Prácticas de Visión Artificial

Práctica 5: Detección de caras

(Parte I)

En esta práctica vamos a trabajar con un detector de caras.

Los temas principales que necesitaremos son:

1) Face Detection.

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

Entrega y codificación:

En los ficheros adjuntos encontraréis un fichero template.m. En el se muestra la cabecera e información que debéis utilizar para resolver los ejercicios planteados. Utilizad este fichero como plantilla para la resolución de las prácticas. Además se pide que entreguéis un fichero en formato Word o PDF con el siguiente contenido:

- Cabecera con el grupo de prácticas y los nombres de los componentes.
- Respuesta a las preguntas formuladas en el enunciado.
- Comentarios y observaciones sobre vuestra solución.
- Problemas encontrados y solución adoptada.

4.2 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.

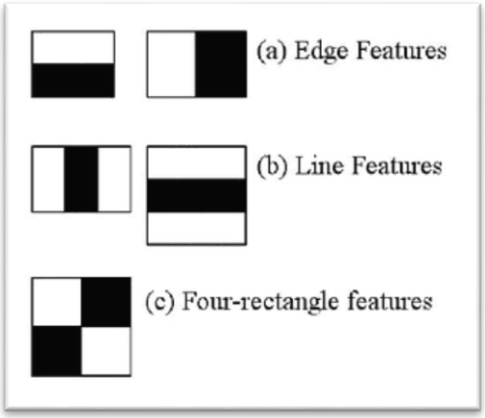

Ilustración 1: Haar-like features

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 de 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 (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 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 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 clasificación 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 cero o positivo y (-1) en caso que se le pase un valor negativo.

A partir de esta información, ejecuta la función ***haarFeatureDemo*** que os hemos facilitado 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 a continuación (por el orden indicado):

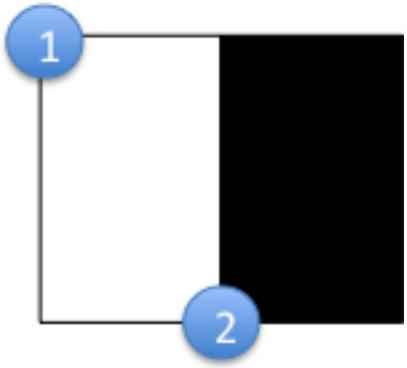

Ilustración 2: Selección de puntos

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. Mira el gráfico que tienes a continuación:

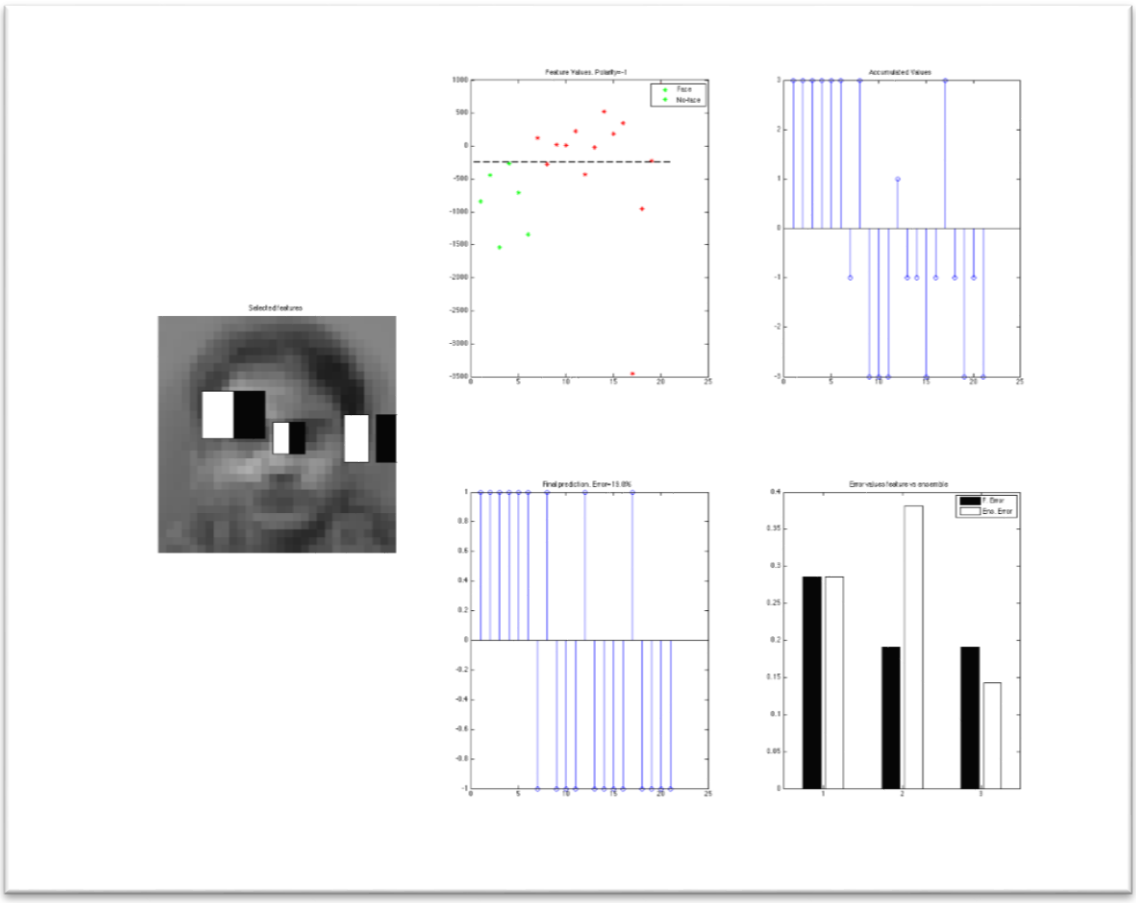

Ilustración 3: Ejemplo de resultados para 3 características.

