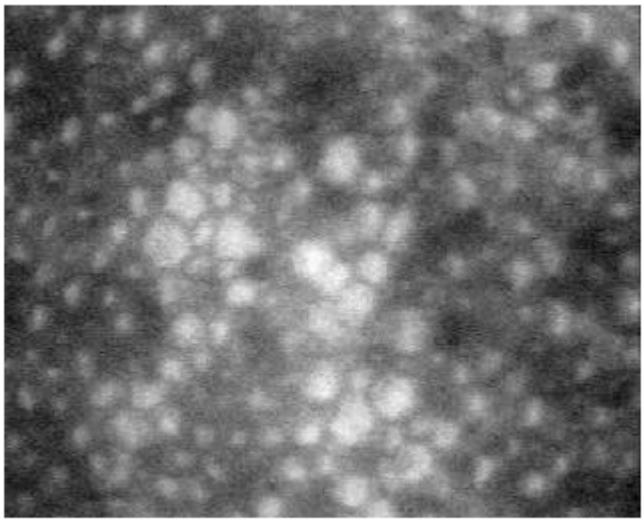
Entrega laboratori 8

Lorenzo Sabater Fandos

Carles Tornel Bonfill

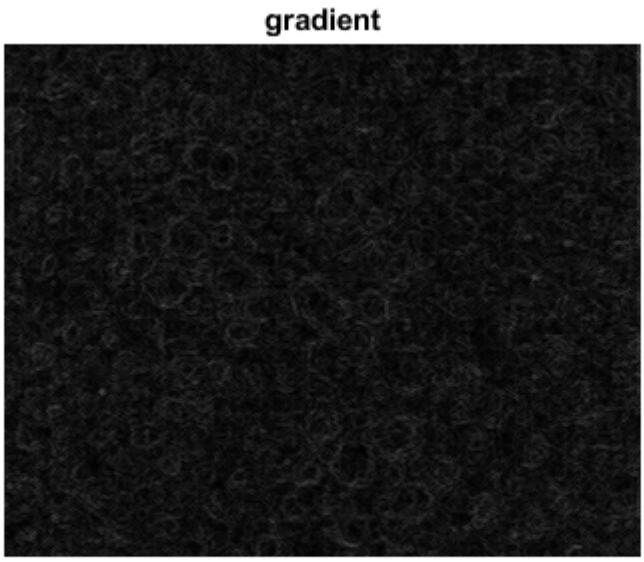
1. Repàs de la segmentació per watershed

```
clear all variables
orig = imread('cornea.tif');
figure, imshow(orig), xlabel('imatge molt sorollosa')
```

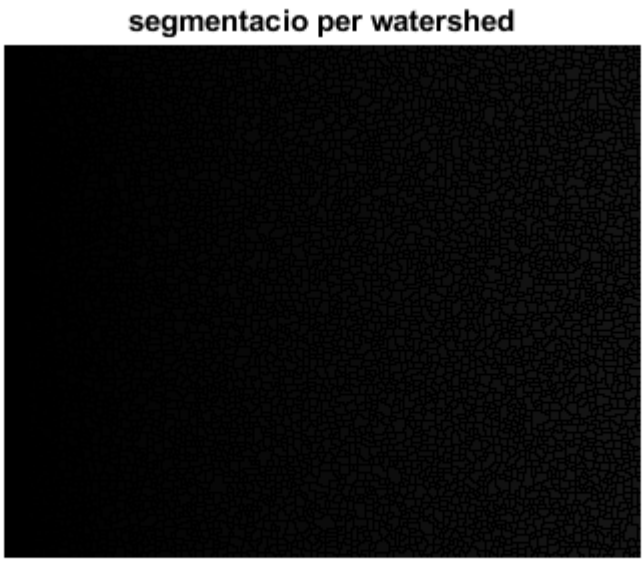


imatge molt sorollosa

```
ee = strel('disk',1);

% calculem el gradient. Ho podem fer morfològic
grad=imsubtract(imdilate(orig,ee),imerode(orig,ee));
figure,imshow(grad),title('gradient')
```

