Práctica 3:

Reconocimiento de caras.

Sistemas de Percepción

Tabla de contenidos

1. Introducción.

Detectar y reconocer rostros humanos en imágenes (y secuencias de video) es un problema cada vez más en auge en el campo de la visión por computador (en inglés, *computer vision*). Existen distintas aplicaciones prácticas como: vigilancia, videoconferencia, interfaces hombre-máquina (HMI), detección de expresiones faciales, control de acceso o sistemas de recuperación de imágenes basadas en contenidos.

El planteamiento de partida es tener un nivel de conocimiento previo de imágenes almacenadas en forma de bases de datos, a partir del cuál se quiere entrenar algún tipo de sistema de aprendizaje que permita resolver el problema de reconocimiento. Por otro lado, es a partir de 1988 con las publicaciones de los investigadores Kirby y Sirovich de la Universidad de Brown (EE.UU.) [Sirovich et al. 87], y después de Turk y Pentland del MIT (EE.UU) basados en el método *Eigenface* [Turk et al. 91] [Pentland et al. 94] cuando se ha abierto un camino al que ha ido contribuyendo un creciente número de grupos de investigación.

Este documento corresponde a la Práctica 3 que tiene como título **Reconocimiento de caras**.

El documento se estructura de la siguiente forma:

1. En el capítulo 2, se introduce al lector en los objetivos del trabajo a realizar.

2. En el capítulo 3, se presenta una descripción del trabajo desarrollado.

3. En el capítulo 4, se realiza una recopilación de la bibliografía utilizada. Todos los enlaces URL están disponibles a fecha de Mayo de 2008.

2. Objetivos del trabajo.

Habitualmente, se considera una imagen digital como una matriz donde cada valor representa el color del píxel correspondiente. La aplicación tanto del álgebra lineal como de análisis estadístico a dichos datos, ha permitido construir toda una línea de desarrollo científico a lo largo de la cual se han ido sumando varios métodos o algoritmos. De acuerdo con la literatura, existen dos alternativas a la hora de plantearse métodos de reconocimiento basados en la imagen. O bien diseñar un clasificador, como por ejemplo algunos modelos de redes neuronales sobre el espacio de todas las caras. O bien aplicar métodos basados en subespacios u otras técnicas de reducción de dimensionalidad para poder aplicar otros métodos de clasificación en más bajas dimensionalidades como *eigenfaces*.

Para el alcance de los objetivos de esta práctica se ha desarrollado, implementado, probado y comparado un clasificador basado en redes neuronales utilizando una base de datos propia[1], en el Departamento de Electrónica de la Universidad de Mondragón, para la resolución del problema de identificación de imágenes de rostros humanos. Son imágenes de 7 individuos diferentes, a los cuales se les tomó 5 fotos en una sesión con variaciones de luz, expresión facial y punto de vista. La dimensión de las fotos es de 100 de alto por 70 de ancho (píxeles). El objetivo final sería caracterizar su eficacia en problemas simulados y su posible aplicación práctica. Se ha hecho pruebas de clasificación con 5 muestras de cada clase, con 5 clases. Entrenando al clasificador con 4 por clase y probando la restante. La implementación del método PCA o *eigenfaces* y la arquitectura de la red neuronal es modular, que beneficiará posibles mejoras o ampliaciones, así como el aislamiento de posibles errores (*software*) que se puedan producir. Como trabajos similares encontrados, los investigadores Prasanna, Sudha y Kamakoti del Instituto de Tecnología de Madrás (India) describen en [Prasanna et al. 05] una red neuronal que utiliza el método PCA, a la que llaman PCNN.

[1] Existen más de una veintena de bases de datos estándar con imágenes de caras para probar cualquier algoritmo. El acceso a ellas es más o menos (no a todas) público y podemos encontrar referencias en [Cevikalp et al. 05].

Su sistema tolera variaciones locales en la cara como cambios en la expresión y dirección de la luz. En este caso, los autores han diseñado un *hardware* cuya implementación en ASIC[2] es capaz de procesar 11.000 entradas por segundo en el proceso de aprendizaje y 19.000 en la fase de recuperación. Lo cuál es, según sus autores, 10 veces superior y 5 veces más rápido que un sistema *software* funcionando en un PC.

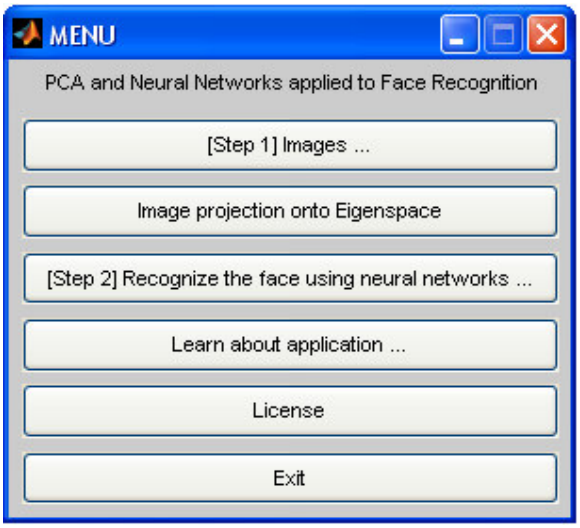

figura 1. Interfaz gráfica (GUI):
PCA and Neural Networks applied to Face Recognition.

3. Trabajo desarrollado.

En este capítulo se describe el trabajo desarrollado. La sección 3.1 y 3.2, responde al paso previo al reconocimiento de caras, dado que conviene tener algo de conciencia sobre las técnicas que suelen emplearse: PCA, *Eigenfaces* (3.1) y redes neuronales (3.2). En la sección 3.3, se atiende a la combinación de los métodos PCA, *Eigenfaces* con redes neuronales. Por último, la sección 3.4, corresponde a la implementación realizada para el entorno de MATLAB, y así alcanzar el objetivo de la práctica.

[2] Circuito integrado de propósito específico.

