos conforme avanza la imagen sobre ellos, es decir, el clasificador dos será más estricto que el clasificador uno y así sucesivamente.

En cada clasificador se segmenta la imagen y se va identificando si hay cara o no.

En el caso de haber cara, este fragmento será pasado al siguiente clasificador. En la figura 8 podemos ver un esquema del funcionamiento de estos clasificadores[5].

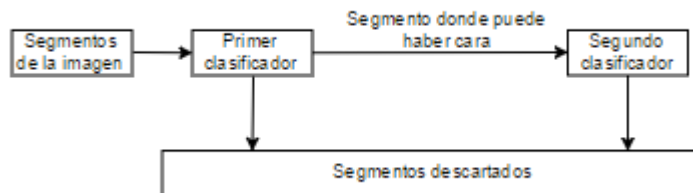


Figura 8: Esquema algoritmo Viola-Jones

3.1.2 TÉCNICAS DE RECONOCIMIENTO FACIAL

El siguiente paso para el desarrollo de este proyecto es el reconocimiento de personas de forma facial. Existen diversas técnicas sobre reconocimiento que se pueden aplicar a esta área específica.

Existen técnicas basadas en la apariencia que se centran en analizar las características de la imagen como la textura, técnicas basadas en modelos que consisten en extraer las características de forma de las imágenes.

Para este proyecto se va hacer uso de las técnicas basadas en la apariencia dado que nos aportarán información más característica que las técnicas basadas en apariencia que suelen utilizarse para realizar modelos en 2D y 3D a partir de imágenes[4].

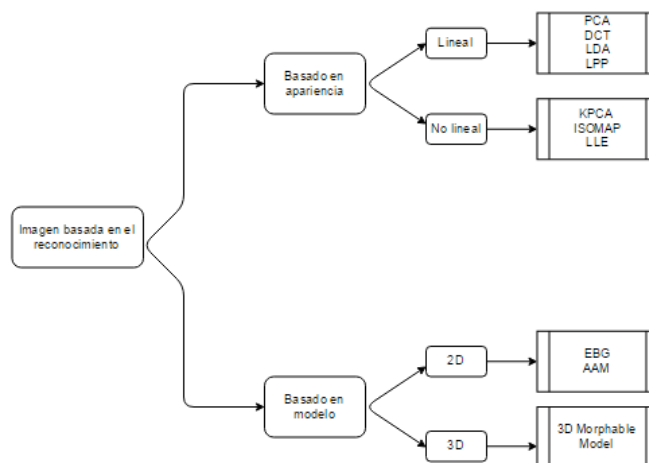


Figura 9: Esquema técnicas de reconocimiento facial

Cómo podemos ver en la figura, dentro de las técnicas basadas en la apariencia se encuentran las técnicas lineales y las técnicas no lineales.

Las técnicas lineales consisten en realizar una transformación de las imágenes de entrada a un nuevo subespacio a través de una matriz de proyección. Cada una de las técnicas lineales utilizan una representación propia en un espacio vectorial de alta dimensionalidad.

Para obtener la representación se utilizan los coeficientes como la representación de cada una de las imágenes.

Las transformaciones lineales conseguidas del vector imagen original cumplen con la siguiente característica:

$$y = W^T * X$$

Donde y es el vector de características de dimensión, W es la matriz de transformación que se utiliza para cada uno de las técnicas y X es la imagen de entrada.

3.1.2.1 PCA (PRINCIPAL COMPONENT ANALYSIS)

PCA es una técnica de detección de tipo lineal que permite reducir la dimensionalidad de un conjunto de datos. Esta técnica consiste en generar un pequeño número de variables incorreladas a partir de un número de variables correladas.

El objetivo de esta técnica consiste en representar una imagen en términos de un sistema de coordenadas óptimo reduciendo el número final de componentes. Con PCA se consigue encontrar los vectores que mejor representan la distribución que existe en un grupo de imágenes.

Para el uso de esta técnica previamente se crea una matriz de proyección. La creación de esta matriz viene a partir de entrenar PCA con N imágenes, estas imágenes deben ser de diferentes personas y a su vez con diferentes ángulos, iluminación y gestos[4][6]

Para calcular la matriz de proyección se realizan los siguientes pasos:

1. Cálculo de la media del vector:

$$\mu = \sum_{i=1}^N X_i$$

2. Estimación de la matriz de covarianza:

$$S_T = \sum_{i=1}^N (X_i - \mu) \cdot (X_i - \mu)^T$$

3. Cálculo de los autovectores y autovalores de S_T y generar W . Donde las columnas de W son los autovectores correspondientes a los autovalores más significativos.

$$S_T \rightarrow W = (e_1, e_2 \dots e_M)$$

En cuanto al reconocimiento, para poder aplicar esta técnica se necesitan realizar las siguientes fases cuando se obtiene una nueva imagen de test en el entrenamiento del sistema.

1. La nueva imagen de test se redimensiona al mismo tamaño que se ha utilizado para la creación de la matriz de proyección.
2. Esta imagen se normaliza en media 0 y norma 1.
3. Eliminación de la media de la clase PCA por filas.

3.1.2.2 LDA (LINEAR DISCRIMINANT ANALYSIS)

LDA pretende convertir un problema de alta dimensionalidad en uno de baja dimensionalidad. Esta técnica se basa en aumentar la distancia existente entre clases para conseguir una mayor discriminación entre ellas[6][7].

La gran diferencia entre LDA y PCA es que esta técnica requiere de una supervisión durante el entrenamiento mientras que PCA utiliza un entrenamiento no supervisado.

Para conseguir la maximización entre clases se hace uso de la siguiente expresión:

$$J(w) = \frac{W^T \cdot S_B \cdot W}{W^T \cdot S_w \cdot W}$$

Donde S_B es la matriz de dispersión dentro de clases, S_w es la matriz de dispersión de clase que puede calcularse con las expresiones siguientes:

$$S_B = \sum_c N_c \cdot (\mu_c - x) \cdot (\mu_c - x)^T$$

$$S_W = \sum_C \sum_{i \in C} (X_i - \mu_C) \cdot (X_i - \mu_C)^T$$

$$\mu_C = \frac{1}{N_C} \cdot \sum_{i \in C} X_i$$

$$x = \frac{1}{N} \cdot \sum_i X_i = \frac{1}{N} \cdot \sum_C N_C \cdot \mu_C$$

Donde, C es el número de clases, N_C el número de casos dentro de la clase C , X_i el conjunto de muestras y μ_C el vector media de cada clase C .

A través de estas ecuaciones se obtienen la matriz de proyección con la que se podrá pasar del problema de alta dimensionalidad al de baja dimensionalidad. Cada columna de esta matriz son las bases del subespacio creado. Con estas bases se consigue una máxima discriminación entre las clases.

En la siguiente figura podemos ver de forma gráfica como proyecta las clases la técnica LDA. Se puede ver como agrupa los propios a la misma clase y separa las diferentes clases.

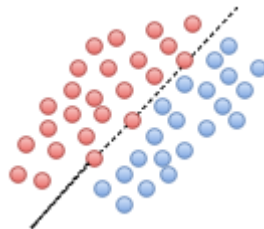


Figura 10: Proyección LDA

3.1.3 SISTEMAS DE RECONOCIMIENTO FACIAL

A lo largo del tiempo se han desarrollado diferentes tipos de reconocimiento facial, por lo tanto se va hacer uso de algún algoritmo ya testado que haya demostrados obtener buenos resultados.

Teniendo en cuenta que para la parte de código del proyecto se va hacer uso de Matlab, ya especificado anteriormente, uno de los algoritmos más adecuados, por su formulación matricial es Eigenfaces, que hace uso de PCA o Fisherfaces que hace uso la técnica de LDA.

3.1.3.1 EIGENFACES

Eigenfaces es un algoritmo desarrollado por Turk y Pentland[5]. Este método de reconocimiento facial se basa en el análisis por componentes principales. Identifica patrones en las imágenes y clasifica a cada individuo en función de las coordenadas

obtenidas en el subespacio que se forma a través de estas componentes.

Para conseguir esta clasificación previamente se necesita de un set de entrenamiento, es decir, se realiza una colección de diferentes imágenes de caras de personas que harán papel de vectores de observación. Estos vectores vendrán determinados por cada pixel de la imagen.

Para poder utilizar este algoritmo, todas las imágenes han de tener el mismo tamaño. Todas las imágenes deben tener $n = W \times H$ píxeles, donde W es el ancho de la imagen y H la altura. Por lo tanto, cada imagen es representada como un punto en un subespacio vectorial de n dimensiones.

Los desarrolladores de este algoritmo utilizaron imágenes de 256x256 píxeles, por lo que las imágenes estaban contenidas en un subespacio vectorial de 65536 dimensiones[5].

