Universidad de Valladolid

UNIVERSIDAD DE VALLADOLID

ESCUELA DE INGENIERIAS INDUSTRIALES

Grado en Ingeniería Electrónica Industrial y Automática

Algoritmo de extracción de características faciales para la clasificación de individuos por edad

Autor:

González Bartolomé, Sergio

Tutor:

De la Fuente López, Eusebio

Departamento de Ingeniería de Sistemas y Automática

Valladolid, julio de 2019.

Agradecimientos

A mis padres, a mi hermana y a todos mis amigos por el apoyo incondicional en esta etapa de mi vida. En especial a mi pareja por estar pendiente de mí motivándome cada día.

Por último, gracias a Google y a YouTube por compartir conmigo esa gran fuente de conocimientos.

Resumen y palabras clave

La visión artificial ha sufrido un crecimiento exponencial, el cual ha sido potenciado por la industria 4.0. Esta tecnología no solo se ha restringido a un uso industrial, sino que en nuestro día a día podemos ver como se integra facilitando con ello un sinfín de tareas.

El neuromarketing es uno de los sectores que está en alza gracias a esta tecnología por su ayuda a la hora de captar emociones o movimientos de los sujetos que estén en un campo de estudio.

Vinculado a todo ello, se pretende crear una aplicación que consiga identificar la edad y el género de los sujetos que estén presentes en una cámara de video. Para ello se emplearán técnicas de clasificación y descriptores de textura, siendo este último el que permitirá cruzar los resultados obtenidos con los datos previamente procesados de una base de datos, dando lugar a la obtención del género y la edad.

Patrones binarios locales, Clasificador HaarCascade, Visión artificial, OpenCV, Base de datos FERET.

Abstract and key words

Artficial vision has been strongly developed in the last decade. It too mainly evolved tanks to the supprt of the 4.0 industry. Not only this technology has an industrial utility but also it can be useful in our daily life helping in a lot o regular activities.

Neuromarketing is now rising as a promising technology due to the development of artificial vision, since it can be used for capturing movements or emotions shown by the subjects included in the study.

Along this, it is been tried to design an application which allows to identify age and sex of the people shown by the camera. To achieve this goal, clasification techniques an texure descriptors are being used. Texture descriptors allow the matching of the results obtained with the data earier available, being the last a processed database. And finally getting the definitive age and sex.

Local Binary Patterns, HaarCascade Classifier, Computer vision, OpenCV, FERET database.

Índice general

Índice de ilustraciones

CAPITULO I

1 Introducción

El presente Trabajo Fin de Grado ha sido desarrollado en la Escuela de Ingenierías Industriales de la Universidad de Valladolid, en colaboración con el Departamento de Ingeniería de Sistemas y Automática.

Con este documento se pretende dar explicación al desarrollo de un sistema de visión artificial dedicado a la extracción de características faciales, tales como la edad o el género, en tiempo real.

1.1 Justificación del estudio

Actualmente, el análisis de imágenes por computación de forma autónoma es un amplio sector, en auge, de la tecnología, el cual participa en diferentes ámbitos de nuestra vida cotidiana. Es utilizado por la industria, tanto para identificar las piezas como para posicionarlas por medio de un robot con un sistema de visión artificial integrado; por empresas de seguridad, a la hora de recabar información de los vídeos obtenidos mediante cámaras de videovigilancia; o incluso por empresas telefónicas tan populares como *Samsung©*, *Apple©* o *Xiaomi©*, las cuales últimamente han incorporado una aplicación a sus terminales para desbloquearlos mediante la cámara frontal del dispositivo cuando reconoce el rostro del titular.

Ilustración 1 - Face ID de Apple [1].

El desarrollo de esta tecnología viene acompañado de grandes empresas que invierten en nuevas herramientas, para mejorar la eficacia de los procesos ya existentes, o se dedican a investigar en nuevos campos en los que aplicar esta tecnología, para facilitar al usuario el desempeño de la tarea. De acuerdo a lo mencionado se observa que la mayor parte del software desarrollado es de uso privado o en su defecto posee un alto coste.

Pese a ello e inspiradas en estos últimos avances, cada vez más compañías enfocadas en el marketing se dedican a usar estas nuevas aplicaciones, las cuales les ayudan a obtener información de una muestra de la población, con la finalidad de transformar el sector del marketing y las ventas mediante la posibilidad de realizar "microestudios de mercado" fácilmente.

Por lo anteriormente expuesto, se elige orientar el presente proyecto dentro de este marco, el del marketing.

Como se puede inferir el proyecto se perfilará bajo la intención de crear una tecnología para el beneficio de pequeños negocios y no sólo de grandes multinacionales, es decir, usando solo herramientas de libre distribución, las cuales abaratan el coste. Con esto en mente, el desarrollo del proyecto se basará en la siguiente premisa:

"Colocar una cámara en el escaparate de una tienda y a su vez conectarla al dispositivo que contenga la aplicación que se va a desarrollar en el presente proyecto. La cámara obtendrá la imagen de los clientes que observen el escaparate. Posteriormente la aplicación procesará la información de esa imagen, con la opción de crear estudios de las personas interesadas en ese escaparate y en los productos que llamen su atención."

En síntesis, este proyecto buscará realizar una aplicación, de bajo coste, con la finalidad de usar la visión artificial para lograr reconocer rostros y extraer las características faciales de las imágenes que vayan a ser tratadas.

1.2 Objetivos

El objetivo principal de este Trabajo Fin de Grado es crear una aplicación que reconozca la cara presente en una imagen, siendo a su vez, capaz de extraer las características faciales de dicho rostro mediante el uso de visión artificial.

Para ello se hará uso del descriptor visual *ULBP (Uniform Local Binary Patterns)*, a la hora de realizar la aplicación, el cual describiremos posteriormente en esta memoria.

Otros objetivos marcados son:

- Realizar una aplicación bajo un estándar multiplataforma, es decir, podrá ser usada en diferentes entornos hardware y/o distintos sistemas operativos.
- Usar software y herramientas de libre distribución.
- La aplicación podrá ser ejecutada en tiempo real evitando decalajes que entorpezcan su funcionamiento.
- Se buscará establecer como funcionalidad el reconocer múltiples rostros a la vez.
- Enfocar el proyecto en una trayectoria autodidacta. Para ello, se almacenará la información de los sujetos en una base de datos (BBDD), lo que permitirá realizar extracciones más precisas a medida que la BBDD aumente.

A fin de lograr lo mencionado, se realiza una división de la memoria en 3 grandes sub-objetivos, los cuales se proceden a mencionar sin llegar a profundizar ya que serán desarrollados posteriormente en sucesivos apartados.

1.2.1 Estudio del estado de la cuestión

Se evaluará un marco teórico sobre las posibles aplicaciones similares existentes en el mercado, para posteriormente realizar un análisis comparativo entre la aplicación a desarrollar y el resto.

A mayores se explicará que es la visión artificial y cuáles son las herramientas que se pretenden usar, todo ello de forma teórica sin mostrar fragmentos de código.

1.2.2 Análisis de los materiales y herramientas

Se explicarán los materiales y herramientas que se emplearán en el proyecto y la razón de su elección frente a otras opciones.

Como se ha mencionado anteriormente, la mayor parte de las aplicaciones que poseen la finalidad que queremos obtener son de pago o de uso privado, por ello se hace necesario dar explicación a la elección de herramientas de libre distribución, las cuales, por lo general, contemplan un estándar multiplataforma para que puedan llegar a ser usadas en los diferentes sistemas operativos existentes hoy en día.

1.2.3 Desarrollo y evaluación de la aplicación

Contendrá la descripción de cómo se ha estructurado y codificado la aplicación. Con el objeto de facilitar la compresión se incluirán diagramas de flujo que mostrarán el camino a seguir de la información a través del código.

Por último, se comentarán los resultados obtenidos y futuras mejoras que se podrán realizar sobre el proyecto final.

1.3 Organización de la memoria

La composición de la memoria por capítulos será la siguiente:

- Capítulo I. Introducción

  Se describe el porqué de la realización de este proyecto, junto con los objetivos que se preten1den alcanzar.

- Capítulo II. Fundamentos teóricos

  Estudio de mercado sobre las aplicaciones existentes que empleen reconocimiento facial. A mayores se expondrán unos fundamentos sobre la visión artificial para facilitar la comprensión del resto del proyecto.

- Capítulo III. Herramientas software

  Descripción y elección de las herramientas informáticas que se emplearán en la elaboración del proyecto.

- Capítulo IV. Desarrollo

  Explicación de los procesos implicados en la aplicación. Se expondrá a mayores la realización de la puesta en marcha y el funcionamiento de la aplicación de forma continua.

- Capítulo V. Resultados

  Se muestran las evidencias obtenidas sobre el funcionamiento tanto correcto como incorrecto de la aplicación desarrollada.

- Capítulo VI. Conclusiones y mejoras

  Se dará solución a los errores expuestos en el capítulo anterior, al igual, que unas mejoras a implementar de cara a una implementación futura.

CAPITULO II


Escuela de Ingenierías Industriales
Fundamentos Teóricos

2 Fundamentos teóricos

Con el objeto de obtener una visión general del marco en el que se encuentra este proyecto, se procede a explicar en qué consiste la visión artificial, junto con las tecnologías actuales que pueden competir con la aplicación que se va a desarrollar.

Asimismo, se ve necesidad de mencionar otras aplicaciones relacionadas con el reconocimiento facial, ya que será el grueso del proyecto.

2.1 Visión artificial

El estudio y la creación del presente software surge del auge que ha sufrido la visión artificial en los últimos años. Atendiendo esto, se realizará un breve análisis sobre los avances y las técnicas que emplea la visión artificial.

2.1.1 Introducción

Antes de entrar en consideración, cabría preguntarse qué es la visión artificial, la cual también es identificada como "Visión por Ordenador" siendo mencionada indistintamente a lo largo de la memoria.

Una definición puede ser, "*La visión artificial es una disciplina científica que incluye métodos para adquirir, procesar y analizar imágenes del mundo real con el fin de producir información que pueda ser tratada por una máquina.*" [2]

Vinculado a dicho concepto, se pasa a enumerar los principales métodos usados por los desarrolladores de esta disciplina. [3]

- *Procesamiento Digital de Imágenes.*

  Consiste en convertir una imagen real en otra modificada, la cual simplifique la extracción de características, a través de la eliminación del fondo o mejorando la calidad de la fotografía tomada.

- *Análisis de Imágenes.*

  Proceso mediante el cual se intentan obtener datos de la imagen de forma autónoma. Véase por ejemplo los procesos de calidad de las industrias, los cuales detectan fallos en la pieza inspeccionada.

- *Reconocimiento de Patrones.*

  Se emplean plantillas para reconocer formas o diferentes tonalidades, de cara a una posible enumeración o selección. Método usado especialmente en el sector de la seguridad.

- *Computación Gráfica.*

  Genera imágenes artificiales las cuales son construidas mediante estructuras físicas conocidas. Técnica empleada en videojuegos o en la industria cinematográfica, véase los efectos especiales.

2.1.2 Evolución

La invención de la fotografía trajo consigo grandes avances tecnológicos [4], por ejemplo, en la fotogrametría a través de la extracción de imágenes aéreas con globos, o en la astronomía gracias a poder capturar fotografías del firmamento e incluso pasando por la medicina, realizando capturas de los organismos visibles a través de un microscopio.

Este hecho, junto con la revolución electrónica, propiciaron un movimiento científico que tenía como finalidad poder analizar imágenes de forma autónoma, sin intervención humana.

"El inicio de la visión artificial, desde el punto de vista práctico, fue marcado por Larry Roberts (considerado uno de los padres de Internet), el cual, en 1961 creó un programa que podía "ver" una estructura de bloques, analizar su contenido y reproducirla desde otra perspectiva, demostrando así a los espectadores que esa información visual que había sido mandada al ordenador por una cámara, había sido procesada adecuadamente por él." [5]

Fue en la década de los 80 cuando este movimiento empezó a tener más inversores del calibre de *IBM©* o *Intel©*, quienes eran los estándares de tecnología en el momento. Hasta 1999, que fue el año en el que surgió la primera versión de *OpenCV©*, la visión artificial fue algo de uso exclusivamente privado, usado y desarrollado por grandes empresas.

A partir de ese momento, el desarrollo de aplicaciones relacionadas con la visión artificial creció exponencialmente, ya que no solo las grandes compañías se dedicaban a crear proyectos, sino que, usuarios empezaron a crear aplicaciones a título personal siendo compartidas y mejoradas gracias al empleo de internet.

Actualmente, la visión artificial está siendo integrada en todos los ámbitos de nuestra vida, desde el neuromarketing, pasando por la robótica colaborativa en nuestros hogares, hospitales, etc. Hasta el punto de favorecer la nueva revolución industrial (Industria 4.0) con la automatización de las factorías y la aplicación de

inteligencia artificial, como por ejemplo empleando robots autónomos con visión artificial integrada.

2.1.3 Configuración de un sistema de Visión Artificial

La visión artificial es aún una disciplina en desarrollo, por lo que actualmente es necesario controlar el lugar de implantación de la misma, teniendo en cuenta luminosidad, ruido y/o distancia. Por estas razones es necesario crear un entorno físico que favorezca el proceso.

Para lograr la creación de dicho espacio operativo, se ha de poner especial énfasis en controlar la formación de la imagen y el posterior procesamiento de las mismas [4].

Estas ideas llevan a una división en subsistemas los cuales han de ser controlados de forma independiente para obtener el resultado esperado del proceso, facilitando con ello la búsqueda de un control global.

- Subsistema de iluminación.

  Realizar un estudio de iluminación es necesario de cara a la obtención de la imagen. Dicho análisis ha de tener en cuenta las circunstancias en las que se trabaja, es decir, si es en interior o exterior. Véase que si el control de este subsistema es en el exterior se hace más tedioso controlar la cantidad lumínica.

  Este control viene acompañado del uso de lámparas, filtros de luz o pantallas fotográficas, ayudando a manejar la exposición del objeto, evitando quemar la imagen o en su defecto la pérdida de información por ser excesivamente oscura.

- Subsistema de captación.

  Como se puede inferir, esta tarea es realizada por una cámara, ya sea para captar fotogramas o videos. Sin embargo, no solo se busca obtener imágenes convencionales, sino que a través de la implementación de cámaras térmicas o de radiación gamma se puede fotografiar el espectro no visible, permitiendo posteriormente analizarlo y procesarlo.

- Subsistema de adquisición.

  Una cámara se encarga de generar una señal eléctrica, la cual ha de ser transmitida, ya sea de forma analógica o digital.

  Actualmente la mayor parte de los periféricos de captación son digital, pero continúan existiendo sistemas que trabajan con vídeo analógico (CCIR, PAL, RS170, NTSC…). Como se puede deducir, para realizar el procesado se necesita que las imágenes o vídeos sean de carácter digital, es por ello que se emplean

tarjetas de adquisición o *"frame grabbers"*, los cuales actúan como convertidores en caso de ser necesario.

- Subsistema de procesamiento.

Se realiza a través de un ordenador o de varios, en función de las necesidades de los algoritmos. El controlador tendrá como entrada la imagen digitalizada proveniente del sistema de adquisición.

Asimismo, como salida se alcanzará otro tipo de información, la cual puede ir desde salidas por pantalla con información significativa de la imagen, hasta señales que incidan en distintos sensores y actuadores de tal forma que regulen el proceso a controlar.

- Subsistemas de periféricos.

Serán los sensores y actuadores que recibirán la información obtenida tras el procesado. Su principal función es realizar una acción ante un evento o retroalimentar el flujo para corregir las posibles fallas presentes.

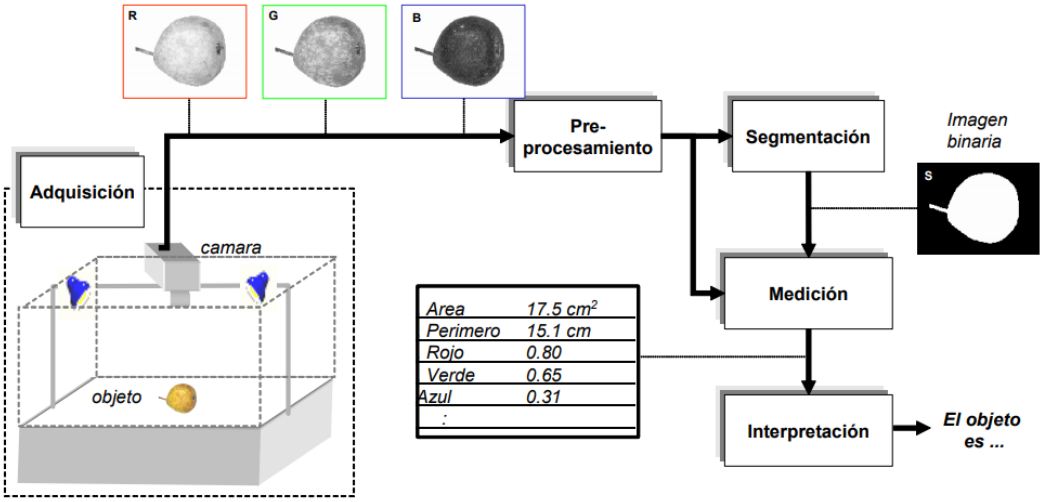


Ilustración 2 - Esquema de un sistema de análisis de imágenes [6].

Los subsistemas prioritarios para este proyecto serán la adquisición de la imagen y el tratamiento de esta, es por ello que se procede a entrar en detalle sobre estos aspectos.

2.1.3.1 Adquisición de la imagen

Al digitalizar una imagen se obtiene una o tres matrices bidimensionales, atendiendo la captura, si es en blanco y negro o en color respectivamente.

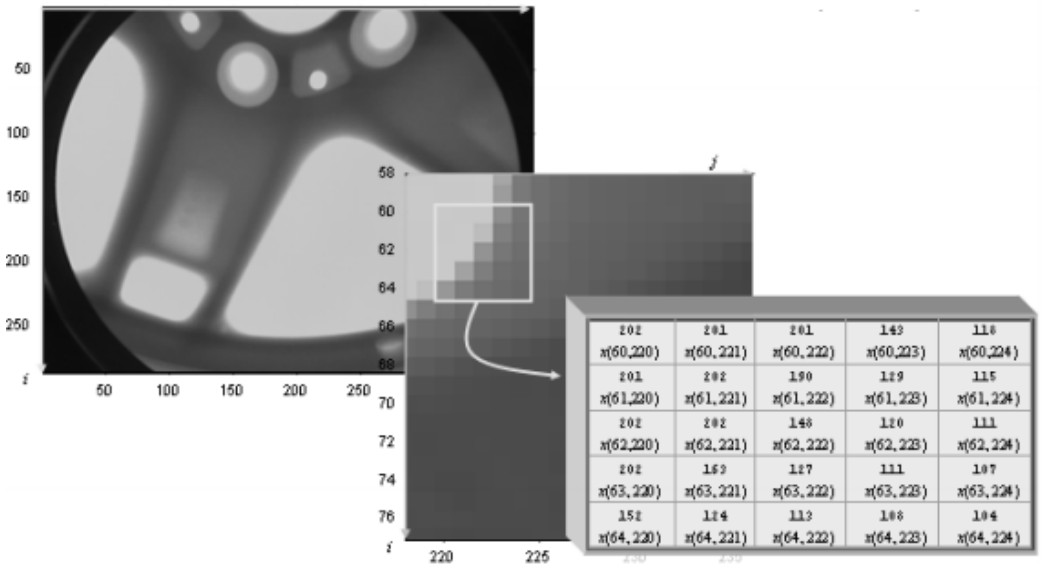


Ilustración 3 - Valor matricial de una imagen [4].

Cada celda constituirá un píxel, el cual entrega diferente información al usuario. Si la imagen es en blanco y negro, la información será el brillo de ésta, en cambio sí es en color, permitirá conocer la intensidad de color (RGB – RED, GREEN, BLUE) en esa región.

Captura

De acuerdo con la captura de fotografías o videos, se puede hablar del uso de dispositivos pasivos (cámara oscura) o activos (escaneo) [7].

Entre los integrantes de los dispositivos pasivos se encuentran las cámaras de fotografía y video convencionales. Estos instrumentos están formados por una caja cerrada, la cual posee un orificio en una pared que permite el paso de luz. Tras el paso de ésta, se proyecta en la pared contraria una imagen invertida de lo que se encuentra ante el orificio.

Estas cámaras obtienen fotogramas en dos dimensiones, pero combinándolas se puede capturar diferentes perspectivas del objeto fotografiado llegando a construir el elemento en 3D.

Por otra parte, tenemos los dispositivos activos o también llamados escáneres. Estos son diametralmente opuestos a los ya mencionados. Un elemento activo, véase un láser, recorre la escena a capturar. El escáner emite rayos de luz los cuales son reflejados al impactar contra el escenario, para posteriormente ser recogidos por un detector de luz. Este proceso se realiza de forma continua para formar la señal que identificará la escena que está siendo analizada. La finalidad principal del escaneo es capturar objetos o escenas en 3D.

Sobre las ideas expuestas se puede deducir que las cámaras son más eficaces y veloces que los escáneres. Es por ello que a la hora de elegir un dispositivo de captura para este proyecto se ha optado por un dispositivo pasivo.

Digitalización

Se procede a explicar el cambio que sufre la información desde que se captura la luz del escenario (sistema analógico) hasta que se obtiene la imagen discretizada en píxeles (sistema digital)[7].

El muestreo es el primer punto, el cual mide a intervalos discretos, en un periodo de tiempo, el valor analógico obtenido. De este proceso son representativas las variables *"frecuencia de muestreo"* y *"resolución espacial"*, las cuales nos indican respectivamente el número de veces que se mide el factor analógico y la distancia existente entre dos píxeles adyacentes respecto el objeto fotografiado.

La segunda operación que se realiza es la cuantización. Esta consiste en discretizar los valores de cada píxel. Por norma general se trabaja con 256 niveles.

Si la imagen solo posee una única matriz de píxeles, se habla de imagen en *niveles de gris* sirviendo el valor "0" del píxel para representar el negro y el "255" para el blanco. En cambio, si por el contrario el fotograma posee tres matrices está en una escala de *niveles en color real* debido a que cuentan con 16 millones de colores (256x256x256).

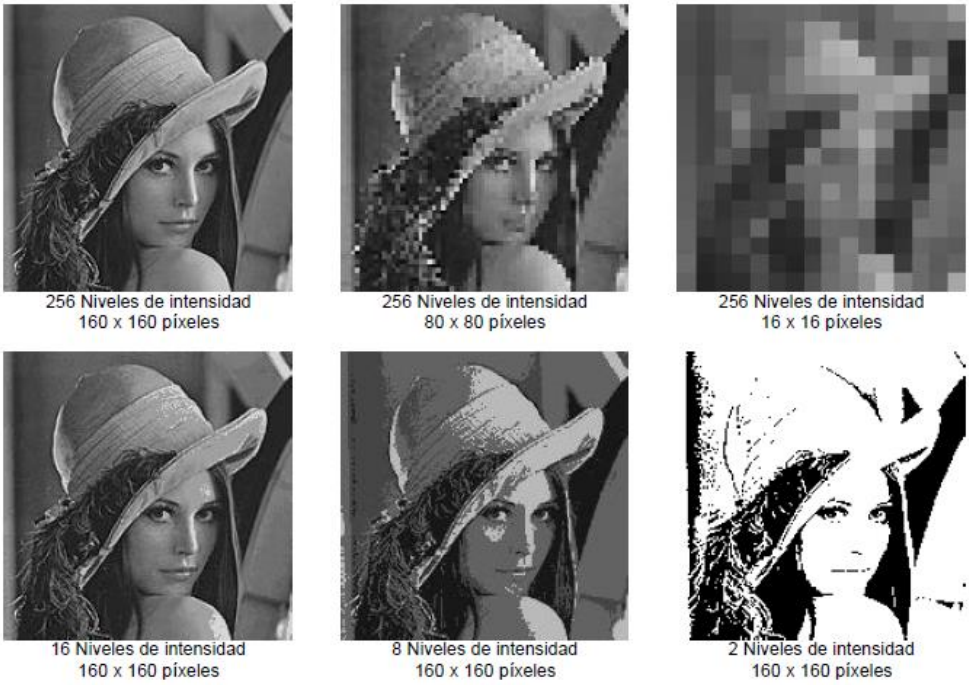


Ilustración 4 - Diferencias entre resolución espacial y cuantización. [7]

En referencia a la ilustración 4. Las imágenes de la fila superior han sido capturadas con un rango de 256 niveles de intensidad, pero variando la resolución espacial. Mientras, la fila inferior se caracteriza por tener la misma resolución espacial con diferentes niveles de intensidad.

Por último, cabe destacar la resolución de imagen. Se debe evaluar en función del resultado que se quiere obtener, ya que el aumento de este parámetro implica que la imagen adquiera un gran tamaño, dificultando el posterior procesamiento de la misma. Este aspecto se ha estandarizado a lo largo del tiempo tal y como podemos apreciar en la siguiente ilustración.

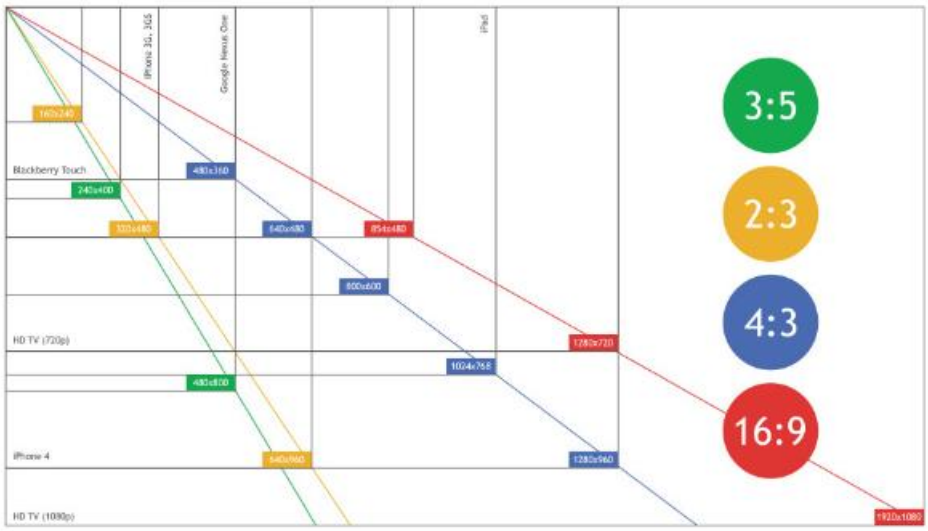


Ilustración 5 - Resoluciones más utilizadas [8].

2.1.3.2 Procesamiento de las imágenes

Una vez se ha obtenido la imagen digital se realiza un procesamiento previo a la segmentación y al reconocimiento. Este tratamiento busca mejorar o destacar elementos en las imágenes para simplificar las etapas posteriores, es decir, se puede comparar esta etapa con una función de transferencia, ya que se recibirá una señal de entrada (Imagen capturada) y se procesará hasta obtener la señal de salida.

Se han seleccionado las principales operaciones que se emplearán a lo largo del proyecto [7].

Operaciones entre píxeles

La clasificación básica en lo referente a estas operaciones es diferenciar entre aritmético-lógicas y geométricas.

Una operación aritmético-lógica se encarga de modificar directamente el valor de los píxeles presentes en la matriz imagen. Se utilizan funciones básicas de matemáticas, estadística y algunas propias de la electrónica digital.

Algunos ejemplos básicos son operaciones AND, OR, Inversa, Suma, Resta, Multiplicación, División, Media, Mediana.... Todas estas necesitan de dos imágenes para poder realizarse, lo que generaría una imagen diferente a las empleadas de partida.

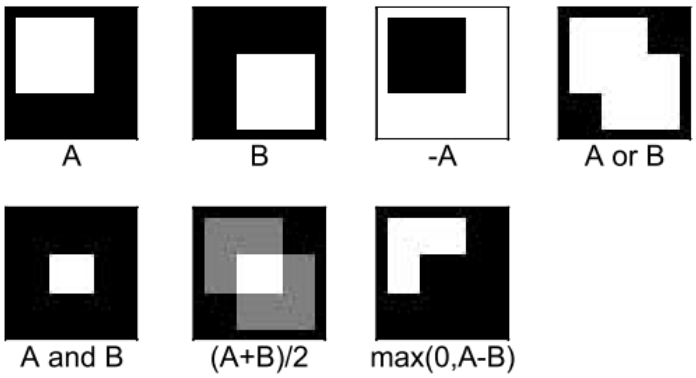


Ilustración 6 - Ejemplos de operaciones aritméticas con imágenes simples [7].

Por otro lado, las operaciones geométricas no se dan a nivel de píxel, sino que estas son aplicadas a toda la imagen al mismo tiempo. Si se trata la imagen como una matriz (x, y) cada píxel estará registrado mediante una coordenada horizontal y otra vertical. Siendo este el caso, se realiza una multiplicación de matrices entre la imagen capturada y una matriz definida para realizar la transformación deseada.

De este tipo de operaciones, las más usadas en este proyecto serán traslación, escalado y rotación. La matriz de transformación no será una imagen como en el caso anterior, sino que sólo contendrá información del tipo de operación y de la medida que se va a aplicar.

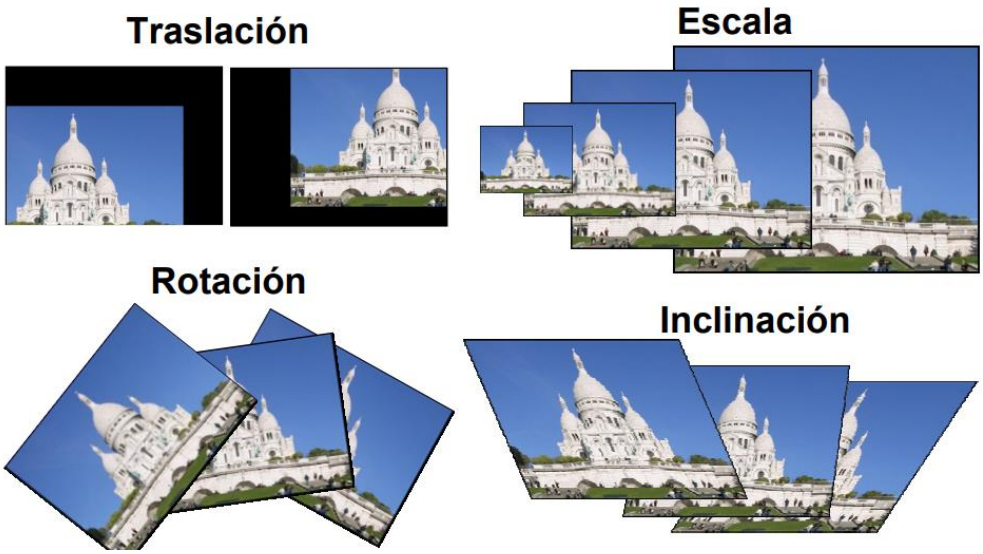


Ilustración 7 - Transformaciones afines o geométricas [9].

Operaciones en el histograma

Un histograma es un diagrama de barras cuyo eje de abscisas son los valores que puede tomar un píxel (0 a 255) y el eje de ordenadas representa el número de píxeles que se encuentran en la imagen con el mismo valor. Para imágenes en escala de gris obtendremos un único histograma, pero en imágenes a color, la representación de esta será a través de tres histogramas, cada uno para cada matriz de color (RGB o HSV).

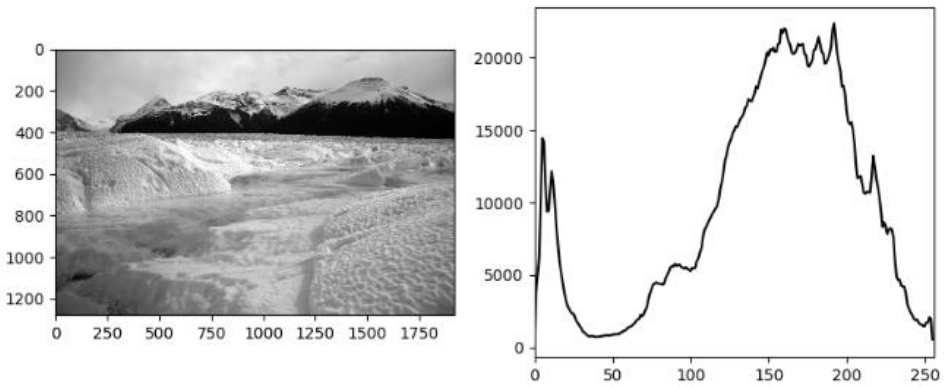


Ilustración 8 - Histograma de una imagen en blanco y negro [10].

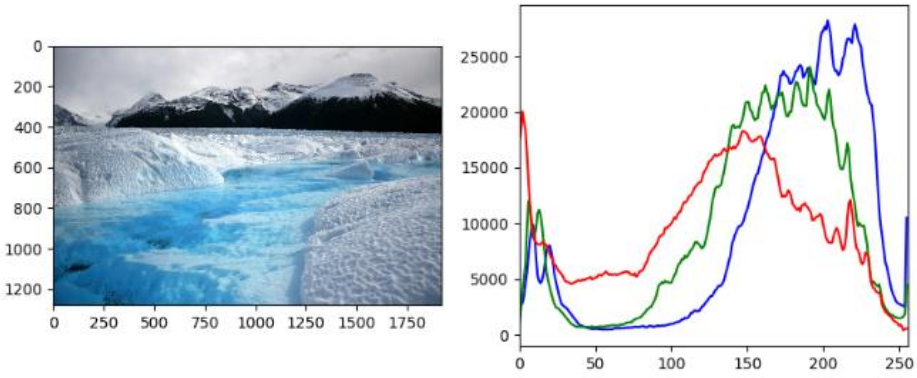


Ilustración 9 - Histograma de una imagen en color con los tres planos RGB [10].

Cuando se construye un histograma, la primera operación a realizar, es la normalización del eje de ordenadas entre 0 y 1, dando lugar un histograma que no dependa del número de píxeles asociados a la imagen.

En lo referente al eje de abscisas, se puede dar el caso en el que todos los valores estén centrados en torno a un punto, generando una campana de gauss o que no exista un patrón claro. A este factor se le conoce como rango dinámico, el cual si es pequeño nos indica que la imagen posee un contraste bajo. Por el contrario, si el rango dinámico ocupa todo el eje de abscisas, se dice que la imagen está saturada.

El análisis de un histograma permite al usuario conocer detalles sobre la calidad y el proceso de adquisición de la imagen. Es decir, si el rango dinámico es amplio se habla de imagen de alta calidad, en cambio, si es muy acotado, la imagen contiene poca información por lo que se suelen descartar de cara a un proceso de segmentación.

Los histogramas serán usados posteriormente en técnicas de segmentación como la binarización (convertir una imagen a blanco y negro exclusivamente) de la cual se hablará posteriormente, pero antes de estudiar estas técnicas se comentarán las operaciones propias sobre un histograma, las cuales pueden mejorar la información de la imagen obtenida [11].

- Expansión del Histograma.

  Consiste en aumentar el rango dinámico de la imagen, transformando las intensidades a través de una función a trozos.

- Ecualización del Histograma.

  Se busca repartir los valores a lo largo del histograma, para obtener uno nuevo con una distribución uniforme. Esta técnica mejora las imágenes saturadas, debido al cambio de los valores de las zonas saturadas a zonas vacías.

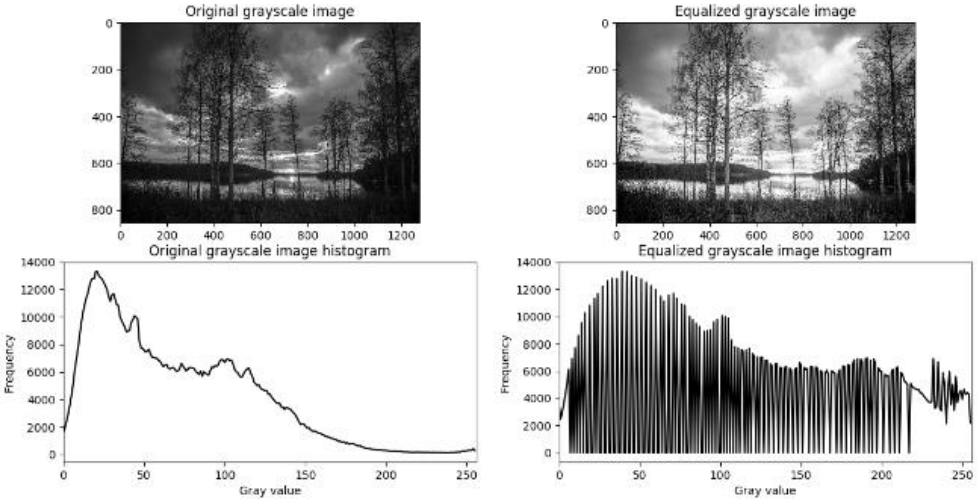


Ilustración 10 - Ecualización del histograma de una imagen en blanco y negro [10].

Filtrado espacial

En este proyecto no se usarán este tipo de operaciones, pero es necesario mencionar su existencia. La razón por la que se obvia su uso, es el elevado coste computacional. Asimismo, el realizar esta operación sobre imágenes que contienen rostros, generaría una gran pérdida de información.

- Filtros paso bajo

Se analiza píxel a píxel la matriz, creando una sub-matriz cuadrada con los píxeles aledaños a este (3x3, 5x5, 7x7…) realizando un promedio de los píxeles adyacentes sobre el evaluado.

Ilustración 11 - Aplicación de filtro paso bajo [7].

