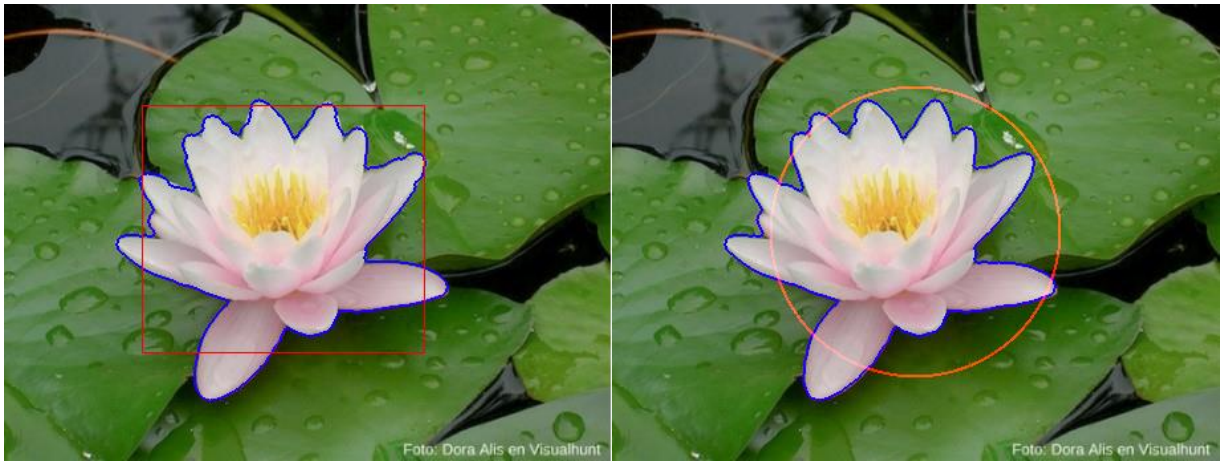
VC: Informe de Laboratori 7

Segmentació mitjançant kmeans



Pere Ginebra Solanellas

12/4/2021 – Q2 Curs 2020-21

Visió per Computador, FIB UPC

1. Introducció

En aquesta sessió treballarem la segmentació mitjançant la funció *kmeans* amb la qual podrem dividir la nostra imatge en diferents conjunts segons diferents paràmetres. Aquests conjunts els podem utilitzar per definir la similitud entre píxels en relació als paràmetres definits i, per tant, distingir objectes amb certes característiques a les del seu entorn / el reste de la imatge. En concret intentarem segmentar un objecte que seleccionarem amb un rectangle.

2. Exercici

2.1. Exercici base

Primer he realitzat la segmentació definint una finestra rectangular que marca l'element usant un *kmeans* dels paràmetres de color de la imatge en format HSV.

Per començar llegeixo la imatge, demano a l'usuari el rectangle defineixo PI, transformo la imatge a format HSV i en separo els canals per facilitar la taula pel *kmeans:*

```
img = imread('flor.jpg');
imshow(img);

rect = getrect;
PI = 3.14159265359;

imgHSV = rgb2hsv(img);
imgH = imgHSV(:,:,1);
imgS = imgHSV(:,:,2);
imgV = imgHSV(:,:,3);
```

Figura 2.1. Importació i preparació inicial de la imatge

A continuació defineixo la taula *O* amb les coordenades circulars del canal H (Hue) i els canals S i V de color de la imatge. Amb aquesta taula faig un *kmeans* de 35 valors, ja que he trobat que és un bon valor per la segmentació de les imatges que he provat. Per visualitzar el resultat del *kmeans* podem representar els diferents conjunts amb colors en una imatge rgb amb les funcions *reshape* i *label2rgb*:

```
O(:,1) = cos(imgH(:)*2*PI);    %Hx
O(:,2) = sin(imgH(:)*2*PI);    %Hy
O(:,3) = imgS(:);
O(:,4) = imgV(:);

k = 35;
C = kmeans(O,k);

[n,m] = size(imgH);
IC = reshape(C,n,m); %transformem la llista C en una matriu
rgb = label2rgb(IC); %donem color als 'labels' de C
figure
imshow(rgb);
```

Figura 2.2. Declaració de la taula usada al *kmeans* i la representació del seu resultat

