VISIÓ PER COMPUTADORS

Informe Sessió 5

Carles Llongueras

Carlos Plana García

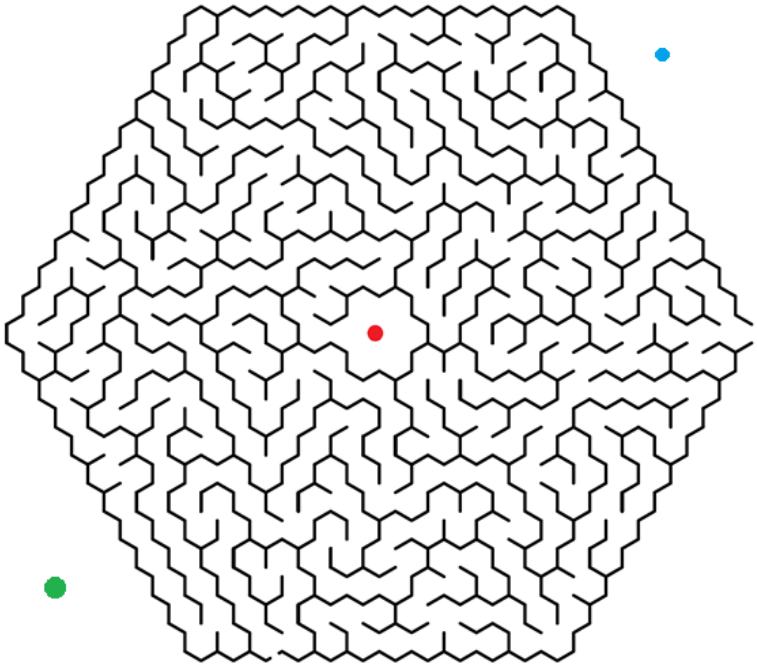
Pablo Gámiz Nieto

Curs 2021-2022

Lectura de la informació de la imatge

Primer, llegim la imatge proporcionada per fer l'exercici i, a continuació, la convertim a escala de grisos per obtenir la informació necessària més endavant.

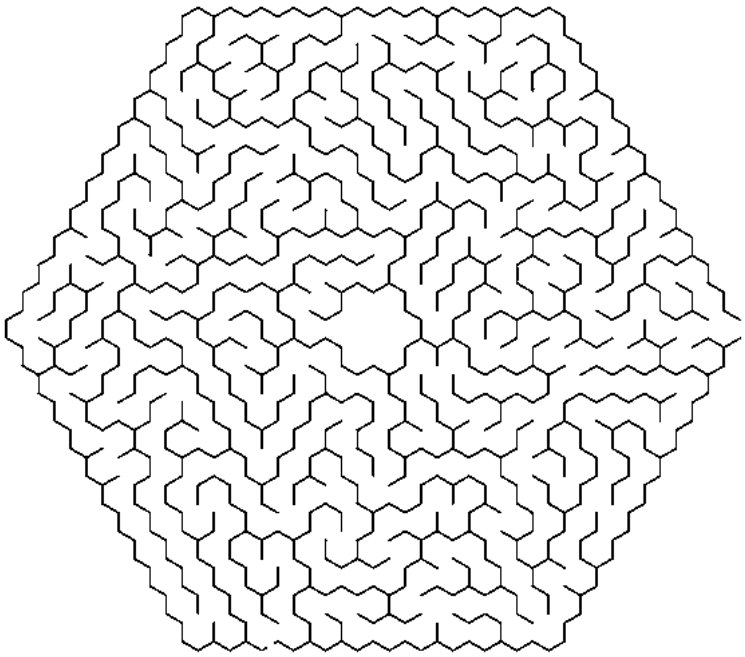
```
A = imread("Laberint.png");
G = rgb2gray(A);
imshow(A);
```



Creem un element estructurant de 3x3 pixels (connectivitat 8) què farem servir per recórrer la figura del laberint. A continuació, creem una matriu binaria per cada color que conté la posició de cadascun d'aquests i, a més, obtenim una matriu que conté la figura del laberint (binarització de la figura original).

```
SE = [1 1 1; 1 1 1; 1 1 1];
%GREEN = [0, 0.5, 0.35] [0, 127.5, 89.25] APROXIMADAMENT
%BLUE = [0.3010, 0.6314, 0.9330] [76.75, 161, 237.915] APROXIMADAMENT
%RED = [1,0,0] [255,0,0] APROXIMADAMENT
GREEN = A(:,:,1) < 50 & A(:,:,2) > 100 & A(:,:,3) < 80;
BLUE = A(:,:,1) < 100 & A(:,:,2) < 200 & A(:,:,3) > 200;
RED = A(:,:,1) > 200 & A(:,:,2) < 50 & A(:,:,3) < 50;
BIN = G>50;
```

```
imshow(BIN);
```



