Information Processing Technology of Internet of Things

Chapter 4 Visual Information Processing

Wu Liu

Beijing Key Lab of Intelligent Telecomm. Software and Multimedia
Beijing University of Posts and Telecommunications

4.2 Image Preprocessing

Introduction

- Pre-processing is the name used for operations on images at the lowest level of abstraction both input and output are intensity images.
- An intensity image usually represented by a matrix or matrices of image function values (brightnesses).
- Pre-processing does not increase image information content. If information is measured using entropy, then pre-processing typically decreases image information content.
- The best way to avoid pre-processing is to concentrate on high-quality image acquisition
- The aim of pre-processing is an improvement of the image data that suppresses undesired distortions or enhances some image features important for further processing.

4.2.1 Pixel brightness transformations

Pixel brightness transformation

- A brightness transformation modifies pixel brightness--the transformation depends on the properties of a pixel itself.
- There are two classes of pixel brightness transformations
  - brightness corrections: modifies the pixel brightness taking into account its original brightness and its position in the image.
  - gray-scale transformations: change brightness without regard to position in the image.

Position-dependent brightness correction

- If image degradation is of a systematic nature, it can be suppressed by brightness correction.
- A multiplicative error coefficient e(i,j) describes the change from the ideal identity transfer function.
- Assume that g(i, j) is the original undegraded image (or desired or true image) and f(i, j) is the image containing degradation. Then

$$f(i,j) = e(i,j)\, g(i,j)$$

- The error coefficient e(i, j) can be obtained if a reference image g(i, j) with known brightnesses is captured, the simplest being an image of constant brightness c. The degraded result is the image fc(i, j).
- Then systematic brightness errors can be suppressed by

$$g(i,j) = \frac{f(i,j)}{e(i,j)} = \frac{c\, f(i,j)}{f_c(i,j)}$$

Position-dependent brightness correction

- This method can be used only if the image degradation process is stable.
- If we wish to suppress this kind of error in the image capturing process, we should perhaps re-calibrate the device (find error coefficients e(i, j)) from time to time.

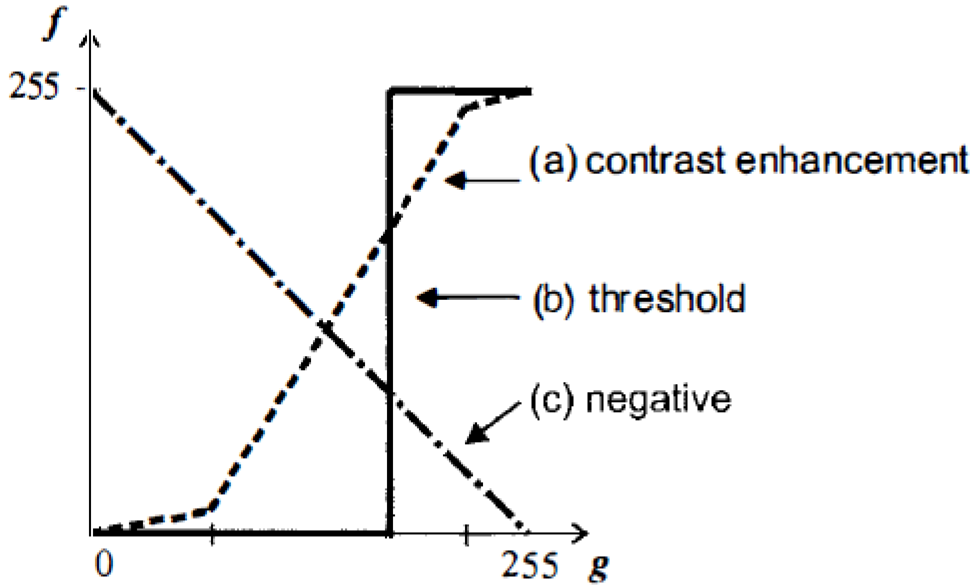
Gray-scale transformation

- Gray-scale transformations do not depend on the position of the pixel in the image.
- A transformation T of the original brightness p from scale [p0 , pk] into brightness q from a new scale [q0, qk] is given by

$$q = \mathcal{T}(p)$$

Gray-scale transformation

- The most common gray-scale transformations are shown
  - the piecewise linear function a enhances the image contrast between brightness values p1 and p2 .
  - The function b is called brightness thresholding and results in a black-and-white image
  - the straight line c denotes the negative transformation.