Obtinguda la llista del *kmeans*, genero una mascara *MASK* per la imatge on tots els píxels que queden a dins del rectangle estan a 1 i la poso en columna al costat de C per formar la taula H. Aquesta taula ens servirà al bucle for per determinar quants píxels de cada conjunt estan a dins o fora del rectangle (si tenen un 1 o un 0 al valor de màscara que l'acompanya). Aquests valors es guarden Hist1 i Hist0 respectivament:

```
MASK = zeros(n,m);
MASK(rect(2):(rect(2)+rect(4)), rect(1):(rect(1)+rect(3))) = 1;

H = [C, MASK(:)];

Hist0 = zeros(1,k);
Hist1 = zeros(1,k);

for i = 1:(n*m)
    if H(i,2) == 1
        Hist1(H(i,1)) = Hist1(H(i,1)) + 1;
    else
        Hist0(H(i,1)) = Hist0(H(i,1)) + 1;
    end
end
```

Figura 2.3. Càlcul del nombre de píxels a dins i fora del rectangle per cada conjunt a les variables *Hist*

Finalment marquem a *RES* els conjunts que tenen més píxels a dins del rectangle que a fora i a *M* els píxels que pertanyen als conjunts de *RES*. Per veure els resultats transformem la llista *M* en una matriu *IM* amb *reshape* i l'imprimim per pantalla, per eliminar petits elements erronis i acabar de agafar bé l'objecte podem aplicar diversos filtres com *medfilt, imopen* i *imclose*:

```
RES(:) = Hist1(:) > Hist0(:);

M(:) = RES(C(:));
IM = reshape(M,n,m);
figure
imshow(IM);

SE1 = strel('sphere', 3);
SE2 = strel('sphere', 5);
IM = medfilt2(IM, [2 2]);
IM = imopen(IM, SE1);
IM = imclose(IM, SE1);
IM = imopen(IM, SE2);
figure
imshow(IM);
```

Figura 2.4. Selecció dels píxels que pertanyen a l'objecte i filtrat de la imatge resultant

Per acabar de mostrar el resultat, calculem el contorn amb una dilatació i una resta i el pintem a la imatge. També podem mostrar el rectangle seleccionat sobre la imatge resultant amb *hold on* i la funció *rectangle*:

```
SE3 = strel('sphere', 2);
IMcontorn = imdilate(IM, SE3);
IMcontorn = IMcontorn - IM;
figure
imshow(IMcontorn);

imgSol(:,:,3) = max(uint8(IMcontorn(:,:)*255), img(:,:,3));
imgSol(:,:,2) = img(:,:,2)-(img(:,:,2).*uint8(IMcontorn(:,:)));
imgSol(:,:,1) = img(:,:,1)-(img(:,:,3).*uint8(IMcontorn(:,:)));

imshow(imgSol);
hold on
rectangle('Position', rect, 'EdgeColor', 'r');
```

Figura 2.5. Pintat del contorn de l'objecte i del rectangle seleccionat

2.2. Exercici usant distància en el *kmeans*

A continuació he modificat lleugerament l'anterior script per tal de afegir la distància de cada píxel al rectangle a la funció *kmeans* i per tenir en compte la quantitat de píxels del color a dins/fora de la finestra relativament a la mida de cada zona.

Per això afegeixo una columna a *O* per tenir en compte la distància de cada píxel al rectangle a l'hora de fer els conjunts del *kmeans*:

```
%calcular la distància de cada pixel al rectangle
distToRect = zeros(n,m);
for i = 1:n
    for j = 1:m
        distToRect(i,j) = distFromRectangle2(i,j,rect);
    end
end
boxImportance = 1.5; %importancia de la distancia del rectangle
O(:,5) = distToRect(:)/(norm([0,0]-[n,m])/boxImportance);
```

Figura 2.6. Càlcul de la distància de cada píxel a l'interior de la zona seleccionada

Per calcular la distància al rectangle he escrit les funcions *distFromRectangle* i *distFromRectangle2*, que calculen la distància a l'exterior i al centre del rectangle respectivament. Les funcions es poden trobar a l'apartat d'Annexos / Scripts.

A més, modifico la línia que marca la llista *RES* per tenir en compte la quantitat de píxels a dins/fora del rectangle en relació a la mida de cada zona:

```
%les divisions per fer el nombre relatiu a la mida del rectangle
RES(:) = Hist1(:)/(rect(3)*rect(4)) > Hist0(:)/((m*n)-(rect(3)*rect(4)));
```

Figura 2.7. Selecció dels píxels de l'objecte calculat relativament a la dimensió de la zona seleccionada

2.3. Exercici sense selecció rectangular

Finalment he modificat el script anterior per realitzar la segmentació sense especificar l'element, interpretant que aquest es troba més o menys centrat en la imatge.

