1. Introducción

Una de la habilidades humanas más notables es el reconocimiento facial. Se desarrolla a lo largo de la infancia y es muy importante para muchos aspectos de nuestra vida social, y junto a otras habilidades como el reconocimiento del estado emocional de la gente con la que interectuamos, ha jugado un papel importante en la evolución. Por todo esto, los humanos somos muy buenos a la hora de reconocer caras y formas complejas. Ni el paso del tiempo hace mella en esta capacidad. Sin duda es una capacidad muy potente, y por lo tanto sería muy interesante que los ordenadores fueran capaces de reconocer caras tan bien como los humanos.

Un sistema de reconocimiento facial podría ayudarnos de muchas formas, como sistemas de vigilancia y control de criminales, sistemas de acceso a un recinto privado, encontrar a gente perdida usando grabaciones de video de los lugares públicos…

Este proyecto trata de desarrollar un sistema de reconocimiento facial con imágenes estáticas.

Hay dos técnicas tradicionales de reconocimiento facial usando imagenes tomadas de frente y con iluminación constante. La primera es el *reconocimiento de características geométricas* de la imagén y la segunda sería la *contraposición de imágenes (template matching)* para hallar cuánto se asemejan. En nuestro proyecto usamos la primera de las técnicas.

1.1. Motivación

El proyecto fue concebido como un reto hacia el estudiante y no como un intento de desarrollo óptimo ya que no era permitido el uso de las herramientas óptimas para este desarrollo. Existe un programa para reconocimento de formas llamado **KS 400** y el estudiante debía obtener un sistema de **reconocimento facial** usando dicho programa. Es conocido que hay métodos mucho mejores para el desarrollo de este tipo de funciones, como por ejemplo Matlab, pero el proyecto no consistía en eso sino en el aprovechamiento de dicho programa.

1.2. Objetivos

- Contrucción de una base de datos con imágenes faciales.
- Análisis de las imágenes y reconocimiento de características.
- Tratar de reconocer a personas de la base de datos con esas imágenes.
- Análisis de los resultados.

1.3. Procedimiento

El proyecto tiene cuatro partes diferentes:

1.3.1. Construcción de la base de datos.

La base de datos fue construída a partir de imagines correspondientes a la gente del laboratorio y estudiantes de la FH München. Cerca de veinte personas fueron fotografiadas con una cámara instalada en el laboratorio.

1.3.2. Análisis facial.

Antes de comenzar con el análisis facial, entrenamos con otras formas. Estas formas eran semillas, a las que podíamos clasificar según su tipo. Cuando nos sentimos suficientemente preparados para un análisis más difícil comenzamos con el análisis facial. Debíamos buscar ciertas características geométricas de la cara y encontramos la boca y los ojos como nuestras referencias principales.

1.3.3. Reconociendo caras.

Después de que las caras de la base de datos fueran analizadas y medidas nos quedaba la clasificación. Insertamos todas las características de las caras en tablas y tras un pequeño procesado de estos datos creamos nuevas imágenes. De estas imágenes más sencillas sí puede aprender nuestro programa. Ya cuando nuestro programa ha aprendido a diferenciar las imágenes de la base de datos sólo queda insertar una para que la clasifique. Más tarde explicaremos el porqué de este extraño proceder.

1.3.4. Análisis de los resultados.

El programa se cambió innumerables veces y se hicieron innumerables pruebas, dada las múltiples posibilidades que teniamos, para ver cuál era la manera óptima de hacer la clasificación. Al final solo los resultados de la forma que parecía la óptima son expuestos.

2. Algoritmo

2.1. El sistema CZ KS 400

El sistema de análisis **CZ KS 400** está diseñado para medir y analizar imágenes estáticas. Ofrece una amplia gama de funciones con las que se pueden realizar un completo procesado de la imagen. Además ofrece funciones de bases de datos y de toma de imágenes. El programa ofrece un modo de programación tipo macro, donde estas instrucciones son ejecutadas una a una y en orden descendente. En este “pseudo-lenguaje” también es posible realizar bucles, y para ello posee funciones como “if” o “while”.

2.2. Nuestro trabajo.

Trabajamos sobre este sistema, el **CZ KS400.** Está instalado en nuestro laboratorio en la FH München. Allí está la versión KS400 3.0 de dicho sistema. El programa necesita una llave hardware para funcionar por lo que no es posible sacarlo del laboratorio. Tampoco se encontraron versiones abiertas para el uso fuera de la FH München por lo que sólo era accesible en el laboratorio.

Tal como ya hemos dicho, nuestro objetivo era realizar un sistema de reconocimiento facial usando el KS400. Realizamos dos versiones principales de nuestro sistema:

- Clasificación con imágenes virtuales.
- Clasificación con imágenes virtuales y cuadrícula de clasificación.

Los códigos están escritos como macros, y estos deben ser interpretados por el KS400 y no por otro programa. Así que para hacer correr nuestros clasificadores nos hace falta el KS400.

Además encontramos muchos problemas y fallos dentro de este sistema, los cuales iremos explicando en las siguientes páginas.

-Estructura del código

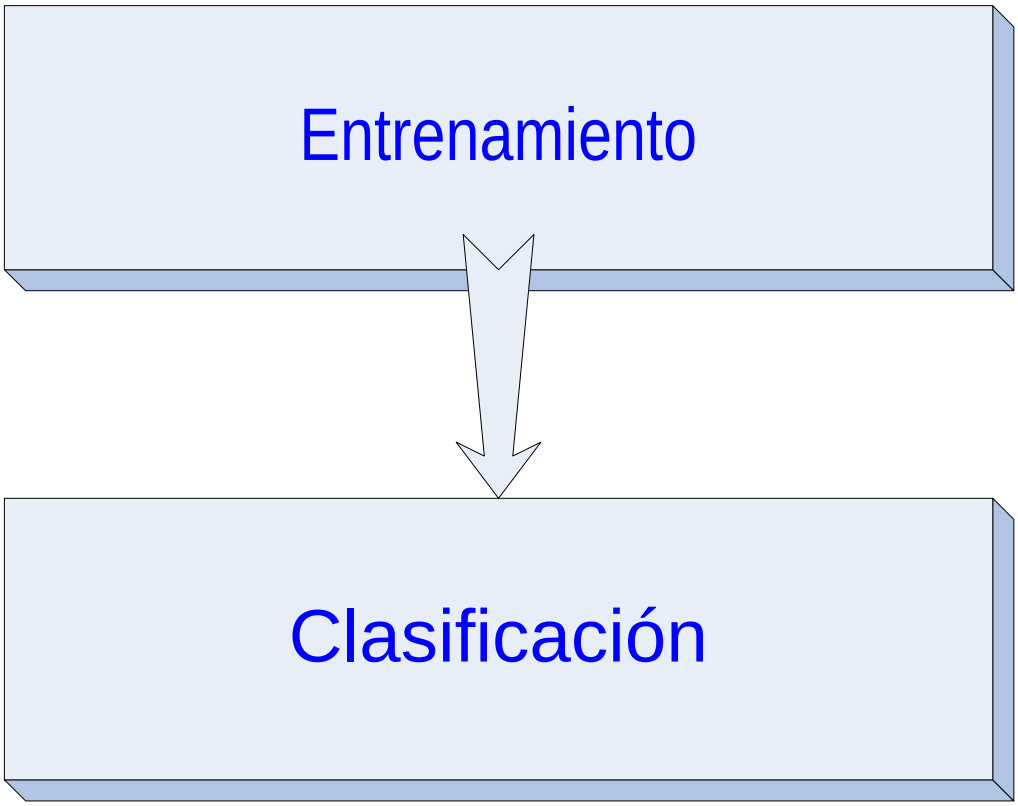
El código tiene dos partes bien diferenciadas. Éstas son:

- El entrenamiento
- La clasificación

En el entrenamiento las imágenes de la base de datos son cargadas, luego analizadas y finalmente sus carácterísticas son guardadas en un clasificador.