3.1. Métodos PCA, Eigenfaces.

La idea de aplicar análisis de componentes principales, en adelante PCA, para representar imágenes de caras en una dimensión baja fue introducida por primera vez por los investigadores Sirovich y Kirby de la Universidad de Brown (EE.UU.), en 1987 [Sirovich et al. 87]. Comenzando con un conjunto original de imágenes de caras calculaban el mejor sistema de coordenadas para comprimir la imagen, donde cada coordenada es una imagen que ellos llaman *eigenpictures*[3], imágenes propias. Esta idea fue tomada poco después por los investigadores Turk y Pentland del MIT [Turk et al. 91], quienes desarrollaron el método de reconocimiento de los investigadores Sirovich y Kirby [Kirby et al. 90] y lo reformularon con la denominación *eigenfaces* indistintamente, lo cual no es del todo correcto. El método PCA[4] es un método general de análisis de datos, por otra parte *eigenfaces* es un método de reconocimiento que utiliza PCA con alguna variación:

Sea una imagen de una cara dada por una matriz *MxN*, puede considerarse como un vector de dimensión *MxN*, de manera que una imagen típica (pequeña) de 100x70 píxeles será un vector de dimensión 7000, o de forma equivalente, un punto en un espacio de 7000 dimensiones. La idea de PCA es encontrar la base que mejor expresan la distribución de las imágenes de las caras dentro del espacio completo. Estos vectores describen la base del subespacio de las imágenes de caras, el espacio de las caras (*eigenfaces space*). Cada vector de dimensión *d=MxN* describe una imagen *MxN*, y es una combinación lineal los vectores base del subespacio. Como estos vectores son los vectores propios de la matriz de covarianza correspondiente al espacio original de las imágenes, y como son parecidas a una cara, se les llama *eigenfaces*.

[3] El término *eigen* lo introdujo por primera vez en este contexto David Hilbert en 1904, significa característico o individual en alemán. Los *eigenvectors* son los vectores propios.
[4] Del mismo modo, los trabajos –independientes entre sí– de los investigadores Kari Karhunen (en 1946) y de Michel Loève, le dieron uno de sus nombres más conocidos: <<transformada KL>> o TDKL, transformada discreta de Karhunen-Loève.

Sea el conjunto de imágenes de caras x_1, x_2, x_3, …, x_m (considerando vectores columna de dimensión *d* podemos construir la matriz X de dimensiones *dx*m). La media del conjunto (o la cara media) se define como:

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

Donde A sería la matriz X normalizada (a cada x_i columna se le ha restado la media). La matriz S tiene una dimensión de *dxd*, lo que hace de extraer sus vectores y valores propios una tarea computacionalmente prohibitiva para imágenes de un tamaño normal. Si el número de puntos en el espacio de imágenes es menor que la dimensión del espacio *m* < *MxN*, habrá como mucho *m-1* vectores propios significativos. Puede resolverse el problema tomando apropiadamente combinaciones lineales de las imágenes. Si en lugar de calcular AA^T consideramos v_i los vectores propios de $A^T A$ (y α_i sus valores propios) resultará que:

$$AA^T A v_i = \alpha_i v_i$$	(2)

Premultiplicando a ambos lados por A, se tiene:

$$A^T A v_i = \alpha_i A v_i$$	(3)

Donde Av_i son los vectores propios de $S = AA^T$. Partiendo de este análisis, se construye la matriz $A^T A$ de dimensión *mxm* y se encuentran los *m* vectores propios.

Según [Turk et al. 91], un algoritmo típico de reconocimiento de caras utilizando las *eigenfaces* sería:

Paso 1: Tomar un conjunto inicial de imágenes (el conjunto de entrenamiento). Este conjunto debería incluir un número de imágenes para cada persona, con variaciones en la expresión y en la iluminación.

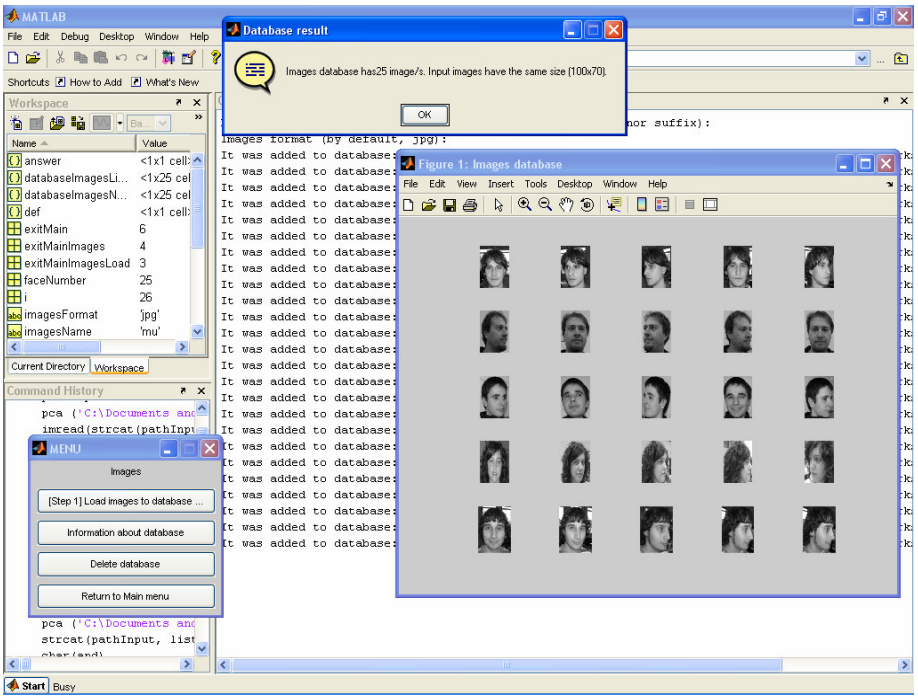

figura 2. Base de datos del Departamento de Electrónica (Universidad de Mondragón).

Paso 2: Calcular la matriz *mxm* normalizada y encontrar sus vectores y valores propios. Elegir los *m'* vectores propios con los valores propios asociados más altos.

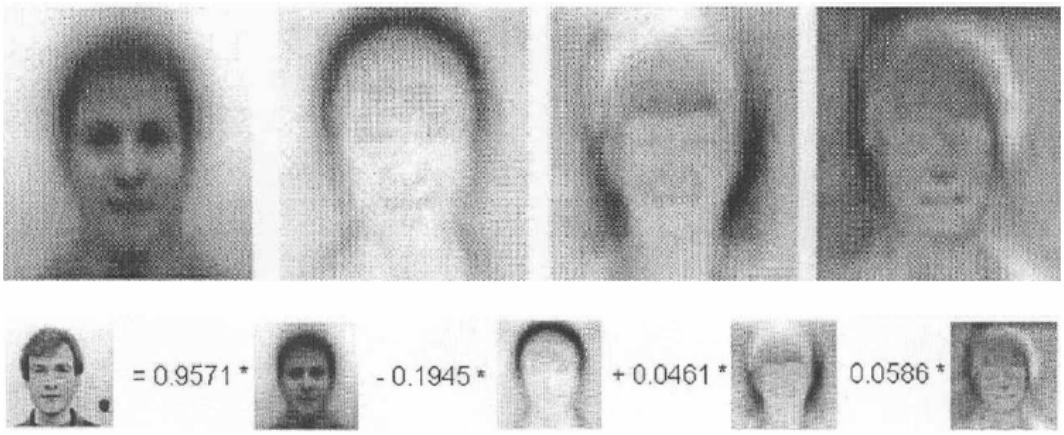

figura 3. Cada cara se podría representar como una combinación lineal de los mejores K vectores propios (fuente: [Turk et al. 91]).

