VC SP QT 2022

Xenia Calisalvo Veciana
Natalia Dai
You Wu

Cuatrimestre de otoño 2022

Índice

1. Introducción

El objetivo final de este trabajo es el de implementar un código propio con la capacidad para la detección y la lectura de matrículas en un conjunto de imágenes dadas de coches en escenarios exteriores. Para hacerlo vamos a desarrollar el proyecto en las siguientes fases:

1. Construir un detector de matrículas específico para el conjunto de imágenes que se nos proporciona.
2. Encontrar la separación entre caracteres a partir de la matrícula detectada anteriormente.
3. Implementar con OCR el reconocimiento de los dígitos en las diferentes subimágenes de la matrícula dividida.

2. Metodología

Ahora nos centraremos en explicar todo el proceso que hemos seguido para conseguir obtener nuestros resultados. Para ello hablaremos paso a paso siguiendo las diferentes fases que constituyen el proyecto.

2.1. Localización de la matrícula

Para la primera parte del proyecto, que es la identificación de la placa de matrícula de cada coche, primero hemos transformado la imagen a escala de grises y a continuación hemos realizado un filtrado de mediana de la imagen con la finalidad de quitar ruidos de esta.

Figura 1

El siguiente paso ha sido el de binarizar. Al principio utilizamos un primer método que consistía en utilizar un *threshold local* junto un *open close* con forma de cuadrado para conseguir que regiones separadas por poco se juntaran y así poder eliminar pequeños agujeros irregulares en las placas. Usábamos este método porque inicialmente parecía dejar los bordes de la matricula bastante marcados.

Figura 2

Aunque en la mayoría de matrículas lo dicho anteriormente es cierto, seguía habiendo una cantidad considerable en la que teníamos problemas con la distinción de la placa, la mayoría de ellos son las imágenes que no tiene mucho contraste entre la placa y el coche. Por lo tanto utilizamos el kernel de convolución y el mismo nivel de tolerancia de gris para decidir el umbral. Una vez hecho la binarización utilizamos la función *erode* con distintos elementos estructurados en forma de líneas, con la intención de alargar algunas líneas de las que componen los cierres de la placa , para terminar de cerrar las bordes de las placas. Primero utilizamos dos *erode*, uno con orientación horizontal y el otro en vertical, y aunque podía intuirse la forma, esto no era suficiente para que se pudiera detectar por visión por computador. De esta forma conseguimos incrementar nuestro porcentaje de aciertos pero seguíamos teniendo ciertos casos con pautas que creíamos que podíamos mejorar.

Figura 3

En el siguiente método seguimos con la filosofía del anterior, añadiendo un *erode* para engrosar las líneas horizontales. Este método se hizo por el mismo motivo que el anterior: algunas placas no acaban de quedar bien delimitadas por sus bordes.

Figura 4

Finalmente, repetimos lo mismo que anteriormente, pero esta vez para las líneas horizontales y no las verticales y en una menor medida para evitar el toque de los caracteres de la matrícula a la placa que podría crear enmarcación de la placa de forma incorrecta.

