Trabajo Fin de Grado
Grado en Ingeniería de las Tecnologías de Telecomunicación

Estudio comparativo de modelos de identificación facial basados en correlación

Autor: María Sierra Zapata
Tutor: Eduardo Fernández Camacho

Dep. Ingeniería de Sistemas y Automática
Escuela Técnica Superior de Ingeniería
Universidad de Sevilla

Sevilla, 2015

Trabajo Fin de Grado
Grado en Ingeniería de las Tecnologías de Telecomunicación

Estudio comparativo de modelos de identificación facial basados en correlación

Autor:
María Sierra Zapata

Tutor:
Eduardo Fernández Camacho
Catedrático

Dep. Ingeniería de Sistemas y Automática
Escuela Técnica Superior de Ingeniería
Universidad de Sevilla
Sevilla, 2015

Trabajo Fin de Grado: Estudio comparativo de modelos de identificación facial basados en correlación

Autor: María Sierra Zapata

Tutor: Eduardo Fernández Camacho

El tribunal nombrado para juzgar el Trabajo arriba indicado, compuesto por los siguientes miembros:

Presidente:

Vocales:

Secretario:

Acuerdan otorgarle la calificación de:

Sevilla, 2015

El Secretario del Tribunal

A todos los que han estado a
mi lado durante estos años

Agradecimientos

Primero, quisiera agradecer este proyecto a mi tutor, Eduardo Fernández Camacho, por brindarme la oportunidad de realizarlo y que tanto me ha guiado durante la elaboración del mismo.

También me gustaría mencionar a mis amigos, tanto de dentro como de fuera de la carrera. Me han acompañado durante este tramo de mi vida apoyándome y ayudándome a superar cada obstáculo. Aunque jamás te reconoceré que me ayudaras en aquel trabajo de programación de primero, Eugenio. Espero seguir contando con todos ellos en las próximas etapas de mi vida.

Por último, quiero hacer una alusión especial a mi familia, que tanto ha sacrificado para poder encontrarme hoy aquí, realizando un último esfuerzo para terminar estos estudios. En especial, a dos personas que forman parte de ella. Mi hermana, Isabel, sin la cual no sabría que hacer. A pesar de todas las peleas y las riñas, al final es la persona con la que puedo contar para todo. Y mi pareja, Ramón, que a pesar de ser una incorporación reciente, para mí ya forma parte de mi familia. Consigue arrojar luz sobre los momentos de oscuridad y siempre está para levantarme en los momentos díficiles. Gracias por este maravilloso tiempo junto a ti que espero poder disfrutar siempre.

Resumen

El presente Trabajo Fin de Grado trata sobre el estudio comparativo de los modelos de identificación facial basados en correlación.

Los objetivos a cumplir son:

- Analizar los métodos de identificación facial más extendidos de una manera exhaustiva, desde su algoritmia matemática hasta su implementación informática.
- Estudio comparativo, donde se incluyen aspectos como tasa de aciertos, falsos positivos o tiempos de identificación.

Para ello, se ha realizado un estudio en profundidad de los modelos de identificación EigenFace, FisherFace y LBPHFace.

Se contemplan para su realización cuatro variables: el tamaño de las muestras de los sujetos, el número de sujetos que formarán el modelo de identificación, el número de muestras por sujeto que forman el modelo y la estructura del modelo empleado. Se observará la influencia de las modificaciones de estas variables en los modelos de identificación.

Tras la obtención de los resultados, se definirán los parámetros de tiempo y fiabilidad que cumple cada modelo. Por último, se finalizará este estudio con las conclusiones obtenidas de los resultados generados.

Abstract

This Degree Final Project deals with the comparative study of face recognition model based on correlation.

The objectives to be fulfilled are:

- Analyze the most spread out face recognition methods which are exhaustively described, from its mathematical algorithms to their computer implementation.
- Comparative study, where aspects such as hit rate, false positive or identification times are included.

For that, a thorough study of the EigenFace, FisherFace and LBPHFace identification models has been made.

For its realization, it is considered four variables: the size of subjects' samples, the number of subjects who forms the identification model, the number of samples per subject which forms the model and the model's structure used. The influence of changes in these variables in the identification model will be observed.

After obtaining the results, time and reliability parameters that fulfills each model will be defined. Finally, this study will be completed with the conclusions drawn from the results generated.

Índice

ÍNDICE DE FIGURAS

Notación

TFG	Trabajo Fin de Grado
PCA	Principal Component Analysis
LDA	Linear Dimension Analysis
LBP	Local Binary Patterns
e.o.c	En otro caso
LBPH	Local Binary Patterns Histograms
GHz	Gigahercios
L3	Level 3
MB	Megabytes
RAM	Random Access Memory
GB	Gigabytes
DDR3	Double Data Rate type 3
”	Pulgadas
GUFD	Glasgow Unfamiliar Face Database
ISO	International Organization for Standardization
GCC	GNU Compiler Collection
Mat	Tipo definido por OpenCV para contener la información relativa a una imagen

1 INTRODUCCIÓN

"I learned that computer sciense is not just about syntax and coding. We can make a difference in people's lives by developing applications."

- Kyle Rector -

El problema de la identificación facial se ha planteado desde múltiples puntos de vista. De esta manera, ha sido implementado en aplicaciones de diversos ámbitos. La identificación facial es empleada en áreas que abarcan desde la seguridad informática hasta el entretenimiento personal. Un ejemplo, dentro del área de seguridad, se tiene en los teléfonos móviles. Estos dispositivos contienen formas diferentes para realizar un patrón de desbloqueo, requiriendo una de ellas el uso de la identificación facial.

Este proyecto versa sobre el funcionamiento de los modelos de identificación facial basados en correlación. Realizando un estudio en profundidad sobre estos modelos, se quiere dar con las respuestas a preguntas como cuánto tiempo se tarda en realizar una identificación, cuál es la tasa de falsos positivos del modelo o cómo de fiables son dichos métodos.

A través de la comparación de los resultados que se obtengan a lo largo de este estudio, se hallarán los parámetros que rigen el funcionamiento de los modelos. Se tendrán cuatro variables: el tamaño de las muestras de los sujetos, el número de sujetos que formarán el modelo de identificación, el número de muestras por sujeto que forman el modelo y la estructura del modelo empleado.

1.1. Motivación

La Visión Artificial es un mercado en auge. Desde que se hicieran accesibles las librerías de desarrollo de este ámbito, el número de aplicaciones existentes en torno a esta rama no hace más que aumentar. Dentro de la Visión Artificial, se encuentra el reconocimiento y la identificación facial, siendo esta última el dominio de estudio de este proyecto.

Así pues, la motivación que va ligada a este trabajo es realizar un estudio en profundidad, comparando los modelos existentes para la identificación facial basados en correlación. Conociendo el comportamiento de estos modelos, se puede elegir el mejor método a utilizar cuando se desee desarrollar una aplicación de identificación facial. De esta forma, se podrá decidir la implementación del modelo a través de los parámetros que se quieran cumplir.

1.2. Objetivos

- Analizar los métodos de identificación facial más extendidos de una manera exhaustiva, desde su algoritmia matemática hasta su implementación informática.
- Estudio comparativo, donde se incluyen aspectos como tasa de aciertos, falsos positivos o tiempos de identificación.

Para alcanzar estos objetivos, se realizará una implementación de dicho estudio, la cual generará una base de datos con los resultados de todas las variantes de los modelos empleados. Obtenidos los resultados, serán procesados para así obtener los parámetros que gobiernan dichos modelos para su posterior análisis.

1.3. Metodología

Se realizará un estudio sobre los modelos de identificación existentes en OpenCV [1]. Existen tres modelos: modelo FisherFace, modelo EigenFace y modelo LBPHFace. Se evaluarán tanto en su forma simple como en una estructura en árbol. En total, se analizarán seis modelos.

Se tendrá una base de datos con cien sujetos, la mitad hombres y la otra mitad mujeres [2]. El estudio se realizará sobre diez, cincuenta y cien clases clasificadoras. Una clase clasificadora es un sujeto único a identificar.

Se comparará cómo afecta el hecho de tener imágenes más pequeñas o más grandes en el modelo, una imagen o dos imágenes por sujeto y el número de clases clasificadoras a la hora de construir el modelo. Así, se obtendrán los parámetros siguientes:

- Tiempo de creación del modelo.
- Tiempo de carga del modelo.
- Tiempo de identificación medio del modelo.
- Porcentaje de falsos positivos.
- Porcentaje de falsos positivos en mujeres.
- Porcentaje de falsos positivos en hombres.
- Porcentaje de tasa de acierto.
- Porcentaje de tasa de acierto en mujeres.
- Porcentaje de tasa de acierto en hombres.

1.4. Estructuración del proyecto

- El capítulo 2 desarrolla en mayor profundidad qué es la visión artificial y el reconocimiento facial. Además, se definirá cada modelo de identificación.
- El capítulo 3 muestra las herramientas empleadas para el desarrollo del estudio a nivel hardware y software.
- El capítulo 4 está compuesto por la implementación del estudio. Se definirá la interfaz del estudio, la programación de visión artificial realizada y cómo se guardan los datos generados.
- El capítulo 5 contiene la implementación para la representación gráfica de los resultados obtenidos y las gráficas generadas.
- Por último, en el capítulo 6 se encuentran las conclusiones obtenidas tras la realización de este estudio y se presentan posibles vías de futuro.

2 VISIÓN ARTIFICIAL

El término Visión Artificial ha sido utilizado a lo largo de toda la introducción. Sin embargo, ¿qué es la Visión Artificial? La Visión Artificial es un campo de la Inteligencia Artificial [3]. Mediante modelos matemáticos intenta conseguir la percepción realizada por el sentido de la vista. Es decir, trata de obtener información a través del procesado de imágenes.

Se intentan extraer características relevantes visuales haciendo uso de procedimientos automáticos. La información extraída puede ser muy variada. No solo se obtienen propiedades físicas, como puede ser el tamaño, la forma o la localización de un objeto, sino también propiedades del material (color, textura...) o qué se está identificando (un objeto, una persona, un animal...).

Normalmente, la información extraída debe de cumplir una serie de requisitos. Se deben tener en cuenta las tasas de fallos, los falsos positivos y los falsos negativos que conllevan la utilización de este tipo de algoritmia. La Visión Artificial no es perfecta y hay que tratar de minimizar el impacto de este tipo de errores en el sistema.

A pesar de no ser perfecta, la Visión Artificial es muy eficaz en tareas visuales repetitivas y alineantes para el hombre. Por ello, es aplicada en muchos sectores diferentes. En sectores como la industria, se utiliza para la inspección de piezas, sobre todo para que pasen pruebas de calidad de forma automática. También es utilizada en vehículos autoguiados, para determinar la trayectoria del vehículo.

En la actualidad, ya no solo es exclusiva para la industria. Existen muchas aplicaciones a nivel de usuario, como ya se ha mencionado en la introducción. Otro ejemplo de ello es que algunos dispositivos móviles tienen la capacidad de ordenar las imágenes de la galería por los sujetos que aparecen en ellas y agrupar todas las imágenes de un mismo sujeto en una carpeta solo para dicho sujeto.

Esta última aplicación pertenece a la rama del reconocimiento y la identificación facial dentro de la Visión Artificial. En las próximas secciones de este capítulo, se hablará de dicha rama y se explicará la base y fundamentación de los métodos y modelos en los que se basan la identificación facial.

La versatilidad de aplicaciones en las que participa la Visión Artificial hace que sea una rama muy interesante de estudio.

2.1. Reconocimiento e identificación facial

El reconocimiento y la identificación facial conllevan dos fases en Visión Artificial. Primero, se debe encontrar a una persona en la imagen a examinar y, a continuación, proceder a la identificación de dicha persona comparándola con una base de datos, todo ello de forma automática. Es importante diferenciar ambos procesos, ya que el presente estudio solo se implicará en la segunda parte, la identificación facial.

Grandes empresas hacen uso de este tipo de algoritmia. Mastercard ha desarrollado una aplicación para que a la hora de pagar con la tarjeta, se valide que es el propietario de la tarjeta el que está realizando la compra[4]. Facebook incluye este tipo de métodos también, cuando preselecciona dónde se encuentra una cara o sugiere cuál de tus amigos es el que se encuentra en el recuadro.

El principal problema de la identifiación facial es que se trata de biométrica. Los algoritmos empleados en esta rama de la Visión Artificial son muy susceptibles a los cambios de iluminación, tamaño de las imágenes, calidad de las imágenes… sin contar con los cambios físicos de las personas, como es el caso a estudiar. Cambios como la barba, el color del pelo o el llevar gafas pueden afectar en la identificación.

Es necesario estudiar y definir todas las variables implicadas en este proceso. Conociéndolas, se puede realizar un sistema más robusto frente a cambios de este tipo. Hay que comprender la base de los modelos para poder utilizarlos de la forma más adecuada posible. Saber el modelo de identificación más adecuado para la aplicación que se desee crear también es importante.

Existen dos grandes agrupaciones, en cuanto a modelos de identificación facial se refiere: los modelos basados en caracteríscas geométricas de la cara y los modelos basados en correlación. Los modelos basados en características geométricas son los más intuituivos [5]. Uno de los primeros sistemas de reconocimiento facial automatizado utilizaba puntos de marcado (posición de los ojos, de las orejas, de la nariz…) para construir un vector de características (distancia entre los puntos, ángulo entre ellos…) [6].

El reconocimiento se realiza calculando la distancia euclídea entre las características de la imagen de prueba y la imagen de referencia. Este método es robusto contra los cambios de iluminación, pero tiene un enorme inconveniente: el registro exacto de los puntos de marcado es complicado. Uno de los últimos métodos implementados basados en geométrica facial demuestran que por sí solo este tipo de método no conlleva suficiente información para el reconocimiento facial [7].

Los modelos basados en correlación son los elegidos en el desarrollo de este esutdio. Para realizar la identificación utilizando las librerías de código abierto de Visión Artificial, OpenCV [1], existen tres tipos diferentes de modelos. Se desarrollará el modelo matemático de cada uno de ellos, explicando su fundamento y funcionamiento paso a paso. Los tres modelos son:

- Modelo EigenFace.
- Modelo FisherFace.
- Modelo LBPHFace.

2.1.1. Modelo EigenFace

El principal problema de la representación de imagénes es su alta dimensionalidad [5]. Imágenes en blanco y negro con dos dimensiones $p\ x\ q$ ocupan un vector con un tamaño $m = p \cdot q$. La cuestión es, ¿todos los datos de dichos vectores son necesarios? Solo se puede tomar una decisión cuando existe una variación en los datos. Por tanto, se busca la información más relevante.

El Análisis de Componente Principal [8] (PCA en inglés) se encarga de que un conjunto de variables posiblemente correladas se conviertan en un conjunto de variables incorreladas. Los métodos PCA encuentran la dirección con la mayor varianza en los datos, llamados componentes principales.

- Descripción del algoritmo

Suponga $X = \{x_1, x_2, \dots, x_n\}$ un vector aleatorio con observaciones $x_i \in R^d$.

1. Compute la media μ

$$\mu = \frac{1}{n}\sum_{i=1}^{n} x_i$$

2. Compute la Covarianza Matriz S

$$S = \frac{1}{n}\sum_{i=1}^{n}(x_i - \mu)(x_i - \mu)^T$$

3. Compute los autovalores λ_i y los autovectores v_i de S

$$Sv_i = \lambda_i v_i, i = 1,2,\dots,n$$

4. Ordene los autovectores de forma descendiente por su autovalor. Los componentes principales k son los autovectores correspondientes a los mayores autovalores k.

Los componentes principales k del vector observado x son dados por:

$$y = W^T(x - \mu)$$

donde $W = (v_1, v_2, \dots, v_k)$.

La reconstrucción desde las bases PCA se hace con:

$$x = Wy + \mu$$

donde $W = (v_1, v_2, \dots, v_k)$.

- Resumen del método

El método EigenFace realiza la identificación facial:

- Proyectando todas las muestras a entrenar en el subespacio PCA.
- Proyectando la imagen a identificar en el subespacio PCA.
- Encontrando el vecino más cercano entre la proyección de las imágenes entrenadas y la proyección de la imagen a identificar.

2.1.2. Modelo FisherFace

El PCA, que es la base del método EigenFace, encuentra combinaciones lineales de características que maximicen la varianza total en los datos [5]. Aunque es claramente una manera bastante poderosa para representar datos, no implica que no pueda perderse mucha información cuando se utiliza ese método.

Imagine una situación donde la varianza de los datos es generada por una fuente externa, como puede ser la iluminación. Los componentes identificados por un PCA no contiene ninguna información discriminativa, así que las muestras proyectadas se correlan juntas y la clasificación, por tanto, es imposible de realizar (en [9] se puede encontrar un ejemplo).

El Análisis Discriminatorio Lineal (LDA en inglés) consigue una reducción en la dimensionalidad de una especificación de una clase. Fue creado por el estadístico Sir R.A. Fisher [10]. Para poder encontrar la combinación de las características que diferencian mejor las clases, el LDA maximiza la proporción entre las clases y la dispersión de las clases, en vez de maximizar simplemente la dispersión total de las clases.

La idea es simple: una clase debe agrupar su proyección lo más pegada posible, mientras que las diferentes clases existentes deben estar tan alejadas unas de otras como sea posible en la representación de las proyecciones. Esto fue reconocido, también, por los investigadores Belhumeur, Hespanha y Kriegman, que aplicaron el LDA en reconocimiento facial en [11].

- Descripción del algoritmo

Sea X un vector aleatorio con las muestras dibujadas de c clases:

$$X = \{X_1, X_2, \dots, X_c\}$$
$$X_i = \{x_1, x_2, \dots, x_n\}$$

Las matrices de dispersión S_B y S_W se calculan como:

$$S_B = \sum_{i=1}^{c} N_i(\mu_i - \mu)(\mu_i - \mu)^T$$
$$S_W = \sum_{i=1}^{c} \sum_{x_j \in X_i} (x_j - \mu_i)(x_j - \mu_i)^T$$

donde μ es la media total:

$$\mu = \frac{1}{n} \sum_{i=1}^{n} x_n$$

y μ_i es la media de la clase $i \in \{1,2, \dots, c\}$:

$$\mu_i = \frac{1}{|X_i|} \sum_{x_j \in X_i} x_j$$

El algoritmo clásico Fisher ahora busca, para una proyección W, maximizar el criterio de separación entre clases:

$$W_{opt} = \arg\max_W \frac{|W^T S_B W|}{|W^T S_W W|}$$

Siguiendo la referencia [11], se obtiene una solución para esta optimización del problema, resuelta por el Problema de Autovalor General:

$$S_B v_i = \lambda_i S_W v_i$$
$$S_W^{-1} S_B v_i = \lambda_i v_i$$

Queda un problema por resolver: El rango de S_W es al menos $(N - c)$, con N muestras y c clases. En los problemas del patrón de reconocimiento, el número de muestras N es casi siempre menor que la dimensión de los datos de entrada (el número de píxeles), así que la matriz de dispersión S_W se convierte en singular (ver [12]). En [11], se resuelve realizando un PCA en los datos proyectando las muestras en el espacio de dimensiones $(N - c)$. Un LDA puede realizarse a continuación para reducir los datos, ya que S_W ya no es singular.

La optimización del problema, por tanto, puede ser reescrita como:

$$W_{pca} = \arg\max_W |W^T S_T W|$$
$$W_{fld} = \arg\max_W \frac{\left|W^T W_{pca}^T S_B W_{pca} W\right|}{\left|W^T W_{pca}^T S_W W_{pca} W\right|}$$

La transformación de la matriz W, que proyecta la muestra en el espacio dimensional $(c - 1)$, es dada por:

$$W = W_{fld}^T W_{pca}^T$$

2.1.3. Modelo LBPHFace

Los modelos EigenFace y FisherFace toman un enfonque holístico para la identificación fácil. La información es tratada como un vector con un espacio dimensional muy alto [5]. El enfoque realizado por EigenFace maximiza la dispersión total, el cual puede conllevar problemas si la varianza se debe por una fuente externa, ya que los componentes con una varianza máxima sobre todas las clases existentes no son necesariamente útiles para la clasificación [9]. Para preservar alguna información discriminatoria, se aplica el LDA y se optimiza, como se describió para el modelo FisherFace.

Ahora, la idea no es mirar la imagen por completo como un vector con una gran dimensionalidad, sino describir solo las características locales de un objeto. Las características que se extraen de esta forma tendrán una baja dimensionalidad implícitamente. En esto se basa el operador de Patrones Binarios Locales (LBP en inglés). Se basa en el análisis de texturas en 2D.

La idea principal de LBP es resumir la estructura local de una imagen mediante la comparación de cada píxel con sus píxeles vecinos. Se toma un píxel como centro y se limita el valor de los vecinos. Es decir, si la intensidad del píxel central es mayor que su vecino, este se denota con un cero, y si, por el contrario, la intensidad del vecino es mayor o igual a la intensidad del vecino central, entonces se denota con un uno.

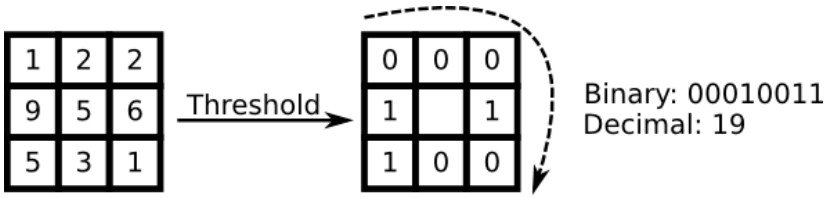


Figura 1. Ejemplo de cómo funciona el operador LBP

- Descripción del algoritmo

Una descripción más formal del operador LBP se puede dar como:

$$LBP(x_c, y_c) = \sum_{p=0}^{P-1} 2^p s(i_p - i_c)$$

con (x_c, y_c) como el píxel central con intensidad i_c; e i_n la intensidad del píxel vecino. s es la función signo definida como:

$$s(x) = \begin{cases} 1 \; si \; x \geq 0 \\ 0 \quad e.o.c. \end{cases}$$

Esta descripción permite capturar detalles con mucha precisión en las imágenes [13]. Para un punto (x_c, y_c), la posición de su vecino (x_p, y_p), $p \in P$, pude ser calculado como:

$$x_p = x_c + R\cos(\frac{2\pi p}{P})$$

$$y_p = y_c - R\sin(\frac{2\pi p}{P})$$

donde R es el radio del círculo que forman los vecinos y P es el número de puntos de muestra.

El operador es una extensión de la codificación original LBP, por eso a veces es llamado LBP Extendido. Si los puntos coordenados en el cícurlo no correponden a las coordenadas de la imagen, el punto debe ser interpolado. OpenCV implementa una interpolación bilineal:

$$f(x,y) \approx [1-x \; x] \begin{bmatrix} f(0,0) & f(0,1) \\ f(1,0) & f(1,1) \end{bmatrix} \begin{bmatrix} 1-y \\ y \end{bmatrix}$$

Por definición, el operador LBP es robusto frente a transformaciones en escala de grises monótonas.

Por último, se incorpora la información espacial en el modelo de identificación facial. La representación

propuesta en [13] es dividir la imagen LBP en m regiones locales y extraer la información del histograma de cada una. El vector de características espaciales mejorado es obtenido mediante la concatenación de los histogramas locales (no una fusión). Estos histogramas son llamados como Histogramas de Patrones Binarios Locales (LBPH en inglés).

3 HERRAMIENTAS EMPLEADAS

Ahora que todos los modelos de identificación facial han sido definidos al detalle, hay que pasar a la descripción de los componentes utizados para la realización de este estudio. Se describirán los elementos empleados a la hora de realizar este estudio desde dos puntos de vista.

Primero, se procederá a la descripción del hardware que se ha usado. Y, a continuación, se describirá el software empleado para la realización del estudio en todos los aspectos, desde el sistema operativo empleado hasta los programas y lenguajes para su formación.

3.1. Descripción Hardware

Para comenzar, la descripción desde nivel de hardware. Para la realización de este TFG se ha empleado un ordenador portátil. Cuenta con un procesador Intel Celeron P4500, a 1.86 GHz. Cuenta con un Dual-Core y una caché L3 de 2 MB.

En términos de memoria, tiene una RAM integrada de 2GB DDR3 y un disco duro de 250 GB. La pantalla, de 15,6”, tiene una resolución de 1366 x 768 píxeles.

3.2. Descripción Software

En la descripción de software empleado, hay que realizar una diferenciación entre dos partes del estudio realizado:

- La programación del estudio, la cual alberga la generación de modelos, el proceso de identificación y la generación de los datos.
- La programación para la representación de los datos que han sido generados a lo largo de los diferentes experimentos realizados.

Hay que diferenciarlos ya que para cada una de estas fases se ha hecho uso de sistemas operativos, programas y lenguajes diferentes.

Para terminar, se hará mención a la base de datos de sujetos utilizada en el estudio.

3.2.1. Descripción Software: Programación del estudio

El Sistema Operativo utilizado para la implementación del estudio ha sido Ubuntu 14.04 LTS. Este Sistema Operativo es un sistema Linux abierto. Además, en dicho Sistema Operativo, se ha procedido a la instalación del programa Eclipse, en su versión Eclipse IDE for C/C++ Developers.

3.2.1.1 Lenguaje de programación empleado

El estudio se ha desarrollado en su totalidad en C++. C++ es un lenguaje de programación de alto nivel [14]. Es un lenguaje abierto estandarizado por ISO. Además, se caracteriza por ser un lenguaje compilado y soporta tanto una comprobación dinámica como estática.

Se ha hecho uso del compilador Linux GCC C++ para generar la aplicación del estudio, a través de la cual, se han obtenido los resultados para su evaluación en este TFG. A parte de C++, se ha incorporado el uso de librerías complementarias, las cuales están descritas a continuación.

3.2.1.2 Librerías complementarias

Ya se ha mencionado cuáles son la librerías de Visión Artificial. Dichas librerías son las que proporciona OpenCV [1], aunque no es la única librería utilizada. El estudio realizado también está basado en Qt [15].

- Qt

Se trata de una herramienta para facilitar la programación de interfaces y su generación. Desarrollan librerías para interfaces con compatibilidad multiplataforma. ¿Y por qué usar esta plataforma? Debido a que simplifican la sincronización entre hilos, el diseño a la hora de realizar aplicaciones, el control sobre los temporizadores con mayor precisión…

Gracias a todas esas simplificaciones, la convierten en una herramienta poderosa para desarrolladores y era muy conveniente introducirla en este estudio. Así, se han podido separar el uso de temporizadores y la generación de la interfaz de la programación de la parte de visión. Dicha separación dan como resultado dos hilos independientes, los cuales se sincronizan mediante el uso de señales.

Por último, para la generación de la aplicación del estudio, se utiliza el compilador de C++, como se ha mencionado más arriba. Para realizar la compilación, es necesario crear archivos Makefile que contengan las instrucciones interpretables por el compilador.

Pues bien, esta herramienta es capaz de generar de manera automática dichos archivos, ahorrando el tiempo que conlleva la creación de los mismos cuando se tienen múltiples archivos y clases. En el Anexo B, se muestra un ejemplo de archivo para la generación automática de archivos Makefile.

La versión utilizada en la elaboración del estudio es la 4.8. Se trata de librerías de código abierto, las cuales están disponibles para descargar en su página oficial.

- OpenCV

Como se ha mencionado, se trata de las librerías para la implementación de algoritmos de Visión Artificial. Su nombre completo, Open Source Computer Vision, significa Código Abierto para Visión Artificial. Es gratuito tanto para su uso académico como comercial. Soporta múltiples lenguajes de programación, así como Sistemas Operativos.

Fue diseñado para realizar aplicaciones con una eficiencia computacional y enfocada sobre todo a las aplicaciones en tiempo real. Debido a esto, es usada desde el arte interactivo hasta la inspección de minas. Son incontables la cantidad de aplicaciones en las que se usan estas librerías. Esta versatilidad de aplicaciones se debe al gran número de funciones que forman OpenCV.

Ofrecen más de quinientas funciones que abarca un gran abanico dentro de la Visión Artificial. Detección de objetos, seguir la trayectoria de éstos, clasificación de los objetos, reconocimiento facial… son solo algunos de los ejemplos que ofrece OpenCV. Dependiendo de cómo se combinen, se obtiene la gran cantidad de aplicaciones que se mencionaba antes.

Para la realización de este estudio, solo se utilizará la parte de identificación facial. Creación de los diferentes modelos existentes para la identificación facial, guardar y cargar dichos modelos y el proceso de identificación de los diferentes sujetos de la base de datos. Estas son las funcionalidades de OpenCV sobre las que se desea realizar el estudio.

La versión utilizada para ello es la 2.4.10 y, al igual que Qt, está disponible para su descarga en la página oficial de OpenCV.

3.2.2. Descripción Software: Programación para la representación

Para la programación de la reprensentación de los datos obtenidos durante todo el estudio, se ha utilizado como Sistema Operativo Windows 7. Sobre dicho Sistema Operativo, se ha realizado la instalación del entorno de desarrollo MATLAB [16] en su versión 7.11.0 (R2010b).

MATLAB es un lenguaje de alto nivel. Es capaz de procesar datos matemáticos a una gran velocidad. Debido a la cantidad de ficheros con resultados que se han generado a lo largo del estudio, hace que sea el lenguaje ideal a la hora de representar de forma gráfica y entendible los datos obtenidos.

Se realizará un programa basado en este lenguaje, en el que se proceda a la lectura de los ficheros generados con los resultados, a la síntesis de dichos resultados y a su representación gráfica de los datos según la influencia del tamaño de la imagen y el número de clase. En el capítulo cinco, se explicará con mayor detalle los archivos que han sido programados y que se encuentran en el Anexo C.

3.2.3. Base de Datos de Sujetos Empleada

La base de datos de sujetos utilizada se denomina GUFD [2]. Todas las imágenes utilizadas fueron tomadas en un área común de la Universidad de Glasgow Caledonian (Escocia, Reino Unido). Dichas imágenes se tomaron con luz natural, la cual iluminaba a los sujetos desde arriba y de frente. También se hizo uso de una lámpara fluorescente no direccional para iluminar el área.

Se han cogido 50 sujetos femeninos y 50 sujetos masculinos. Cada sujeto cuenta con tres imágenes diferentes, tomadas con tres cámaras distintas. Las dos primeras fueron tomadas con cámaras digitales. La primera de ellas con una Olympus Camedia C-350 Zoom de 3 megapíxeles y la segunda con una Fugifilm FinePix 0800Zoom de 6 megapíxeles. La tercera imagen fue tomada con una videocámara Panasonic NV-DS29B DS29.

Para la creación de los modelos, se han usado 10, 50 y 100 sujetos, la mitad femeninos y la otra mitad masculinos. Además, dichos modelos fueron creados con una imagen por sujeto o dos imágenes por sujeto, para comprobar la influencia de tener una o más de una imagen por sujeto.

La identificación, independientemente del número de sujetos que formaba el modelo, se ha realizado sobre los 100 sujetos escogidos. En caso de tener una imagen por sujeto, se han identificado las otras dos imágenes que se tenían de cada sujeto. En cambio, cuando se tenían dos imágenes por sujeto, se ha realizado la identificación sobre la imagen que no formaba parte del modelo.

4 IMPLEMENTACIÓN DEL ESTUDIO

Tras conocer las herramientas utilizadas a lo largo de este proyecto, se puede proceder a la explicación del código creado para la implementación del estudio. Dicho estudio es una aplicación construida en C++ formada por tres clases:

- QConfig: Encargada de la lectura del fichero de configuración de la interfaz y de la escritura de los datos generados por el estudio.
- QFormPrincipal: Tiene el manejo de la interfaz. Realiza la comunicación con el hilo encargado de la Visión Artificial y tiene el control sobre los temporizadores.
- QProceso: Contiene la programación de Visión Artificial. Se encarga de crear los modelos, guardarlos, cargarlos y realizar la identificación cuando la interfaz se lo pida.

Toda la documentación del código se encuentra en el Anexo A. Este capítulo hace un resumen de ello, explicando el proceso paso a paso.

Al comenzar la aplicación, la función inicial main procede a la lectura del fichero de configuración a través de la clase QConfig. Una vez cargados los datos, se inicializa la clase QFormPrincipal, dando paso a la creación de la interfaz, cuyo funcionamiento está detalladamente explicado en el siguiente punto.

4.1. Programación de la Interfaz

Como se ha dicho anteriormente, primero se crea la interfaz. Con los valores de configuración ya cargados, se definen los parámetros de la ventana de la interfaz. Este paso va seguido de la inicialización del hilo proceso, nombre con el que se declara la clase QProceso en QFormPrincipal.

A continuación, se definen las conexiones entre el hilo principal y el hilo proceso. Estas conexiones representan las comunicaciones entre los hilos, es decir, qué señal activa qué función. También se procede a conectar los temporizadores (timer, timer2 y timer3) con el hilo principal.

Los temporizadores son una herramienta muy útil, ya que permiten medir el tiempo en la que se realiza una tarea específica. En este estudio, la relación de los temporizadores con las funciones que desempeñan se muestra en la siguiente figura:

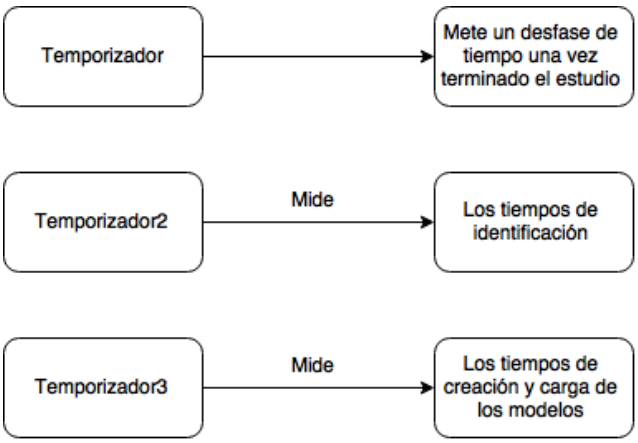


Figura 2. Relación temporizadores-tareas desempeñadas

Una vez finalizado el contructor de la clase QFormPrincipal, se procede al resto de fases del estudio. La primera de ellas es la fase de la creación de todos los modelos identificadores implicados en este estudio. Durante la duración de esta fase, la interfaz tiene este aspecto:

Figura 3. Interfaz durante la creación de modelos.

Creados y guardados todos los modelos, la siguiente fase es la de carga de un modelo:

Figura 4. Interfaz durante la carga de un modelo.

Y, después de cargar el modelo, se procede a la identificación de la base de datos de sujetos:

Figura 5. Interfaz durante la identificación de la base de datos.

La imagen anterior se mantiene mientras se procede a la identificación de los 100 sujetos que componen la base. Estas dos fases, cargando modelo e identificando, se repite hasta que se hayan identificado los 100 sujetos con cada uno de los modelos. En este estudio, en total hay seis modelos. Identificados los sujetos de un modelo, se guardan los datos generados en un fichero.

Cuando se termina la identificación de los seis modelos, se procede al cierre del estudio. Se guardan los tiempos de creación y carga de los modelos en un archivo y se procede al cierre de la aplicación:

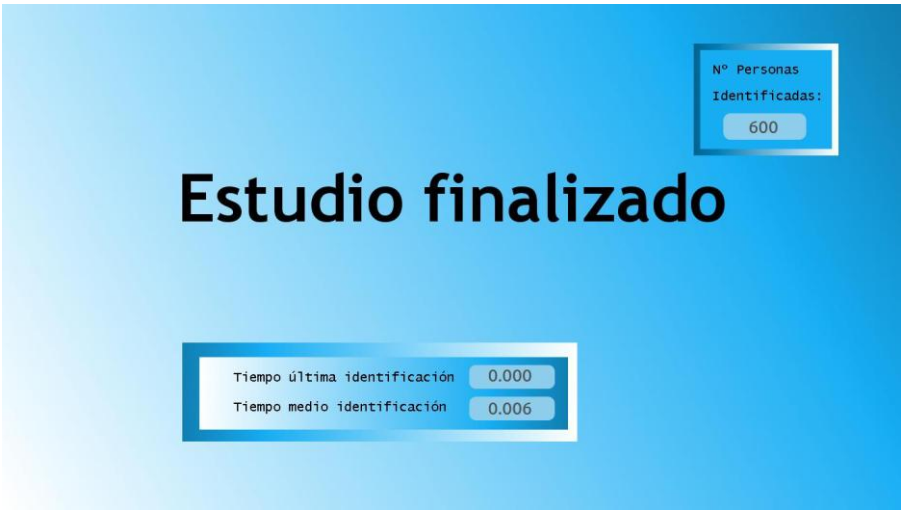


Figura 6. Interfaz cuando se finaliza el estudio.

En el útlimo punto de este capítulo se hablará con mayor profundidad de los ficheros con los datos generados a lo largo del estudio.

A lo largo de las fases anteriores es cuando se da la comunicación entre los hilos. El hilo principal, durante la creación de modelos, emite seis señales distintas para crear cada uno de los seis modelos de identificación facial. Durante la creación de cada modelo, se activa el temporizador3 (timer3) para medir los tiempos de creación de los modelos.

Creado el modelo, el hilo proceso emite una señal de fin de creación de modelo. Recibida por el hilo principal, reestablece el temporizador3 y emite la señal para crear el siguiente modelo. Una vez creados los seis modelos, se puede proceder a la carga del primer modelo.

Para ello, el hilo principal emite una señal, al igual que ocurría en el caso de creación, pero indicando la carga del modelo. Y, como antes, el tiempo de carga también es controlado por el temporizador3. Aunque, a diferencia de la creación de modelos, cuando el hilo proceso emite la señal de fin de carga, empieza la identificación de los 100 sujetos.

Realizada la identificación de todos los sujetos, se procede a la carga del siguiente modelo de la misma forma. Un esquema de la comunicación entre hilos se muestra en la siguiente figura:

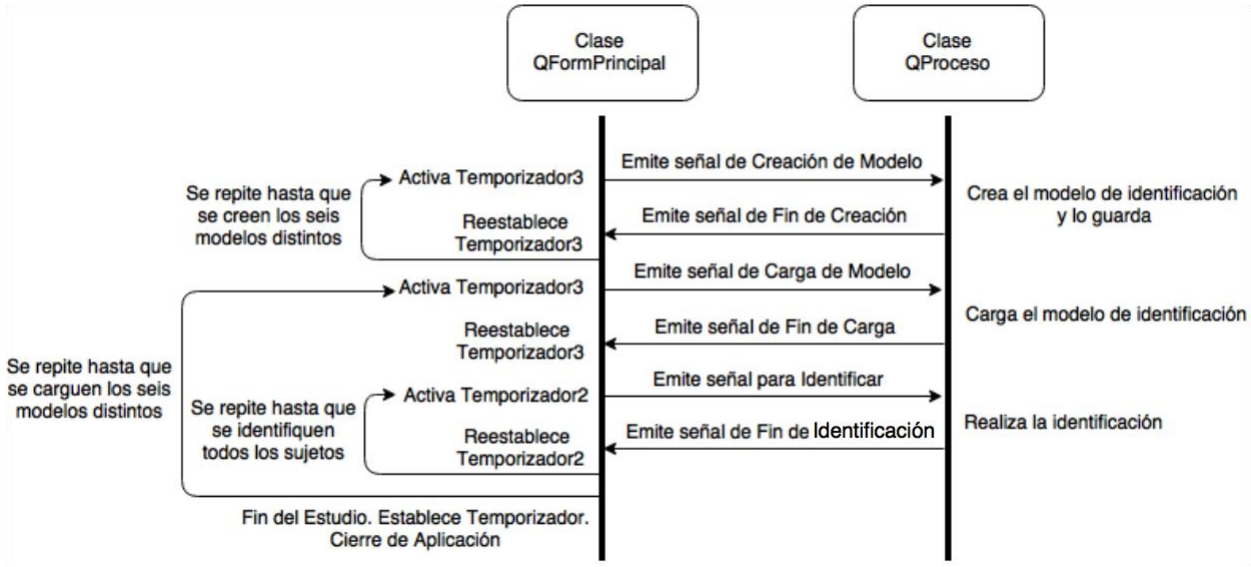


Figura 7. Esquema comunicación hilos.

Explicado todo el código que conlleva la interfaz, queda explicar la parte de Visión Artificial, la clase QProceso.

4.2. Programación de Visión Artificial

El procesado de la parte de Visión Artificial está definido en las funciones que forman la clase QProceso. Son tres las tareas de Visión Artificial que realiza QProceso:

- Crear y guardar los seis modelos.
- Cargar los modelos.
- Realizar la fase de identificación.

Los seis modelos empleados son: el modelo EigenFace, el modelo FisherFace y el modelo LBPHFace, todos en su forma simple y con estructura en árbol.

- Descripción de los modelos en su forma simple

Todos los modelos utilizan las mismas funciones para desempeñar las tareas arriba descritas. La única diferencia existente está en la creación del modelo. Cada modelo se crea de una forma distinta:

- Creación del modelo EigenFace:

```
model = createEigenFaceRecognizer();
```

- Creación del modelo FisherFace:

```
model = createFisherFaceRecognizer();
```

- Creación del modelo LBPHFace:

```
model = createLBPHFaceRecognizer();
```

Realizada la creación, se tiene un modelo vacío. Para tener un modelo capaz de identificar, se debe entrenar. Se utiliza la función train:

```
model->train(images,labels);
```

Los parámetros de entrada corresponden a:

- images: vector de Mat. Contiene las muestras utilizadas por el modelo de los diferentes sujetos.
- labels: vector de enteros. Relaciona el número de la clase correspondiente a un sujeto con la imagen de muestra que se encuentra en la misma posición dentro del vector de Mat.

Ya se tiene un modelo capaz de identificar. Se puede proceder, por tanto, a guardar el modelo con la función save, así podrá cargarse y ser utilizado para identificar:

```
model->save(fn_model);
```

fn_model hace referencia al nombre con el que se desea guardar el modelo.

Al tenerse el modelo completo creado, puede emitirse la señal de fin de creación del modelo. Un ejemplo de los pasos de la función completa para la creación de un modelo se puede ver a continuación (se utiliza como ejemplos el modelo EigenFace):

```
73      //Creacion del modelo
74      model = createEigenFaceRecognizer();
75      model->train(images,labels);
76
77      //Se guarda el modelo creado
78      model->save(fn_model);
79
80      //Emision de la señal de fin de creacion del modelo
81      emit SignalFinCreacionModelo();
82      model->~FaceRecognizer();
```

Para cargar un modelo, se hace uso de la función load. La función load tiene como parámetro de entrada el mismo que la función save. Hay que crear un modelo vacío antes de proceder a la carga. Así, se tiene para cargar un modelo la función siguiente:

```
111      //Carga del modelo
112      model = createEigenFaceRecognizer();
113      model->load(fn_model);
114
115      //Emision de la señal de fin de carga del modelo
116      emit SignalFinCargaModelo();
```

Al igual que cuando se creaba, una vez está cargado el modelo y listo para su utilización emite una señal indicándolo.

Por último, para realizar la identificación se utiliza la función predict:

```
model->predict(test_im,predictedLabel,confidence);
```

Los parámetros de entrada son:

- test_im: Imagen del sujeto a identificar.
- predictedLabel: Entero donde se guardará la predicción de la clase a la que pertenece el sujeto a identificar.
- confidence: Decimal en el que se guardará la confianza con la que se ha hecho la predicción.

Como pasaba en las funciones anteriores, al terminar la identificación se emite una señal indicándolo, tal y como se muestra en la función completa del proceso de identificación:

```
131      //Identificacion del sujeto
132      predictedLabel=-1;
```

```
133     confidence=0.0;
134     model->predict(test_im,predictedLabel,confidence);
135
136     //Emision de la señal de fin de identificacion del sujeto
137     emit SignalFinIdentificacion(confidence,predictedLabel);
```

- Descripción de los modelos con estructura en árbol

Al igual que en la forma simple, se crea el árbol, se guarda, se carga y se utiliza para identificar. Aunque ahora ya no se tiene un único modelo, sino un vector de modelos. Cada modelo representa una de las ramas que conforman el árbol.

El árbol está formado por ramas con modelos binarios y ramas con modelos de cinco clases. Esto es debido a que, para la formación de los modelos, se han usado 10, 50 y 100 clases identificadoras, siendo estos múltiplos de dos y de cinco.

Todas los modelos que conforman las ramas de un árbol están creados con el mismo tipo de modelo. Así, si se trata de una estructura árbol con modelo EigenFace, todos los modelos creados para las diferentes ramas serán de este tipo de modelo. Lo mismo ocurre para la estructura en árbol con modelo FisherFace y la estructura en árbol con modelo LBPHFace.

Crear un modelo que compone una rama del árbol es como crear un modelo simple. Siguen los mismos pasos. La primera diferencia reside en que, a medida que se avanza en la creación de los modelos que componen el árbol, cada vez se tiene un menor número de sujetos para formar el modelo. En el siguiente ejemplo se muestra cuántos sujetos componen cada modelo de cada rama para el caso de 10 clases identificadoras:

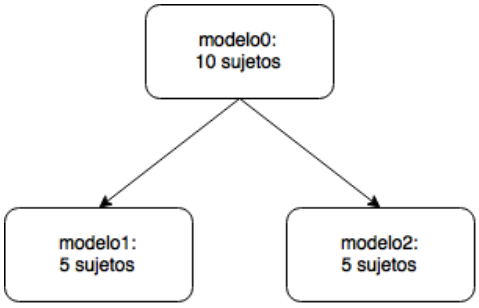


Figura 8. Modelos creados para un árbol con 10 clases identificadoras.

La otra diferencia reside en que, en el caso de los modelos binarios, como su nombre indica, solo se tienen dos clases identificadoras, y los modelos con cinco clases, pues tienen cinco clases identificadoras. Al final, dependiendo de la predicción de cada modelo, se irán recorriendo las ramas del árbol hasta llegar a la identificación del sujeto, tal como se puede ver en la siguiente figura, en el caso de tener 10 sujetos para formar el modelo:

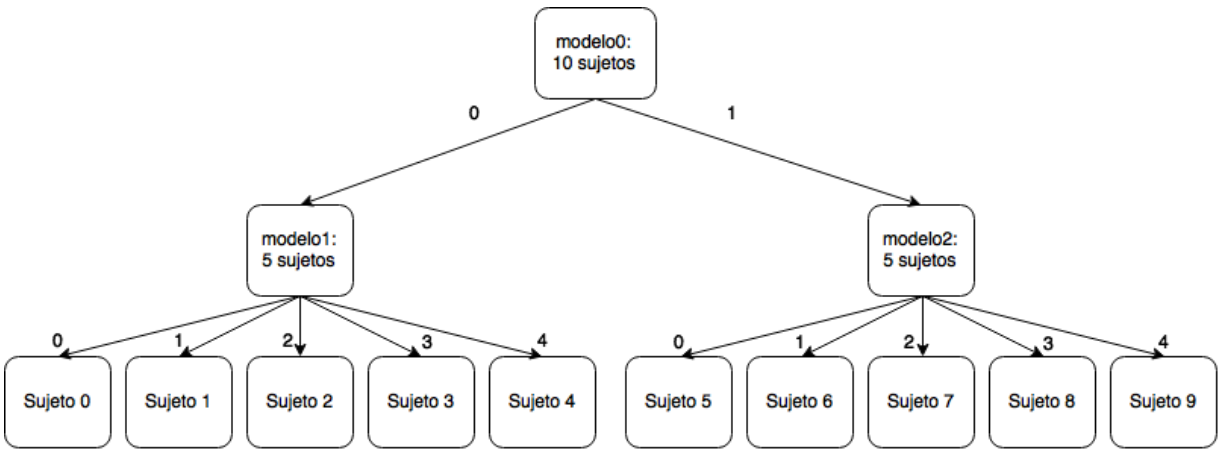


Figura 9. Estructura en árbol con 10 clases identificadoras.

Todo lo anteriormente explicado ha sido implementado en funciones. A continuación, se muestra el diagrama que sigue la función de creación el árbol:

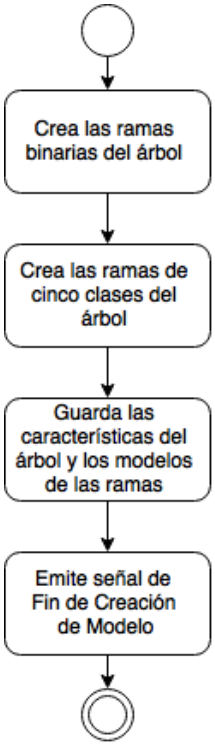


Figura 10. Diagrama de creación de estructura en árbol.

El proceso de carga es el siguiente:

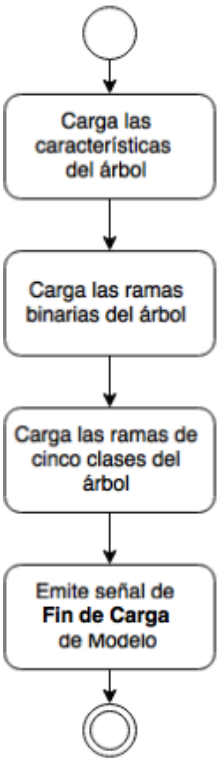


Figura 11. Diagrama de carga de estructura en árbol.

Por último, el proceso de identificación llevado a cabo por la estructura en árbol:

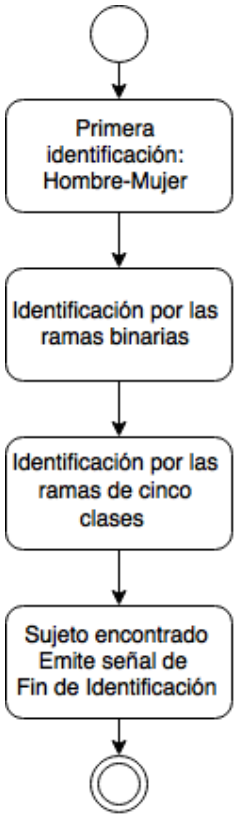


Figura 12. Diagrama de identificación de estructura en árbol.

4.3. Datos generados

Los experimientos que se han realizado han sido los siguientes:

- Influencia del tamaño de la imagen en los modelos. Se han utilizado cuatro tamaños distintos en las muestras tomadas de los sujetos: 50x50 píxeles, 100x100 píxeles, 150x150 píxeles y 200x200 píxeles.
- Influencia del número de sujetos para formar el modelo de identificación. Se han usado 10, 50 y 100 sujetos para la creación de los modelos.
- Influencia del número de imágenes por sujeto. Cómo influye el tener una o dos imágenes por sujeto en los modelos.
- Influencia en el uso de un modelo simple o con estructura en árbol con cada tipo de modelo existente. Se van a evaluar seis modelos distintos.

En total se tienen 144 variantes a estudiar, todas ellas integradas en este estudio. Debido a todas las variantes, se ha generado un gran número de datos. Estos datos han sido escritos en ficheros. A continuación se presentan los ficheros que se han obtenido:

- 24 ficheros que contienen los tiempos de creación y carga de los modelos.
- 216 ficheros que contienen los datos generados por la identificación de los sujetos femeninos.
- 216 ficheros que contienen los datos generados por la identificación de los sujetos masculinos.

Todos estos ficheros han sido escritos por la clase QConfig.

El primer tipo de fichero es el que contiene los tiempos de creación y de carga. Dichos tiempos son los tiempos de los seis modelos que se han creado y cargado durante una de las ejecuciones de la aplicación. Este fichero se genera tras la ejecución de las variantes de tamaño, número de sujeto para los modelos y número de imágenes por sujeto.

Los ficheros con los datos debido a la identificación de mujeres han sido generados cada vez que un modelo identificaba a los cincuenta sujetos femeninos que se tienen en la base de datos utilizada. Se generan más de 144 ficheros ya que cuando se usa solo una imagen por sujeto, se procede a identificar las dos imágenes que se tienen de cada sujeto. Debido a esto, es por lo que se obtienen 216 archivos.

Los ficheros con los datos de las identificaciones masculinas se generan de la misma manera que los ficheros con los datos de identificación femenina. Por ello, se obtiene el mismo número de archivos para el caso femenino y para el caso masculino.

Cada tipo de fichero es generado por funciones distintas definidas en la clase QConfig. Por tanto, la clase QConfig contiene en total cuatro funciones:

- Función encargada de la lectura del archivo de configuración de la interfaz.
- Función encargada de la escritura del archivo con los tiempos de creación y de carga de los modelos.
- Función encargada de la escritura del archivo con los datos de identificación de los sujetos femeninos.
- Función encargada de la escritura del archivo con los datos de identificación de los sujetos masculinos.

5 REPRESENTACIÓN DE RESULTADOS

Teniendo todos los datos recogidos en ficheros, el siguiente paso es extraer dichos resultados, sintetizarlos y respresentarlos gráficamente. Así, en el último capítulo, se comentarán las conclusiones obtenidas por la realización de este estudio.

Debido a la gran cantidad de ficheros con datos a procesar, se hace uso de MATLAB [16] para realizar estas tareas. MATLAB es capaz de procesar grandes cargas computacionales de una forma rápida. Además, dispone de herramientas de representación gráfica.

A lo largo de este capítulo, se explicará el código creado en MATLAB, adjunto en el Anexo C; los datos obtenidos para la representación gráfica y, por último, se incluirán todas las gráficas que contendrán todos los resultados obtenidos.

5.1. Programación en MATLAB

Para la representación gráfica en MATLAB, han sido creado los siguientes ficheros, todos adjuntos en el Anexo C:

- analizaFicheroTiempos.m: Abre todos los ficheros que contienen los tiempos de creación y de carga de los modelos, haciendo uso de la función leeFicheroTiempos. Realiza la representación de dichos tiempos en gráficas. Dichas gráficas enfrenta los datos de los distintos tamaños de imagen utilizados, el número de sujetos empleados en el modelo, si se tiene una o dos imágenes por sujeto y la estructura simple con la estructura en árbol.
- leeFicheroTiempos.m: Fichero que contiene la función leeFicheroTiempos. Devuelve en una matriz los datos contenidos en el fichero de entrada.
- analizaFicheroPersona50x50.m: Realiza la lectura y síntesis de todos los datos obtenidos de las identificaciones, tanto los ficheros de sujetos masculinos como los ficheros de sujetos femeninos, con un tamaño de muestra de 50x50 píxeles. Para ello, hace uso de la función leeFicheroPersona.
- analizaFicheroPersona100x100.m: Realiza la lectura y síntesis de todos los datos obtenidos de las identificaciones, tonto los ficheros de sujetos masculinos como los ficheros de sujetos femeninos, con un tamaño de muestra de 100x100 píxeles. Para ello, hace uso de la función leeFicheroPersona.

- analizaFicheroPersona150x150.m: Realiza la lectura y síntesis de todos los datos obtenidos de las identificaciones, tanto los ficheros de sujetos masculinos como los ficheros de sujetos femeninos, con un tamaño de muestra de 150x150 píxeles. Para ello, hace uso de la función leeFicheroPersona.
- analizaFicheroPersona200x200.m: Realiza la lectura y síntesis de todos los datos obtenidos de las identificaciones, tanto los ficheros de sujetos masculinos como los ficheros de sujetos femeninos, con un tamaño de muestra de 200x200 píxeles. Para ello, hace uso de la función leeFicheroPersona.
- leeFicheroPersona.m: Fichero que contiene la función leeFicheroPersona. Devuelve en una matriz los datos contenidos en el fichero de entrada.
- dibujaDatos.m: Realiza la representación gráfica de los datos obtenidos por los ficheros analizaFicheroPersona50x50.m, analizaFicheroPersona100x100.m, analizaFicheroPersona150x150.m, analizaFicheroPersona200x200.m. Se deben ejecutar estos ficheros antes de ejecutar dibujaDatos.m. Representa las gráficas de los siguientes parámetros:
    - Tiempo de identificación medio: Representa los tiempos de identificación medio de un mismo tipo de modelo frente a los distintos tamaños de imagen utilizados, el número de sujetos empleados en el modelo, si se tiene una o dos imágenes por sujeto y la estructura simple con la estructura en árbol.
    - Porcentaje de falsos positivos: Representa el porcentaje de falsos positivos de un mismo tipo de modelo frente a los distintos tamaños de imagen utilizados, el número de sujetos empleados en el modelo, si se tiene una o dos imágenes por sujeto y la estructura simple con la estructura en árbol. Un falso positivo ocurre cuando la predicción realizada no coincide con la identidad de la clase identificadora del sujeto.
    - Porcentaje de falsos positivos en mujeres: Representa el porcentaje de falsos positivos en mujeres de un mismo tipo de modelo frente a los distintos tamaños de imagen utilizados, el número de sujetos empleados en el modelo, si se tiene una o dos imágenes por sujeto y la estructura simple con la estructura en árbol. Un falso positivo en mujeres ocurre cuando un sujeto masculino es considerado como un sujeto femenino.
    - Porcentaje de falsos positivos en hombres: Representa el porcentaje de falsos positivos en hombres de un mismo tipo de modelo frente a los distintos tamaños de imagen utilizados, el número de sujetos empleados en el modelo, si se tiene una o dos imágenes por sujeto y la estructura simple con la estructura en árbol. Un falso positivo en hombres ocurre cuando un sujeto femenino es considerado como un sujeto masculino.
    - Porcentaje de tasa de acierto: Representa la tasa de acierto de un mismo tipo de modelo frente a los distintos tamaños de imagen utilizados, el número de sujetos empleados en el modelo, si se tiene una o dos imágenes por sujeto y la estructura simple con la estructura en árbol. Un acierto ocurre cuando la predicción realizada coincide con la identidad de la clase identificadora del sujeto.
    - Porcentaje de tasa de acierto en mujeres: Representa la tasa de acierto en mujeres de un mismo tipo de modelo frente a los distintos tamaños de imagen utilizados, el número de sujetos empleados en el modelo, si se tiene una o dos imágenes por sujeto y la estructura simple con la estructura en árbol. Un acierto en mujeres ocurre cuando la predicción realizada coincide con la identidad de la clase identificadora de un sujeto femenino.
    - Porcentaje de tasa de acierto en hombres: Representa la tasa de acierto en hombres de un mismo tipo de modelo frente a los distintos tamaños de imagen utilizados, el número de sujetos empleados en el modelo, si se tiene una o dos imágenes por sujeto y la estructura simple con la estructura en árbol. Un acierto en hombres ocurre cuando la predicción realizada coincide con la identidad de la clase identificadora de un sujeto masculino.

5.2. Resultados obtenidos

Serán agrupados por el tipo de parámetro que representan. La leyenda que aparece en todas las gráficas significa:

- S – 1:1. Estructura simple del modelo – Una muestra por sujeto.
- A – 1:1. Estructura en árbol del modelo – Una muestra por sujeto.
- S – 2:1. Estructura simple del modelo – Dos muestras por sujeto.
- A – 2:1. Estructura en árbol del modelo – Dos muestras por sujeto.

