Visió per computador

Sessió 9 de Laboratori

Tardor 2019

Martí Ferret

Maria Vila

Aquest laboratori té com a objectiu reconèixer un conjunt de lletres i números donada la imatge següent com a mostra:

I realitzar els tests amb les següents imatges (deformades) per comprovarl'eficàcia del programa.

Per la realització d'aquest exercici hem separat el programa en un codi principal i quatre funcions, d'aquesta manera aconseguim que l'exercici sigui més entenedor i més còmode a l'hora de treballar-lo.

Codi main.m

En el codi del main llegim les imatges de mostra i test corresponents, ens creem un vector **`clauSample`** on l'índex ens indica la posició que ocupa cada caracter en la imatge de mostra i el valor la seva representació en *chars.* Procedim a obtenir les característiques de cada lletra de les imatges de mostra i de test amb la funció `obtain_text_characteristics(I)`. Finalment ens creem la sortida **`clauTest`** (vector amb el mateix format que clauSample però de la imatge de Test) cridant a la funció `nearest_neighbour_of_text(...)` que implementa l'algorisme de veïns més propers. Per últim generem la matriu de confusió i calculem l'error cridant a la funció `obtain_confusion_matrix(...)`. Es pot observar a continuació el codi que hem desenvolupat.

```
% Llegim la imatge de Mostra i la imatge de Test
Imostra = rgb2gray(imread('Joc_de_caracters.jpg'));
Itest = rgb2gray(imread('Joc_de_caracters_deformats.jpg'));
% Creem les llistes de caracteristiques
clauSample = ['0'; '1'; '2'; '3'; '4'; '5'; '6'; '7'; '8'; '9'; 'B';
'C'; 'D'; 'F'; 'G'; 'H'; 'J'; 'K'; 'L'; 'M'; 'N'; 'P'; 'R'; 'S'; 'T';
'V'; 'W'; 'X'; 'Y'; 'Z'];
% Obtenim normalized characteristics of sample image
caracteristiquesSample = obtain_text_characteristics(Imostra);
% Obtenim normalized characteristics of test image
caracteristiquesTest = obtain_text_characteristics(Itest);
% Veí més proper (Distancia euclideana)
clauTest = nearest_neighbour_of_text(clauSample, caracteristiquesSample,
caracteristiquesTest);
% Creem la matriu de confusió, obtenim l'error de test i les confusions
[matriu_confusio, error] = obtain_confusion_matrix(clauSample,
clauTest);
error % printem l'error
```

Funció obtain_text_characteristics.m

Aquesta funció rep com a paràmetres d'entrada una imatge **`I`** i retorna una matriu de **característiques** on cada fila correspon a les característiques d'una lletra. Per fer-ho binaritzem la imatge d'entrada i executem regionprops, llavors per cada lletra Calculem la seva BoundingBox i ampliem amb + 1 el seu marc; i amb imcrop, obtenim la imatge de la lletra a tractar. Procedim a cridar la funció obtaincharacteristics amb la imatge com a paràmetre d'entrada per obtenir les característiques d'aquesta lletra i les afegim a la matriu. Per últim necessitem normalitzar els valors de la matriu de característiques, així totes tindran el mateix pes i podrem aplicar la distància euclidiana. El codi de la funció es pot veure a continuació:

```
function [ caracteristiques ] = obtain_text_characteristics( I )
    % Binaritzem imatge
    I = I<180;
    % Cridem a region props per obtenir les imatges de les lletres
    LlistaLletres = regionprops(I, 'BoundingBox');
    % Caracteristiques es una matriu
    caracteristiques = [];
    for i = 1:30 % per cada lletra fem:
        %obtenim la BB de la lletra
        BoundingBoxLletra = LlistaLletres(i).BoundingBox;
        %augmentem els marcs
        BoundingBoxLletra = BoundingBoxLletra + [-1 -1 1 1];
        %obtenim la imatge de la lletra
        Illetra = imcrop(I, BoundingBoxLletra);
        %calculem les caracteristiques d'aquesta
        carac = obtaincharacteristics(Illetra);
        %afegim el valor a la variable de retorn
        caracteristiques = [caracteristiques; carac];
    end
    %Normalitzem les caracteristiques perque totes tinguin un rang
similar (entre 0 i 1)
    [f, c] = size(caracteristiques);
    for i =  1:c
        carac = caracteristiques(:, i);
        maxim = max(carac);
        minim = min(carac);
        for j = 1:f
            caracteristiques(j, i) = (maxim - caracteristiques(j, i)) /
(maxim - minim);
        end
    end
end
```