Otros ejemplos son el *filtro de la mediana*, el cual sustituye el valor del píxel por el de la mediana de la sub-matriz, o el *filtro del bicho* raro. Este compara la intensidad del píxel examinado con el promedio de los adyacentes, si la diferencia

entre ambos es superior a un umbral marcado anteriormente, se sustituye el valor del píxel por el del promedio.

- Filtros paso alto

Mediante la derivada direccional se pueden observar los cambios de dirección, estos se corresponden con los bordes o límites de las imágenes. Aplicando esto a una imagen, se puede generar otra que contenga los contornos horizontales o verticales de la original. A este proceso se le define como filtro paso alto.

Ilustración 12 - Aplicación filtro paso alto [7].

En esta categoría podemos encontrar los filtros de Sobel, Roberts y Prewitt.

- Filtro de la laplaciana.

Este tipo de filtro hace uso de la segunda derivada, la cual nos indica los cambios de signo en la primera derivada. Aplicando este filtro se obtiene una nueva imagen, la cual es la convolución de la original, tal y como se observa en la ilustración 13.

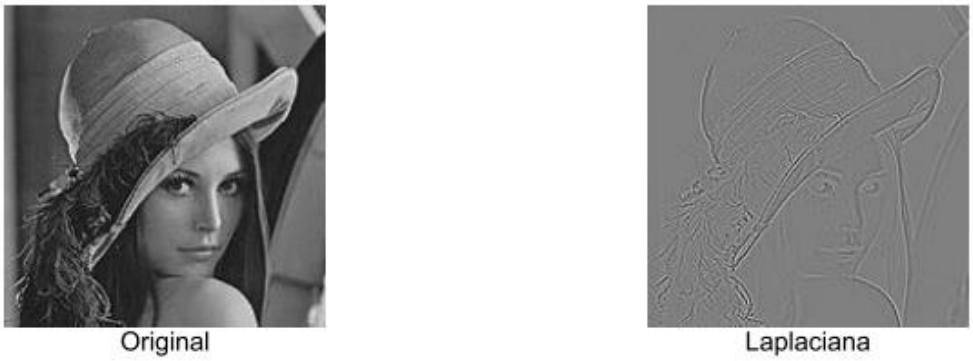


Ilustración 13 - Aplicación filtro Laplaciana [7].

Operaciones morfológicas

Estas operaciones se dedican a tratar y analizar las formas presentes en una imagen. Para favorecerlas, se usan imágenes bitonales o binarias y en menor medida se emplean imágenes en niveles de gris. Las imágenes binarias son conjuntos 2D, ya que solo se contemplan dos colores, blanco o negro. Mientras las imágenes en niveles de gris son conjuntos 3D, siendo la tercera componente la que marque el nivel de intensidad [12].

Las operaciones básicas son:

- Dilatación

Esta operación dilata el aspecto del objeto sobre el que se aplica. Es decir, los píxeles adyacentes al contorno de la forma analizada adquieren el mismo valor, ya sea en el contorno exterior o interior del objeto.

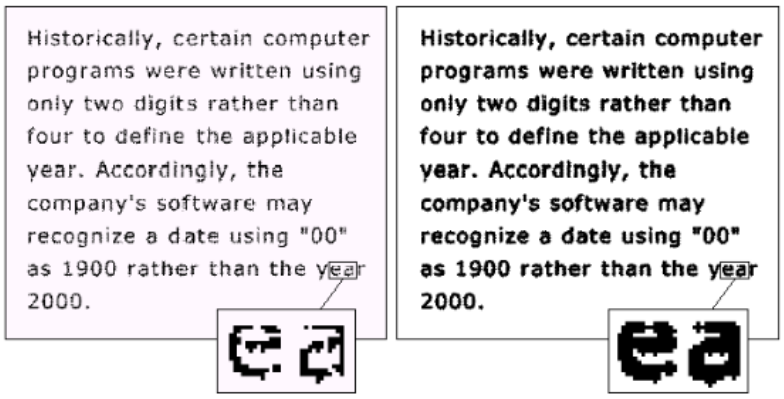


Ilustración 14 - Operación morfológica de dilatación aplicada a un texto [12].

- Erosión

Produce un adelgazamiento de la figura en cuestión, es decir, esta operación produce el efecto contrario a la dilatación. Los píxeles externos de la forma adquieren el valor del fondo de la imagen.

Ilustración 15 - Operación morfológica de erosión. Se aplica una dilatación sobre la segunda imagen, generando la tercera [12].

- Apertura

El aplicar esta operación tiene el fin de suavizar los contornos del objeto y eliminar los posibles salientes generados en una mala binarización. Todo ello se logra combinando las dos operaciones anteriores, primero se realiza una erosión, la cual elimina todos los salientes, para posteriormente emplear una dilatación, recuperando el tamaño del objeto inicial.

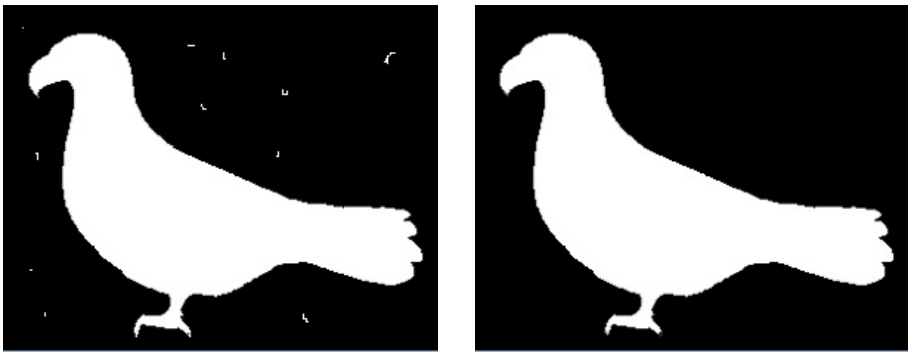


Ilustración 16 - Operación morfológica de apertura [12].

- Cierre

Las operaciones de cierre y apertura son comunes, ya que ambas emplean dilatación y erosión. Pero al contrario que en el caso de apertura, en este se emplea primeramente el proceso de dilatación seguido de una erosión.

El objetivo de aplicar un cierre es eliminar los huecos internos de la forma analizada e incluso de suavizar los golfos que esta posea.

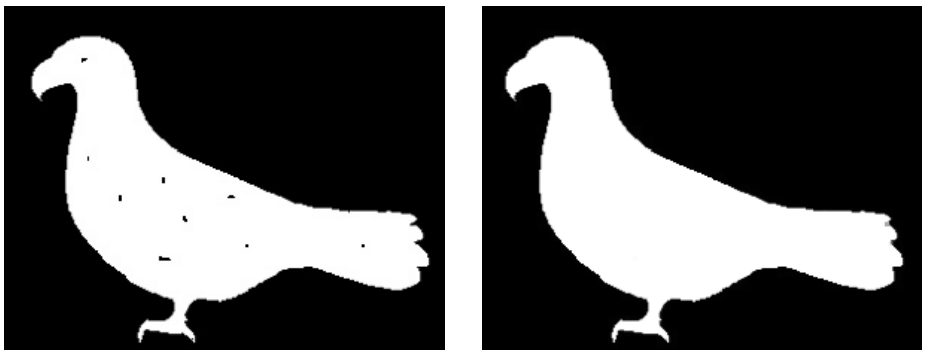


Ilustración 17 - Operación morfológica de cierre [12].

Dentro de las operaciones morfológicas se deben tener en cuenta a mayores los filtros morfológicos. Estos emplean las operaciones morfológicas anteriormente descritas con el fin de mejorar o acomodar el procesamiento de la imagen capturada [13]. A continuación, se indican los que serán usados en este proyecto.

- Eliminación de ruido

Al adquirir imágenes se puede dar la casuística de que estas presenten perturbaciones. Estas son conocidas como ruido, véase la imagen original de la ilustración 18.

Ilustración 18 - Filtros morfológicos dedicados a la eliminación de ruido [12].

Estos filtros, en esencia, eliminarán los objetos menores de un área señalada mediante un proceso de apertura. Posteriormente buscará recuperar la misma forma realizando un cierre.

- Extracción de contornos

Su finalidad es dar una descripción a los objetos presentes en la imagen. Existen múltiples algoritmos para realizar este proceso, los cuales se emplean en diferentes situaciones, como al tener una gran presencia de objetos pequeños o una única existencia de un elemento con gran detalle (rueda dentada) en la imagen.

Ilustración 19 - Aplicación filtro Canny para extraer contornos [14].

El algoritmo base de este filtro aplica una erosión en las zonas claras de la imagen y una dilatación en las zonas oscuras, obteniendo dos imágenes. Sucesivamente se realiza una resta entre ambas obteniendo el contorno del objeto en cuestión.

- Relleno de agujeros

Para aplicar este filtro es necesario conocer las zonas que se desean rellenar. El proceso comenzaría localizando un punto del vacío, es indiferente donde esté situado. Para rellenar el vacío se realiza una iteración constante similar a la operación de dilatación, la cual va asignando el valor de intensidad del objeto a los píxeles vacíos.

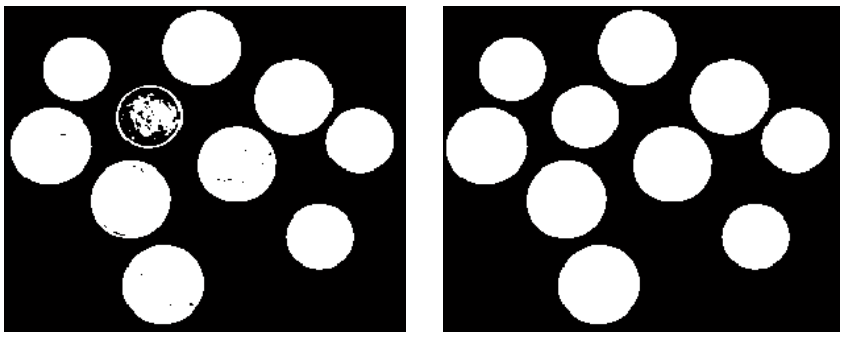


Ilustración 20 - Aplicación filtro Imfill a una imagen binarizada [15].

- Adelgazamiento

Como indica el propio nombre del filtro, éste consiste en adelgazar los objetos presentes en la imagen hasta obtenerlos de forma esquemática.

Se aplicará la operación de erosión sobre las zonas que estén sobre definidas, es decir, se erosionarán los bordes hasta obtener un esquema de la pieza en el cual se puedan percibir mejor los detalles. Un claro ejemplo de aplicación es sobre una rueda dentada, ya que aplicando este filtro se pretende obtener solo el esqueleto de la imagen, los dientes y el cuerpo, facilitando la enumeración de dientes presentes en la imagen.

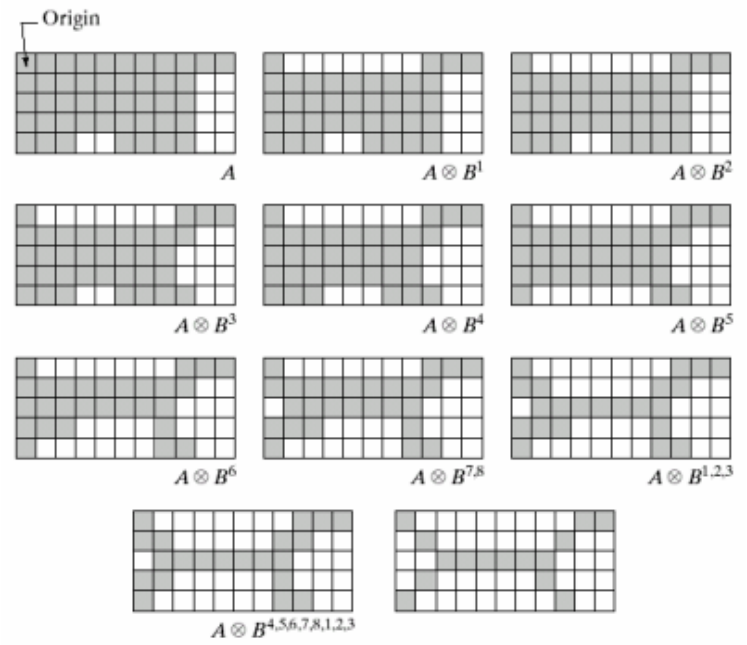


Ilustración 21 - Aplicación filtro adelgazamiento [12].

2.1.3.3 Segmentación

Una vez se ha realizado un procesamiento previo de la imagen capturada, el cual ha mejorado la calidad de ésta y/o de los elementos presentes, se busca segmentar la foto para facilitar el análisis digital o reconocimiento automático.

Este fin se logra dividiendo la imagen en zonas homogéneas referentes a una sola característica, como pueden ser el color o el brillo. Este proceso no es algo estandarizado y aplicable a cualquier imagen, depende del problema y del objetivo que se persiga, siendo así necesario o no aplicar este proceso [7].

Como se ha mencionado aplicar este proceso no es imprescindible. Por ejemplo, si se aplica segmentación a una fotografía paisajística se crearán una multitud de regiones homogéneas, las cuales no transmitirán ninguna información relevante sobre la imagen.

Un ejemplo válido es el separar una palabra en cada letra, obteniendo tantas nuevas imágenes como caracteres tenga la misma.

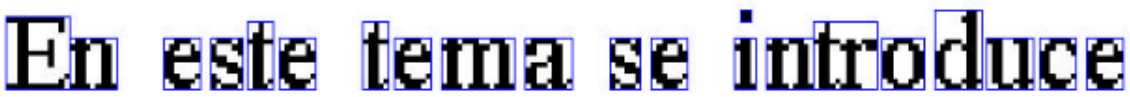


Ilustración 22 - Frase segmentada. Cada carácter es etiquetado [7].

Segmentar una imagen genera una nueva, en la que cada elemento esté etiquetado con sus características, pudiendo crear una lista con los objetos presentes en la imagen.

Para clasificar los elementos de una imagen se atiende a factores como son los bordes o la textura, de los cuales se hablará a continuación.

Textura

La rugosidad o regularidad presente en una imagen, en sus diferentes grados, se puede definir como la textura de la misma.

"Uno o más patrones locales que se repiten de manera periódica" [7].

Definiendo una textura se verán dos enfoques, descendente y ascendente. El primero, también llamado *"top-down"*, se basa en un téxel (elemento básico de textura) y en una regla o patrón que sitúa donde se encuentran estos. Este enfoque se emplea en imágenes regulares, véase un muro de ladrillos.

Por otro lado, se encuentra el enfoque ascendente o *"bottom-up"* en el que a través de probabilidad busca componentes difícilmente visibles provocados por el entorno, el cual no sigue un patrón establecido. Un ejemplo sería una imagen cuyo fondo sea hierba.

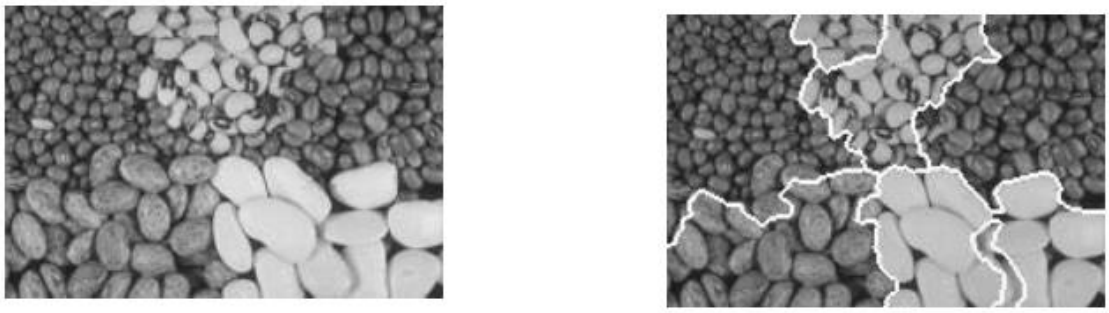


Ilustración 23 - Segmentación basada en textura.

El proyecto utilizará este enfoque de segmentación de cara a extraer características de los rostros a analizar, tales como la edad y el género. Por ello este apartado lo veremos con mayor profundidad en posteriores puntos de la presente memoria.

Segmentación por umbral

La umbralización convierte una imagen en niveles de gris en una binarizada, es decir, los valores que tomarán los píxeles serán únicamente 0 o 1, identificando blanco o negro. Este proceso tiene bajo coste computacional, por lo que puede ser ejecutado al mismo tiempo que se captura la imagen.

Se deben distinguir dos tipos, umbralización fija y generalizada.

La primera actúa en función de un umbral fijado con anterioridad, cambiando el valor de todos los píxeles con valores inferiores a este por 0 y los superiores por 1.

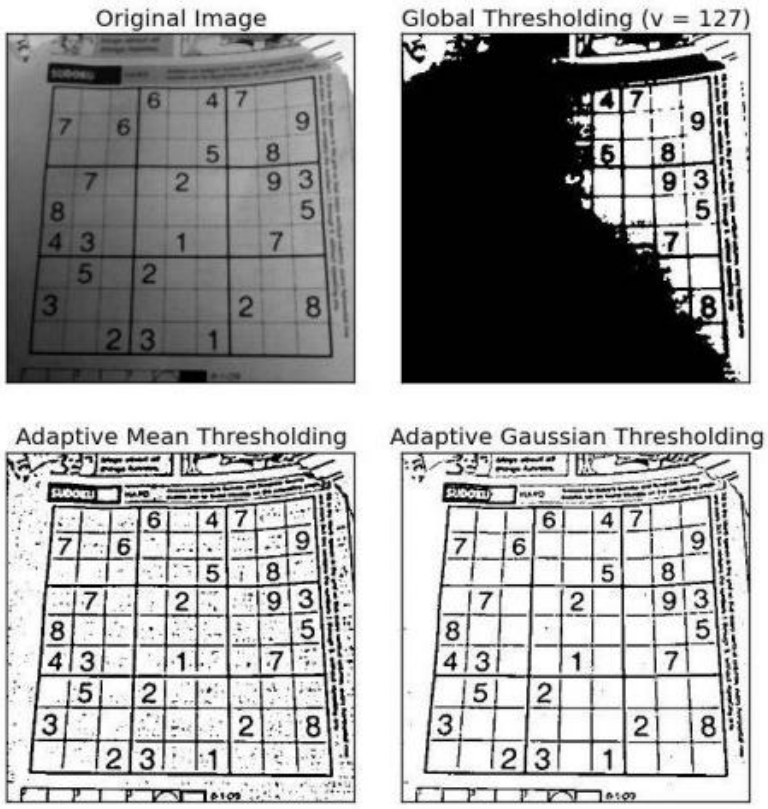


Ilustración 24 - Segmentación por umbralización aplicando diferentes algoritmos [16].

Este tipo de binarización requiere un estudio previo por parte del usuario para conocer el umbral que permite obtener un resultado adecuado a las características que se quieren remarcar de la imagen. Para facilitar esta elección se suele usar el histograma de la función, el cual indica dónde se encuentran la mayor parte de los valores de la imagen, lugar del que se parte como referencia.

En lo referente a la binarización generalizada se procede a hablar del caso adaptativo, por ser el tipo que se usará en el código de la aplicación. Mencionar que, entre los tipos restantes destaca la umbralización de banda, en la que hay 2 umbrales siendo el interior de estos lo que quedará a 1 y el resto a 0.

La umbralización adaptativa es la única que no necesita fijar el valor de un umbral sobre el cual realizar la binarización.

El algoritmo funciona dividiendo la imagen original en subimágenes sobre las cuales determina un umbral independiente. En caso de no poder elegir el umbral en una subimagen asignará un umbral interpolando los umbrales vecinos. Posteriormente, procesa cada subimagen con su umbral y reúne estas para conformar la imagen original binarizada. Este algoritmo lo podemos ver en la ilustración 24. Las dos imágenes inferiores surgen de aplicar binarización adaptativa. Por lo expuesto anteriormente, se deduce que la aplicación de este tipo de binarización es en imágenes con iluminación no uniforme.

Detección de bordes

Este proceso de segmentación tiene como funcionalidad reconocer los objetos que poseen el mismo color, diferenciándolos del fondo y obteniéndolos por separado, como se puede apreciar en la ilustración 25. Para localizar el núcleo se ha procedido mediante un proceso iterativo, despreciando en cada iteración aquellas partes de la imagen con diferente tonalidad de la que se busca.

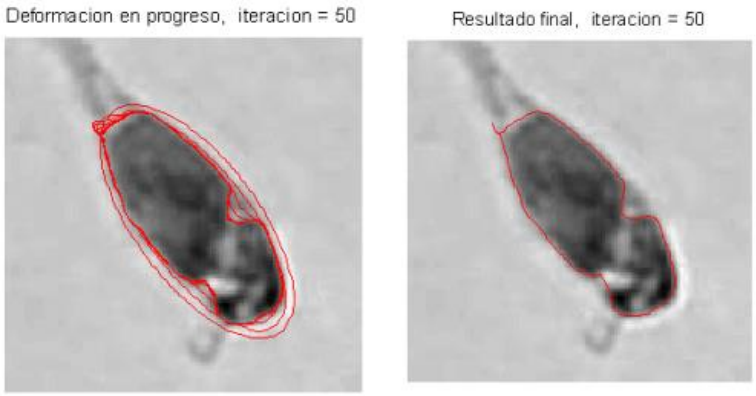


Ilustración 25 - Segmentación del núcleo mediante la detección del borde [7].

Para facilitar este proceso se suele trabajar con imágenes binarizadas, pero no es estrictamente necesario, siendo posible usar imágenes en niveles de gris o en color. Para conseguir esto es necesario aportar información adicional sobre los objetos presentes en la imagen.

El algoritmo básico se encarga de etiquetar cada píxel asociándolo a un conjunto, dando así lugar a un objeto presente en la imagen. Para ello, se recorre la imagen de izquierda a derecha y de arriba abajo buscando un píxel negro. Una vez situado en él observa los vecinos, en caso de ser píxeles negros continúa moviéndose a ellos y etiquetándolos con el mismo valor asignado al primero. Este proceso se realiza iterativamente hasta que todos los píxeles vecinos del conjunto poseen un valor distinto. Posteriormente el algoritmo busca otro píxel negro en la imagen repitiendo el proceso de forma recursiva hasta haber examinado la imagen en su totalidad.

2.1.3.4 Clasificadores

Lo anteriormente expuesto facilita el análisis de una imagen, pudiendo devolver al usuario cierta información de forma automática, como por ejemplo el número de elementos presentes en una imagen. Pero, para identificar qué objetos son cada elemento de la imagen es necesaria la presencia de un operario humano o de elaborar un software robusto, el cual pueda salvar cualquier imprevisto.

Ilustración 26 - Ejemplo de clasificación morfológica.

La ilustración 26, ejemplifica cómo mediante un proceso morfológico, es decir, valiéndose del tamaño y forma, se puede llegar a clasificar objetos y etiquetarlos. Como se puede prever, esto es válido en entornos muy estandarizados, ya que un mínimo cambio de luminosidad, puede malograr la clasificación.

Para facilitar este proceso y llevarlo a un plano automático se usan clasificadores. Estos, son algoritmos que permiten clasificar los objetos presentes en una imagen diferenciándolos entre ellos y asignándoles nombres propios tales como tornillo, arandela o tuerca.

El presente proyecto se vale de un clasificador estandarizado para reconocer los rostros presentes en una imagen conocido como *"HaarCascade"*. Por lo que se pretende describir el uso y empleo de este descriptor.

HaarCascades

El proceso de aplicación de estos descriptores se basa en las siguientes tres etapas [17]:

Primero, se genera una imagen integral seguida de una extracción de características, para finalmente realizar la clasificación buscada.

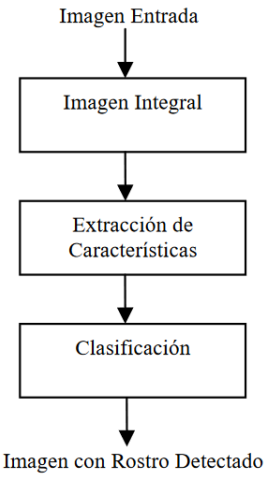


Ilustración 27 - Flujo descriptor HaarCascade [17].

Para formar la imagen integral, como indica su nombre, se realiza la integral del sumatorio del píxel evaluado y de los vecinos. Está imagen permite conocer características a diferentes niveles de interpretación, ya que posteriormente no se trabajará con los valores de intensidad de los píxeles sino con una imagen acumulativa.

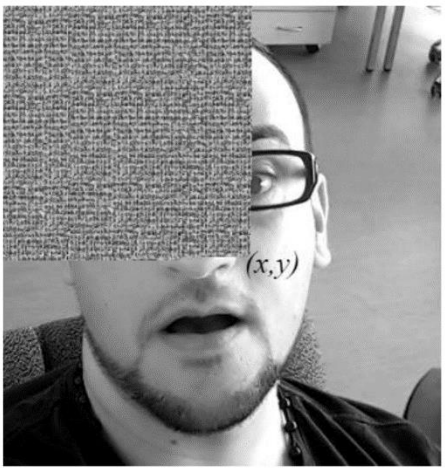


Ilustración 28 - Ejemplo de imagen integral [17].

Este descriptor aplica posteriormente filtros con bases *"Haar"* para lograr la extracción de características de la imagen de estudio. Estos son calculados sobre la imagen integral obtenida, permitiendo discriminar la orientación espacial del objeto,

al igual que su frecuencia. En la ilustración 29 se puede ver una muestra de estos y de cómo pueden verse modificados respecto a su escala y orientación.

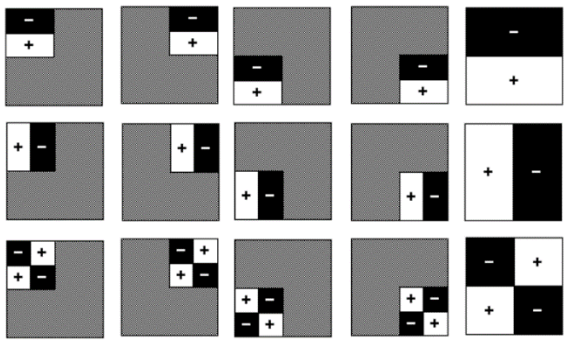


Ilustración 29 - Filtros Haar rotados, trasladados y con cambios de escala [17].

Los filtros con base *"Haar"* codifican las diferencias de valor de los píxeles obteniendo los contornos, todo ello contrastando las regiones que comparan con cada iteración.

Por último, el algoritmo entra en la etapa de clasificación, donde asignará al conjunto la clase con la que encuentre una mayor similitud en función de los templates o patrones introducidos inicialmente.

Ilustración 30 - Aplicación descriptor HaarCascade [18].

En la ilustración 30, se aprecia el reconocimiento de medio perfil, rostro, ojos y nariz. Para cada región se ha usado un template diferente.

Se puede encontrar en internet una gran cantidad de clasificadores que funcionen con la metodología *"haarcascade"*. Entre ellos destacan la detección de vehículos o viandantes, la localización de rostros en una imagen, como es el ejemplo anterior, o incluso rostros de mascotas.

2.1.4 Aplicaciones

Atendiendo a lo previamente indicado a lo largo de este apartado de visión artificial, se pretende, como punto final, dar una fotografía de las aplicaciones en las que más involucrada está dicha tecnología.

Actualmente, la visión por ordenador no es desarrollada únicamente por grandes compañías, sino por usuarios de menor relevancia. Debido al gran campo que abarca esta disciplina, solo se mentarán aplicaciones de uso industrial y/o comercial.

2.1.4.1 Visión industrial

La expresión “industria 4.0” se empezó a acuñar en el año 2011, debido a la revolución que estaban sufriendo las factorías por la implementación de inteligencia artificial, Big Data y visión artificial, siendo esta última de gran ayuda para automatizar líneas completas de producción.

En esta categoría, la visión artificial trabaja junto con sensores y actuadores en pro de favorecer los movimientos que deben realizar los brazos robóticos a la hora de colocar una pieza de forma determinada o ayudando en la orientación y movimiento de las plataformas de transporte de material. Las aplicaciones más usadas en este sector son las siguientes.

- Geometría Epipolar

  Al capturar una imagen se pierde información como puede ser la profundidad, la cual es apreciable por un ser humano sin ninguna dificultad. Esto se debe a la transformación espacial realizada al pasar de un entorno 3D como es la realidad a una imagen 2D [19].

  Para solventar este problema, se decide imitar el esquema de los ojos humanos utilizando 2 cámaras para generar una “visión estéreo”.

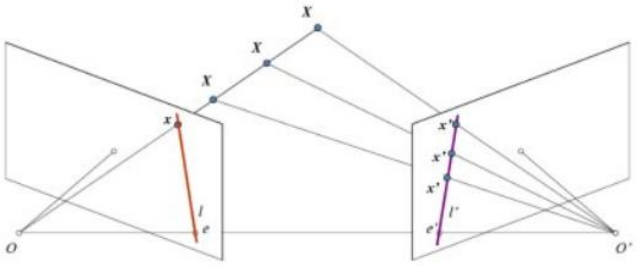


Ilustración 31 - Esquema configuración "visión estéreo" [19].

Como se observa en la ilustración 32, la combinación de ambas cámaras permite asignar a cada píxel una profundidad. Un ejemplo claro en la industria que emplea esta tecnología son las carretillas de transporte que emplea *Amazon©*.

Ilustración 32 – Ejemplo uso visión industrial con geometría epipolar [20].

- Reconocimiento de objetos

A nivel industrial, los sistemas más utilizados de visión por ordenador son los siguientes [21]:

  - Sistema de presencia – ausencia

    Se realiza una inspección del área en la que se emite un valor u otro en función de la existencia de una pieza o elemento en la zona analizada. Para ello, se extrae el fondo en el que va a estar funcionando nuestro sistema, posteriormente se compara este fondo con la imagen actual para ver la presencia o ausencia de la pieza.

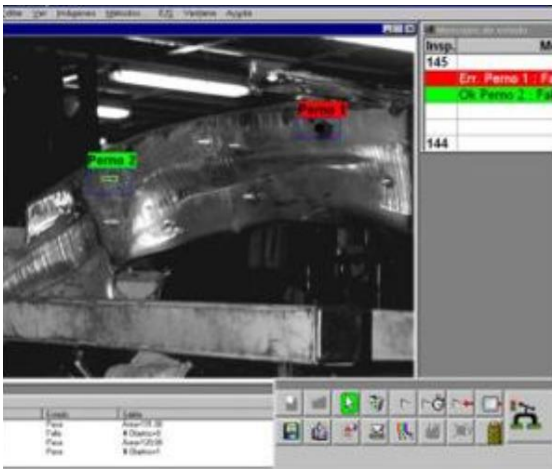


Ilustración 33 - Ejemplo sistema de presencia – ausencia [21].

- Sistema de pick-up & place

También llamado guiado de robots. Identifica la situación de un objeto al mismo tiempo que le asigna unas coordenadas para que pueda ser recogido por un brazo robótico. Posteriormente, detecta que el objeto ha sido entregado en su destino.

Ilustración 34 - Ejemplo sistema de pick-up & place [21].

- Sistema de control de calidad

Como indica el propio nombre, este sistema verifica que se cumplan ciertos objetivos y especificaciones en una pieza. La implementación requiere usar cámaras de gran precisión, es decir, se necesita tomar imágenes con una alta resolución para no perder información.

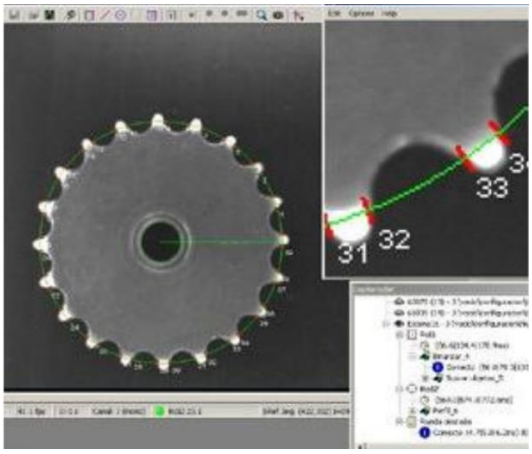


Ilustración 35 - Ejemplo sistema de calidad [21].

- Sistema de metrología

Mediante visión artificial se pueden inspeccionar y analizar características de una pieza tales como sus dimensiones, superficie, distancia entre bordes, ángulos, posición de taladros…

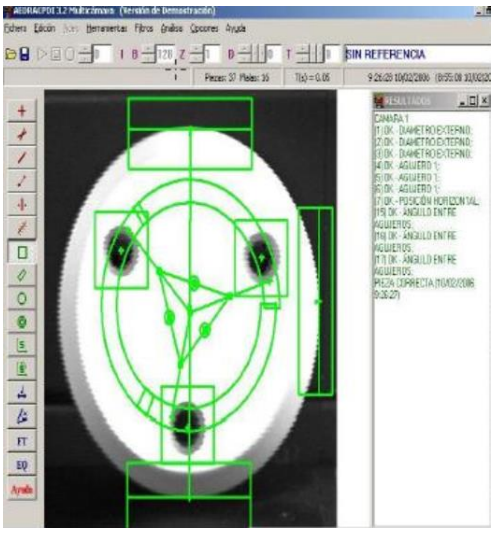


Ilustración 36 - Ejemplo sistema de metrología [21].

- Sistema de clasificación industrial

Como se ha mencionado anteriormente, se pueden utilizar templates para favorecer la clasificación de objetos. En este caso, permiten identificar por área, color, código de barras, patrón…

Ilustración 37 - Ejemplo sistema de clasificación [21].

2.1.4.2 Reconocimiento óptico de caracteres

En este ámbito se pueden encontrar aplicaciones muy variadas, tanto para uso corporativo como para uso público.

Los radares de tráfico son un ejemplo de uso empresarial. La aplicación de visión artificial permite reconocer las matrículas de las imágenes que se obtienen, transformando la información de la imagen a caracteres con los cuales se puede trabajar.

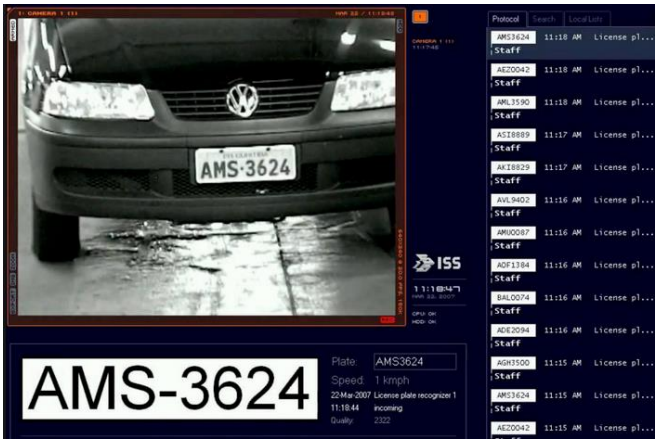


Ilustración 38 - Ejemplo reconocimiento y segmentación de una matrícula [22].

En cuanto a aplicaciones para usuarios finales, se puede ver el crecimiento del OCR, el cual consiste en transcribir el texto presente en una imagen a formato digital, exportándolo a *Microsoft Word©* o a algún otro editor de texto.

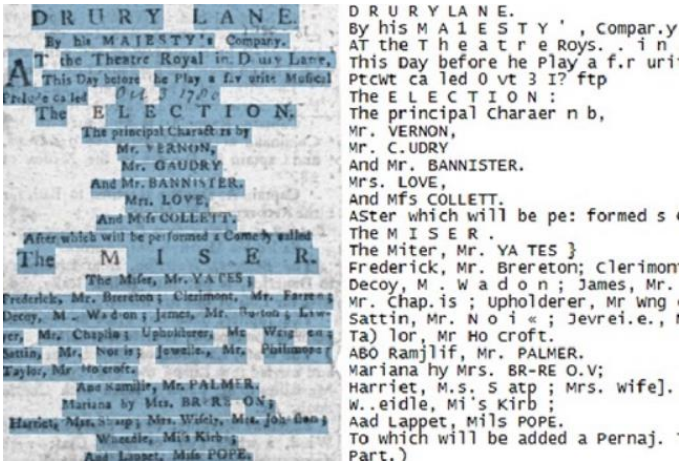


Ilustración 39 - Ejemplo reconocimiento óptico de caracteres. OCR [23].

Como ejemplo de aplicaciones con este fin podemos ver *Adobe Acrobat©* o *Tesseract©*, siendo esta última *"Open Source"*.

2.1.4.3 Teledetección

Debido a la gran cantidad de imágenes que se obtienen hoy en día de los satélites y a la información que estas ofrecen, por la mejora de la resolución, el ámbito de la teledetección está en auge [24].

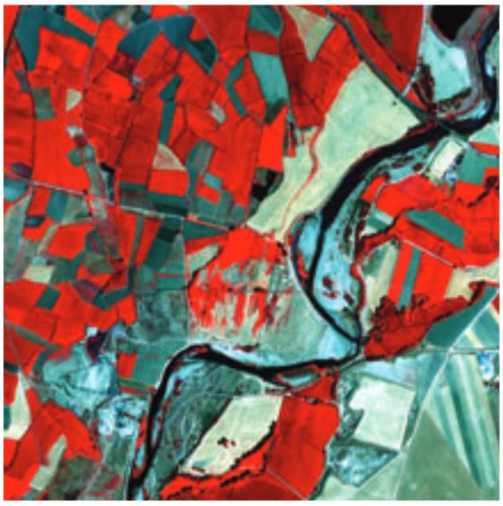


Ilustración 40 - Ejemplo Teledetección. En rojo zonas agrícolas de un mismo cereal [24].