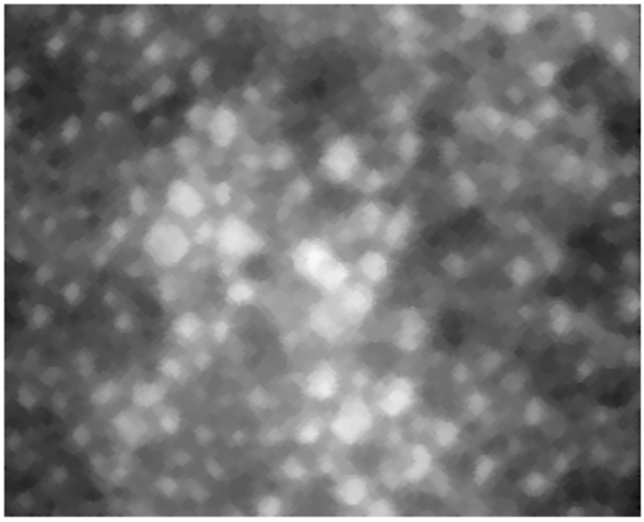



```
% mirem de segmentar les cèlules fent watershed sobre la imatge gradient
segm=watershed(grad);
figure,imshow(segm), title('segmentacio per watershed')
```

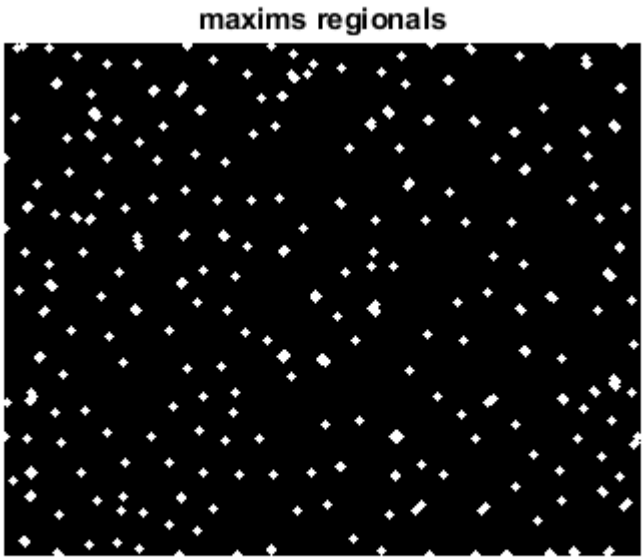



```
% resultat sobresegmentat. Cal treballar amb markers
%% marker-controlled watershed
% usarem els màxims regionals com a markers de les cèlules
% la imatge és molt sorollosa. Cal filtrar abans
ee=strel('disk',2);
```

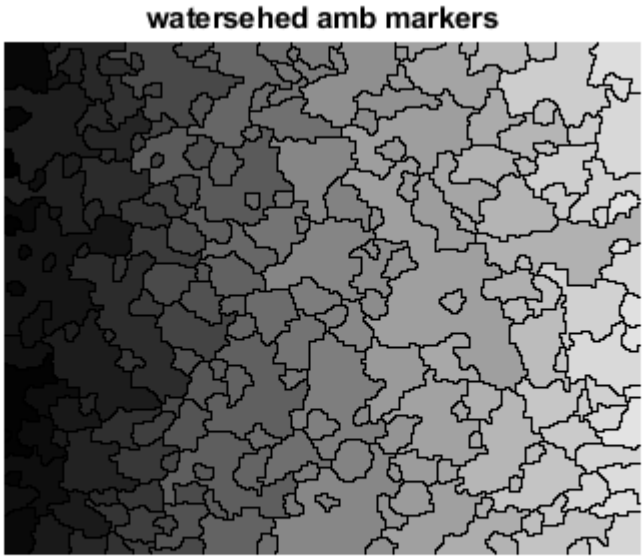
```
filt=imopen(imclose(orig,ee),ee); %filtre OC
figure,imshow(filt)
```



