Image Processing and Analysis
Lecture 4、Image Enhencement in Spatial Filtering (II)

Weiqiang Wang
School of Computer Science and Technology, UCAS

September 28, 2023

Outline

1. Histogram Processing and Function Plotting
2. Spatial Filtering
3. Standard Spatial Filters in Image Processing Toolbox

Outline

1. Histogram Processing and Function Plotting
2. Spatial Filtering
3. Standard Spatial Filters in Image Processing Toolbox

Outline

1. Histogram Processing and Function Plotting
2. Spatial Filtering
3. Standard Spatial Filters in Image Processing Toolbox

Histogram Matching

- Histogram matching is similar to histogram equalization, except that instead of trying to make the output image have a flat histogram, we would like it to have a histogram of a specified shape.
- Consider for a moment continuous levels that are normalized to the interval [0, 1], and let r and z denote the intensity levels of the input and output images. The input levels have probability density function $p_r(r)$ and the output levels have the specified probability density function $p_z(z)$.

Histogram Matching

- We know from the discussion in the previous section that the transformation:

$$s = T(r) = \int_0^r p_r(w)dw$$

result in a ideal equalized histogram $p_s(s)$.
- Suppose now we define a variable z with the property

$$H(z) = \int_0^z p_z(w)dw = s$$

- From the preceding two equations, it follows that

$$z = H^{-1}(s) = H^{-1}(T(r));$$

- We can find $T(r)$ from the input image (this is the histogram equalization transformation discussed in the previous section), so it follows that we can use the preceding equation to find the transformed levels z whose PDF is the specified $p_z(z)$ as long as we can find H^{-1}.

Histogram Matching

- The toolbox implements histogram matching using the following syntax in histeq:

  g = histeq(f, hspec)

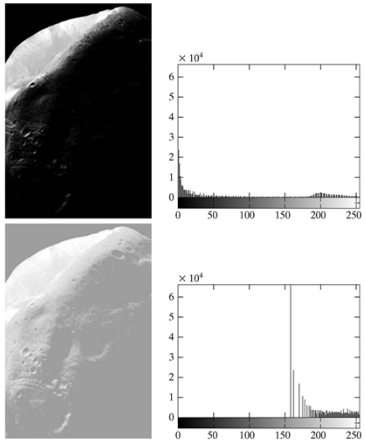
  - where f is the input image, hspec is the specified histogram (a row vector of specified values), and g is the output image, whose histogram approximates the specified histogram, hspec.
- This vector should contain integer counts corresponding to equally spaced bins. A property of histeq is that the histogram of g generally better matches hspec when length (hspec) is much smaller than the number of intensity levels in f.

Histogram Matching

- f=imread('Fig0310(a)(Moon Phobos).tif');
- f1=histeq(f,256);
- imshow(f1);

- It shows that histogram equalization in fact did not produce a particularly good result in this case.
- The reason for this can be seen by studying the histogram of the equalized image.

Histogram Matching



a b
c d

FIGURE 3.10 (a) Image of the Mars moon Phobos. (b) Histogram. (c) Histogram-equalized image. (d) Histogram of (c). (Original image courtesy of NASA).

Histogram Matching

- One possibility for remedying this situation is to use histogram matching, with the desired histogram having a lesser concentration of components in the low end of the gray scale, and maintaining the general shape of the histogram of the original image.
- We note that the histogram of original image is basically bimodal, with one large mode at the origin, and another, smaller, mode at the high end of the gray scale. These types of histograms can be modeled, for example, by using multimodal Gaussian functions.

Histogram Matching

- The following M-function computes a bimodal Gaussian function normalized to unit area, so it can be used as a specified histogram.
  - Function twomodegauss:

$$p(x) = k + \frac{A_1}{\sqrt{2\pi}\sigma_1}\exp\left(-\frac{(x-m_1)^2}{2\sigma_1^2}\right) + \frac{A_2}{\sqrt{2\pi}\sigma_2}\exp\left(-\frac{(x-m_2)^2}{2\sigma_2^2}\right)$$

- The following interactive function accepts inputs from a keyboard and plots the resulting Gaussian function. Refer to Section 2.10.5 for an explanation of the functions input and str2num. Note how the limits of the plots are set.
  - Function manualhist

Histogram Matching

- Since the problem with histogram equalization in this example is due primarily to a large concentration of pixels in the original image with levels near 0,a reasonable approach is to modify the histogram of that image so that it does not have this property.
- Figure 3.11(a) shows a plot of a function that preserves the general shape of the original histogram, but has a smoother transition of levels in the dark region of the intensity scale. The output of the program, p, consists of 256 equally spaced points from this function and is the desired specified histogram.