Se trabaja con algoritmos de corrección geométrica, software de detección de objetos, reconocimiento de patrones…. Todo ello con el fin de obtener software que ayude en temas como el catastro, hidrología, agricultura, urbanismo, explotaciones forestales, meteorología….

2.1.4.4 Reconocimiento facial

Estas aplicaciones permiten identificar rostros analizando las características del mismo. La identificación de estas características se basa en procesos matemáticos y algoritmos de coincidencia [25].

Este proceso se distingue por no ser intrusivo a la hora de ejecutarse, pudiendo ser completamente autónomo sin tener necesidad de un controlador humano. Se pueden diferenciar en dos vertientes principales:

- Sistemas geométricos

  Se vale de los rasgos característicos de un rostro humano para ubicar la imagen, siendo estos atributos los ojos, nariz, boca…

- Sistemas fotométricos

Buscan coincidencia completa del rostro con imágenes almacenadas en bases de datos. Estos sistemas se usan para seguridad principalmente.

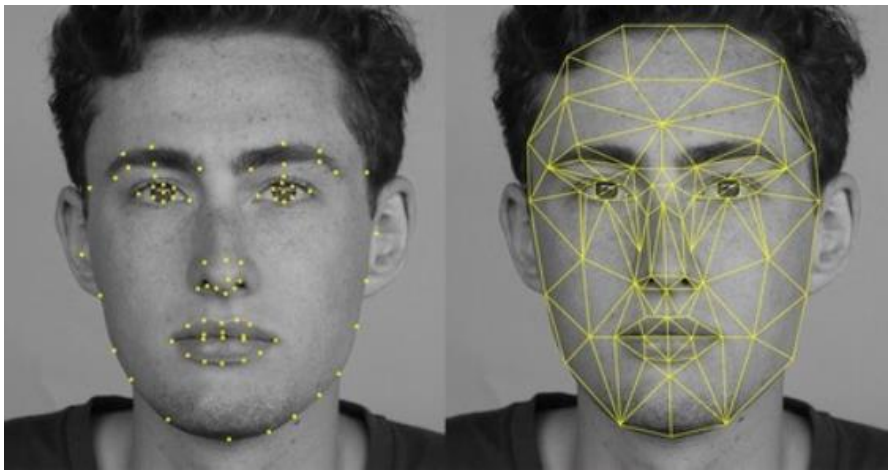


Ilustración 41 - Ejemplo reconocimiento facial biométrico [26].

En este proyecto se busca utilizar un sistema geométrico de búsqueda facial, empleando el descriptor "haarcascade" explicado anteriormente. En el capítulo IV del proyecto se explicará cómo se emplea esta tecnología a nivel práctico.

2.2 Tecnologías actuales

Se procede a exponer en este apartado todas aquellas tecnologías y aplicaciones que se encuentran en el mercado y que poseen la misma finalidad que el software que se va a desarrollar.

Para ello, se plantea dividir esta búsqueda, en aplicaciones destinadas al reconocimiento facial, explicando que finalidades tienen que cumplir y las aplicaciones que se usan para reconocer el género y la edad.

2.2.1 Reconocimiento facial

Se entiende por aplicación de reconocimiento facial, un software que identifica un rostro en una imagen digital o video. Esta definición puede ser aplicada a muchos ámbitos de la vida cotidiana, generando con ello múltiples sistemas en los cuales se puede aplicar esta tecnología.

Esta se basa en realizar un análisis de los rasgos característicos del individuo a estudiar y comparar dichos rasgos con los almacenados en una base de datos. Como se puede inferir, esta tecnología es usada principalmente en ciberseguridad, ya que el rostro humano viene a ser como una huella dactilar, siendo algo único y personal.

Pero no solo se reducen a este ámbito las aplicaciones, sino que se pueden encontrar en diferentes casuísticas menos comunes.

2.2.1.1 Pagos por reconocimiento facial

Face++© es una empresa china puntera en el sector del reconocimiento facial, ofreciendo múltiples aplicaciones a otras empresas que necesiten usar esta tecnología, como por ejemplo *Alipay*© [27].

Esta empresa es la plataforma líder de pagos online en china. El software que se emplea es una comparación de rostros, entre el capturado por la cámara y el almacenado previamente. Para realizar el pago, el usuario únicamente ha de sonreír a la cámara.

Con esta misma finalidad, se están abriendo en Pekín tiendas automatizadas sin personal. Los clientes usan una aplicación móvil para poder acceder a la tienda. Una vez dentro están vigilados por varias cámaras de seguridad programadas para detectar posibles hurtos. Para finalizar la compra, el pago se realiza cuando el usuario enseña su rostro a una cámara situada a la salida, la cual efectúa el pago y habilita las puertas para permitir la salida.

2.2.1.2 Localización de personas desaparecidas

Entre 2012 y 2017 desaparecieron más de 240.000 niños en la India, es por ello que el gobierno puso en funcionamiento un programa llamado *"TrackChild"*. Se basa en una base de datos en línea, en la cual se sube una imagen del niño en cuestión y a través de algoritmos fotométricos se compara la fotografía con las imágenes almacenadas de las cámaras de videovigilancia en busca de posibles coincidencias [28].

Este software fue desarrollado por *Bachpan Bachao Andolan*©, valiéndose de la tecnología que nos ocupa llegaron a identificar cerca de 3000 niños partiendo de una base de datos de 45000 fotografías.

Al ver este éxito, otras compañías de diferentes países se han propuesto reproducirlo. Pero han entrado en conflicto con obstáculos legales, por lo que actualmente estas herramientas se limitan a ser usadas por agencias gubernamentales con fines de búsqueda de fugitivos o criminales.

2.2.1.3 Tramites aeroportuarios

Con el fin de ahorrar las largas colas de acceso a los vuelos, el aeropuerto de Orlando, implementó en 2018 en todos sus vuelos, tecnología de reconocimiento facial. Mediante el empleo de cámaras de alta resolución, el sistema permite capturar una fotografía del pasajero y los documentos, como el pasaporte y el billete. Una vez obtenida esta, se compara con la base de datos que posee la Agencia de Aduanas. Este proceso se realiza en aproximadamente 2 segundos con un porcentaje de acierto del 99% [29].

2.2.1.4 Desbloqueo facial terminal móvil

Los usuarios de *iPhone X*© o de los nuevos terminales de *Samsung*©, poseen está tecnología en sus dispositivos, la cual permite el desbloqueo inmediato mediante el rostro del titular.

Se basa en el uso de la cámara frontal del dispositivo, la cual se activa al detectar presencia mediante infrarrojos. Mediante esta cámara infrarroja se fija la posición de la cara, la forma, el tamaño de los ojos y la boca, formando un sistema de puntos 3D del rostro, para ser comparado posteriormente con el patrón almacenado en la base de datos del dispositivo.

2.2.1.5 Inclusión de personas con discapacidad visual

La firma *Novartis*© junto con el departamento *Microsoft Cognitive Services*© de la misma marca, han lanzado la aplicación *"ViaOpta Hello"*, la cual es pionera en el campo de ayuda a personas con discapacidad visual o ceguera [30].

Dicho software identifica los objetos y personas presentes en una estancia mediante el uso de un dispositivo móvil. Primeramente, se toma una foto y automáticamente se le informa al usuario de lo que tiene a su alrededor a través del altavoz del dispositivo. Para reconocer a personas es necesario que el usuario de la app registre a sus amigos o familiares anteriormente.

2.2.1.6 Accesos restringidos

Como se puede prever, esta tecnología tiene un gran interés para permitir el acceso a personal autorizado a instalaciones de seguridad. Es por ello que se está extendiendo este uso a otros ámbitos, los cuales están más ligados con la higiene [31].

Ciertos hospitales de Estados Unidos, han incorporado sistemas de reconocimiento facial a la entrada de quirófanos o salas blancas, evitando con ello el contacto entre el médico y la puerta, manteniendo la higiene y esterilidad. Dicha implementación es válida para mantener un control de acceso y de presencia.

2.2.2 Reconocimiento de género y edad

La aplicación que se pretende desarrollar está enfocada en este campo, es decir, en la extracción de características a través del rostro capturado. Este campo se comenzó a explotar a lo largo de 2014, ya que fue el año de lanzamiento de diversos dispositivos móviles como el Mi4, perteneciente a la firma *Xiaomi©*.

Este terminal fue de los primeros en incluir en la aplicación nativa de la cámara un software que permitiera analizar mediante los "selfies" la edad y el género de la persona fotografiada [32].

Recientemente se ha empezado a dar mayor importancia a este tipo de aplicaciones y a su potencial, ya que este tipo de tecnología había sido relegado únicamente al ocio.

Ciertas empresas como *Smart Me Up©*, expusieron en el CES de 2016, nuevas aplicaciones que daban un paso adelante en el reconocimiento facial y en la detección de características [33]. El software presentado por la compañía francesa realiza un análisis continuo en tiempo real del individuo presente ante la cámara. Con el paso del tiempo el resultado es cada vez más preciso, llegando a acotar casi a la perfección la edad, el género o estados de ánimo, como signos de cansancio, la atención o algún otro tipo de emoción.

La compañía propuso como posibles integraciones de este software los ámbitos de los vehículos autónomos, hogares inteligentes, análisis de consumidores en los comercios o robótica.

Este último es el más interesante, ya que se facilitaría en un futuro la integración de robots colaborativos junto con los humanos, analizando las emociones presentes en los usuarios cercanos.

CAPITULO III

3 Herramientas software

En este apartado se pretende realizar un estudio exhaustivo de las tecnologías en las cuales se va a apoyar el desarrollo del software anteriormente explicado.

Para ello, se realizará una comparativa entre las diferentes propuestas que ofrece el mercado, exponiendo las ventajas e inconvenientes entre ellas. Todo ello primando siempre el software libre, por ser uno de los objetivos que se marca este proyecto.

Se comenzará el análisis desde un enfoque superlativo, para ir reduciendo el rango de elección con cada herramienta seleccionada.

3.1 Sistema operativo

Se comenzará eligiendo el sistema operativo, ya que a la hora de seguir eligiendo herramientas limitará las opciones. Los sistemas operativos más populares e importantes a día de hoy son *MacOS©*, *Windows©* y *Linux©*.

A continuación, se van a exponer los puntos fuertes de cada SO, haciendo referencia a las ventajas que ofrecen a la hora de programar en ellos.

3.1.1 MacOS Mojave

El actual sistema operativo de la compañía *Apple©* es *MacOS Mojave©*. Dicho SO, al igual que sus predecesores, está escrito en los lenguajes C, C++, Objective – C y Swift.

Esta plataforma ha sido enfocada a lo largo de los años al diseño gráfico, es decir, *Apple©* ha buscado una simetría entre software y hardware a la hora de reducir el impacto de los recursos visuales consumidos, lo que facilita el uso de programas de diseño como *Illustrator©*, *Photoshop©*, *Premiere©*....

En lo referente a la programación, *MacOS©*, está orientado al desarrollo de aplicaciones para sus dispositivos, es decir, para ordenadores *Macintosh©* y dispositivos móviles o tablets con *iOS©*. Este hecho es contrario al desarrollo de la aplicación, ya que se requiere que esta pueda ser multiplataforma.

Asimismo, el coste de un dispositivo *Macintosh©* es muy elevado. Este hecho, sumado a lo anteriormente comentado, ocasiona el rechazo de este sistema operativo en el desarrollo de este proyecto.

3.1.2 Windows 10

Es el sistema operativo más vendido y usado a nivel mundial. Perteneciente a *Microsoft©,* está codificado en C++ y lenguaje ensamblador.

Windows 10© es la versión más reciente, la cual ofrece una compatibilidad absoluta con cualquier software privativo. Este sistema operativo puede ser instalado en cualquier equipo, permitiendo así, la libre elección del hardware en el que se quiere ejecutar.

Los desarrolladores eligen esta plataforma principalmente porque posee todas las herramientas dedicadas al desarrollo de videojuegos y animaciones. Aunque *Windows 10©*, es mucho más versátil, permitiendo usar todo tipo de herramienta de programación.

3.1.3 Linux

Es un sistema operativo basado en *UNIX©*, el cual está basado en software libre y es de código abierto. El núcleo *Linux©*, está escrito en C y ensamblador, lo que favorece el rendimiento por encima de los anteriores sistemas operativos.

Linux© cuenta con multitud de distribuciones, las cuales se actualizan de forma paralela, siempre bajo una comunidad de desarrolladores pública. Se pueden mencionar *Ubuntu©*, *Devian©*, *Arch Linux©* o *Linux Mint©*, como las distribuciones más notables de esta plataforma.

El mayor inconveniente es la escasez de software privativo como *Adobe©* o *Autodesk©*, debido a que es una plataforma minoritaria. Aunque en el presente proyecto, la falta de dicho software no impide la viabilidad del mismo.

El principal punto fuerte de *Linux©* a la hora de programar, es su uso de la "multitarea", la cual permite la ejecución de varios procesos conectados entre sí al mismo tiempo en un único equipo, siendo algo muy atractivo a la hora de realizar desarrollos de la misma índole que el aquí expuesto.

Windows© y *Linux©* son muy similares, pero la opción multitarea y el rendimiento que ofrece *Linux©* lo hacen ser la elección definitiva de cara al desarrollo de este proyecto. Con se opta por la distribución *Ubuntu©*, debido a su estabilidad y su comunidad de desarrolladores.

3.2 Librerías de visión artificial

Para llevar a cabo este proyecto es necesario elegir una librería de visión artificial que facilite el desarrollo, de tal forma que se aprovechen las funciones predefinidas que ofrecen estos paquetes.

Se analizarán exclusivamente las librerías de código abierto (*open-source*), más influyentes. Mentar la existencia de empresas desarrolladoras de este tipo de software con fines privados como *Sherlock*, desarrollado por *Dalsa©* o *Halcon* comercializado por *MVTec©*.

3.2.1 Point Cloud Library

Es una librería que permite trabajar con nubes de puntos 3D, lo cual favorece el procesamiento geométrico. Contiene algoritmos dedicados a la reconstrucción de superficies, tratamiento 3D o segmentación de objetos. Fue diseñada y lanzada por *Willow Garage©* en marzo de 2010, siendo desarrollada en C++.

Entre las funciones que posee dicha librería destacan [34]:

- Filtros.
- Extracción de características.
- Kdtree y octree.
- Segmentación.
- Reconocimiento de patrones.

Esta librería está diseñada para trabajar en entornos de 3D, lo que genera una gran carga computacional. Teniendo esto en cuenta y que el presente proyecto pretende trabajar únicamente con imágenes planas, se descarta el uso de esta librería.

3.2.2 JavaVis

El desarrollo de esta librería nace como un trabajo de la Universidad de Alicante en 1998/99. El lenguaje en el que se ha desarrollado es C bajo Unix/Linux, pero la implementación, como su propio nombre indica, es a través de Java [35].

La ventaja de *JavaVis©* es la facilidad de adición de algoritmos ya codificados, los cuales simplifican las tareas, como puede ser el almacenamiento en baterías de frames capturados. Los principales puntos débiles de este framework, son su portabilidad y su interfaz gráfica, la cual carece de muchas herramientas, que

incorporan sus competidoras. Por ello, esta librería no entra en consideración para realizar este proyecto.

3.2.3 Matlab

Este sistema, el cual ofrece una gran variedad de aplicaciones permite trabajar y procesar imágenes. Este hecho es debido a que el propio *Matlab©* incorpora librerías especificas destinadas a esta tarea, no obstante, es posible incluir bibliotecas externas como *OpenCV©* para trabajar en este campo.

Matlab ofrece un desarrollo continuo de sus aplicaciones con actualizaciones bianuales, al igual que una estandarización multiplataforma en cualquier sistema operativo.

Las tareas más usuales que puede desarrollar este sistema en el ámbito de la visión artificial son [36]:

- Detección, reconocimiento, identificación y seguimiento de objetos.
- Análisis de movimiento.
- Reconstrucción de escenas.
- Restauración de imágenes.

La mayor desventaja de esta herramienta es el alto coste de la licencia, que asciende a 800€ si es anual o a 2000€ eligiéndola de forma perpetua. Es por ello que se descarta Matlab como posible entorno de trabajo de cara al desarrollo.

3.2.4 OpenCV

Desarrollada por *Intel©*, *OpenCV©* es la librería de código abierto más usada a nivel global, pudiendo ser empleada para fines comerciales y de investigación, debido a su licencia BSD. Ha sido y es desarrollada en C y C++, con una programación optimizada aprovechando la capacidad de los procesadores multinúcleos [37].

OpenCV© se puede ejecutar en *Linux©*, *MacOS X©*, *Windows©* y *Android©*, dándole así un carácter multiplataforma, el cual favorece el cumplimiento de uno de los objetivos del proyecto. Actualmente cuenta con más de 2500 algoritmos y una comunidad de 47000 personas desarrollando módulos complementarios.

Entre los paquetes que incorpora esta librería destacan los siguientes [38]:

- Núcleo funcional.

  Contiene un modelo funcional el cual permite asignar imágenes a matrices, crear vectores multidimensionales como objetos y funciones básicas usadas por el resto de módulos.

- Procesamiento de imágenes.

  Los filtros anteriormente descritos están incluidos en este paquete, junto con funciones predefinidas dedicadas a la transformación geométrica, conversión de color, histogramas y su tratamiento…

- Análisis de video.

  En lo referente a imágenes en movimiento, se permite, extraer el fondo, emplear algoritmos de seguimiento ya incluidos o estimar la velocidad de un objeto etiquetado previamente.

- Calibración de cámara y Reconstrucción 3D.

  A través de este paquete, *OpenCV*© permite conectar varias cámaras simultáneamente y procesar sus imágenes obteniendo un escenario 3D, dotando de profundidad a los objetos presentes y permitiendo calcular distancias entre ellos.

- Reconocimiento de plantillas.

  Permite el uso de descriptores, detectores y plantillas, los cuales ha sido desarrollados por terceros. Entre ellos se encuentra el descriptor "HaarCascade" el cual será usado en el desarrollo del proyecto.

- Detección de objetos.

  Nativamente, se encuentran unas plantillas que permiten localizar personas, coches, caras u ojos, las cuales no son tan precisas como las desarrolladas por terceros, pero están mejor codificadas y son más ligeras.

- Conectividad hardware.

  Permite usar una gran cantidad de interfaces de captura de video externas a la aplicación, a través de una implementación rápida y sencilla, instalando ciertos codecs junto con la biblioteca *OpenCV*©.

Estas serán las librerías de visión artificial que se usarán de cara al desarrollo de la aplicación anteriormente descrita.

3.3 Base de datos FERET

En septiembre de 1993, el instituto nacional de estándares y tecnología (NIST) de los Estados Unidos, junto con el Departamento de Defensa, promovieron el desarrollo del programa “Facial Recognition Technology” (FERET). El objetivo de este proyecto era y sigue siendo, desarrollar nuevas técnicas, tecnología y algoritmos para automatizar el reconocimiento de los rostros humanos [39].

El proyecto fue dividido en tres fases. En la primera fase se marcó el objetivo de estudiar la viabilidad de los algoritmos de reconocimiento facial y averiguar una línea base para establecer su posible uso. Las fases 2 y 3, fueron marcadas por la mejora del desarrollo anteriormente efectuado. En 1998, como fin del programa FERET, la oficina de desarrollo de tecnología antidrogas del Departamento de Defensa, implementó está tecnología en sistemas de tiempo real, véase cámaras de videovigilancia [40].

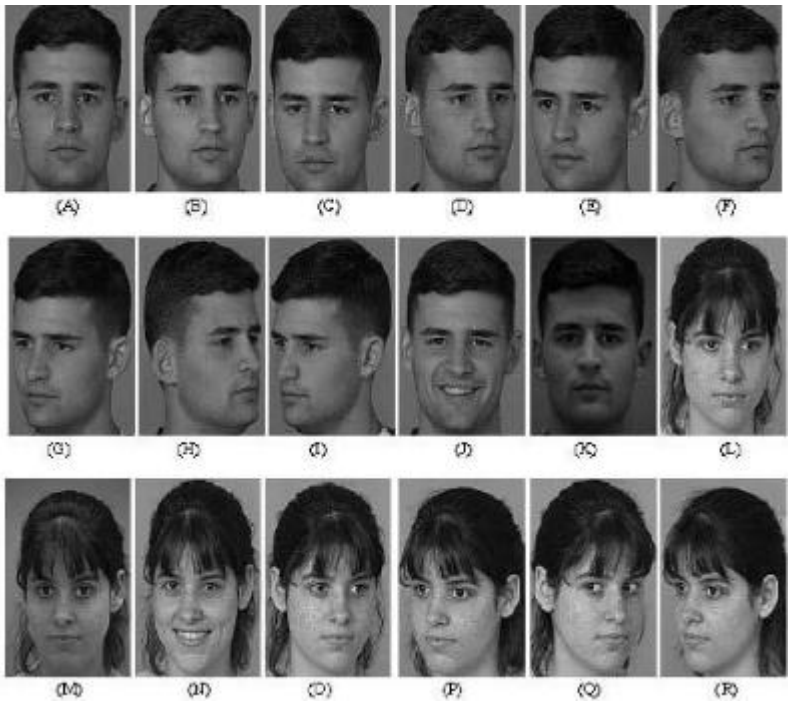


Ilustración 42 - Muestra de la base de datos FERET.

Entre diciembre de 1993 y agosto de 1996 se recolectó, en 15 sesiones, una base de datos compuesta por imágenes de 1199 individuos de diferentes edades y géneros, llegando a tener 14126 imágenes de estos individuos, en diferentes ángulos y con distintas expresiones faciales. Estas imágenes fueron tomadas en un entorno semi controlado, para mantener la coherencia a la hora de replicar todas las capturas de las imágenes.

Algunos objetivos planteados a posteriori son [41]:

- Búsqueda automática en bases de datos usando fotos tomadas por cámaras de vigilancia, bocetos de artistas o descripciones de testigos.
- Control de acceso a instalaciones o equipos restringidos.
- Proceso de acreditación de personal para verificar antecedentes.
- Monitorización de áreas susceptibles, como aeropuertos, cruces fronterizos, instalaciones oficiales…
- Encontrar y registrar múltiples apariciones de personas en un periodo de tiempo en videos de vigilancia, ya sea en tiempo real o grabados.

Ilustración 43 - Ejemplo de uso de la base de datos FERET.

Otros usos que se plantearon y no se han llegado a realizar:

- Verificación de identidad en cajeros automáticos.
- Verificación de identidad a la hora de expedir carnets y/o licencias.
- Búsqueda en los registros con la finalidad de evitar fraudes, como, conducción de usuario no siendo el titular del vehículo.

3.3.1 Tratamiento de la base de datos

La propia base de datos FERET, está dividida en 2 grandes grupos, siendo el primero perteneciente a las primeras capturas, con todas las imágenes en escala de grises y el segundo grupo contiene todas las imágenes en color, las cuales fueron procesamientos digitales respecto el primer grupo y nuevas capturas, para incrementar el número de individuos.

El set de imágenes de cada individuo varía, es decir, no todos tienen la misma cantidad de fotografías, ni las mismas posiciones. Es por ello que se ha seguido un proceso de catalogación exhaustivo.

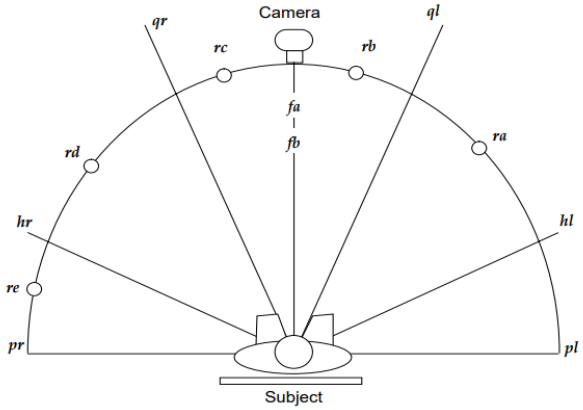


Ilustración 44 - Esquema sobre la toma de fotos, para la base de datos FERET.

La ilustración 44, representa un esquema de cómo se realizaron las capturas. Las fotografías frontales son nombradas como *"fa"* y *"fb"*, sugiriendo al sujeto que, para esta segunda, cambie la expresión facial, ya que es tomada segundos después. También se recogen imágenes de perfil derecha *"pr"* e izquierda *"pl"*, perfil medio *"hr y hl"* y cinco ubicaciones adicionales, espaciadas aleatoriamente entre las anteriormente descritas. A mayores, a ciertos sujetos se les pide usar gafas o complementos para realizar un segundo set sobre él mismo [41]. En la Ilustración 45, se observa un set como ejemplo.

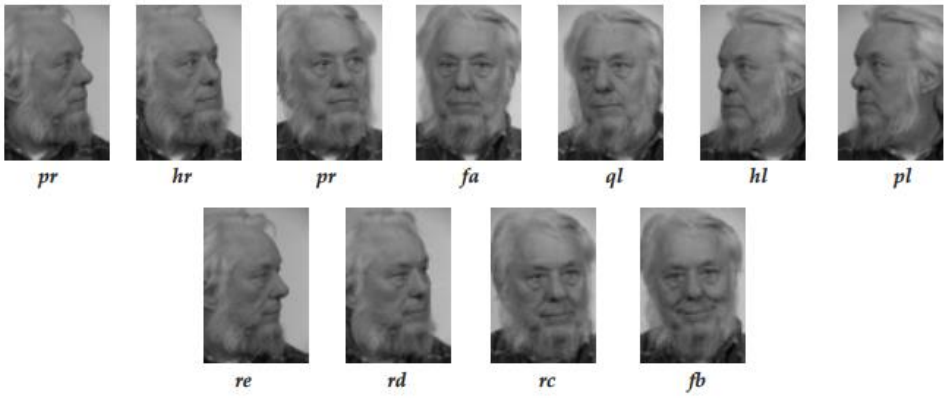


Ilustración 45 - Muestra de un individuo de la base de datos FERET.

Las imágenes pasan a ser nombradas en base al esquema presente en la ilustración 46.

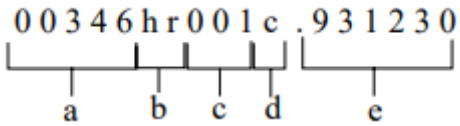


Ilustración 46 - Denominación de una imagen.

- A. ID único para cada fotografía.
- B. Pose, frontal, perfil, medio perfil…
- C. Flags. Estas cifras indican si la imagen tiene permiso para publicarse o no, cuál de las dos cámaras ha sido usada y si el histograma ha sido ecualizado de cara a mejorar la imagen. Se indican en el mismo orden mediante un 1 o 0.
- D. La letra indica ciertas características de la imagen, como si lleva gafas, si el brillo de la imagen o el contraste ha sido modificado, si el color ha sido modificado digitalmente….
- E. Fecha en la que se realizó la toma de esa imagen, aammdd.

En lo referente al desarrollo de este proyecto. De cara a simplificar la aplicación se plantea reconocer rostros frontales, por lo que para ello se debe hacer una selección de las imágenes que trae está base de datos. Ya que cada individuo presenta dos fotografías con estas características, se selecciona solo aquellas con código “fa” en su nombre.

La base de datos FERET, incorpora a mayores, un fichero TXT por cada imagen en el que se indican los siguientes valores.

id=cfrS00001
gender=Male
yob=1943
race=White

Con el año de nacimiento y la fecha de toma de la foto, la cual viene en el nombre de la imagen, se puede conocer la edad y el género de los individuos que se usarán para el desarrollo del proyecto.

3.4 Local Binary Patterns

En este apartado se introducirá el concepto *"Local Binary Pattern"* (LBP) y se profundizará más en el análisis anteriormente efectuado de los descriptores de textura.

Cabría preguntarse qué es LBP. Es una técnica descriptiva, que permite clasificar objetos en el ámbito que nos ocupa, la visión artificial. Se encarga de filtrar los píxeles vecinos respecto del analizado, obteniendo un valor representativo.

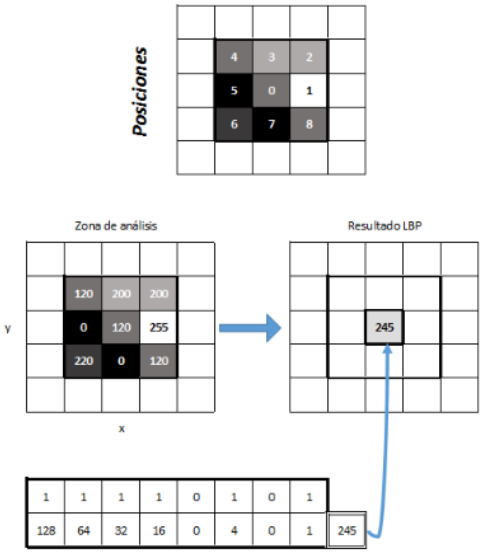


Ilustración 47 - Asignación del valor mediante LBP.

Para trabajar con este algoritmo es preferible usar imágenes en escala de grises, ya que contribuye a la robustez del algoritmo en cuanto a variaciones lumínicas, una de sus características más importantes. Cuando el píxel vecino sea mayor o igual al que se está analizando se asignará un 1 o un 0 en caso contrario. Este proceso se repite hasta acabar con los píxeles vecinos, obteniendo un array de 0 y 1 de longitud 8 por lo general.

Hay que tener especial cuidado en los bordes y esquinas de las imágenes, ya que la estrategia a emplear no es idéntica a la del interior. Por lo general se suelen ignorar estas zonas, debido a que la información obtenida suele conducir a errores en el resultado final.

Un ejemplo del array que se obtendría es el '11110101', el cual se convierte a código decimal '245', siendo éste el valor de intensidad de la imagen en ese píxel analizado.

En este punto, el algoritmo presenta dos variantes principales, la que se viene explicando y ULBP (Uniform Local Binary Patterns), siendo esta última la que se emplee durante el proyecto.

3.4.1 Uniform Local Binary Patterns

Esta variante es posiblemente la más usada. Esto se debe a que restringe el uso de cualquier array obtenido a solo aquellos que sean uniformes, mejorando con ello los clasificadores. A mayores, este algoritmo permite ser usado para obtener la invarianza ante rotaciones en la matriz de intensidad [42].

El concepto uniforme se define a posteriori de obtener el array, ya que se cuentan los saltos de valor en el propio vector, permitiendo como máximo realizar dos transiciones.

En el ejemplo '00111000' se aprecian solo dos saltos, el primero entre la segunda y la tercera posición. El otro salto está entre la quinta y sexta. Por lo que se puede afirmar que este array es uniforme. Si se trabaja con '01010101' se pueden aprecian siete transiciones, despreciando este vector como uniforme.

Ilustración 48 - Comparación entre imagen LBP y ULBP.

Posteriormente se obtienen los valores decimales de los vectores uniformes. A los patrones no uniformes se les asigna un mismo valor. Todo ello para obtener el histograma de la imagen en el que el eje de abscisas tenga solo 59 espacios en vez de los 256 habituales.

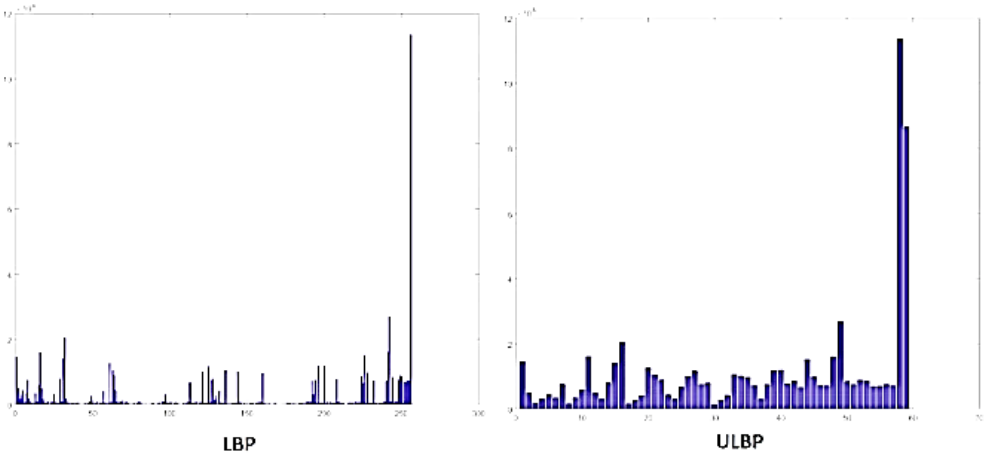


Ilustración 49 - Comparación histogramas LBP y ULBP.

En el caso de esta aplicación, una vez se han obtenido los valores decimales de los arráis, se introducirá dicho valor en una Look Up Table (LUT), de tal forma que será esta matriz la que dará el valor final al píxel.

La razón por la que se obviarán los patrones no uniformes a la hora de crear el algoritmo, es debido a que, en imágenes naturales, es decir, aquellas que no han sido creadas y/o modificadas por ordenador, la presencia de patrones uniformes es altamente significativa, llegando a estar presentes en más del 90% de la imagen.

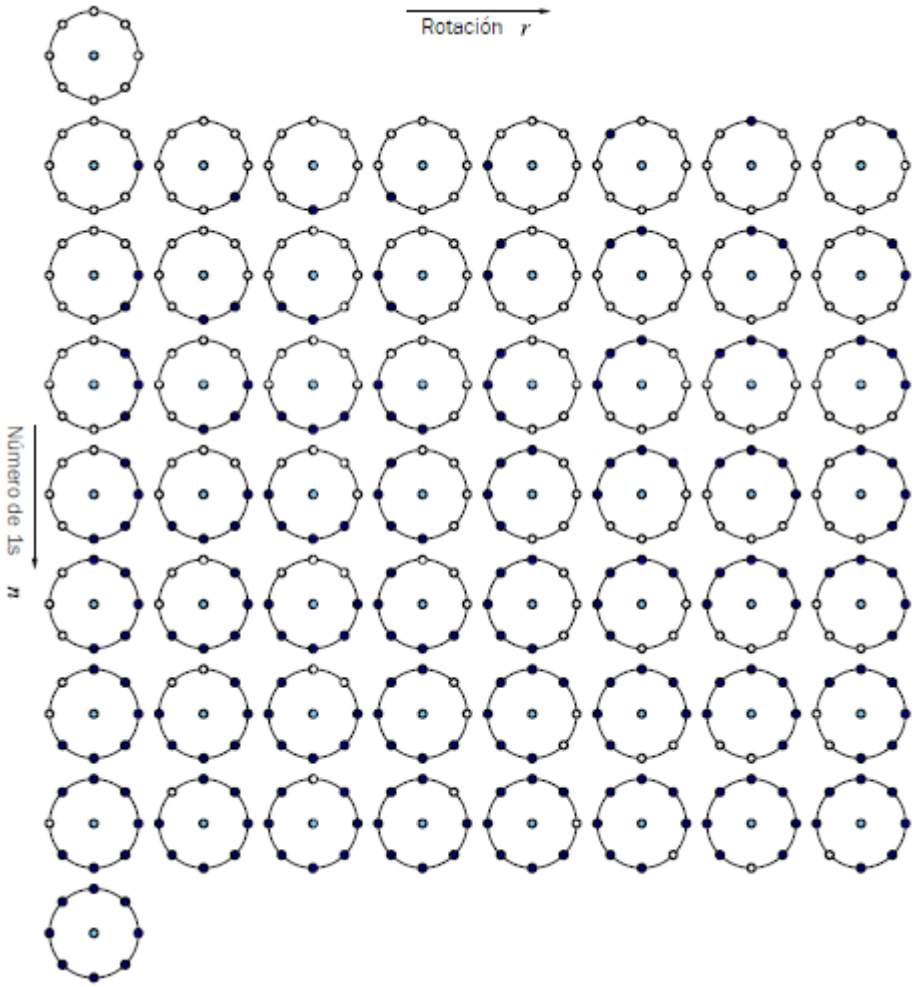


Ilustración 50 - Muestra de los patrones uniformes con R=1 y P=8.

CAPITULO IV

4 Desarrollo

El algoritmo que se procede a describir ha sido desarrollado en lenguaje C++, valiéndose de las librerías *OpenCV©*. Para facilitar la elaboración del código de la aplicación se ha empleado el IDE (*"Integrated Development Environment") Code::Blocks©*, el cual está especializado en los lenguajes C y C++. A mayores como se ha mencionado anteriormente se ha empleado la distribución Ubuntu, perteneciente a la familia *Linux©*.

Los capítulos previos han tenido como finalidad elaborar un contexto en el que acotar la aplicación. Es por ello que en el primer capítulo se fijaron los objetivos que debía cumplir. Posteriormente se ha explicado la visión artificial y las herramientas que ofrece, para finalmente exponer la base de datos que se empleará junto con el operador ULBP.

El software desarrollado se divide en tres procesos cuyas funciones están diferenciadas y secuenciadas, ya que todo ello ha de funcionar en tiempo real. Primeramente, se dará una visión general de la aplicación y de cómo interactúan los procesos entre sí. Posteriormente se analizará cada proceso de forma independiente, evaluando únicamente las tareas en el realizadas. Para finalizar en el capítulo V se expondrán los resultados obtenidos.

4.1 Descripción Global

Atendiendo a la búsqueda de una visión general de la aplicación, se procede a representar mediante un diagrama los grandes bloques que compondrán el desarrollo.

En la ilustración 51 se pueden ver los procesos principales.

- Proceso 1: Proc_base_datos

  Se procesarán las imágenes de la base de datos *FERET©* junto con los archivos TXT que vienen adjuntos a cada individuo. El resultado será la obtención de un recorte de la zona facial y su posterior clasificación a partir de la edad y el género.

