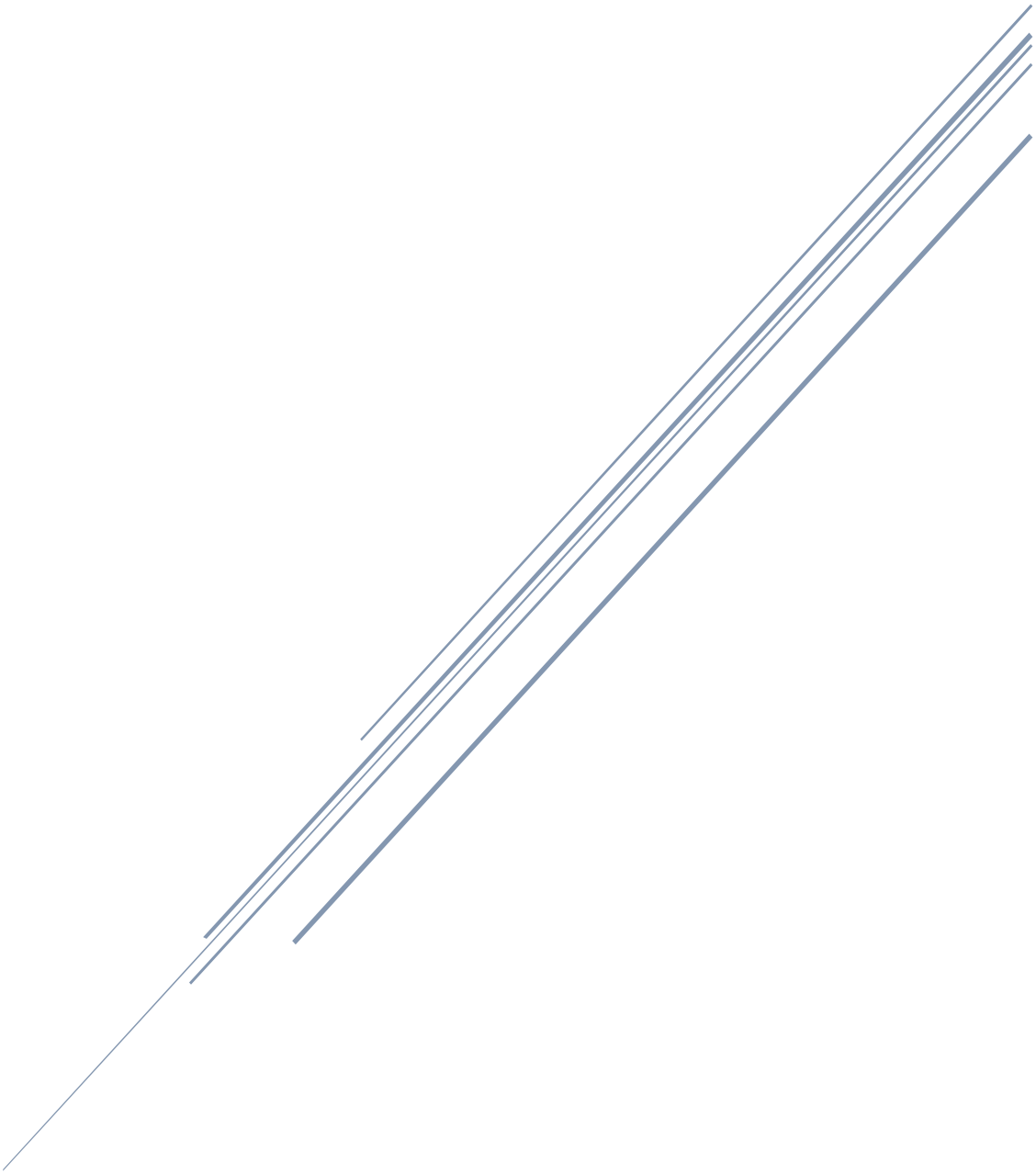
PRÁCTICA 4 – RECONOCIMIENTO DE OBJETOS

Sistemas de Percepción – 4 GIERM



Pablo León Barriga
2021/2022

Introducción

La práctica se ha desarrollado en dos códigos principales, uno para la construcción del clasificador bayesiano mediante el análisis de imágenes de matrículas en las que se conoce de antemano el número presente en las mismas, "**MAIN_pr4_construccion.m**" y otro archivo en el que se validará el clasificador anteriormente construido, además de implementar un clasificador de mínima distancia, aprovechando que ya se dispone de la media de cada una de las características de los números, "**MAIN_pr4_validacion.m**".

