UNIVERSIDAD POLITECNICA DE VALENCIA

ESCUELA POLITECNICA SUPERIOR DE GANDIA

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

"Detección facial y reconocimiento anímico mediante expresiones faciales"

TRABAJO FINAL DE GRADO

Autor/a:

Raquel Bartual González

Tutor/a:

Jose Ignacio Herranz Herruzo

Jose Pelegri Sebastia

GANDIA, 2017

Resumen

Este proyecto trata sobre el reconocimiento anímico de personas a partir de sus expresiones faciales. Para ello, se ha desarrollado un software mediante los programas LabView y Matlab, que se puede subdividir en varias fases.

En la primera de ellas el software será capaz de detectar la cara de cualquier persona en un video de alta calidad. Está detección o recorte del rostro de la persona será tratado mediante herramientas de procesado de imágenes.

En la segunda fase, la imagen se procesa para poder estimar el estado de ánimo que esta persona muestra según su cara. Se diferencia en un principio entre los seis estados de ánimo básicos de las personas, los cuales son: sorpresa, asco, tristeza, ira, miedo y alegría.

Y la última fase nos mostrará una gráfica con los diferentes estados de ánimo.

"Casi todo el mundo piensa que sabe que es una emoción hasta que intenta definirla. En ese momento prácticamente nadie afirma poder entenderla" (Wenger, Jones y Jones, 1962, pg. 3)

Abstract

The project is based on a software development elaborated with the program LabView and Matlab.

The above-mentioned program can be subdivided it in several phases. In phase one, the software will be capable of detecting the face of any person in a high-quality video. Once the face is detected, it will be processed with image processing tools and phase two will begin.

Continuing to the second phase, the image will be processed and we will be able to determine the emotional state of the individual according to his face. The software will be able to differentiate between the six basic emotional states, which are: surprise, disgust, sadness, rage, fear and happiness.

Finally the last phase will show us a graph with the different emotional states.

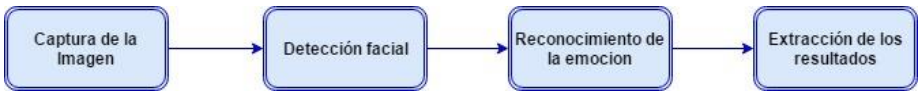


Figura1: Esquema general del programa

Palabras Clave

Detección, cara, emociones, Viola-Jones, Red Neuronal

Key Words

Detection, face, emotions, Viola, Jones, Neural- Network

Contenido

Tabla de figuras

1. Introducción

1.1. Presentación

Reconocer los estados de ánimo de los usuarios ha sido siempre un sistema muy solicitado para diversas áreas. Desde el lanzamiento de un producto, un videojuego, una película o incluso música se desea siempre poder agradar al público para que estos puedan salir adelante con grandes beneficios. Con el avance tecnológico se ha abierto un gran campo para poder reconocer los estados de ánimo de los usuarios.

En este proyecto se encarga de poder reconocer los estados de ánimo de uno o más sujetos de forma simultánea mientras son grabados por una cámara. Los estados de ánimo que se quieren distinguir son los estados de ánimo básicos según los psicólogos, alegría, ira, tristeza, neutro, sorpresa, asco y miedo.

A través de este reconocimiento se mostrará el resultado final de las emociones de los sujetos de forma gráfica que ayudará a valorar como se sienten los usuarios durante un periodo.

Para poder lograr la identificación de los estados de ánimo, previamente se ha de realizar una detección facial. Esta detección facial se realizará durante la grabación de los sujetos donde se podrán localizar diversas personas de forma simultánea.

Una vez se ha tenido esta detección facial se procederá al reconocimiento de estados de los estados de ánimo mediante varias técnicas que serán explicadas a lo largo del trabajo.

1.2. Objetivos del proyecto

El objetivo principal del proyecto es crear un programa en el cual se diferenciará entre 7 tipos de emociones diferentes.

En primer lugar, para realizar el proyecto de forma coherente y consciente, se ha recopilado la información necesaria sobre las diferentes fases en las que se divide este proyecto, empezando por los distintos tipos de descriptores en el procesamiento de imágenes, el tratamiento que se hace de ellos, la extracción de los valores fundamentales, y finalmente los tipos de clasificadores que existen.

Además, se ha investigado más concretamente los descriptores necesarios para reconocer las emociones a partir de la cara de una persona. El objetivo es que el programa, a partir del vídeo, sea capaz de detectar automáticamente a que clase o emoción en este caso pertenece cada persona detectada.

1.3. Etapas del proyecto

1.3.1. Etapa 1

Como etapa inicial de este proyecto se ha realizado una investigación de las técnicas, más actuales de detección y reconocimiento facial. Esto es lo que hemos considerado como "Estado del arte" que se desarrolla a lo largo de los apartados 3, 4 ,5, 6 y 7. El tiempo que se ha empleado en esta etapa ha sido de 1 mes.

Aun así, esta etapa no se ha dejado de lado en ningún momento, ya que se ha ido actualizando y buscando más información para mejorar, las técnicas con las que se analiza la imagen.

1.3.2. Etapa 2

En esta etapa se eligió el material con el cual íbamos a trabajar, sobre todo la cámara, la cual decidimos que fuera la Axis 1354 Network Camera, se configuro y se creó el software necesario para la grabación y almacenamiento de las imágenes, mediante Labview.

Todo este proceso nos llevó unos 15 días.

1.3.3. Etapa 3

Con esta etapa entramos en el grueso del proyecto, la realización del código de detección y reconocimiento de emociones.

En un principio este código se hizo a través de Matlab y posteriormente se pasó al programa de Labview, también se desarrolló la red neuronal. Esta etapa ha sido la que más tiempo nos ha llevado, siendo más de 3 meses lo que hemos tardado en desarrollar la programación, ejecución y corrección de errores del código final.

1.3.4. Etapa 4

Finalmente, la última etapa ha sido el desarrollo de la interfaz del usuario a través de LabView y el periodo de pruebas en tiempo real. Esto no ha llevado 3 semanas.

Finalmente, este proyecto se ha realizado en el trascurso de 6 meses desde febrero de 2017 hasta julio de 2017.

2. Material

El material utilizado para este proyecto consta de varios softwares para desarrollar el programa realizado, además de una serie de dispositivos los cuales se van a especificar detalladamente a continuación:

- Cámara de video (Axis 1354 Network Camera)
- Fuente de alimentación
- Ordenador con el software necesario
    - Labview 2016 + Vision and Motion Package
    - Matlab 2017ª + Computer Vision tolos

En primer lugar, la cámara que se ha utilizado ha sido una, Axis 1354 Network Camera.

Después de probar varios modelos de cámaras de la misma marca decidimos utilizar este modelo ya que nos permitía conectarnos a ella de forma sencilla, con una conexión Ethernet la cual se estableció y se dio de alta previamente. Y además teníamos acceso a su configuración interna y podíamos variar desde la base el formato de la imagen los fps etc.

La conexión con la cámara se realiza a través del programa LabView, para conectarnos de forma automática debemos saber la dirección IP de la cámara. La forma más sencilla de identificar la dirección IP, el nombre y las demás propiedades de la cámara es a través de una herramienta que nos facilita el propio National Instruments, empresa responsable de LabView.

El programa NI MAX es el que nos facilita toda la información de la cámara además de permitirnos un acceso rápido a ella para visualizar la imagen o incluso grabar.

En el menú que encontramos a la izquierda debemos seleccionar My System > Devices and Interfaces > Network Devices y en nuestro caso elegiremos la cam1.

En la Figura 2 podemos ver una muestra del programa.

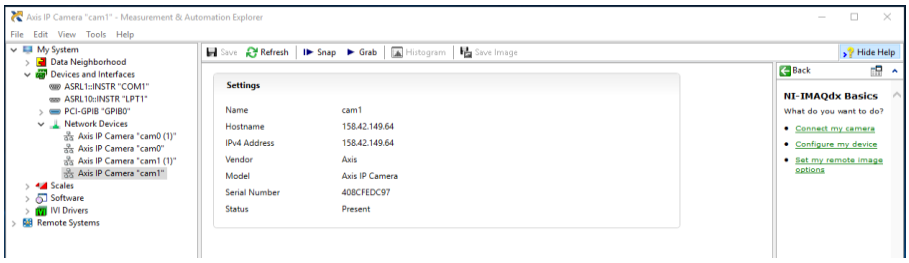


Figura 2: Datos de la Cámara

Una vez conseguido acceso a la cámara desde el programa se probó y configuró las diferentes especificaciones de esta, desde la Url que proporciona Axis para la configuración de estas cámaras a través de la red.

En esta Figura 3 se muestra la configuración que hemos decidido ponerle a la cámara, la cual se explicará en el Capítulo 8.1 con el inicio del proyecto.

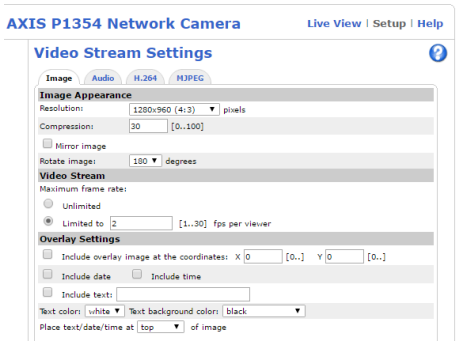


Figura 3: Configuración de la cámara

En este caso la cámara la hemos situado en el laboratorio de Instrumentación Avanzada, para ello necesitamos su conexión Ethernet y además una fuente de alimentación, que nos permita escoger la tensión necesaria para la cámara.

Por lo tanto, el montaje en el laboratorio de la cámara para realizar las pruebas se quedaba como en la Figura -4.

Figura 4: Montaje de la cámara en el Laboratorio

El resto de material utilizado son los programas LabView y Matlab ambos instalados en los ordenadores del laboratorio con los paquetes extras para la adquisición y procesamiento de imágenes. Todas las licencias necesarias para estos programas han sido facilitadas por la universidad.

Figura 5 : Logos de LabView y Matlab

3. Estado del arte: Descriptores

Los descriptores son valores específicos que se utilizan en las imágenes digitales para poder caracterizar los objetos presentes. Sirven por ejemplo para descartar objetos no deseados o manchas, diferenciar la forma de los objetos,

etc.…

Los descriptores más básicos que se pueden encontrar son los de forma, brillo o textura. Todos ellos se suelen utilizar para encontrar y delimitar las diferentes regiones que hay en una imagen, ya sea por color textura o forma.

En general, los descriptores se pueden clasificar en los siguientes tipos:

Los descriptores más básicos:

- De brillo (color/gris): se puede distinguir los diferentes cambios de color en una imagen, para eso la imagen debe de estar en un entorno muy controlado en cuanto a la iluminación.
- De tamaño: estos descriptores son más restrictivos en cuanto a las condiciones de captación, ya que se tienen que aplicar en cámaras fijas o conocer el tamaño real de la imagen.
- De forma: estos son los descriptores ideales para diferenciar los objetos entre sí. En general se pueden dividir entre descriptores de frontera o de región.

Como regla general, estos descriptores deben cumplir una serie de características: unicidad (un único valor posible por objeto) e invarianza con la translación, rotación y escalado [1].

Los descriptores más evidentes de una imagen digital son los valores de sus propios píxeles. Sin embargo, la enorme cantidad de ellos hace inviable cualquier tipo de clasificación. Se necesita, por tanto, una reducción de la dimensionalidad mediante por ejemplo los métodos PCA o LDA. Otros descriptores más avanzados sintetizan la información de los píxeles de diversas formas. Entre los más empleados, destacan los descriptores tipo HOG o Haar.

3.1. Descriptores HOG

En imágenes de escenas reales es difícil utilizar los descriptores básicos para detectar objetos o incluso personas. Por eso se hace uso de los descriptores más avanzados, como son los tipos HOG.

En una escena de una calle no se puede realizar una segmentación por umbralización debido a la gran variedad que hay de objetos, escenas, iluminación, etc.