El primer paso a realizar es el cálculo de la matriz de covariancia S_x

$$S_x = \frac{1}{M} \sum_{k=1}^M (x_k - \mu) * (x_k - \mu)^T = \frac{AA^T}{M}$$

Donde, M es el número de imágenes de entrenamiento a utilizar y X son los vectores formados con los valores de los píxeles de cada imagen. El principal problema de esta formulación es que resulta una matriz $n \times n$ por lo que se obtienen demasiados valores para realizar el cálculo de los autovectores.

Por ello se utiliza el cálculo de la matriz $A^T A$ que tiene unas dimensiones $M \times M$. Por lo tanto es necesario tener una relación de estos autovectores y autovalores con la matriz S_x . Para ellos se hace uso de la siguiente definición donde los autovectores de $A^T * A$, μ_i , cumplen la siguiente ecuación:

$$\frac{1}{M} * A^T * A * \mu_i = \sigma_i * \mu_i$$

Donde σ_i es el autovalor correspondiente al autovector μ_i . A partir de esta ecuación se multiplica a la ecuación por la matriz A :

$$\frac{1}{M} * A^T * A * A \mu_i = S_x(A \mu_i) = S_x(A \sigma_i)$$

Con esto, llegamos a que el conjunto de autovectores utilizados para Eigenfaces viene por la matriz P :

$$P_{n \times M} = A_{n \times M} [\mu_1 \mu_2 \dots \mu_n]$$

El conjunto de estos autovectores resultantes se denominan Eigenfaces. Una vez se han calculado los Eigenfaces, se procede a reducir la dimensionalidad de los datos. Para poder realizar este paso, se desechan los Eigenfaces que tienen autovalores más bajos haciendo uso sólo de los que corresponden a los valores más altos.

Durante el entrenamiento con Eigenfaces para cada imagen patrón de cada persona se guardan sus componentes en el subespacio formado por Eigenfaces.

Una vez se ha terminado con la parte de entrenamiento, cuando se quiere realizar un reconocimiento, la imagen que entra se proyecta en el subespacio descrito anteriormente y se usan las coordenadas de esta proyección para tomar una decisión con los patrones que existen guardados. Para poder realizar este reconocimiento se realizan los siguientes pasos[5]:

1. Normalización de la imagen de entrada y proyección del resultado en el subespacio de Eigenfaces obtenido en el entrenamiento.

$$y' = p^T * (x' - \mu) = \begin{bmatrix} y'_1 \\ \dots \\ y'_m \end{bmatrix}$$

Donde m es el número de Eigenfaces utilizadas e y' son las coordenadas o descriptores de la imagen de entrada.

2. Cálculo de la distancia Euclídea d entre la imagen de entrada y cada patrón almacenado (ρ_l)

$$d = \|\rho_l - y'\|$$

3. Por último, si esta distancia Euclídea es inferior a un umbral establecido previamente, se decidirá que la imagen de entrada es el individuo correspondiente a ese patrón. Por el contrario, si no cumple esta restricción, se decidirá que la imagen de entrada no pertenece a la persona existente en el sistema.

$$h(x') = \begin{cases} 1 & \text{Si } d < \text{Umbral} \\ 0 & \text{Si } d > \text{Umbral} \end{cases}$$

3.1.1.1 FISHESFACES

Como se ha explicado anteriormente, existe otro método alternativo de reconocimiento de personas llamado Fisherfaces. Este sistema en diferencia, al basado en Eigenfaces, utiliza LDA como técnica para el reconocimiento facial. Con este sistema se pretende maximizar la distancia entre clases y minimizar la distancia dentro de la misma clases[5].

Este algoritmo fue propuesto por Belhumeur, que consiste en usar LDA para reducir la dimensión de los datos a $N - C$, donde N es el número de imágenes y C es el número de individuos que se encuentran en el set de entrenamiento, también llamado clases. La técnica LDA reduce la dimensionalidad hasta $C - 1$, que será el número de autovectores obtenidos distintos de cero.

Para aplicar Fisherfaces se hace uso de la matriz P formado por los vectores columna:

$$P = P_{PCA} * P_{LDA} \rightarrow P^T = P_{LDA}^T * P_{PCA}^T$$

Igual que en Eigenfaces se realiza una etapa de entrenamiento donde se almacenan los patrones de las personas que se deseen clasificar. El procedimiento es el mismo que en caso anterior donde se proyecta la imagen de entrada al subespacio formado por las Fisherfaces.

En Fisherfaces, se hace uso de la siguiente ecuación para calcular las coordenadas x e y de la imagen de entrada en el subespacio de Fisherfaces:

$$y = P^T * x = P_{LDA}^T * P_{PCA}^T * x$$

Fisherfaces necesita conocer previamente el etiquetado de cada imagen del set de entrenamiento. Esa es la diferencia fundamental respecto al método de Eigenfaces.

Como se ha mencionado anteriormente, una vez se le pasa esta información al algoritmo procede a la extracción de las características Fisherfaces de cada clase. Como se observa en la ecuación siguiente, Fisherfaces utiliza tanto la técnica PCA para reducir la dimensión de espacio de características y después se aplica LDA. Es decir, se proyecta la imagen de forma más óptima a partir de:

$$W_{opt}^T = W_{lda}^T * W_{pca}^T$$

Cuando se han obtenido los pesos de Fisherfaces de cada clase y estos son proyectados en el subespacio se emplean para concluir si la imagen de entrada pertenece a una de las clases existentes en la base de datos. Se puede hacer uso de varios sistemas de decisión que calculen la distancia entre la imagen de entrada y a cada uno de los patrones de la base de datos. Este método de decisión se tratará en el siguiente apartado.

De forma sintética un algoritmo propio de Fisherfaces seguirá las siguientes características[7]:

1. Construcción de la matriz de proyección con los vectores propios S_W y S_B correspondientes a los valores no nulos.
2. Reducción de dimensionalidad de los vectores de características.
3. Construcción de un vector de características proyectando la matriz anterior por cada una de las clases.
4. Clasificación por distancia a los vectores de características.

4.1.1 MÉTODOS DE DECISIÓN

El último paso a realizar es la decisión sobre a quién pertenece la imagen de entrada una vez se han extraído todas las características necesarias.

Existen diversos métodos de decisión aplicables, como es SVM, que se encarga de definir un algoritmo de aprendizaje supervisado para poder determinar a qué clase de las existentes pertenece una nueva muestra[8].

Un método de decisión sencillo y versátil es K-Vecinos basado en la distancia Euclídea. Este algoritmo mide la distancia entre clases mediante la distancia Euclídea y se utiliza K-Vecinos como clasificador.

La base de la distancia Euclídea viene determinada por el Teorema de Pitágoras sobre triángulos rectángulos donde la distancia euclídea es la longitud de la hipotenusa del triángulo recto conformado por cada punto y los vectores proyectados sobre los ejes directores al nivel de la hipotenusa[9].

En un espacio de N dimensiones la distancia euclídea entre dos puntos A $(a_1, a_2, a_3 \dots a_N)$ y B $(b_1, b_2, b_3 \dots b_N)$ se calcula a partir de la ecuación:

$$d(A, B) = \sqrt{\sum_{i=1}^N (b_i - a_i)^2}$$

Esta distancia se calcula a cada una de las proyecciones existentes por cada imagen patrón de cada una de las clases existentes y después se hace uso de la referencia al clasificador K-Vecinos.

El clasificador K-Vecinos se fundamenta en que el nuevo caso existente se clasificará en la clase más frecuente a la que pertenecen sus K vecinos más cercanos.

En la siguiente figura podemos ver de forma visual el método de clasificación K-Vecinos.

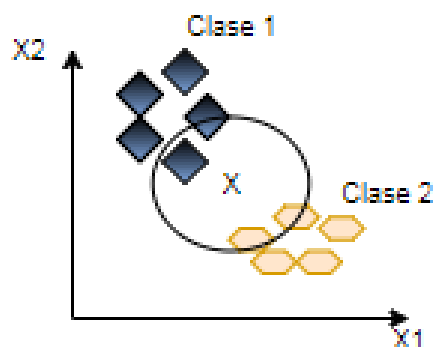


Figura 11: Método K-Vecinos

5. CÓDIGO GENERADO PARA LA APLICACIÓN

Como se ha explicado en la primera parte de la documentación. El software ha sido desarrollado con Labview para la interfaz y uso de la cámara y Matlab para el procesado de la imagen.

5.1 PRIMERA FASE

La primera fase del proyecto se ha realizado mediante Labview. Una vez conectada la cámara de video vigilancia a la fuente de alimentación y al cable de red. Mediante Labview se realiza la captación y control de la cámara.

Para realizar el control de la cámara de video vigilancia y la captación de la imagen se ha creado un .vi de Labview haciendo uso del módulo de Labview 'Vision and Motion'.

Con este módulo, utilizando 'Vision Acquisition' y 'Vision Express', seleccionando la cámara pertinente, obtenemos el control de la cámara en funcionamiento. Desde el panel frontal podremos ver la emisión de la cámara en directo y de forma interna guardaremos la imagen.

