Software de Reconocimiento De Placas Vehiculares con Lógica Difusa en Matlab

A. Delgado. *Student at Ricardo Palma University-Peru, alvarodelgado15@outlook.com*
Adviser P. Huamaní, member Ricardo Palma University-Perú, phuamani@urp.edu.pe

Abstract. - There is no denying that there are a lot of security issues in our country, and car theft is not an exception. Security in parking lots has improved in the last few years, however, this brings an increment in parking fee costs. I propose the use of a vehicle license plate recognition software to lessen these costs and improve security, having the possibility to access a database of stolen cars and check if the incoming car is on the list. This software would have to be able to recognize every type of license plate here in Lima, which is a challenging task because of all the varieties we have, and the fact that the "O", "0" and "D" are fairly similar. This can be solved with the use of Fuzzy Logic Edge Detection. In the following paper, I'll explain the work done to code this software.

***Palabras Claves*— Reconocimiento, Seguridad, Vehículos.**

I. INTRODUCCIÓN

No es novedad hablar de las deficiencias en el sistema de estacionamientos que tenemos en nuestra ciudad. Según el diario Correo en 2015, tan solo entre Surco, Miraflores, San Isidro y San Borja, se tiene un déficit de 40000 espacios de estacionamiento. Mariana Alegre, coordinadora del observatorio "Lima Cómo Vamos", menciona que esto se solucionará solamente con el control del precio de parqueo, eliminación del parqueo gratuito, promoción del manejo de bicicletas, etc. Su opinión se fundamente en la comparación con la situación de Londres, donde el edificio empresarial más grande solo tiene 40 espacios para autos. Explícitamente, la crisis del parqueo de Lima se debe a que tener parqueos por toda la ciudad promueve el uso de los automóviles en vez de otros medios de transporte, y lo que disminuirá el problema será menos parqueos, y que estos sean mejor manejados. [1]

En el espectro del mejor manejo de playas de estacionamiento, uno de los mayores requerimientos es la seguridad y el conteo de la cantidad de espacios disponibles.

En el primer caso, mantener una base de datos de los códigos de la placa de cada automóvil dentro del parqueo supone un control preciso de tiempos de entrada y salida, además de permitir acceder a la lista de autos requisitoriados e identificar si alguno de los autos en la playa de estacionamiento está relacionado a algún delito.

También, en caso del conteo de espacios disponibles, varios estacionamientos han tomado la iniciativa de colocar sensores en el parqueo para identificar qué espacios están libres. Sin embargo, estos sensores solo iluminan un pequeño LED cerca al espacio libre, y no lleva un conteo de los espacios disponibles al personal en la entrada, de manera que estos no son capaces de saber si el estacionamiento está lleno o no.

La propuesta de un software de reconocimiento de placas vehiculares soluciona estos problemas, siendo que mantiene una base de datos en Excel que puede ser manejada de la manera que el cliente requiera, y con esto es capaz de hacer el procesamiento mencionado, ya sea revisar si el auto está relacionado a un delito, llevar un conteo de la cantidad de autos en la playa, o ambos.

II. PROCESAMIENTO DE IMÁGENES

El algoritmo inicia tomando la imagen capturada y convirtiéndola en escala de grises, luego procede a dilatar la imagen ligeramente y transformarla en imagen de blanco y negro. La dilatación en este caso se está utilizando para reducir el ruido y/o suavizar la imagen. Además, el transformar la imagen a blanco y negro permite eliminar bastantes de las características de la misma, dejando únicamente un fondo blanco con formas negras. Luego, esta imagen es invertida para hacer que el fondo sea negro (ceros), y las formas sean blancas (unos), tal que se puede volver a hacer un procesamiento morfológico. Esto se visualiza en las figuras 1, que será la imagen sin modificar, y la figura 2, que será la imagen alterada.

Figura 1. Imagen de ejemplo sin alterar.

Fuente: Elaboración propia.

Observamos que hay varias características que debemos eliminar, como la bandera, el recuadro negro en la esquina superior derecha, los caracteres de "PERU", los círculos negros de los tornillos, y los caracteres de la parte inferior. Además, se tienen curvas en la parte trasera de los caracteres principales.