La forma de analizar la imagen evitando los problemas anteriores es empleando los descriptores HOG (Histogram of Oriented Gradients). Estos descriptores dividen la imagen en un subconjunto de celdas uniformes, en las cuales representa las orientaciones de los bordes dominantes en cada zona de la imagen.

Se calcula el gradiente de cada pixel y luego se calcula un histograma de cada una de las M orientaciones de los gradientes [2].

3.2. Descriptores Haar

Los descriptores Haar son unos de los más eficientes que se utilizan, debido a su bajo coste computacional. Éstos son los utilizados por el famoso algoritmo de de detección de caras Viola-Jones.

Estas características vienen de lo que se conocía como las wavelets de Haar. Buscan una región en la cual existe una gran diferencia de luminosidad y la subdivide en dos o vareas áreas siendo rectángulos o cuadrados. Estas diferencias de luminosidad entre regiones pueden revelar la existencia de un objeto, por ejemplo, de una cara [3].

3.3. Extracción de descriptores

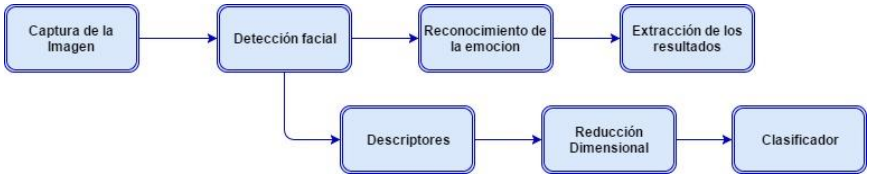


Figura 6: Esquema extracción de descriptores

En esta primera fase se explica cómo utilizarán estas técnicas para extraer las características básicas de una imagen y más adelante poder detectar o identificar una cara dentro de la propia imagen. La extracción de descriptores asume una reducción de dimensionalidad.

Cada una de las técnicas explicadas a continuación extrae de diferente forma los valores característicos más relevantes de una imagen, haciendo más óptimas las técnicas dependiendo de cuál es la finalidad que se requiere de ellas.

En cuanto a las diferentes técnicas de extracción tenemos dos grandes bloques: las técnicas basadas en la apariencia, dentro de las cuales tenemos las lineales y las no lineales, y las técnicas basadas en modelos de 2 dimensiones o 3 dimensiones.

Describiremos brevemente las técnicas más relevantes de cada bloque para luego poder centrarnos en otros tipos de descriptores que serán los que utilicemos durante el proyecto.

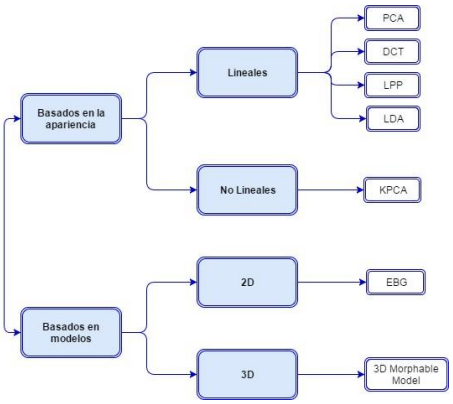


Figura 7: Esquema de técnicas de extracción

3.3.1. Lineal

Para empezar, las técnicas lineales son las técnicas más sencillas de extracción de características.

En primer lugar, tenemos:

- PCA (Principal Component Analysis)

Este método es un algoritmo de reducción dimensional, extrae las características principales de la imagen que mejor la define y las agrupa en diferentes vectores, cada uno de los cuales pertenece a una elipse.

En total este método encuentra 7 vectores de valores incorrelados de la imagen. Al mismo tiempo agrupa los valores de un mismo individuo y los separa del resto de valores, creando así una clase diferenciada para cada individuo [4].

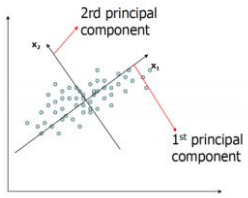


Figura 8 : Imagen PCA extraída de [1]

- DCT (Discrete Cosine Transform)

En este caso tenemos una transformación que crea unas funciones cosenoidales finitas por cada punto, oscilando cada una a diferentes frecuencias, y así representa cada punto como una suma de todos los valores extraídos de esas funciones.

Este método tiene dos grandes ventajas en comparación con PCA, el coste computacional es mucho menor y además el set de imágenes que se necesitan para el entrenamiento puede ser totalmente diferente a las del set de test. Este es un detalle que el método PCA no admite, ambos grupos de imágenes han de ser iguales [4].

3.3.2. Otras Técnicas

A parte de las técnicas lineales que están basadas en la apariencia, existen también las técnicas no lineales, las cuales no son tan utilizadas por su alto coste computacional, de entre muchas, una de ellas es KPCA:

- KPCA (Kernel Principal Component Analysis)

  Esta es una forma de extracción de datos basada en PCA con una pequeña variación, los datos extraídos de una imagen se almacenan en forma de matriz organizándolos según las fórmulas de Kernel.

  Además, en cuanto a la decisión de si un objeto pertenece a una clase u otra se utiliza la técnica del vecino más próximo. Es decir, se mide la distancia a las diferentes clases y se comparan entre ellas, la distancia más pequeña definirá la clase a la cual pertenece [5].

Por otro lado, se encuentran las técnicas que están directamente basados en modelos y nos permiten reconstruir imágenes de una forma mucho más visual que las basadas en la apariencia.

En primer lugar, tenemos las de 2 dimensiones:

- EBG (Elastic Brunch Graph)

  Funciona de forma totalmente diferente, mediante las ondas Gabor extrae los diferentes bordes y texturas de la imagen, los mapea en un entorno grafico diferente y sitúa todos los datos extraídos pudiendo recrear así la imagen de nuevo [6].

Y por último tenemos las técnicas de 3 dimensiones:

- 3D Morphable Model

  Este formato puede generar caras de forma automática a partir de una o más fotografías, pueden recrear la forma y el tamaño de la persona.

  Hay veces que no es necesario que tenga la propia imagen de una persona, sino que a través de unas características básicas o imágenes parciales de ella y mediante su amplia base de datos puede recrear a esa persona [7].

4. Estado del arte: Estados de ánimo

En 1991 un psicólogo llamado Izard quiso cuantificar de alguna forma las diferentes emociones que podía sentir el ser humano. Empezó creando una lista con los diferentes requisitos que debían cumplir las emociones para ser consideradas como básicas:

- Tener un sustrato neural específico y distintivo.

- Tener una expresión o configuración facial específica y distintiva.
- Poseer sentimientos específicos distintivos.
- Derivar de procesos biológicos evolutivos.
- Manifestar propiedades motivacionales y organizativas de funciones adaptativas.

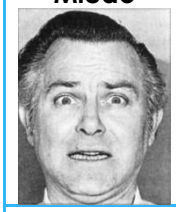
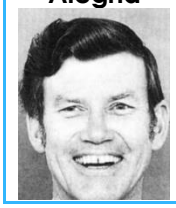
Con todo esto Izard llego a la conclusión de que había 8 emociones que cumplían con cada uno de los requisitos anteriores (Placer, interés, sorpresa, tristeza, ira, asco, miedo y desprecio). Pero finalmente fue Ekman, otro de los autores relevantes en el estudio de la emoción quien determinó según los requisitos anteriores 6 emociones básicas: ira, alegría, asco, tristeza, sorpresa y miedo.

Mucho antes de que en los años 90 estos psicólogos determinaran las características que diferencian cada emoción, otros intelectuales también repararon en el origen genético de algunas expresiones faciales.

Aristóteles dijo "Hay expresiones de la cara características que son observables para acompañar la cólera el miedo, la excitación erótica y todas otras pasiones".

En el siglo XIX fueron Darwin y Guillaume Duchenne, posteriormente Silvan Tomkis en el siglo XX, hasta que llegó Paul Ekman [8].

EMOCIÓN	DEFINICIÓN	EXPRESIÓN
Ira 	El antagonismo hacia una persona o un objeto a menudo se sentía después de que usted siente que ha sido agraviado u ofendido	La reducción de cejas, apretar y estrechar los labios, los ojos mirando fijamente, apretando los parpados inferiores, con menos frecuencia, empujando la mandíbula hacia delante
Asco 	Desagrado intenso o condena causada por algo ofensivo o repulsivo	La reducción de cejas, curvando el labio superior, arrugando la nariz.
Tristeza 	Sentimiento de infelicidad o tristeza	Los parpados caídos, la reducción de las esquinas de la boca, labios fruncidos, los ojos bajos.
Sorpresa 	Sensación de Malestar o sorpresa ante un hecho inesperado	Levantando las cejas altas (que puede causar arrugas en la frente), abriendo los ojos como platos, dejando caer la mandíbula.

Miedo 	Sensación de aprehensión provocada por la percepción de peligro, amenaza o imposición de dolor.	Levantando las cejas/ dibujar las cejas juntas, tensando los parpados inferiores, que se extiende horizontalmente labios, la boca ligeramente abierta.
Alegría 	Agradable sensación de satisfacción y bienestar	Sonriendo- tirando hacia arriba las comisuras de los labios, contrayendo los músculos grandes orbitales alrededor de los ojos.

[8]

Después de decidir que emociones vamos a reconocer y de saber cuáles son las características fisiológicas que las diferencia unas de otras hemos establecido cuáles serán los descriptores que utilizaremos nosotros.

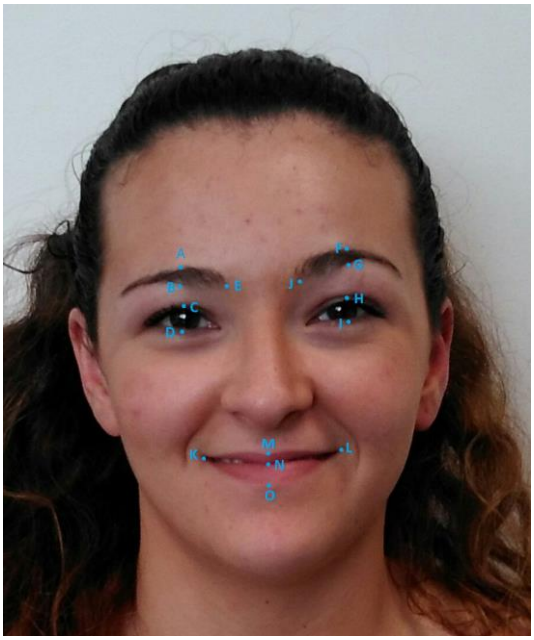


Figura 9: Imagen con los descriptores

Solo escogeremos las distancias más relevantes, aunque marquemos otros puntos para tenerlos como referencia [9].

- Descriptor 1: distancia del ancho del ojo izquierdo, definida entre los puntos C y D.
- Descriptor 2: separación entre el punto más bajo de la ceja (B) y el punto más alto del ojo (C).
- Descriptor 3: distancia del ancho del ojo derecho, definida entro los puntos H y I.
- Descriptor 4: la separación entre el punto más bajo de la ceja izquierda (G) y el más alto del ojo (H).
- Descriptor 5: separación entre ambas cejas, definidas por E y J.
- Descriptor 6: el largo de la boca, definida por K y L, ambos marcados en

las comisuras.
- Descriptor 7: el ancho de la boca, definida por los puntos M y O.
- Descriptor 8: distancia de la comisura izquierda al punto central de la boca, definida por los puntos K y N.
- Descriptor 9: distancia de la comisura derecha al punto central de la boca, definida por los puntos N y L
- Descriptor 10: altura del centro de la boca a la recta que une las comisuras de la boca.
- Descriptor 11: si en el centro de la boca hay blanco entonces tenemos dientes y está sonriendo.
- Descriptor 12: media de los descriptores D2 y D4.

5. Estado del arte: Clasificadores

Existen múltiples tipos de clasificadores de los cuales aquí se van a citar y a explicar los más importantes y que tengan un menor coste computacional.