Histogram Matching

- An image with the specified histogram was generated using the command

  ```
  >> g = histeq(f, p);
  ```

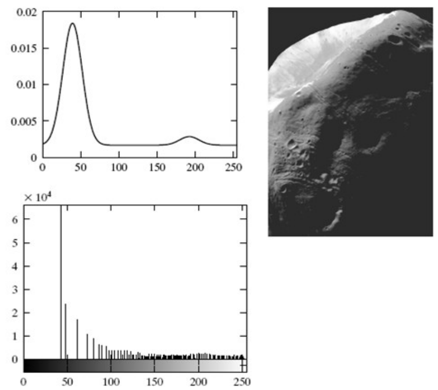
  all commands are:

  ```
  >> f=imread('Fig0310(a)(Moon Phobos).tif');
  >> g=histeq(f,manualhist);
  Enter m1, sig1, m2, sig2, A1, A2, k OR x to quit:x;
  >> imshow(g);
  ```


Histogram Matching

a b
c

FIGURE 3.11 (a) Specified histogram. (b) Result of enhancement by histogram matching. (c) Histogram of (b).



Spatial Filtering

- In spatial filtering (vs. frequency domain filtering), the output image is computed directly by simple calculations on the pixels of the input image.
- Spatial filtering can be either linear or non-linear.
- For each output pixel, some neighborhood of input pixels is used in the computation.

Linear Spatial Filtering

- In general, linear filtering of an image f of size $M \times N$ with a filter mask of size $m \times n$ is given by

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

  where $a = (m-1)/2$ and $b = (n-1)/2$
- This concept called convolution. Filter masks are sometimes called convolution masks or convolution kernels.

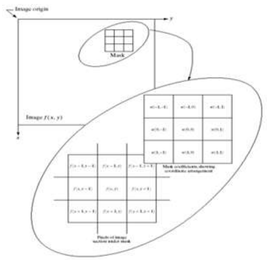
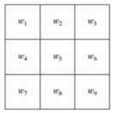


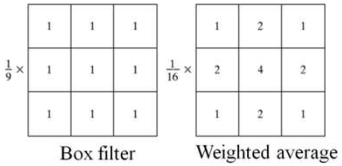
FIGURE 3.32 The mechanics of spatial filtering. The magnified drawing shows a 3 × 3 mask and the image section directly under it; the image section is shown displaced out from under the mask for ease of readability.

FIGURE 3.33 Another representation of a general 3 × 3 spatial filter mask.



Linear Spatial Filtering(cont.)

- Smoothing linear filters
  - Averaging filters (Lowpass filters in Chapter 4))
    - Box filter
    - Weighted average filter



a b

FIGURE 3.34 Two 3×3 smoothing (averaging) filter masks. The constant multipli er in front of each mask is equal to the sum of the values of its coefficients, as is required to compute an average.

Linear Spatial Filtering(cont.)

- 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 ω by the image array f in the manner described in Fig. 3.32.
- Mechanically, convolution is the same process, except that ω is rotated by 180 degree prior to passing it by f.
- These two concepts are best explained by some simple examples.

Linear Spatial Filtering(cont.)

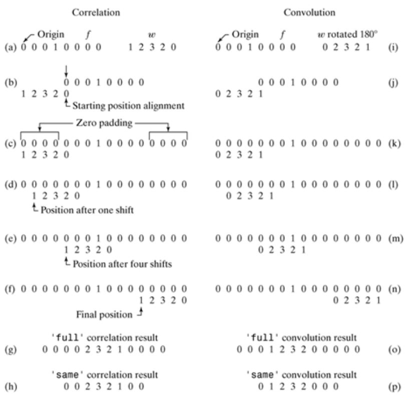


FIGURE 3.13 Illustration of one-dimensional correlation and convolution.

Linear Spatial Filtering(cont.)

