Universitat Politècnica de Catalunya

Visió per Computadors

Informe Sessió 10b: Reconeixement de caràcters

Carlota Catot Bragós
Alejandro Domínguez Besserer

Quadrimestre Tardor 2018-2019

Per resoldre el problema de reconeixement de caràcters plantejat, hem agafat la figura seguent, realitzarem un seguit de passos per tal de poder detectar quina és la lletra o número en les imatges de prova.

Figura 1: Imatge a analitzar

En primer lloc, calcularem 6 descriptors (invariants a la mida) per cadascun dels caràcters de la figura. Per començar el primer pas, primer, binaritzarem la figura amb un factor de 20 per tal de poder detectar els caràcters, un cop binaritzada utilitzarem la propietat *connected components* per poder extreure tots els caràcters i amb la propietat *regionprops* totes les seves característiques.

```
I = imread('Joc_de_caracters.jpg');
[F C] = size(I);
ndg = rgb2gray(I);
BW = ndg < 20;
imshow(BW);
cc = bwconncomp(BW);
stats = regionprops(cc,'all');
```

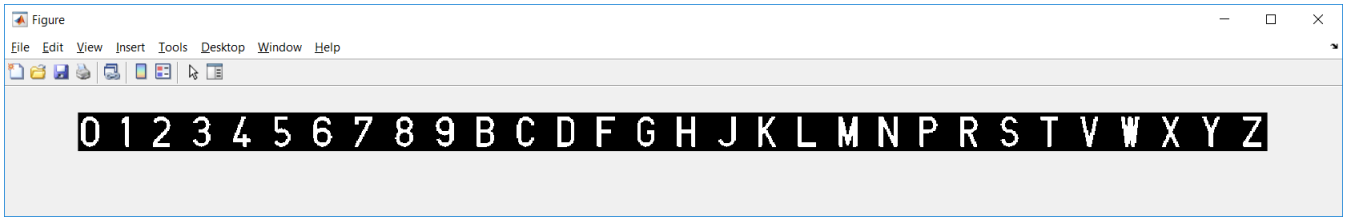

Figura 2: imshow(BW)

Ara a la variable stats tindrem totes les característiques que ofereix *regionprops* i ens quedarem amb 6 característiques, invariants a la mida, que serviran com a descriptors dels nostres caràcters. Els descriptors seleccionats han estat:

- C1: Número de forats (Amb el número d'euler)
- C2: Costat més llarg del polígon (MajorAxisLength)
- C3: Eccentricity
- C4: Relació de píxels dins de la Bounding box (Extent)
- C5: (Perimetre * Perimetre)/Area
- C6: Area/ConvexArea (Solidity)

Per tal d'utilitzar les variables, cal normalitzar-les, per tant, quan calculem cadascuna de les propietats seleccionades, calcularem el seu valor normalitzat.

```
maxMAL = max([stats.MajorAxisLength]);
maxCo = 0;
for i = 1:30
    if (stats(i).Perimeter * stats(i).Perimeter)/stats(i).Area > maxCo
        maxCo = (stats(i).Perimeter * stats(i).Perimeter)/stats(i).Area;
    end
end
```

```
for i = 1:30     %per cada caràcter calculem 8 propietats
    C1(i) = (stats(i).EulerNumber+1)/2;
    C2(i) = stats(i).MajorAxisLength/maxMAL;
    C3(i) = stats(i).Eccentricity;
    C4(i) = stats(i).Extent;
    C5(i) = ((stats(i).Perimeter * stats(i).Perimeter)/stats(i).Area)/maxCo;
    C6(i) = stats(i).Solidity;
end
```

A continuació implementarem un predictor per tal d'analitzar, a través de les propietats, quin és el caràcter en qüestió. Per començar a mirar això hem llegit i binaritzat la imatge següent, que seran el nostre joc de prova, per tal de predir sobre aquestes imatges, el resultat:

```
J = imread('Joc_de_caracters_deformats.jpg');
ndg2 = rgb2gray(J);
BW2 = ndg2 < 20;
imshow(BW2);
```

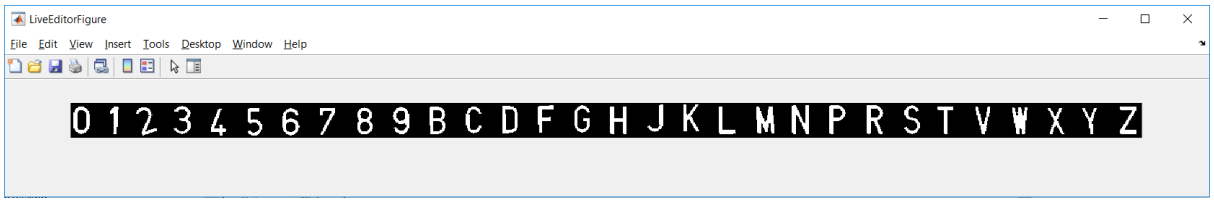

Figura 3: imshow(BW2)