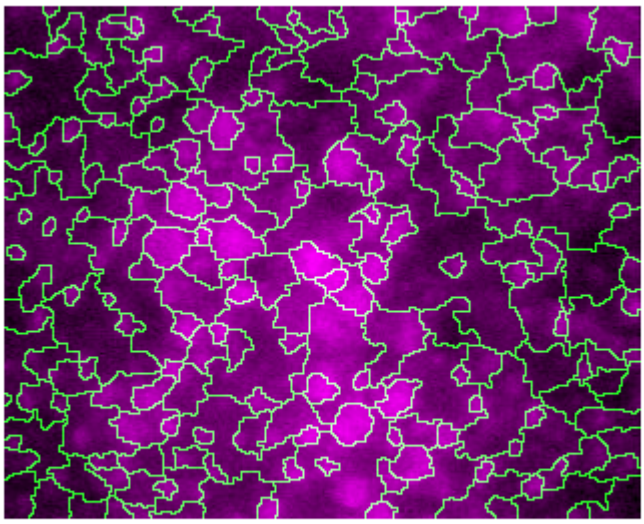
```
rm=imregionalmax(filt);
figure,imshow(rm),title('maxims regionals')
```




```
% Usem els maxims regionals com a markers pel watershed
segm=watershed(imimposemin(grad,rm));
figure,imshow(segm),title('watersehed amb markers')
```

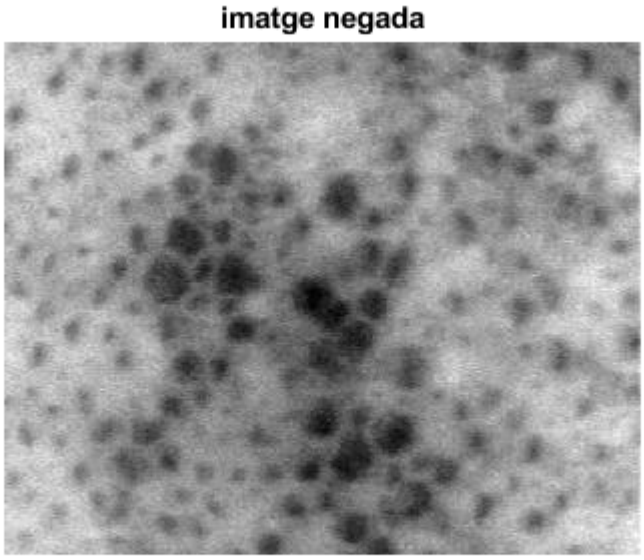



```
%Feu un overlay, en color, dels contorns obtinguts sobre la imatge original
%Us sembla correcta la segmentació? O ens em oblidat alguna cosa ?

s = segm == 0;
imshowpair(s, orig)
```

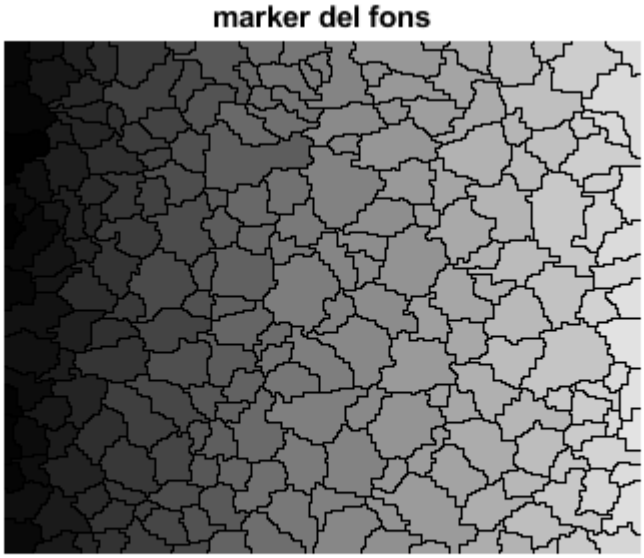


```
%% Cal trobar un marker pel fons
% L'obtenim fent el watershed de la imatge negada.
%Usem com a markers els mateixos maxims d'abans
```

```
Norig=imcomplement(orig);
figure,imshow(Norig),title('imatge negada')
```




```
fons=watershed(imimposemin(Norig,rm));
figure,imshow(fons),title('marker del fons')
```