- Tiempos de creación

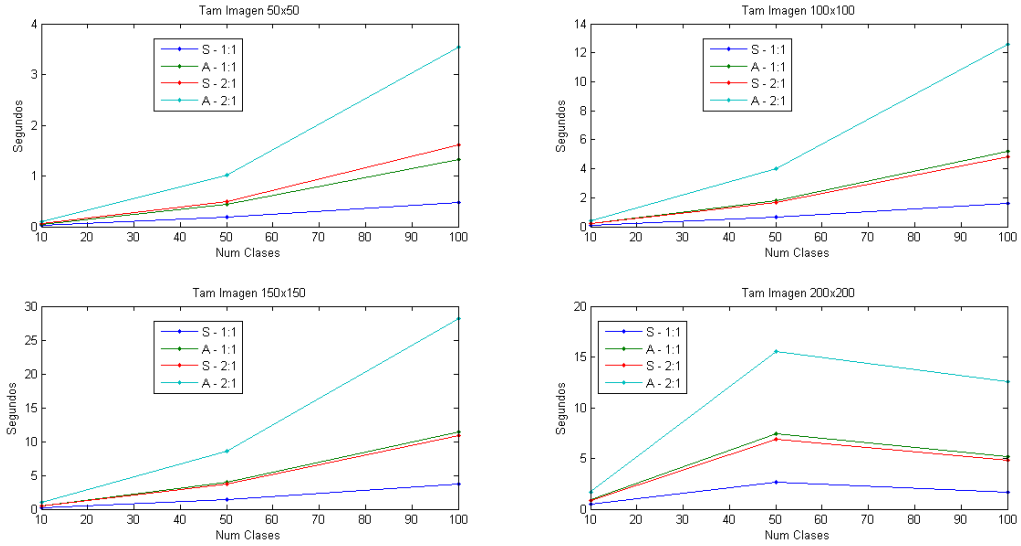


Figura 13. Tiempos de creación del modelo EigenFace.

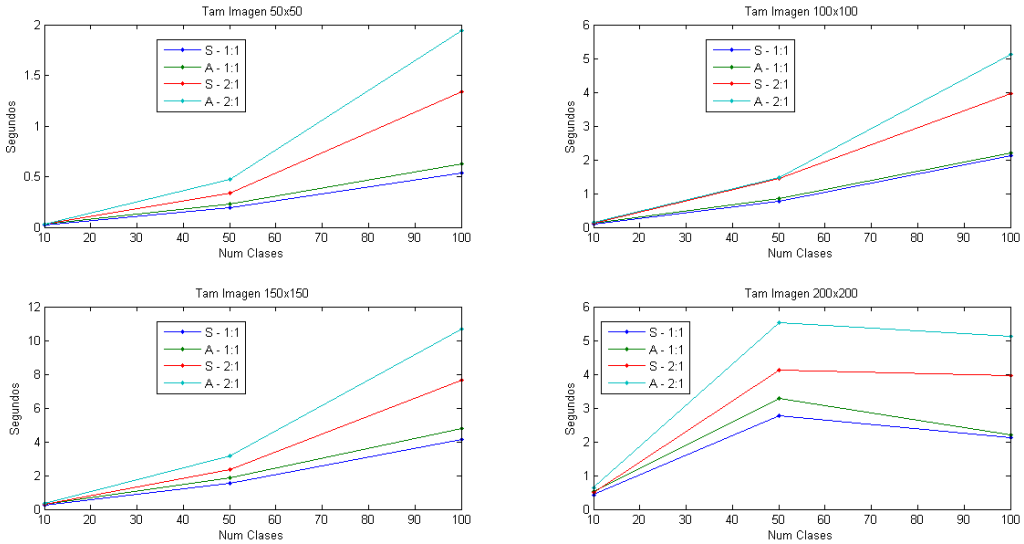


Figura 14. Tiempos de creación del modelo FisherFace.

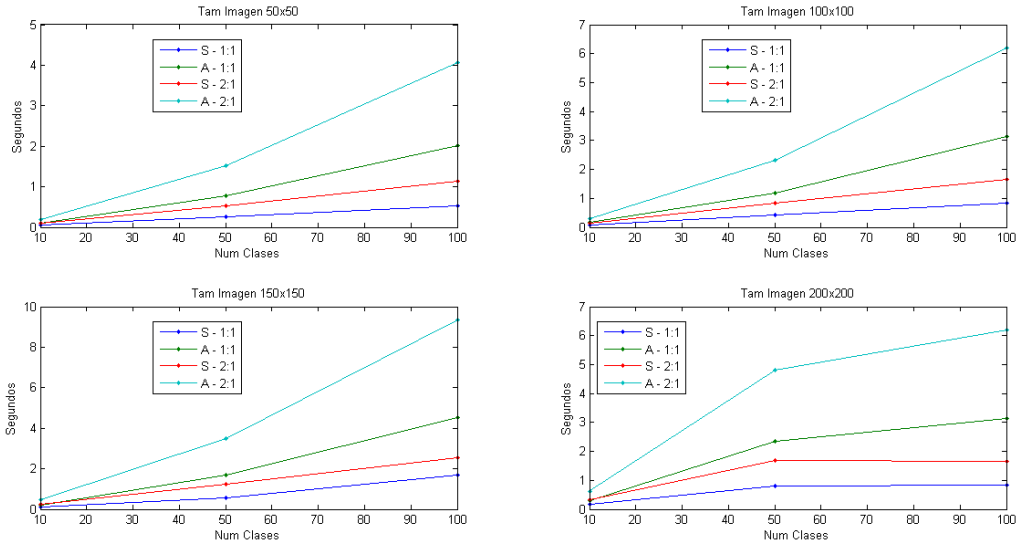


Figura 15. Tiempos de creación del modelo LBPHFace.

- Tiempos de carga

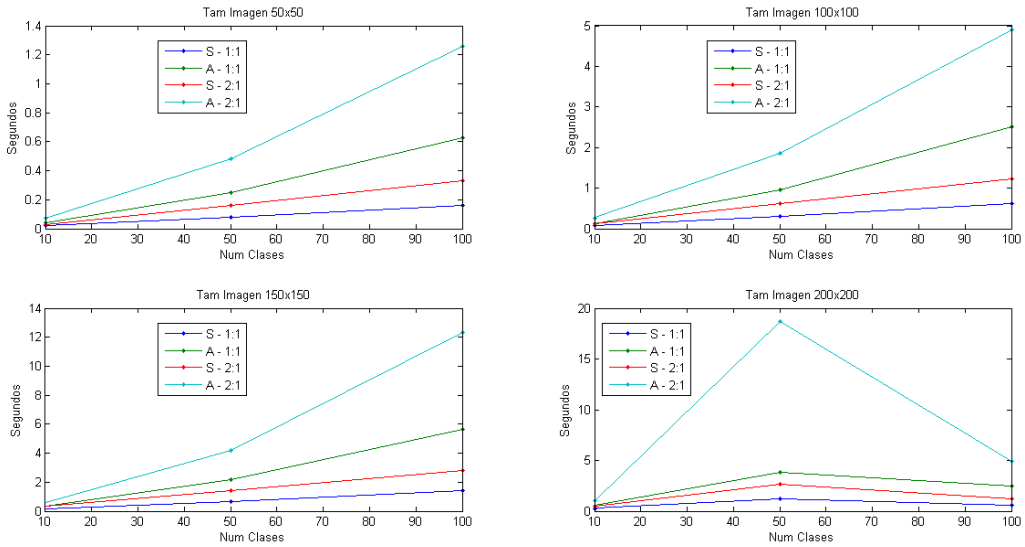


Figura 16. Tiempos de carga del modelo EigenFace.

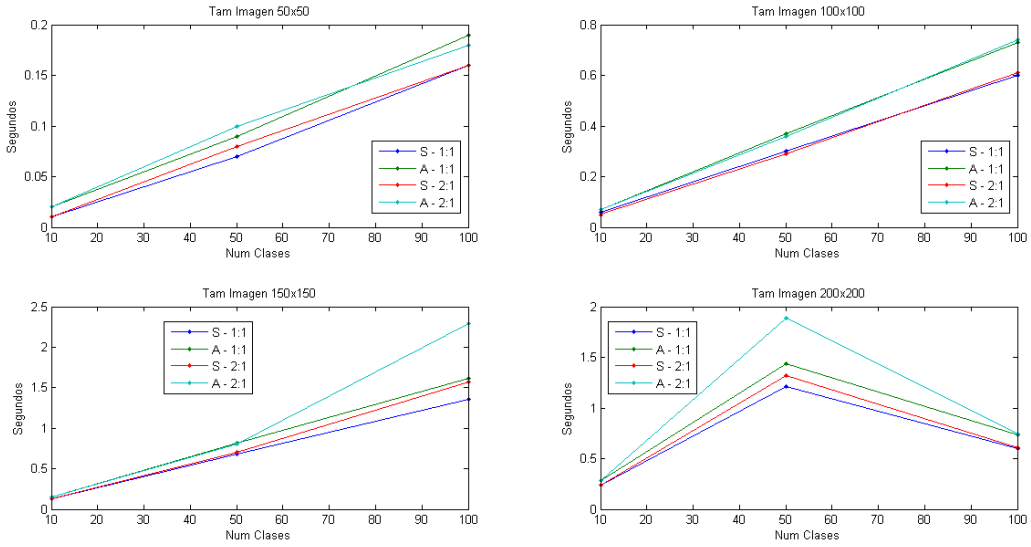


Figura 17. Tiempos de carga del modelo FisherFace.

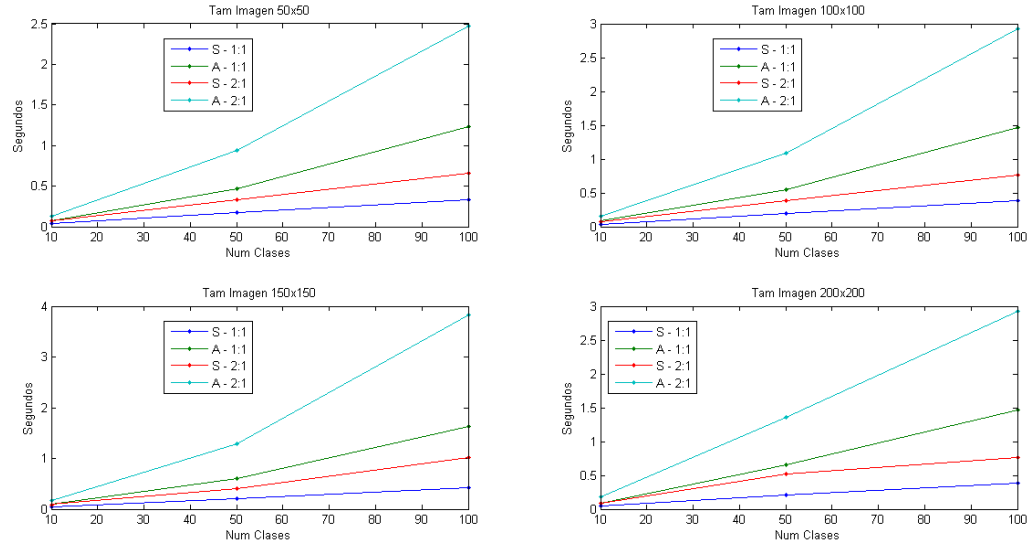


Figura 18. Tiempos de carga del modelo LBPHFace.

- Tiempos medios de identificación

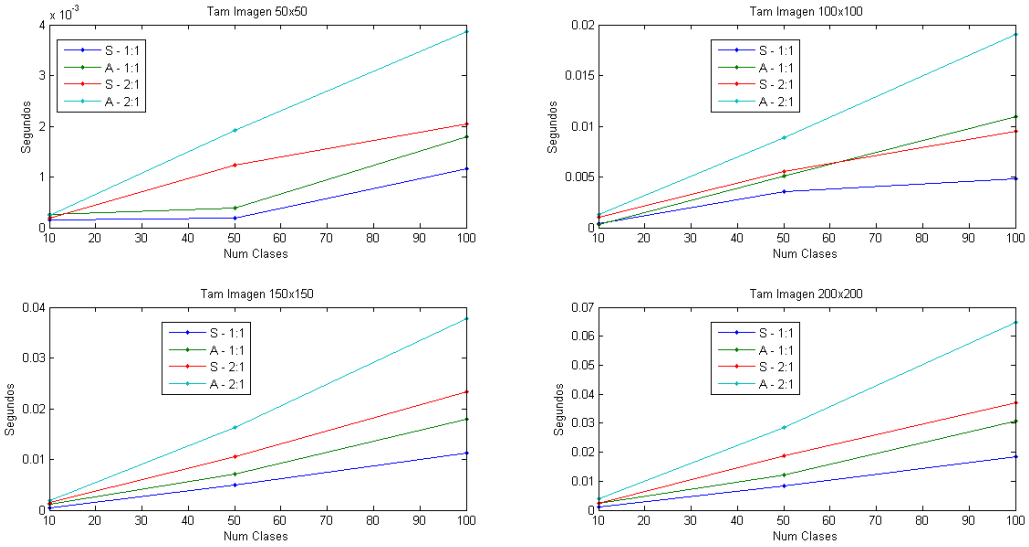


Figura 19. Tiempos medios de identificación del modelo EigenFace.

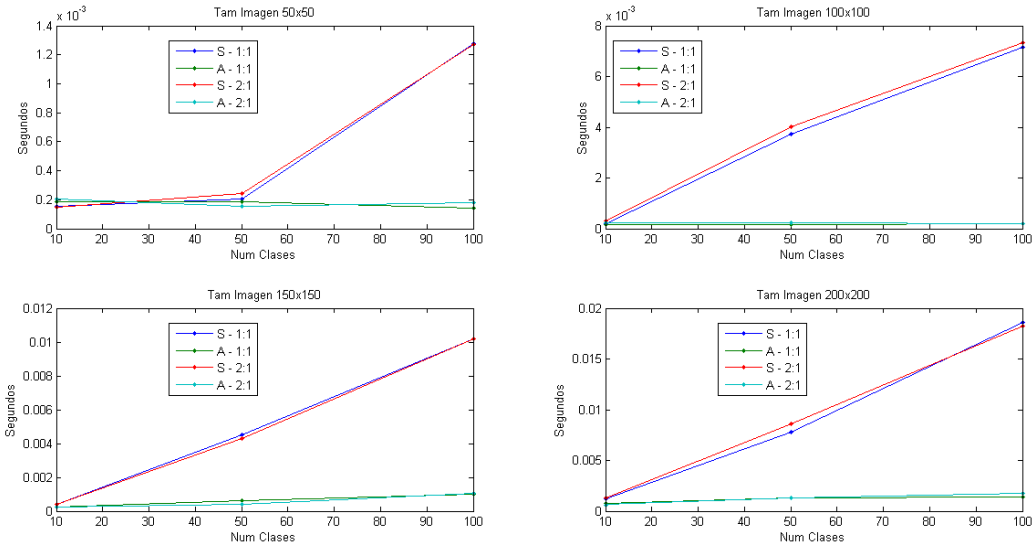


Figura 20. Tiempos medios de identifiación del modelo FisherFace.

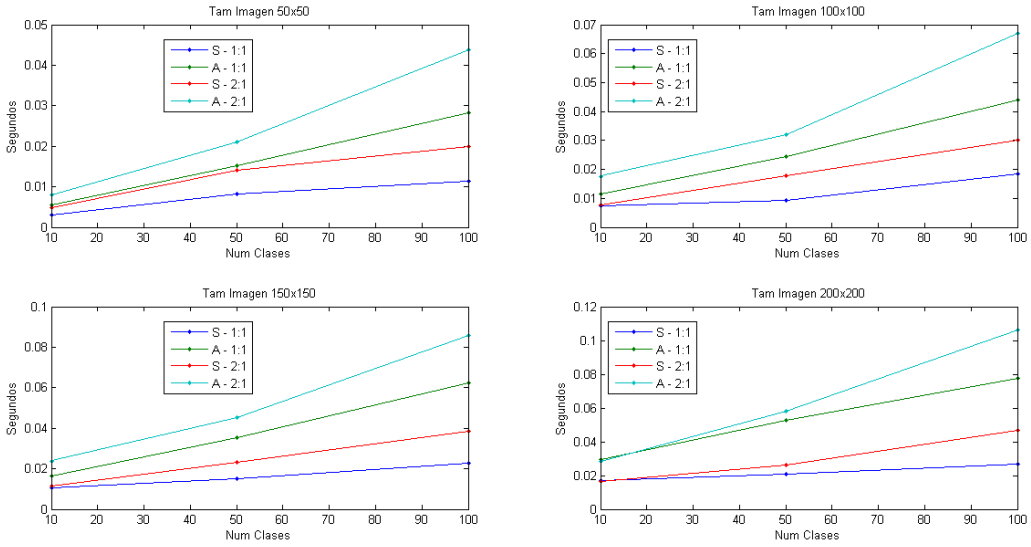


Figura 21. Tiempos medios de identificación del modelo LBPHFace.

- Porcentajes de falsos positivos

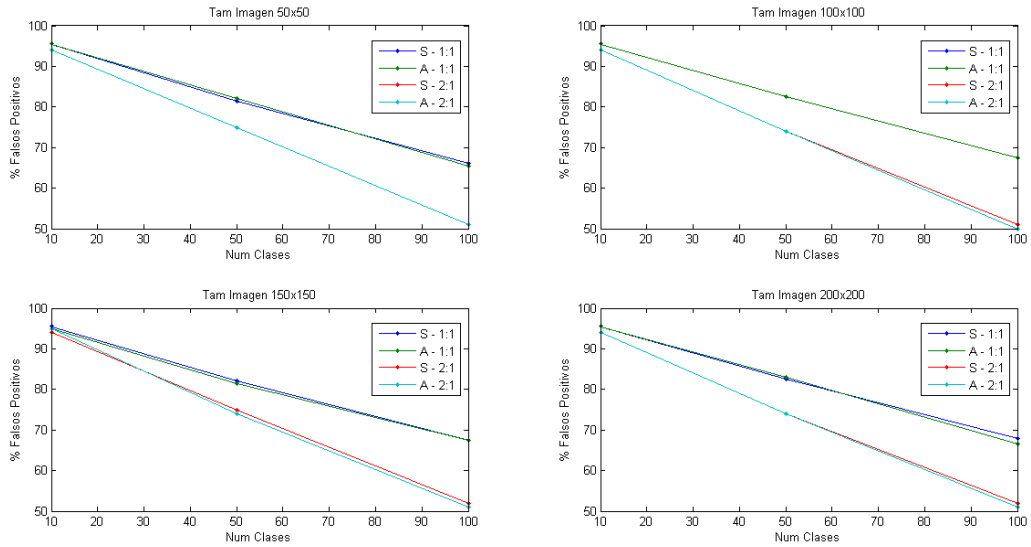


Figura 22. Porcentajes de falsos positivos del modelo EigenFace.

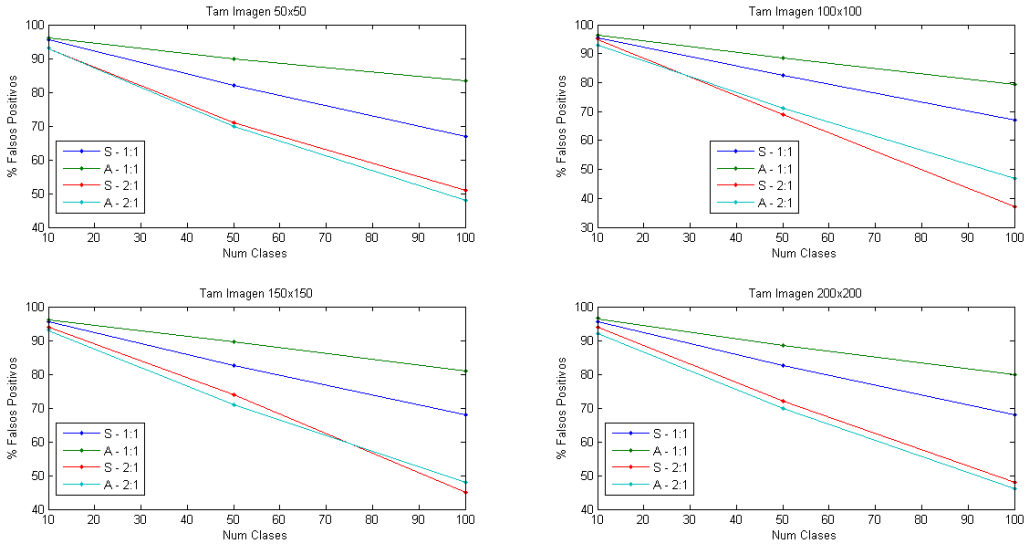


Figura 23. Porcentajes de falsos positivos del modelo FisherFace

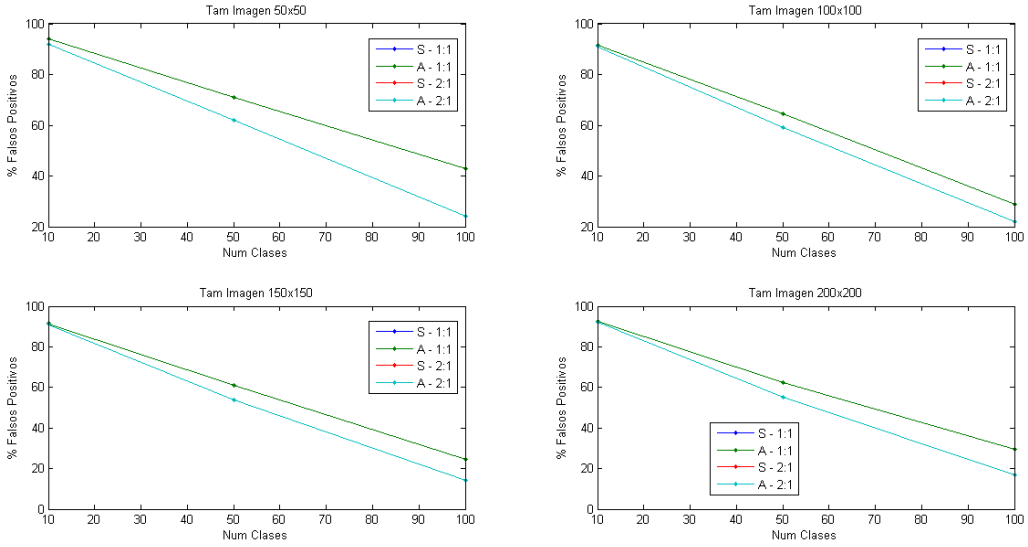


Figura 24. Porcentajes de falsos positivos del modelo LBPHFace.

- Porcentajes de falsos positivos en mujeres

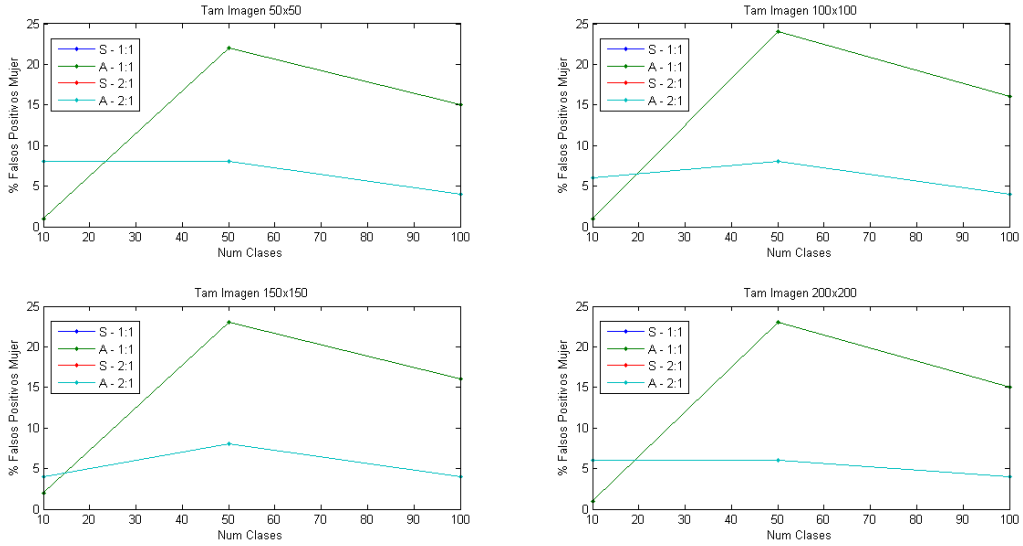


Figura 25. Porcentajes de falsos positivos en mujeres del modelo EigenFace.

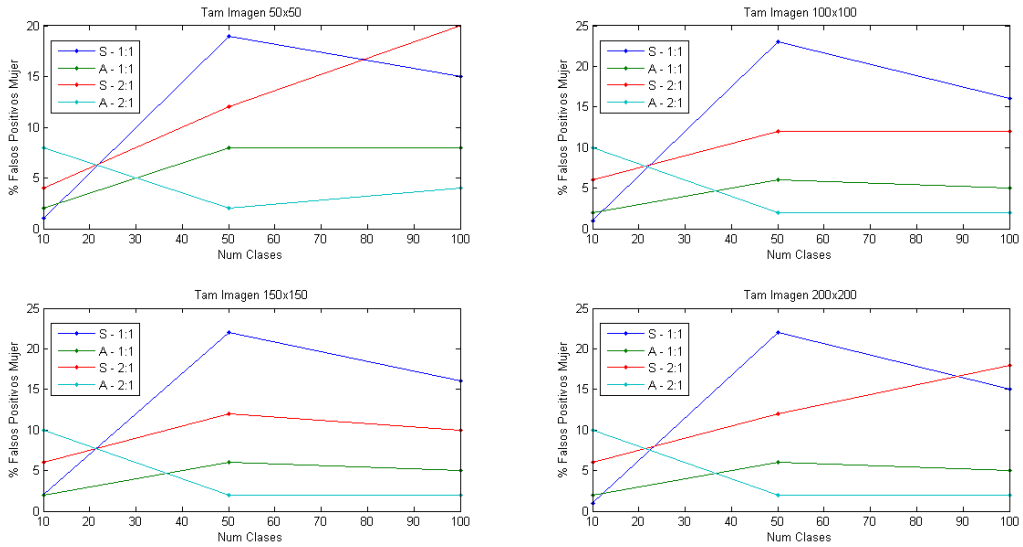


Figura 26. Porcentajes de falsos positivos en mujeres del modelo FisherFace.

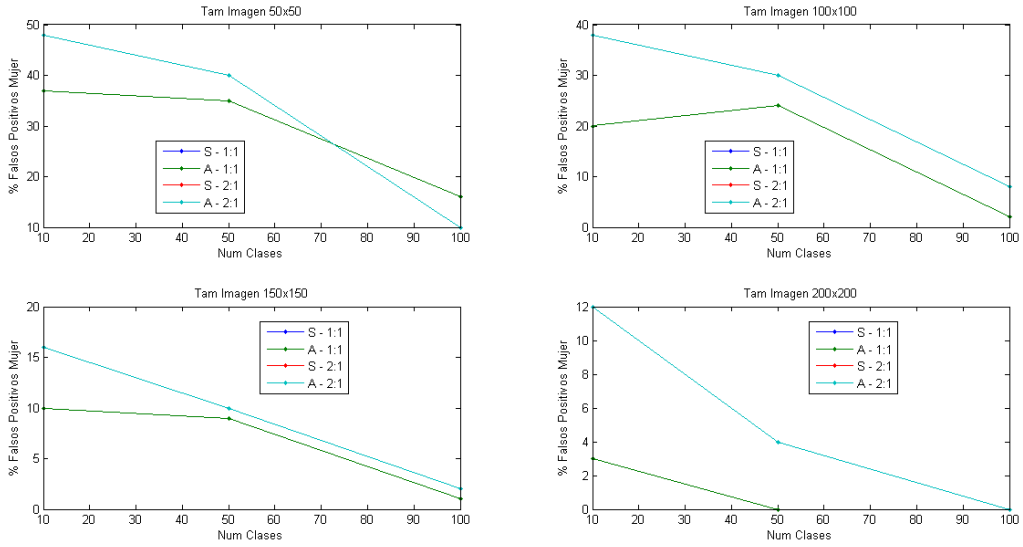


Figura 27. Porcentajes de falsos positivos en mujeres del modelo LBPHFace.

- Porcentajes de falsos positivos en hombres

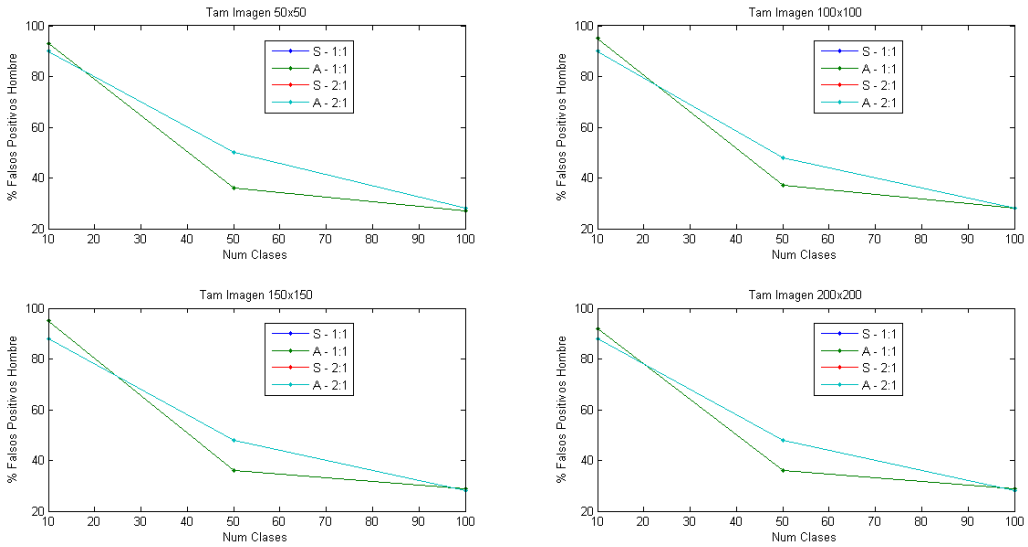


Figura 28. Porcentajes de falsos positivos en hombres del modelo EigenFace.

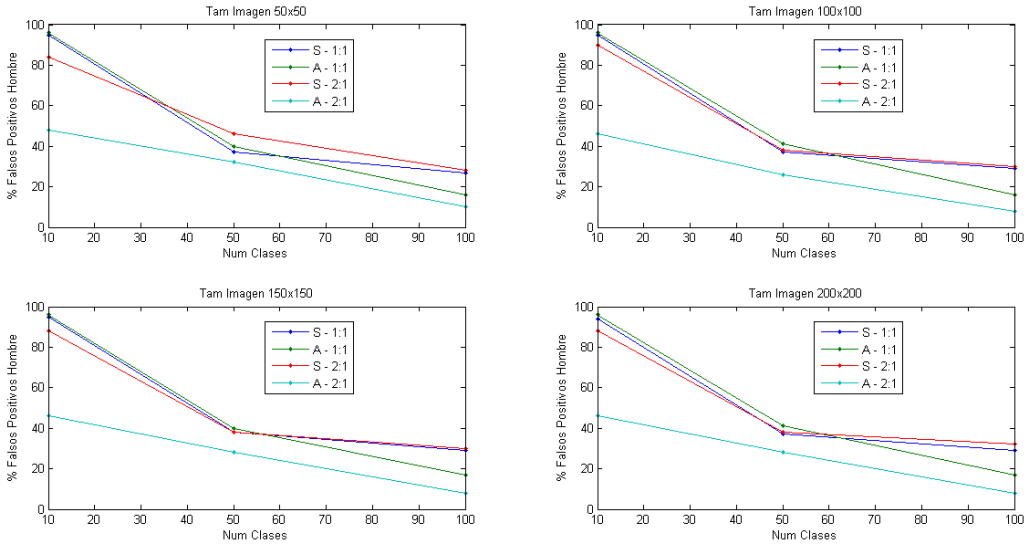


Figura 29. Porcentajes de falsos positivos en hombres del modelo FisherFace.

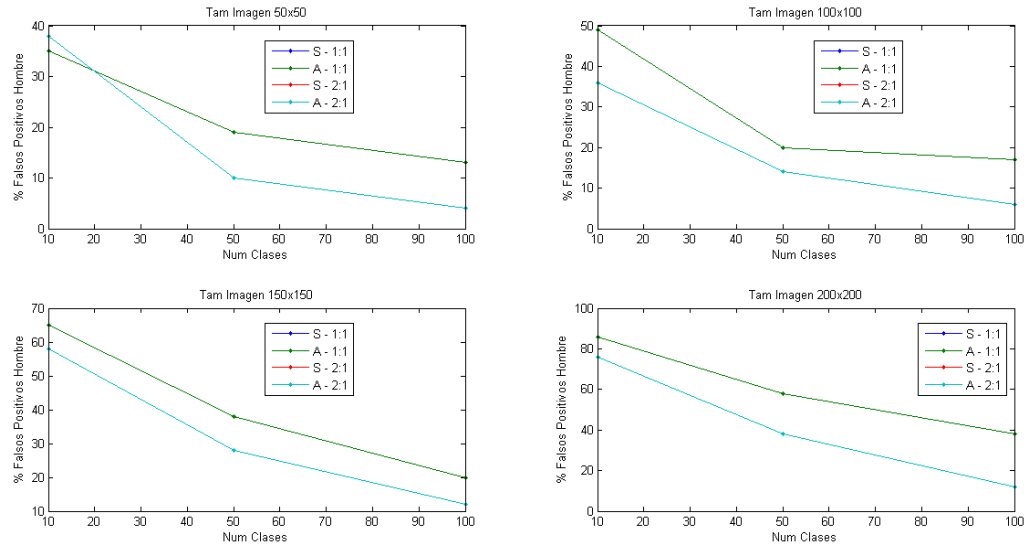


Figura 30. Porcentajes de falsos positivos en hombres del modelo LBPHFace

- Porcentajes de tasas de acierto

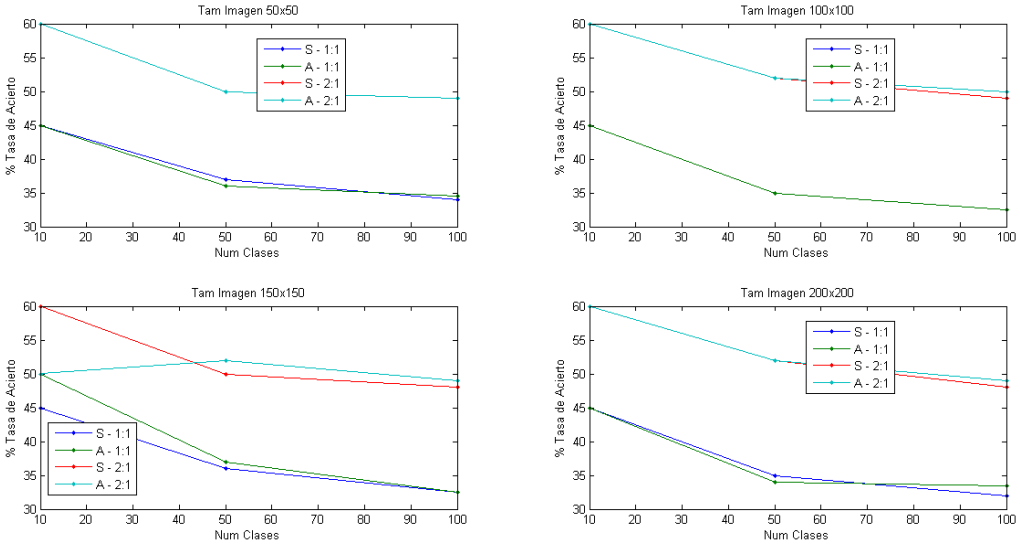


Figura 31. Porcentajes de tasas de acierto del modelo EigenFace.

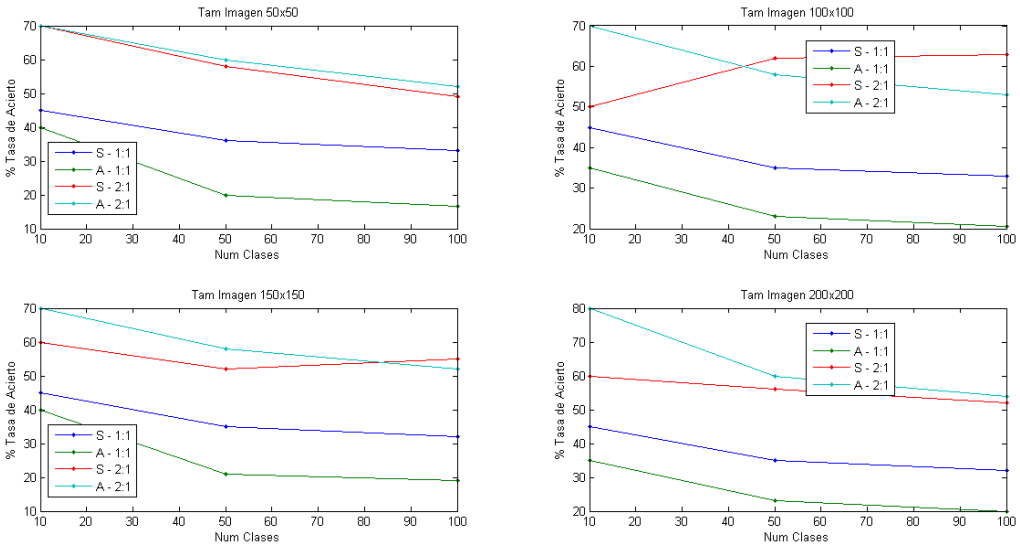


Figura 32. Porcentajes de tasas de acierto del modelo FisherFace.

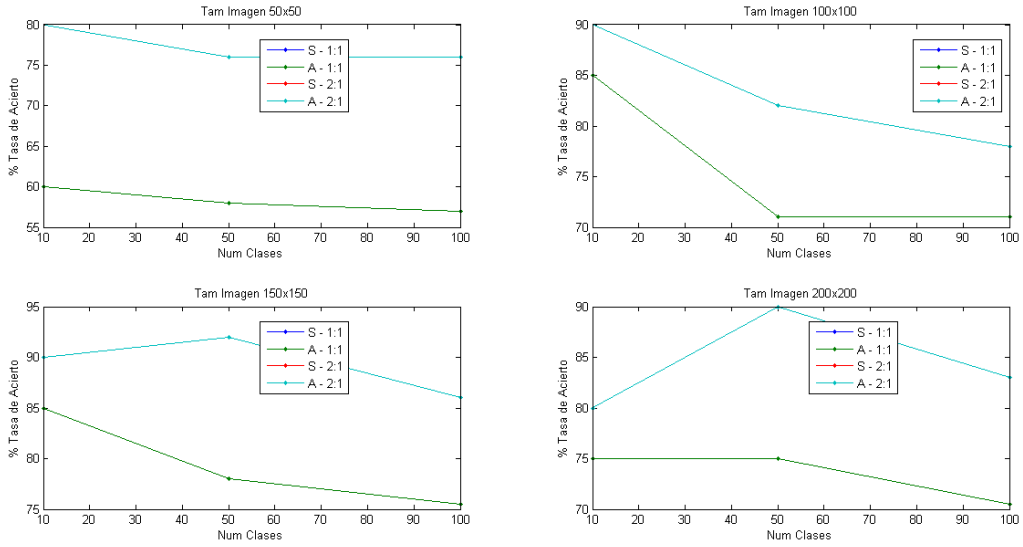


Figura 33. Porcentajes de tasas de acierto del modelo LBPHFace.

- Porcentajes de tasas de acierto en mujeres

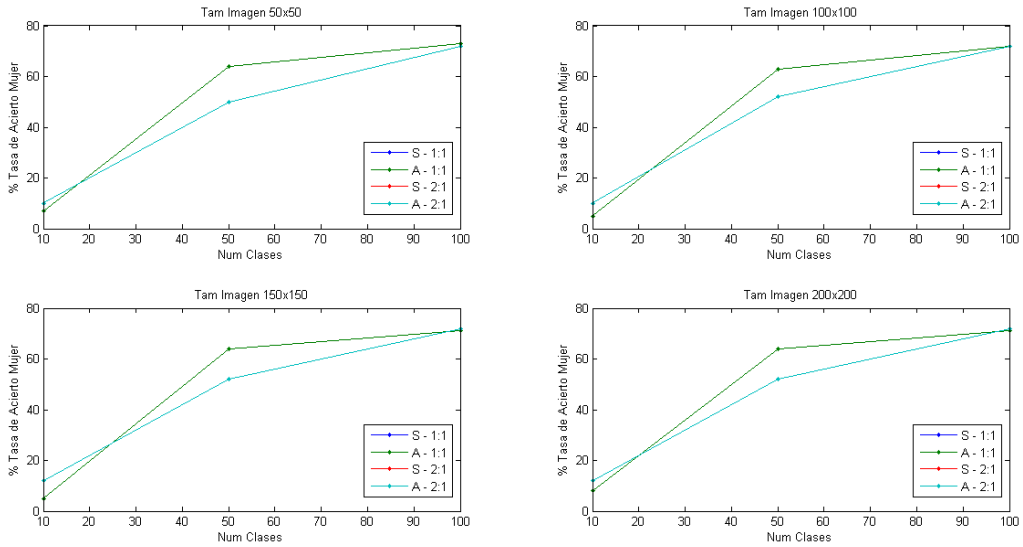


Figura 34. Porcentajes de tasas de acierto en mujeres del modelo EigenFace.

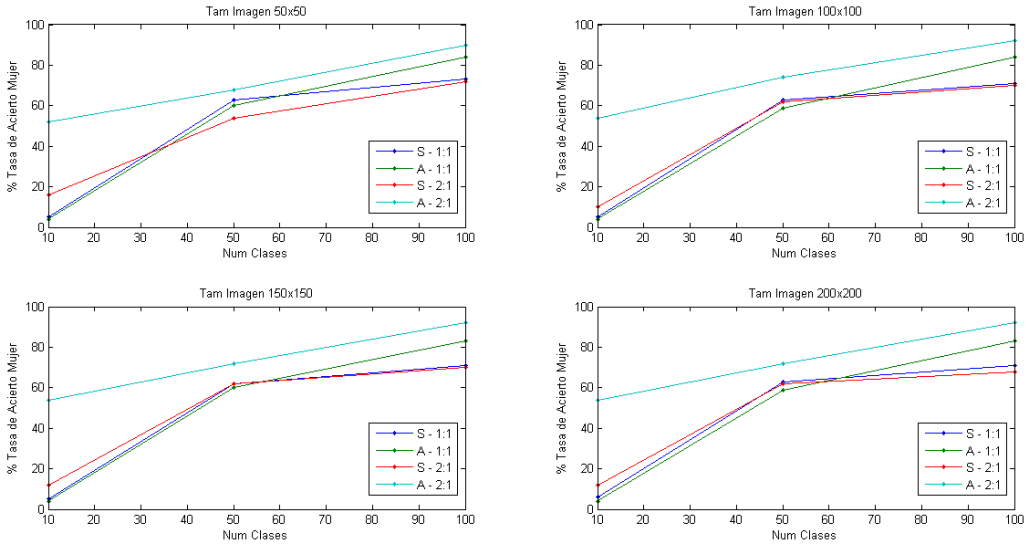


Figura 35. Porcentajes de tasas de acierto en mujeres del modelo FisherFace.

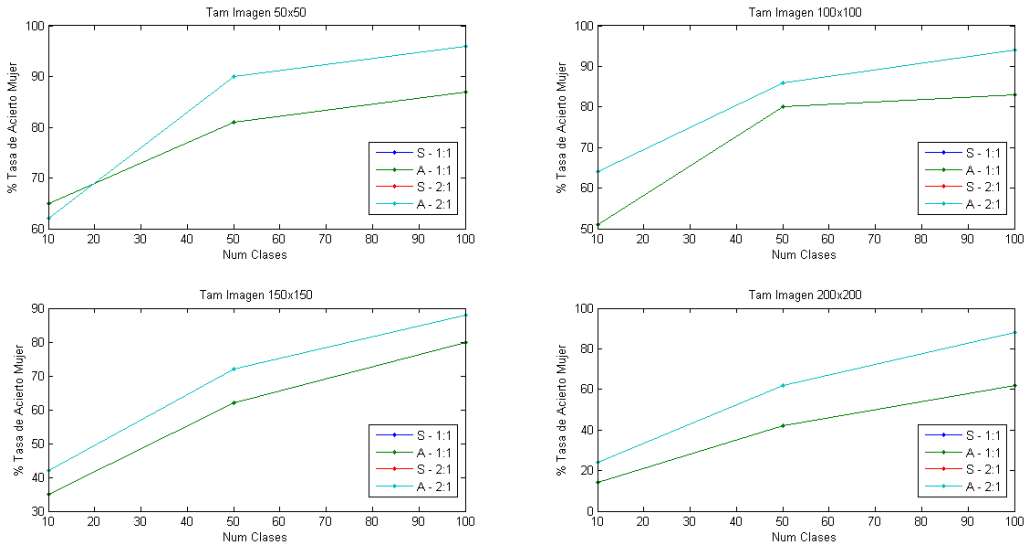


Figura 36. Porcentajes de tasas de acierto en mujeres del modelo LBPHFace.

- Porcentajes de tasas de acierto en hombres

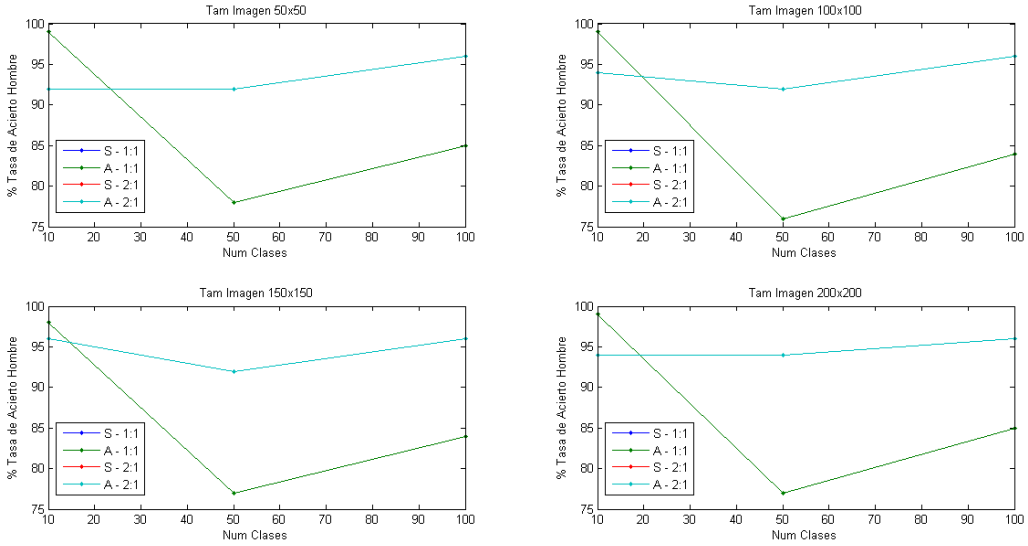


Figura 37. Porcentajes de tasas de acierto en hombres del modelo EigenFace.

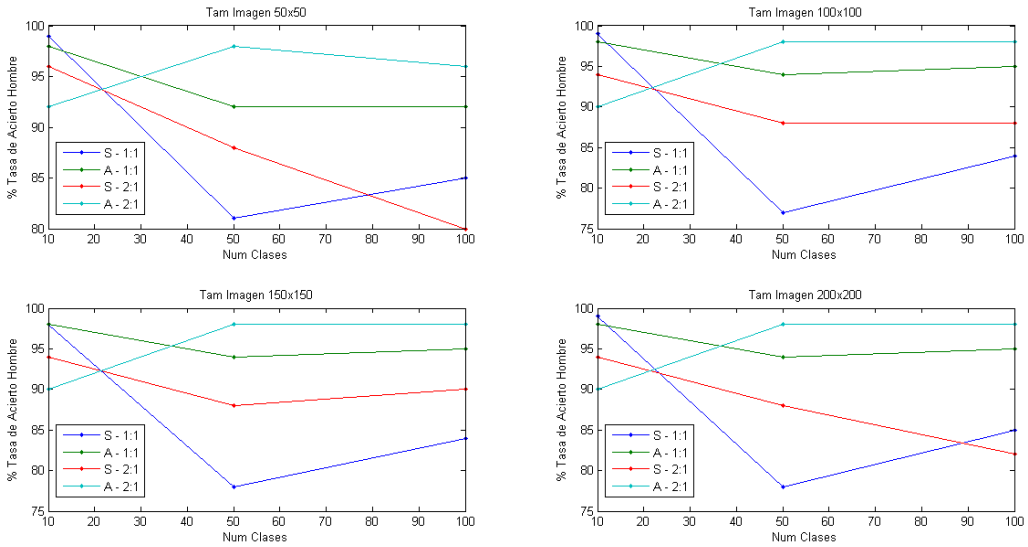


Figura 38. Porcentajes de tasas de acierto en hombres del modelo FisherFace.

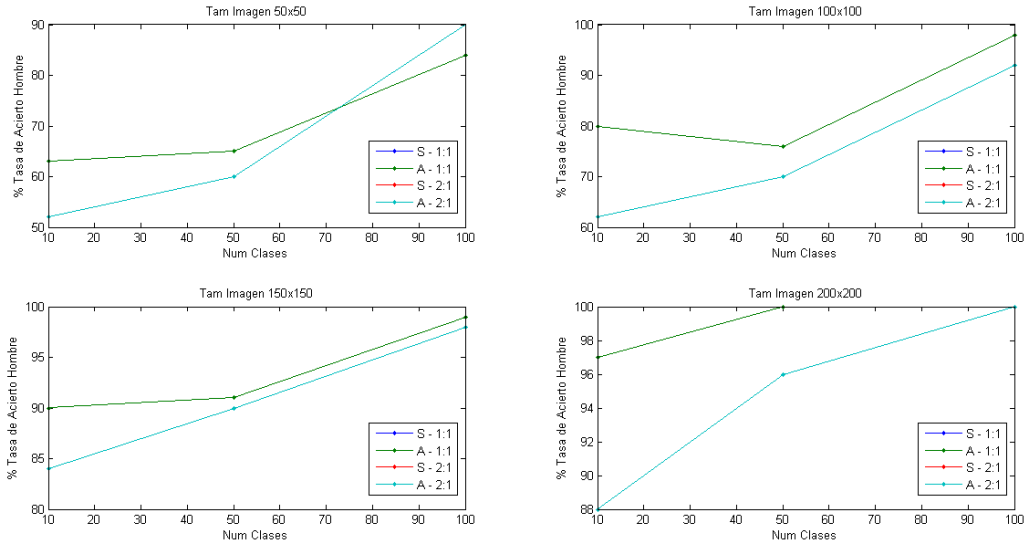


Figura 39. Porcentajes de tasas de acierto en hombres del modelo LBPHFace.

6 ASPECTOS FINALES

Obtenidos todos los resultados de forma gráfica, durante este último capítulo se procederá a comentar las conclusiones más relevantes extraídas. Por último, y para concluir esta memoria, se añadirán las posibles vías de futuro.

6.1. Conclusiones

Comparando los tiempos de creación y de carga de los modelos, se observa que el primero es mayor que el segundo. Se tarda más en crear un modelo que en que éste sea cargado. La acción de crear un modelo solo es necesaria la primera vez. Una vez éste se ha guardado, puede ser cargado y utilizado tantas veces como se quiera. Además, cuando se carga, se puede dejar funcionando el modelo tanto tiempo como se quiera. No es necesario estar continuamente recargándolo para realizar la identificación.

El siguiente parámetro de interés es el tiempo medio de identificación. Es muy importante, ya que dependiendo del tipo de aplicación que se quiera desarrollar, se deberá usar un modelo u otro. Por ejemplo, si se quiere una aplicación en la que se identifiquen personas sin necesidad de que sea en tiempo real, se puede escoger el método más fiable aunque sea más lento. Por el contrario, si se desea realizar una aplicación en tiempo real con varias identificaciones por imagen, se deberá escoger aquella cuyo tiempo de identificación medio no supere el tiempo de toma de muestra de la cámara.

Otro aspecto a tener en cuenta para el tiempo medio de identificación es el número de sujetos que forman la base de datos a utilizar. Cuanto mayor sea el número de sujetos, mayor será este tiempo. Por ello, llega un momento en la que el modelo simple necesita realizar tantas iteraciones que supera el tiempo medio de identificación del modelo con estructura en árbol. Este caso solo se consigue observar en el modelo FisherFace, pero es extrapolable a los demás modelos. No se llega a observar debido a que el número de sujetos es muy bajo.

Queda por evaluar la fiabilidad de los modelos. Lo primero que se observa es cómo se mejora la tasa de acierto en casi todos los modelos cuando se tienen dos muestras por sujeto. Es un cambio bastante signficativo, ya que, en la gran mayoría, dicha tasa mejora entre un 10% y un 20%. Claro está, mayor número de muestras implica un mayor tiempo medio de identificación. Aún así, en el estudio llevado a cabo, el máximo tiempo medio de identificación es de 0.1 segundos, valor bastante asumible cuando se traten de aplicaciones de identificación en tiempo real.

Respecto a la variación del tamaño de las muestras de los sujetos tomadas para la identificación, en los modelos EigenFace y FisherFace, no varía mucho la tasa de acierto obtenida. En cambio, en el modelo LBPHFace afecta en la tasa de acierto el tener menor o mayor tamaño. De hecho, en las imágenes con un tamaño de 150x150 píxeles se han obtenido los mejores resultados en cuanto a tasa de acierto se refiere.

Esto puede ser observado en la implementación de los modelos matemáticamente. Mientras que EigenFace y FisherFace trabajan con proyecciones de las muestras totales, LBPHFace realiza comparaciones píxel a píxel con sus vecinos. Por ello, si la muestra es demasiado chica, identifica peor, ya que no consigue obtener tanta información como de una muestra de mayor tamaño. Y, si las muestras son demasiado grandes, tiene demasiada redundancia, haciendo que también identifique peor.

No es posible tener un modelo que realice el 100% de las identificaciones de forma correcta. Esto es debido a que, al estar utilizando biométrica, se debe convivir con los falsos positivos. Aún así, se han conseguido resultados bastante buenos. El mejor modelo realizando la identificación ha sido el modelo LBPHFace. Es el modelo que consigue mayor tasa de acierto en todas las variantes. A continuación, se muestran los resultados con muestras de 150x150 píxeles:

- Cuando se tenían 10 sujetos formando el modelo de identificación:
  - De las 20 imágenes a identificar cuando se tenía una muestra por sujeto, se han conseguido identificar correctamente 17 de esas imágenes, es decir, un 85%.
  - De las 10 imágenes a identificar cuando se tenían dos muestras por sujeto, se han conseguido identificar correctamente 9 de esas imágenes, es decir, un 90%.
- Cuando se tenían 50 sujetos formando el modelo de identificación:
  - De las 100 imágenes a identificar cuando se tenía una muestra por sujeto, se han conseguido identificar correctamente 78 de esas imágenes, es decir, un 78%.
  - De las 50 imágenes a identificar cuando se tenían dos muestras por sujeto, se han conseguido identificar correctamente 46 de esas imágenes, es decir, un 92%.
- Cuando se tenían 100 sujetos formando el modelo de identificación:
  - De las 200 imágenes a identificar cuando se tenía una muestra por sujeto, se han conseguido identificar correctamente 150 de esas imágenes, es decir, un 75%.
  - De las 100 imágenes a identificar cuando se tenían dos muestras por sujeto, se han conseguido identificar correctamente 86 de esas imágenes, es decir, un 86%.

El estudio no se ha limitado solo al estudio de la tasa de acierto en la correcta identificación, sino también se ha querido ver cuál era el modelo capaz de realizar la mejor clasificación por género. Los modelos que han obtenido los mejores resultados realizando la clasificación de sujetos femeninos son los siguientes:

- Cuando se tenían 10 sujetos formando el modelo de identificación: Modelo LBPHFace con una muestra por sujeto y muestras con tamaño 50x50 píxeles.
- Cuando se tenían 50 sujetos formando el modelo de identificación: Modelo LBPHFace con dos muestras por sujeto y muestras con tamaño 50x50 píxeles.
- Cuando se tenían 100 sujetos formando el modelo de identificación: Modelo LBPHFace con dos muestras por sujeto y muestras con tamaño 50x50 píxeles.

En el caso de la clasificación de sujetos masculinos, tanto para 10, 50 y 100 sujetos formando el modelo de identificación, se ha obtenido la misma configuración. El modelo que ha obtenido los mejores resultados ha sido el modelo LBPHFace en su configuración de una muestra por sujeto y muestras con un tamaño de 200x200 píxeles.

6.2. Vías de futuro

La finalización de este estudio no implica que aquí acabe la parte de investigación en este campo. A partir de este estudio, existen muchas vías a tomar. Un ejemplo podría ser comprobar la influencia de ampliar la base de sujetos tanto en tamaño como en número de imágenes por sujeto. También, en lugar de realizar la identificación con imágenes estáticas, que la entrada para la identificación fuesen vídeos en los que aparezcan diferentes personas de diferente sexo.

Incluso, podría estudiarse el hecho de realizar una identificación facial incorporando métodos basados en características de la cara. Características como la silueta de la cara o la distancia entre los ojos se incorporarían a los métodos de identificación estudiados. Así, algunos parámetros que se comprobarían serían el beneficio de esta incorporación en la tarea de identificación, la modificación en tiempos como el de identificación…

Otra vía a tomar podría ser realizar un estudio de viabilidad para integrar procesamiento de reconocimiento e identificación facial a placas que utilicen procesadores menos potentes. Un ejemplo de estas placas podría ser la Raspberry Pi. Esta placa está siendo muy utilizada actualmente para desarrollar aplicaciones en ámbitos tan importantes e innovadores como pueden ser el de la domótica o el del Internet de las Cosas (IoT en inglés). Sería muy interesante verificar si pudiera ser compatible. Se podrían desarrollar muchísimas aplicaciones nuevas en torno a este ámbito.

Evidentemente, aquí solo se hace mención a algunas de las vías que pueden ser interesantes a estudiar de aquí a un futuro relativamente cercano. Viene bien recordar que, al final, el objetivo principal de la Visión Artificial y, en particular, del reconocimiento e identificación facial es emular el sentido de la vista.

“El cielo es el límite.”

-Wayne Dyer-

REFERENCIAS

[1] «OpenCV (Open Source Coputer Vision),» 2015. [En línea] Disponible en: http://opencv.org/

[2] Burton, A.M., White, D., & McNeill, A. (2010). *The Glasgow Face Matching Test*. Behavior Research Methods, 42(1), 286–291. doi:10.3758/BRM.42.1.286

[3] «Visión artificial e interacción sin mando,» 2010. [En línea] Disponible en: http://sabia.tic.udc.es/gc/Contenidos%20adicionales/trabajos/3D/VisionArtificial/index.html

[4] «¿Un selfie para validar una compra? Mastercard va en serio con el reconocimiento facial,» 2015. [En línea]. Disponible en: http://www.xataka.com/aplicaciones/un-selfie-para-validar-una-compra-mastercard-va-en-serio-con-el-reconocimiento-facial

[5] «Face recognition with OpenCV,» 2015. [En línea] Disponible en: http://docs.opencv.org/modules/contrib/doc/facerec/facerec_tutorial.html

[6] Kanade, T. *Picture processing system by computer complex and recognition of human faces.* PhD thesis, Kyoto University, November 1973.

