Sesión 11. 09/05

Continuació local features

```
J = rgb2gray(imread("Abecedari.png"));
BW = J < 200;
imshow(BW);
```


```
BWU = BW;
BWU(end/2:end,:) = 0;
```

```
Warning: Integer operands are required for colon operator when used as index.
```

```
BWD = BW;
BWD(1:end/2,:) = 0;
```

```
Warning: Integer operands are required for colon operator when used as index.
```

```
CCU = bwconncomp(BWU);
CCD = bwconncomp(BWD);

propsU = regionprops('table',CCU,'Centroid');
propsD = regionprops('table',CCD,'Centroid');

FU = extractHOGFeatures(BWU, propsU.Centroid,'CellSize', [16,16], 'BlockSize',[3,3]);
FD = extractHOGFeatures(BWD, propsD.Centroid,'CellSize', [16,16], 'BlockSize',[3,3]);

numObj = CCU.NumObjects;

A = zeros(numObj,numObj);
for i=1:numObj
    for j =1:numObj
        A(j,i) = norm(FU(i,:) - FD(j,:));
    end
end

costUnmatched = max(A,[],'all');
Assig = matchpairs(A,costUnmatched);

imshow(BW);
```

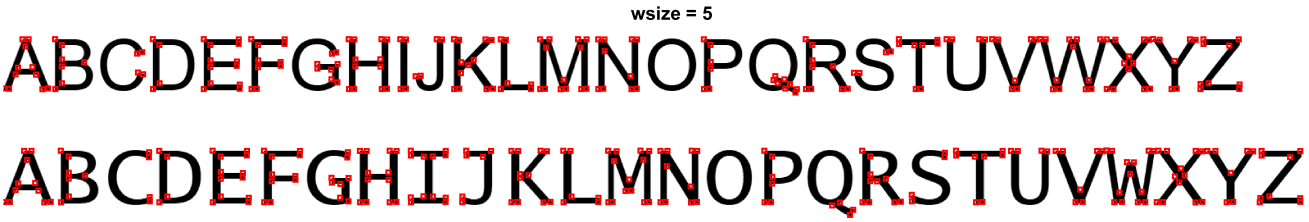
```
for i=1:numObj
    line([propsU.Centroid(i,1), propsD.Centroid(Assig(i),1)],[propsU.Centroid(i,2), propsD.Cent
end
```

Algorithm: Harris corner Detector

Implementar l'algoritme de Harris per detectar els vèrtexs de la imatge de l'abecedari.

```
im = rgb2gray(imread("Abecedari.png"));
imshow(im);
wsize = 5;
Kp = Harris(double(im),wsize);

for i=1:size(Kp,1)
    rectangle('Position',[Kp.Centroid(i,1)-wsize/2,Kp.Centroid(i,2)-wsize/2,wsize,wsize],'EdgeC
end
title('wsize = 5');
```

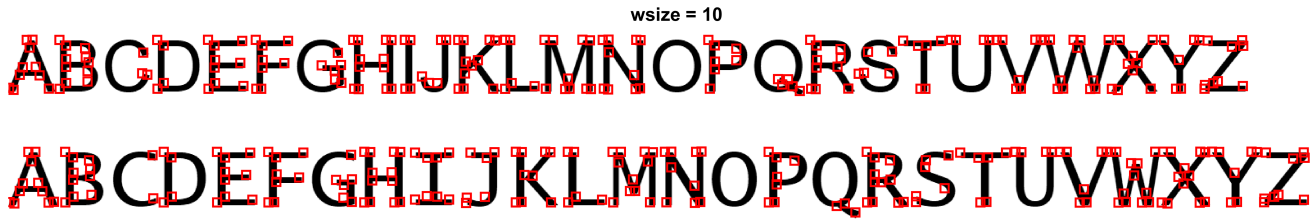


Veient la imatge resultant, podem veure correctament quin són els vèrtexs de cada lletra. Almenys amb una mida de 5. Però si augmentem la mida d'aquesta finestra, podem veure com cada cop s'aniran superposants rectangles de vèrtexs molt aprop, fins al punt, de si es molt gran, pot acabar despareixent la marcació de vertexs. Amb una mida de 10 encara podem veure com marca els vèrtexs pero es van superposant o desplaçant. Alemnys marca els vèrtexs. Però si posem una mida més gran, com per exmeple 20, ja veiem com una regió agrupa més d'un vèrtex resultant i doncs resulta ensenyant no tots els vèrtexs.

