"Visión Artificial"

Entrega 2

Universidad Rey Juan Carlos – Ingeniería Informática

Irem Deniz Gunduz (Y4458509-L)

Iván Conde Carretero (03940213-Z)

Lara Camila Sánchez Correa (Y0479214-T)

Tabla de contenido

Apartado 1 2
1.1 Umbralizado 2
1.2 Contornos 3
1.3 Filtrar 4
1.4 Inconvenientes 4
Apartado 2 4
2.1 Umbral, contornos y coordenadas. 5
2.2 Vector de características 5
2.3 LDA 6
2.4 Clasificador 6
2.5 Output 7
Estadísticas 8

En esta entrega usaremos los conceptos aprendidos en la práctica anterior para extender nuestros conocimientos en el campo de detección de patrones, en concreto, detección de la matrícula de los vehículos según su frontal.

La práctica se compone de tres partes principales:

Apartado 1

En este apartado realizaremos la detección de los caracteres.

Empezaremos en el método main, el cual cargará las imágenes del testing y del ocr. Dicha carga se realiza en nivel de gris, poniendo un 0 en el segundo parámetro de la función cv2.imread (mismo método usado en la práctica anterior).

Al final tendremos una estructura donde en cada posición tendremos un duo de imagen a color e imagen en escala de grises.

```
def main():
    cargar_imagen(imagenes_testing, "./testing_full_system")
    cargar_imagen(imagenes_testing, "./testing_ocr")
    infoMatricula = localizar_matricula(imagenes_testing)
```

Cargando las imágenes en la misma estructura

Dicha estructura: **imágenes_testing**, será pasada como parámetro a la función **localizar_matricula.**

La idea principal del método es detectar los diferentes trozos de cada matricula. Lo primero que se ejecutará será un bucle for que recorra la estructura que le hemos pasado.

Usaremos la imagen en escala de grises en el método **detectMultiSacale**, cuyo resultado será la imagen por procesar.

```
for i in date_image:
    im = i[0]
    imc = i[1]

    imagenP = matricula_cascade.detectMultiScale(im, 1.1, 5)
```

Imagen en gris

Escala y vecinos

1.1 Umbralizado

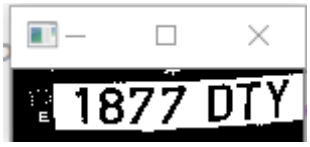
Primero **recortaremos** la imagen en gris.

Imagen en gris:

Imagen recortada resultante de la matrícula:

Usaremos el **método cv2.threshhold con la binarización de Otsu**:

```
ret, th3 = cv2.threshold(imagen_GrisRecortada, 0, 255, cv2.THRESH_OTSU)
```



1.2 Contornos

Con el método **cv2.findContours** encontraremos los contornos pasándolo como parámetros la imagen umbralizada.

```
contours, hierarchy = cv2.findContours(th3, cv2.RETR_TREE, cv2.CHAIN_APPROX_SIMPLE)
```

Imagen umbralizada

Modo y método

Creando dos estructuras, los contornos y la jerarquía.

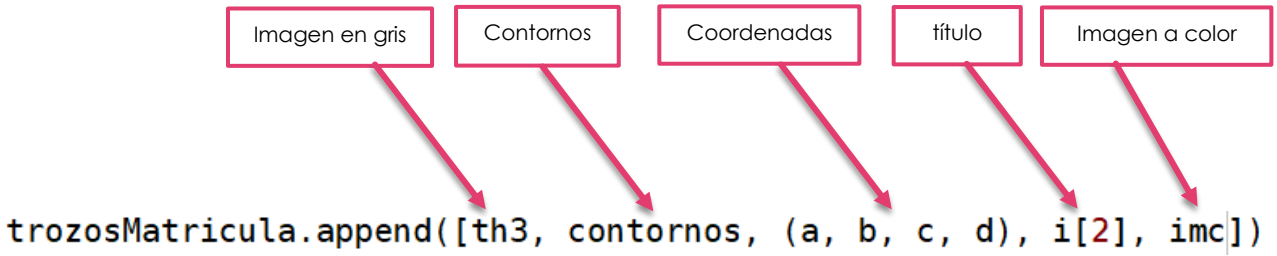
En la estructura de contornos iremos añadiendo aquellos que vayamos encontrando en cada imagen usando el método **cv2.boundingRect** (sacando las coordenadas).

1.3 Filtrar

Filtraremos las coordenadas detectadas: la altura debe que ser mayor que la anchura y se eliminan los que son demasiado cortos o demasiado finos.

```
if d > 1.1 * c and c > 5 and d > 12:
    contornos.append(cor)
```

Finalmente, el método **localizar_matrícula** devolverá una estructura que contiene la imagen en gris, sus contornos, sus coordenadas, su título y su imagen normal.