- SVM
- K-Vecinos
- Redes Neuronales
- Adaboost

SVM

El algoritmo Support Vector Machines, es un clasificador lineal. Utiliza hiperplanos de margen máximo que son más robustos y dan menos errores de clasificación. Este sistema es eficiente en el caso de datos no lineales y resistente a los sobre ajustes [10].

K-Vecinos

Este algoritmo se basa en la distancia que existe entre la muestra de entrada con las diferentes muestras que se tienen ya clasificadas por clases. Dependiendo del número de vecinos en los que se fije determinará la clase de la muestra. Ésta corresponderá a la clase más dominante entre los K vecinos más próximos [1].

Redes Neuronales

Existen diferentes tipos de redes neuronales. En este caso se han investigado las de reconocimiento de patrones, ya que será el tipo de clasificador que se utilizará para clasificar las diferentes emociones que tengamos. Las redes neuronales artificiales tienen un número de neuronas establecido, que puede variar a elección del usuario y hace que el modelo se ajuste mejor a lo deseado. No obstante, no por tener mayor número de neuronas es mejor el resultado obtenido.

En nuestro caso se explicará más adelante el número de neuronas que gastamos para hacer las diferentes pruebas y se comprobará como, un mayor número de neuronas no significa un mejor resultado [11].

AdaBoost

La clasificación de Boosting se utiliza para aumentar el rendimiento de los árboles de decisión. Se emplea esta técnica porque mejora en las decisiones difíciles, es decir conforme pasa las por las diferentes etapas de aprendizaje, descarta los errores para luego posteriormente centrarse en esos y tener una mejor tasa de acierto [12].

6. Algoritmo Viola-Jones

Viola-Jones es un algoritmo que se ha definido como uno de los detectores faciales más eficientes que existen en este momento, gracias a su bajo coste computacional y su alta tasa de acierto. Este algoritmo fue creado en 2001 por Paul Viola y Michael Jones. En un principio fue creado para detectar objetos en tiempo real de forma competitiva. Utilizado para la detección de personas, aunque también se puede entrenar para detectar otro tipo de objetos.

En la Figura 10 podemos ver el recorrido de una muestra imagen cuando se somete al algoritmo de detección facial de Viola-Jones.

Figura 10 : Esquema algoritmo Viola-Jones

Este algoritmo tiene un funcionamiento en cascada mediante la concatenación de varios clasificadores débiles. Primero como en cualquier imagen se extraen los descriptores característicos, en este caso son de tipo Haar, y posteriormente se clasifican esos descriptores con un clasificador llamado AdaBoost o "Boosting", que divide todas las imágenes procesadas en dos tipos de imagen, clase 1 "Cara" y clase 2 "No cara".

En primer lugar, hace una selección de las diferentes características Haar de cada imagen (o frame de un video), y después va evaluando sector a sector el valor de la suma de los pixeles. Si existen una gran diferencia entre ambas secciones entonces dicha imagen pasará a la siguiente fase.

Dentro de esta primera etapa pasa por diferentes fases en las que se divide la imagen en diferentes regiones y se sigue evaluando el valor de la suma de todos los valores de los pixeles de esa región con la región contigua.

Como se muestra en la siguiente imagen, las diferentes formas de dividir la imagen.

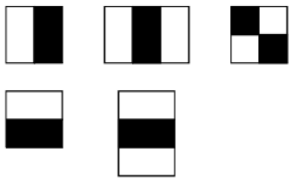


Figura 11: Segmentaciones de Viola-Jones para cambios de iluminación

En la segunda etapa se hace una representación de la imagen integral que incluye en la posición correspondiente (x, y), el valor de la suma de los pixeles por encima y por la izquierda de dicha posición.

Como tercera etapa entramos en la formación de un clasificador AdaBoost. Estos clasificadores son del tipo impulsado. Se crea un árbol de decisión para identificar la clase a la que pertenecen siendo 1 el valor positivo y -1 el negativo, identificando así que no pertenece a la clase.

La gran ventaja de este proceso es que incluye en su árbol de decisión la tasa de error, pudiendo así reajustarse en las siguientes decisiones.

Y por último llegamos a los clasificadores en cascada. Si se añaden X número de clasificadores como los explicados en el párrafo anterior la tasa de acierto aumenta con cada clasificador añadido.

En definitiva, podemos concluir que Viola-Jones no es solo una técnica de detección de caras o no caras, sino que es un algoritmo el cual goza de dos procesos bien diferenciados: la detección y la clasificación, en la que podemos ajustar su umbral de decisión para que cometa una menor tasa de error.

7. Bases de Datos

Después de un amplio estudio en cuanto a la base de datos que utilizaríamos se eligió la que más sujetos tenían y aportaba unas mejores condiciones.

NOMBRE	FORMATO	MODALIDAD	TAMAÑO	Nº SUJETOS	EMOCIONES	ACCESO	REF	ENCONTRADA
Humaine Database	Video	Facial, Corporal	40 sesiones	40	6 emociones	Disponible	[13]	No Registro
Belfast Naturalistic Database	Video	Facial, Corporal	125 sesiones	125	6 emociones	Disponible	[14]	No Registro
SAL Database	Video	Facial	80 sesiones	20	1 dimensión	Bajo demanda	[13]	No Registro
Indian Institute of Technology Kanpur Database	Imagen	Facial	240 imágenes	40	6 emociones	Disponible	[15]	Si
The Yale Face Database	Imagen	Facial	165 imágenes	15	3 emociones	Bajo demanda	[16]	No Registro
Caltech Frontal Face DB	Imagen	Facial	450	27	¿? No esta descrito	Disponible	[17]	Si
HumanScan BioID Face BD	Imagen	Facial	1500 imágenes	23	¿? No esta descrito	Disponible	[18]	Si
The Database of Faces AT&T	Imagen	Facial	400 imágenes	40	¿? No esta descrito	Disponible	[19]	Si

Semaine	Video	Facial, corporal	50 sesiones	20	6 emociones	Disponible	[20]	No Registro
Cohn Kanade DB	Video	Facial, Corporal	500 sesiones	100	6 emociones	Bajo demanda	[21]	Si
The Japanese Female expresión	Imagen	Facial	70 imágenes	10	6 emociones	Bajo Demanda	[22]	Si
Extended Cohn Database	Video	Facial, Corporal	610 sesiones	127	6 emociones	Bajo Demanda	[20]	Si
Karolinska Directed Emotional Faces	Imagen	Facial	4900 Imágenes	70	7 emociones	Bajo Demanda	[23]	Si

Figura 12 : Tabla estudio de las Bases de Datos [24]

Finalmente, la Base de datos elegida, es la última Karolinska Directed Emotional Faces, ya que está formada por una gran cantidad de imágenes y sujetos.

Hemos tenido que escoger las imágenes necesarias, ya que nuestro código estará implementado de forma que detecte las caras y las emociones en imágenes frontales. Por eso se ha reducido considerablemente el número de imágenes por sujeto ya que ellos tenían imágenes de 5 ángulos diferentes por cada emoción y sujeto.

De esta forma ahora contamos con 140 imágenes por cada emoción.

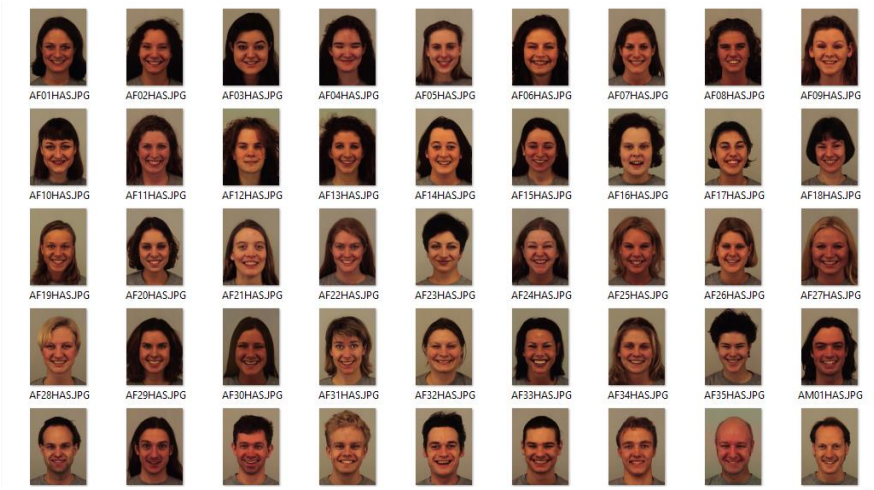


Figura 13 : Ejemplo de la base de datos

8. División del proyecto por fases

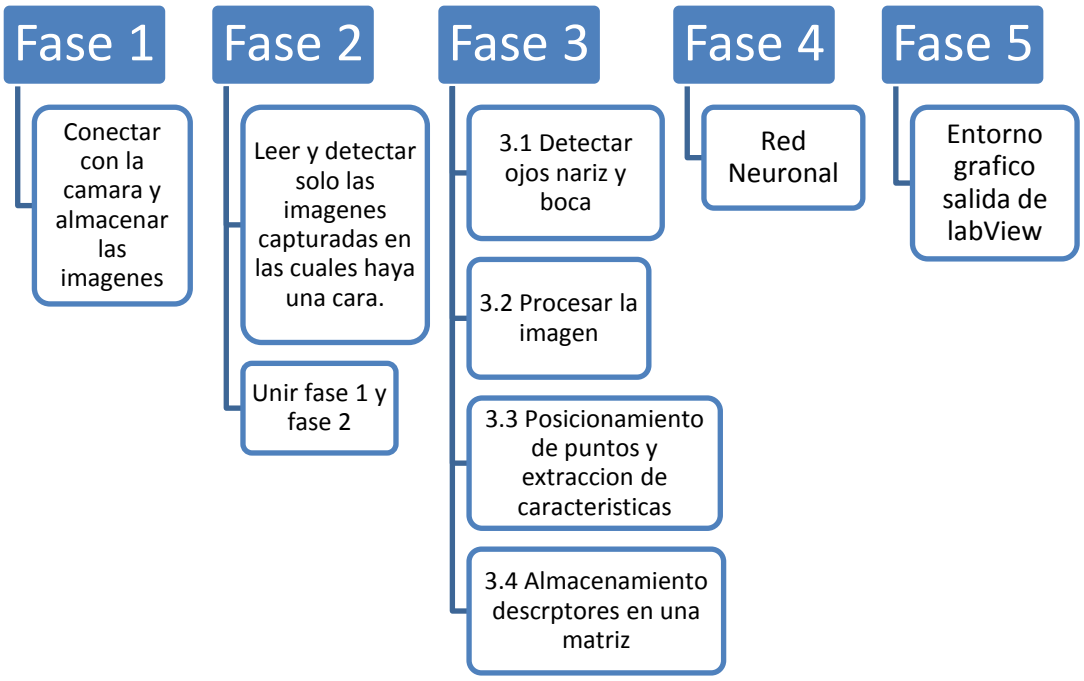


Figura 14 : División de las fases del proyecto

Después de realizar todo el proceso de búsqueda de información y decidir cuales iban a ser los descriptores más óptimos, se comenzó con el proceso de creación del código.

Este código lo hemos dividido en diferentes fases, las cuales se irán explicando poco a poco, para tener un concepto claro de la función completa del programa.

8.1. Fase 1

En primer lugar, nos enfrentamos al reto de conectar la cámara, para ello se ha realizado un pequeño SubVi con el entorno de programación de LabView.

Este SubVi, fue creado con la herramienta de LabView Vision Acquisition Express, (esta herramienta pertenece al bloque de funciones de Vision and Motion que fue instalada exclusivamente para este proyecto).

El módulo Acquisittion express, nos permitió en 5 minutos configurar, una entrada para la cámara, dándole simplemente el nombre de esta.

En la Figura 15 mostrada a continuación se ve el diagrama de bloques, es decir la programación de este subVi. Junto a la cual encontraremos una explicación de cada uno de los objetos que hay.