- Proceso 2: Hist_base_datos

  Partiendo de los recortes obtenidos y de su clasificación, se prosigue con la obtención del histograma. Por ello se convierten las imágenes a ULBP y se dividen en teselas para obtener el histograma ULBP de cada una de ellas. Todo ello da lugar a una normalización de los histogramas pertenecientes a cada rango de edad y género almacenados en formato xml y png.

- Proceso 3: Proc_Principal

Establece la conexión con la cámara obteniendo video en tiempo real. Analiza cada frame realizando recortes de los rostros presentes y analizando estos con los datos anteriormente obtenidos. La información resultante es mostrada junto con el video de la cámara en tiempo real.

- Proceso 4: Proc_Principal

Para realizar un feedback y mejorar la base de datos y con ella la fiabilidad de los resultados, se implementa un guardado de cada rostro analizado en la BBDD. Se almacenará tanto el recorte como un TXT con la misma configuración que los otorgados por la base de datos *FERET©*, generando así una estandarización del proceso.

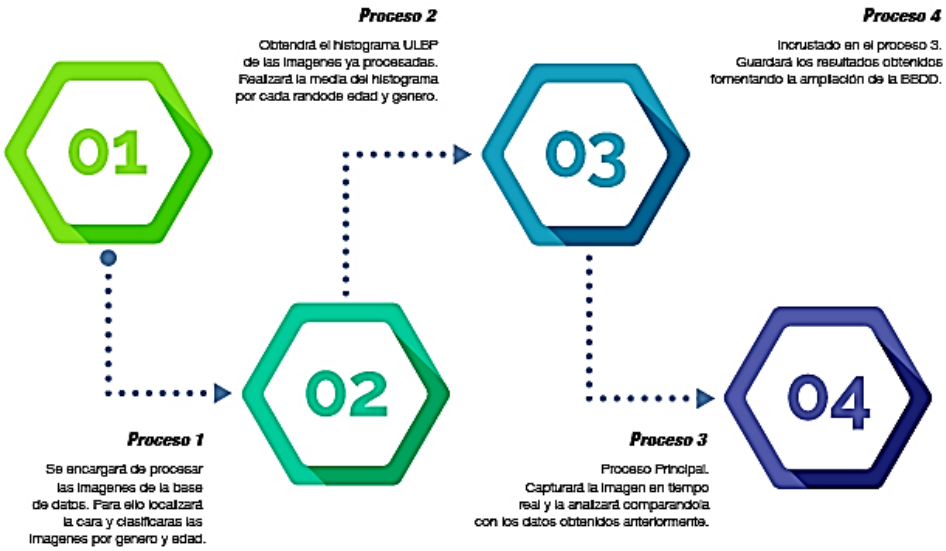


Ilustración 51 - Diagrama de flujo general.

4.1.1 Estructura de carpetas

Para la realización de la aplicación se observa la necesidad de trabajar con el almacenamiento de datos, para que estos puedan ser usados posteriormente en el resto de módulos o en posibles futuras mejoras.

Asimismo, se piensa en una posible inserción del software en diferentes dispositivos. Para facilitar esto se estructura la aplicación como un bloque el cual solo pueda ser transferido con facilidad.

Con esto se busca que la aplicación exclusivamente necesite la instalación de las librerías *OpenCV©* para poder ser ejecutada en cualquier terminal que contenga un compilador C++.

La estructura de carpetas propuesta se muestra en la ilustración 52. Se puede apreciar que es un circuito cerrado de datos, lo que facilita lo anteriormente expuesto, la portabilidad de la aplicación a diferentes dispositivos, con diferentes sistemas operativos.

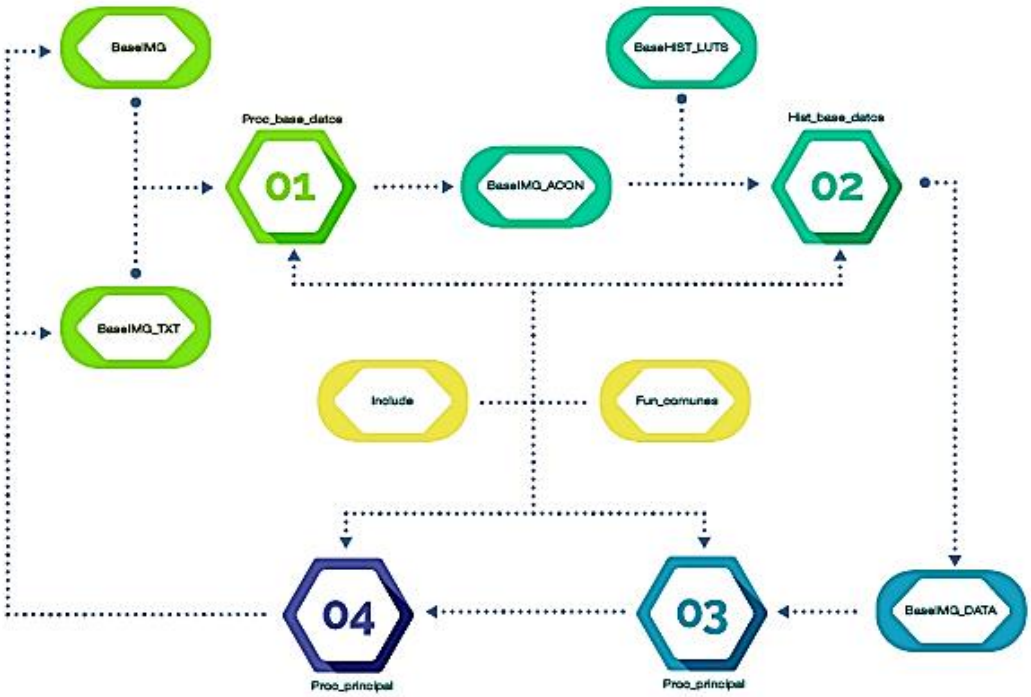


Ilustración 52 - Estructura de carpetas

- BaseIMG

Esta carpeta contiene todas las imágenes que incorpora la base de datos *FERET©*. Todas ellas han sido seleccionadas para que el rostro este de frente, descartando con ello el resto. De este proceso se hablará en el apartado 4.2.

Un ejemplo de las imágenes que se encuentran es el ilustrado en la siguiente imagen.

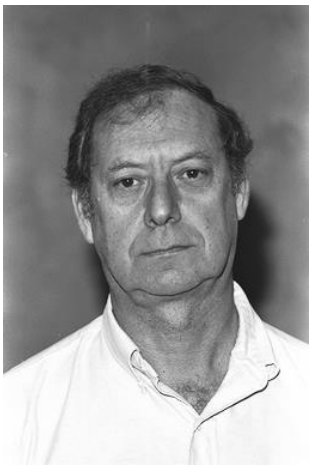


Ilustración 53 - Imagen 00005_930831. FERET.

- BaseIMG_TXT

Se han almacenado los TXT pertenecientes a la base de datos *FERET©* en esta carpeta. Los cuales como se ha visto anteriormente contienen los datos del género y el año de nacimiento de la persona fotografiada.

Remarcar que no todas las imágenes procedentes de la *FERET©* poseen TXT, es por ello que en el primer proceso se evaluará esta casuística.

- BaseIMG_Acon

Como se observa en la ilustración 52, esta carpeta contendrá todas las fotografías acondicionadas que han sido resultado del primer proceso "Proc_base_datos". La organización interna es realizada primeramente por rangos de edad, para posteriormente en cada subcarpeta dividir las imágenes por género. Todo ello es realizado por el primer proceso de forma autónoma.

Si se tienen en cuenta las imágenes de la base de datos *FERET©* exclusivamente, se obtiene lo siguiente.

- 10-19
  - Hombres: 11
  - Mujeres: 7
- 20-29
  - Hombres: 210
  - Mujeres: 234
- 30-39
  - Hombres: 173
  - Mujeres: 85
- 40-49
  - Hombres: 116
  - Mujeres: 47
- 50-59
  - Hombres: 55
  - Mujeres: 25
- 60-69
  - Hombres: 22
  - Mujeres: 2
- 70-79
  - Hombres: 1
  - Mujeres: 1

El número indicado anteriormente indica los recortes que son válidos y que serán usados en el siguiente proceso. En total en la puesta en marcha de la aplicación se procesan correctamente 989 retratos de los 1199 totales que posee la base de datos.

- BaseHIST_Data

Esta carpeta mantiene una estructura idéntica a la estipulada en el anterior apartado. En ella se almacenarán los histogramas que se obtienen al ejecutar el proceso "Hist_base_datos". Los formatos en los que se almacenarán son png, para que una visualización inmediata; y en xml, que será el formato que empleará el siguiente proceso para leer dichos histogramas.

En el apartado 4.3 se verá más a fondo la técnica empleada para almacenarlo. Como ejemplo en la ilustración 54 se puede ver un histograma png perteneciente a una tesela. El eje de ordenadas de estos, muestra los 59 posibles niveles de intensidad.

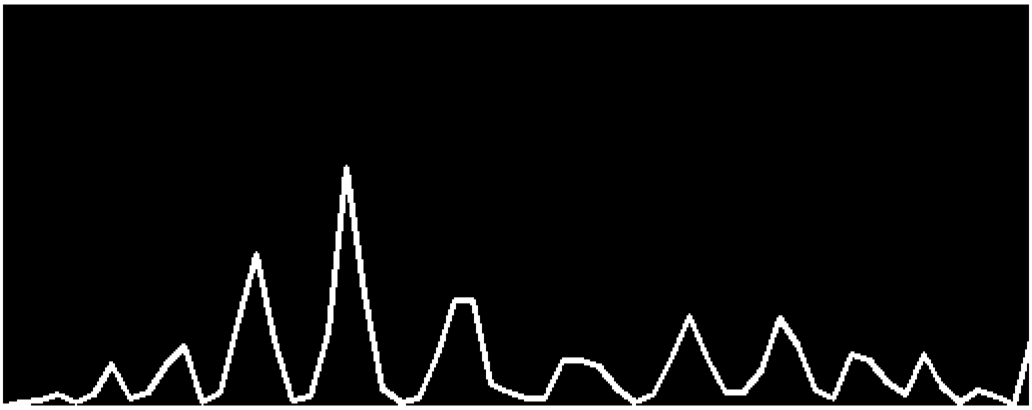


Ilustración 54 - Histograma de una tesela.

- BaseHIST_Luts

Solo contiene un archivo llamado “ULBP_LUT_59.dat”. Este archivo como indica su nombre es usado por una LUT (“Look Up Table”) para transformar el histograma de 256 niveles a solo 59 y así poder utilizar el algoritmo ULBP anteriormente descrito.

La LUT contiene la siguiente secuencia numérica:

0 1 2 3 4 58 5 6 7 58 58 58 8 58 9 10 11 58 58 58 58 58 58 58 12 58 58 58 13 58 14 15 16 58 58 58 58 58 58 58 58 58 58 58 58 58 58 58 17 58 58 58 58 58 58 58 18 58 58 58 19 58 20 21 22 58 58 58 58 58 58 58 58 58 58 58 58 58 58 58 58 58 58 58 58 58 58 58 58 58 58 58 58 58 58 58 23 58 58 58 58 58 58 58 58 58 58 58 58 58 58 58 24 58 58 58 58 58 58 58 25 58 58 58 26 58 27 28 29 30 58 31 58 58 58 32 58 58 58 58 58 58 58 33 58 58 58 58 58 58 58 58 58 58 58 58 58 58 58 34 58 58 58 58 58 58 58 58 58 58 58 58 58 58 58 58 58 58 58 58 58 58 58 58 58 58 58 58 58 58 58 58 35 36 37 58 38 58 58 58 39 58 58 58 58 58 58 58 40 58 58 58 58 58 58 58 58 58 58 58 58 58 58 58 41 42 43 58 44 58 58 58 45 58 58 58 58 58 58 58 46 47 48 58 49 58 58 58 50 51 52 58 53 54 55 56 57

- Include y Fun_comunes

Las dos carpetas restantes, tal y como se indica en la ilustración 52, son usadas por los procesos ya que en ellas se incluyen las librerías y funciones que son comunes a todos ellos y que han sido desarrolladas para esta aplicación. Esto facilita el diseño modular de la aplicación pudiendo realizar nuevas funciones sin tener la necesidad de modificar el código de los procesos.

Las librerías y funciones son las anexadas en la siguiente tabla:

Librerías h	Archivo C++
compara.h	compara.cpp
estadist.h	estadist.cpp
fichero.h	fichero.cpp
Fun_Histo.h	Fun_Histo.cpp
histo.h	histo.cpp
imBBDD.h	imBBDD.cpp
lbp.h	lbp.cpp
visualiza.h	visualiza.cpp
ulbp.hpp	

A lo largo de este capítulo se entrará en el detalle de cada archivo en el que se verá que función realiza.

4.1.2 Comunicaciones entre procesos y señales

Hasta el presente se ha explicado de forma general las funciones de los procesos involucrados en la aplicación y la estructura de carpetas en la que se van a depositar los datos resultantes de estos.

Es debido a la naturaleza secuencial de los procesos que se necesita indicar el inicio de estos mismos, es decir, el segundo proceso “Hist_base_datos” no puede comenzar su ejecución hasta que se haya finalizado el tratamiento de todas las imágenes de la base de datos a través del primer proceso.

Asimismo, se intenta proteger la aplicación ante cierres ocasionados por agentes externos. Para ello se propone reiniciar la ejecución de cada proceso cada vez que se cierre por circunstancias ajenas a la propia ejecución, véase un cierre inesperado.

Por lo anteriormente descrito se propone implementar una comunicación entre los procesos y un control de señales para optimizar la ejecución del software desarrollado.

En la ilustración 55 se muestra un esquema del planteamiento de señales y de comunicación que se plantea emplear. Solo se incorporan las carpetas limitantes, es decir, aquellas en las que la salida de un proceso es usada por el siguiente.

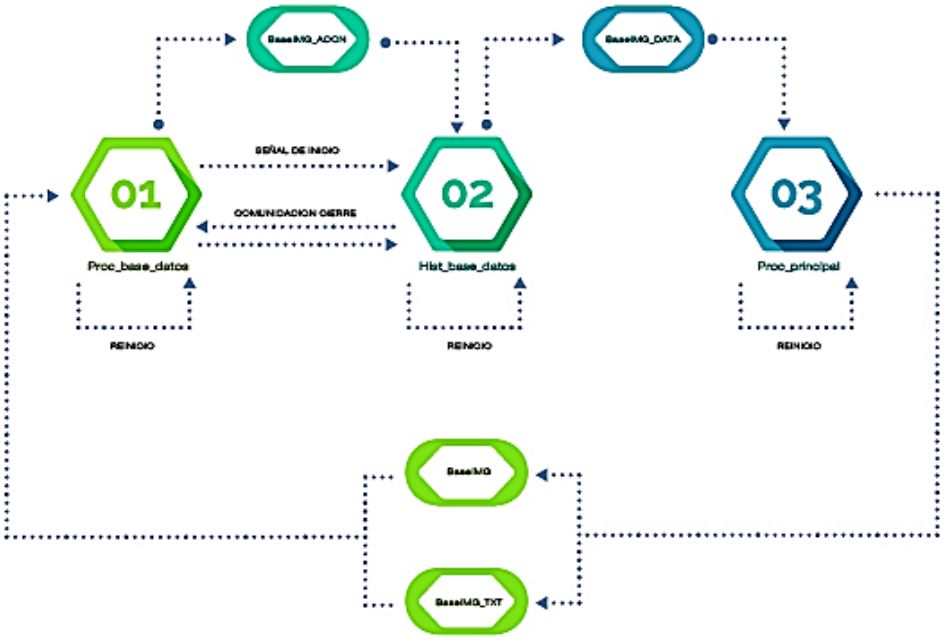


Ilustración 55 - Diagrama comunicación y señales.

4.1.2.1 Comunicación

Atendiendo las consideraciones anteriores, se procede a exponer la comunicación realizada entre los procesos “Proc_base_datos” e “Hist_base_datos”.

De acuerdo al uso de un sistema operativo basado en *Linux*© se usarán los mecanismos IPC (InterProcess Comunication) que ofrece el propio Ubuntu. Dentro de los mecanismos IPC entre los que se encuentran colas de mensajes, semáforos y memorias compartidas, se opta por implementar estas últimas, debido a las múltiples opciones que ofrecen, en especial orientadas a futuras mejoras.

Una memoria compartida es una zona común de memoria entre dos o más procesos, la cual permite compartir datos en ese segmento de memoria.

El primer proceso, “Proc_base_datos” será el encargado de crear las dos memorias compartidas reflejadas en la ilustración 55.

La creación de la memoria pasa por obtener una clave para que solo los procesos que se desea puedan acceder a dicho segmento de memoria. Posteriormente la creación se implementa a través de una llamada al sistema “shmget()”.

Como se ha mencionado, se crearán 2 memorias compartidas entre ambos procesos.

- Memoria compartida de inicio

  Esta primera está orientada en una sola dirección, es decir, el flujo de datos se realiza del primer proceso al segundo exclusivamente, siendo el primero un emisor y el segundo el lector de la información.

  La información que se comparte entre ambos será booleana indicando “true” o “false”.

  Está memoria enviará por defecto un “false”, el cual indica que el segundo proceso debe estar en espera hasta su activación. Se enviará “true” cuando el primer proceso haya finalizado el tratamiento de las imágenes de la base de datos, dando con ello comienzo a la obtención de los histogramas ULBP del segundo proceso.

  Una vez se inicie el nuevo proceso, se efectuará de nuevo un cambio en la variable recibida a “false”, para evitar ejecuciones simultaneas entre ambos procesos. Por lo que siempre habrá un único proceso activo, estando el complementario a la espera del “true”.

- Memoria compartida de cierre

  En este caso, la memoria compartida está asociada a señales de cierre. Si alguno de los dos procesos recibe una señal de cierre este comparte dicha información con el otro. Eso es debido a la naturaleza de estos dos primeros procesos, los cuales como ya se ha explicado funcionan de forma secuencial y en caso de cierre de uno de ellos se debe reiniciar ambos en pro de un correcto funcionamiento.

  Al contrario que en el caso anterior, se tiene acceso de lectura y escritura por parte de ambos procesos. El dato que se comunica al segmento de memoria es igualmente un booleano, el cual permanece a “false” por defecto y solo cambia a “true” cuando a alguno de los procesos se le informa con realizar un cierre.

  Cada vez que se ejecute el reinicio de estos procesos, se cerrarán ambas memorias compartidas para no causar posibles violaciones de segmento.

Al crearse ambas memorias por mecanismos IPC, se deben cerrar al acabar su función, ya que en caso contrario el sistema operativo las mantendrá activas hasta el cierre o reinicio del mismo. Cada vez que se ejecute el reinicio de estos procesos, se cerrarán ambas memorias compartidas para no causar posibles violaciones de segmento.

Pero en caso de que la aplicación sufra un cierre no controlado, ambos segmentos de memoria compartida permanecerán activos. Para ello se empleará el comando “ipcrm -a”, el cual cierra las memorias presentes en el sistema y que no sean inherentes a él.

Este comando Unix, se ejecutará previamente a la ejecución de la aplicación, realizando con ello una limpieza de las posibles memorias que estuviesen abiertas. En el apartado 4.5 se verá el momento de su ejecución y su implementación mediante un “makefile”.

4.1.2.2 Señales

Una señal es un evento que informa a un proceso que un evento ha ocurrido. Estas son generalmente asíncronas con lo cual no se conoce el momento en el que van a ser notificadas, por lo que se recomienda realizar un tratamiento de estas para controlar los eventos puedan ser informados. Un ejemplo claro es “Ctrl+C”, el cual envía una señal de interrupción al proceso realizando un cierre abrupto en este.

Las señales más comunes son las siguientes:

SIGHUP 1 /* hangup */
SIGINT 2 /* interrupt */
SIGQUIT 3 /* quit */
SIGILL 4 /* illegal instruction */
SIGABRT 6 /* used by abort */
SIGKILL 9 /* hard kill */
SIGALRM 14 /* alarm clock */
SIGCONT 19 /* continue a stopped process */
SIGCHLD 20 /* to parent on child stop or exit */

En la aplicación se ha decidido tratar la señal número 2 “SIGINT”. Esta señal es recibida por un proceso cuando se presiona la combinación “Ctrl+C”, generando el cierre del proceso, como se ha comentado.

Para atender su llamada se emplea el manejador “signal()”, el cual al recibir la señal cambia la acción por defecto por una llamada a una función, en la cual el usuario puede codificar las acciones que él requiera.

En el caso de esta aplicación al presionar “Ctrl+C” desde alguno de los dos primeros procesos la ejecución del proceso se verá interrumpida. Posteriormente se enviará al segmento de memoria compartida de cierre un “true” para que ambos procesos finalicen. Por último, se cerrarán las memorias compartidas y se reiniciará el proceso mediante la consola llamando de nuevo a ambos procesos.

En lo referente al “Proc_Principal”, el funcionamiento es similar. Cuando se presione la combinación “Ctrl+C” el programa se reiniciará, pero en este caso, el proceso carece de memorias compartidas por lo que se obvian esos pasos en relación a los procesos previos.

4.2 Descripción Individual de los procesos

Anteriormente se ha descrito una visión global de la aplicación y de los procesos que se ejecutan en paralelo para un correcto funcionamiento, al igual que las conexiones y directorios que se emplean.

En este apartado se detallará las tareas que los procesos, anteriormente mencionados, realizan. Para ello se seguirá el orden de ejecución de la aplicación, es decir, comenzando por el proceso "Proc_base_datos" y finalizando con el "Proc_Principal".

4.2.1 Proc_base_datos

Con el objeto de estudiar este proceso, se realiza una división en 3 campos, los cuales son el tratamiento previo que se realizará a la base de datos FERET, el propio proceso y las salidas que serán obtenidas una vez se finaliza la ejecución.

4.2.1.1 Tratamiento previo base de datos FERET

Previamente, en el capítulo III apartado 3.4, se ha expuesto la base de datos *FERET©* de la cual se vale esta aplicación. Como se ha mentado, cada individuo fotografiado posee como mínimo 8 imágenes asociadas a diferentes posiciones.

A mayores, el nombre con el que aparecen las fotos es poco descriptivo, aunque posee información que es necesaria para conocer el ángulo de captura y el día en el que fue tomado.

Por todo lo anterior, se procede a realizar un tratamiento previo a la base de datos en el que se eliminen todas aquellas fotos en las que el rostro no esté situado de frente, es decir, las que en su nomenclatura no aparezca "fa" o "fb". Para ello nos valdremos de un "Shell-script" en la consola de *Linux©*.

Una vez realizado esto, se obtienen 2418 fotografías, número que se reduce a la mitad, ya que después de analizar las imágenes, se toma la decisión de eliminar las "fb". De esta forma se obtienen 1209 fotos de frente del mismo número de individuos.

Respecto a la nomenclatura empleada por *FERET©*, se decide realizar un cambio para simplificar la adición de nuevas fotografías. La nomenclatura pasará a contener exclusivamente un número identificativo y la fecha de toma de la fotografía. Este cambio se realiza mediante otro "Shell-script" debido al gran número de imágenes.

00118hr001c.931230.tif ⟶ 00118_931230.tif

De esta forma el código a la hora de cargar las fotografías de la base de datos, solamente tendrá que almacenar dos datos, el número de fotografía y la fecha, siendo esta necesaria para obtener la edad de la persona fotografiada.

4.2.1.2 Procesamiento

A lo largo de este apartado se explicará el flujo que realizarán las imágenes almacenadas en la base de datos a través de este primer proceso. Anteriormente se ha mencionado que "Proc_base_datos" es el encargado de crear las memorias compartidas entre este y el siguiente proceso, siendo esto descrito en el apartado 4.1.2.

Primeramente, el proceso analiza el número de archivos que existen en la carpeta BaseIMG. En el caso en el que la cantidad de imágenes en la base de datos no se corresponda con el número de imágenes tratadas en la carpeta BaseIMG_ACON, comenzará la ejecución. En caso contrario el proceso principal estará a la espera de una modificación en la base de datos. Esto se ha codificado para favorecer el ahorro de recursos consumidos por el proceso, de esta forma no estará en ejecución de forma continua, solo cuando se actualice dicho directorio.

Para realizar la lectura del tamaño de ambas carpetas se emplean la librería "dirent.h" la cual permite conocer el número de archivos en un directorio o incluso cargar en un vector los nombres de los archivos, como es el caso de los que están contenidos en la carpeta BaseIMG. La carga se realiza sobre un vector string, el cual será ordenado posteriormente obteniendo un listado de todas las imágenes presentes en la base de datos.

Como se ha mencionado la ejecución se realiza al cumplir la casuística anteriormente indicada.

El proceso entra en un bucle que recorre el vector que contiene los nombres de todas las imágenes presentes en la base de datos. Con la cadena de caracteres que define la imagen se va realizando la carga a través de la función "imread()", siendo esta la función más básica de las librerías OpenCV. Al ser cargada la imagen se realiza un filtro paso bajo para eliminar el posible ruido que haya en la imagen. Estos filtros han sido explicados anteriormente en el apartado 2.1.3.2.4.

Junto con la carga de las imágenes se realiza la carga del clasificador "HaarCascade". Como se ha explicado en el apartado 2.1.3.4.1, existe una gran

cantidad de estos clasificadores, por lo que se realiza una batería de pruebas con varios descriptores dedicados a la búsqueda de rostros humanos. A continuación, se puede apreciar las diferencias que existen entre unos y otros a la hora de localizar rostros.

Ilustración 56 - Muestra descriptores HaarCascade.

Como se puede ver en la ilustración 56, existen múltiples descriptores. Los que se muestran en la ilustración son los más recomendados y usados para localizar rostros en imágenes. Se aprecia que los que más ajustan la cara eliminando el fondo son “haarcascade_frontalface_alt” y “haarcascade_frontalface_default”.

Por los resultados obtenidos, al hacer una batería de pruebas, el descriptor que más encaja con lo que se busca para la aplicación es “haarcascade_frontalface_alt.xml”, ya que este limita la región facial eliminando gran parte del fondo, centrándose únicamente en la imagen.

Una vez se tiene la imagen y el clasificador cargados, se procede con un tratamiento para esta primera, el cual facilita la el proceso del "HaarCascade".

- Reescalado

Tras la realización de múltiples pruebas, se observa que el clasificador funciona mejor en imágenes de alta calidad y gran tamaño. Las imágenes de la base de datos poseen diferentes tamaños, aunque en un gran número de casos se repite la dimensión 256 x 384. Por ello se estandariza la entrada al doble del tamaño.

Para ello se emplea una operación entre píeles explicada en el apartado 2.1.3.2.1. La aplicación de esta operación solicita un método de interpolación, siendo los siguientes los ofrecidos por las librerías [43].

- INTER_NEAREST - Una interpolación del vecino más cercano.
- INTER_LINEAR - Una interpolación bilineal.
- INTER_AREA - Remuestreo utilizando la relación del área de píxeles.
- INTER_CUBIC - Una interpolación bicúbica sobre los vecinos de 4x4 píxeles.
- INTER_LANCZOS4 - Una interpolación de Lanczos en los vecinos de 8x8 píxeles.

En este y en los sucesivos casos, la aplicación empleará la interpolación bilineal "INTER_LINEAR".

- Cambio de color

La base de datos *FERET*© adjunta todas las imágenes en blanco y negro o en color. Pero en un futuro las imágenes que se adjunten no tienen por qué ser en blanco y negro. Es por ello que se realiza un cambio a escala de grises.

- Ecualización del histograma

Como se ha expuesto en el apartado 2.1.3.2.2, las imágenes saturadas pueden ser solventadas mediante una ecualización del histograma. Estas imágenes como se ha mencionado anteriormente se pueden dar bajo diferentes circunstancias lumínicas, es por ello, que al desconocer la zona de aplicación se realiza esta operación.

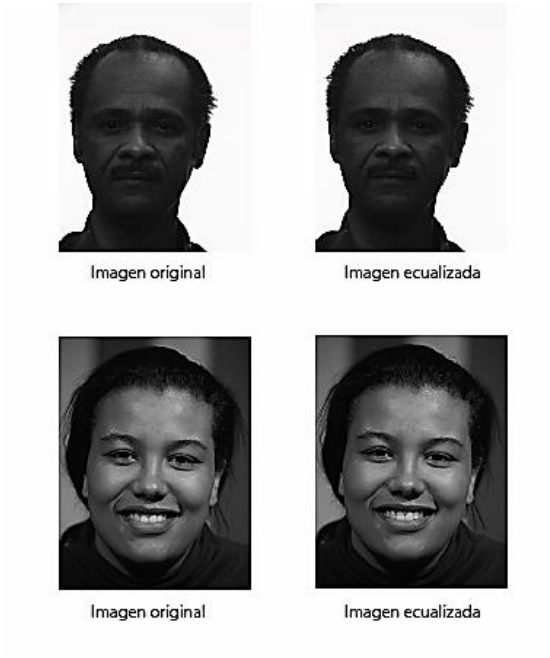


Ilustración 57 - Ejemplos de rostros ecualizados por su histograma.

Posteriormente a la acomodación de la imagen al clasificador, se procede a aplicar este. La salida que se obtendrá es un vector de puntos, los cuales marcan las zonas en las que se encuentra el objeto buscado y como entradas se tendrá la imagen. A mayores se pueden configurar cuatro parámetros, los cuales son [44]:

- Factor de escala

  Se explicó anteriormente que la aplicación de estos clasificadores se realiza de forma iterada, por lo que este parámetro ajusta la escala a la que se reduce la imagen con cada iteración. El valor por defecto es 1.1, el cual da un resultado aceptable, pero no lo sufrientemente aproximado, por lo que se usará un factor de 1.01.

  Al reducir este factor, la obtención de los rostros es más lenta, pero con un alto índice de aciertos.

- Vecinos

  Al aplicar los filtros "Haar" descritos anteriormente, se solicita conocer los píxeles que se van a comparar de la imagen integral. El valor de este parámetro establecido por defecto es 3x3. En caso de ser modificado se deben elegir parámetros impares 5x5, 7x7…

  Al realizar pruebas para ajustarlo, se observa que el valor por defecto se ajusta a lo que se busca obtener, ya que ampliar este parámetro ralentiza la ejecución.

- Flags

Se emplea este parámetro para favorecer la búsqueda del elemento deseado. Para lograr esta optimización se pueden emplear las siguientes opciones [45].

  - Defecto

    Esta opción aplica el clasificador sin tener en cuenta ninguna consideración limitante.

  - DO_CANNY_PRUNING

    Esta función emplea el detector de bordes Canny para obviar las zonas de la imagen que contengan pocos o demasiados bordes, lo que acelera la búsqueda al tener que analizar menos regiones de la imagen.

  - SCALE_IMAGE

    Al aplicar esta configuración se reduce la escala de la imagen en lugar de ampliar las coordenadas.

  - FIND_BIGGEST_OBJECT

    Esta configuración busca el objeto más grande presente en la imagen, el cual será la salida del clasificador. En caso de no ser encontrado devolverá un 0, indicando que la imagen no posee objetos principales.

  - DO_ROUGH_SEARCH

    Este filtro se aplica únicamente junto con FIND_BIGGEST_OBJECT y con un valor de vecinos superior a 0. En este caso el clasificador dejará de buscar candidatos de menor tamaño al encontrar ya un objeto presente, realizando una búsqueda de orden superior. La ventaja de este flag es la velocidad que adquiere el proceso, limitando con ello la precisión de búsqueda.

Debido a que las imágenes que se van a procesar son de menos de medio cuerpo, el objeto más grande presente en estas será el rostro del individuo. Por lo anteriormente descrito, se emplea FIND_BIGGEST_OBJECT como flag del clasificador.

- Tamaño mínimo

  El último parámetro que se ajustará es el tamaño mínimo en píxeles que podrá tener el rostro en la imagen, despreciando todos aquellos que su dimensión sea menor. Para la aplicación se observa que el tamaño de las imágenes de la base de datos oscila en torno a 256x384, con lo que se sitúa este parámetro en 100x100.

Como se ha mencionado se obtiene un vector de tipo “Rect” el cual contiene las coordenadas X Y del punto superior izquierda y la altura y ancho del rectángulo que contiene el rostro, delimitando con ello la zona que se está buscando.

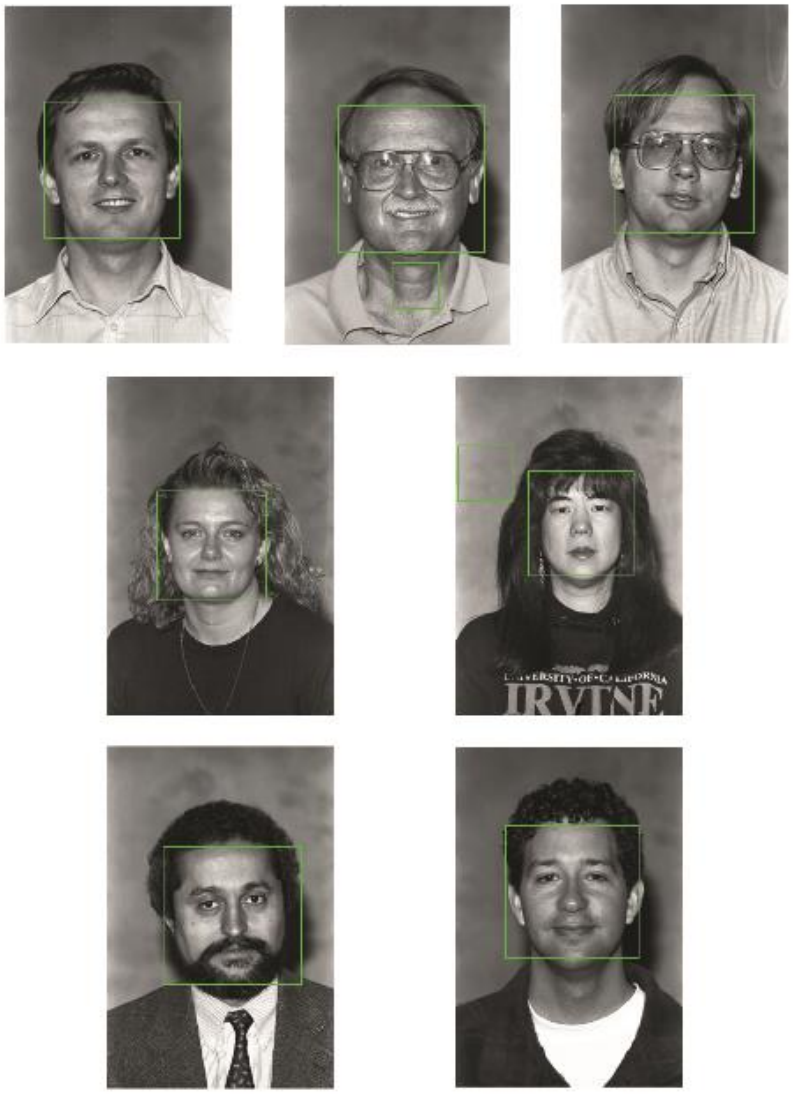


Ilustración 58 - Ejemplos aplicación HaarCascade.

En la ilustración 58 se aprecian diferentes ejemplos sobre los que se ha aplicado el clasificador. En las imágenes número 2 y 5 se observa que el clasificador ha encontrado más de un posible rostro, pero al ser seleccionado como parámetro FIND_BIGGEST_OBJECT, las coordenadas finales que se tendrán serán las del objeto más grande, en el caso de esta aplicación es el rostro.

Con estas coordenadas se puede situar el rostro en la imagen original y realizar un recorte para trabajar con la zona facial exclusivamente. Este recorte es reescalado posteriormente a una dimensión de 200x300, para estandarizar todas las imágenes procesadas, las cuales serán almacenadas en la carpeta BaseIMG_ACON.

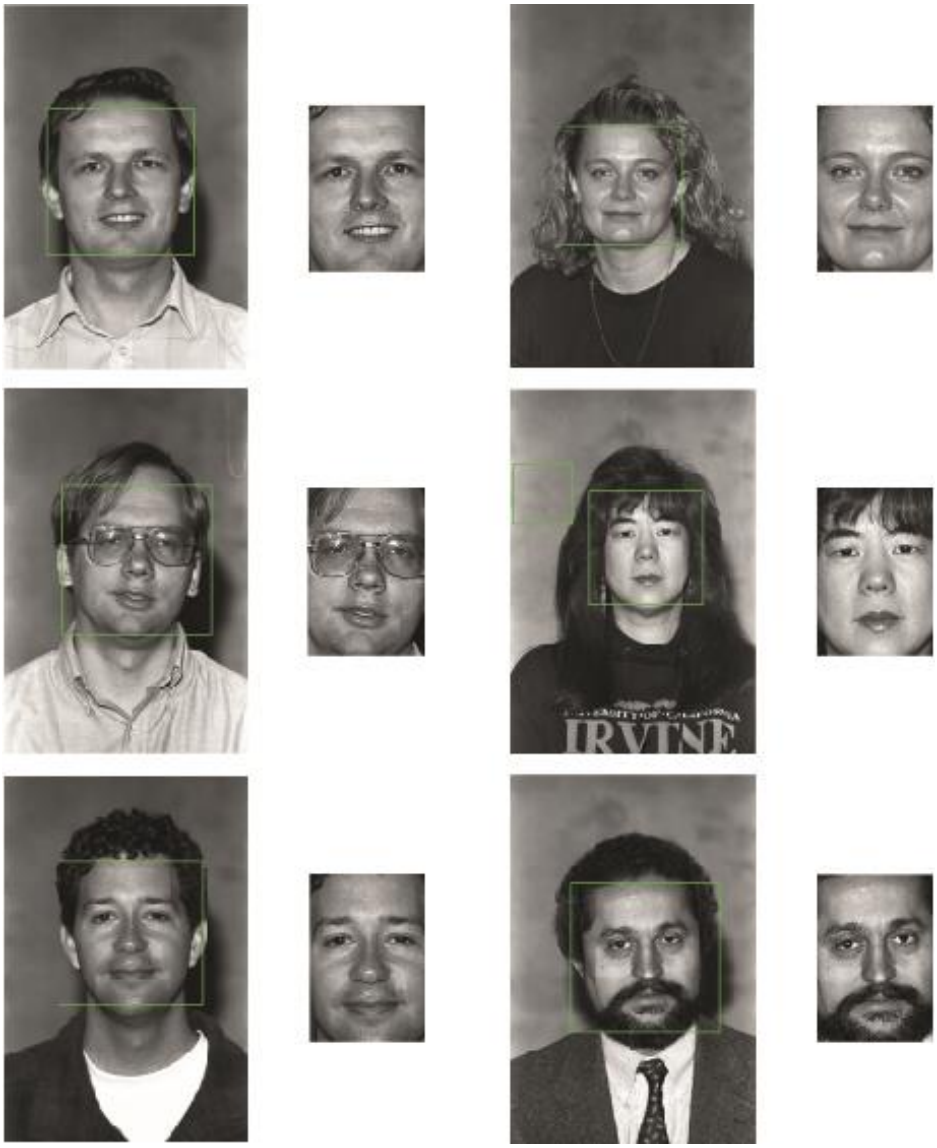


Ilustración 59 - Recortes obtenidos tras el procesado.