[7] Brunelli, R., Poggio, T. *Face Recognition through Geometrical Features.* European Conference on Computer Vision (ECCV) 1992, S. 792–800.

[8] Turk, M., and Pentland, A. *Eigenfaces for recognition.* Journal of Cognitive Neuroscience 3 (1991), 71–86.

[9] «Principal Component Analysis and Linear Discriminant Analysis with GNU Octave.» 2011. [En línea]. Disponible en: http://www.bytefish.de/blog/pca_lda_with_gnu_octave/

[10] Fisher, R.A. *The use of multiple measurementsin taxonomic problems.* Annals Eugen. 7 (1936), 179-188.

[11] Belhumeur, P. N., Hespanha, J., and Kriegman, D. *Eigenfaces vs. Fisherfaces: Recognition Using Class Specific Linear Projection*. IEEE Transactions on Pattern Analysis and Machine Intelligence 19, 7 (1997), 711–720.

[12] S. Raudys and A.K. Jain. *Small sample size effects in statistical pattern recognition: Recommendations for practitioneers.* - IEEE Transactions on Pattern Analysis and Machine Intelligence 13, 3 (1991), 252-264.

[13] Ahonen, T., Hadid, A., and Pietikainen, M. *Face Recognition with Local Binary Patterns. Computer Vision* - ECCV 2004 (2004), 469–481.

[14] «Cplusplus,» 2015. [En línea] Disponible en: http://www.cplusplus.com/info/description/

[15] «Qt,» 2015. [En línea] Dispobible en: http://www.qt.io/

[16] «MATLAB,» 2015. [En línea] Disponible en: http://es.mathworks.com/products/matlab/

ANEXO A. DOCUMENTACIÓN DEL CÓDIGO (DOXYGEN)

Documentación de las clases

Referencia de la Clase QConfig

Se encarga de la lectura del fichero de configuracion para la interfaz de la aplicacion y la escritura de los ficheros con los datos generados por los distintos modelos de identificacion.

```
#include <Config.h>
```

Métodos públicos

- **QConfig** ()
  *Constructor de la clase **QConfig**.*
- virtual **~QConfig** ()
  *Destructor de la clase **QConfig**.*
- void **Inicializar** ()
  Funcion Inicializar.
- void **CargarArchivo** (int indice)
  Funcion CargarArchivo. Carga el archivo con los valores de configuracion de la interfaz de la aplicacion.
- void **EscribeValoresMujer** (int id_modelo, int id_CSV)
  Funcion EscribeValoresMujer. Escritura de los valores de identificacion de sujetos femeninos.
- void **EscribeValoresHombre** (int id_modelo, int id_CSV)
  Funcion EscribeValoresHombre. Escritura de los valores de identificacion de sujetos masculinos.
- void **EscribeTiemposModelos** (int id_CSV)
  Funcion EscribeTiemposModelos. Escritura de los valores de los tiempos por creacion y carga.

Atributos públicos

- QString **background**
- int **resolucion_w**
- int **resolucion_h**
- int **label_personas_x**
- int **label_personas_y**
- int **label_personas_w**
- int **label_personas_h**
- int **label_personas_marg**
- int **label_personas_spac**
- int **frame_tiempos_x**
- int **frame_tiempos_y**
- int **frame_tiempos_w**
- int **frame_tiempos_h**
- int **frame_tiempos_marg**
- int **frame_tiempos_spac**
- float **seg_CSV**
- vector< float > **seg_Modelos**
- vector< float > **seg_Carga**
- vector< float > **media_segundos**
- vector< float > **confidence**
- vector< int > **prediccion**

Atributos privados

- QSettings * **settings**

Descripción detallada

Se encarga de la lectura del fichero de configuracion para la interfaz de la aplicacion y la escritura de los ficheros con los datos generados por los distintos modelos de identificacion.

Documentación del constructor y destructor

QConfig::QConfig ()

Constructor de la clase **QConfig**.

Constructor vacio.

```
14 {
15
16 }
```

QConfig::~QConfig () [virtual]

Destructor de la clase **QConfig**.

Destructor vacio.

```
19 {
20
21 }
```

Documentación de las funciones miembro

void QConfig::CargarArchivo (int *indice*)

Funcion CargarArchivo. Carga el archivo con los valores de configuracion de la interfaz de la aplicacion.

Lectura del fichero de configuracion de la interfaz de la aplicacion. Inicializacion de las variables de configuracion usadas para realizar la interfaz de la aplicacion.

- **Parámetros:**

indice	Entero con el que se controla el archivo de configuracion a cargar.

```
29 {
30     //Apertura del fichero a leer
31     QString nombre_fichero = QString("etc/Config_%1.inp").arg(indice);
32     settings = new QSettings(nombre_fichero, QSettings::IniFormat);
33     qDebug() << "Cargando archivo " << nombre_fichero;
34
35     //Lectura del nombre de la imagen de background
36     settings->beginGroup("GENERALES");
37     background          = settings->value("background").toString();
38     settings->endGroup();
39
40     //Lectura de las propiedadess a configurar en la interfaz de la aplicacion
41     settings->beginGroup("INTERFAZ");
42     resolucion_w            = settings->value("resolucion_w").toInt();
43     resolucion_h            = settings->value("resolucion_h").toInt();
44
45     label_personas_x        = settings->value("label_personas_x").toInt();
46     label_personas_y        = settings->value("label_personas_y").toInt();
47     label_personas_w        = settings->value("label_personas_w").toInt();
48     label_personas_h        = settings->value("label_personas_h").toInt();
49     label_personas_marg     = settings->value("label_personas_marg").toInt();
50     label_personas_spac     = settings->value("label_personas_spac").toInt();
51
52     frame_tiempos_x         = settings->value("frame_tiempos_x").toInt();
53     frame_tiempos_y         = settings->value("frame_tiempos_y").toInt();
54     frame_tiempos_w         = settings->value("frame_tiempos_w").toInt();
55     frame_tiempos_h         = settings->value("frame_tiempos_h").toInt();
56     frame_tiempos_marg      = settings->value("frame_tiempos_marg").toInt();
57     frame_tiempos_spac      = settings->value("frame_tiempos_spac").toInt();
```

```
58      settings->endGroup();
59  }
```

void QConfig::EscribeTiemposModelos (int *id_CSV*)

Funcion EscribeTiemposModelos. Escritura de los valores de los tiempos por creacion y carga.

Escribe en un fichero los valores de los tiempos de creacion y carga de cada modelo utilizado en la aplicacion para un archivo CSV determinado.

Parámetros:

id_CSV	Identificador del archivo CSV

```
101 {
102     //Apertura del fichero de escritura de los datos
103     QString nombre_fichero =
QString("etc/Resultados/50x50/TiemposModelos_%1.m").arg(id_CSV);
104     settings = new QSettings(nombre_fichero, QSettings::IniFormat);
105
106     qDebug() << "Resultados: Guardando archivo " << nombre_fichero;
107
108     //Escritura de los datos en el fichero
109     settings->beginGroup(QString("TIEMPOS_MODELOS"));
110     for(uint i=0; i<seg_Modelos.size(); i++)
111     {
112         QString numero=QString("%1").arg(i+1);
113         QString linea=QString("%1   %2").arg((seg_Modelos[i]+seg_CSV), 0, 'f',
2).arg(seg_Carga[i],0,'f',2);
114         settings->setValue(numero,  linea);
115     }
116     settings->endGroup();
117 }
```

void QConfig::EscribeValoresHombre (int *id_modelo*, int *id_CSV*)

Funcion EscribeValoresHombre. Escritura de los valores de identificacion de sujetos masculinos.

Escribe en un fichero los valores de identificacion de la base de datos de los sujetos masculinos generados por un modelo y archivo CSV especificos.

Parámetros:

id_modelo	Identificador del modelo que ha generado los datos
id_CSV	Identificador del archivo CSV

```
82 {
83     //Apertura del fichero de escritura de los datos
84     QString nombre_fichero =
QString("etc/Resultados/50x50/ResultadosHombreM3_%1%2.m").arg(id_CSV).arg(id_modelo);
85     settings = new QSettings(nombre_fichero, QSettings::IniFormat);
86
87     qDebug() << "Resultados: Guardando archivo " << nombre_fichero;
88
89     //Escritura de los datos en el fichero
90     settings->beginGroup(QString("IDENTIFICACION_HOMBRE"));
91     for(uint i=0; i<media_segundos.size(); i++)
92     {
93         QString numero=QString("%1").arg(i+1);
94         QString linea=QString("%1   %2   %3").arg(media_segundos[i], 0, 'f',
3).arg(confidence[i],0,'f',4).arg(prediccion[i]+1);
95         settings->setValue(numero,  linea);
96     }
97     settings->endGroup();
98 }
```

void QConfig::EscribeValoresMujer (int *id_modelo*, int *id_CSV*)

Funcion EscribeValoresMujer. Escritura de los valores de identificacion de sujetos femeninos.

Escribe en un fichero los valores de identificacion de la base de datos de los sujetos femeninos generados por un modelo y archivo CSV especificos.

Parámetros:

id_modelo	Identificador del modelo que ha generado los datos

id_CSV	Identificador del archivo CSV

```
63 {
64     //Apertura del fichero de escritura de los datos
65     QString nombre_fichero =
QString("etc/Resultados/50x50/ResultadosMujerM3_%1%2.m").arg(id_CSV).arg(id_modelo);
66     settings = new QSettings(nombre_fichero, QSettings::IniFormat);
67
68     qDebug() << "Resultados: Guardando archivo " << nombre_fichero;
69
70     //Escritura de los datos en el fichero
71     settings->beginGroup(QString("IDENTIFICACION_MUJER"));
72     for(uint i=0; i<media_segundos.size(); i++)
73     {
74         QString numero=QString("%1").arg(i+1);
75         QString linea=QString("%1   %2   %3").arg(media_segundos[i], 0, 'f',
3).arg(confidence[i],0,'f',4).arg(prediccion[i]+1);
76         settings->setValue(numero,  linea);
77     }
78     settings->endGroup();
79 }
```

void QConfig::Inicializar ()

Funcion Inicializar.

Llama a **CargarArchivo()**;

```
24 {
25     CargarArchivo(0);
26 }
```

Documentación de los datos miembro

QString QConfig::background

Contiene el nombre de la imagen de background.

vector<float> QConfig::confidence

Confianza de la identificacion.

int QConfig::frame_tiempos_h

Alto del recuadro de tiempos de identificacion.

int QConfig::frame_tiempos_marg

Margen del recuadro de tiempos de identificacion.

int QConfig::frame_tiempos_spac

Espaciado del recuadro de tiempos de identificacion.

int QConfig::frame_tiempos_w

Ancho del recuadro de tiempos de identificacion.

int QConfig::frame_tiempos_x

Posicion x en la pantalla para el recuadro de tiempos de identificacion.

int QConfig::frame_tiempos_y

Posicion y en la pantalla para el recuadro de tiempos de identificacion.

int QConfig::label_personas_h

Alto del recuadro para el numero de sujetos identificados.

int QConfig::label_personas_marg

Margen del recuadro para el numero de sujetos identificados.

int QConfig::label_personas_spac

Espaciado del recuadro para el numero de sujetos identificados.

int QConfig::label_personas_w

Ancho del recuadro para el numero de sujetos identificados.

int QConfig::label_personas_x

Posicion x en la pantalla para el numero de sujetos identificados.

int QConfig::label_personas_y

Posicion y en la pantalla para el numero de sujetos identificados.

vector<float> QConfig::media_segundos

Tiempos de identificacion.

vector<int> QConfig::prediccion

Prediccion realizada por el modelo de identificacion.

int QConfig::resolucion_h

Resolucion de la pantalla en alto.

int QConfig::resolucion_w

Resolucion de la pantalla en ancho.

vector<float> QConfig::seg_Carga

Tiempos de carga de los distintos modelos de identificacion.

float QConfig::seg_CSV

Tiempo de carga del archivo CSV.

vector<float> QConfig::seg_Modelos

Tiempos de creacion de los distintos modelos de identificacion.

QSettings* QConfig::settings `[private]`

Utilizada para la lectura y escritura de ficheros

La documentación para esta clase fue generada a partir de los siguientes ficheros:

- src/**Config.h**
- src/**Config.cpp**

Referencia de la Clase QFormPrincipal

Hereda de QWidget. Se encarga de gestionar la interfaz del programa.

`#include <FormPrincipal.h>`

Diagrama de herencias de QFormPrincipal

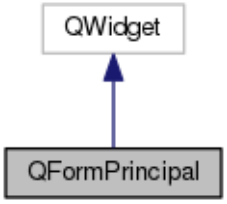
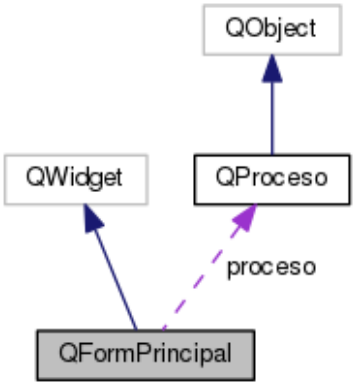


Diagrama de colaboración para QFormPrincipal:



Slots públicos

- void **SlotIniciar** ()
  Funcion SlotIniciar. Pone visibles los componentes de la interfaz de la aplicacion.
- void **SlotFinAperturaCSV** ()
  Funcion SlotFinAperturaCSV. Indica la creacion del primer modelo identificador.
- void **SlotActualizarTimer3** ()
  Funcion SlotActualizarTimer3.
- void **SlotSiguienteModelo** ()
  Funcion SlotSiguienteModelo. Indica la creacion del siguente modelo identificador.
- void **SlotCargaModelo** ()
  Funcion SlotCargaModelo. Indica la carga del modelo que se va a utilizar en las identificaciones.
- void **SlotFinCargaModelo** ()
  Funcion SlotFinCargaModelo. Indica el fin de la carga del modelo que se va a utilizar.
- void **SlotCargaImagen** ()
  Funcion SlotCargaImagen. Carga la imagen a identificar.
- void **SlotActualizarTimer2** ()
  Funcion SlotActualizarTimer2.
- void **SlotNuevaIdentificacion** (float conf, int predLabel)
  Funcion SlotNuevaIdentificacion. Actualiza los valores producidos por la identificacion.

Señales

- void **SignalAperturaCSV** (QString fn_csv)
  Señal SignalAperturaCSV.
- void **SignalCreaModeloFisherFace** (QString fn_model)
  Señal SignalCreaModeloFisherFace.

- void **SignalCreaModeloEigenFace** (QString fn_model)
  Señal SignalCreaModeloEigenFace.
- void **SignalCreaModeloLBPHFace** (QString fn_model)
  Señal SignalCreaModeloLBPHFace.
- void **SignalCreaArbolFisherFace** (QString fn_model, int **id_CSV**)
  Señal SignalCreaArbolFisherFace.
- void **SignalCreaArbolEigenFace** (QString fn_model, int **id_CSV**)
  Señal SignalCreaArbolEigenFace.
- void **SignalCreaArbolLBPHFace** (QString fn_model, int **id_CSV**)
  Señal SignalCreaArbolLBPHFace.
- void **SignalCargaModeloFisherFace** (QString fn_model)
  Señal SignalCargaModeloFisherFace.
- void **SignalCargaModeloEigenFace** (QString fn_model)
  Señal SignalCargaModeloEigenFace.
- void **SignalCargaModeloLBPHFace** (QString fn_model)
  Señal SignalCargaModeloLBPHFace.
- void **SignalCargaArbolFisherFace** (QString fn_model, int **id_CSV**)
  Señal SignalCargaArbolFisherFace.
- void **SignalCargaArbolEigenFace** (QString fn_model, int **id_CSV**)
  Señal SignalCargaArbolEigenFace.
- void **SignalCargaArbolLBPHFace** (QString fn_model, int **id_CSV**)
  Señal SignalCargaArbolLBPHFace.
- void **SignalIdentificar** (Mat ***test_im**)
  Señal SignalIdentificar.
- void **SignalIdentificarArbol** (Mat ***test_im**)
  Señal SignalIdentificarArbol.

Métodos públicos

- **QFormPrincipal** (QWidget *parent=0)
  *Constructor de la clase **QFormPrincipal**.*
- **~QFormPrincipal** ()
  *Destructor de la clase **QFormPrincipal**.*
- void **SetupUI** ()
  Funcion SetupUI. Se encarga de la inicializacion de la interfaz de la aplicacion.
- void **Comenzar** ()
  Funcion Comenzar.
- void **keyPressEvent** (QKeyEvent *e)
  Funcion keyPressEvent. Gestor de eventos de teclado.

Atributos públicos

- QFrame * frame_general
- QLabel * label_fondo
- QLabel * label_imagen
- QLabel * label_num_personas
- QFrame * frame_tiempos
- QVBoxLayout * **layout_tiempos**
- QLabel * label_text_tiempo_iden
- QLabel * label_text_tiempo_med
- QTimer * **timer**
- QTimer * **timer2**
- QTimer * **timer3**
- QProceso * proceso

- QThread * **thread**

Métodos privados

- void **ActualizarMedia** ()
  Funcion ActualizarMedia. Actualiza el valor de tiempo medio de identificacion.

Atributos privados

- float **segundos**
- float **seg_CSV**
- vector< string > **nom_Modelos**
- int **id_modelo**
- vector< float > **seg_Modelos**
- vector< float > **seg_Carga**
- int **id_persona**
- int **tam**
- Mat **test_im**
- Mat * **identifica**
- float **med_seg**
- vector< float > **media_segundos**
- vector< float > **confidence**
- vector< int > **prediccion**
- int **id_CSV**
- bool **genero**
- int **num_personas**

Descripción detallada

Hereda de QWidget. Se encarga de gestionar la interfaz del programa.

Documentación del constructor y destructor

QFormPrincipal::QFormPrincipal (QWidget * *parent* = 0)

Constructor de la clase **QFormPrincipal**.

Inicializa los valores por defecto de las variables de la clase. Inicializa la interfaz de la aplicacion. Conecta los temporizadores y los hilos entre si.

Parámetros:

parent	Valor por defecto cero. Referencia a la clase padre.

```
14     : QWidget(parent)
15 {
16     //Inicializacion de valores
17     segundos=0;
18     id modelo=0;
19     id_persona=1;
20     tam=50;
21     id_CSV=0;
22     genero=true;
23     med_seg=0;
24     num_personas=0;
25
26     //Inicializacion de la interfaz
27     SetupUI();
28
29     //Inicializacion de los temporizadores
30     timer = new QTimer(this);
31     timer2 = new QTimer(this);
32     timer3 = new QTimer(this);
33     proceso = new QProceso();
```

```
34
35     //Inicializacion del hilo proceso
36     thread = new QThread();
37     proceso->moveToThread(thread);
38
39     //Conexion de las señales emitidas por este hilo y el hilo proceso con las funciones a
ejecutar
40     //Apertura del archivo CSV
41     connect(this,       SIGNAL(SignalAperturaCSV(QString)),                proceso,
SLOT(SlotAbrirCSV(QString)));
42     connect(proceso,    SIGNAL(SignalFinAperturaCSV()),                    this,
SLOT(SlotFinAperturaCSV()));
43
44     //Creacion de los diferentes modelos
45     connect(this,       SIGNAL(SignalCreaModeloFisherFace(QString)),       proceso,
SLOT(SlotCreaModeloFisherFace(QString)));
46     connect(this,       SIGNAL(SignalCreaModeloEigenFace(QString)),        proceso,
SLOT(SlotCreaModeloEigenFace(QString)));
47     connect(this,       SIGNAL(SignalCreaModeloLBPHFace(QString)),         proceso,
SLOT(SlotCreaModeloLBPHFace(QString)));
48     connect(this,       SIGNAL(SignalCreaArbolFisherFace(QString, int)),   proceso,
SLOT(SlotCreaArbolFisherFace(QString, int)));
49     connect(this,       SIGNAL(SignalCreaArbolEigenFace(QString, int)),    proceso,
SLOT(SlotCreaArbolEigenFace(QString, int)));
50     connect(this,       SIGNAL(SignalCreaArbolLBPHFace(QString, int)),     proceso,
SLOT(SlotCreaArbolLBPHFace(QString, int)));
51     connect(proceso,    SIGNAL(SignalFinCreacionModelo()),                 this,
SLOT(SlotSiguienteModelo()));
52
53     //Carga de los diferentes modelos
54     connect(this,       SIGNAL(SignalCargaModeloFisherFace(QString)),      proceso,
SLOT(SlotCargaModeloFisherFace(QString)));
55     connect(this,       SIGNAL(SignalCargaModeloEigenFace(QString)),       proceso,
SLOT(SlotCargaModeloEigenFace(QString)));
56     connect(this,       SIGNAL(SignalCargaModeloLBPHFace(QString)),        proceso,
SLOT(SlotCargaModeloLBPHFace(QString)));
57     connect(this,       SIGNAL(SignalCargaArbolFisherFace(QString, int)),  proceso,
SLOT(SlotCargaArbolFisherFace(QString, int)));
58     connect(this,       SIGNAL(SignalCargaArbolEigenFace(QString, int)),   proceso,
SLOT(SlotCargaArbolEigenFace(QString, int)));
59     connect(this,       SIGNAL(SignalCargaArbolLBPHFace(QString, int)),    proceso,
SLOT(SlotCargaArbolLBPHFace(QString, int)));
60     connect(proceso,    SIGNAL(SignalFinCargaModelo()),                    this,
SLOT(SlotFinCargaModelo()));
61
62     //Identificacion de un sujeto
63     connect(this,       SIGNAL(SignalIdentificar(Mat *)),                  proceso,
SLOT(SlotIdentificar(Mat *)));
64     connect(this,       SIGNAL(SignalIdentificarArbol(Mat *)),             proceso,
SLOT(SlotIdentificarArbol(Mat *)));
65     connect(proceso,    SIGNAL(SignalFinIdentificacion(float, int)),       this,
SLOT(SlotNuevaIdentificacion(float, int)));
66
67     //Conexion de las señales emitidas por los temporizadores con las funciones a ejecutar
de este hilo
68     connect(timer,      SIGNAL(timeout()),                                 this,
SLOT(SlotTerminar()));
69     connect(timer2,     SIGNAL(timeout()),                                 this,
SLOT(SlotActualizarTimer2()));
70     connect(timer3,     SIGNAL(timeout()),                                 this,
SLOT(SlotActualizarTimer3()));
71
72 }
```

QFormPrincipal::~QFormPrincipal ()

Destructor de la clase **QFormPrincipal**.

Elimina la variable proceso.

```
146 {
147     delete proceso;
148 }
```

Documentación de las funciones miembro

void QFormPrincipal::ActualizarMedia () [private]

Funcion ActualizarMedia. Actualiza el valor de tiempo medio de identificacion.

Una vez realizada la identificacion, se actualiza el valor de tiempo medio de identificacion en la interfaz de la aplicacion.

```
495 {
496     med_seg=0;
497     media_segundos.push_back(segundos);
498     uint i;
499 
500     //Calculo de la nueva media de tiempo de identificacion
501     for(i=0;i<media_segundos.size();i++)
502         med_seg+=media_segundos[i];
503     med_seg=(med_seg/media_segundos.size());
504 
505     //Actualizacion en la interfaz del tiempo medio de identificacion
506     label_text_tiempo_med->setText(QString("%1").arg(med_seg, 0, 'f', 3));
507 }
```

void QFormPrincipal::Comenzar ()

Funcion Comenzar.

Llama a **SlotIniciar()**;

```
151 {
152     SlotIniciar();
153 }
```

void QFormPrincipal::keyPressEvent (QKeyEvent * *e*)

Funcion keyPressEvent. Gestor de eventos de teclado.

En caso de que se pulse "esc", se interrumpira la ejecucion de la aplicacion, saliendose de esta.

Parámetros:

e	Valor correspondiente a la tecla pulsada

```
532 {
533     if(ke->key()==Qt::Key_Escape)
534         SlotTerminar();
535 }
```

void QFormPrincipal::SetupUI ()

Funcion SetupUI. Se encarga de la inicializacion de la interfaz de la aplicacion.

Se inicializan y se asocian los valores cargados por **QConfig** a las variables encargadas de la interfaz de la aplicacion.

```
75 {
76     //Inicializacion de la fuente utilizada para la interfaz de la aplicacion
77     QFont font1;
78     font1.setPointSize(20);
79 
80     this->resize(config.resolucion_w, config.resolucion_h);
81 
82     QPalette palette;
83     palette.setBrush(QPalette::All, QPalette::Window, QBrush(Qt::black));
84 
85     this->setPalette(palette);
86 
87     //Inicializacion del espacio reservado para la interfaz de la aplicacion
88     frame_general = new QFrame(this);
89     frame_general->setGeometry(QRect(0, 0, config.resolucion_w, config.resolucion_h));
90 
91     //Inicializacion de la imagen de fondo de la aplicacion
92     label_imagen = new QLabel(frame_general);
```

```
93     label_imagen->setGeometry(QRect(0, 0, config.resolucion_w, config.resolucion_h));
94     label_imagen->setFrameShape(QFrame::NoFrame);
95     label_imagen->setAlignment(Qt::AlignCenter);
96     QPixmap pixmap_imagen = QPixmap(QString("resources/media/imagen0"));
97     label_imagen->setPixmap(pixmap_imagen);
98     label_imagen->setMask(pixmap_imagen.mask());
99
100     //Inicializacion del numero de identificaciones realizadas
101     label_num_personas = new QLabel(frame_general);
102     label_num_personas->setGeometry(QRect(config.label_personas_x,
config.label_personas_y, config.label_personas_w, config.label_personas_h));
103     label_num_personas->setMargin(config.label_personas_marg);
104     label_num_personas->setFrameShape(QFrame::NoFrame);
105     label_num_personas->setAlignment(Qt::AlignCenter);
106     label_num_personas->setFont(font1);
107
108     //Inicializacion de la imagen de background de la aplicacion
109     label_fondo = new QLabel(frame_general);
110     label_fondo->setGeometry(QRect(0, 0, config.resolucion_w, config.resolucion_h));
111     label_fondo->setFrameShape(QFrame::NoFrame);
112     QPixmap pixmap_background = QPixmap(config.background);
113     label_fondo->setPixmap(pixmap_background);
114     label_fondo->setMask(pixmap_background.mask());
115
116     //Inicializacion de la informacion de tiempos
117     frame_tiempos = new QFrame(frame_general);
118     frame_tiempos->setGeometry(config.frame_tiempos_x, config.frame_tiempos_y,
config.frame_tiempos_w, config.frame_tiempos_h);
119
120     layout_tiempos = new QVBoxLayout();
121     layout_tiempos->setMargin(config.frame_tiempos_marg);
122     layout_tiempos->setSpacing(config.frame_tiempos_spac);
123
124     label_text_tiempo_iden = new QLabel();
125     label_text_tiempo_med = new QLabel();
126
127     label_text_tiempo_iden->setAlignment(Qt::AlignCenter);
128     label_text_tiempo_iden->setFont(font1);
129     label_text_tiempo_med->setAlignment(Qt::AlignCenter);
130     label_text_tiempo_med->setFont(font1);
131
132     layout_tiempos->addWidget(label_text_tiempo_iden);
133     layout_tiempos->addWidget(label_text_tiempo_med);
134
135     frame_tiempos->setLayout(layout_tiempos);
136
137     //Colocacion de los objetos en la interfaz de la aplicacion
138     frame_general->             raise();
139     label_imagen->              raise();
140     label_fondo->               raise();
141     frame_tiempos->             raise();
142     label_num_personas->        raise();
143 }
```

void QFormPrincipal::SignalAperturaCSV (QString *fn_csv*) [signal]

Señal SignalAperturaCSV.

Señal que indica la apertura del fichero CSV.

Parámetros:

fn_csv	Nombre del archivo CSV a cargar.

void QFormPrincipal::SignalCargaArbolEigenFace (QString *fn_model*, int *id_CSV*) [signal]

Señal SignalCargaArbolEigenFace.

Señal que indica la carga del modelo EigenFace con estructura en arbol.

Parámetros:

fn_model	Nombre del modelo a cargar.
id_CSV	Identificador de CSV.

void QFormPrincipal::SignalCargaArbolFisherFace (QString *fn_model*, int *id_CSV*) `[signal]`

Señal SignalCargaArbolFisherFace.

Señal que indica la carga del modelo FisherFace con estructura en arbol.

Parámetros:

fn_model	Nombre del modelo a cargar.
id_CSV	Identificador de CSV.

void QFormPrincipal::SignalCargaArbolLBPHFace (QString *fn_model*, int *id_CSV*) `[signal]`

Señal SignalCargaArbolLBPHFace.

Señal que indica la carga del modelo LBPHFace con estructura en arbol.

Parámetros:

fn_model	Nombre del modelo a cargar.
id_CSV	Identificador de CSV.

void QFormPrincipal::SignalCargaModeloEigenFace (QString *fn_model*) `[signal]`

Señal SignalCargaModeloEigenFace.

Señal que indica la carga del modelo EigenFace.

Parámetros:

fn_model	Nombre del modelo a cargar.

void QFormPrincipal::SignalCargaModeloFisherFace (QString *fn_model*) `[signal]`

Señal SignalCargaModeloFisherFace.

Señal que indica la carga del modelo FisherFace.

Parámetros:

fn_model	Nombre del modelo a cargar.

void QFormPrincipal::SignalCargaModeloLBPHFace (QString *fn_model*) `[signal]`

Señal SignalCargaModeloLBPHFace.

Señal que indica la carga del modelo LBPHFace.

Parámetros:

fn_model	Nombre del modelo a cargar.

void QFormPrincipal::SignalCreaArbolEigenFace (QString *fn_model*, int *id_CSV*) `[signal]`

Señal SignalCreaArbolEigenFace.

Señal que indica la creacion del modelo EigenFace con estructura en arbol.

Parámetros:

fn_model	Nombre del modelo a crear.
id_CSV	Identificador de CSV.

void QFormPrincipal::SignalCreaArbolFisherFace (QString *fn_model*, int *id_CSV*) `[signal]`

Señal SignalCreaArbolFisherFace.

Señal que indica la creacion del modelo FisherFace con estructura en arbol.

Parámetros:

fn_model	Nombre del modelo a crear.
id_CSV	Identificador de CSV.

void QFormPrincipal::SignalCreaArbolLBPHFace (QString *fn_model*, int *id_CSV*) `[signal]`

Señal SignalCreaArbolLBPHFace.

Señal que indica la creacion del modelo LBPHFace con estructura en arbol.

Parámetros:

fn_model	Nombre del modelo a crear.
id_CSV	Identificador de CSV.

void QFormPrincipal::SignalCreaModeloEigenFace (QString *fn_model*) `[signal]`

Señal SignalCreaModeloEigenFace.

Señal que indica la creacion del modelo EigenFace.

Parámetros:

fn_model	Nombre del modelo a crear.

void QFormPrincipal::SignalCreaModeloFisherFace (QString *fn_model*) `[signal]`

Señal SignalCreaModeloFisherFace.

Señal que indica la creacion del modelo FisherFace.

Parámetros:

fn_model	Nombre del modelo a crear.

void QFormPrincipal::SignalCreaModeloLBPHFace (QString *fn_model*) `[signal]`

Señal SignalCreaModeloLBPHFace.

Señal que indica la creacion del modelo LBPHFace.

Parámetros:

fn_model	Nombre del modelo a crear.

void QFormPrincipal::SignalIdentificar (Mat * *test_im*) `[signal]`

Señal SignalIdentificar.

Señal que indica que se va a proceder a identificar el sujeto.

Parámetros:

test_im	Sujeto a identificar.

void QFormPrincipal::SignalIdentificarArbol (Mat * *test_im*) `[signal]`

Señal SignalIdentificarArbol.

Señal que indica que se va a proceder a identificar el sujeto en un modelo con estructura en arbol.

Parámetros:

test_im	Sujeto a identificar.

void QFormPrincipal::SlotActualizarTimer2 () `[slot]`

Funcion SlotActualizarTimer2.

Actualiza la cuenta del temporizador Timer2 y el tiempo de identificacion de la interfaz de la aplicacion.

```
467 {
468     segundos+=0.001;
469     label_text_tiempo_iden->setText(QString("%1").arg(segundos, 0, 'f', 3));
470 }
```

void QFormPrincipal::SlotActualizarTimer3 () [slot]

Funcion SlotActualizarTimer3.

Actualiza la cuenta del temporizador Timer3.

```
179 {
180     segundos+=0.01;
181 }
```

void QFormPrincipal::SlotCargaImagen () [slot]

Funcion SlotCargaImagen. Carga la imagen a identificar.

Se encarga de cargar la imagen a identificar de la base de datos de sujetos femeninos y masculinos. Inicializa el temporizador timer2 e indica el comienzo para una nueva identificacion. Una vez realizada la identificacion a todos los sujetos femeninos y masculinos, se guardan los datos generados por ese modelo en un fichero.

```
366 {
367     if(genero)
368     {
369         //Carga imagen de la base de datos de sujetos femeninos
370         char buf[40]={0};
371         sprintf(  buf, "resources/media/Fotos/M3/female/%03d.jpg", id_persona);
372         test_im=imread(buf,0);
373         if(!test_im.empty())
374         {
375             id_persona++;
376             cv::resize(test_im,test_im,Size(50,50));
377             identifica = new Mat();
378             test im.copyTo(*identifica);
379 
380             //Activacion temporizador de identificacion
381             timer2->start(1);
382 
383             //Emite la señal para realizar la identificacion en funcion si es una
estructura simple o en arbol
384             if(id_modelo<=2)
385                 emit SignalIdentificar(identifica);
386             else
387                 emit SignalIdentificarArbol(identifica);
388         }
389         else if(id_persona>tam&&id_modelo<=5)
390         {
391             id_persona=1;
392             segundos=0;
393 
394             //Todos los sujetos femeninos identificados. Se guardan los datos generados
en un fichero
395             config.media_segundos=this->media_segundos;
396             config.confidence=this->confidence;
397             config.prediccion=this->prediccion;
398             config.EscribeValoresMujer(id_modelo,id_CSV);
399             this->media_segundos.clear();
400             this->confidence.clear();
401             this->prediccion.clear();
402 
403             //Comienzo de la identificacion de los sujetos masculinos
404             genero=false;
405             SlotCargaImagen();
406         }
407     }
408     else
409     {
410         //Carga imagen de la base de datos de sujetos masculinos
```

```
411         char buf[40]={0};
412         sprintf(  buf, "resources/media/Fotos/M3/male/%03d.jpg", id_persona);
413         test_im=imread(buf,0);
414         if(!test_im.empty())
415         {
416             id_persona++;
417             cv::resize(test_im,test_im,Size(50,50));
418             identifica = new Mat();
419             test_im.copyTo(*identifica);
420 
421             //Activacion temporizador de identificacion
422             timer2->start(1);
423 
424             //Emite la señal para realizar la identificacion en funcion si es una
estructura simple o en arbol
425             if(id_modelo<=2)
426                 emit SignalIdentificar(identifica);
427             else
428                 emit SignalIdentificarArbol(identifica);
429         }
430         else if(id persona>tam&&id modelo<5)
431         {
432             id_persona=1;
433             segundos=0;
434 
435             //Todos los sujetos masculinos identificados. Se guardan los datos generados
en un fichero
436             config.media_segundos=this->media_segundos;
437             config.confidence=this->confidence;
438             config.prediccion=this->prediccion;
439             config.EscribeValoresHombre(id_modelo,id_CSV);
440             this->media_segundos.clear();
441             this->confidence.clear();
442             this->prediccion.clear();
443             genero=true;
444             id_modelo++;
445 
446             //Carga del siguiente modelo identificador
447             SlotCargaModelo();
448         }
449         else if(id_modelo==5)  //Todos los modelos identificadores han sido cargados y
probados
450         {
451             //Todos los sujetos masculinos identificados. Se guardan los datos generados
en un fichero
452             config.media segundos=this->media segundos;
453             config.confidence=this->confidence;
454             config.prediccion=this->prediccion;
455             config.EscribeValoresHombre(id_modelo,id_CSV);
456             this->media_segundos.clear();
457             this->confidence.clear();
458             this->prediccion.clear();
459 
460             //Se procede al cierre de la aplicacion
461             SlotSalir();
462         }
463     }
464 }
```

void QFormPrincipal::SlotCargaModelo () [slot]

Funcion SlotCargaModelo. Indica la carga del modelo que se va a utilizar en las identificaciones.

Cambia la imagen de fondo. Reinicializa el temporizador timer3 y se indica que comience la carga del modelo a utilizar para la proxima identificacion de la base de datos de sujetos completa.

```
291 {
292     //Carga de nueva imagen de fondo
293     QPixmap pixmap_imagen= QPixmap(QString("resources/media/imagen1"));
294     label_imagen->setPixmap(pixmap_imagen);
295     label_imagen->setMask(pixmap_imagen.mask());
296     label_text_tiempo_med->setText(QString("%1").arg(segundos, 0, 'f', 3));
297     if(id_modelo==0)
298     {
299         //Temporizador activado
300         timer3->start(10);
301 
```

```
302         //Emision de señal para cargar el modelo FisherFace
303         emit SignalCargaModeloFisherFace(QString::fromStdString(nom_Modelos[id_modelo]));
304     }
305     else if(id_modelo==1)
306     {
307         //Temporizador activado
308         timer3->start(10);
309 
310         //Emision de señal para cargar el modelo EigenFace
311         emit SignalCargaModeloEigenFace(QString::fromStdString(nom Modelos[id modelo]));
312     }
313     else if(id_modelo==2)
314     {
315         //Temporizador activado
316         timer3->start(10);
317 
318         //Emision de señal para cargar el modelo LBPHFace
319         emit SignalCargaModeloLBPHFace(QString::fromStdString(nom_Modelos[id_modelo]));
320     }
321     else if(id_modelo==3)
322     {
323         //Temporizador activado
324         timer3->start(10);
325 
326         //Emision de señal para cargar el modelo FisherFace con estructura en arbol
327         emit SignalCargaArbolFisherFace(QString::fromStdString(nom_Modelos[id_modelo]),
id_CSV);
328     }
329     else if(id_modelo==4)
330     {
331         //Temporizador activado
332         timer3->start(10);
333 
334         //Emision de señal para cargar el modelo EigenFace con estructura en arbol
335         emit SignalCargaArbolEigenFace(QString::fromStdString(nom_Modelos[id_modelo]),
id_CSV);
336     }
337     else if(id_modelo==5)
338     {
339         //Temporizador activado
340         timer3->start(10);
341 
342         //Emision de señal para cargar el modelo LBPHFace con estructura en arbol
343         emit SignalCargaArbolLBPHFace(QString::fromStdString(nom_Modelos[id_modelo]),
id_CSV);
344     }
345 }
```

void QFormPrincipal::SlotFinAperturaCSV () [slot]

Funcion SlotFinAperturaCSV. Indica la creacion del primer modelo identificador.

Una vez cargadas las imagenes utilizadas por los modelos identificadores, se para el temporizador timer3 y se guarda el resultado. Se reinicializa el temporizador y se indica que comience la creacion del primer modelo de identificacion.

```
184 {
185     //Detencion del temporizador
186     timer3->stop();
187 
188     //Asociacion de los segundos que se ha tardado en cargar el archivo CSV
189     seg_CSV=segundos;
190 
191     //Emision de la señal para la creacion del modelo de identificacion FisherFace
192     nom_Modelos.push_back("etc/Resultados/50x50/FisherFace_4.yml");
193     segundos=0;
194     timer3->start(10);
195     emit SignalCreaModeloFisherFace(QString::fromStdString(nom_Modelos[id_modelo]));
196 }
```

void QFormPrincipal::SlotFinCargaModelo () [slot]

Funcion SlotFinCargaModelo. Indica el fin de la carga del modelo que se va a utilizar.

Terminada la carga del modelo de identificacion, se para el temporizador timer3 y se guarda el tiempo de carga del modelo. Se cambia la imagen de fondo y llama a **SlotCargaImagen()**.

```
348 {
349     //Detencion temporizador
350     timer3->stop();
351
352     //Asociacion de los segundos que se ha tardado en crear el modelo
353     seg_Carga.push_back(segundos);
354     segundos=0;
355
356     //Carga nueva imagen de fondo
357     QPixmap pixmap_imagen= QPixmap(QString("resources/media/imagen2"));
358     label_imagen->setPixmap(pixmap_imagen);
359     label_imagen->setMask(pixmap_imagen.mask());
360
361     //Comienzo de la identificacion
362     SlotCargaImagen();
363 }
```

void QFormPrincipal::SlotIniciar () [slot]

Funcion SlotIniciar. Pone visibles los componentes de la interfaz de la aplicacion.

Comienzo del hilo thread. Los componentes de la interfaz de la aplicacion se hacen visibles. Comienzo del temporizador timer3. Da comienzo a la carga de las imagenes utilizadas para crear los distintos modelos de identificacion.

```
156 {
157     //Comienzo del hilo thread
158     thread->start();
159
160     //Objetos de la interfaz de la aplicacion visibles
161     label_imagen->              setVisible(true);
162     label_fondo->               setVisible(true);
163     label_num_personas->        setVisible(true);
164     frame_tiempos->             setVisible(true);
165
166     //Valores iniciales de sujetos identificados y tiempos
167     label_num_personas->setText(QString("%1").arg(num_personas));
168     label_text_tiempo_iden->setText(QString("%1").arg(segundos, 0, 'f', 3));
169     label_text_tiempo_med->setText(QString("%1").arg(med_seg, 0, 'f', 3));
170
171     //Temporizador activado
172     timer3->start(10);
173
174     //Emision de la señal para proceder a la apertura del archivo CSV
175     emit SignalAperturaCSV("resources/media/CSV0.txt");
176 }
```

void QFormPrincipal::SlotNuevaIdentificacion (float *conf*, int *predLabel*) [slot]

Funcion SlotNuevaIdentificacion. Actualiza los valores producidos por la identificacion.

Terminada la identificacion actual, se detiene el temporizador timer2, se actualizan los valores correspondientes a una identificacion y se llama a **SlotCargaImagen()** para proceder a la carga de la siguiente imagen a identificar.

Parámetros:

conf	El valor de confianza devuelto por el modelo tras la identificacion
predLabel	El valor de la prediccion del sujeto que se ha identificado.

```
473 {
474     //Detencion del temporizador de identificacion
475     timer2->stop();
476
477     //Asociacion de los nuevos datos generados por la identificacion
478     confidence.push_back(conf);
479     prediccion.push_back(predLabel);
480     num_personas++;
481     ActualizarMedia();
482     segundos=0;
483
484     //Actualizacion en la interfaz del numero de personas identificadas
485     label_num_personas->setText(QString("%1").arg(num_personas));
```

```
486
487     //Actualizacion en la interfaz del tiempo de identificacion para la proxima
identificacion
488     label_text_tiempo_iden->setText(QString("%1").arg(segundos, 0, 'f', 3));
489
490     //Siguiente identificacion
491     SlotCargaImagen();
492 }
```

void QFormPrincipal::SlotSalir () [slot]

Funcion SlotSalir. Guarda los tiempos de creacion y carga de los modelos de identificacion.

Cambia la imagen de fondo. Se guardan los tiempos de creacion y carga de los diferentes modelos de identificacion empleados. Se inicializa el temporizador timer1.

```
510 {
511     //Carga nueva imagen de fondo
512     QPixmap pixmap_imagen= QPixmap(QString("resources/media/imagen3"));
513     label_imagen->setPixmap(pixmap_imagen);
514     label_imagen->setMask(pixmap_imagen.mask());
515
516     //Escritura de los tiempos de creacion y carga de los modelos de identificacion
empleados
517     config.seg_CSV=this->seg_CSV;
518     config.seg_Modelos=this->seg_Modelos;
519     config.seg_Carga=this->seg_Carga;
520     config.EscribeTiemposModelos(id_CSV);
521
522     //Activacion temporizador final
523     timer->start(500);
524 }
```

void QFormPrincipal::SlotSiguienteModelo () [slot]

Funcion SlotSiguienteModelo. Indica la creacion del siguente modelo identificador.

Terminada la creacion del modelo identificador, se para el temporizador timer3 y se guarda el tiempo de creacion del modelo. Se reinicializa el temporizador y se indica que comience la creacion del siguiente modelo de identificacion. Se repite hasta que se hayan creado los seis modelos diferentes.

```
199 {
200     if(id_modelo==0)
201     {
202         //Detencion del temporizador
203         timer3->stop();
204
205         //Asociacion de los segundos que se ha tardado en crear el modelo
206         seg_Modelos.push_back(segundos);
207         segundos=0;
208         id_modelo=1;
209
210         //Emision de la señal para la creacion del modelo de identificacion EigenFace
211         nom_Modelos.push_back("etc/Resultados/50x50/EigenFace_4.yml");
212         timer3->start(10);
213         emit SignalCreaModeloEigenFace(QString::fromStdString(nom_Modelos[id_modelo]));
214     }
215     else if(id_modelo==1)
216     {
217         //Detencion del temporizador
218         timer3->stop();
219
220         //Asociacion de los segundos que se ha tardado en crear el modelo
221         seg_Modelos.push_back(segundos);
222         segundos=0;
223         id_modelo=2;
224
225         //Emision de la señal para la creacion del modelo de identificacion LBPHFace
226         nom_Modelos.push_back("etc/Resultados/50x50/LBPHFace_4.yml");
227         timer3->start(10);
228         emit SignalCreaModeloLBPHFace(QString::fromStdString(nom_Modelos[id_modelo]));
229     }
230     else if(id_modelo==2)
231     {
```

```
232         //Detencion del temporizador
233         timer3->stop();
234
235         //Asociacion de los segundos que se ha tardado en crear el modelo
236         seg_Modelos.push_back(segundos);
237         segundos=0;
238
239         //Emision de la señal para la creacion del modelo de identificacion FisherFace
con estructura en arbol.
240         nom_Modelos.push_back("etc/Resultados/50x50/Arbol/FisherFace_%d%d.yml");
241         id_modelo=3;
242         timer3->start(10);
243         emit SignalCreaArbolFisherFace(QString::fromStdString(nom Modelos[id modelo]),
id_CSV);
244     }
245     else if(id_modelo==3)
246     {
247         //Detencion del temporizador
248         timer3->stop();
249
250         //Asociacion de los segundos que se ha tardado en crear el modelo
251         seg_Modelos.push_back(segundos);
252         segundos=0;
253         id_modelo=4;
254
255         //Emision de la señal para la creacion del modelo de identificacion EigenFace con
estructura en arbol.
256         nom_Modelos.push_back("etc/Resultados/50x50/Arbol/EigenFace_%d%d.yml");
257         timer3->start(10);
258         emit SignalCreaArbolEigenFace(QString::fromStdString(nom_Modelos[id_modelo]),
id_CSV);
259     }
260     else if(id_modelo==4)
261     {
262         //Detencion del temporizador
263         timer3->stop();
264
265         //Asociacion de los segundos que se ha tardado en crear el modelo
266         seg_Modelos.push_back(segundos);
267         segundos=0;
268         id_modelo=5;
269
270         //Emision de la señal para la creacion del modelo de identificacion LBPHFace con
estructura en arbol.
271         nom_Modelos.push_back("etc/Resultados/50x50/Arbol/LBPHFace_%d%d.yml");
272         timer3->start(10);
273         emit SignalCreaArbolLBPHFace(QString::fromStdString(nom_Modelos[id_modelo]),
id_CSV);
274     }
275     else if(id_modelo==5)
276     {
277         //Detencion del temporizador
278         timer3->stop();
279
280         //Asociacion de los segundos que se ha tardado en crear el modelo
281         seg_Modelos.push_back(segundos);
282         segundos=0;
283         id_modelo=0;
284
285         //Carga del primer modelo
286         SlotCargaModelo();
287     }
288 }
```

void QFormPrincipal::SlotTerminar () [slot]

Funcion SlotTerminar.

Cierra la clase actual

```
527 {
528     this->close();
529 }
```

Documentación de los datos miembro

vector<float> QFormPrincipal::confidence [private]

Valores de confianza de las identificaciones realizadas

QFrame* QFormPrincipal::frame_general

Espacio reservado para la interfaz de la aplicacion.

QFrame* QFormPrincipal::frame_tiempos

Espacio reservado para la informacion de tiempos.

bool QFormPrincipal::genero [private]

Genero sobre el que se esta identificando

int QFormPrincipal::id_CSV [private]

Identificador del archivo CSV

int QFormPrincipal::id_modelo [private]

Identificador para el modelo que se este usando

int QFormPrincipal::id_persona [private]

Identificador para cargar el sujeto a identificar

Mat* QFormPrincipal::identifica [private]

Utilizado para transferir la imagen del sujeto a identificar a la clase **QProceso**

QLabel* QFormPrincipal::label_fondo

Carga la imagen de background de la aplicacion.

QLabel* QFormPrincipal::label_imagen

Carga las imagenes de fondo que apareceran a lo largo de la aplicacion.

QLabel* QFormPrincipal::label_num_personas

Se encarga de actualizar el valor del numero de identificaciones realizadas.

QLabel* QFormPrincipal::label_text_tiempo_iden

Se encarga de actualizar el valor de tiempo empleado en la identificacion.

QLabel* QFormPrincipal::label_text_tiempo_med

Se encarga de acutalizar el valor de tiempo medio de las identificaciones.

QVBoxLayout* QFormPrincipal::layout_tiempos

Se encarga del layout de la informacion de tiempos.

float QFormPrincipal::med_seg [private]

Media del tiempo empleado en las identificaciones

vector<float> QFormPrincipal::media_segundos [private]

Tiempos empleados en las distintas identificaciones realizadas

vector<string> QFormPrincipal::nom_Modelos [private]

Nombres utilizados para guardar los distintos modelos de identificacion

int QFormPrincipal::num_personas `[private]`

Numero total de personas identificadas

vector<int> QFormPrincipal::prediccion `[private]`

Valores de prediccion de las identificaciones realizadas

QProceso* QFormPrincipal::proceso

Instancia de la clase **QProceso**.

vector<float> QFormPrincipal::seg_Carga `[private]`

Tiempos empleados para la carga de los diferentes modelos de identificacion

float QFormPrincipal::seg_CSV `[private]`

Guarda el tiempo que se tarda en abir el archivo CSV

vector<float> QFormPrincipal::seg_Modelos `[private]`

Tiempos empleados para la creacion de los diferentes modelos de identificacion

float QFormPrincipal::segundos `[private]`

Usada por los temporizadores

int QFormPrincipal::tam `[private]`

Tamaño de la base de datos de sujetos a identificar

Mat QFormPrincipal::test_im `[private]`

Imagen del sujeto a identificar

QThread* QFormPrincipal::thread

Instancia de la clase QThread.

QTimer* QFormPrincipal::timer

Temporizador para la finalizacion de la aplicacion.

QTimer* QFormPrincipal::timer2

Temporizador para obtener los tiempos de identificacion.

QTimer* QFormPrincipal::timer3

Temporizador para obtener los tiempo de creacion y carga de los diferentes modelos.

La documentación para esta clase fue generada a partir de los siguientes ficheros:

- src/**FormPrincipal.h**
- src/**FormPrincipal.cpp**

Referencia de la Clase QProceso

Hereda de QObject. Se encarga de la creacion, carga e identificacion llevada a cabo por los modelos.

`#include <Proceso.h>`

Diagrama de herencias de QProceso

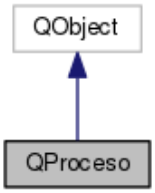
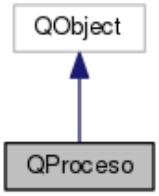


Diagrama de colaboración para QProceso:



Slots públicos

- void **SlotAbrirCSV** (QString fn_csv)
  Funcion SlotAbrirCSV. Indica cuando se ha terminado la lectura del fichero CSV.
- void **SlotCreaModeloFisherFace** (QString fn_model)
  Funcion SlotCreaModeloFisherFace. Crea el modelo FisherFace.
- void **SlotCreaModeloEigenFace** (QString fn_model)
  Funcion SlotCreaModeloEigenFace. Crea el modelo EigenFace.
- void **SlotCreaModeloLBPHFace** (QString fn_model)
  Funcion SlotCreaModeloLBPHFace. Crea el modelo LBPHFace.
- void **SlotCargaModeloFisherFace** (QString fn_model)
  Funcion SlotCargaModeloFisherFace. Carga el modelo FisherFace.
- void **SlotCargaModeloEigenFace** (QString fn_model)
  Funcion SlotCargaModeloEigenFace. Carga el modelo EigenFace.
- void **SlotCargaModeloLBPHFace** (QString fn_model)
  Funcion SlotCargaModeloLBPHFace. Carga el modelo LBPHFace.
- void **SlotIdentificar** (Mat *test_im)
  Funcion SlotIdentificar. Identifica a un sujeto.
- void **SlotCreaArbolFisherFace** (QString fn_model, int id_CSV)
  Funcion SlotCreaArbolFisherFace. Crea el modelo FisherFace con estructura en arbol.
- void **SlotCreaArbolEigenFace** (QString fn_model, int id_CSV)
  Funcion SlotCreaArbolEigenFace. Crea el modelo EigenFace con estructura en arbol.
- void **SlotCreaArbolLBPHFace** (QString fn_model, int id_CSV)
  Funcion SlotCreaArbolLBPHFace. Crea el modelo LBPHFace con estructura en arbol.
- void **SlotCargaArbolFisherFace** (QString fn_model, int id_CSV)
  Funcion SlotCargaArbolFisherFace. Carga el modelo FisherFace con estructura en arbol.
- void **SlotCargaArbolEigenFace** (QString fn_model, int id_CSV)
  Funcion SlotCargaArbolEigenFace. Carga el modelo EigenFace con estructura en arbol.
- void **SlotCargaArbolLBPHFace** (QString fn_model, int id_CSV)
  Funcion SlotCargaArbolLBPHFace. Carga el modelo LBPHFace con estructura en arbol.
- void **SlotIdentificarArbol** (Mat *test_im)

Funcion SlotIdentificar. Identifica a un sujeto con un modelo con estructura en arbol.

Señales

- void **SignalFinAperturaCSV** ()
  Señal SignalFinAperturaCSV.
- void **SignalFinCreacionModelo** ()
  Señal SignalFinCreacionModelo.
- void **SignalFinCargaModelo** ()
  Señal SignalFinCargaModelo.
- void **SignalFinIdentificacion** (float conf, int predLabel)
  Señal SignalFinIdentificacion.

Métodos públicos

- **QProceso** ()
  *Constructor de la clase **QProceso**.*
- virtual **~QProceso** ()
  *Destructor de la clase **QProceso**.*

Métodos privados

- void **read_csv** (const string &filename, vector< Mat > &**images**, vector< int > &**labels**, char separator)
  Funcion read_csv. Lectura del archivo CSV.

Atributos privados

- vector< Mat > **images**
- vector< Mat > **im_arbol**
- Mat **transicion**
- vector< int > **labels**
- Ptr< FaceRecognizer > **model**
- vector< Ptr< FaceRecognizer > > **arbol**
- vector< int > **posiciones**
- int **num_saltos2**
- int **num_saltos5**
- int **num_images**
- int **num_modelos**
- int **im_size**
- int **tam_posiciones**
- int **predictedLabel**
- int **predictedLabelArbol**
- int **prox_model**
- double **confidence**
- ofstream **ficheroArbol**
- ifstream **ficheroArbolLec**
- bool **fin**

Descripción detallada

Hereda de QObject. Se encarga de la creacion, carga e identificacion llevada a cabo por los modelos.

Documentación del constructor y destructor

QProceso::QProceso ()

Constructor de la clase **QProceso**.

Constructor vacio.

```
14 {
15
16 }
```

QProceso::~QProceso () [virtual]

Destructor de la clase **QProceso**.

Libera la memoria reservada por images.

```
19 {
20     images.clear();
21 }
```

Documentación de las funciones miembro

void QProceso::read_csv (const string & *filename*, vector< Mat > & *images*, vector< int > & *labels*, char *separator*) [private]

Funcion read_csv. Lectura del archivo CSV.

Carga de las imagenes utilizadas para la creacion del modelo, asi como las relacion de las imagenes con las clases realizando la lectura de un fichero CSV.

Parámetros:

filename	Nombre del fichero CSV.
images	Vector en el que se cargaran las imagenes.
labels	Vector en el que se cargara la relacion imagenes - clases del identificador.
separator	Caracter con el cual se separa el nombre de la imagen a cargar con la clase que la identifica.

```
23
{
24
25     //Apertura del fichero
26     std::ifstream file(filename.c_str(), std::ifstream::in);
27     if (!file) {
28         string error_message = "No valid input file was given, please check the given
filename.";
29         CV_Error(CV_StsBadArg, error_message);
30     }
31     Mat cambio;
32     string line, path, classlabel;
33     //Lectura linea a linea
34     while (getline(file, line)) {
35         stringstream liness(line);
36         getline(liness, path, separator);
37         getline(liness, classlabel);
38         if(!path.empty() && !classlabel.empty()) {
39             //Apertura de la imagen
40             cambio=imread(path,0);
41             resize(cambio,cambio,Size(50,50));
42             //Incorporacion de la imagen y la clase de identificacion correpondiente
43             images.push_back(cambio);
44             labels.push_back(atoi(classlabel.c_str()));
45         }
46     }
47 }
```

void QProceso::SignalFinAperturaCSV () [signal]

Señal SignalFinAperturaCSV.

Señal que indica que se ha finalizado la apertura del fichero CSV.

void QProceso::SignalFinCargaModelo () [signal]

Señal SignalFinCargaModelo.

Señal que indica que se ha finalizado la carga de un modelo.

void QProceso::SignalFinCreacionModelo () [signal]

Señal SignalFinCreacionModelo.

Señal que indica que se ha finalizado la creacion de un modelo.

void QProceso::SignalFinIdentificacion (float *conf*, int *predLabel*) [signal]

Señal SignalFinCreacionIdentificacion.

Señal que indica que se ha finalizado la identificacion de un sujeto.

Parámetros:

conf	Confianza de la prediccion.
predLabel	Prediccion de la identificacion.

void QProceso::SlotAbrirCSV (QString *fn_csv*) [slot]

Funcion SlotAbrirCSV. Indica cuando se ha terminado la lectura del fichero CSV.

Llama a la funcion **read_csv()**; Indica que se ha finalizado con la lectura y carga del fichero CSV.

Parámetros:

fn_csv	Nombre del fichero CSV.

```
50 {
51     read_csv(fn_csv.toUtf8().constData(), images,labels,';');
52 
53     //Emision de la señal de fin de lectura del fichero CSV
54     emit SignalFinAperturaCSV();
55 }
```

void QProceso::SlotCargaArbolEigenFace (QString *fn_model*, int *id_CSV*) [slot]

Funcion SlotCargaArbolEigenFace. Carga el modelo EigenFace con estructura en arbol.

