Lect10

## linear spatial filtering Implementation using Matlab :

the toolbox implement linear spatial filtering using function imfilter, with the following syntax :

g = imfilter (f, w, fitering-mode, boundary-option, size-option);

this function fitters the multidimensional array f (image) with the multidimensional filter w (filter mask), the result g (filtered image) has the same size and class as f.

Each element of the output g is computed using double-precision floating point.

if f is an integer array, then output element that exceed the range of the integer type are truncated, and fractional value are rounded,

- filtering-mode :

| option | Description |
|---|---|
| 'Corr' | imfilter performs multidemensional filtering using correlation ( detault option) |
| 'Conv' | Filtering is done using convolution. |

- Boundary-option :

| option | Description |
|---|---|
| X | input array values outside the bounds of the array are implicitly assumed to have the value X, when no boundary option is specified, imfilter uses X = 0 (zero padding) |
| 'symmetric' | input array values outside the bounds of the array are computed by mirror-reflecting the array across the array (image) border. |
| 'replicate' | the size of the image is extended by replicating the values in its outer border |
| 'Circular' | the size of the image is extended by treating the image as one period a 2-D periodic function. |

- Size options:

- 'same' the output image is the same size as the input image, this is the default.

'full' the output image is the full filtered result.

the most common syntax for imfilter is:

```
g = imfilter(f, w, 'replicate')
```

This syntax is used when implementing IPT standard linear spatial filters.

Examples: (example 1)

1) Read a color image into the workspace and view it

```
>> rgb = imread('peppers.png');
>> imview(rgb);
```

2) Create a filter, h, that can be used to approximate linear camera motion:

```
>> h = fspecial('motion', 50, 45);
```

3) Apply the filter, using imfilter, to the image rgb to Create a new image, rgb2:

```
>> rgb2 = imfilter(rgb, h);
>> imview(rgb2); >> whos
```

4) Specifies the replicate boundary option:

```
>> brrgb = imfilter(rgb, h, 'replicate');
>> imview(brrgb)
```

(example 2): "Averaging filters"

```
>> I = imread('coins.png');
>> h = ones(5,5)/25;
>> I2 = imfilter(I, h);
>> imshow(I), title('original Image');
>> figure, imshow(I2), title('Filtered Image')
```

example 3: Zero padding of outside pixels:

Zero padding can result in a dark band around the edge of the image. as shown in the next example:

```
>> I = imread('eight.tif');
>> h = ones(5,5)/25;
>> I2 = imfilter(I,h);
>> imshow(I), title('original image');
>> figure, imshow(I2), title('Filtered Image with Black Border')
```

To eleminate the zero-padding artifacts around the edge of the image, we can use boundary padding called border replicaton

example 4:

```
>> I3 = imfilter(I,h,'replicate');
>> figure, imshow(I3);
>> title('Filtered Image with Border Replication').
```

filtering Using Convolution :

when we working with filters using convolution method, we have two options to do that :

1) using the following syntax :

```
g = imfilter(f, w, 'conv', 'replicate')
```

2) using rot90(w,2) to rotate the window 180° (preprocessing) and then use imfilter(f,w,'replicate'),

- rot90(w,k): rotates w by k * 90 degrees, where k is an integer.

example 5: (script file)

```
f1 = zeros(256,256);
f2 = ones(256,256);
f3 = ones(256,256);
f4 = zeros(256,256);
f = [f1, f2; f3, f4];
imshow(f)
% two symmetric filters
w = ones(31); % Accumulator
w1 = ones(31,31)/(31^2) % Average filter
```

```
% default boundary option, padding the border of
% the image with zeros, the edge between black & white
% in the filtered image are blurred.
    g = imfilter(f,w);
    figure(2), imshow(g,[]);
% to solve this problem use
% 1. use boundary option 'replicate'
    g1 = imfilter(f,w,'replicate');
    figure(3), imshow(g1,[]);
% 2. use boundary option 'symmetric'
    g2 = imfilter(f,w,'symmetric');
    figure(4), imshow(g2,[]);
% using boundary options 'Circular'
% the edge between black & white in the filtered image
% are blurred, the same problem as zero padding
    g3 = imfilter(f,w,'circular');
    figure(5), imshow(g3,[]);
% scaling effect of using imfilter with an image of
% class uint8, clipping caused some data loss
% Coefficients of the mask did not sum the the range
% [0,1], resulting in filtered values outside the
% [0 255] range.
% To avoid this problem, we have the option of
% normalizing the coefficients by (31)^2, so the
% sum would be 1, or inputting the data in
% double format.
    f8 = im2uint8(f);
    g8 = imfilter(f8,w,'replicate');
    figure(6), imshow(g8,[]);
```