Per començar calculem la distància de cada punt al centre i ho guardem a la matriu *distToCenter*, que com abans utilitzem com a columna a *O* pel càlcul del *kmeans*:

```
distToCenter = zeros(n,m);
for i = 1:n
    for j = 1:m
        distToCenter(i,j) = norm([i,j]-[n/2,m/2]);
    end
end
boxImportance = 1; %importancia de la distancia del rectangle
O(:,5) = distToCenter(:)/(norm([0,0]-[n,m])/boxImportance);
```

Figura 2.8. Càlcul de la distància de cada píxel al centre de la imatge

A l'hora de calcular els píxels de l'element utilitzem la mateixa matriu *distToCenter* per la taula *H*, i definim un llindar per definir un cercle de selecció com fèiem amb el rectangle anterior:

```
H = [C, distToCenter(:)];
distanceToFocus = 0.33; %how many half diagonals do you want the circle's diagonal to be
diagonal = sqrt(n^2+m^2)/2; %half diagonal (distance from centre to a corner)

for i = 1:(n*m)
    if H(i,2) < distanceToFocus*diagonal
        Hist1(H(i,1)) = Hist1(H(i,1)) + 1;
    else
        Hist0(H(i,1)) = Hist0(H(i,1)) + 1;
    end
end
```

Figura 2.9. Càlcul del nombre de píxels de cada conjunt que es troba a dins d'un cercle

Alternativament, en comptes de sumar 1 per cada píxel a *Hist,* també he provat donant un pes proporcional a cada píxel segons la seva distància al centre:

```
for i = 1:(n*m)
    if H(i,2) < distanceToFocus*diagonal
        Hist1(H(i,1)) = Hist1(H(i,1)) + (distanceToFocus*diagonal - H(i,2)) + 0.2;
    else
        Hist0(H(i,1)) = Hist0(H(i,1)) + (H(i,2) - distanceToFocus*diagonal) + 0.2;
    end
end
```

Figura 2.10. Càlcul de la presencia d'un conjunt a dins i fora del cercle ponderada segons la distància de cada píxel d'aquest al centre

Finalment, després de pintar el contorn, afegeixo el cercle que s'ha utilitzat per la selecció a la imatge resultant:

```
%pinta el cercle que s'utilitza com a finestra
imgSol(:,:,1) = max(uint8(distanceToFocus*diagonal > distToCenter(:,:)-1 & distToCenter(:,:)+1 > distanceToFocus*diagonal)*255, imgSol(:,:,1));
imshow(imgSol);
```

Figura 2.11. Pintat del cercle usat en la segmentació sobre la imatge resultant

3. Resultats

A continuació tenim els resultats intermitjos del primer script, hi podem veure la imatge de conjunts en color *IC*, la imatge obtinguda amb la segmentació *IM,* la mateixa després d'un filtrat, i finalment el contorn de l'anterior imatge pintat en blau per sobre de la original:

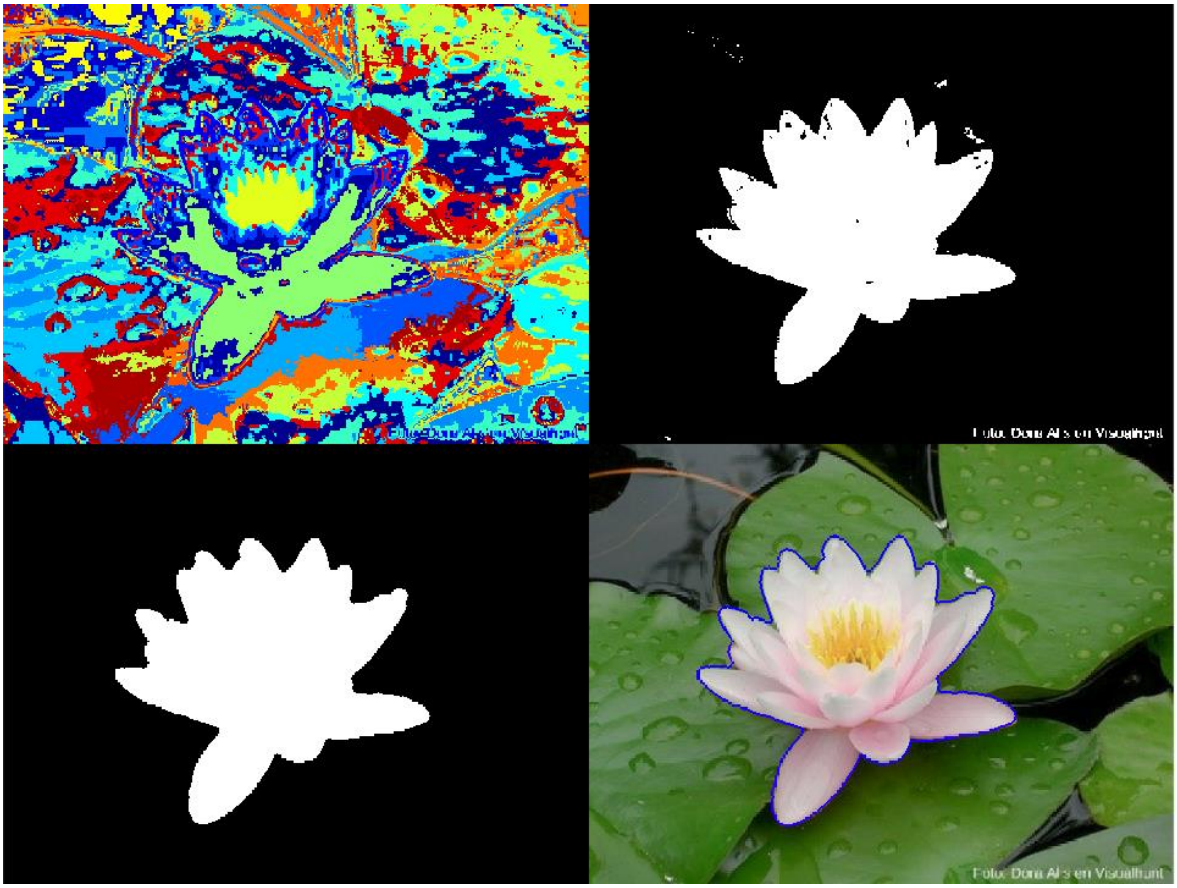


Figura 3.1. Resultats intermitjos del primer script

El primer resultat final és el següent:

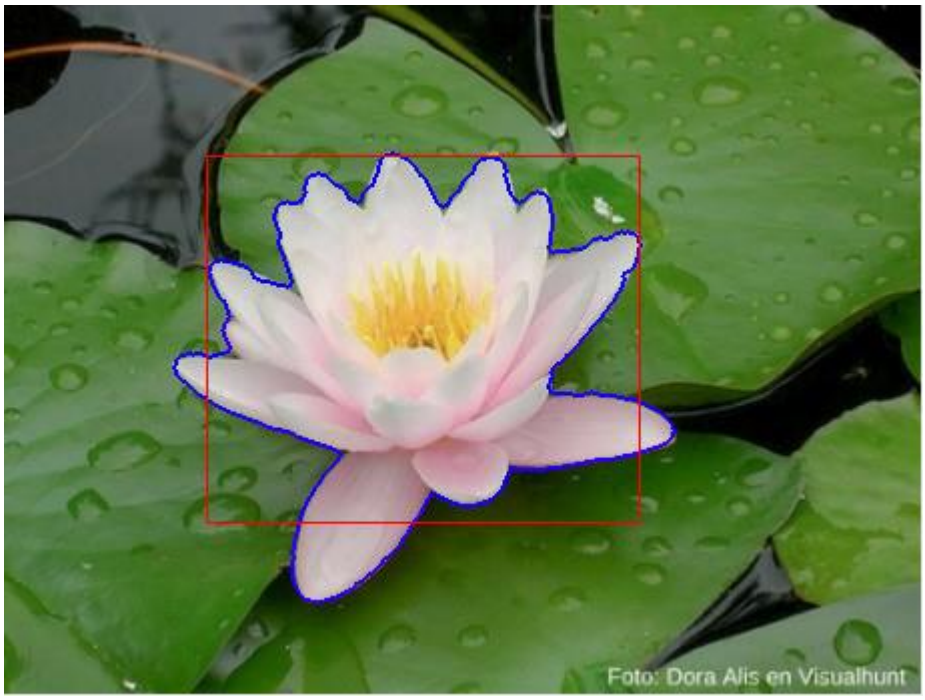


Figura 3.2. Resultat final del primer script

El primer script funciona molt bé per a elements amb colors poc freqüents a la resta de la imatge, però per a objectes que tenen colors comuns al fons el segon script obté millors resultats aproximats, ja que afegeix a l'objecte colors que el primer no agafaria. D'altra banda, el segon sol agafar més píxels que no pertanyen a l'objecte com a conseqüència:

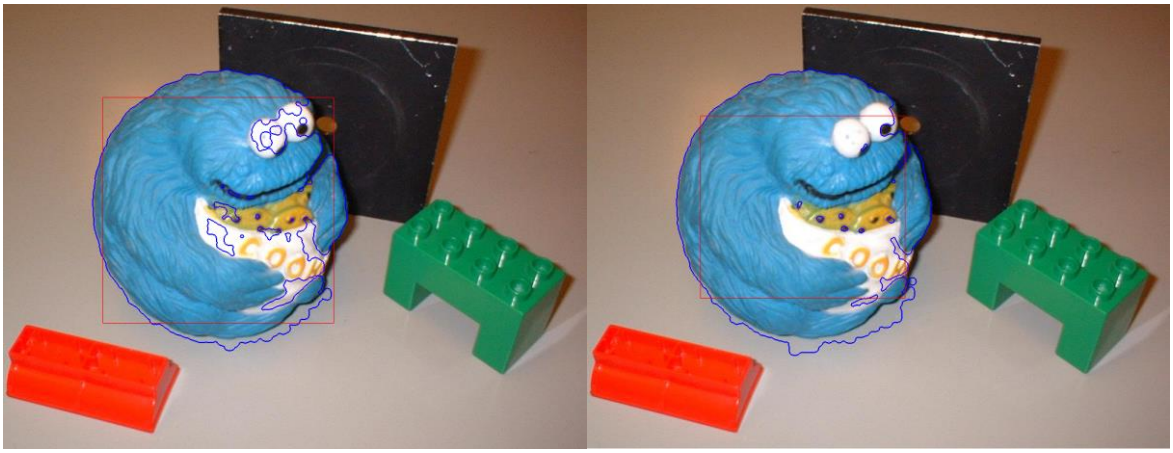


Figura 3.3. Comparació dels resultats dels dos primers scripts. A l'esquerra el primer, a la dreta el segon

Els resultats de les dos versions del script que seleccionen l'element central de la imatge solen donar resultats prou similars en les imatges provades:

Figura 3.4. Comparació dels dos scripts de segmentació de l'element central

La imatge de l'esquerra es el script base que segmenta l'element central, la de la dreta utilitza la distància de cada píxel per calcular *Hist0* i *Hist1*. He trobat que el millor radi central per aquesta imatge està al voltant de 0,3 – 0,4 vegades el radi de la imatge, i que a la segona versió d'aquest script li va millor un radi lleugerament més gran que la primera. Les imatges s'han fet amb un radi de 0,33 i 0,38 respectivament.

Cal dir que per utilitzar aquestes dos ultimes versions cal analitzar una imatge en la que l'objecte està ben centrat, sinó no funciona gaire bé ja que el cercle agafa parts d'elements no desitjats. Dit això, jugant amb la variable *boxImportance* podem refinar lleugerament els resultats en alguns casos:

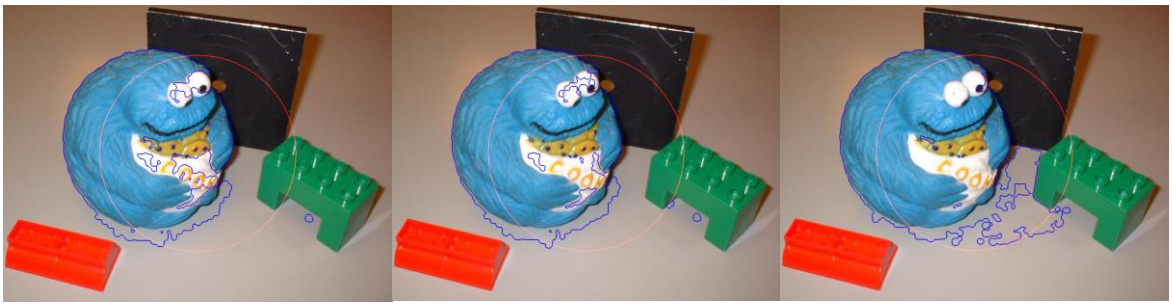


Figura 3.5. Comparació dels resultats variant *boxImportance* a l'últim script amb un objecte descentrat

En les imatges anteriors podem veure les diferències entre utilitzar valors de *boxImportance* de 0, 0.5 i 1 respectivament. Aquesta variable, com s'ha explicat a l'apartat de l'exercici, defineix la importància de la distància de cada píxel a l'interior cercle.