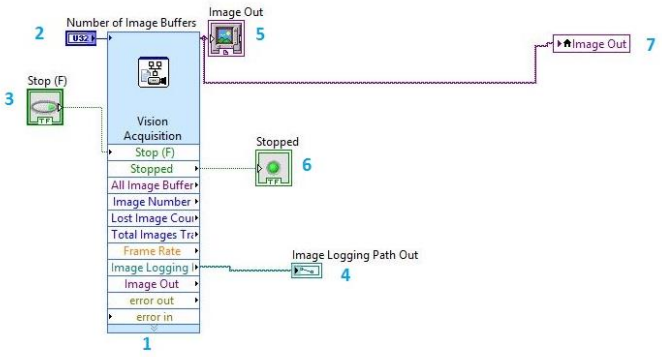


Figura 15 : Diagrama de bloques del Vi Adquisición de imágenes

En primer lugar, en el numero 1 tenemos el bloque del Acquisittion Expres. Podemos ver que nos permite modificar y programar múltiples variables. Pero en este caso nosotros solo modificaremos algunas de ellas.

En el número 2 se observa que se le añadió un control. Este es un campo vacío en el que se puede pedir un número concreto de imágenes que se almacenan en el buffer.

En el número 3 contamos con un botón para poder detener todos los procesos de grabar y guardar las imágenes.

En el número 4 tenemos un indicador, el cual nos muestra la ruta en la que se están guardando las imágenes. Ésta no es modificable ya que se configuró al principio de crear el proyecto, y no nos interesa que el cliente final pueda modificar esa carpeta.

En el número 5 contamos con un display que nos muestra la imagen de la cámara en tiempo real. Esto es interesante ya que así el usuario podrá saber en qué momentos está o no grabando y cuál es el campo de visión de la cámara.

En el numero 6 hemos colocado un led que indicará si el programa se encuentra parado o no.

Por último, contamos con el número 7 que es una variable local que se crea para poder gastar esas mismas funciones de forma externa.

Este es el primer SubVi que utilizaremos a lo largo del proyecto por eso debemos crear este tipo de variables locales externas para poder gastar estos mismos botones en los sucesivos SubVis que creemos a partir de este.

Por último, mostramos una imagen de como se ha configurado el panel frontal del SubVi, ya que este panel será el que vera el usuario final.

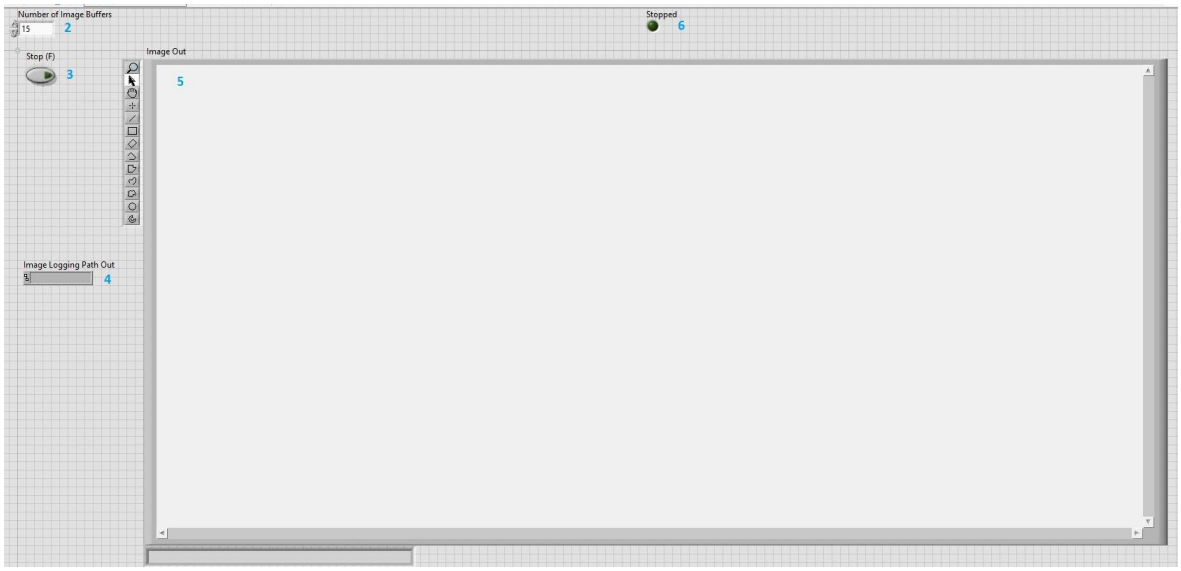


Figura 16 : Panel frontal del Vi Adquisición de imágenes

Aquí también se pueden encontrar los números haciendo referencia a las explicaciones anteriores.

Durante este proceso nos encontramos con un problema. No podíamos regular desde LabView el número de Frames por segundo que capturaba la cámara y almacenaba en la carpeta. Por eso nos vimos obligados a modificarlo desde la configuración propia de la cámara en las páginas que nos proporciona el fabricante, dejando así una tasa de 2 imágenes por segundo.

Es un valor pequeño pero puesto que el procesamiento que se debe hacer luego de dichas imágenes es un proceso costoso computacionalmente, era necesario tener una tasa de captura más pequeña de lo normal.

8.2. Fase 2

Una vez tenemos las imágenes almacenadas en nuestra carpeta de "prueba express", vamos a realizar el primero de muchos procesamientos para cada imagen.

Como se ha citado anteriormente hemos utilizado el algoritmo de Viola-Jones que está ya implementado en Matlab para el primer análisis de la imagen, en busca de poder dividirlas en dos clases diferentes, clase 1 "CARAS", clase 2 "NO CARAS".

Gastamos Matlab y LabView indistintamente ya que LabView es compatible con Matlab.

```
while length(lee_archivos) > 0

archivo = lee_archivos(k).name; %Obtiene el nombre de los archivos
nombre = 'C:\Users\rabargon\Desktop\prueba express\'; %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 = 6;
BB = step(FDetect,I); % Bounding Box coordenadas de la imagen don existe una cara
mat_image=cell(1,numel(BB,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 par
```

Figura 17: Código de Matlab para leer imágenes y detectar la cara

Lo primero de este código es hacer un bucle, el cual no acabará hasta que la carpeta en la cual se van almacenando las imágenes se quede vacía.

En primer lugar, se hace una lista con los documentos que hay en la carpeta "prueba express" y se accede al primero de ellos.

Se lee la imagen y con la función FDetect = vision.CascadeObjectDetector extraigo unos valores llamados BB, los cuales hacen referencia al cuadrado de la imagen en el cual detectan una cara.

Los valores Bounding Box es una matriz de vectores, en el caso de que encuentre más de un objeto, y en caso contrario un vector con 4 valores de coordenadas [x y weight hide]. Gracias a estas coordenadas podemos recortar la imagen a ese tamaño y quedarnos únicamente con la cara.

Tuvimos algunos problemas con el umbral utilizado para la detección ya que se colaban caras que en realidad no lo eran. Así que aumentamos el umbral (MergeThreshold) para que fuera más estricto el clasificador y no nos diera falsos positivos.

Numero de imágenes totales	Umbral	Caras correctas	Caras incorrectas
70	4	62	7
70	6	60	1
70	7	59	1

Figura 18 : Tabla umbral detección de caras

Finalmente decidimos aumentar a 6 el umbral, ya que preferíamos que cogiera un falso positivo y tener mayor número de imágenes correctas, a que nos siguiera dando un falso positivo, pero ir reduciendo el número de Caras que nos detectaba.

A continuación, se observa la diferencia entre la imagen original y la imagen recortada con el Viola-Jones.

Figura 19: Lectura de la imagen y Recorte con FDetect

Cuando se ha procesado la imagen, si encuentra una cara se guarda en otra carpeta, y luego se elimina dicha imagen de la carpeta principal. Si no encuentra ninguna cara la imagen también es eliminada.

Este código finaliza cuando la carpeta de "prueba Express" se queda vacía.

Todo este código el cual se encuentra completo en el Anexo 1, se introduce en un módulo de LabView, llamado Matlab Script, para crear así nuestro siguiente SubVi, el cual mostramos a continuación. En este caso este SubVi solo tiene la ventana de diagrama de bloques, ya que no tiene ninguna salida que queramos que se muestre al usuario final.

No obstante, sí que tiene una salida que es un flag que sirve para indicar que el código ha acabado y la carpeta prueba imágenes está vacía.

Figura 20 : Diagrama de bloques Vi de Viola-Jones

8.2.1. Unificación Fase 1 y Fase 2

En la siguiente imagen vamos a ver como unimos los dos SubVis anteriores en uno nuevo, el cual será una fase beta del entorno grafico que verán los usuarios.

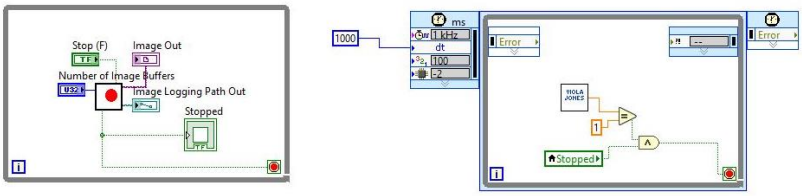


Figura 21: Diagrama de bloques Vi principal

En esta figura se ve claramente en la parte izquierda un "While Loop" que engloba el primer SubVi creado para la captura de imágenes, conectado al propio botón de stop del bucle para poder sacar una variable local externa.

En el lado de la derecha hay un "Timed Loop" que tiene un temporizador. Este temporizador hará que se inicie 1000ms más tarde el SubVi de Viola-Jones, ya que si se inicia al mismo tiempo da error porque la carpeta de "prueba express" aún está vacía.

Además, se llama a la variable local del bucle contiguo "stopped" comparándola así con el flag del código de Matlab. Esto quiere decir que el nuevo SubVi finalizara cuando el flag esté a uno y previamente le hayamos dado al botón de stop.

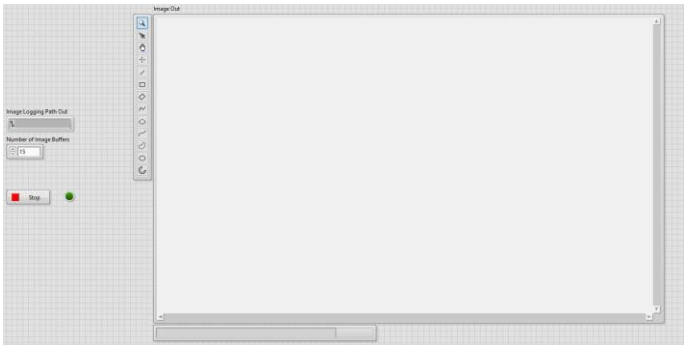


Figura 22 : Panel frontal Vi principal

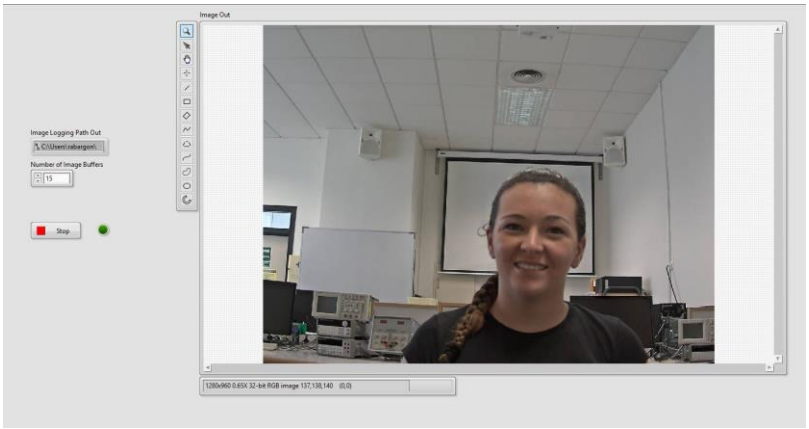


Figura 23 : Panel frontal en funcionamiento

8.3. Fase 3

Una vez se ha detectado la cara deberemos de segmentar la imagen para calcular los descriptores detallados anteriormente. Se ha decidido que la forma más óptima es detectando los ojos, la nariz y la boca.