Carga el modelo EigenFace con estructura en arbol. Primero carga los modelos binarios. A continuacion, carga los modelos con cinco clases identificadoras. Una vez cargadas todas las ramas del arbol, se indica la finalizacion de la carga del arbol.

Parámetros:

fn_model	Nombre con el cual se cargaran los modelos de las ramas de arbol.
id_CSV	Identificador del fichero CSV.

```
1570 {
1571     //Apertura y lectura de las caracteristicas del modelo con estructura en arbol
1572     string line;
1573 
ficheroArbolLec.open(QString("etc/Resultados/50x50/ParamArbolEigenFace%1.txt").arg(id_CSV).to
Utf8().constData());
1574     getline(ficheroArbolLec,line);
1575     num_modelos=atoi(line.c_str());
1576     getline(ficheroArbolLec,line);
1577     num_saltos2=atoi(line.c_str());
1578     getline(ficheroArbolLec,line);
1579     num_saltos5=atoi(line.c_str());
1580     ficheroArbolLec.close();
1581     arbol.clear();
1582 
1583     //Carga de los modelos de cada rama que forma el arbol
1584     for(int i=0;i<num_modelos;i++)
1585     {
1586         model.release();
1587         model=createEigenFaceRecognizer();
1588         arbol.push_back(model);
1589         char intermedio[100]={0};
1590         sprintf(intermedio,fn_model.toUtf8().constData(),id_CSV,i);
1591         arbol[i]->load(intermedio);
```

```
1592     }
1593
1594     //Emision de la señal de fin de carga del modelo
1595     emit SignalFinCargaModelo();
1596 }
```

void QProceso::SlotCargaArbolFisherFace (QString *fn_model*, int *id_CSV*) [slot]

Funcion SlotCargaArbolFisherFace. Carga el modelo FisherFace con estructura en arbol.

Carga el modelo FisherFace con estructura en arbol. Primero carga los modelos binarios. A continuacion, carga los modelos con cinco clases identificadoras. Una vez cargadas todas las ramas del arbol, se indica la finalizacion de la carga del arbol.

Parámetros:

fn_model	Nombre con el cual se cargaran los modelos de las ramas de arbol.
id_CSV	Identificador del fichero CSV.

```
1542 {
1543     //Apertura y lectura de las caracteristicas del modelo con estructura en arbol
1544     string line;
1545
ficheroArbolLec.open(QString("etc/Resultados/50x50/ParamArbolFisherFace%1.txt").arg(id CSV).t
oUtf8().constData());
1546     getline(ficheroArbolLec,line);
1547     num_modelos=atoi(line.c_str());
1548     getline(ficheroArbolLec,line);
1549     num saltos2=atoi(line.c str());
1550     getline(ficheroArbolLec,line);
1551     num_saltos5=atoi(line.c_str());
1552     ficheroArbolLec.close();
1553
1554     //Carga de los modelos de cada rama que forma el arbol
1555     for(int i=0;i<num_modelos;i++)
1556     {
1557         model.release();
1558         model=createFisherFaceRecognizer();
1559         arbol.push_back(model);
1560         char intermedio[100]={0};
1561         sprintf(intermedio,fn model.toUtf8().constData(),id CSV,i);
1562         arbol[i]->load(intermedio);
1563     }
1564
1565     //Emision de la señal de fin de carga del modelo
1566     emit SignalFinCargaModelo();
1567 }
```

void QProceso::SlotCargaArbolLBPHFace (QString *fn_model*, int *id_CSV*) [slot]

Funcion SlotCargaArbolLBPHFace. Carga el modelo LBPHFace con estructura en arbol.

Carga el modelo LBPHFace con estructura en arbol. Primero carga los modelos binarios. A continuacion, carga los modelos con cinco clases identificadoras. Una vez cargadas todas las ramas del arbol, se indica la finalizacion de la carga del arbol.

Parámetros:

fn_model	Nombre con el cual se cargaran los modelos de las ramas de arbol.
id_CSV	Identificador del fichero CSV.

```
1599 {
1600     //Apertura y lectura de las caracteristicas del modelo con estructura en arbol
1601     string line;
1602
ficheroArbolLec.open(QString("etc/Resultados/50x50/ParamArbolLBPHFace%1.txt").arg(id_CSV).toU
tf8().constData());
1603     getline(ficheroArbolLec,line);
1604     num_modelos=atoi(line.c_str());
1605     getline(ficheroArbolLec,line);
1606     num_saltos2=atoi(line.c_str());
1607     getline(ficheroArbolLec,line);
1608     num_saltos5=atoi(line.c_str());
1609     ficheroArbolLec.close();
1610     arbol.clear();
```

```
1611
1612     //Carga de los modelos de cada rama que forma el arbol
1613     for(int i=0;i<num_modelos;i++)
1614     {
1615         model.release();
1616         model=createLBPHFaceRecognizer();
1617         arbol.push_back(model);
1618         char intermedio[100]={0};
1619         sprintf(intermedio,fn_model.toUtf8().constData(),id_CSV,i);
1620         arbol[i]->load(intermedio);
1621     }
1622
1623     //Emision de la señal de fin de carga del modelo
1624     emit SignalFinCargaModelo();
1625 }
```

void QProceso::SlotCargaModeloEigenFace (QString *fn_model*) [slot]

Funcion SlotCargaModeloEigenFace. Carga el modelo EigenFace.

Carga el modelo EigenFace creado anteriormente. Indica la finalizacion de la carga del modelo.

Parámetros:

fn_model	Nombre para cargar el modelo.

```
110 {
111     //Carga del modelo
112     model = createEigenFaceRecognizer();
113     model->load(fn_model.toUtf8().constData());
114
115     //Emision de la señal de fin de carga del modelo
116     emit SignalFinCargaModelo();
117 }
```

void QProceso::SlotCargaModeloFisherFace (QString *fn_model*) [slot]

Funcion SlotCargaModeloFisherFace. Carga el modelo FisherFace.

Carga el modelo FisherFace creado anteriormente. Indica la finalizacion de la carga del modelo.

Parámetros:

fn_model	Nombre para cargar el modelo.

```
100 {
101     //Carga del modelo
102     model = createFisherFaceRecognizer();
103     model->load(fn_model.toUtf8().constData());
104
105     //Emision de la señal de fin de carga del modelo
106     emit SignalFinCargaModelo();
107 }
```

void QProceso::SlotCargaModeloLBPHFace (QString *fn_model*) [slot]

Funcion SlotCargaModeloLBPHFace. Carga el modelo LBPHFace.

Carga el modelo LBPHFace creado anteriormente. Indica la finalizacion de la carga del modelo.

Parámetros:

fn_model	Nombre para cargar el modelo.

```
120 {
121     //Carga del modelo
122     model = createLBPHFaceRecognizer();
123     model->load(fn_model.toUtf8().constData());
124
125     //Emision de la señal de fin de carga del modelo
126     emit SignalFinCargaModelo();
127 }
```

void QProceso::SlotCreaArbolEigenFace (QString *fn_model*, int *id_CSV*) [slot]

Funcion SlotCreaArbolEigenFace. Crea el modelo EigenFace con estructura en arbol.

Crea el modelo EigenFace con estructura en arbol. Primero crea los modelos binarios. A continuacion, crea los modelos con cinco clases identificadoras. Una vez creadas todas las ramas del arbol, procede a guardar el arbol completo. Por ultimo, indica la finalizacion de la creacion del arbol.

Parámetros:

fn_model	Nombre con el cual se guardaran las ramas de arbol.
id_CSV	Identificador del fichero CSV.

```
608 {
609     num_saltos2=0;
610     num_saltos5=0;
611     num_modelos=0;
612 
613     //Decision de una imagen por clase o dos imagenes por clase
614     if(id_CSV%2!=0)
615     {
616         im_size=images.size()/2;
617         num_images=images.size()/2;
618     }
619     else
620     {
621         im_size=images.size();
622         num_images=images.size();
623     }
624 
625     //Creacion de las ramas de modelo binario
626     while(num_images%2==0&&(int)posiciones.size()!=im_size)
627     {
628         //Relacion una imagen por clase
629         if(id_CSV%2==0)
630         {
631             labels.clear();
632             for(int i=0;i<(num_images)/2;i++)
633                 labels.push_back(0);
634             for(int i=0;i<(num_images)/2;i++)
635                 labels.push_back(1);
636         }
637         //Relacion dos imagenes por clase
638         else
639         {
640             labels.clear();
641             for(int i=0;i<(num_images)/2;i++)
642             {
643                 labels.push_back(0);
644                 labels.push_back(0);
645             }
646             for(int i=0;i<(num_images)/2;i++)
647             {
648                 labels.push_back(1);
649                 labels.push_back(1);
650             }
651         }
652         //Creacion del modelo mujer - hombre
653         if(num_saltos2==0)
654         {
655             model = createEigenFaceRecognizer();
656             arbol.push_back(model);
657             arbol[num_modelos]->train(images,labels);
658             num_modelos++;
659             num_saltos2++;
660             posiciones.push_back(im_size/2-1);
661             posiciones.push_back(im_size-1);
662             num_images/=2;
663         }
664         //Creacion del resto de ramas binarias
665         else
666         {
667             for(uint i=0;i<(posiciones.size()/2);i++)
668             {
669                 //Seleccion una imagen por clase
670                 if(id_CSV%2==0)
671                 {
672                     for(int j=(2*num_images*i);j<=posiciones[2*i];j++)
673                     {
674                         images[j].copyTo(transicion);
675                         im_arbol.push_back(transicion);
```

```
676                         transicion.release();
677                     }
678                 }
679                 //Seleccion dos imagenes por clase
680                 else
681                 {
682                     for(int j=(2*num_images*i);j<=posiciones[2*i];j++)
683                     {
684                         images[2*j].copyTo(transicion);
685                         im_arbol.push_back(transicion);
686                         transicion.release();
687                         images[2*j+1].copyTo(transicion);
688                         im arbol.push back(transicion);
689                         transicion.release();
690                     }
691                 }
692                 //Creacion modelo binario
693                 model.release();
694                 model = createEigenFaceRecognizer();
695                 arbol.push_back(model);
696                 arbol[num modelos]->train(im arbol,labels);
697                 num_modelos++;
698                 im_arbol.clear();
699                 //Seleccion una imagen por clase
700                 if(id CSV%2==0)
701                 {
702                     for(int j=posiciones[2*i]+1;j<=posiciones[2*i+1];j++)
703                     {
704                         images[j].copyTo(transicion);
705                         im_arbol.push_back(transicion);
706                         transicion.release();
707                     }
708                 }
709                 //Seleccion dos imagenes por clase
710                 else
711                 {
712                     for(int j=posiciones[2*i]+1;j<=posiciones[2*i+1];j++)
713                     {
714                         images[2*j].copyTo(transicion);
715                         im_arbol.push_back(transicion);
716                         transicion.release();
717                         images[2*j+1].copyTo(transicion);
718                         im_arbol.push_back(transicion);
719                         transicion.release();
720                     }
721                 }
722                 //Creacion modelo binario
723                 model.release();
724                 model = createEigenFaceRecognizer();
725                 arbol.push_back(model);
726                 arbol[num_modelos]->train(im_arbol,labels);
727                 num_modelos++;
728                 im_arbol.clear();
729             }
730             //Tramos para los siguientes modelos
731             tam_posiciones=posiciones.size();
732             for(int i=0;i<(tam_posiciones/2);i++)
733             {
734                 posiciones.push back(((posiciones[2*i]+1)/2)-1);
735                 posiciones.push_back(posiciones[tam_posiciones+2*i]+num_images);
736             }
737             std::sort(posiciones.begin(),posiciones.end());
738             num saltos2++;
739             num_images/=2;
740         }
741     }
742
743     //Creacion de las ramas de modelo con cinco clases identificadoras
744     while(num_images%5==0&&(int)posiciones.size()!=im_size)
745     {
746         //Relacion una imagen por clase
747         if(id_CSV%2==0)
748         {
749             labels.clear();
750             for(int i=0;i<(num images)/5;i++)
751                 labels.push_back(0);
752             for(int i=0;i<(num_images)/5;i++)
753                 labels.push_back(1);
754             for(int i=0;i<(num_images)/5;i++)
```

```
755                 labels.push_back(2);
756             for(int i=0;i<(num_images)/5;i++)
757                 labels.push_back(3);
758             for(int i=0;i<(num_images)/5;i++)
759                 labels.push_back(4);
760         }
761         //Relacion dos imagenes por clase
762         else
763         {
764             labels.clear();
765             for(int i=0;i<(num_images)/5;i++)
766             {
767                 labels.push_back(0);
768                 labels.push_back(0);
769             }
770             for(int i=0;i<(num_images)/5;i++)
771             {
772                 labels.push_back(1);
773                 labels.push_back(1);
774             }
775             for(int i=0;i<(num_images)/5;i++)
776             {
777                 labels.push_back(2);
778                 labels.push_back(2);
779             }
780             for(int i=0;i<(num_images)/5;i++)
781             {
782                 labels.push_back(3);
783                 labels.push_back(3);
784             }
785             for(int i=0;i<(num_images)/5;i++)
786             {
787                 labels.push_back(4);
788                 labels.push_back(4);
789             }
790         }
791         //Creacion de los modelos de cinco clases despues de las ramas binarias
792         if(num_saltos5==0)
793         {
794             for(uint i=0;i<(posiciones.size()/2);i++)
795             {
796                 //Seleccion una imagen por clase
797                 if(id_CSV%2==0)
798                 {
799                     for(int j=(2*num_images*i);j<=posiciones[2*i];j++)
800                     {
801                         images[j].copyTo(transicion);
802                         im_arbol.push_back(transicion);
803                         transicion.release();
804                     }
805                 }
806                 //Seleccion dos imagenes por clase
807                 else
808                 {
809                     for(int j=(2*num_images*i);j<=posiciones[2*i];j++)
810                     {
811                         images[2*j].copyTo(transicion);
812                         im_arbol.push_back(transicion);
813                         transicion.release();
814                         images[2*j+1].copyTo(transicion);
815                         im_arbol.push_back(transicion);
816                         transicion.release();
817                     }
818                 }
819                 //Creacion del modelo de cinco clases identificadoras
820                 model.release();
821                 model = createEigenFaceRecognizer();
822                 arbol.push_back(model);
823                 arbol[num_modelos]->train(im_arbol,labels);
824                 num_modelos++;
825                 im_arbol.clear();
826                 //Seleccion una imagen por clase
827                 if(id_CSV%2==0)
828                 {
829                     for(int j=posiciones[2*i]+1;j<=posiciones[2*i+1];j++)
830                     {
831                         images[j].copyTo(transicion);
```

```
832                         im_arbol.push_back(transicion);
833                         transicion.release();
834                     }
835                 }
836                 //Seleccion dos imagenes por clase
837                 else
838                 {
839                     for(int j=posiciones[2*i]+1;j<=posiciones[2*i+1];j++)
840                     {
841                         images[2*j].copyTo(transicion);
842                         im_arbol.push_back(transicion);
843                         transicion.release();
844                         images[2*j+1].copyTo(transicion);
845                         im_arbol.push_back(transicion);
846                         transicion.release();
847                     }
848                 }
849                 //Creacion del modelo de cinco clases identificadoras
850                 model.release();
851                 model = createEigenFaceRecognizer();
852                 arbol.push back(model);
853                 arbol[num_modelos]->train(im_arbol,labels);
854                 num_modelos++;
855                 im_arbol.clear();
856             }
857             //Tramos para los siguientes modelos
858             tam_posiciones=posiciones.size();
859             for(int i=0;i<(tam_posiciones-1);i++)
860             {
861                 if(i==0)
862                 {
863                     posiciones.push_back(((posiciones[0]+1)/5)-1);
864                     posiciones.push_back((2*(posiciones[0]+1)/5)-1);
865                     posiciones.push_back((3*(posiciones[0]+1)/5)-1);
866                     posiciones.push_back((4*(posiciones[0]+1)/5)-1);
867                 }
868                 posiciones.push_back(((posiciones[0]+1)/5)+posiciones[i]);
869                 posiciones.push_back((2*(posiciones[0]+1)/5)+posiciones[i]);
870                 posiciones.push_back((3*(posiciones[0]+1)/5)+posiciones[i]);
871                 posiciones.push_back((4*(posiciones[0]+1)/5)+posiciones[i]);
872             }
873             std::sort(posiciones.begin(),posiciones.end());
874             num_saltos5++;
875             num_images/=5;
876         }
877         //Creacion de los modelos de cinco clases despues de una rama de un modelo con
cinco clases
878         else
879         {
880             for(uint i=0;i<(posiciones.size()/5);i++)
881             {
882                 //Seleccion una imagen por clase
883                 if(id_CSV%2==0)
884                 {
885                     for(int j=(5*num_images*i);j<=posiciones[5*i];j++)
886                     {
887                         images[j].copyTo(transicion);
888                         im_arbol.push_back(transicion);
889                         transicion.release();
890                     }
891                 }
892                 //Seleccion dos imagenes por clase
893                 else
894                 {
895                     for(int j=(5*num_images*i);j<=posiciones[5*i];j++)
896                     {
897                         images[2*j].copyTo(transicion);
898                         im_arbol.push_back(transicion);
899                         transicion.release();
900                         images[2*j+1].copyTo(transicion);
901                         im_arbol.push_back(transicion);
902                         transicion.release();
903                     }
904                 }
905                 //Creacion del modelo de cinco clases identificadoras
906                 model.release();
907                 model = createEigenFaceRecognizer();
908                 arbol.push_back(model);
909                 arbol[num_modelos]->train(im_arbol,labels);
```

```
910                 num_modelos++;
911                 im_arbol.clear();
912                 //Seleccion una imagen por clase
913                 if(id_CSV%2==0)
914                 {
915                     for(int j=posiciones[5*i]+1;j<=posiciones[5*i+1];j++)
916                     {
917                         images[j].copyTo(transicion);
918                         im_arbol.push_back(transicion);
919                         transicion.release();
920                     }
921                 }
922                 //Seleccion dos imagenes por clase
923                 else
924                 {
925                     for(int j=posiciones[5*i]+1;j<=posiciones[5*i+1];j++)
926                     {
927                         images[2*j].copyTo(transicion);
928                         im arbol.push back(transicion);
929                         transicion.release();
930                         images[2*j+1].copyTo(transicion);
931                         im_arbol.push_back(transicion);
932                         transicion.release();
933                     }
934                 }
935                 //Creacion del modelo de cinco clases identificadoras
936                 model.release();
937                 model = createEigenFaceRecognizer();
938                 arbol.push_back(model);
939                 arbol[num_modelos]->train(im_arbol,labels);
940                 num modelos++;
941                 im_arbol.clear();
942                 //Seleccion una imagen por clase
943                 if(id_CSV%2==0)
944                 {
945                     for(int j=posiciones[5*i+1]+1;j<=posiciones[5*i+2];j++)
946                     {
947                         images[j].copyTo(transicion);
948                         im_arbol.push_back(transicion);
949                         transicion.release();
950                     }
951                 }
952                 //Seleccion dos imagenes por clase
953                 else
954                 {
955                     for(int j=posiciones[5*i+1]+1;j<=posiciones[5*i+2];j++)
956                     {
957                         images[2*j].copyTo(transicion);
958                         im_arbol.push_back(transicion);
959                         transicion.release();
960                         images[2*j+1].copyTo(transicion);
961                         im arbol.push back(transicion);
962                         transicion.release();
963                     }
964                 }
965                 //Creacion del modelo de cinco clases identificadoras
966                 model.release();
967                 model = createEigenFaceRecognizer();
968                 arbol.push_back(model);
969                 arbol[num modelos]->train(im arbol,labels);
970                 num_modelos++;
971                 im_arbol.clear();
972                 //Seleccion una imagen por clase
973                 if(id_CSV%2==0)
974                 {
975                     for(int j=posiciones[5*i+2]+1;j<=posiciones[5*i+3];j++)
976                     {
977                         images[j].copyTo(transicion);
978                         im arbol.push back(transicion);
979                         transicion.release();
980                     }
981                 }
982                 //Seleccion dos imagenes por clase
983                 else
984                 {
985                     for(int j=posiciones[5*i+2]+1;j<=posiciones[5*i+3];j++)
986                     {
```

```
                        images[2*j].copyTo(transicion);
                        im_arbol.push_back(transicion);
                        transicion.release();
                        images[2*j+1].copyTo(transicion);
                        im_arbol.push_back(transicion);
                        transicion.release();
                    }
                }
                //Creacion del modelo de cinco clases identificadoras
                model.release();
                model = createEigenFaceRecognizer();
                arbol.push_back(model);
                arbol[num_modelos]->train(im_arbol,labels);
                 num_modelos++;
                 im_arbol.clear();
                 //Seleccion una imagen por clase
                 if(id_CSV%2==0)
                 {
                     for(int j=posiciones[5*i+3]+1;j<=posiciones[5*i+4];j++)
                     {
                         images[j].copyTo(transicion);
                         im_arbol.push_back(transicion);
                         transicion.release();
                     }
                 }
                 //Seleccion dos imagenes por clase
                 else
                 {
                     for(int j=posiciones[5*i+3]+1;j<=posiciones[5*i+4];j++)
                     {
                         images[2*j].copyTo(transicion);
                         im_arbol.push_back(transicion);
                         transicion.release();
                         images[2*j+1].copyTo(transicion);
                         im_arbol.push_back(transicion);
                         transicion.release();
                     }
                 }
                 //Creacion del modelo de cinco clases identificadoras
                 model.release();
                 model = createEigenFaceRecognizer();
                 arbol.push back(model);
                 arbol[num_modelos]->train(im_arbol,labels);
                 num_modelos++;
                 im_arbol.clear();
             }
             //Tramos para los siguientes modelos
             tam_posiciones=posiciones.size();
             for(int i=0;i<(tam_posiciones-1);i++)
             {
                 if(i==0)
                 {
                     posiciones.push_back(((posiciones[0]+1)/5)-1);
                     posiciones.push back((2*(posiciones[0]+1)/5)-1);
                     posiciones.push_back((3*(posiciones[0]+1)/5)-1);
                     posiciones.push_back((4*(posiciones[0]+1)/5)-1);
                 }
                 posiciones.push_back(((posiciones[0]+1)/5)+posiciones[i]);
                 posiciones.push back((2*(posiciones[0]+1)/5)+posiciones[i]);
                 posiciones.push_back((3*(posiciones[0]+1)/5)+posiciones[i]);
                 posiciones.push_back((4*(posiciones[0]+1)/5)+posiciones[i]);
             }
             std::sort(posiciones.begin(),posiciones.end());
             num_saltos5++;
             num_images/=5;
         }
     }

     //Modelo arbol creado completo. Se procede a guardarlo
     for(uint i=0;i<arbol.size();i++)
     {
         char intermedio[100]={0};
         sprintf(intermedio,fn_model.toUtf8().constData(),id_CSV,i);
         arbol[i]->save(intermedio);
         }
     arbol.clear();
     char buf[40]={0};
     sprintf(  buf, "%d\n%d\n%d", num_modelos, num_saltos2, num_saltos5);
```

```
1065
ficheroArbol.open(QString("etc/Resultados/50x50/ParamArbolEigenFace%1.txt").arg(id_CSV).toUtf
8().constData());
1066     ficheroArbol<<buf;
1067     ficheroArbol.close();
1068     posiciones.clear();
1069
1070     //Emision de la señal de fin de creacion del modelo
1071     emit SignalFinCreacionModelo();
1072 }
```

void QProceso::SlotCreaArbolFisherFace (QString *fn_model*, int *id_CSV*) [slot]

Funcion SlotCreaArbolFisherFace. Crea el modelo FisherFace con estructura en arbol.

Crea el modelo FisherFace con estructura en arbol. Primero crea los modelos binarios. A continuacion, crea los modelos con cinco clases identificadoras. Una vez creadas todas las ramas del arbol, procede a guardar el arbol completo. Por ultimo, indica la finalizacion de la creacion del arbol.

Parámetros:

fn_model	Nombre con el cual se guardaran las ramas de arbol.
id_CSV	Identificador del fichero CSV.

```
141 {
142     num_saltos2=0;
143     num_saltos5=0;
144     num_modelos=0;
145
146     //Decision de una imagen por clase o dos imagenes por clase
147     if(id_CSV%2!=0)
148     {
149         im_size=images.size()/2;
150         num_images=images.size()/2;
151     }
152     else
153     {
154         im_size=images.size();
155         num_images=images.size();
156     }
157
158     //Creacion de las ramas de modelo binario
159     while(num_images%2==0&&(int)posiciones.size()!=im_size)
160     {
161         //Relacion una imagen por clase
162         if(id_CSV%2==0)
163         {
164             labels.clear();
165             for(int i=0;i<(num_images)/2;i++)
166                 labels.push_back(0);
167             for(int i=0;i<(num_images)/2;i++)
168                 labels.push_back(1);
169         }
170         //Relacion dos imagenes por clase
171         else
172         {
173             labels.clear();
174             for(int i=0;i<(num images)/2;i++)
175             {
176                 labels.push_back(0);
177                 labels.push_back(0);
178             }
179             for(int i=0;i<(num_images)/2;i++)
180             {
181                 labels.push_back(1);
182                 labels.push_back(1);
183             }
184         }
185         //Creacion del modelo mujer - hombre
186         if(num_saltos2==0)
187         {
188             model = createFisherFaceRecognizer();
189             arbol.push_back(model);
190             arbol[num_modelos]->train(images,labels);
191             num_modelos++;
192             num_saltos2++;
```

```
193             posiciones.push_back(im_size/2-1);
194             posiciones.push_back(im_size-1);
195             num_images/=2;
196         }
197         //Creacion del resto de ramas binarias
198         else
199         {
200             for(uint i=0;i<(posiciones.size()/2);i++)
201             {
202                 //Seleccion una imagen por clase
203                 if(id_CSV%2==0)
204                 {
205                     for(int j=(2*num images*i);j<=posiciones[2*i];j++)
206                     {
207                         images[j].copyTo(transicion);
208                         im_arbol.push_back(transicion);
209                         transicion.release();
210                     }
211                 }
212                 //Seleccion dos imagenes por clase
213                 else
214                 {
215                     for(int j=(2*num_images*i);j<=posiciones[2*i];j++)
216                     {
217                         images[2*j].copyTo(transicion);
218                         im_arbol.push_back(transicion);
219                         transicion.release();
220                         images[2*j+1].copyTo(transicion);
221                         im_arbol.push_back(transicion);
222                         transicion.release();
223                     }
224                 }
225                 //Creacion modelo binario
226                 model.release();
227                 model = createFisherFaceRecognizer();
228                 arbol.push_back(model);
229                 arbol[num_modelos]->train(im_arbol,labels);
230                 num_modelos++;
231                 im_arbol.clear();
232                 //Seleccion una imagen por clase
233                 if(id_CSV%2==0)
234                 {
235                     for(int j=posiciones[2*i]+1;j<=posiciones[2*i+1];j++)
236                     {
237                         images[j].copyTo(transicion);
238                         im arbol.push back(transicion);
239                         transicion.release();
240                     }
241                 }
242                 //Seleccion dos imagenes por clase
243                 else
244                 {
245                     for(int j=posiciones[2*i]+1;j<=posiciones[2*i+1];j++)
246                     {
247                         images[2*j].copyTo(transicion);
248                         im_arbol.push_back(transicion);
249                         transicion.release();
250                         images[2*j+1].copyTo(transicion);
251                         im arbol.push back(transicion);
252                         transicion.release();
253                     }
254                 }
255                 //Creacion modelo binario
256                 model.release();
257                 model = createFisherFaceRecognizer();
258                 arbol.push_back(model);
259                 arbol[num_modelos]->train(im_arbol,labels);
260                 num_modelos++;
261                 im_arbol.clear();
262             }
263             //Tramos para los siguientes modelos
264             tam_posiciones=posiciones.size();
265             for(int i=0;i<(tam_posiciones/2);i++)
266             {
267                 posiciones.push back(((posiciones[2*i]+1)/2)-1);
268                 posiciones.push_back(posiciones[tam_posiciones+2*i]+num_images);
269             }
270             std::sort(posiciones.begin(),posiciones.end());
271             num_saltos2++;
```

```
272             num_images/=2;
273         }
274     }
275
276     //Creacion de las ramas de modelo con cinco clases identificadoras
277     while(num_images%5==0&&(int)posiciones.size()!=im_size)
278     {
279         //Relacion una imagen por clase
280         if(id_CSV%2==0)
281         {
282             labels.clear();
283             for(int i=0;i<(num_images)/5;i++)
284                 labels.push_back(0);
285             for(int i=0;i<(num_images)/5;i++)
286                 labels.push back(1);
287             for(int i=0;i<(num_images)/5;i++)
288                 labels.push_back(2);
289             for(int i=0;i<(num_images)/5;i++)
290                 labels.push back(3);
291             for(int i=0;i<(num_images)/5;i++)
292                 labels.push_back(4);
293         }
294         //Relacion dos imagenes por clase
295         else
296         {
297             labels.clear();
298             for(int i=0;i<(num_images)/5;i++)
299             {
300                 labels.push_back(0);
301                 labels.push_back(0);
302             }
303             for(int i=0;i<(num_images)/5;i++)
304             {
305                 labels.push_back(1);
306                 labels.push_back(1);
307             }
308             for(int i=0;i<(num_images)/5;i++)
309             {
310                 labels.push_back(2);
311                 labels.push_back(2);
312             }
313             for(int i=0;i<(num_images)/5;i++)
314             {
315                 labels.push_back(3);
316                 labels.push_back(3);
317             }
318             for(int i=0;i<(num_images)/5;i++)
319             {
320                 labels.push_back(4);
321                 labels.push_back(4);
322             }
323         }
324         //Creacion de los modelos de cinco clases despues de las ramas binarias
325         if(num_saltos5==0)
326         {
327             for(uint i=0;i<(posiciones.size()/2);i++)
328             {
329                 //Seleccion una imagen por clase
330                 if(id_CSV%2==0)
331                 {
332                     for(int j=(2*num_images*i);j<=posiciones[2*i];j++)
333                     {
334                         images[j].copyTo(transicion);
335                         im_arbol.push_back(transicion);
336                         transicion.release();
337                     }
338                 }
339                 //Seleccion dos imagenes por clase
340                 else
341                 {
342                     for(int j=(2*num_images*i);j<=posiciones[2*i];j++)
343                     {
344                         images[2*j].copyTo(transicion);
345                         im_arbol.push_back(transicion);
346                         transicion.release();
347                         images[2*j+1].copyTo(transicion);
348                         im_arbol.push_back(transicion);
```

```
                        transicion.release();
                    }
                }
                //Creacion del modelo de cinco clases identificadoras
                model.release();
                model = createFisherFaceRecognizer();
                arbol.push_back(model);
                arbol[num_modelos]->train(im_arbol,labels);
                num_modelos++;
                im_arbol.clear();
                //Seleccion una imagen por clase
                if(id_CSV%2==0)
                {
                    for(int j=posiciones[2*i]+1;j<=posiciones[2*i+1];j++)
                    {
                        images[j].copyTo(transicion);
                        im_arbol.push_back(transicion);
                        transicion.release();
                    }
                }
                //Seleccion dos imagenes por clase
                else
                {
                    for(int j=posiciones[2*i]+1;j<=posiciones[2*i+1];j++)
                    {
                        images[2*j].copyTo(transicion);
                        im_arbol.push_back(transicion);
                        transicion.release();
                        images[2*j+1].copyTo(transicion);
                        im_arbol.push_back(transicion);
                        transicion.release();
                    }
                }
                //Creacion del modelo de cinco clases identificadoras
                model.release();
                model = createFisherFaceRecognizer();
                arbol.push_back(model);
                arbol[num_modelos]->train(im_arbol,labels);
                num_modelos++;
                im_arbol.clear();
            }
            //Tramos para los siguientes modelos
            tam_posiciones=posiciones.size();
            for(int i=0;i<(tam_posiciones-1);i++)
            {
                if(i==0)
                {
                    posiciones.push_back(((posiciones[0]+1)/5)-1);
                    posiciones.push_back((2*(posiciones[0]+1)/5)-1);
                    posiciones.push_back((3*(posiciones[0]+1)/5)-1);
                    posiciones.push_back((4*(posiciones[0]+1)/5)-1);
                }
                posiciones.push_back(((posiciones[0]+1)/5)+posiciones[i]);
                posiciones.push_back((2*(posiciones[0]+1)/5)+posiciones[i]);
                posiciones.push_back((3*(posiciones[0]+1)/5)+posiciones[i]);
                posiciones.push_back((4*(posiciones[0]+1)/5)+posiciones[i]);
            }
            std::sort(posiciones.begin(),posiciones.end());
            num_saltos5++;
            num_images/=5;
        }
        //Creacion de los modelos de cinco clases despues de una rama de un modelo con
cinco clases
        else
        {
            for(uint i=0;i<(posiciones.size()/5);i++)
            {
                //Seleccion una imagen por clase
                if(id_CSV%2==0)
                {
                    for(int j=(5*num_images*i);j<=posiciones[5*i];j++)
                    {
                        images[j].copyTo(transicion);
                        im_arbol.push_back(transicion);
                        transicion.release();
                    }
                }
                //Seleccion dos imagenes por clase
                else
```

```
427                 {
428                     for(int j=(5*num_images*i);j<=posiciones[5*i];j++)
429                     {
430                         images[2*j].copyTo(transicion);
431                         im_arbol.push_back(transicion);
432                         transicion.release();
433                         images[2*j+1].copyTo(transicion);
434                         im_arbol.push_back(transicion);
435                         transicion.release();
436                     }
437                 }
438                 //Creacion del modelo de cinco clases identificadoras
439                 model.release();
440                 model = createFisherFaceRecognizer();
441                 arbol.push back(model);
442                 arbol[num_modelos]->train(im_arbol,labels);
443                 num_modelos++;
444                 im_arbol.clear();
445                 //Seleccion una imagen por clase
446                 if(id_CSV%2==0)
447                 {
448                     for(int j=posiciones[5*i]+1;j<=posiciones[5*i+1];j++)
449                     {
450                         images[j].copyTo(transicion);
451                         im_arbol.push_back(transicion);
452                         transicion.release();
453                     }
454                 }
455                 //Seleccion dos imagenes por clase
456                 else
457                 {
458                     for(int j=posiciones[5*i]+1;j<=posiciones[5*i+1];j++)
459                     {
460                         images[2*j].copyTo(transicion);
461                         im_arbol.push_back(transicion);
462                         transicion.release();
463                         images[2*j+1].copyTo(transicion);
464                         im_arbol.push_back(transicion);
465                         transicion.release();
466                     }
467                 }
468                 //Creacion del modelo de cinco clases identificadoras
469                 model.release();
470                 model = createFisherFaceRecognizer();
471                 arbol.push_back(model);
472                 arbol[num_modelos]->train(im_arbol,labels);
473                 num_modelos++;
474                 im arbol.clear();
475                 //Seleccion una imagen por clase
476                 if(id_CSV%2==0)
477                 {
478                     for(int j=posiciones[5*i+1]+1;j<=posiciones[5*i+2];j++)
479                     {
480                         images[j].copyTo(transicion);
481                         im_arbol.push_back(transicion);
482                         transicion.release();
483                     }
484                 }
485                 //Seleccion dos imagenes por clase
486                 else
487                 {
488                     for(int j=posiciones[5*i+1]+1;j<=posiciones[5*i+2];j++)
489                     {
490                         images[2*j].copyTo(transicion);
491                         im_arbol.push_back(transicion);
492                         transicion.release();
493                         images[2*j+1].copyTo(transicion);
494                         im_arbol.push_back(transicion);
495                         transicion.release();
496                     }
497                 }
498                 //Creacion del modelo de cinco clases identificadoras
499                 model.release();
500                 model = createFisherFaceRecognizer();
501                 arbol.push_back(model);
502                 arbol[num_modelos]->train(im_arbol,labels);
503                 num_modelos++;
```

```
504                 im_arbol.clear();
505                 //Seleccion una imagen por clase
506                 if(id_CSV%2==0)
507                 {
508                     for(int j=posiciones[5*i+2]+1;j<=posiciones[5*i+3];j++)
509                     {
510                         images[j].copyTo(transicion);
511                         im arbol.push back(transicion);
512                         transicion.release();
513                     }
514                 }
515                 //Seleccion dos imagenes por clase
516                 else
517                 {
518                     for(int j=posiciones[5*i+2]+1;j<=posiciones[5*i+3];j++)
519                     {
520                         images[2*j].copyTo(transicion);
521                         im_arbol.push_back(transicion);
522                         transicion.release();
523                         images[2*j+1].copyTo(transicion);
524                         im arbol.push back(transicion);
525                         transicion.release();
526                     }
527                 }
528                 //Creacion del modelo de cinco clases identificadoras
529                 model.release();
530                 model = createFisherFaceRecognizer();
531                 arbol.push_back(model);
532                 arbol[num_modelos]->train(im_arbol,labels);
533                 num_modelos++;
534                 im_arbol.clear();
535                 //Seleccion una imagen por clase
536                 if(id_CSV%2==0)
537                 {
538                     for(int j=posiciones[5*i+3]+1;j<=posiciones[5*i+4];j++)
539                     {
540                         images[j].copyTo(transicion);
541                         im_arbol.push_back(transicion);
542                         transicion.release();
543                     }
544                 }
545                 //Seleccion dos imagenes por clase
546                 else
547                 {
548                     for(int j=posiciones[5*i+3]+1;j<=posiciones[5*i+4];j++)
549                     {
550                         images[2*j].copyTo(transicion);
551                         im_arbol.push_back(transicion);
552                         transicion.release();
553                         images[2*j+1].copyTo(transicion);
554                         im_arbol.push_back(transicion);
555                         transicion.release();
556                     }
557                 }
558                 //Creacion del modelo de cinco clases identificadoras
559                 model.release();
560                 model = createFisherFaceRecognizer();
561                 arbol.push_back(model);
562                 arbol[num modelos]->train(im arbol,labels);
563                 num_modelos++;
564                 im_arbol.clear();
565             }
566             //Tramos para los siguientes modelos
567             tam_posiciones=posiciones.size();
568             for(int i=0;i<(tam_posiciones-1);i++)
569             {
570                 if(i==0)
571                 {
572                     posiciones.push_back(((posiciones[0]+1)/5)-1);
573                     posiciones.push_back((2*(posiciones[0]+1)/5)-1);
574                     posiciones.push_back((3*(posiciones[0]+1)/5)-1);
575                     posiciones.push_back((4*(posiciones[0]+1)/5)-1);
576                 }
577                 posiciones.push_back(((posiciones[0]+1)/5)+posiciones[i]);
578                 posiciones.push_back((2*(posiciones[0]+1)/5)+posiciones[i]);
579                 posiciones.push_back((3*(posiciones[0]+1)/5)+posiciones[i]);
580                 posiciones.push_back((4*(posiciones[0]+1)/5)+posiciones[i]);
581             }
582             std::sort(posiciones.begin(),posiciones.end());
```

```
583             num_saltos5++;
584             num_images/=5;
585         }
586     }
587
588     //Modelo arbol creado completo. Se procede a guardarlo
589     for(uint i=0;i<arbol.size();i++)
590     {
591         char intermedio[100]={0};
592         sprintf(intermedio,fn model.toUtf8().constData(),id CSV,i);
593         arbol[i]->save(intermedio);
594     }
595     arbol.clear();
596     char buf[40]={0};
597     sprintf(  buf, "%d\n%d\n%d\n", num modelos, num saltos2, num saltos5);
598
ficheroArbol.open(QString("etc/Resultados/50x50/ParamArbolFisherFace%1.txt").arg(id_CSV).toUt
f8().constData());
599     ficheroArbol<<buf;
600     ficheroArbol.close();
601     posiciones.clear();
602
603     //Emision de la señal de fin de creacion del modelo
604     emit SignalFinCreacionModelo();
605 }
```

void QProceso::SlotCreaArbolLBPHFace (QString *fn_model*, int *id_CSV*) [slot]

Funcion SlotCreaArbolLBPHFace. Crea el modelo LBPHFace con estructura en arbol.

Crea el modelo LBPHFace con estructura en arbol. Primero crea los modelos binarios. A continuacion, crea los modelos con cinco clases identificadoras. Una vez creadas todas las ramas del arbol, procede a guardar el arbol completo. Por ultimo, indica la finalizacion de la creacion del arbol.

Parámetros:

fn_model	Nombre con el cual se guardaran las ramas de arbol.
id_CSV	Identificador del fichero CSV.

```
1075 {
1076     num saltos2=0;
1077     num_saltos5=0;
1078     num_modelos=0;
1079
1080     //Decision de una imagen por clase o dos imagenes por clase
1081     if(id_CSV%2!=0)
1082     {
1083         im_size=images.size()/2;
1084         num_images=images.size()/2;
1085     }
1086     else
1087     {
1088         im_size=images.size();
1089         num_images=images.size();
1090     }
1091
1092     //Creacion de las ramas de modelo binario
1093     while(num images%2==0&&(int)posiciones.size()!=im size)
1094     {
1095         //Relacion una imagen por clase
1096         if(id_CSV%2==0)
1097         {
1098             labels.clear();
1099             for(int i=0;i<(num_images)/2;i++)
1100                 labels.push_back(0);
1101             for(int i=0;i<(num_images)/2;i++)
1102                 labels.push_back(1);
1103         }
1104         //Relacion dos imagenes por clase
1105         else
1106         {
1107             labels.clear();
1108             for(int i=0;i<(num_images)/2;i++)
1109             {
1110                 labels.push_back(0);
1111                 labels.push_back(0);
```

```
            }
            for(int i=0;i<(num_images)/2;i++)
            {
                labels.push_back(1);
                labels.push_back(1);
            }
        }
        //Creacion del modelo mujer - hombre
        if(num_saltos2==0)
        {
            model = createLBPHFaceRecognizer();
            arbol.push_back(model);
            arbol[num_modelos]->train(images,labels);
            num_modelos++;
            num_saltos2++;
            posiciones.push_back(im_size/2-1);
            posiciones.push_back(im_size-1);
            num_images/=2;
        }
        //Creacion del resto de ramas binarias
        else
        {
            for(uint i=0;i<(posiciones.size()/2);i++)
            {
                //Seleccion una imagen por clase
                if(id_CSV%2==0)
                {
                    for(int j=(2*num_images*i);j<=posiciones[2*i];j++)
                    {
                        images[j].copyTo(transicion);
                        im_arbol.push_back(transicion);
                        transicion.release();
                    }
                }
                //Seleccion dos imagen por clase
                else
                {
                    for(int j=(2*num_images*i);j<=posiciones[2*i];j++)
                    {
                        images[2*j].copyTo(transicion);
                        im_arbol.push_back(transicion);
                        transicion.release();
                        images[2*j+1].copyTo(transicion);
                        im_arbol.push_back(transicion);
                        transicion.release();
                    }
                }
                //Creacion modelo binario
                model.release();
                model = createLBPHFaceRecognizer();
                arbol.push_back(model);
                arbol[num_modelos]->train(im_arbol,labels);
                num_modelos++;
                im_arbol.clear();
                //Seleccion una imagen por clase
                if(id_CSV%2==0)
                {
                    for(int j=posiciones[2*i]+1;j<=posiciones[2*i+1];j++)
                    {
                        images[j].copyTo(transicion);
                        im_arbol.push_back(transicion);
                        transicion.release();
                    }
                }
                //Seleccion dos imagenes por clase
                else
                {
                    for(int j=posiciones[2*i]+1;j<=posiciones[2*i+1];j++)
                    {
                        images[2*j].copyTo(transicion);
                        im_arbol.push_back(transicion);
                        transicion.release();
                        images[2*j+1].copyTo(transicion);
                        im_arbol.push_back(transicion);
                        transicion.release();
                    }
                }
                //Creacion modelo binario
                model.release();
```

```
1191                model = createLBPHFaceRecognizer();
1192                arbol.push_back(model);
1193                arbol[num_modelos]->train(im_arbol,labels);
1194                num_modelos++;
1195                im_arbol.clear();
1196            }
1197            //Tramos para los siguientes modelos
1198            tam_posiciones=posiciones.size();
1199            for(int i=0;i<(tam_posiciones/2);i++)
1200            {
1201                posiciones.push_back(((posiciones[2*i]+1)/2)-1);
1202                posiciones.push_back(posiciones[tam_posiciones+2*i]+num_images);
1203            }
1204            std::sort(posiciones.begin(),posiciones.end());
1205            num saltos2++;
1206            num_images/=2;
1207        }
1208    }
1209
1210    //Creacion de las ramas de modelo con cinco clases identificadoras
1211    while(num_images%5==0&&(int)posiciones.size()!=im_size)
1212    {
1213        //Relacion una imagen por clase
1214        if(id_CSV%2==0)
1215        {
1216            labels.clear();
1217            for(int i=0;i<(num_images)/5;i++)
1218                labels.push_back(0);
1219            for(int i=0;i<(num_images)/5;i++)
1220                labels.push_back(1);
1221            for(int i=0;i<(num images)/5;i++)
1222                labels.push_back(2);
1223            for(int i=0;i<(num_images)/5;i++)
1224                labels.push_back(3);
1225            for(int i=0;i<(num_images)/5;i++)
1226                labels.push_back(4);
1227        }
1228        //Relacion dos imagenes por clase
1229        else
1230        {
1231            labels.clear();
1232            for(int i=0;i<(num_images)/5;i++)
1233            {
1234                labels.push_back(0);
1235                labels.push_back(0);
1236            }
1237            for(int i=0;i<(num_images)/5;i++)
1238            {
1239                labels.push_back(1);
1240                labels.push_back(1);
1241            }
1242            for(int i=0;i<(num images)/5;i++)
1243            {
1244                labels.push_back(2);
1245                labels.push_back(2);
1246            }
1247            for(int i=0;i<(num_images)/5;i++)
1248            {
1249                labels.push_back(3);
1250                labels.push back(3);
1251            }
1252            for(int i=0;i<(num_images)/5;i++)
1253            {
1254                labels.push_back(4);
1255                labels.push_back(4);
1256            }
1257        }
1258        //Creacion de los modelos de cinco clases despues de las ramas binarias
1259        if(num saltos5==0)
1260        {
1261            for(uint i=0;i<(posiciones.size()/2);i++)
1262            {
1263                //Seleccion una imagen por clase
1264                if(id_CSV%2==0)
1265                {
1266                    for(int j=(2*num_images*i);j<=posiciones[2*i];j++)
1267                    {
```

```
1268                        images[j].copyTo(transicion);
1269                        im_arbol.push_back(transicion);
1270                        transicion.release();
1271                    }
1272                }
1273                //Seleccion dos imagenes por clase
1274                else
1275                {
1276                    for(int j=(2*num_images*i);j<=posiciones[2*i];j++)
1277                    {
1278                        images[2*j].copyTo(transicion);
1279                        im_arbol.push_back(transicion);
1280                        transicion.release();
1281                        images[2*j+1].copyTo(transicion);
1282                        im_arbol.push_back(transicion);
1283                        transicion.release();
1284                    }
1285                }
1286                //Creacion del modelo de cinco clases identificadoras
1287                model.release();
1288                model = createLBPHFaceRecognizer();
1289                arbol.push_back(model);
1290                arbol[num_modelos]->train(im_arbol,labels);
1291                num_modelos++;
1292                im arbol.clear();
1293                //Seleccion una imagen por clase
1294                if(id_CSV%2==0)
1295                {
1296                    for(int j=posiciones[2*i]+1;j<=posiciones[2*i+1];j++)
1297                    {
1298                        images[j].copyTo(transicion);
1299                        im_arbol.push_back(transicion);
1300                        transicion.release();
1301                    }
1302                }
1303                //Seleccion dos imagenes por clase
1304                else
1305                {
1306                    for(int j=posiciones[2*i]+1;j<=posiciones[2*i+1];j++)
1307                    {
1308                        images[2*j].copyTo(transicion);
1309                        im arbol.push back(transicion);
1310                        transicion.release();
1311                        images[2*j+1].copyTo(transicion);
1312                        im_arbol.push_back(transicion);
1313                        transicion.release();
1314                    }
1315                }
1316                //Creacion del modelo de cinco clases identificadoras
1317                model.release();
1318                model = createLBPHFaceRecognizer();
1319                arbol.push_back(model);
1320                arbol[num_modelos]->train(im_arbol,labels);
1321                num modelos++;
1322                im_arbol.clear();
1323            }
1324            //Tramos para los siguientes modelos
1325            tam_posiciones=posiciones.size();
1326            for(int i=0;i<(tam posiciones-1);i++)
1327            {
1328                if(i==0)
1329                {
1330                    posiciones.push back(((posiciones[0]+1)/5)-1);
1331                    posiciones.push_back((2*(posiciones[0]+1)/5)-1);
1332                    posiciones.push_back((3*(posiciones[0]+1)/5)-1);
1333                    posiciones.push_back((4*(posiciones[0]+1)/5)-1);
1334                }
1335                posiciones.push_back(((posiciones[0]+1)/5)+posiciones[i]);
1336                posiciones.push_back((2*(posiciones[0]+1)/5)+posiciones[i]);
1337                posiciones.push_back((3*(posiciones[0]+1)/5)+posiciones[i]);
1338                posiciones.push_back((4*(posiciones[0]+1)/5)+posiciones[i]);
1339            }
1340            std::sort(posiciones.begin(),posiciones.end());
1341            num_saltos5++;
1342            num images/=5;
1343        }
1344        //Creacion de los modelos de cinco clases despues de una rama de un modelo con
cinco clases
1345        else
```

```
1346         {
1347             for(uint i=0;i<(posiciones.size()/5);i++)
1348             {
1349                 //Seleccion una imagen por clase
1350                 if(id_CSV%2==0)
1351                 {
1352                     for(int j=(5*num_images*i);j<=posiciones[5*i];j++)
1353                     {
1354                         images[j].copyTo(transicion);
1355                         im_arbol.push_back(transicion);
1356                         transicion.release();
1357                     }
1358                 }
1359                 //Seleccion dos imagenes por clase
1360                 else
1361                 {
1362                     for(int j=(5*num_images*i);j<=posiciones[5*i];j++)
1363                     {
1364                         images[2*j].copyTo(transicion);
1365                         im_arbol.push_back(transicion);
1366                         transicion.release();
1367                         images[2*j+1].copyTo(transicion);
1368                         im_arbol.push_back(transicion);
1369                         transicion.release();
1370                     }
1371                 }
1372                 //Creacion del modelo de cinco clases identificadoras
1373                 model.release();
1374                 model = createLBPHFaceRecognizer();
1375                 arbol.push_back(model);
1376                 arbol[num_modelos]->train(im_arbol,labels);
1377                 num_modelos++;
1378                 im_arbol.clear();
1379                 //Seleccion una imagen por clase
1380                 if(id_CSV%2==0)
1381                 {
1382                     for(int j=posiciones[5*i]+1;j<=posiciones[5*i+1];j++)
1383                     {
1384                         images[j].copyTo(transicion);
1385                         im_arbol.push_back(transicion);
1386                         transicion.release();
1387                     }
1388                 }
1389                 //Seleccion dos imagenes por clase
1390                 else
1391                 {
1392                     for(int j=posiciones[5*i]+1;j<=posiciones[5*i+1];j++)
1393                     {
1394                         images[2*j].copyTo(transicion);
1395                         im_arbol.push_back(transicion);
1396                         transicion.release();
1397                         images[2*j+1].copyTo(transicion);
1398                         im_arbol.push_back(transicion);
1399                         transicion.release();
1400                     }
1401                 }
1402                 //Creacion del modelo de cinco clases identificadoras
1403                 model.release();
1404                 model = createLBPHFaceRecognizer();
1405                 arbol.push back(model);
1406                 arbol[num_modelos]->train(im_arbol,labels);
1407                 num_modelos++;
1408                 im_arbol.clear();
1409                 //Seleccion una imagen por clase
1410                 if(id_CSV%2==0)
1411                 {
1412                     for(int j=posiciones[5*i+1]+1;j<=posiciones[5*i+2];j++)
1413                     {
1414                         images[j].copyTo(transicion);
1415                         im_arbol.push_back(transicion);
1416                         transicion.release();
1417                     }
1418                 }
1419                 //Seleccion dos imagenes por clase
1420                 else
1421                 {
1422                     for(int j=posiciones[5*i+1]+1;j<=posiciones[5*i+2];j++)
```

```
1423                    {
1424                        images[2*j].copyTo(transicion);
1425                        im_arbol.push_back(transicion);
1426                        transicion.release();
1427                        images[2*j+1].copyTo(transicion);
1428                        im_arbol.push_back(transicion);
1429                        transicion.release();
1430                    }
1431                }
1432                //Creacion del modelo de cinco clases identificadoras
1433                model.release();
1434                model = createLBPHFaceRecognizer();
1435                arbol.push back(model);
1436                arbol[num_modelos]->train(im_arbol,labels);
1437                num_modelos++;
1438                im_arbol.clear();
1439                //Seleccion una imagen por clase
1440                if(id_CSV%2==0)
1441                {
1442                    for(int j=posiciones[5*i+2]+1;j<=posiciones[5*i+3];j++)
1443                    {
1444                        images[j].copyTo(transicion);
1445                        im_arbol.push_back(transicion);
1446                        transicion.release();
1447                    }
1448                }
1449                //Seleccion dos imagenes por clase
1450                else
1451                {
1452                    for(int j=posiciones[5*i+2]+1;j<=posiciones[5*i+3];j++)
1453                    {
1454                        images[2*j].copyTo(transicion);
1455                        im_arbol.push_back(transicion);
1456                        transicion.release();
1457                        images[2*j+1].copyTo(transicion);
1458                        im_arbol.push_back(transicion);
1459                        transicion.release();
1460                    }
1461                }
1462                //Creacion del modelo de cinco clases identificadoras
1463                model.release();
1464                model = createLBPHFaceRecognizer();
1465                arbol.push_back(model);
1466                arbol[num_modelos]->train(im_arbol,labels);
1467                num_modelos++;
1468                im arbol.clear();
1469                //Seleccion una imagen por clase
1470                if(id_CSV%2==0)
1471                {
1472                    for(int j=posiciones[5*i+3]+1;j<=posiciones[5*i+4];j++)
1473                    {
1474                        images[j].copyTo(transicion);
1475                        im_arbol.push_back(transicion);
1476                        transicion.release();
1477                    }
1478                }
1479                //Seleccion dos imagenes por clase
1480                else
1481                {
1482                    for(int j=posiciones[5*i+3]+1;j<=posiciones[5*i+4];j++)
1483                    {
1484                        images[2*j].copyTo(transicion);
1485                        im arbol.push back(transicion);
1486                        transicion.release();
1487                        images[2*j+1].copyTo(transicion);
1488                        im_arbol.push_back(transicion);
1489                        transicion.release();
1490                    }
1491                }
1492                //Creacion del modelo de cinco clases identificadoras
1493                model.release();
1494                model = createLBPHFaceRecognizer();
1495                arbol.push_back(model);
1496                arbol[num_modelos]->train(im_arbol,labels);
1497                num modelos++;
1498                im_arbol.clear();
1499            }
1500            //Tramos para los siguientes modelos
1501            tam_posiciones=posiciones.size();
```

```
1502             for(int i=0;i<(tam_posiciones-1);i++)
1503             {
1504                 if(i==0)
1505                 {
1506                     posiciones.push_back(((posiciones[0]+1)/5)-1);
1507                     posiciones.push_back((2*(posiciones[0]+1)/5)-1);
1508                     posiciones.push_back((3*(posiciones[0]+1)/5)-1);
1509                     posiciones.push_back((4*(posiciones[0]+1)/5)-1);
1510                 }
1511                 posiciones.push back(((posiciones[0]+1)/5)+posiciones[i]);
1512                 posiciones.push_back((2*(posiciones[0]+1)/5)+posiciones[i]);
1513                 posiciones.push_back((3*(posiciones[0]+1)/5)+posiciones[i]);
1514                 posiciones.push_back((4*(posiciones[0]+1)/5)+posiciones[i]);
1515             }
1516             std::sort(posiciones.begin(),posiciones.end());
1517             num_saltos5++;
1518             num_images/=5;
1519         }
1520     }
1521
1522     //Modelo arbol creado completo. Se procede a guardarlo
1523     for(uint i=0;i<arbol.size();i++)
1524     {
1525         char intermedio[100]={0};
1526         sprintf(intermedio,fn_model.toUtf8().constData(),id_CSV,i);
1527         arbol[i]->save(intermedio);
1528     }
1529     arbol.clear();
1530     char buf[40]={0};
1531     sprintf(  buf, "%d\n%d\n%d", num_modelos, num_saltos2, num_saltos5);
1532
ficheroArbol.open(QString("etc/Resultados/50x50/ParamArbolLBPHFace%1.txt").arg(id_CSV).toUtf8
().constData());
1533     ficheroArbol<<buf;
1534     ficheroArbol.close();
1535     posiciones.clear();
1536
1537     //Emision de la señal de fin de creacion del modelo
1538     emit SignalFinCreacionModelo();
1539 }
```

void QProceso::SlotCreaModeloEigenFace (QString *fn_model*) [slot]

Funcion SlotCreaModeloEigenFace. Crea el modelo EigenFace.

Crea el modelo EigenFace. Entrena dicho modelo con las imagenes y clases del identificador del fichero CSV, cargado anteriormente. Por ultimo, guarda el modelo creado en un fichero e indica la finalizacion de la creacion del modelo.

Parámetros:

fn_model	Nombre con el cual se guardara el modelo.

```
72 {
73     //Creacion del modelo
74     model = createEigenFaceRecognizer();
75     model->train(images,labels);
76
77     //Se guarda el modelo creado
78     model->save(fn_model.toUtf8().constData());
79
80     //Emision de la señal de fin de creacion del modelo
81     emit SignalFinCreacionModelo();
82     model->~FaceRecognizer();
83 }
```

void QProceso::SlotCreaModeloFisherFace (QString *fn_model*) [slot]

Funcion SlotCreaModeloFisherFace. Crea el modelo FisherFace.

Crea el modelo FisherFace. Entrena dicho modelo con las imagenes y clases del identificador del fichero CSV, cargado anteriormente. Por ultimo, guarda el modelo creado en un fichero e indica la finalizacion de la creacion del modelo.

Parámetros:

fn_model	Nombre con el cual se guardara el modelo.

```
58 {
59     //Creacion del modelo
60     model = createFisherFaceRecognizer();
61     model->train(images,labels);
62 
63     //Se guarda el modelo creado
64     model->save(fn_model.toUtf8().constData());
65 
66     //Emision de la señal de fin de creacion del modelo
67     emit SignalFinCreacionModelo();
68     model->~FaceRecognizer();
69 }
```

void QProceso::SlotCreaModeloLBPHFace (QString *fn_model*) [slot]

Funcion SlotCreaModeloLBPHFace. Crea el modelo LBPHFace.