Funció obtaincharacteristics.m

Aquesta funció rep com a paràmetres d'entrada una Imatge **`I`** (que és una lletra). Retorna un vector de 8 característiques, totes elles invariants a la mida, que són:

- **Número de forats**: amb la funció bwconncomp obtenim el nombre de components connexos, per tant en imatge complementaria seran el nombre de forats -1 per treure el background.
- **Rectangularitat**: és l'àrea que ocupa la superfície del nombre partida per l'àrea total de la bounding box, ens ajudem de les propietats que hem obtingut de regionprops.
- **Distància del polígon més llarg respecte el perímetre**: com el seu nom indica és la distància màxima dels polígons simplificats dividida entre el perímetre total que formen la lletra, per obtenir-lo hem fet ús de les funcions boundary i reducem.
- **Angle del polígon més llarg**: un cop tenim el polígon més llarg podem calcular l'angle que forma respecte l'eix horitzontal.
- **Aspect Ratio de la Bounding Box:** relació d'aspecte de la bounding box.
- **Compacitat:** amb les propietats que ens retorna regionprops tenim el perímetre i l'àrea, amb aquests dos valors calculem la compacitat de la lletra a tractar.
- **Posició x del centroide**: regionprops també ens retorna el centroide de la figura, amb això calculem la seva posició respecte l'amplada de la bounding box.
- **Posició y del centroide**: calculem la posició y del centroide de la mateixa manera.

El codi implementat per aconseguir totes aquestes característiques es pot veure a continuació:

```
function [ carac ] = obtaincharacteristics( I )
    % Cridem a regionprops per aconseguir algunes propietats basiques
    Properties = regionprops(I, 'Perimeter', 'Area', 'BoundingBox',
'Centroid');
    Perimetre = Properties.Perimeter;
    BoundingBoxLletra = Properties.BoundingBox;
    Width = BoundingBoxLletra(3);
    Heigth = BoundingBoxLletra(4);
    Area = Properties.Area;
    %Numero de forats de la lletra:
    CC = bwconncomp(not(I), 8);
    % Treiem el fons com a forat
    car_num_forats = CC.NumObjects -1;
    %Rectangularitat
    car_rectangularitat = Area/(Width*Heigth);
    % simplified polygonal boundary of a BW image
    CC = bwconncomp(I);
    idxlists = CC.PixelIdxList;
    pixels = idxlists{1};
    [F,C] = ind2sub(size(I), pixels);
    % exterior boundary
    k = boundary([F,C],0.90); % loose factor = 0.15
```

```
    % reduce polygonal
    [RF,RC] = reducem(F(k),C(k),5); % tolerance = 5 degrees
    [quantitat_poligons, ~] = size(RF);
    poligon_mes_llarg = 0;
    angle_poligon_llarg =0;
    for x = 1:quantitat_poligons-1
        dist = (RF(x) - RF(x+1))^2 + (RC(x) - RC(x+1))^2;
        if dist > poligon_mes_llarg
            poligon_mes_llarg = dist;
            base = abs(RF(x) - RF(x+1));
            angle_poligon_llarg = sin(base/sqrt(dist));
        end
    end
    %Poligon mes llarg en relacio al perimetre
    car_largest_pol = poligon_mes_llarg / Perimetre;
    %Angle del poligon mes llarg
    car_angle_largest_pol = angle_poligon_llarg;
    %Aspect ratio de la BB
    car_aspect_BB = Width / Heigth;
    %Compacitat
    car_compacitat = Perimetre^2/Area;
    %Centroide x
    car_centroide_x = Properties.Centroid(1)/Width;
    %Centroide y
    car_centroide_y = Properties.Centroid(2)/Heigth;
    %Obtenim el resultat
    carac = [car_num_forats car_rectangularitat car_largest_pol
car_angle_largest_pol car_aspect_BB car_compacitat car_centroide_x
car_centroide_y];
end
```

Funció nearest_neighbour_of_text.m

Un cop disposem de la matriu de característiques de la imatge de mostra i la de test, podem procedir a executar l'algorisme del veí més proper. La funció nearest_neighbour_of_text rep com a paràmetres d'entrada el vector **clauSample** i les característiques de mostra i test. Retorna un vector **clauTest** amb els resultats calculats. El que fa la funció és comparar per cada lletra de test amb les de mostra i mira quina distància les separa, guardant-se sempre l'índex i la distància; finalment omple la posició de la lletra a tractar del vector **clauTest** amb el caràcter assignat per l'algorisme. El codi de la funció es pot veure a continuació.

```
function [ clauTest ] = nearest_neighbour_of_text( clauSample,
caracteristiquesSample, caracteristiquesTest )
    [f, c] = size(caracteristiquesSample);
    [fs, cs] = size(clauSample);
    clauTest = zeros(fs, cs);
    for lletra_test = 1:f
       nearest_neighbour = 0;
       min_dist = Inf;
       for lletra_sample = 1:f
           euclidean_distance = 0;
           for caracteristica = 1:c
               euclidean_distance = euclidean_distance +
(caracteristiquesTest(lletra_test, caracteristica) -
caracteristiquesSample(lletra_sample, caracteristica))^2;
           end
           if euclidean_distance < min_dist
               min_dist = euclidean_distance;
               nearest_neighbour = lletra_sample;
           end
       end
       clauTest(lletra_test) = clauSample(nearest_neighbour);
    end
end
```