En la figura 12 podemos ver el bloque de Vision Acquisition utilizado para el control de la grabación de la cámara. En este caso los parámetros que hemos extraído de este bloque es el stop para que el usuario pueda detener la grabación, un buffer con todas las imágenes, número de imágenes así como el número de cada imagen, el ratio de frames de la captación, el path donde la imagen se almacena, la imagen de salida y los errores.

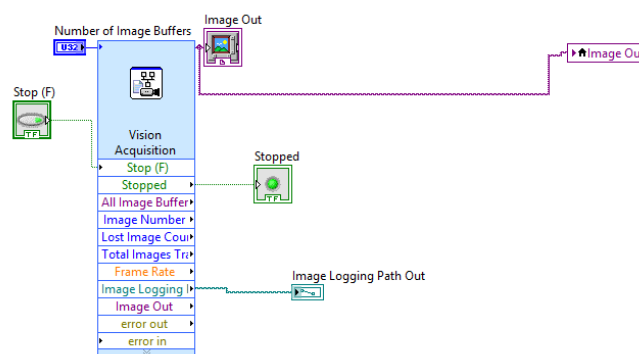


Figura 12: .VI adquisición de imagen

Dado que esto es solamente la parte de manejo y captación, que deberá ir unido con el procesado de la imagen, generamos este .vi como un subvi que será llamado desde otro .vi principal para poder hacer uso de él. En la figura 13 podemos ver la forma que tiene este subvi.

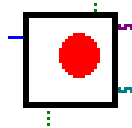


Figura 13: Icono subvi adquisición de imagen

El siguiente paso es realizar otro subvi donde se procesará la imagen. Como se ha explicado anteriormente el procesamiento de la imagen se realizará mediante Matlab, por lo que se hará uso de un Matlab Script de Labview.

En la figura 14 podemos ver la parte interna de este nuevo subvi, en el tenemos simplemente el Matlab Script donde se insertará el código Matlab.

```

numComponentsToKeep = 0; %Número de componentes a almacenar NO PUEDE SER MAYOR A c
A = face; %Matriz de imágenes de la base de datos
A = A;
N = size(A,1); % Número de imágenes totales

m_database = mean(A,1); %Media de las imágenes de la base de datos
A = A - repmat(m_database,size(A,1),1);
labels = unique(imageOwner);
c = length(labels); % Número de clases

numComponentsToKeep = min(numComponentsToKeep,(c-1));

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% PCA
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
resta = N-c;
[n_pca,d_pca] = size(A);
mu = mean(A);
Xm_pca = A - repmat(mu, size(A,1), 1);
if(n_pca>d_pca)
    C_pca = Xm_pca*Xm_pca;
    [W_pca,D_pca] = eig(C_pca);
    % sort eigenvalues and eigenvectors
    [D_pca,I_pca] = sort(diag(D_pca), 'descend');
    W_pca = W_pca(I_pca);
    % keep k components
        
```

Figura 14: .VI procesado de imagen

Como salida tendremos una variable real que nos indicará la clase a la que pertenecerá la persona captada después de haber sido procesada y un flag que indicará que se han procesado todas las imágenes existentes. En la figura 15 podemos ver el icono de este nuevo subvi.

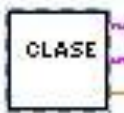


Figura 15: Icono subvi procesado de imagen

Tendremos un tercer subvi de donde se extraerá la clase a la cual pertenece la persona que se encuentre delante de la cámara, en la figura 14 podemos ver el icono de este tercer subvi. Este subvi incorporará como salida la hora y el nombre de la persona.



Figura 16: Subvi que extrae el nombre de la persona.

Por último, generaremos el .vi principal en el cual se realiza la conexión de estos dos subvis. En la imagen 15 podemos ver el conjunto íntegro del .vi principal.

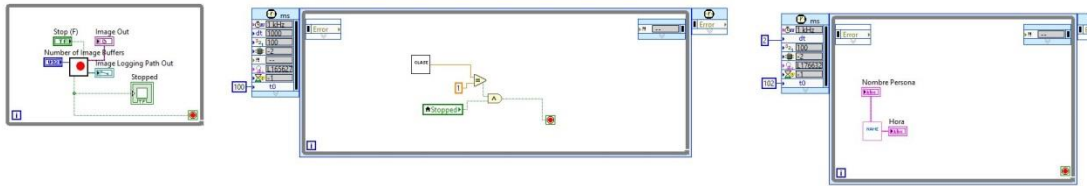


Figura 17: .VI principal

Como se observa , para ejecutar el primer subvi se hace uso de un While Loop donde se extraen las imágenes y se almacenan en el Path de salida. Al usuario se le da permisos para poder detener la captura y se muestra la imagen de salida.

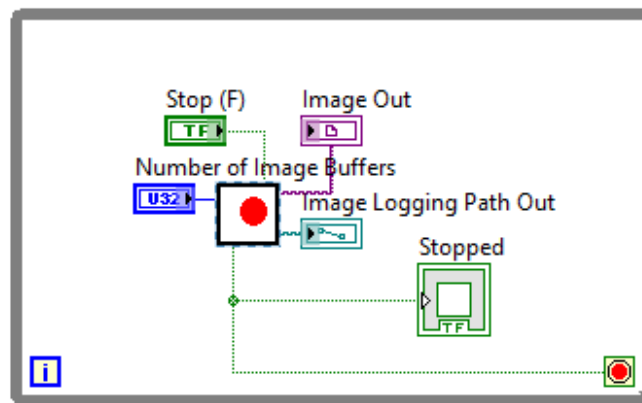


Figura 18: Parte de la adquisición de la imagen en el .vi principal

En la siguiente figura, podemos ver como el subvi donde se procesa la imagen utiliza un Timed Loop con el que se controlará cuando debe comenzar a ejecutarse este subvi. En este caso se hace uso de un temporizador que retardará la ejecución de este subvi de 100ms con lo que nos aseguraremos de que la cámara habrá capturado imágenes que serán procesadas.

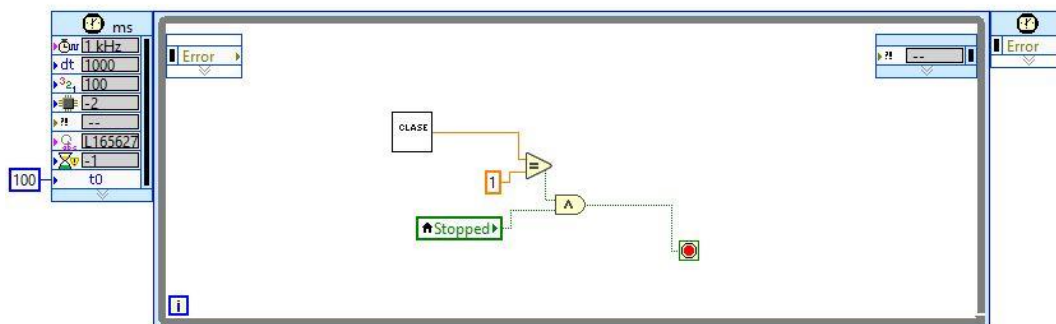


Figura 19: Parte del procesamiento de imagen en el .vi principal

En la figura 20 podemos ver la estructura que tiene el tercer subvi que extraerá el nombre de la persona. Consiste en otro While Loop que tras un periodo de retardo y añadiendo una frecuencia de ejecución extraerá la hora y el nombre de la persona. Este subvi será ejecutado a los 102ms de que se haya ejecutado el primer subvi y tendrá un periodo interno de ejecución de 2 ms.

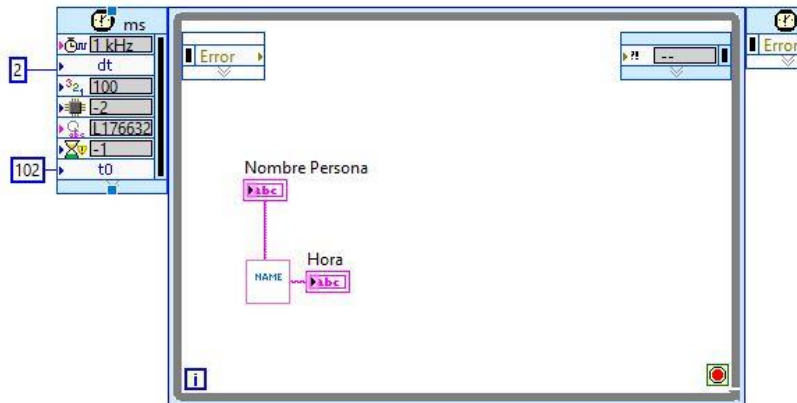


Figura 20: Método para incorporar el subvi que extraerá el nombre y la hora

Con este .vi principal conseguimos controlar la cámara de video vigilancia y el procesamiento de las imágenes que se obtienen así como el momento de ejecución de cada parte. En la última fase se explicará la interfaz asociada a este vi principal generado.

5.2 SEGUNDA FASE

En esta segunda fase, realizada con Matlab se ha procedido a la detección facial para las imágenes que se inserten en la base de datos.