Crea el modelo LBPHFace. Entrena dicho modelo con las imagenes y clases del identificador del fichero CSV, cargado anteriormente. Por ultimo, guarda el modelo creado en un fichero e indica la finalizacion de la creacion del modelo.

Parámetros:

fn_model	Nombre con el cual se guardara el modelo.

```
86 {
87     //Creacion del modelo
88     model = createLBPHFaceRecognizer();
89 
90     //Se guarda el modelo creado
91     model->train(images,labels);
92     model->save(fn_model.toUtf8().constData());
93 
94     //Emision de la señal de fin de creacion del modelo
95     emit SignalFinCreacionModelo();
96     model->~FaceRecognizer();
97 }
```

void QProceso::SlotIdentificar (Mat * *test_im*) [slot]

Funcion SlotIdentificar. Identifica a un sujeto.

Identifica un sujeto, obteniendo la confianza de la prediccion, asi como la prediccion de la clase identificadora a la que pertenece dicho sujeto. Indica la finalizacion de la identificacion del sujeto.

Parámetros:

test_im	Contiene la imagen del sujeto a identificar.

```
130 {
131     //Identificacion del sujeto
132     predictedLabel=-1;
133     confidence=0.0;
134     model->predict(*test_im,predictedLabel,confidence);
135 
136     //Emision de la señal de fin de identificacion del sujeto
137     emit SignalFinIdentificacion(confidence,predictedLabel);
138 }
```

void QProceso::SlotIdentificarArbol (Mat * *test_im*) [slot]

Funcion SlotIdentificar. Identifica a un sujeto con un modelo con estructura en arbol.

Identifica un sujeto. Primero, comprueba si es hombre o mujer. Despues, continua identificandopor las demas rama del arbol, hasta que llega a la base del arbol. Se obtiene la confianza de la prediccion, asi como la prediccion de la clase identificadora a la que pertenece dicho sujeto. Al final, se indica la finalizacion de la identificacion del sujeto.

Parámetros:

test_im	Contiene la imagen del sujeto a identificar.

```
1628 {
1629     num_images=im_size;
```

```
1630     predictedLabelArbol=0;
1631     prox_model=0;
1632     num_modelosBinarios=0;
1633     fin=false;
1634 
1635     //Identificacion del sujeto por las ramas binarias
1636     for(int i=0;(i<num_saltos2)&&(!fin);i++)
1637     {
1638         confidence=0.0;
1639         predictedLabel=-1;
1640         arbol[prox_model]->predict(*test_im,predictedLabel,confidence);
1641         //Si no identifica al sujeto, se termina la ejecucion del arbol
1642         if(predictedLabel==-1)
1643         {
1644             fin=true;
1645             if(i==0)
1646                 predictedLabelArbol=-1;
1647             else if(predictedLabelArbol<im_size/2)
1648                 predictedLabelArbol=-2;
1649             else
1650                 predictedLabelArbol=-3;
1651         }
1652         //Si el sujeto se identifica, pasa a la siguiente rama del arbol binario
1653         else
1654         {
1655             predictedLabelArbol+=predictedLabel*num_images/2;
1656             num_images/=2;
1657             prox_model=2*prox_model+predictedLabel+1;
1658             num_modelosBinarios+=(i+1)*2;
1659         }
1660     }
1661 
1662     //Identificacion del sujeto por las ramas de los modelos de cinco clases
identificadoras
1663     for(int i=0;(i<num_saltos5)&&(!fin);i++)
1664     {
1665         confidence=0.0;
1666         predictedLabel=-1;
1667         arbol[prox_model]->predict(*test_im,predictedLabel,confidence);
1668         //Si no identifica al sujeto, se termina la ejecucion del arbol
1669         if(predictedLabel==-1)
1670         {
1671             fin=true;
1672             if(predictedLabelArbol<im_size/2)
1673                 predictedLabelArbol=-2;
1674             else
1675                 predictedLabelArbol=-3;
1676         }
1677         //Si el sujeto se identifica, pasa a la siguiente rama de los modelos con cinco
clases identificadoras
1678         else
1679         {
1680             predictedLabelArbol+=predictedLabel*num_images/5;
1681             num_images/=5;
1682             prox_model=num_modelosBinarios+(prox_model-
(num_modelosBinarios/2))*5+predictedLabel+1;
1683         }
1684     }
1685 
1686     //Emision de la señal de fin de identificacion del sujeto
1687     emit SignalFinIdentificacion(confidence,predictedLabelArbol);
1688 }
```

Documentación de los datos miembro

vector<Ptr<FaceRecognizer> > QProceso::arbol [private]

Utilizado para la crecacion y carga de los distintos modelos con estructura en arbol

double QProceso::confidence [private]

Confianza de la identificacion realizada

ofstream QProceso::ficheroArbol [private]

Fichero de escritura utilizado en la estructura en arbol para guardar el modelo

ifstream QProceso::ficheroArbolLec [private]

Fichero de lectura utilizado en la estructura en arbol para cargar el modelo

bool QProceso::fin [private]

Indica el final en la identificacion del modelo con estructura en arbol

vector<Mat> QProceso::im_arbol [private]

Imagenes usadas para la creacion de los modelos con estructura en arbol

int QProceso::im_size [private]

Numero de imagenes que forman el vector images

vector<Mat> QProceso::images [private]

Imagenes usadas para la creacion de los modelos

vector<int> QProceso::labels [private]

Contiene la correspondencia imagen - clase a identificar para la creacion de los modelos

Ptr<FaceRecognizer> QProceso::model [private]

Utilizado para la creacion y carga de los distintos modelos de identificacion

int QProceso::num_images [private]

Numero de imagenes que forman el vector images. Se modifica en la estructura en arbol

int QProceso::num_modelos [private]

Numero de modelos generados para la estructura en arbol

int QProceso::num_modelosBinarios [private]

Numero total de modelos binarios en la estructura en arbol

int QProceso::num_saltos2 [private]

Numero de saltos de modelos binarios en la estructura en arbol

int QProceso::num_saltos5 [private]

Numero de saltos de modelos con cinco clases en la estructura en arbol

vector<int> QProceso::posiciones [private]

Utilizado para la definicion de las ramas de los modelos con estructura en arbol

int QProceso::predictedLabel [private]

Prediccion realizada por un modelo de identificacion

int QProceso::predictedLabelArbol [private]

Prediccion realizada por un modelo con estructura en arbol

int QProceso::prox_model [private]

Utilizado para conocer el siguiente modelo a identificar en la estructura en arbol

int QProceso::tam_posiciones [private]

Tamaño del vector posiciones

Mat QProceso::transicion `[private]`

Utilizada para la seleccion de imagenes de cada rama del modelo con estructura en arbol

---

La documentación para esta clase fue generada a partir de los siguientes ficheros:

- src/**Proceso.h**
- src/**Proceso.cpp**

Documentación de archivos

Referencia del Archivo src/Cabeceras.h

Contiene la inclusion de las clases y los ficheros *.h necesarios. Se debe incluir en todos los ficheros del proyecto.

```
#include "Tipos.h"
#include "FormPrincipal.h"
#include "Proceso.h"
#include "Config.h"
#include <QtGui>
#include "opencv2/opencv.hpp"
#include "opencv2/core/core.hpp"
#include "opencv2/highgui/highgui.hpp"
#include <sys/time.h>
#include <unistd.h>
#include <iostream>
#include <vector>
```

Dependencia gráfica adjunta para Cabeceras.h:

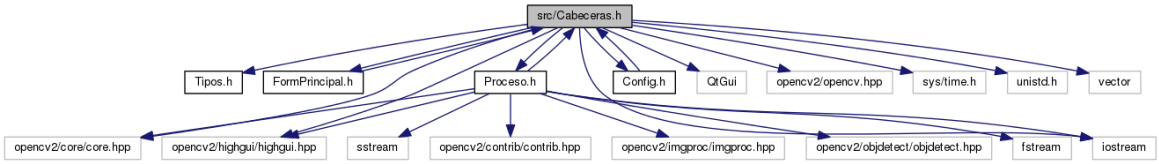
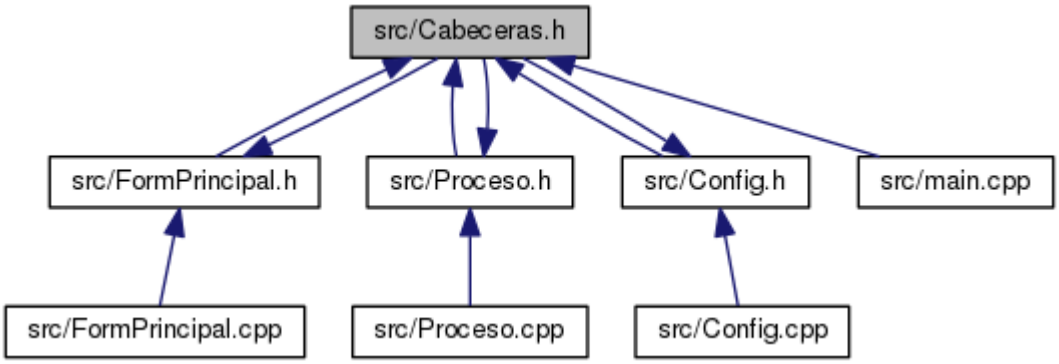


Gráfico de los archivos que directa o indirectamente incluyen a este archivo:



Descripción detallada

Contiene la inclusion de las clases y los ficheros *.h necesarios. Se debe incluir en todos los ficheros del proyecto.

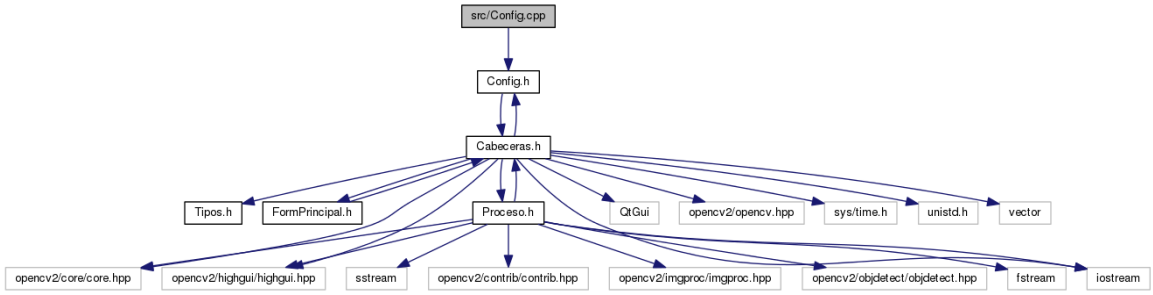
Autor:
Maria Sierra Zapata

Versión:
3.0

Fecha:
29 Junio 2015

Referencia del Archivo src/Config.cpp

Contiene la definicion de la clase **QConfig**.
`#include "Config.h"`
Dependencia gráfica adjunta para Config.cpp:



Descripción detallada

Contiene la definicion de la clase **QConfig**.

Autor:
Maria Sierra Zapata

Versión:
3.0

Fecha:
29 Junio 2015

Referencia del Archivo src/Config.h

Contiene la cabecera de la clase **QConfig**.

```
#include "Cabeceras.h"
```

Dependencia gráfica adjunta para Config.h:

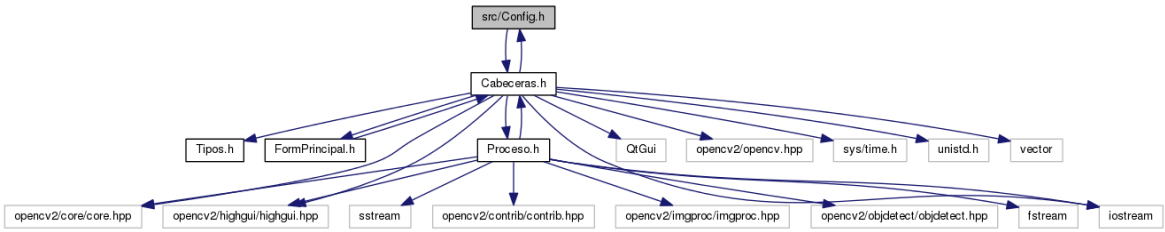
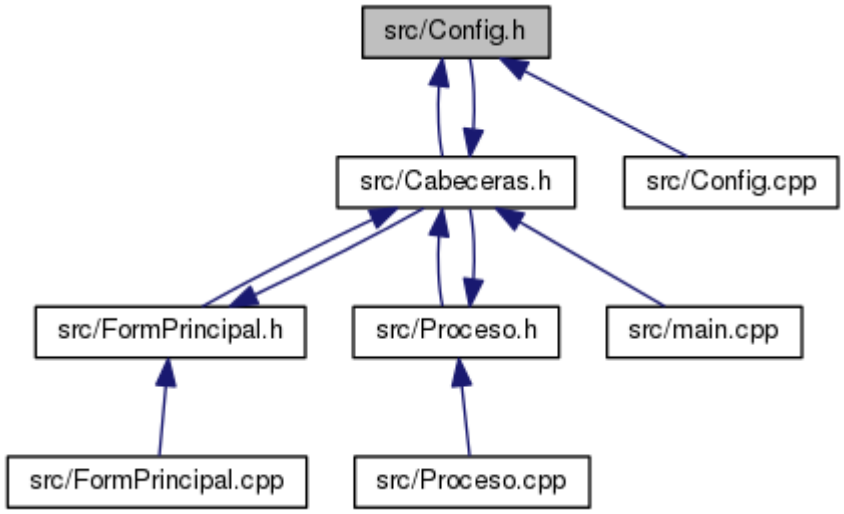


Gráfico de los archivos que directa o indirectamente incluyen a este archivo:



Clases

- class **QConfig**

Se encarga de la lectura del fichero de configuracion para la interfaz de la aplicacion y la escritura de los ficheros con los datos generados por los distintos modelos de identificacion. Variables

- **QConfig config**

Descripción detallada

Contiene la cabecera de la clase **QConfig**.

Autor:
Maria Sierra Zapata

Versión:
3.0

Fecha:
29 Junio 2015

Documentación de las variables

QConfig config

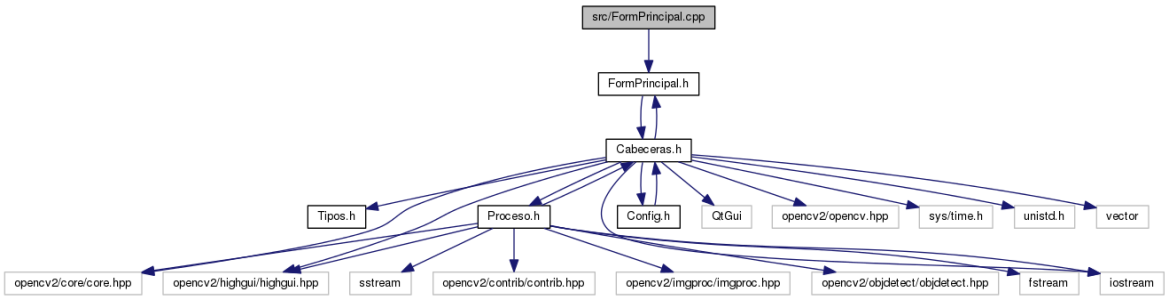
Instancia de la clase **QConfig**

Referencia del Archivo src/FormPrincipal.cpp

Contiene la definicion de la clase **QFormPrincipal**.

```
#include "FormPrincipal.h"
```

Dependencia gráfica adjunta para FormPrincipal.cpp:



Descripción detallada

Contiene la definicion de la clase **QFormPrincipal**.

Autor:

Maria Sierra Zapata

Versión:

3.0

Fecha:

29 Junio 2015

Referencia del Archivo src/FormPrincipal.h

Contiene la cabecera de la clase **QFormPrincipal**.
`#include "Cabeceras.h"`
Dependencia gráfica adjunta para FormPrincipal.h:

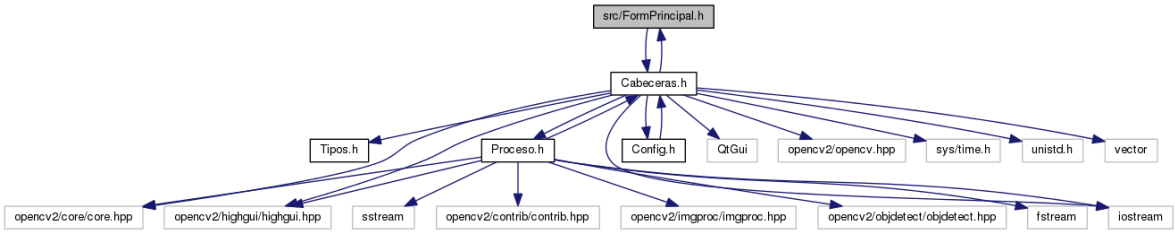
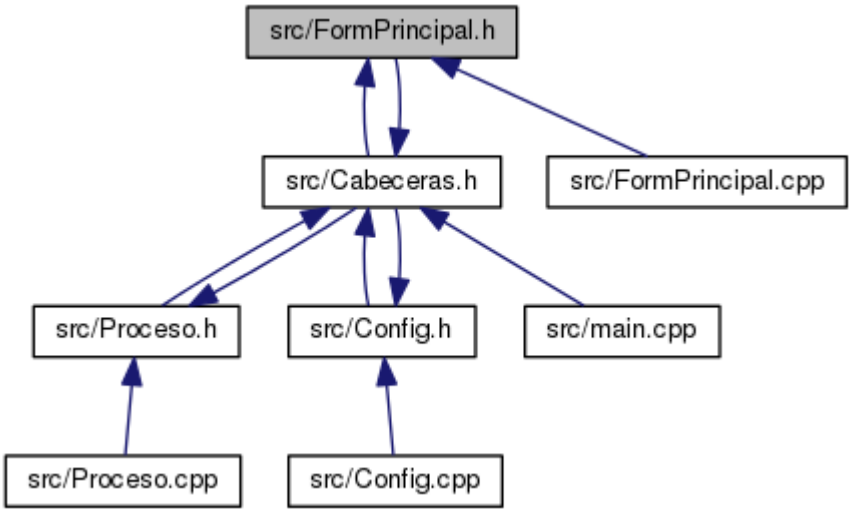


Gráfico de los archivos que directa o indirectamente incluyen a este archivo:



Clases

class **QFormPrincipal**

Hereda de QWidget. Se encarga de gestionar la interfaz del programa. Variables

QConfig config

Descripción detallada

Contiene la cabecera de la clase **QFormPrincipal**.

Autor:
Maria Sierra Zapata

Versión:
3.0

Fecha:
29 Junio 2015

Documentación de las variables

QConfig config

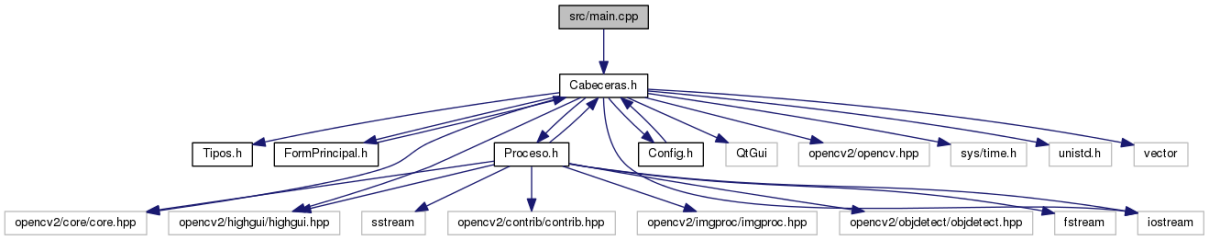
Instancia de la clase **QConfig**

Referencia del Archivo src/main.cpp

Contiene funcion main. Inicializa la aplicacion.

```
#include "Cabeceras.h"
```

Dependencia gráfica adjunta para main.cpp:



Funciones

- int **main** (int argc, char *argv[])
  Funcion main. Invoca a la aplicacion.

Variables

- **QConfig config**

Descripción detallada

Contiene funcion main. Inicializa la aplicacion.

Autor:
Maria Sierra Zapata

Versión:
3.0

Fecha:
29 Junio 2015

Versiones

1.0 - Realizacion de la interfaz utilizando Qt.

1.1 - Incorporacion de un modelo de identificacion. Sincronizacion entre los hilos de las clases **QProceso** y **QFormPrincipal**. Base de datos de diez sujetos.

1.2 - Primera prueba de escritura de ficheros generados.

2.0 - Incorporacion de los tres modelos existentes en OpenCV.

2.1 - Base de datos de cien sujetos, cincuenta sujetos femeninos y cincuenta sujetos masculinos.

2.2 - Cambios en la escritura de ficheros con los datos generados. Separacion por identificador del modelo usado, por identificador del fichero CSV utilizado y por genero.

3.0 - Incorporacion de una estructura en arbol por cada modelo existente en OpenCV.

Documentación de las funciones

int main (int *argc*, char * *argv*[])

Funcion main. Invoca a la aplicacion.

Se encarga de cargar la configuracion inicial de la interfaz, asi como del arranque de la aplicacion.

Parámetros:

argc	Numero de parametros que se pasan por linea de comandos.
argv	Parametros pasados por linea de comandos.

Devuelve:

a.exec() Devuelve un valor generado por la aplicacion, dependiendo de como termine la misma.

```
56 {
57     // Carga del fichero de configuracion para la interfaz de la aplicacion.
58     config.Inicializar();
59 
60     // Arranque de la aplicacion
61     QApplication a(argc, argv);
62     a.setApplicationName("EstudioTFG");
63 
64     // Arranque del gestor de la interfaz
65     QFormPrincipal w;
66     w.showFullScreen();
67     w.Comenzar();
68 
69     return a.exec();
70 }
```

Documentación de las variables

QConfig config

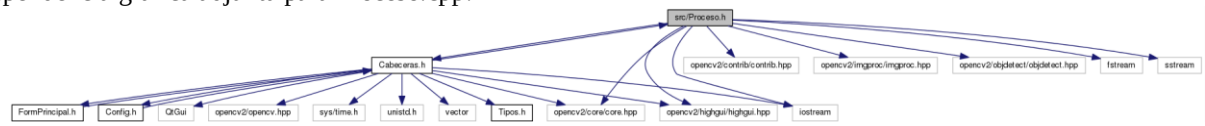
Instancia de la clase **QConfig**

Referencia del Archivo src/Proceso.cpp

Contiene la definicion de la clase **QProceso**.

```
#include "Proceso.h"
```

Dependencia gráfica adjunta para Proceso.cpp:



Descripción detallada

Contiene la definicion de la clase **QProceso**.

Autor:

Maria Sierra Zapata

Versión:

3.0

Fecha:

29 Junio 2015

Referencia del Archivo src/Proceso.h

Contiene la cabecera de la clase **QProceso**.

```
#include "Cabeceras.h"
#include "opencv2/core/core.hpp"
#include "opencv2/contrib/contrib.hpp"
#include "opencv2/highgui/highgui.hpp"
#include "opencv2/imgproc/imgproc.hpp"
#include "opencv2/objdetect/objdetect.hpp"
#include <iostream>
#include <fstream>
#include <sstream>
```

Dependencia gráfica adjunta para Proceso.h:

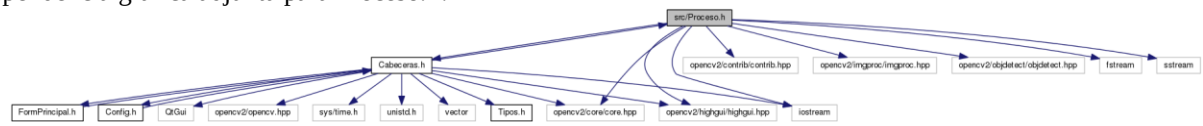
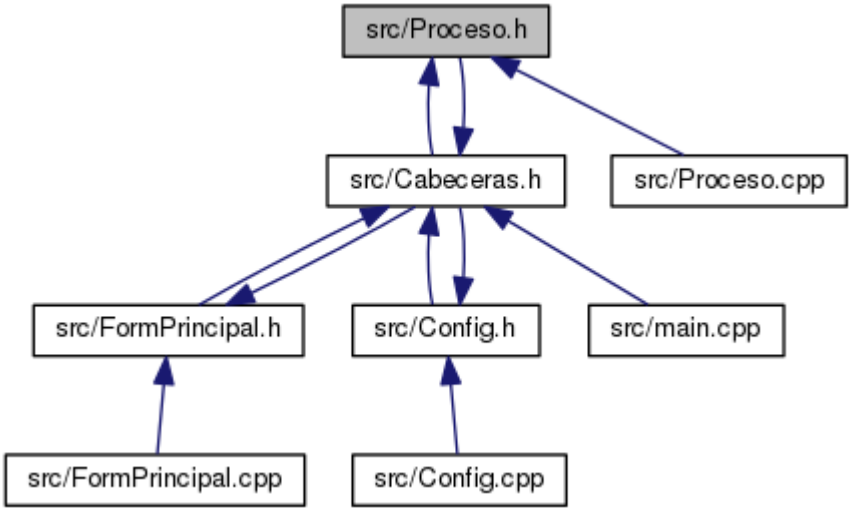


Gráfico de los archivos que directa o indirectamente incluyen a este archivo:



Clases

- class **QProceso**

Hereda de QObject. Se encarga de la creacion, carga e identificacion llevada a cabo por los modelos.

Descripción detallada

Contiene la cabecera de la clase **QProceso**.

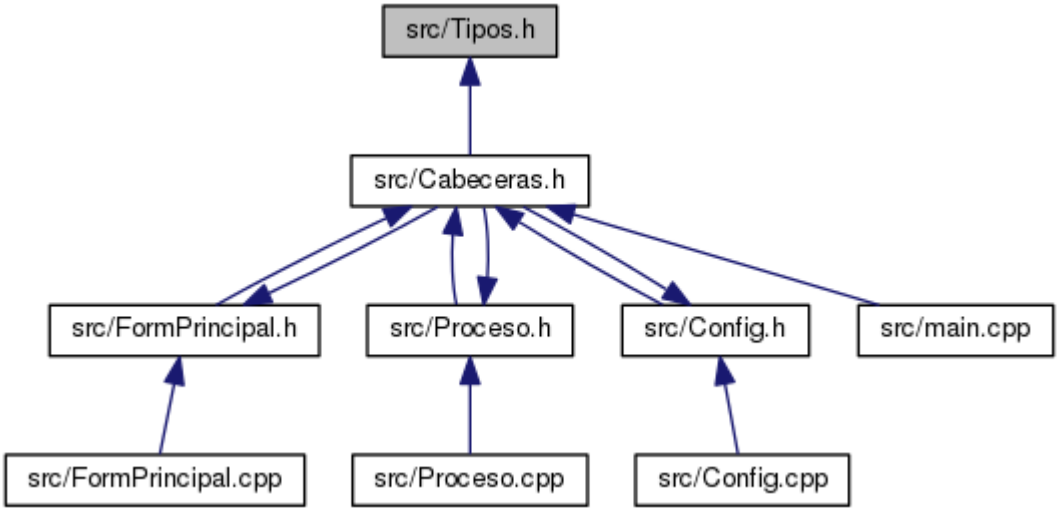
Autor:
Maria Sierra Zapata

Versión:
3.0

Fecha:
29 Junio 2015

Referencia del Archivo src/Tipos.h

Contiene la definicion de tipos de enteros y decimales especiales.
Gráfico de los archivos que directa o indirectamente incluyen a este archivo:



'typedefs'

- typedef signed char **int8**
- typedef unsigned char **uInt8**
- typedef signed short **int16**
- typedef unsigned short **uInt16**
- typedef signed long **int32**
- typedef unsigned long **uInt32**
- typedef float **float32**
- typedef double **float64**
- typedef unsigned long long **uInt64**

Descripción detallada

Contiene la definicion de tipos de enteros y decimales especiales.

Autor:
Maria Sierra Zapata

Versión:
3.0

Fecha:
29 Junio 2015

Documentación de los 'typedefs'

typedef float float32

typedef double float64

typedef signed short int16

typedef signed long int32

typedef signed char int8

typedef unsigned short uInt16

typedef unsigned long uInt32

typedef unsigned long long uInt64

typedef unsigned char uInt8

ANEXO B. GENERADOR DE FICHEROS MAKEFILE AUTOMÁTICO

```
TEMPLATE = app
VERSION = 3.0
QT += core \
    gui
CONFIG += qt \
    debug_and_release \
    build_all
HEADERS += src/Tipos.h \
    src/Config.h \
    src/Proceso.h \
    src/Cabeceras.h \
    src/FormPrincipal.h
SOURCES +=  src/Config.cpp \
            src/Proceso.cpp \
            src/FormPrincipal.cpp \
            src/main.cpp
LIBS += -L/usr/lib/ \
        -L/usr/local/lib/ \
        -lopencv_core \
        -lopencv_highgui \
        -lopencv_imgproc \
        -lopencv_video \
        -lopencv_objdetect \
        -lopencv_contrib \
        -lopencv_legacy

unix:DEFINES = _TTY_POSIX_
win32:DEFINES = _TTY_WIN_ \
    QWT_DLL \
    QT_DLL

CONFIG(debug, debug|release) {
    OBJECTS_DIR = .build/debug/obj
    MOC_DIR = .build/debug/moc
    DESTDIR = .build/debug/
    TARGET = EstudioTFGd
}
else {
    OBJECTS_DIR = .build/release/obj
    MOC_DIR = .build/release/moc
    DESTDIR = .build/release/
    TARGET = EstudioTFG

    target.path = EstudioTFG/
    config.path = EstudioTFG/etc
    config.files = etc/*
    resources.path = EstudioTFG/resources
    resources.files = resources/*
    INSTALLS += target config resources
}
```

ANEXO C. FICHEROS MATLAB PARA LA GENERACION DE LAS GRAFICAS

Fichero analizaFicheroTiempos.m:

```
%
% Fichero: analizaFicheroTiempos.m
% Autor: Maria Sierra Zapata
% Fecha: 30/06/2015
% Version: 0.1
%
% Breve Descripcion:
%       Apertura de los ficheros generados debido a la creacion y carga de
%       los diferentes modelos estudiados. Generacion de las graficas
%       correspondientes a los tiempos de los modelos.
%
% Datos que se obtienen:
%       Tiempo de creacion de todos los modelos
%       Tiempo de carga de todos los modelos
%
% Graficas generadas:
%       Tiempo de creacion - Numero de clases
%       Tiempo de carga - Numero de clases
%

cadena='%d=%f %f';
num_images=[10 50 100];

% Apertura y lectura del fichero
nom_fichero='50x50/TiemposModelos_0.m';
tam50x50_0=leeFicheroTiempos(nom_fichero,cadena);
nom_fichero='50x50/TiemposModelos_1.m';
tam50x50_1=leeFicheroTiempos(nom_fichero,cadena);
nom_fichero='50x50/TiemposModelos_2.m';
tam50x50_2=leeFicheroTiempos(nom_fichero,cadena);
nom_fichero='50x50/TiemposModelos_3.m';
tam50x50_3=leeFicheroTiempos(nom_fichero,cadena);
nom_fichero='50x50/TiemposModelos_4.m';
tam50x50_4=leeFicheroTiempos(nom_fichero,cadena);
nom_fichero='50x50/TiemposModelos_5.m';
tam50x50_5=leeFicheroTiempos(nom_fichero,cadena);

nom_fichero='100x100/TiemposModelos_0.m';
tam100x100_0=leeFicheroTiempos(nom_fichero,cadena);
nom_fichero='100x100/TiemposModelos_1.m';
tam100x100_1=leeFicheroTiempos(nom_fichero,cadena);
nom_fichero='100x100/TiemposModelos_2.m';
tam100x100_2=leeFicheroTiempos(nom_fichero,cadena);
nom_fichero='100x100/TiemposModelos_3.m';
tam100x100_3=leeFicheroTiempos(nom_fichero,cadena);
nom_fichero='100x100/TiemposModelos_4.m';
tam100x100_4=leeFicheroTiempos(nom_fichero,cadena);
nom_fichero='100x100/TiemposModelos_5.m';
tam100x100_5=leeFicheroTiempos(nom_fichero,cadena);

nom_fichero='150x150/TiemposModelos_0.m';
tam150x150_0=leeFicheroTiempos(nom_fichero,cadena);
nom_fichero='150x150/TiemposModelos_1.m';
tam150x150_1=leeFicheroTiempos(nom_fichero,cadena);
nom_fichero='150x150/TiemposModelos_2.m';
tam150x150_2=leeFicheroTiempos(nom_fichero,cadena);
nom_fichero='150x150/TiemposModelos_3.m';
tam150x150_3=leeFicheroTiempos(nom_fichero,cadena);
nom_fichero='150x150/TiemposModelos_4.m';
tam150x150_4=leeFicheroTiempos(nom_fichero,cadena);
nom_fichero='150x150/TiemposModelos_5.m';
tam150x150_5=leeFicheroTiempos(nom_fichero,cadena);

nom_fichero='200x200/TiemposModelos_0.m';
tam200x200_0=leeFicheroTiempos(nom_fichero,cadena);
nom_fichero='200x200/TiemposModelos_1.m';
tam200x200_1=leeFicheroTiempos(nom_fichero,cadena);
nom_fichero='200x200/TiemposModelos_2.m';
tam200x200_2=leeFicheroTiempos(nom_fichero,cadena);
nom_fichero='200x200/TiemposModelos_3.m';
tam200x200_3=leeFicheroTiempos(nom_fichero,cadena);
nom_fichero='200x200/TiemposModelos_4.m';
tam200x200_4=leeFicheroTiempos(nom_fichero,cadena);
nom_fichero='200x200/TiemposModelos_5.m';
tam200x200_5=leeFicheroTiempos(nom_fichero,cadena);
```

```
% Creacion de la grafica para un modelo especifico. Modificando y, se
% selecciona la grafica de tiempo de carga o de tiempo de creacion.
% Modificando x se selecciona el modelo a representar. Representacion en
% funcion del numero de clases.
x=3;
y=3;

tiempoSimUna=[tam50x50_0(x,y) tam50x50_2(x,y) tam50x50_4(x,y)];
tiempoArbUna=[tam50x50_0(x+3,y) tam50x50_2(x+3,y) tam50x50_4(x+3,y)];
tiempoSimDos=[tam50x50_1(x,y) tam50x50_3(x,y) tam50x50_5(x,y)];
tiempoArbDos=[tam50x50_1(x+3,y) tam50x50_3(x+3,y) tam50x50_5(x+3,y)];

subplot(2,2,1)
plot(num_images, tiempoSimUna,'.-', num_images, tiempoArbUna, '.-', num_images, tiempoSimDos, '.-
', num_images, tiempoArbDos, '.-')
xlabel('Num Clases')
ylabel('Segundos')
legend('S - 1:1','A - 1:1', 'S - 2:1', 'A - 2:1', 'Location', 'Best')
title('Tam Imagen 50x50')

tiempoSimUna=[tam100x100_0(x,y) tam100x100_2(x,y) tam100x100_4(x,y)];
tiempoArbUna=[tam100x100_0(x+3,y) tam100x100_2(x+3,y) tam100x100_4(x+3,y)];
tiempoSimDos=[tam100x100_1(x,y) tam100x100_3(x,y) tam100x100_5(x,y)];
tiempoArbDos=[tam100x100_1(x+3,y) tam100x100_3(x+3,y) tam100x100_5(x+3,y)];

subplot(2,2,2)
plot(num_images, tiempoSimUna,'.-', num_images, tiempoArbUna, '.-', num_images, tiempoSimDos, '.-
', num_images, tiempoArbDos, '.-')
xlabel('Num Clases')
ylabel('Segundos')
legend('S - 1:1','A - 1:1', 'S - 2:1', 'A - 2:1', 'Location', 'Best')
title('Tam Imagen 100x100')

tiempoSimUna=[tam150x150_0(x,y) tam150x150_2(x,y) tam150x150_4(x,y)];
tiempoArbUna=[tam150x150_0(x+3,y) tam150x150_2(x+3,y) tam150x150_4(x+3,y)];
tiempoSimDos=[tam150x150_1(x,y) tam150x150_3(x,y) tam150x150_5(x,y)];
tiempoArbDos=[tam150x150_1(x+3,y) tam150x150_3(x+3,y) tam150x150_5(x+3,y)];

subplot(2,2,3)
plot(num_images, tiempoSimUna,'.-', num_images, tiempoArbUna, '.-', num_images, tiempoSimDos, '.-
', num_images, tiempoArbDos, '.-')
xlabel('Num Clases')
ylabel('Segundos')
legend('S - 1:1','A - 1:1', 'S - 2:1', 'A - 2:1', 'Location', 'Best')
title('Tam Imagen 150x150')

tiempoSimUna=[tam200x200_0(x,y) tam200x200_2(x,y) tam100x100_4(x,y)];
tiempoArbUna=[tam200x200_0(x+3,y) tam200x200_2(x+3,y) tam100x100_4(x+3,y)];
tiempoSimDos=[tam200x200_1(x,y) tam200x200_3(x,y) tam100x100_5(x,y)];
tiempoArbDos=[tam200x200_1(x+3,y) tam200x200_3(x+3,y) tam100x100_5(x+3,y)];

subplot(2,2,4)
plot(num_images, tiempoSimUna,'.-', num_images, tiempoArbUna, '.-', num_images, tiempoSimDos, '.-
', num_images, tiempoArbDos, '.-')
xlabel('Num Clases')
ylabel('Segundos')
legend('S - 1:1','A - 1:1', 'S - 2:1', 'A - 2:1', 'Location', 'Best')
title('Tam Imagen 200x200')
```

Fichero de la función leeFicheroTiempos.m:

```
%
% Fichero: leeFicheroTiempos.m
% Autor: Maria Sierra Zapata
% Fecha: 30/06/2015
% Version: 0.1
%
% Breve Descripcion:
%       Funcion que se encarga de leer los datos de un fichero con un
%       formato especifico.
%
% Parametros de Entrada:
%       nom_fichero :  Contiene la cadena del fichero del que se quiere
%                      leer los datos.
```

```
%       cadena :  Contiene el formato con el que se desea leer el fichero.
%
% Parametros de Salida:
%       Y :  Matriz que contiene los datos obtenidos del fichero.
%

function Y = leeFicheroTiempos(nom_fichero, cadena)

% Apertura del fichero
fichero=fopen(nom_fichero,'r');
fgets(fichero);
% Lectura del fichero
X=fscanf(fichero,cadena);
Y=reshape(X,3,size(X,1)/3)';
% Cierre del fichero
fclose(fichero);

end
```