# Image processing Toolbox Standard Spatial Filters:

## • linear spatial Filters

To generate 2-D linear spatial filters masks, w, we use fspecial functions with the following syntax:

w = fspecial('type', parameters).

type: the filter type, parameters: applicable parameters for each filter.

special filters supported by function fspecial:

1) fspecial('average', [r c]).

generates a rectangular averaging filter of size r×c, the default is 3×3, A single number instead of [r c] Specifies a square filter.

example:

w = fspecial('average', [3 3]);

$$w = \begin{bmatrix} \frac{1}{9} & \frac{1}{9} & \frac{1}{9} \\ \frac{1}{9} & \frac{1}{9} & \frac{1}{9} \\ \frac{1}{9} & \frac{1}{9} & \frac{1}{9} \end{bmatrix} \rightarrow \text{According to the equation} \rightarrow R = \frac{1}{9} \cdot \sum_{i=1}^{9} z_i$$

So $w_i = \frac{1}{9}$

- this operation results in image smoothing. (Low pass filter)

2) fspecial('disk', r);

A Circular Averaging filter (within a square of size 2r+1) with radius r, the default r = 5;

3) fspecial('guassian', [r c], sig).

A Guassian Lowpass filter of size [r c] and standard diviation sig (positive), the default 3×3 and 0.5.

" In some application, we have a continuous function of two variables, and the objective is to obtain a spatial filter mask based on that function.

Gaussian function: $h(x,y) = e^{-\frac{x^2+y^2}{2\sigma^2}}$.

$\sigma$ - standard diviation.

To generate a 3×3 filter mask from this function, we sample it about its center.

Thus : $w_1 = h(-1,-1)$, $w_2 = h(-1,0), \ldots, w_9 = h(1,1)$.

An mxn filter mask is generated in a similar manner.

4) fspecial('laplacian', alpha) :

Generates A 3×3 laplacian filter whose shape is specified by alpha, a number in the range [0 1], the default value of alpha = 0.5.

The laplacian of an image $f(x,y)$, denoted $\nabla^2 f(x,y)$ is defined as

$$\nabla^2 f(x,y) = \frac{\partial^2 f(x,y)}{\partial x^2} + \frac{\partial^2 f(x,y)}{\partial y^2}.$$

Commonly used digital approximation of the second derivatives are : $\frac{\partial^2 f}{\partial x^2} = f(x+1,y) + f(x-1,y) - 2f(x,y)$ and

$$\frac{\partial^2 f}{\partial y^2} = f(x,y+1) + f(x,y-1) - 2f(x,y)$$

So that :

$$\nabla^2 f = [f(x+1,y) + f(x-1,y) + f(x,y+1) + f(x,y-1)] - 4f(x,y)$$

This expression can be implemented at all points (x,y) in an image by convolving the image with the following spatial mask : $w = \begin{bmatrix} 0 & 1 & 0 \\ 1 & -4 & 1 \\ 0 & 1 & 0 \end{bmatrix}$ or an alteranate definition of the digital second derivatives takes into account diagonal elements, and can be implemented using the mask : $\begin{bmatrix} 1 & 1 & 1 \\ 1 & -8 & 1 \\ 1 & 1 & 1 \end{bmatrix}$

Enhancement using the laplacian is based on the equation :

$$g(x,y) = f(x,y) + c \cdot [\nabla^2 f(x,y)]$$

Where $f(x,y)$ - input image, $g(x,y)$ - enhanced image,

$c = 1$ if the center of the mask is positive.