Paso 3: Combinar el conjunto de entrenamiento con los vectores propios para calcular las *m' eigenfaces*. Se ha comprobado que conforme mayor sea la dimensión mejor se reconocerá, pero a partir de un cierto valor, la magnitud de los autovalores expresará la poca información que aportan los vectores asociados y, por lo tanto, la tasa de reconocimiento no se verá aumentada.

Paso 4: Para cada clase (para cada individuo) elegir al menos una (o promediando más de una) de sus imágenes y calcular el vector de clase (proyectándolo en el espacio de caras) Ω_k .

Paso 5: Para cada nueva imagen y calcular su vector $\Omega = \upsilon_k (y - \upsilon)$ y calcular la distancia a cada vector de clase.

Uno de los objetivos es encontrar una proyección lineal de las imágenes de las caras en un espacio menor de características insensible a las variaciones de la luz y la expresión facial. Sin embargo, al utilizar PCA para reducir la dimensión se mantienen variaciones indeseadas debidas a cambios en la luz y en la expresión facial. En [Belhumeur et al. 96] se insiste en que las variaciones entre las imágenes de la misma cara debidas a la dirección del punto de vista y la luz son siempre más grandes que las variaciones en la identidad de la cara.

3.2. Redes neuronales.

Como hemos visto en la anterior sección, una propiedad de los vectores propios es que cada uno de ellos tiene un valor propio asociado. Más importante, los vectores propios con valores propios más altos proporcionan más información sobre la variación de la cara. Después de que las *eigenfaces* se extraen de la matriz de covarianza (A) del conjunto de caras, cada cara se proyecta en el espacio de las caras (*eigenfaces space*) y se representa como una combinación lineal de *eigenfaces*, donde se obtiene un descriptor correspondiente a un punto dentro del espacio de dimensiones con los *eigenfaces* como ejes. Si nosotros usamos todas las *eigenfaces* para representar las caras, ésas en el conjunto de imágenes inicial, se pueden reconstruir completamente –véase figura 4–.

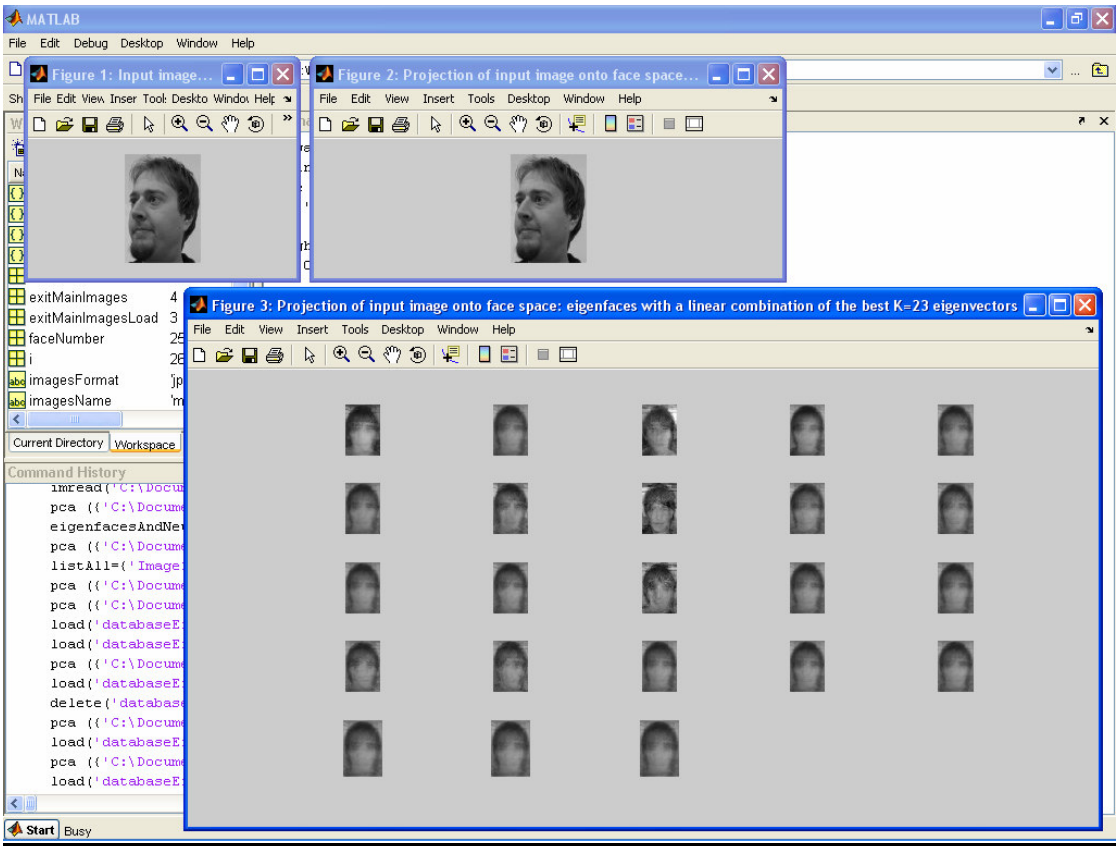

figura 4. Descriptor correspondiente a un punto dentro del espacio de dimensiones con los *eigenfaces* como ejes.

De acuerdo con la literatura, está claro que nosotros deberíamos usar *eigenfaces* con valores propios más altos para reconstruir las caras porque proporcionan mucha más información sobre la variación de la cara. Nosotros usamos las primeras K (a determinar por el usuario a través de la interfaz gráfica) *eigenfaces*. La cara que intentamos reconocer se proyecta sobre las primeras K *eigenfaces*. Ello produce una nueva descripción de la cara de sólo K números reales.

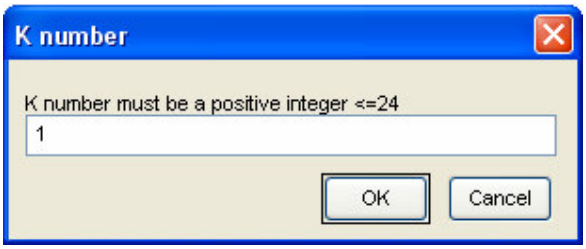

figura 5. Descriptor de la cara de tamaño K.