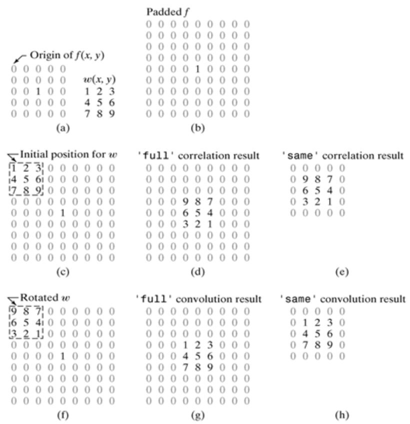


FIGURE 3.14 Illustration of two-dimensional correlation and convolution. The 0s are shown in gray to simplify viewing.

Linear Spatial Filtering(cont.)

- 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 Table 3.2.

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.

Linear Spatial Filtering(cont.)

- Each element of the filtered image is computed using double-precision, floating-point arithmetic. However, imfilter converts the output image to the same class of the input.Therefore, if input is an integer array, then output elements that exceed the range of the integer type are truncated, and fractional values are rounded. If more precision is desired in the result, then should be converted to class double by using im2double or double before using imfilter.
  - f=imread('Fig0315(a)(original_test_pattern).tif');
  - f=im2double(f);
  - w=ones(31);
  - gd = imfilter(f,w);
  - imshow(gd,[])

Linear Spatial Filtering(cont.)

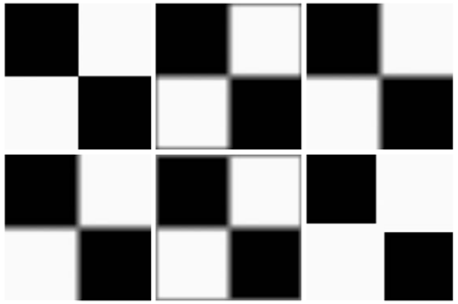
- >>gr = imfilter(f, w, 'replicate');
- >>figure, imshow(gr, [ ])
- >>gc = imfilter(f, w, 'circular');
- >>figure, imshow(gc, [ ])

Please help me choose the corresponding output!!

Linear Spatial Filtering(cont.)

- f=imread('Fig0315(a)(original_test_pattern).tif');
- w=ones(31);
- gd = imfilter(f,w);
- imshow(gd,[])

Linear Spatial Filtering(cont.)



a b c
d e f

FIGURE 3.15 (a) Original image. (b) Result of using `imfilter` with default zero padding. (c) Result with the `'replicate'` option. (d) Result with the `'symmetric'` option. (e) Result with the `'circular'` option. (f) Result of converting the original image to class `uint8` and then filtering with the `'replicate'` option. A filter of size 31 × 31 with all 1s was used throughout.

Nonlinear Spatial Filtering

- Nonlinear spatial filtering usually uses a neighborhood too, but some other mathematical operations are used. For example, letting the response at each center point be equal to the maximum pixel value in its neighborhood is a nonlinear filtering operation.
- Another basic difference is that the concept of a mask is not as prevalent in nonlinear processing.
- The toolbox provides two functions for performing general nonlinear filtering: nlfilter and colfilt.
- The former performs operations directly in 2-D. use the command open nlfilter to see the source code.
- colfilt organizes the data in the form of columns. requires more memory, but executes significantly faster than nlfilter.

Nonlinear Spatial Filtering(cont.)

- Given an input image, f, of size $M \times N$ and a neighborhood of size $m \times n$, function colfilt generates a matrix, call it A, of maximum size $mn \times MN$
- each column corresponds to the pixels encompassed by the neighborhood centered at a location in the image.
- For example, the first column corresponds to the pixels encompassed by the neighborhood when its center is located at the top, leftmost point in f.
- The former performs operations directly in 2-D. use the command open nlfilter to see the source code.
- colfilt organizes the data in the form of columns. requires more memory, but executes significantly faster than nlfilter.

Nonlinear Spatial Filtering(cont.)

- The syntax of function colfilt is

  g = colfilt(f, [m n], 'sliding', @fun, parameters)

  - where, m and n are the dimensions of the filter region
  - 'sliding' indicates that the process is one of sliding the region from pixel to pixel in the input image f
  - @fun references a function, which we denote arbitrarily as fun
  - parameters indicates parameters (separated by commas) that may be required by function fun.
- Because of the way in which matrix A is organized, function fun must operate on each of the columns of A individually and return a row vector, g, containing the results for all the columns.
- The kth element of g is the result of the operation performed by fun on the kth column of A.

Nonlinear Spatial Filtering(cont.)