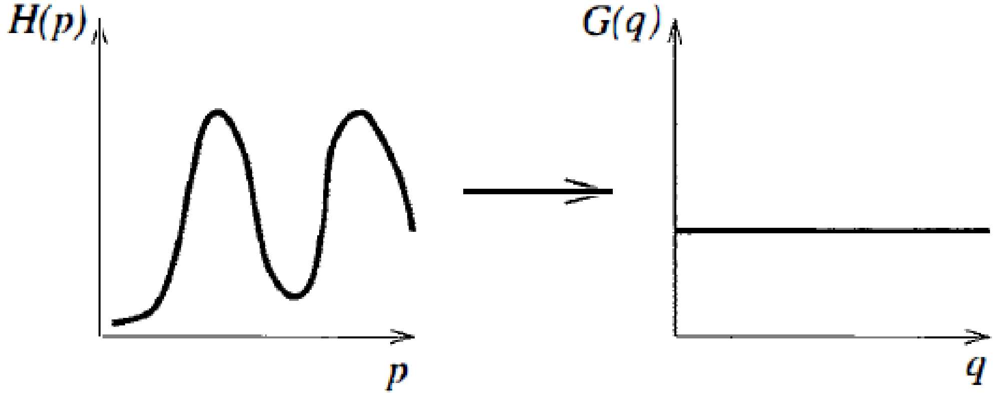


Gray-scale transformation

- Gray-scale transformations are used mainly when an image is viewed by a human observer, and a transformed image might be more easily interpreted if the contrast is enhanced.
- A gray-scale transformation for contrast enhancement is usually found automatically using the histogram equalization technique.
  - The aim is to create an image with **equally distributed brightness levels over the whole brightness scale**
  - Histogram equalization enhances contrast for brightness values close to histogram maxima, and decreases contrast near minima.

Gray-scale transformation-- histogram equalization

- Denote the input histogram by H(p) and recall that the input gray-scale is [p0, pk].
- The intention is to find a monotonic pixel brightness transformation q = T(p) such that the desired output histogram G(q) is uniform over the whole output brightness scale [q0 , qk].



Gray-scale transformation-- histogram equalization

- The histogram can be treated as a discrete probability density function. The monotonic property of the transform T implies

$$\sum_{i=0}^{k} G(q_i) = \sum_{i=0}^{k} H(p_i) \qquad (5.4)$$

- Assume that the image has N rows and columns; then the equalized histogram G(q) corresponds to the uniform probability density function f whose function value is a constant:

$$f = \frac{N^2}{q_k - q_0}$$

Gray-scale transformation-- histogram equalization

- The equalized histogram can be obtained precisely only for the 'idealized' continuous probability density, in which case equation (5.4) becomes

$$N^2 \int_{q_0}^{q} \frac{1}{q_k - q_0}\, \mathrm{d}s = \frac{N^2 (q - q_0)}{q_k - q_0} = \int_{p_0}^{p} H(s)\, \mathrm{d}s$$

- The desired pixel brightness transformation T can then be derived as

$$q = \mathcal{T}(p) = \frac{q_k - q_0}{N^2} \int_{p_0}^{p} H(s)\, \mathrm{d}s + q_0 \qquad (5.7)$$

- The integral in equation (5.7) is called the cumulative histogram, which is approximated by a sum in digital images, so the resulting histogram is not equalized ideally.