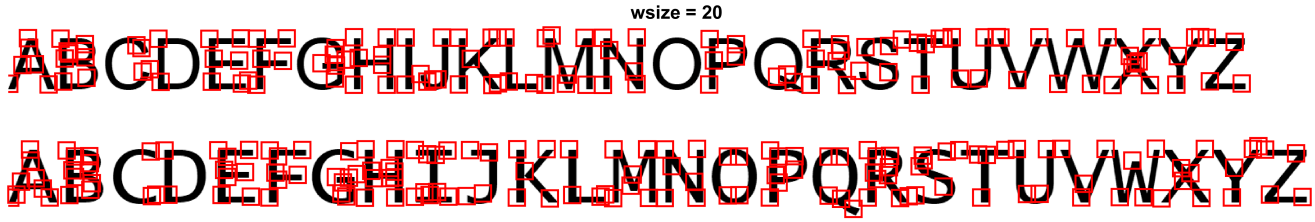
També, en funció de la mida, un vèrtex pot ser-ho o no. Per això, veiem en el cas de 20, diferents resultats que amb una mida de 5.

```
% --- test amb 10
im = rgb2gray(imread("Abecedari.png"));
imshow(im);
wsize = 10;
```

```
Kp = Harris(double(im),wsize);
for i=1:size(Kp,1)
    rectangle('Position',[Kp.Centroid(i,1)-wsize/2,Kp.Centroid(i,2)-wsize/2,wsize,wsize],'EdgeC
end
title('wsize = 10');
```




```
% --- test amb 20
im = rgb2gray(imread("Abecedari.png"));
imshow(im);
wsize = 20;
Kp = Harris(double(im),wsize);
for i=1:size(Kp,1)
    rectangle('Position',[Kp.Centroid(i,1)-wsize/2,Kp.Centroid(i,2)-wsize/2,wsize,wsize],'EdgeC
end
title('wsize = 20');
```




```
function [Kp] = Harris(im,size)
    % 1. sobel. derivada en x e y. (imatge de doubles)
    hx = [1,0,-1;2,0,-2;1,0,-1];
    hy = hx';
    Dx = imfilter(im,hx,'replicate');
    Dy = imfilter(im,hy,'replicate');

    % 2. mult Dx * Dy px a px dx2 = dx.*dx ...
    Dx2 = Dx.*Dx;
    Dy2 = Dy.*Dy;
    Dxy = Dx.*Dy;

    % 3. sum -> imfilter amb una mida que li pasem com a param
    s = fspecial('gaussian',size,size/4);
    sumDx = imfilter(Dx2,s,'replicate');
    sumDy = imfilter(Dy2,s,'replicate');
```

```
    sumDxy = imfilter(Dxy,s,'replicate');

    % 4. calcular R a cada un dels px.
    R = sumDx.*sumDy - sumDxy.^2 - 0.05*((sumDx+sumDy).^2);

    % 5. posar un llindar: ens quedem amb els maxim locals
    % T = mean(R) + 0.5desviacio estandard (std(R,'all'))
    % iR = (R == RD) & (R>T). sent RD (R dilatada)
    RD = imdilate(R,ones(2,2));
    T = mean(R,'all') + 0.5*std(R,[],'all');
    iR = R == RD & R>T;

    % regionprops(calcular els centroides)
    Kp = regionprops('table',iR,'Centroid');
end
```