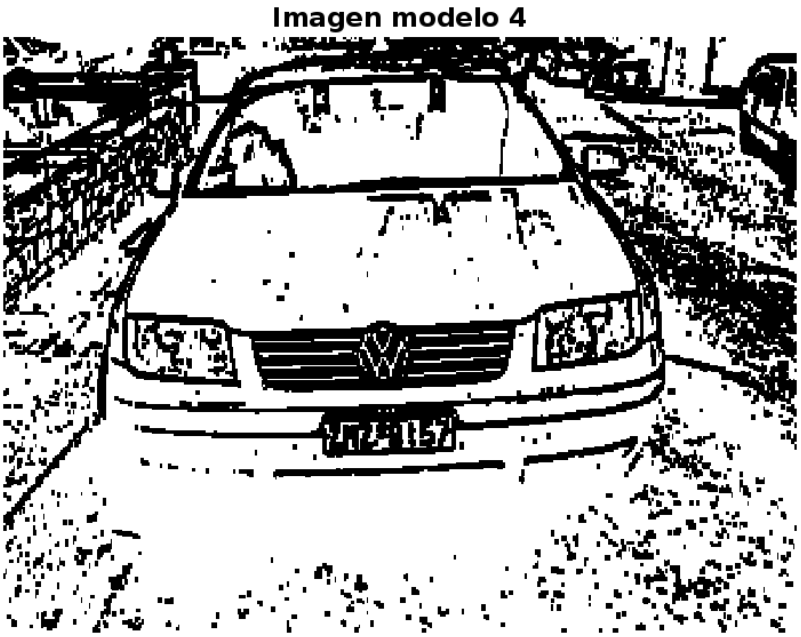

Figura 5

En todos los métodos anteriores que hemos servido para binarizar la imagen, se utiliza el mismo proceso para enmarcar la placa de la matrícula. Para ello hemos creado una función llamada *placa* que nos permite encapsular esta parte del código para que sea mucho mas legible. Dentro de esta utilizamos una función llamada *regionprops* que nos permite obtener las propiedades de una región y le pasamos como parámetro los competentes conectados. Con esto obtenemos la *BoundingBox* y el *EulerNumber* que nos permiten definir las condiciones para definir la placa. La primera propiedad la utilizamos para asegurarnos de que una matrícula siempre va a ser más ancha que alta, concretamente definimos que para considerarse matrícula la anchura debía ser mayor que 2'3 alturas y menor que 8 alturas y también que la altura debía ser mayor que 0.0375 veces la altura de la imagen y más pequeña que 8'5 veces la imagen. Además, añadimos que la anchura debía ser menor que la mitad de la imagen. Estos parámetros fueron seleccionados primeramente a ojo a partir de las imágenes y poco a poco los fuimos retocando midiéndolo con *impixelinfo* hasta conseguir los que nos parecía que daban unos mejores resultados. El parámetro de *euler* define el número de agujeros que hay en la imagen y lo utilizamos con la finalidad de descartar aún mas imágenes que tenían demasiadas irregularidades y que normalmente hacían referencia a vegetación u otros.

Figura 6: Placa Enmarcada de la imagen original

Figura 7: Placa de la imagen

2.2. Localización de los caracteres

Una vez conseguidas las placas, el objetivo de ahora será hacer lo mismo pero para la localización de cada uno de los caracteres. Esta parte del codigo la hemos encapsulado en una función propia llamada *numsBoundingBox* de forma que se pueda llamar varias veces y no tener código redundante. Para ello usaremos como imagen inicial el trozo recortado de la placa y le aplicaremos un *threshold local*. A continuación utilizaremos la función *markers* con tal de limpiar la imagen y dejar solo los dígitos/letras sin los lados.

Figura 8: Recorte de la placa

Una vez que tenemos la imagen de esta forma procedemos a seguir un proceso similar al de detección de la matrícula. Como haremos este proceso para carácter también, hemos creado una función llamada *caracteres* con esta finalidad.
-Primeramente, etiquetamos los componentes conectados y conseguimos las propiedades de cada uno de ellos de la misma forma que antes. También definimos la altura y la anchura de la imagen total ya que nos servirá para comparaciones más adelante.
-Comprobamos que hayamos encontrado algún componente ya que en caso de que la placa no se hubiera recortado correctamente se puede dar el caso de que no encontramos nada y por tanto no sigamos con el proceso ya que nos encontramos delante de un error.
-En caso contrario, procederemos con las comprobaciones para ver si el componente en cuestión se trata de una letra. Estas comprobaciones las hacemos porque en la imagen binarizada podemos encontrar ruido de fondo y por tanto, componentes que no tengan nada que ver con los números. Las comprobaciones que hacemos son ver que el doble de la altura de la letra sea mayor que la altura de la imagen, siete veces la anchura de la letra más pequeña que la de la imagen y que 1'75 veces la altura de la imagen es mayor que su anchura. Estas comprobaciones son muy flexibles ya que lo único que queremos descartar aquí son unas pocas irregularidades e igual algunas figuras que aparecen en las imágenes que forman parte de la matricula pero que no son ni letras ni números.
-Una vez hecho esto, el siguiente paso es comprobar que hayamos encontrado el número correcto de caracteres. En este caso, comprobamos que como mínimo hayamos encontrado 3, esto lo hacemos porque en caso negativo descartamos ya la binarización de la placa que estamos utilizando y procedemos a probar con otra ya que consideramos que esta no es lo suficientemente buena.
-En caso de que la anterior condición se cumpla tenemos 3 opciones: que des del principio ya consigamos 7 caracteres, que teóricamente son los que buscamos, que encontremos de menos o que encontremos de más.
-En caso de que encontremos de menos o de más cogeremos la imagen e iremos variando el *threshold* y recalculando las *BoundingBox* de los supuestos caracteres hasta encontrar los 7 que queremos o superar unos límites que hemos establecido en los cuales consideramos que ya no encontraremos nada de provecho en esas imágenes. Cuando tenemos 7 caracteres, los cuales suponemos que ya son correctos al cumplir las condiciones, los enmarcaremos uno a uno para visualizarlos, adjuntamos un ejemplo abajo, y procederemos con la siguiente fase de la predicción.

