University of Diyala
Computer Science Department
Image Processing
3rd Class
Lecturer: Dr. Jumana Waleed Salih

Image Processing

معالجة صور

6 th lecture

Spatial Filtering

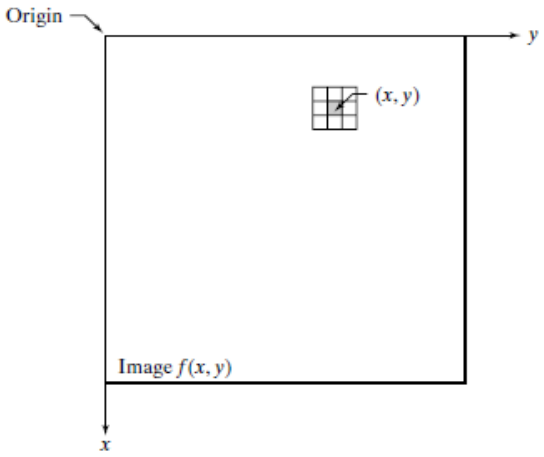
Removing or intensifying some components in an image by using enhancement processing is called filtering. Using spatial masks to process an image is called spatial filtering

Background

Spatial domain techniques operate directly on the pixels of an image. The spatial domain processes are denoted by the expression:

$$g(x,y) = T\,[f(x,y)]$$

Where f(x,y) is the input image, g(x,y) is the output (processed) image, and T is an operator on defined over a specified neighborhood about point (x,y). The principal approach for defining spatial neighborhoods about a point is to use a square or rectangular region centered as shown in the figure below. The center of the region is moved from pixel to pixel starting, say, at the top, left corner, and, as it moves, it encompasses different neighborhoods.



neighborhood of size 3 × 3 about a point (x, y) in an image.

Neighborhood processing consists of:

(1) Defining a center point,

(2) Performing an operation that involves only the pixels in a predefined neighborhood about that center point;

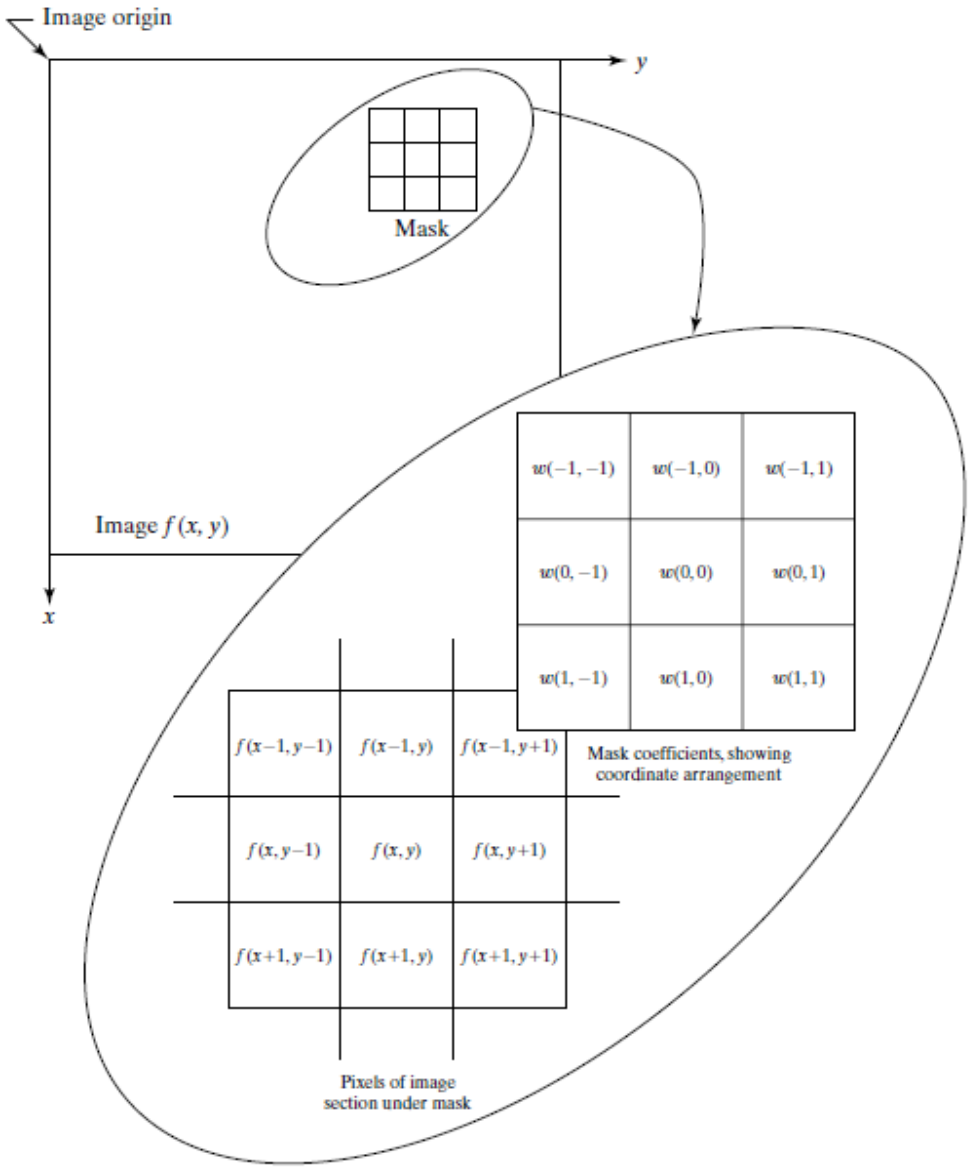
(3) Letting the result of that operation be the "response" of the process at that point;