Una vez que se ha extraído el rostro de la imagen se procede a su clasificación en la carpeta de salida, la cual ha sido descrita anteriormente en el apartado 4.1.1.

Para ello se abre el fichero TXT adjuntado en la base de datos FERET, el cual contiene el año de nacimiento del individuo. Este dato ligado al año en el que se ha tomado la foto nos da la edad del sujeto fotografiado en el mismo día en el que se realizó la misma.

A mayores este fichero lleva anotado el género del sujeto. La unión de ambos datos, edad y género, permiten realizar el guardado del recorte facial en la carpeta correspondiente.

Este subproceso de guardado está controlado por la existencia del fichero TXT, ya que como se mencionó en el apartado 3.3.1, no todas las imágenes poseen este fichero por ser un programa desarrollado a lo largo de 3 años. Es decir, todas aquellas imágenes que carecen de archivo TXT asociado, serán descartadas a la hora de distribuirlas en la carpeta BaseIMG_ACON.

En este punto, el programa finaliza la ejecución del bucle y manda un “true” a través de la memoria compartida al segundo proceso para que comience su ejecución.

Por último, este proceso vuelve a estar en reposo esperando un cambio en la carpeta BaseIMG, ya sea por la adición de una nueva imagen o por la sustracción de alguna ya presente. El caso de la eliminación se contempla debido a posibles adulteraciones en la base de datos, es decir, que una imagen que haya sido añadida se haya procesado de forma errónea y modifique los valores finales de la aplicación.

En la ilustración 60 se puede observar un ejemplo de esta casuística. En este caso el clasificador no ha sido capaz de reconocer el rostro debido a la presencia del flequillo. Esta casuística se verá con mayor profundidad en el capítulo V - Resultados.

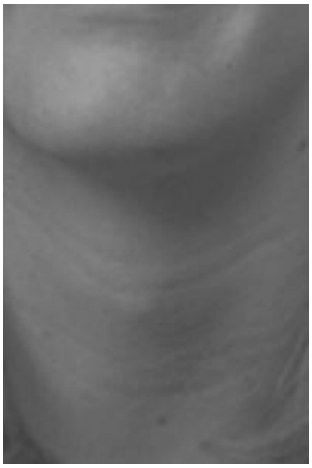


Ilustración 60 - Imagen mal procesada.

4.2.1.3 Salida del proceso

A continuación, se muestran varios ejemplos de imágenes procesadas.

Ilustración 61 - Imágenes acondicionadas. Hombre 30-39 años.

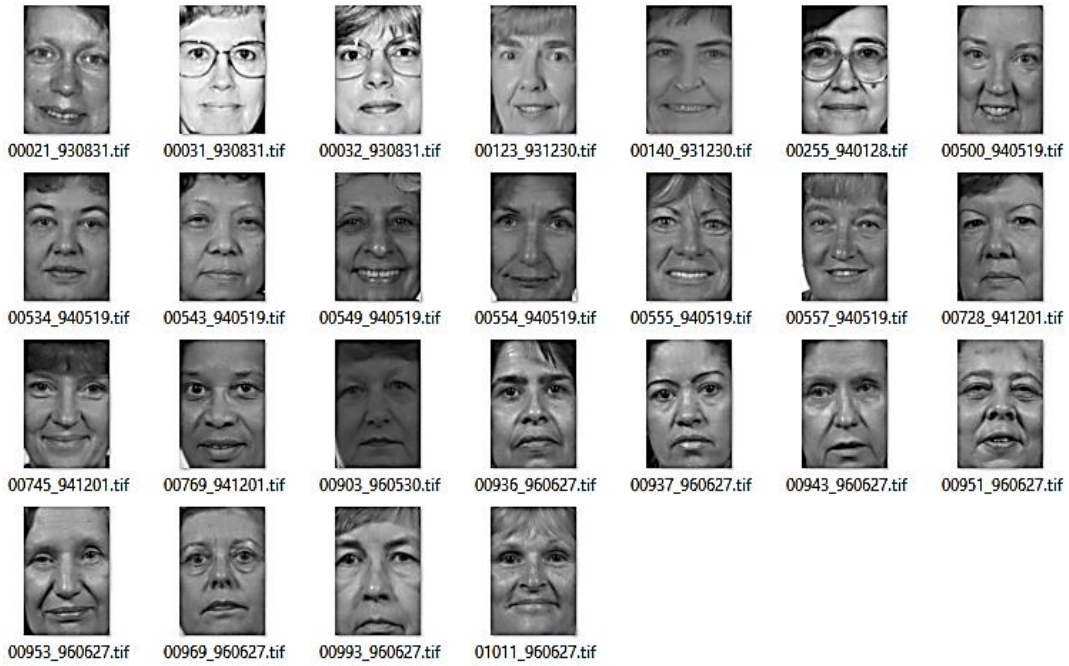


Ilustración 62 - Imágenes acondicionadas. Mujer 50-59 años.

4.2.2 Hist_base_datos

El segundo proceso, como se ha mencionado anteriormente, será el encargado de calcular los histogramas ULBP de las imágenes resultantes del proceso ya descrito.

4.2.2.1 Inicialización

La señal de comienzo de ejecución de este proceso viene marcada por la finalización del bucle de tratamiento de imágenes de "Proc_base_datos". Como se ha mencionado, esta indicación de comienzo llega a mediante un segmento de memoria reservado para la comunicación entre ambos procesos.

Por ello al ejecutar el proceso que nos ocupa, lo primero que hace es conectarse a las dos memorias compartidas creadas por "Proc_base_datos". En el apartado 4.1.2 se puede ver más información acerca de los segmentos de memoria compartidos.

Posteriormente el proceso accede a la carpeta BaseIMG_ACON, cuya estructura ha sido descrita en el apartado 4.1.1. Se conoce el número de subcarpetas en las que se divide el directorio mencionado, siendo 14 carpetas en total. Estas corresponden a las 7 divisiones en rangos de edad y a la posterior subdivisión en géneros.

Una vez se accede al directorio se cargan las imágenes de forma individual, para el posterior cálculo del histograma. La imagen es cargada en blanco y negro de forma directa a través del "imread()", ya que la escala de color de las imágenes obtenidas en el anterior proceso es escala de grises.

Previo al paso del cálculo del histograma ULBP se necesitan conocer los siguientes datos, los cuales se obtienen secuencialmente a partir del dato anterior.

- Media

  Se procesa la imagen cargada para obtener una nueva matriz imagen resultante. Para ello se aplicarán operaciones entre píxeles, las cuales han sido descritas en el apartado 2.1.3.2.1.

  El recorrido de la matriz de píxeles perteneciente a la imagen inicial se realiza sin tener en cuenta un marco de tamaño un píxel. Al situarse sobre un píxel a analizar, se calcula la media de los 8 píxeles vecinos asignando este valor en la misma posición del píxel analizado de la nueva matriz resultante.

En la ilustración 63 se muestran ejemplos de este procesamiento. La imagen media se observa que es más oscura, debido a que se ha normalizado su histograma antes de mostrar la imagen.

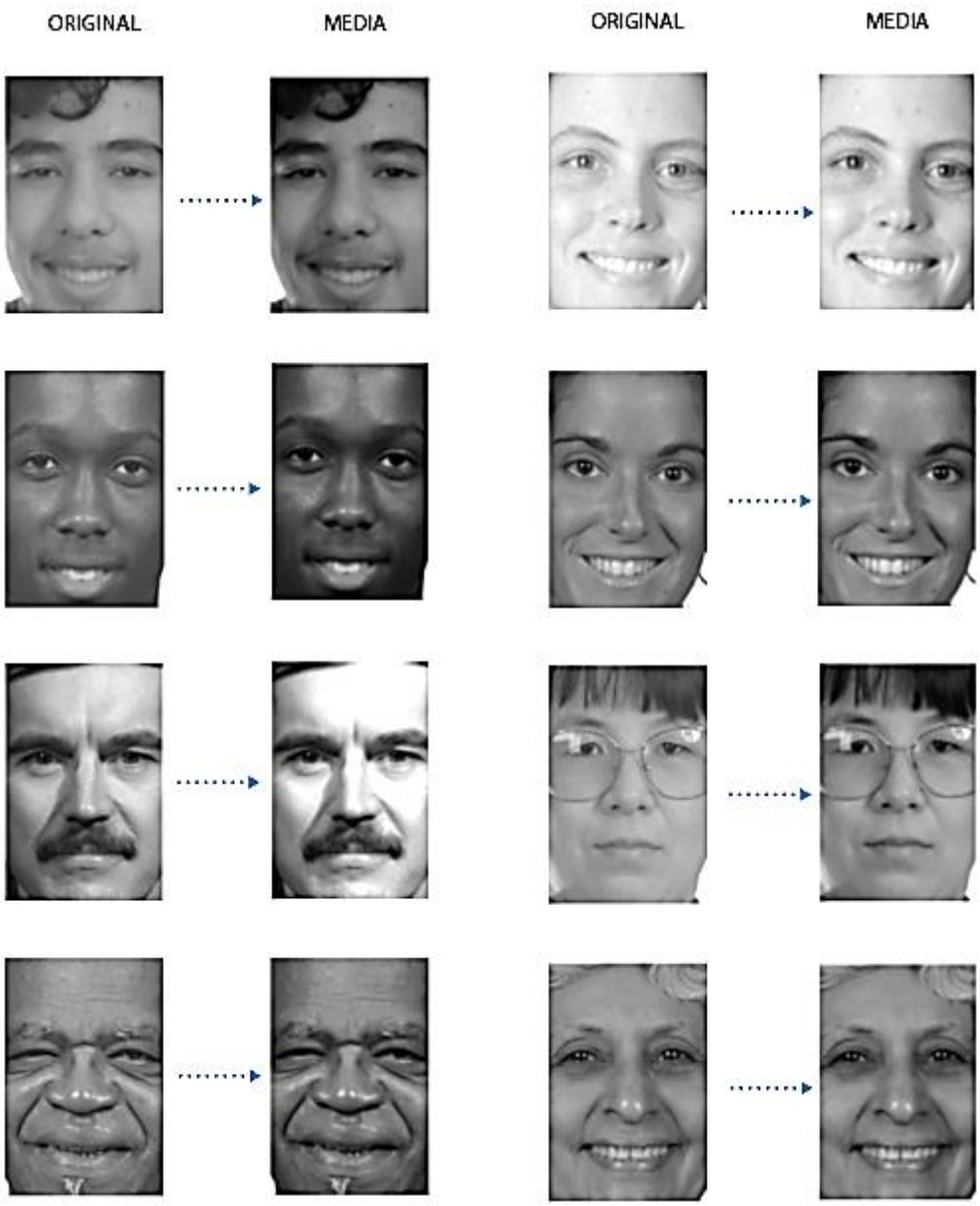


Ilustración 63 - Ejemplos obtención imagen media.

- Imagen LBP

La imagen resultante del cálculo anterior es la entrada para obtener la imagen LBP. En el aparatado 3.4 se ha explicado cómo obtener una imagen LBP y ULBP. La imagen LBP que obtendremos en esta función sigue ese mismo proceso.

Antes de comentar el proceso se fijan dos ratios, el número de vecinos que se analizarán y el radio de acción. En esta aplicación se empleará un radio de valor 1 y el número de vecinos establecido será 8. En la ilustración 64 se puede observar cómo influye la modificación de estos parámetros.

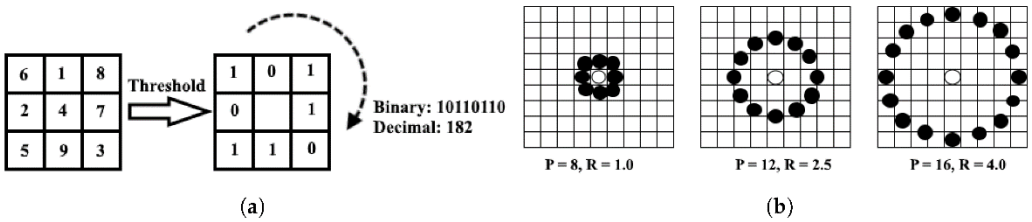


Ilustración 64 - Variaciones parámetros R y P.

A mayores la aplicación ha sido preparada para posibles usos con parámetros diferentes a los anteriormente mencionados. En estos casos se realizará una interpolación, para conseguir un resultado lineal. Se procede a explicar la obtención de la imagen LBP para los valores empleados, R=1 y P=8.

Se parte de la imagen media obtenida anteriormente y la imagen original. Como explicamos en el apartado 3.4, se empleará un valor umbral para conocer si el píxel vecino posee un valor superior o inferior para asignarle posteriormente 1 o 0 respectivamente.

Al analizar un píxel de la imagen original se almacenan el valor de los 8 vecinos y se comparan con el valor umbral. Este dato se obtiene de la imagen media obtenida anteriormente, cogiendo el valor del píxel situado en el mismo lugar espacial que el analizado de la imagen original.

Tras eso se compone el número binario de 8 dígitos. Se comenzará desde la esquina superior izquierda y se irán asociando en el sentido de las agujas del reloj. Un ejemplo de este proceso se observa en la ilustración 65.

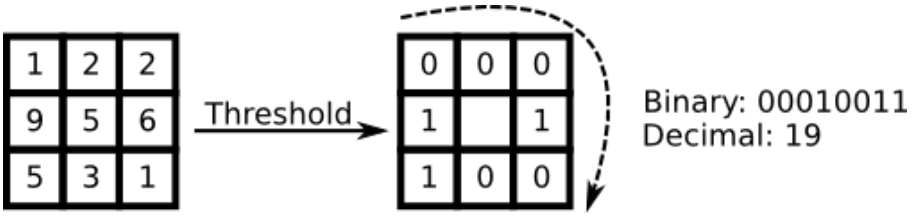


Ilustración 65 - Ejemplo aplicación algoritmo LBP.

Una vez se obtiene el número binario se convierte a formato decimal para ser asignado en una nueva matriz imagen resultante. El conjunto de todos los valores decimales genera la imagen LBP.

A continuación, se muestra una batería de ejemplos de esta transformación. Como se puede observar en estos ejemplos, lo más destacable de las imágenes LBP son los rasgos faciales y arrugas, ya que como se ha dicho este algoritmo se emplea en búsqueda de texturas.

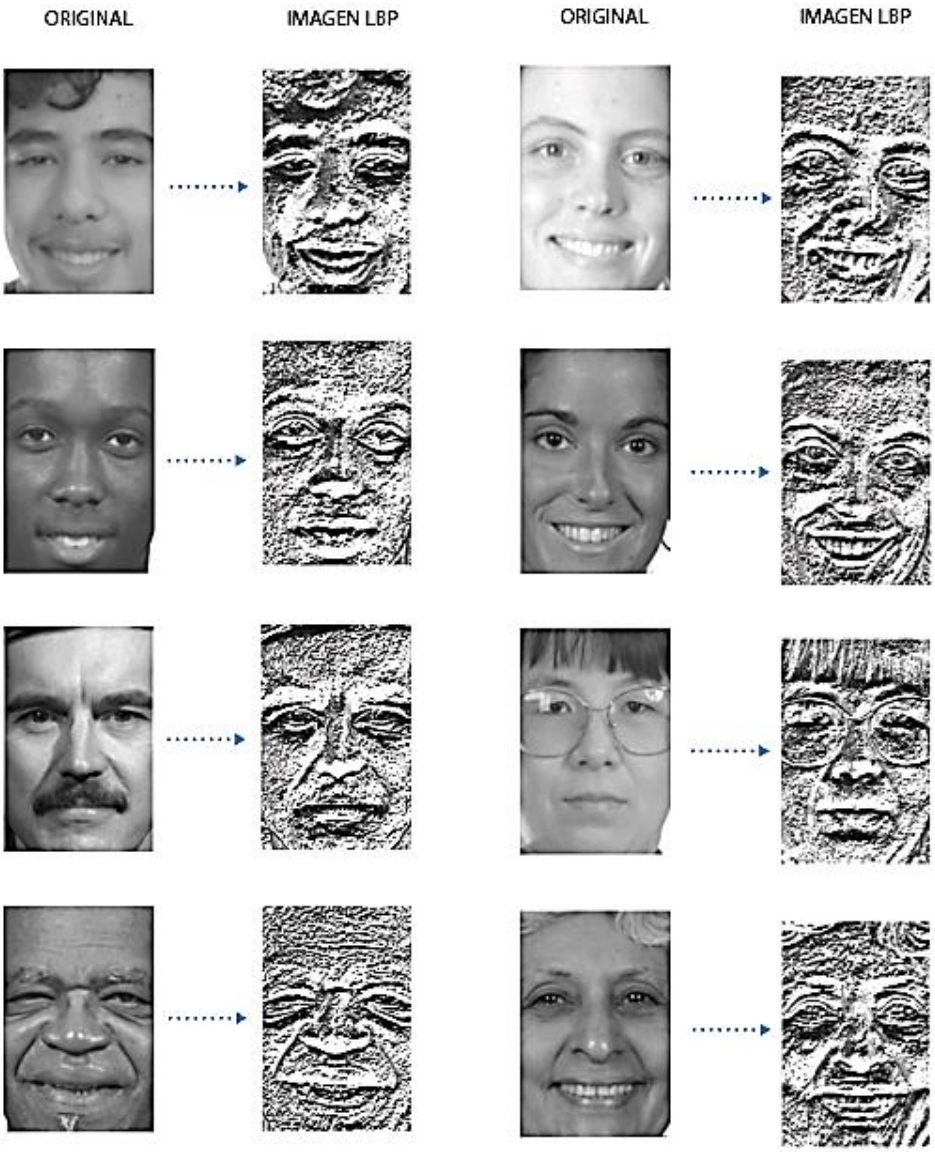


Ilustración 66 - Ejemplos obtención imagen LBP.

- Conversión LBP a ULBP

Finalmente, la imagen LBP obtenida en el paso anterior es transformada a ULBP, ya que este es el algoritmo que se desea aplicar. Para ello se carga en una LUT (Look Up Table) la secuencia de datos almacenada en un fichero DAT en la carpeta BaseHIST_Luts.

Este fichero contiene la secuencia de datos indicada en el apartado 4.1.1, la cual permite realizar una conversión entre los 256 niveles de intensidad de la imagen LBP a solo 59 niveles.

La secuencia de esta función es similar a las anteriores. Se recorre la matriz de píxeles de la imagen LBP. El valor de intensidad del píxel analizado se introduce en la LUT obteniendo uno nuevo, el cual es insertado, en la misma posición que el analizado, en una nueva matriz. Esta última matriz será la imagen ULBP respectivamente de la original introducida.

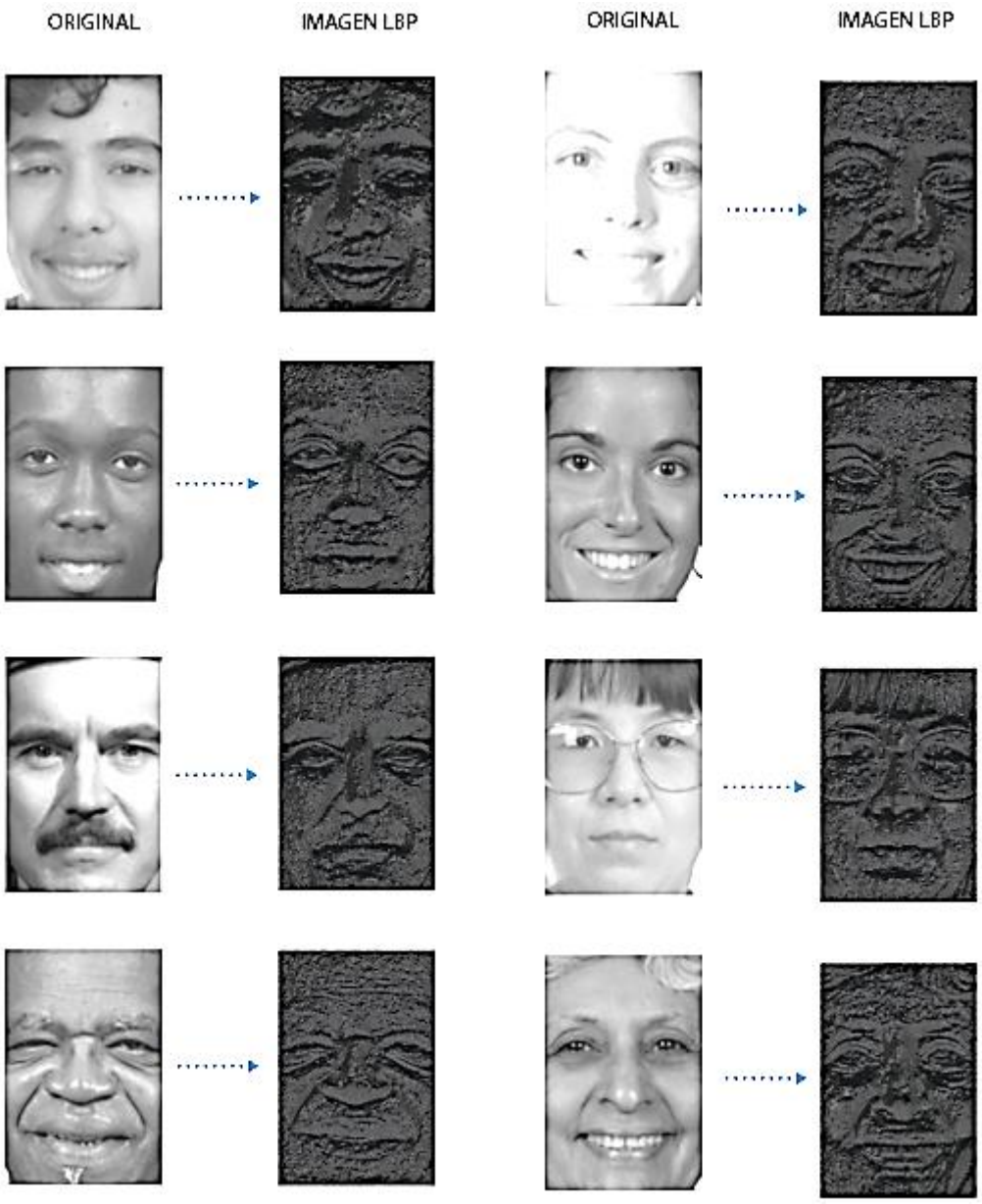


Ilustración 67 - Ejemplos obtención imagen ULBP.

4.2.2.2 Cálculo histograma ULBP

Con esta última imagen calculada, la ULBP, se procede a realizar el cálculo de su histograma.

El cálculo se realiza dividiendo la imagen en celdillas o teselas, de esta forma la obtención del histograma de cada una de ellas permite una mayor exactitud en los resultados finales en comparación con procesar la imagen completa para obtener un único histograma. Tras la realización de una gran batería de pruebas se decide hacer teselas de tamaño 25x25 píxeles. La imagen ULBP, al igual que la original, posee unas dimensiones de 200x300 píxeles. Esto genera 96 subimágenes referenciadas a la imagen ULBP.

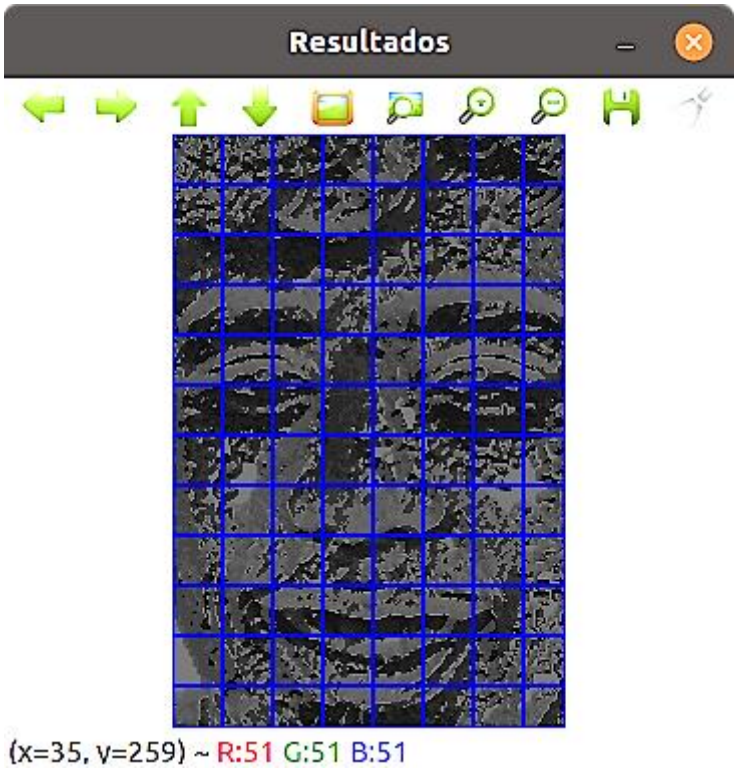


Ilustración 68 - Ejemplo división de celdillas sobre imagen original.

Cada celdilla es procesada obteniendo el histograma en respectivo. Para obtener el histograma la librería *OpenCV©* ofrece la función "cv::calcHist()". Para el resultado que se busca obtener, solo se empleará un único canal, sin mascara y el resultado será en una dimensión. En la ilustración 69 se muestran varias teselas y el resultado que se obtiene tras este proceso.

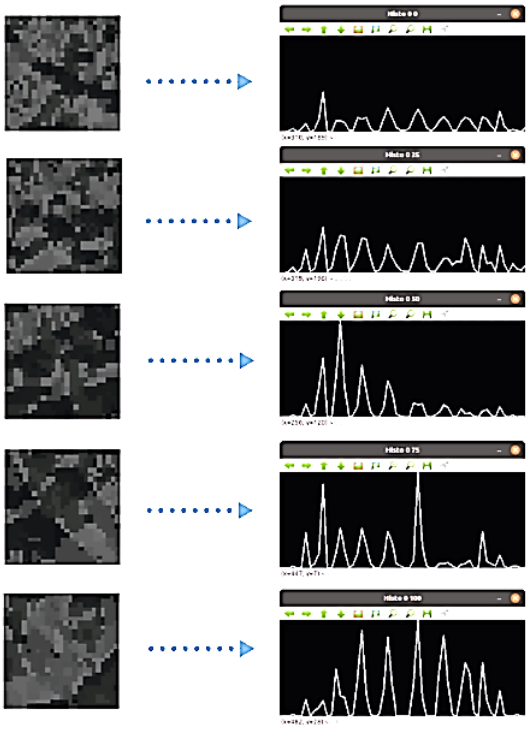


Ilustración 69 - Histograma respecto celdilla ULBP.

Una vez finaliza el bucle que se encarga de obtener los histogramas ULBP de la imagen a analizar, se deben sumar de forma concatenada para obtener un único histograma de la imagen ULBP. En el caso de la aplicación se ha descartado este proceso ya que en posteriores procedimientos el tratamiento se hace más tedioso y delicado. En el siguiente proceso, "Proc_Principal" se necesita poder acceder a cada histograma de forma singular.

Por todo ello se procede a almacenar por separado los histogramas ULBP, evitando una posterior desconcatenación en el "Proc_Principal". Este vector tendrá una longitud de 96 posiciones en las cuales se almacenará cada histograma obtenido manteniendo un orden establecido.

Cada vez que se analiza una imagen sus histogramas se van almacenando en un vector suma del mismo tamaño que el anteriormente descrito. De esta forma, al finalizar el tratamiento de todas las imágenes de una carpeta del directorio "BaseIMG_ACON" se habrá obtenido un único vector de tamaño 96 con todos los histogramas sumados. Es decir, en la primera posición se encontrará la suma de todos los histogramas correspondientes a la primera celdilla de todas las imágenes pertenecientes al rango de edad y genero analizado.

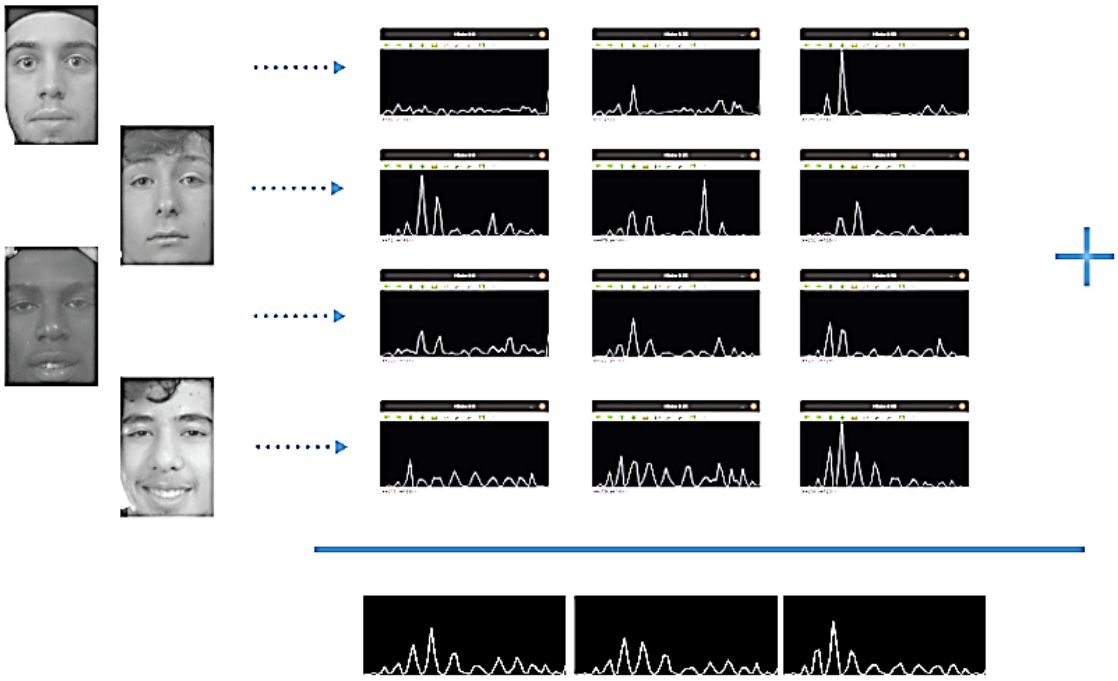


Ilustración 70 - Ejemplo suma de histogramas por tesela.

Una vez se obtiene el vector suma final y antes de pasar al siguiente conjunto de imágenes se procede a un post-procesado y a su guardado. Este procesado consiste en realizar una normalización cada histograma obtenido. Esta técnica ha sido descrita anteriormente en el apartado 2.1.3.2.2. Para realizarlo se emplea la función "normalize()" la cual genera un histograma de 59 niveles que no se verá afectado por el número de píxeles de la imagen.

Por último, se procede al guardado de este. Para ello se almacenará en la carpeta "BaseHIST_Data" con dos formatos distintos. El primer formato es png es decir como una imagen. Esto permitirá analizar a un controlador los histogramas obtenidos para detectar posibles variaciones anómalas o errores de obtención.

El segundo formato es XML. Este almacenará los datos de forma decimal, siendo estos los que se usarán en el siguiente proceso para su carga, ya que ofrece una gran exactitud. Un ejemplo de este formato es el siguiente, el cual ha sido extraído de la primera tesela del rango de edad 30-39 correspondiente al género femenino.

<?xml version="1.0"?>
<opencv_storage>
<Date>"Wed May 8 16:54:17 2019"</Date>
<radio>1</radio>
<numVecinos>8</numVecinos>
<histoULBP type_id="opencv-matrix">
<rows>59</rows>
<cols>1</cols>
<dt>f</dt>
<data> 0. 1.26161263e-03 3.89952981e-03 1.61715802e-02
2.29384098e-03 1.39924306e-02 9.47356373e-02 8.71659629e-03
2.40853317e-02 1.03452235e-01 1.71694010e-01 1.26161263e-03
2.80995537e-02 2.69411623e-01 3.84447753e-01 1.23752728e-01
5.16114244e-03 1.55981192e-02 2.20896900e-01 7.63504982e-01
2.96134889e-01 3.09668556e-02 1.37630466e-03 1.37630468e-02
9.23271030e-02 2.53698826e-01 2.48193607e-01 3.56692299e-02
1.20426659e-02 5.50521864e-03 2.00711098e-02 1.34763166e-01
1.63436174e-01 1.11251295e-01 2.08739545e-02 3.89952981e-03
1.53687354e-02 1.18820965e-01 2.93841034e-01 1.53113887e-01
1.83507279e-02 7.22559961e-03 6.77830055e-02 1.75478846e-01
1.26046568e-01 2.31677946e-02 4.01422195e-03 1.02305315e-01
7.94815943e-02 1.99564174e-02 7.34029151e-03 1.02764085e-01
2.15621069e-02 2.75260932e-03 2.38559470e-02 7.56967580e-03
3.67014576e-03 2.21355669e-02 1.</data>
</histoULBP>
</opencv_storage>

Una vez se realiza el guardado de los 96 histogramas, se procede con el procesado del siguiente directorio hasta finalizar todos ellos.

Al finalizar la lectura de todos los directorios el programa envía al segmento de memoria compartida con el primer proceso, una señal de finalización, quedando este, “Hist_base_datos”, en pausa a la espera de una nueva indicación del proceso “Proc_base_datos” que vuelva a poner en marcha la ejecución.

4.2.2.3 Salida del proceso

Con este proceso se obtiene 96 archivos PNG y XML de cada rango de edad y género, es decir de las 14 subcarpetas que posee el directorio “BaseIMG_Acon”. La siguiente ilustración representa los 96 histogramas pertenecientes al rango de edad 40-49 de género femenino. Todos ellos han sido compuestos en una sola imagen representando la ubicación de la tesela de la que han sido obtenidos.

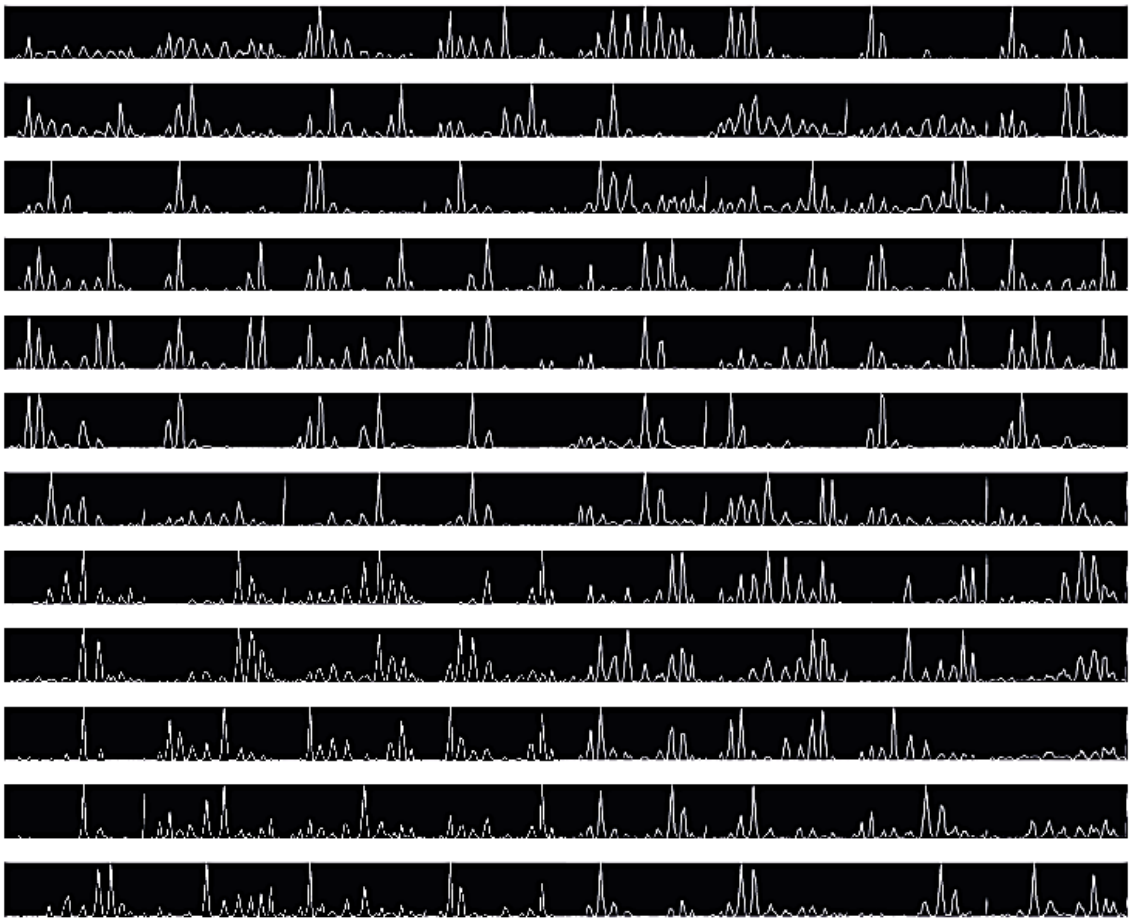


Ilustración 71 - Histogramas ULBP resultantes rango 40-49 F.