Además de esos dos archivos se han desarrollado una serie de funciones para realizar operaciones de obtención de medianas, giro y recorte de las imágenes de entrenamiento y validación, o para el cálculo de las características, por ejemplo.

Archivo MAIN_pr4_construccion.m

Se comienza inicializando algunos valores que serán de utilidad a lo largo del código, como el número de clases, la probabilidad a priori de cada una de esas clases, y el número de características.

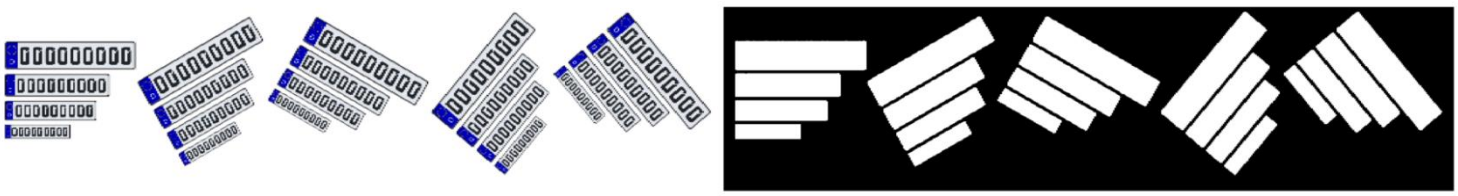
```
%% Practica 4 - Sistemas de Percepcion - Pablo Leon Barriga
clear;close all;clc

%% Variables a utilizar
% Numero de clases, en este caso 10 (10 posibles numeros, 0 - 9).
Nclases = 10;

% Numero de caracteristicas, en este caso 13 (x1 - x13).
Nx = 13;

% La probabilidad a priori de las clases es 1 / Nclases
pC = 1/Nclases;

% Arrancar ventana grafica
pIMSHOW = imshow(logical(zeros(128,64)));
```

A continuación se inicializa un bucle en el que se prepara el clasificador para cada una de las diez clases de las que se obtendrán las características. Se comienza leyendo la imagen correspondiente al número a "entrenar", se pasa a escala de grises y se binariza utilizando un umbral de 245. Además, se realiza una operación de dilatado para obtener las matrículas de forma más homogénea. Se etiqueta la imagen y se obtiene la información de cada uno de los objetos etiquetados en la misma.

```
%% Bucle para entrenar cada una de las clases (numeros)
for i = 1:Nclases

    % Matriz de caracteristicas
    X = [];

    % Numero de muestras de entrenamiento
    nMuestras = 0;

    % Leer imager, pasar a escala de grises y binarizarla con umbral.
    imagenLeer = sprintf("imagenes/entrenamiento%d.jpg", i - 1);
    fOriginal = imread(imagenLeer);
    [nFilas, nColumnas, ~] = size(fOriginal);
    fGray = rgb2gray(fOriginal);
    fBW = fGray < 245;
```

```
    % Operaciones morfologicas para obtener imagen mas homogenea
    % Dilatar la imagen para obtener cada matrícula.
    elem = strel('disk', 2);
    fBW = imdilate(fBW, elem);

    % Etiquetado y obtencion de datos de la imagen
    % Etiquetar la imagen para realizar regionprops
    fBWlabeled = logical(fBW);

    % Obtener los datos de cada matricula
    datosMatriculas = regionprops(fBWlabeled, 'Area', 'BoundingBox', 'Centroid',
'MinorAxisLength', 'MajorAxisLength', 'Orientation');
    nDatosMatriculas = length(datosMatriculas(:,1));
```

A continuación se muestra una imagen de entrenamiento, y su versión binarizada.