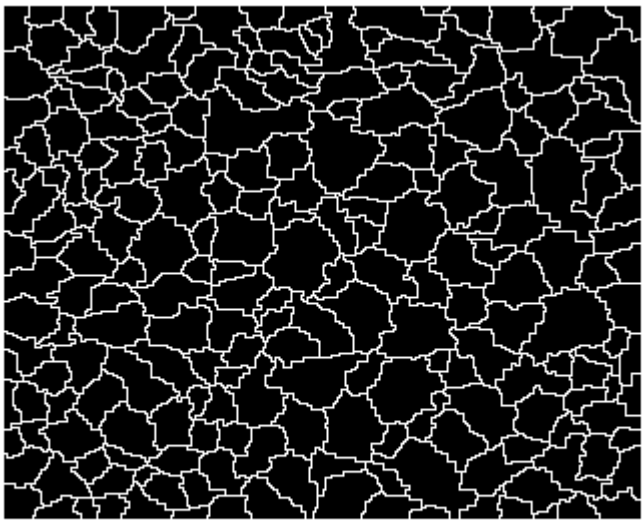



```
% Aquesta part corregeix els errors del document original, podem veure que
% el problema que ens pot sorgir és la superposició de marques, i per tant,
% hem de donar prioritat a les marques de les cèl·lules. Per fer això, hem
% d'eliminar les unions entre aquestes marques i la resta, d'aquesta forma,
```

```
% el watershed les detectará com elements distints.

% Fem una imatge de markers a partir de les celules i del fons
% Eliminem la part de les línees que no ens interessen degut que
% intersecten les dues marques
ss = strel("square", 1)
```

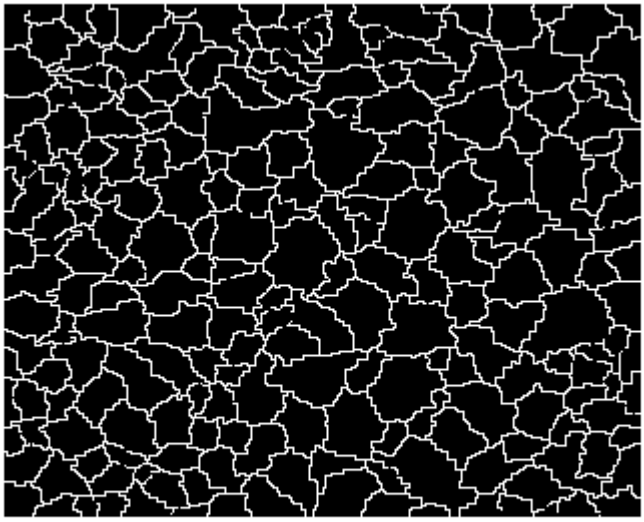
```
ss =
strel is a square shaped structuring element with properties:

      Neighborhood: 1
    Dimensionality: 2
```

```
auxrm= rm;
f = fons==0;
f2 = f;
imshow(f)
```

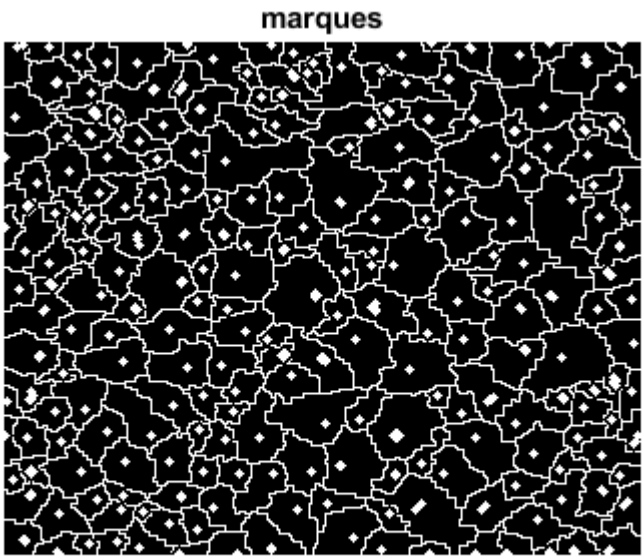


```
% Trobem els punts on es superposen
f = f&imdilate(auxrm, ee);
imshow(f)
```

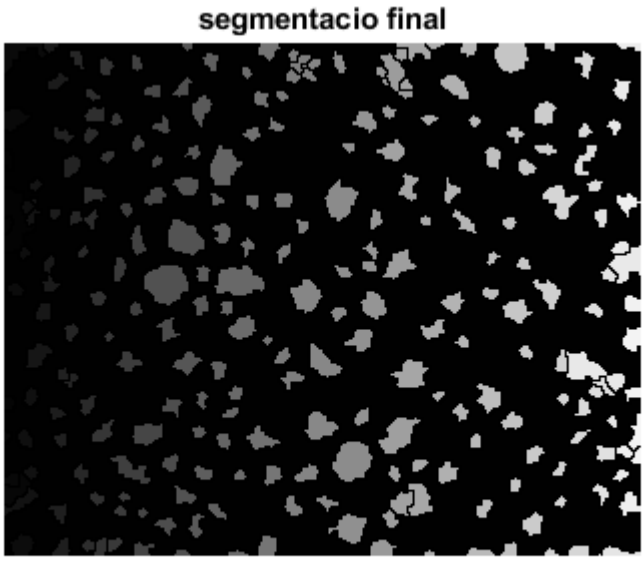
```
% Eliminem les línees que no volem
fons = f2&~f;

imshow(fons)
```