En la clasificación tratamos una imagen de igual manera que antes y luego aplicamos el clasificador resultado del entrenamiento para obtener la clasificación.

Fig 1. Funcionamiento externo del programa



Esta es la manera de funcionamiento a grandes rasgos.

Estos dos procesos son muy similares, ya que en realidad consisten en lo mismo excepto que uno trata todas la imágenes de la base de datos y el otro sólo a una.

Al ser muy similares están divididos en las mismas partes. Éstas son las cuatro que a continuación se describen:

- Medida de las imágenes de la base de datos.
- Construcción de la base de datos “FACE”
- Transformación y normalización de la base de datos “FACE”
- Creación de imágenes virtuales y según parte aprendizaje del clasificador o clasificación.

Es un esquema muy sencillo. Cogemos los objetos, medimos, guardamos los resultados y luego entrenamos al clasificador. De las partes que tenemos sólo la que corresponde a la creación de imágenes virtuales puede parecer un poco extraña. Realizamos este paso por culpa de un “fallo” del programa y no porque nos pareciera necesario.

El primero de los pasos “Medida de las imágenes de la base de datos” consiste exactamente en eso. Cargamos las imágenes de la base de datos y medimos carácterísticas de ellas (como los ojos y la boca). Para realizarlo necesitamos filtros de color y otros tipos de filtros. Al final todos los datos son grabados en las bases de datos llamadas ‘Testmouth’ y ‘Testeyes’.

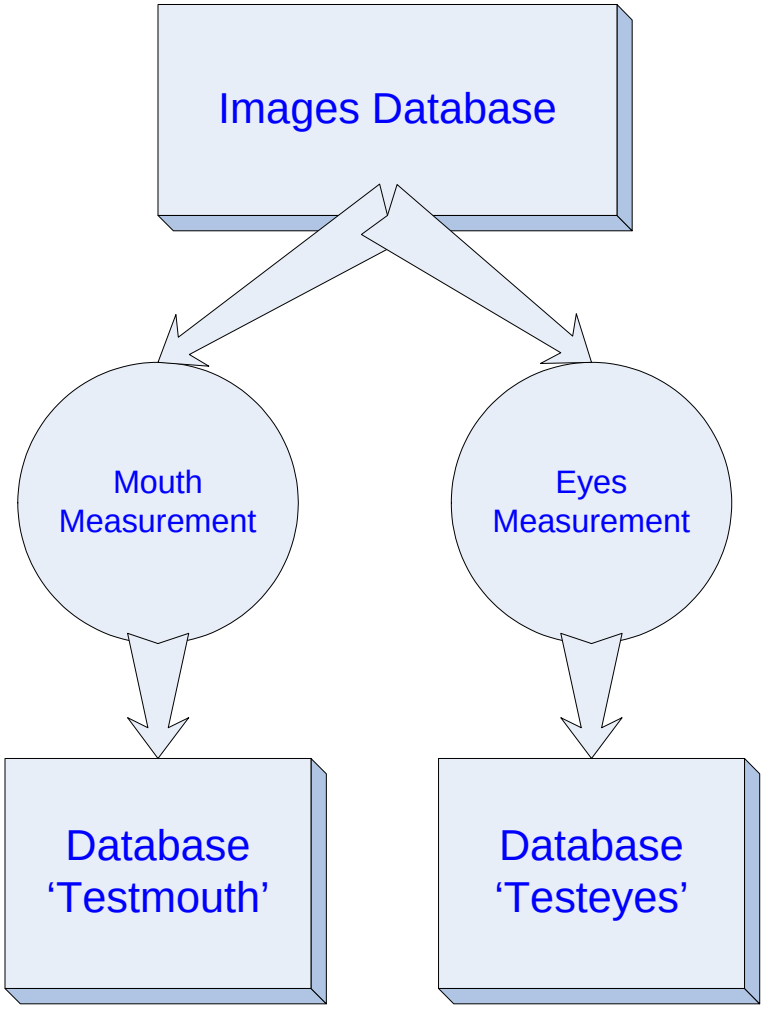


Fig 2. Aquí vemos el esquema de lo antes expuesto.

Hemos medido, en nuestro caso, la boca y los ojos. Se podrían haber medido más cosas pero debido a la lentitud del programa y a otras causas como el modo de clasificación que debimos utilizar, nos conformamos con estos dos elementos.

Primero debemos definir varios parámetros del clasificador, éstos son:

- Clases, para realizar la clasificación.
- Características de la región a medir.
- Condiciones sobre estas características.
- Tipo de clasificación.
- Bases de datos.

Luego empezamos la medición de las características. Comenzamos por la boca:

- Medición de la boca

En la boca medimos las siguientes características:

- Largo
- Ancho
- Posición de la boca

A continuación exponemos el proceso a grandes rasgos:

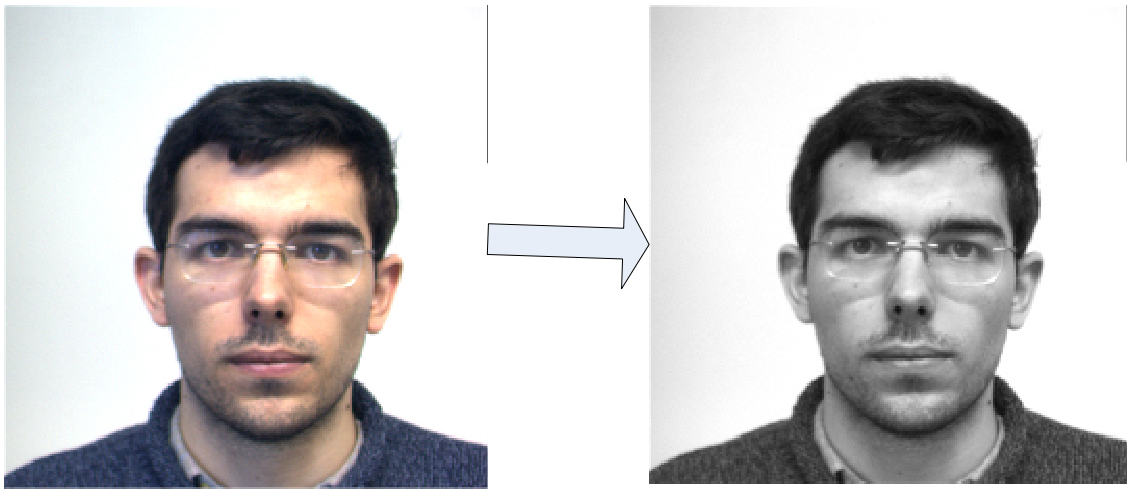


Fig. 3 Paso de color a escala de grises

Primero obtenemos la imagen es escala de grises y a continuación hacemos una selección de las partes más interesantes para medir.