8.3.1. Fase 3.1

El código utilizado para detectar la nariz, la boca y los ojos, es también una variante de Viola-Jones, implementada específicamente para la detección de dichos objetos.

A continuación, vemos una parte del código para la detección de la nariz.

```
NoseDetect = vision.CascadeObjectDetector('Nose','MergeThreshold',16);
     for iN = 1:size(BB,1)
        BBnose=step(NoseDetect,mat_image{i});
        figure(iN);
        hold on
        rectangle('Position',BBnose(1,:),'LineWidth',1.5,'LineStyle','-','EdgeColor','b');
        Nose=imcrop(mat_image{i},BBnose(1,:));

     end
```

Figura 24 : Código detector de Nariz

Se decidió que la detección de nariz se hiciera antes que la de boca y ojos, ya que la tasa de error que tiene es mucho menor. Es decir, el número de imágenes en las cuales detecta la nariz y lo hace de forma correcta en nuestro caso es de casi un 100%.

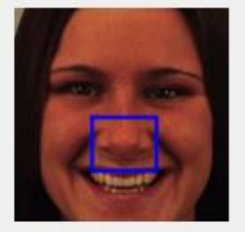


Figura 25 : Ejemplo de la detección de nariz

De esta forma, tenemos un apoyo más para el detector de la boca, ya que este detector sí que tenía una tasa de error mayor puesto que marcaba algún ojo como boca.

Con una serie de bucles se obligó a que el recuadro de la boca quedara siempre por debajo del de la nariz. Además, se aumentó de forma considerable el umbral de detección de la boca hasta 80.

Estos bucles se pueden observar en la siguiente figura.

```
MouthDetect = vision.CascadeObjectDetector('Mouth');
MouthDetect.MergeThreshold= 20;
   for iM = 1:size(BB,1)
      BBmouth=step(MouthDetect,mat_image{i});
      for jM= 1: size(BBmouth,1)
          if (BBmouth(jM,2) > BBnose(1,2))
               BBmouth2= [ (BBmouth(:,1)) (BBmouth(:,2)+3) (BBmouth(:,3)) (BBmouth(:,4)-3)];
               rectangle('Position',BBmouth2(jM,:),'LineWidth',1,'LineStyle','-','EdgeColor','m');
               title('Mouth Detection');
               Mouth=imcrop(mat_image{i},BBmouth2(jM,:));
               x_Mouth = BBmouth2(jM,1);
               y_Mouth = BBmouth2(jM,2);
               H_Mouth = BBmouth2(jM,4);
               W_Mouth = BBmouth2(jM,3);
```

Figura 26: Código detección de boca

En los bucles se especifica que, si la segunda coordenada del vector Bounding Box de la boca es mayor que la segunda coordenada del vector correspondiente a la nariz, dibujará un rectángulo en la posición de la boca. Para asegurarnos que el detector ya implementado no nos recorta las comisuras de la boca, hemos incrementado de forma manual el tamaño de dicho rectángulo, en los pixeles que hemos estimado después de una serie de pruebas.

Posteriormente nos hemos almacenado cada coordenada del vector llamado BBmouth2 para poder utilizarlas más adelante en el marcaje de descriptores por puntos.

Aquí se muestra una detección correcta de la boca, y un error de la misma.

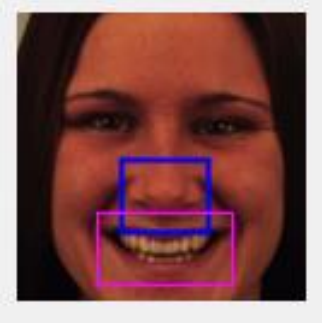


Figura 27: Ejemplo detección de boca

Por último, tenemos el detector de los ojos, en este caso no hemos necesitado ponerle ningún tipo de condición ya que tiene una tasa de acierto de detección casi del 100%.

```
EyeDetect = vision.CascadeObjectDetector('EyePairBig');
    for iE = 1:size(BB,1)
        BBeye=step(EyeDetect,mat_image{i});
        BBeye2 = [(BBeye(:,1)-2) (BBeye(:,2)-10) (BBeye(:,3)) (BBeye(:,4)+10)];
        figure(iE);
        rectangle('Position',BBeye2(1,:),'LineWidth',1.5,'LineStyle','-','EdgeColor','b');
        imshow (BBeye2);
        title('Eye Detection');
        Eye=imcrop(mat_image{i},BBeye2(1,:));
        x_Eye = BBeye2(:,1);
        y_Eye = BBeye2(:,2);
        H_Eye = BBeye2(:,4);
        W_Eye = BBeye2(:,3);
```

Figura 28: Código detector de ojos

Matlab dispone de tres tipos de funciones para la detección de ojos en el CascadeObject. Se puede detectar el ojo izquierdo, por un lado, el ojo derecho por otro, o los dos pares de ojos, que es el que hemos escogido nosotros.

La elección de detectar el par de ojos viene dada porque al mismo tiempo que se detectan los ojos queremos detectar las cejas de los individuos. Según la lista de descriptores que realizamos en el punto 7, a nosotros nos interesa la distancia del ojo a la ceja como uno de los descriptores.

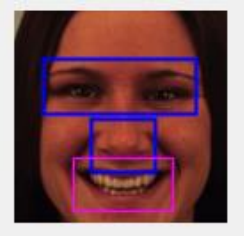


Figura 29 : Ejemplo detector de ojos

Por este mismo motivo obligamos a que el Bounding Box de los ojos sea mucho más grande, para así capturar la ceja al mismo tiempo.

8.3.2. Fase 3.2

En este punto se centra gran parte del grueso del trabajo, debido a que, si se hace un mal tratamiento de la imagen, perderíamos mucha información.

Además, es un proceso delicado, puesto que lo tenemos que adaptar al mayor número de imágenes posibles.

Finalmente se ha conseguido adaptar el programa para que calcule correctamente las distancias de los descriptores necesarias en más de 90% de imágenes de la base de datos.

Emoción	Porcentaje
Alegría	97,8%
Asco	93,53%
Ira	96,42 %
Miedo	98,5%
Neutro	94,24%
Tristeza	94,96%
Sorpresa	95,67%

Figura 30 : Tabla porcentaje imágenes en la base de datos

El mayor problema lo teníamos en la umbralización, binarización, la dilatación y la erosión de la imagen etc…

El proceso y los valores de umbralización que iban bien para una imagen para las siguientes se quedaban demasiado distorsionados y no funcionaba.

A continuación, se explicará todo el proceso que se lleva a cabo con cada una de las segmentaciones de la imagen. Comenzaremos en el orden que las tenemos en el código, en primer lugar, tenemos la boca.

```
Mouth= rgb2gray(Mouth);
F = adaptthresh(Mouth,0.45,'ForegroundPolarity','dark','NeighborhoodSize',29);
BW_Mouth = not(imbinarize(Mouth,F));
BW_Mouth = bwpropfilt(BW_Mouth,'area', [100 inf]);
[L Ne]=bwlabel(BW_Mouth);
propied= regionprops(L);
b =[propied.BoundingBox];
x_Mouth = b(:,1);
y_Mouth = b(:,2);
H_Mouth = b(:,4);
W_Mouth = b(:,3);
nombre2 = 'Test\';
imwrite(BW_Mouth,[nombre2 'mouth' num2str(iM) '.jpg'],'jpg');
imshow(BW_Mouth);
```

Figura 31: Código procesamiento del segmento de la boca

En primer lugar, pasamos el recorte de la boca, a escala de grises (Figura 32), para posteriormente poder hacer una umbralización adaptativa de la imagen.

Figura 32 : Boca en escala de Grises

En este caso la umbralización adaptativa la hacemos hacia el oscuro, con una sensibilidad del 0.45 y fijándonos en los 29 vecinos de cada pixel.

Después hacemos la binarización en función del nivel calculado anteriormente, y nos guardaremos el inverso de la binarización.

Figura 33 : Binarización de la boca

Para poder encontrar la boca dentro del Bounding Box del detector hacemos un contador de objetos de los cuales extraemos los propios Bounding Box de cada objeto, y almacenamos sus coordenadas.

En el caso de los ojos se sigue el mismo proceso, con algunas modificaciones.

```
Eye= rgb2gray (Eye);
imshow(Eye);
T = adaptthresh(Eye,0.45,'ForegroundPolarity','dark','NeighborhoodSize',9);
BW_Eye = not(imbinarize(Eye,T));
imshow (BW_Eye);

BW_Eye2 = bwpropfilt(BW_Eye,'area', [30 800]);
IM2 = bwmorph(BW_Eye2,'thicken');
IM3 = bwmorph(IM2,'thin');
imshow (BW_Eye2);
imshow(IM3);
nombre2 = 'Test\';
imwrite(IM3,[nombre2 'Eye' num2str(iE) '.jpg'],'jpg');
hold off;
```

Figura 34 : Código procesamiento de los ojos

En las primeras líneas tenemos el mismo procedimiento que el de la boca, a diferencia que el umbral adaptativo se fija solo en los 9 vecinos más cercanos.

En este caso añadimos un filtro por área para que nos elimine todos los objetos con un número de píxeles menor a 30. Esto nos permite eliminar pequeñas áreas que se podrían confundir como otros objetos. Para poder encontrar los puntos que se han nombrado en el capítulo 5 debemos exagerar los objetos y definir más sus bordes.

Para ello dilatamos la imagen gastando la opción 'thicken', ya que ésta añadirá una fila de pixeles en el borde del objeto siempre y cuando no haga que se junten dos objetos previamente inconexos. Así evitaremos que se junte el ojo con la ceja. Y después erosionamos la imagen con la opción 'Thin'.

Este proceso se está gastando con los dos ojos al mismo tiempo. Ahora debemos recortar y dejar el ojo y la ceja izquierda por un lado y el ojo y la ceja derecha por otro lado.

```
    Eye= rgb2gray (Eye);
    imshow(Eye);
    T = adaptthresh(Eye,0.45,'ForegroundPolarity','dark','NeighborhoodSize',9);
    BW_Eye = not(imbinarize(Eye,T));
    imshow (BW_Eye);

    BW_Eye2 = bwpropfilt(BW_Eye,'area', [30 800]);
    IM2 = bwmorph(BW_Eye2,'thicken');
    IM3 = bwmorph(IM2,'thin');
    imshow (BW_Eye2);
    imshow(IM3);
    nombre2 = 'Test\';
    imwrite(IM3,[nombre2 'Eye' num2str(iE) '.jpg'],'jpg');
    hold off;
end
```

Figura 35 : Código procesamiento de cada ojo por separado

En primer lugar, recortamos la imagen por la mitad y hacemos un contador de objetos, ordenándolos y quedándonos con el objeto cuyo centroide tiene su segunda coordenada mayor que los demás, es decir será el objeto que este arriba del todo, el primero.

Una vez se han etiquetado los objetos e identificado sus posiciones los etiquetamos como ceja y ojo y empezamos a trabajar individualmente con cada uno de ellos.

Se ha añadido una dilatación para todos los objetos que etiquetamos como ojo, haciéndolo con un elemento estructurante que es una línea de un pixel de grosor y cinco pixeles de largo.

Esto nos permite unir los objetos que previamente estaban desconectados entre sí pero en el eje horizontal. Esto se ha realizado porque a veces el ojo se separaba en dos objetos debido al blanco de los ojos. Trataba la imagen de forma que detectaba un cambio brusco de iluminación entre el iris y el blanco del ojo y lo dividía entre sí como dos objetos.

Figura 36: Dilatación de ambos ojos

Una vez tenemos el ojo procesado se repite la operación previa de etiquetado de los objetos.

De esta forma conseguimos tener en la mayoría de los casos el ojo y la ceja por separado.

Figura 37 : Ceja y ojo Derecha

Durante el desarrollo del proyecto nos dimos cuenta que no siempre detectaba la ceja en la posición que tocaba. En el caso de la ira sobre todo la ceja y el ojo están tan juntos que detecta un único objeto.