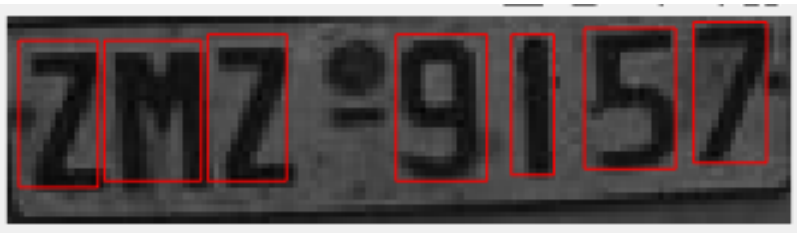

Figura 9: Recorte de la placa con caracteres enmarcados

2.3. Reconocimiento de caracteres

En esta sección vamos a centrarnos en la detección de los caracteres. Para llegar a esta fase, como ya hemos dicho anteriormente, nos encontramos con los siete supuestos caracteres delimitados por sus correspondientes *BoundingBox*.
Lo dividimos en dos partes: dígitos y letras. Para el primer caso, primero intentamos usar el *training* propio que tenía ya *Matlab* de los dígitos con la función de *extractHOGFeatures*, y para el segundo, utilizamos las imágenes del disco de la asignatura cogiendo solamente la parte de las letras usando las funciones de *extractFeatures* y *matchFeatures*. Viendo que los números se confundían mucho unos con los otros, y las letras no daban buenos resultados con esas dos funciones (como bien señala el aviso que se subió en Racó) tampoco, decidimos implementar algo diferente: utilizamos la clasificación de caracteres mediante las características de HOG para ambos casos, pero con un entrenamiento nuestro, que para nuestra sorpresa, fue mejor. Creamos dos carpetas, una para las letras y otra para los números, cada una de ellas conteniendo los subdirectorios que contienen las imágenes de entrenamiento las cuales nos permiten generar el clasificador de números y de letras.

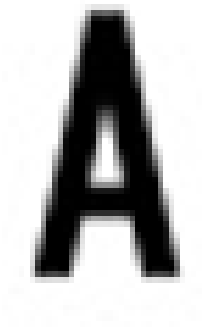

Figura 10: Ejemplo de la letra A

Figura 11: Otro ejemplo de la letra A

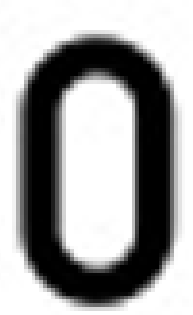

Figura 12: Ejemplo del numero 0

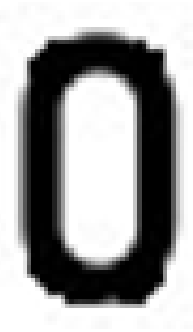

Figura 13: Otro ejemplo del numero 0

Ambos clasificadores se generan mediante la función *setClassifier*, utilizando la carpeta de números y la de letras, utilizamos la función *extractHOGFeatures*, comentada anteriormente, para extraer las características de HOG con medida de 4x4 . Y para clasificar utilizamos la función *predictUsingHOG* en el que le pasamos como parámetro la imagen del carácter que queremos clasificar y el clasificador de números o de letras según lo que necesitamos y nos retorna el resultado de clasificación. Teniendo en cuenta la mayoría de los casos la matrículas de coche esta formado por tres letras seguidos de cuatro números, hemos decidido clasificar las tres primeras mediante el clasificador de letras y la resta como números. Nos aseguramos de redimensionar los tamaños del conjunto de entrenamiento con la finalidad de que sean más parecidos a los caracteres a los que nos encontramos en nuestras placas. Esto se hace dentro de la función *extractHOGFeatures* y *predictUsingHOG*. Además, el uso de la función *extractHOGFeatures* con *cellSize* 4x4 nos permite sacar el mismo número de características de HOG para poder dar a cabo la predicción del carácter.