Una vez obtenidos los datos de cada objeto se itera sobre cada uno de estos, comprobando antes que el área sea mayor que un umbral arbitrario, con el objetivo de eliminar posible ruido a la hora de etiquetar la imagen, que haya provocado que pequeñas "islas" de píxeles sean consideradas matrículas. Una vez comprobado, se procede a girar y recortar la imagen, utilizando la función propia **f_recortar_girar_imagen.m**, de manera que se obtiene la matrícula recortada y en posición horizontal, para proceder a aislar en este caso cada número por separado.

```
    % ---------------------------------------------------------------------
    % Bucle de tratamiento de imagenes para obtener cada matricula por separado
    % ---------------------------------------------------------------------
    for j = 1:nDatosMatriculas
        % Eliminar posibles objetos detectados pero que no se tratan de
        % matriculas por el area.
        if datosMatriculas(j).Area > 500

            % Recortar y girar la imagen
            fEntrenamiento = f_recortar_girar_imagen(datosMatriculas(j), fGray);

            % Etiquetar la imagen
            fEntrenamientoLabelled = logical(fEntrenamiento);

            % Obtener los datos de cada elemento de la matricula aislada
            datosNumeros = regionprops(fEntrenamientoLabelled, 'Area', 'BoundingBox', 'Centroid',
'MinorAxisLength', 'MajorAxisLength', 'Orientation');
            nDatosNumeros = length(datosNumeros(:,1));

            % Obtener la mediana de las areas de los objetos, para obtener
            % el area de un numero individual.
            areaMediana = f_Obtener_Mediana_Area(nDatosNumeros,datosNumeros);
```

El cálculo de la mediana de las áreas, realizado en la función **f_Obtener_Mediana_Area.m,** permite tener una estimación del área de cada número, y así evitar tratar de entrenar con el marco de la matrícula, o con el objeto creado en el interior de la letra "D" de la matrícula. A continuación se muestra un ejemplo de una matrícula aislada y binarizada.

El siguiente paso consiste en iterar a través de cada uno de los objetos etiquetados en cada matrícula aislada. Para ello se comprueba primero que el área del objeto se encuentre entre un 90% y un 110% de la mediana del área calculada previamente, consiguiendo filtrar de esta manera el marco de la matrícula y el interior de la "D".

Una vez comprobado que se trata de un número, se procede a aislar dicho número, normalizar su tamaño a una imagen de 64x128 píxeles (128 filas, 64 columnas), y calcular las características que se utilizarán para fabricar los clasificadores bayesianos y de mínima distancia. El cálculo de características se realiza en la función propia **f_calculo_caracteristicas.m**.

```
        % ------------------------------------------------
        % Bucle para tratar cada numero de cada matricula
        % ------------------------------------------------


        % Una vez obtenida la mediana, analizar cada numero del vector
        % para realizar el entrenamiento, comprobando nuevamente que se
        % trata de un numero.
        for k = 1:nDatosNumeros
            areaNumero = round(datosNumeros(k).Area);

            % Comprobar que el area esta dentro de unos margenes
            % arbitrarios, para descartar el borde y la D, asi como
            % otros posibles ruidos.
            if (areaNumero > (areaMediana * 0.9)) && (areaNumero < (areaMediana * 1.1))
                nMuestras = nMuestras + 1;

                % "Recortar" cada numero de la matricula
                fNumero = f_recortar_girar_imagen(datosNumeros(k),fEntrenamiento, 1);
                % Una vez se tiene el numero, normalizar tamano a 128 filas x 64 columnas pix
                fNumeroResized = imresize(fNumero, [128 64]);
                set(pIMSHOW, 'CData', fNumeroResized);
                pause(0.000000001);

                % Una vez normalizado, se pasan a calcular las caracteristicas
                % Xi.
                Xaux = f_calculo_caracteristicas(fNumeroResized);
                X = [X; Xaux];
            end
        end
    end
end
```

Una vez se tienen todas las características de cada número, se procede a construir el clasificador. Para ello se hará uso de una clase en la que se irán almacenando los valores de la media aritmética de cada característica, la matriz de varianza y su inversa, la probabilidad de cada clase, ya definida al comienzo del programa, y por último se pre-calcula el valor de ***fk***, que se utilizará en la validación de los clasificadores, para evitar realizar la operación múltiples veces.