El filtro transforma imágenes con formato RGB a formato escala de grises con valores de 0 a 255 (de negro a blanco). Como se puede observar, nos quedan la boca y los ojos muy claros y fáciles de seleccionar.

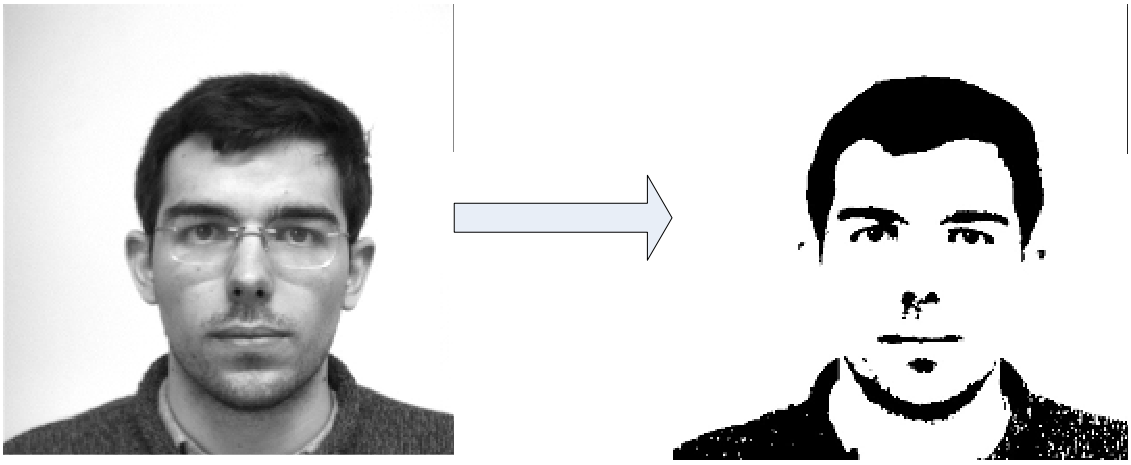


Fig 15. Aplicación de un filtro de escala de grises.

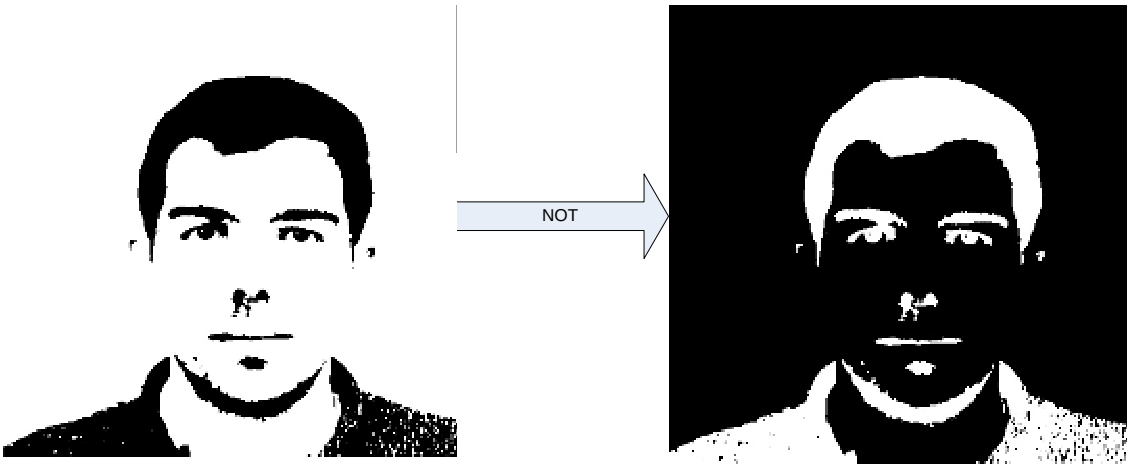


Fig.5 Paso al negativo de la imagen.

Luego necesitamos pasar la foto a negativo para su medición.

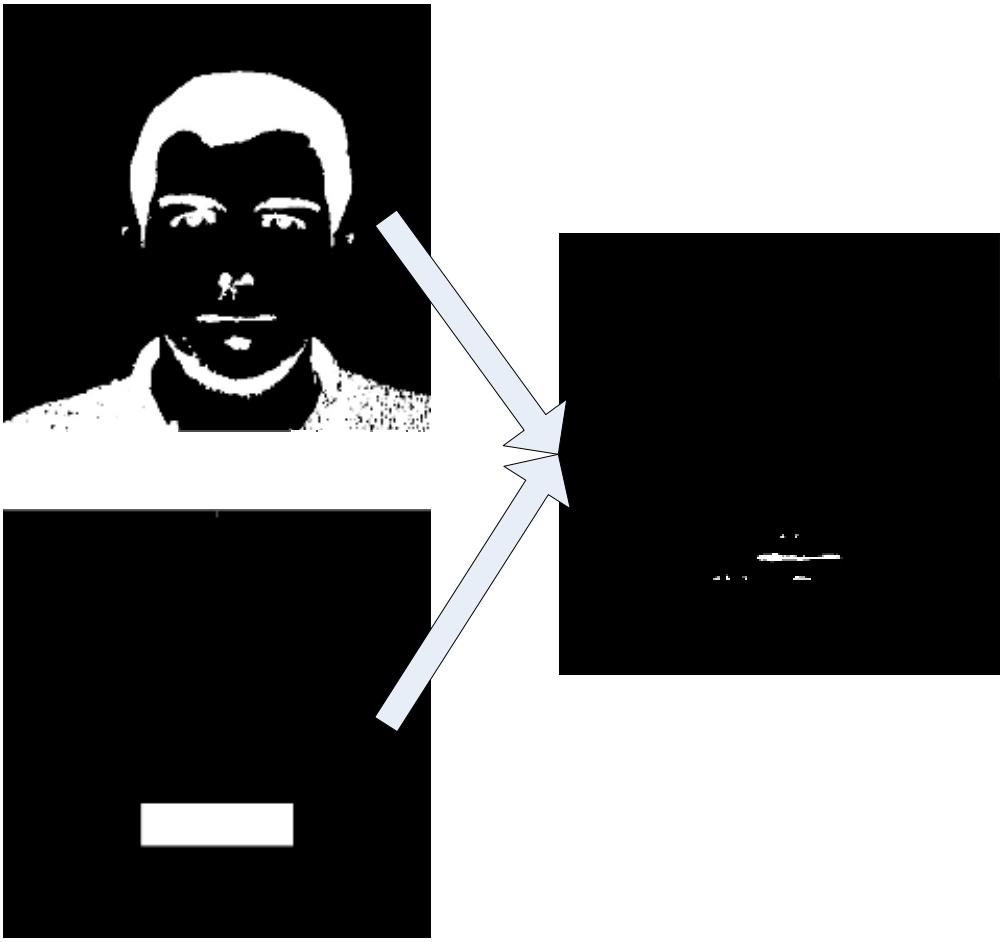


Fig. 6 Aplicamos una mascara para separar a la boca del resto

La máscara que aparece en la figura no es como la que se usa en realidad, en ésta se ha exagerado su tamaño para hacer ver lo que realiza más claramente. La máscara de la realidad es mucho más grande, lo cual permite que en todos los casos la boca quede dentro del cuadro. Bueno ya nos queda sólo la boca, nos queda realizar la medida.

