RECONOCIMIENTO DE CARACTERES ÓPTICOS (OCR) USANDO MATLAB

Diego Barragán, Pablo Vallejo
UNIVERSIDAD TÉCNICA PARTICULAR DE LOJA
ESCUELA DE ELECTRÓNICA Y TELECOMUNICACIONES
dobarragan@utpl.edu.ec, prvallejo@utpl.edu.ec

Resumen.- *el presente artículo muestra la forma de usar Matlab y funciones de su toolbox de procesamiento de imágenes para reconocer una palabra o conjunto de palabras y números en una imagen.*

Palabras clave.- correlación, OCR, IPT, filtrado, imagen binaria, imagen RGB.

INTRODUCCIÓN

La tecnología OCR proporciona a los sistemas de reproducción por escáner y sistemas de imágenes la habilidad de convertir imágenes de caracteres en letra de máquina, en caracteres capaces de ser interpretados o reconocidos por una computadora. Así, las imágenes de caracteres en letra de máquina son extraídas de un mapa de bits de la imagen reproducida por el escáner [1].

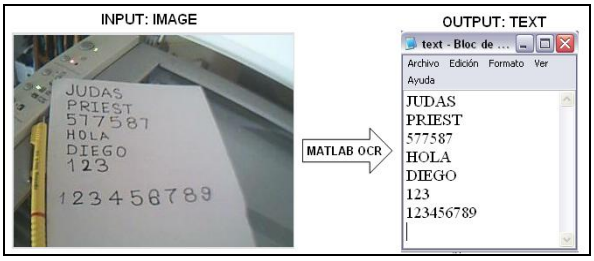

Fig.1. Esquema de un OCR.

El proceso OCR envuelve varios aspectos como segmentación, extracción de características y clasificación [2].

Image Processing Toolbox proporciona a MATLAB un conjunto de funciones que amplía las capacidades del producto para realizar desarrollo de aplicaciones y de nuevos algoritmos en el campo del proceso y análisis de imágenes. El entorno matemático y de creación de MATLAB es ideal para el procesado de imágenes, ya que estas imágenes son, al fin y al cabo, matrices. Este toolbox incorpora funciones para:

- Diseño de filtros.
- Mejora y retocado de imágenes.
- Análisis y estadística de imágenes.
- Operaciones morfológicas, geométricas y de color.
- Transformaciones 2D.

El procesamiento de imágenes es un campo de trabajo absolutamente crucial para aquellos colectivos e industrias que estén trabajando en áreas como diagnóstico médico, astronomía, geofísica, ciencias medioambientales, análisis de datos en laboratorios, inspección industrial, etc. [3].

DESARROLLO DEL PROGRAMA

SEGMENTACIÓN

Como primer paso, se recorta la imagen para ajustarla al texto. Luego de esto, se separa línea por línea. La función que realiza el recorte de la imagen se muestra a continuación:

```
function imgn=clip(imagen)
% Crops a black letter with white background.
%Example:
% imagen=imread('metal.bmp');
% imgn=clip(imagen);
% subplot(2,1,1);imshow(imagen);
%title('INPUT IMAGE')
% subplot(2,1,2);imshow(~imgn);
%title('OUTPUT IMAGE')
if ~islogical(imagen)
    imagen=im2bw(imagen,0.99);
end
a=~imagen;
[f c]=find(a);
lmaxc=max(c);lminc=min(c);
lmaxf=max(f);lminf=min(f);
imgn=a(lminf:lmaxf,lminc:lmaxc);%Clip image
```

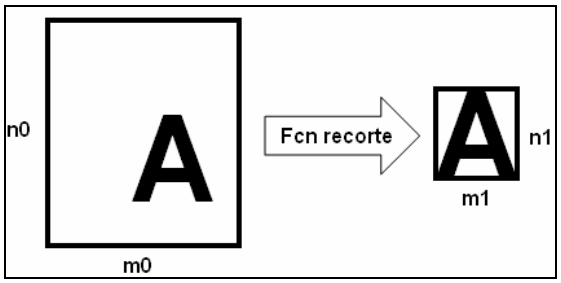

Fig.2. Esquema de función que recorta la imagen al tamaño de la letra.