Trobar el camí més curt

Per trobar el camí més curt des d'un dels colors fora del laberint fins al centre, hem creat la funció **pixelPath** que ens ajudarà a trobar-lo i ens dirà el nombre de píxels d'aquest. Aquesta funció té com a paràmetres *color_ini* que representa la matriu del color inicial des d'on partirem, *color_end* que representa la matriu del color final on hem d'arribar, *expansion_shape* que representa l'element estructurant que farem servir per anar dilatant el camí i *BIN* que representa la matriu binària que conté la forma del laberint. Dins de la funció, primer creem *Aux* on tindrem la posició actual. En segon lloc, passem la matriu del color inicial a double i la guardem a *Dist_Mat*, on guardarem el camí actual, i creem una variable *Pixels*, on guardarem el nombre de píxels actual del camí. A continuació, fem un bucle que iterarà fins que les matrius *Aux* i color_end coincideixin, és a dir, fins que la posició actual no sigui la del color final. Mentre no hàgim arribat, augmentem en 1 el valor de píxels recorreguts i dilatem la figura que es troba a la matriu *Aux* amb l'element estructurant passat per paràmetre. Llavors, restem la matriu abans de dilatar a la que hem obtingut després de fer-ho obtenint la nova posició actual, la multipliquem pel nombre de píxels i la sumem a la matriu *Dist_Mat*. Una vegada hem arribat al color final, la funció acaba i retornem els valor de *Dist_Mat* i *Píxels*, que contindran respectivament el camí des del color inicial fins al final i el nombre de píxels d'aquest.

```
function [Dist_Mat,Pixels] = pixelPath(color_ini,color_end,expansion_shape,BIN)
    Aux = color_ini;
    Dist_Mat = im2double(color_ini);
```

```
    Pixels = 1;
    while sum(sum(Aux&color_end)) == 0
        Pixels = Pixels + 1;
        Before = Aux;
        Aux = imdilate(Aux,expansion_shape);
        Aux = Aux&BIN;
        Painted = Aux - Before;
        Dist_Mat = Dist_Mat + Painted*Pixels;
    end
end
```

Executem la funció dues vegades per cada color, una per anar del color inicial al final, i un altre per anar del color final a l'inicial. Això ens retornarà dues matrius de distància per cada color i el número de píxels mínims per anar del color inicial al color final. Mostrant el valor de N_GREEN i N_BLUE, trobem que el nombre de píxels del camí que comença pel color verd és 1001, mentre que el del camí que comença pel color blau és 1332.

```
[GREEN_PIXELS, N_GREEN] = pixelPath(GREEN,RED,SE,BIN);
N_GREEN
```

```
N_GREEN = 1011
```

```
[GREEN_INVERSE_PIXELS, IDEM] = pixelPath(RED,GREEN,SE,BIN);
[BLUE_PIXELS, N_BLUE] = pixelPath(BLUE,RED,SE,BIN);
N_BLUE
```

```
N_BLUE = 1332
```

```
[BLUE_INVERSE_PIXELS, IDEM2] = pixelPath(RED,BLUE,SE,BIN);
```

Amb la funció **representation** obtindrem el camí final amb una línia. Els paràmetres de la funció inclouen els dos camins obtinguts anteriorment amb la funció pixelPath i la imatge binarizada amb el contorn del laberint. En primer lloc fem la suma de les dues imatges dels camins i després agafem els píxels que siguin iguals a la suma dels dos camins més 1, ja que tots el punts del camí més curt han de tenir aquest valor (la suma d'anar d'un punt inicial a un punt A i anar del punt final al punt A és equivalent a la distancia de tot el camí). Després d'això utilitzem la funció bwskel que donat una forma en 2D redueix aquesta a un pixel d'amplada mantenint la topologia, per aconseguir així el camí més curt. Per últim fem un dilate al camí perquè es vegi millor quan es faci la superposició amb la imatge original.