Debido a que la proyección sobre el espacio de las caras describe la variación de la distribución de caras, es natural tender a usar estos nuevos descriptores de caras para la clasificación. Las caras se reconocen comparando los nuevos descriptores de caras con la base de datos de caras que se ha codificado de la misma manera. Un enfoque para encontrar el patrón de caras es calcular la distancia Euclidea entre el descriptor de la imagen de entrada y cada modelo de cara conocido en la base de datos [Turk et al. 91].

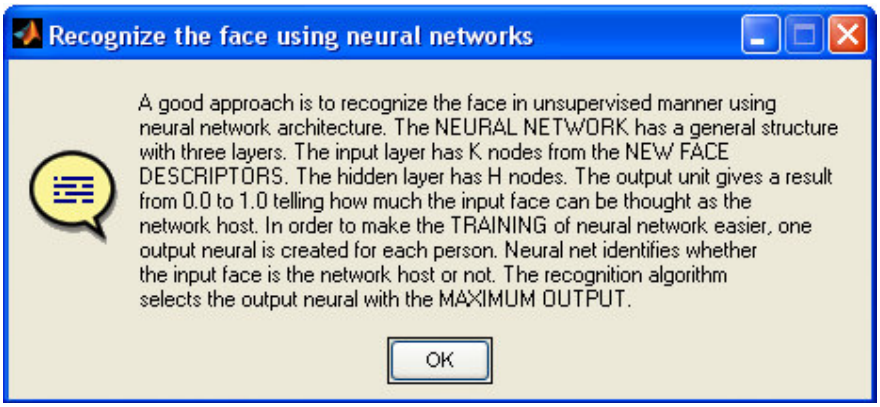

figura 6. *Recognize the face using neural networks*.

En [Jiang 01] encuentran que un mejor enfoque es reconocer la cara de un modo no supervisado utilizando una arquitectura basada en redes neuronales: se coleccionan típicas caras de un individuo, se proyectan en un espacio de las caras y la red neuronal aprende cómo clasificarlas con el descriptor de caras como entrada de la red. Para nuestro caso, la red neuronal tiene una estructura *feed-forward* con tres capas[5]. Esto es, la red está compuesta por una capa de nodos de entrada, una capa de salida, y un número de capas ocultas. La capa de entrada tiene K nodos cuyos valores son los nuevos descriptores de caras. La capa oculta viene dada por 5 nodos. Si bien, se puede determinar a petición del usuario a través de la interfaz gráfica (GUI). Para cada nodo de salida, se da un resultado de 0.0 a 1.0 indicando la posible pertenencia de la cara a reconocer con respecto a un individuo registrado (a través de su nombre). Para hacer la red neuronal más fácil, una neurona se crea para cada individuo registrado. El algoritmo de reconocimiento selecciona el nodo con máxima salida. Del mismo modo, durante el entrenamiento de esta red se calculan un conjunto de pesos $w_{ji,}$ de un conjunto de ejemplos [Izaguirre et al. 08].

3.2.1 Entrenamiento de la red neuronal.

Después de que la red neuronal se ha establecido (con los valores para K y H, *[Step 1] Initialize the neural network architecture*), el siguiente paso es preparar los ejemplos de entrenamiento (*[Step 2] Prepare the training examples*). En el inicio del entrenamiento, nosotros seleccionamos un número de imágenes de caras de cada persona. Estas caras se usarán como ejemplos positivos para sus propios nodos de salida y ejemplos negativos para otros nodos. Asumimos que se tratan de imágenes siempre vinculadas a caras. Después de que la red neuronal se ha creado, lo ejecutamos sobre nuevas caras de los individuos de nuestra base de datos.

[5] De acuerdo con la literatura, se ha demostrado que tres capas son suficientes para determinar cualquier función de discriminación arbitraria, asumiendo que la red está compuesta por un número de nodos. Dependiendo de la complejidad del problema de clasificación se requerirá una arquitectura de red u otra.

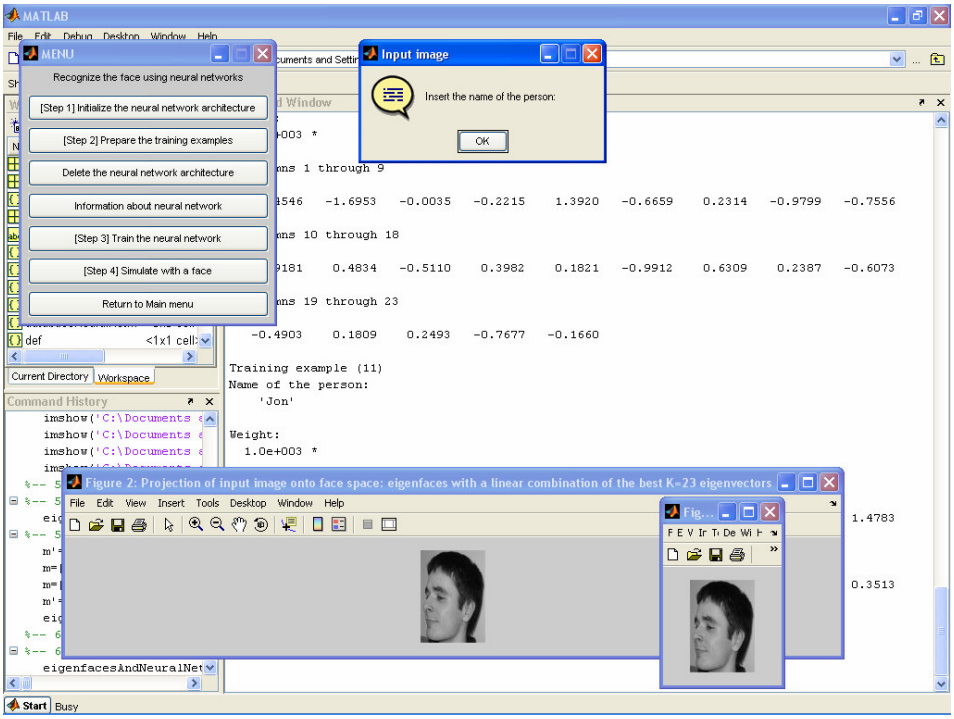

figura 7. *Prepare the training examples*.

En [Jiang 01] cada cara, como resultado del reconocimiento, cae dentro de una de las cuatro categorías que siguen:

- La cara tiene un valor alto en la salida ([0.0 1.0]) sobre su nodo de pertenencia y bajo valor de salida para el resto de nodos, asociado a los otros individuos: ninguna acción a realizar.