- When using colfilt, the input image must be padded explicitly before filtering. For this we use function padarray, which, for 2-D functions, has the syntax

  fp = padarray(f, [r c], method, direction)

  where f is the input image, fp is the padded image, [r c] gives the number of rows and columns by which to pad f, and method and direction are as explained in Table 3.3.

TABLE 3.3 Options for function `padarray`.

Options	Description
Method	
`'symmetric'`	The size of the image is extended by mirror-reflecting it across its 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 of a 2-D periodic function.
Direction	
`'pre'`	Pad before the first element of each dimension.
`'post'`	Pad after the last element of each dimension.
`'both'`	Pad before the first element and after the last element of each dimension. This is the default.

Linear Spatial Filtering

- 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.

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.

Sharpening Spatial Filters

- The principal objective of sharping is to highlight fine detail in an image or to enhance detail that has been blurred.
- sharping is generally accomplished by spatial differentiation.
- The derivatives of a digital function are defined in terms of differences.
- For a first derivative, we require that it must satisfy:
  1. must be zero in flat segments(areas of constant gray-level values)
  2. must be nonzero at the onset of a gray-level step or ramp
  3. must be nonzero along ramps
- For a second derivative, we require that it must satisfy:
  1. must be zero in flat segments(areas of constant gray-level values).
  2. must be nonzero at the onset and end of a gray-level step or ramp.
  3. must be zero along ramps of constant slope.

Sharpening Spatial Filters(cont.)

- A basic definition of the first-order derivative of a one-dimensional function $f(x)$ is the difference

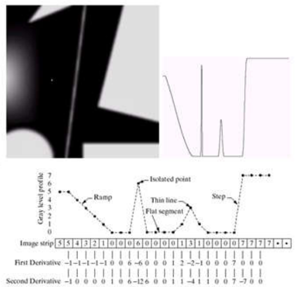
$$\frac{\partial f}{\partial x} = f(x+1) - f(x)$$

- Similarly, we define the second derivative as

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

a b
c

FIGURE 3.38 (a) A simple image. (b) 1-D horizontal gray-level profile along the center of the image and including the isolated noise point. (c) Simplified profile (the points are joined by dashed lines to simplify interpretation).



Laplacian Filter

- Laplacian function is defined as

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

- Function fspecial('laplacian', alpha) implements a more general Laplacian mask:

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

- We now proceed to enhance the image in Fig. 3.16(a) using the Laplacian. This image is a mildly blurred image of the North Pole of the moon. Enhancement in this case consists of sharpening the image, while preserving as much of its gray tonality as possible.

Image Enhancement with Laplacian Filters

- $g(x,y) = f(x,y) - \nabla^2 f(x,y)$

a b
c d

FIGURE 3.16 (a) Image of the North Pole of the moon. (b) Laplacian filtered image, using `uint8` formats. (c) Laplacian filtered image obtained using `double` formats. (d) Enhanced result, obtained by subtracting (c) from (a). (Original image courtesy of NASA.)

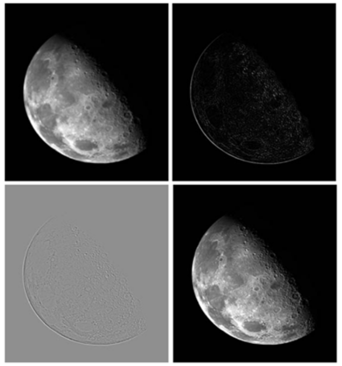
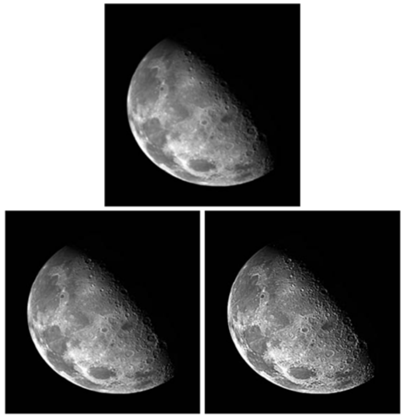


Image Enhancement with Laplacian Filters(cont.)

- Enhancement problems often require the specification of filters beyond those available in the toolbox. The Laplacian is a good example. The toolbox supports a 3×3 Laplacian filter with a -4 in the center. Usually, sharper enhancement is obtained by using the 3×3 Laplacian filter that has a -8 in the center and is surrounded by 1s, as discussed earlier.

```
>>f = imread('Fig0316(a)(moon).tif');
>>w4 = fspecial('laplacian', 0);
>>w8 = [1 1 1; 1 -8 1; 1 1 1];
>>f = im2double(f);
>>g4 = f-imfilter(f, w4, 'replicate');
>>g8 = f-imfilter(f, w8, 'replicate');
>>imshow(f)
>>figure, imshow(g4)
>>figure, imshow(g8)
```


Image Enhancement with Laplacian Filters(cont.)



