

## VISIÓ PER COMPUTADOR: EXERCICI 7

En aquest exercici implementarem un sistema per segmentar objectes o animals en imatges en color, l'objectiu és que aquesta primera implementació permeti a l'usuari seleccionar una regió rectangular que emmarqui aproximadament el cos a segmentar.



**Figura 1:** Imatge inicial utilitzada.

En primer lloc, carreguem la imatge que utilitzarem de base per desenvolupar l'aplicatiu. Com que està pensat per treballar en imatges RGB, el primer pas serà comprovar-ho mitjançant la funció *size*:

```
I = imread('zebra.jpg');  
  
% Comprovem que sigui una imatge RGB. Si rgb  
% té un valor diferent a 3, I no serà RGB:  
rgb = size(I,3);  
  
if (rgb ~= 3)  
msg = msgbox('La imatge no és RGB!');  
end  
  
% Mostrem la imatge per pantalla:  
imshow(I);
```

Si la imatge carregada no és en format RGB, saltarà un missatge emergent informant a l'usuari.

Un cop el sistema dona la imatge per bona, ja es pot procedir a permetre a l'usuari seleccionar la regió rectangular en qüestió, per fer-ho emprem la funció *getrect*, que permetrà seleccionar interactivament la regió:

```
% Permet a l'usuari seleccionar el rectangle  
% de la imatge sobre el que buscar l'objecte:  
rect = getrect;
```

Així doncs, ara ja tindrem una regió de la imatge especificada per la variable *rect*, que cal constar que té el format  $(x, y, w, h)$ , és a dir, la posició especificada per  $x$  i  $y$ , i l'amplada i l'alçada ( $w$  i  $h$ ).

Tot seguit, cal obtenir la imatge en format HSV i podem començar a preparar la taula pel *kmeans*. Aquesta taula, que l'anomenarem *O*, tindrà tantes files com píxels i quatre columnes:  $[H_x H_y S V]$ , sent  $H_x$  i  $H_y$  les coordenades circulars del hue,  $S$  la saturació i  $V$  el valor de la tonalitat:

```

Ihsv = rgb2hsv(I);

H = Ihsv(:,:,1);
Hx = cos(H); Hx = reshape(Hx,[],1);
Hy = sin(H); Hy = reshape(Hy,[],1);
S = Ihsv(:,:,2); S = reshape(S,[],1);
V = Ihsv(:,:,3); V = reshape(V,[],1);

O = [Hx Hy S V];

```

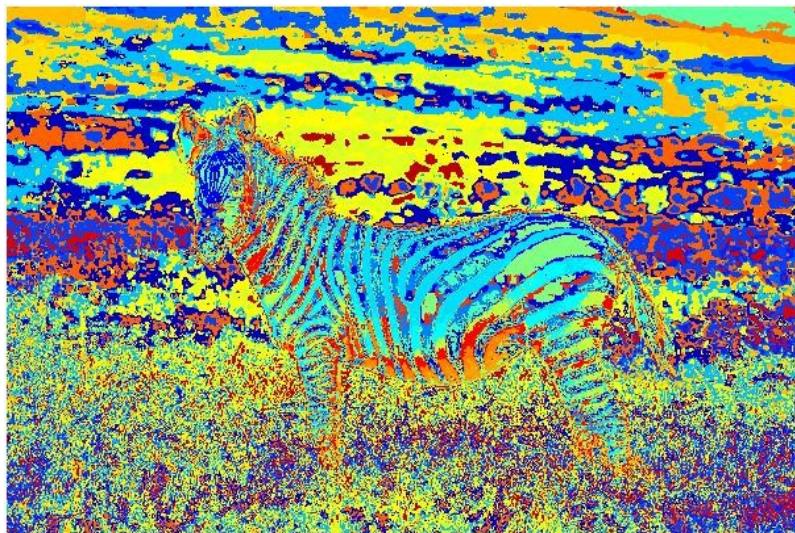
Seguidament, agrupem els colors en  $k$  classes amb *kmeans* i obtenim la classificació  $C$ , a més, definim de nou la mida de la imatge. S'han provat diversos valors de  $k$ , però sembla ser que els millors resultats s'aconsegueixen amb el valor proporcionat inicialment (20). Ho visualitzarem per fer-nos una idea de com està avançant el procés:

```

K = 20; C = kmeans(O,K);

mida = size(I);
IC = reshape(C,mida(1),mida(2)); NOVA = label2rgb(IC); imshow(NOVA);

