Prácticas de Visión Artificial

Práctica 5: Detección y reconocimiento de caras

En esta práctica inicialmente vamos a trabajar con un detector de caras, y finalmente vamos a ver cómo podemos reconocer caras en imágenes.

Los temas principales que necesitaremos son:

1) Face Detection.
2) Face Recognition.

Para completar la práctica es necesario conocer y aplicar los conceptos básicos de detección y reconocimiento de caras. Para esto, consultar el material de teoría.

5.1 Características de Haar y clasificación

El primer paso para la detección de caras y de cualquier objeto en general es la descripción de las imágenes con características. Una de las más utilizadas en el caso de la detección de caras son las Haar-like features o características de Haar.

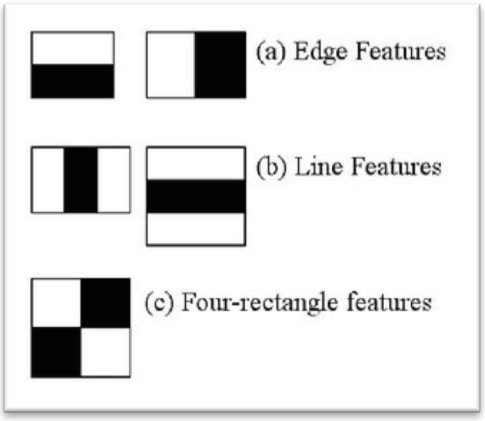

Figura 1: Haar-like features

Son una discretización de los filtros de Haar, muy parecidos a los utilizados en la práctica P4, para describir y detectar texturas. Cada característica está formada por una región excitadora (blanca) y una inhibidora (negra), y el resultado de la característica se obtiene calculando el valor de todos los píxeles de cada una de las regiones por separado y restando el total de la región inhibidora del total de la excitadora.

Cuando tenemos el valor de la característica, el siguiente paso es convertir este valor en una decisión (clasificar); en el caso de la detección de caras, decidir si ese valor corresponde a una cara (1) o no (-1). Para hacer esta decisión, hay que elegir un valor de corte o umbral (threshold), a partir del cual decidiremos si una imagen corresponde a una cara o no de la siguiente forma:

$$h(f, thr, pol) = \begin{cases} +1 & pol * f \geq pol * thr \\ -1 & pol * f < pol * thr \end{cases}$$

donde ***f*** es el valor de la característica, ***thr*** el valor de corte y ***pol*** el valor de polaridad que permite decidir si los valores de las caras son los que están por encima (+1) o por debajo (-1) del valor de corte. Estos clasificadores h(f, thr, pol) basados en una característica y un valor de corte se denominan detectores débiles. Generalmente un clasificador débil no es suficiente, y hay que combinar varios de estos para construir un clasificador fuerte. Si consideramos que la salida de cada clasificador débil es un voto a que la imagen sea cara o no cara, una forma de crear clasificadores fuertes es escoger para cada imagen la clase más votada:

$$H(I) = signo\left(\sum_{i=1..n} h_i(I)\right)$$

donde ***I*** es una imagen y cada $\boldsymbol{h_i}$ es el resultado de aplicar un clasificador débil a esta imagen. La función signo devuelve (+1) si el valor es mayor o igual a cero y (-1) en caso que se le pase un valor negativo.

A partir de esta información, ejecuta la función ***haarFeatureDemo*** facilitada con el enunciado. Al ejecutar se os mostrará una imagen promedio de una cara. Puedes agrandar la ventana, y debes seleccionar los puntos correspondientes a la región excitadora de una característica de Haar de tipo detector de contorno vertical. Se seleccionan los puntos indicados en la Figura 2, por el mismo orden.

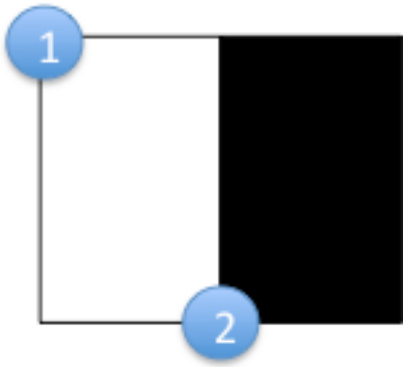

Figura 2: Selección de puntos

Una vez seleccionados los puntos, se genera la característica y se calcula su valor. El código permite seleccionar un número arbitrario de características (por defecto 3), y muestra los resultados de clasificación. Observa el gráfico de la Figura 3.