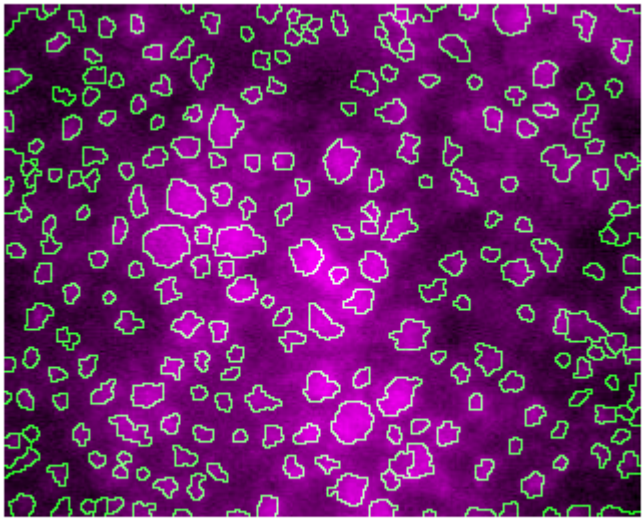
```
markers=fons|rm;
figure,imshow(markers),title('marques')
```




```
% fem el watershed amb les noves marques
segm=watershed(imimposemin(grad,markers));
figure,imshow(segm),title('segmentacio final')
```




```
s = segm == 0;
imshowpair(s, orig)
```