(4) Repeating the process for every point in the image.

The process of moving the center point creates new neighborhoods, one for each pixel in the input image. The two principal terms used to identify this operation are neighborhood processing and spatial filtering.

Linear Spatial Filtering

The linear operations consist of multiplying each pixel in the neighborhood by a corresponding coefficient and summing the results to obtain the response at each point (x,y). If the neighborhood is of size $m \times n$, mn coefficients are required. The coefficients are arranged as a matrix, called a filter, mask, filter mask, kernel, template, or window. The mechanics of linear spatial filtering are illustrated in this figure.



mechanics of linear spatial filtering. The magnified drawing shows a 3×3 mask and the corresponding image neighborhood directly under it. The neighborhood is shown displaced out from under the mask for ease of readability.

The process consists simply of moving the center of the filter *w* mask from point to point in an image, *f*. At each point the response of the filter at that point (x, y) is the sum of products of the filter coefficients and the corresponding neighborhood pixels in the area spanned by the filter mask.

There are two closely related concepts that must be understood clearly when performing linear spatial filtering. One is correlation; the other is convolution. Correlation is the process of passing the mask *w* by the image array *f* in the manner described in the previous figure. Mechanically, convolution is the same process, except that *w* is rotated by 180° prior to passing it by *f*.

Example:

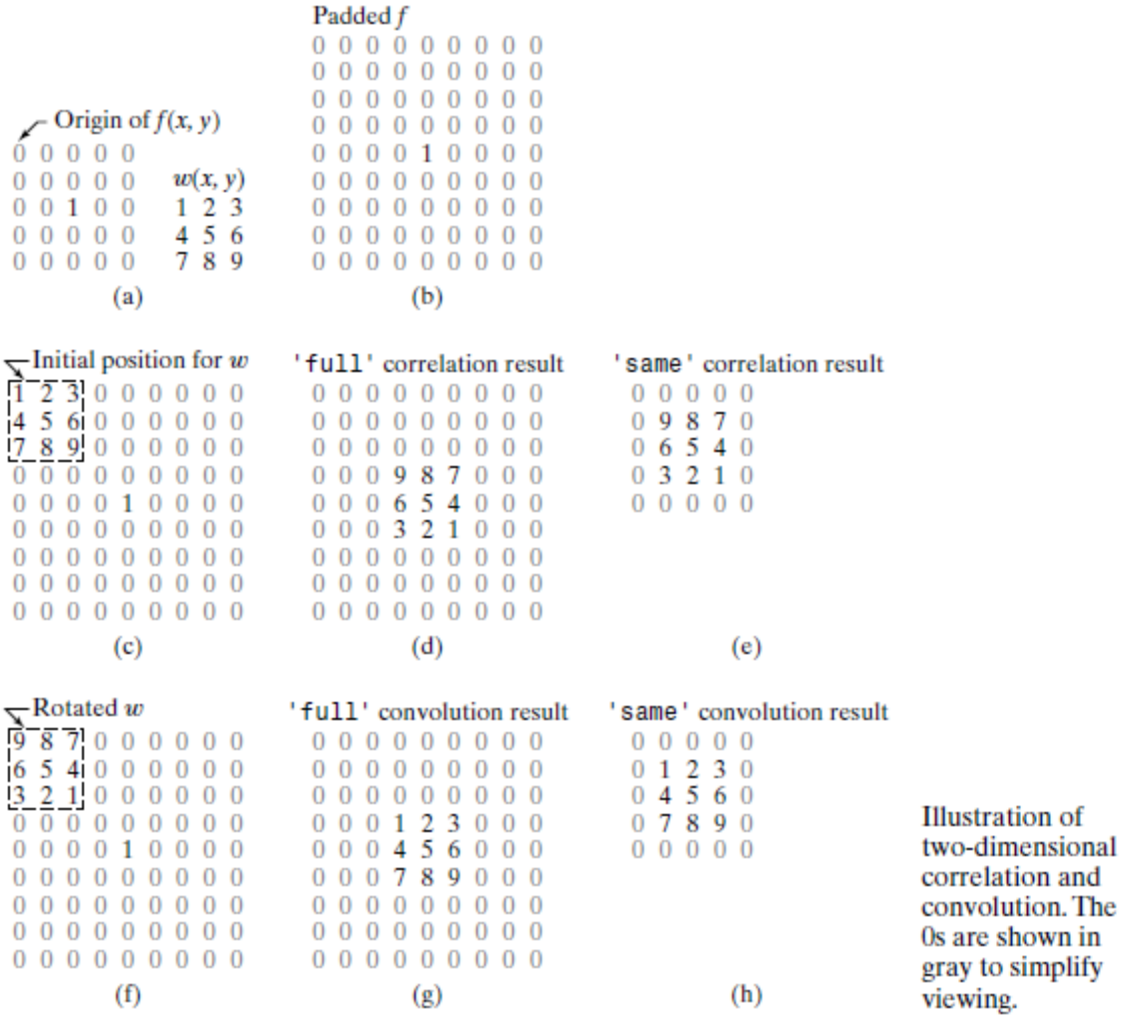


Illustration of two-dimensional correlation and convolution. The 0s are shown in gray to simplify viewing.

The toolbox implements linear spatial filtering using function imfilter, which has the following syntax:

g = imfilter(f, w, filtering_mode, boundary_options, size_options)

Where f is the input image, w is the filter mask, g is the filtered result, and the other parameters are summarized in the next table.

Options	Description
Filtering Mode	
`'corr'`	Filtering is done using correlation (see Figs. 3.13 and 3.14). This is the default.
`'conv'`	Filtering is done using convolution (see Figs. 3.13 and 3.14).
Boundary Options	
`P`	The boundaries of the input image are extended by padding with a value, `P` (written without quotes). This is the default, with value 0.
`'replicate'`	The size of the image is extended by replicating the values in its outer border.
`'symmetric'`	The size of the image is extended by mirror-reflecting it across its border.
`'circular'`	The size of the image is extended by treating the image as one period a 2-D periodic function.
Size Options	
`'full'`	The output is of the same size as the extended (padded) image (see Figs. 3.13 and 3.14).
`'same'`	The output is of the same size as the input. This is achieved by limiting the excursions of the center of the filter mask to points contained in the original image (see Figs. 3.13 and 3.14). This is the default.

Example:

```
>> f=[0 0 0 0 0; 0 0 0 0 0; 0 0 1 0 0; 0 0 0 0 0; 0 0 0 0 0];

>> w=[1 2 3; 4 5 6; 7 8 9];

>> g=imfilter(f,w,'corr','replicate','full')

g =

     0     0     0     0     0     0     0
     0     0     0     0     0     0     0
     0     0     9     8     7     0     0
     0     0     6     5     4     0     0
     0     0     3     2     1     0     0
     0     0     0     0     0     0     0
     0     0     0     0     0     0     0
```

***Smoothing Spatial Filters**

Smoothing Filters are used for blurring and for noise reduction.

Averaging filters

It is called smooth processing or low-pass filters in frequency domain, which is used for blurring and for noise reduction.

We can have all sizes of filters,

$$\frac{1}{9}\times\begin{bmatrix}1&1&1\\1&1&1\\1&1&1\end{bmatrix} \qquad \frac{1}{25}\times\begin{bmatrix}1&1&1&1&1\\1&1&1&1&1\\1&1&1&1&1\\1&1&1&1&1\\1&1&1&1&1\end{bmatrix}$$

***Order Statistics Nonlinear Filters**

Median filters

It replaces the value of a pixel by the median of the intensity values in the neighborhood of that pixel. Median filters are quite popular because, for certain types of random noise (salt and pepper noise), they provide excellent noise reduction capabilities, with less blurring than linear Smoothing Filtering.