Previamente se realiza una lectura de cada uno de los archivos que existen en el directorio donde se encuentra la base de datos. El directorio tiene una forma específica. Dentro de la carpeta principal tendremos una carpeta donde se almacenaran las caras y tantas carpetas como clases tengamos. Es necesario que cada carpeta contenga como nombre, el nombre de la clase de forma numérica y que en ella se encuentren las imágenes de la persona que pertenece a dicha clase.

En la siguiente figura podemos ver un ejemplo donde se muestra la carpeta principal llamada 'BBDD', en ella encontramos una carpeta llamada 'Todas' que no deberá ser modificada por el usuario y otras 9 carpetas nombradas del 1 al 9, que especifican cada una de las clases.

En cada una de estas nueve carpetas se almacenan las fotografías de la base de datos de los nueve sujetos diferentes.

De esta forma el usuario podrá ampliar dicha base de datos o eliminar sujetos que ya no se desee tener en la Base de Datos siguiendo este formato y nomenclatura.

1	03/07/2017 19:20	Carpeta de archivos
2	03/07/2017 19:20	Carpeta de archivos
3	03/07/2017 19:20	Carpeta de archivos
4	03/07/2017 19:20	Carpeta de archivos
5	03/07/2017 19:20	Carpeta de archivos
6	03/07/2017 19:20	Carpeta de archivos
7	03/07/2017 19:20	Carpeta de archivos
8	03/07/2017 19:20	Carpeta de archivos
9	03/07/2017 19:20	Carpeta de archivos
Todas	03/07/2017 19:22	Carpeta de archivos

Figura 21: Formato en el que han de estar las imágenes de la Base de Datos para tratar con ellas

Una vez se han leído las imágenes de la Base de Datos y se ha obtenido el vector necesario para realizar Fisherfaces, se detectará la cara de las imágenes existentes en ella. En la siguiente figura podemos ver el procesamiento de cada imagen y como se ha detectado si existe cara o no con el uso de Viola-Jones.

```
clear all
num_carpetas_previo =
load('C:\Users\necaena\Desktop\texto\lista_carpetas.mat',
'list_dir');
sice_carpetas_anterior = numel(num_carpetas_previo.list_dir);
load('C:\Users\necaena\Desktop\texto\imageOwner.txt');
list_dir=dir(fullfile('E:\Nerea\Fisherfaces\Prueba para
Labview\BBDD\')); %%Contar carpetas en el directorio de la BBDD
if (numel(list_dir) ~= sice_carpetas_anterior)
nombre_carpeta = [];
imageOwner = [];
for i = 1:numel(list_dir)
nombre_carpeta = list_dir(i).name;
nombrel = str2num(nombre_carpeta);
logical = isempty(nombrel);
if logical == 0
tipo = '\*.jpg';
directorio = strcat(num2str(nombrel), tipo);
lee_carpeta = dir(['E:\Nerea\Fisherfaces\Pruebas para
Labview\BBDD\' directorio]);
for j=1:numel(lee_carpeta)
barra = '\';
nombre_foto = (lee_carpeta(j).name);
nombre_completo =
strcat(nombre_carpeta, barra, nombre_foto);
foto = imread(['E:\Nerea\Fisherfaces\Prueba para
Labview\BBDD\' nombre_completo]);
imwrite(foto, ['E:\Nerea\Fisherfaces\Prueba para
Labview\BBDD\Todas\' nombre_foto], 'jpg')
imageOwner=[imageOwner nombrel];
end
end
save('C:\Users\necaena\Desktop\texto\lista_carpetas.mat',
'list_dir', '-mat');
save('C:\Users\necaena\Desktop\texto\imageOwner.txt',
'imageOwner', '-ascii');
```

```

end
end

lee_archivos = dir('E:\Nerea\Fisherfaces\Pruebas para
Labview\BBDD\Todas\*.jpg'); %el formato de imagen puede ser
modificado.
i0=1;
k=1;
Npx=120;
while length(lee_archivos) > 0 %%
archivo = lee_archivos(1).name; %Obtiene el nombre de los
archivos
nombre = 'E:\Nerea\Fisherfaces\Prueba para Labview\BBDD\Todas\';
%Recore el diretorio

I = imread(strcat(nombre,archivo));% lee la primera imagen
FDetect = vision.CascadeObjectDetector;
%Devuelve una matriz con el numero de objetos encontrados y sus
posiciones.
FDetect.MergeThreshold = 8;
BB = step(FDetect,I); % Bounding Box coordenadas de la imagen
don existe una cara
BB2 = [BB(:,1) BB(:,2) BB(:,3) (BB(:,4)+50)];
mat_image=cell(1,numel(BB2,1)); % Es una celda de matrices con
las coordenadas de BB
archivo0=archivo(1:end-4);% eliminamos las 4 ultimos caracteres
del nombre de la imagen el .jpg para no sobrescribirlo
for i = 1:size(BB2,1)
mat_image{i}=imcrop(I,BB2(i,:));
mat_image{i}=imresize(mat_image{i}, [Npx, Npx]);
nombre2 = 'E:\Nerea\Fisherfaces\Prueba para
Labview\BBDD\Todas\Caras\';
B= (strcat(nombre2,num2str(str2num(archivo0),'%-
0.4i')));
imwrite(mat_image{i},[B '_' num2str(i,'%0.4i')
'.jpg'],'jpg');
i0=i0+1;
end
delete(strcat(nombre,archivo)); %Borrar imagen analizada
lee_archivos = dir('E:\Nerea\Fisherfaces\Prueba para
Labview\BBDD\Todas\*.jpg'); %el formato de imagen puede ser
modificado.
end
%end

```

Figura 22: Código Matlab realizado para leer las imágenes que el usuario almacene en la Base de Datos

En el código podemos ver como se leen los archivos existentes en formato .jpg en cada una de las subcarpetas donde se encuentran las imágenes.

Se lee cada una de las imágenes que existen en la carpeta 'Todas' donde se procesa con la función 'vision.CascadeObjectDetector' que es la función propia de Matlab para utilizar Viola-Jones. En este caso se le ha puesto un umbral de 8 obtenido por varias pruebas realizadas para evitar que nos quedemos con alguna imagen que realmente no es una cara.

Con esta función, cuando existe una cara, nos quedamos con las coordenadas en las que ha detectado la cara. Viola-Jones siempre devuelve coordenadas formando un cuadrado, por lo que luego la imagen podrá ser redimensionada sin problema. Una vez

tenemos estas coordenadas, recortamos la imagen original, quedándonos solamente con la cara.

La imagen original que se encuentra en 'Todas' será eliminada para ahorrar espacio. Por último, cada una de estas imágenes almacenadas se redimensionan en un tamaño de 120x120 poder trabajar con ellas.

Una vez tenemos la cara, se continúa con el procesamiento de la imagen que se ha almacenado donde se ha detectado una cara. En la siguiente figura podemos ver el procesamiento de Fisherfaces del que se hace uso para la base de datos creada.

```
lee_caras = dir('E:\Nerea\Fisherfaces\Prueba para
Labview\BBDD\Todas\Caras\*.jpg');
numero_ficheros = numel(lee_caras);
face=zeros([120*120 numero_ficheros]); %Matriz face para almacenar las
caras

for i = 1:numero_ficheros
    archivo = lee_caras(i).name; %Obtiene el nombre de los archivos
    nombre = 'E:\Nerea\Fisherfaces\Prueba para
Labview\BBDD\Todas\Caras\'; %Recorre el directorio
    face0 = rgb2gray(imread(strcat(nombre,archivo))); %%Redimensionar y
cargar imagenes en escala de grises
    face(:,i)=face0(:); %Matriz con las caras almacenadas con tamaño
120x120 y en gray
end

nRow = 120; %Numero de filas
nCol = 120; %Numero de columnas
V_Clases = imageOwner';

numComponentsToKeep = 9; %Número de componentes a almacenar NO PUEDE
SER MAYOR A c

A = face; %Matriz de imágenes de la base de datos
A = A';
N = size(A,1); % Número de imagenes totales
m_database = mean(A,1); %Media de las imágenes de la base de datos
A = A - repmat(m_database,size(A,1),1);
labels = unique(imageOwner);
c = length(labels); % Número de clases

numComponentsToKeep = min(numComponentsToKeep, (c-1));
[Wpca, mu] = pca(A, imageOwner, (N-c));
proj = (A-repmat(mu, N, 1))*Wpca;
[Wlda, mu_lda] = lda(proj, imageOwner, numComponentsToKeep);
prinComponents = Wpca*Wlda;
weightCols = prinComponents'*A';
```

Figura 23: Código Matlab para la extracción de las Fisherfaces de las imágenes de la Base de Datos

En este código se leen las caras existentes en la base de datos que han sido localizadas, recortadas y redimensionadas en la fase anterior. Estas imágenes se almacenan en escala de grises.

Se crea una matriz donde se encuentran cada una de estas imágenes, para poder realizar la extracción de los Fisherfaces. Cada una de las columnas de la matriz será una imagen.