Figura 2. Imagen de ejemplo en blanco y negro.

Fuente: Elaboración propia.

La siguiente parte del algoritmo incluye procesamiento morfológico para eliminar dichas características no deseadas de la imagen, y el código se muestra a continuación en la figura 3:

```
conc=strel('disk',1);
gi=imdilate(g,conc); % Dilata g y la almacena en ji
ge=imerode(g,conc);  % Erosiona g y la almacena en ge
gdiff=imsubtract(gi,ge); % Diferencia entre gi y ge
gdiff=mat2gray(gdiff);
gdiff=conv2(gdiff,[1 1;1 1]);
gdiff=imadjust(gdiff,[0.5 0.7],[0 1],.1);
B=logical(gdiff);     % Matriz de diferencia a lógica
[a1, b1]=size(B);
figure, imshow(B)
er=imerode(B,strel('line',100,0));
figure, imshow(er)
out1=imsubtract(B,er);
F=imfill(out1,'holes');    % Rellenando los agujeros
H=bwmorph(F,'thin',1);
H=imerode(H,strel('line',3,90));
figure, imshow(H)
```

Como podemos observar, el código es sencillo y principalmente trabaja con las operaciones de dilatación y erosión. De esta manera los detalles pequeños se pierden. Además, al hacer la diferencia entre la imagen dilatada y la imagen erosionada, obtenemos los contornos de la imagen, tal que, aplicando la función de rellenar, conseguimos más claramente la figura de los caracteres.

En la figura 3 se observa el resultado de este procesamiento morfológico.

Figura 3. Imagen de ejemplo tras procesamiento morfológico.

Fuente: Elaboración propia.

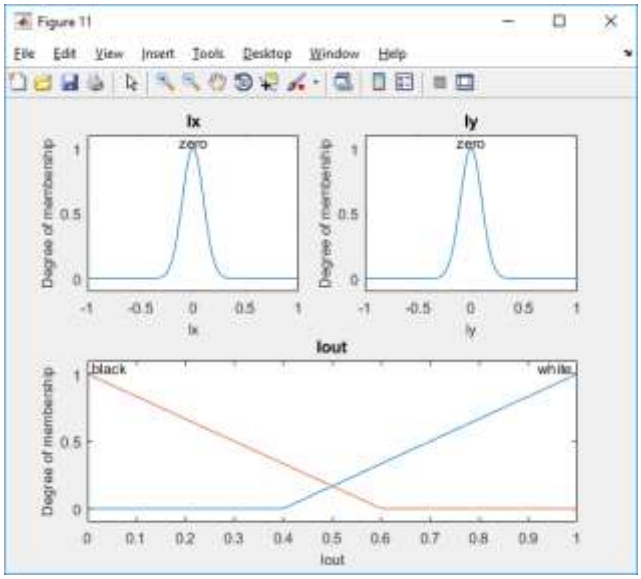
Seguidamente, el algoritmo procede a llamar nuevamente a la imagen transformada en escala de grises que se mencionó al inicio de este capítulo, tal que aplica lógica difusa.