Por último, se guarda el entrenamiento en el archivo **clase.mat**.

```
    % -----------------------------------
    % -------Clasificador Bayesiano-------
    % -----------------------------------
    % Una vez finalizada con todas las matriculas de la imagen, se pasa a
    % construir el clasificador Bayesiano. Se escogen las caracteristicas
    % x6~x13, que permite construir una matriz inversa sin tener un numero
    % de condicion excesivamente alto (ill-conditioned matrix).

    % Se tiene la matriz de caracteristicas X, cada columna se trata de una de
    % las caracteristicas x1 - x13.
    clase(i).X = X(:,6:13);

    % Media de cada caracteristica (diap. 42, tema 4). La media tambien
    % servirá para el clasificador lineal.
    clase(i).mu = mean(X(:,6:13))';

    % Varianza de la clase (diap. 42, tema 4), media de las varianzas de cada
    % caracteristica.
    clase(i).V = (clase(i).X'-clase(i).mu)*(clase(i).X'-clase(i).mu)' / nMuestras;
    clase(i).Vinv = pinv(clase(i).V);

    % Calculo de fk, ya que depende de la probabilidad total de la clase, y de
    % su varianza, que son constantes.
    clase(i).fk = log(pC) - 1/2 * log(det(clase(i).V));

    % Probabilidad a priori de cada clase
    clase(i).pC = pC;
end

%% Guardar el entrenamiento
save('clase.mat', 'clase');
```

Función f_recortar_girar_imagen.m

Función que recibe una imagen y una estructura tipo regionprops y realiza un recorte y un giro de la imagen (el giro se realiza en caso de que se trate una matrícula, si se aísla un número no se produce ningún giro).

Se comienza obteniendo los datos de los argumentos de la función, comprobando si el argumento **numero** es un 0 o un 1. Se procede a acotar la plantilla, girarla, y, en caso de que la matrícula estuviese girada, acotar de nuevo la imagen para obtener la matrícula. Por último se binariza la imagen a partir de la imagen en escala de grises.

```
function fEntrenamiento = f_recortar_girar_imagen(datos, f, numero)
%f_recortar_girar_imagen Funcion que recorta y gira la imagen para aislar
%la matricula.

% Obtener datos para facilitar lectura
BoundBox = round(datos.BoundingBox);
anguloGiro = -round(datos.Orientation);  % Negativo puesto que se quiere deshacer el giro
ejeXlong = datos.MajorAxisLength;
ejeYlong = datos.MinorAxisLength;

% En caso de que se quiera aislar un numero no se contempla borde.
if numero == 0
    borde = 30;
else
    borde = 0;
    anguloGiro = 0;
end

% Acotar la plantilla para tener unicamente
fAcotadaGray = f(BoundBox(2) : BoundBox(2) + BoundBox(4)-1, BoundBox(1) : BoundBox(1) +
BoundBox(3)-1);

% Imagen acotada y rotada
fAcotadaRotadaGray = imrotate(fAcotadaGray, anguloGiro);

% En caso de que el angulo de giro sea distinto de 0 (solo se da en las
% matriculas), se debera acotar aun mas la imagen
if anguloGiro ~= 0
    [nFilasLocal, nColumnasLocal, ~] = size(fAcotadaRotadaGray);
    acotarFilaMin = max([round(nFilasLocal / 2 - ejeYlong / 2 ), 1]);
    acotarFilaMax = min([round(nFilasLocal / 2 + ejeYlong / 2), nFilasLocal]);
    acotarColumnaMin = max([round(nColumnasLocal / 2 - ejeXlong / 2 + borde), 1]);
    acotarColumnaMax = min([round(nColumnasLocal / 2 + ejeXlong / 2 - borde), nColumnasLocal]);
    fAcotadaRotadaGray = fAcotadaRotadaGray(acotarFilaMin:acotarFilaMax, acotarColumnaMin:
acotarColumnaMax);
end

% Obtener la imagen binaria a partir de la imagen en escala de grises
if numero == 0
    fEntrenamiento = fAcotadaRotadaGray < 120 & fAcotadaRotadaGray > 0;
end
end
```

Función f_Obtener_Mediana_Area.m

Función que recibe una lista de objetos y calcula la mediana de las áreas, filtrando el posible ruido en forma de áreas extremadamente pequeñas.

```
function areaMediana = f_Obtener_Mediana_Area(nDatosNumeros,datosNumeros)
%f_Obtener_Mediana_Area Funcion que recibe los datos de los objetos obtenidos con
%regionprops y calcula la mediana de las areas de los objetos, que será la del numero

%   Se realiza una comprobacion para eliminar posibles objetos detectados mediante
%   regionprops pero que sean demasiado pequenos como para tratarse de un numero.

% Obtener la mediana de las areas, para identificar el area de un numero.
indiceareaV = 1;
areaVector = 0;

% Tener unicamente en cuenta aquellos objetos cuya area sea
% mayor que 500 pixeles, para eliminar posible ruido al
% binarizar la imagen.
for k = 1:nDatosNumeros
    if datosNumeros(k).Area >  500
        areaVector(indiceareaV) = datosNumeros(k).Area;
        indiceareaV = indiceareaV + 1;
    end
end
areaMediana = median(areaVector);
end
```