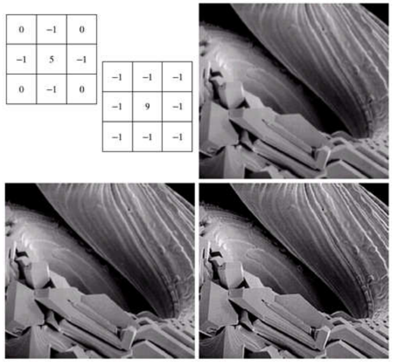
a
b c

FIGURE 3.17 (a) Image of the North Pole of the moon. (b) Image enhanced using the Laplacian filter `'laplacian'`, which has a -4 in the center. (c) Image enhanced using a Laplacian filter with a -8 in the center.

Image Enhancement with Laplacian Filters(cont.)

Simplification:

$$g(x,y) = f(x,y) - \nabla^2 f(x,y) = 5f(x,y) - [f(x+1,y) + f(x-1,y) + f(x,y+1) + f(x,y-1)]$$



a b c
d e

FIGURE 3.41 (a) Composite Laplacian mask. (b) A second composite mask. (c) Scanning electron microscope image. (d) and (e) Results of filtering with the masks in (a) and (b), respectively. Note how much sharper (e) is than (d). (Original image courtesy of Mr. Michael Shaffer, Department of Geological Sciences, University of Oregon, Eugene.)

Unsharp masking and high-boost filtering

- Unsharp masking is implemented by subtract a blurred version of an image from the image itself, i.e.,

  $$f_s(x,y) = f(x,y) - \bar{f}(x,y)$$

  where $f_s(x,y)$ denotes the sharpened image and $\bar{f}(x,y)$ is a blurred version of $f(x,y)$
- A slight further generalizaiton of unsharp masking is called high-boost filtering, and a high-boost filtered image, $f_{hb}(x,y)$, is defined as

  $$f_{hb}(x,y) = Af(x,y) - \bar{f}(x,y)$$

  where $A \geq 1$, and $\bar{f}(x,y)$ is a blurred version of $f(x,y)$
- Further, we can obtain

  $$f_{hb}(x,y) = (A-1)f(x,y) + f(x,y) - \bar{f}(x,y)$$
  $$f_{hb}(x,y) = (A-1)f(x,y) + f_s(x,y)$$
- If we use the Laplacian to compute $f_s(x,y)$, we have

  $$f_{hb}(x,y) = (A-1)f(x,y) - \nabla^2 f(x,y)$$
  or $f_{hb}(x,y) = (A-1)f(x,y) + \nabla^2 f(x,y)$

High-Boost Filtering with Laplacian Filters(cont.)

a b
c d

FIGURE 3.43 (a) Same as Fig. 3.41(c), but darker. (a) Laplacian of (a) computed with the mask in Fig. 3.42(b) using $A = 0$. (c) Laplacian enhanced image using the mask in Fig. 3.42(b) with $A = 1$. (d) Same as (c), but using $A = 1.7$.

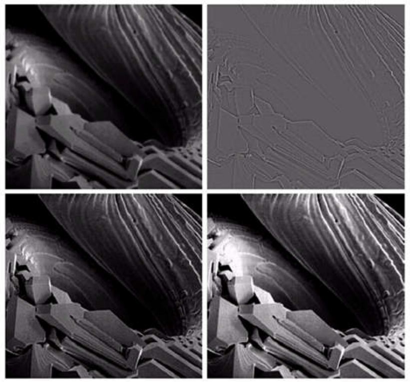


Image Enhancement with the Gradient

a
b c
d e

FIGURE 3.44 A 3 × 3 region of an image (the z's are gray-level values) and masks used to compute the gradient at point labeled z_5. All masks coefficients sum to zero, as expected of a derivative operator.

