Universitat Politècnica de Catalunya

Visió per Computador
Grau en Enginyeria Informàtica

EXERCICI 7 DE LABORATORI

SEGMENTATION USING *K*-MEANS CLUSTERING

Autor:
Daniel DONATE

Professor:
Manel FRIGOLA
Q2 Curs 2020-2021

13 d'abril del 2021

UNIVERSITAT POLITÈCNICA DE CATALUNYA
BARCELONATECH
Facultat d'Informàtica de Barcelona

FIB

Enunciat de l'exercici

En aquesta setena sessió de laboratori implementarem un sistema semi-automàtic per segmentar objectes/animals en imatges RGB mitjançant una segmentació simple per color, utilitzant l'algoritme de *k*-means (demanarem a l'usuari que indiqui l'objecte a segmentar amb un rectangle). Les imatges a tractar presentaran un objecte molt diferenciat del fons pels seus colors, com la del lotus de la *Figura 1* que es dona a l'enunciat.

Figura 1: Nelumbo nucífera. Veurem la construcció del nostre algoritme a partir d'aquesta foto. ***Autora****: Dora Alis*

Construcció d'un algoritme per segmentar imatges

1. Llegint la imatge i obtenint el rectangle que emmarca l'objecte

Primer, llegim la imatge, comprovem que efectivament es tracta d'una imatge RGB i demanem a l'usuari que emmarqui l'objecte que vol segmentar mitjançant la funció *getrect*, que retorna un vector de quatre elements amb els valors $[xmin, ymin, width, height]$ del rectangle.

```
[file,path] = uigetfile('*.*');
if isequal(file,0)
   disp('User selected Cancel');
else
   I = imread(fullfile(path,file));
   [f,c,nChannels] = size(I);
   if nChannels ~= 3
       disp('You have to choose an RGB image as input');
   else
        imshow(I);
        rect = getrect(); % rect és un array tq [xmin ymin width height]
        ...
```

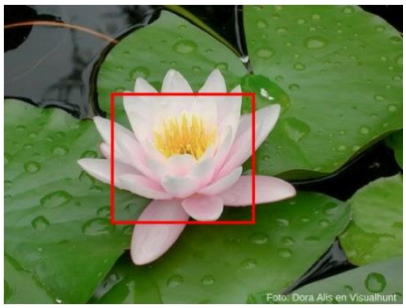

Figura 2: Rectangle introduït per l'usuari que emmarca l'objecte a segmentar

2. Segmentat per *k*-means

Ara convertirem la imatge a l'espai HSV i prepararem la taula O pel *k*-means, que tindrà $f \cdot c$ files (on f i c són el nombre de files i columnes de la imatge original I, respectivament) i les columnes corresponents a les components H, S, V de la imatge en espai HSV.

Agrupem els colors en k classes mitjançant la funció *kmeans*, tot obtenint la classificació C, i visualitzem el resultat provisional obtingut mitjançant les funcions *reshape* (per tornar a passar de la taula a una matriu f per c) i *label2rgb* (per fer un *labeling* dels píxels en pseudo-color).

```
hsv = rgb2hsv(I);
O=reshape(hsv,f*c,3);
k = 20;
C = kmeans(O,k);
IC = reshape(C,f,c);
rgb = label2rgb(IC);
imshow(rgb);
```

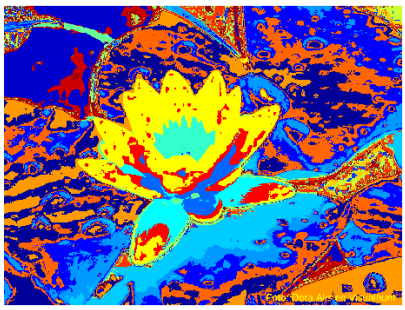

Figura 3: Labeling dels píxels en pseudo-color

3. Construint una màscara per al rectangle que conté l'objecte a segmentar

A continuació construirem una màscara, $MASK$, per veure quins píxels queden a dins del rectangle introduït per l'usuari, i quins a fora. La màscara és simplement una imatge binària amb zeros a fora del rectangle i uns a dins. Podem crear-la amb una sentència com aquesta:

```
MASK = zeros(f,c);
% MASK(top:bottom,left:right) = 1
MASK(rect(2):rect(2)+rect(4),rect(1):rect(1)+rect(3)) = 1;
```

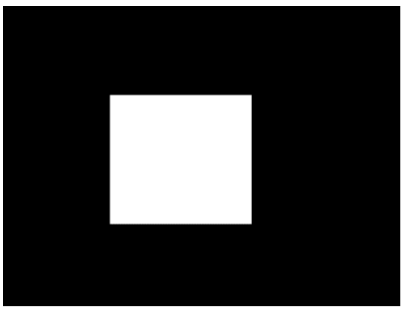

Figura 4: Màscara que ens indica si un píxel està dins o fora del rectangle introduït per l'usuari

Ara podem construir un vector H que ens digui si un color cau dins del rectangle o no:

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

Observem que H és una taula que indica si tal color C cau a dins o a fora de la màscara.