```



**Figura 2:** Resultat de la primera etapa de procés en pseudo-color, la zebra es pot distingir prou bé.

```

% Crearem una màscara a partir del rectangle seleccionat, els
% valors fora del rectangle seran 0, altrament seran 1:
MASK = zeros(mida(1),mida(2));
MASK(rect(2):rect(2)+rect(4),rect(1):rect(1)+rect(3)) = 1;

imshow(MASK);

```



**Figura 3:** Màscara binària d'una àrea seleccionada per l'usuari.

A continuació, els passos finals per aconseguir la nostra imatge processada:

```
% Construïm un vector per indicar si un color
% cau dins de la regió seleccionada, o no:
H = [C,MASK(:)];

% Comptem per cada color quants píxels són fora
% del rectangle i quants a dins, guardarem els
% resultats en dos vectors:
Hist0 = hist(H(H(:,2)==0),[1:K]);
Hist1 = hist(H(H(:,2)==1),[1:K]);

% Decidim si un color pertany o no a la figura a
% partir de comparar les aparicions dins i fora del
% rectangle:
RES = Hist1 > Hist0;

% Decidim quins píxels formen part de la figura:
M = RES(H(:,1));

Mimage = reshape(M,mida(1),mida(2));

imshow(Mimage);
```

Finalment, aplicarem un senzill joc de filtres per intentar millorar-ne lleugerament els resultats, el primer per omplir espais buits i el segon per eliminar els píxels de soroll:

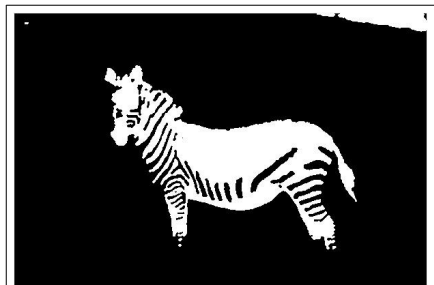
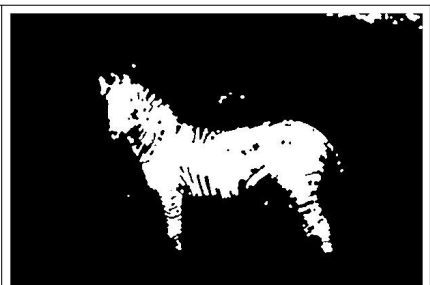
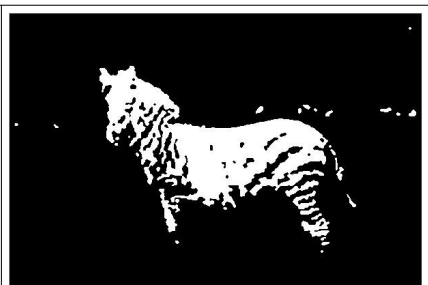
```
Mimage = bwmorph(Mimage, 'fill', 10);
Mimage = bwmorph(Mimage, 'majority', 10);

imshow(Mimage);
```



**Figura 4:** A l'esquerra: el resultat provisional, la zebra és detectada gairebé a la perfecció. A la dreta: el resultat final, la zebra és detectada més consistentment, i s'ha minimitzat el soroll.