4.2.3 Proc_Principal

El último proceso, como se viene exponiendo, es el encargado de obtener las imágenes en tiempo real y de asignar a los rostros presentes un género y edad. A mayores, tal y como muestra la ilustración 51, existe un método de guardado para incrementar la base de datos con imágenes nuevas. Este subproceso está ligado a "Proc_Principal" y se expondrá posteriormente su uso y funcionamiento.

4.2.3.1 Captura y carga inicial

Como se explicó en el apartado 4.1.2, este proceso carece de memorias compartidas con alguno de los otros ya descritos, por lo que "Proc_Principal" es independiente en lo referente a la ejecución. No obstante, en ese mismo apartado se explicó la inclusión de tratamiento de señales para controlar cierres en la aplicación.

Las entradas que tendrá el proceso principal al ser ejecutado serán los archivos XML de la carpeta "BaseHIST_Data", la LUT del directorio "BaseHIST_Luts" y el clasificador Haar, siendo el mismo que el empleado en el primer proceso "Proc_base_datos". La carga de la LUT y del clasificador Haar se realiza de idéntica forma a las ya mencionadas en anteriores puntos del capítulo IV.

Los archivos XML generados anteriormente se cargarán al inicio de la aplicación, es decir cualquier modificación en estos durante la ejecución no será válida hasta el reinicio del proceso. De esta forma se protege la aplicación ante fallos derivados de esta carga durante el propio proceso.

La carga de estos ficheros se realiza diferenciando entre los rangos y géneros establecidos en la subdivisión de carpetas. La función encargada de ello, lee del fichero XML los puntos almacenados para posteriormente formar de nuevo el histograma. Una vez se han cargado los 96 archivos correspondientes a un rango de edad y género, se almacenan en un vector para poder acceder a ellos en cualquier momento. El resultado de esta función será un vector de vectores con una dimensión 14 y 96 respectivamente, almacenando así los 1344 histogramas.

La siguiente sentencia que se ejecuta es el acceso a la cámara. Para este paso *OpenCV©* ofrece un constructor llamado VideoCapture, el cual permite obtener imágenes de cualquier videocámara, no solo de la webcam. Para establecer un flujo continuo de imágenes, es necesario establecer un bucle infinito en el proceso, de no ser así se analizaría exclusivamente el primer frame capturado.

La ejecución del análisis de las imágenes obtenidas es idéntica al empleado en el primer proceso.

El frame que se analiza se transforma de color a escala de grises para mejorar el tratamiento del algoritmo ULBP. Para mejorar posibles fallos de saturación en la imagen se procede a ecualizar su histograma, como se puede apreciar en la ilustración 72. En el apartado 2.1.3.2.2 se expone su funcionamiento y utilidad.

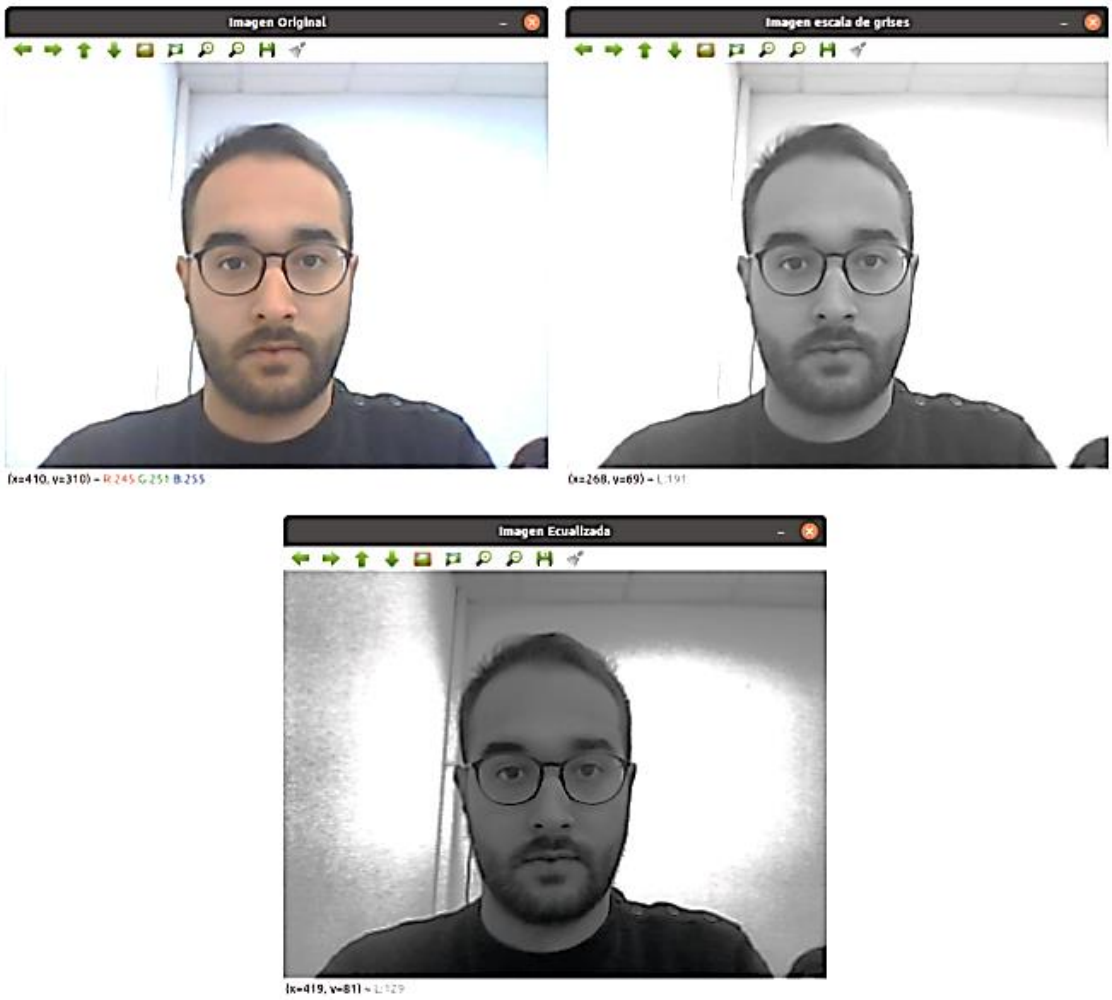


Ilustración 72 - Imagen original capturada, en escala de grises y ecualizada.

Una vez se obtiene la imagen ecualizada y en escala de grises, se procede a aplicar el clasificador HaarCascade para realizar la búsqueda de los rostros presentes en el frame que se está analizando. Este paso ha sido descrito anteriormente en el apartado 4.2.1.2, en el cual se explican todos los miembros necesarios en la función.

Tras detectar los rostros presentes se procede con u análisis de forma individual. Para ello primeramente se modificará el recorte facial dotándolo de unas dimensiones 200x300. Esta técnica ha sido explicada anteriormente en el punto 4.2.1.2.

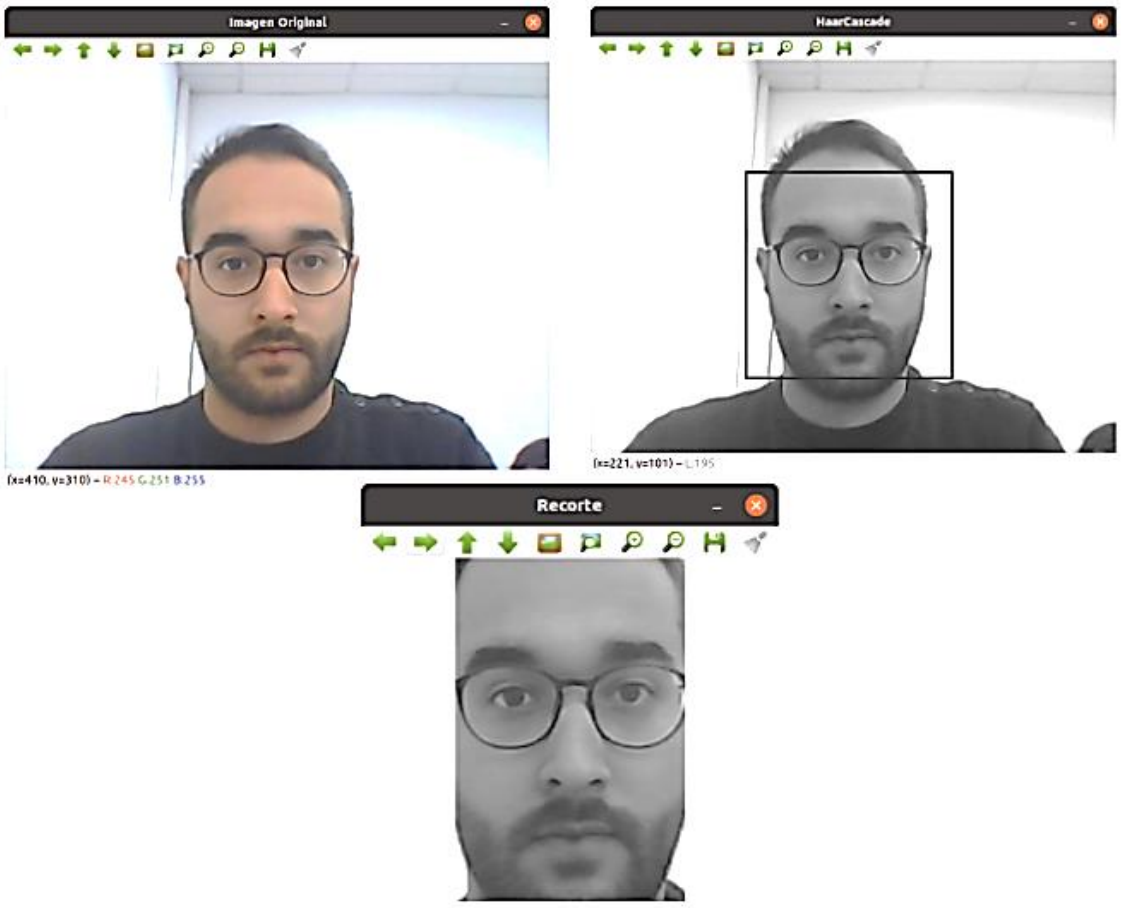


Ilustración 73 - Detección del rostro y recorte.

Antes de realizar el cálculo de la edad se aplican al recorte las tres operaciones ya descritas en el apartado 4.2.2.1, siendo estas el cálculo de la imagen media, la imagen LBP y la imagen ULBP.

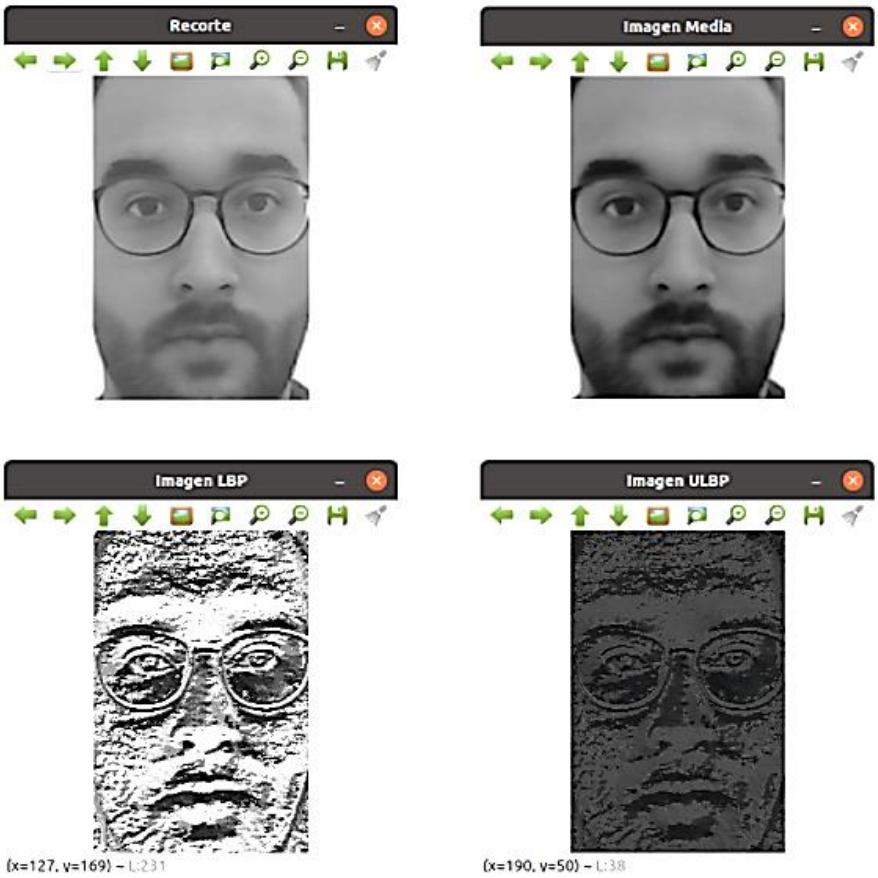


Ilustración 74 - Obtención imagen media, LBP y ULBP del recorte.

Una vez se obtiene la imagen ULBP, se comienza con el análisis de texturas por comparación para asignar género y edad al rostro del recorte.

4.2.3.2 Cálculo de edad y género

La obtención de estos factores se va a realizar a través de una comparación mediante histogramas ULBP, entre los almacenados anteriormente por el proceso "Hist_base_datos" y los que se van a obtener del recorte realizado anteriormente.

Para realizar la comparación se recorre el vector de vectores que contiene los histogramas separados por género y edad. Cada vector contiene los 96 histogramas correspondientes al mismo número de teselas de la media del rango de edad y género.

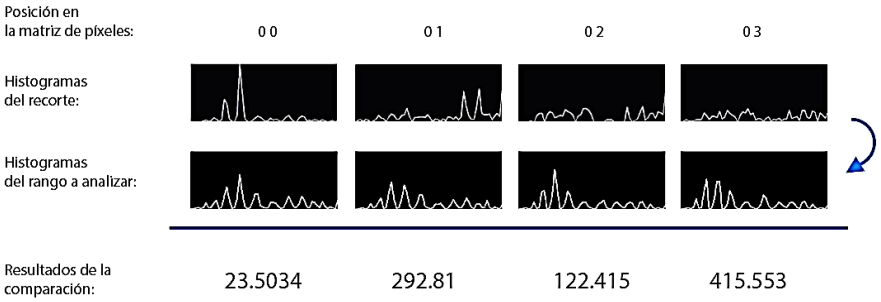


Ilustración 75 - Comparación entre histogramas.

La comparación se hará a nivel de celdillas, por lo que es necesario realizar la subdivisión del recorte en 96 teselas. Esto se logra siguiendo el procedimiento explicado en el apartado 4.2.2.2. Posteriormente se obtendrá el histograma de esta y será normalizado.

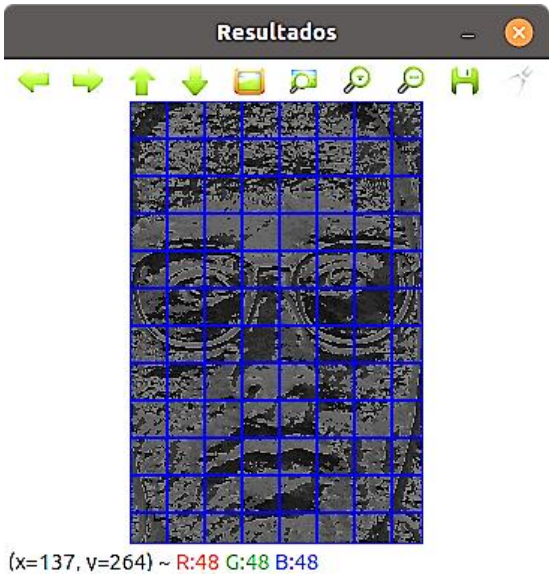


Ilustración 76 - Imagen ULBP dividida en 96 teselas.

Al acceder a cada celdilla resultante se compara con el histograma respectivo calculado en el proceso "Hist_base_datos". Para esta comparación el programa se

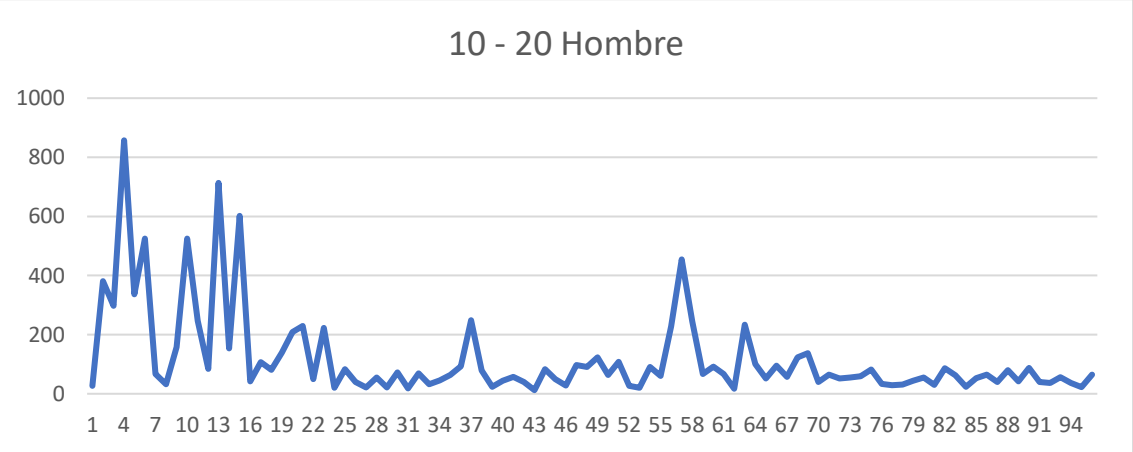
vale de la función “cv::compareHist()”, la cual está incluida en las librerías *OpenCV©* [46].

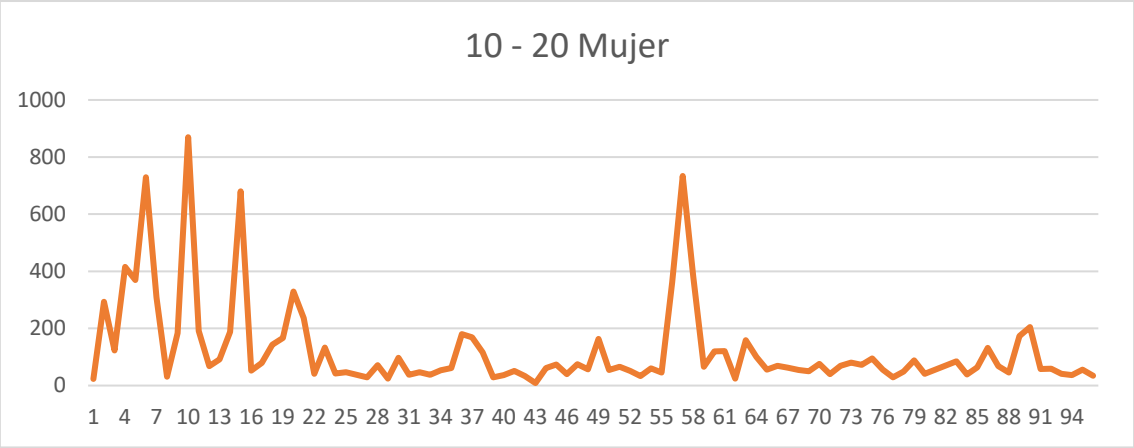
Esta función necesita tres argumentos. Los dos primeros son los histogramas que se desean comparar, en este caso el de la tesela que está siendo analizada del frame obtenido y el de la celdilla respectiva obtenida en el anterior proceso. El tercer argumento es el método de comparación que se desee emplear. Las librerías ofrecen los siguientes:

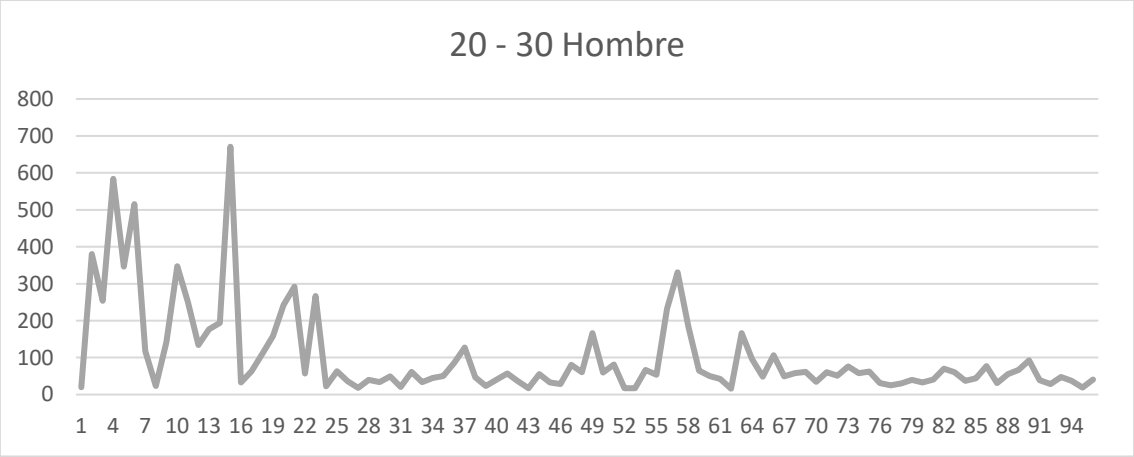
- CV_COMP_CORREL – Correlación
  - Coincidencia completa: 1.0
  - Intervalo de salida: 0 - 1
- CV_COMP_CHISQR – Chi-Cuadrado
  - Coincidencia completa: 0.0
  - Intervalo de salida: 0 - ∞
- CV_COMP_INTERSECT – Intersección
  - Coincidencia completa: ∞
  - Intervalo de salida: 0 - ∞
- CV_COMP_BHATTACHARYYA – Distancia de Bhattacharyya
  - Coincidencia completa: 0.0
  - Intervalo de salida: 0 – 1

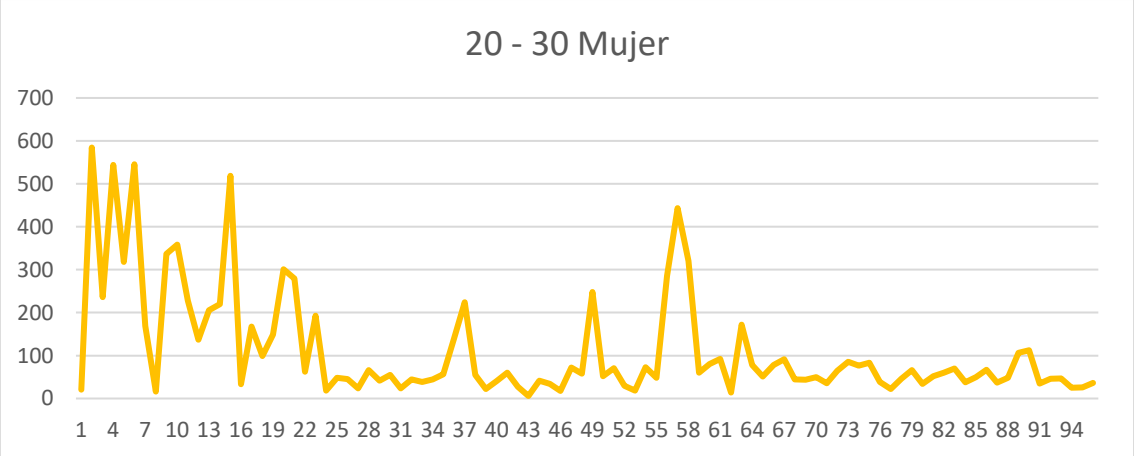
El que se empleará en este programa será “CV_COMP_CHISQR”, es decir a través del algoritmo Chi-Cuadrado. Por lo que cuando se tenga un caso desfavorable el valor resultante de la función será muy alto y en contraposición, para casos favorables el resultado será cercano a 0.

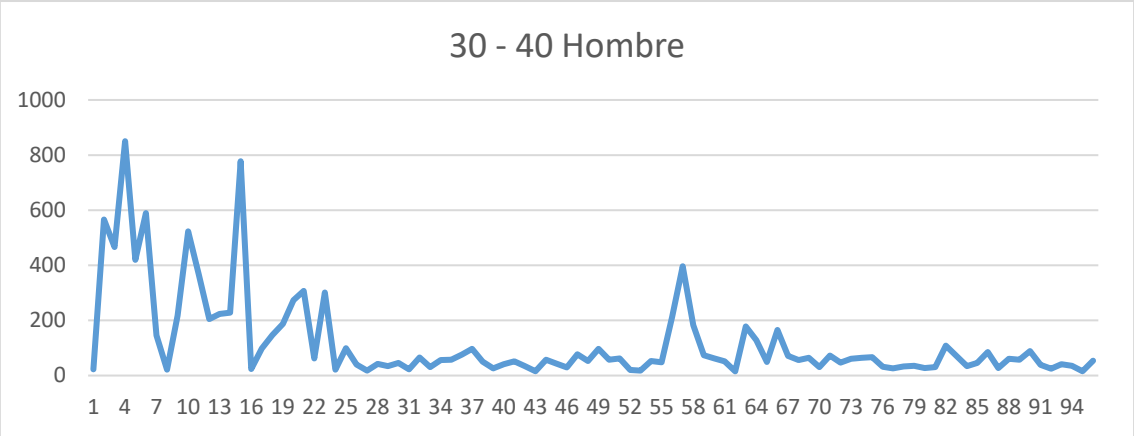
Estos resultados se irán almacenando en un vector, obteniendo con ello 96 datos resultantes de la comparación, cada uno correspondiente con la zona de la imagen que se está analizando. Se empezará por la tesela superior izquierda y se acabará en la inferior derecha dando lugar a 96 posiciones en el eje de ordenadas. A continuación, se muestra un ejemplo de los 14 rangos.

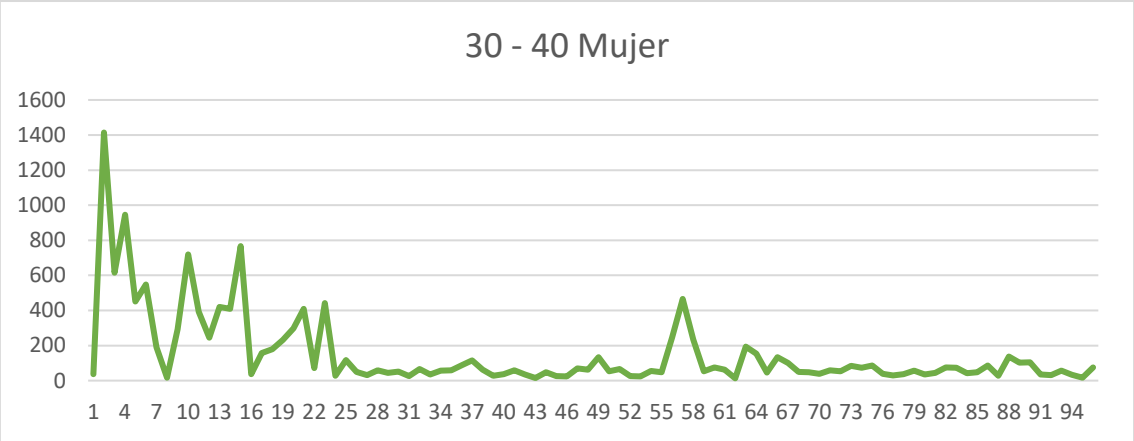


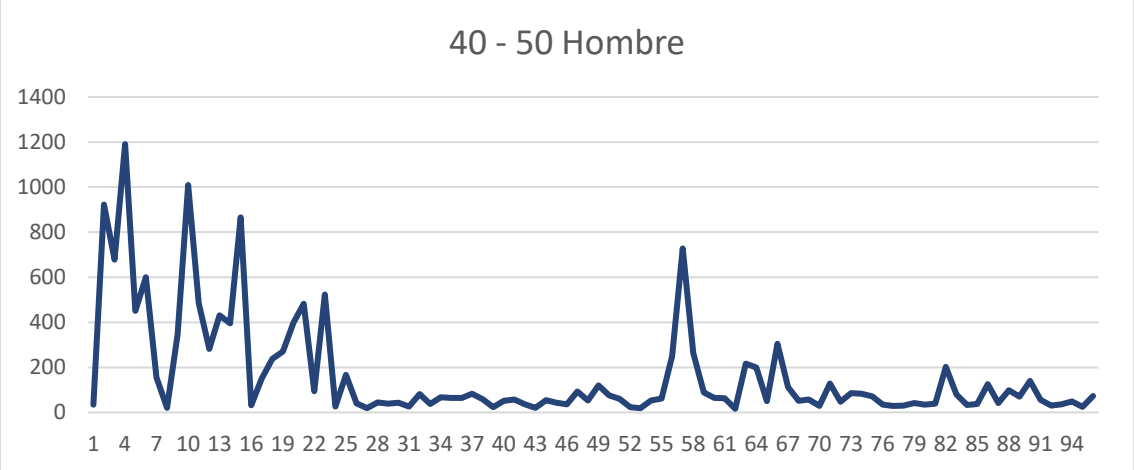

10 - 20 Mujer
1000
800
600
400
200
0
1 4 7 10 13 16 19 22 25 28 31 34 37 40 43 46 49 52 55 58 61 64 67 70 73 76 79 82 85 88 91 94

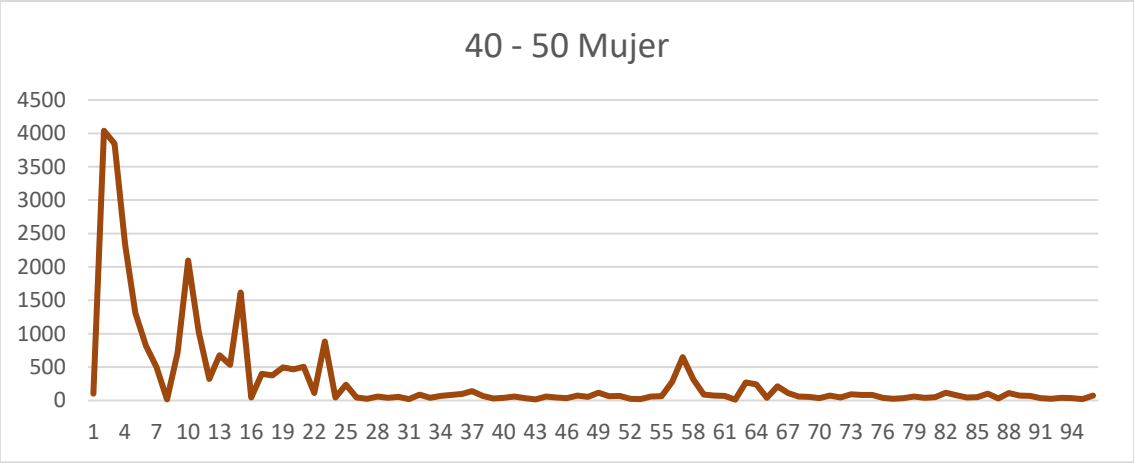

20 - 30 Hombre
800
700
600
500
400
300
200
100
0
1 4 7 10 13 16 19 22 25 28 31 34 37 40 43 46 49 52 55 58 61 64 67 70 73 76 79 82 85 88 91 94

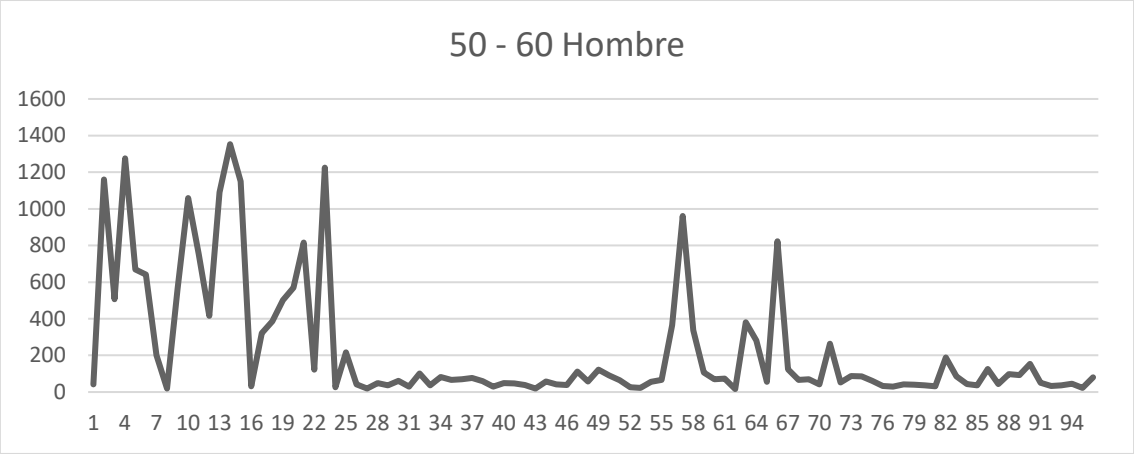

20 - 30 Mujer
700
600
500
400
300
200
100
0
1 4 7 10 13 16 19 22 25 28 31 34 37 40 43 46 49 52 55 58 61 64 67 70 73 76 79 82 85 88 91 94

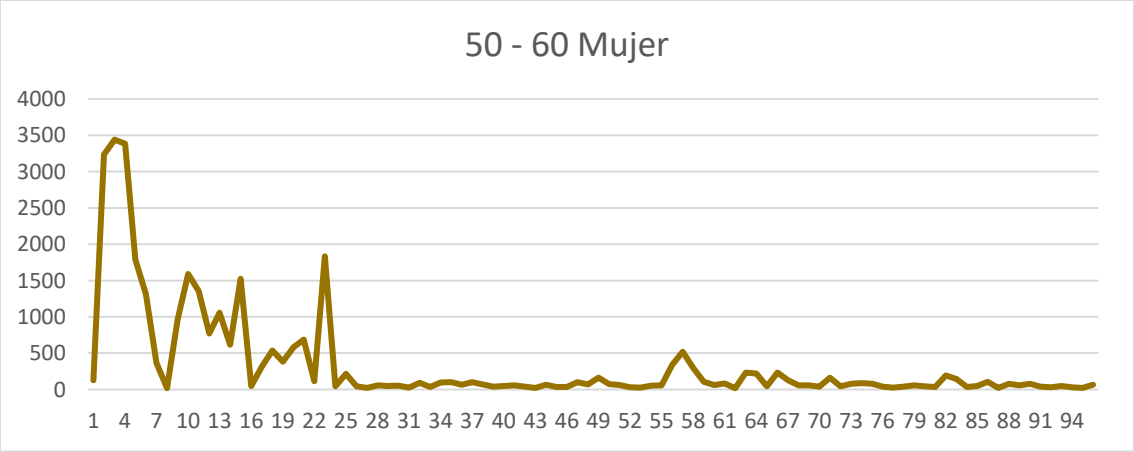

30 - 40 Hombre
1000
800
600
400
200
0
1 4 7 10 13 16 19 22 25 28 31 34 37 40 43 46 49 52 55 58 61 64 67 70 73 76 79 82 85 88 91 94

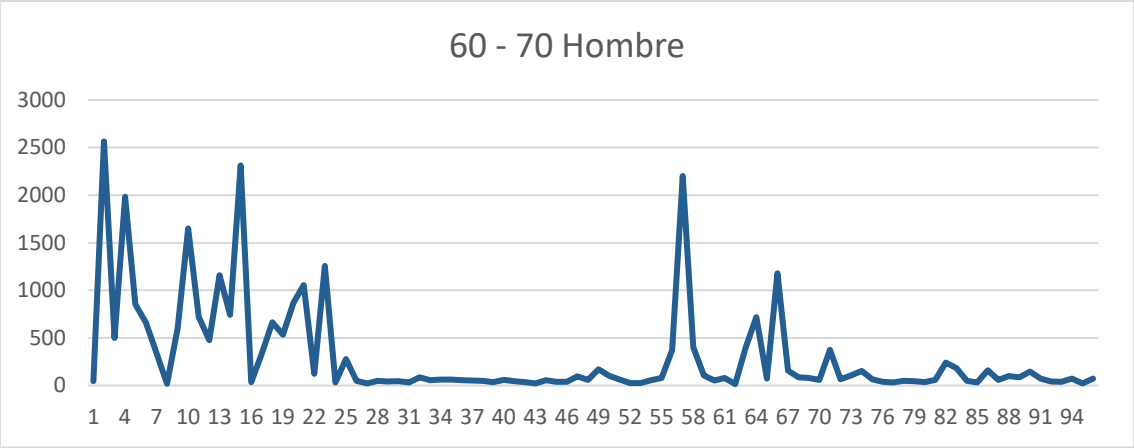

30 - 40 Mujer
1600
1400
1200
1000
800
600
400
200
0
1 4 7 10 13 16 19 22 25 28 31 34 37 40 43 46 49 52 55 58 61 64 67 70 73 76 79 82 85 88 91 94