Por ello, se ha creado una condición de que en el caso que solo detecte un objeto, ése será el ojo y la distancia de la ceja a éste será 0.

Figura 38 : Ejemplo Ceja y Ojo unidos

8.3.3. Fase 3.3

Una vez tenemos identificados la boca, el ojo y la ceja, procedemos a la búsqueda de los puntos característicos de los ojos y la boca, para calcular las distancias necesarias.

En este caso, el marcaje de los puntos lo hemos hecho de varias formas diferentes dependiendo de cuál era la información que queríamos.

En la figura 39 podemos ver el primer procedimiento que se utiliza para la extracción de los puntos.

```
[HC WC]= find (L_L==ind_L(1));

for Ci=1:H_EyeG
    Cj= (WC(1,1)+round(W_Eye/2));
    if (Left_Eye(Ci,Cj) ==1)
        puntoCx = Cj;
        puntoCy = Ci;
        imshow(Left_Eye);hold on; plot(puntoCx,puntoCy, '+','MarkerSize',10)
        break
    end
end
```

Figura 39 : Código extracción del punto C

En este primer caso extraemos el tamaño del objeto y comenzamos recorrer todas las filas de la matriz dejando la columna fija a la central. En este caso queremos que nos marque el primer punto del ojo en vertical.

Figura 40 : Primer punto del ojo Punto C

En el momento que encuentre el primer pixel igual a 1, guardamos las posiciones de este pixel y la almacenamos como coordenada (x, y) de punto C, en este caso.

El break que se encuentra al final es para que cuando encuentre el primer punto

En la figura 41 tenemos otro de los procedimientos que se ha utilizado para localizar en este caso el borde inferior del ojo.

```
for Di= 1: H_EyeG
    Dj= (WC(1,1)+ round(W_Eye/2));
    if (Left_Eye(Di,Dj) ==1)
        puntoDx = Dj;
        puntoDy = Di;
        imshow(Left_Eye);hold on; plot(puntoDx,puntoDy, '+','MarkerSize',10)
    end
end
```

Figura 41 : Código extracción punto D

En este caso el código es exactamente igual menos el break, lo que hace que almacene las coordenadas del ultimo pixel que encuentre con valor 1 en la vertical.

Figura 42: Último punto del ojo Punto D

También tenemos un tercer procedimiento que se puede ver en la figura 43, totalmente diferente a los anteriores.

```
[WA1 HA1 Ei]= find(WA,1,'last');
[WA2 HA2 Ej]= find(HA,1,'last');
puntoEx = Ei;
puntoEy = Ej;
imshow(Eyebrow_Left);hold on; plot(puntoEx,puntoEy, '+','MarkerSize',10)
```

Figura 43 : Código extracción punto E

Extraemos directamente las ultimas coordenadas del objeto, y las almacenamos sin ningún tipo de tratamiento, localizando así el borde derecho de la ceja izquierda.

Y como ultimo procedimiento tenemos el que aparece en la figura 44.

```
[WJ1 HJ1 Ji]= find(WF,1,'first');
[WJ2 HJ2 Jj]= find(HF,1,'first');
puntoJx = Ji;
puntoJy = Jj;
imshow(Eyebrow_Right);hold on; plot(puntoJx,puntoJy, '+','MarkerSize',10)
```

Figura 44 : Código extracción Punto J

Aquí tenemos el inverso del anterior, en este caso se localiza el borde izquierdo de la ceja derecha, para ello buscamos el primer punto del objeto y almacenamos esas coordenadas.

Una vez se han extraído todos los puntos que necesitábamos y se han guardado internamente sus coordenadas, entonces calculamos la distancia que existe entre ellos, con la siguiente formula.

$$D = \sqrt{(x - x_o)^2 + (y - y_o)^2}$$

Este procedimiento de extracción de los puntos es el que se ha llevado a cabo para los descriptores del 1 al 9, explicados en el apartado 4.

En cuanto al descriptor número 10 se ha utilizado la fórmula de la altura del triángulo [25]. Para calcular la distancia del centro de la boca a la recta que une las comisuras.

Los lados del triángulo son las distancias que se describen entre los puntos K-N (Descriptor 8), los puntos N – L (Descriptor 9) y los puntos K-L (Descriptor 6).

```
hc=(1/2)* (sqrt(2*(D9^2 + D8^2) - D6^2)) ;
D10= hc;
```

Figura 45: Código áltura del centro de la boca a la recta de las comisuras

El descriptor numero 11 es de tipo binario, se ha procesado el segmento de la boca de forma diferente, en vez de umbralizar al "dark" (oscuro), se ha umbralizado al "bright" (claro). Y se ha buscado el valor del pixel central para determinar si hay diente o no hay un diente, en el caso afirmativo el valor del descriptor será un 1 y en caso negativo será 0.

```
F2 = adaptthresh(Mouth,0.45,'ForegroundPolarity','bright','NeighborhoodSize',5);
BW_Mouth2 = imbinarize(Mouth,F2);
imshow(BW_Mouth2)
x_smile = puntoNx+5;
y_smile = puntoNy+5;
area_smile=  imcrop (BW_Mouth2,[x_smile y_smile 10 10]);
[ar Nearea]=bwlabel(area_smile);
prop= regionprops(ar);
area=[prop.Area];
area_smile=sum(area);

if (area_smile >= 1)
    Teeth=1;
else
    Teeth=0;
end
D11= Teeth;
```

Figura 46 : Código detector de diente

Para finaliza el ultimo descriptor gastado es una media de dos de los descriptores calculados previamente, la distancia de la ceja al ojo en ambos casos, es decir la media del Descriptor 2 y del Descriptor 4.

```
Average = (D2+D4)/2;
```

Figura 47: Código de la media

8.3.3.1. Pruebas y soluciones

Durante el desarrollo de esta fase del proyecto se fueron realizando diferentes pruebas para comprobar que el código realizado marcaba bien los puntos en la imagen y calculaba bien las distancias.

Desde la primera versión que se creó del programa nos encontramos con una gran cantidad de errores que hemos solventado. Ahora se va a redactar una lista de los errores más comunes que obteníamos. Y posteriormente se explicará el motivo del error y como se solucionó.

En la primera prueba.

Con 35 imágenes de Miedo, solo funcionaba correctamente con 9 imágenes.

ERRORES	Nº DE SUJETOS	SOLUCIÓN
No le detecta la boca.	5	Reducir el MergeThreshold del detector de la boca. De 60 a 20, ya que era extremadamente restrictivo y obviaba muchas veces la boca. En un principio se elevó tanto porque cogía otras zonas de la cara como boca.

No encuentra la ceja porque el área es mayor que el rango que estamos marcándole [60 150].	8	Cambiar el rango, aunque no se aumenta para que entren todas las anteriores áreas porque si no en algún caso me lo confundirá con el ojo. Aun así, aumentamos el rango de [60 195]
Detecta el área de la ceja como la más grande e invierte los objetos, la ceja es el ojo y viceversa.	2	Intentar que cumpla dos condiciones, que en el caso de la ceja me coja el área más pequeña y el objeto que este arriba del todo. En el caso del ojo será al revés, que tenga el área más grande, ya que el ojo suele ser más grande que la ceja, y que se encuentre en segunda posición el objeto por debajo de la ceja.
Se unen la ceja y el ojo.	3	Erosionar la imagen y cambiar el tratamiento que se hace de ésta.
La binarización de la boca no es correcta y hay un espacio entre el labio superior y el inferior.	3	No hay modificaciones
Varios errores diferentes.	5	No hay modificaciones

Figura 48 : Tabla errores Primera prueba

Segunda Prueba

Una vez resolvimos la gran mayoría de estos errores volvimos a probar el código con 40 imágenes de la carpeta de miedo (20 mujeres y 20 hombres). Solo conseguimos que el código funcionara en 10 imágenes. En las demás nos daba errores distintos que el anterior, pero en las mismas imágenes.

ERRORES	Nº DE SUJETOS	SOLUCIÓN
La binarización de la boca no es correcta y hay un espacio entre el labio superior y el inferior.	4	Se puede variar el nivel de umbralización o en este caso cambiamos el bucle para detectar los puntos de la imagen.
No sé qué área está cogiendo.	2	Dado el número de imágenes nos podemos permitir el error.
Están unidas ceja y ojo.	3	En caso de que solo exista un objeto, crear un condicional para que tanto si hay dos objetos como si hay uno que encuentre sus puntos característicos.
Coge el ojo como ceja y viceversa.	9	Al haber aumentado el rango para que me detectara la ceja en la prueba anterior ahora nos encontramos con algún caso en el que el área de la ceja es mayor que la del ojo y ésta la etiqueta como ceja. Puesto que en un principio el ojo se etiquetaba al encontrar el área más grande.
la altura de la ceja no está bien calculada.	3	En un principio los puntos de la ceja se calculaban de forma muy mecánica, obligando a encontrar los puntos a una altura de (1/4) empezando por la parte superior.
Aparecen muchos objetos pequeñas	2	La solución fue poner un filtro para eliminar todas las áreas que fueran menores de 30 pixeles.
Elimina la ceja porque está por debajo del rango.	5	Volver a cambiar el rango del área de la ceja o poner 2 condiciones para etiquetar la ceja que sea el área más pequeña y además que esté en la primera posición
No le detecta la boca.	1	Reducir aún más el MergeThreshold del detector de boca.

Figura 49: Tabla errores Segunda Prueba

Tercera Prueba

Después de resolver estos errores volvimos a probar el código con 40 imágenes de la carpeta de miedo (20 mujeres y 20 hombres). Se aumentó considerablemente el número de acierto a 29, pero seguíamos teniendo errores que teníamos que resolver.

ERRORES	Nº DE SUJETOS	SOLUCIÓN
No detecta un punto del ojo porque están juntos la ceja y el ojo.	4	Crear un if para que detecte solamente un objeto, y la distancia de la ceja al ojo sea siempre 0.
No detecta la boca.	2	La solución es disminuir aún más el MergeThreshold del detector de la boca, pero no es una opción viable, ya que si se disminuye más detecta otras zonas de la cara como boca. Y el error seria mayor.
El primer objeto está más alto que la ceja.	2	Este es un error que debemos de admitir ya que la binarización no siempre va a ser la adecuada para todas las imágenes, pero si es un porcentaje pequeño de toda la base de datos entonces la propia red neuronal descartará una distancia si es mucho mayor que las demás.
No detecta el punto en la mitad del ojo porque no está exactamente en la mitad de la sección que se ha llamado ojo	4	Sumar la posición en la cual empieza el objeto y a partir de ese punto se sumaría la mitad del ancho o alto según se quiera del propio objeto.
Binariza mal la boca.	1	Al igual que en el error numero 3 este es un error que debemos de admitir ya que la binarización no siempre va a ser la adecuada para todas las imágenes.

Figura 50 : Tabla errores Prueba 3

Cuarta Prueba

Después de resolver estos errores volvimos a probar el código con 20 imágenes de la carpeta de miedo (10 mujeres y 10 hombres). Se aumentó

considerablemente el número de acierto a 16, pero seguíamos teniendo errores que teníamos que resolver.

ERRORES	Nº DE SUJETOS	SOLUCIÓN
Solo tiene un objeto, ceja y ojo son uno.	2	Revisar el if creado para que el único objeto existente se guarde como ojo y no como ceja
No detecta la boca.	1	Tanto el error 2 y el 3 son errores que debemos de aceptar, ya que en un porcentaje pequeño se debe aceptar.
No tiene ceja.	1	Tanto el error 2 y el 3 son errores que debemos de aceptar, ya que en un porcentaje pequeño se debe aceptar.

Figura 51 :Tabla errores Prueba 4

Quinta Prueba

Finalmente se realizó una última prueba con 20 imágenes de cada una de las emociones (10 mujeres y 10 hombres). Y se obtuvieron los siguientes errores, diferenciados de cada emoción.