Fichero analizaFicheroPersona50x50.m:

```
%
% Fichero: analizaFicheroPersona50x50.m
% Autor: Maria Sierra Zapata
% Fecha: 01/07/2015
% Version: 0.1
%
% Breve Descripcion:
%       Apertura de los ficheros generados debido a la
%       identificacion de personas para un modelo y con tamaño de imagen
%       50x50 pixeles. Seleccion de los datos para la generacion de las
%       graficas para el estudio.
%
% Datos que se obtienen:
%       Tiempo Medio de Identificacion
%       Falsos Positivos Mujer
%       Falsos Positivos Hombre
%       Falsos Positivos
%       Tasa de Acierto Hombre
%       Tasa de Acierto Mujer
%       Tasa de Acierto
%

cadena='%d=%f %f %d';
num_images=[10 50 100];

% Apertura y lectura del fichero
nom_fichero='50x50/ResultadosHombreM2_02.m';
dat=leeFicheroPersona(nom_fichero,cadena);
% Obtencion de los datos del fichero
[m,n]=size(dat);
for i=1:m
    if i==1
        if dat(i,2)==0
            tmi50x50S1=0.0001;
        else
            tmi50x50S1=dat(i,2);
        end
        if dat(i,4)>5
            fpm50x50S1=0;
            tah50x50S1=1;
        else
            fpm50x50S1=1;
            tah50x50S1=0;
        end
        if dat(i,4)==i+5
            fp50x50S1=0;
            ta50x50S1=1;
        else
            fp50x50S1=1;
            ta50x50S1=0;
        end
    else
        if dat(i,2)==0
            tmi50x50S1=[tmi50x50S1 0.0001];
        else
```

```
            tmi50x50S1=[tmi50x50S1 dat(i,2)];
        end
        if dat(i,4)>5
            fpm50x50S1=[fpm50x50S1 0];
            tah50x50S1=[tah50x50S1 1];
        else
            fpm50x50S1=[fpm50x50S1 1];
            tah50x50S1=[tah50x50S1 0];
        end
        if dat(i,4)==i+5
            fp50x50S1=[fp50x50S1 0];
            ta50x50S1=[ta50x50S1 1];
        else
            fp50x50S1=[fp50x50S1 1];
            ta50x50S1=[ta50x50S1 0];
        end
    end
end
nom_fichero='50x50/ResultadosMujerM2_02.m';
dat=leeFicheroPersona(nom_fichero,cadena);
[m,n]=size(dat);
for i=1:m
    if dat(i,2)==0
        tmi50x50S1=[tmi50x50S1 0.0001];
    else
        tmi50x50S1=[tmi50x50S1 dat(i,2)];
    end
    if dat(i,4)==i&&dat(i,4)<=5
        fp50x50S1=[fp50x50S1 0];
        ta50x50S1=[ta50x50S1 1];
    else
        fp50x50S1=[fp50x50S1 1];
        ta50x50S1=[ta50x50S1 0];
    end
    if i==1
        if dat(i,4)<=5
            fph50x50S1=0;
            tam50x50S1=1;
        else
            fph50x50S1=1;
            tam50x50S1=0;
        end
    else
        if dat(i,4)<=5
            fph50x50S1=[fph50x50S1 0];
            tam50x50S1=[tam50x50S1 1];
        else
            fph50x50S1=[fph50x50S1 1];
            tam50x50S1=[tam50x50S1 0];
        end
    end
end
nom_fichero='50x50/ResultadosHombreM2_05.m';
dat=leeFicheroPersona(nom_fichero,cadena);
[m,n]=size(dat);
for i=1:m
    if i==1
        if dat(i,2)==0
            tmi50x50A1=0.0001;
        else
            tmi50x50A1=dat(i,2);
        end
        if dat(i,4)>5
            fpm50x50A1=0;
            tah50x50A1=1;
        else
            fpm50x50A1=1;
            tah50x50A1=0;
        end
        if dat(i,4)==i+5
            fp50x50A1=0;
            ta50x50A1=1;
        else
            fp50x50A1=1;
            ta50x50A1=0;
        end
    else
```

```
        if dat(i,2)==0
            tmi50x50A1=[tmi50x50A1 0.0001];
        else
            tmi50x50A1=[tmi50x50A1 dat(i,2)];
        end
        if dat(i,4)>5
            fpm50x50A1=[fpm50x50A1 0];
            tah50x50A1=[tah50x50A1 1];
        else
            fpm50x50A1=[fpm50x50A1 1];
            tah50x50A1=[tah50x50A1 0];
        end
        if dat(i,4)==i+5
            fp50x50A1=[fp50x50A1 0];
            ta50x50A1=[ta50x50A1 1];
        else
            fp50x50A1=[fp50x50A1 1];
            ta50x50A1=[ta50x50A1 0];
        end
    end
end
nom_fichero='50x50/ResultadosMujerM2_05.m';
dat=leeFicheroPersona(nom_fichero,cadena);
[m,n]=size(dat);
for i=1:m
    if dat(i,2)==0
        tmi50x50A1=[tmi50x50A1 0.0001];
    else
        tmi50x50A1=[tmi50x50A1 dat(i,2)];
    end
    if dat(i,4)==i&&dat(i,4)<=5
        fp50x50A1=[fp50x50A1 0];
        ta50x50A1=[ta50x50A1 1];
    else
        fp50x50A1=[fp50x50A1 1];
        ta50x50A1=[ta50x50A1 0];
    end
    if i==1
        if dat(i,4)<=5
            fph50x50A1=0;
            tam50x50A1=1;
        else
            fph50x50A1=1;
            tam50x50A1=0;
        end
    else
        if dat(i,4)<=5
            fph50x50A1=[fph50x50A1 0];
            tam50x50A1=[tam50x50A1 1];
        else
            fph50x50A1=[fph50x50A1 1];
            tam50x50A1=[tam50x50A1 0];
        end
    end
end
nom_fichero='50x50/ResultadosHombreM2_22.m';
dat=leeFicheroPersona(nom_fichero,cadena);
[m,n]=size(dat);
for i=1:m

    if dat(i,2)==0
        tmi50x50S1=[tmi50x50S1 0.0001];
    else
        tmi50x50S1=[tmi50x50S1 dat(i,2)];
    end
    if dat(i,4)>25
        fpm50x50S1=[fpm50x50S1 0];
        tah50x50S1=[tah50x50S1 1];
    else
        fpm50x50S1=[fpm50x50S1 1];
        tah50x50S1=[tah50x50S1 0];
    end
    if dat(i,4)==i+25
        fp50x50S1=[fp50x50S1 0];
        ta50x50S1=[ta50x50S1 1];
    else
        fp50x50S1=[fp50x50S1 1];
        ta50x50S1=[ta50x50S1 0];
```

```
    end
end
nom_fichero='50x50/ResultadosMujerM2_22.m';
dat=leeFicheroPersona(nom_fichero,cadena);
[m,n]=size(dat);
for i=1:m
    if dat(i,2)==0
        tmi50x50S1=[tmi50x50S1 0.0001];
    else
        tmi50x50S1=[tmi50x50S1 dat(i,2)];
    end
    if dat(i,4)==i&&dat(i,4)<=25
        fp50x50S1=[fp50x50S1 0];
        ta50x50S1=[ta50x50S1 1];
    else
        fp50x50S1=[fp50x50S1 1];
        ta50x50S1=[ta50x50S1 0];
    end
    if dat(i,4)<=25
        fph50x50S1=[fph50x50S1 0];
        tam50x50S1=[tam50x50S1 1];
    else
        fph50x50S1=[fph50x50S1 1];
        tam50x50S1=[tam50x50S1 0];
    end
end
nom_fichero='50x50/ResultadosHombreM2_25.m';
dat=leeFicheroPersona(nom_fichero,cadena);
[m,n]=size(dat);
for i=1:m

    if dat(i,2)==0
        tmi50x50A1=[tmi50x50A1 0.0001];
    else
        tmi50x50A1=[tmi50x50A1 dat(i,2)];
    end
    if dat(i,4)>25
        fpm50x50A1=[fpm50x50A1 0];
        tah50x50A1=[tah50x50A1 1];
    else
        fpm50x50A1=[fpm50x50A1 1];
        tah50x50A1=[tah50x50A1 0];
    end
    if dat(i,4)==i+25
        fp50x50A1=[fp50x50A1 0];
        ta50x50A1=[ta50x50A1 1];
    else
        fp50x50A1=[fp50x50A1 1];
        ta50x50A1=[ta50x50A1 0];
    end
end
nom_fichero='50x50/ResultadosMujerM2_25.m';
dat=leeFicheroPersona(nom_fichero,cadena);
[m,n]=size(dat);
for i=1:m
    if dat(i,2)==0
        tmi50x50A1=[tmi50x50A1 0.0001];
    else
        tmi50x50A1=[tmi50x50A1 dat(i,2)];
    end
    if dat(i,4)==i&&dat(i,4)<=25
        fp50x50A1=[fp50x50A1 0];
        ta50x50A1=[ta50x50A1 1];
    else
        fp50x50A1=[fp50x50A1 1];
        ta50x50A1=[ta50x50A1 0];
    end
    if dat(i,4)<=25
        fph50x50A1=[fph50x50A1 0];
        tam50x50A1=[tam50x50A1 1];
    else
        fph50x50A1=[fph50x50A1 1];
        tam50x50A1=[tam50x50A1 0];
    end
end
nom_fichero='50x50/ResultadosHombreM2_42.m';
dat=leeFicheroPersona(nom_fichero,cadena);
```

```
[m,n]=size(dat);
for i=1:m

    if dat(i,2)==0
        tmi50x50S1=[tmi50x50S1 0.0001];
    else
        tmi50x50S1=[tmi50x50S1 dat(i,2)];
    end
    if dat(i,4)>50
        fpm50x50S1=[fpm50x50S1 0];
        tah50x50S1=[tah50x50S1 1];
    else
        fpm50x50S1=[fpm50x50S1 1];
        tah50x50S1=[tah50x50S1 0];
    end
    if dat(i,4)==i+50
        fp50x50S1=[fp50x50S1 0];
        ta50x50S1=[ta50x50S1 1];
    else
        fp50x50S1=[fp50x50S1 1];
        ta50x50S1=[ta50x50S1 0];
    end
end
nom_fichero='50x50/ResultadosMujerM2_42.m';
dat=leeFicheroPersona(nom_fichero,cadena);
[m,n]=size(dat);
for i=1:m
    if dat(i,2)==0
        tmi50x50S1=[tmi50x50S1 0.0001];
    else
        tmi50x50S1=[tmi50x50S1 dat(i,2)];
    end
    if dat(i,4)==i&&dat(i,4)<=50
        fp50x50S1=[fp50x50S1 0];
        ta50x50S1=[ta50x50S1 1];
    else
        fp50x50S1=[fp50x50S1 1];
        ta50x50S1=[ta50x50S1 0];
    end
    if dat(i,4)<=50
        fph50x50S1=[fph50x50S1 0];
        tam50x50S1=[tam50x50S1 1];
    else
        fph50x50S1=[fph50x50S1 1];
        tam50x50S1=[tam50x50S1 0];
    end
end
nom_fichero='50x50/ResultadosHombreM2_45.m';
dat=leeFicheroPersona(nom_fichero,cadena);
[m,n]=size(dat);
for i=1:m

    if dat(i,2)==0
        tmi50x50A1=[tmi50x50A1 0.0001];
    else
        tmi50x50A1=[tmi50x50A1 dat(i,2)];
    end
    if dat(i,4)>50
        fpm50x50A1=[fpm50x50A1 0];
        tah50x50A1=[tah50x50A1 1];
    else
        fpm50x50A1=[fpm50x50A1 1];
        tah50x50A1=[tah50x50A1 0];
    end
    if dat(i,4)==i+50
        fp50x50A1=[fp50x50A1 0];
        ta50x50A1=[ta50x50A1 1];
    else
        fp50x50A1=[fp50x50A1 1];
        ta50x50A1=[ta50x50A1 0];
    end
end
nom_fichero='50x50/ResultadosMujerM2_45.m';
dat=leeFicheroPersona(nom_fichero,cadena);
[m,n]=size(dat);
for i=1:m
    if dat(i,2)==0
        tmi50x50A1=[tmi50x50A1 0.0001];
```

```
    else
        tmi50x50A1=[tmi50x50A1 dat(i,2)];
    end
    if dat(i,4)==i&&dat(i,4)<=50
        fp50x50A1=[fp50x50A1 0];
        ta50x50A1=[ta50x50A1 1];
    else
        fp50x50A1=[fp50x50A1 1];
        ta50x50A1=[ta50x50A1 0];
    end
    if dat(i,4)<=50
        fph50x50A1=[fph50x50A1 0];
        tam50x50A1=[tam50x50A1 1];
    else
        fph50x50A1=[fph50x50A1 1];
        tam50x50A1=[tam50x50A1 0];
    end
end

nom_fichero='50x50/ResultadosHombreM3_02.m';
dat=leeFicheroPersona(nom_fichero,cadena);
[m,n]=size(dat);
for i=1:m

    if dat(i,2)==0
        tmi50x50S1=[tmi50x50S1 0.0001];
    else
        tmi50x50S1=[tmi50x50S1 dat(i,2)];
    end
    if dat(i,4)>5
        fpm50x50S1=[fpm50x50S1 0];
        tah50x50S1=[tah50x50S1 1];
    else
        fpm50x50S1=[fpm50x50S1 1];
        tah50x50S1=[tah50x50S1 0];
    end
    if dat(i,4)==i+5
        fp50x50S1=[fp50x50S1 0];
        ta50x50S1=[ta50x50S1 1];
    else
        fp50x50S1=[fp50x50S1 1];
        ta50x50S1=[ta50x50S1 0];
    end
end
nom_fichero='50x50/ResultadosMujerM3_02.m';
dat=leeFicheroPersona(nom_fichero,cadena);
[m,n]=size(dat);
for i=1:m
    if dat(i,2)==0
        tmi50x50S1=[tmi50x50S1 0.0001];
    else
        tmi50x50S1=[tmi50x50S1 dat(i,2)];
    end
    if dat(i,4)==i&&dat(i,4)<=5
        fp50x50S1=[fp50x50S1 0];
        ta50x50S1=[ta50x50S1 1];
    else
        fp50x50S1=[fp50x50S1 1];
        ta50x50S1=[ta50x50S1 0];
    end
    if dat(i,4)<=5
        fph50x50S1=[fph50x50S1 0];
        tam50x50S1=[tam50x50S1 1];
    else
        fph50x50S1=[fph50x50S1 1];
        tam50x50S1=[tam50x50S1 0];
    end
end
nom_fichero='50x50/ResultadosHombreM3_05.m';
dat=leeFicheroPersona(nom_fichero,cadena);
[m,n]=size(dat);
for i=1:m

    if dat(i,2)==0
        tmi50x50A1=[tmi50x50A1 0.0001];
    else
        tmi50x50A1=[tmi50x50A1 dat(i,2)];
```

```
    end
    if dat(i,4)>5
        fpm50x50A1=[fpm50x50A1 0];
        tah50x50A1=[tah50x50A1 1];
    else
        fpm50x50A1=[fpm50x50A1 1];
        tah50x50A1=[tah50x50A1 0];
    end
    if dat(i,4)==i+5
        fp50x50A1=[fp50x50A1 0];
        ta50x50A1=[ta50x50A1 1];
    else
        fp50x50A1=[fp50x50A1 1];
        ta50x50A1=[ta50x50A1 0];
    end
end
nom_fichero='50x50/ResultadosMujerM3_05.m';
dat=leeFicheroPersona(nom_fichero,cadena);
[m,n]=size(dat);
for i=1:m
    if dat(i,2)==0
        tmi50x50A1=[tmi50x50A1 0.0001];
    else
        tmi50x50A1=[tmi50x50A1 dat(i,2)];
    end
    if dat(i,4)==i&&dat(i,4)<=5
        fp50x50A1=[fp50x50A1 0];
        ta50x50A1=[ta50x50A1 1];
    else
        fp50x50A1=[fp50x50A1 1];
        ta50x50A1=[ta50x50A1 0];
    end
    if dat(i,4)<=5
        fph50x50A1=[fph50x50A1 0];
        tam50x50A1=[tam50x50A1 1];
    else
        fph50x50A1=[fph50x50A1 1];
        tam50x50A1=[tam50x50A1 0];
    end
end
nom_fichero='50x50/ResultadosHombreM3_12.m';
dat=leeFicheroPersona(nom_fichero,cadena);
[m,n]=size(dat);
for i=1:m
    if i==1
        if dat(i,2)==0
            tmi50x50S2=0.0001;
        else
            tmi50x50S2=dat(i,2);
        end
        if dat(i,4)>5
            fpm50x50S2=0;
            tah50x50S2=1;
        else
            fpm50x50S2=1;
            tah50x50S2=0;
        end
        if dat(i,4)==i+5
            fp50x50S2=0;
            ta50x50S2=1;
        else
            fp50x50S2=1;
            ta50x50S2=0;
        end
    else
        if dat(i,2)==0
            tmi50x50S2=[tmi50x50S2 0.0001];
        else
            tmi50x50S2=[tmi50x50S2 dat(i,2)];
        end
        if dat(i,4)>5
            fpm50x50S2=[fpm50x50S2 0];
            tah50x50S2=[tah50x50S2 1];
        else
            fpm50x50S2=[fpm50x50S2 1];
            tah50x50S2=[tah50x50S2 0];
        end
        if dat(i,4)==i+5
```

```
            fp50x50S2=[fp50x50S2 0];
            ta50x50S2=[ta50x50S2 1];
        else
            fp50x50S2=[fp50x50S2 1];
            ta50x50S2=[ta50x50S2 0];
        end
    end
end
nom_fichero='50x50/ResultadosMujerM3_12.m';
dat=leeFicheroPersona(nom_fichero,cadena);
[m,n]=size(dat);
for i=1:m
    if dat(i,2)==0
        tmi50x50S2=[tmi50x50S2 0.0001];
    else
        tmi50x50S2=[tmi50x50S2 dat(i,2)];
    end
    if dat(i,4)==i&&dat(i,4)<=5
        fp50x50S2=[fp50x50S2 0];
        ta50x50S2=[ta50x50S2 1];
    else
        fp50x50S2=[fp50x50S2 1];
        ta50x50S2=[ta50x50S2 0];
    end
    if i==1
        if dat(i,4)<=5
            fph50x50S2=0;
            tam50x50S2=1;
        else
            fph50x50S2=1;
            tam50x50S2=0;
        end
    else
        if dat(i,4)<=5
            fph50x50S2=[fph50x50S2 0];
            tam50x50S2=[tam50x50S2 1];
        else
            fph50x50S2=[fph50x50S2 1];
            tam50x50S2=[tam50x50S2 0];
        end
    end
end
nom_fichero='50x50/ResultadosHombreM3_15.m';
dat=leeFicheroPersona(nom_fichero,cadena);
[m,n]=size(dat);
for i=1:m
    if i==1
        if dat(i,2)==0
            tmi50x50A2=0.0001;
        else
            tmi50x50A2=dat(i,2);
        end
        if dat(i,4)>5
            fpm50x50A2=0;
            tah50x50A2=1;
        else
            fpm50x50A2=1;
            tah50x50A2=0;
        end
        if dat(i,4)==i+5
            fp50x50A2=0;
            ta50x50A2=1;
        else
            fp50x50A2=1;
            ta50x50A2=0;
        end
    else
        if dat(i,2)==0
            tmi50x50A2=[tmi50x50A2 0.0001];
        else
            tmi50x50A2=[tmi50x50A2 dat(i,2)];
        end
        if dat(i,4)>5
            fpm50x50A2=[fpm50x50A2 0];
            tah50x50A2=[tah50x50A2 1];
        else
            fpm50x50A2=[fpm50x50A2 1];
```

```
            tah50x50A2=[tah50x50A2 0];
        end
        if dat(i,4)==i+5
            fp50x50A2=[fp50x50A2 0];
            ta50x50A2=[ta50x50A2 1];
        else
            fp50x50A2=[fp50x50A2 1];
            ta50x50A2=[ta50x50A2 0];
        end
    end
end
nom_fichero='50x50/ResultadosMujerM3_15.m';
dat=leeFicheroPersona(nom_fichero,cadena);
[m,n]=size(dat);
for i=1:m
    if dat(i,2)==0
        tmi50x50A2=[tmi50x50A2 0.0001];
    else
        tmi50x50A2=[tmi50x50A2 dat(i,2)];
    end
    if dat(i,4)==i&&dat(i,4)<=5
        fp50x50A2=[fp50x50A2 0];
        ta50x50A2=[ta50x50A2 1];
    else
        fp50x50A2=[fp50x50A2 1];
        ta50x50A2=[ta50x50A2 0];
    end
    if i==1
        if dat(i,4)<=5
            fph50x50A2=0;
            tam50x50A2=1;
        else
            fph50x50A2=1;
            tam50x50A2=0;
        end
    else
        if dat(i,4)<=5
            fph50x50A2=[fph50x50A2 0];
            tam50x50A2=[tam50x50A2 1];
        else
            fph50x50A2=[fph50x50A2 1];
            tam50x50A2=[tam50x50A2 0];
        end
    end
end
nom_fichero='50x50/ResultadosHombreM3_22.m';
dat=leeFicheroPersona(nom_fichero,cadena);
[m,n]=size(dat);
for i=1:m

    if dat(i,2)==0
        tmi50x50S1=[tmi50x50S1 0.0001];
    else
        tmi50x50S1=[tmi50x50S1 dat(i,2)];
    end
    if dat(i,4)>25
        fpm50x50S1=[fpm50x50S1 0];
        tah50x50S1=[tah50x50S1 1];
    else
        fpm50x50S1=[fpm50x50S1 1];
        tah50x50S1=[tah50x50S1 0];
    end
    if dat(i,4)==i+25
        fp50x50S1=[fp50x50S1 0];
        ta50x50S1=[ta50x50S1 1];
    else
        fp50x50S1=[fp50x50S1 1];
        ta50x50S1=[ta50x50S1 0];
    end
end
nom_fichero='50x50/ResultadosMujerM3_22.m';
dat=leeFicheroPersona(nom_fichero,cadena);
[m,n]=size(dat);
for i=1:m
    if dat(i,2)==0
        tmi50x50S1=[tmi50x50S1 0.0001];
    else
        tmi50x50S1=[tmi50x50S1 dat(i,2)];
```

```
    end
    if dat(i,4)==i&&dat(i,4)<=25
        fp50x50S1=[fp50x50S1 0];
        ta50x50S1=[ta50x50S1 1];
    else
        fp50x50S1=[fp50x50S1 1];
        ta50x50S1=[ta50x50S1 0];
    end
    if dat(i,4)<=25
        fph50x50S1=[fph50x50S1 0];
        tam50x50S1=[tam50x50S1 1];
    else
        fph50x50S1=[fph50x50S1 1];
        tam50x50S1=[tam50x50S1 0];
    end
end
nom_fichero='50x50/ResultadosHombreM3_25.m';
dat=leeFicheroPersona(nom_fichero,cadena);
[m,n]=size(dat);
for i=1:m

    if dat(i,2)==0
        tmi50x50A1=[tmi50x50A1 0.0001];
    else
        tmi50x50A1=[tmi50x50A1 dat(i,2)];
    end
    if dat(i,4)>25
        fpm50x50A1=[fpm50x50A1 0];
        tah50x50A1=[tah50x50A1 1];
    else
        fpm50x50A1=[fpm50x50A1 1];
        tah50x50A1=[tah50x50A1 0];
    end
    if dat(i,4)==i+25
        fp50x50A1=[fp50x50A1 0];
        ta50x50A1=[ta50x50A1 1];
    else
        fp50x50A1=[fp50x50A1 1];
        ta50x50A1=[ta50x50A1 0];
    end
end
nom_fichero='50x50/ResultadosMujerM3_25.m';
dat=leeFicheroPersona(nom_fichero,cadena);
[m,n]=size(dat);
for i=1:m
    if dat(i,2)==0
        tmi50x50A1=[tmi50x50A1 0.0001];
    else
        tmi50x50A1=[tmi50x50A1 dat(i,2)];
    end
    if dat(i,4)==i&&dat(i,4)<=25
        fp50x50A1=[fp50x50A1 0];
        ta50x50A1=[ta50x50A1 1];
    else
        fp50x50A1=[fp50x50A1 1];
        ta50x50A1=[ta50x50A1 0];
    end
    if dat(i,4)<=25
        fph50x50A1=[fph50x50A1 0];
        tam50x50A1=[tam50x50A1 1];
    else
        fph50x50A1=[fph50x50A1 1];
        tam50x50A1=[tam50x50A1 0];
    end
end
nom_fichero='50x50/ResultadosHombreM3_32.m';
dat=leeFicheroPersona(nom_fichero,cadena);
[m,n]=size(dat);
for i=1:m

    if dat(i,2)==0
        tmi50x50S2=[tmi50x50S2 0.0001];
    else
        tmi50x50S2=[tmi50x50S2 dat(i,2)];
    end
    if dat(i,4)>25
        fpm50x50S2=[fpm50x50S2 0];
```

```
        tah50x50S2=[tah50x50S2 1];
    else
        fpm50x50S2=[fpm50x50S2 1];
        tah50x50S2=[tah50x50S2 0];
    end
    if dat(i,4)==i+25
        fp50x50S2=[fp50x50S2 0];
        ta50x50S2=[ta50x50S2 1];
    else
        fp50x50S2=[fp50x50S2 1];
        ta50x50S2=[ta50x50S2 0];
    end
end
nom_fichero='50x50/ResultadosMujerM3_32.m';
dat=leeFicheroPersona(nom_fichero,cadena);
[m,n]=size(dat);
for i=1:m
    if dat(i,2)==0
        tmi50x50S2=[tmi50x50S2 0.0001];
    else
        tmi50x50S2=[tmi50x50S2 dat(i,2)];
    end
    if dat(i,4)==i&&dat(i,4)<=25
        fp50x50S2=[fp50x50S2 0];
        ta50x50S2=[ta50x50S2 1];
    else
        fp50x50S2=[fp50x50S2 1];
        ta50x50S2=[ta50x50S2 0];
    end
    if dat(i,4)<=25
        fph50x50S2=[fph50x50S2 0];
        tam50x50S2=[tam50x50S2 1];
    else
        fph50x50S2=[fph50x50S2 1];
        tam50x50S2=[tam50x50S2 0];
    end
end
nom_fichero='50x50/ResultadosHombreM3_35.m';
dat=leeFicheroPersona(nom_fichero,cadena);
[m,n]=size(dat);
for i=1:m

    if dat(i,2)==0
        tmi50x50A2=[tmi50x50A2 0.0001];
    else
        tmi50x50A2=[tmi50x50A2 dat(i,2)];
    end
    if dat(i,4)>25
        fpm50x50A2=[fpm50x50A2 0];
        tah50x50A2=[tah50x50A2 1];
    else
        fpm50x50A2=[fpm50x50A2 1];
        tah50x50A2=[tah50x50A2 0];
    end
    if dat(i,4)==i+25
        fp50x50A2=[fp50x50A2 0];
        ta50x50A2=[ta50x50A2 1];
    else
        fp50x50A2=[fp50x50A2 1];
        ta50x50A2=[ta50x50A2 0];
    end
end
nom_fichero='50x50/ResultadosMujerM3_35.m';
dat=leeFicheroPersona(nom_fichero,cadena);
[m,n]=size(dat);
for i=1:m
    if dat(i,2)==0
        tmi50x50A2=[tmi50x50A2 0.0001];
    else
        tmi50x50A2=[tmi50x50A2 dat(i,2)];
    end
    if dat(i,4)==i&&dat(i,4)<=25
        fp50x50A2=[fp50x50A2 0];
        ta50x50A2=[ta50x50A2 1];
    else
        fp50x50A2=[fp50x50A2 1];
        ta50x50A2=[ta50x50A2 0];
    end
```

```
    if dat(i,4)<=25
        fph50x50A2=[fph50x50A2 0];
        tam50x50A2=[tam50x50A2 1];
    else
        fph50x50A2=[fph50x50A2 1];
        tam50x50A2=[tam50x50A2 0];
    end
end
nom_fichero='50x50/ResultadosHombreM3_42.m';
dat=leeFicheroPersona(nom_fichero,cadena);
[m,n]=size(dat);
for i=1:m

    if dat(i,2)==0
        tmi50x50S1=[tmi50x50S1 0.0001];
    else
        tmi50x50S1=[tmi50x50S1 dat(i,2)];
    end
    if dat(i,4)>50
        fpm50x50S1=[fpm50x50S1 0];
        tah50x50S1=[tah50x50S1 1];
    else
        fpm50x50S1=[fpm50x50S1 1];
        tah50x50S1=[tah50x50S1 0];
    end
    if dat(i,4)==i+50
        fp50x50S1=[fp50x50S1 0];
        ta50x50S1=[ta50x50S1 1];
    else
        fp50x50S1=[fp50x50S1 1];
        ta50x50S1=[ta50x50S1 0];
    end
end
nom_fichero='50x50/ResultadosMujerM3_42.m';
dat=leeFicheroPersona(nom_fichero,cadena);
[m,n]=size(dat);
for i=1:m
    if dat(i,2)==0
        tmi50x50S1=[tmi50x50S1 0.0001];
    else
        tmi50x50S1=[tmi50x50S1 dat(i,2)];
    end
    if dat(i,4)==i&&dat(i,4)<=50
        fp50x50S1=[fp50x50S1 0];
        ta50x50S1=[ta50x50S1 1];
    else
        fp50x50S1=[fp50x50S1 1];
        ta50x50S1=[ta50x50S1 0];
    end
    if dat(i,4)<=50
        fph50x50S1=[fph50x50S1 0];
        tam50x50S1=[tam50x50S1 1];
    else
        fph50x50S1=[fph50x50S1 1];
        tam50x50S1=[tam50x50S1 0];
    end
end
nom_fichero='50x50/ResultadosHombreM3_45.m';
dat=leeFicheroPersona(nom_fichero,cadena);
[m,n]=size(dat);
for i=1:m

    if dat(i,2)==0
        tmi50x50A1=[tmi50x50A1 0.0001];
    else
        tmi50x50A1=[tmi50x50A1 dat(i,2)];
    end
    if dat(i,4)>50
        fpm50x50A1=[fpm50x50A1 0];
        tah50x50A1=[tah50x50A1 1];
    else
        fpm50x50A1=[fpm50x50A1 1];
        tah50x50A1=[tah50x50A1 0];
    end
    if dat(i,4)==i+50
        fp50x50A1=[fp50x50A1 0];
        ta50x50A1=[ta50x50A1 1];
```

```
    else
        fp50x50A1=[fp50x50A1 1];
        ta50x50A1=[ta50x50A1 0];
    end
end
nom_fichero='50x50/ResultadosMujerM3_45.m';
dat=leeFicheroPersona(nom_fichero,cadena);
[m,n]=size(dat);
for i=1:m
    if dat(i,2)==0
        tmi50x50A1=[tmi50x50A1 0.0001];
    else
        tmi50x50A1=[tmi50x50A1 dat(i,2)];
    end
    if dat(i,4)==i&&dat(i,4)<=50
        fp50x50A1=[fp50x50A1 0];
        ta50x50A1=[ta50x50A1 1];
    else
        fp50x50A1=[fp50x50A1 1];
        ta50x50A1=[ta50x50A1 0];
    end
    if dat(i,4)<=50
        fph50x50A1=[fph50x50A1 0];
        tam50x50A1=[tam50x50A1 1];
    else
        fph50x50A1=[fph50x50A1 1];
        tam50x50A1=[tam50x50A1 0];
    end
end
nom_fichero='50x50/ResultadosHombreM3_52.m';
dat=leeFicheroPersona(nom_fichero,cadena);
[m,n]=size(dat);
for i=1:m

    if dat(i,2)==0
        tmi50x50S2=[tmi50x50S2 0.0001];
    else
        tmi50x50S2=[tmi50x50S2 dat(i,2)];
    end
    if dat(i,4)>50
        fpm50x50S2=[fpm50x50S2 0];
        tah50x50S2=[tah50x50S2 1];
    else
        fpm50x50S2=[fpm50x50S2 1];
        tah50x50S2=[tah50x50S2 0];
    end
    if dat(i,4)==i+50
        fp50x50S2=[fp50x50S2 0];
        ta50x50S2=[ta50x50S2 1];
    else
        fp50x50S2=[fp50x50S2 1];
        ta50x50S2=[ta50x50S2 0];
    end
end
nom_fichero='50x50/ResultadosMujerM3_52.m';
dat=leeFicheroPersona(nom_fichero,cadena);
[m,n]=size(dat);
for i=1:m
    if dat(i,2)==0
        tmi50x50S2=[tmi50x50S2 0.0001];
    else
        tmi50x50S2=[tmi50x50S2 dat(i,2)];
    end
    if dat(i,4)==i&&dat(i,4)<=50
        fp50x50S2=[fp50x50S2 0];
        ta50x50S2=[ta50x50S2 1];
    else
        fp50x50S2=[fp50x50S2 1];
        ta50x50S2=[ta50x50S2 0];
    end
    if dat(i,4)<=50
        fph50x50S2=[fph50x50S2 0];
        tam50x50S2=[tam50x50S2 1];
    else
        fph50x50S2=[fph50x50S2 1];
        tam50x50S2=[tam50x50S2 0];
    end
end
```

```
nom_fichero='50x50/ResultadosHombreM3_55.m';
dat=leeFicheroPersona(nom_fichero,cadena);
[m,n]=size(dat);
for i=1:m

    if dat(i,2)==0
        tmi50x50A2=[tmi50x50A2 0.0001];
    else
        tmi50x50A2=[tmi50x50A2 dat(i,2)];
    end
    if dat(i,4)>50
        fpm50x50A2=[fpm50x50A2 0];
        tah50x50A2=[tah50x50A2 1];
    else
        fpm50x50A2=[fpm50x50A2 1];
        tah50x50A2=[tah50x50A2 0];
    end
    if dat(i,4)==i+50
        fp50x50A2=[fp50x50A2 0];
        ta50x50A2=[ta50x50A2 1];
    else
        fp50x50A2=[fp50x50A2 1];
        ta50x50A2=[ta50x50A2 0];
    end
end
nom_fichero='50x50/ResultadosMujerM3_55.m';
dat=leeFicheroPersona(nom_fichero,cadena);
[m,n]=size(dat);
for i=1:m
    if dat(i,2)==0
        tmi50x50A2=[tmi50x50A2 0.0001];
    else
        tmi50x50A2=[tmi50x50A2 dat(i,2)];
    end
    if dat(i,4)==i&&dat(i,4)<=50
        fp50x50A2=[fp50x50A2 0];
        ta50x50A2=[ta50x50A2 1];
    else
        fp50x50A2=[fp50x50A2 1];
        ta50x50A2=[ta50x50A2 0];
    end
    if dat(i,4)<=50
        fph50x50A2=[fph50x50A2 0];
        tam50x50A2=[tam50x50A2 1];
    else
        fph50x50A2=[fph50x50A2 1];
        tam50x50A2=[tam50x50A2 0];
    end
end

% Generados todos los datos, se hace la clasificacion para 10 clases,
% 50 clases y 100 clases.
m=size(tmi50x50S1,2);
tmi50x50S12=reshape(tmi50x50S1,m/6,6);
tmi50x50S1=[(sum(tmi50x50S12(:,1))+sum(tmi50x50S12(:,4)))/(m/3)
(sum(tmi50x50S12(:,2))+sum(tmi50x50S12(:,5)))/(m/3)
(sum(tmi50x50S12(:,3))+sum(tmi50x50S12(:,6)))/(m/3)];
tmi50x50S12=0;

m=size(tmi50x50A1,2);
tmi50x50A12=reshape(tmi50x50A1,m/6,6);
tmi50x50A1=[(sum(tmi50x50A12(:,1))+sum(tmi50x50A12(:,4)))/(m/3)
(sum(tmi50x50A12(:,2))+sum(tmi50x50A12(:,5)))/(m/3)
(sum(tmi50x50A12(:,3))+sum(tmi50x50A12(:,6)))/(m/3)];
tmi50x50A12=0;

m=size(tmi50x50S2,2);
tmi50x50S22=reshape(tmi50x50S2,m/3,3);
tmi50x50S2=[sum(tmi50x50S22(:,1))/(m/3) sum(tmi50x50S22(:,2))/(m/3) sum(tmi50x50S22(:,3))/(m/3)];
tmi50x50S22=0;

m=size(tmi50x50A2,2);
tmi50x50A22=reshape(tmi50x50A2,m/3,3);
tmi50x50A2=[sum(tmi50x50A22(:,1))/(m/3) sum(tmi50x50A22(:,2))/(m/3) sum(tmi50x50A22(:,3))/(m/3)];
tmi50x50A22=0;
```

```
m=size(fpm50x50S1,2);
fpm50x50S12=reshape(fpm50x50S1,m/6,6);
fpm50x50S1=[(sum(fpm50x50S12(:,1))+sum(fpm50x50S12(:,4)))/(m/3)
(sum(fpm50x50S12(:,2))+sum(fpm50x50S12(:,5)))/(m/3)
(sum(fpm50x50S12(:,3))+sum(fpm50x50S12(:,6)))/(m/3)];
fpm50x50S12=0;

m=size(fpm50x50A1,2);
fpm50x50A12=reshape(fpm50x50A1,m/6,6);
fpm50x50A1=[(sum(fpm50x50A12(:,1))+sum(fpm50x50A12(:,4)))/(m/3)
(sum(fpm50x50A12(:,2))+sum(fpm50x50A12(:,5)))/(m/3)
(sum(fpm50x50A12(:,3))+sum(fpm50x50A12(:,6)))/(m/3)];
fpm50x50A12=0;

m=size(fpm50x50S2,2);
fpm50x50S22=reshape(fpm50x50S2,m/3,3);
fpm50x50S2=[sum(fpm50x50S22(:,1))/(m/3) sum(fpm50x50S22(:,2))/(m/3) sum(fpm50x50S22(:,3))/(m/3)];
fpm50x50S22=0;

m=size(fpm50x50A2,2);
fpm50x50A22=reshape(fpm50x50A2,m/3,3);
fpm50x50A2=[sum(fpm50x50A22(:,1))/(m/3) sum(fpm50x50A22(:,2))/(m/3) sum(fpm50x50A22(:,3))/(m/3)];
fpm50x50A22=0;


m=size(fph50x50S1,2);
fph50x50S12=reshape(fph50x50S1,m/6,6);
fph50x50S1=[(sum(fph50x50S12(:,1))+sum(fph50x50S12(:,4)))/(m/3)
(sum(fph50x50S12(:,2))+sum(fph50x50S12(:,5)))/(m/3)
(sum(fph50x50S12(:,3))+sum(fph50x50S12(:,6)))/(m/3)];
fph50x50S12=0;

m=size(fph50x50A1,2);
fph50x50A12=reshape(fph50x50A1,m/6,6);
fph50x50A1=[(sum(fph50x50A12(:,1))+sum(fph50x50A12(:,4)))/(m/3)
(sum(fph50x50A12(:,2))+sum(fph50x50A12(:,5)))/(m/3)
(sum(fph50x50A12(:,3))+sum(fph50x50A12(:,6)))/(m/3)];
fph50x50A12=0;

m=size(fph50x50S2,2);
fph50x50S22=reshape(fph50x50S2,m/3,3);
fph50x50S2=[sum(fph50x50S22(:,1))/(m/3) sum(fph50x50S22(:,2))/(m/3) sum(fph50x50S22(:,3))/(m/3)];
fph50x50S22=0;

m=size(fph50x50A2,2);
fph50x50A22=reshape(fph50x50A2,m/3,3);
fph50x50A2=[sum(fph50x50A22(:,1))/(m/3) sum(fph50x50A22(:,2))/(m/3) sum(fph50x50A22(:,3))/(m/3)];
fph50x50A22=0;


m=size(tah50x50S1,2);
tah50x50S12=reshape(tah50x50S1,m/6,6);
tah50x50S1=[(sum(tah50x50S12(:,1))+sum(tah50x50S12(:,4)))/(m/3)
(sum(tah50x50S12(:,2))+sum(tah50x50S12(:,5)))/(m/3)
(sum(tah50x50S12(:,3))+sum(tah50x50S12(:,6)))/(m/3)];
tah50x50S12=0;

m=size(tah50x50A1,2);
tah50x50A12=reshape(tah50x50A1,m/6,6);
tah50x50A1=[(sum(tah50x50A12(:,1))+sum(tah50x50A12(:,4)))/(m/3)
(sum(tah50x50A12(:,2))+sum(tah50x50A12(:,5)))/(m/3)
(sum(tah50x50A12(:,3))+sum(tah50x50A12(:,6)))/(m/3)];
tah50x50A12=0;

m=size(tah50x50S2,2);
tah50x50S22=reshape(tah50x50S2,m/3,3);
tah50x50S2=[sum(tah50x50S22(:,1))/(m/3) sum(tah50x50S22(:,2))/(m/3) sum(tah50x50S22(:,3))/(m/3)];
tah50x50S22=0;

m=size(tah50x50A2,2);
tah50x50A22=reshape(tah50x50A2,m/3,3);
tah50x50A2=[sum(tah50x50A22(:,1))/(m/3) sum(tah50x50A22(:,2))/(m/3) sum(tah50x50A22(:,3))/(m/3)];
tah50x50A22=0;


m=size(tam50x50S1,2);
tam50x50S12=reshape(tam50x50S1,m/6,6);
```

```
tam50x50S1=[(sum(tam50x50S12(:,1))+sum(tam50x50S12(:,4)))/(m/3)
(sum(tam50x50S12(:,2))+sum(tam50x50S12(:,5)))/(m/3)
(sum(tam50x50S12(:,3))+sum(tam50x50S12(:,6)))/(m/3)];
tam50x50S12=0;

m=size(tam50x50A1,2);
tam50x50A12=reshape(tam50x50A1,m/6,6);
tam50x50A1=[(sum(tam50x50A12(:,1))+sum(tam50x50A12(:,4)))/(m/3)
(sum(tam50x50A12(:,2))+sum(tam50x50A12(:,5)))/(m/3)
(sum(tam50x50A12(:,3))+sum(tam50x50A12(:,6)))/(m/3)];
tam50x50A12=0;

m=size(tam50x50S2,2);
tam50x50S22=reshape(tam50x50S2,m/3,3);
tam50x50S2=[sum(tam50x50S22(:,1))/(m/3) sum(tam50x50S22(:,2))/(m/3) sum(tam50x50S22(:,3))/(m/3)];
tam50x50S22=0;

m=size(tam50x50A2,2);
tam50x50A22=reshape(tam50x50A2,m/3,3);
tam50x50A2=[sum(tam50x50A22(:,1))/(m/3) sum(tam50x50A22(:,2))/(m/3) sum(tam50x50A22(:,3))/(m/3)];
tam50x50A22=0;


m=size(fp50x50S1,2);
fp50x50S12=reshape(fp50x50S1,m/6,6);
fp50x50S1=[(sum(fp50x50S12(:,1))+sum(fp50x50S12(:,4)))/(m/3)
(sum(fp50x50S12(:,2))+sum(fp50x50S12(:,5)))/(m/3)
(sum(fp50x50S12(:,3))+sum(fp50x50S12(:,6)))/(m/3)];
fp50x50S12=0;

m=size(fp50x50A1,2);
fp50x50A12=reshape(fp50x50A1,m/6,6);
fp50x50A1=[(sum(fp50x50A12(:,1))+sum(fp50x50A12(:,4)))/(m/3)
(sum(fp50x50A12(:,2))+sum(fp50x50A12(:,5)))/(m/3)
(sum(fp50x50A12(:,3))+sum(fp50x50A12(:,6)))/(m/3)];
fp50x50A12=0;

m=size(fp50x50S2,2);
fp50x50S22=reshape(fp50x50S2,m/3,3);
fp50x50S2=[sum(fp50x50S22(:,1))/(m/3) sum(fp50x50S22(:,2))/(m/3) sum(fp50x50S22(:,3))/(m/3)];
fp50x50S22=0;

fp50x50A2;
m=size(fp50x50A2,2);
fp50x50A22=reshape(fp50x50A2,m/3,3);
fp50x50A2=[sum(fp50x50A22(:,1))/(m/3) sum(fp50x50A22(:,2))/(m/3) sum(fp50x50A22(:,3))/(m/3)];
fp50x50A22=0;


m=size(ta50x50S1,2);
ta50x50S12=reshape(ta50x50S1,m/6,6);
ta50x50S1=[(sum(ta50x50S12(:,1))+sum(ta50x50S12(:,4)))/20
(sum(ta50x50S12(:,2))+sum(ta50x50S12(:,5)))/100 (sum(ta50x50S12(:,3))+sum(ta50x50S12(:,6)))/200];
ta50x50S12=0;

m=size(ta50x50A1,2);
ta50x50A12=reshape(ta50x50A1,m/6,6);
ta50x50A1=[(sum(ta50x50A12(:,1))+sum(ta50x50A12(:,4)))/20
(sum(ta50x50A12(:,2))+sum(ta50x50A12(:,5)))/100 (sum(ta50x50A12(:,3))+sum(ta50x50A12(:,6)))/200];
ta50x50A12=0;

m=size(ta50x50S2,2);
ta50x50S22=reshape(ta50x50S2,m/3,3);
ta50x50S2=[sum(ta50x50S22(:,1))/10 sum(ta50x50S22(:,2))/50 sum(ta50x50S22(:,3))/100];
ta50x50S22=0;

ta50x50A2;
m=size(ta50x50A2,2);
ta50x50A22=reshape(ta50x50A2,m/3,3);
ta50x50A2=[sum(ta50x50A22(:,1))/10 sum(ta50x50A22(:,2))/50 sum(ta50x50A22(:,3))/100];
ta50x50A22=0;
```

Fichero analizaFicheroPersona100x100.m:

```
%
% Fichero: analizaFicheroPersona100x100.m
% Autor: Maria Sierra Zapata
% Fecha: 01/07/2015
% Version: 0.1
%
% Breve Descripcion:
%       Apertura de los ficheros generados debido a la
%       identificacion de personas para un modelo y con tamaño de imagen
%       100x100 pixeles. Seleccion de los datos para la generacion de las
%       graficas para el estudio.
%
% Datos que se obtienen:
%       Tiempo Medio de Identificacion
%       Falsos Positivos Mujer
%       Falsos Positivos Hombre
%       Falsos Positivos
%       Tasa de Acierto Hombre
%       Tasa de Acierto Mujer
%       Tasa de Acierto
%

cadena='%d=%f %f %d';
num_images=[10 50 100];

% Apertura y lectura del fichero
nom_fichero='100x100/ResultadosHombreM2_02.m';
dat=leeFicheroPersona(nom_fichero,cadena);
% Obtencion de los datos del fichero
[m,n]=size(dat);
for i=1:m
    if i==1
        if dat(i,2)==0
            tmi100x100S1=0.0001;
        else
            tmi100x100S1=dat(i,2);
        end
        if dat(i,4)>5
            fpm100x100S1=0;
            tah100x100S1=1;
        else
            fpm100x100S1=1;
            tah100x100S1=0;
        end
        if dat(i,4)==i+5
            fp100x100S1=0;
            ta100x100S1=1;
        else
            fp100x100S1=1;
            ta100x100S1=0;
        end
    else
        if dat(i,2)==0
            tmi100x100S1=[tmi100x100S1 0.0001];
        else
            tmi100x100S1=[tmi100x100S1 dat(i,2)];
        end
        if dat(i,4)>5
            fpm100x100S1=[fpm100x100S1 0];
            tah100x100S1=[tah100x100S1 1];
        else
            fpm100x100S1=[fpm100x100S1 1];
            tah100x100S1=[tah100x100S1 0];
        end
        if dat(i,4)==i+5
            fp100x100S1=[fp100x100S1 0];
            ta100x100S1=[ta100x100S1 1];
        else
            fp100x100S1=[fp100x100S1 1];
            ta100x100S1=[ta100x100S1 0];
        end
    end
end
nom_fichero='100x100/ResultadosMujerM2_02.m';
dat=leeFicheroPersona(nom_fichero,cadena);
```

```
[m,n]=size(dat);
for i=1:m
    if dat(i,2)==0
        tmi100x100S1=[tmi100x100S1 0.0001];
    else
        tmi100x100S1=[tmi100x100S1 dat(i,2)];
    end
    if dat(i,4)==i&&dat(i,4)<=5
        fp100x100S1=[fp100x100S1 0];
        ta100x100S1=[ta100x100S1 1];
    else
        fp100x100S1=[fp100x100S1 1];
        ta100x100S1=[ta100x100S1 0];
    end
    if i==1
        if dat(i,4)<=5
            fph100x100S1=0;
            tam100x100S1=1;
        else
            fph100x100S1=1;
            tam100x100S1=0;
        end
    else
        if dat(i,4)<=5
            fph100x100S1=[fph100x100S1 0];
            tam100x100S1=[tam100x100S1 1];
        else
            fph100x100S1=[fph100x100S1 1];
            tam100x100S1=[tam100x100S1 0];
        end
    end
end
nom_fichero='100x100/ResultadosHombreM2_05.m';
dat=leeFicheroPersona(nom_fichero,cadena);
[m,n]=size(dat);
for i=1:m
    if i==1
        if dat(i,2)==0
            tmi100x100A1=0.0001;
        else
            tmi100x100A1=dat(i,2);
        end
        if dat(i,4)>5
            fpm100x100A1=0;
            tah100x100A1=1;
        else
            fpm100x100A1=1;
            tah100x100A1=0;
        end
        if dat(i,4)==i+5
            fp100x100A1=0;
            ta100x100A1=1;
        else
            fp100x100A1=1;
            ta100x100A1=0;
        end
    else
        if dat(i,2)==0
            tmi100x100A1=[tmi100x100A1 0.0001];
        else
            tmi100x100A1=[tmi100x100A1 dat(i,2)];
        end
        if dat(i,4)>5
            fpm100x100A1=[fpm100x100A1 0];
            tah100x100A1=[tah100x100A1 1];
        else
            fpm100x100A1=[fpm100x100A1 1];
            tah100x100A1=[tah100x100A1 0];
        end
        if dat(i,4)==i+5
            fp100x100A1=[fp100x100A1 0];
            ta100x100A1=[ta100x100A1 1];
        else
            fp100x100A1=[fp100x100A1 1];
            ta100x100A1=[ta100x100A1 0];
        end
    end
```

```
end
nom_fichero='100x100/ResultadosMujerM2_05.m';
dat=leeFicheroPersona(nom_fichero,cadena);
[m,n]=size(dat);
for i=1:m
    if dat(i,2)==0
        tmi100x100A1=[tmi100x100A1 0.0001];
    else
        tmi100x100A1=[tmi100x100A1 dat(i,2)];
    end
    if dat(i,4)==i&&dat(i,4)<=5
        fp100x100A1=[fp100x100A1 0];
        ta100x100A1=[ta100x100A1 1];
    else
        fp100x100A1=[fp100x100A1 1];
        ta100x100A1=[ta100x100A1 0];
    end
    if i==1
        if dat(i,4)<=5
            fph100x100A1=0;
            tam100x100A1=1;
        else
            fph100x100A1=1;
            tam100x100A1=0;
        end
    else
        if dat(i,4)<=5
            fph100x100A1=[fph100x100A1 0];
            tam100x100A1=[tam100x100A1 1];
        else
            fph100x100A1=[fph100x100A1 1];
            tam100x100A1=[tam100x100A1 0];
        end
    end
end
nom_fichero='100x100/ResultadosHombreM2_22.m';
dat=leeFicheroPersona(nom_fichero,cadena);
[m,n]=size(dat);
for i=1:m

    if dat(i,2)==0
        tmi100x100S1=[tmi100x100S1 0.0001];
    else
        tmi100x100S1=[tmi100x100S1 dat(i,2)];
    end
    if dat(i,4)>25
        fpm100x100S1=[fpm100x100S1 0];
        tah100x100S1=[tah100x100S1 1];
    else
        fpm100x100S1=[fpm100x100S1 1];
        tah100x100S1=[tah100x100S1 0];
    end
    if dat(i,4)==i+25
        fp100x100S1=[fp100x100S1 0];
        ta100x100S1=[ta100x100S1 1];
    else
        fp100x100S1=[fp100x100S1 1];
        ta100x100S1=[ta100x100S1 0];
    end
end
nom_fichero='100x100/ResultadosMujerM2_22.m';
dat=leeFicheroPersona(nom_fichero,cadena);
[m,n]=size(dat);
for i=1:m
    if dat(i,2)==0
        tmi100x100S1=[tmi100x100S1 0.0001];
    else
        tmi100x100S1=[tmi100x100S1 dat(i,2)];
    end
    if dat(i,4)==i&&dat(i,4)<=25
        fp100x100S1=[fp100x100S1 0];
        ta100x100S1=[ta100x100S1 1];
    else
        fp100x100S1=[fp100x100S1 1];
        ta100x100S1=[ta100x100S1 0];
    end
    if dat(i,4)<=25
        fph100x100S1=[fph100x100S1 0];
```

```
        tam100x100S1=[tam100x100S1 1];
    else
        fph100x100S1=[fph100x100S1 1];
        tam100x100S1=[tam100x100S1 0];
    end
end
nom_fichero='100x100/ResultadosHombreM2_25.m';
dat=leeFicheroPersona(nom_fichero,cadena);
[m,n]=size(dat);
for i=1:m

    if dat(i,2)==0
        tmi100x100A1=[tmi100x100A1 0.0001];
    else
        tmi100x100A1=[tmi100x100A1 dat(i,2)];
    end
    if dat(i,4)>25
        fpm100x100A1=[fpm100x100A1 0];
        tah100x100A1=[tah100x100A1 1];
    else
        fpm100x100A1=[fpm100x100A1 1];
        tah100x100A1=[tah100x100A1 0];
    end
    if dat(i,4)==i+25
        fp100x100A1=[fp100x100A1 0];
        ta100x100A1=[ta100x100A1 1];
    else
        fp100x100A1=[fp100x100A1 1];
        ta100x100A1=[ta100x100A1 0];
    end
end
nom_fichero='100x100/ResultadosMujerM2_25.m';
dat=leeFicheroPersona(nom_fichero,cadena);
[m,n]=size(dat);
for i=1:m
    if dat(i,2)==0
        tmi100x100A1=[tmi100x100A1 0.0001];
    else
        tmi100x100A1=[tmi100x100A1 dat(i,2)];
    end
    if dat(i,4)==i&&dat(i,4)<=25
        fp100x100A1=[fp100x100A1 0];
        ta100x100A1=[ta100x100A1 1];
    else
        fp100x100A1=[fp100x100A1 1];
        ta100x100A1=[ta100x100A1 0];
    end
    if dat(i,4)<=25
        fph100x100A1=[fph100x100A1 0];
        tam100x100A1=[tam100x100A1 1];
    else
        fph100x100A1=[fph100x100A1 1];
        tam100x100A1=[tam100x100A1 0];
    end
end
nom_fichero='100x100/ResultadosHombreM2_42.m';
dat=leeFicheroPersona(nom_fichero,cadena);
[m,n]=size(dat);
for i=1:m

    if dat(i,2)==0
        tmi100x100S1=[tmi100x100S1 0.0001];
    else
        tmi100x100S1=[tmi100x100S1 dat(i,2)];
    end
    if dat(i,4)>50
        fpm100x100S1=[fpm100x100S1 0];
        tah100x100S1=[tah100x100S1 1];
    else
        fpm100x100S1=[fpm100x100S1 1];
        tah100x100S1=[tah100x100S1 0];
    end
    if dat(i,4)==i+50
        fp100x100S1=[fp100x100S1 0];
        ta100x100S1=[ta100x100S1 1];
    else
        fp100x100S1=[fp100x100S1 1];
```

```
        ta100x100S1=[ta100x100S1 0];
    end
end
nom_fichero='100x100/ResultadosMujerM2_42.m';
dat=leeFicheroPersona(nom_fichero,cadena);
[m,n]=size(dat);
for i=1:m
    if dat(i,2)==0
        tmi100x100S1=[tmi100x100S1 0.0001];
    else
        tmi100x100S1=[tmi100x100S1 dat(i,2)];
    end
    if dat(i,4)==i&&dat(i,4)<=50
        fp100x100S1=[fp100x100S1 0];
        ta100x100S1=[ta100x100S1 1];
    else
        fp100x100S1=[fp100x100S1 1];
        ta100x100S1=[ta100x100S1 0];
    end
    if dat(i,4)<=50
        fph100x100S1=[fph100x100S1 0];
        tam100x100S1=[tam100x100S1 1];
    else
        fph100x100S1=[fph100x100S1 1];
        tam100x100S1=[tam100x100S1 0];
    end
end
nom_fichero='100x100/ResultadosHombreM2_45.m';
dat=leeFicheroPersona(nom_fichero,cadena);
[m,n]=size(dat);
for i=1:m

    if dat(i,2)==0
        tmi100x100A1=[tmi100x100A1 0.0001];
    else
        tmi100x100A1=[tmi100x100A1 dat(i,2)];
    end
    if dat(i,4)>50
        fpm100x100A1=[fpm100x100A1 0];
        tah100x100A1=[tah100x100A1 1];
    else
        fpm100x100A1=[fpm100x100A1 1];
        tah100x100A1=[tah100x100A1 0];
    end
    if dat(i,4)==i+50
        fp100x100A1=[fp100x100A1 0];
        ta100x100A1=[ta100x100A1 1];
    else
        fp100x100A1=[fp100x100A1 1];
        ta100x100A1=[ta100x100A1 0];
    end
end
nom_fichero='100x100/ResultadosMujerM2_45.m';
dat=leeFicheroPersona(nom_fichero,cadena);
[m,n]=size(dat);
for i=1:m
    if dat(i,2)==0
        tmi100x100A1=[tmi100x100A1 0.0001];
    else
        tmi100x100A1=[tmi100x100A1 dat(i,2)];
    end
    if dat(i,4)==i&&dat(i,4)<=50
        fp100x100A1=[fp100x100A1 0];
        ta100x100A1=[ta100x100A1 1];
    else
        fp100x100A1=[fp100x100A1 1];
        ta100x100A1=[ta100x100A1 0];
    end
    if dat(i,4)<=50
        fph100x100A1=[fph100x100A1 0];
        tam100x100A1=[tam100x100A1 1];
    else
        fph100x100A1=[fph100x100A1 1];
        tam100x100A1=[tam100x100A1 0];
    end
end

nom_fichero='100x100/ResultadosHombreM3_02.m';
```

```
dat=leeFicheroPersona(nom_fichero,cadena);
[m,n]=size(dat);
for i=1:m

    if dat(i,2)==0
        tmi100x100S1=[tmi100x100S1 0.0001];
    else
        tmi100x100S1=[tmi100x100S1 dat(i,2)];
    end
    if dat(i,4)>5
        fpm100x100S1=[fpm100x100S1 0];
        tah100x100S1=[tah100x100S1 1];
    else
        fpm100x100S1=[fpm100x100S1 1];
        tah100x100S1=[tah100x100S1 0];
    end
    if dat(i,4)==i+5
        fp100x100S1=[fp100x100S1 0];
        ta100x100S1=[ta100x100S1 1];
    else
        fp100x100S1=[fp100x100S1 1];
        ta100x100S1=[ta100x100S1 0];
    end
end
nom_fichero='100x100/ResultadosMujerM3_02.m';
dat=leeFicheroPersona(nom_fichero,cadena);
[m,n]=size(dat);
for i=1:m
    if dat(i,2)==0
        tmi100x100S1=[tmi100x100S1 0.0001];
    else
        tmi100x100S1=[tmi100x100S1 dat(i,2)];
    end
    if dat(i,4)==i&&dat(i,4)<=5
        fp100x100S1=[fp100x100S1 0];
        ta100x100S1=[ta100x100S1 1];
    else
        fp100x100S1=[fp100x100S1 1];
        ta100x100S1=[ta100x100S1 0];
    end
    if dat(i,4)<=5
        fph100x100S1=[fph100x100S1 0];
        tam100x100S1=[tam100x100S1 1];
    else
        fph100x100S1=[fph100x100S1 1];
        tam100x100S1=[tam100x100S1 0];
    end
end
nom_fichero='100x100/ResultadosHombreM3_05.m';
dat=leeFicheroPersona(nom_fichero,cadena);
[m,n]=size(dat);
for i=1:m

    if dat(i,2)==0
        tmi100x100A1=[tmi100x100A1 0.0001];
    else
        tmi100x100A1=[tmi100x100A1 dat(i,2)];
    end
    if dat(i,4)>5
        fpm100x100A1=[fpm100x100A1 0];
        tah100x100A1=[tah100x100A1 1];
    else
        fpm100x100A1=[fpm100x100A1 1];
        tah100x100A1=[tah100x100A1 0];
    end
    if dat(i,4)==i+5
        fp100x100A1=[fp100x100A1 0];
        ta100x100A1=[ta100x100A1 1];
    else
        fp100x100A1=[fp100x100A1 1];
        ta100x100A1=[ta100x100A1 0];
    end
end
nom_fichero='100x100/ResultadosMujerM3_05.m';
dat=leeFicheroPersona(nom_fichero,cadena);
[m,n]=size(dat);
for i=1:m
```

```
    if dat(i,2)==0
        tmi100x100A1=[tmi100x100A1 0.0001];
    else
        tmi100x100A1=[tmi100x100A1 dat(i,2)];
    end
    if dat(i,4)==i&&dat(i,4)<=5
        fp100x100A1=[fp100x100A1 0];
        ta100x100A1=[ta100x100A1 1];
    else
        fp100x100A1=[fp100x100A1 1];
        ta100x100A1=[ta100x100A1 0];
    end
    if dat(i,4)<=5
        fph100x100A1=[fph100x100A1 0];
        tam100x100A1=[tam100x100A1 1];
    else
        fph100x100A1=[fph100x100A1 1];
        tam100x100A1=[tam100x100A1 0];
    end
end
nom_fichero='100x100/ResultadosHombreM3_12.m';
dat=leeFicheroPersona(nom_fichero,cadena);
[m,n]=size(dat);
for i=1:m
    if i==1
        if dat(i,2)==0
            tmi100x100S2=0.0001;
        else
            tmi100x100S2=dat(i,2);
        end
        if dat(i,4)>5
            fpm100x100S2=0;
            tah100x100S2=1;
        else
            fpm100x100S2=1;
            tah100x100S2=0;
        end
        if dat(i,4)==i+5
            fp100x100S2=0;
            ta100x100S2=1;
        else
            fp100x100S2=1;
            ta100x100S2=0;
        end
    else
        if dat(i,2)==0
            tmi100x100S2=[tmi100x100S2 0.0001];
        else
            tmi100x100S2=[tmi100x100S2 dat(i,2)];
        end
        if dat(i,4)>5
            fpm100x100S2=[fpm100x100S2 0];
            tah100x100S2=[tah100x100S2 1];
        else
            fpm100x100S2=[fpm100x100S2 1];
            tah100x100S2=[tah100x100S2 0];
        end
        if dat(i,4)==i+5
            fp100x100S2=[fp100x100S2 0];
            ta100x100S2=[ta100x100S2 1];
        else
            fp100x100S2=[fp100x100S2 1];
            ta100x100S2=[ta100x100S2 0];
        end
    end
end
nom_fichero='100x100/ResultadosMujerM3_12.m';
dat=leeFicheroPersona(nom_fichero,cadena);
[m,n]=size(dat);
for i=1:m
    if dat(i,2)==0
        tmi100x100S2=[tmi100x100S2 0.0001];
    else
        tmi100x100S2=[tmi100x100S2 dat(i,2)];
    end
    if dat(i,4)==i&&dat(i,4)<=5
        fp100x100S2=[fp100x100S2 0];
        ta100x100S2=[ta100x100S2 1];
```

```
    else
        fp100x100S2=[fp100x100S2 1];
        ta100x100S2=[ta100x100S2 0];
    end
    if i==1
        if dat(i,4)<=5
            fph100x100S2=0;
            tam100x100S2=1;
        else
            fph100x100S2=1;
            tam100x100S2=0;
        end
    else
        if dat(i,4)<=5
            fph100x100S2=[fph100x100S2 0];
            tam100x100S2=[tam100x100S2 1];
        else
            fph100x100S2=[fph100x100S2 1];
            tam100x100S2=[tam100x100S2 0];
        end
    end
end
nom_fichero='100x100/ResultadosHombreM3_15.m';
dat=leeFicheroPersona(nom_fichero,cadena);
[m,n]=size(dat);
for i=1:m
    if i==1
        if dat(i,2)==0
            tmi100x100A2=0.0001;
        else
            tmi100x100A2=dat(i,2);
        end
        if dat(i,4)>5
            fpm100x100A2=0;
            tah100x100A2=1;
        else
            fpm100x100A2=1;
            tah100x100A2=0;
        end
        if dat(i,4)==i+5
            fp100x100A2=0;
            ta100x100A2=1;
        else
            fp100x100A2=1;
            ta100x100A2=0;
        end
    else
        if dat(i,2)==0
            tmi100x100A2=[tmi100x100A2 0.0001];
        else
            tmi100x100A2=[tmi100x100A2 dat(i,2)];
        end
        if dat(i,4)>5
            fpm100x100A2=[fpm100x100A2 0];
            tah100x100A2=[tah100x100A2 1];
        else
            fpm100x100A2=[fpm100x100A2 1];
            tah100x100A2=[tah100x100A2 0];
        end
        if dat(i,4)==i+5
            fp100x100A2=[fp100x100A2 0];
            ta100x100A2=[ta100x100A2 1];
        else
            fp100x100A2=[fp100x100A2 1];
            ta100x100A2=[ta100x100A2 0];
        end
    end
end
nom_fichero='100x100/ResultadosMujerM3_15.m';
dat=leeFicheroPersona(nom_fichero,cadena);
[m,n]=size(dat);
for i=1:m
    if dat(i,2)==0
        tmi100x100A2=[tmi100x100A2 0.0001];
    else
        tmi100x100A2=[tmi100x100A2 dat(i,2)];
    end
```

```
    if dat(i,4)==i&&dat(i,4)<=5
        fp100x100A2=[fp100x100A2 0];
        ta100x100A2=[ta100x100A2 1];
    else
        fp100x100A2=[fp100x100A2 1];
        ta100x100A2=[ta100x100A2 0];
    end
    if i==1
        if dat(i,4)<=5
            fph100x100A2=0;
            tam100x100A2=1;
        else
            fph100x100A2=1;
            tam100x100A2=0;
        end
    else
        if dat(i,4)<=5
            fph100x100A2=[fph100x100A2 0];
            tam100x100A2=[tam100x100A2 1];
        else
            fph100x100A2=[fph100x100A2 1];
            tam100x100A2=[tam100x100A2 0];
        end
    end
end
nom_fichero='100x100/ResultadosHombreM3_22.m';
dat=leeFicheroPersona(nom_fichero,cadena);
[m,n]=size(dat);
for i=1:m

    if dat(i,2)==0
        tmi100x100S1=[tmi100x100S1 0.0001];
    else
        tmi100x100S1=[tmi100x100S1 dat(i,2)];
    end
    if dat(i,4)>25
        fpm100x100S1=[fpm100x100S1 0];
        tah100x100S1=[tah100x100S1 1];
    else
        fpm100x100S1=[fpm100x100S1 1];
        tah100x100S1=[tah100x100S1 0];
    end
    if dat(i,4)==i+25
        fp100x100S1=[fp100x100S1 0];
        ta100x100S1=[ta100x100S1 1];
    else
        fp100x100S1=[fp100x100S1 1];
        ta100x100S1=[ta100x100S1 0];
    end
end
nom_fichero='100x100/ResultadosMujerM3_22.m';
dat=leeFicheroPersona(nom_fichero,cadena);
[m,n]=size(dat);
for i=1:m
    if dat(i,2)==0
        tmi100x100S1=[tmi100x100S1 0.0001];
    else
        tmi100x100S1=[tmi100x100S1 dat(i,2)];
    end
    if dat(i,4)==i&&dat(i,4)<=25
        fp100x100S1=[fp100x100S1 0];
        ta100x100S1=[ta100x100S1 1];
    else
        fp100x100S1=[fp100x100S1 1];
        ta100x100S1=[ta100x100S1 0];
    end
    if dat(i,4)<=25
        fph100x100S1=[fph100x100S1 0];
        tam100x100S1=[tam100x100S1 1];
    else
        fph100x100S1=[fph100x100S1 1];
        tam100x100S1=[tam100x100S1 0];
    end
end
nom_fichero='100x100/ResultadosHombreM3_25.m';
dat=leeFicheroPersona(nom_fichero,cadena);
[m,n]=size(dat);
for i=1:m
```

```
    if dat(i,2)==0
        tmi100x100A1=[tmi100x100A1 0.0001];
    else
        tmi100x100A1=[tmi100x100A1 dat(i,2)];
    end
    if dat(i,4)>25
        fpm100x100A1=[fpm100x100A1 0];
        tah100x100A1=[tah100x100A1 1];
    else
        fpm100x100A1=[fpm100x100A1 1];
        tah100x100A1=[tah100x100A1 0];
    end
    if dat(i,4)==i+25
        fp100x100A1=[fp100x100A1 0];
        ta100x100A1=[ta100x100A1 1];
    else
        fp100x100A1=[fp100x100A1 1];
        ta100x100A1=[ta100x100A1 0];
    end
end
nom_fichero='100x100/ResultadosMujerM3_25.m';
dat=leeFicheroPersona(nom_fichero,cadena);
[m,n]=size(dat);
for i=1:m
    if dat(i,2)==0
        tmi100x100A1=[tmi100x100A1 0.0001];
    else
        tmi100x100A1=[tmi100x100A1 dat(i,2)];
    end
    if dat(i,4)==i&&dat(i,4)<=25
        fp100x100A1=[fp100x100A1 0];
        ta100x100A1=[ta100x100A1 1];
    else
        fp100x100A1=[fp100x100A1 1];
        ta100x100A1=[ta100x100A1 0];
    end
    if dat(i,4)<=25
        fph100x100A1=[fph100x100A1 0];
        tam100x100A1=[tam100x100A1 1];
    else
        fph100x100A1=[fph100x100A1 1];
        tam100x100A1=[tam100x100A1 0];
    end
end
nom_fichero='100x100/ResultadosHombreM3_32.m';
dat=leeFicheroPersona(nom_fichero,cadena);
[m,n]=size(dat);
for i=1:m

    if dat(i,2)==0
        tmi100x100S2=[tmi100x100S2 0.0001];
    else
        tmi100x100S2=[tmi100x100S2 dat(i,2)];
    end
    if dat(i,4)>25
        fpm100x100S2=[fpm100x100S2 0];
        tah100x100S2=[tah100x100S2 1];
    else
        fpm100x100S2=[fpm100x100S2 1];
        tah100x100S2=[tah100x100S2 0];
    end
    if dat(i,4)==i+25
        fp100x100S2=[fp100x100S2 0];
        ta100x100S2=[ta100x100S2 1];
    else
        fp100x100S2=[fp100x100S2 1];
        ta100x100S2=[ta100x100S2 0];
    end
end
nom_fichero='100x100/ResultadosMujerM3_32.m';
dat=leeFicheroPersona(nom_fichero,cadena);
[m,n]=size(dat);
for i=1:m
    if dat(i,2)==0
        tmi100x100S2=[tmi100x100S2 0.0001];
    else
```

```
        tmi100x100S2=[tmi100x100S2 dat(i,2)];
    end
    if dat(i,4)==i&&dat(i,4)<=25
        fp100x100S2=[fp100x100S2 0];
        ta100x100S2=[ta100x100S2 1];
    else
        fp100x100S2=[fp100x100S2 1];
        ta100x100S2=[ta100x100S2 0];
    end
    if dat(i,4)<=25
        fph100x100S2=[fph100x100S2 0];
        tam100x100S2=[tam100x100S2 1];
    else
        fph100x100S2=[fph100x100S2 1];
        tam100x100S2=[tam100x100S2 0];
    end
end
nom_fichero='100x100/ResultadosHombreM3_35.m';
dat=leeFicheroPersona(nom_fichero,cadena);
[m,n]=size(dat);
for i=1:m

    if dat(i,2)==0
        tmi100x100A2=[tmi100x100A2 0.0001];
    else
        tmi100x100A2=[tmi100x100A2 dat(i,2)];
    end
    if dat(i,4)>25
        fpm100x100A2=[fpm100x100A2 0];
        tah100x100A2=[tah100x100A2 1];
    else
        fpm100x100A2=[fpm100x100A2 1];
        tah100x100A2=[tah100x100A2 0];
    end
    if dat(i,4)==i+25
        fp100x100A2=[fp100x100A2 0];
        ta100x100A2=[ta100x100A2 1];
    else
        fp100x100A2=[fp100x100A2 1];
        ta100x100A2=[ta100x100A2 0];
    end
end
nom_fichero='100x100/ResultadosMujerM3_35.m';
dat=leeFicheroPersona(nom_fichero,cadena);
[m,n]=size(dat);
for i=1:m
    if dat(i,2)==0
        tmi100x100A2=[tmi100x100A2 0.0001];
    else
        tmi100x100A2=[tmi100x100A2 dat(i,2)];
    end
    if dat(i,4)==i&&dat(i,4)<=25
        fp100x100A2=[fp100x100A2 0];
        ta100x100A2=[ta100x100A2 1];
    else
        fp100x100A2=[fp100x100A2 1];
        ta100x100A2=[ta100x100A2 0];
    end
    if dat(i,4)<=25
        fph100x100A2=[fph100x100A2 0];
        tam100x100A2=[tam100x100A2 1];
    else
        fph100x100A2=[fph100x100A2 1];
        tam100x100A2=[tam100x100A2 0];
    end
end
nom_fichero='100x100/ResultadosHombreM3_42.m';
dat=leeFicheroPersona(nom_fichero,cadena);
[m,n]=size(dat);
for i=1:m

    if dat(i,2)==0
        tmi100x100S1=[tmi100x100S1 0.0001];
    else
        tmi100x100S1=[tmi100x100S1 dat(i,2)];
    end
    if dat(i,4)>50
        fpm100x100S1=[fpm100x100S1 0];
```

```
        tah100x100S1=[tah100x100S1 1];
    else
        fpm100x100S1=[fpm100x100S1 1];
        tah100x100S1=[tah100x100S1 0];
    end
    if dat(i,4)==i+50
        fp100x100S1=[fp100x100S1 0];
        ta100x100S1=[ta100x100S1 1];
    else
        fp100x100S1=[fp100x100S1 1];
        ta100x100S1=[ta100x100S1 0];
    end
end
nom_fichero='100x100/ResultadosMujerM3_42.m';
dat=leeFicheroPersona(nom_fichero,cadena);
[m,n]=size(dat);
for i=1:m
    if dat(i,2)==0
        tmi100x100S1=[tmi100x100S1 0.0001];
    else
        tmi100x100S1=[tmi100x100S1 dat(i,2)];
    end
    if dat(i,4)==i&&dat(i,4)<=50
        fp100x100S1=[fp100x100S1 0];
        ta100x100S1=[ta100x100S1 1];
    else
        fp100x100S1=[fp100x100S1 1];
        ta100x100S1=[ta100x100S1 0];
    end
    if dat(i,4)<=50
        fph100x100S1=[fph100x100S1 0];
        tam100x100S1=[tam100x100S1 1];
    else
        fph100x100S1=[fph100x100S1 1];
        tam100x100S1=[tam100x100S1 0];
    end
end
nom_fichero='100x100/ResultadosHombreM3_45.m';
dat=leeFicheroPersona(nom_fichero,cadena);
[m,n]=size(dat);
for i=1:m

    if dat(i,2)==0
        tmi100x100A1=[tmi100x100A1 0.0001];
    else
        tmi100x100A1=[tmi100x100A1 dat(i,2)];
    end
    if dat(i,4)>50
        fpm100x100A1=[fpm100x100A1 0];
        tah100x100A1=[tah100x100A1 1];
    else
        fpm100x100A1=[fpm100x100A1 1];
        tah100x100A1=[tah100x100A1 0];
    end
    if dat(i,4)==i+50
        fp100x100A1=[fp100x100A1 0];
        ta100x100A1=[ta100x100A1 1];
    else
        fp100x100A1=[fp100x100A1 1];
        ta100x100A1=[ta100x100A1 0];
    end
end
nom_fichero='100x100/ResultadosMujerM3_45.m';
dat=leeFicheroPersona(nom_fichero,cadena);
[m,n]=size(dat);
for i=1:m
    if dat(i,2)==0
        tmi100x100A1=[tmi100x100A1 0.0001];
    else
        tmi100x100A1=[tmi100x100A1 dat(i,2)];
    end
    if dat(i,4)==i&&dat(i,4)<=50
        fp100x100A1=[fp100x100A1 0];
        ta100x100A1=[ta100x100A1 1];
    else
        fp100x100A1=[fp100x100A1 1];
        ta100x100A1=[ta100x100A1 0];
```

```
    end
    if dat(i,4)<=50
        fph100x100A1=[fph100x100A1 0];
        tam100x100A1=[tam100x100A1 1];
    else
        fph100x100A1=[fph100x100A1 1];
        tam100x100A1=[tam100x100A1 0];
    end
end
nom_fichero='100x100/ResultadosHombreM3_52.m';
dat=leeFicheroPersona(nom_fichero,cadena);
[m,n]=size(dat);
for i=1:m

    if dat(i,2)==0
        tmi100x100S2=[tmi100x100S2 0.0001];
    else
        tmi100x100S2=[tmi100x100S2 dat(i,2)];
    end
    if dat(i,4)>50
        fpm100x100S2=[fpm100x100S2 0];
        tah100x100S2=[tah100x100S2 1];
    else
        fpm100x100S2=[fpm100x100S2 1];
        tah100x100S2=[tah100x100S2 0];
    end
    if dat(i,4)==i+50
        fp100x100S2=[fp100x100S2 0];
        ta100x100S2=[ta100x100S2 1];
    else
        fp100x100S2=[fp100x100S2 1];
        ta100x100S2=[ta100x100S2 0];
    end
end
nom_fichero='100x100/ResultadosMujerM3_52.m';
dat=leeFicheroPersona(nom_fichero,cadena);
[m,n]=size(dat);
for i=1:m
    if dat(i,2)==0
        tmi100x100S2=[tmi100x100S2 0.0001];
    else
        tmi100x100S2=[tmi100x100S2 dat(i,2)];
    end
    if dat(i,4)==i&&dat(i,4)<=50
        fp100x100S2=[fp100x100S2 0];
        ta100x100S2=[ta100x100S2 1];
    else
        fp100x100S2=[fp100x100S2 1];
        ta100x100S2=[ta100x100S2 0];
    end
    if dat(i,4)<=50
        fph100x100S2=[fph100x100S2 0];
        tam100x100S2=[tam100x100S2 1];
    else
        fph100x100S2=[fph100x100S2 1];
        tam100x100S2=[tam100x100S2 0];
    end
end
nom_fichero='100x100/ResultadosHombreM3_55.m';
dat=leeFicheroPersona(nom_fichero,cadena);
[m,n]=size(dat);
for i=1:m

    if dat(i,2)==0
        tmi100x100A2=[tmi100x100A2 0.0001];
    else
        tmi100x100A2=[tmi100x100A2 dat(i,2)];
    end
    if dat(i,4)>50
        fpm100x100A2=[fpm100x100A2 0];
        tah100x100A2=[tah100x100A2 1];
    else
        fpm100x100A2=[fpm100x100A2 1];
        tah100x100A2=[tah100x100A2 0];
    end
    if dat(i,4)==i+50
        fp100x100A2=[fp100x100A2 0];
        ta100x100A2=[ta100x100A2 1];
```

```
    else
        fp100x100A2=[fp100x100A2 1];
        ta100x100A2=[ta100x100A2 0];
    end
end
nom_fichero='100x100/ResultadosMujerM3_55.m';
dat=leeFicheroPersona(nom_fichero,cadena);
[m,n]=size(dat);
for i=1:m
    if dat(i,2)==0
        tmi100x100A2=[tmi100x100A2 0.0001];
    else
        tmi100x100A2=[tmi100x100A2 dat(i,2)];
    end
    if dat(i,4)==i&&dat(i,4)<=50
        fp100x100A2=[fp100x100A2 0];
        ta100x100A2=[ta100x100A2 1];
    else
        fp100x100A2=[fp100x100A2 1];
        ta100x100A2=[ta100x100A2 0];
    end
    if dat(i,4)<=50
        fph100x100A2=[fph100x100A2 0];
        tam100x100A2=[tam100x100A2 1];
    else
        fph100x100A2=[fph100x100A2 1];
        tam100x100A2=[tam100x100A2 0];
    end
end

% Generados todos los datos, se hace la clasificacion para 10 clases,
% 50 clases y 100 clases.
m=size(tmi100x100S1,2);
tmi100x100S12=reshape(tmi100x100S1,m/6,6);
tmi100x100S1=[(sum(tmi100x100S12(:,1))+sum(tmi100x100S12(:,4)))/(m/3)
(sum(tmi100x100S12(:,2))+sum(tmi100x100S12(:,5)))/(m/3)
(sum(tmi100x100S12(:,3))+sum(tmi100x100S12(:,6)))/(m/3)];
tmi100x100S12=0;

m=size(tmi100x100A1,2);
tmi100x100A12=reshape(tmi100x100A1,m/6,6);
tmi100x100A1=[(sum(tmi100x100A12(:,1))+sum(tmi100x100A12(:,4)))/(m/3)
(sum(tmi100x100A12(:,2))+sum(tmi100x100A12(:,5)))/(m/3)
(sum(tmi100x100A12(:,3))+sum(tmi100x100A12(:,6)))/(m/3)];
tmi100x100A12=0;

m=size(tmi100x100S2,2);
tmi100x100S22=reshape(tmi100x100S2,m/3,3);
tmi100x100S2=[sum(tmi100x100S22(:,1))/(m/3) sum(tmi100x100S22(:,2))/(m/3)
sum(tmi100x100S22(:,3))/(m/3)];
tmi100x100S22=0;

m=size(tmi100x100A2,2);
tmi100x100A22=reshape(tmi100x100A2,m/3,3);
tmi100x100A2=[sum(tmi100x100A22(:,1))/(m/3) sum(tmi100x100A22(:,2))/(m/3)
sum(tmi100x100A22(:,3))/(m/3)];
tmi100x100A22=0;


m=size(fpm100x100S1,2);
fpm100x100S12=reshape(fpm100x100S1,m/6,6);
fpm100x100S1=[(sum(fpm100x100S12(:,1))+sum(fpm100x100S12(:,4)))/(m/3)
(sum(fpm100x100S12(:,2))+sum(fpm100x100S12(:,5)))/(m/3)
(sum(fpm100x100S12(:,3))+sum(fpm100x100S12(:,6)))/(m/3)];
fpm100x100S12=0;

m=size(fpm100x100A1,2);
fpm100x100A12=reshape(fpm100x100A1,m/6,6);
fpm100x100A1=[(sum(fpm100x100A12(:,1))+sum(fpm100x100A12(:,4)))/(m/3)
(sum(fpm100x100A12(:,2))+sum(fpm100x100A12(:,5)))/(m/3)
(sum(fpm100x100A12(:,3))+sum(fpm100x100A12(:,6)))/(m/3)];
fpm100x100A12=0;

m=size(fpm100x100S2,2);
fpm100x100S22=reshape(fpm100x100S2,m/3,3);
fpm100x100S2=[sum(fpm100x100S22(:,1))/(m/3) sum(fpm100x100S22(:,2))/(m/3)
sum(fpm100x100S22(:,3))/(m/3)];
```

```
fpm100x100S22=0;

m=size(fpm100x100A2,2);
fpm100x100A22=reshape(fpm100x100A2,m/3,3);
fpm100x100A2=[sum(fpm100x100A22(:,1))/(m/3) sum(fpm100x100A22(:,2))/(m/3)
sum(fpm100x100A22(:,3))/(m/3)];
fpm100x100A22=0;


m=size(fph100x100S1,2);
fph100x100S12=reshape(fph100x100S1,m/6,6);
fph100x100S1=[(sum(fph100x100S12(:,1))+sum(fph100x100S12(:,4)))/(m/3)
(sum(fph100x100S12(:,2))+sum(fph100x100S12(:,5)))/(m/3)
(sum(fph100x100S12(:,3))+sum(fph100x100S12(:,6)))/(m/3)];
fph100x100S12=0;

m=size(fph100x100A1,2);
fph100x100A12=reshape(fph100x100A1,m/6,6);
fph100x100A1=[(sum(fph100x100A12(:,1))+sum(fph100x100A12(:,4)))/(m/3)
(sum(fph100x100A12(:,2))+sum(fph100x100A12(:,5)))/(m/3)
(sum(fph100x100A12(:,3))+sum(fph100x100A12(:,6)))/(m/3)];
fph100x100A12=0;

m=size(fph100x100S2,2);
fph100x100S22=reshape(fph100x100S2,m/3,3);
fph100x100S2=[sum(fph100x100S22(:,1))/(m/3) sum(fph100x100S22(:,2))/(m/3)
sum(fph100x100S22(:,3))/(m/3)];
fph100x100S22=0;

m=size(fph100x100A2,2);
fph100x100A22=reshape(fph100x100A2,m/3,3);
fph100x100A2=[sum(fph100x100A22(:,1))/(m/3) sum(fph100x100A22(:,2))/(m/3)
sum(fph100x100A22(:,3))/(m/3)];
fph100x100A22=0;


m=size(tah100x100S1,2);
tah100x100S12=reshape(tah100x100S1,m/6,6);
tah100x100S1=[(sum(tah100x100S12(:,1))+sum(tah100x100S12(:,4)))/(m/3)
(sum(tah100x100S12(:,2))+sum(tah100x100S12(:,5)))/(m/3)
(sum(tah100x100S12(:,3))+sum(tah100x100S12(:,6)))/(m/3)];
tah100x100S12=0;

m=size(tah100x100A1,2);
tah100x100A12=reshape(tah100x100A1,m/6,6);
tah100x100A1=[(sum(tah100x100A12(:,1))+sum(tah100x100A12(:,4)))/(m/3)
(sum(tah100x100A12(:,2))+sum(tah100x100A12(:,5)))/(m/3)
(sum(tah100x100A12(:,3))+sum(tah100x100A12(:,6)))/(m/3)];
tah100x100A12=0;

m=size(tah100x100S2,2);
tah100x100S22=reshape(tah100x100S2,m/3,3);
tah100x100S2=[sum(tah100x100S22(:,1))/(m/3) sum(tah100x100S22(:,2))/(m/3)
sum(tah100x100S22(:,3))/(m/3)];
tah100x100S22=0;

m=size(tah100x100A2,2);
tah100x100A22=reshape(tah100x100A2,m/3,3);
tah100x100A2=[sum(tah100x100A22(:,1))/(m/3) sum(tah100x100A22(:,2))/(m/3)
sum(tah100x100A22(:,3))/(m/3)];
tah100x100A22=0;


m=size(tam100x100S1,2);
tam100x100S12=reshape(tam100x100S1,m/6,6);
tam100x100S1=[(sum(tam100x100S12(:,1))+sum(tam100x100S12(:,4)))/(m/3)
(sum(tam100x100S12(:,2))+sum(tam100x100S12(:,5)))/(m/3)
(sum(tam100x100S12(:,3))+sum(tam100x100S12(:,6)))/(m/3)];
tam100x100S12=0;

m=size(tam100x100A1,2);
tam100x100A12=reshape(tam100x100A1,m/6,6);
tam100x100A1=[(sum(tam100x100A12(:,1))+sum(tam100x100A12(:,4)))/(m/3)
(sum(tam100x100A12(:,2))+sum(tam100x100A12(:,5)))/(m/3)
(sum(tam100x100A12(:,3))+sum(tam100x100A12(:,6)))/(m/3)];
tam100x100A12=0;

m=size(tam100x100S2,2);
```

```
tam100x100S22=reshape(tam100x100S2,m/3,3);
tam100x100S2=[sum(tam100x100S22(:,1))/(m/3) sum(tam100x100S22(:,2))/(m/3)
sum(tam100x100S22(:,3))/(m/3)];
tam100x100S22=0;

m=size(tam100x100A2,2);
tam100x100A22=reshape(tam100x100A2,m/3,3);
tam100x100A2=[sum(tam100x100A22(:,1))/(m/3) sum(tam100x100A22(:,2))/(m/3)
sum(tam100x100A22(:,3))/(m/3)];
tam100x100A22=0;


m=size(fp100x100S1,2);
fp100x100S12=reshape(fp100x100S1,m/6,6);
fp100x100S1=[(sum(fp100x100S12(:,1))+sum(fp100x100S12(:,4)))/(m/3)
(sum(fp100x100S12(:,2))+sum(fp100x100S12(:,5)))/(m/3)
(sum(fp100x100S12(:,3))+sum(fp100x100S12(:,6)))/(m/3)];
fp100x100S12=0;

m=size(fp100x100A1,2);
fp100x100A12=reshape(fp100x100A1,m/6,6);
fp100x100A1=[(sum(fp100x100A12(:,1))+sum(fp100x100A12(:,4)))/(m/3)
(sum(fp100x100A12(:,2))+sum(fp100x100A12(:,5)))/(m/3)
(sum(fp100x100A12(:,3))+sum(fp100x100A12(:,6)))/(m/3)];
fp100x100A12=0;

m=size(fp100x100S2,2);
fp100x100S22=reshape(fp100x100S2,m/3,3);
fp100x100S2=[sum(fp100x100S22(:,1))/(m/3) sum(fp100x100S22(:,2))/(m/3)
sum(fp100x100S22(:,3))/(m/3)];
fp100x100S22=0;

fp100x100A2;
m=size(fp100x100A2,2);
fp100x100A22=reshape(fp100x100A2,m/3,3);
fp100x100A2=[sum(fp100x100A22(:,1))/(m/3) sum(fp100x100A22(:,2))/(m/3)
sum(fp100x100A22(:,3))/(m/3)];
fp100x100A22=0;


m=size(ta100x100S1,2);
ta100x100S12=reshape(ta100x100S1,m/6,6);
ta100x100S1=[(sum(ta100x100S12(:,1))+sum(ta100x100S12(:,4)))/20
(sum(ta100x100S12(:,2))+sum(ta100x100S12(:,5)))/100
(sum(ta100x100S12(:,3))+sum(ta100x100S12(:,6)))/200];
ta100x100S12=0;

m=size(ta100x100A1,2);
ta100x100A12=reshape(ta100x100A1,m/6,6);
ta100x100A1=[(sum(ta100x100A12(:,1))+sum(ta100x100A12(:,4)))/20
(sum(ta100x100A12(:,2))+sum(ta100x100A12(:,5)))/100
(sum(ta100x100A12(:,3))+sum(ta100x100A12(:,6)))/200];
ta100x100A12=0;

m=size(ta100x100S2,2);
ta100x100S22=reshape(ta100x100S2,m/3,3);
ta100x100S2=[sum(ta100x100S22(:,1))/10 sum(ta100x100S22(:,2))/50 sum(ta100x100S22(:,3))/100];
ta100x100S22=0;

ta100x100A2;
m=size(ta100x100A2,2);
ta100x100A22=reshape(ta100x100A2,m/3,3);
ta100x100A2=[sum(ta100x100A22(:,1))/10 sum(ta100x100A22(:,2))/50 sum(ta100x100A22(:,3))/100];
ta100x100A22=0;
```