4. Annexos / Scripts

```
clear
close all

img = imread('flor.jpg');
imshow(img);

rect = getrect;
PI = 3.14159265359;

imgHSV = rgb2hsv(img);
imgH = imgHSV(:,:,1);
imgS = imgHSV(:,:,2);
imgV = imgHSV(:,:,3);

O(:,1) = cos(imgH(:)*2*PI);   %Hx
O(:,2) = sin(imgH(:)*2*PI);   %Hy
O(:,3) = imgS(:);
O(:,4) = imgV(:);

k = 35;
C = kmeans(O,k);

[n,m] = size(imgH);
IC = reshape(C,n,m); %transformem la llista C en una matriu
rgb = label2rgb(IC); %donem color als 'labels' de C
figure
imshow(rgb);

MASK = zeros(n,m);
MASK(rect(2):(rect(2)+rect(4)), rect(1):(rect(1)+rect(3))) = 1;

H = [C, MASK(:)];

Hist0 = zeros(1,k);
Hist1 = zeros(1,k);

for i = 1:(n*m)
    if H(i,2) == 1
        Hist1(H(i,1)) = Hist1(H(i,1)) + 1;
    else
        Hist0(H(i,1)) = Hist0(H(i,1)) + 1;
    end
end

RES(:) = Hist1(:) > Hist0(:);

M(:) = RES(C(:));
IM = reshape(M,n,m);
figure
imshow(IM);

SE1 = strel('sphere', 3);
SE2 = strel('sphere', 5);
IM = medfilt2(IM, [2 2]);
IM = imopen(IM, SE1);
IM = imclose(IM, SE1);
IM = imopen(IM, SE2);
figure
imshow(IM);

SE3 = strel('sphere', 2);
IMcontorn = imdilate(IM, SE3);
IMcontorn = IMcontorn - IM;
figure
imshow(IMcontorn);

imgSol(:,:,3) = max(uint8(IMcontorn(:,:)*255), img(:,:,3));
imgSol(:,:,2) = img(:,:,2)-(img(:,:,2).*uint8(IMcontorn(:,:)));
imgSol(:,:,1) = img(:,:,1)-(img(:,:,3).*uint8(IMcontorn(:,:)));

imshow(imgSol);
hold on
rectangle('Position', rect, 'EdgeColor', 'r');
```

Figura 4.1. Script inicial

```
clear
close all

img = imread('flor.jpg');
imshow(img);

rect = getrect;
PI = 3.14159265359;

imgHSV = rgb2hsv(img);
imgH = imgHSV(:,:,1);
imgS = imgHSV(:,:,2);
imgV = imgHSV(:,:,3);
[n,m] = size(imgH);

O(:,1) = cos(imgH(:)*2*PI);   %Hx
O(:,2) = sin(imgH(:)*2*PI);   %Hy
O(:,3) = imgS(:);
O(:,4) = imgV(:);
%calcular la distància de cada pixel al rectangle
distToRect = zeros(n,m);
for i = 1:n
    for j = 1:m
        distToRect(i,j) = distFromRectangle2(i,j,rect);
    end
end
boxImportance = 1.5; %importancia de la distancia del rectangle
O(:,5) = distToRect(:)/(norm([0,0]-[n,m])/boxImportance);

k = 35;
C = kmeans(O,k);

IC = reshape(C,n,m);
rgb = label2rgb(IC);
figure
imshow(rgb);

MASK = zeros(n,m);
MASK(rect(2):(rect(2)+rect(4)), rect(1):(rect(1)+rect(3))) = 1;

H = [C, MASK(:)];
Hist0 = zeros(1,k);
Hist1 = zeros(1,k);
for i = 1:(n*m)
    if H(i,2) == 1
        Hist1(H(i,1)) = Hist1(H(i,1)) + 1;
    else
        Hist0(H(i,1)) = Hist0(H(i,1)) + 1;
    end
end

%les divisions per fer el nombre relatiu a la mida del rectangle
RES(:) = Hist1(:)/(rect(3)*rect(4)) > Hist0(:)/((m*n)-(rect(3)*rect(4)));

M(:) = RES(C(:));
IM = reshape(M,n,m);
figure
imshow(IM);

SE1 = strel('sphere', 3);
SE2 = strel('sphere', 5);
IM = medfilt2(IM, [2 2]);
IM = imopen(IM, SE1);
IM = imclose(IM, SE1);
IM = imopen(IM, SE2);
figure
imshow(IM);

SE3 = strel('sphere', 2);
IMcontorn = imdilate(IM, SE3);
IMcontorn = IMcontorn - IM;
figure
imshow(IMcontorn);

imgSol(:,:,3) = max(uint8(IMcontorn(:,:)*255), img(:,:,3));
imgSol(:,:,2) = img(:,:,2)-(img(:,:,2).*uint8(IMcontorn(:,:)));
imgSol(:,:,1) = img(:,:,1)-(img(:,:,3).*uint8(IMcontorn(:,:)));
imshow(imgSol);
hold on
rectangle('Position', rect, 'EdgeColor', 'r');
```