Posteriormente se calcula la media de cada uno de los píxeles de cada imagen y se le restará al conjunto de todas las matrices para equilibrar las condiciones de iluminación.

Teniendo el vector con el orden de cada clase se indica un número de componentes a extraer que nunca podrá superar al número de clases existentes menos uno.

Se realiza la extracción de las componentes PCA y LDA del conjunto de datos, así como sus proyecciones, calculamos los componentes para Fisherfaces mediante la multiplicación de estas dos proyecciones y por último extraemos los pesos de Fisherfaces multiplicando estas componentes por la matriz con las imágenes.

En la figura 24 y 25 podemos ver las funciones PCA y LDA que se utilizan para extraer los componentes necesarios.

```
function [W, mu] = pca(X, y, k)
[n,d] = size(X);
mu = mean(X);
Xm = X - repmat(mu, size(X,1), 1);
if(n>d)
    C = Xm'*Xm;
    [W,D] = eig(C);
    % sort eigenvalues and eigenvectors
    [D, i] = sort(diag(D), 'descend');
    W = W(:,i);
    % keep k components
    W = W(:,1:k);
else
    C = Xm*Xm';
    %C = cov(Xm');
    [W,D] = eig(C);
    % multiply with data matrix
    W = Xm'*W;
    % normalize eigenvectors
    for i=1:n
        W(:,i) = W(:,i)/norm(W(:,i));
    end
    % sort eigenvalues and eigenvectors
    [D, i] = sort(diag(D), 'descend');
    W = W(:,i);
    % keep k components
    W = W(:,1:k);
end
end
```

Figura 24: Función PCA aplicada para la extracción de las Fisherfaces

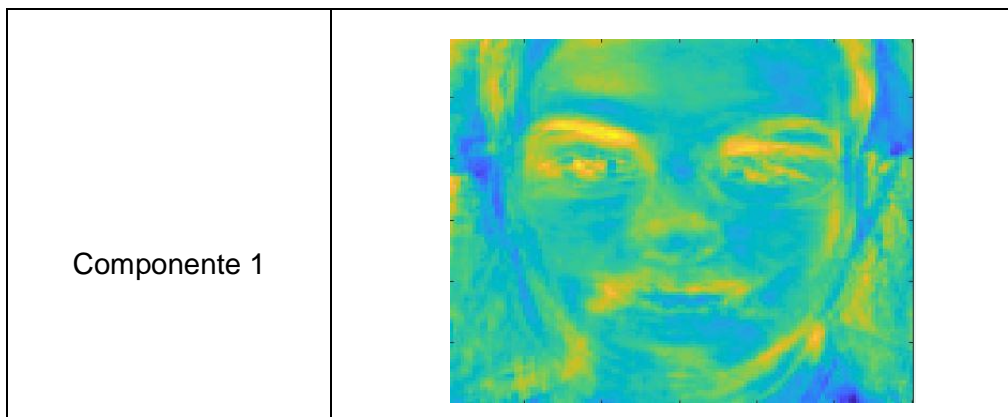
```

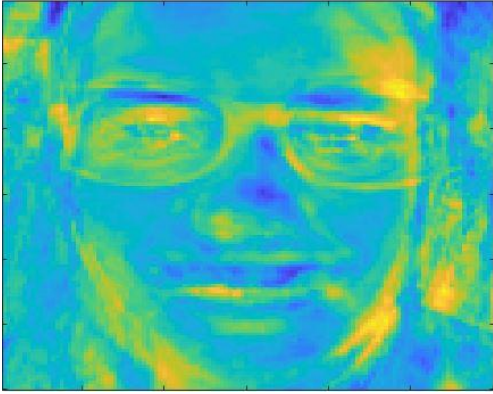
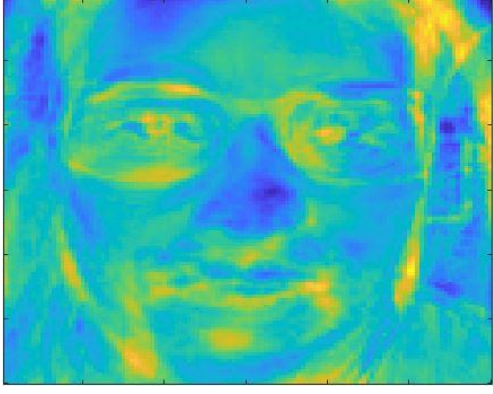
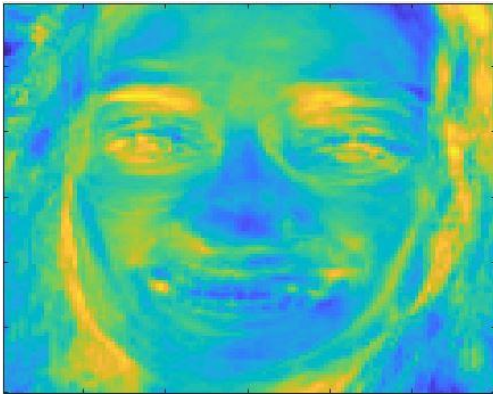
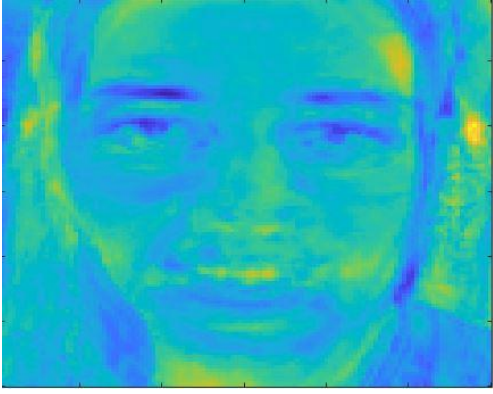
function [W, mu] = lda(X,y,k)
    [n,d] = size(X);
    % number of classes
    labels = unique(y);
    C = length(labels);
    % allocate scatter matrices
    Sw = zeros(d,d);
    Sb = zeros(d,d);
    % total mean
    mu = mean(X);
    % calculate scatter matrices
    for i = 1:C
        Xi = X(find(y == labels(i)),:);
        n = size(Xi,1);
        mu_i = mean(Xi); % mean vector for current class
        Xi = Xi - repmat(mu_i, n, 1);
        Sw = Sw + Xi'*Xi;
        Sb = Sb + n * (mu_i - mu)'*(mu_i - mu);
    end
    % solve general eigenvalue problem
    [W, D] = eig(Sb, Sw);
    % sort eigenvectors
    [D, i] = sort(diag(D), 'descend');
    W = W(:,i);
    % keep at most (c-1) eigenvectors
    W = W(:,1:k);
end

```

Figura 25: Función LDA aplicada para extracción de las Fisherfaces

En la siguiente imagen podemos ver las características Fisherfaces de forma gráfica que se obtienen de las imágenes de la base de datos. Dado que el número de prueba en esta base de datos es de nueve, se van a obtener un máximo de ocho componentes Fisher.



Componente 2	
Componente 3	
Componente 4	
Componente 5	

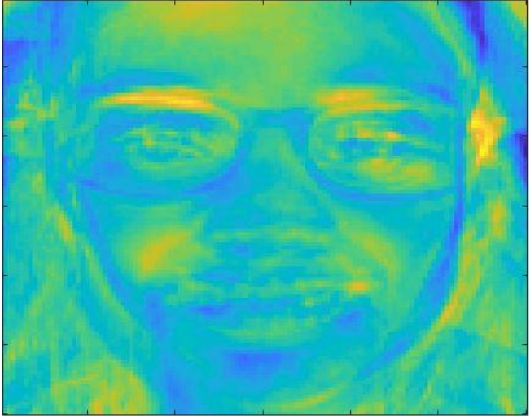

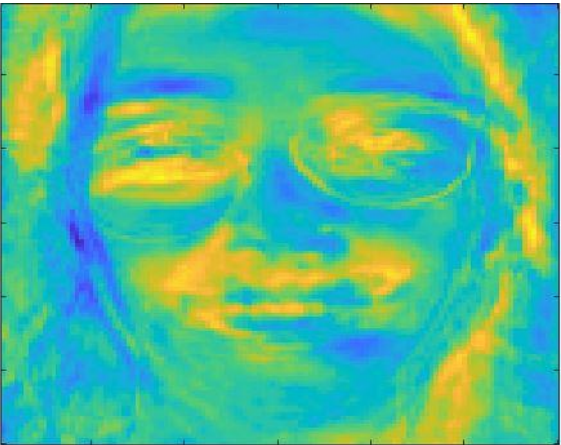
Componente 6	
Componente 7	
Componente 8	

Figura 26: Fisherfaces de la base de datos

5.3 TERCERA FASE

En esta tercera fase, con Matlab, se ha procedido a la detección facial de las imágenes que se extraen de la cámara de video vigilancia. En la siguiente figura podemos ver el código que se ha realizado para poder procesar cada una de las

imágenes captadas por Labview, la detección facial en cada una de ellas y su posterior almacenamiento.

```
lee_archivos_in = dir('C:\Users\necaena\Desktop\prueba
express\*.jpg'); %el formato de imagen puede ser modificado.
l0=1;
nn = 1;
while length(lee_archivos_in) > 0
archivo_in = lee_archivos_in(nn).name; %Obtiene el nombre de los
archivos
nombre_in = 'C:\Users\necaena\Desktop\prueba express\'; %Recore
el directorio
Pers = imread(strcat(nombre_in,archivo_in));% lee la primera
imagen
FDetect = vision.CascadeObjectDetector;%Devuelve una matriz con
el numero de objetos encontrados y sus posiciones.
FDetect.MergeThreshold = 8;
BB_in = step(FDetect,Pers); % Bounding Box coordenadas de la
imagen don existe una cara
BB_in2 = [BB_in(:,1) BB_in(:,2) BB_in(:,3) (BB_in(:,4)+50)];
Pers = imcrop(Pers, BB_in2);%Recorte de la cara de la imagen de
entrada
```

Figura 27: Código Matlab para la extracción de las imágenes captadas por la imagen y detección facial.