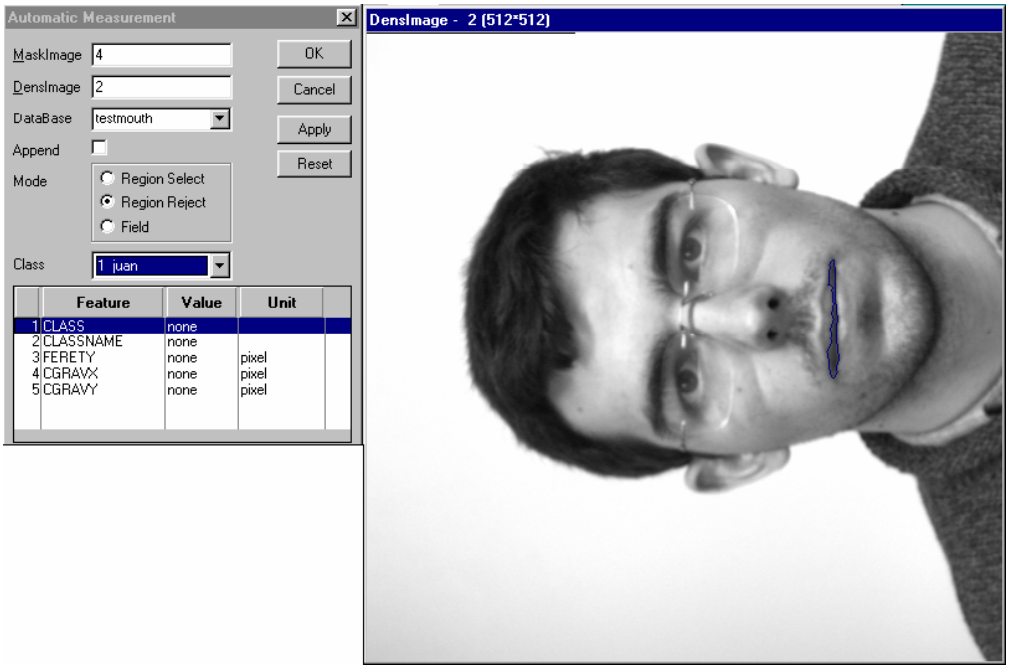


Fig 7. Medición de la boca.

Queda bastante claro que la longitud y la posición de la boca quedan perfectamente delimitada en el proceso. El ancho sólo será utilizado para diferenciar la boca de otros elementos colindantes como la barbilla. Al final todos estos datos quedan grabados en la base de datos 'Testmouth'.

- Medición de los ojos

En los ojos medimos las siguientes características:

- Area.
- Circularidad.
- Posición de los ojos.
- Longitud
- Ancho.

El proceso de medición de los ojos es muy similar al proceso de medición de la boca. Los filtros no son los mismos pero el proceso a grandes rasgos es igual.

A continuación exponemos el proceso de medición igual que antes, pero obviamos los primeros pasos al ser prácticamente iguales a los anteriores.

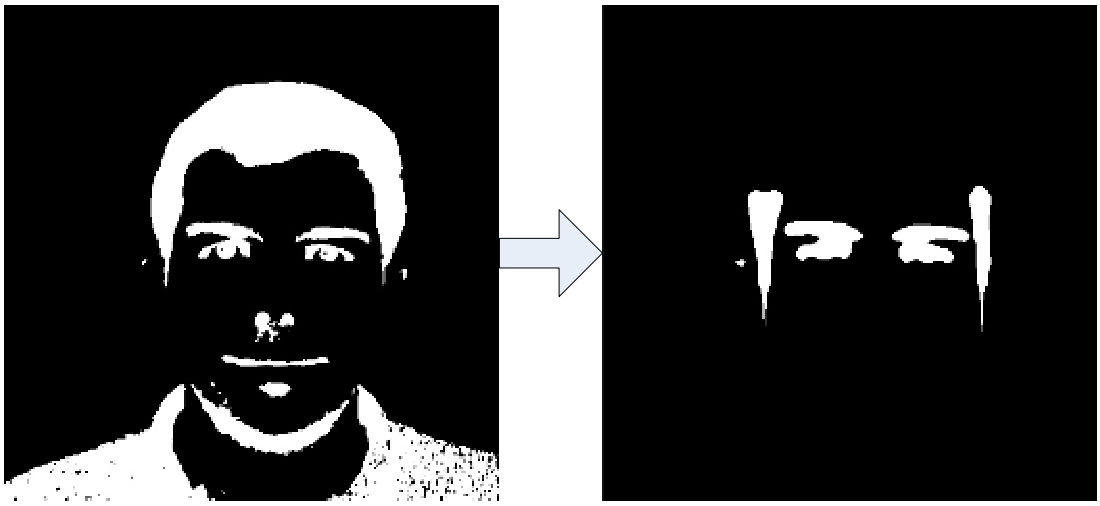
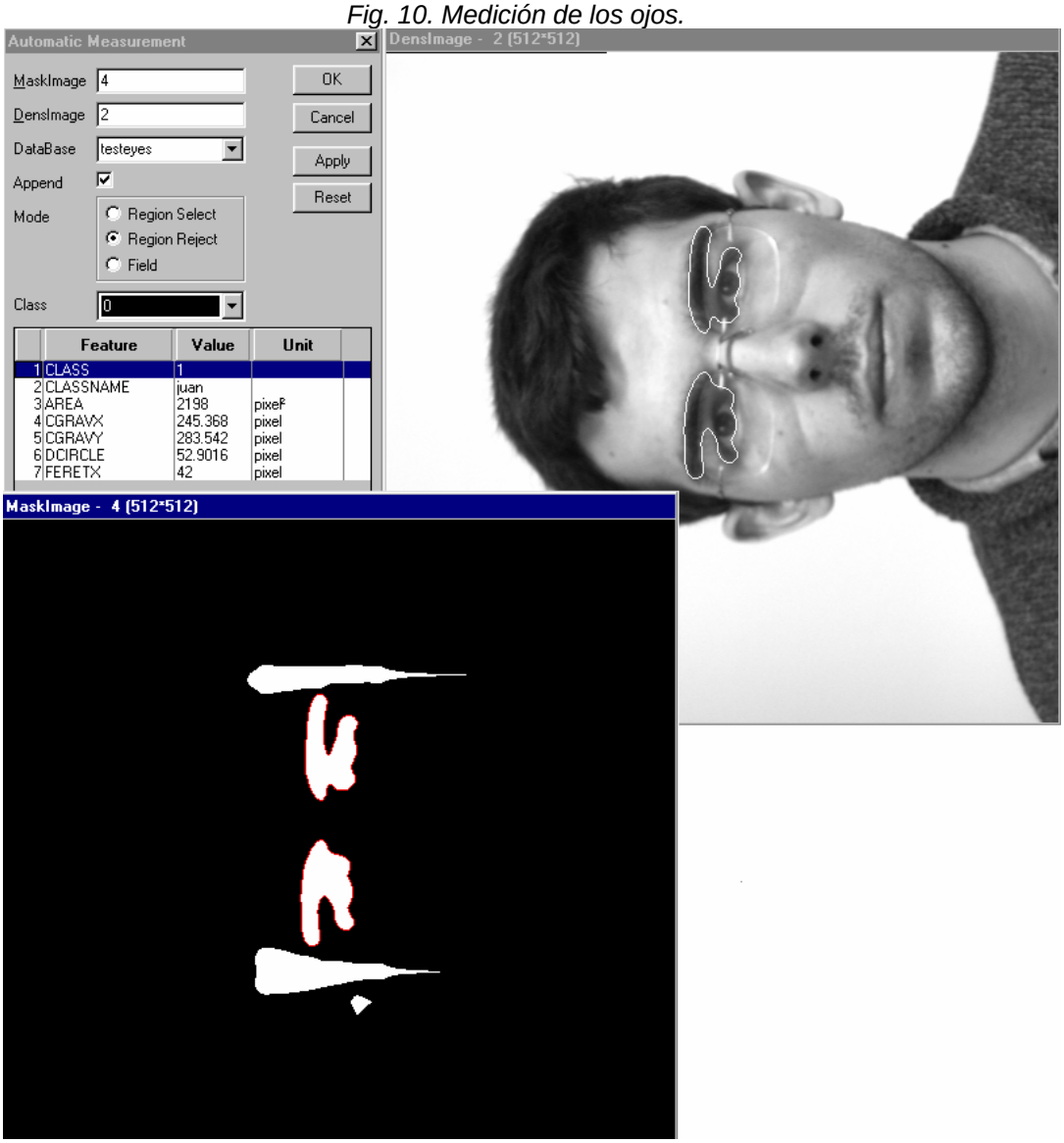


