Linear Spatial Filters

IPT supports a number of predefined 2-D linear spatial filters, obtained by using function *fspecial*, which generates a filter mask, **w**, using the syntax

```
h = fspecial(type)
h = fspecial(type, parameters)
```

where '**type**' specifies the filter type, and parameters further define the specified filter. The spatial filters supported by *fspecial* are summarized in Table 1, including applicable parameters for each filter.

h = fspecial('average', hsize) returns an averaging filter h of size hsize. The argument hsize can be a vector specifying the number of rows and columns in h, or it can be a scalar, in which case h is a square matrix. The default value for hsize is [3 3].

Ex. >> h = fspecial('average')

h =

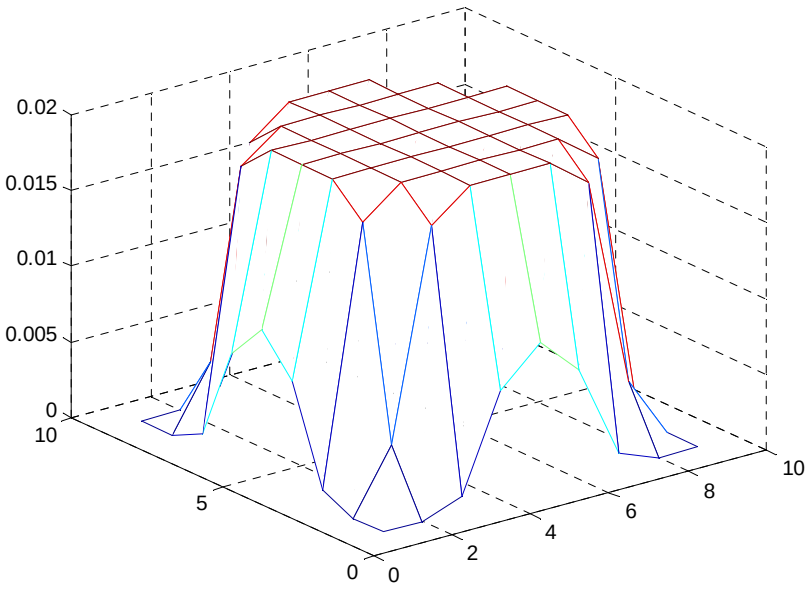
```
   0.1111   0.1111   0.1111
   0.1111   0.1111   0.1111
   0.1111   0.1111   0.1111
```

h = fspecial('disk', radius) returns a circular averaging filter (pillbox) within the square matrix of side 2*radius+1. The default radius is 5.

Ex. >> h = fspecial('disk', 4)

h =

```
         0         0    0.0010    0.0072    0.0097    0.0072    0.0010         0         0
         0    0.0041    0.0179    0.0199    0.0199    0.0199    0.0179    0.0041         0
    0.0010    0.0179    0.0199    0.0199    0.0199    0.0199    0.0199    0.0179    0.0010
    0.0072    0.0199    0.0199    0.0199    0.0199    0.0199    0.0199    0.0199    0.0072
    0.0097    0.0199    0.0199    0.0199    0.0199    0.0199    0.0199    0.0199    0.0097
    0.0072    0.0199    0.0199    0.0199    0.0199    0.0199    0.0199    0.0199    0.0072
    0.0010    0.0179    0.0199    0.0199    0.0199    0.0199    0.0199    0.0179    0.0010
         0    0.0041    0.0179    0.0199    0.0199    0.0199    0.0179    0.0041         0
         0         0    0.0010    0.0072    0.0097    0.0072    0.0010         0         0
```



h = fspecial('gaussian', hsize, sigma) returns a rotationally symmetric Gaussian lowpass filter of size hsize with standard deviation sigma (positive). hsize can be a vector specifying the number of rows and columns in h, or it can be a scalar, in which case h is a square matrix. The default value for hsize is [3 3]; the default value for sigma is 0.5.