Con la variable 'lee_archivos' se leen los archivos existentes en formato .jpg en el directorio establecido. Este directorio es el directorio en el cual se han almacenado las imágenes obtenidas en la primera fase.

Se procesan todas las imágenes existentes dado que se utiliza la condición de que se realice el mismo proceso hasta que dicho directorio no contenga ningún archivo .jpg.

Posteriormente se lee cada una de las imágenes, leyendo siempre la primera que irá cambiando dado que una vez esta sea procesada será borrada de este directorio para poder ahorrar costes en almacenamiento.

Una vez leída se le aplica la función 'vision.CascadeObjectDetector' misma función aplicada en la fase anterior para detectar si existe cara.

Al igual que en el caso anterior nos quedamos con las coordenadas en las que ha detectado la cara, recortamos la imagen original para estas coordenadas para quedarnos solamente con la cara y posteriormente comparar con la base de datos.

Esta imagen será redimensionada de la misma forma que las imágenes de la base de datos para que ambas imágenes tenga el mismo tamaño y puedan ser comparadas. Por último, como se ha indicado antes, la imagen procesada será eliminada para ahorrar espacio de almacenamiento.

5.4 CUARTA FASE

En la cuarta fase, con Matlab también, se proyectará la imagen de entrada en el espacio de Fisherfaces obtenido de la base de datos y se realizará la comparación con la proyección de todas las imágenes de la base de datos a partir del cálculo de la Distancia Euclídea y el clasificador K-Vecinos.

En la figura 28 se encuentra la forma en la que se calculan los pesos de las imágenes de entrada. En ella podemos ver como se pasa la imagen a blanco y negro y después de ponerla en forma de columna, se resta la media de la base de datos obtenida en la fase anterior para acercar las condiciones de iluminación. Por último se calculan los pesos multiplicando los componentes que se han obtenido con el uso de las funciones de PCA y LDA por dicha resta.

```
Pers = imresize(Pers, [Npx,Npx]);
%imwrite (Pers, 'BBDD\Test\Cara\1.jpg');
Pers = rgb2gray(Pers);
Pers = Pers(:);
Difference = double(Pers)-m_database.'; %Diferencia entre la
imagen de entrada y la media de la base de datos
weightCols_in = prinComponents'*Difference;
```

Figura 28: Extracción de las características Fisherfaces de las imágenes de entrada

Una vez se han calculado los pesos, se procede a medir la distancia Euclídea de esta imagen de entrada con los pesos de cada una de las imágenes patrón. Para ello utilizamos la ecuación que se ha descrito anteriormente.

Estas distancias las almacenamos en un vector y las ordenamos de forma ascendente, quedando en primera posición la distancia más pequeña para poder aplicar el clasificador K-vecinos. El valor de K varía en función del número de imágenes que haya por cada clase en la base de datos, para este caso se ha utilizado una K = 5.

Aplicando K-Vecinos, tendremos que la clase más repetida dentro de esas 5 primeras distancias, la más repetida será a la cual pertenece la imagen de entrada.

Además, se tendrá en cuenta un umbral en el cual si la distancia mínima existente es mayor a dicho umbral se concluirá que la persona no se encuentra en la base de datos.

En la figura 29 podemos ver como se realiza este cálculo y se determina la clase.

```
Ao = weightCols_in(1,1);
Bo = weightCols_in(2,1);
Co = weightCols_in(3,1);
Do = weightCols_in(4,1);
Eo = weightCols_in(5,1);
Fo = weightCols_in(6,1);
Go = weightCols_in(7,1);
Ho = weightCols_in(8,1);

Distancias=zeros(N,1);

for V=1: N
```

```

        Distancias(V)=sqrt(((weightCols(1,V)-
Ao)^2)+((weightCols(2,V)-Bo)^2)+((weightCols(3,V)-
Co)^2)+((weightCols(4,V)-Do)^2)+((weightCols(5,V)-
Eo)^2)+((weightCols(6,V)-Fo)^2)+((weightCols(7,V)-
Go)^2)+((weightCols(8,V)-Ho)^2));
        end

        Clases_Distancias = [V_Classes Distancias]; %Clases y
Distancias
        [imagen, L] = sort(Clases_Distancias(:,2)); %Orden de las
distancias
        imagen = [Clases_Distancias(L) imagen];
        d_min = imagen(1,2);
        clase = [];
        if d_min > 3e+03
            clase = 0;
        else
            k_vecinos = imagen(1:5)'; %Almacenar las 5 primeras
clases con menores distancias
            vecino_repetido = mode(k_vecinos);
            clase = vecino_repetido;
        end
end

```

Figura 29: Código Matlab del cálculo de la distancia Euclídea y clasificador K-Vecinos

5.5 QUINTA FASE

En esta fase, volviendo a utilizar Labview se procesa la imagen desde el MathScript que incorpora Labview. En este subvi se introduce el código Matlab generado para esto.

Para poder realizar el reconocimiento facial de la persona en tiempo real se ha añadido a este código que se escriba la clase en un documento de texto.

En el tercer subvi generado se leerá este archivo de texto y dependiendo del número que indica la clase que se extraiga se indicará el nombre de la persona a la que pertenece dicha clase.

A su vez, se ha incorporado un reloj desde Labview para que se pueda ver la hora a la que sucede cada uno de los reconocimientos.

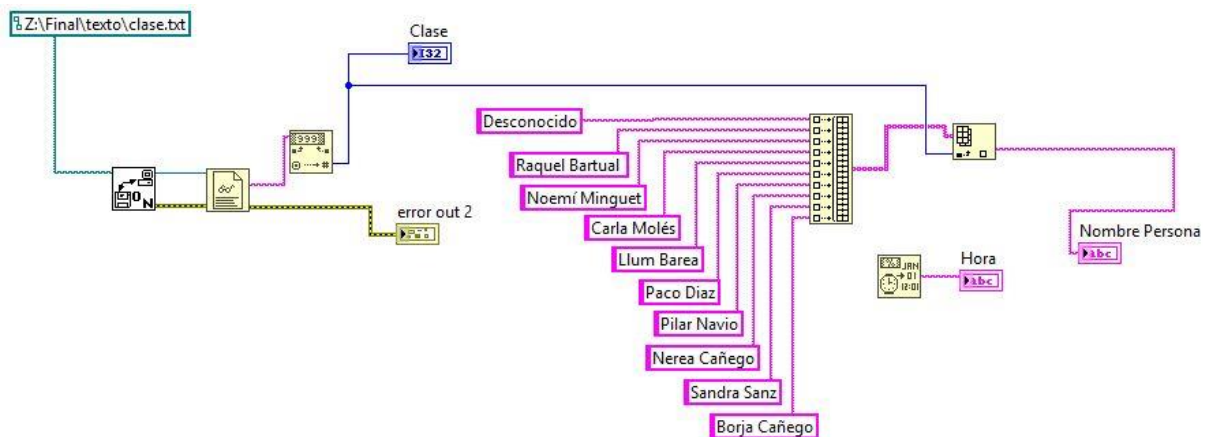


Figura 30: .Vi de lectura de la clase para mostrarlo en la interfaz

Como podemos ver la figura se realiza la lectura de este archivo de texto, donde transformando el contenido del archivo, obteniendo el número de la clase.

Este número corresponde al índice en el cual se encuentra el nombre de la persona que será mostrado por pantalla.

De forma independiente tenemos la extracción de la hora para poder mostrarlo en la interfaz.

5.6 SEXTA FASE

En la última fase se ha realizado la interfaz por Labview del software completo a través del .vi principal generado.

En la siguiente figura podemos ver cómo queda la interfaz construida que consta de una ventana donde se muestra la imagen recibida de la cámara, un botón de 'stop' para detener la grabación de la misma junto a una luz que se encontrará en rojo cuando se esté grabando y en verde cuando no.