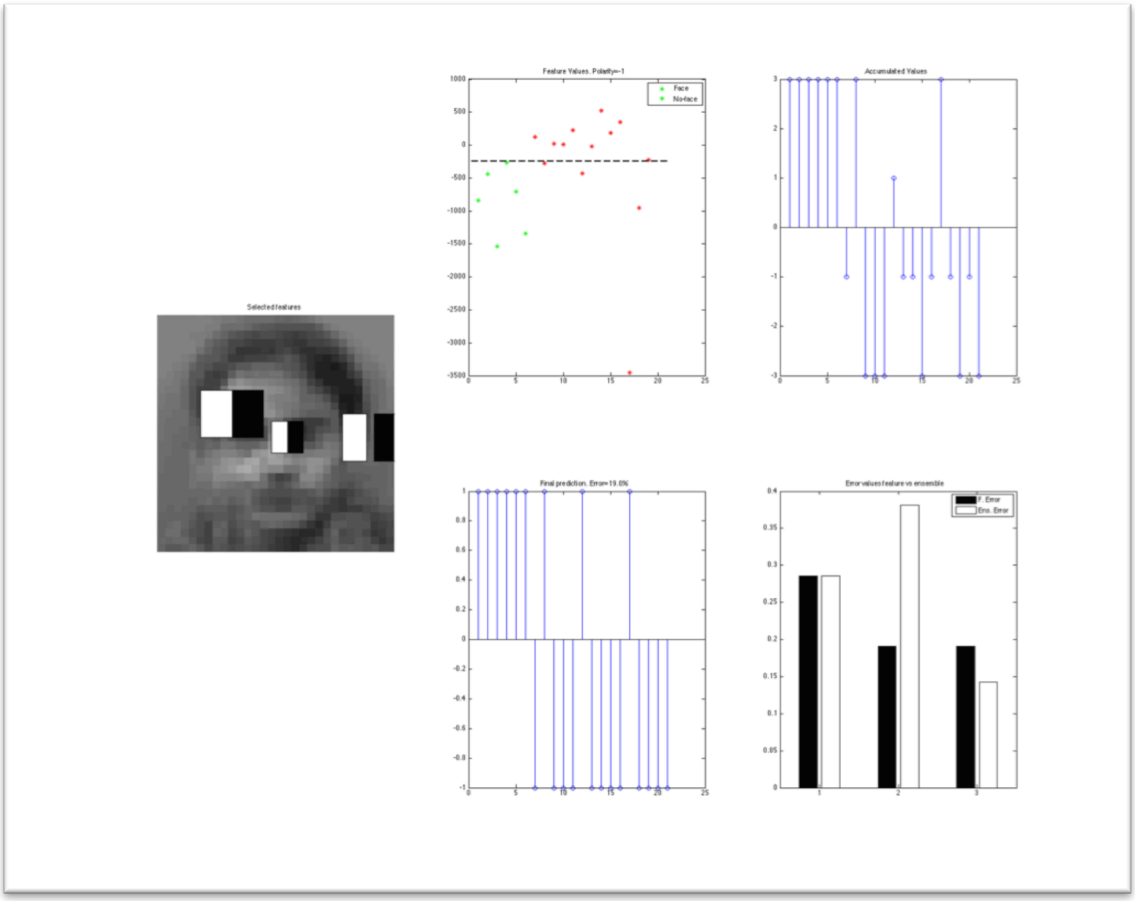

Figura 3: Ejemplo de resultados para 3 características.

A partir del código facilitado, se pide:

- **a)** Selecciona tres características y explica los resultados que aparecen en las distintas gráficas de la figura (adjunta la figura en el documento).
- **b)** A partir de los resultados obtenidos para distintas características, indicar si hay algunos criterios que puedan ser utilizados para seleccionar una buena característica.
- **c)** ¿Qué criterio se utiliza para seleccionar el valor de corte?
- **d)** Combinar características no siempre mejora el resultado. ¿A qué se debe?

5.2 Cascadas de clasificación

Una forma de optimizar el poder discriminativo de los clasificadores es combinarlos en forma de cascada, tal como se muestra en la Figura 4.