Fichero analizaFicheroPersona150x150.m:

```
%
% Fichero: analizaFicheroPersona150x150.m
% Autor: Maria Sierra Zapata
% Fecha: 01/07/2015
% Version: 0.1
%
% Breve Descripcion:
%       Apertura de los ficheros generados debido a la
%       identificacion de personas para un modelo y con tamaño de imagen
```

```
%       150x150 pixeles. Seleccion de los datos para la generacion de las
%       graficas para el estudio.
%
% Datos que se obtienen:
%       Tiempo Medio de Identificacion
%       Falsos Positivos Mujer
%       Falsos Positivos Hombre
%       Falsos Positivos
%       Tasa de Acierto Hombre
%       Tasa de Acierto Mujer
%       Tasa de Acierto
%

cadena='%d=%f %f %d';
num_images=[10 50 100];

% Apertura y lectura del fichero
nom_fichero='150x150/ResultadosHombreM2_02.m';
dat=leeFicheroPersona(nom_fichero,cadena);
% Obtencion de los datos del fichero
[m,n]=size(dat);
for i=1:m
    if i==1
        if dat(i,2)==0
            tmi150x150S1=0.0001;
        else
            tmi150x150S1=dat(i,2);
        end
        if dat(i,4)>5
            fpm150x150S1=0;
            tah150x150S1=1;
        else
            fpm150x150S1=1;
            tah150x150S1=0;
        end
        if dat(i,4)==i+5
            fp150x150S1=0;
            ta150x150S1=1;
        else
            fp150x150S1=1;
            ta150x150S1=0;
        end
    else
        if dat(i,2)==0
            tmi150x150S1=[tmi150x150S1 0.0001];
        else
            tmi150x150S1=[tmi150x150S1 dat(i,2)];
        end
        if dat(i,4)>5
            fpm150x150S1=[fpm150x150S1 0];
            tah150x150S1=[tah150x150S1 1];
        else
            fpm150x150S1=[fpm150x150S1 1];
            tah150x150S1=[tah150x150S1 0];
        end
        if dat(i,4)==i+5
            fp150x150S1=[fp150x150S1 0];
            ta150x150S1=[ta150x150S1 1];
        else
            fp150x150S1=[fp150x150S1 1];
            ta150x150S1=[ta150x150S1 0];
        end
    end
end
nom_fichero='150x150/ResultadosMujerM2_02.m';
dat=leeFicheroPersona(nom_fichero,cadena);
[m,n]=size(dat);
for i=1:m
    if dat(i,2)==0
        tmi150x150S1=[tmi150x150S1 0.0001];
    else
        tmi150x150S1=[tmi150x150S1 dat(i,2)];
    end
    if dat(i,4)==i&&dat(i,4)<=5
        fp150x150S1=[fp150x150S1 0];
        ta150x150S1=[ta150x150S1 1];
    else
        fp150x150S1=[fp150x150S1 1];
```

```
        ta150x150S1=[ta150x150S1 0];
    end
    if i==1
        if dat(i,4)<=5
            fph150x150S1=0;
            tam150x150S1=1;
        else
            fph150x150S1=1;
            tam150x150S1=0;
        end
    else
        if dat(i,4)<=5
            fph150x150S1=[fph150x150S1 0];
            tam150x150S1=[tam150x150S1 1];
        else
            fph150x150S1=[fph150x150S1 1];
            tam150x150S1=[tam150x150S1 0];
        end
    end
end
nom_fichero='150x150/ResultadosHombreM2_05.m';
dat=leeFicheroPersona(nom_fichero,cadena);
[m,n]=size(dat);
for i=1:m
    if i==1
        if dat(i,2)==0
            tmi150x150A1=0.0001;
        else
            tmi150x150A1=dat(i,2);
        end
        if dat(i,4)>5
            fpm150x150A1=0;
            tah150x150A1=1;
        else
            fpm150x150A1=1;
            tah150x150A1=0;
        end
        if dat(i,4)==i+5
            fp150x150A1=0;
            ta150x150A1=1;
        else
            fp150x150A1=1;
            ta150x150A1=0;
        end
    else
        if dat(i,2)==0
            tmi150x150A1=[tmi150x150A1 0.0001];
        else
            tmi150x150A1=[tmi150x150A1 dat(i,2)];
        end
        if dat(i,4)>5
            fpm150x150A1=[fpm150x150A1 0];
            tah150x150A1=[tah150x150A1 1];
        else
            fpm150x150A1=[fpm150x150A1 1];
            tah150x150A1=[tah150x150A1 0];
        end
        if dat(i,4)==i+5
            fp150x150A1=[fp150x150A1 0];
            ta150x150A1=[ta150x150A1 1];
        else
            fp150x150A1=[fp150x150A1 1];
            ta150x150A1=[ta150x150A1 0];
        end
    end
end
nom_fichero='150x150/ResultadosMujerM2_05.m';
dat=leeFicheroPersona(nom_fichero,cadena);
[m,n]=size(dat);
for i=1:m
    if dat(i,2)==0
        tmi150x150A1=[tmi150x150A1 0.0001];
    else
        tmi150x150A1=[tmi150x150A1 dat(i,2)];
    end
    if dat(i,4)==i&&dat(i,4)<=5
        fp150x150A1=[fp150x150A1 0];
```

```
        ta150x150A1=[ta150x150A1 1];
    else
        fp150x150A1=[fp150x150A1 1];
        ta150x150A1=[ta150x150A1 0];
    end
    if i==1
        if dat(i,4)<=5
            fph150x150A1=0;
            tam150x150A1=1;
        else
            fph150x150A1=1;
            tam150x150A1=0;
        end
    else
        if dat(i,4)<=5
            fph150x150A1=[fph150x150A1 0];
            tam150x150A1=[tam150x150A1 1];
        else
            fph150x150A1=[fph150x150A1 1];
            tam150x150A1=[tam150x150A1 0];
        end
    end
end
nom_fichero='150x150/ResultadosHombreM2_22.m';
dat=leeFicheroPersona(nom_fichero,cadena);
[m,n]=size(dat);
for i=1:m

    if dat(i,2)==0
        tmi150x150S1=[tmi150x150S1 0.0001];
    else
        tmi150x150S1=[tmi150x150S1 dat(i,2)];
    end
    if dat(i,4)>25
        fpm150x150S1=[fpm150x150S1 0];
        tah150x150S1=[tah150x150S1 1];
    else
        fpm150x150S1=[fpm150x150S1 1];
        tah150x150S1=[tah150x150S1 0];
    end
    if dat(i,4)==i+25
        fp150x150S1=[fp150x150S1 0];
        ta150x150S1=[ta150x150S1 1];
    else
        fp150x150S1=[fp150x150S1 1];
        ta150x150S1=[ta150x150S1 0];
    end
end
nom_fichero='150x150/ResultadosMujerM2_22.m';
dat=leeFicheroPersona(nom_fichero,cadena);
[m,n]=size(dat);
for i=1:m
    if dat(i,2)==0
        tmi150x150S1=[tmi150x150S1 0.0001];
    else
        tmi150x150S1=[tmi150x150S1 dat(i,2)];
    end
    if dat(i,4)==i&&dat(i,4)<=25
        fp150x150S1=[fp150x150S1 0];
        ta150x150S1=[ta150x150S1 1];
    else
        fp150x150S1=[fp150x150S1 1];
        ta150x150S1=[ta150x150S1 0];
    end
    if dat(i,4)<=25
        fph150x150S1=[fph150x150S1 0];
        tam150x150S1=[tam150x150S1 1];
    else
        fph150x150S1=[fph150x150S1 1];
        tam150x150S1=[tam150x150S1 0];
    end
end
nom_fichero='150x150/ResultadosHombreM2_25.m';
dat=leeFicheroPersona(nom_fichero,cadena);
[m,n]=size(dat);
for i=1:m

    if dat(i,2)==0
```

```
        tmi150x150A1=[tmi150x150A1 0.0001];
    else
        tmi150x150A1=[tmi150x150A1 dat(i,2)];
    end
    if dat(i,4)>25
        fpm150x150A1=[fpm150x150A1 0];
        tah150x150A1=[tah150x150A1 1];
    else
        fpm150x150A1=[fpm150x150A1 1];
        tah150x150A1=[tah150x150A1 0];
    end
    if dat(i,4)==i+25
        fp150x150A1=[fp150x150A1 0];
        ta150x150A1=[ta150x150A1 1];
    else
        fp150x150A1=[fp150x150A1 1];
        ta150x150A1=[ta150x150A1 0];
    end
end
nom_fichero='150x150/ResultadosMujerM2_25.m';
dat=leeFicheroPersona(nom_fichero,cadena);
[m,n]=size(dat);
for i=1:m
    if dat(i,2)==0
        tmi150x150A1=[tmi150x150A1 0.0001];
    else
        tmi150x150A1=[tmi150x150A1 dat(i,2)];
    end
    if dat(i,4)==i&&dat(i,4)<=25
        fp150x150A1=[fp150x150A1 0];
        ta150x150A1=[ta150x150A1 1];
    else
        fp150x150A1=[fp150x150A1 1];
        ta150x150A1=[ta150x150A1 0];
    end
    if dat(i,4)<=25
        fph150x150A1=[fph150x150A1 0];
        tam150x150A1=[tam150x150A1 1];
    else
        fph150x150A1=[fph150x150A1 1];
        tam150x150A1=[tam150x150A1 0];
    end
end
nom_fichero='150x150/ResultadosHombreM2_42.m';
dat=leeFicheroPersona(nom_fichero,cadena);
[m,n]=size(dat);
for i=1:m

    if dat(i,2)==0
        tmi150x150S1=[tmi150x150S1 0.0001];
    else
        tmi150x150S1=[tmi150x150S1 dat(i,2)];
    end
    if dat(i,4)>50
        fpm150x150S1=[fpm150x150S1 0];
        tah150x150S1=[tah150x150S1 1];
    else
        fpm150x150S1=[fpm150x150S1 1];
        tah150x150S1=[tah150x150S1 0];
    end
    if dat(i,4)==i+50
        fp150x150S1=[fp150x150S1 0];
        ta150x150S1=[ta150x150S1 1];
    else
        fp150x150S1=[fp150x150S1 1];
        ta150x150S1=[ta150x150S1 0];
    end
end
nom_fichero='150x150/ResultadosMujerM2_42.m';
dat=leeFicheroPersona(nom_fichero,cadena);
[m,n]=size(dat);
for i=1:m
    if dat(i,2)==0
        tmi150x150S1=[tmi150x150S1 0.0001];
    else
        tmi150x150S1=[tmi150x150S1 dat(i,2)];
    end
```

```
    if dat(i,4)==i&&dat(i,4)<=50
        fp150x150S1=[fp150x150S1 0];
        ta150x150S1=[ta150x150S1 1];
    else
        fp150x150S1=[fp150x150S1 1];
        ta150x150S1=[ta150x150S1 0];
    end
    if dat(i,4)<=50
        fph150x150S1=[fph150x150S1 0];
        tam150x150S1=[tam150x150S1 1];
    else
        fph150x150S1=[fph150x150S1 1];
        tam150x150S1=[tam150x150S1 0];
    end
end
nom_fichero='150x150/ResultadosHombreM2_45.m';
dat=leeFicheroPersona(nom_fichero,cadena);
[m,n]=size(dat);
for i=1:m

    if dat(i,2)==0
        tmi150x150A1=[tmi150x150A1 0.0001];
    else
        tmi150x150A1=[tmi150x150A1 dat(i,2)];
    end
    if dat(i,4)>50
        fpm150x150A1=[fpm150x150A1 0];
        tah150x150A1=[tah150x150A1 1];
    else
        fpm150x150A1=[fpm150x150A1 1];
        tah150x150A1=[tah150x150A1 0];
    end
    if dat(i,4)==i+50
        fp150x150A1=[fp150x150A1 0];
        ta150x150A1=[ta150x150A1 1];
    else
        fp150x150A1=[fp150x150A1 1];
        ta150x150A1=[ta150x150A1 0];
    end
end
nom_fichero='150x150/ResultadosMujerM2_45.m';
dat=leeFicheroPersona(nom_fichero,cadena);
[m,n]=size(dat);
for i=1:m
    if dat(i,2)==0
        tmi150x150A1=[tmi150x150A1 0.0001];
    else
        tmi150x150A1=[tmi150x150A1 dat(i,2)];
    end
    if dat(i,4)==i&&dat(i,4)<=50
        fp150x150A1=[fp150x150A1 0];
        ta150x150A1=[ta150x150A1 1];
    else
        fp150x150A1=[fp150x150A1 1];
        ta150x150A1=[ta150x150A1 0];
    end
    if dat(i,4)<=50
        fph150x150A1=[fph150x150A1 0];
        tam150x150A1=[tam150x150A1 1];
    else
        fph150x150A1=[fph150x150A1 1];
        tam150x150A1=[tam150x150A1 0];
    end
end

nom_fichero='150x150/ResultadosHombreM3_02.m';
dat=leeFicheroPersona(nom_fichero,cadena);
[m,n]=size(dat);
for i=1:m

    if dat(i,2)==0
        tmi150x150S1=[tmi150x150S1 0.0001];
    else
        tmi150x150S1=[tmi150x150S1 dat(i,2)];
    end
    if dat(i,4)>5
        fpm150x150S1=[fpm150x150S1 0];
        tah150x150S1=[tah150x150S1 1];
```

```
    else
        fpm150x150S1=[fpm150x150S1 1];
        tah150x150S1=[tah150x150S1 0];
    end
    if dat(i,4)==i+5
        fp150x150S1=[fp150x150S1 0];
        ta150x150S1=[ta150x150S1 1];
    else
        fp150x150S1=[fp150x150S1 1];
        ta150x150S1=[ta150x150S1 0];
    end
end
nom_fichero='150x150/ResultadosMujerM3_02.m';
dat=leeFicheroPersona(nom_fichero,cadena);
[m,n]=size(dat);
for i=1:m
    if dat(i,2)==0
        tmi150x150S1=[tmi150x150S1 0.0001];
    else
        tmi150x150S1=[tmi150x150S1 dat(i,2)];
    end
    if dat(i,4)==i&&dat(i,4)<=5
        fp150x150S1=[fp150x150S1 0];
        ta150x150S1=[ta150x150S1 1];
    else
        fp150x150S1=[fp150x150S1 1];
        ta150x150S1=[ta150x150S1 0];
    end
    if dat(i,4)<=5
        fph150x150S1=[fph150x150S1 0];
        tam150x150S1=[tam150x150S1 1];
    else
        fph150x150S1=[fph150x150S1 1];
        tam150x150S1=[tam150x150S1 0];
    end
end
nom_fichero='150x150/ResultadosHombreM3_05.m';
dat=leeFicheroPersona(nom_fichero,cadena);
[m,n]=size(dat);
for i=1:m

    if dat(i,2)==0
        tmi150x150A1=[tmi150x150A1 0.0001];
    else
        tmi150x150A1=[tmi150x150A1 dat(i,2)];
    end
    if dat(i,4)>5
        fpm150x150A1=[fpm150x150A1 0];
        tah150x150A1=[tah150x150A1 1];
    else
        fpm150x150A1=[fpm150x150A1 1];
        tah150x150A1=[tah150x150A1 0];
    end
    if dat(i,4)==i+5
        fp150x150A1=[fp150x150A1 0];
        ta150x150A1=[ta150x150A1 1];
    else
        fp150x150A1=[fp150x150A1 1];
        ta150x150A1=[ta150x150A1 0];
    end
end
nom_fichero='150x150/ResultadosMujerM3_05.m';
dat=leeFicheroPersona(nom_fichero,cadena);
[m,n]=size(dat);
for i=1:m
    if dat(i,2)==0
        tmi150x150A1=[tmi150x150A1 0.0001];
    else
        tmi150x150A1=[tmi150x150A1 dat(i,2)];
    end
    if dat(i,4)==i&&dat(i,4)<=5
        fp150x150A1=[fp150x150A1 0];
        ta150x150A1=[ta150x150A1 1];
    else
        fp150x150A1=[fp150x150A1 1];
        ta150x150A1=[ta150x150A1 0];
    end
```

```
    if dat(i,4)<=5
        fph150x150A1=[fph150x150A1 0];
        tam150x150A1=[tam150x150A1 1];
    else
        fph150x150A1=[fph150x150A1 1];
        tam150x150A1=[tam150x150A1 0];
    end
end
nom_fichero='150x150/ResultadosHombreM3_12.m';
dat=leeFicheroPersona(nom_fichero,cadena);
[m,n]=size(dat);
for i=1:m
    if i==1
        if dat(i,2)==0
            tmi150x150S2=0.0001;
        else
            tmi150x150S2=dat(i,2);
        end
        if dat(i,4)>5
            fpm150x150S2=0;
            tah150x150S2=1;
        else
            fpm150x150S2=1;
            tah150x150S2=0;
        end
        if dat(i,4)==i+5
            fp150x150S2=0;
            ta150x150S2=1;
        else
            fp150x150S2=1;
            ta150x150S2=0;
        end
    else
        if dat(i,2)==0
            tmi150x150S2=[tmi150x150S2 0.0001];
        else
            tmi150x150S2=[tmi150x150S2 dat(i,2)];
        end
        if dat(i,4)>5
            fpm150x150S2=[fpm150x150S2 0];
            tah150x150S2=[tah150x150S2 1];
        else
            fpm150x150S2=[fpm150x150S2 1];
            tah150x150S2=[tah150x150S2 0];
        end
        if dat(i,4)==i+5
            fp150x150S2=[fp150x150S2 0];
            ta150x150S2=[ta150x150S2 1];
        else
            fp150x150S2=[fp150x150S2 1];
            ta150x150S2=[ta150x150S2 0];
        end
    end
end
nom_fichero='150x150/ResultadosMujerM3_12.m';
dat=leeFicheroPersona(nom_fichero,cadena);
[m,n]=size(dat);
for i=1:m
    if dat(i,2)==0
        tmi150x150S2=[tmi150x150S2 0.0001];
    else
        tmi150x150S2=[tmi150x150S2 dat(i,2)];
    end
    if dat(i,4)==i&&dat(i,4)<=5
        fp150x150S2=[fp150x150S2 0];
        ta150x150S2=[ta150x150S2 1];
    else
        fp150x150S2=[fp150x150S2 1];
        ta150x150S2=[ta150x150S2 0];
    end
    if i==1
        if dat(i,4)<=5
            fph150x150S2=0;
            tam150x150S2=1;
        else
            fph150x150S2=1;
            tam150x150S2=0;
        end
```

```
    else
        if dat(i,4)<=5
            fph150x150S2=[fph150x150S2 0];
            tam150x150S2=[tam150x150S2 1];
        else
            fph150x150S2=[fph150x150S2 1];
            tam150x150S2=[tam150x150S2 0];
        end
    end
end
nom_fichero='150x150/ResultadosHombreM3_15.m';
dat=leeFicheroPersona(nom_fichero,cadena);
[m,n]=size(dat);
for i=1:m
    if i==1
        if dat(i,2)==0
            tmi150x150A2=0.0001;
        else
            tmi150x150A2=dat(i,2);
        end
        if dat(i,4)>5
            fpm150x150A2=0;
            tah150x150A2=1;
        else
            fpm150x150A2=1;
            tah150x150A2=0;
        end
        if dat(i,4)==i+5
            fp150x150A2=0;
            ta150x150A2=1;
        else
            fp150x150A2=1;
            ta150x150A2=0;
        end
    else
        if dat(i,2)==0
            tmi150x150A2=[tmi150x150A2 0.0001];
        else
            tmi150x150A2=[tmi150x150A2 dat(i,2)];
        end
        if dat(i,4)>5
            fpm150x150A2=[fpm150x150A2 0];
            tah150x150A2=[tah150x150A2 1];
        else
            fpm150x150A2=[fpm150x150A2 1];
            tah150x150A2=[tah150x150A2 0];
        end
        if dat(i,4)==i+5
            fp150x150A2=[fp150x150A2 0];
            ta150x150A2=[ta150x150A2 1];
        else
            fp150x150A2=[fp150x150A2 1];
            ta150x150A2=[ta150x150A2 0];
        end
    end
end
nom_fichero='150x150/ResultadosMujerM3_15.m';
dat=leeFicheroPersona(nom_fichero,cadena);
[m,n]=size(dat);
for i=1:m
    if dat(i,2)==0
        tmi150x150A2=[tmi150x150A2 0.0001];
    else
        tmi150x150A2=[tmi150x150A2 dat(i,2)];
    end
    if dat(i,4)==i&&dat(i,4)<=5
        fp150x150A2=[fp150x150A2 0];
        ta150x150A2=[ta150x150A2 1];
    else
        fp150x150A2=[fp150x150A2 1];
        ta150x150A2=[ta150x150A2 0];
    end
    if i==1
        if dat(i,4)<=5
            fph150x150A2=0;
            tam150x150A2=1;
        else
```

```
            fph150x150A2=1;
            tam150x150A2=0;
        end
    else
        if dat(i,4)<=5
            fph150x150A2=[fph150x150A2 0];
            tam150x150A2=[tam150x150A2 1];
        else
            fph150x150A2=[fph150x150A2 1];
            tam150x150A2=[tam150x150A2 0];
        end
    end
end
nom_fichero='150x150/ResultadosHombreM3_22.m';
dat=leeFicheroPersona(nom_fichero,cadena);
[m,n]=size(dat);
for i=1:m

    if dat(i,2)==0
        tmi150x150S1=[tmi150x150S1 0.0001];
    else
        tmi150x150S1=[tmi150x150S1 dat(i,2)];
    end
    if dat(i,4)>25
        fpm150x150S1=[fpm150x150S1 0];
        tah150x150S1=[tah150x150S1 1];
    else
        fpm150x150S1=[fpm150x150S1 1];
        tah150x150S1=[tah150x150S1 0];
    end
    if dat(i,4)==i+25
        fp150x150S1=[fp150x150S1 0];
        ta150x150S1=[ta150x150S1 1];
    else
        fp150x150S1=[fp150x150S1 1];
        ta150x150S1=[ta150x150S1 0];
    end
end
nom_fichero='150x150/ResultadosMujerM3_22.m';
dat=leeFicheroPersona(nom_fichero,cadena);
[m,n]=size(dat);
for i=1:m
    if dat(i,2)==0
        tmi150x150S1=[tmi150x150S1 0.0001];
    else
        tmi150x150S1=[tmi150x150S1 dat(i,2)];
    end
    if dat(i,4)==i&&dat(i,4)<=25
        fp150x150S1=[fp150x150S1 0];
        ta150x150S1=[ta150x150S1 1];
    else
        fp150x150S1=[fp150x150S1 1];
        ta150x150S1=[ta150x150S1 0];
    end
    if dat(i,4)<=25
        fph150x150S1=[fph150x150S1 0];
        tam150x150S1=[tam150x150S1 1];
    else
        fph150x150S1=[fph150x150S1 1];
        tam150x150S1=[tam150x150S1 0];
    end
end
nom_fichero='150x150/ResultadosHombreM3_25.m';
dat=leeFicheroPersona(nom_fichero,cadena);
[m,n]=size(dat);
for i=1:m

    if dat(i,2)==0
        tmi150x150A1=[tmi150x150A1 0.0001];
    else
        tmi150x150A1=[tmi150x150A1 dat(i,2)];
    end
    if dat(i,4)>25
        fpm150x150A1=[fpm150x150A1 0];
        tah150x150A1=[tah150x150A1 1];
    else
        fpm150x150A1=[fpm150x150A1 1];
        tah150x150A1=[tah150x150A1 0];
```

```
    end
    if dat(i,4)==i+25
        fp150x150A1=[fp150x150A1 0];
        ta150x150A1=[ta150x150A1 1];
    else
        fp150x150A1=[fp150x150A1 1];
        ta150x150A1=[ta150x150A1 0];
    end
end
nom_fichero='150x150/ResultadosMujerM3_25.m';
dat=leeFicheroPersona(nom_fichero,cadena);
[m,n]=size(dat);
for i=1:m
    if dat(i,2)==0
        tmi150x150A1=[tmi150x150A1 0.0001];
    else
        tmi150x150A1=[tmi150x150A1 dat(i,2)];
    end
    if dat(i,4)==i&&dat(i,4)<=25
        fp150x150A1=[fp150x150A1 0];
        ta150x150A1=[ta150x150A1 1];
    else
        fp150x150A1=[fp150x150A1 1];
        ta150x150A1=[ta150x150A1 0];
    end
    if dat(i,4)<=25
        fph150x150A1=[fph150x150A1 0];
        tam150x150A1=[tam150x150A1 1];
    else
        fph150x150A1=[fph150x150A1 1];
        tam150x150A1=[tam150x150A1 0];
    end
end
nom_fichero='150x150/ResultadosHombreM3_32.m';
dat=leeFicheroPersona(nom_fichero,cadena);
[m,n]=size(dat);
for i=1:m

    if dat(i,2)==0
        tmi150x150S2=[tmi150x150S2 0.0001];
    else
        tmi150x150S2=[tmi150x150S2 dat(i,2)];
    end
    if dat(i,4)>25
        fpm150x150S2=[fpm150x150S2 0];
        tah150x150S2=[tah150x150S2 1];
    else
        fpm150x150S2=[fpm150x150S2 1];
        tah150x150S2=[tah150x150S2 0];
    end
    if dat(i,4)==i+25
        fp150x150S2=[fp150x150S2 0];
        ta150x150S2=[ta150x150S2 1];
    else
        fp150x150S2=[fp150x150S2 1];
        ta150x150S2=[ta150x150S2 0];
    end
end
nom_fichero='150x150/ResultadosMujerM3_32.m';
dat=leeFicheroPersona(nom_fichero,cadena);
[m,n]=size(dat);
for i=1:m
    if dat(i,2)==0
        tmi150x150S2=[tmi150x150S2 0.0001];
    else
        tmi150x150S2=[tmi150x150S2 dat(i,2)];
    end
    if dat(i,4)==i&&dat(i,4)<=25
        fp150x150S2=[fp150x150S2 0];
        ta150x150S2=[ta150x150S2 1];
    else
        fp150x150S2=[fp150x150S2 1];
        ta150x150S2=[ta150x150S2 0];
    end
    if dat(i,4)<=25
        fph150x150S2=[fph150x150S2 0];
        tam150x150S2=[tam150x150S2 1];
```

```
    else
        fph150x150S2=[fph150x150S2 1];
        tam150x150S2=[tam150x150S2 0];
    end
end
nom_fichero='150x150/ResultadosHombreM3_35.m';
dat=leeFicheroPersona(nom_fichero,cadena);
[m,n]=size(dat);
for i=1:m

    if dat(i,2)==0
        tmi150x150A2=[tmi150x150A2 0.0001];
    else
        tmi150x150A2=[tmi150x150A2 dat(i,2)];
    end
    if dat(i,4)>25
        fpm150x150A2=[fpm150x150A2 0];
        tah150x150A2=[tah150x150A2 1];
    else
        fpm150x150A2=[fpm150x150A2 1];
        tah150x150A2=[tah150x150A2 0];
    end
    if dat(i,4)==i+25
        fp150x150A2=[fp150x150A2 0];
        ta150x150A2=[ta150x150A2 1];
    else
        fp150x150A2=[fp150x150A2 1];
        ta150x150A2=[ta150x150A2 0];
    end
end
nom_fichero='150x150/ResultadosMujerM3_35.m';
dat=leeFicheroPersona(nom_fichero,cadena);
[m,n]=size(dat);
for i=1:m
    if dat(i,2)==0
        tmi150x150A2=[tmi150x150A2 0.0001];
    else
        tmi150x150A2=[tmi150x150A2 dat(i,2)];
    end
    if dat(i,4)==i&&dat(i,4)<=25
        fp150x150A2=[fp150x150A2 0];
        ta150x150A2=[ta150x150A2 1];
    else
        fp150x150A2=[fp150x150A2 1];
        ta150x150A2=[ta150x150A2 0];
    end
    if dat(i,4)<=25
        fph150x150A2=[fph150x150A2 0];
        tam150x150A2=[tam150x150A2 1];
    else
        fph150x150A2=[fph150x150A2 1];
        tam150x150A2=[tam150x150A2 0];
    end
end
nom_fichero='150x150/ResultadosHombreM3_42.m';
dat=leeFicheroPersona(nom_fichero,cadena);
[m,n]=size(dat);
for i=1:m

    if dat(i,2)==0
        tmi150x150S1=[tmi150x150S1 0.0001];
    else
        tmi150x150S1=[tmi150x150S1 dat(i,2)];
    end
    if dat(i,4)>50
        fpm150x150S1=[fpm150x150S1 0];
        tah150x150S1=[tah150x150S1 1];
    else
        fpm150x150S1=[fpm150x150S1 1];
        tah150x150S1=[tah150x150S1 0];
    end
    if dat(i,4)==i+50
        fp150x150S1=[fp150x150S1 0];
        ta150x150S1=[ta150x150S1 1];
    else
        fp150x150S1=[fp150x150S1 1];
        ta150x150S1=[ta150x150S1 0];
    end
```

```
end
nom_fichero='150x150/ResultadosMujerM3_42.m';
dat=leeFicheroPersona(nom_fichero,cadena);
[m,n]=size(dat);
for i=1:m
    if dat(i,2)==0
        tmi150x150S1=[tmi150x150S1 0.0001];
    else
        tmi150x150S1=[tmi150x150S1 dat(i,2)];
    end
    if dat(i,4)==i&&dat(i,4)<=50
        fp150x150S1=[fp150x150S1 0];
        ta150x150S1=[ta150x150S1 1];
    else
        fp150x150S1=[fp150x150S1 1];
        ta150x150S1=[ta150x150S1 0];
    end
    if dat(i,4)<=50
        fph150x150S1=[fph150x150S1 0];
        tam150x150S1=[tam150x150S1 1];
    else
        fph150x150S1=[fph150x150S1 1];
        tam150x150S1=[tam150x150S1 0];
    end
end
nom_fichero='150x150/ResultadosHombreM3_45.m';
dat=leeFicheroPersona(nom_fichero,cadena);
[m,n]=size(dat);
for i=1:m

    if dat(i,2)==0
        tmi150x150A1=[tmi150x150A1 0.0001];
    else
        tmi150x150A1=[tmi150x150A1 dat(i,2)];
    end
    if dat(i,4)>50
        fpm150x150A1=[fpm150x150A1 0];
        tah150x150A1=[tah150x150A1 1];
    else
        fpm150x150A1=[fpm150x150A1 1];
        tah150x150A1=[tah150x150A1 0];
    end
    if dat(i,4)==i+50
        fp150x150A1=[fp150x150A1 0];
        ta150x150A1=[ta150x150A1 1];
    else
        fp150x150A1=[fp150x150A1 1];
        ta150x150A1=[ta150x150A1 0];
    end
end
nom_fichero='150x150/ResultadosMujerM3_45.m';
dat=leeFicheroPersona(nom_fichero,cadena);
[m,n]=size(dat);
for i=1:m
    if dat(i,2)==0
        tmi150x150A1=[tmi150x150A1 0.0001];
    else
        tmi150x150A1=[tmi150x150A1 dat(i,2)];
    end
    if dat(i,4)==i&&dat(i,4)<=50
        fp150x150A1=[fp150x150A1 0];
        ta150x150A1=[ta150x150A1 1];
    else
        fp150x150A1=[fp150x150A1 1];
        ta150x150A1=[ta150x150A1 0];
    end
    if dat(i,4)<=50
        fph150x150A1=[fph150x150A1 0];
        tam150x150A1=[tam150x150A1 1];
    else
        fph150x150A1=[fph150x150A1 1];
        tam150x150A1=[tam150x150A1 0];
    end
end
nom_fichero='150x150/ResultadosHombreM3_52.m';
dat=leeFicheroPersona(nom_fichero,cadena);
[m,n]=size(dat);
```

```
for i=1:m

    if dat(i,2)==0
        tmi150x150S2=[tmi150x150S2 0.0001];
    else
        tmi150x150S2=[tmi150x150S2 dat(i,2)];
    end
    if dat(i,4)>50
        fpm150x150S2=[fpm150x150S2 0];
        tah150x150S2=[tah150x150S2 1];
    else
        fpm150x150S2=[fpm150x150S2 1];
        tah150x150S2=[tah150x150S2 0];
    end
    if dat(i,4)==i+50
        fp150x150S2=[fp150x150S2 0];
        ta150x150S2=[ta150x150S2 1];
    else
        fp150x150S2=[fp150x150S2 1];
        ta150x150S2=[ta150x150S2 0];
    end
end
nom_fichero='150x150/ResultadosMujerM3_52.m';
dat=leeFicheroPersona(nom_fichero,cadena);
[m,n]=size(dat);
for i=1:m
    if dat(i,2)==0
        tmi150x150S2=[tmi150x150S2 0.0001];
    else
        tmi150x150S2=[tmi150x150S2 dat(i,2)];
    end
    if dat(i,4)==i&&dat(i,4)<=50
        fp150x150S2=[fp150x150S2 0];
        ta150x150S2=[ta150x150S2 1];
    else
        fp150x150S2=[fp150x150S2 1];
        ta150x150S2=[ta150x150S2 0];
    end
    if dat(i,4)<=50
        fph150x150S2=[fph150x150S2 0];
        tam150x150S2=[tam150x150S2 1];
    else
        fph150x150S2=[fph150x150S2 1];
        tam150x150S2=[tam150x150S2 0];
    end
end
nom_fichero='150x150/ResultadosHombreM3_55.m';
dat=leeFicheroPersona(nom_fichero,cadena);
[m,n]=size(dat);
for i=1:m

    if dat(i,2)==0
        tmi150x150A2=[tmi150x150A2 0.0001];
    else
        tmi150x150A2=[tmi150x150A2 dat(i,2)];
    end
    if dat(i,4)>50
        fpm150x150A2=[fpm150x150A2 0];
        tah150x150A2=[tah150x150A2 1];
    else
        fpm150x150A2=[fpm150x150A2 1];
        tah150x150A2=[tah150x150A2 0];
    end
    if dat(i,4)==i+50
        fp150x150A2=[fp150x150A2 0];
        ta150x150A2=[ta150x150A2 1];
    else
        fp150x150A2=[fp150x150A2 1];
        ta150x150A2=[ta150x150A2 0];
    end
end
nom_fichero='150x150/ResultadosMujerM3_55.m';
dat=leeFicheroPersona(nom_fichero,cadena);
[m,n]=size(dat);
for i=1:m
    if dat(i,2)==0
        tmi150x150A2=[tmi150x150A2 0.0001];
    else
```

```
        tmi150x150A2=[tmi150x150A2 dat(i,2)];
    end
    if dat(i,4)==i&&dat(i,4)<=50
        fp150x150A2=[fp150x150A2 0];
        ta150x150A2=[ta150x150A2 1];
    else
        fp150x150A2=[fp150x150A2 1];
        ta150x150A2=[ta150x150A2 0];
    end
    if dat(i,4)<=50
        fph150x150A2=[fph150x150A2 0];
        tam150x150A2=[tam150x150A2 1];
    else
        fph150x150A2=[fph150x150A2 1];
        tam150x150A2=[tam150x150A2 0];
    end
end

% Generados todos los datos, se hace la clasificacion para 10 clases,
% 50 clases y 100 clases.
m=size(tmi150x150S1,2);
tmi150x150S12=reshape(tmi150x150S1,m/6,6);
tmi150x150S1=[(sum(tmi150x150S12(:,1))+sum(tmi150x150S12(:,4)))/(m/3)
(sum(tmi150x150S12(:,2))+sum(tmi150x150S12(:,5)))/(m/3)
(sum(tmi150x150S12(:,3))+sum(tmi150x150S12(:,6)))/(m/3)];
tmi150x150S12=0;

m=size(tmi150x150A1,2);
tmi150x150A12=reshape(tmi150x150A1,m/6,6);
tmi150x150A1=[(sum(tmi150x150A12(:,1))+sum(tmi150x150A12(:,4)))/(m/3)
(sum(tmi150x150A12(:,2))+sum(tmi150x150A12(:,5)))/(m/3)
(sum(tmi150x150A12(:,3))+sum(tmi150x150A12(:,6)))/(m/3)];
tmi150x150A12=0;

m=size(tmi150x150S2,2);
tmi150x150S22=reshape(tmi150x150S2,m/3,3);
tmi150x150S2=[sum(tmi150x150S22(:,1))/(m/3) sum(tmi150x150S22(:,2))/(m/3)
sum(tmi150x150S22(:,3))/(m/3)];
tmi150x150S22=0;

m=size(tmi150x150A2,2);
tmi150x150A22=reshape(tmi150x150A2,m/3,3);
tmi150x150A2=[sum(tmi150x150A22(:,1))/(m/3) sum(tmi150x150A22(:,2))/(m/3)
sum(tmi150x150A22(:,3))/(m/3)];
tmi150x150A22=0;


m=size(fpm150x150S1,2);
fpm150x150S12=reshape(fpm150x150S1,m/6,6);
fpm150x150S1=[(sum(fpm150x150S12(:,1))+sum(fpm150x150S12(:,4)))/(m/3)
(sum(fpm150x150S12(:,2))+sum(fpm150x150S12(:,5)))/(m/3)
(sum(fpm150x150S12(:,3))+sum(fpm150x150S12(:,6)))/(m/3)];
fpm150x150S12=0;

m=size(fpm150x150A1,2);
fpm150x150A12=reshape(fpm150x150A1,m/6,6);
fpm150x150A1=[(sum(fpm150x150A12(:,1))+sum(fpm150x150A12(:,4)))/(m/3)
(sum(fpm150x150A12(:,2))+sum(fpm150x150A12(:,5)))/(m/3)
(sum(fpm150x150A12(:,3))+sum(fpm150x150A12(:,6)))/(m/3)];
fpm150x150A12=0;

m=size(fpm150x150S2,2);
fpm150x150S22=reshape(fpm150x150S2,m/3,3);
fpm150x150S2=[sum(fpm150x150S22(:,1))/(m/3) sum(fpm150x150S22(:,2))/(m/3)
sum(fpm150x150S22(:,3))/(m/3)];
fpm150x150S22=0;

m=size(fpm150x150A2,2);
fpm150x150A22=reshape(fpm150x150A2,m/3,3);
fpm150x150A2=[sum(fpm150x150A22(:,1))/(m/3) sum(fpm150x150A22(:,2))/(m/3)
sum(fpm150x150A22(:,3))/(m/3)];
fpm150x150A22=0;


m=size(fph150x150S1,2);
fph150x150S12=reshape(fph150x150S1,m/6,6);
```

```
fph150x150S1=[(sum(fph150x150S12(:,1))+sum(fph150x150S12(:,4)))/(m/3)
(sum(fph150x150S12(:,2))+sum(fph150x150S12(:,5)))/(m/3)
(sum(fph150x150S12(:,3))+sum(fph150x150S12(:,6)))/(m/3)];
fph150x150S12=0;

m=size(fph150x150A1,2);
fph150x150A12=reshape(fph150x150A1,m/6,6);
fph150x150A1=[(sum(fph150x150A12(:,1))+sum(fph150x150A12(:,4)))/(m/3)
(sum(fph150x150A12(:,2))+sum(fph150x150A12(:,5)))/(m/3)
(sum(fph150x150A12(:,3))+sum(fph150x150A12(:,6)))/(m/3)];
fph150x150A12=0;

m=size(fph150x150S2,2);
fph150x150S22=reshape(fph150x150S2,m/3,3);
fph150x150S2=[sum(fph150x150S22(:,1))/(m/3) sum(fph150x150S22(:,2))/(m/3)
sum(fph150x150S22(:,3))/(m/3)];
fph150x150S22=0;

m=size(fph150x150A2,2);
fph150x150A22=reshape(fph150x150A2,m/3,3);
fph150x150A2=[sum(fph150x150A22(:,1))/(m/3) sum(fph150x150A22(:,2))/(m/3)
sum(fph150x150A22(:,3))/(m/3)];
fph150x150A22=0;


m=size(tah150x150S1,2);
tah150x150S12=reshape(tah150x150S1,m/6,6);
tah150x150S1=[(sum(tah150x150S12(:,1))+sum(tah150x150S12(:,4)))/(m/3)
(sum(tah150x150S12(:,2))+sum(tah150x150S12(:,5)))/(m/3)
(sum(tah150x150S12(:,3))+sum(tah150x150S12(:,6)))/(m/3)];
tah150x150S12=0;

m=size(tah150x150A1,2);
tah150x150A12=reshape(tah150x150A1,m/6,6);
tah150x150A1=[(sum(tah150x150A12(:,1))+sum(tah150x150A12(:,4)))/(m/3)
(sum(tah150x150A12(:,2))+sum(tah150x150A12(:,5)))/(m/3)
(sum(tah150x150A12(:,3))+sum(tah150x150A12(:,6)))/(m/3)];
tah150x150A12=0;

m=size(tah150x150S2,2);
tah150x150S22=reshape(tah150x150S2,m/3,3);
tah150x150S2=[sum(tah150x150S22(:,1))/(m/3) sum(tah150x150S22(:,2))/(m/3)
sum(tah150x150S22(:,3))/(m/3)];
tah150x150S22=0;

m=size(tah150x150A2,2);
tah150x150A22=reshape(tah150x150A2,m/3,3);
tah150x150A2=[sum(tah150x150A22(:,1))/(m/3) sum(tah150x150A22(:,2))/(m/3)
sum(tah150x150A22(:,3))/(m/3)];
tah150x150A22=0;


m=size(tam150x150S1,2);
tam150x150S12=reshape(tam150x150S1,m/6,6);
tam150x150S1=[(sum(tam150x150S12(:,1))+sum(tam150x150S12(:,4)))/(m/3)
(sum(tam150x150S12(:,2))+sum(tam150x150S12(:,5)))/(m/3)
(sum(tam150x150S12(:,3))+sum(tam150x150S12(:,6)))/(m/3)];
tam150x150S12=0;

m=size(tam150x150A1,2);
tam150x150A12=reshape(tam150x150A1,m/6,6);
tam150x150A1=[(sum(tam150x150A12(:,1))+sum(tam150x150A12(:,4)))/(m/3)
(sum(tam150x150A12(:,2))+sum(tam150x150A12(:,5)))/(m/3)
(sum(tam150x150A12(:,3))+sum(tam150x150A12(:,6)))/(m/3)];
tam150x150A12=0;

m=size(tam150x150S2,2);
tam150x150S22=reshape(tam150x150S2,m/3,3);
tam150x150S2=[sum(tam150x150S22(:,1))/(m/3) sum(tam150x150S22(:,2))/(m/3)
sum(tam150x150S22(:,3))/(m/3)];
tam150x150S22=0;

m=size(tam150x150A2,2);
tam150x150A22=reshape(tam150x150A2,m/3,3);
tam150x150A2=[sum(tam150x150A22(:,1))/(m/3) sum(tam150x150A22(:,2))/(m/3)
sum(tam150x150A22(:,3))/(m/3)];
tam150x150A22=0;
```

```
m=size(fp150x150S1,2);
fp150x150S12=reshape(fp150x150S1,m/6,6);
fp150x150S1=[(sum(fp150x150S12(:,1))+sum(fp150x150S12(:,4)))/(m/3)
(sum(fp150x150S12(:,2))+sum(fp150x150S12(:,5)))/(m/3)
(sum(fp150x150S12(:,3))+sum(fp150x150S12(:,6)))/(m/3)];
fp150x150S12=0;

m=size(fp150x150A1,2);
fp150x150A12=reshape(fp150x150A1,m/6,6);
fp150x150A1=[(sum(fp150x150A12(:,1))+sum(fp150x150A12(:,4)))/(m/3)
(sum(fp150x150A12(:,2))+sum(fp150x150A12(:,5)))/(m/3)
(sum(fp150x150A12(:,3))+sum(fp150x150A12(:,6)))/(m/3)];
fp150x150A12=0;

m=size(fp150x150S2,2);
fp150x150S22=reshape(fp150x150S2,m/3,3);
fp150x150S2=[sum(fp150x150S22(:,1))/(m/3) sum(fp150x150S22(:,2))/(m/3)
sum(fp150x150S22(:,3))/(m/3)];
fp150x150S22=0;

fp150x150A2;
m=size(fp150x150A2,2);
fp150x150A22=reshape(fp150x150A2,m/3,3);
fp150x150A2=[sum(fp150x150A22(:,1))/(m/3) sum(fp150x150A22(:,2))/(m/3)
sum(fp150x150A22(:,3))/(m/3)];
fp150x150A22=0;


m=size(ta150x150S1,2);
ta150x150S12=reshape(ta150x150S1,m/6,6);
ta150x150S1=[(sum(ta150x150S12(:,1))+sum(ta150x150S12(:,4)))/20
(sum(ta150x150S12(:,2))+sum(ta150x150S12(:,5)))/100
(sum(ta150x150S12(:,3))+sum(ta150x150S12(:,6)))/200];
ta150x150S12=0;

m=size(ta150x150A1,2);
ta150x150A12=reshape(ta150x150A1,m/6,6);
ta150x150A1=[(sum(ta150x150A12(:,1))+sum(ta150x150A12(:,4)))/20
(sum(ta150x150A12(:,2))+sum(ta150x150A12(:,5)))/100
(sum(ta150x150A12(:,3))+sum(ta150x150A12(:,6)))/200];
ta150x150A12=0;

m=size(ta150x150S2,2);
ta150x150S22=reshape(ta150x150S2,m/3,3);
ta150x150S2=[sum(ta150x150S22(:,1))/10 sum(ta150x150S22(:,2))/50 sum(ta150x150S22(:,3))/100];
ta150x150S22=0;

ta150x150A2;
m=size(ta150x150A2,2);
ta150x150A22=reshape(ta150x150A2,m/3,3);
ta150x150A2=[sum(ta150x150A22(:,1))/10 sum(ta150x150A22(:,2))/50 sum(ta150x150A22(:,3))/100];
ta150x150A22=0;
```