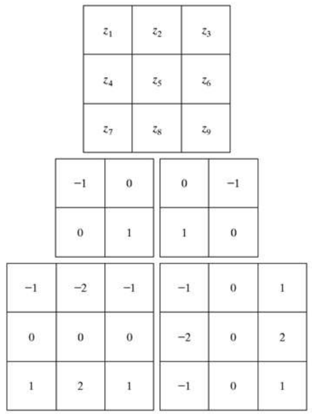
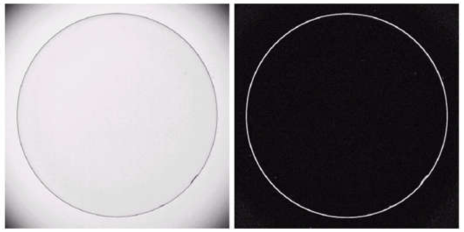


Image Enhancement with the Gradient(cont.)



a b

FIGURE 3.45 Optical image of contact lens (note defects on the boundary at 4 and 5 o'clock). (b) Sobel gradient. (Original image courtesy of Mr. Pete Sites, Perceptics Corporation.)

Nonlinear Spatial Filters

- A commonly-used tool for generating nonlinear spatial filters in IPT is function ordfilt2, which generates order-statistic filters (also called rank filters).
- The syntax of function ordfilt2 is

  g = ordfilt2(f, order, domain)

  This function creates the output image g by replacing each element of f by the order-th element in the sorted set of neighbors specified by the nonzero elements in domain.
- For example, to implement *a min filter* (order 1) of size $m \times n$ we use the syntax

  g = ordfilt2(f, 1, ones(m, n))

  g = ordfilt2(f, m*n, ones(m, n))

  g = ordfilt2(f, median(1:m*n), ones(m, n))

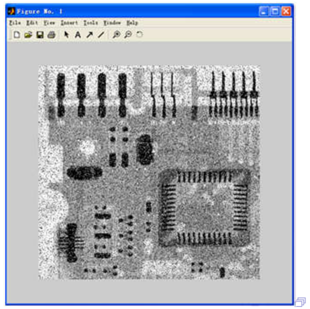
Nonlinear Spatial Filters-Medium Filter

- The best-known order-statistic filter in digital image processing is the median filter
- Because of its practical importance, the toolbox provides a specialized implementation of the 2-D median filter:
  - g = medfilt2(f, [m n], padopt)

    where the tuple [m n] defines a neighborhood of size m×n over which the median is computed, and padopt specifies one of three possible border padding options: 'zeros' (the default), 'symmetric' in which f is extended symmetrically by mirror-reflecting it across its border, and 'indexed', in which f is padded with 1s if it is of class double and with 0s otherwise.

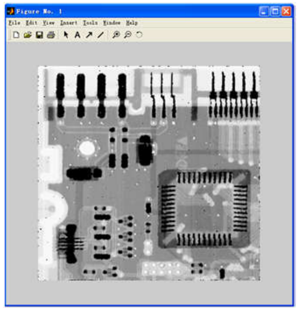
Nonlinear Spatial Filters(cont.)

- Median filtering is a useful tool for reducing salt-and-pepper noise in an image.
  - f=imread('Fig0318(a)(ckt-board-orig).tif');
  - fn=imnoise(f, 'salt & pepper', 0.2);
  - imshow(fn)



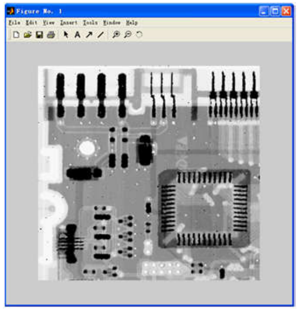
Nonlinear Spatial Filters(cont.)

- gm = medfilt2(fn);
- imshow(gm)

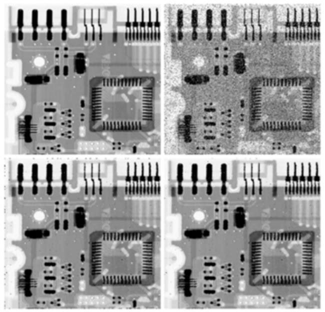


Nonlinear Spatial Filters(cont.)

- gms = medfilt2(fn, 'symmetric');
- imshow(gms)



Nonlinear Spatial Filters(cont.)



a b
c d

FIGURE 3.18 Median filtering. (a) X-ray image. (b) Image corrupted by salt-and-pepper noise. (c) Result of median filtering with `medfilt2` using the default settings. (d) Result of median filtering using the `'symmetric'` image extension option. Note the improvement in border behavior between (d) and (c). (Original image courtesy of Lixi, Inc.)