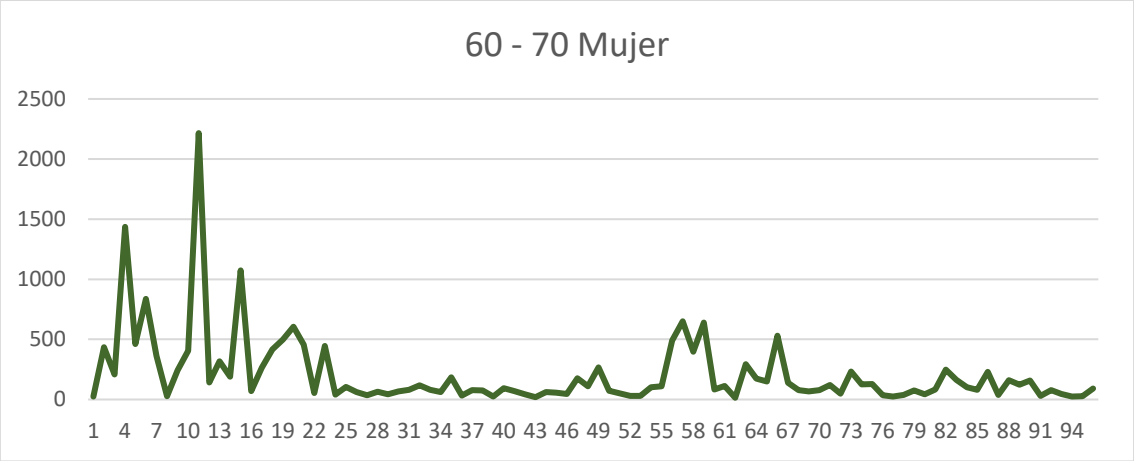

40 - 50 Hombre
1400
1200
1000
800
600
400
200
0
1 4 7 10 13 16 19 22 25 28 31 34 37 40 43 46 49 52 55 58 61 64 67 70 73 76 79 82 85 88 91 94

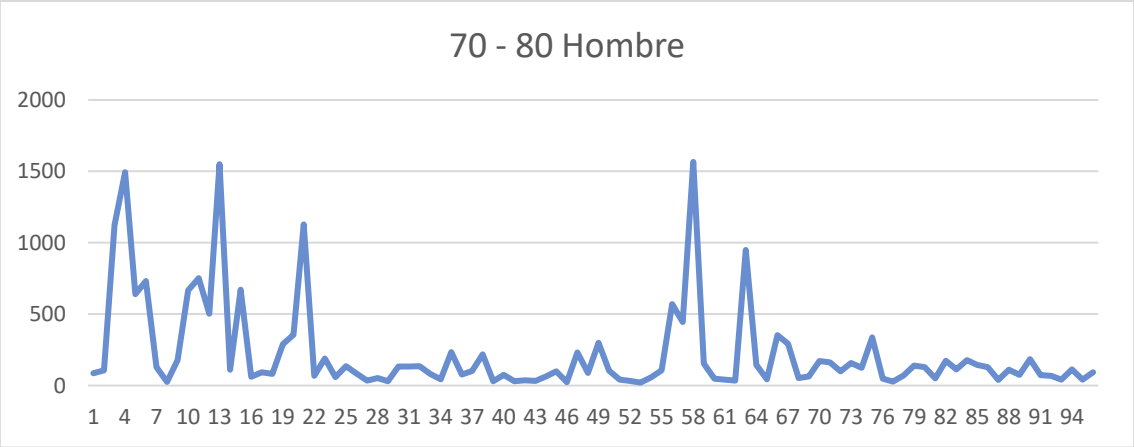

40 - 50 Mujer
4500
4000
3500
3000
2500
2000
1500
1000
500
0
1 4 7 10 13 16 19 22 25 28 31 34 37 40 43 46 49 52 55 58 61 64 67 70 73 76 79 82 85 88 91 94

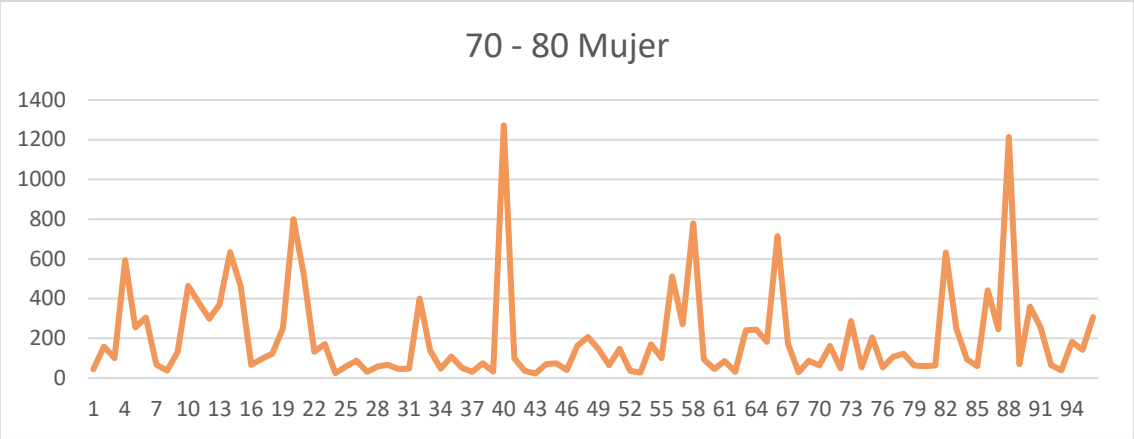

50 - 60 Hombre
1600
1400
1200
1000
800
600
400
200
0
1 4 7 10 13 16 19 22 25 28 31 34 37 40 43 46 49 52 55 58 61 64 67 70 73 76 79 82 85 88 91 94


50 - 60 Mujer
4000
3500
3000
2500
2000
1500
1000
500
0
1 4 7 10 13 16 19 22 25 28 31 34 37 40 43 46 49 52 55 58 61 64 67 70 73 76 79 82 85 88 91 94


60 - 70 Hombre
3000
2500
2000
1500
1000
500
0
1 4 7 10 13 16 19 22 25 28 31 34 37 40 43 46 49 52 55 58 61 64 67 70 73 76 79 82 85 88 91 94


60 - 70 Mujer
2500
2000
1500
1000
500
0
1 4 7 10 13 16 19 22 25 28 31 34 37 40 43 46 49 52 55 58 61 64 67 70 73 76 79 82 85 88 91 94


70 - 80 Hombre
2000
1500
1000
500
0
1 4 7 10 13 16 19 22 25 28 31 34 37 40 43 46 49 52 55 58 61 64 67 70 73 76 79 82 85 88 91 94



Posteriormente se analizan estos datos respecto de un umbral establecido inicialmente. Este dato se fija en 60 posteriormente a la realización de una batería de pruebas para ajustarlo. Para el análisis de los datos obtenidos anteriormente se procede con la realización de una comparación entre el valor umbral y estos.

En caso de que el dato obtenido sea inferior al umbral se considerará que se acerca esa tesela a la celdilla patrón obtenida en el proceso anterior. En la siguiente ilustración podemos ver el resultado de dicha comparación. Los cuadros en rojo indican que el valor obtenido es menor que el umbral.

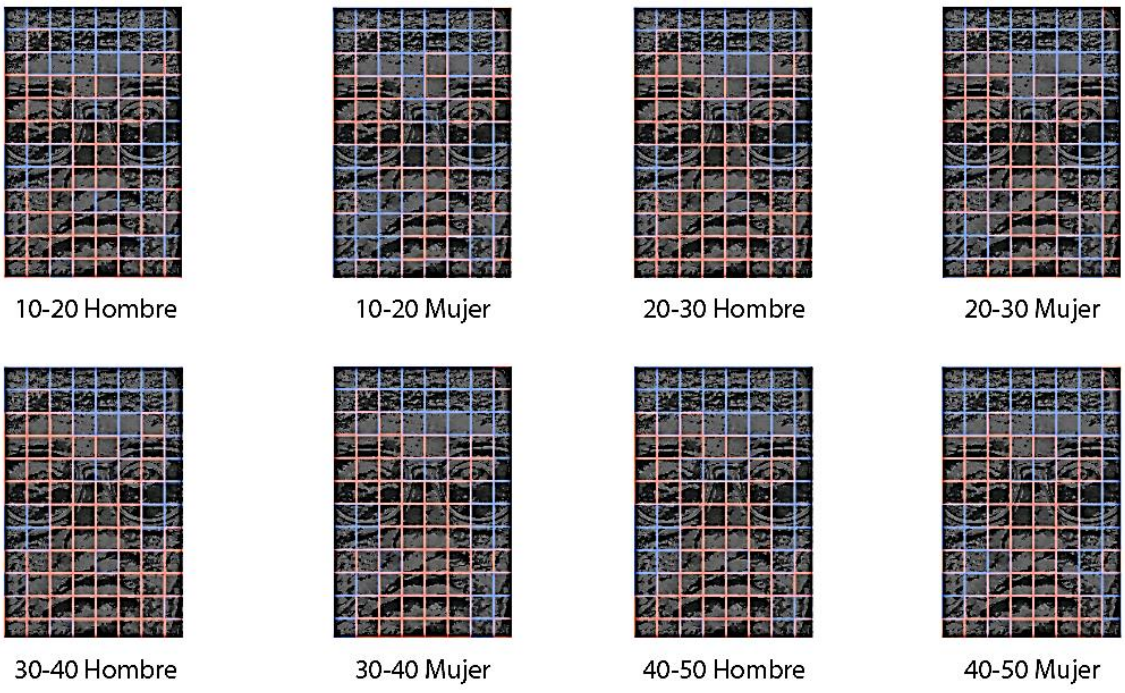


Ilustración 77 - Comparación recorte con histogramas por rangos I.

Ilustración 78 - Comparación recorte con histogramas por rangos II.

Para decidir cuál es el rango más acertado se usará el número de teselas que suponen aciertos, es decir las que estén en color rojo. En la siguiente ilustración se muestra un ejemplo de ello.

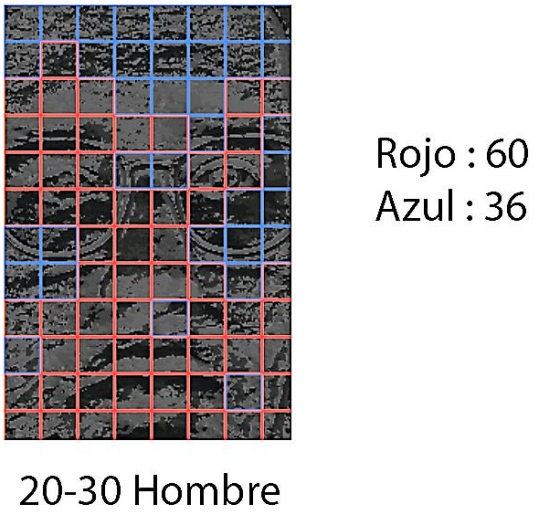


Ilustración 79 - Conteo teselas con aciertos.

Para el cálculo de la edad se suman los datos de los rangos de edades indiferentemente del género asignando a ese rostro el rango que contenga el valor

más alto. En el caso del género se repetirá esta fórmula, se suman los valores correspondientes a cada género, sin considerar el rango de edad.

A continuación, se da una muestra de los datos que se obtienen:

- Rango: 10-20 Suma Teselas: 124
- Rango: 20-30 Suma Teselas: 129
- Rango: 30-40 Suma Teselas: 124
- Rango: 40-50 Suma Teselas: 113
- Rango: 50-60 Suma Teselas: 105
- Rango: 60-70 Suma Teselas: 84
- Rango: 70-80 Suma Teselas: 67
- Número de veces hombre: 386
- Número de veces mujer: 360

En este punto se conoce el rango de edad y el género que supuestamente posee el rostro que se está analizando en tiempo real. Estos datos son muy fluctuantes debido a la iluminación en la imagen, si se trabaja con una cámara con autoajuste, etc....

Por los posibles fallos anteriormente descritos, se decide discretizar el resultado respecto al tiempo. Es decir, se ponderan los valores obtenidos en un tiempo de un segundo y medio.

Entre el segundo 0 y el 1.5 (sin incluirlo) se sumarán por separado los datos obtenidos. En el caso del género se sumará cuantas veces el resultado es Hombre o Mujer. Para la edad, se sumará el número de años asignado a ese rostro.

Cuando el tiempo de ejecución llega o supera los 1.5 segundos, se reinicia el tiempo poniéndolo a 0. Es en este momento cuando se vuelve a asignar los valores que se imprimirán en la imagen. El nuevo género que aparecerá es el que se haya obtenido mayoritariamente durante los anteriores instantes. En cuanto al género, se realizará una media entre los valores obtenidos. Es decir, la suma que se ha ido haciendo será dividida por el número de veces que se ha obtenido un valor.

Un ejemplo de los datos que se pueden obtener son los siguientes:

- Media Hombre: 24
- Suma Edad: 620
- nveces Hombre: 25
- nveces Mujer: 0

El dato “Media Hombre” será el valor de la edad que aparecerá sobre la imagen. Este es obtenido al hacer la división de “Suma Edad” entre la suma de “nveces Hombre” y “nveces Mujer”. El género se asignará en función del dato más grande entre “nveces Hombre” y “nveces Mujer”.

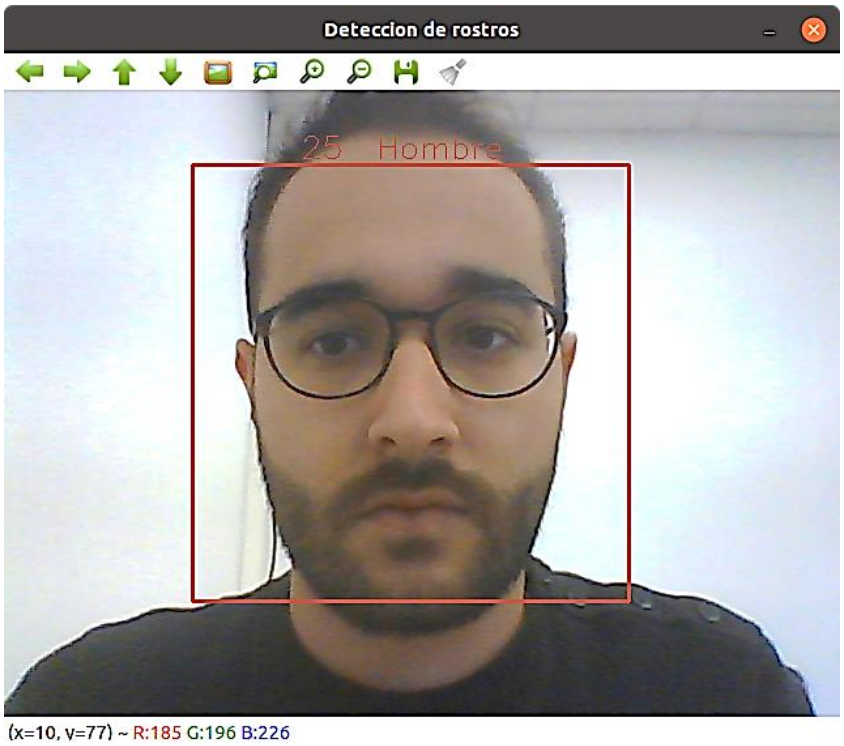


Ilustración 80 - Resultado impreso en la imagen.

Para finalizar el análisis de los resultados, estos se imprimirán en la imagen original obtenida mediante la cámara. En la ilustración 80 podemos ver un ejemplo del resultado.

Este proceso de cálculo ha sido descrito para un único rostro, pero la aplicación es capaz de procesar más de un rostro al mismo tiempo. Para ello sería necesario aumentar la tasa de refresco de datos por pantalla, pasando del 1.5s actual a 2s en el caso de dos rostros presentes.

Una vez se ha finalizado el análisis de los rostros presentes en ese frame, el programa pasará al siguiente frame para ejecutar el análisis. Este proceso se ejecuta de forma continua hasta que un operador decida interrumpirlo.

4.2.3.3 Guardado

La interrupción de la ejecución, está acompañada del guardado de los rostros presentes en el último frame. De esta forma se realimenta la base de datos con nuevos individuos ya analizados.

En la ilustración 50, se ve este proceso en el flujo de la aplicación. Pero solo se llamará a este proceso en caso de que el operador decida realizar un guardado del mismo.

Para realizar el guardado se partirá del último frame obtenido, de los rectángulos resultantes del HaarCascade y de los valores que se han obtenido tras el análisis de histogramas.

Lo primero será obtener los rostros en la imagen, para ello se realizan los recortes pertinentes a partir de la imagen y del vector de rectángulos resultante de haber aplicado el clasificador Haar.

Acto seguido se mostrarán por pantalla los recortes de forma individual y en el terminal los valores de edad y género asociados a dichos recortes. La aplicación preguntará si los datos obtenidos son correctos, para lo que se esperará la respuesta del operador. En caso positivo se realizará el guardado y en caso negativo se pedirán los valores correctos al usuario a través del teclado.

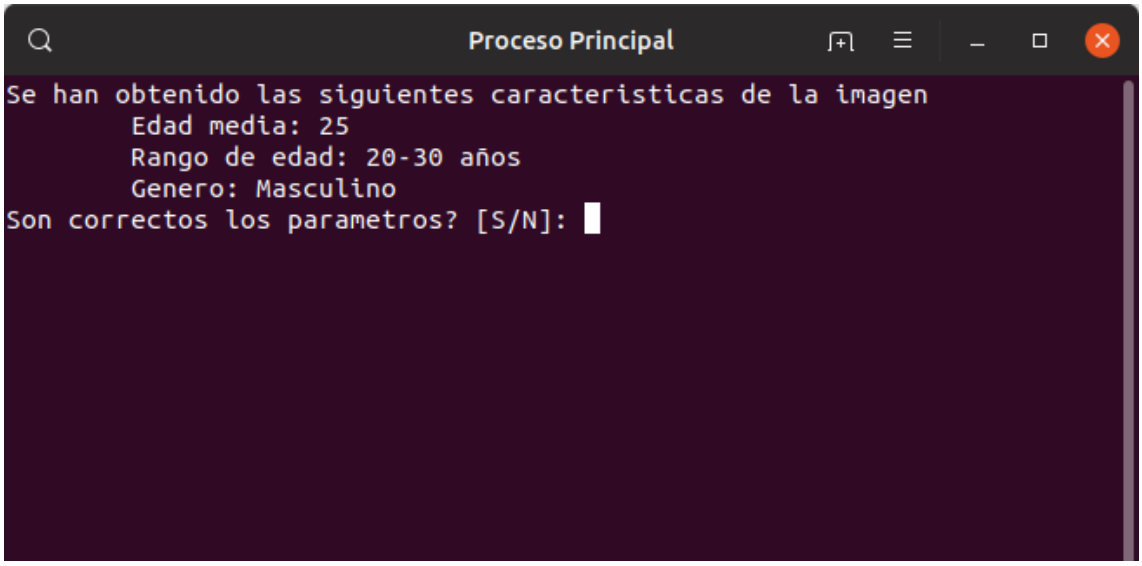


Ilustración 81 - Solicitud de guardado.

El guardado se enfoca en seguir el estándar otorgado por la base de datos FERET. Es decir, se guardará el recorte con las dimensiones 200x300, para mantener una misma escala y se generará un fichero TXT con las mismas entradas que los presentes en la base de datos. De esta forma la aplicación no deberá ser modificada.

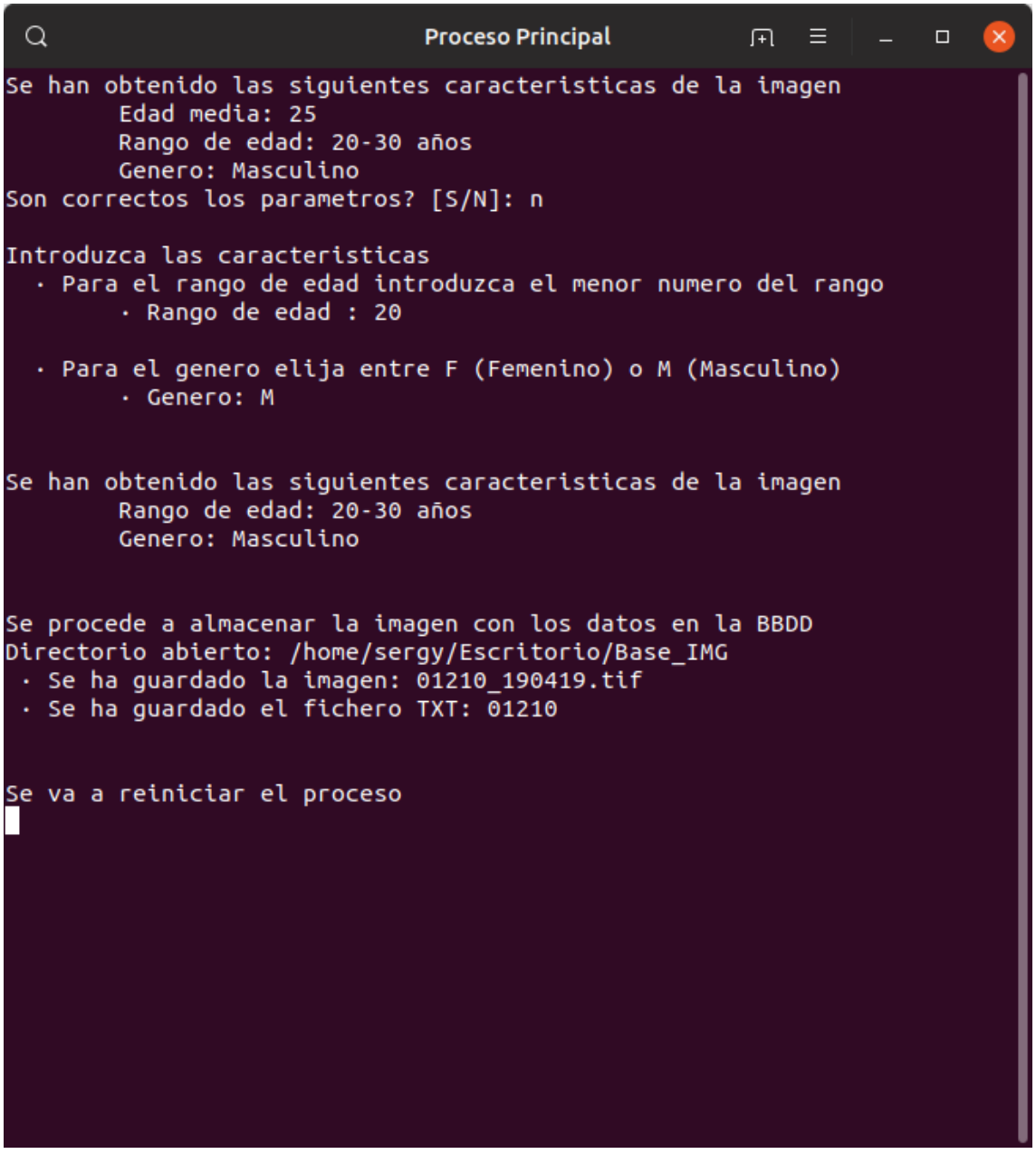


Ilustración 82 - Corrección de parámetros previa al guardado.

Por último, el proceso “Proc_Principal” se reiniciará autónomamente sin la necesidad de ejecutar la orden de inicio, es decir se abrirá un nuevo terminal el cual comenzará la ejecución de este proceso.

4.3 Puesta en marcha y ejecución continua

Para finalizar este apartado de desarrollo se procede a explicar la puesta en marcha y ejecución de la aplicación. Por las características propias de la aplicación se presupone una ejecución continua y autónoma, a excepción de las intervenciones de los operarios, es por ello que para llegar a este modo de funcionamiento se debe hacer primeramente una puesta en marcha en el lugar de implantación.

Lo primero de todo es ver es instalar las librerías de *OpenCV©* en el dispositivo en el que se pretende ejecutar la aplicación.

Debido a que esta puede ser instalada en un gran número de dispositivos hardware, se ha creado un "makefile" el cual facilite la tarea de la puesta en marcha y de la ejecución en caso de posibles fallos.

```
#########################################################################################################################################################
CC= g++
CFLAGS= -g -Wall -std=c++11 -pedantic -fexceptions -Wparentheses
FlagOpencv = `pkg-config --cflags --libs opencv`
EJS = Proc_base_datos Hist_base_datos Proc_Principal
EJS1 = Proc_base_datos_exe Hist_base_datos_exe Proc_Principal_exe
EJS2 = archivos.o outputs memorias
FunGenerales = Fun_comunes/compara.cpp Fun_comunes/estadist.cpp Fun_comunes/fichero.cpp Fun_comunes/Fun_Histo.cpp Fun_comunes/histo.cpp
Fun_comunes/lbp.cpp Fun_comunes/visualiza.cpp Fun_comunes/imBBDD.cpp
Proc_base_datos_o = Proc_base_datos/main.o
Proc_Principal_o = Proc_Principal/main.o
Hist_base_datos_o = Hist_base_datos/main.o
LibreriaLibb = /usr/lib/x86_64-linux-gnu/libtbb.so
#########################################################################################################################################################
compile: $(EJS)
Proc_base_datos: $(Proc_base_datos_o)
    $(CC) $(CFLAGS) Proc_base_datos/main.cpp $(FunGenerales) $(LibreriaLibb) -o Proc_base_datos/output $(FlagOpencv) | true
Hist_base_datos: $(Hist_base_datos_o)
    $(CC) $(CFLAGS) Hist_base_datos/main.cpp $(FunGenerales) $(LibreriaLibb) -o Hist_base_datos/output $(FlagOpencv) | true
Proc_Principal: $(Proc_Principal_o)
    $(CC) $(CFLAGS) Proc_Principal/main.cpp $(FunGenerales) $(LibreriaLibb) -o Proc_Principal/output $(FlagOpencv) | true
execute: $(EJS1)
Proc_base_datos_exe:
    gnome-terminal --command ./Proc_base_datos/output --hide-menubar -t "Procesamiento base de datos" --geometry 72x19+0+0 | true
Hist_base_datos_exe:
    gnome-terminal --command ./Hist_base_datos/output --hide-menubar -t "Obtención Histogramas"  --geometry 72x18+0+1000 | true
Proc_Principal_exe:
    gnome-terminal --command ./Proc_Principal/output --hide-menubar -t "Proceso Principal" --geometry 71x39+1000+0 | true
clean: $(EJS2)
archivos.o:
        rm Proc_base_datos/main.o | true
        rm Hist_base_datos/main.o | true
        rm Proc_Principal/main.o | true
outputs:
        rm Proc_base_datos/output | true
        rm Hist_base_datos/output | true
        rm Proc_Principal/output | true
memorias:
    ipcrm -a
```

Ilustración 83 - Makefile.

Las dos primeras instrucciones que se deben ejecutar serán “Clean” y “Compile”. La primera borrará todos los archivos residuales, en caso de existir, de compilaciones anteriores y las memorias que hayan sido creadas con anterioridad.

```
sergy@sergy-K53SC:~/Escritorio/TFGvFINAL$ make clean

 Borrando binarios *.o

rm Proc_base_datos/main.o | true
rm Hist_base_datos/main.o | true
rm Proc_Principal/main.o | true

 Borrando archivos ejecutables

rm Proc_base_datos/output | true
rm Hist_base_datos/output | true
rm Proc_Principal/output | true
ipcrm -a


sergy@sergy-K53SC:~/Escritorio/TFGvFINAL$
```

Ilustración 84 - Ejecución Make "Clean".

Y la instrucción “Compile”, como su nombre indica, compilará todos los fuentes de la aplicación, generando con ello los respectivos archivos binarios que pasarán a ser ejecutados.

```
sergy@sergy-K53SC:~/Escritorio/TFGvFINAL$ make compile
g++     -c -o Proc_base_datos/main.o Proc_base_datos/main.cpp
 Compilando Proc_base_datos

g++ -g -Wall -std=c++11 -pedantic -fexceptions -Wparentheses  Proc_base_datos/main.cpp Fun_com
unes/compara.cpp Fun_comunes/estadist.cpp Fun_comunes/fichero.cpp Fun_comunes/Fun_Histo.cpp Fu
n_comunes/histo.cpp Fun_comunes/lbp.cpp Fun_comunes/visualiza.cpp Fun_comunes/imBBDD.cpp /usr/
lib/x86_64-linux-gnu/libtbb.so -o Proc_base_datos/output `pkg-config --cflags --libs opencv` |
 true


g++     -c -o Hist_base_datos/main.o Hist_base_datos/main.cpp
 Compilando Hist_base_datos

g++ -g -Wall -std=c++11 -pedantic -fexceptions -Wparentheses  Hist_base_datos/main.cpp Fun_com
unes/compara.cpp Fun_comunes/estadist.cpp Fun_comunes/fichero.cpp Fun_comunes/Fun_Histo.cpp Fu
n_comunes/histo.cpp Fun_comunes/lbp.cpp Fun_comunes/visualiza.cpp Fun_comunes/imBBDD.cpp /usr/
lib/x86_64-linux-gnu/libtbb.so -o Hist_base_datos/output `pkg-config --cflags --libs opencv` |
 true


g++     -c -o Proc_Principal/main.o Proc_Principal/main.cpp
 Compilando Proc_principal

g++ -g -Wall -std=c++11 -pedantic -fexceptions -Wparentheses  Proc_Principal/main.cpp Fun_comu
nes/compara.cpp Fun_comunes/estadist.cpp Fun_comunes/fichero.cpp Fun_comunes/Fun_Histo.cpp Fun
_comunes/histo.cpp Fun_comunes/lbp.cpp Fun_comunes/visualiza.cpp Fun_comunes/imBBDD.cpp /usr/l
ib/x86_64-linux-gnu/libtbb.so -o Proc_Principal/output `pkg-config --cflags --libs opencv` | t
rue


sergy@sergy-K53SC:~/Escritorio/TFGvFINAL$
```

Ilustración 85 - Ejecución Make "Compile".

Habiendo ejecutado las dos instrucciones anteriores se procederá con la ejecución de estos ficheros binarios mediante el comando “Execute”.

```
sergy@sergy-K53SC:~/Escritorio/TFGvFINAL$ make execute

 Ejecutando el procesamiento
de la base de datos

gnome-terminal --command ./Proc_base_datos/output --hide-menubar -t "Procesamiento base de datos" --geometry 72x19+0+0 | true
# La opción «--command» está obsoleta y se eliminará en una versión futura de gnome-terminal.
# Use «-- » para terminar las opciones y coloque el comando que ejecutar después.
# _g_io_module_get_default: Found default implementation gvfs (GDaemonVfs) for ‘gio-vfs’
# _g_io_module_get_default: Found default implementation dconf (DConfSettingsBackend) for ‘gsettings-backend’
# watch_fast: "/org/gnome/terminal/legacy/" (establishing: 0, active: 0)
# unwatch_fast: "/org/gnome/terminal/legacy/" (active: 0, establishing: 1)
# watch_established: "/org/gnome/terminal/legacy/" (establishing: 0)



 Ejecutando obtención de histogramas

gnome-terminal --command ./Hist_base_datos/output --hide-menubar -t "Obtención Histogramas"  --geometry 72x18+0+1000 | true
# La opción «--command» está obsoleta y se eliminará en una versión futura de gnome-terminal.
# Use «-- » para terminar las opciones y coloque el comando que ejecutar después.
# _g_io_module_get_default: Found default implementation gvfs (GDaemonVfs) for ‘gio-vfs’
# _g_io_module_get_default: Found default implementation dconf (DConfSettingsBackend) for ‘gsettings-backend’
# watch_fast: "/org/gnome/terminal/legacy/" (establishing: 0, active: 0)
# unwatch_fast: "/org/gnome/terminal/legacy/" (active: 0, establishing: 1)
# watch_established: "/org/gnome/terminal/legacy/" (establishing: 0)



 Ejecutando el proceso general

gnome-terminal --command ./Proc_Principal/output --hide-menubar -t "Proceso Principal" --geometry 71x39+1000+0 | true
# La opción «--command» está obsoleta y se eliminará en una versión futura de gnome-terminal.
# Use «-- » para terminar las opciones y coloque el comando que ejecutar después.
# _g_io_module_get_default: Found default implementation gvfs (GDaemonVfs) for ‘gio-vfs’
# _g_io_module_get_default: Found default implementation dconf (DConfSettingsBackend) for ‘gsettings-backend’
# watch_fast: "/org/gnome/terminal/legacy/" (establishing: 0, active: 0)
# unwatch_fast: "/org/gnome/terminal/legacy/" (active: 0, establishing: 1)
# watch_established: "/org/gnome/terminal/legacy/" (establishing: 0)
```

Ilustración 86 - Ejecución Make "Execute".

La introducción por terminal de este comando implicará la apertura de 3 ventanas en las que estarán corriendo al unísono los procesos descritos anteriormente en el apartado 4.2.

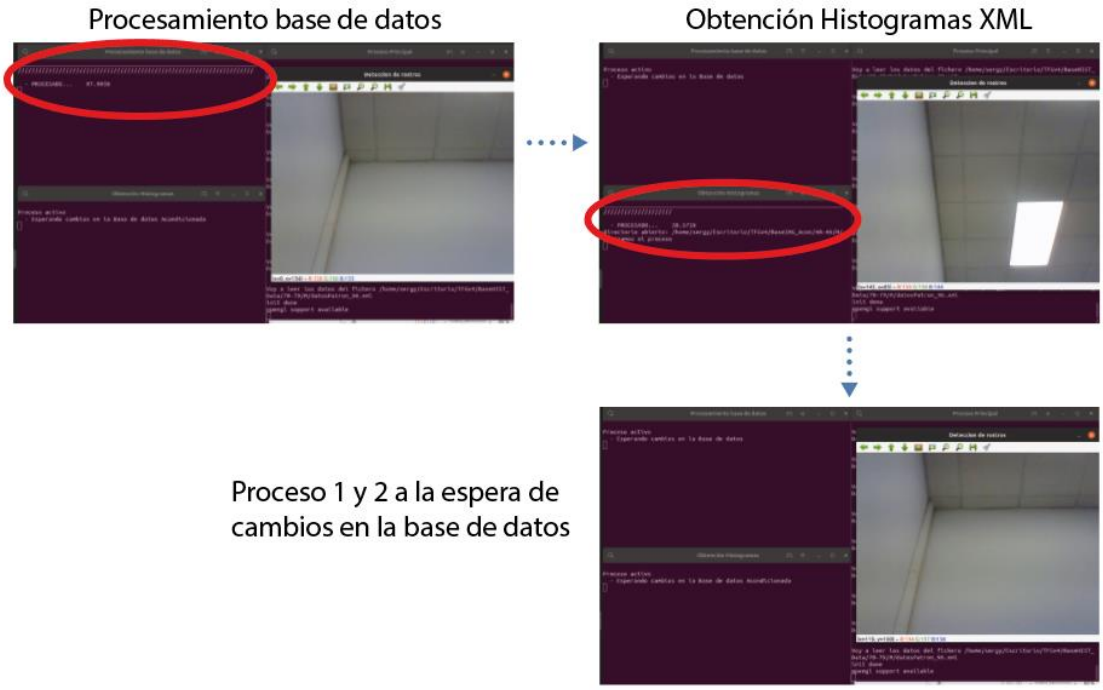


Ilustración 87 - Ejecución simultanea de los tres procesos.

La puesta en marcha de la aplicación se realiza ejecutando exclusivamente los procesos “Proc_base_datos” y “Hist_base_datos”. Estos dos procesos como se ha expuesto anteriormente, generarán las imágenes de la base de datos procesadas y sus correspondientes histogramas respecto al rango de edad. Sin estos el “Proc_principal” no puede entrar en funcionamiento. Una vez hayan finalizado ambos procesos, se podrá ejecutar la aplicación con total normalidad.

En caso de iniciar el “Proc_Principal” sin haber realizado anteriormente una puesta en marcha, este lanzará un mensaje de error en cuanto no detecte los histogramas almacenados con formato XML.

En una situación cotidiana de la aplicación, los procesos “Proc_base_datos” y “Hist_base_datos” estarán a la espera de modificaciones en la carpeta que almacena las imágenes de la base de datos. Es por ello que una ejecución continua en la que no intervengan operarios, bastaría con ejecutar solo el “Proc_principal”.