- La cara tiene un valor alto en la salida en sus propios nodos de salida y en algunos otros nodos. Estas caras se usarán como ejemplos negativos para otros nodos.

- La cara tiene un valor bajo de salida en sus propios nodos de salida. Las caras se usarán como ejemplos positivos para sus propios nodos y ejemplos negativos para otros nodos.

- La cara tiene un valor bajo en la salida de sus propios nodos. Si ellos tienen un valor alto de salida en alguno de los otros nodos, se incluirá como ejemplos negativos para otros nodos. De lo contrario, se ignorará.

Una vez que se van consiguiendo nuevas caras de los individuos registrados, se podrían añadir como nuevos ejemplos de entrenamiento y así se reentrena la red neuronal. Los errores de reconocimiento tenderán a corregirse y el rendimiento total mejorará. El proceso de entrenamiento de la red podría continuar hasta que no se detecten errores significativos de reconocimiento.

De hecho, podríamos pasarnos horas modificando los pesos tratando de acercarlos a la solución, pero parece más sencillo dejar que los pesos se calculen automáticamente. Para ello, tenemos que entrenar la red (*[Step 3] Train the neural network*). Aparece un gráfico en el que vemos la evolución del entrenamiento. El gradiente es 1,91095e-008. Se podría probar diferentes estructuras para la red neuronal: número de neuronas por capa oculta, función de activación de cada capa (por defecto, se utilizará *logsig*) de las neuronas. La primera se puede modificar a petición del usuario –véase la figura 8–. Si bien, esto último conlleva la eliminación de la arquitectura actual de la red neuronal (con sus ejemplos de entrenamiento, etc.). Para nuestros experimentos, nosotros establecemos a esta variable H el valor 5.

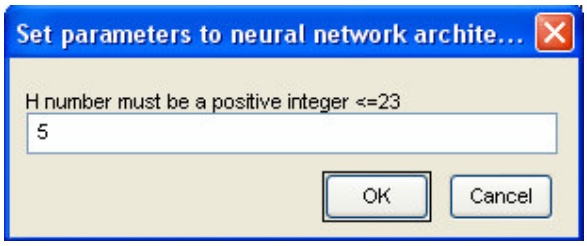

figura 8. *H number.*

Vemos por la salida resultante (*output*) de la siguiente figura 9 que hemos aprendido.

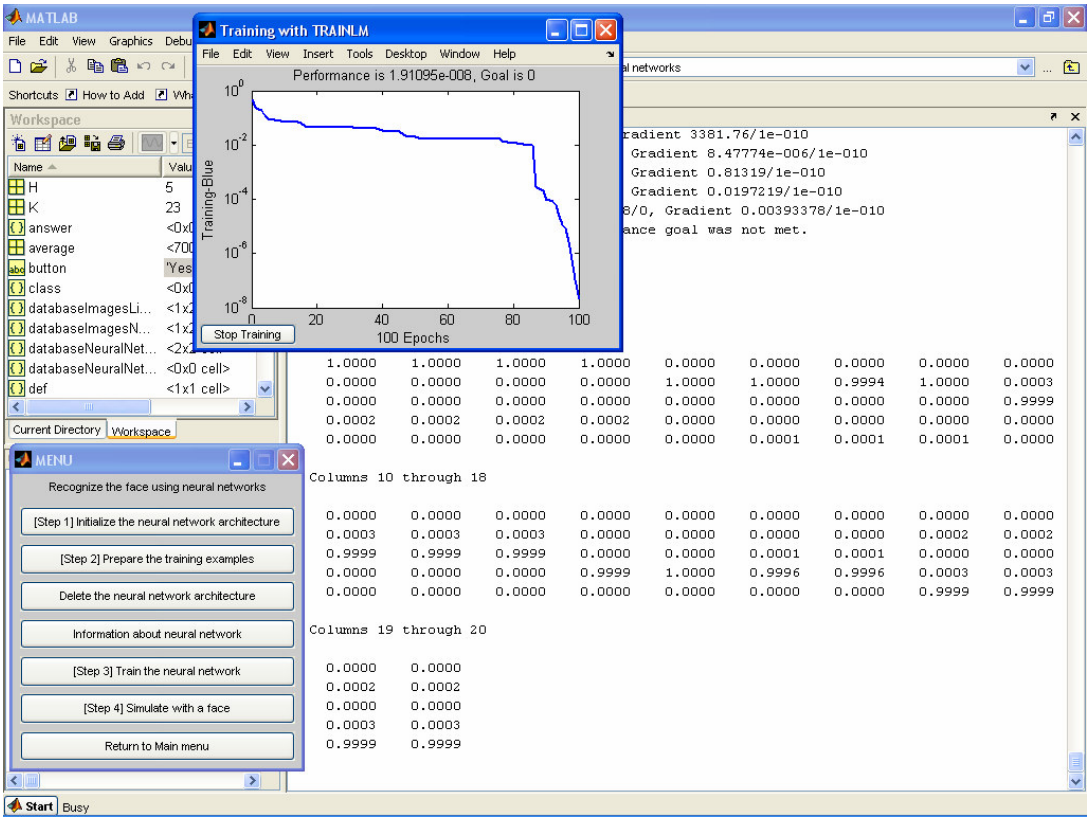

figura 9. *[Step 3] Train the neural network.*

3.3. Combinación de PCA, Eigenfaces con redes neuronales.

Combinando las técnicas comentadas arriba, hemos construido un sistema que reconoce rostros humanos utilizando una base de datos propia, basándonos en el método PCA o *eigenfaces*, y en el uso de redes neuronales como clasificador. El sistema predice la pertenencia de una imagen de entrada a un individuo registrado en la base de datos, siempre que pase un valor umbral predefinido (0.8), notificándolo así al usuario de la aplicación. Los experimentos se han llevado a cabo aproximadamente bajo condiciones de igual iluminación. Las caras utilizadas son básicamente una vista frontal sin cambios de orientación significante. En los test, cada imagen en la base de datos de la Universidad de Mondragón es de 100x70 píxeles. Un ratio de reconocimiento

del 65% se ha alcanzado (*What is the name of this person?. Response: I think that is …*).