Tal como se ve en la función, el umbral para la transformación a imagen binaria es 0.99 (`bn=im2bw(imagen,0.99)`). Este umbral se tomó para que colores con valores RGB muy cercanos a 255 (valor máximo) sean considerados como 0 en la imagen binaria.

Una vez recortada la imagen, el siguiente paso es separar cada línea. Para esto se usó la siguiente función:

```
function [fl re]=lines(aa)
%Divide text in lines.
```

```
%aa->input image; fl->first line;re->remain
line
%Example:
%aa=imread('heavy_metal.bmp');
%[fl re]=lines(aa);
%subplot(3,1,1);imshow(aa);
%title('INPUT IMAGE')
%subplot(3,1,2);imshow(fl);title('FIRST LINE')
%subplot(3,1,3);imshow(re);
%title('REMAIN LINES')
aa=clip(aa);
[r c]=size(aa);
for s=1:r
    if sum(aa(s,:))==0
        nm=aa(1:s-1,1:end);%First line matrix
        rm=aa(s:end,1:end);%Remain line matrix
        fl=~clip(~nm);
        re=~clip(~rm);
     %Uncomment lines below to see the result
        %         subplot(2,1,1);imshow(fl);
        %         subplot(2,1,2);imshow(re);
        break
    else
        fl=~aa;% Only one line.
        re=[];
    end
end
```

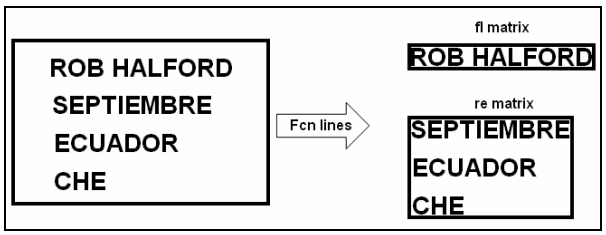

Fig.3. Esquema de la función para separar líneas en la imagen.

EXTRACCIÓN

Una vez obtenidas por separado cada línea de la imagen, se procede a extraer letra por letra de la imagen matriz *fl*. Para esto se usó la función *bwlabel*, la cual etiqueta los componentes conectados de la imagen. En otras palabras, esta función cuenta los trazos continuos y los enumera. Para separar cada letra se usó el siguiente código:

```
%-*-Calculating connected components*-*-*-*-*
    %Code from:
%http://www.mathworks.com/matlabcentral/fileex
change/loadFile.do?objectId=8031&objectType=FI
LE
    L = bwlabel(imgn);
    mx=max(max(L));
    BW = edge(double(imgn),'sobel');
    [imx,imy]=size(BW);
    for n=1:mx
        [r,c] = find(L==n);
        rc = [r c];
        [sx sy]=size(rc);
        n1=zeros(imx,imy);
        for i=1:sx
            x1=rc(i,1);
            y1=rc(i,2);
            n1(x1,y1)=255;
        end
%*-*-END Calculating connected components*-*-
```

Luego cada letra es normalizada a un tamaño de 42 x 24 píxeles, que es el tamaño de la plantilla con la que realizará la correlación. Para la normalización se usó la siguiente función:

```
function img_r=same_dim(imagen_g)
%Example:
% imagen_g=imread('a_reducir.bmp');
% img_r=same_dim(imagen_g);
% subplot(2,1,1);imshow(imagen_g);
% title('Image m x n')
% subplot(2,1,2);imshow(img_r);
% title('Image 42 x 24')
[a b]=size(imagen_g);
img_r=imagen_g(1:a/42:end,1:b/24:end);
```

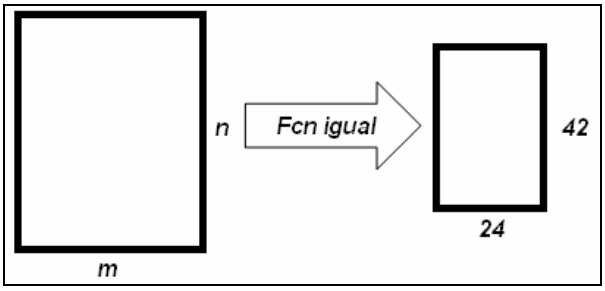

Fig.4. Esquema de la función de normalización de tamaño de muestra.