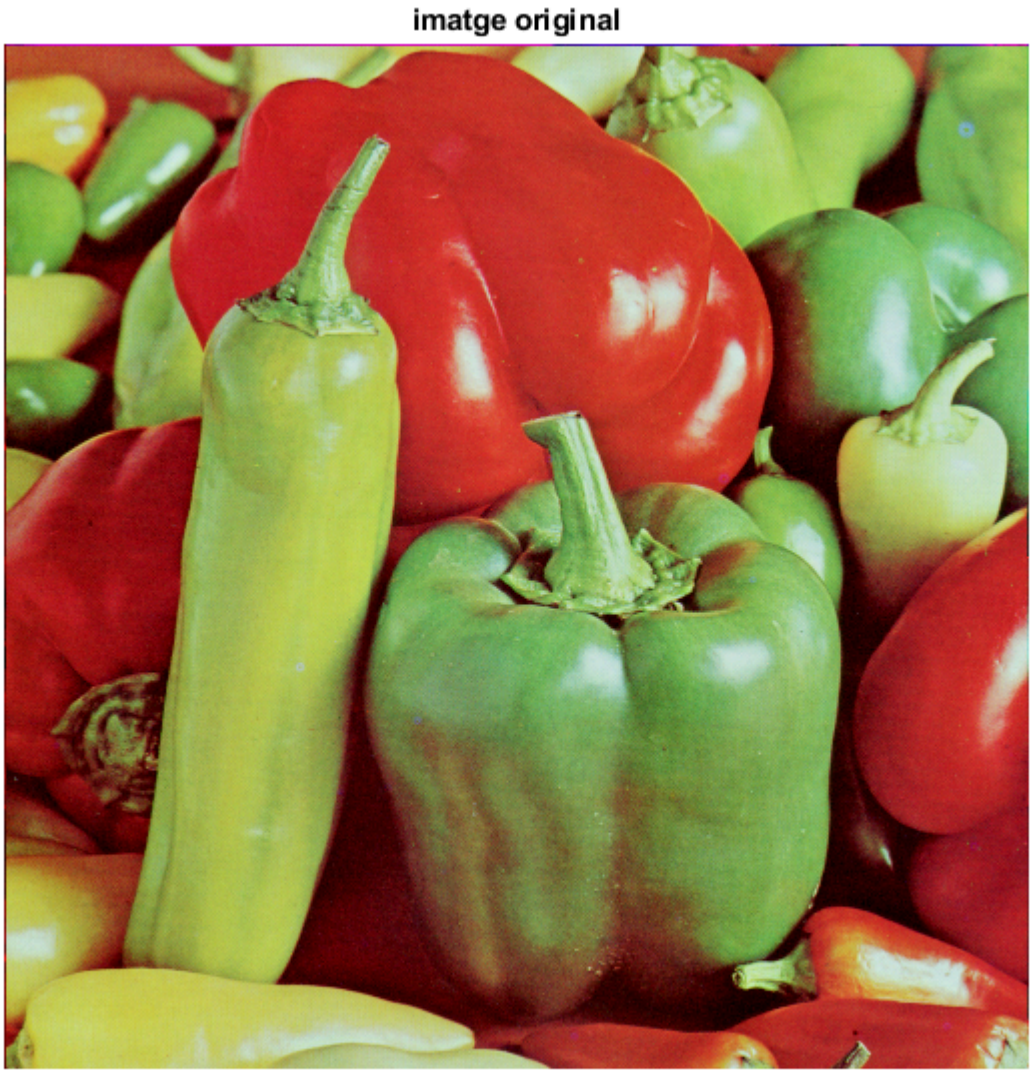


2. Segmentació per clustering. K-means

```
clear all variables
im = imread('peppers.png');
[MAXFILA MAXCOL chan]=size(im)
```

```
MAXFILA =
   512
MAXCOL =
   512
chan =
     3
```

```
figure,imshow(im),title('imatge original')
```




```
% la segmentarem per color. Treballarem en l'espai Hue-Sat
hsv=rgb2hsv(im);
hs=hsv(:,:,1:2);

vect=reshape(hs,MAXFILA*MAXCOL,2); % feature vector. 2 features per píxel
Nclusters=3; %vermell, verd, i negre

% Usarem la funció kmeans amb distancia citiblock
[cluster_idx, cluster_center] = kmeans(vect,Nclusters,'distance','cityblock');

% help("kmeans")
% obtenim la imatge etiquetada pel número cluster
eti=reshape(cluster_idx,MAXFILA,MAXCOL);
figure,imshow(eti,[]),colormap(colorcube), title('imatge etiquetada')
```


```
%Algo no va. Els pebrots vermells queden mal segmentats. (COMTE! EL RESULTAT DE
%KMEANS ÉS SEMPRE DIFERENT. DEPÈN DE L%EXECUCIÓ, EL RESULTAT PODRIA
%SORTIT MILLOR O PITJOR)
```

H conté angles que representen els colors, com el vermell conté colors a [0,30] i [330,360] no detecta que els colors siguin propers.

```
% Representem els píxels en l'espai hue-sat
figure,scatter(vect(:,1),vect(:,2),1,cluster_idx);
xlabel('hue');ylabel('Sat')
title('HS space')
```

```
% El hue és un angle. És cíclic.
```

```
% Píxels vermells amb hue molt similars queden a banda i banda de l’espai hue-sat
% per culpa del pas per zero
```

```
% Ara observem com es comporta amb el k-means que hem implementatat que
% utilitza el hue únicament, ja que no hem cregut necessària la saturació en
% aquest cas, ja que els problemes venen donats per els reflexos i
% l'aritmètica cíclica del hue.

% Per no desaprofitar el rang sencer del hue, normalitzem per fer més
% estables els valor i dsitingir millor els colors
im2 = round(normalize(hsv(:,:,1),'range')*360);

% varem observar que la primera fila y columna tenien valors anòmals,
% posiblement degutda la transformació de RGB a HSV així que varem decidir
% eliminar aquestes franges
[r, c] = k_meansCiclic(im2(2:end-1, 2:end-1),3, 10, 360);
```

```
it =
    10
```

```
c
```

```
c = 1×3
    71   106     5
```

```
imshow(r,[]),colormap(colorcube), title('imatge etiquetada')
```

Els resultats són els esperats, ara si que detectem bé el vermell.

A més, distingim entre els dos verds, tot i que els reflexos entre els vermell i verd distorsionen la silueta completa, com podem veure al preve verd gran. La part blava, correspon al color original. La part verda, es deguda al lleuger reflezxe del preve vermell i el verd propers i, finalment, la part inferor te un gran reflexe vermell que impedeix la correcta detecció de la forma.

