VISIÓ PER

COMPUTADOR

Informe de laboratori 5

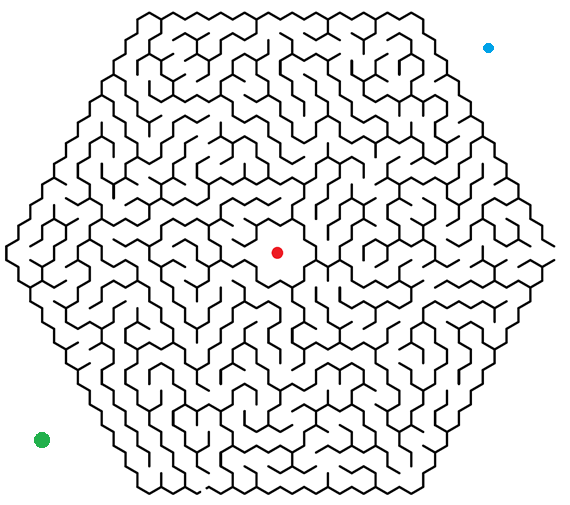
Marc Cervilla Rovira

24/03/2021

Introducció:

En aquesta sessió de laboratori, l’objectiu serà crear un programa de visió per computador que ens trobi el camí més curt entre dos punts, tenint en compte que podem tenir obstacles per al mig.

En el nostre cas, tindrem un laberint com a obstacle, dos punts d’origen(verd i blau) i un punt de destí (vermell) tal com es mostra a la imatge següent.



Considerarem que l’objecte té el tamany d’un pixel i el problema ens demana:

* Indicar la distància en píxels del camí més curt desde cadascun dels punts perifèrics fins al centre
* I mostrar aquest camí més curt de color groc a la imatge.

Implementació:

En primer lloc, llegim la imatge i la llegeixo també en gris per si ho necessitem.

I = imread('Laberint.png');

G = rgb2gray(I);

Creo l’element estructurant que utilitzarem, en aquest cas de veïnatge 8, ja que vull que el camí pugui anar per qualsevol dels píxels veins.

Si volguessim utilitzar veïnatge 4 tan sols hauriem de treure els 1s de la diagonal.

SE = [1,1,1;1,1,1;1,1,1];

A continuació creo les imatges binàries que dilatarem més endevant amb els 3 punts que ens interessen, verd i blau seràn els punts inicials (per canviar els punts només hem de comentar i descomentar) i el vermell serà el punt final.

Per trobar els punts utilitzo uns llindars aplicats a les components RGB.

%punt inicial verd

ini = I(:,:,2)>100 & I(:,:,1)<100 & I(:,:,3)<100;

%punt inicial blau

%ini = I(:,:,2)<200 & I(:,:,1)<50 & I(:,:,3)>100;

%punt final vermell

fin = I(:,:,1)>100 & I(:,:,2)<100 & I(:,:,3)<100;

També hem de generar la imatge binaria del laberint per a que actuï com a imatge condició, per fer-ho utilitzo un llindar en la imatge en nivells de gris.

%imatge condicio (laberint)

lab = G>50;

Per arribar desde un punt perifèric fins al punt vermell utilitzem la dilatació condicional molts cops, és a dir, una reconstrucció.

El que fem és, a cada iteració del bucle, anar dilatant el punt inicial i, just després, aplicar la intersecció amb la imatge condició, en el nostre cas la imatge binària del laberint, per tant, com que anem de píxel en píxel no creua mai les parets.

Executem el bucle fins que la dilatació arriba al punt final, en el meu cas, he pensat que seria quan la intersecció entre el punt final i el nostre camí donessin algun punt en comú i, per tant, quan la suma de la matriu intersecció sigui diferent de 0 sortim del bucle.