CLASIFICACIÓN

La operación principal que se empleó para la clasificación fue la correlación en dos dimensiones. Esta operación da un valor de la semejanza entre dos matrices (imágenes). La función *corr2* desarrolla esta operación según la siguiente ecuación [4]:

$$r = \frac{\sum_m \sum_n \left(A_{mn} - \overline{A}\right)\left(B_{mn} - \overline{B}\right)}{\sqrt{\left(\sum_m \sum_n \left(A_{mn} - \overline{A}\right)^2\right)\left(\sum_m \sum_n \left(B_{mn} - \overline{B}\right)^2\right)}}$$

Donde $\overline{A}$ =mean2(A) y $\overline{B}$ =mean2(B).

La siguiente función realiza la correlación entre las plantillas y cada letra extraída:

```
function letter=read_letter(imagn)
%Compute the correlation between template and
%input image and its output is a string
containing the letter.
%Size of 'imagn' must be 42 x 24 pixels
%Example:
% imagn=imread('D.bmp');
% letter=read_letter(imagn)
comp=[];
load templates
for n=1:36
    sem=corr2(templates{1,n},imagn);
    comp=[comp sem];
end
vd=find(comp==max(comp));
%*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-
if vd==1 letter='A';
elseif vd==2 letter='B';
elseif vd==3 letter='C';
elseif vd==4 letter='D';
elseif vd==5 letter='E';
elseif vd==6 letter='F';
```

```
elseif vd==7 letter='G';
elseif vd==8 letter='H';
elseif vd==9 letter='I';
elseif vd==10 letter='J';
elseif vd==11 letter='K';
elseif vd==12 letter='L';
elseif vd==13 letter='M';
elseif vd==14 letter='N';
elseif vd==15 letter='O';
elseif vd==16 letter='P';
elseif vd==17 letter='Q';
elseif vd==18 letter='R';
elseif vd==19 letter='S';
elseif vd==20 letter='T';
elseif vd==21 letter='U';
elseif vd==22 letter='V';
elseif vd==23 letter='W';
elseif vd==24 letter='X';
elseif vd==25 letter='Y';
elseif vd==26 letter='Z';
%*-*-*-*-**-*-*-*-**-*-*-
elseif vd==27 letter='1';
elseif vd==28 letter='2';
elseif vd==29 letter='3';
elseif vd==30 letter='4';
elseif vd==31 letter='5';
elseif vd==32 letter='6';
elseif vd==33 letter='7';
elseif vd==34 letter='8';
elseif vd==35 letter='9';
else   letter='0';
end
```

PLANTILLAS

Cada plantilla es una imagen binaria bmp de 42 x 24 píxeles. El script para almacenar las plantillas en una estructura de celda es el siguiente:

```
%CREATE TEMPLATES
%Letter
A=imread('A.bmp');B=imread('B.bmp');
C=imread('C.bmp');D=imread('D.bmp');
E=imread('E.bmp');F=imread('F.bmp');
G=imread('G.bmp');H=imread('H.bmp');
I=imread('I.bmp');J=imread('J.bmp');
K=imread('K.bmp');L=imread('L.bmp');
M=imread('M.bmp');N=imread('N.bmp');
O=imread('O.bmp');P=imread('P.bmp');
Q=imread('Q.bmp');R=imread('R.bmp');
S=imread('S.bmp');T=imread('T.bmp');
U=imread('U.bmp');V=imread('V.bmp');
W=imread('W.bmp');X=imread('X.bmp');
Y=imread('Y.bmp');Z=imread('Z.bmp');
%Number
one=imread('1.bmp');  two=imread('2.bmp');
three=imread('3.bmp');four=imread('4.bmp');
five=imread('5.bmp'); six=imread('6.bmp');
seven=imread('7.bmp');eight=imread('8.bmp');
nine=imread('9.bmp'); zero=imread('0.bmp');
%*-*-*-*-*-*-*-*-*-*-*-
letter=[A B C D E F G H I J K L M...
        N O P Q R S T U V W X Y Z];
number=[one two three four five...
        six seven eight nine zero];
character=[letter number];
templates=mat2cell(character,42,[24 24 24 24
24 24 24 ...
    24 24 24 24 24 24 24 ...
    24 24 24 24 24 24 24 ...
    24 24 24 24 24 24 24 ...
    24 24 24 24 24 24 24 24]);
save ('templates','templates')
```