EMOCIÓN	ERROR
Miedo	➔ No detecta la boca
Ira	➔ No binariza bien la boca. ➔ No detecta la boca. ➔ La ceja es otro objeto.
Asco	➔ No binariza bien la imagen. ➔ No detecta la boca.
Alegría	➔ No detecta la nariz.
Neutro	➔ No binariza bien la cara. (2 sujetos)
Tristeza	➔ No detecta la boca. ➔ No binariza bien la boca. (2 sujetos)
Sorpresa	➔ No detecta la boca. (4 sujetos)

Figura 52 : Tabla errores Prueba 5

Finalmente, estos errores no los hemos podido solucionar, ya que si aumentábamos o disminuíamos los niveles de umbralización en el detector de

nariz o boca, nos repercutía en otras fotos y era mayor el número de error. Nos detectaba un mayor número de bocas, lo que inducia a errores mayores.

Estos han sido los resultados que se han obtenido con el código explicado al principio del apartado.

Todos los errores y pruebas que se han llevado a cabo han desembocado en el código que ahora mismo tenemos entre nuestras manos.

8.3.4. Fase 3.4

Por último, se ha creado un bucle para que lea todas las imágenes de la base de datos a la vez, calcule los descriptores y las almacene en una matriz de 12x M, siendo M el número de imágenes de la base de datos de cada emoción.

Para poder introducir estas matrices en la red neuronal debemos tenerlas todas en una misma matriz. Para ello se han leído de nuevo los archivos de cada matriz y se han concatenado todas las emociones. Creando así una matriz final de 12x835 elementos.

8.4. Fase 4

En Matlab existe una app llamada Neural Network Pattern Recognition, que nos permite crear una red neuronal de forma muy rápida y sencilla. En este apartado, vamos a explicar paso a paso el proceso que hemos seguido hasta obtener la función de Matlab de nuestra propia red neuronal.

En primer lugar, se escribió un pequeño código para crear una matriz de etiquetas. Esta es una matriz llamada targets de 7 emociones x las 835 muestras que tenemos en la base de datos.

En este caso no tenemos el mismo número de imágenes por cada emoción. Por eso se creó una matriz de ceros y posteriormente se fue rellenando con unos.

Obteniendo así una etiqueta en binario para cada imagen.

Alegría =	1000000
Asco =	0100000
Ira =	0010000
Miedo =	0001000
Neutro =	0000100
Sorpresa =	0000010
Tristeza =	0000001

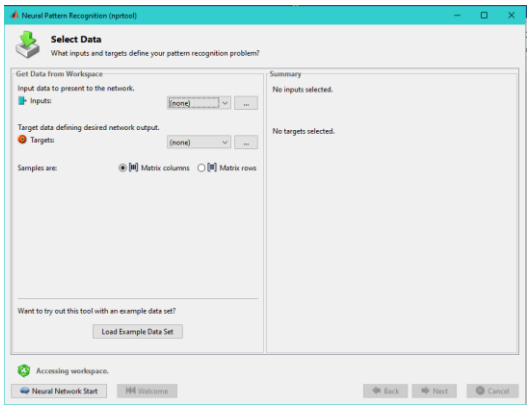


Figura 53 : Primer paso red neuronal

En primer lugar, en la aplicación tenemos una ventana que nos permite cargar las matrices de entrada y la de etiquetas.

Esta aplicación te permite seleccionar un porcentaje de imágenes para validación y otro de test, dentro de la misma base de datos que se le ha introducido como entradas para entrenar.

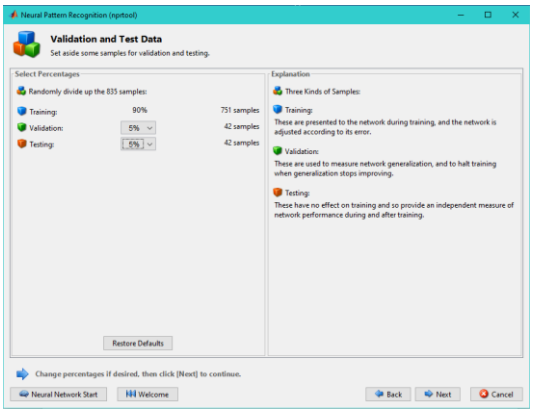


Figura 54: Segundo paso App Red Neuronal

En mi este caso solo hemos elegido un 5% en cada uno de los casos puesto que ya tenemos 15 imágenes por emoción guardadas para hacer nuestro propio test.

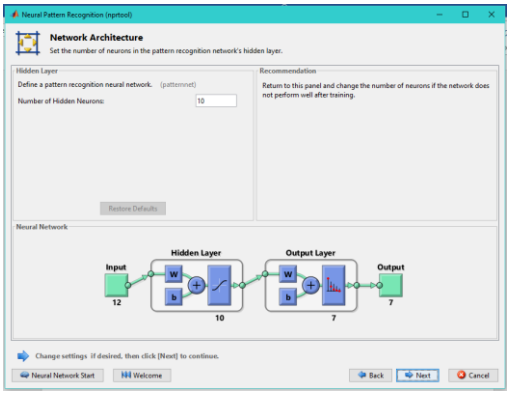


Figura 55 : Tercer paso App Red Neuronal

Por último, se elige el número de neuronas de la red y se entrena.

Después de realizar pruebas con diferentes números de neuronas y reentrenándolo hemos llegado a las siguientes conclusiones.

Nº Neuronas	Porcentaje de Validación	Tasa de Error
10	10 %	47,6%
10	15%	45,8%
12	10%	46,8%
12	15%	45,9%
15	10%	47,4%
15	15%	42,2%

Figura 56 : Tabla tasa de error de las redes neuronales

Por lo tanto, como se puede observar en la tabla anterior la red neuronal que menor tasa de error tiene está formada de 15 neuronas, y además se ha incrementado el porcentaje de imágenes que se gastan como validación. Se ha probado con dos porcentajes diferentes en el set de imágenes de validación, ya que lo que hace con este conjunto de imágenes es ajustar la tasa de error del set de entrenamiento con el de validación, forzando así la red neuronal a que el error en ese punto decaiga más.

	Alegria	Asco	Ira	Miedo	Neutro	Sorpresa	Tristeza	Total
Alegria	88 13,20%	4 0,60%	1 0,10%	1 0,10%	3 0,40%	0 0,00%	4 0,60%	87,10% 12,90%
Asco	0 0,00%	62 9,30%	11 1,60%	17 2,50%	2 0,30%	1 0,10%	13 1,90%	58,50% 41,50%
Ira	1 0,10%	12 1,80%	66 9,90%	8 1,20%	12 1,80%	0 0,00%	8 1,20%	61,70% 38,30%
Miedo	0 0,00%	7 1%	3 0,40%	22 3,30%	9 1,30%	9 1,30%	15 2,20%	33,80% 66,20%
Neutro	5 0,70%	4 0,60%	7 1,00%	15 2,20%	54 8,10%	6 0,90%	33 4,90%	43,50% 56,50%
Sorpresa	1 0,10%	0 0,00%	3 0,40%	28 4,20%	7 1,00%	71 0,60%	5 0,70%	61,70% 38,30%
Tristeza	0 0,00%	5 0,70%	6 0,90%	12 1,80%	7 1,00%	2 0,30%	18 2,70%	36,00% 64,00%
Total	92,60% 7,40%	66,00% 34,00%	68,00% 32,00%	21,40% 78,60%	57,40% 42,60%	79,80% 20,20%	18,80% 81,30%	57,00% 43,00%

Figura 57: Matriz de Confusión Red Neuronal Set de entrenamiento

	Alegria	Asco	Ira	Miedo	Neutro	Sorpresa	Tristeza	Total
Alegria	111 13,30%	5 0,60%	1 0,10%	2 0,20%	4 0,50%	0 0,00%	4 0,50%	87,40% 12,60%
Asco	0 0,00%	75 9,00%	18 2,20%	18 2,20%	2 0,20%	2 0,20%	16 1,90%	57,30% 42,70%
Ira	1 0,10%	15 1,85%	79 9,50%	9 1,10%	13 1,60%	1 0,10%	11 1,30%	61,20% 38,80%
Miedo	0 0,00%	9 1,10%	3 0,40%	29 3,50%	11 1,30%	12 1,40%	21 2,50%	34,10% 65,90%
Neutro	6 0,70%	5 0,60%	8 1,00%	16 1,90%	71 8,50%	7 0,80%	39 4,70%	46,70% 53,30%
Sorpresa	1 0,10%	1 0,10%	3 0,40%	34 4,10%	7 0,80%	95 11,40%	5 0,60%	65,10% 34,90%
Tristeza	3 45,00%	6 75,00%	8 1,00%	15 1,80%	9 1,10%	2 0,20%	22 2,60%	33,80% 66,20%
Total	91,05 9,00%	64,70% 35,30%	65,80% 34,20%	23,60% 76,40%	60,70% 29,30%	79,80% 20,20%	78,60% 21,40%	57,70% 42,30%

Figura 58 : Matriz de confusión red neuronal Total

En este tipo de tablas, no solo se nos muestra la tasa de acierto y de error, sino un gráfico de confusión indicando cuales son las emociones que más se equivocan entre ellas, teniendo en la diagonal el acierto.

En la segunda tabla podemos observar la matriz de confusión general, es decir se juntan la de entrenamiento, la de validación y la de test. De esta forma podemos ver que la emoción en la que más falla es el Miedo y el Neutro.

Muchas de las imágenes que corresponden a estas emociones las etiqueta como otra emoción.

8.5. Fase 5

Finalmente, como última fase se ha introducido todo el código de Matlab en SubVis de LabView.

El código del SubVi de Viola-Jones, explicado en el apartado 10 (Como conectar con la cámara), lo hemos modificado de forma que ahora no solo detecta y recorta la cara, sino que este es el SubVi extrae los descriptores de cada imagen capturada.

Como ya se ha explicado en apartados anteriores, los descriptores que se han utilizado son las distancias más características de la cara de una persona para determinar sus expresiones faciales, y a su vez estas convertirlas en las emociones.

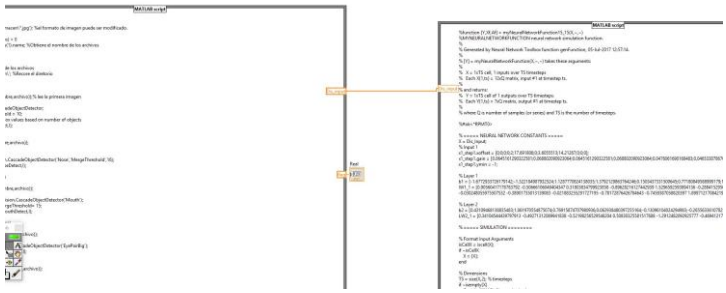


Figura 59: Diagrama de bloque Vi Guardar Distancias

En esta imagen se puede ver como se han enlazado los dos Matlab Scripts, el de la derecha realiza el cálculo de los descriptores de todas las imágenes que se han guardado en la carpeta de almacenamiento de la cámara (imágenes que una vez las analiza las elimina de la carpeta, para ahorrar espacio y disminuir el coste computacional). Entonces nos devuelve dos salidas, ambas de tipo "double" pero diferentes en el formato. Por un lado, se encuentra un flag que se pone a 1 cuando la carpeta de almacenamiento está vacía. Y por otro lado una variable con los descriptores a la que llamamos Dis_input, la cual será la entrada del siguiente script(a la derecha), la red neuronal que se ha creado en Matlab.

La red neuronal nos devuelve otro vector llamado "Y", con el parecido de cada imagen a cada una de las emociones. Hay casos en los que las puntuaciones son muy distintas entre ellas y la emoción a la que pertenece se diferencia claramente. Para ello lo que hacemos es sacarnos la posición del valor más alto del vector de parecidos. Este índice es casi nuestro resultado final, ya que cada posición equivale a una emoción diferente, teniéndola así ya identificada.

