Visió per Computador

ENTREGA 3

UNIVERSITAT POLITÈCNICA DE CATALUNYA
BARCELONATECH
Facultat d'Informàtica de Barcelona

FIB

Raúl García Fuentes

Kleber Enrique Reyes Illescas

9 d'Octubre de 2018

1. **Implementar d'una manera eficient un filtre de mitjana intentant minimitzar el nombre total d'operacions (sumes, productes, etc.) realitzades pel filtre.**

```
>> I = imread('Bird24b.bmp');
>> G = rgb2gray(I);
>> imshow(G);

>> G = double(G);

>> out = func3_1(G);
>> figure; imshow(out,[]);
```

```
function [Res] = func3_1(I)
    [nRows, nCols] = size(I);
    Res = I;

    for i = 3:nRows-3
        auxS = sum(I(i, 1:5));
        for j = 3:nCols-3
            auxS2 = sum(I(i-2:i+2, j));
            Res(i,j) = auxS+auxS2;
            auxS = auxS - I(i, j-2) + I(i, j+3);
        end

    end

end
```

1.1 *Imatge original*

1.2 *Imatge processada*

2. **Implementar un filtre no lineal que elimini el soroll del tipus 'sal i pebre' (píxels de soroll aïllats que són totalment blancs o negres).**

```
>> I = imread('Bird24b.bmp');
>> G = rgb2gray(I);
>> figure; imshow(G);

>> imN = imnoise(G,'salt & pepper', 0.2);
>> figure; imshow(imN);

>> J = Anti_sp(imN);
>> figure; imshow(J,[]);
```

```
function [ Res ] = Anti_sp( I )

[nRows, nCols] = size(I);
Res = I;
    for i = 3:nRows-2
        for j = 3:nCols-2
            if I(i,j) == 0 | I(i,j) == 255
                window = [I(i-1,j-1),I(i-1,j),I(i,j+1),I(i,j-1),
                                        I(i,j),I(i,j+1),I(i+1,j-1),
                                        I(i+1,j),I(i+1,j+1)];
                Res(i,j) = median(window);
            end
        end

    end

end
```

1.1 *Imatge original*

2.2 *Imatge amb soroll*

2.3 *Imatge processada*

3. Implementar una funció que realci els contorns (els contorns queden marcats però no es perden els nivells de grisos).

```
>> I = imread('airplane.tif');
>> G = rgb2gray(I);

>> H = [-1, -2, -1; 0, 0, 0; 1, 2, 1];
>> GX = imfilter(G, H);
>> GY = imfilter(G, H');

>> C = abs(GX) + abs(GY);
>> Res = G + C;

>> figure; imshow(Res, []);
```

3.1 *Imatge original*

3.2 *Imatge processada*

4. **Implementar una funció que esborroni (*motion blur*) una imatge en una direcció (un angle passat per paràmetre). El codi d'aquesta funció ha de ser un codi propi.**

```
function[res] = myMotion(I,n,angle)
%n := length of the motion; angle := angle of the motion
H = 1/2 * [0,0,0;1,1,0;0,0,0];
H = imrotate(H, angle);

Res = I;
J = imfilter(Res, H);

for i = 1:n
      Res = Res + J;
      J = imfilter(J,H);
end

end
```

```
>> I = imread('Bird24b.bmp');
>> G = rgb2gray(I);
>> imshow(G);

>> G = double(G);

>> Res = myMotion(G,40,135);
>> figure;imshow(Res,[]);
```

4.1 *Imatge original*

4.2 *Imatge processada*

5. **[opcional] Implementar una funció que donada una imatge esborronada utilitzant la funció implementada anteriorment recuperi (en major o menor mesura) la imatge original.**

És molt difícil obtenir una imatge nítida a partir d'una imatge esborronada, a menys que sapiguem de quina manera ha sigut distorsionada.

En aquest cas únicament caldria aplicar algun algoritme de *deblurring* indicant-li els paràmetres que hem fet servir per distorsionar la imatge prèviament.

Un algoritme conegut és "**el filtre de Wiener**", que ja ve implementat a Matlab i el podem executar amb: J = deconvwnr(I,psf,nsr);

- **I**: la imatge esborronada.
- **psf**: la funció de difusió per punts.
- **nsr**: ratio de soroll-a-senyal del soroll additiu.
  - **J**: Imatge retornada.

En cas contrari, si desconeixem el procediment d'esborronament de la imatge és pràcticament impossible recuperar una imatge nítida. Tot i així existeixen algoritmes com el "***Blind deconvolution algorithm***". Aquest algoritme també es troba implementat a Matlab, i s'executa amb:
[J, psfr] = deconvblind(I,psfi);

- **I**: la imatge esborronada.
- **psfi**: l'estimador inicial de la funció **psf**.
  - **J**: Imatge retornada.
  - **psfr**: funció psf perfeccionada.

L'algoritme maximitza la probabilitat de que la imatge resultant, quan es converteixi amb el **psfr** resultant, sigui una instància de la imatge borrosa, assumint una distribució *Poisson* del soroll.
Es a dir, l'algoritme va restaurant la imatge a mesura que perfecciona la funció **psf**, usant un procés iteratiu similar a un altre algoritme conegut, "L'algoritme de **Lucy-Richardson** accelerat".