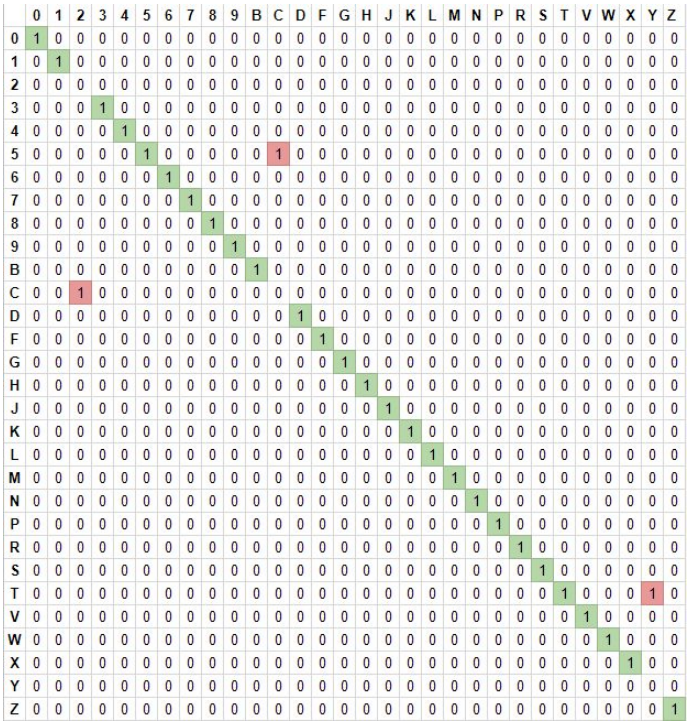
Funció obtain_confusion_matrix.m

Aquesta funció ens genera la matriu de confusió del resultat final. Per fer-ho rep com a paràmetres d'entrada la **clauSample** i la **clauTest**. Retorna la matriu i també l'error obtingut. Primer delcarem l'error i la matriu de confusió a 0. Llavors, per cada lletra de clauSample, compara les lletres de clauTest, si són iguals posem un 1 a l'elemt de la matriu indicat, si els índex són diferents sumem 1 a l'error i mostrem per pantalla la confusió que ha tingut. Finalment per tenir un error en percentatge, dividim aquest per la mida. El codi que implementa aquesta funció es pot veure a continuació.

```
function [ confusion_matrix, error ] =
obtain_confusion_matrix(clauSample, clauTest)
    [mida, ~]= size(clauSample);
    error = 0;
    confusion_matrix = zeros(mida);
    for lletra_sample = 1:mida
        for lletra_test = 1:mida
            if clauSample(lletra_sample) == clauTest(lletra_test)
                confusion_matrix(lletra_sample, lletra_test) =
confusion_matrix(lletra_sample, lletra_test) + 1;
                if lletra_sample ~= lletra_test
                    error = error + 1;
                    confon = [clauSample(lletra_sample),
clauSample(lletra_test)]
                end
            end
        end
    end
    error = error/mida;
end
```

Així doncs, amb el codi explicat anteriorment obtenim els resultats següents:

- Amb la imatge Joc_de_caracters_deformats.jpg obtenim un **10%** d'error. Concretament el nostre algorisme únicament confón el caràcter 'C' pel '5', el caràcter '2' pel 'C' i el caràcter 'T' per 'Y'. La matriu de confusió generada és:



	0	1	2	3	4	5	6	7	8	9	B	C	D	F	G	H	J	K	L	M	N	P	R	S	T	V	W	X	Y	Z
0	1	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	0	0	0	0
1	0	1	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	0	0	0
2	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	0	0	0	0	0
3	0	0	0	1	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	0
4	0	0	0	0	1	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
5	0	0	0	0	0	1	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
6	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
7	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
8	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
9	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
B	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
C	0	0	1	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	0	0
D	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
F	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
G	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
H	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0
J	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0
K	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0
L	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0
M	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0
N	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0
P	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0
S	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0
T	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	1	0
V	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	1	0	0	0	0
W	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	0	1	0	0	0
X	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	0	0	1	0	0
Y	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	0	0	0	0	0
Z	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	0	0	0	0	1