Debajo de esta pantalla de visualización se tiene la hora actual y el nombre de la persona que se encuentra delante de ella.










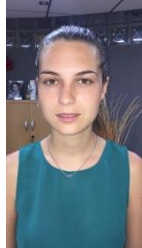



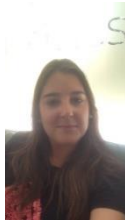


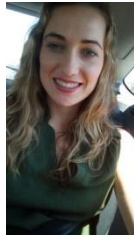





Figura 31: Interfaz programa

6. BASE DE DATOS GENERADA

La base de datos que se ha creado para este proyecto, contiene 9 clases diferentes con 4 imágenes por cada clase haciendo un total de 44 imágenes. Es decir, existen 9 personas diferentes a identificar en la base de datos.

Como se ha utilizado el algoritmo Fisherfaces, se ha podido utilizar imágenes con diferentes iluminaciones, poses e incluso complementos en estas personas como pueden ser las gafas.

En la siguiente figura podemos ver las imágenes utilizada para la base de datos. Esta base de datos, como se ha repetido en varias ocasiones podría ser modificada en cualquier momento haciendo uso del formato requerido. Es decir, será necesario que si se desea añadir una décima persona a reconocer en el directorio donde está la carpeta de 'BBDD' se cree una carpeta con nombre '10' y se introduzcan las imágenes de esta nueva persona. El número de imágenes por personas ha de ser siempre el mismo.

FOTO 1	FOTO 2	FOTO 3	FOTO 4	CLAS E
				1
				2
				3
				4
				5

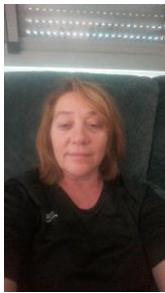
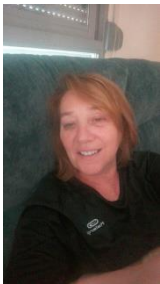

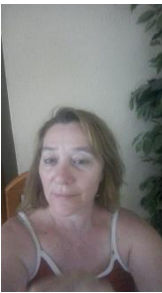










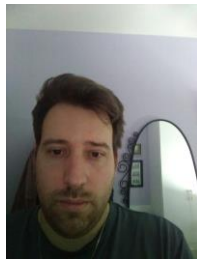
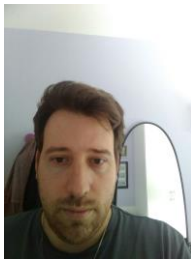
				6
				7
				8
				9

Figura 32: Base de Datos generada en la aplicación

Esta base de datos ha sido generada a partir de personas conocidas. Se han pedido fotos a 15 personas, de las cuales, sólo 12 son válidas dado que se han pasado fotos recortadas o modificadas que venían con pérdida de información. La base de datos se ha creado de 9 personas y se han guardado las fotos de las otras 4 personas para poder calcular el umbral necesario donde se deberá considerar que la persona no pertenece a la base de datos.

Se debe tener en cuenta que al procesar la imagen, especialmente al convertirla a escala de grises, se pierde información, es importante que las imágenes de la base de datos estén sin procesar previamente. Sin embargo, no es necesario que las fotografías estén realizadas ni con la misma cámara, ni con la misma iluminación y

como se ha dicho antes, es indiferente si la persona lleva objetos puesto como gafas, collares, diademas etc...

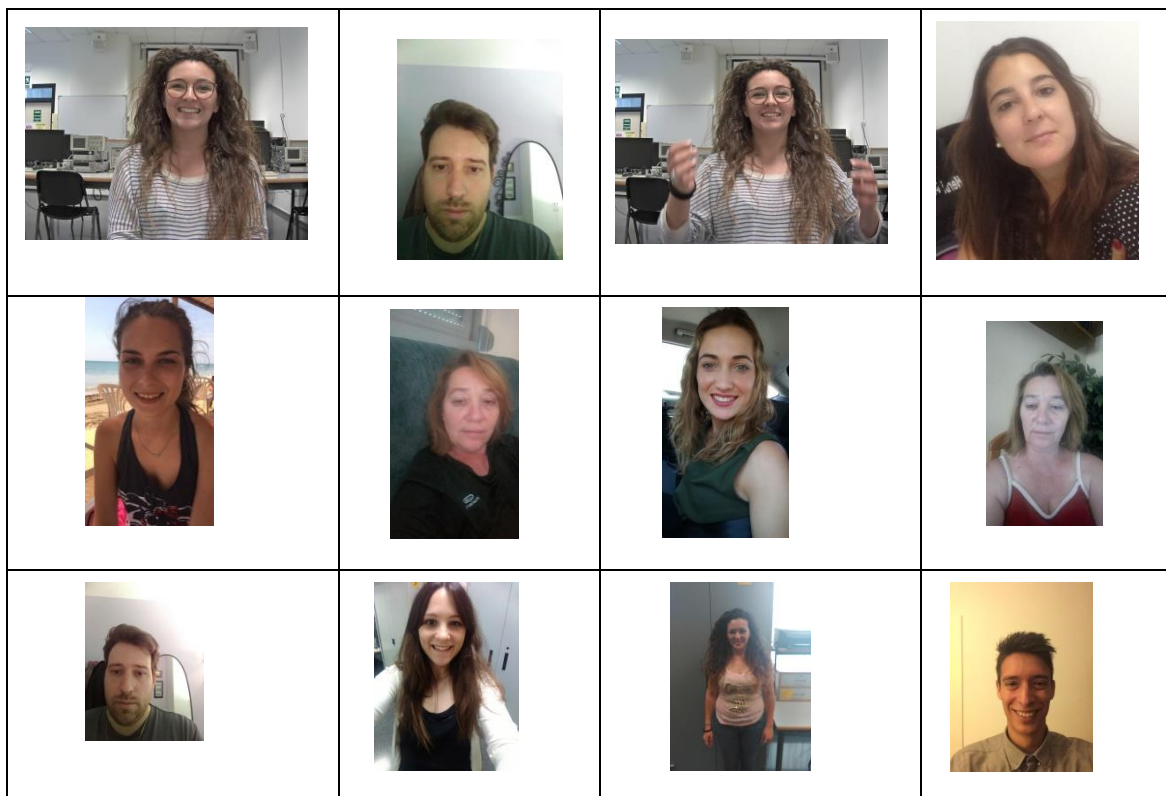
7. RESULTADOS

Se han realizado diferentes pruebas sobre el procesado de la imagen, es decir, se ha probado si el código Fisherfaces utilizado para determinar si la clase de la persona funciona de forma correcta.

En cuanto a las pruebas realizadas para comprobar el funcionamiento del código, éstas se han dividido en nueve pruebas diferentes. Primero se han ido realizando pruebas de menor a mayor número de imágenes por clase en la base de datos, desde dos imágenes por clase hasta cuatro imágenes por clase. No se puede realizar con una única imagen por clase dado que para que el código funcione es necesario que existan dos fotografías por cada una de las personas que se encuentran en la base de datos.

A su vez, se ha ido variando el número de componentes Fisherfaces calculadas. Se ha probado con cuatro componentes, seis componentes y ocho componentes, máximo permitido para esta base de datos dado que existen nueve clases diferentes.

Para todas estas pruebas se han utilizado las mismas 20 imágenes de test, de las cuales 14 imágenes pertenecen a personas que existe en la base de datos y las otras 6 imágenes son de personas que no existen en la base de datos. Estas imágenes de test se pueden ver en la siguiente figura.



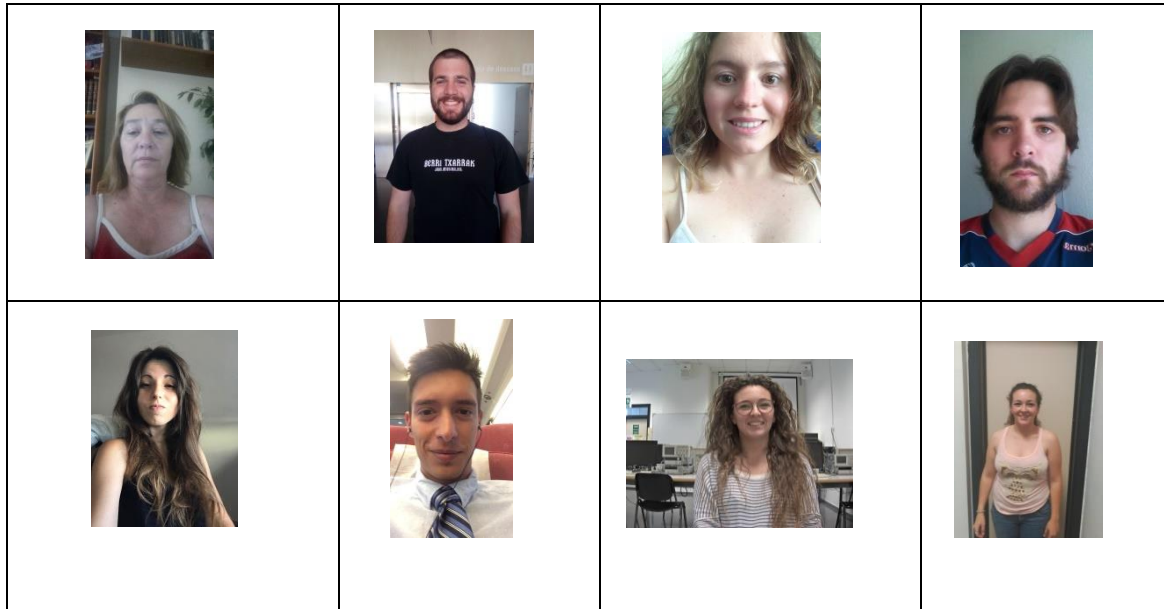


Figura 33: Imágenes de testeo utilizadas para las pruebas

7.1 RESULTADOS DOS IMÁGENES POR CLASE

En esta prueba se tienen dos imágenes por clase en la base de datos, por lo que se han calculado los Fisherfaces para un número total de 18 imágenes y luego se ha hecho el testeo con las 20 imágenes que tenemos.

