Práctica 2: K-Means

Mario Fernández Simón

- Obriu una imatge i comprovar que és una imatge RGB, fent servir la instrucció *size.*
- Mostrar la imatge per pantalla. (fig. 3)

```
I = imread("nenufar.jpg");
[files, cols, dim] = size(I);

if dim ~= 3
    error('No es una imagen a color');
else
    imshow(I);
end
```

- Obtenir del usuari el rectangle que emmarca l'objecte (fig. 4) que vol segmentar. Utilitzeu *rect = getrect;.* Vigileu que la variable *rect* té el format (x, y, w, h); posició, amplada alçada (en lloc de fila i columna).

```
rect = getrect;
```

- La gama de colors en RGB és enorme caldrà reduir el nombre de colors. Us podeu ajudar de l'algoritme k-means per reduir la quantitat de colors de la imatge i facilitar la classificació.
- Podeu reduir el nombre de colors en RGB o millor en HSV. Si ho fem per pel hue podem obtenir una imatge **HSV**però caldrà vigilar amb el problema de l'aritmètica cíclica dels angles

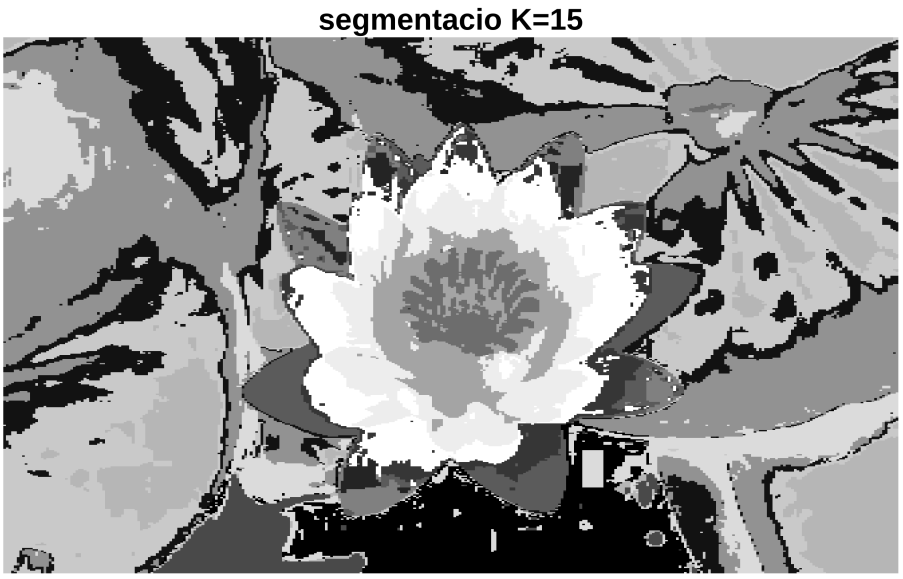
```
hsv = rgb2hsv(I);
vect=reshape(hsv, files*cols, 3);  % [H, S, V]

Hx = sin(2*pi*vect(:,1));
Hy = cos(2*pi*vect(:,1));

O = [Hx, Hy, vect(:, 2:3)];
```

- Agrupem els colors en **k** classes amb kmeans i obtenim la classificació **C**, per exemple

k = 15; **C** = kmeans(**O**,k);

- El valor més adequat del paràmetre *k* dependrà de la complexitat de la imatge, però normalmente es mou entre 10 a 20 colors.
- Ara caldrà decidir quins colors de C formen part de l'objecte i quins no. Podem comptar quins colors cauen majoritàriament dins de la finestra. No busquem un 100% d'efectivitat, en que obtingueu quelcom semblant a la figura 5 serà suficient.

```
k = 15;
C = kmeans(O, k);

eti = reshape(C, files, cols);
figure, imshow(eti, []), title('segmentacio K=15')
```



- Per tal fi, construïm una imatge de valors booleans anomenada **MASK**de la mateixa mida que la imatge inicial amb els valors a zero si està fora del rectangle i 1 si està dins (fig 5). Nota: és molt fàcil construir-la amb una sola sentència, no us compliqueu a fer funcions de si un píxel està dins o fora d'un rectangle...
- Ara construïm un vector **H** que indicarà si un color cau dins del rectangle o no. Tal que H = [C, MASK(:)]; Observeu que H és una taula que indica si tal color C és dins o fora de la mascara.

```
% Coordenadas de imagen recortada
x1 = round(rect(1, 1));
x2 = round(rect(1, 1) + rect(1, 3));
```

```
y1 = round(rect(1, 2));
y2 = round(rect(1, 2) + rect(1, 4));

% Usaremos 2 sentencias para mas claridad
MASK = false(files, cols);
MASK(y1:y2, x1:x2) = true;
```