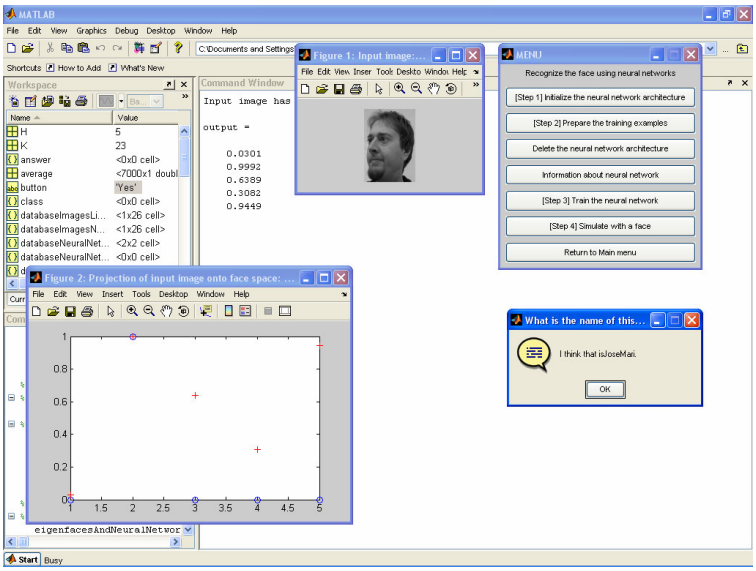

figura 10. *[Step 4] Simulate with a face. Correct.*

Por otro lado, podemos ver la salida con respecto al objetivo en la parte inferior-izquierda de la figura anterior 10 y siguiente 11.

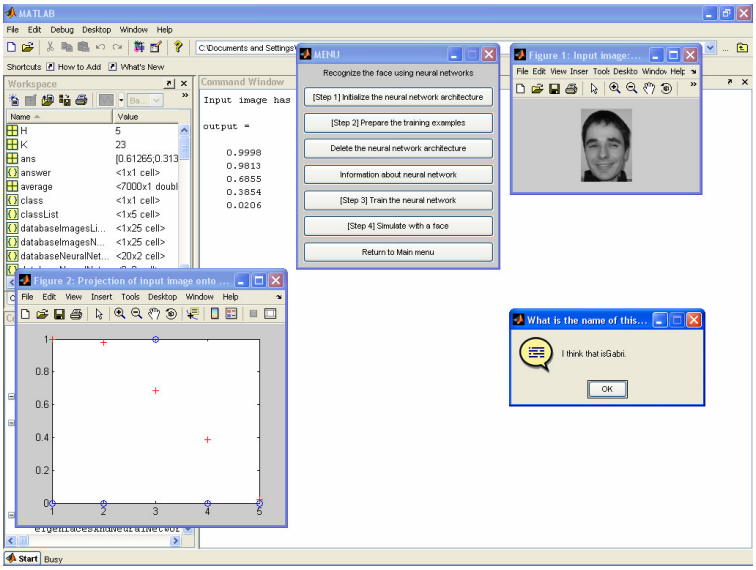

figura 11. *[Step 4] Simulate with a face. Incorrect.*

En el inicio, nuestra base de datos contiene 4 personas. Después de que la base de datos se ha creado, nosotros hemos añadido otra persona más. Se ha demostrado que ésta se puede expandir fácilmente. Nosotros podríamos mejorar el ratio de reconocimiento incrementando el número de ejemplos utilizados en el entrenamiento de la red neuronal. Ésta ha sido de 4 caras por cada individuo, y 5 individuos, dejando una cara restante de cada individuo para simular la red neuronal, y así calcular las salidas.

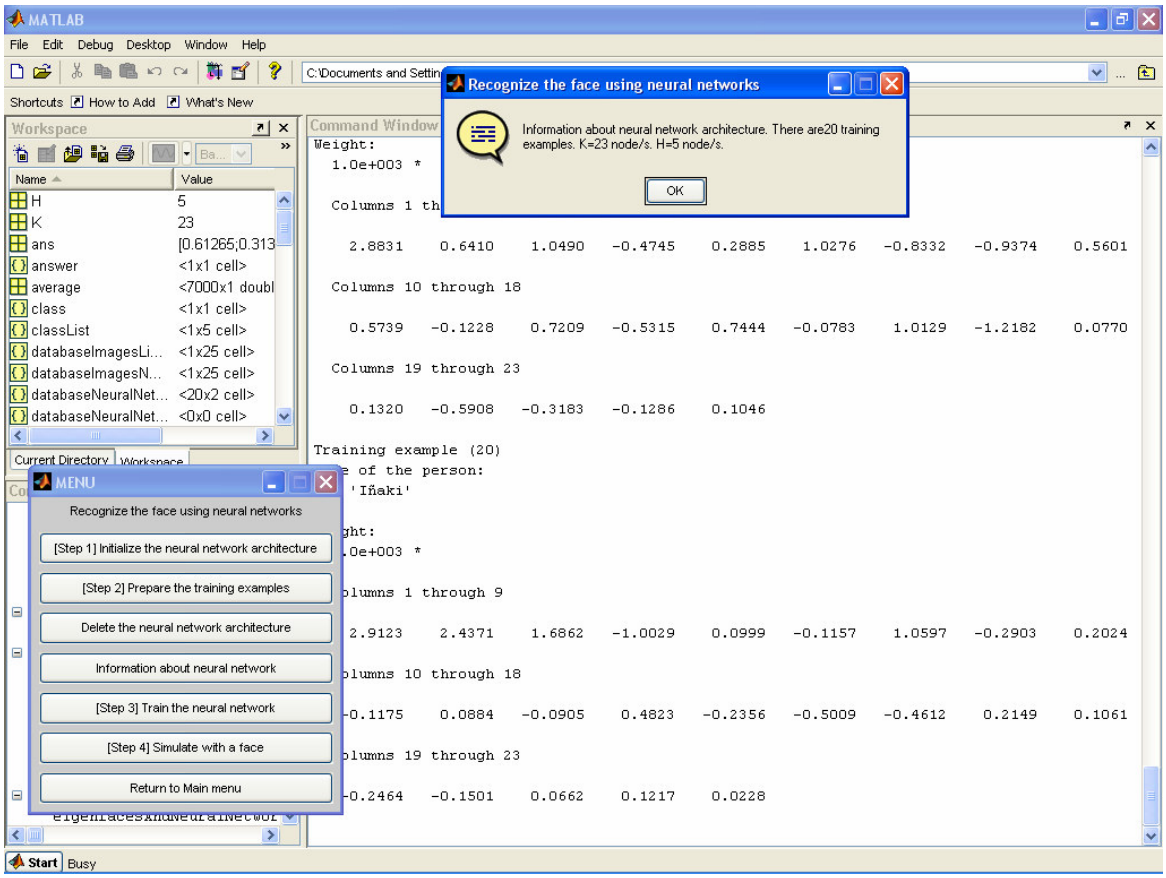

figura 12. *Information about neural architecture. There are 20 training examples. K=23 node/s. H=5 node/s.*

Del mismo modo, existen publicaciones dentro de esta línea de trabajo que usan de media para cada individuo 140 ejemplos de entrenamiento. Así que, aunque fuera del alcance de la práctica, la aplicación permite probar una base de datos bien conocida de entre la comunidad científica, perteneciente a AT&T,

tomadas en laboratorio entre 1992 y 1994, y que cuenta con un mayor número de imágenes de cada individuo, 10, y 40 individuos diferentes[6] (Paso: *[Step 1] Images … → Delete database → [Step 1] Load images to database ... → From selected directory*).