1.4 Inconvenientes

En un primer momento dividimos el problema en tres casos, cuando la matricula era localizada, el coche era localizado y cuando ni la matricula ni el coche eran localizados.

En el segundo caso, localizamos todos los contornos, y lo filtrábamos por su tamaño y área, mediante *cv2.contourArea(contorno)*, obteníamos el área del contorno. Después de tener los contornos, intentábamos obtener los contornos alineados, para ello usábamos el algoritmo de RANSAC, guardábamos las coordenadas X e Y en dos listas, una para cada eje, y después mediante la biblioteca **sklearn**, usábamos *LinearRegression*() para realizar el ajuste de mínimos cuadrados. Para acabar obteníamos los contornos que estaban cerca de la recta calculada.

Al realizar el segundo apartado, nos encontramos con una serie de problemas, ya que al ejecutar el programa nos notificaba de unos errores al redimensionar los contornos localizados en este segundo caso. Por falta de tiempo no nos pudimos centrar en resolver el problema, así que eliminamos el segundo caso.

Dicho código se podrá encontrar comentado en el fichero leerCoche.py.

Apartado 2

En este apartado construiremos el sistema que clasifique los caracteres de la matrícula.

Se ha utilizado el conjunto de entrenamiento proporcionado.

Llamaremos al método **Leer_matricula**, el cual recibirá como parámetros la información adquirida en el método anterior y las imágenes cargadas del **testing.**

2.1 Umbral, contornos y coordenadas.

Primero, crearemos el vector de características. Cargaremos y umbralizaremos las imágenes de los caracteres del entrenamiento con el método **cv2.threshold** y sacaremos sus contornos con el método **cv2.findContours.**

```
for caracter in img_training:
    cont1 = cont1 + 1
    # Umbralizar el caracter
    ret, th1 = cv2.threshold(caracter[0], 0, 255, cv2.THRESH_OTSU)

    # Sacar los contornos
    contours, hierarchy = cv2.findContours(th1, cv2.RETR_TREE, cv2.CHAIN_APPROX_NONE)
```

Dentro del bucle, usaremos el método **cv2.boundingRect** para sacar las coordenadas del contorno calculado, añadiéndolo si cumple el filtrado.

A continuación, sacaremos la umbralización del carácter y lo recortaremos.

Seguidamente, redimensionaremos los caracteres con el método **cv2.resize**:

2.2 Vector de características

Creamos el vector de características asociado al carácter de entrenamiento. Agruparemos las características en una matriz, mediante un bucle. Tendremos en cuenta que el carácter final no esté vacío y no pertenezca a las vocales ya que no aparecen en la matrícula, es decir, no se entrenarán.

```
if caracter_redim is not () and not(2500 <= cont1 < 2750 or 3750 <= cont1 < 4000 or 4750 <= cont1 < 5000 or 6250 <= cont1 < 6500 or 7750 <= cont1 < 8000):
    for i in range(caracter_redim.shape[0]):
        for j in range(caracter_redim.shape[1]):
            caracteristicas[0][a] = caracter_redim[i][j]
            a = a + 1
    matrizC.append(caracteristicas)
```

2.3 LDA

Recorreremos toda la matriz que hemos creado (C, de características), que tiene todas las características de cada clase.

Creamos las clases con el método **crearEtiquetas.** Dicho método creará las clases en las que se debe clasificar cada uno de los dígitos de aprendizaje, como tendremos 32 clases ([0-9],[B-Z] sin vocales) habrá 250 imágenes por cada letra o número, por ello, el '0' irá desde 0 a 249 y así sucesivamente.

```
def crearEtiquetas():
    clases = np.zeros((8000), np.uint8)
    i=0
    while i<8000:
        if (0 <= i < 250):
            clases[i] = 0
        elif (250 <= i < 500):
            clases[i] = 1
        elif (500 <= i < 750):
            clases[i] = 2
```

Creamos el **objeto LDA,** llamando al método LDA() de **discrimant_analysis.py.**

Creamos la matriz de proyección LDA con el método **fit,** pasándole como parámetro la matriz de características y el vector de etiquetas.

Lo siguiente será crear un bucle para recortar los dígitos de la matrícula según los contornos, recorreremos cada posición de **infoMatricula** que contiene la imagen en gris, sus contornos, sus coordenadas, su título y su imagen normal de cada vehículo.