Función f_calculo_caracteristicas.m

Función que recibe la imagen de cada número aislado y calcula las características deseadas del mismo, siendo estas el porcentaje de pixeles ocupados de cada región en la que se dividirá.

```
%function x = f_calculo_caracteristicas(fNumeroResized)
%f_calculo_caracteristicas Funcion que recibe la imagen del numero y

% Obtener numero de filas y columnas
[nFilas, nColumnas, ~] = size(fNumeroResized);

% Inicializar variables
x1 = 0; x2 = 0; x3 = 0; x4 = 0; x5 = 0;
x6 = 0; x7 = 0; x8 = 0; x9 = 0; x10 = 0;
x11 = 0; x12 = 0; x13 = 0;

% Recorrer la imagen y
for i = 1:nFilas
    for j = 1:nColumnas

        % Imagen entera
        x1 = x1 + fNumeroResized(i, j);

        % Imagen divida en 4 segmentos
        %
        % |---------|---------|
        % |         |         |
        % |   x2    |    x3   |
        % |         |         |
        % |---------|---------|
        % |         |         |
        % |   x4    |    x5   |
        % |         |         |
        % |---------|---------|
        %
        if (i <= nFilas / 2) && (j <= nColumnas / 2)
            x2 = x2 + fNumeroResized(i, j);
        elseif (i <= nFilas / 2) && (j >= nColumnas / 2)
            x3 = x3 + fNumeroResized(i, j);
        elseif (i >= nFilas / 2) && (j <= nColumnas / 2)
            x4 = x4 + fNumeroResized(i, j);
        elseif (i >= nFilas / 2) && (j >= nColumnas / 2)
            x5 = x5 + fNumeroResized(i, j);
        end



        % Imagen divida en 8 segmentos
        %
        % |---------|---------|
        % |   x6    |   x7    |
        % |---------|---------|
        % |   x8    |   x9    |
        % |---------|---------|
        % |   x10   |   x11   |
        % |---------|---------|
        % |   x12   |   x13   |
        % |---------|---------|
        %
        if (i <= nFilas / 4) && (j <= nColumnas / 2)
            x6 = x6 + fNumeroResized(i, j);
        elseif (i <= nFilas / 4) && (j >= nColumnas / 2)
            x7 = x7 + fNumeroResized(i, j);
        elseif (i >= nFilas / 4 && i <= nFilas / 2) && (j <= nColumnas / 2)
```

```
            x8 = x8 + fNumeroResized(i, j);
        elseif (i >= nFilas / 4 && i <= nFilas / 2) && (j >= nColumnas / 2)
            x9 = x9 + fNumeroResized(i, j);
        elseif (i >= nFilas / 2 && i <= nFilas * 3/4) && (j <= nColumnas / 2)
            x10 = x10 + fNumeroResized(i, j);
        elseif (i >= nFilas / 2 && i <= nFilas * 3/4) && (j >= nColumnas / 2)
            x11 = x11 + fNumeroResized(i, j);
        elseif (i >= nFilas * 3/4) && (j <= nColumnas / 2)
            x12 = x12 + fNumeroResized(i, j);
        elseif (i >= nFilas * 3/4) && (j >= nColumnas / 2)
            x13 = x13 + fNumeroResized(i, j);
        end
    end
end
% Montar vector X
x = [x1 x2 x3 x4 x5 x6 x7 x8 x9 x10 x11 x12 x13];
end
```