PROGRAMA PRINCIPAL

El programa principal se muestra a continuación:

```
%OCR (Optical Character Recognition).
%PRINCIPAL PROGRAM
%Private Technical University of Loja
% (ECUADOR-SUDAMERICA)
%***********************************
warning off,clc, close all, clear all
imagen=imread('judas.jpg');%Read Binary Image
%*-*-*Filter Image Noise*-*-*-*
if length(size(imagen))==3 %RGB image
    imagen=rgb2gray(imagen);
end
imagen = medfilt2(imagen);
[f c]=size(imagen);
imagen (1,1)=255;
imagen (f,1)=255;
imagen (1,c)=255;
imagen (f,c)=255;
%*-*-*END Filter Image Noise*-*-*-*
word=[];%Matrix to store word from image
re=imagen;
fid = fopen('text.txt', 'wt');%Open text.txt as
file for write
while 1
    [fl re]=lines(re);%Fcn 'lines' separate
                      %lines in text
    imgn=~fl;
    %*-*Uncomment line below to see
       %sentences*-*-*-*
    %imshow(fl);pause(1)
    %*-*--*-*-*-*-*-*-
    %*-Calculating connected components*-*
    %Code from:
%http://www.mathworks.com/matlabcentral/fileexch
ange/loadFile.do?objectId=8031&objectType=FILE
    L = bwlabel(imgn);
    mx=max(max(L));
    BW = edge(double(imgn),'sobel');
    [imx,imy]=size(BW);
    for n=1:mx
        [r,c] = find(L==n);
        rc = [r c];
        [sx sy]=size(rc);
        n1=zeros(imx,imy);
        for i=1:sx
            x1=rc(i,1);
            y1=rc(i,2);
            n1(x1,y1)=255;
        end
        %-END Calculating connected components*-
        n1=~n1;
        n1=~clip(n1);
        img_r=same_dim(n1);%To size 42 X 24
        %Uncomment line below to see
        %letters
        % imshow(img_r);pause(1)
        %*-*-*-*-*-*-*-*
        letter=read_letter(img_r);%img to text
        word=[word letter];
    end
    fprintf(fid,'%s\n',word);%Write 'word' in
                             %text file
    word=[];%Clear 'word' variable
    %*-*When sentences finish, breaks the loop**
    if isempty(re)%See variable 're'
                  %in Fcn 'lines'
        break
    end
    %*-*-*-*-*-*--*-*-*-*-*-*-*-*--*-*-*-*-*-*-*-*-*-*
end
fclose(fid);
winopen('text.txt')%Open 'text.txt' file
```

ENSAYO DEL PROGRAMA

Una vez terminado el programa, al ejecutarse puede usarse las funciones *tic* y *toc* al inicio y final del código para medir su tiempo de respuesta. Cuando termine la ejecución se abrirá un archivo de texto que contiene las palabras de la imagen:

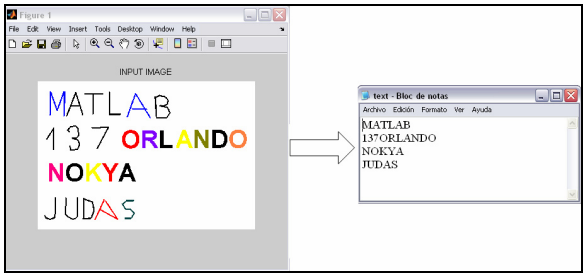

Fig. 5. Resultado del programa.

Otro ensayo que se realizó fue con una hoja escaneada con una resolución de 200 PPP y un brillo de 128 (parámetros por defecto del escáner) que contenía varias palabras manuscritas y algunos números. La muestra se indica en la figura siguiente:

Fig. 6. Texto de prueba a escanear.