A partir del código facilitado, se pide:

- **a)** Selecciona tres características y explica los resultados que aparecen en las distintas graficas de la figura (adjunta la figura en el documento).
- **b)** A partir de los resultados obtenidos para distintas características, puedes indicar si hay algunos criterios que puedes utilizar 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?

4.3 Cascadas de clasificación

Una forma de optimizar el poder discriminativo de los clasificadores es combinándolos en forma de cascada, tal como se muestra en la figura siguiente.

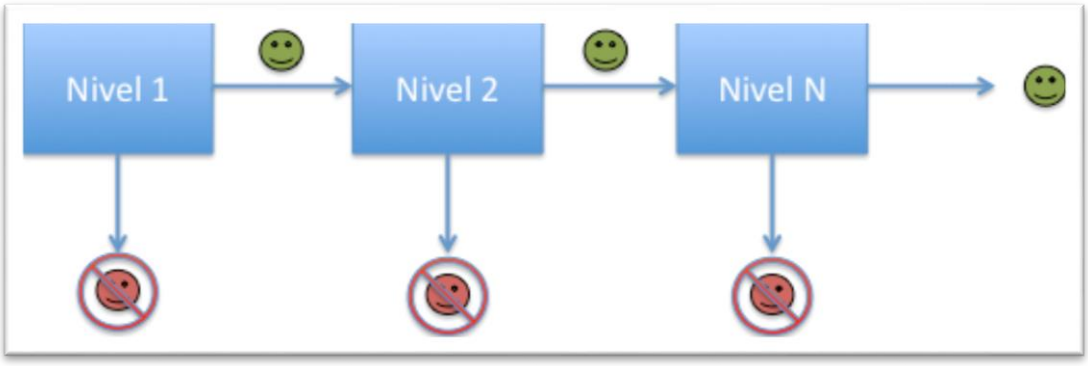

Ilustración 4: Cascada de clasificadores.

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 ex43 que utilizando el método ***haarFeatureDemo*** que has utilizado en el 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 grafico del error de la predicción final para cada nivel de la cascada. Debe tener un aspecto como el que sigue:

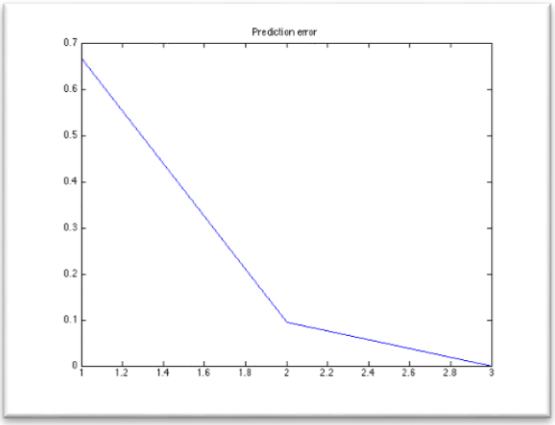

Ilustración 5: Evolución del error

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.

4.4 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:

- 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, el número de ejemplos necesarios para que el detector pueda generalizar, se requiere un conjunto de caras muy elevado, y añadir nuevos ejemplos de imágenes de no caras a cada nivel.

En este apartado utilizaremos un detector de caras ya aprendido, y veremos su funcionamiento. Tenéis el código del detector en la carpeta *ViolaJones* del material facilitado con el enunciado. Crea un fichero ***ex44*** para la solución.

d) 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?

e) 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=GetIntergralImages(image,defaultoptions);
    imagesc(intImageStruct.ii);colormap(jet);
end
```

Para que Matlab encuentre las funciones del detector se requiere añadir sus directorios. Para esto añade al principio del fichero ex44.m la siguiente línea de código:

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

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

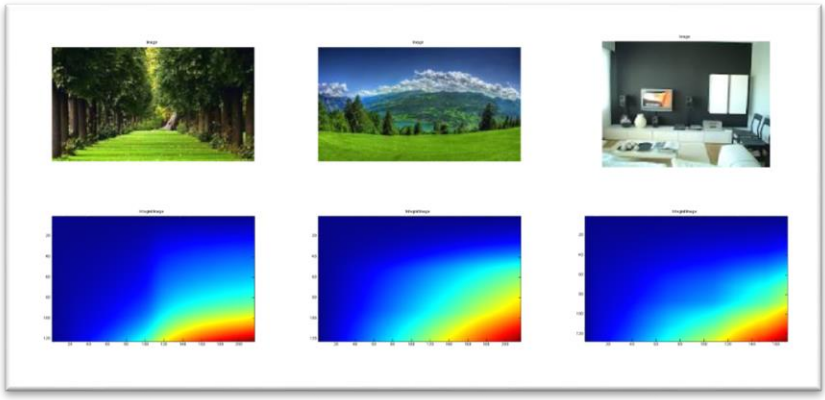

Ilustración 6: Imágenes Integrales

f) Utiliza los ejemplos 1 y 2 que hay en la documentación del método ***ObjectDetection*** en Matlab (lookfor ObjectDetection) para probar el detector con las imágenes testFaces1.jpg y testFaces2.jpg.

Ilustración 7: Ejemplo Detección de caras

Entrega: Esta parte de la práctica junto con la II parte de la práctica 5 (sobre reconocimiento de caras) se ha de entregar por Campus Virtual hasta el 21.12.2015, 12h al mediodía.