Archivo MAIN_pr4_validación.m

En este segundo archivo se realiza la validación de los clasificadores creados en el archivo **MAIN_pr4_construccion.m**. El procedimiento para obtener cada matrícula por separado es idéntico al utilizado anteriormente. Se comienza cargando el archivo con el clasificador **clase.mat**, leyendo la imagen de validación y binarizándola. Una vez etiquetada, nuevamente se itera por cada una de las matrículas encontradas.

```
%% Practica 4 - Sistemas de Percepcion - Pablo Leon Barriga
clear;close all;clc

%% Variables a utilizar

% Cargar el archivo clase.mat, que contiene la estructura con la
% informacion de cada clase
load('clase.mat');
nClases = length(clase);

%% Leer imager, pasar a escala de grises y binarizarla con umbral.
imagenLeer = sprintf("imagenes/validacion.jpg");
fOriginal = imread(imagenLeer);
[nFilas, nColumnas, ~] = size(fOriginal);
fGray = rgb2gray(fOriginal);
fBW = fGray < 245;

%% Operaciones morfologicas para obtener imagen mas homogenea
% Dilatar la imagen para obtener cada matrícula.
elem = strel('disk', 2);
fBW = imdilate(fBW, elem);

%% Etiquetado y obtencion de datos de la imagen
% Etiquetar la imagen para realizar regionprops
fBWlabeled = logical(fBW);

% Obtener los datos de cada matricula
datosMatriculas = regionprops(fBWlabeled, 'Area', 'BoundingBox', 'Centroid', 'MinorAxisLength',
'MajorAxisLength', 'Orientation');
nDatosMatriculas = length(datosMatriculas(:,1));

%% Bucle de tratamiento de imagenes para obtener cada matricula por separado
for j = 1:nDatosMatriculas
    if datosMatriculas(j).Area > 500
        % Recortar y girar la imagen
        fValidacion = f_recortar_girar_imagen(datosMatriculas(j), fGray, 0);

        % Etiquetar la imagen
        fValidacionLabelled = logical(fValidacion);

        % Obtener los datos de cada elemento de la matricula aislada
        datosNumeros = regionprops(fValidacionLabelled, 'Area', 'BoundingBox', 'Centroid',
'MinorAxisLength', 'MajorAxisLength', 'Orientation');
        nDatosNumeros = length(datosNumeros(:,1));
```

Una vez se tiene la información de cada objeto, se comprueba primero si la orientación de la matrícula es "correcta", esto es, que la zona azul de esta con las estrellas y la "D" se encuentren a la izquierda de la imagen. Esto se realiza con la función propia **f_Comprobar_Orientacion.m**, donde, en caso de tener que girar 180 grados la imagen, se recalcula la información de cada objeto de la imagen. Nuevamente se calcula la mediana del área con la función **f_Obtener_Mediana_Area.m** explicada anteriormente. El siguiente paso consiste en ordenar los

números por orden de aparición, para obtener el número de la matricula de forma correcta. Para ello se utiliza la función **f_Obtener_Numeros_Ordenados.m**.

```
        % Comprobar orientacion
        [fValidacionLabelled,datosNumeros, nDatosNumeros]  =
f_Comprobar_Orientacion(datosNumeros, nDatosNumeros, fValidacionLabelled);

        % Obtener la mediana de las areas de los objetos, para obtener
        % el area de un numero individual.
        areaMediana = f_Obtener_Mediana_Area(nDatosNumeros,datosNumeros);

        % Una vez obtenida la mediana, analizar cada numero y
        % obtener la matriz con cada numero ordenado por orden de aparicion
        % en la matricula.
        [matrizOrdenar, numeroNumeros] = f_Obtener_Numeros_Ordenados(nDatosNumeros,datosNumeros,
areaMediana);
```