El código de esta sección es el siguiente:

```
%% Detección de bordes con lógica difusa

        I = double(Igray);
        Gx = [-1 1];
        Gy = Gx';
        Ix = conv2(I,Gx,'same');
        Iy = conv2(I,Gy,'same');


        figure
        image(Ix,'CDataMapping','scaled')
        colormap('gray')
        title('Ix')


        figure
        image(Iy,'CDataMapping','scaled')
        colormap('gray')
        title('Iy')
        edgeFIS = newfis('edgeDetection');
        edgeFIS = addvar(edgeFIS,'input','Ix',[-1 1]);
        edgeFIS = addvar(edgeFIS,'input','Iy',[-1 1]);
        sx = 0.1;
        sy = 0.1;
        edgeFIS = addmf(edgeFIS,'input',1,'zero','gaussmf',[sx 0]);
        edgeFIS = addmf(edgeFIS,'input',2,'zero','gaussmf',[sy 0]);
        edgeFIS = addvar(edgeFIS,'output','Iout',[0 1]);
        wa = 0.4;
        wb = 1;
        wc = 1;
        ba = 0;
        bb = 0;
        bc = 0.6;
        edgeFIS = addmf(edgeFIS,'output',1,'white','trimf',[wa wb wc]);
        edgeFIS = addmf(edgeFIS,'output',1,'black','trimf',[ba bb bc]);
        figure
        subplot(2,2,1)
        plotmf(edgeFIS,'input',1)
        title('Ix')
        subplot(2,2,2)
        plotmf(edgeFIS,'input',2)
        title('Iy')
        subplot(2,2,[3 4])
        plotmf(edgeFIS,'output',1)
        title('Iout')
        r1 = 'If Ix is zero and Iy is zero then Iout is white';
        r2 = 'If Ix is not zero or Iy is not zero then Iout is black';
        r = char(r1,r2);
        edgeFIS = parsrule(edgeFIS,r);
        showrule(edgeFIS)
        Ieval = zeros(size(I));
        for ii = 1:size(I,1)
            Ieval(ii,:) = evalfis([(Ix(ii,:));(Iy(ii,:));]',edgeFIS);
        end
        Ieval = im2bw(Ieval,0.7);
        figure
        image(Ieval,'CDataMapping','scaled')
        colormap('gray')
        title('Detección de bordes con Lógica Difusa')
```

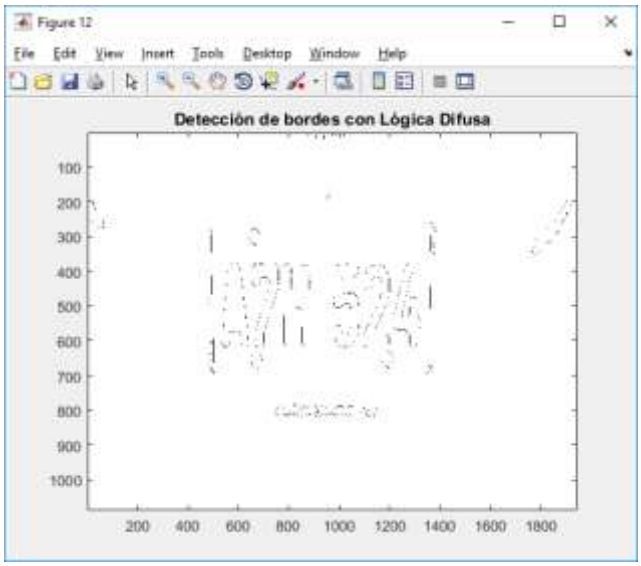
Es cierto que es una lógica difusa relativamente sencilla, sin embargo, sus efectos son notablemente importantes sobre la imagen, como se observa en las figuras 4, 5 y 6.

Figura 4. Gráficas de grados de pertenencia – Lógica Difusa.



Fuente: Elaboración propia.

Figura 5. Imagen de ejemplo tras procesamiento difuso.



Fuente: Elaboración propia.

Finalmente, estos bordes son dilatados y restados con la imagen 3, tal que solo quedan los caracteres principales. Esto se observa en la figura 6.

Figura 6. Imagen final tras procesamiento.

Fuente: Elaboración propia.

III. RECONOCIMIENTO DE CARACTERES Y ALMACENAMIENTO EN BASE DE DATOS

El reconocimiento de los caracteres ahora se hace sencillo gracias al trabajo de procesamiento que se ha hecho en la imagen. Nótese que los caracteres tienen formas correctas, con las partes planas siendo planas y las curvas bien detalladas. Esto permite un correcto reconocimiento de los caracteres sin la confusión mencionada al inicio de este documento.

El algoritmo accede a una base de datos de caracteres preestablecidos en forma de imagen .bmp, y compara cada carácter reconocido con estas muestras. Cuando existe coincidencia, almacena el carácter en un vector.