Finalmente, todos los resultados que nos retorna de una placa los vamos concatenando dentro de una variable hasta haber clasificado los 7 caracteres que es cuando lo vamos a mostrar por la terminal para poder comprobar el resultado.

3. Resultados experimentación

Para realizar la experimentación, hemos separado los resultados entre parciales y totales. Los resultados totales son aquellos que hacen referencia al total de imágenes que tenemos, es decir siempre se evalúan sobre 67. En cambio, los resultados parciales utilizan el valor que se ha obtenido en la fase anterior para calcular su porcentaje. Por ejemplo, si en la primera fase, el reconocimiento de placas, hemos obtenido 50 placas de 67 el porcentaje parcial de la fase 2, enmarcar caracteres, se calculará sobre estas 50 placas. Adjuntamos una tabla con los resultados descritos a continuación.

Resultados de la experimentación			
Resultados	Reconocimiento de placas	Enmarcar caracteres	Reconocimiento de caracteres
Aciertos sobre 67	57	51	35
Parcial (%)	-	89,5	68,6
Total (%)	85,1	76,1	52,2

Cuadro 1: Resultados totales y parciales de la experimentación en cada fase

En nuestra experimentación también hemos hecho un recuento de los falsos positivos, falsos negativos y aciertos en el reconocimiento de caracteres. Para hacerlo contamos como aciertos aquellas veces que esperábamos un carácter en concreto y hemos obtenido este mismo, falsos positivos cuando esperábamos un carácter y ha reconocido otro pero al contabilizar bien el número de caracteres se ha considerado que la matricula era correcta y falsos negativos, el carácter que ha sido confundido por otro. Adjuntamos una tabla con el recuentro total de los caracteres reconocidos de todas las matriculas

Falsos negativos/positivos																								
	A	B	E	H	I	K	M	N	O	P	T	X	Y	Z	0	1	2	3	4	5	6	7	8	9
A	35	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
B	0	35	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
E	0	0	35	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
H	0	0	0	35	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
I	0	0	0	0	34	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0
K	0	0	0	0	1	31	0	2	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0
M	0	0	0	0	0	0	35	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
N	0	0	0	0	0	0	0	35	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
P	0	0	0	0	0	0	0	0	35	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
O	0	0	0	0	0	0	0	0	0	35	0	0	0	0	0	0	0	0	0	0	0	0	0	0
T	0	0	0	0	0	0	0	0	0	0	35	0	0	0	0	0	0	0	0	0	0	0	0	0
X	0	0	0	0	0	0	0	0	0	0	0	35	0	0	0	0	0	0	0	0	0	0	0	0
Y	0	0	0	0	1	0	0	1	0	0	0	0	33	0	0	0	0	0	0	0	0	0	0	0
Z	0	0	0	0	1	0	0	0	0	0	0	0	0	34	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	33	2	0	0	0	0	0	0	0	0
1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	35	0	0	0	0	0	0	0	0
2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	34	0	0	0	0	1	0	0
3	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	35	0	0	0	0	0	0
4	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	34	0	0	0	0	0
5	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	35	0	0	0	0
6	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	34	0	0	0
7	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	7	0	0	0	0	28	0	0
8	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	6	0	0	0	0	0	34	1
9	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	6	0	0	0	0	0	0	35

Cuadro 2: Falsos positivos, falsos negativos y aciertos en el reconocimiento de letras

Como podemos ver, mayoritariamente los falsos aciertos se concentran en el reconocimiento de los 2 donde podemos ver que algunos de ellos se han predecido como 7, 8 y 9. Esto suele ser debido a ruidos de la imagen que hacen que o se borre parte del 2 o se le añadan píxeles a la componente conexa haciendo que al buscar sus características se confunda con otros caracteres. A parte de este error más remarcable, los otros parecen ser casos aislados que no llegan a ser significativos.