	4 componentes	6 componentes	8 componentes
Aciertos	10	10	11
Falsos	10	10	9
Tasa de acierto	50%	50%	55%

Figura 34: Tabla de resultado con dos imágenes por clase

Como se puede observar a través de la tabla de resultados, se tiene una tasa de acierto de hasta el 55% utilizando el máximo de componentes Fisher, siendo esta tasa de acierto mayor a los otros dos casos. Sin embargo, esta tasa de acierto es pequeña en cuanto a los objetivos previstos, esto se debe a que tenemos pocas imágenes por cada una de las clases en la base de datos.

7.2 RESULTADOS TRES IMÁGENES POR CLASE

En esta prueba tenemos tres imágenes diferentes en cada una de las clases existentes en la base de datos. Se calculan los Fisherfaces para un total de 27 imágenes y el testeo se realiza con las mismas imágenes de test que en el caso anterior.

	4 componentes	6 componentes	8 componentes
Aciertos	14	14	14
Falsos	6	6	6
Tasa de acierto	70%	70%	70%

Figura 35: Tabla de resultado con tres imágenes por clase

En este caso se obtiene una tasa de acierto igual en las tres pruebas realizadas, la tasa de acierto aumenta bastante en comparación con la prueba anterior debido a que tenemos una imagen más por cada una de las clases, sin embargo, no existe un aumento con el cambio de los componentes Fisherfaces.

7.3 RESULTADOS CUATRO IMÁGENES POR CLASE

En esta prueba tenemos cuatro imágenes por cada una de las clases en la base de datos. Se calculan los Fisherfaces para un total de 36 imágenes y el testeo se realiza con las mismas imágenes de test que en el caso anterior.

	4 componentes	6 componentes	8 componentes
Aciertos	15	16	16
Falsos	5	4	8
Tasa de acierto	75%	80%	80%

Figura 36: Tabla de resultado con cuatro imágenes por clase

Con esta última prueba se obtiene una tasa de acierto del 80% que supera en un 10% el objetivo previsto en el proyecto. Esta tasa de acierto ha aumentado considerablemente debido a que utilizamos cuatro imágenes por cada una de las clases existentes.

Podemos observar como el número de componentes Fisherfaces a extraer aumenta la tasa de acierto en algunos casos pero de forma muy lenta, sin embargo se puede ver como con el aumento de la cantidad de imágenes por clase de la base de datos, la tasa de acierto aumenta considerablemente. Por lo tanto, la mejor opción es tener una base de datos grande en cuanto a número de imágenes por clase y utilizar el máximo número de componentes Fisherfaces para esta base de datos, es decir utilizar siempre un número de componentes Fisherfaces menor en uno a la cantidad de clases existentes.

Para esta esta base de datos, se obtendrá un mejor resultado utilizando las cuatro imágenes por clases y con la extracción de ocho características Fisherfaces.

8. CONCLUSIONES

Según se ha ido viendo a lo largo del proyecto, se ha decidido utilizar cada una de las técnicas que se han descrito por ser las más adecuadas en cuanto a eficiencia y complejidad.

A lo largo del proyecto nos hemos encontrado con diversos problemas de conocimientos que se han ido ampliando poco a poco, por lo que se ha logrado el objetivo anteriormente indicado sobre ampliar el conocimiento existente en el tratamiento de imágenes.

Sin embargo, el conocimiento sobre bases de datos ya existentes no se ha ampliado en gran cantidad el conocimiento personal dado que para este proyecto era conveniente tener una base de datos de personas conocidas.

Por otro lado, si se ha conseguido adquirir un mayor manejo de los programas que se han utilizado, es decir, después de terminar el proyecto se ha obtenido un mayor conocimiento de Labview y Matlab

En cuanto a la tasa de acierto se ha conseguido un 80% haciendo uso de cuatro imágenes por persona y teniendo ocho componentes Fisher. Se puede concluir por lo tanto que se ha alcanzado el objetivo especificado al comienzo del proyecto.

9. TRABAJO FUTURO

Continuando con el proyecto, sería posible mejorar la tasa de acierto del programa consiguiendo más imágenes para cada una de las clases de la base de datos. También se podría bajar la tasa de aciertos intentando añadir otro método de decisión además del K-Vecinos y la distancia Euclídea.

Otra forma para bajar la tasa de acierto sería corregir la orientación de la imagen en caso de que la persona tuviera la cara inclinada en ese momento, por lo que si tenemos las caras tanto en la base de datos como en la imagen de entrada con la misma orientación la comprobación sería más acertada.

En cuanto al sistema de vídeo se podría añadir mediante código Matlab un seguimiento facial, para poder encontrar la cara de forma más rápida y poder bajar el retraso que se introduce al calcular todos los parámetros necesarios para obtener la identificación de la persona.

10. BIBLIOGRAFÍA

1. **Axis Communications.** Cámara de red AXIS P1354. [En línea] [Citado el: 19 de 04 de 2017.] <https://www.axis.com/global/es/products/axis-p1354>.
2. **Banerjee, Asit Kumar Datta Madhura Datta Pradipta Kumat.** *Face Detecction and Recognition: Theory and Practice*. New York : CRC Press, 2016.
3. **Bartlett, Marian Stewart.** *Face Image Analysis by unsupervised learning*. San Diego : University of California, 2011.
4. **Hernández, Roger Gimeno.** *Estudio de técnicas de reconocimiento facial*. Barcelona : Escola Técnica Superior d'Enginyeria de Telecomunicació de Barcelona, 2010.
5. **Imparato, Abdón Alejandro Vivas.** *Desarrollo de un sistema de reconocimiento facial*. Madrid : Universidad Polécnica de Madrid, 2014.
6. **Iborra, Marcelo J. Armengot.** *Análisis comparativo de métodos basados en subespacios aplicados al reconocimiento de caras*. Valencia : Universitat de València, 2006.
7. **Martinez, Rafael Cazorla.** *Software para la detección y el reconocimiento de rostros*. Barcelona : Universitat Autònoma de Barcelona, 2016.
8. **Caselles, Ernest Valveny Jordi Gonzàlez Sabaté Ramon Baldrich.** *Lecture 11 - Support Vector Machines(SVM): Conceptos básicos*. [Vídeo] Barcelona : Coursera, 2017.
9. **EcuRed.** EcuRed. [En línea] 17 de Junio de 2012. [Citado el: 15 de Junio de 2017.] https://www.ecured.cu/Distancia_eucl%C3%ADdea.
10. **Mathworks.** *Mathworks*. [En línea] <https://es.mathworks.com/>.
11. <http://www.ni.com/es-es.html>. National Instruments. *National Instruments*. [En línea]
12. **Moreno, Robinson Jimenez.** *Sistema de detección de nivel de cansancio en conductores mediante técnicas de visión por computador*. Colombia : Universidad Nacional de Colombia, 2011.
13. **Liu, Haowei.** *Face Detectio an Recognition on Mobile Devices*. Waltham : Elsevier, 2015.
14. **Bartlett, Marian Stewart.** *Wikipedia*. San Diego : University of California, 2011.
15. **Shaogang Gong, Stephen J McKenna, Alexandra Psarrou.** *Wikipedia*. 2000 : Imperial College Press, 2016.
16. **Jun-Bao Li, Shu-Chuan Chu.** *Kernel Learning*. New York : Springer, 2014.
17. **Zhang, Yu-Jin.** *Advances in Face Image*. New York : Reference, 2011.
18. **Gérard Medioni, Sven Dickinson.** *Boosting-Based Face Detection*. s.l. : Morgan & Claypool Publishers, 2010.

19. **Claude C. Chibulushi, Fabrice Bourel.** *Facial Expression Recognition: A Brief Tutorial Overview* . UK : Staffordshire University, 2002.

20. **Zor, Cemre.** *Facial Expression Recognition*. Guildford, Surrey : University of Surrey, 2008.