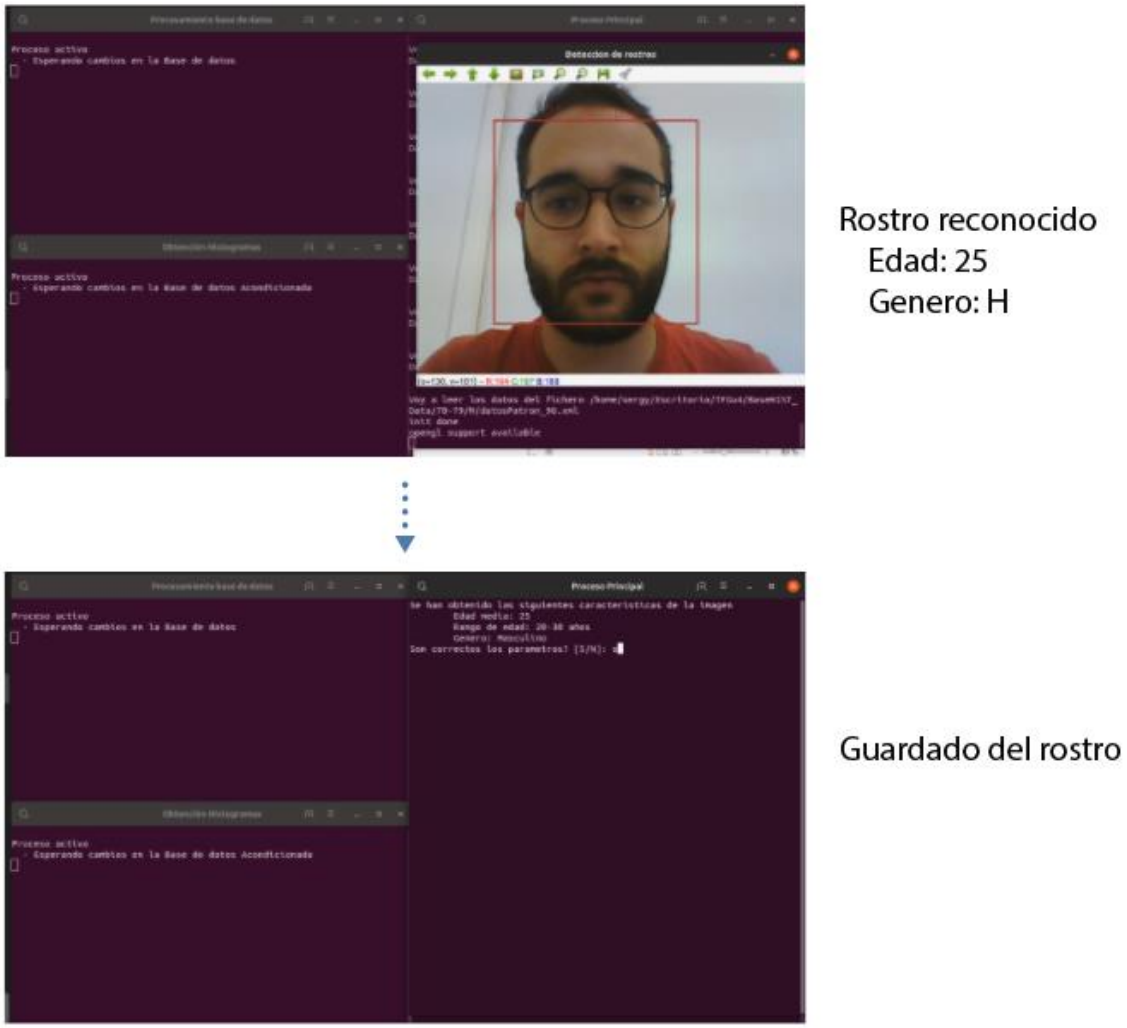


Ilustración 88 - Guardado de un rostro.

Pero uno de los objetivos marcados era conseguir una aplicación de mejora continua. Esto se logra incrementando el número de sujetos en la base de datos, por lo que se hace inevitable la presencia de un operario. En esta situación será necesario ejecutar los tres procesos para que una vez se almacene una imagen el "Proc_base_datos" comience su ejecución para añadir esa imagen recién obtenida.

CAPITULO V

5 Resultados

Como seguimiento del desarrollo de la aplicación se induce el apartado de resultados, en el que se aportarán evidencias sobre los procesos mencionados anteriormente referentes a los aciertos o fallos de los mismos.

La estructura que se sigue en este capítulo será proceso a proceso, mostrando al final varios ejemplos con diferentes sujetos, los cuales serán pixelados para seguir la LOPD (Ley Orgánica de Protección de Datos [47]).

5.1 Proc_base_datos

Anteriormente se ha descrito el funcionamiento de este proceso, en el cual se procesan las imágenes y son almacenadas de forma estructurada a través de los directorios anteriormente mencionados.

La funcionalidad que puede generar errores en este apartado es la búsqueda del rostro en las imágenes de la base de datos. El HaarCascade en ocasiones puede generar errores ya sea por la luminosidad de la imagen, por la saturación de esta misma o incluso en la aplicación de los filtros Haar.

Como punto de partida se poseen 989 imágenes con archivo TXT asociado. Tras un primer procesado con los parámetros del clasificador ya ajustados, se obtienen los siguientes resultados:

- Rostros correctos: 973
- Rostros incorrectos: 16

Estos datos nos dan un porcentaje de acierto del 98,38%.

A continuación, se muestran en la siguiente ilustración lo que se define como rostro incorrecto.

Ilustración 89 - Reconocimiento incorrecto de rostros.

5.2 Hist_base_datos

En este proceso no se ha detectado ninguna zona conflictiva en la que puedan salir fallos que repercutan en el funcionamiento general de la aplicación.

Ahora bien, sí que es cierto que las salidas del primer proceso, las cuales derivan en entradas en este, no están controladas, es decir, las imágenes expuestas en la ilustración 89 seguirán siendo procesadas. Debido a que es un porcentaje ínfimo de rostros no reconocidos correctamente, se ha decantado por exponer esto para una futura mejora, la cual será ampliada en el siguiente capítulo.

5.3 Proc_Principal

Por último, el proceso principal, como se expuesto anteriormente, será el encargado de reconocer, asignar una edad y género y guardar los rostros presentes en la imagen obtenida de la cámara.

El primer subproceso que se realiza es la detección del rostro. Esto al ser en tiempo real se corrige de forma automática en caso de que en un frame haya habido algún error, por lo que en ningún momento se dará la casuística explicada en el apartado 5.1.

Dicho reconocimiento se ha enfocado desde un primer momento a fotos frontales, pero se ha visto que rostros ladeados también son obtenidos. En la siguiente ilustración se pueden ver diferentes casos.

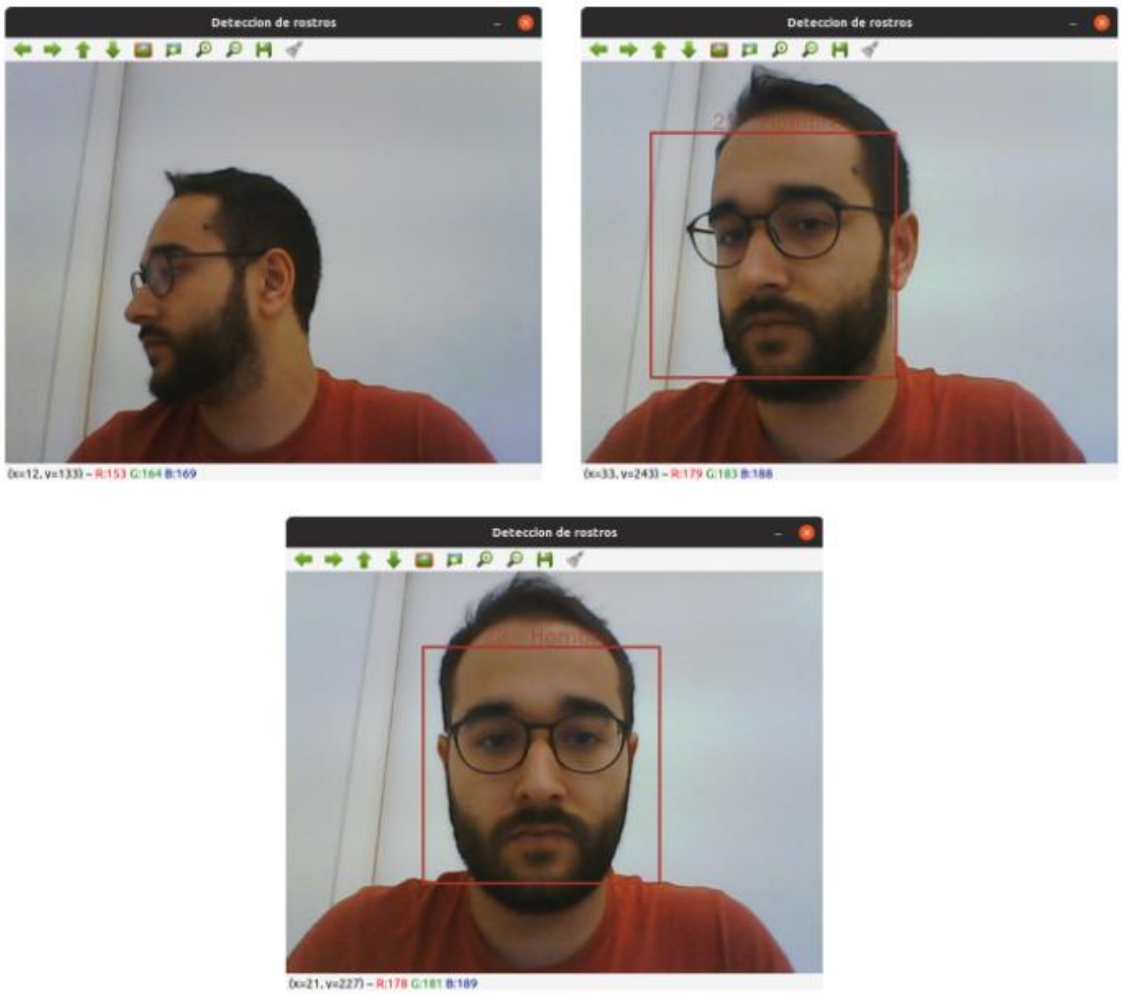


Ilustración 90 - Detección de rostros rotados.

A mayores y relacionado con el reconocimiento de rostros, se marcó como objetivo reconocer todos los presentes en un mismo frame. Por las limitaciones de la cámara, es decir los píxeles que posee y por el gran angular, se han realizado pruebas con 2 sujetos presentes en la imagen de forma simultánea.

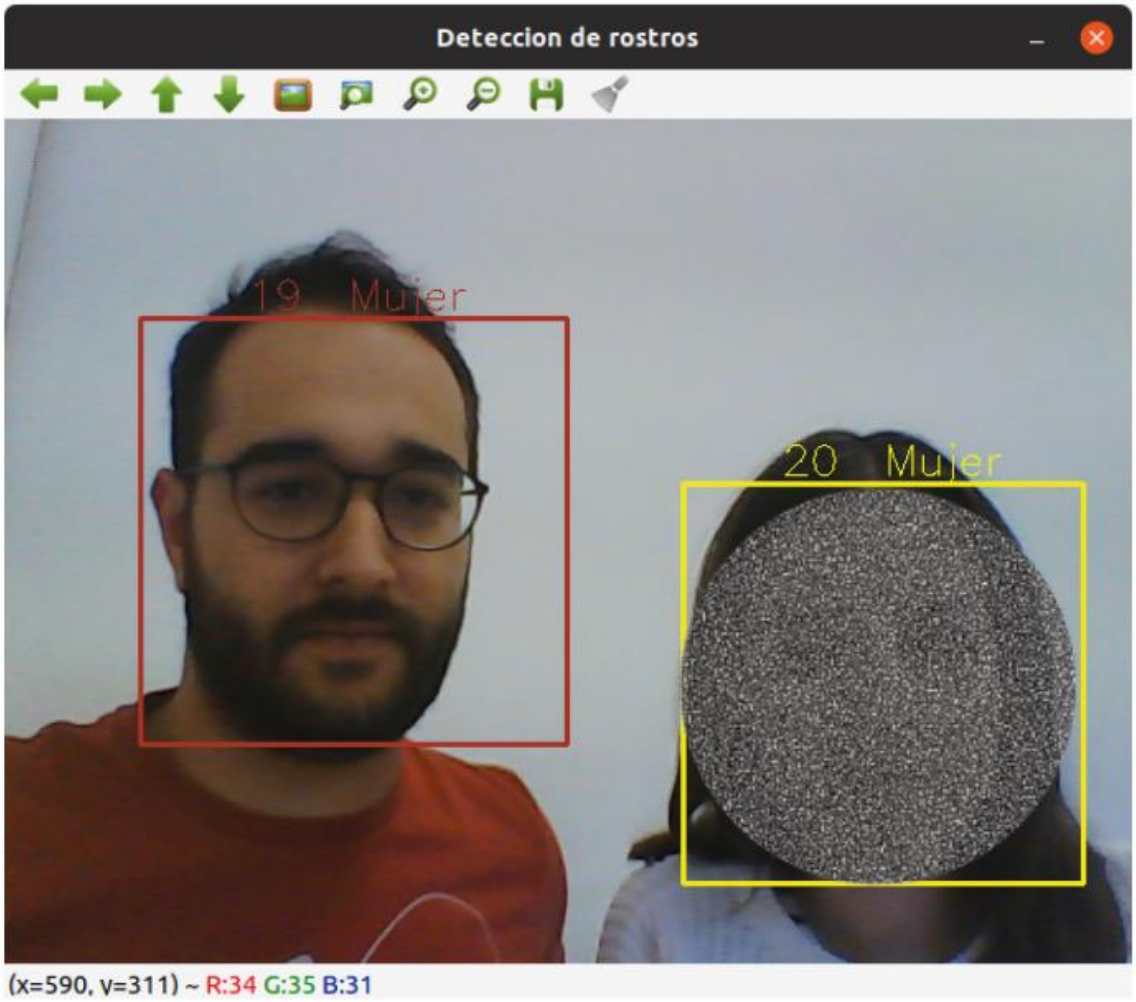


Ilustración 91 - Reconocimiento de 2 rostros.

Como se observa en la ilustración 91 se verifica esta casuística. En lo referente a asignar la edad a dos personas se observan fallos. Después de la realización de varias pruebas se comprueba que es debido a la distancia del rostro respecto a la cámara, ya que cuanto más alejado menor resolución poseerá el recorte que se obtenga y con ello el reescalado tendrá que interpretar una mayor cantidad de píxeles. Por ello se achaca este fallo a la cámara y no al software.

Esta casuística se presenta igualmente cuando se foca a una sola persona. En la siguiente ilustración podemos ver una batería de imágenes en las que se presenta este problema.

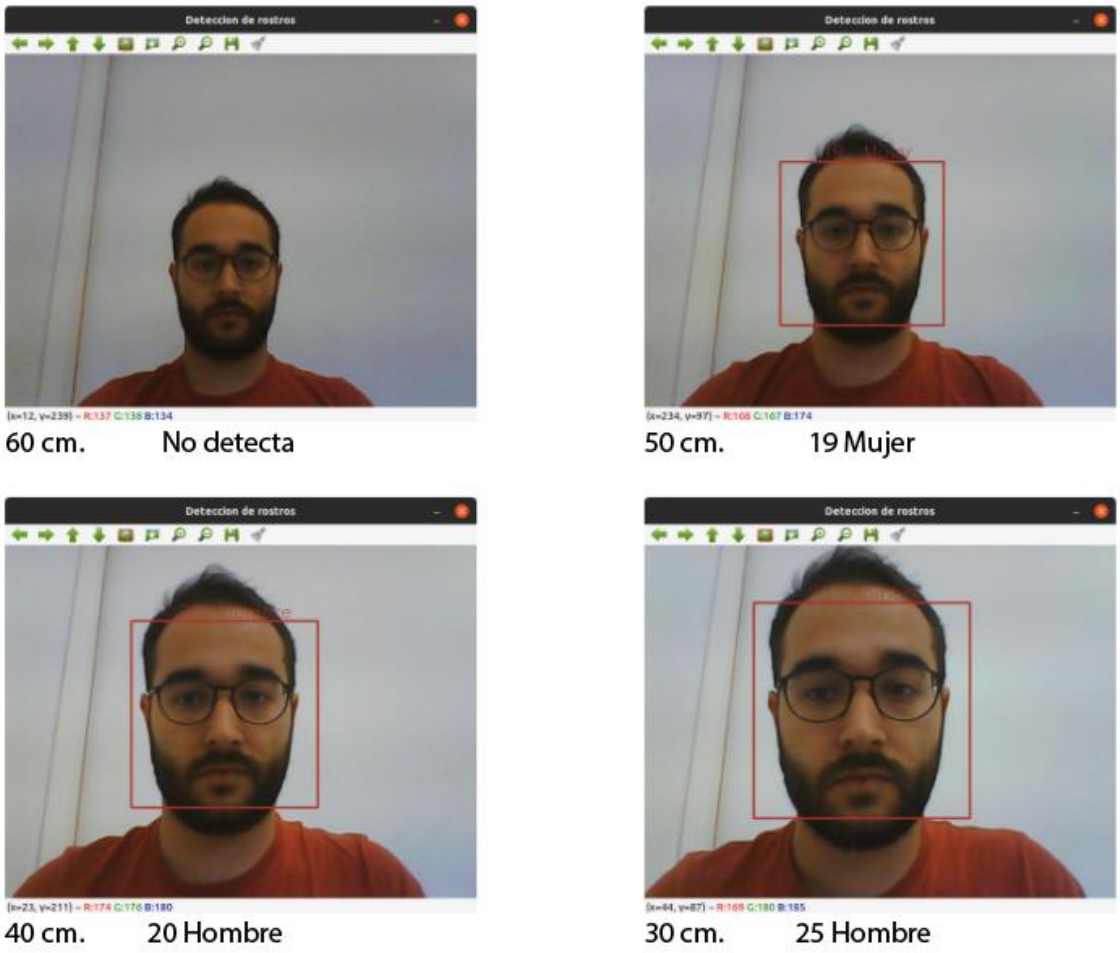


Ilustración 92 - Detección con diferentes distancias.

Como se puede observar en la ilustración 92 a menor distancia respecto la cámara, mayor es la tasa de aciertos de la aplicación. Todo ello deriva de la calidad del hardware y no de fallos propiciados en el software desarrollado.

CAPITULO VI


Escuela de Ingenierías Industriales
Conclusiones y mejoras

6 Conclusiones y mejoras

6.1 Conclusiones

La realización del presente proyecto buscaba la creación de una aplicación basada en la visión artificial, que permitiese a los usuarios de la misma conocer la edad y el género de los sujetos presentes en una imagen. Todo ello a través del uso de algoritmos de textura. Para lograrlo se marcaron una serie de objetivos descritos en el apartado 1.2.

El primer objetivo fue crear dicha aplicación bajo un estándar multiplataforma, el cual se ha logrado utilizando herramientas comunes a una multitud de hardware. Véase el empleo de *Linux*© como el lenguaje C++ o el sistema operativo de desarrollo, el cual permite extrapolar sin grandes inconvenientes a diferentes plataformas, como *Raspberry Pi*© o servidores basados en *Linux*©.

Ligado a ese objetivo se marcó emplear exclusivamente herramientas software de libre distribución, como *Code::Blocks*©, el cual ha sido el IDE empleado para desarrollar la aplicación; las librerías *OpenCV*©, clasificadores HaarCascade y la base de datos *FERET*©, la cual es propiedad del gobierno estadounidense, pero permite su uso en proyectos de investigación.

Asimismo, se propuso que la aplicación pudiese ser ejecutada de forma continua en tiempo real y que reconociese más de un solo rostro. Como se ha podido observar a lo largo de la presente memoria, estos objetivos se cumplen excepto el reconocimiento de varias caras, ya que sí, reconoce los rostros, pero el algoritmo de extracción de características falla debido al hardware que se ha empleado para realizar las pruebas.

Por último, se propuso incluir en la aplicación un módulo que funcionase como entrenamiento supervisado, incorporando las imágenes que van llegando, hecho que se ha logrado mediante el uso de un operario, el cual decidirá el momento en el que el proceso de guardado debe ejecutarse.

Todos estos objetivos han servido para configurar la aplicación anteriormente descrita en el capítulo IV, la cual funciona correctamente.

En el capítulo anterior se explica cómo se procesa la base de datos con un 98,38% de aciertos. Los fallos que se dan se deben a las condiciones de luminosidad o saturación en la imagen por regla general. En cuanto al proceso principal, es decir el encargado de determinar la edad y el género de los sujetos, posee un alto índice de aciertos a una distancia de 30cm respecto la cámara. Para distancias mayores, la precisión de los aciertos es menor, debido a la calidad de la cámara que se ha empleado. A mayores se ha observado como el software es capaz de reconocer rostros ladeados, al igual que reconocer y asignar edad y género a varias personas presentes ante la cámara.

6.2 Líneas futuras

A continuación, se proponen unas líneas de cara a futuras mejoras de la aplicación. Los puntos que se exponen están enfocados en la búsqueda de una mayor autonomía de la aplicación, una implementación real y mejoras funcionales.

En lo referente al hardware, se ha mostrado anteriormente en el capítulo V que los resultados de la cámara son fiables con distancias menores a 30cm. Es por ello que la primera mejora propuesta es realizar pruebas con cámaras de mayor resolución de las cuales se puedan obtener imágenes con mayor calidad, es decir, con una matriz de píxeles mayor. De esta forma a la hora de realizar el reescalado, no se perderá ningún tipo de información.

Aunado a este cambio, habría que buscar procesadores a medida, ya que es inevitable que al subir la resolución de la imagen se necesite mayor capacidad de procesado para la misma y más cuando la aplicación desarrollada tiene por objetivo una ejecución en tiempo real.

Un ejemplo económico y práctico para comenzar las pruebas de la aplicación sería una *Raspberry Pi©*, ya que es una placa computadora basada en *Linux©* y que posee cámara propia. Todo ello favorece la implementación de esta aplicación en dicho hardware.

Por otra parte, en lo referente al software y pensando en el uso de esta aplicación por parte de usuarios finales, se propone implementar una interfaz gráfica con *QT©* [48], para facilitar a los operarios de esta su uso y entendimiento.

La base de datos *FERET©*, como se ha comentado en el apartado 3.3, exclusivamente posee sujetos con edades de 10 en 10 años, es decir, 10,20,30… Por ello se debería trabajar en aumentar esta base de datos para poder obtener cifras más exactas, las cuales no dependan de una discretización respecto al tiempo.

Respecto a las mejoras del propio código de la aplicación. A muchas variables se les ha asignado un valor determinado, fruto de las pruebas realizadas. Sería conveniente intentar configurar estos valores en función de la situación en la que se vaya a implementar la aplicación, ya sea a través de un estudio autónomo que realice la aplicación o de una marca que establezca el operario a la hora de instalar dicha aplicación.

Otro factor que afecta al reconocimiento de edad y género es el fondo que se vislumbra al hacer el recorte facial, ya que en muchas ocasiones quedan restos, siendo estos de diferentes tonalidades afectando con ello al histograma. Es por ello que se propone como mejora una extracción del fondo quedando exclusivamente los rostros presentes ante la cámara.

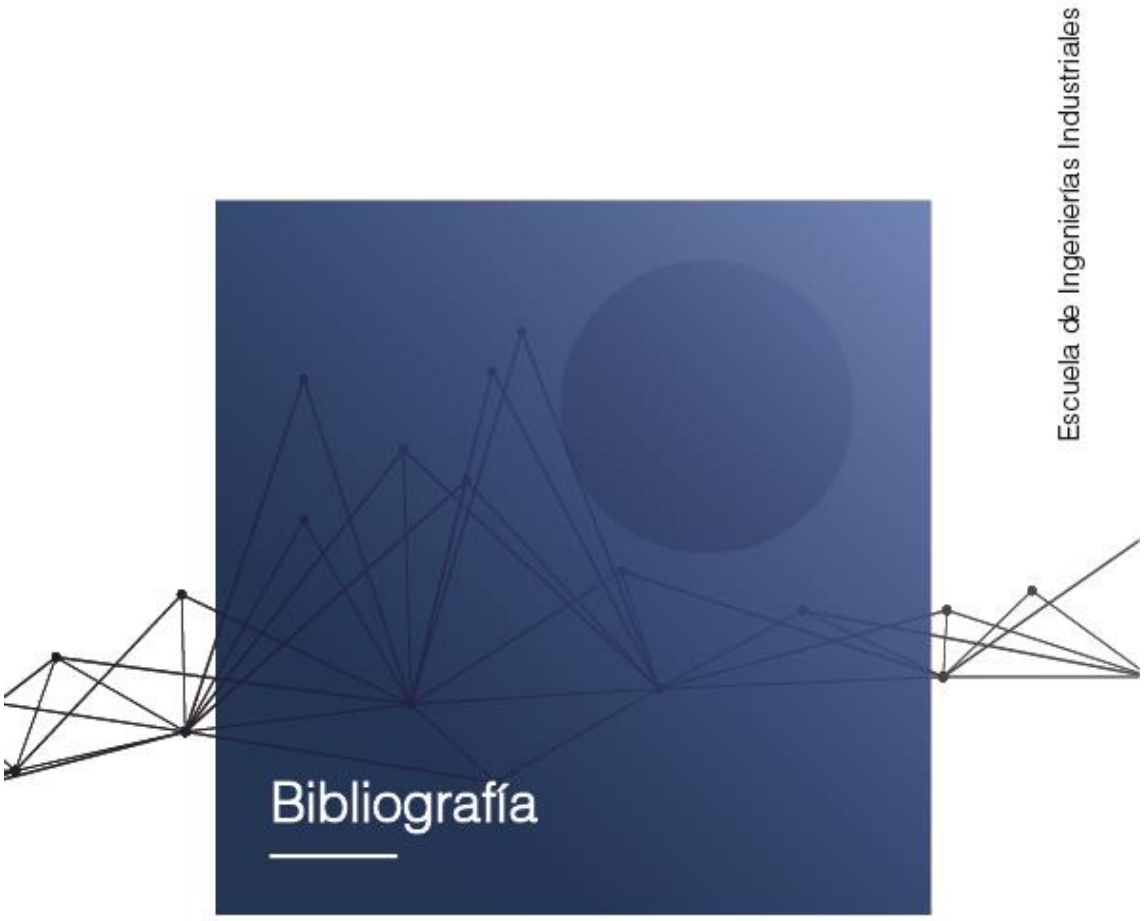
Siguiendo esta línea se podría mejorar el resultado corrigiendo la luminosidad presente en la imagen. Esto se ha realizado en la presente memoria, pero cuando se han realizado pruebas con luz natural directa, el ecualizar el histograma no es suficiente. Por ello habría que emplear un algoritmo que corrija la luminosidad de la

imagen en función del lugar en el que se haya implementado la aplicación y de las horas del día en caso de ser en una zona exterior.

Siguiendo la mejora de código se podría implementar una mejora que consistiese en rescindir la ejecución de los tres procesos asíncronamente, en pro de ejecutarlos cuando sea necesario. Este nuevo desarrollo consistiría en que cada vez que el operario decida guardar los rostros presentes en la imagen que está observando, comenzase la ejecución del primer proceso “Proc_base_datos” en vez de estar este en ejecución esperando cambios en la base de datos. Se debería implementar la misma casuística en el segundo proceso, consiguiendo con todo ello la eliminación de las memorias compartidas.

Por último, se podría ahondar en la posibilidad de eliminar la presencia del operador humano a la hora de guardar los rostros. De esta forma la aplicación sería completamente autónoma y de aprendizaje continuo.

CAPITULO VII



7 Bibliografía

1. Usar Face ID en el iPhone o iPad Pro - Soporte técnico de Apple [Internet]. [Accedido 24 de junio de 2019]. Disponible en: https://support.apple.com/es-es/HT208109
2. CONTAVAL [Internet]. [Accedido 3 de abril de 2019]. Disponible en: https://contaval.es/que-es-la-vision-artificial-y-para-que-sirve/
3. Mery D. Visión por Computador [Internet]. 2004 [Accedido 3 de abril de 2019]. Disponible en: http://www.ing.puc.cl/
4. Carlos Platero Dueñas. Introducción a la Visión [Internet]. 2015 [Accedido 4 de abril de 2019]. Disponible en: http://www.elai.upm.es/webantigua/spain/Asignaturas/MIP_VisionArtificial/ApuntesVA/cap1IntroVA.pdf
5. Paredes JC. Teoría y Aplicación de la Informática 2 Visión Artificial [Internet]. [Accedido 21 de febrero de 2019]. Disponible en: http://jeuazarru.com/wp-content/uploads/2015/11/vision_artificial.pdf
6. Mery D, Pedreschi F. Segmentation of colour food images using a robust algorithm. [Accedido 24 de junio de 2019]; Disponible en: www.ing.puc.cl/~dmery.
7. Vélez Serrano JF, Moreno Díaz AB, Sanchez Calle Á, Esteban Sánchez-Maríb JL. VisionPorComputador.
8. Estadísticas resolución de pantallas más utilizadas en la web « Los Tiempos Cambian [Internet]. [Accedido 24 de junio de 2019]. Disponible en: http://www.lostiemposcambian.com/blog/usabilidad/estadisticas-resolucion-de-pantallas-mas-utilizadas-en-la-web/
9. Procesamiento de imágenes (Traslación, Escala, Rotación, Inclinación) - Monografias.com [Internet]. [Accedido 24 de junio de 2019]. Disponible en: https://www.monografias.com/trabajos108/procesamiento-imagenes-traslacion-escala-rotacion-inclinacion/procesamiento-imagenes-traslacion-escala-rotacion-inclinacion.shtml
10. Understanding image histograms with OpenCV [Internet]. [Accedido 24 de junio de 2019]. Disponible en: https://lmcaraig.com/image-histograms-histograms-equalization-and-histograms-comparison/
11. Woods G. Introducción a las imágenes digitales Segunda parte [Internet]. [Accedido 6 de abril de 2019]. Disponible en: http://alojamientos.us.es/gtocoma/pid/tema1-2.pdf
12. Woods G. Tema 5: Morfología Primera parte [Internet]. [Accedido 7 de abril de 2019]. Disponible en: http://alojamientos.us.es/gtocoma/pid/tema5-1.pdf
13. GERMÁN AGUILAR CARRERA G. PROCESAMIENTO DIGITAL DE IMÁGENES UTILIZANDO FILTROS MORFOLÓGICO [Internet]. [Accedido 7 de abril de 2019]. Disponible en: http://bibdigital.epn.edu.ec/bitstream/15000/5249/1/T171.pdf

14. Canny Edge Detection in Python with OpenCV | henrydangprg [Internet]. [Accedido 28 de junio de 2019]. Disponible en: https://henrydangprg.com/2016/12/11/canny-edge-detection-in-python-with-opencv/

15. c++ - Fill the holes in OpenCV - Stack Overflow [Internet]. [Accedido 24 de junio de 2019]. Disponible en: https://stackoverflow.com/questions/1716274/fill-the-holes-in-opencv

16. Chapter 4: Image Processing in OpenCV - Programmer Sought [Internet]. [Accedido 25 de junio de 2019]. Disponible en: http://www.programmersought.com/article/689566561/;jsessionid=F32380C8483D94A569D8B481931DE08D

17. Alhamzawi HAM. Faces and eyes Detection in Digital Images Using Cascade Classifiers. Comput Eng Appl J. 2018;7(1):57-66.

18. Davis M, Popov S, Surlea C. Real-Time Face Recognition from Surveillance Video. En 2011 [Accedido 25 de junio de 2019]. p. 155-94. Disponible en: http://link.springer.com/10.1007/978-3-642-17554-1_8

19. Geometría Epipolar [Internet]. [Accedido 9 de abril de 2019]. Disponible en: https://unipython.com/geometria-epipolar/

20. Amazon: Ein Lager mit künstlicher Intelligenz | FINK.HAMBURG [Internet]. [Accedido 25 de junio de 2019]. Disponible en: https://fink.hamburg/2017/05/amazon-erstes-teilautomatisiertes-lager-geplant/

21. Generalitat de Catalunya. Aplicación práctica de la visión artificial en el control de procesos industriales [Internet]. [Accedido 10 de abril de 2019]. Disponible en: http://visionartificial.fpcat.cat/wp-content/uploads/UD_1_didac_Conceptos_previos.pdf

22. Soluciones para Estacionamientos | Pemica S.R.L [Internet]. [Accedido 25 de junio de 2019]. Disponible en: https://www.pemica.com.do/soluciones-para-estacionamientos/

23. NEH Invites Proposals that Respond to Historical and Multilingual OCR Report | National Endowment for the Humanities (NEH) [Internet]. [Accedido 25 de junio de 2019]. Disponible en: https://www.neh.gov/blog/neh-invites-proposals-respond-historical-and-multilingual-ocr-report

24. INDRA. SISTEMAS DE TELEDETECCIÓN [Internet]. [Accedido 11 de abril de 2019]. Disponible en: https://www.indracompany.com/sites/default/files/teledeteccion.pdf

25. Reconocimiento facial [Internet]. [Accedido 11 de abril de 2019]. Disponible en: https://www.serbanbiometrics.es/es/soluciones/reconocimiento-facial

26. Top 10 Facial Recognition APIs (Updated for 2018) | RapidAPI - Liberal Dictionary [Internet]. [Accedido 25 de junio de 2019]. Disponible en: http://www.liberaldictionary.com/face-recognition/top-10-facial-recognition-apis-updated-for-2018-rapidapi/

27. Customers - Face++ Cognitive Services [Internet]. [Accedido 25 de abril de

2019]. Disponible en: https://www.faceplusplus.com/customers/

28. Missing & Vulnerable ChildrenTrackChild 2.0 | National Tracking System for [Internet]. [Accedido 24 de abril de 2019]. Disponible en: https://trackthemissingchild.gov.in/trackchild/index.php

29. Press E. El aeropuerto de Orlando, el primero de EEUU en usar reconocimiento facial en todos sus vuelos internacionales. [Accedido 24 de abril de 2019]; Disponible en: https://www.europapress.es/portaltic/sector/noticia-aeropuerto-orlando-primero-eeuu-usar-reconocimiento-facial-todos-vuelos-internacionales-20180622132420.html

30. ViaOpta Hello - Aplicaciones en Google Play [Internet]. [Accedido 24 de abril de 2019]. Disponible en: https://play.google.com/store/apps/details?id=com.novartis.pharma.global.viaoptahello.ext

31. Control de acceso facial en hospitales y farmacéuticas | Kimaldi [Internet]. [Accedido 25 de abril de 2019]. Disponible en: https://www.kimaldi.com/aplicaciones/reconocimiento_facial/reconocimiento_facial_3d_para_control_de_acceso_y_presencia_en_hospitales_y_farmaceuticas/

32. Xiaomi crea aplicación que adivina edad y género de usuarios a través de una selfie [Internet]. [Accedido 25 de abril de 2019]. Disponible en: https://ohmygeek.net/2015/04/07/xiaomi-app-edad-genero-selfie/

33. La revolución del reconocimiento facial: software que identifica edad, cansancio, emociones... - Observatorio IA [Internet]. [Accedido 25 de abril de 2019]. Disponible en: http://observatorio-ia.com/nuevo-software-de-reconocimiento-facia

34. Documentation - Point Cloud Library (PCL) [Internet]. [Accedido 15 de abril de 2019]. Disponible en: http://www.pointclouds.org/documentation/

35. Cazorla M, Colomina O, Compañ P, Escolano F, Zamora JL. JavaVis: Una librería para visión artificial en Java [Internet]. [Accedido 1 de marzo de 2019]. Disponible en: https://rua.ua.es/dspace/bitstream/10045/12363/6/Jenui01-1.pdf

36. Visión Artificial - MATLAB y Simulink [Internet]. [Accedido 15 de abril de 2019]. Disponible en: https://es.mathworks.com/discovery/computer-vision.html

37. OpenCV library [Internet]. [Accedido 16 de abril de 2019]. Disponible en: https://opencv.org/

38. OpenCV: Introduction [Internet]. [Accedido 16 de abril de 2019]. Disponible en: https://docs.opencv.org/master/d1/dfb/intro.html

39. Color FERET Database - Data.gov [Internet]. [Accedido 17 de abril de 2019]. Disponible en: https://catalog.data.gov/dataset/color-feret-database

40. Face Recognition Technology (FERET) | NIST [Internet]. [Accedido 17 de abril de 2019]. Disponible en: https://www.nist.gov/programs-projects/face-recognition-technology-feret

41. Phillips PJ, Rauss PJ, Der SZ. FERET (Face Recognition Technology) Recognition

Algorithm Development and Test Results [Internet]. 1996 [Accedido 17 de abril de 2019]. Disponible en: https://www.nist.gov/sites/default/files/documents/2016/12/15/feret3.pdf

42. Battiato S, Gallo G, Schettini R, Stanco F. Image analysis and processing -- ICIAP 2017 : 19th International Conference, Catania, Italy, September 11-15, 2017, Proceedings. Part II [Internet]. [Accedido 30 de abril de 2019]. 795 p. Disponible en: https://books.google.es/books?id=Iec5DwAAQBAJ&pg=PA206&lpg=PA206&dq=ulbp+algorithm&source=bl&ots=hMs2Olsp6Z&sig=ACfU3U34xd0BQVDED9VE6CCuEofvLG1Dyw&hl=es&sa=X&ved=2ahUKEwjOlaefr_jhAhWJ4YUKHb1nDI8Q6AEwC3oECAkQAQ#v=onepage&q=ulbp algorithm&f=false

43. Geometric Image Transformations — OpenCV 2.4.13.7 documentation [Internet]. [Accedido 13 de mayo de 2019]. Disponible en: https://docs.opencv.org/2.4/modules/imgproc/doc/geometric_transformations.html

44. Cascade Classification — OpenCV 2.4.13.7 documentation [Internet]. [Accedido 13 de mayo de 2019]. Disponible en: https://docs.opencv.org/2.4/modules/objdetect/doc/cascade_classification.html

45. HAAR_DETECTION_TYPE Enumeration [Internet]. [Accedido 13 de mayo de 2019]. Disponible en: http://www.emgu.com/wiki/files/1.5.0.0/Help/html/e2278977-87ea-8fa9-b78e-0e52cfe6406a.htm

46. Histograms — OpenCV 2.4.13.7 documentation [Internet]. [Accedido 19 de mayo de 2019]. Disponible en: https://docs.opencv.org/2.4/modules/imgproc/doc/histograms.html?highlight=comparehist#comparehist

47. Agencia Estatal Boletín Oficial del Estado. Ley Orgánica 15/1999, de 13 de diciembre, de Protección de Datos de Carácter Personal. [Internet]. [Accedido 10 de junio de 2019]. Disponible en: https://www.boe.es/buscar/act.php?id=BOE-A-1999-23750

48. Qt | Cross-platform software development for embedded [Internet]. [Accedido 24 de junio de 2019]. Disponible en: https://www.qt.io/