4. Conclusión

Como conclusión de este proyecto, nos gustaría hacer una introspección en lo positivo y en lo negativo de nuestro trabajo.
Por un lado estamos contentas con el resultado final dado a que obviamente no es algo que hubiéramos podido hacer antes de empezar esta asignatura y es muy gratificante el hecho de que se trate de algo tan visual y práctico en la vida real. También estamos ciertamente satisfechas con el reconocimiento de las placas ya que a mitad del proyecto se nos avisó de que igual esta forma de solucionar el problema daba lugar a una mayor complejidad de este, pero conseguimos obtener unos buenos resultados de todas formas. Además hay que destacar que aunque la asignatura no esté enfocada en el aprendizaje de *Matlab* sino en el uso de las técnicas para desarrollar la visión por computador, es innegable que nuestra destreza con esta herramienta ha mejorado a lo largo del uso. De la misma forma, nuestra habilidad para buscar recursos para poder avanzar y familiarizarnos con una gran cantidad de funciones relacionas con el ámbito también han mejorado notablemente.
Por el otro lado nos quedamos con las ganas de aún haber mejorado más nuestros resultados. La posible falta de tiempo al final y el constante problema de coordinación entre los miembros del grupo al final al tener diferentes horarios de exámenes no nos ha permitido incrementar todo lo que nos hubiera gustado nuestro porcentaje de aciertos totales. Aún así los resultados de usar OCR no han sido malos y los resultados parciales de estos nos parecen lo suficientemente buenos aunque nos habría gustado experimentar más con diferentes tipos de ruidos en las imágenes de entrenamiento y igual algún tipo de rotación con tal de aumentar el número de *matches*.
Así que en definitiva, como en todo proyecto siempre te quedas con ganas de aprender más y de explorar aún más las posibilidades de este pero aún así estamos satisfechas con el trabajo realizado y los resultados obtenidos.