Per tal que el nostre predictor analitzi quina és el caràcter més semblant a partir de les propietats, per cadascun dels elements, mirarem a quin caràcter del vector inicial de caràcters (cada propietat CX) s'assembla més, per realitzar això hem generat el següent codi (aquest codi es correrà dos cops, un per cada imatge de joc de proves):

```
cc2 = bwconncomp(BW2);
stats2 = regionprops(cc2,'all');
idxlists2 = cc2.PixelIdxList;
%regionprops deixa les propietats dels caràcters ordenates de esquerra a dreta
assig = zeros(1,30);
expected = zeros(1,30);
for i = 1:30
    minDist = 4326546;
    minIndex = 0;
    expected(i) = i;
    for j = 1:30
        distij=0;
        distij = distij + abs(C1(j) - (stats2(i).EulerNumber+1)/2);
        distij = distij + abs(C2(j) - (stats2(i).MajorAxisLength)/maxMAL);
        distij = distij + abs(C3(j) - (stats2(i).Eccentricity));
        distij = distij + abs(C4(j) - (stats2(i).Extent));
        distij = distij + abs(C5(j) - ((stats2(i).Perimeter * stats2(i).Perimeter)/stats2(i).A
        distij = distij + abs(C6(j) - (stats2(i).Solidity));

        if(distij < minDist)
            minDist = distij;
            minIndex = j;
        end
```

```
        end
        assig(i)=minIndex;
    end
```

D'aquesta manera, el vector *assig* tindrà els valors del caràcter que el nostre predictor detecta com el més semblant i per conseqüent, el detecta com a igual.

Per tal de mirar el nostre marge d'error, passem al següent pas, analitzar la matriu de confusió, per realitzar això hem utilitzat propietats pròpies de matLab, *confusionmat* i *confusionchart*, per cadascun dels jocs de proves. Per tal d'utilitzar aquesta propietat, el passem com a paràmetres una llista amb els valors esperats i la llista *assig*transposada *assig*.

```
result = transpose(assig);
C = confusionmat(expected, result);
cm = confusionchart(expected, result);
```

A partir de la matriu de confusió posada a continuació podem determinar que el nostre predictor no és gaire precís, ja que té un 77% d'accert, ja que veiem que hi ha 7 errors, per tant no podem determinar que sigui un bon predictor.

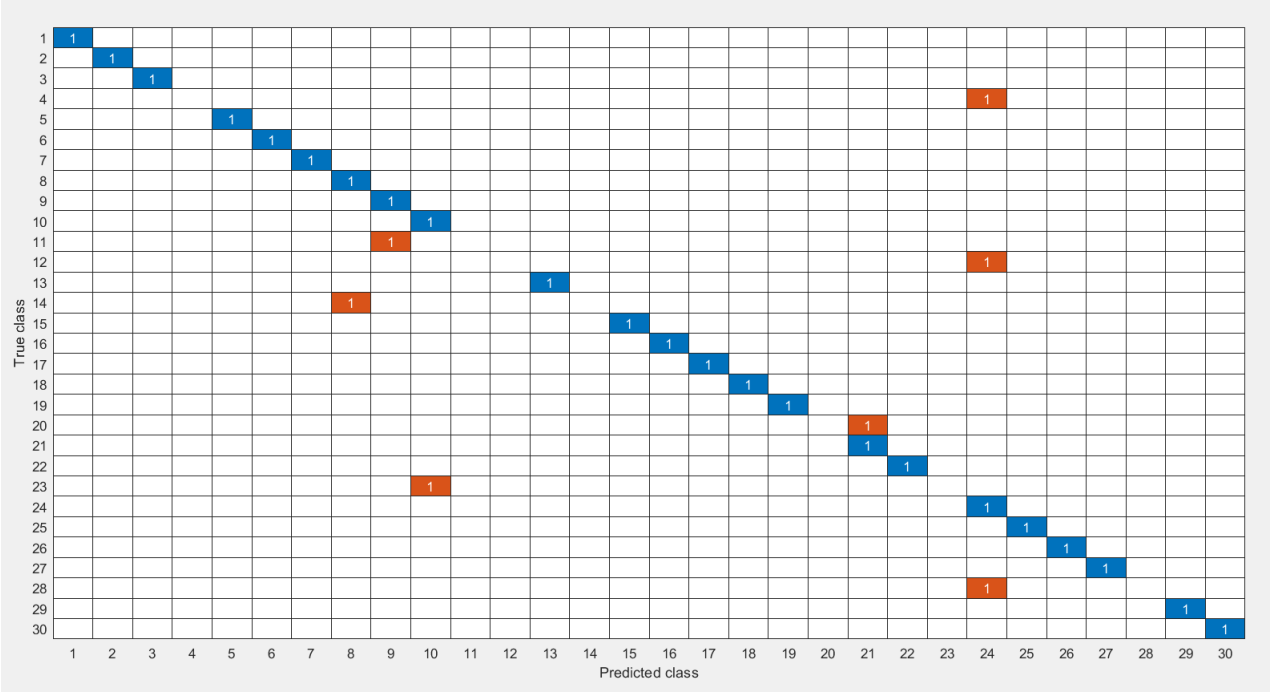

Figura 4: matriu de confusio per la imatge 'Joc_de_caracters_deformat.jpg'