$c = -1$ " " " " " " is negative.

" because the laplacian is a derivative operator, it shapens the image"

function fspecial ("laplacian", alpha) implements a more general lapalacian mask :

$$\begin{bmatrix} \frac{\alpha}{1+\alpha} & \frac{1-\alpha}{1+\alpha} & \frac{\alpha}{1+\alpha} \\ \frac{1-\alpha}{1+\alpha} & \frac{-4}{1+\alpha} & \frac{1-\alpha}{1+\alpha} \\ \frac{\alpha}{1+\alpha} & \frac{1-\alpha}{1+\alpha} & \frac{\alpha}{1+\alpha} \end{bmatrix}$$

example1: enhancement (sharpening) the image:

```
% Generate and display the laplacian filter:
>> w = fspecial('laplacian', 0)

w =
    0.0000    1.0000    0.0000
    1.0000   -4.0000    1.0000
    0.0000    1.0000    0.0000

% We could just as easily have specified this shape
  manually as
>> w = [0 1 0; 1 -4 1; 0 1 0];
% apply the mask w to the input image f, which is of
% class uint8:
>> g1 = imfilter(f, w, 'replicate');
>> imshow(g1, [])      % negative center mask (negative
>> f2 = im2double(f);  % value are truncated, so we
>> g2 = imfilter(f2, w, 'replicate'); % must convert f to class double
>> imshow(g2, [])      % befor filtering it.
% g2 - is sharpening as should look laplacian image.
>> g = f2 - g2;
>> imshow(g);  the final result: sharper image.
```

example2: two laplacian formulations:

- supported by IPT with a -4 in the center.
- manually created filter mask with a -8 in the center.

- Compare the results obtained by two laplacian filters:

```
>> f = imread('moon.tif');          % Manually
>> w4 = fspecial('laplacian', 0);   % specifying filters
>> w8 = [1 1 1; 1 -8 1; 1 1 1];     % and comparing
>> f = im2double(f);                % techniques
>> g4 = f - imfilter(f, w4, 'replicate');
>> g8 = f - imfilter(f, w8, 'replicate');
>> imshow(f)
>> figure, imshow(g4), figure, imshow(g8)
```

5) fspecial('log', [r c], sig).

laplacian of a Guassian (LoG) filter of the size r x c and standard deviation sig (positive), the defaut 5x5 and 0.5.

6) fspecial('motion', len, theta):

approximates linear motion of a camera (with respect to the image) of len pixels. The direction of motion is theta, measured in degree, counterclockwise from the horizontal, the default len = 9, theta = 0.

7) fspecial('prewitt').

outputs a 3x3 Prewitt mask, wv, that approximates a vertical gradient. A mask for horizontal gradient is obtained by transposing the result: sh = sv'.

8) fspecial('Sobel').

outputs a 3x3 Sobel mask, sv, that approximates a vertical gradient. A mask for horizontal gradient is obtained by transposing the result: sh = sv'.

9) fspecial('unsharp', alpha).

Outputs a 3x3 unsharp filter. Parameter alpha controls the shape, $0 \le \text{alpha} \le 1$, the default = 0.2.

· Smoothing linear Filters: (Weighted Average Filter)

The mask:

$$\frac{1}{16} \times \begin{bmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{bmatrix} \qquad \frac{1}{9} \times \begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}$$

Weighted Average filter — Average filter

weighted Average filter: the pixel at the center of the mask is multipleted by a higher value than any other. the other pixels are inversely weighted as a function of their distance from the center of mask.

the general form for filtering an MxN image with a weighted filter of size mxn is

$$= \frac{\sum_{s=-a}^{a} \sum_{t=-b}^{b} w(s,t) \cdot f(x+s, y+t)}{\sum_{s=-a}^{a} \sum_{t=-b}^{b} w(s,t)}$$