Aprofitant que s'ha finalitzat satisfactòriament el desenvolupament de l'aplicació demanada, procedim a jugar una mica amb les diverses opcions del programa. Per exemple, provarem a modificar les característiques de la taula *O*, sense modificar cap altre aspecte del codi:

		
<p>Deixant la taula tal que <math>O = [V]</math>, s'observa com malgrat que la zebra es distingeix prou bé de la resta de la imatge, les ratlles negres de la seva pell no s'han identificat prou bé, i que realment en bona part es deu als filtres implementats, sense ells segurament el resultat encara seria pitjor.</p>	<p>Pel que fa a la taula sent <math>O = [S]</math>, el resultat implica una mica més de soroll dispers que en la versió prèvia, tot i que a <i>grosso modo</i> les ratlles de la zebra són detectades de forma més precisa. En general, tal com en el cas anterior, la zebra es pot distingir bé de la resta de l'escena.</p>	<p>Finalment, tenim el cas de la taula sent <math>O = [Hx Hy]</math>. Aquest és, de totes tres proves, la que dona pitjors resultats. La zebra es segueix distingint mínimament bé, però és el cas en què la seva forma queda més desdibuixada: directament la porta del davant pràcticament desapareix, i tot el contorn inferior de l'animal és molt poc precís.</p>

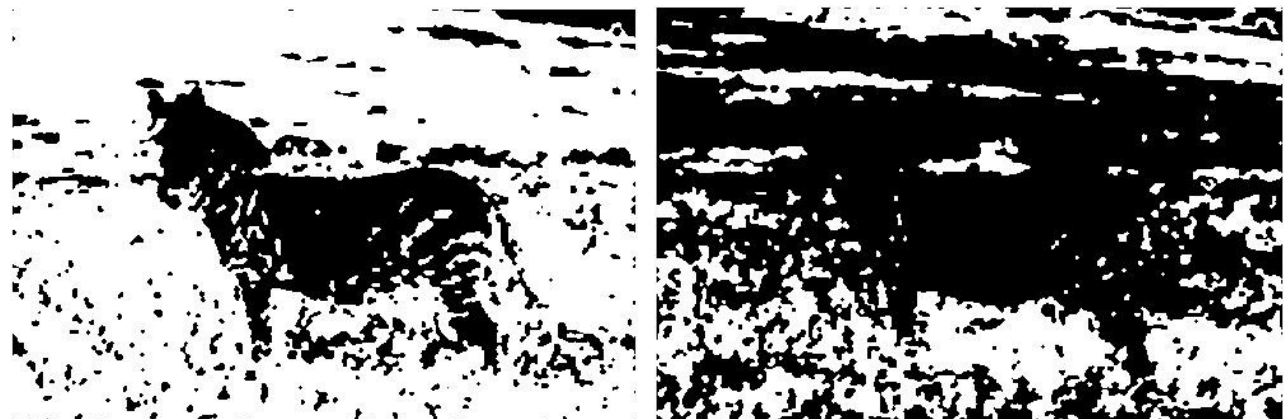
També plantegem un concepte diferent (tornant al concepte inicial de la taula  $O$ ) en què l'usuari no ha de seleccionar cap regió, sinó que **senzillament ha d'especificar un punt**, el color del qual s'utilitzarà com a referent per trobar la resta de punts de la imatge que siguin semblants.

Bàsicament hem portat a terme dues modificacions, per una banda, el *getrect* és substituït pel *getpts*, aquesta funcionalitat permet a l'usuari assenyalar tants punts com vulgui, assumirem que només n'assenyalarà un per cada execució. Seguidament, computem la variable *HPoint*, i l'emprarem per generar una màscara:

```
[ptx, pty] = getpts;
ptx = ceil(ptx);
pty = ceil(pty);
Ihsv = rgb2hsv(I);
```

[...]

```
HPoint = H(ptx, pty);
MASK = (H >= HPoint - 0.03) & (H <= HPoint + 0.03);
```



**Figura 5:** A l'esquerra el resultat de clicar a una de les ratlles negres de la zebra. A la dreta el resultat de clicar una de les ratlles blanques de la zebra.