Figura 4.2. Segon Script (utilitza distància al rectangle)

```
clear
close all

img = imread('flor.jpg');
imshow(img);

PI = 3.14159265359;

imgHSV = rgb2hsv(img);
imgH = imgHSV(:,:,1);
imgS = imgHSV(:,:,2);
imgV = imgHSV(:,:,3);

[n,m] = size(imgH);

O(:,1) = cos(imgH(:)*2*PI);   %Hx
O(:,2) = sin(imgH(:)*2*PI);   %Hy
O(:,3) = imgS(:);
O(:,4) = imgV(:);
distToCenter = zeros(n,m);
for i = 1:n
    for j = 1:m
        distToCenter(i,j) = norm([i,j]-[n/2,m/2]);
    end
end
boxImportance = 1; %importancia de la distancia del rectangle
O(:,5) = distToCenter(:)/(norm([0,0]-[n,m])/boxImportance);

k = 35;
C = kmeans(O,k);

IC = reshape(C,n,m);
rgb = label2rgb(IC);
figure
imshow(rgb);

Hist0 = zeros(1,k);
Hist1 = zeros(1,k);

H = [C, distToCenter(:)];
distanceToFocus = 0.33; %how many half diagonals do you want the circle's diagonal to be
diagonal = sqrt(n^2+m^2)/2; %half diagonal (distance from centre to a corner)

for i = 1:(n*m)
    if H(i,2) < distanceToFocus*diagonal
        Hist1(H(i,1)) = Hist1(H(i,1)) + 1;
    else
        Hist0(H(i,1)) = Hist0(H(i,1)) + 1;
    end
end

RES(:) = Hist1(:) > Hist0(:);

M(:) = RES(C(:));
IM = reshape(M,n,m);
figure
imshow(IM);

SE1 = strel('sphere', 3);
SE2 = strel('sphere', 5);
IM = medfilt2(IM, [2 2]);
IM = imopen(IM, SE1);
IM = imclose(IM, SE1);
IM = imopen(IM, SE2);
figure
imshow(IM);

%pinta el contorn de l'objecte
SE3 = strel('sphere', 2);
IMcontorn = imdilate(IM, SE3);
IMcontorn = IMcontorn - IM;
figure
imshow(IMcontorn);

imgSol(:,:,3) = max(uint8(IMcontorn(:,:)*255), img(:,:,3));
imgSol(:,:,2) = img(:,:,2)-(img(:,:,2).*uint8(IMcontorn(:,:)));
imgSol(:,:,1) = img(:,:,1)-(img(:,:,3).*uint8(IMcontorn(:,:)));

%pinta el cercle que s'utilitza com a finestra
imgSol(:,:,1) = max(uint8(distanceToFocus*diagonal > distToCenter(:,:)-1 & distToCenter(:,:)+1 >
distanceToFocus*diagonal)*255, imgSol(:,:,1));
imshow(imgSol);
```