Al escanear la imagen se produce cierto ruido (puntos negros) en ésta. La solución es implementar un filtro al inicio del código para eliminar el ruido.

```
%*-*-*Filter Image Noise*-*-*-*
if length(size(imagen))==3 %RGB image
    imagen=rgb2gray(imagen);
end
imagen = medfilt2(imagen);
[f c]=size(imagen);
imagen (1,1)=255;
imagen (f,1)=255;
imagen (1,c)=255;
imagen (f,c)=255;
%*-*-*END Filter Image Noise*-*-*-*
```

Al aplicar el programa se obtuvo los siguientes resultados:

Fig. 7. Texto reconocido.

Se puede deducir que el error presente en los tres primeros sietes se debe a la pequeña curvatura que posee la plantilla, que sí la posee el último 7.

CODE METRICS

MATLAB posee dos herramientas que ayudan a mejorar el código. La una es el *profile*, la cual, entre otras cosas, calcula el tiempo de ejecución de cada función, y muestra que variables puedes descartarse del código para acelerarlo.

La segunda herramienta es *code metrics*, un programa que está disponible en el *File Exchange* de *MathWorks* [5]. Este programa determina si los nombres de las funciones que estamos usando no tienen conflicto con funciones ya definidas en el path del MATLAB, una medida de la complejidad del programa (*cyclomatic complexity*) y sugerencias muy prácticas para mejorar el funcionamiento del programa.

El resultado de aplicar *code metrics* al OCR es el siguiente:

Summary

Criteria	Maximum Cyclomatic Complexity	Average M-Lint Messages	Average Help Warnings	Total Filename Clashes
Raw Score (lower is better)	37	0.2	0.3	1

Detailed Metrics

Filename	Line Count	Percent Comments	Cyclomatic Complexity	M-Lint Messages	Help Warnings
OCR.m	78	40%	6		1
clip.m	14	50%	1		
create_templates.m	28	14%	1		1
lines.m	29	48%	3		
read_letter.m	89	9%	37	1	
same_dim.m	8	63%	1		

CONCLUSIONES

- El tamaño de la letra debe no ser menor a 42 x 24 píxeles.
- La imagen de entrada puede ser con letras de color o no.
- Si la imagen presenta ruido en los bordes mayor al que puede filtrar la función *medfilt2*, será necesario usar la función *imcrop* para recortar la imagen.

- Si el tamaño de la letra es delgado, se debe usar la función *imdilate* para aumentar su grosor antes de pasar al programa.
- Para tener como resultado un texto en minúsculas, se usa la función *lower* sobre la variable *word* del programa principal.
- El tiempo de procesamiento del texto de prueba de la figura 6 fue: >> *tic;OCR;toc Elapsed time is 3.806446 seconds.* Sin embargo, en un PC con 1 GB de RAM el tiempo de procesamiento fue de 2.4 segundos.
- Este tiempo de procesamiento se reduciría si se usa una plantilla de 20 x 20 píxeles.

REFERENCIAS

[1] http://www.pearsonncs.com/
[2] A MATLAB PROJECT IN OPTICAL CHARACTER RECOGNITION (OCR), Jesse Hansen
[3] http://www.eldish.net/hp/automat/matlab.htm
[4] RECONOCEDOR DE DÍGITOS MANUSCRITOS, Marcelo Valdiviezo C., Revista "En Corto Circuito", núm. 11, Enero de 2007.
[5] Link de code metrics en mathworks: http://www.mathworks.com/matlabcentral/fileexchange/loadFile.do?objectId=10514&objectType=file

BIOGRAFÍAS

Diego Barragán Guerrero
Profesional en formación
E-mail: dobarragan@utpl.edu.ec
Experiencias: editor de la Revista de Electrónica y Telecomunicaciones "En Corto Circuito", actualmente se desempeña como programador independiente.

Pablo Vallejo Zúñiga
Profesional en formación
E-mail: prvallejo@ieee.org
Experiencias: gestión productiva en el departamento de Electricidad y Sistemas Electrónicos en Diseño y Mantenimiento Eléctrico.