La funció utilitzada comenta totes les parts, què fan i per quins motius si és necessari.

```
function [c, cets] = k_meansCiclic(A, clusters, iter, cicle_length)

    [f, c] = size(A);

    % Només pot haver 1 clúster per enter entre [0:360]
    if (cicle_length < clusters)
        error('Too many clusters')
```

```
    end

    % No té sentit fer només un clúster, a més, descartem errors per valors
    % negatius introduits
    if (clusters < 2)
        error('Not enough clusters')
    end

    % Varem observar si deixant un mètode no aleatori funcionava millor,
    % però no es el cas, el fet de que els centroides inicials siguin
    % aleatories es part del que fa que funcioni k-means.

    %for i = [1:clusters]
    %    centroids(1,i) = i*(cicle_length/clusters);
    %end

    % Aquest mètode serveix per a un nombre petit de clústers
    % per evitar clústers repetits
    % Podria tenir problemes en cas de un gran nombre de clústers

    diff = true;
    centroids = zeros(1, clusters);
    it = 0;


    while (diff)
        diff = false;
        rVals = rand(1, clusters);
        for i = [1:clusters]
            centroids(1, i) = round(rVals(1, i)*cicle_length);
            for j = [1:i]
                if(i ~= j)
                    if(centroids(1,i) == centroids(1,j))
                        diff = true;
                    end
                end
            end
        end

    end

    % La convergència en k-means és complicada, degut que o es compara la
    % imatge resultat anterior amb la següent, o s'han de comparar els
    % clústers. El problema es que pot entrar en bucle si roten
    % constantment els cústers, per tant, també posem un límit de
    % iteracions

    results = zeros(f,c);
    conv = false;
    matVal = zeros(f, c, clusters);

    while (~conv && it < iter)

        ant = centroids;
```

```
% Generem les matrius resultat de les distancies
for i = [1:clusters]
    matVal(:,:,i) = (abs(A(:,:) - centroids(1,i)));
end

% Aquí guardarem la suma de distàncies de cada clúster, per
% posteriorment generar els nous centroides si és necessari
calcC = zeros(2, clusters);

% Hem de recorrer la matriu per dur a terme una sèrie de operacions
for i = [1:f]
    for j = [1:c]

        % Ajustem la distància per tenir en compte que és cíclica
        % a continuació elevem la distància al quadrat
        for k = [1:clusters]
            if(matVal(i,j,k) > cicle_length/2)
                matVal(i,j,k) = (cicle_length - matVal(i,j,k))^2;
            else
                matVal(i,j,k) = (matVal(i,j,k))^2;
            end
        end
        max = 1;
        % Cerquem el centroide amb menor distància al punt
        for k = [1:clusters]
            if(matVal(i,j,max) >= matVal(i,j,k))
                max = k;
            end
        end

        % Guaram els valors necessaris per posteriorment recalcular
        % els centroides
        % Per fer que la distància de 0 a 350 sigui la mateixa que
        % de 0 a 10 i es tengui en compte a l'hora de calcular el
        % centroide, s'interpreta com a negatiu de l'anlge
        % complementari

        % De 0 a 10 hi ha le mateix que de 0 a -10, -10
        % representaria 350
        if(A(i,j) > 180)
            calcC(1, max) = calcC(1,max) - (360-A(i,j));
        else
            calcC(1, max) = calcC(1,max) + A(i,j);
        end
        calcC(2, max) = calcC(2,max) + 1;
        results(i, j) = max;
    end
end

% Calculem els que serien els nous centroides, com hem reduit a
% Si obtenim un centroide negatiu, significa que tenim un angle
% major a 180 com a centroide, per tant, reinterpretem l'angle com
% 360 + el centroide negatiu obtingut
```

```
        % Si tenim un centroide -50 realment tenim el centroide 310
        for i = [1:clusters]
               centroids(1,i) = round(calcC(1,i)/calcC(2,i));
               if(centroids(1,i) < 0)
                   centroids(1,i) = 360 + centroids(1,i);
               end
        end

        % Comprovem la convergència
        if (ant == centroids)
            conv = true;
        end

        it = it + 1;
    end
    if (conv)
        cets = centroids;
    else
        cets = ant;
    end
    it
    c = results;
end
```