Gray-scale transformation-- histogram equalization

- The discrete approximation of the continuous pixel brightness transformation from equation (5.7) is

$$q = T(p) = \frac{q_k - q_0}{N^2} \sum_{i=p_0}^{p} H(i) + q_0$$

Algorithm 5.1: Histogram equalization

1. For an $N \times M$ image of G gray-levels (often 256), create an array H of length G initialized with 0 values.
2. Form the image histogram: Scan every pixel and increment the relevant member of H—if pixel p has intensity g_p, perform

$$H[g_p] = H[g_p] + 1 \, .$$

3. Form the cumulative image histogram H_c:

$$H_c[0] = H[0] \, ,$$
$$H_c[p] = H_c[p-1] + H[p] \, , \quad p = 1, 2, \ldots, G-1 \, .$$

4. Set

$$T[p] = \mathrm{round}\left(\frac{G-1}{NM} H_c[p]\right) \, .$$

   (This step obviously lends itself to more efficient implementation by constructing a look-up table of the multiples of $(G-1)/NM$, and making comparisons with the values in H_c, which are monotonically increasing.)

5. Rescan the image and write an output image with gray-levels g_q, setting

$$g_q = T[g_p] \, .$$

Gray-scale transformation-- histogram equalization

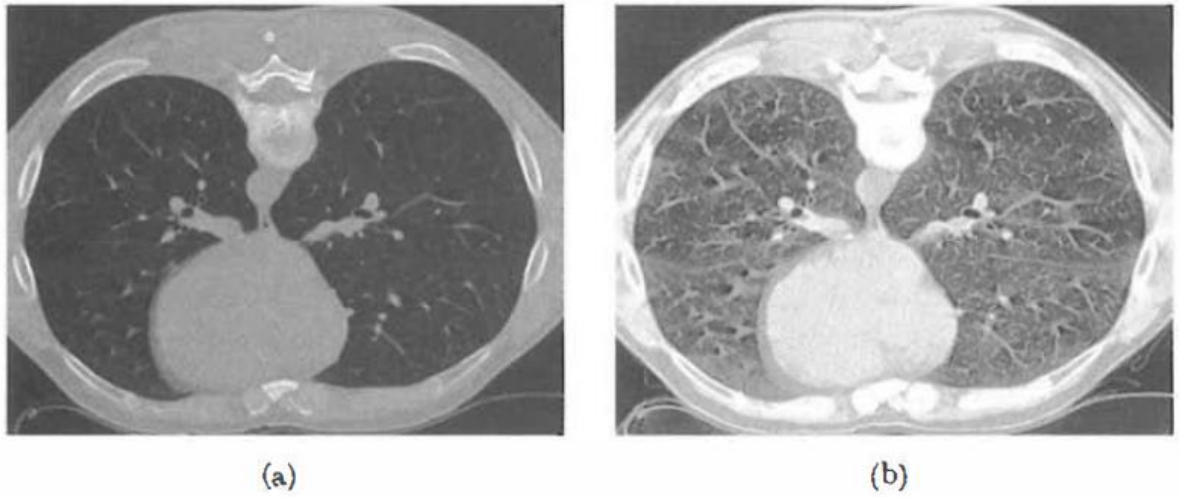


Figure 5.3: Histogram equalization. (a) Original image. (b) Equalized image.