F = ini;

pixels = 0;

while sum(sum(F&fin))==0

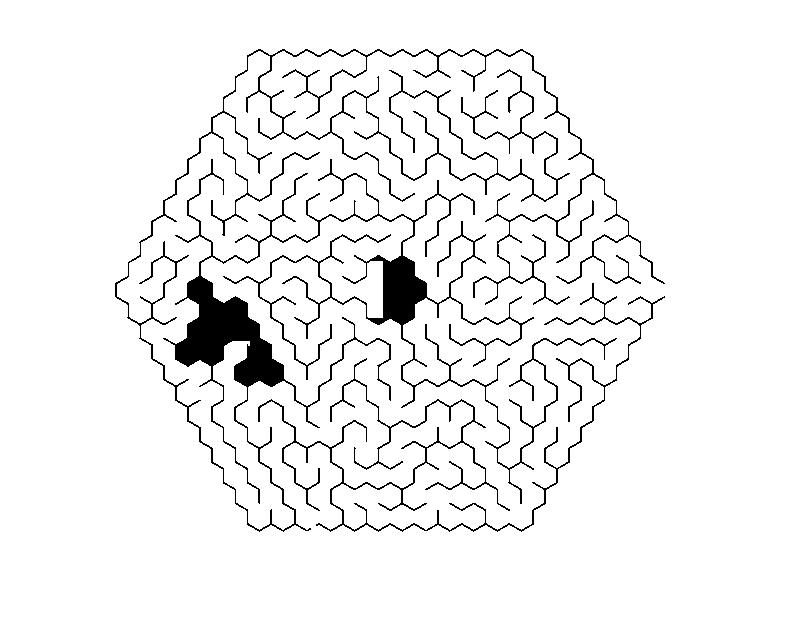
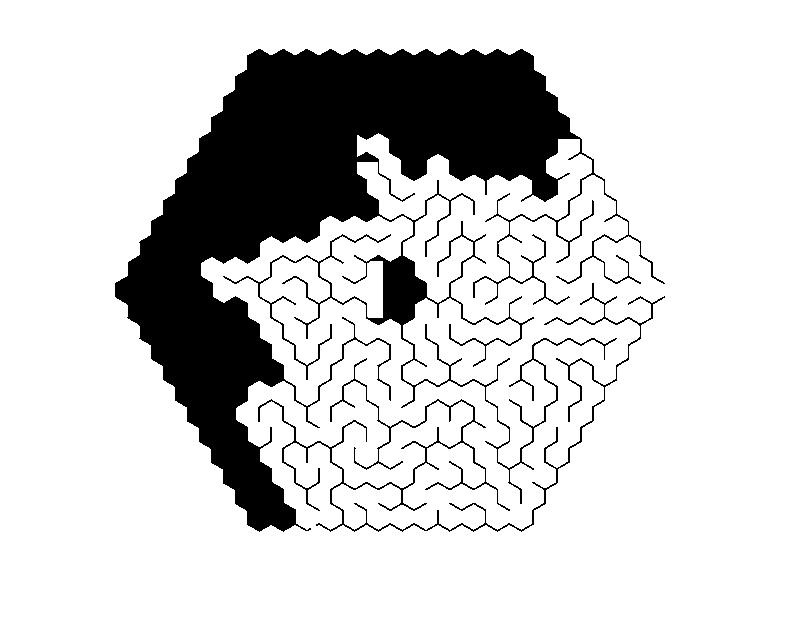
pixels = pixels+1;

F = imdilate(F,SE);

F = F&lab;

end

En les següents imatges podem veure com es fa aquesta reconstrucció en el moment final que hem arribat al centre:



desde el punt verd desde el punt blau

Com que a cada pas del bucle només avancem 1 píxel en qualsevol direcció, això vol dir que en el moment que arribem al punt final haurem recorregut el mateix nombre de píxels que de iteracions, per tant, ficant un contador podem obtenir la primera part del problema que ens demanava el camí més curt:

Camí més curt verd: 1011

Camí més curt blau: 1332

En la segona part del problema ens demanen pintar aquest camí, així que la cosa es complica una mica. Per a resoldre-ho, el que he fet ha sigut crear una matriu de distàncies que ens mostrarà, per cada píxel que hem pintat amb les dilatacions, a quina distància es troben del punt inicial. Per fer-ho he modificat el bucle anterior afegint una part on calculem quins són els píxels que hem pintat i, a partir d'aquí, podem trobar a quina distància es troben de la següent manera:

*(He utilitzat matrius de doubles ja que el rang era major que 255)*

dist = im2double(ini);

while sum(sum(F&fin))==0

pixels = pixels+1;

abans = F;

F = imdilate(F,SE);

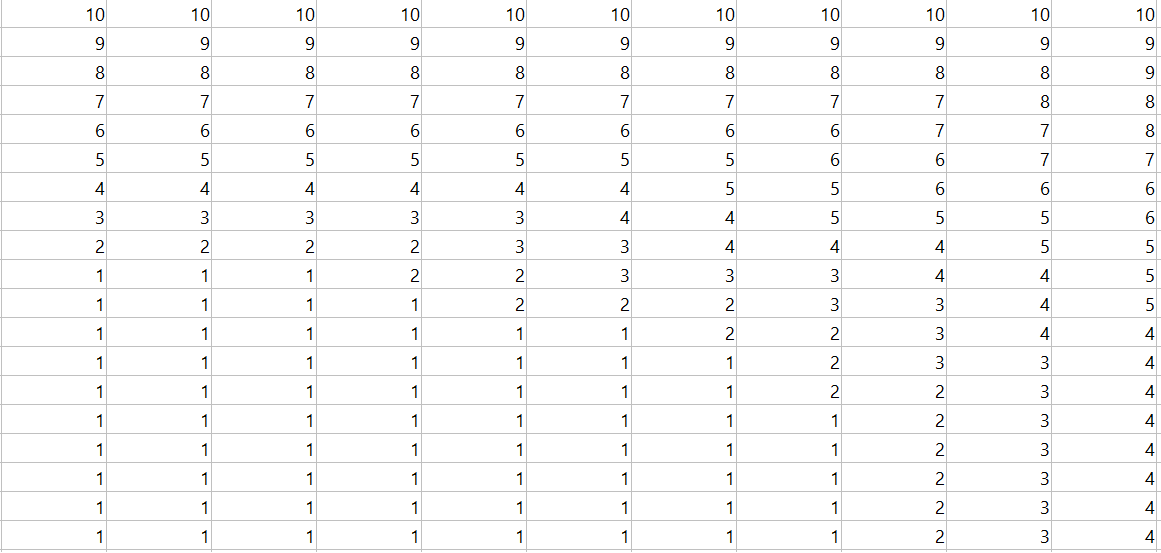
F = F&lab;

pintats = F-abans;

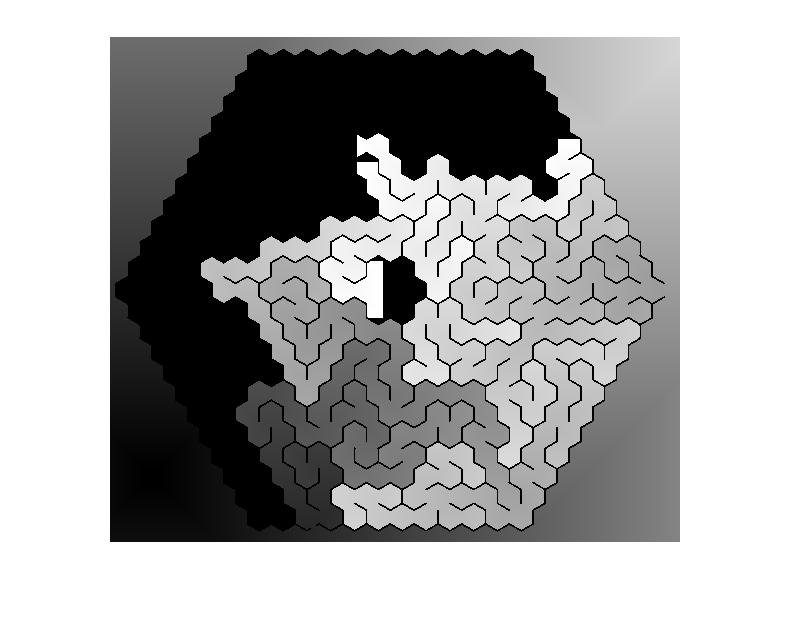
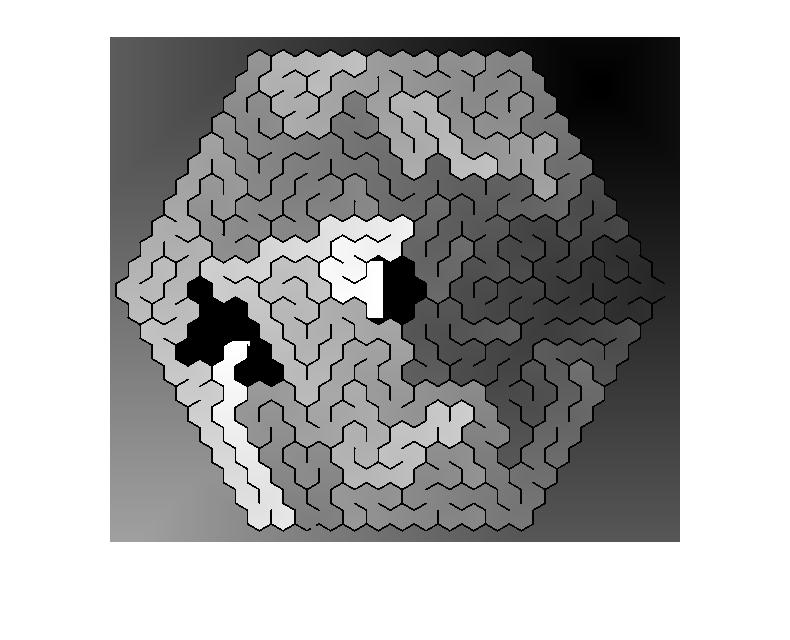
dist = dist + pintats\*pixels;