4. Construint els histogrames dels píxels a fora i a dins de la màscara

Per construir els histogrames dels píxels que cauen a fora de la màscara ($Hist0$) i a dins de la màscara ($Hist1$) simplement haurem de comptar, per a cada color de C, quants píxels cauen a fora de $MASK$ (i, per tant, s'afegiran a la seva corresponent entrada a $Hist0$) i quants píxels cauen a dins (la mateixa operació, però afegint-los a la seva entrada a $Hist1$). Com és lògic, la mida dels dos histogrames, $Hist0$ i $Hist1$ ha de ser igual al nombre de classes que hem establert anteriorment, k.

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

5. Decidint quins píxels formen part de l'objecte a segmentar

A partir dels histogrames podem decidir per a cada píxel de la matriu H si forma part o no de la figura que volem segmentar, comparant les seves aparicions dins i fora de la màscara. Guardem la decisió en un vector RES que es pot definir de la següent forma:

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

Així, només cal guardar el resultat en un vector M i mostrar el resultat per pantalla:

```
M = zeros(size(H(:,1)));
for i = 1:size(M)
    class = H(i,1);
    if RES(class) == 1
        M(i) = 1;
    end
end

M = reshape(M,f,c); figure, imshow(M,[]);
```

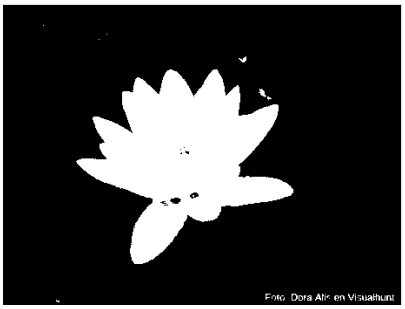

Figura 5: Resultat provisional, sense filtrar

6. Filtrant el resultat provisional

Finalment, filtrem la imatge per a què el resultat no sigui tant sorollós i sigui més compacte. Una possible forma de fer-ho és mitjançant operadors morfològics, amb un *Open(Close(im))*:

```
SE=strel('disk',5);
filt=imopen(imclose(M,SE),SE); % filtre OC
```

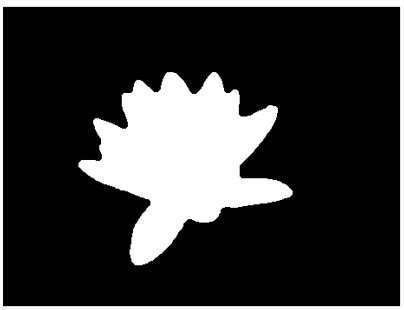

Figura 6: Figura segmentada i filtrada

Amb tot, el perfil del lotus que hem aconseguim a partir de la segmentació realitzada és aquest:

Figura 7: Perfil (en vermell) de l'objecte emmarcat per l'usuari

Resultats obtinguts amb altres imatges

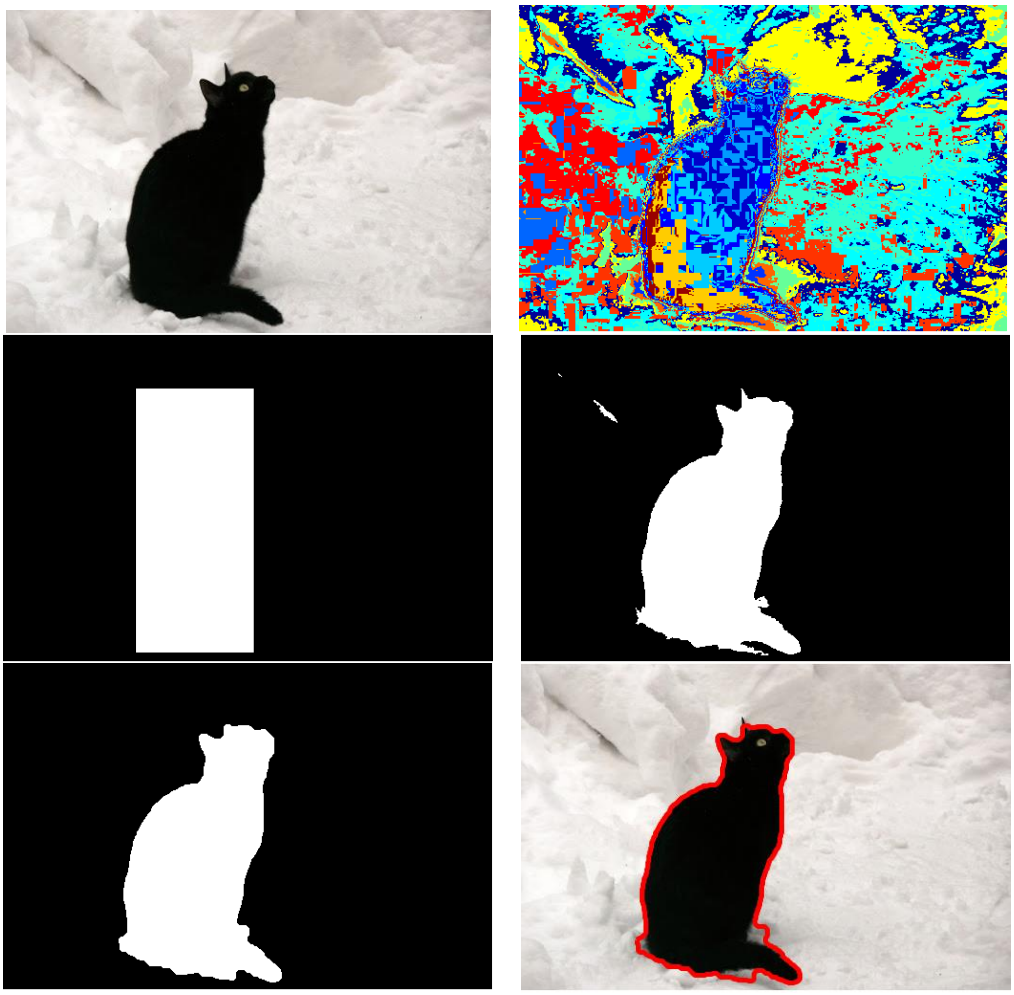

Figura 8: Segmentació d'un gat negre a la neu a partir de la màscara de l'esquerra

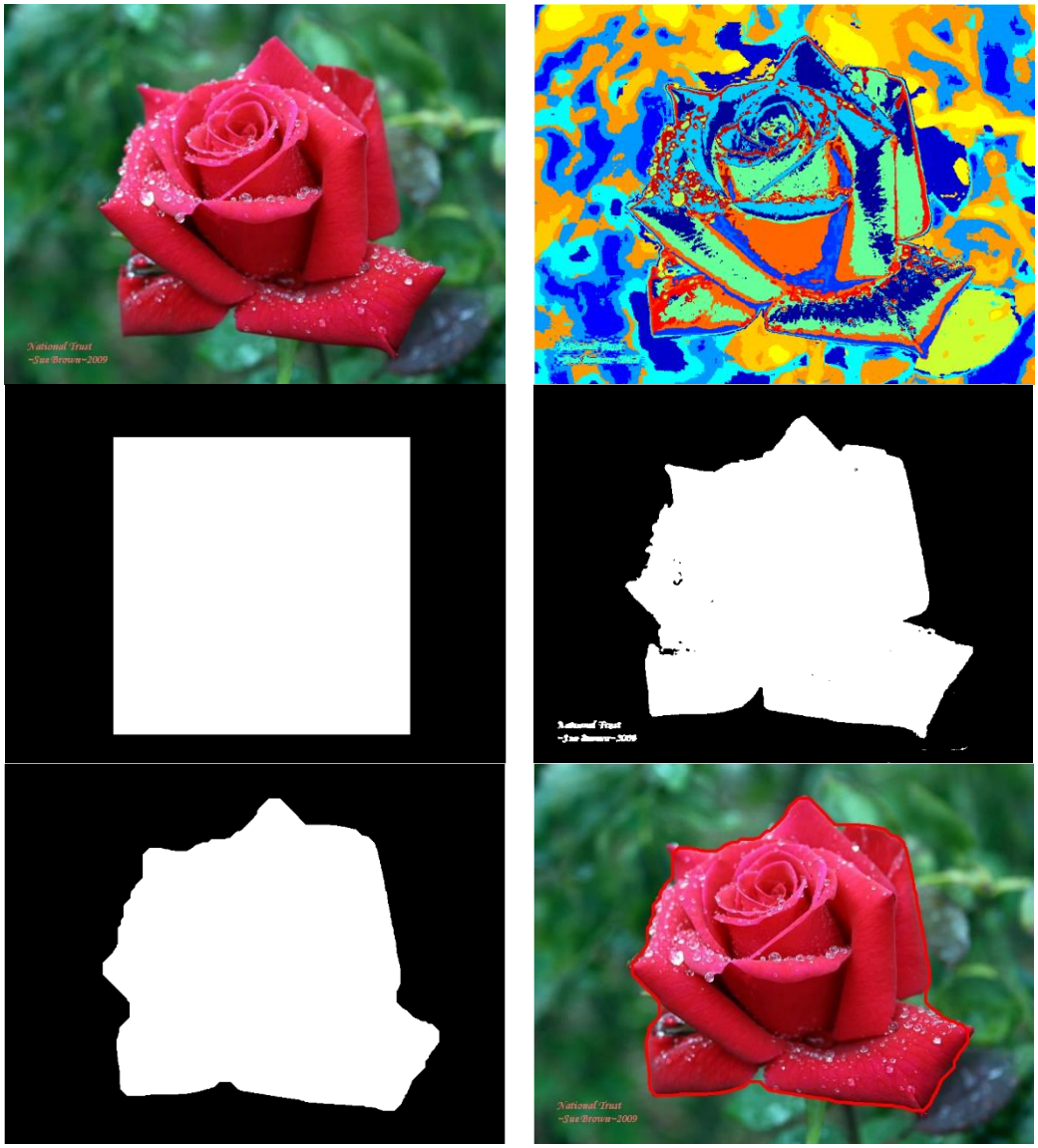

Figura 9: Segmentació d'una rosa a partir de la màscara de l'esquerra