2.4 Clasificador

Sacaremos los contornos y los filtraremos como hemos hecho anteriormente y obtendremos las características de cada matrícula con el método **obtenerCarcteristicasMatricula.**

Dicho método umbralizará los caracteres recortados de la matrícula, los recortará y agrupará las características de los dígitos de la matrícula en una matriz, siendo el valor devuelto.

Una vez que tengamos las características, usaremos la predicción de LDA con el método **predict,** haciendo un append del resultado a un vector.

Construiremos la matrícula a partir del vector anterior, con el método **construirMatricula**, el cual traducirá los valores predichos a letras y números para construir la matrícula resultante.

El método de construcción otorgará un valor u otro dependiendo del valor del dígito.

Una vez construida la matrícula, recorreremos cada letra en el vector. Le daremos formato a la fuente con **cv2.FONT_HERSHEY_COMPLEX** y lo mostraremos por pantalla mediante el método **cv2.putText** sobre la imagen del vehículo.

A su vez, imprimiremos por consola los datos obtenidos: el nombre de la imagen procesando y el número de matrícula detectado.

2.5 Output

Como salida tendremos los diferentes vehículos con la detección de la matrícula en un color azul.

```
Imagen  1 : coche.jpg ['ES', '7', '4', '1', '5', 'H', '0', 'H']
Imagen  2 : cocheFrontal.jpg ['6', '2', '4', '7', 'F', 'N', 'S']
Imagen  3 : coches.jpg ['4', 'G', '5', '2', '6', 'H', 'R']
Imagen  4 : coches_2.jpg []
Imagen  5 : sta50002.jpg ['ES', '8', '1', '7', '9', '1', 'H']
Imagen  6 : sta50003.jpg ['ES', '4', '6', '1', '8', 'C', 'W', 'J']
Imagen  7 : sta50003_2.jpg []
Imagen  8 : sta50004.jpg ['1', '1', '4', 'Z', '9', 'B', 'H', 'W']
Imagen  9 : sta50007.jpg ['W', '5', '9', '7', '4', 'C', 'M', 'T']
Imagen  10 : sta50009.jpg ['9', '0', '8', '4', 'B', 'R', 'L']
Imagen  11 : sta50016.jpg ['ES', '7', '0', '5', '2', 'B', 'G', 'K']
Imagen  12 : sta50017.jpg ['ES', '7', '0', '5', '2', 'B', 'G', 'K']
Imagen  13 : sta50079.jpg []
Imagen  14 : sta50079.jpg []
Imagen  15 : sta50091.jpg ['4', '1', '4', 'B', 'W', 'G']
Imagen  16 : sta50092.jpg ['0', '5', '1', '3', 'D', 'G', 'X']
Imagen  17 : sta50133.jpg ['1', '1', '3', '4', '7', '0', 'Y', 'B']
Imagen  18 : sta50135.jpg ['ES', '2', '1', '1', '7', 'C', 'W', 'P']
Imagen  19 : sta50163.jpg ['ES', '2', '9', '3', 'C', 'L', 'C']
Imagen  20 : sta50185.jpg ['3']
Imagen  21 : sta50187.jpg ['2', 'L', 'W']
Imagen  22 : sta50188.jpg ['1', '8', '6', '1', '1', '1']
Imagen  23 : test1.jpg ['9', '3', '2', '6', 'C', 'V', 'Z']
Imagen  24 : test2.jpg ['1', '3', '5', '6', '5', 'G', 'X', 'J']
Imagen  25 : test3.jpg ['W', '8', '7', '1', '2', 'H', 'J', 'G']
Imagen  26 : test5.jpg ['ES', '3', '5', '8', '9', 'B', 'M', 'N']
Imagen  27 : test6.jpg ['1', '9', '9', '7', '8', 'H', 'K', 'G']
Imagen  28 : frontal_1.jpg ['8', '6', '7', 'Q']
Imagen  29 : frontal_10.jpg ['0', '1', '9', '5', '0', 'T']
Imagen  30 : frontal_12.jpg ['9', '7', 'C', 'N', 'W']
Imagen  31 : frontal_13.jpg ['Q', '2', '3', '8', '7', '0', 'H', 'J']
Imagen  32 : frontal_15.jpg ['ES', '1', '5', '1', '2', 'D', 'L', 'C']
```