Fig. 9 Aplicación máscara y dilatación.

Como vemos, no seleccionamos los ojos, sino el conjunto ojos-cejas, por ser más sencillo de medir y ofrecer casi las mismas características el centro mismo de los ojos.

Fig. 10. Medición de los ojos.



Como vemos los ojos quedan seleccionados y medidos sin confusión alguna. Esto lo realiza gracias a la imposición de condiciones sobre los objetos a elegir tales como la longitud o el ancho. Los valores de estas condiciones se obtienen mediante la observación de resultados. Así los objetos como la boca y los ojos son automáticamente reconocidos y el proceso se hace totalmente automático.

Finalmente todos estos datos quedan guardados en la base de datos 'Testeyes'.

Así acaba la parte de medición del código.

- Construcción de la base de datos 'Face'

Lo que hacemos a continuación es unir ambas bases de datos en una base de datos única. Creando además nuevas características que son el resultado de la interacción de las dos anteriores como puede ser la distancia de los ojos a la boca o el ángulo que forman entre ellos.

KS400 nos ofrece una serie de comandos para realizar estos pasos para la creación de la base de datos nueva.

Finalmente tras este proceso, las características que nos quedan son las siguientes:

- Longitud de la boca.
- Área del ojo izquierdo.
- Área del ojo derecho.
- Distancia entre ojos.*
- Distancia del ojo izquierdo a la boca.*
- Distancia del ojo derecho a la boca.*
- Ángulo entre el ojo izquierdo y la boca.*
- Ángulo entre el ojo derecho y la boca.*
- Circularidad del ojo izquierdo.
- Circularidad del ojo derecho.

Todas las características marcadas con (*) son el resultado de la interacción de las dos bases de datos. Finalmente tenemos diez características para poder usar en la clasificación.

- Transformación y normalización de la base de datos "FACE"

En este punto nos enfrentamos a los problemas del KS400. Si pudiéramos aprender de la base de datos directamente y no de una imagen, no sería ningún problema la utilización directa de la base de datos "FACE". Pero éste no es el caso. Nos topamos con que no podemos hacer aprender al programa con nuestra base de datos y tenemos que buscar una solución. La solución tomada: La creación de imágenes virtuales en base a los datos de "FACE". Con estas imágenes sí podríamos entrenar a un clasificador. Pero esto concierne al siguiente punto, porque antes, para poder crear estas imágenes, deberíamos normalizar los valores para poder insertarlos.

Por ejemplo: Cuando creamos las imágenes, sólo podemos usar números enteros (representan a los píxeles y no se puede dibujar medio píxel) y los valores que tenemos en la base contienen decimales o por otra parte éstos pueden ser muy grandes (caso de las áreas) con lo que deberíamos reducir su valor con una normalización.

A partir de aquí también vamos a diferenciar las dos versiones del código que implementamos. Se van a diferenciar por la normalización que será a 100 o a 10 según versiones.

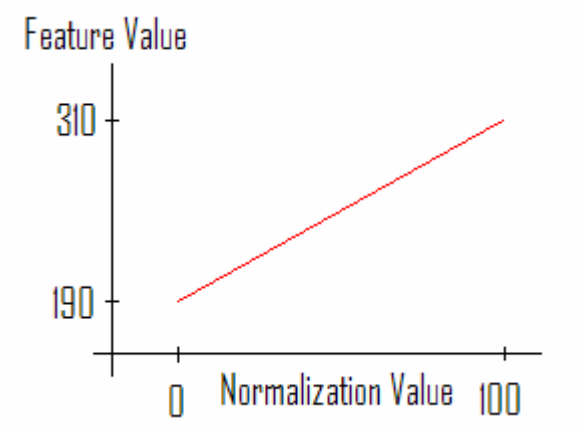


Fig 11. Método de normalización.

Aquí tenemos un ejemplo de normalización para un valor que oscila entre 200 y 300 (damos un margen para nuevas entradas del 10%).

También señalamos que en el código hay una parte que puede parecer "cómica" ya que realiza la conversión de decimal a entero de una manera casi "prehistórica", esto se debe a que en el KS400 la función de pasar de decimal a entero no está correctamente implementada!!.

Al final del proceso nos quedan menos características ya que hemos fundido características pertenecientes a los dos ojos en una, realizando la media entre las mismas para cada cara. Todo queda guardado en la base de datos 'Facenorm'.

- Creación de imágenes virtuales y, según parte, aprendizaje del clasificador o clasificación.

Ahora llega el momento del aprendizaje del clasificador. El proceso consiste en la definición de un clasificador y el posterior aprendizaje a partir de las imágenes creadas a partir de la base de datos 'Facenorm'. En caso de ser parte de la clasificación lo que se hace es aplicar el clasificador a la imagen creada.

Las imágenes son rectángulos cuyas dimensiones y posición son dependientes de los valores tomados de la base de datos. O sea, hacemos una traducción de las características de la imagen (longitud de boca por ejemplo) en otras como por ejemplo longitud del cuadrado. Hemos usado el rectángulo por ser la imagen más compleja (con más variables) que nos deja construir el KS400.

Los rectángulos son dibujados sobre una imagen vacía y esa imagen será inmediatamente medida para tomar los valores del rectángulo. El tamaño de esta imagen depende de la versión que estemos utilizando del código. En la versión normal ésta es de 200x200 píxeles mientras que en la extendida es de 2000x2000 píxeles.