Algorithm:

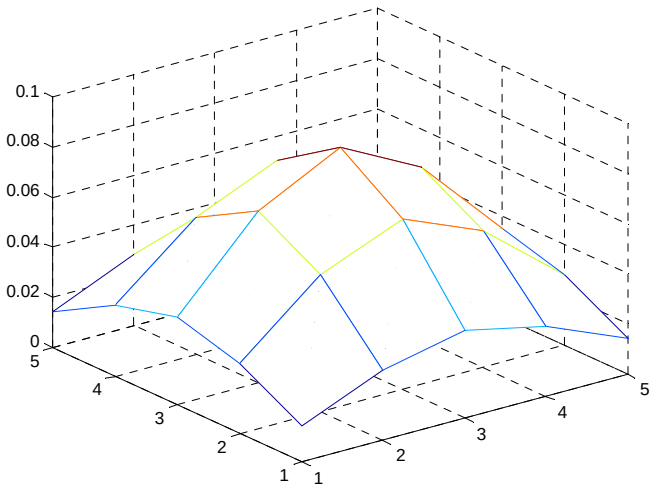
$$h_g(n_1, n_2) = e^{-\left(n_1^2 + n_2^2\right)/2\sigma^2}$$

$$h(n_1, n_2) = \frac{h_g(n_1, n_2)}{\sum_{n_1}\sum_{n_2} h_g}$$

Ex. >> h = fspecial('gaussian', 5, 1.5)

h =

0.0144	0.0281	0.0351	0.0281	0.0144
0.0281	0.0547	0.0683	0.0547	0.0281
0.0351	0.0683	0.0853	0.0683	0.0351
0.0281	0.0547	0.0683	0.0547	0.0281
0.0144	0.0281	0.0351	0.0281	0.0144



h = fspecial('laplacian', alpha) returns a 3-by-3 filter approximating the shape of the two-dimensional Laplacian operator. The parameter alpha controls the shape of the Laplacian and must be in the range 0.0 to 1.0. The default value for alpha is 0.2.

Algorithm:

$$\nabla^2 = \frac{\partial^2}{\partial x^2} + \frac{\partial^2}{\partial y^2}$$

$$\nabla^2 = \frac{4}{(\alpha+1)} \begin{bmatrix} \frac{\alpha}{4} & \frac{1-\alpha}{4} & \frac{\alpha}{4} \\ \frac{1-\alpha}{4} & -1 & \frac{1-\alpha}{4} \\ \frac{\alpha}{4} & \frac{1-\alpha}{4} & \frac{\alpha}{4} \end{bmatrix}$$

Ex. >> h = fspecial('laplacian', 0.5)

h =

0.3333	0.3333	0.3333
0.3333	-2.6667	0.3333
0.3333	0.3333	0.3333

h = fspecial('log', hsize, sigma) returns a rotationally symmetric Laplacian of Gaussian filter of size hsize with standard deviation sigma (positive). hsize can be a vector specifying the number of rows and columns in h, or it can be a scalar, in which case h is a square matrix. The default value for hsize is [5 5] and 0.5 for sigma.

Algorithm:

$$h_g(n_1, n_2) = e^{-\left(n_1^2 + n_2^2\right)/2\sigma^2}$$

$$h_g(n_1, n_2) = \frac{\left(n_1^2 + n_1^2 - 2\sigma^2\right) h_g(n_1, n_2)}{2\pi\sigma^6 \sum_{n_1} \sum_{n_2} h_g}$$

Ex. >> h = fspecial('log')

h =

0.0448	0.0468	0.0564	0.0468	0.0448
0.0468	0.3167	0.7146	0.3167	0.0468
0.0564	0.7146	-4.9048	0.7146	0.0564
0.0468	0.3167	0.7146	0.3167	0.0468
0.0448	0.0468	0.0564	0.0468	0.0448

h = fspecial('motion', len, theta) returns a filter to approximate, once convolved with an image, the linear motion of a camera by len pixels, with an angle of theta degrees in a counterclockwise direction. The filter becomes a vector for horizontal and vertical motions. The default len is 9 and the default theta is 0, which corresponds to a horizontal motion of nine pixels.