- Ara construïm un vector **H** que indicarà si un color cau dins del rectangle o no. Tal que H = [C, MASK(:)]; Observeu que H és una taula que indica si tal color C és dins o fora de la mascara.
- A continuació, comptem per a cada color de C quants píxels han caigut a fora i quants a dins. Guardem els resultats en dos arrays **Hist0** i **Hist1**. Evidentment, la mida de Hist0 i Hist1 ha de ser igual al nombre de classes *k*.

```
H = [C, MASK(:)];

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

for i=1:size(H, 1)
    color = H(i, 1);
    if H(i, 2) == 0
        Hist0(color, 1) = Hist0(color, 1) + 1;
    else
        Hist1(color, 1) = Hist1(color, 1) + 1;
    end
end
```

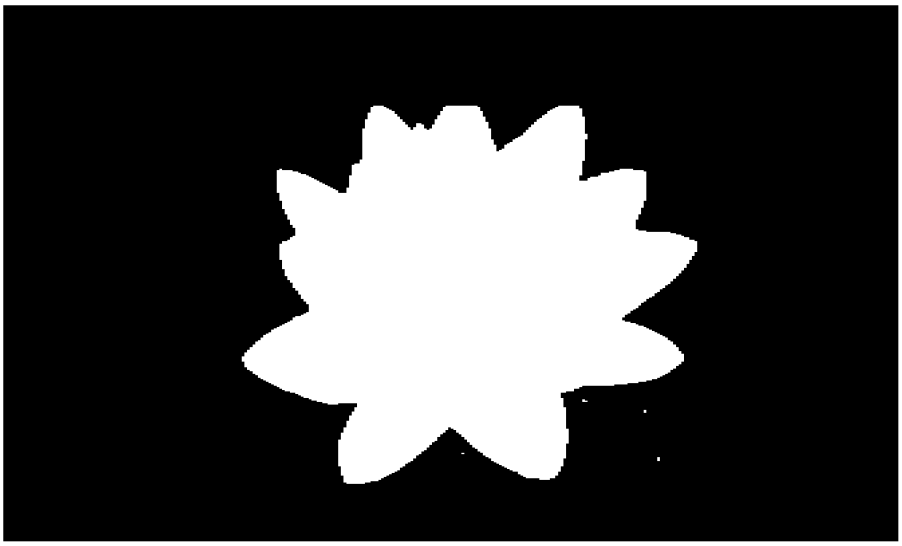
- Llavors decidim si un representant de color pertany a la figura que es vol segmentar comparant les seves aparicions dins i fora del rectangle. Guardem la decisió en un vector anomenat ***RES***. Quelcom com RES = Hist1 > Hist0;

```
RES = Hist1 > Hist0;
```

- Casi per últim, decidim per a cada píxel de la matriu **H** si forma part de la figura o no utilitzant la informació que conté el vector **RES**. Guardem el resultat en un vector **M** i el mostrem per pantalla
- I finalment filtrem el resultat i/o mostrem l'objecte segmentat de mida més gran.

```
M = H;

for i=1:size(H, 1)
    entra = RES(H(i, 1));
    M(i, 2) = entra && H(i, 2);
end
```

```
%% NO MOSTRAMOS EL VECTOR PORQUE ES DEMASIADO GRANDE

% Objeto segmetado
final = reshape(M(:, 2), files, cols);
imshow(final)
```



Post-Procesado

Depuramos el resultado eliminando residuos que puedan quedar por un criterio de area.

```
% Eliminamos con un open pixels sueltos o pequeños conjuntos de pixels
IO = imopen(final, strel('disk', 5));
```

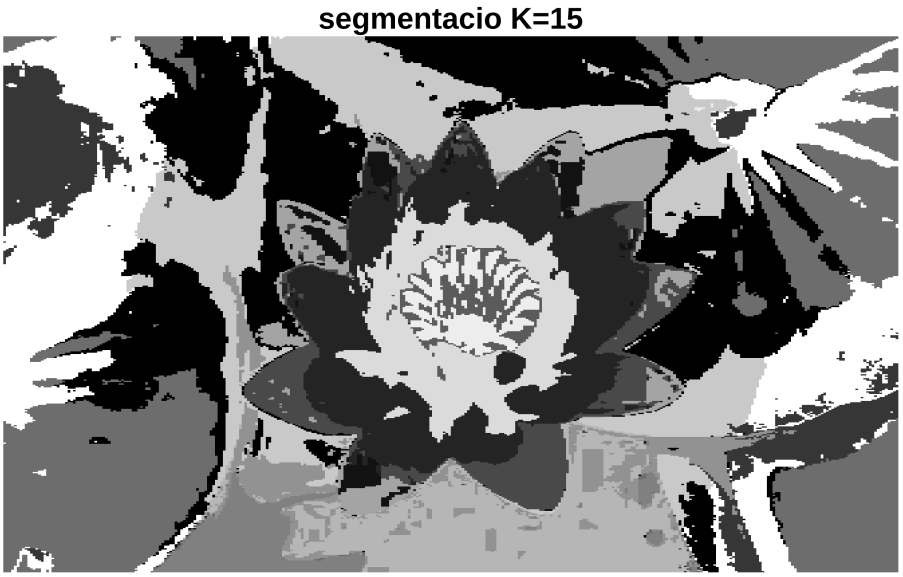
Resultado final

```
% Detectamos el borde del resultado
BC = edge(IO, 'Canny');
% Ampliamos el marco porque apenas se aprecia
BCGrueso = imdilate(BC, strel('disk', 2));
% Mostramos resultado inicial y final
o = imoverlay(I, BCGrueso, 'red');
montage({I, o})
```