***Sharpening filters**

It is called high-pass filtering in frequency domain, which is used for highlighting fine detail or enhancing detail that has been blurred.

$$\frac{1}{9}\times\begin{bmatrix}-1&-1&-1\\-1&w&-1\\-1&-1&-1\end{bmatrix} \qquad \frac{1}{25}\times\begin{bmatrix}-1&-1&-1&-1&-1\\-1&-1&-1&-1&-1\\-1&-1&w&-1&-1\\-1&-1&-1&-1&-1\\-1&-1&-1&-1&-1\end{bmatrix}$$

Image Processing Toolbox Standard Spatial Filters

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

w = 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 the following:

Type	Syntax and Parameters
`'average'`	`fspecial('average', [r c])`. A rectangular averaging filter of size `r` × `c`. The default is 3 × 3. A single number instead of `[r c]` specifies a square filter.
`'disk'`	`fspecial('disk', r)`. A circular averaging filter (within a square of size `2r + 1`) with radius `r`. The default radius is 5.
`'gaussian'`	`fspecial('gaussian', [r c], sig)`. A Gaussian lowpass filter of size `r` × `c` and standard deviation `sig` (positive). The defaults are 3 × 3 and 0.5. A single number instead of `[r c]` specifies a square filter.
`'laplacian'`	`fspecial('laplacian', alpha)`. A 3 × 3 Laplacian filter whose shape is specified by `alpha`, a number in the range [0, 1]. The default value for `alpha` is 0.5.
`'log'`	`fspecial('log', [r c], sig)`. Laplacian of a Gaussian (LoG) filter of size `r` × `c` and standard deviation `sig` (positive). The defaults are 5 × 5 and 0.5. A single number instead of `[r c]` specifies a square filter.
`'motion'`	`fspecial('motion', len, theta)`. Outputs a filter that, when convolved with an image, approximates linear motion (of a camera with respect to the image) of `len` pixels. The direction of motion is `theta`, measured in degrees, counterclockwise from the horizontal. The defaults are 9 and 0, which represents a motion of 9 pixels in the horizontal direction.
`'prewitt'`	`fspecial('prewitt')`. Outputs a 3 × 3 Prewitt mask, `wv`, that approximates a vertical gradient. A mask for the horizontal gradient is obtained by transposing the result: `wh = wv'`.
`'sobel'`	`fspecial('sobel')`. Outputs a 3 × 3 Sobel mask, `sv`, that approximates a vertical gradient. A mask for the horizontal gradient is obtained by transposing the result: `sh = sv'`.
`'unsharp'`	`fspecial('unsharp', alpha)`. Outputs a 3 × 3 unsharp filter. Parameter `alpha` controls the shape; it must be greater than 0 and less than or equal to 1.0; the default is 0.2.

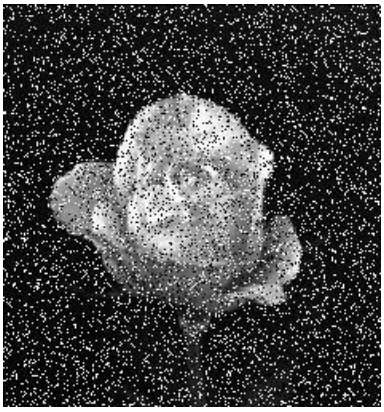
Because of its practical importance, the toolbox provides a specialized implementation of the 2-D median filter:

g = medfilt2(f)

Example:

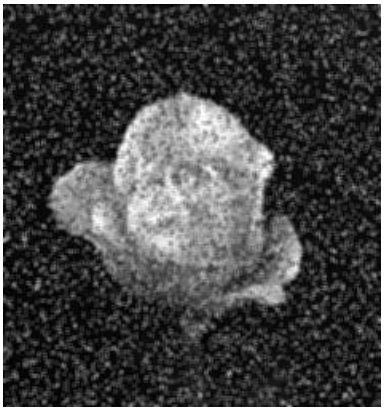
```
>> f=imread('E:\rose.jpg');
>> fn = imnoise(f, 'salt & pepper', 0.2);
>> imshow(fn);
>> g = medfilt2(fn);
>> figure; imshow(g);
>> w=fspecial('average', [3 3])          or          >> w=fspecial('average')
w =
    0.1111    0.1111    0.1111
    0.1111    0.1111    0.1111
    0.1111    0.1111    0.1111
>> g=imfilter(fn,w,'replicate');
>> figure; imshow(g);
```

Rose Image



Noisy Image

Median Filtering



Averaging Filtering