5. Anexo

```
close all
clear all

%Directorio para cargar las im genes:
a = dir('/MATLAB Drive/proyecto_vc/*.jpg');

nf = size(a);

%Cargar los directorios de n meros y letras
numerosDir =  fullfile('/MATLAB Drive/proyecto_vc/numeros');
letrasDir =  fullfile('/MATLAB Drive/proyecto_vc/letras');

trainLetras = imageDatastore(letrasDir,'IncludeSubfolders',true,'LabelSource','foldernames
clasificadorLetras = setClassifier(trainLetras);
trainNumeros = imageDatastore(numerosDir,'IncludeSubfolders',true,'LabelSource','foldername
clasificadorNumeros = setClassifier(trainNumeros);

results = strings(size(a));

for i=1:size(a)

    close all
    filename = horzcat(a(i).folder,'/',a(i).name);
    I = (imread(filename));
    % preprocess
    figure,imshow(I), title(a(i).name);
    Ig = rgb2gray(I);
    Im = medfilt2(Ig);
    Ibw = thresholdLocally(Im, 70);
    k3 = strel('square', 3);
    Io = imopen(Ibw, k3);
    k4 = strel('square', 2);
    Ic = imclose(Io, k4);

    matricula1 = placa(Ic);

    Im = medfilt2(Ig);
    im=Ig;
    h=ones(15)/15/15;
    promig=imfilter(im,h,'conv','replicate');
    imbw=im>(promig-15);
    op = imerode(imbw, strel('line', 3,0));
    op = imerode(op, strel('line', 3,90));
    matricula2 = placa(op);

    op2 = imerode(op, strel('line', 6,0));
    matricula3 = placa(op2);

    op = imerode(op, strel('line', 3,90));
    matricula4 = placa(op);
```

```
matricula = [matricula1;matricula2;matricula3; matricula4];

if (size(matricula) ~= 0)
    for j = 1 : size(matricula)
        crop = imcrop(Ig, matricula(j,:));
        th = 20;
        [nums,sN] = numsBoundingBoxs(crop,20);
        sizeNums = size(nums,1);
        if (sizeNums >= 3)
            rectangle('Position', matricula(j,:), 'EdgeColor', [1 0 0.0])
            while (sizeNums < 7 & th-1 > 0)
                th = th - 1;
                [nums,sN] = numsBoundingBoxs(crop, th);
                sizeNums = size(nums,1);
            end
            while (sizeNums > 7 & th < 100)
                th = th + 1;
                [nums,sN] = numsBoundingBoxs(crop, th);
                sizeNums = size(nums,1);
            end
        end
        [numsT,sNT] = numsBoundingBoxs(crop,-1);
        if (sizeNums < size(numsT,1))
            sizeNums = size(numsT,1);
            nums = numsT;
        end
        if (sizeNums == 7)
            figure, imshow(crop),title(i)
            matriculaCoche = "";
            gray = im2gray(crop);
            for k = 1:3
                nums(k,1:2) = nums(k,1:2)-1;
                nums(k,3:4) = nums(k,3:4)+1;
                rectangle('Position', nums(k,:), 'EdgeColor', [1 0 0.0])
                img = imcrop(crop, nums(k,:));
                pred = string(predictUsingHOG(img,clasificadorLetras));
                matriculaCoche = strcat(matriculaCoche,pred);
            end
            matriculaCoche = strcat(matriculaCoche, '-');
            for k = 4:7
                nums(k,1:2) = nums(k,1:2)-1;
                nums(k,3:4) = nums(k,3:4)+1;
                rectangle('Position', nums(k,:), 'EdgeColor', [1 0 0.0])
                img = imcrop(crop, nums(k,:));
                pred = string(predictUsingHOG(img,clasificadorNumeros));
                matriculaCoche = strcat(matriculaCoche,pred);
            end
            annotation('textbox', [0.45, 0.3, 0, 0], 'string', matriculaCoche)
            matriculaCoche
            results(i) = matriculaCoche;
            %pause
            break
        end
    end
```

```
    end
    drawnow
    pause
end

results

function imbw = thresholdLocally(Im, step)

    f = size(Im,1);
    c = size(Im,2);
    mat = zeros(f,c, 'double');

    for i = 1:step:f
        for j = 1:step:c
            iend = i + step;
            if(iend > f)
                iend = f;
            end
            jend = j + step;
            if(jend > c)
                jend = c;
            end
            thresh = graythresh(Im(i:iend,j:jend));
            mat(i:iend,j:jend) =  thresh;
        end
    end

    k = fspecial('average', step);
    mat = imfilter(mat, k);

    imbw = im2double(Im);
    imbw(imbw < mat) = 0;
    imbw(imbw >= mat) = 1;

end

function img = markers(im)
    mark = im;
    mark(2:end-1,2:end-1) = 0;
    rec = imreconstruct(mark, im);
    img = imsubtract(im, rec);
end

function img = placa(im)

    [L, n] = bwlabel(im);
    Iwidth = size(im, 2);
    Iheight = size(im, 1);
    P = regionprops(L, 'BoundingBox', 'EulerNumber');
    euler = cat(1, P.EulerNumber);
    boundingBox = cat(1, P.BoundingBox);
    height = boundingBox(:,4);
    width = boundingBox(:,3);
```

```
    img = boundingBox(euler <-1 & width > 2.3*height & width < 8*height & height > 0.0375*I

end

function [nums, sN] = numsBoundingBoxs(crop, th)

    if (th == -1) bw = imbinarize(crop);
    else bw = thresholdLocally(crop, th); % poner crop, 1
    end

    n = markers(~bw);
    [nums,sN] = caracteres(n);
end

function [c, n] = caracteres(im)

    [L, n] = bwlabel(im);
    Iwidth = size(im, 2);
    Iheight = size(im, 1);
    P = regionprops(L, 'BoundingBox', 'EulerNumber');
    if (n > 0)
        boundingBox = cat(1, P.BoundingBox);
        height = boundingBox(:,4);
        width = boundingBox(:,3);

        c = boundingBox( 2.5*height > Iheight & width*7 < Iwidth & 1.75*height >= width,:);
    else
        boundingBox = cat(1, P.BoundingBox);
        c = boundingBox(:,:);
    end

end

function classifier = setClassifier(trainingSet)

    numImages = numel(trainingSet.Files);
    trainingFeatures = [];

    for i = 1:numImages
        im = readimage(trainingSet,i);
        im = im2gray(imresize(im, [32,32]));
        im = imbinarize(im);

        trainingFeatures(i, :) = extractHOGFeatures(im,'CellSize', [4 4]);
    end

    trainingLabels = trainingSet.Labels;
    classifier = fitcecoc(trainingFeatures, trainingLabels);

end

function prediction = predictUsingHOG(img, classifier)
```

```
    img = imresize(img, [32,32]);
    img = imdilate(img, strel('disk',1));
    [Features] = extractHOGFeatures(imbinarize(img), 'CellSize', [4 4]);
    predictedLabels = predict(classifier, Features);
    prediction = mode(predictedLabels);

end
```