Segona part: sense el rectangle

```
centerDist = zeros(files, cols);

for i = 1:files
    for j = 1:cols
        centerDist(i, j) = norm([i, j] - [files/2, cols/2]);
    end
end

% Normalizar los valores de centerDist entre 0 y 1
centerDistNorm = centerDist / max(centerDist(:));

% Convertir centerDist en una matriz de una sola columna
O(:, 5) = centerDistNorm(:);
```

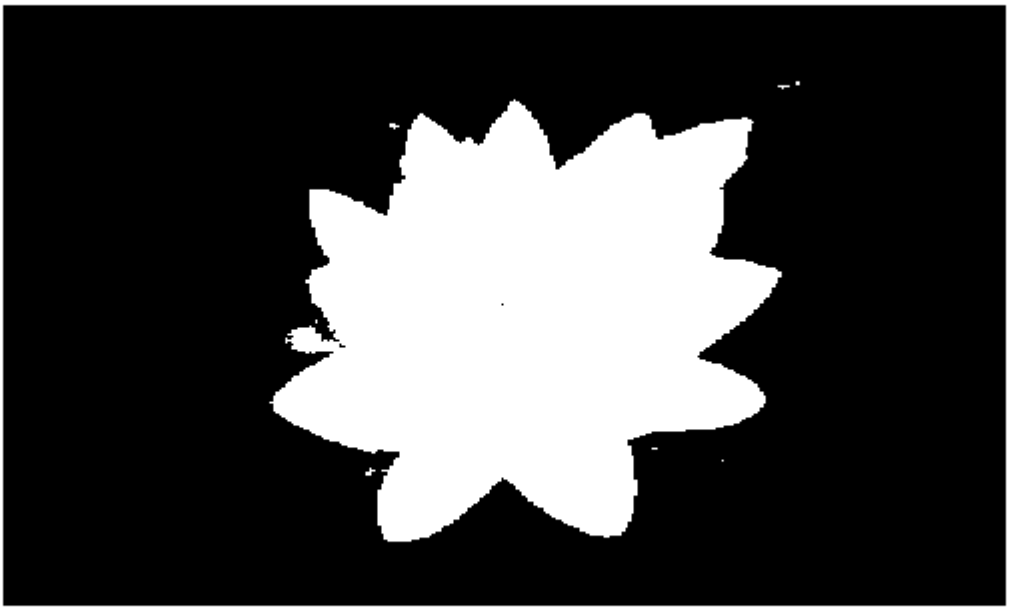
```
k = 15;
C = kmeans(O, k);

eti = reshape(C, files, cols);
figure, imshow(eti, []), title('segmentacio K=15')
```




```
H = [C, centerDist(:)];

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

% Diagonal de la imagen para saber las proporciones
diagonalIM = sqrt(files^2 + cols^2)/2;
proporcio = 0.4;

for i=1:size(H, 1)
    color = H(i, 1);
    if H(i, 2) > diagonalIM * proporcio
        HistO(color, 1) = HistO(color, 1) + 1;
    else
        Hist1(color, 1) = Hist1(color, 1) + 1;
    end
end

RES = Hist1 > HistO;
```

```
M = H;

for i=1:size(H, 1)
```

```
    entra = RES(H(i, 1));
    M(i, 2) = entra && H(i, 2);
end

%% NO MOSTRAMOS EL VECTOR PORQUE ES DEMASIADO GRANDE

% Objeto segmetado
final = reshape(M(:, 2), files, cols);
imshow(final)
```



Aplicamos lo mismo de antes

```
% Eliminamos con un open pixels sueltos o pequeños conjuntos de pixels
IO = imopen(final, strel('disk', 5));

% Detectamos el borde del resultado
BC = edge(IO, 'Canny');
% Ampliamos el marco porque apenas se aprecia
BCGrueso = imdilate(BC, strel('disk', 2));
% Mostramos resultado inicial y final
o = imoverlay(I, BCGrueso, 'red');
montage({I, o})
```