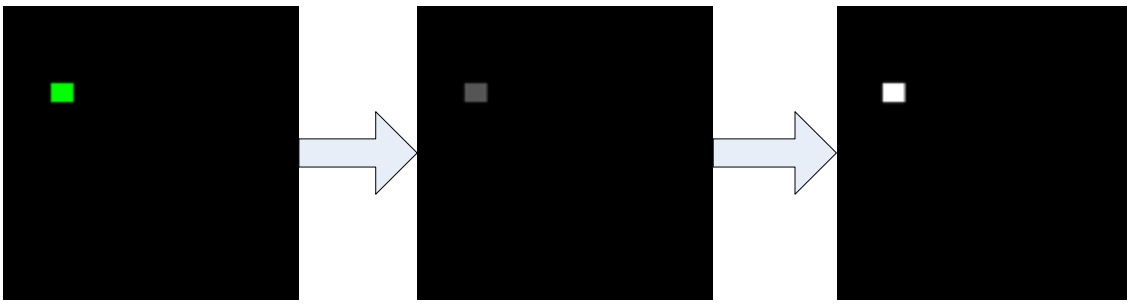


Fig 12. Imágenes virtuales y obtención de la máscara.

Ahora vamos a mostrar cómo son las imágenes en las que insertamos el rectángulo.

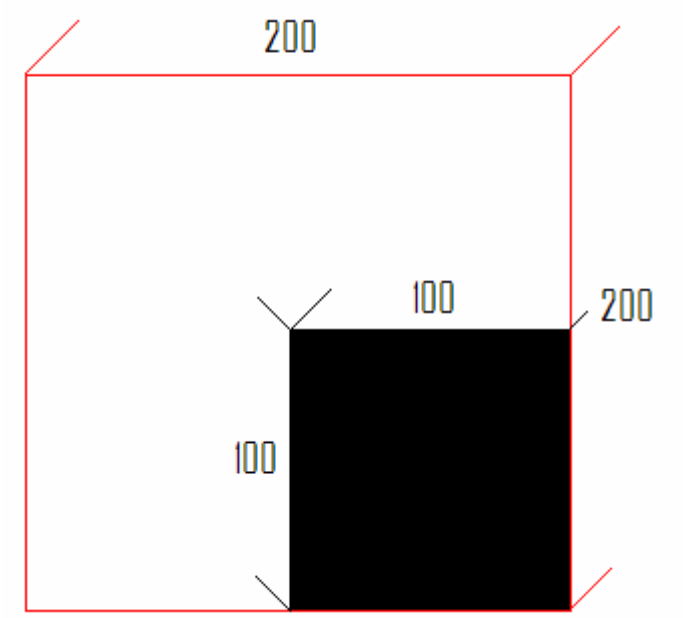


Fig 13. Dimensiones de la cuadrícula base.

Esta cuadrícula es la imagen a medir en el caso de la versión sencilla y la cuadrícula base en caso de la versión expandida. Tiene un tamaño de 200x200 y vemos que está diseñada para que el rectángulo no rebase en ningún caso los límites de la misma.

En la versión extendida lo que hacemos es usar una imagen de 2000x2000 dónde diferenciamos 10x10 o sea 100 cuadrículas de las anteriores. Con esto lo que conseguimos es poder usar dos variables más, que se mueven de cero a nueve. Con lo cual multiplicamos por 100 las posibilidades y aumentamos hasta seis las variables utilizadas en la clasificación. Como desventaja tenemos el que el programa se hace más lento.

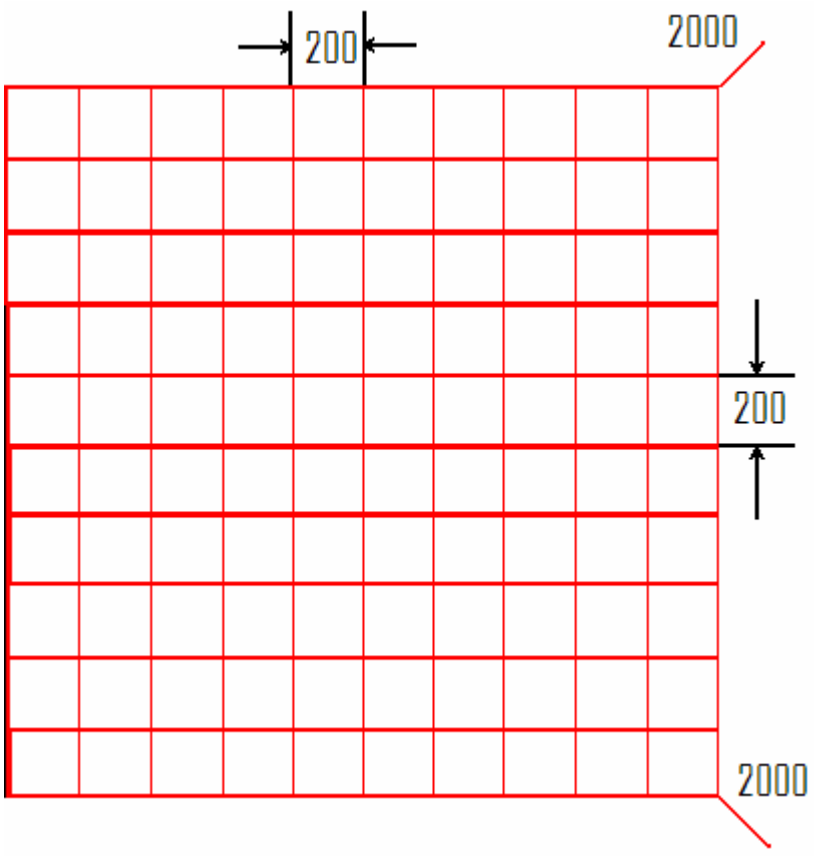


Fig 14. Nueva cuadrícula usada para la clasificación.

El código está programado para poder tomar las fotos para la clasificación en el mismo momento de hacerla con lo que se podría hacer reconocimiento en tiempo real.

Podemos usar los diferentes clasificadores ya sea de mínima distancia, bayesiano o mahalanobiano. Los resultados de la clasificación serán guardados en una base de datos y además podrán ser presentados por pantalla si se desea.

En el siguiente esquema se puede entender mejor el proceso. Se ve cómo tomamos las caracteríticas de la imagen. Después las normalizamos y finalmente traducimos en otras caracteríticas correspondientes a una figura, un rectángulo en nuestro caso. Para poder introducir más caracteríticas se utilizaron las cuadrículas. El uso de las cuadrículas consiste en dividir la imagen virtual en cuadros (cuadrículas) y dependiendo de dos valores(características) representar en un determinado cuadro el rectángulo. Así en vez de introducir una característica en una variable podemos introducir dos. Así podemos utilizar todas las caracteríticas seleccionadas (seis) en cuatro variables