end

La matriu de distància que ens genera és d’aquest estil:



Per veure-ho més gràficament la podem pintar, a mesura que s’acosta és més blanc:



desde el punt verd desde el punt blau

Però amb les distàncies del punt inicial al final no són suficients per trobar el camí més curt, per fer-ho torno a fer el mateix procediment però, aquest cop, per calcular les distancies desde el punt final fins al inicial, simplement canviem els punts inicial i final:

ini2 = fin;

fin2 = ini;

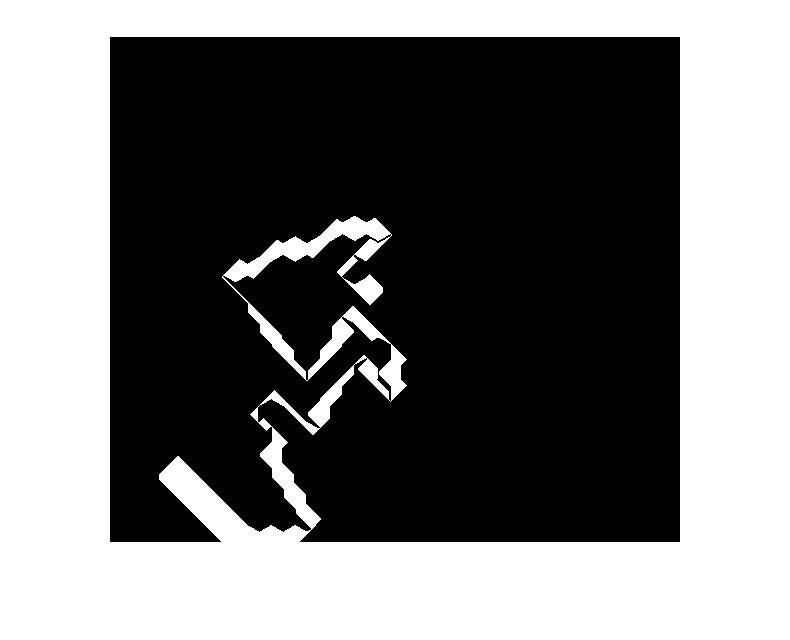
i fem exactament el mateix bucle que abans per calcular la matriu *dist2*.