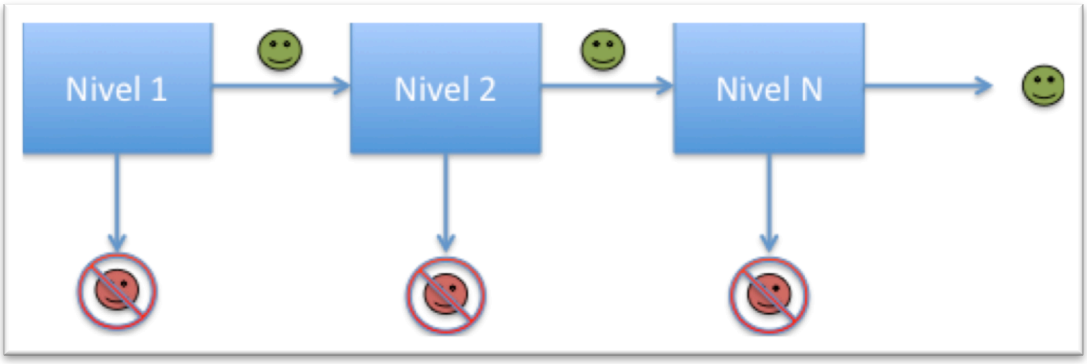

Figura 4: Cascada de clasificadores.

Cada nivel deja pasar sólo las imágenes que considera caras, y el siguiente nivel se especializa en descartar los falsos positivos del nivel anterior.

a) Crea una función ***ej52*** que, utilizando el método ***haarFeatureDemo*** del apartado anterior, entrene una cascada de tres niveles y dos detectores débiles por nivel. Para hacerlo consulta la documentación del método y ten en cuenta:

- El conjunto de datos (X e Y) se deben crear en el primer nivel y reutilizar en los siguientes niveles.
- El primer nivel no tiene predicciones previas.

Muestra el gráfico del error de la predicción final para cada nivel de la cascada. Debe tener un aspecto como el de la Figura 5.

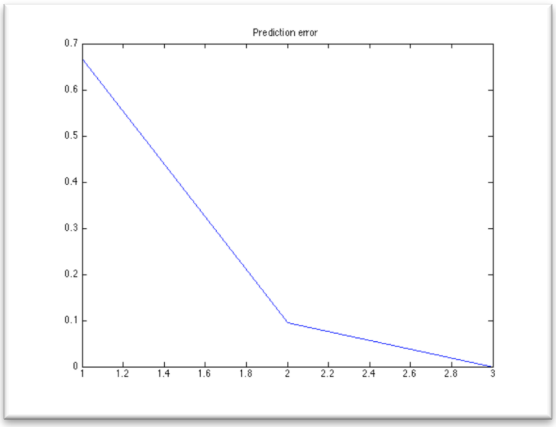

Figura 5: Evolución del error

b) Repite el experimento anterior con dos configuraciones:

a. Un nivel de cascada con 3 detectores débiles.

b. Tres niveles de cascada con un detector débil en cada nivel.

Intenta seleccionar las mismas 3 características en los dos experimentos. ¿El valor del error es el mismo en ambos casos? Aunque las características

sean las mismas, ¿Los detectores débiles son los mismos? Comenta los resultados obtenidos.

c) **[OPCIONAL]** Añadir más caras a la base de imágenes y observar si el resultado de la detección de caras mejora.

5.3 Detección de caras

Los ejercicios anteriores sirven para entender el funcionamiento de los detectores de caras basados en cascada de clasificadores y características de Haar. El proceso que has seguido es una simplificación del proceso de aprendizaje del detector. Los puntos importantes a tener en cuenta son:

- Se entrena con un tamaño de referencia (en el caso anterior de 30x30), y los detectores débiles se definen a partir de las coordenadas de las características dentro de este tamaño de referencia y los valores de corte.
- Un detector es una cascada (conjunto) de niveles (stages) con varios detectores débiles por nivel.

En la realidad, se requiere un método que seleccione las características de forma automática y teniendo en cuenta cuál es la aportación de cada característica a las seleccionadas anteriormente. Además, se requiere un número de ejemplos de caras muy elevado para que el detector pueda generalizar, y añadir nuevos ejemplos de imágenes de no caras a cada nivel.

En este apartado utilizaremos un detector de caras ya entrenado, cuyo código está en la carpeta ***ViolaJones*** del material facilitado con el enunciado, y veremos su funcionamiento. Crear un fichero ***ej53*** para la solución, e incluir la siguiente línea de código al principio del mismo para que Matlab encuentre las funciones del detector:

addpath('ViolaJones','ViolaJones/SubFunctions');

a) Dentro de esta carpeta veréis que hay otra carpeta ***HaarCascades***, que contiene un fichero XML con la cascada de detección ya aprendida. Si abres el fichero verás que contiene los detectores débiles para distintos niveles, y que el tamaño de referencia es 20x20. El primer detector está basado en la siguiente definición de característica:

```
<feature>
      <rects>
            <_>3 7 14 4 -1.</_>
            <_>3 9 14 2 2.</_>
      </rects>
</feature>
```