To compute the filter coefficients, h, for 'motion':

1. Construct an ideal line segment with the desired length and angle, centered at the center coefficient of h.
2. For each coefficient location (i,j), compute the nearest distance between that location and the ideal line segment.
3. h = max(1 - nearest_distance, 0);
4. Normalize h:h = h/(sum(h(:))

Ex. >> h = fspecial('motion', 3,45)

h =

0	0.0754	0.1883
0.0754	0.3215	0.0754
0.1883	0.0754	0

h = fspecial('prewitt') returns the 3-by-3 filter h (shown below) that emphasizes horizontal edges by approximating a vertical gradient. If you need to emphasize vertical edges, transpose the filter h'.
Ex. >> h = fspecial('prewitt')

h =

1	1	1
0	0	0
-1	-1	-1

To find vertical edges, or for x-derivatives, use h'.

h = fspecial('sobel') returns a 3-by-3 filter h (shown below) that emphasizes horizontal edges using the smoothing effect by approximating a vertical gradient. If you need to emphasize vertical edges, transpose the filter h'.
Ex. >> h = fspecial('sobel')

h =

```
    1    2    1
    0    0    0
   -1   -2   -1
```

h = fspecial('unsharp', alpha) returns a 3-by-3 unsharp contrast enhancement filter. fspecial creates the unsharp filter from the negative of the Laplacian filter with parameter alpha. alpha controls the shape of the Laplacian and must be in the range 0.0 to 1.0. The default value for alpha is 0.2.
Algorithm:

$$h = \frac{1}{\alpha+1}\begin{bmatrix} -\alpha & \alpha-1 & -\alpha \\ \alpha-1 & \alpha+5 & \alpha-1 \\ -\alpha & \alpha-1 & -\alpha \end{bmatrix}$$

Note Do not be confused by the name of this filter: an unsharp filter is an operator used to sharpen images. The name comes from a publishing industry process in which an image is sharpened by subtracting a blurred (unsharp) version of the image from itself.

Examples

```
I = imread('cameraman.tif');
subplot(2,2,1);
imshow(I); title('Original Image');

H = fspecial('motion',20,45);
MotionBlur = imfilter(I,H,'replicate');
subplot(2,2,2);
imshow(MotionBlur);title('Motion Blurred Image');

H = fspecial('disk',10);
blurred = imfilter(I,H,'replicate');
subplot(2,2,3);
imshow(blurred); title('Blurred Image');

H = fspecial('unsharp');
sharpened = imfilter(I,H,'replicate');
subplot(2,2,4);
imshow(sharpened); title('Sharpened Image');
```

จากตัวกรองสัญญาณที่สร้างจากฟังก์ชัน fspecial ซึ่งมีอยู่หลายรูปแบบดังที่ได้อธิบายไปแล้ว ให้สร้างตัวกรองแต่ละแบบ แล้วนำมาดำเนินการกับข้อมูลภาพ 'cameraman.tif' ที่เก็บไว้ในตัวแปร f และให้ใช้ฟังก์ชัน imfilter ที่มีรูปแบบการใช้ดังนี้

Syntax

```
B = imfilter(f, h)
```

B = imfilter(f, h, option1, option2,...)

Description

B = imfilter(A, H) filters the multidimensional array A with the multidimensional filter H. The array A can be logical or a nonsparse numeric array of any class and dimension. The result B has the same size and class as A.

Each element of the output B is computed using double-precision floating point. If A is an integer or logical array, then output elements that exceed the range of the integer type are truncated, and fractional values are rounded.

ให้อภิปรายผลลัพธ์ที่ได้จากการกรองข้อมูลภาพด้วยตัวกรองแต่ละชนิด