Figura 4.3. Tercer Script (sense rectangle, amb l'element centrat)

```
clear
close all

img = imread('flor.jpg');
imshow(img);

PI = 3.14159265359;

imgHSV = rgb2hsv(img);
imgH = imgHSV(:,:,1);
imgS = imgHSV(:,:,2);
imgV = imgHSV(:,:,3);

[n,m] = size(imgH);

O(:,1) = cos(imgH(:)*2*PI);   %Hx
O(:,2) = sin(imgH(:)*2*PI);   %Hy
O(:,3) = imgS(:);
O(:,4) = imgV(:);
distToCenter = zeros(n,m);
for i = 1:n
    for j = 1:m
        distToCenter(i,j) = norm([i,j]-[n/2,m/2]);
    end
end
boxImportance = 1; %importancia de la distancia del rectangle
O(:,5) = distToCenter(:)/(norm([0,0]-[n,m])/boxImportance);

k = 35;
C = kmeans(O,k);

IC = reshape(C,n,m);
rgb = label2rgb(IC);
figure
imshow(rgb);

Hist0 = zeros(1,k);
Hist1 = zeros(1,k);

H = [C, distToCenter(:)];
distanceToFocus = 0.33; %how many half diagonals do you want the circle's diagonal to be
diagonal = sqrt(n^2+m^2)/2; %half diagonal (distance from centre to a corner)

for i = 1:(n*m)
    if H(i,2) < distanceToFocus*diagonal
        Hist1(H(i,1)) = Hist1(H(i,1)) + (distanceToFocus*diagonal - H(i,2)) + 0.2;
    else
        Hist0(H(i,1)) = Hist0(H(i,1)) + (H(i,2) - distanceToFocus*diagonal) + 0.2;
    end
end

RES(:) = Hist1(:) > Hist0(:);

M(:) = RES(C(:));
IM = reshape(M,n,m);
figure
imshow(IM);

SE1 = strel('sphere', 3);
SE2 = strel('sphere', 5);
IM = medfilt2(IM, [2 2]);
IM = imopen(IM, SE1);
IM = imclose(IM, SE1);
IM = imopen(IM, SE2);
figure
imshow(IM);

%pinta el contorn de l'objecte
SE3 = strel('sphere', 2);
IMcontorn = imdilate(IM, SE3);
IMcontorn = IMcontorn - IM;
figure
imshow(IMcontorn);

imgSol(:,:,3) = max(uint8(IMcontorn(:,:)*255), img(:,:,3));
imgSol(:,:,2) = img(:,:,2)-(img(:,:,2).*uint8(IMcontorn(:,:)));
imgSol(:,:,1) = img(:,:,1)-(img(:,:,3).*uint8(IMcontorn(:,:)));

%pinta el cercle que s'utilitza com a finestra
imgSol(:,:,1) = max(uint8(distanceToFocus*diagonal > distToCenter(:,:)-1 & distToCenter(:,:)+1 >
distanceToFocus*diagonal)*255, imgSol(:,:,1));
imshow(imgSol);
```

Figura 4.4. Quart Script (calcula *Hist* tenint en compte la distància de cada píxel al centre)

```
function [dist] = distFromRectangle(x,y,rect)
    rx = rect(1);
    ry = rect(2);
    rxx = rect(1)+rect(3);
    ryy = rect(2)+rect(4);

    if x > rx & x < rxx
        if y > rect(2) && y < ryy
            dist = 0;
        else
            dist = min(abs(y-rect(2)), abs(y-ryy));
        end
    elseif y > ry &  y < ryy
            dist = min(abs(x-rect(1)), abs(x-rxx));
    else
        dist = min([norm([x,y]-[rx,ry]), norm([x,y]-[rx,ryy]), norm([x,y]-
[rxx,ry]), norm([x,y]-[rxx,ryy])]);
    end
end
```

Figura 4.5. Funció ***distFromRectangle***, calcula la distància d'un punt al contorn d'un rectangle

```
function [dist] = distFromRectangle2(x,y,rect)
    if x > rect(1) & x < (rect(1)+rect(3)) & y > rect(2) & y < (rect(2)+rect(4))
        dist = 0;
    else
        rectCenter = [rect(1)+(rect(3)/2),rect(2)+(rect(4)/2)];
        dist = norm([x,y]-rectCenter);
    end
end
```

Figura 4.6. Funció ***distFromRectangle2***, calcula la distància d'un punt al centre d'un rectangle i 0 si el punt es del rectangle

Aquestes dos funcions s'utilitzen per calcular la distància de cada píxel a la finestra de segmentació, que s'aplica a la segona versió del script per afegir-la al càlcul del *kmeans*.