Una vez ordenados los números (junto con sus bounding boxes y centroides), se pasa a iterar cada número, normalizando su tamaño a 64x128 pixeles, calculando sus características con la función **f_calculo_caracteristicas.m** ya explicada anteriormente, y clasificando según un clasificador bayesiano y un clasificador de distancia mínima.

```
        % Una vez se tiene cada numero de la matricula por separado, se
        % detecta cada numero.
        matriculaBayesiano = zeros(1, numeroNumeros - 1);
        matriculaTextoBayesiano = '';
        matriculaLineal = zeros(1, numeroNumeros - 1);
        matriculaTextoLineal = '';
        for k = 1 : numeroNumeros - 1

            % "Recortar" cada numero de la matricula
            fNumero = fValidacionLabelled(matrizOrdenar(k, 1):matrizOrdenar(k, 2),
matrizOrdenar(k, 3):matrizOrdenar(k, 4));

            % Una vez se tiene el numero, normalizar tamano a 128 filas x 64 columnas pix.
            fNumeroResized = imresize(fNumero, [128 64]);

            % Una vez normalizado, se pasan a calcular las caracteristicas
            % Xi.
            Xaux = f_calculo_caracteristicas(fNumeroResized);
            X = Xaux(6:13);

            % Una vez obtenidas las caracteristicas, se pasa a
            % comprobar que numero del entrenamiento se corresponde con
            % el detectado.
            fdkBayesiano = zeros(1, 10);
            for l = 1 : nClases
                dist = (X - clase(l).mu') * clase(l).Vinv * (X - clase(l).mu')';
                fdkBayesiano(l) = -1/2*dist^2 + clase(l).fk;
            end
            [~, indiceMinimoBayesiano] = min(abs(fdkBayesiano));
            matriculaBayesiano(k) = indiceMinimoBayesiano - 1;
            matriculaTextoBayesiano = strcat(matriculaTextoBayesiano,
num2str(matriculaBayesiano(k)));

            % Clasificador de minima distancia
            fdkLineal = zeros(1, 10);
            for l = 1 : nClases
                fdkLineal(l) = norm(clase(l).mu' - X);  % Distancia a cada clase
            end
            [~, indiceMinimoLineal] = min(abs(fdkLineal));
            matriculaLineal(k) = indiceMinimoLineal - 1;
```

```
            matriculaTextoLineal = strcat(matriculaTextoLineal, num2str(matriculaLineal(k)));

        end
        f_Mostrar_Matricula_Numero(fValidacionLabelled, matrizOrdenar, numeroNumeros,
matriculaBayesiano, matriculaTextoBayesiano, matriculaLineal, matriculaTextoLineal);
        pause;
    end
end
```

Con cada iteración del bucle se va construyendo la matrícula en forma de texto, tanto con el clasificador bayesiano como con el clasificador de distancia mínima. Para el clasificador de distancia mínima se calcula la norma del vector que resulta de restar las características obtenidas del número con la media de características obtenidas en la construcción del clasificador. Por último, se llama a la función **f_Mostrar_Matricula_Numero.m**, que recibe tanto los vectores como las cadenas de caracteres de cada clasificador, y las muestra superpuestas a la matrícula que se está detectando.

A continuación se pasan a explicar aquellas funciones nuevas que han aparecido en este segundo código.

Función f_Comprobar_Orientacion.m

Función que recibe una estructura de regionprops y una imagen, y comprueba que la parte azul de la matrícula se encuentre en la parte izquierda de la misma. En caso contrario, realiza un giro de 180 grados y recalcula la información de los objetos. En caso de no ser necesario el giro, la función actúa como "bypass" y devuelve los argumentos que recibió.

```
function [fValidacionLabelled,datosNumeros, nDatosNumeros]  =
f_Comprobar_Orientacion(datosNumeros, nDatosNumeros, fValidacionLabelled)
%f_Comprobar_Orientacion Funcion que comprueba la orientacion de la
%matricula, girando 180 grados para obtenerla del derecho.
%   Detailed explanation goes here

% Inicializar variable para obtener el objeto de mayor area
datoMayorArea = datosNumeros(1);

[~, nColumnas, ~] = size(fValidacionLabelled);

% Obtener el objeto de mayor area, que se corresponde con la parte azul de
% la matricula.
for i = 2:nDatosNumeros
    if datosNumeros(i).Area > datoMayorArea.Area
        datoMayorArea = datosNumeros(i);
    end
end

% Si la variable horizontal del centroide de la parte azul se encuentra mas
% alla de la mitad de la imagen, se debe voltear 180 grados la imagen.
if datoMayorArea.Centroid(1) > nColumnas / 2
    fValidacionLabelled = imrotate(fValidacionLabelled, 180);

    % Reobtener los datos de cada elemento de la matricula una vez girada
    % 180 grados.
    datosNumeros = regionprops(fValidacionLabelled, 'Area', 'BoundingBox', 'Centroid',
'MinorAxisLength', 'MajorAxisLength', 'Orientation');
    nDatosNumeros = length(datosNumeros(:,1));
end
end
```