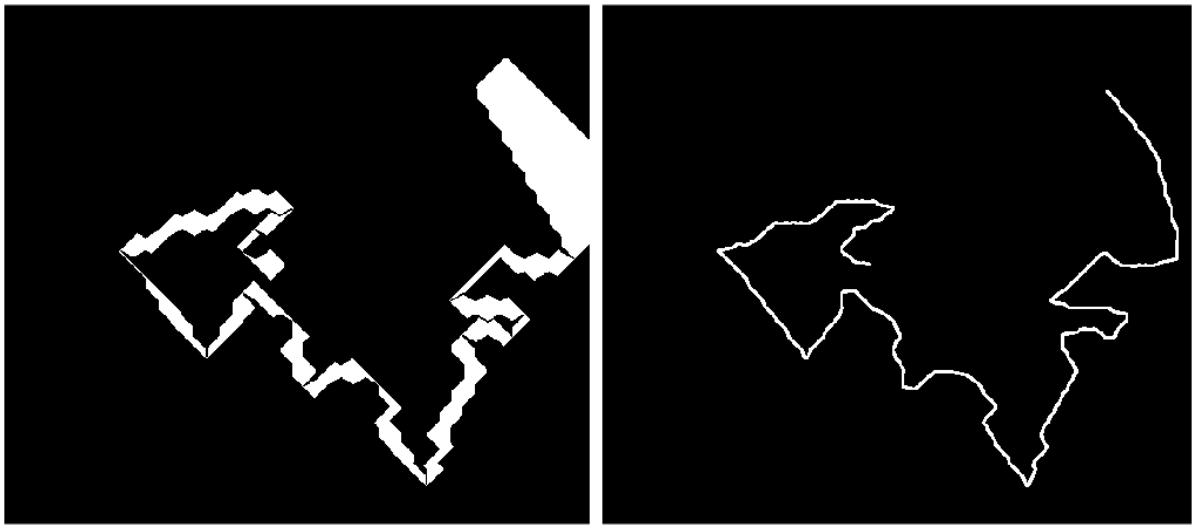
```
function Dist_total = representation(Color_Pixels,Inverser_Color_Pixels, N)
    SE = [1 1 1; 1 1 1; 1 1 1];
    Dist_total = Color_Pixels + Inverser_Color_Pixels;
    Dist_total = Dist_total == N+1;
```

```
    imshow(Dist_total);
    Dist_total = bwskel(Dist_total);
    Dist_total = imdilate(Dist_total,SE);
    imshow(Dist_total);
end
```

Aquesta funció l'executem una vegada per al color verd i una altra per al blau, obtenint el camí més curt entre el color verd i el vermell, i el color blau i el color vermell.

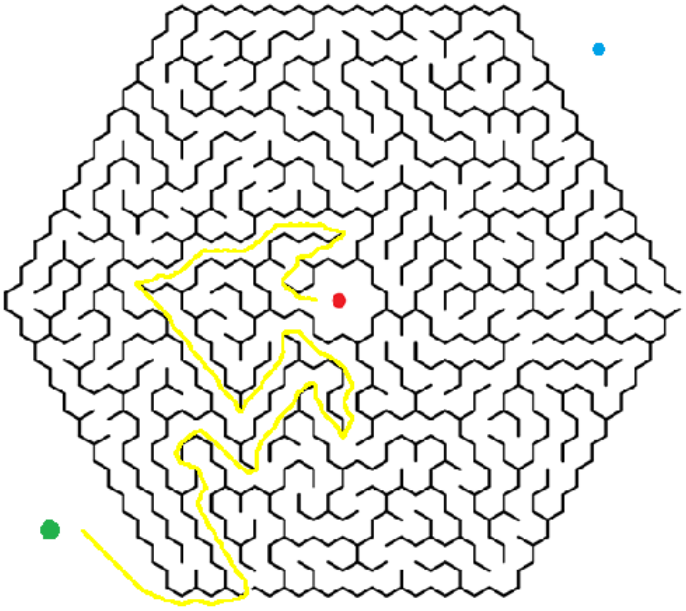
```
GREEN_TOTAL = representation(GREEN_PIXELS,GREEN_INVERSE_PIXELS,N_GREEN);
```

```
BLUE_TOTAL = representation(BLUE_PIXELS,BLUE_INVERSE_PIXELS,N_BLUE);
```



Per últim fem overlay de la solució amb la imatge original per veure el resultat final.

```
GREEN_END = imoverlay(A,GREEN_TOTAL,'yellow');
BLUE_END = imoverlay(A,BLUE_TOTAL,'yellow');
imshow(GREEN_END);
```



```
imshow(BLUE_END);
```