Finalmente, el vector obtiene todos los caracteres y los muestra en pantalla, mientras procesa el almacenamiento de los mismos en una hoja de Excel.

Si la placa leída ya ha sido leída antes, y el código ya se encuentra en la base de datos, el algoritmo no lo vuelve a escribir, y envía una alerta a la ventana de trabajo. Esta acción se puede configurar para cualquier requerimiento del usuario.

Lo que se visualiza en la ventana de trabajo se muestra en la figura 7:

Figura 8. Imagen final tras procesamiento.

```
placa =

A7H524

Coincidencia encontrada en posición: 4
Ingrese 1 para tomar foto, 0 para finalizar:
```

Fuente: Elaboración propia.

En la figura 8 se muestra un ejemplo de la base de datos creada en los ensayos del algoritmo.

Figura 8. Imagen final tras procesamiento.

	A	B	C	D	E	F
1	1	A	6	8	2	A
2	A	V	A	6	0	2
3	R	I	H	1	6	6
4	A	7	H	5	2	4
5	A	1	A	6	0	2
6	A	K	H	3	4	3
7	A	F	R	4	Z	0
8						

Fuente: Elaboración propia.

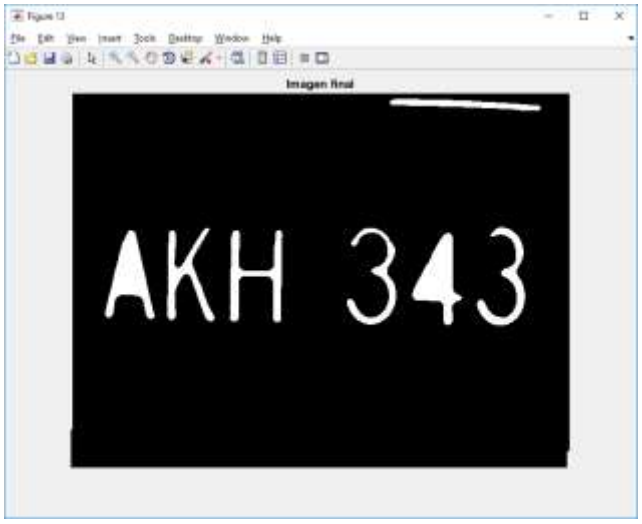
Cabe destacar que efectivamente la placa A7H524 se encuentra en la fila 4 de la base de datos.

Para finalizar, y comprobar que el algoritmo funciona correctamente, se muestran las figuras 9, 10, 11, 12, las cuales muestran más ejemplos comparativos entre imagen original e imagen final.

Figura 9. Segunda imagen de ejemplo sin modificar.

Fuente: Elaboración propia.

Figura 10. Segunda imagen de ejemplo procesada totalmente.



Fuente: Elaboración propia.

Figura 11. Tercera imagen de ejemplo sin modificar.

Fuente: Elaboración propia.

Figura 11. Tercera imagen de ejemplo procesada totalmente.

Fuente: Elaboración propia.

IV. CONCLUSIONES

a) Es posible desarrollar un software de reconocimiento de placas vehiculares con capacidad de almacenar el código en una base de datos.

b) El procesamiento morfológico es sumamente importante para el correcto reconocimiento de los caracteres.

c) Es posible aún mejorar sustancialmente el software en cuanto a tiempo de procesamiento, calidad del procesamiento, y versatilidad.

V. BIBLIOGRAFÍA

[1] Leyton, F. (2015, 01 de marzo). Crisis de parqueos ahoga a cuatro distritos de Lima. Correo. Extraído el 1 de Julio, 2017, de http://diariocorreo.pe/ciudad/crisis-de-parqueos-ahoga-a-cuatro-distritos-de-lima-568663/

VI. BIOGRAFÍA

Alvaro Gonzalo Delgado Boza, estudiante de Ingeniería Mecatrónica – URP Perú, 8vo Ciclo.
Especialista en sistemas de video-seguridad MOBOTIX y asesor de seguridad corporativa – Pre-Visión S.A.C.
alvarodelgado15@outlook.com (+51) 997 556 713