Fichero analizaFicheroPersona200x200.m:

```
%
% Fichero: analizaFicheroPersona200x200.m
% Autor: Maria Sierra Zapata
% Fecha: 01/07/2015
% Version: 0.1
%
% Breve Descripcion:
%       Apertura de los ficheros generados debido a la
%       identificacion de personas para un modelo y con tamaño de imagen
%       200x200 pixeles. Seleccion de los datos para la generacion de las
%       graficas para el estudio.
%
% Datos que se obtienen:
%       Tiempo Medio de Identificacion
%       Falsos Positivos Mujer
%       Falsos Positivos Hombre
%       Falsos Positivos
%       Tasa de Acierto Hombre
%       Tasa de Acierto Mujer
%       Tasa de Acierto
```

```
%

cadena='%d=%f %f %d';
num_images=[10 50 100];

% Apertura y lectura del fichero
nom_fichero='200x200/ResultadosHombreM2_02.m';
dat=leeFicheroPersona(nom_fichero,cadena);
% Obtencion de los datos del fichero
[m,n]=size(dat);
for i=1:m
    if i==1
        if dat(i,2)==0
            tmi200x200S1=0.0001;
        else
            tmi200x200S1=dat(i,2);
        end
        if dat(i,4)>5
            fpm200x200S1=0;
            tah200x200S1=1;
        else
            fpm200x200S1=1;
            tah200x200S1=0;
        end
        if dat(i,4)==i+5
            fp200x200S1=0;
            ta200x200S1=1;
        else
            fp200x200S1=1;
            ta200x200S1=0;
        end
    else
        if dat(i,2)==0
            tmi200x200S1=[tmi200x200S1 0.0001];
        else
            tmi200x200S1=[tmi200x200S1 dat(i,2)];
        end
        if dat(i,4)>5
            fpm200x200S1=[fpm200x200S1 0];
            tah200x200S1=[tah200x200S1 1];
        else
            fpm200x200S1=[fpm200x200S1 1];
            tah200x200S1=[tah200x200S1 0];
        end
        if dat(i,4)==i+5
            fp200x200S1=[fp200x200S1 0];
            ta200x200S1=[ta200x200S1 1];
        else
            fp200x200S1=[fp200x200S1 1];
            ta200x200S1=[ta200x200S1 0];
        end
    end
end
nom_fichero='200x200/ResultadosMujerM2_02.m';
dat=leeFicheroPersona(nom_fichero,cadena);
[m,n]=size(dat);
for i=1:m
    if dat(i,2)==0
        tmi200x200S1=[tmi200x200S1 0.0001];
    else
        tmi200x200S1=[tmi200x200S1 dat(i,2)];
    end
    if dat(i,4)==i&&dat(i,4)<=5
        fp200x200S1=[fp200x200S1 0];
        ta200x200S1=[ta200x200S1 1];
    else
        fp200x200S1=[fp200x200S1 1];
        ta200x200S1=[ta200x200S1 0];
    end
    if i==1
        if dat(i,4)<=5
            fph200x200S1=0;
            tam200x200S1=1;
        else
            fph200x200S1=1;
            tam200x200S1=0;
        end
    else
```

```
        if dat(i,4)<=5
            fph200x200S1=[fph200x200S1 0];
            tam200x200S1=[tam200x200S1 1];
        else
            fph200x200S1=[fph200x200S1 1];
            tam200x200S1=[tam200x200S1 0];
        end
    end
end
nom_fichero='200x200/ResultadosHombreM2_05.m';
dat=leeFicheroPersona(nom_fichero,cadena);
[m,n]=size(dat);
for i=1:m
    if i==1
        if dat(i,2)==0
            tmi200x200A1=0.0001;
        else
            tmi200x200A1=dat(i,2);
        end
        if dat(i,4)>5
            fpm200x200A1=0;
            tah200x200A1=1;
        else
            fpm200x200A1=1;
            tah200x200A1=0;
        end
        if dat(i,4)==i+5
            fp200x200A1=0;
            ta200x200A1=1;
        else
            fp200x200A1=1;
            ta200x200A1=0;
        end
    else
        if dat(i,2)==0
            tmi200x200A1=[tmi200x200A1 0.0001];
        else
            tmi200x200A1=[tmi200x200A1 dat(i,2)];
        end
        if dat(i,4)>5
            fpm200x200A1=[fpm200x200A1 0];
            tah200x200A1=[tah200x200A1 1];
        else
            fpm200x200A1=[fpm200x200A1 1];
            tah200x200A1=[tah200x200A1 0];
        end
        if dat(i,4)==i+5
            fp200x200A1=[fp200x200A1 0];
            ta200x200A1=[ta200x200A1 1];
        else
            fp200x200A1=[fp200x200A1 1];
            ta200x200A1=[ta200x200A1 0];
        end
    end
end
nom_fichero='200x200/ResultadosMujerM2_05.m';
dat=leeFicheroPersona(nom_fichero,cadena);
[m,n]=size(dat);
for i=1:m
    if dat(i,2)==0
        tmi200x200A1=[tmi200x200A1 0.0001];
    else
        tmi200x200A1=[tmi200x200A1 dat(i,2)];
    end
    if dat(i,4)==i&&dat(i,4)<=5
        fp200x200A1=[fp200x200A1 0];
        ta200x200A1=[ta200x200A1 1];
    else
        fp200x200A1=[fp200x200A1 1];
        ta200x200A1=[ta200x200A1 0];
    end
    if i==1
        if dat(i,4)<=5
            fph200x200A1=0;
            tam200x200A1=1;
        else
            fph200x200A1=1;
```

```
            tam200x200A1=0;
        end
    else
        if dat(i,4)<=5
            fph200x200A1=[fph200x200A1 0];
            tam200x200A1=[tam200x200A1 1];
        else
            fph200x200A1=[fph200x200A1 1];
            tam200x200A1=[tam200x200A1 0];
        end
    end
end
nom_fichero='200x200/ResultadosHombreM2_22.m';
dat=leeFicheroPersona(nom_fichero,cadena);
[m,n]=size(dat);
for i=1:m

    if dat(i,2)==0
        tmi200x200S1=[tmi200x200S1 0.0001];
    else
        tmi200x200S1=[tmi200x200S1 dat(i,2)];
    end
    if dat(i,4)>25
        fpm200x200S1=[fpm200x200S1 0];
        tah200x200S1=[tah200x200S1 1];
    else
        fpm200x200S1=[fpm200x200S1 1];
        tah200x200S1=[tah200x200S1 0];
    end
    if dat(i,4)==i+25
        fp200x200S1=[fp200x200S1 0];
        ta200x200S1=[ta200x200S1 1];
    else
        fp200x200S1=[fp200x200S1 1];
        ta200x200S1=[ta200x200S1 0];
    end
end
nom_fichero='200x200/ResultadosMujerM2_22.m';
dat=leeFicheroPersona(nom_fichero,cadena);
[m,n]=size(dat);
for i=1:m
    if dat(i,2)==0
        tmi200x200S1=[tmi200x200S1 0.0001];
    else
        tmi200x200S1=[tmi200x200S1 dat(i,2)];
    end
    if dat(i,4)==i&&dat(i,4)<=25
        fp200x200S1=[fp200x200S1 0];
        ta200x200S1=[ta200x200S1 1];
    else
        fp200x200S1=[fp200x200S1 1];
        ta200x200S1=[ta200x200S1 0];
    end
    if dat(i,4)<=25
        fph200x200S1=[fph200x200S1 0];
        tam200x200S1=[tam200x200S1 1];
    else
        fph200x200S1=[fph200x200S1 1];
        tam200x200S1=[tam200x200S1 0];
    end
end
nom_fichero='200x200/ResultadosHombreM2_25.m';
dat=leeFicheroPersona(nom_fichero,cadena);
[m,n]=size(dat);
for i=1:m

    if dat(i,2)==0
        tmi200x200A1=[tmi200x200A1 0.0001];
    else
        tmi200x200A1=[tmi200x200A1 dat(i,2)];
    end
    if dat(i,4)>25
        fpm200x200A1=[fpm200x200A1 0];
        tah200x200A1=[tah200x200A1 1];
    else
        fpm200x200A1=[fpm200x200A1 1];
        tah200x200A1=[tah200x200A1 0];
    end
```

```
    if dat(i,4)==i+25
        fp200x200A1=[fp200x200A1 0];
        ta200x200A1=[ta200x200A1 1];
    else
        fp200x200A1=[fp200x200A1 1];
        ta200x200A1=[ta200x200A1 0];
    end
end
nom_fichero='200x200/ResultadosMujerM2_25.m';
dat=leeFicheroPersona(nom_fichero,cadena);
[m,n]=size(dat);
for i=1:m
    if dat(i,2)==0
        tmi200x200A1=[tmi200x200A1 0.0001];
    else
        tmi200x200A1=[tmi200x200A1 dat(i,2)];
    end
    if dat(i,4)==i&&dat(i,4)<=25
        fp200x200A1=[fp200x200A1 0];
        ta200x200A1=[ta200x200A1 1];
    else
        fp200x200A1=[fp200x200A1 1];
        ta200x200A1=[ta200x200A1 0];
    end
    if dat(i,4)<=25
        fph200x200A1=[fph200x200A1 0];
        tam200x200A1=[tam200x200A1 1];
    else
        fph200x200A1=[fph200x200A1 1];
        tam200x200A1=[tam200x200A1 0];
    end
end
nom_fichero='200x200/ResultadosHombreM2_42.m';
dat=leeFicheroPersona(nom_fichero,cadena);
[m,n]=size(dat);
for i=1:m

    if dat(i,2)==0
        tmi200x200S1=[tmi200x200S1 0.0001];
    else
        tmi200x200S1=[tmi200x200S1 dat(i,2)];
    end
    if dat(i,4)>50
        fpm200x200S1=[fpm200x200S1 0];
        tah200x200S1=[tah200x200S1 1];
    else
        fpm200x200S1=[fpm200x200S1 1];
        tah200x200S1=[tah200x200S1 0];
    end
    if dat(i,4)==i+50
        fp200x200S1=[fp200x200S1 0];
        ta200x200S1=[ta200x200S1 1];
    else
        fp200x200S1=[fp200x200S1 1];
        ta200x200S1=[ta200x200S1 0];
    end
end
nom_fichero='200x200/ResultadosMujerM2_42.m';
dat=leeFicheroPersona(nom_fichero,cadena);
[m,n]=size(dat);
for i=1:m
    if dat(i,2)==0
        tmi200x200S1=[tmi200x200S1 0.0001];
    else
        tmi200x200S1=[tmi200x200S1 dat(i,2)];
    end
    if dat(i,4)==i&&dat(i,4)<=50
        fp200x200S1=[fp200x200S1 0];
        ta200x200S1=[ta200x200S1 1];
    else
        fp200x200S1=[fp200x200S1 1];
        ta200x200S1=[ta200x200S1 0];
    end
    if dat(i,4)<=50
        fph200x200S1=[fph200x200S1 0];
        tam200x200S1=[tam200x200S1 1];
    else
```

```
        fph200x200S1=[fph200x200S1 1];
        tam200x200S1=[tam200x200S1 0];
    end
end
nom_fichero='200x200/ResultadosHombreM2_45.m';
dat=leeFicheroPersona(nom_fichero,cadena);
[m,n]=size(dat);
for i=1:m

    if dat(i,2)==0
        tmi200x200A1=[tmi200x200A1 0.0001];
    else
        tmi200x200A1=[tmi200x200A1 dat(i,2)];
    end
    if dat(i,4)>50
        fpm200x200A1=[fpm200x200A1 0];
        tah200x200A1=[tah200x200A1 1];
    else
        fpm200x200A1=[fpm200x200A1 1];
        tah200x200A1=[tah200x200A1 0];
    end
    if dat(i,4)==i+50
        fp200x200A1=[fp200x200A1 0];
        ta200x200A1=[ta200x200A1 1];
    else
        fp200x200A1=[fp200x200A1 1];
        ta200x200A1=[ta200x200A1 0];
    end
end
nom_fichero='200x200/ResultadosMujerM2_45.m';
dat=leeFicheroPersona(nom_fichero,cadena);
[m,n]=size(dat);
for i=1:m
    if dat(i,2)==0
        tmi200x200A1=[tmi200x200A1 0.0001];
    else
        tmi200x200A1=[tmi200x200A1 dat(i,2)];
    end
    if dat(i,4)==i&&dat(i,4)<=50
        fp200x200A1=[fp200x200A1 0];
        ta200x200A1=[ta200x200A1 1];
    else
        fp200x200A1=[fp200x200A1 1];
        ta200x200A1=[ta200x200A1 0];
    end
    if dat(i,4)<=50
        fph200x200A1=[fph200x200A1 0];
        tam200x200A1=[tam200x200A1 1];
    else
        fph200x200A1=[fph200x200A1 1];
        tam200x200A1=[tam200x200A1 0];
    end
end

nom_fichero='200x200/ResultadosHombreM3_02.m';
dat=leeFicheroPersona(nom_fichero,cadena);
[m,n]=size(dat);
for i=1:m

    if dat(i,2)==0
        tmi200x200S1=[tmi200x200S1 0.0001];
    else
        tmi200x200S1=[tmi200x200S1 dat(i,2)];
    end
    if dat(i,4)>5
        fpm200x200S1=[fpm200x200S1 0];
        tah200x200S1=[tah200x200S1 1];
    else
        fpm200x200S1=[fpm200x200S1 1];
        tah200x200S1=[tah200x200S1 0];
    end
    if dat(i,4)==i+5
        fp200x200S1=[fp200x200S1 0];
        ta200x200S1=[ta200x200S1 1];
    else
        fp200x200S1=[fp200x200S1 1];
        ta200x200S1=[ta200x200S1 0];
    end
```

```
end
nom_fichero='200x200/ResultadosMujerM3_02.m';
dat=leeFicheroPersona(nom_fichero,cadena);
[m,n]=size(dat);
for i=1:m
    if dat(i,2)==0
        tmi200x200S1=[tmi200x200S1 0.0001];
    else
        tmi200x200S1=[tmi200x200S1 dat(i,2)];
    end
    if dat(i,4)==i&&dat(i,4)<=5
        fp200x200S1=[fp200x200S1 0];
        ta200x200S1=[ta200x200S1 1];
    else
        fp200x200S1=[fp200x200S1 1];
        ta200x200S1=[ta200x200S1 0];
    end
    if dat(i,4)<=5
        fph200x200S1=[fph200x200S1 0];
        tam200x200S1=[tam200x200S1 1];
    else
        fph200x200S1=[fph200x200S1 1];
        tam200x200S1=[tam200x200S1 0];
    end
end
nom_fichero='200x200/ResultadosHombreM3_05.m';
dat=leeFicheroPersona(nom_fichero,cadena);
[m,n]=size(dat);
for i=1:m

    if dat(i,2)==0
        tmi200x200A1=[tmi200x200A1 0.0001];
    else
        tmi200x200A1=[tmi200x200A1 dat(i,2)];
    end
    if dat(i,4)>5
        fpm200x200A1=[fpm200x200A1 0];
        tah200x200A1=[tah200x200A1 1];
    else
        fpm200x200A1=[fpm200x200A1 1];
        tah200x200A1=[tah200x200A1 0];
    end
    if dat(i,4)==i+5
        fp200x200A1=[fp200x200A1 0];
        ta200x200A1=[ta200x200A1 1];
    else
        fp200x200A1=[fp200x200A1 1];
        ta200x200A1=[ta200x200A1 0];
    end
end
nom_fichero='200x200/ResultadosMujerM3_05.m';
dat=leeFicheroPersona(nom_fichero,cadena);
[m,n]=size(dat);
for i=1:m
    if dat(i,2)==0
        tmi200x200A1=[tmi200x200A1 0.0001];
    else
        tmi200x200A1=[tmi200x200A1 dat(i,2)];
    end
    if dat(i,4)==i&&dat(i,4)<=5
        fp200x200A1=[fp200x200A1 0];
        ta200x200A1=[ta200x200A1 1];
    else
        fp200x200A1=[fp200x200A1 1];
        ta200x200A1=[ta200x200A1 0];
    end
    if dat(i,4)<=5
        fph200x200A1=[fph200x200A1 0];
        tam200x200A1=[tam200x200A1 1];
    else
        fph200x200A1=[fph200x200A1 1];
        tam200x200A1=[tam200x200A1 0];
    end
end
nom_fichero='200x200/ResultadosHombreM3_12.m';
dat=leeFicheroPersona(nom_fichero,cadena);
[m,n]=size(dat);
```

```
for i=1:m
    if i==1
        if dat(i,2)==0
            tmi200x200S2=0.0001;
        else
            tmi200x200S2=dat(i,2);
        end
        if dat(i,4)>5
            fpm200x200S2=0;
            tah200x200S2=1;
        else
            fpm200x200S2=1;
            tah200x200S2=0;
        end
        if dat(i,4)==i+5
            fp200x200S2=0;
            ta200x200S2=1;
        else
            fp200x200S2=1;
            ta200x200S2=0;
        end
    else
        if dat(i,2)==0
            tmi200x200S2=[tmi200x200S2 0.0001];
        else
            tmi200x200S2=[tmi200x200S2 dat(i,2)];
        end
        if dat(i,4)>5
            fpm200x200S2=[fpm200x200S2 0];
            tah200x200S2=[tah200x200S2 1];
        else
            fpm200x200S2=[fpm200x200S2 1];
            tah200x200S2=[tah200x200S2 0];
        end
        if dat(i,4)==i+5
            fp200x200S2=[fp200x200S2 0];
            ta200x200S2=[ta200x200S2 1];
        else
            fp200x200S2=[fp200x200S2 1];
            ta200x200S2=[ta200x200S2 0];
        end
    end
end
nom_fichero='200x200/ResultadosMujerM3_12.m';
dat=leeFicheroPersona(nom_fichero,cadena);
[m,n]=size(dat);
for i=1:m
    if dat(i,2)==0
        tmi200x200S2=[tmi200x200S2 0.0001];
    else
        tmi200x200S2=[tmi200x200S2 dat(i,2)];
    end
    if dat(i,4)==i&&dat(i,4)<=5
        fp200x200S2=[fp200x200S2 0];
        ta200x200S2=[ta200x200S2 1];
    else
        fp200x200S2=[fp200x200S2 1];
        ta200x200S2=[ta200x200S2 0];
    end
    if i==1
        if dat(i,4)<=5
            fph200x200S2=0;
            tam200x200S2=1;
        else
            fph200x200S2=1;
            tam200x200S2=0;
        end
    else
        if dat(i,4)<=5
            fph200x200S2=[fph200x200S2 0];
            tam200x200S2=[tam200x200S2 1];
        else
            fph200x200S2=[fph200x200S2 1];
            tam200x200S2=[tam200x200S2 0];
        end
    end
end
nom_fichero='200x200/ResultadosHombreM3_15.m';
```

```
dat=leeFicheroPersona(nom_fichero,cadena);
[m,n]=size(dat);
for i=1:m
    if i==1
        if dat(i,2)==0
            tmi200x200A2=0.0001;
        else
            tmi200x200A2=dat(i,2);
        end
        if dat(i,4)>5
            fpm200x200A2=0;
            tah200x200A2=1;
        else
            fpm200x200A2=1;
            tah200x200A2=0;
        end
        if dat(i,4)==i+5
            fp200x200A2=0;
            ta200x200A2=1;
        else
            fp200x200A2=1;
            ta200x200A2=0;
        end
    else
        if dat(i,2)==0
            tmi200x200A2=[tmi200x200A2 0.0001];
        else
            tmi200x200A2=[tmi200x200A2 dat(i,2)];
        end
        if dat(i,4)>5
            fpm200x200A2=[fpm200x200A2 0];
            tah200x200A2=[tah200x200A2 1];
        else
            fpm200x200A2=[fpm200x200A2 1];
            tah200x200A2=[tah200x200A2 0];
        end
        if dat(i,4)==i+5
            fp200x200A2=[fp200x200A2 0];
            ta200x200A2=[ta200x200A2 1];
        else
            fp200x200A2=[fp200x200A2 1];
            ta200x200A2=[ta200x200A2 0];
        end
    end
end
nom_fichero='200x200/ResultadosMujerM3_15.m';
dat=leeFicheroPersona(nom_fichero,cadena);
[m,n]=size(dat);
for i=1:m
    if dat(i,2)==0
        tmi200x200A2=[tmi200x200A2 0.0001];
    else
        tmi200x200A2=[tmi200x200A2 dat(i,2)];
    end
    if dat(i,4)==i&&dat(i,4)<=5
        fp200x200A2=[fp200x200A2 0];
        ta200x200A2=[ta200x200A2 1];
    else
        fp200x200A2=[fp200x200A2 1];
        ta200x200A2=[ta200x200A2 0];
    end
    if i==1
        if dat(i,4)<=5
            fph200x200A2=0;
            tam200x200A2=1;
        else
            fph200x200A2=1;
            tam200x200A2=0;
        end
    else
        if dat(i,4)<=5
            fph200x200A2=[fph200x200A2 0];
            tam200x200A2=[tam200x200A2 1];
        else
            fph200x200A2=[fph200x200A2 1];
            tam200x200A2=[tam200x200A2 0];
        end
```

```
    end
end
nom_fichero='200x200/ResultadosHombreM3_22.m';
dat=leeFicheroPersona(nom_fichero,cadena);
[m,n]=size(dat);
for i=1:m

    if dat(i,2)==0
        tmi200x200S1=[tmi200x200S1 0.0001];
    else
        tmi200x200S1=[tmi200x200S1 dat(i,2)];
    end
    if dat(i,4)>25
        fpm200x200S1=[fpm200x200S1 0];
        tah200x200S1=[tah200x200S1 1];
    else
        fpm200x200S1=[fpm200x200S1 1];
        tah200x200S1=[tah200x200S1 0];
    end
    if dat(i,4)==i+25
        fp200x200S1=[fp200x200S1 0];
        ta200x200S1=[ta200x200S1 1];
    else
        fp200x200S1=[fp200x200S1 1];
        ta200x200S1=[ta200x200S1 0];
    end
end
nom_fichero='200x200/ResultadosMujerM3_22.m';
dat=leeFicheroPersona(nom_fichero,cadena);
[m,n]=size(dat);
for i=1:m
    if dat(i,2)==0
        tmi200x200S1=[tmi200x200S1 0.0001];
    else
        tmi200x200S1=[tmi200x200S1 dat(i,2)];
    end
    if dat(i,4)==i&&dat(i,4)<=25
        fp200x200S1=[fp200x200S1 0];
        ta200x200S1=[ta200x200S1 1];
    else
        fp200x200S1=[fp200x200S1 1];
        ta200x200S1=[ta200x200S1 0];
    end
    if dat(i,4)<=25
        fph200x200S1=[fph200x200S1 0];
        tam200x200S1=[tam200x200S1 1];
    else
        fph200x200S1=[fph200x200S1 1];
        tam200x200S1=[tam200x200S1 0];
    end
end
nom_fichero='200x200/ResultadosHombreM3_25.m';
dat=leeFicheroPersona(nom_fichero,cadena);
[m,n]=size(dat);
for i=1:m

    if dat(i,2)==0
        tmi200x200A1=[tmi200x200A1 0.0001];
    else
        tmi200x200A1=[tmi200x200A1 dat(i,2)];
    end
    if dat(i,4)>25
        fpm200x200A1=[fpm200x200A1 0];
        tah200x200A1=[tah200x200A1 1];
    else
        fpm200x200A1=[fpm200x200A1 1];
        tah200x200A1=[tah200x200A1 0];
    end
    if dat(i,4)==i+25
        fp200x200A1=[fp200x200A1 0];
        ta200x200A1=[ta200x200A1 1];
    else
        fp200x200A1=[fp200x200A1 1];
        ta200x200A1=[ta200x200A1 0];
    end
end
nom_fichero='200x200/ResultadosMujerM3_25.m';
dat=leeFicheroPersona(nom_fichero,cadena);
```

```
[m,n]=size(dat);
for i=1:m
    if dat(i,2)==0
        tmi200x200A1=[tmi200x200A1 0.0001];
    else
        tmi200x200A1=[tmi200x200A1 dat(i,2)];
    end
    if dat(i,4)==i&&dat(i,4)<=25
        fp200x200A1=[fp200x200A1 0];
        ta200x200A1=[ta200x200A1 1];
    else
        fp200x200A1=[fp200x200A1 1];
        ta200x200A1=[ta200x200A1 0];
    end
    if dat(i,4)<=25
        fph200x200A1=[fph200x200A1 0];
        tam200x200A1=[tam200x200A1 1];
    else
        fph200x200A1=[fph200x200A1 1];
        tam200x200A1=[tam200x200A1 0];
    end
end
nom_fichero='200x200/ResultadosHombreM3_32.m';
dat=leeFicheroPersona(nom_fichero,cadena);
[m,n]=size(dat);
for i=1:m

    if dat(i,2)==0
        tmi200x200S2=[tmi200x200S2 0.0001];
    else
        tmi200x200S2=[tmi200x200S2 dat(i,2)];
    end
    if dat(i,4)>25
        fpm200x200S2=[fpm200x200S2 0];
        tah200x200S2=[tah200x200S2 1];
    else
        fpm200x200S2=[fpm200x200S2 1];
        tah200x200S2=[tah200x200S2 0];
    end
    if dat(i,4)==i+25
        fp200x200S2=[fp200x200S2 0];
        ta200x200S2=[ta200x200S2 1];
    else
        fp200x200S2=[fp200x200S2 1];
        ta200x200S2=[ta200x200S2 0];
    end
end
nom_fichero='200x200/ResultadosMujerM3_32.m';
dat=leeFicheroPersona(nom_fichero,cadena);
[m,n]=size(dat);
for i=1:m
    if dat(i,2)==0
        tmi200x200S2=[tmi200x200S2 0.0001];
    else
        tmi200x200S2=[tmi200x200S2 dat(i,2)];
    end
    if dat(i,4)==i&&dat(i,4)<=25
        fp200x200S2=[fp200x200S2 0];
        ta200x200S2=[ta200x200S2 1];
    else
        fp200x200S2=[fp200x200S2 1];
        ta200x200S2=[ta200x200S2 0];
    end
    if dat(i,4)<=25
        fph200x200S2=[fph200x200S2 0];
        tam200x200S2=[tam200x200S2 1];
    else
        fph200x200S2=[fph200x200S2 1];
        tam200x200S2=[tam200x200S2 0];
    end
end
nom_fichero='200x200/ResultadosHombreM3_35.m';
dat=leeFicheroPersona(nom_fichero,cadena);
[m,n]=size(dat);
for i=1:m

    if dat(i,2)==0
```

```
        tmi200x200A2=[tmi200x200A2 0.0001];
    else
        tmi200x200A2=[tmi200x200A2 dat(i,2)];
    end
    if dat(i,4)>25
        fpm200x200A2=[fpm200x200A2 0];
        tah200x200A2=[tah200x200A2 1];
    else
        fpm200x200A2=[fpm200x200A2 1];
        tah200x200A2=[tah200x200A2 0];
    end
    if dat(i,4)==i+25
        fp200x200A2=[fp200x200A2 0];
        ta200x200A2=[ta200x200A2 1];
    else
        fp200x200A2=[fp200x200A2 1];
        ta200x200A2=[ta200x200A2 0];
    end
end
nom_fichero='200x200/ResultadosMujerM3_35.m';
dat=leeFicheroPersona(nom_fichero,cadena);
[m,n]=size(dat);
for i=1:m
    if dat(i,2)==0
        tmi200x200A2=[tmi200x200A2 0.0001];
    else
        tmi200x200A2=[tmi200x200A2 dat(i,2)];
    end
    if dat(i,4)==i&&dat(i,4)<=25
        fp200x200A2=[fp200x200A2 0];
        ta200x200A2=[ta200x200A2 1];
    else
        fp200x200A2=[fp200x200A2 1];
        ta200x200A2=[ta200x200A2 0];
    end
    if dat(i,4)<=25
        fph200x200A2=[fph200x200A2 0];
        tam200x200A2=[tam200x200A2 1];
    else
        fph200x200A2=[fph200x200A2 1];
        tam200x200A2=[tam200x200A2 0];
    end
end
nom_fichero='200x200/ResultadosHombreM3_42.m';
dat=leeFicheroPersona(nom_fichero,cadena);
[m,n]=size(dat);
for i=1:m

    if dat(i,2)==0
        tmi200x200S1=[tmi200x200S1 0.0001];
    else
        tmi200x200S1=[tmi200x200S1 dat(i,2)];
    end
    if dat(i,4)>50
        fpm200x200S1=[fpm200x200S1 0];
        tah200x200S1=[tah200x200S1 1];
    else
        fpm200x200S1=[fpm200x200S1 1];
        tah200x200S1=[tah200x200S1 0];
    end
    if dat(i,4)==i+50
        fp200x200S1=[fp200x200S1 0];
        ta200x200S1=[ta200x200S1 1];
    else
        fp200x200S1=[fp200x200S1 1];
        ta200x200S1=[ta200x200S1 0];
    end
end
nom_fichero='200x200/ResultadosMujerM3_42.m';
dat=leeFicheroPersona(nom_fichero,cadena);
[m,n]=size(dat);
for i=1:m
    if dat(i,2)==0
        tmi200x200S1=[tmi200x200S1 0.0001];
    else
        tmi200x200S1=[tmi200x200S1 dat(i,2)];
    end
    if dat(i,4)==i&&dat(i,4)<=50
```

```
        fp200x200S1=[fp200x200S1 0];
        ta200x200S1=[ta200x200S1 1];
    else
        fp200x200S1=[fp200x200S1 1];
        ta200x200S1=[ta200x200S1 0];
    end
    if dat(i,4)<=50
        fph200x200S1=[fph200x200S1 0];
        tam200x200S1=[tam200x200S1 1];
    else
        fph200x200S1=[fph200x200S1 1];
        tam200x200S1=[tam200x200S1 0];
    end
end
nom_fichero='200x200/ResultadosHombreM3_45.m';
dat=leeFicheroPersona(nom_fichero,cadena);
[m,n]=size(dat);
for i=1:m

    if dat(i,2)==0
        tmi200x200A1=[tmi200x200A1 0.0001];
    else
        tmi200x200A1=[tmi200x200A1 dat(i,2)];
    end
    if dat(i,4)>50
        fpm200x200A1=[fpm200x200A1 0];
        tah200x200A1=[tah200x200A1 1];
    else
        fpm200x200A1=[fpm200x200A1 1];
        tah200x200A1=[tah200x200A1 0];
    end
    if dat(i,4)==i+50
        fp200x200A1=[fp200x200A1 0];
        ta200x200A1=[ta200x200A1 1];
    else
        fp200x200A1=[fp200x200A1 1];
        ta200x200A1=[ta200x200A1 0];
    end
end
nom_fichero='200x200/ResultadosMujerM3_45.m';
dat=leeFicheroPersona(nom_fichero,cadena);
[m,n]=size(dat);
for i=1:m
    if dat(i,2)==0
        tmi200x200A1=[tmi200x200A1 0.0001];
    else
        tmi200x200A1=[tmi200x200A1 dat(i,2)];
    end
    if dat(i,4)==i&&dat(i,4)<=50
        fp200x200A1=[fp200x200A1 0];
        ta200x200A1=[ta200x200A1 1];
    else
        fp200x200A1=[fp200x200A1 1];
        ta200x200A1=[ta200x200A1 0];
    end
    if dat(i,4)<=50
        fph200x200A1=[fph200x200A1 0];
        tam200x200A1=[tam200x200A1 1];
    else
        fph200x200A1=[fph200x200A1 1];
        tam200x200A1=[tam200x200A1 0];
    end
end
nom_fichero='200x200/ResultadosHombreM3_52.m';
dat=leeFicheroPersona(nom_fichero,cadena);
[m,n]=size(dat);
for i=1:m

    if dat(i,2)==0
        tmi200x200S2=[tmi200x200S2 0.0001];
    else
        tmi200x200S2=[tmi200x200S2 dat(i,2)];
    end
    if dat(i,4)>50
        fpm200x200S2=[fpm200x200S2 0];
        tah200x200S2=[tah200x200S2 1];
    else
```

```
        fpm200x200S2=[fpm200x200S2 1];
        tah200x200S2=[tah200x200S2 0];
    end
    if dat(i,4)==i+50
        fp200x200S2=[fp200x200S2 0];
        ta200x200S2=[ta200x200S2 1];
    else
        fp200x200S2=[fp200x200S2 1];
        ta200x200S2=[ta200x200S2 0];
    end
end
nom_fichero='200x200/ResultadosMujerM3_52.m';
dat=leeFicheroPersona(nom_fichero,cadena);
[m,n]=size(dat);
for i=1:m
    if dat(i,2)==0
        tmi200x200S2=[tmi200x200S2 0.0001];
    else
        tmi200x200S2=[tmi200x200S2 dat(i,2)];
    end
    if dat(i,4)==i&&dat(i,4)<=50
        fp200x200S2=[fp200x200S2 0];
        ta200x200S2=[ta200x200S2 1];
    else
        fp200x200S2=[fp200x200S2 1];
        ta200x200S2=[ta200x200S2 0];
    end
    if dat(i,4)<=50
        fph200x200S2=[fph200x200S2 0];
        tam200x200S2=[tam200x200S2 1];
    else
        fph200x200S2=[fph200x200S2 1];
        tam200x200S2=[tam200x200S2 0];
    end
end
nom_fichero='200x200/ResultadosHombreM3_55.m';
dat=leeFicheroPersona(nom_fichero,cadena);
[m,n]=size(dat);
for i=1:m

    if dat(i,2)==0
        tmi200x200A2=[tmi200x200A2 0.0001];
    else
        tmi200x200A2=[tmi200x200A2 dat(i,2)];
    end
    if dat(i,4)>50
        fpm200x200A2=[fpm200x200A2 0];
        tah200x200A2=[tah200x200A2 1];
    else
        fpm200x200A2=[fpm200x200A2 1];
        tah200x200A2=[tah200x200A2 0];
    end
    if dat(i,4)==i+50
        fp200x200A2=[fp200x200A2 0];
        ta200x200A2=[ta200x200A2 1];
    else
        fp200x200A2=[fp200x200A2 1];
        ta200x200A2=[ta200x200A2 0];
    end
end
nom_fichero='200x200/ResultadosMujerM3_55.m';
dat=leeFicheroPersona(nom_fichero,cadena);
[m,n]=size(dat);
for i=1:m
    if dat(i,2)==0
        tmi200x200A2=[tmi200x200A2 0.0001];
    else
        tmi200x200A2=[tmi200x200A2 dat(i,2)];
    end
    if dat(i,4)==i&&dat(i,4)<=50
        fp200x200A2=[fp200x200A2 0];
        ta200x200A2=[ta200x200A2 1];
    else
        fp200x200A2=[fp200x200A2 1];
        ta200x200A2=[ta200x200A2 0];
    end
    if dat(i,4)<=50
        fph200x200A2=[fph200x200A2 0];
```

```
        tam200x200A2=[tam200x200A2 1];
    else
        fph200x200A2=[fph200x200A2 1];
        tam200x200A2=[tam200x200A2 0];
    end
end

% Generados todos los datos, se hace la clasificacion para 10 clases,
% 50 clases y 100 clases.
m=size(tmi200x200S1,2);
tmi200x200S12=reshape(tmi200x200S1,m/6,6);
tmi200x200S1=[(sum(tmi200x200S12(:,1))+sum(tmi200x200S12(:,4)))/(m/3)
(sum(tmi200x200S12(:,2))+sum(tmi200x200S12(:,5)))/(m/3)
(sum(tmi200x200S12(:,3))+sum(tmi200x200S12(:,6)))/(m/3)];
tmi200x200S12=0;

m=size(tmi200x200A1,2);
tmi200x200A12=reshape(tmi200x200A1,m/6,6);
tmi200x200A1=[(sum(tmi200x200A12(:,1))+sum(tmi200x200A12(:,4)))/(m/3)
(sum(tmi200x200A12(:,2))+sum(tmi200x200A12(:,5)))/(m/3)
(sum(tmi200x200A12(:,3))+sum(tmi200x200A12(:,6)))/(m/3)];
tmi200x200A12=0;

m=size(tmi200x200S2,2);
tmi200x200S22=reshape(tmi200x200S2,m/3,3);
tmi200x200S2=[sum(tmi200x200S22(:,1))/(m/3) sum(tmi200x200S22(:,2))/(m/3)
sum(tmi200x200S22(:,3))/(m/3)];
tmi200x200S22=0;

m=size(tmi200x200A2,2);
tmi200x200A22=reshape(tmi200x200A2,m/3,3);
tmi200x200A2=[sum(tmi200x200A22(:,1))/(m/3) sum(tmi200x200A22(:,2))/(m/3)
sum(tmi200x200A22(:,3))/(m/3)];
tmi200x200A22=0;


m=size(fpm200x200S1,2);
fpm200x200S12=reshape(fpm200x200S1,m/6,6);
fpm200x200S1=[(sum(fpm200x200S12(:,1))+sum(fpm200x200S12(:,4)))/(m/3)
(sum(fpm200x200S12(:,2))+sum(fpm200x200S12(:,5)))/(m/3)
(sum(fpm200x200S12(:,3))+sum(fpm200x200S12(:,6)))/(m/3)];
fpm200x200S12=0;

m=size(fpm200x200A1,2);
fpm200x200A12=reshape(fpm200x200A1,m/6,6);
fpm200x200A1=[(sum(fpm200x200A12(:,1))+sum(fpm200x200A12(:,4)))/(m/3)
(sum(fpm200x200A12(:,2))+sum(fpm200x200A12(:,5)))/(m/3)
(sum(fpm200x200A12(:,3))+sum(fpm200x200A12(:,6)))/(m/3)];
fpm200x200A12=0;

m=size(fpm200x200S2,2);
fpm200x200S22=reshape(fpm200x200S2,m/3,3);
fpm200x200S2=[sum(fpm200x200S22(:,1))/(m/3) sum(fpm200x200S22(:,2))/(m/3)
sum(fpm200x200S22(:,3))/(m/3)];
fpm200x200S22=0;

m=size(fpm200x200A2,2);
fpm200x200A22=reshape(fpm200x200A2,m/3,3);
fpm200x200A2=[sum(fpm200x200A22(:,1))/(m/3) sum(fpm200x200A22(:,2))/(m/3)
sum(fpm200x200A22(:,3))/(m/3)];
fpm200x200A22=0;


m=size(fph200x200S1,2);
fph200x200S12=reshape(fph200x200S1,m/6,6);
fph200x200S1=[(sum(fph200x200S12(:,1))+sum(fph200x200S12(:,4)))/(m/3)
(sum(fph200x200S12(:,2))+sum(fph200x200S12(:,5)))/(m/3)
(sum(fph200x200S12(:,3))+sum(fph200x200S12(:,6)))/(m/3)];
fph200x200S12=0;

m=size(fph200x200A1,2);
fph200x200A12=reshape(fph200x200A1,m/6,6);
fph200x200A1=[(sum(fph200x200A12(:,1))+sum(fph200x200A12(:,4)))/(m/3)
(sum(fph200x200A12(:,2))+sum(fph200x200A12(:,5)))/(m/3)
(sum(fph200x200A12(:,3))+sum(fph200x200A12(:,6)))/(m/3)];
fph200x200A12=0;
```

```
m=size(fph200x200S2,2);
fph200x200S22=reshape(fph200x200S2,m/3,3);
fph200x200S2=[sum(fph200x200S22(:,1))/(m/3) sum(fph200x200S22(:,2))/(m/3)
sum(fph200x200S22(:,3))/(m/3)];
fph200x200S22=0;

m=size(fph200x200A2,2);
fph200x200A22=reshape(fph200x200A2,m/3,3);
fph200x200A2=[sum(fph200x200A22(:,1))/(m/3) sum(fph200x200A22(:,2))/(m/3)
sum(fph200x200A22(:,3))/(m/3)];
fph200x200A22=0;


m=size(tah200x200S1,2);
tah200x200S12=reshape(tah200x200S1,m/6,6);
tah200x200S1=[(sum(tah200x200S12(:,1))+sum(tah200x200S12(:,4)))/(m/3)
(sum(tah200x200S12(:,2))+sum(tah200x200S12(:,5)))/(m/3)
(sum(tah200x200S12(:,3))+sum(tah200x200S12(:,6)))/(m/3)];
tah200x200S12=0;

m=size(tah200x200A1,2);
tah200x200A12=reshape(tah200x200A1,m/6,6);
tah200x200A1=[(sum(tah200x200A12(:,1))+sum(tah200x200A12(:,4)))/(m/3)
(sum(tah200x200A12(:,2))+sum(tah200x200A12(:,5)))/(m/3)
(sum(tah200x200A12(:,3))+sum(tah200x200A12(:,6)))/(m/3)];
tah200x200A12=0;

m=size(tah200x200S2,2);
tah200x200S22=reshape(tah200x200S2,m/3,3);
tah200x200S2=[sum(tah200x200S22(:,1))/(m/3) sum(tah200x200S22(:,2))/(m/3)
sum(tah200x200S22(:,3))/(m/3)];
tah200x200S22=0;

m=size(tah200x200A2,2);
tah200x200A22=reshape(tah200x200A2,m/3,3);
tah200x200A2=[sum(tah200x200A22(:,1))/(m/3) sum(tah200x200A22(:,2))/(m/3)
sum(tah200x200A22(:,3))/(m/3)];
tah200x200A22=0;


m=size(tam200x200S1,2);
tam200x200S12=reshape(tam200x200S1,m/6,6);
tam200x200S1=[(sum(tam200x200S12(:,1))+sum(tam200x200S12(:,4)))/(m/3)
(sum(tam200x200S12(:,2))+sum(tam200x200S12(:,5)))/(m/3)
(sum(tam200x200S12(:,3))+sum(tam200x200S12(:,6)))/(m/3)];
tam200x200S12=0;

m=size(tam200x200A1,2);
tam200x200A12=reshape(tam200x200A1,m/6,6);
tam200x200A1=[(sum(tam200x200A12(:,1))+sum(tam200x200A12(:,4)))/(m/3)
(sum(tam200x200A12(:,2))+sum(tam200x200A12(:,5)))/(m/3)
(sum(tam200x200A12(:,3))+sum(tam200x200A12(:,6)))/(m/3)];
tam200x200A12=0;

m=size(tam200x200S2,2);
tam200x200S22=reshape(tam200x200S2,m/3,3);
tam200x200S2=[sum(tam200x200S22(:,1))/(m/3) sum(tam200x200S22(:,2))/(m/3)
sum(tam200x200S22(:,3))/(m/3)];
tam200x200S22=0;

m=size(tam200x200A2,2);
tam200x200A22=reshape(tam200x200A2,m/3,3);
tam200x200A2=[sum(tam200x200A22(:,1))/(m/3) sum(tam200x200A22(:,2))/(m/3)
sum(tam200x200A22(:,3))/(m/3)];
tam200x200A22=0;


m=size(fp200x200S1,2);
fp200x200S12=reshape(fp200x200S1,m/6,6);
fp200x200S1=[(sum(fp200x200S12(:,1))+sum(fp200x200S12(:,4)))/(m/3)
(sum(fp200x200S12(:,2))+sum(fp200x200S12(:,5)))/(m/3)
(sum(fp200x200S12(:,3))+sum(fp200x200S12(:,6)))/(m/3)];
fp200x200S12=0;

m=size(fp200x200A1,2);
fp200x200A12=reshape(fp200x200A1,m/6,6);
```

```
fp200x200A1=[(sum(fp200x200A12(:,1))+sum(fp200x200A12(:,4)))/(m/3)
(sum(fp200x200A12(:,2))+sum(fp200x200A12(:,5)))/(m/3)
(sum(fp200x200A12(:,3))+sum(fp200x200A12(:,6)))/(m/3)];
fp200x200A12=0;

m=size(fp200x200S2,2);
fp200x200S22=reshape(fp200x200S2,m/3,3);
fp200x200S2=[sum(fp200x200S22(:,1))/(m/3) sum(fp200x200S22(:,2))/(m/3)
sum(fp200x200S22(:,3))/(m/3)];
fp200x200S22=0;

fp200x200A2;
m=size(fp200x200A2,2);
fp200x200A22=reshape(fp200x200A2,m/3,3);
fp200x200A2=[sum(fp200x200A22(:,1))/(m/3) sum(fp200x200A22(:,2))/(m/3)
sum(fp200x200A22(:,3))/(m/3)];
fp200x200A22=0;


m=size(ta200x200S1,2);
ta200x200S12=reshape(ta200x200S1,m/6,6);
ta200x200S1=[(sum(ta200x200S12(:,1))+sum(ta200x200S12(:,4)))/20
(sum(ta200x200S12(:,2))+sum(ta200x200S12(:,5)))/100
(sum(ta200x200S12(:,3))+sum(ta200x200S12(:,6)))/200];
ta200x200S12=0;

m=size(ta200x200A1,2);
ta200x200A12=reshape(ta200x200A1,m/6,6);
ta200x200A1=[(sum(ta200x200A12(:,1))+sum(ta200x200A12(:,4)))/20
(sum(ta200x200A12(:,2))+sum(ta200x200A12(:,5)))/100
(sum(ta200x200A12(:,3))+sum(ta200x200A12(:,6)))/200];
ta200x200A12=0;

m=size(ta200x200S2,2);
ta200x200S22=reshape(ta200x200S2,m/3,3);
ta200x200S2=[sum(ta200x200S22(:,1))/10 sum(ta200x200S22(:,2))/50 sum(ta200x200S22(:,3))/100];
ta200x200S22=0;

ta200x200A2;
m=size(ta200x200A2,2);
ta200x200A22=reshape(ta200x200A2,m/3,3);
ta200x200A2=[sum(ta200x200A22(:,1))/10 sum(ta200x200A22(:,2))/50 sum(ta200x200A22(:,3))/100];
ta200x200A22=0;
```

Fichero de la función leeFicheroPersona.m:

```
%
% Fichero: leeFicheroPersona.m
% Autor: Maria Sierra Zapata
% Fecha: 01/07/2015
% Version: 0.1
%
% Breve Descripcion:
%       Funcion que se encarga de leer los datos de un fichero con un
%       formato especifico.
%
% Parametros de Entrada:
%       nom_fichero :  Contiene la cadena del fichero del que se quiere
%                      leer los datos.
%       cadena :  Contiene el formato con el que se desea leer el fichero.
%
% Parametros de Salida:
%       Y :  Matriz que contiene los datos obtenidos del fichero.
%

function Y = leeFicheroPersona(nom_fichero, cadena)

% Apertura del fichero
fichero=fopen(nom_fichero,'r');
fgets(fichero);
% Lectura del fichero
X=fscanf(fichero,cadena);
Y=reshape(X,4,size(X,1)/4)';
% Cierre del fichero
fclose(fichero);
end
```

Fichero dibujaDatos.m:

```
%
% Fichero: dibujaDatos.m
% Autor: Maria Sierra Zapata
% Fecha: 01/07/2015
% Version: 0.1
%
% Breve Descripcion:
%       Representacion grafica de los datos generados por los ficheros
%       analizaFichero50x50.m, analizaFichero100x100.m,
%       analizaFichero150x150.m y analizaFichero200x200.m.
%
% Graficas generadas:
%       Tiempo Medio de Identificacion - Numero de clases
%       % Falsos Positivos Mujer - Numero de clases
%       % Falsos Positivos Hombre - Numero de clases
%       % Falsos Positivos - Numero de clases
%       % Tasa de Acierto Mujer - Numero de clases
%       % Tasa de Acierto Hombre - Numero de clases
%       % Tasa de Acierto - Numero de clases
%

figure(1)
subplot(2,2,1)
plot(num_images, tmi50x50S1,'.-', num_images, tmi50x50A1, '.-', num_images, tmi50x50S2, '.-',
num_images, tmi50x50A2, '.-')
xlabel('Num Clases')
ylabel('Segundos')
legend('S - 1:1','A - 1:1', 'S - 2:1', 'A - 2:1', 'Location', 'Best')
title('Tam Imagen 50x50')

subplot(2,2,2)
plot(num_images, tmi100x100S1,'.-', num_images, tmi100x100A1, '.-', num_images, tmi100x100S2, '.-
', num_images, tmi100x100A2, '.-')
xlabel('Num Clases')
ylabel('Segundos')
legend('S - 1:1','A - 1:1', 'S - 2:1', 'A - 2:1', 'Location', 'Best')
title('Tam Imagen 100x100')

subplot(2,2,3)
plot(num_images, tmi150x150S1,'.-', num_images, tmi150x150A1, '.-', num_images, tmi150x150S2, '.-
', num_images, tmi150x150A2, '.-')
xlabel('Num Clases')
ylabel('Segundos')
legend('S - 1:1','A - 1:1', 'S - 2:1', 'A - 2:1', 'Location', 'Best')
title('Tam Imagen 150x150')

subplot(2,2,4)
plot(num_images, tmi200x200S1,'.-', num_images, tmi200x200A1, '.-', num_images, tmi200x200S2, '.-
', num_images, tmi200x200A2, '.-')
xlabel('Num Clases')
ylabel('Segundos')
legend('S - 1:1','A - 1:1', 'S - 2:1', 'A - 2:1', 'Location', 'Best')
title('Tam Imagen 200x200')

figure(2)
subplot(2,2,1)
plot(num_images, 100.*fp50x50S1,'.-', num_images, 100.*fp50x50A1,'.-', num_images,
100.*fp50x50S2, '.-', num_images, 100.*fp50x50A2,'.-')
xlabel('Num Clases')
ylabel('% Falsos Positivos')
legend('S - 1:1','A - 1:1', 'S - 2:1', 'A - 2:1', 'Location', 'Best')
title('Tam Imagen 50x50')

subplot(2,2,2)
plot(num_images, 100.*fp100x100S1,'.-', num_images, 100.*fp100x100A1,'.-', num_images,
100.*fp100x100S2, '.-', num_images, 100.*fp100x100A2, '.-')
xlabel('Num Clases')
ylabel('% Falsos Positivos')
legend('S - 1:1','A - 1:1', 'S - 2:1', 'A - 2:1', 'Location', 'Best')
title('Tam Imagen 100x100')

subplot(2,2,3)
plot(num_images, 100.*fp150x150S1,'.-', num_images, 100.*fp150x150A1,'.-', num_images,
100.*fp150x150S2, '.-', num_images, 100.*fp150x150A2, '.-')
xlabel('Num Clases')
```

```
ylabel('% Falsos Positivos')
legend('S - 1:1','A - 1:1', 'S - 2:1', 'A - 2:1', 'Location', 'Best')
title('Tam Imagen 150x150')

subplot(2,2,4)
plot(num_images, 100.*fp200x200S1,'.-', num_images, 100.*fp200x200A1,'.-', num_images,
100.*fp200x200S2, '.-', num_images, 100.*fp200x200A2, '.-')
xlabel('Num Clases')
ylabel('% Falsos Positivos')
legend('S - 1:1','A - 1:1', 'S - 2:1', 'A - 2:1', 'Location', 'Best')
title('Tam Imagen 200x200')

figure(3)
subplot(2,2,1)
plot(num_images, 100.*fph50x50S1,'.-', num_images, 100.*fph50x50A1,'.-', num_images,
100.*fph50x50S2, '.-', num_images, 100.*fph50x50A2, '.-')
xlabel('Num Clases')
ylabel('% Falsos Positivos Hombre')
legend('S - 1:1','A - 1:1', 'S - 2:1', 'A - 2:1', 'Location', 'Best')
title('Tam Imagen 50x50')

subplot(2,2,2)
plot(num_images, 100.*fph100x100S1,'.-', num_images, 100.*fph100x100A1,'.-', num_images,
100.*fph100x100S2, '.-', num_images, 100.*fph100x100A2, '.-')
xlabel('Num Clases')
ylabel('% Falsos Positivos Hombre')
legend('S - 1:1','A - 1:1', 'S - 2:1', 'A - 2:1', 'Location', 'Best')
title('Tam Imagen 100x100')

subplot(2,2,3)
plot(num_images, 100.*fph150x150S1,'.-', num_images, 100.*fph150x150A1,'.-', num_images,
100.*fph150x150S2, '.-', num_images, 100.*fph150x150A2, '.-')
xlabel('Num Clases')
ylabel('% Falsos Positivos Hombre')
legend('S - 1:1','A - 1:1', 'S - 2:1', 'A - 2:1', 'Location', 'Best')
title('Tam Imagen 150x150')

subplot(2,2,4)
plot(num_images, 100.*fph200x200S1,'.-', num_images, 100.*fph200x200A1,'.-', num_images,
100.*fph200x200S2, '.-', num_images, 100.*fph200x200A2, '.-')
xlabel('Num Clases')
ylabel('% Falsos Positivos Hombre')
legend('S - 1:1','A - 1:1', 'S - 2:1', 'A - 2:1', 'Location', 'Best')
title('Tam Imagen 200x200')

figure(4)
subplot(2,2,1)
plot(num_images, 100.*fpm50x50S1,'.-', num_images, 100.*fpm50x50A1,'.-', num_images,
100.*fpm50x50S2, '.-', num_images, 100.*fpm50x50A2, '.-')
xlabel('Num Clases')
ylabel('% Falsos Positivos Mujer')
legend('S - 1:1','A - 1:1', 'S - 2:1', 'A - 2:1', 'Location', 'Best')
title('Tam Imagen 50x50')

subplot(2,2,2)
plot(num_images, 100.*fpm100x100S1,'.-', num_images, 100.*fpm100x100A1,'.-', num_images,
100.*fpm100x100S2, '.-', num_images, 100.*fpm100x100A2, '.-')
xlabel('Num Clases')
ylabel('% Falsos Positivos Mujer')
legend('S - 1:1','A - 1:1', 'S - 2:1', 'A - 2:1', 'Location', 'Best')
title('Tam Imagen 100x100')

subplot(2,2,3)
plot(num_images, 100.*fpm150x150S1,'.-', num_images, 100.*fpm150x150A1,'.-', num_images,
100.*fpm150x150S2, '.-', num_images, 100.*fpm150x150A2, '.-')
xlabel('Num Clases')
ylabel('% Falsos Positivos Mujer')
legend('S - 1:1','A - 1:1', 'S - 2:1', 'A - 2:1', 'Location', 'Best')
title('Tam Imagen 150x150')

subplot(2,2,4)
plot(num_images, 100.*fpm200x200S1,'.-', num_images, 100.*fpm200x200A1,'.-', num_images,
100.*fpm200x200S2, '.-', num_images, 100.*fpm200x200A2, '.-')
xlabel('Num Clases')
ylabel('% Falsos Positivos Mujer')
legend('S - 1:1','A - 1:1', 'S - 2:1', 'A - 2:1', 'Location', 'Best')
title('Tam Imagen 200x200')
```

```
figure(5)
subplot(2,2,1)
plot(num_images, 100.*ta50x50S1,'.-', num_images, 100.*ta50x50A1,'.-', num_images,
100.*ta50x50S2, '.-', num_images, 100.*ta50x50A2, '.-')
xlabel('Num Clases')
ylabel('% Tasa de Acierto')
legend('S - 1:1','A - 1:1', 'S - 2:1', 'A - 2:1', 'Location', 'Best')
title('Tam Imagen 50x50')

subplot(2,2,2)
plot(num_images, 100.*ta100x100S1,'.-', num_images, 100.*ta100x100A1,'.-', num_images,
100.*ta100x100S2, '.-', num_images, 100.*ta100x100A2, '.-')
xlabel('Num Clases')
ylabel('% Tasa de Acierto')
legend('S - 1:1','A - 1:1', 'S - 2:1', 'A - 2:1', 'Location', 'Best')
title('Tam Imagen 100x100')

subplot(2,2,3)
plot(num_images, 100.*ta150x150S1,'.-', num_images, 100.*ta150x150A1,'.-', num_images,
100.*ta150x150S2, '.-', num_images, 100.*ta150x150A2, '.-')
xlabel('Num Clases')
ylabel('% Tasa de Acierto')
legend('S - 1:1','A - 1:1', 'S - 2:1', 'A - 2:1', 'Location', 'Best')
title('Tam Imagen 150x150')

subplot(2,2,4)
plot(num_images, 100.*ta200x200S1,'.-', num_images, 100.*ta200x200A1,'.-', num_images,
100.*ta200x200S2, '.-', num_images, 100.*ta200x200A2, '.-')
xlabel('Num Clases')
ylabel('% Tasa de Acierto')
legend('S - 1:1','A - 1:1', 'S - 2:1', 'A - 2:1', 'Location', 'Best')
title('Tam Imagen 200x200')

figure(6)
subplot(2,2,1)
plot(num_images, 100.*tah50x50S1,'.-', num_images, 100.*tah50x50A1,'.-', num_images,
100.*tah50x50S2, '.-', num_images, 100.*tah50x50A2, '.-')
xlabel('Num Clases')
ylabel('% Tasa de Acierto Hombre')
legend('S - 1:1','A - 1:1', 'S - 2:1', 'A - 2:1', 'Location', 'Best')
title('Tam Imagen 50x50')

subplot(2,2,2)
plot(num_images, 100.*tah100x100S1,'.-', num_images, 100.*tah100x100A1,'.-', num_images,
100.*tah100x100S2, '.-', num_images, 100.*tah100x100A2, '.-')
xlabel('Num Clases')
ylabel('% Tasa de Acierto Hombre')
legend('S - 1:1','A - 1:1', 'S - 2:1', 'A - 2:1', 'Location', 'Best')
title('Tam Imagen 100x100')

subplot(2,2,3)
plot(num_images, 100.*tah150x150S1,'.-', num_images, 100.*tah150x150A1,'.-', num_images,
100.*tah150x150S2, '.-', num_images, 100.*tah150x150A2, '.-')
xlabel('Num Clases')
ylabel('% Tasa de Acierto Hombre')
legend('S - 1:1','A - 1:1', 'S - 2:1', 'A - 2:1', 'Location', 'Best')
title('Tam Imagen 150x150')

subplot(2,2,4)
plot(num_images, 100.*tah200x200S1,'.-', num_images, 100.*tah200x200A1,'.-', num_images,
100.*tah200x200S2, '.-', num_images, 100.*tah200x200A2, '.-')
xlabel('Num Clases')
ylabel('% Tasa de Acierto Hombre')
legend('S - 1:1','A - 1:1', 'S - 2:1', 'A - 2:1', 'Location', 'Best')
title('Tam Imagen 200x200')

figure(7)
subplot(2,2,1)
plot(num_images, 100.*tam50x50S1,'.-', num_images, 100.*tam50x50A1,'.-', num_images,
100.*tam50x50S2, '.-', num_images, 100.*tam50x50A2, '.-')
xlabel('Num Clases')
ylabel('% Tasa de Acierto Mujer')
legend('S - 1:1','A - 1:1', 'S - 2:1', 'A - 2:1', 'Location', 'Best')
title('Tam Imagen 50x50')

subplot(2,2,2)
```

```
plot(num_images, 100.*tam100x100S1,'.-', num_images, 100.*tam100x100A1,'.-', num_images,
100.*tam100x100S2, '.-', num_images, 100.*tam100x100A2, '.-')
xlabel('Num Clases')
ylabel('% Tasa de Acierto Mujer')
legend('S - 1:1','A - 1:1', 'S - 2:1', 'A - 2:1', 'Location', 'Best')
title('Tam Imagen 100x100')

subplot(2,2,3)
plot(num_images, 100.*tam150x150S1,'.-', num_images, 100.*tam150x150A1,'.-', num_images,
100.*tam150x150S2, '.-', num_images, 100.*tam150x150A2, '.-')
xlabel('Num Clases')
ylabel('% Tasa de Acierto Mujer')
legend('S - 1:1','A - 1:1', 'S - 2:1', 'A - 2:1', 'Location', 'Best')
title('Tam Imagen 150x150')

subplot(2,2,4)
plot(num_images, 100.*tam200x200S1,'.-', num_images, 100.*tam200x200A1,'.-', num_images,
100.*tam200x200S2, '.-', num_images, 100.*tam200x200A2, '.-')
xlabel('Num Clases')
ylabel('% Tasa de Acierto Mujer')
legend('S - 1:1','A - 1:1', 'S - 2:1', 'A - 2:1', 'Location', 'Best')
title('Tam Imagen 200x200')
```