Función f_Obtener_Numeros_Ordenados.m

Función que recibe una tabla de objetos de regionprops y crea una matriz que contiene los datos de la bounding box y del centroide de cada número, según el orden de aparición de izquierda a derecha.

```
function [matrizOrdenar, indiceNumeros] = f_Obtener_Numeros_Ordenados(nDatosNumeros,datosNumeros,
areaMediana)
%f_Obtener_Numeros_Ordenados Funcion que recibe todos los objetos, filtra los numeros y los
%ordena segun orden de apariencia en la matricula (de izquierda a derecha).

indiceNumeros = 1;
matrizOrdenar = [];
for k = 1:nDatosNumeros
    areaNumero = round(datosNumeros(k).Area);

    % Comprobar que el area se encuentra dentro de unos margenes
    % arbitrarios, para descartar el borde y la D y el resto de
    % objetos no deseados.
    if areaNumero > areaMediana *0.8 && areaNumero < areaMediana *1.2
        indiceNumeros = indiceNumeros + 1;

        % Introducir en una matriz los valores de fila y columna
        % para acotar, así como el indice, para poder ordenar segun
        % aparezcan en la matricula.

        numeroBoundingBox = round(datosNumeros(k).BoundingBox);
        numeroFilaMin = numeroBoundingBox(2);
        numeroFilaMax = numeroBoundingBox(2) + numeroBoundingBox(4);
        numeroColumnaMin = numeroBoundingBox(1);
        numeroColumnaMax = numeroBoundingBox(1) + numeroBoundingBox(3);
        numeroCentroide = round(datosNumeros(k).Centroid);

        matrizOrdenar = [matrizOrdenar; numeroFilaMin numeroFilaMax numeroColumnaMin
numeroColumnaMax numeroCentroide];

    end
end
% Ordenar la matriz en orden ascendente de distancia hacia la
% izquierda, de manera que se tienen en orden los numeros segun
% aparecen en la matricula.
matrizOrdenar = sortrows(matrizOrdenar, 3);
end
```

Función f_Mostrar_Matricula_Numero.m

Recibe una imagen de una matrícula, así como la matrícula reconocida por ambos clasificadores, y la muestra por pantalla.

```
function f_Mostrar_Matricula_Numero(fMatricula, matrizOrdenar, numeroNumeros, matriculaBayesiano,
matriculaTextoBayesiano, matriculaLineal, matriculaTextoLineal)
%UNTITLED5 Summary of this function goes here
%   Detailed explanation goes here

[nFilas, ~, ~] = size(fMatricula);
% Obtener imagen en RGB para poder mostrar los numeros detectas mas
% adelante.
fValidacionRGB = [];
for RGB = 1:3
    fValidacionRGB(:,:,RGB) = uint8(fMatricula * 150);
end

% Mostrar imagen
fValidacionRGB = uint8(fValidacionRGB);
figure('WindowState','fullscreen');
imshow(fMatricula);
pause(0.00001);
for k = 1:numeroNumeros - 1
    text(matrizOrdenar(k, 5), nFilas / 2 - 20, sprintf("%d", matriculaBayesiano(k)), 'Color',
'cyan', 'FontSize', 40, 'FontWeight', 'bold');
    text(matrizOrdenar(k, 5), nFilas / 2 + 20, sprintf("%d", matriculaLineal(k)), 'Color', 'red',
'FontSize', 40, 'FontWeight', 'bold');
end
title(sprintf("Matricula clasificador bayesiano: %s. Matricula clasificador lineal: %s",
matriculaTextoBayesiano, matriculaTextoLineal));
xlabel("Azul: clasificador bayesiano. Rojo: clasificador distancia minima");
end
```

Resultados

A continuación se muestran los resultados de la validación. También se adjunta un pequeño vídeo que muestra tanto el entrenamiento como los resultados.

Azul: clasificador bayesiano. Rojo: clasificador distancia minima

Azul: clasificador bayesiano. Rojo: clasificador distancia minima

Azul: clasificador bayesiano. Rojo: clasificador distancia minima

Azul: clasificador bayesiano. Rojo: clasificador distancia minima

Matricula clasificador bayesiano: 012345678. Matricula clasificador distancia minima: 012345678

Azul: clasificador bayesiano. Rojo: clasificador distancia minima

Matricula clasificador bayesiano: 721609081. Matricula clasificador distancia minima: 721609081

Azul: clasificador bayesiano. Rojo: clasificador distancia minima

Matricula clasificador bayesiano: 213345555. Matricula clasificador distancia minima: 213345555

Azul: clasificador bayesiano. Rojo: clasificador distancia minima

Matricula clasificador bayesiano: 123456789. Matricula clasificador distancia minima: 123456789

Azul: clasificador bayesiano. Rojo: clasificador distancia minima