- Amb la imatge Joc_de_caracters_deformats II.png obtenim un **23,33%** d'error. Concretament el nostre algorisme confón el caràcter 'S' pel '5', el caràcter '6' pel '9', el caràcter 'K' pel 'C', el caràcter H' pel 'G', el caràcter 'Z' pel 'S', el caràcter 'M' pel 'W' i el caràcter 'X' pel '2'. La matriu de confusió generada és:

	0	1	2	3	4	5	6	7	8	9	B	C	D	F	G	H	J	K	L	M	N	P	R	S	T	V	W	X	Y	Z
0	1	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	0	0	0	0
1	0	1	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	0	0	0
2	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	0	0	0	0	0
3	0	0	0	1	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	0
4	0	0	0	0	1	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
5	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0
6	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	0	0	0	0	0
7	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
8	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
9	0	0	0	0	0	0	1	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
B	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
C	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0
D	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
F	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
G	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0
H	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	0	0	0	0	0
J	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0
K	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	0	0	0	0	0
L	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0
M	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	0	0	0	0	0
N	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0
P	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0
S	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	1
T	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0
V	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	1	0	0	0	0
W	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	1	0	0	0
X	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0
Y	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	0	0	0	1	0
Z	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	0	0	0	0	0

En resum, podem determinar que el nostre algorisme obté bons resultats ja que en mitjana té un **16,67%** d'errors.

Seguidament calcularem quina és la característica usada més dèbil. Per a fer-ho, executarem el codi vuit vegades i, en cada una, només usarem set característiques. Per tant, en cada una de les execucions treurem una característica diferent. Els resultats obtinguts es poden observar a la taula següent:

	Primer joc de proves		**Segon joc de proves**	
Característica extreta	**Error**	**Augment de l'error en relació amb l'obtingut amb 8 característiques**	**Error**	**Augment de l'error en relació amb l'obtingut amb 8 característiquess**
Número de forats	13.33%	3.33%	30%	6.67%
Rectangularitat	13.33%	3.33%	30%	6.67%
Longitud del polígon més llarg respecte al perímetre	13.33%	3.33%	46.67%	23.33%
Angle del polígon més llarg	16.67%	6.67%	26.67%	3.33%
Aspect ratio de la bounding box	10%	0%	20%	-3.33%
Compacitat	13.33%	3.33%	23.33%	0%
Posició x del centroide respecte l'amplada	20%	10%	30%	6.67%
Posició y del centroide respecte l'alçada	10%	0%	33.33%	10%

En la taula anterior es mostra l'error obtingut en cada prova i l'augment de l'error que implica treure la característica que s'està provant respecte als resultats obtinguts amb vuit característiques. És important indicar que en alguns casos les lletres confuses han variat. Per exemple, en l'execució del primer test sense la característica compacitat, els caràcters confosos són '2' per '4' i 'K' per 'F' en comptes de '2' per '5' i 'K' per 'H'.

A la taula següent es mostra la mitjana de l'error per a cada característica extreta i la mitjana de l'augment de l'error que implica treure la característica que s'està testejant.

Característica extreta	**Error mitjà**	**Augment de l'error mitjà en relació amb l'obtingut amb 8 característiques**
Número de forats	21.66%	5%
Rectangularitat	21.66%	5%
Longitud del polígon més llarg respecte al perímetre	30%	13.33%
Angle del polígon més llarg	21.66%	5%
Aspect ratio de la bounding box	15%	-1.66%
Compacitat	18.33%	1.66%
Posició x del centroide respecte l'amplada	25%	8.33%
Posició y del centroide respecte l'alçada	21.66%	5%

Amb els resultats de la taula anterior podem observar que la característica més dèbil és el aspect ratio de la bounding box. Això és degut a que el mínim augment de l'error respete a l'error obtingut amb vuit característiques és el provocat per l'extracció d'aquesta característica. Concretament, al extreure aquesta característica l'error disminueix un 1,67% pel que els resultats milloren. Això és degut probablement al fet de que el aspect ratio de la bounding box no és invariant a la rotació i un canvi en la rotació fa que el seu valor varïi molt. Per contra, la característica més forta és la longitud del polígon més llarg respecte al perímetre Això és degut a que el màxim augment de l'error respecte als resultats obtinguts amb vuit característiques es duu a terme quan extraiem aquesta característica; concretament l'error augmenta un 13,33%.