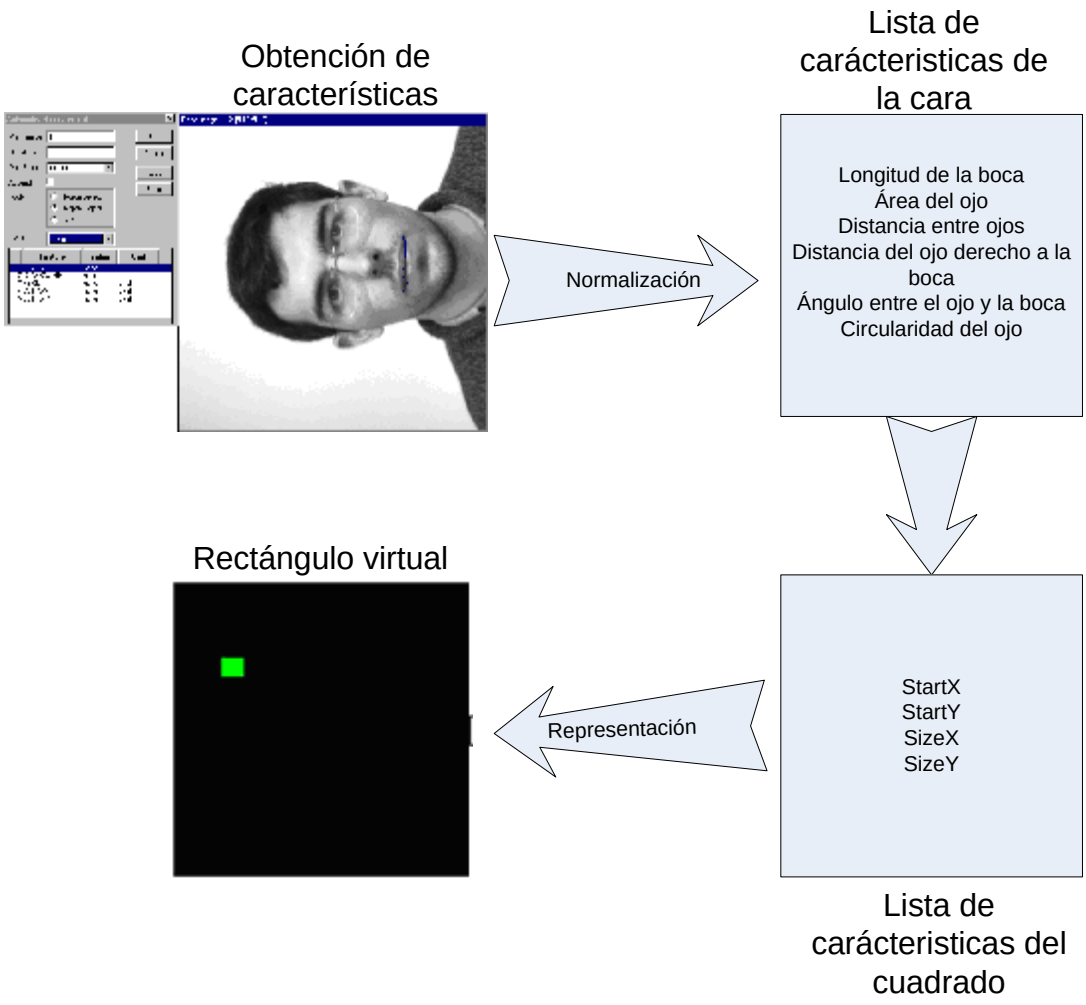


Fig 15. Creación de un rectángulo virtual

3. Resultados

3.1. Imágenes de la base de datos.

3.1.1. Protocolo

El protocolo seguido fue muy simple. Instalamos un puesto fotográfico en el laboratorio con los elementos que encontramos en él. Ningún elemento fue comprado para usarlo allí.

Después fijamos varias reglas para la realización de las fotos. Las más importantes las concernientes a la luz y la posición por su intervención decisiva en la clasificación.

Fig 16. Cámara y pantalla.

Éste es el puesto fotográfico. La distancia y la posición a la cámara están fijadas mediante unas marcas en el suelo y la altura es regulada mediante el soporte dónde está fijada la cámara.

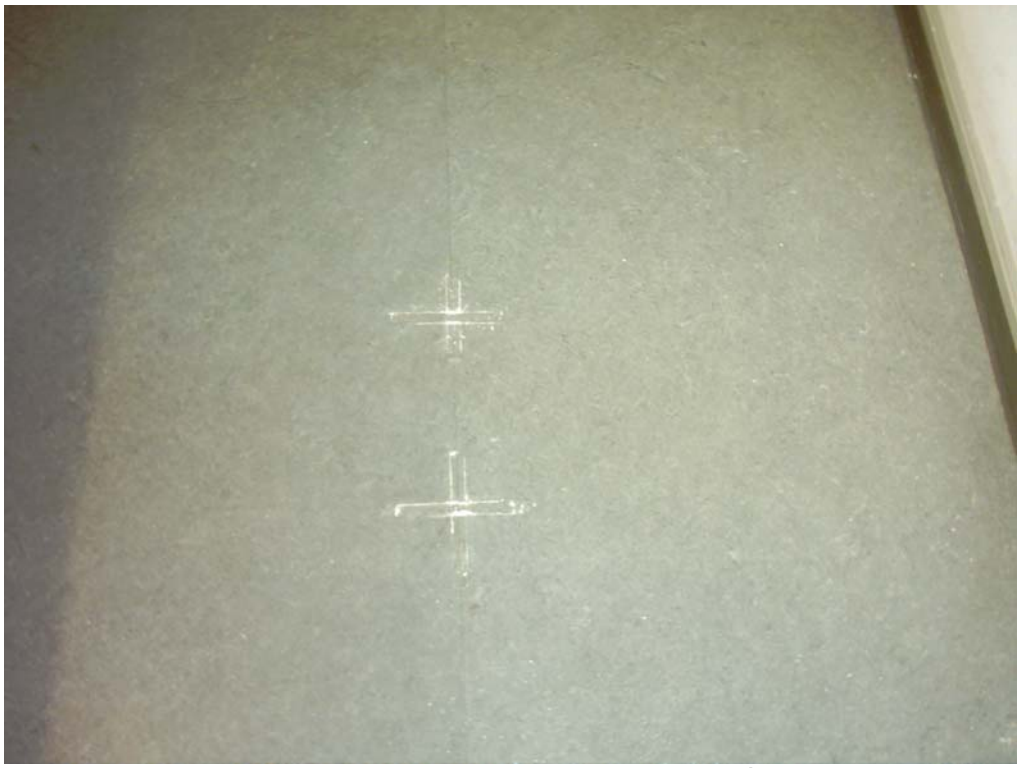


Fig 17. Marcas para fijar la posición.