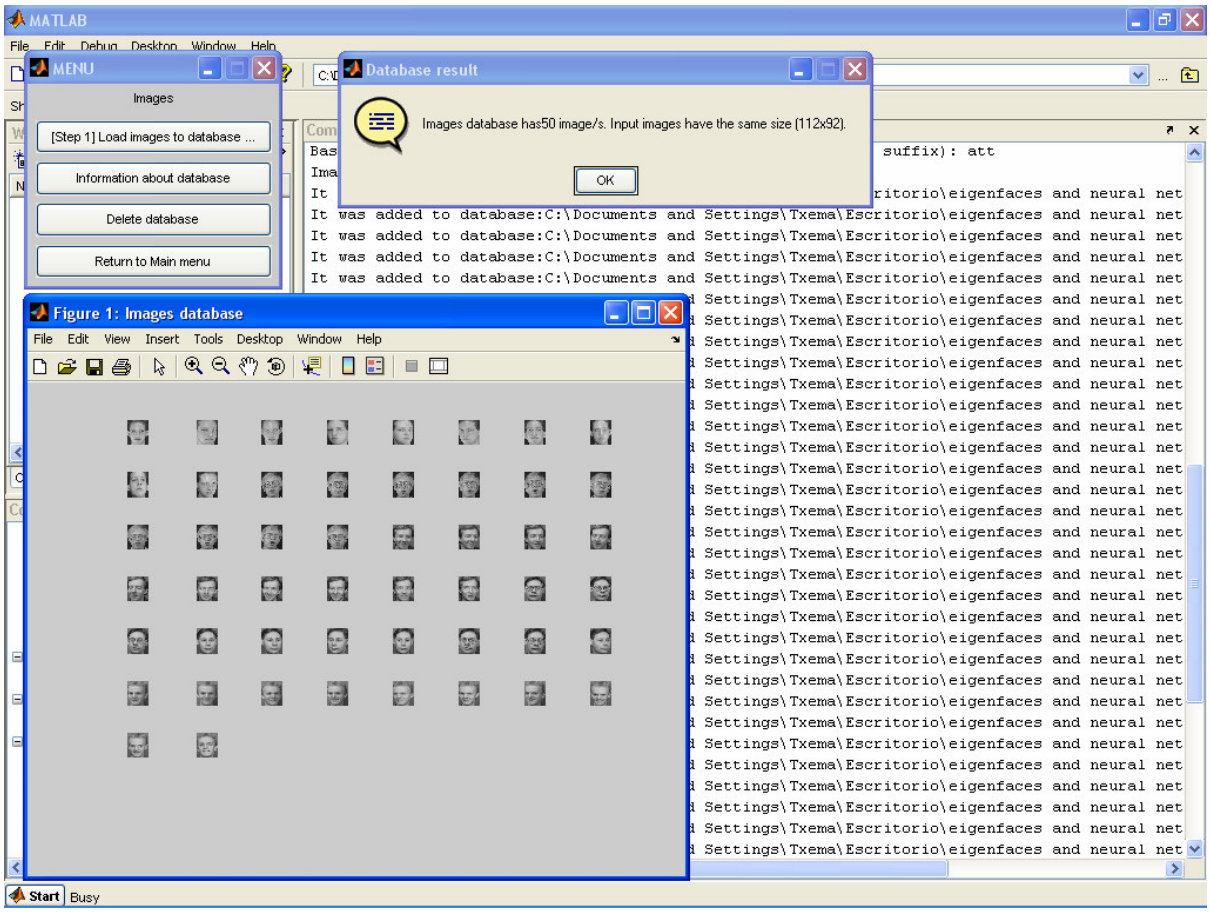

figura 13. Base de datos de AT&T.

3.4. Implementación en MATLAB.

Por último, estos resultados de la anterior sección 3.3 se han obtenido implementando los métodos sobre un entorno de Matlab [MATLAB 08]. El código fuente resultante se organiza en tres ficheros .m:

[6] http://www.cl.cam.ac.uk/research/dtg/attarchive/facedatabase.html

eigenfacesAndNeuralNetworks.m: Corresponde al menú de la aplicación, facilita la entrada (*input*) de valores, y realiza su tratamiento de errores. Del mismo modo, llama a las funciones de: *pca.m* y *neuralnetworks.m*.