Utiliza el método ***getDataBase*** para crear una imagen promedio de las caras a tamaño 20x20, tal como se hace en el método ***haarFeatureDemo***, y dibuja sobre esta imagen la característica representada por estos parámetros utilizando el comando ***rectangle*** (tenéis un ejemplo de uso en

el mismo fichero ***haarFeatureDemo***). ¿Porqué un rectángulo se define con 5 parámetros en este caso? ¿Tiene sentido esta característica?

b) Para optimizar el cómputo de las características de Haar se utiliza una representación de la imagen conocida como *imagen integral*. Utiliza el siguiente código para visualizar una imagen integral:

```
function showIntegralImage(image)
        defaultoptions.Resize=false;
        intImageStruct=GetIntegralImages(image,defaultoptions);
        imagesc(intImageStruct.ii);colormap(jet);
end
```

Muestra la imagen integral para dos o tres imágenes distintas (ver Figura 6) y explica qué tienen en común. Justifica tu respuesta.

Calcula la característica Haar del punto a) a partir de la imagen integral.

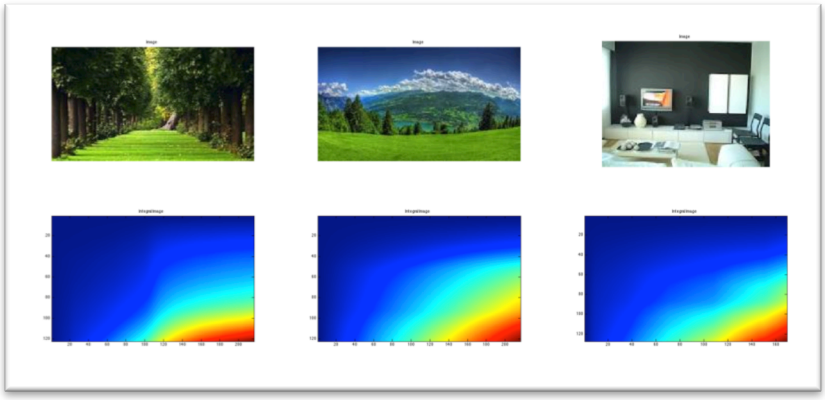

Figura 6: Imágenes Integrales

c) Utiliza los ejemplos 1 y 2 que hay en la documentación del método ***ObjectDetection*** en Matlab (incluido en la carpeta ***ViolaJones*** facilitada con el enunciado) para probar el detector con las imágenes 'testFaces1.jpg' y 'testFaces2.jpg' (ver Figura 7).

Figura 7: Ejemplo Detección de caras

d) Baja de internet 5 imágenes con caras y 5 sin caras y aplica el método. Comenta el resultado.
Nota: Más ejemplos e información se pueden encontrar en la página de Matlab.

5.4 Reconocimiento de caras con el método Eigenfaces

En esta apartado vamos a ver cómo podemos reconocer caras en imágenes. En el material proporcionado con el enunciado encontraréis los ficheros de Matlab ***train.m*** y ***test.m***, que realizan respectivamente el aprendizaje de un modelo de caras y el reconocimiento de caras utilizando el modelo aprendido.

En la fase de aprendizaje (train), las imágenes de caras conocidas se utilizan en el método de análisis de componentes principales, obteniendo la matriz de vectores propios y el promedio de los datos, que permite proyectar las caras en un nuevo espacio.

En la fase de test, tenemos una cara desconocida y queremos saber de quién es. Para esto utilizamos los resultados del aprendizaje para proyectar esta cara y buscar las caras más parecidas, seleccionando la clase de salida con vecino más próximo.

El código proporcionado está incompleto, con comentarios que indican las operaciones a realizar o los valores a calcular donde sea necesario (buscar etiquetas **#to-do**). Completa el código de ambos ficheros, y ejecútalo analizando los resultados proporcionados.

Contesta a las preguntas que se formulan en los comentarios incluidos en el código, pudiendo añadir tus propios comentarios como respuesta.

Finalmente contesta a las siguientes preguntas:

1. ¿Qué sucede si aplicas el reconocimiento sobre el mismo conjunto training_data?
2. ¿Qué sucede si rotas las imágenes de las caras de test?

Entrega de la práctica

Tiempo máximo de entrega: jueves 29 de Diciembre **a las 23:00h**.

Material para entregar: archivo "nombres_apellidos_P5.zip" que contenga:

- Archivos .m de cada ejercicio con las funciones creadas. **El código debe estar comentado en inglés**.
- Fichero .pdf que incluya las respuestas planteadas en los distintos ejercicios.