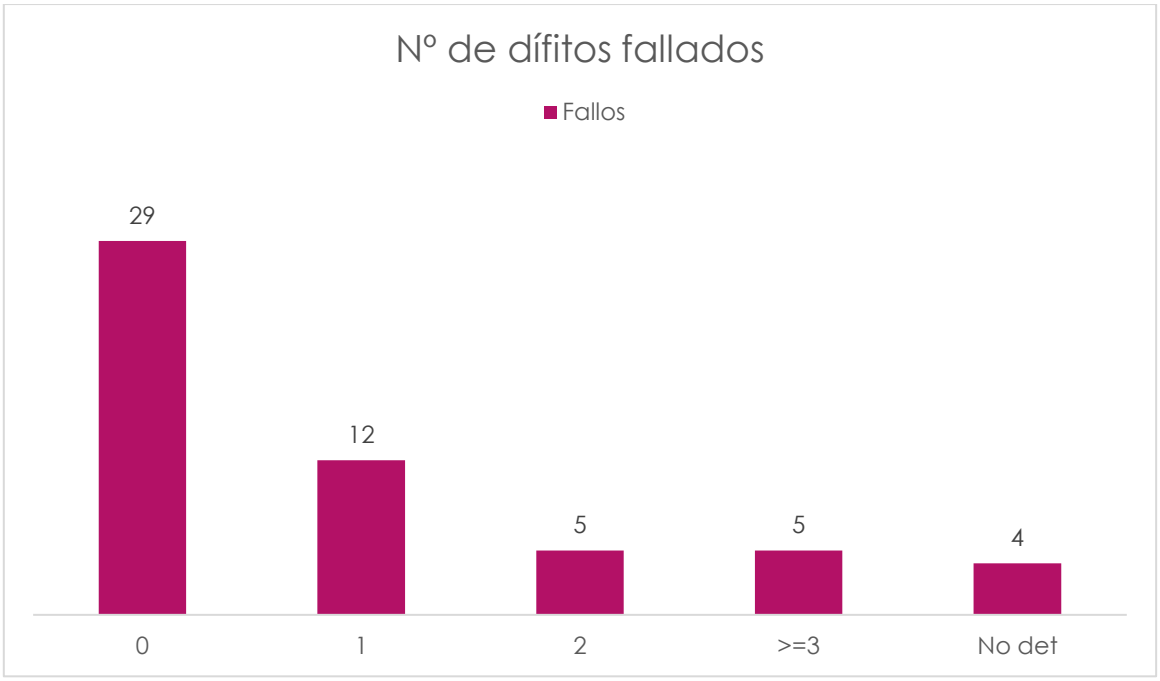
Estadísticas

En la siguiente tabla veremos desglosado la salida del programa, ordenado por orden de aparición: marcado en rojo los caracteres mal detectados.

Nombre imagen	Matrícula	Matrícula detectada	Nº de dígitos fallados
coche	7415 HDH	7415 H0H	**1**
cocheFrontal	6247 FNS	6247 FNS	**0**
coches	4052 GRR	4G52 6HR	**3**
coches_2	**4343 DKF**	No detectada	**NO DET**
sta50002	8179 DSH	8179 1H	**2**
sta50003	4618 CWJ	4618 CWJ	**0**
sta50003_2	**M 0848**	No detectada	**NO DET**
sta50004	1479 BHW	1479 BHW	**0**
sta50007	5974 CMT	5974 CMT	**0**
sta50009	9084 BRL	9084 BRL	**0**
sta50016	7052 BGK	7052 BGK	**0**
sta50017	7052 BGK	7052 BGK	**0**
sta50079	**5991 CYV**	No detectada	**NO DET**
sta50079	**5991 CYV**	No detectada	**NO DET**
sta50091	7414 BWG	414 BWG	**1**
sta50092	0513 DGX	0513 DGX	**0**
sta50133	1347 DYB	1347 OYB	**1**