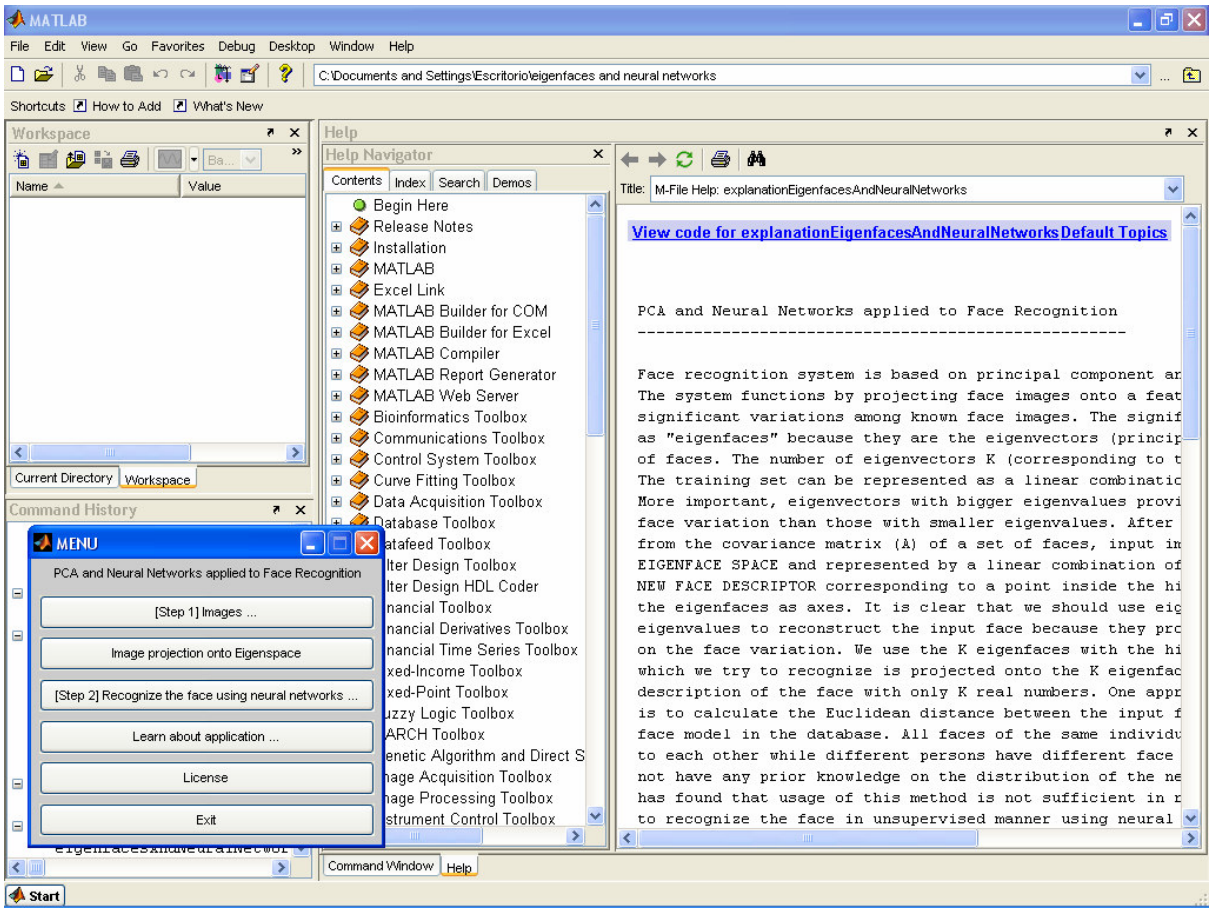

figura 14. *Learn about application …*

pca.m: Implementa la técnica de análisis de componentes principales usado para reducir datos multivariantes para análisis en el campo del reconocimiento de caras, a partir de los argumentos de entrada.

```
>> help pca

  PCA implements the technique (Principal components analysis) used to reduce multidimensional data sets to lower dimensions f
  Syntax:
    PCA(databaseListPath, databaseNameImages, spaceNameImages, K, inputImageRL)
  Input arguments:
    databaseListPath: It is a cell array of n columns. It contains the directory structure (path) to open an image for each el
    databaseNameImages: It is a cell array of n columns. It contains the corresponding image name (see databaseListPath).
    spaceNameImages: It is a cell array of n columns. It contains the Eigenfaces Space (image names).
    K: Number of eigenvectors K (corresponding to the K largest eigenvalues). The training set can be represented as a linear
    inputImageURL: It is a string representation. It contains the directory structure (path) and its corresponding image name
  Output:
    Saves the resultant information (PCA technique), such as representingFaces, eigenFaces, average or dimension of the images

  License: GNU GPL (General Public License). See http://www.gnu.org/licenses/#GPL for details.




>>
```

figura 15. help pca.

neuralnetworks.m: Crea una red neuronal con una estructura *feed-forward* de tres capas y como función de activación de cada capa de las neuronas *logsig*, a partir de los argumentos de entrada.

```
>> help neuralnetworks

  NEURALNETWORKS implements a neural network architecture. The Neural Network has a general feed-forward structure with three
  Syntax:
    NEURALNETWORKS (databaseNeuralNetworks, trainNumber, rangeInputNeuralNet, K, H)
  Input arguments:
    databaseNeuralNetworks: It is a cell array of m rows and n columns. It contains the neural input and its class.
    trainNumber: Number of training examples (see databaseNeuralNetworks).
    rangeInputNeuralNet: It is an array of 1 row and 2 columns which corresponds with the range of input neural net.
    K: Number of input neurals.
    H: Number of nodes in the hidden layer.
  Output:
    Saves the resultant information (neural network architecture), such as databaseNeuralNetworks, trainNumber, valuesInputNet,
    targetOutputNet, K and H in databaseNeuralNetworks.dat.

  License: GNU GPL (General Public License). See http://www.gnu.org/licenses/#GPL for details.




>>
```

figura 16. help neuralnetworks.

4. Bibliografía.

Las siguientes referencias bibliográficas se han utilizado:

[Belhumeur et al. 96]	Belhumeur, P. N., Hespanha, J. P., y Kriegman, D. J. *"Eigenfaces vs. Fisherfaces: Recognition using class especific linear projection"*. Computer Conference on Computer Vision, 1996.

[Cevikalp et al. 05]	Cevikalp, H., Neamtu, M., Wilkes, M., y Barkana, A. *"Discriminative common vectors for face recognition"*. IEEE Computer Society, 2005.

[Izaguirre et al. 08]	Izaguirre, A, y Mendikute, M. *"Sistemas de percepción"*. Apuntes de las clases magistrales, 2008.

[Jiang 01]	Jiang, Q. *"Principal Component Analysis and Neural Network Based Face Recognition"*. PhD Thesis, Department of Computer Science, University of Chicago (EE.UU.), 2001.

[Kirby et al. 90]	Kirby, M., y Sirovich, L. *"Application of the Karhunen-Loeve procedure for the characterization of human faces"*. IEEE Transactions on Pattern Analysis and Machine Intelligence, 12(1), 1990.

[MATLAB 08]	MATLAB – The Language of Technical Computing. *"MATLAB Help"*. The MathWorks, Inc., 2008.

[Pentland et al. 94]	Pentland, A., Moghaddam, B., y Starnet, T. *"View-based and modular eigenspaces for face recognition"*. Proceedings IEEE Conference on Computer Vision and Pattern Recognition, páginas 84-91, Junio 1994.

[Prasanna et al. 05]	Prasanna, C. S. S., Sudha, N., y Kamakoti, V. *"A principal component neural netowork-based face recognition system and ASIC implementation"*. VLSI Design, 2005. 18th International Conference.

[Sirovich et al. 87]	Sirovich, L., y Kirby, M. *"Low-dimensional procedure for the characterization of human faces"*. Journal of the Optical Society of America A, 4(3), 519-524, 1987.

[Turk et al. 91]	Turk, M., y Pentland, A. *"Eigenfaces for recognition"*. Journal of Cognitive Neuroscience, vol. 3, número 1, páginas 71-86.

5. Anexo.

5.1. Tabla de figuras.