**VC: Informe de Laboratori 4**

**Processat morfològic d’imatges**

****

**Pere Ginebra Solanellas**

**15/3/2021 – Q2 Curs 2020-21**

**Visió per Computador, FIB UPC**

1. **Introducció**

En aquesta sessió començarem a utilitzar operadors morfològics a imatges binàries (obtingudes d’una imatge RGB). En concret utilitzarem aquests operadors per a aïllar patrons de colors i formes per tal de trobar una figura amagada en una imatge (un home vestit de ratlles grogues i negres).

1. **Exercici**

Per començar, importem la imatge li apliquem un filtre de mediana als seus 3 canals de color amb *medfilt2* i la convertim a format HSV amb *rgb2hsv*:

|  |
| --- |
|  |
| Figura 2.1. Importació i suavitzat de la imatge inicial |

A continuació, per tal de poder trobar les ratlles grogues i negres de l’Odlaw, aïllarem els elements groc i negres de la imatge en dos imatges lògiques (o binàries). Això ho podem fer marcant els píxels amb un valor de Hue entre 48 i 52 (0.133 i 0.144 en format double) pels grocs i els píxels amb saturació inferior a 20 (0.2 en format double):

|  |
| --- |
|  |
| Figura 2.2. Extracció de dos imatges binàries corresponent amb els elements grocs i negres |

Ara ja podem declarar els elements estructurals necessaris i utilitzar-los per a aïllar les zones de les imatges binàries amb ratlles. Per això primer aplicarem un operador d’erosió amb un element en forma de recta vertical ( *imerode(\_, SE7)* ) per eliminar totes les ratlles horitzontals de la imatge, i a continuació restarem el resultat a la seva imatge lògica inicial, obtenint així les línies horitzontals aïllades. Fet això, ajuntarem ara les línies properes amb un operador de tancament i eliminarem les que quedin soles amb un altre d’obertura, ambdós amb elements circulars.

|  |
| --- |
|  |
| Figura 2.3. Aïllament dels conjunts de línies horitzontals properes a cada imatge binària |

Els operadors de tancament “tanquen” els forats més petits que l’element estructural entre dos figures (en aquest cas forats entre línies properes ), i els de obertura eliminen les figures més petites que l’element (en aquest cas les línies que queden aïllades).

Finalment només ens queda intersecar les dos imatges binàries amb una operació *&* (‘and’) i ampliar l’element comú amb un operador de dilatació per marcar tota la figura de l’Odlaw:

|  |
| --- |
|  |
| Figura 2.4. Intersecció dels conjunts de línies per trobar una zona amb ratlles grogues i negres |

Per representar el resultat, convertirem la imatge binaria a RGB posant els píxels a 1 de color vermell i la resta de color negre i la imatge original a escala de grisos RGB amb *rgb2gray* i tornant-la a concatenar en 3 dimensions. Fet això sumarem les dos imatges RGB per obtenir la localització de l’Odlaw marcada amb una taca vermella.

|  |
| --- |
|  |
| Figura 2.5. Representació i impressió dels resultats obtinguts |

Per visualitzar aquests resultats imprimirem per pantalla dos muntatges, un amb la imatge inicial i la final i un altre amb uns quants passos entremitjos per poder observar els resultats d’alguns operadors.

1. **Resultats**

|  |
| --- |
|  |
| Figura 3.1. Imatge resultant a on podem veure l’Odlaw a sota la taca vermella |

|  |
| --- |
|  |
| Figura 3.2. Imatge resultant presentada junt amb la original |

|  |
| --- |
|  |
| Figura 3.3. Resultats intermitjos. Original, píxels grocs, negres i conjunts de ratlles grogues i negres, i resultat final: d’esquerra a dreta i de dalt a baix. |

En aquest últim muntatge podem veure les imatges binaries originals, en les que podem veure l’abundància de groc a la imatge i l’escassedat de negre i tons molt foscos. També podem veure que el resultat d’aïllar els conjunts de línies ens acota el nombre de elements foscos possibles a 2 i els grocs a uns quants més però també reduït. Finalment podem veure com la intersecció de les dos penúltimes imatges ens dona el resultat final.

1. **Annexos / Script**

|  |
| --- |
| clear  close all    img = imread('Wally.png');  img1(:,:,1) = medfilt2(img(:,:,1), [1 4]);  img1(:,:,2) = medfilt2(img(:,:,2), [1 4]);  img1(:,:,3) = medfilt2(img(:,:,3), [1 4]);  imgHSV = rgb2hsv(img1);    yello = 0.133 <= imgHSV(:,:,1) <= 0.144;  blck = imgHSV(:,:,3) <= 0.2;    SE7 = strel('line',7,90);  SE14 = strel('sphere',14);  SE4 = strel('sphere',4);  SE2 = strel('sphere',2);  SE1 = strel('sphere',1);    ystripes = imerode(yello, SE7);  ystripes = yello - ystripes;  ystripes = imclose(ystripes, SE2);  ystripes = imopen(ystripes, SE4);  ystripes = imdilate(ystripes, SE1);    bstripes1 = imerode(blck, SE7);  bstripes = blck - bstripes1;  bstripes = imclose(bstripes, SE2);  bstripes = imopen(bstripes, SE2);  bstripes = imdilate(bstripes, SE1);    res = bstripes & ystripes;  res = imdilate(res, SE14);    resRGB(:,:,1) = uint8(255\*res(:,:));  resRGB(:,:,2) = 0;  resRGB(:,:,3) = 0;    backGray = rgb2gray(img);  backGrayRGB = cat(3, backGray, backGray, backGray);    finalRGB = backGrayRGB + resRGB;    figure  montage({img, yello, blck, ystripes, bstripes, finalRGB})  figure  montage({img, finalRGB})  figure  imshow(finalRGB); |
| Figura 4.1. Script utilitzat per la sessió |

1. **Bibliografia / Documentació**

* [RGB grayscale](https://es.mathworks.com/matlabcentral/answers/67137-convert-gray-image-back-to-rgb)
* <http://colorizer.org/>