Un cop tenim les 2 matrius en les dues direccions, si les sumem ens donarà un camí a distancia mínima, que serà les celes que tenen el valor del camí més curt (pixels+1):

distTOT = dist+dist2;

distTOT = distTOT==pixels+1;

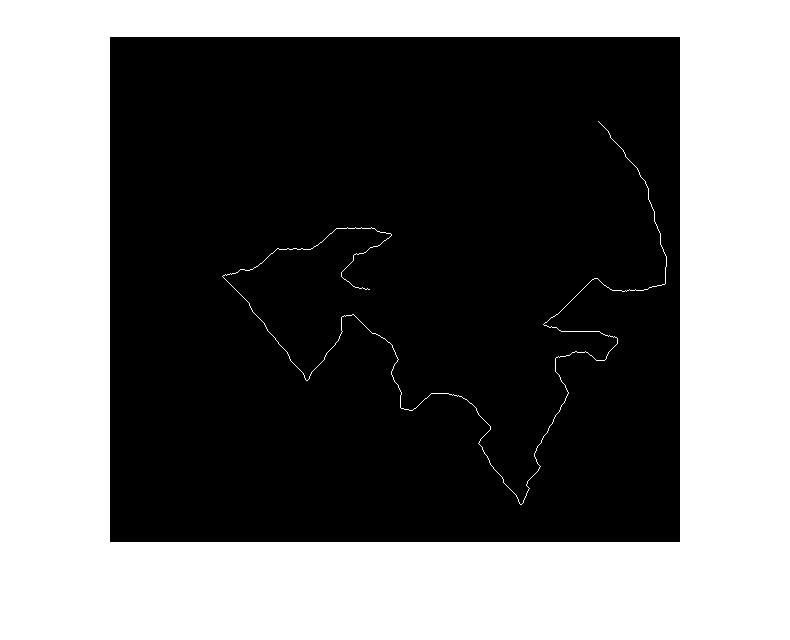
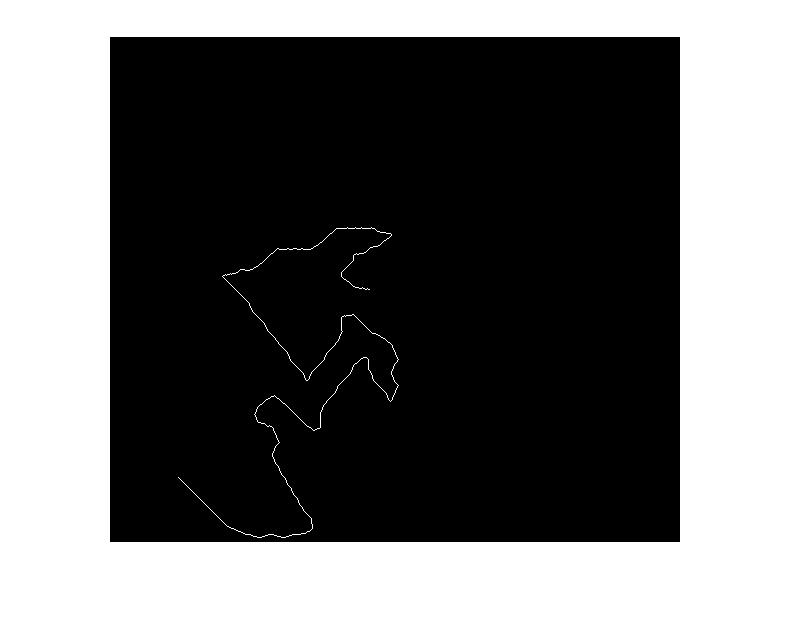
Si ho mostrem veurem el següent:



Finalment, el que he fet ha sigut utilitzar la funció *bwskel* que redueix el camí anterior per a que sigui per un sol píxel.

En aquest punt el camí deixa de tocar ben bé els punts de colors, però penso que el resultat es veu bé, si volguessim que toques podriem utilitzar erode.

distTOT = bwskel(distTOT);



Per últim, com que és veu bastant petit he fet un altre dilatació per a que s’apreci millor (per veure únicament els punts del camí l’hauriem de treure i ja està) i aplicant un *overlay* a la imatge ens queda el resultat final següent:

distTOT = imdilate(distTOT,SE);

FINAL = imoverlay(I,distTOT,'yellow');