Teniendo ya el SubVi "Guardar distancias" insertado en el diagrama de bloques de nuestro Vi principal, (dentro del bucle "Timed Loop" para que empiece más tarde), le hemos insertado otro script de Matlab para que nos diga el número de imágenes que se han detectado de cada emoción. Esto se almacena en un vector, dejando a 0 la emoción que no tiene ninguna imagen, y haciendo un contador de enteros en las emociones correspondientes.

Por último, el vector obtenido en este último Matlab Script se dibuja en una gráfica de tipo tarta en el panel frontal de LabView.

En la siguiente figura podemos observar el Vi principal.

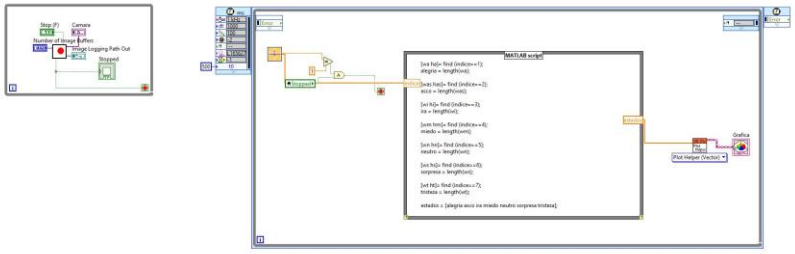


Figura 60: Diagrama de bloques Vi principal

Aquí se muestra el panel frontal del Vi principal, el cual es el entorno gráfico con el que el usuario interactuará, pudiendo verse por la pantalla, parar el proceso de grabación y esperar al análisis de las emociones y su posterior gráfica.

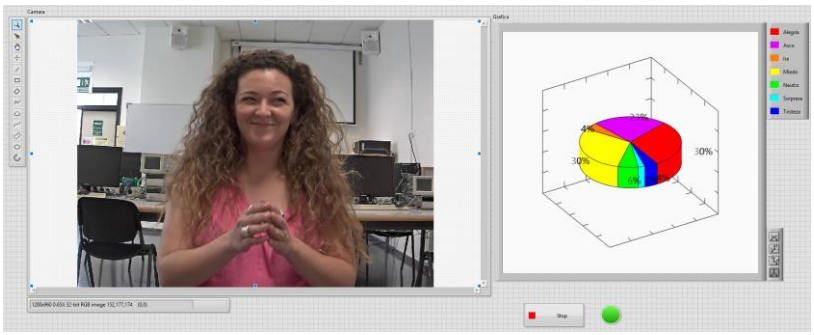


Figura 61: Panel Frontal del Vi principal

9. Pruebas

Después de todas las pruebas realizadas para el código de Matlab con las imágenes de la base de datos, y las pruebas para testear la red neuronal con un set de imágenes de test, se han llevado a cabo las últimas pruebas para validar el programa. Hasta este momento las pruebas se habían realizado con imágenes fijas de la base de datos, mientras que esta última ha sido hecha con el video que captamos con la propia cámara del laboratorio.

Para comprobar que el código de Matlab funcionaba de forma correcta también se han realizado las últimas pruebas con las imágenes de la cámara, en las cuales sigue existiendo la misma tasa de error para diferenciar las emociones.

Las últimas pruebas realizadas, han sido hechas desde LabView para testear que las imágenes de la cámara de video las analizaba bien.

Pese a haber corregido el código de Matlab para que sirviera en el mayor número posible de imágenes de la base de datos, aun se han tenido que ajustar más los niveles de umbralización y el MergeThreshold de los detectores de ojos, nariz y boca.

No es lo más apropiado pero el entorno del laboratorio en el cual se han realizado las pruebas con la cámara está mucho más iluminado, y la cámara tiene una mayor resolución y un mayor campo de visión que las imágenes con las cuales se probó y se hizo la base de datos.

Una vez analizados y corregidos esos errores, se han realizado varias pruebas de entre 10 y 15 segundos en las cuales el sujeto experimentaba 2 o 3 emociones.
Pero en la gráfica nos aparecen siempre las 7 emociones, ya no solo por la tasa de error propia, sino porque al ser un video existen Frames que son justo la transición de una emoción a otra, y nuestro programa lo detecta como un conjunto de do emociones.

10. Conclusiones

Como conclusión principal, el programa ha cumplido sus objetivos marcados. No solo se ha desarrollado un detector de 6 emociones, como era el objetivo del proyecto, sino que durante su proceso de realización se ha aumentado el número de emociones a 7. Ya que las emociones principales eran Alegría, Asco, Ira, Miedo, Sorpresa y Tristeza, y se pensó que, si se utilizaba este programa en una situación normal, por ejemplo, trabajando cara al ordenador, lo más común es estar en un estado neutral, sin mostrar ningún tipo de emoción. Por este motivo se introdujo 1 estado más, el Neutro, teniendo así finalmente nuestras 7 emociones.

Por otro lado, aunque era lo necesario añadir este estado, esto ha provocado que la tasa de error crezca, por eso se propone varias posibles mejoras del código.

En primer lugar, una de las más sencillas es reducir el número de emociones. Se ha probado reduciéndolo solo a 4 y la tasa de error se puede reducir a menos de un 30%. Por lo tanto, tampoco es una solución muy aceptable ya que eliminando 3 emociones debería verse reducida la tasa de error en un porcentaje mucho mayor.

Como segunda opción podríamos aumentar de forma considerable el número de imágenes para la base de datos, ya que cuantas más imágenes se tienen en la base de datos mayor es la precisión de la red neuronal a la hora de acotar una serie de distancias.

Y para finalizar se podría modificar el código utilizado para extraer los descriptores de la imagen, gastando algoritmos de bio detección, calculando las proporciones áureas del rostro de cada sujeto. Creando así un programa mucho más eficiente. Y utilizando descriptores Haar.

Todas estas soluciones nombradas anteriormente, son las fases que se estudiaran en el futuro como para mejorar el programa, reduciendo la tasa de error y calculando los descriptores de forma más eficiente.

Además, se debería intentar reducir el coste computacional para el procesado de las imágenes y así el retardo que tiene desde que el usuario deja de grabar hasta que le sale la gráfica de las emociones.

11. Bibliografía

1. Herruzo, Jose Ignacio Herranz. Analisis de imagenes- Visión artificial- Representación y descripción. Valencia : Universitat Politecnica de Valencia, 2017.

2. Histograms of orientes Gradients for Human Dtection. Dalal, Navneet y Triggs, Bill. Washington, DC, EEUU : IEEE Computer Societ of Washington, 2005, Vol. 1.

3. Visión artificialy Robótica Caracteristicas y segmentación. [En línea] [Citado el: 4 de Junio de 2017.] https://moodle2015-16.ua.es/moodle/pluginfile.php/105463/mod_page/content/43/3%20Caracteristicas%20y%20segmentacion.pdf.

4. Roger, Gimeno Hernandez. Estudio de tecnicas de reconocimiento facial. Barcelona : Universitat Politecnica de Catalunya, 2010.

5. Angel Orozco G, Alvaro, Álvarez, Mauricio y Fetecua Valencia, Juan Gabriel. Scientia et technica. [En línea] Junio de 2008. [Citado el: 10 de Marzo de 2017.] http://revistas.utp.edu.co/index.php/revistaciencia/article/view/3683/2073.

6. Wiskott, Laurenz, P.Würtz, Rolf y Westphal, Günter. Elastic Bunch Graph Matching. [En línea] 12 de Septiembre de 2014. [Citado el: 15 de Marzo de 2017.] http://www.scholarpedia.org/article/Elastic_Bunch_Graph_Matching.

7. Vetter, Volker Blanz and Thomas. graphics and vision. [En línea] http://gravis.dmi.unibas.ch/Sigg99.html#top.

8. Las seis emociones basicas. Las seis emociones basicas. [En línea] 26 de Febrero de 2014. [Citado el: 4 de Noviembre de 2016.] http://germansalinas.blogspot.com.es/2014/02/las-seis-emociones-basicas.html.

9. Facial expression (mood) recognition from facial images using committee neural networks. [En línea] 5 de Agosto de 2009. [Citado el: 20 de Abril de 2017.] https://www.ncbi.nlm.nih.gov/pmc/articles/PMC2731770/.

10. Álvarez, Jana. Analitica Web. [En línea] 22 de Diciembre de 2016. [Citado el: 25 de Mayo de 2017.] http://www.analiticaweb.es/machine-learning-y-support-vector-machines-porque-el-tiempo-es-dinero-2/.

11. Redes Neuronales Artificiales con MATLAB. [En línea] [Citado el: 10 de Junio de 2017.] https://es.mathworks.com/discovery/redes-neuronales.html.

12. Brownlee, Jason. Boosting and AdaBoost for Machine Learning. [En línea] 25 de Abril de 2016. [Citado el: 4 de Junio de 2017.] http://machinelearningmastery.com/boosting-and-adaboost-for-machine-learning/.

13. AAAC Emotion-Research.net. [En línea] [Citado el: 28 de Mayo de 2017.] http://emotion-research.net/projects/humaine/aboutHUMAINE.

14. Belfast Naturilistic Database. [En línea] [Citado el: 28 de Mayo de 2017.] https://belfast-naturalistic-db.sspnet.eu/.

15. IITK. [En línea] 2005. [Citado el: 28 de Mayo de 2017.] http://www.iitk.ac.in/infocell/iitk/newhtml/storyoftheweek24.htm.

16. Yale face Databaes. [En línea] 10 de Septiembre de 1997. [Citado el: 28 de Mayo de 2017.] http://cvc.cs.yale.edu/cvc/projects/yalefaces/yalefaces.html.

17. Archive. [En línea] 17 de Marzo de 2005. [Citado el: 28 de Mayo de 2017.] http://www.vision.caltech.edu/html-files/archive.html.

18. BioID be recogniced. [En línea] 2017. [Citado el: 28 de Mayo de 2017.] https://www.bioid.com/About/BioID-Face-Database.

19. AT&T Lboratories Cambridge. [En línea] 2012. [Citado el: 28 de Mayo de 2017.] http://www.cl.cam.ac.uk/research/dtg/attarchive/facedatabase.html.

20. Semaine Database. [En línea] [Citado el: 28 de Mayo de 2017.] https://semaine-db.eu/.

21. AFFECT ANALYSIS GROUP. [En línea] [Citado el: 28 de Mayo de 2017.] http://www.pitt.edu/~emotion/ck-spread.htm.

22. [En línea] [Citado el: 28 de Mayo de 2017.] http://www.kasrl.org/jaffe_info.html.

23. [En línea]

24. Caroline Pacheco do E. Silva. Behance. [En línea] 8 de Enero de 2017. [Citado el: 28 de Mayo de 2017.] https://www.behance.net/gallery/10675283/Facial-Expression-Public-Databases.

25. Vitutor.net. [En línea] 2015. [Citado el: 15 de Junio de 2017.] http://www.vitutor.net/1/22.html.

26. Caroline, Pacheco do E.Silva. Facial Expression Public Databases. Facial Expression Public Databases. [En línea] Bèhance, 1 de Septiembre de 2013. [Citado el: 1 de Junio de 2017.] https://www.behance.net/gallery/10675283/Facial-Expression-Public-Databases.

27. Montañes, Mariano Choliz. Psicologia de la emoción el proceso emocional. Valencia : Universidad de Valencia, 2005.

28. Wiskott, Laurenz, P.Würtz, Rolf y Westphal, Günter. Elastic Bunch Graph Matching. [En línea]

29. [En línea] http://courses.cs.tamu.edu/rgutier/cpsc689_s07/.

30. Banerjee, Asit Kumar Datta Madhura Datta Pradipta Kumat. Face Detecction and Recognition: Theory and Practice. New York : CRC Press, 2016.

31. Bartlett, Marian Stewart. Face Image Analysis by unsupervised learning. San Diego : University of California, 2011.

32. Shaogang Gong, Stephen J McKenna, Alexandra Psarrou. Wikipedia. 2000 :

Imperial College Press, 2016.

33. Mathworks. Mathworks. [En línea] https://es.mathworks.com/

34. http://www.ni.com/es-es.html. National Instruments. National Instruments. [En línea]