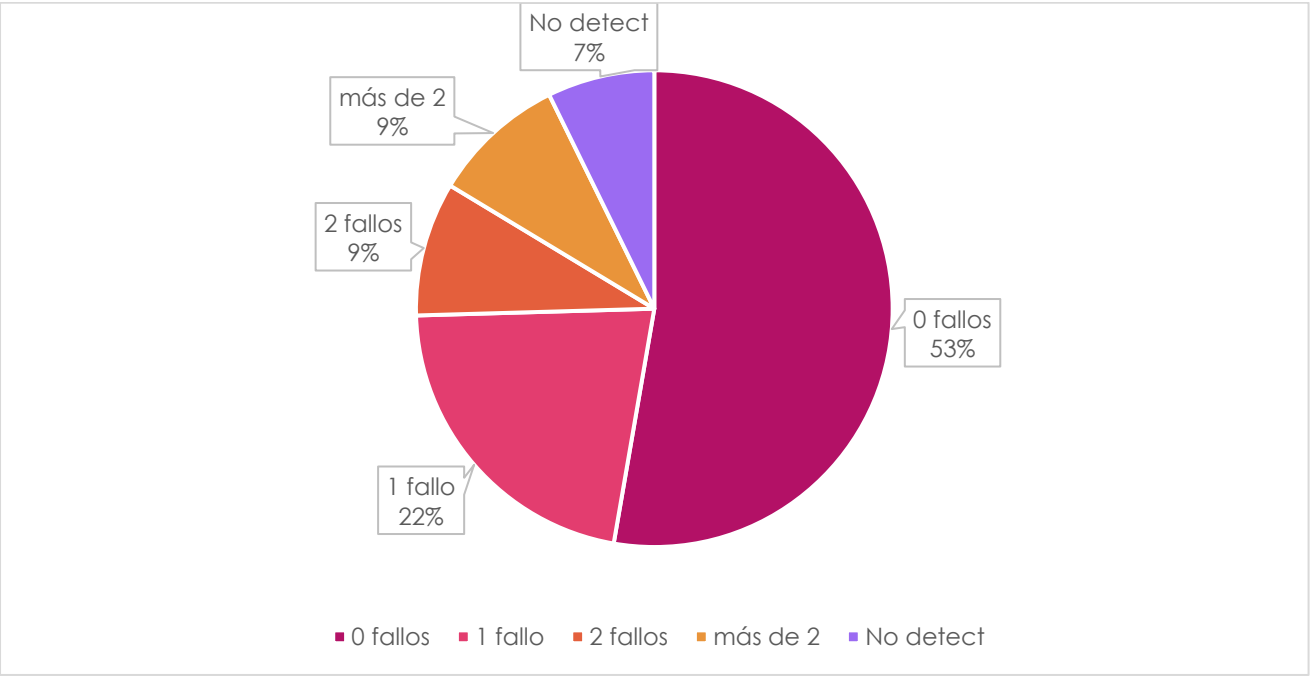
sta50135	2117 CWP	2117 CWP	**0**
sta50163	293**3** CLC	293 CLC	**1**
sta50185	**763****8 DKZ**	3	**6**
sta50187	**861**2 **DR**W	2 LW	**5**
sta50188	861**2 DRW**	8611 1	**4**
test1	9236 CVZ	9236 CVZ	**0**
test2	3565 GXJ	3565 GXJ	**0**
test3	8712 HJG	8712 HJG	**0**
test5	3589 BMN	3589 BMN	**0**
test6	9978 HKG	9978 HKG	**0**
frontal_1	186**7 CGS**	186Z Q	**4**
frontal_10	0195 **CD**T	0195 OT	**2**
frontal_12	9597 CNW	97 CNW	**2**
frontal_13	2387 **D**HJ	2387 0HJ	**1**
frontal_15	1512 DLC	1512 DLC	**0**
frontal_17	0408 FCF	0408 FCF	**0**
frontal_18	1851 **D**GZ	1851 0GZ	**1**
frontal_22	3069 CDF	3069 CDF	**0**
frontal_24	3976 DBT	3976 DBT	**0**
frontal_25	8989 CWL	8989 CWL	**0**
frontal_29	2988 **D**CN	2988 0CN	**1**
frontal_30	6615 DB**D**	6615 DB	**1**
frontal_31	6652 CMC	6652 CMC	**0**
frontal_32	7711 CPS	7711 CPS	**0**
frontal_34	6616 **D**NL	6616 ONL	**1**
frontal_35	2520 CVC	2520 CVC	**0**
frontal_36	1765 FZG	1765 FZG	**0**
frontal_37	35**7**1 CYP	35Z1 CYP	**1**
frontal_39	4951 FT**D**	4951 FT	**1**
frontal_40	9800 CLR	9800 CLR	**0**
frontal_41	3737 GCB	3737 GCB	**0**
frontal_42	1212 **D**M**D**	1212 GM	**2**
frontal_46	3049 CLG	3049 CLG	**0**
frontal_47	1043 **D**ZT	1042 OZT	**1**
frontal_48	1040 DLR	1040 DLR	**0**
frontal_6	6330 CRT	6330 CRT	**0**
frontal_7	1748 **D**J**D**	1748 OJ	**2**
frontal_8	1877 DTY	1877 DTY	**0**

Tenemos 55 imágenes, vamos a ver las estadísticas en profundidad:



Sin embargo, si vemos en profundidad la detección de matrículas podemos ver que existe un problema con la letra 'D' dado que la confunde con otras, si se hubiese detectado correctamente el número de matrículas detectadas correctamente hubiera subido de 29 a 38. Es decir, casi un 70% de detección correcta.

En cuanto a porcentajes tenemos:



Es decir, el 53% de las veces nuestro sistema detectará correctamente, y un 22% con un solo fallo.