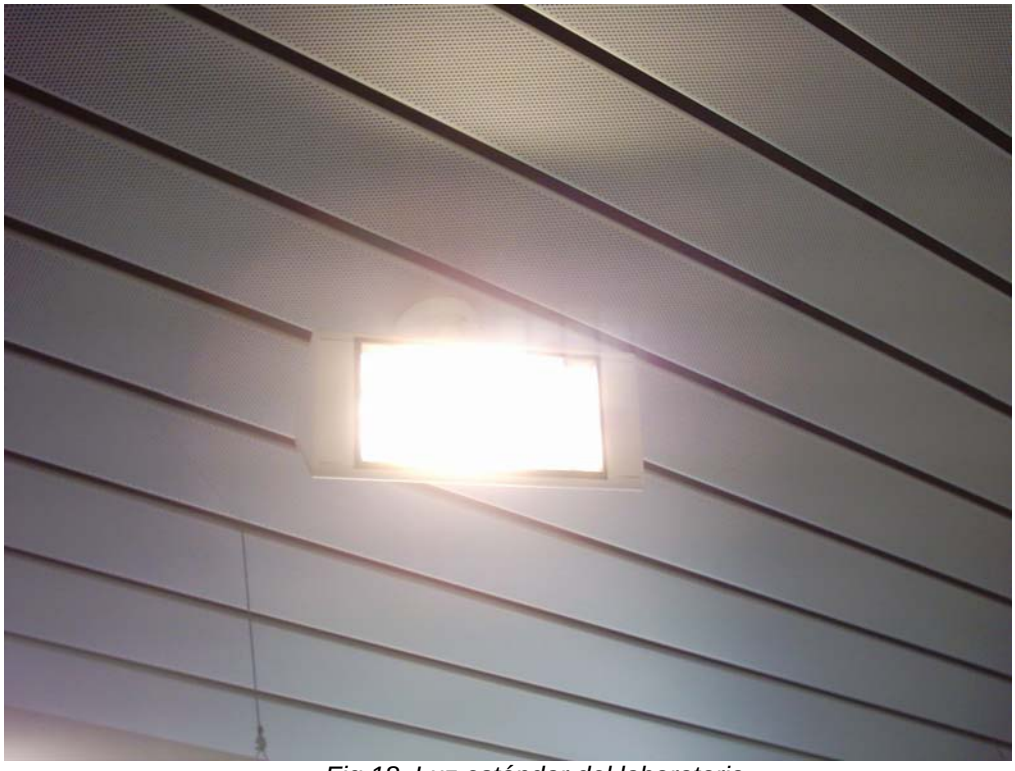


Fig 18. Luz estándar del laboratorio.

3.1.2. Imágenes

3.1.2.1. Base de datos

Nuestra base de datos consta de once personas. Cada una tiene al menos tres fotos. Éstas pueden además pertenecer a diferentes sesiones. En un principio nuestra base era de alrededor de veinte personas pero debido a que el programa se hacía demasiado lento o a que las fotos no siempre cumplían con todos los requisitos, se decidió reducir el número de sujetos a once.

Fig 19. Ejemplo de imágenes

Las fotos son de 512x512 pixels. Las condiciones como luz o distancia son como las que vemos en el ejemplo.

3.1.3. Material

3.1.3.1. Luces

Las luces usadas fueron:

- Luces normales de laboratorio. Ver figura 17.
- Un foco profesional. Primalux 2500. 2500W max.

Fig 20. Vista trasera del foco.

3.1.3.2. Cámara

Usamos una cámara SONY XC-003/003P. La XC-003/003P es una cámara diseñada para el control de procesos y para aplicaciones de procesado de imágenes.

Fig 21. SONY XC-003/003P CAMERA.

3.2. Resultados

3.2.1. Resultados de los experimentos

3.2.1.1. Primer experimento

La primera parte de los experimentos consistió en probar si al introducir las imágenes tomadas en el laboratorio el algoritmo daba problemas. Como vemos en los resultados el 79% de las imágenes que habíamos fue procesado correctamente por el algoritmo. Esto quiere decir midió correctamente las características pedidas y fueron iniciados los clasificadores (de los tres tipos y en ambas versiones) sin problemas.

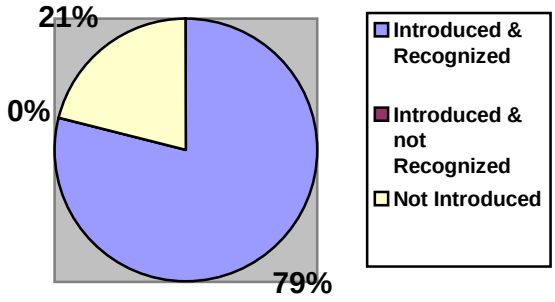


Fig 22. Resultados del experimento.

En realidad el porcentaje de acierto es mayor ya que hemos considerado que si el clasificador no puede ser iniciado a causa del error en un individuo, todas las muestras asociadas a éste son erróneas (aunque sólo una de ellas produzca el error). Las muestras que produjeron error fueron imágenes en las que se ocultaban rasgos, como por ejemplo el cabello delante de los ojos, o mucha barba.

A partir de aquí, sólo nos quedaba la clasificación de las imágenes que teníamos. Pusimos a reconocer las imágenes que ya habíamos clasificado. Reconoció sin un solo error las muestras de todos los individuos (100% de acierto, ni una sóla imagen muestreada fue no reconocida posteriormente). Esto demostraba que no existían solapamientos entre muestras.

4.2.1.2 Segundo experimento.

Luego comenzamos el segundo experimento que consistía en reconocer a muestras de personas que no estaban dentro del clasificador. Este funcionamiento es lo que puede resultar más lógico ya que reconocer imágenes ya conocidas no tiene mucho sentido práctico.

Utilizamos las imágenes que ya pudimos introducir en el primer experimento. Pero tenemos un problema, de algunos sujetos sólo tenemos tres imágenes con lo que cuando quitemos una del clasificador sólo tendremos dos. Esto hace aún más difícil la clasificación con lo que los resultados deberán ser peores de lo que en realidad podríamos obtener.

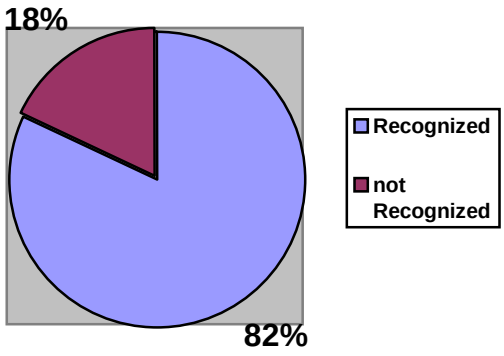


Fig 23. Resultados del segundo experimento.

El resultado obtenido fue muy bueno, obteniendo un porcentaje de éxito del 82%.

En las pruebas se utilizaron los tres tipos de clasificadores (Mínima distancia, bayesiano y mahalanobiano). Se realizaron para las mismas imágenes las mismas pruebas usando los diferentes clasificadores. No se observaron en estas pruebas diferencias en los resultados entre los clasificadores. Esto se puede deber a que el número de pruebas no fue lo suficientemente alto (alrededor de 40) y/o a que el porcentaje de acierto fue muy elevado. Está claro que si no hay error tampoco podemos apreciar las diferencias entre los clasificadores. En las pruebas finales se optó por usar el clasificador bayesiano.

Otro punto a analizar son las diferentes versiones del programa. Como ya hemos dicho, desarrollamos dos versiones del programa, una con una cuadrícula para contar con mayor número de características (seis) y otra sin ella pero con mayor precisión en las variables. Se probaron ambas versiones para evaluar las diferencias entre ellas. El resultado es que ambas versiones obtienen un resultado similar en cuanto al acierto. Pero entre estas versiones existe una diferencia. Durante el aprendizaje, si las muestras de un mismo individuo presentan valores similares en las características (esto es, misma longitud de la boca por ejemplo) el clasificador no se puede iniciar ya que tiene una varianza 0 en sus muestras. Así, la versión con seis caracteríticas presenta menos problemas aquí ya que la coincidencia es más extraña cuanto más variables (con la misma precisión) tengamos. El problema es que esta versión es bastante más lenta que la otra.