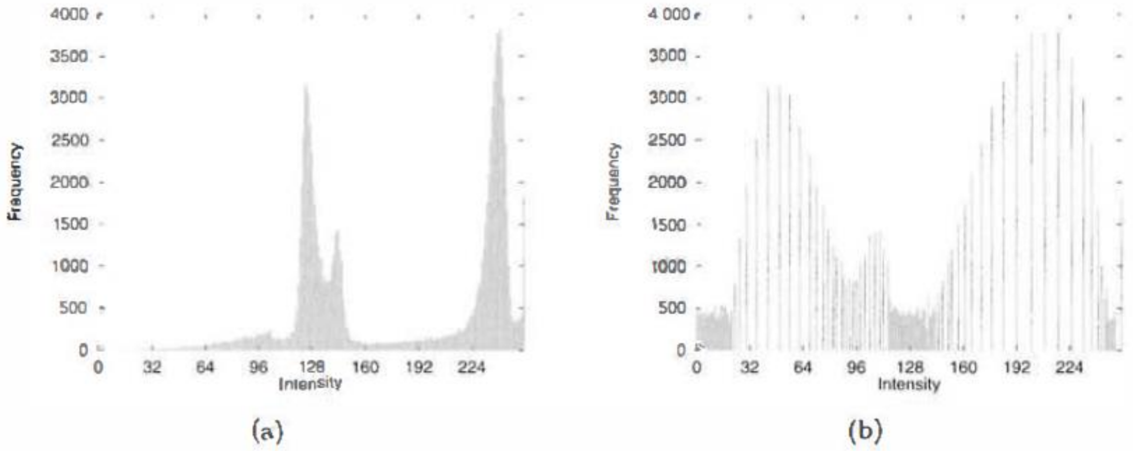


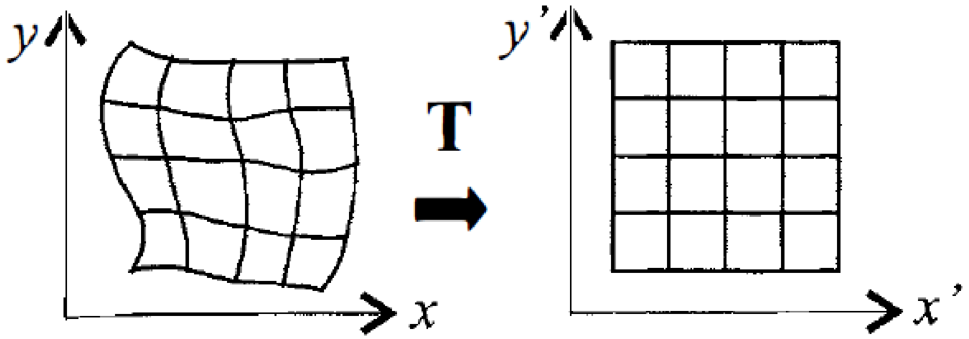
Figure 5.4: Histogram equalization: Original and equalized histograms corresponding to Figure 5.3a,b.

4.2.2 Geometric transformations

Geometric transformations

- Geometric transforms permit elimination of the geometric distortion that occurs when an image is captured.
- If one attempts to match two different images of the same object, a geometric transformation may be needed.
- A geometric transform is a vector function T that maps the pixel (x, y) to a new position (x', y')
- T is defined by its two component equations

$$x' = T_x(x,y), \qquad y' = T_y(x,y)$$



Geometric transformations

- A geometric transform consists of two basic steps.
  - First is the pixel co-ordinate transformation, which maps the co-ordinates of the input image pixel to the point in the output image. The output point co-ordinates should be computed as continuous values (real numbers) , as the position does not necessarily match the digital grid after the transform.
  - The second step is to find the point in the digital raster which matches the transformed point and determine its brightness value. The brightness is usually computed as an **interpolation** of the brightnesses of several points in the neighborhood.

Pixel co-ordinate transformations

- Find the co-ordinates of a point in the output image after a geometric transform. It is usually approximated by a polynomial equation:

$$x' = \sum_{r=0}^{m} \sum_{k=0}^{m-r} a_{rk}\, x^r\, y^k\,, \qquad y' = \sum_{r=0}^{m} \sum_{k=0}^{m-r} b_{rk}\, x^r\, y^k$$

- This transform is linear with respect to the coefficients a_{rk} , b_{rk}

Pixel co-ordinate transformations

- In practice, T is approximated by a bilinear transform for which four pairs of corresponding points are sufficient to find the transformation coefficients

$$x' = a_0 + a_1\,x + a_2\,y + a_3\,x\,y\,,$$
$$y' = b_0 + b_1\,x + b_2\,y + b_3\,x\,y\,.$$

- Even simpler is the affine transformation, for which three pairs of corresponding points are sufficient to find the coefficients

$$x' = a_0 + a_1\,x + a_2\,y\,,$$
$$y' = b_0 + b_1\,x + b_2\,y\,.$$

Pixel co-ordinate transformations

- A geometric transform applied to the whole image may change the co-ordinate system, and a Jacobian J provides information about how the co-ordinate system changes

$$J = \left|\frac{\partial(x', y')}{\partial(x, y)}\right| = \begin{vmatrix} \partial x'/\partial x & \partial x'/\partial y \\ \partial y'/\partial x & \partial y'/\partial y \end{vmatrix}$$

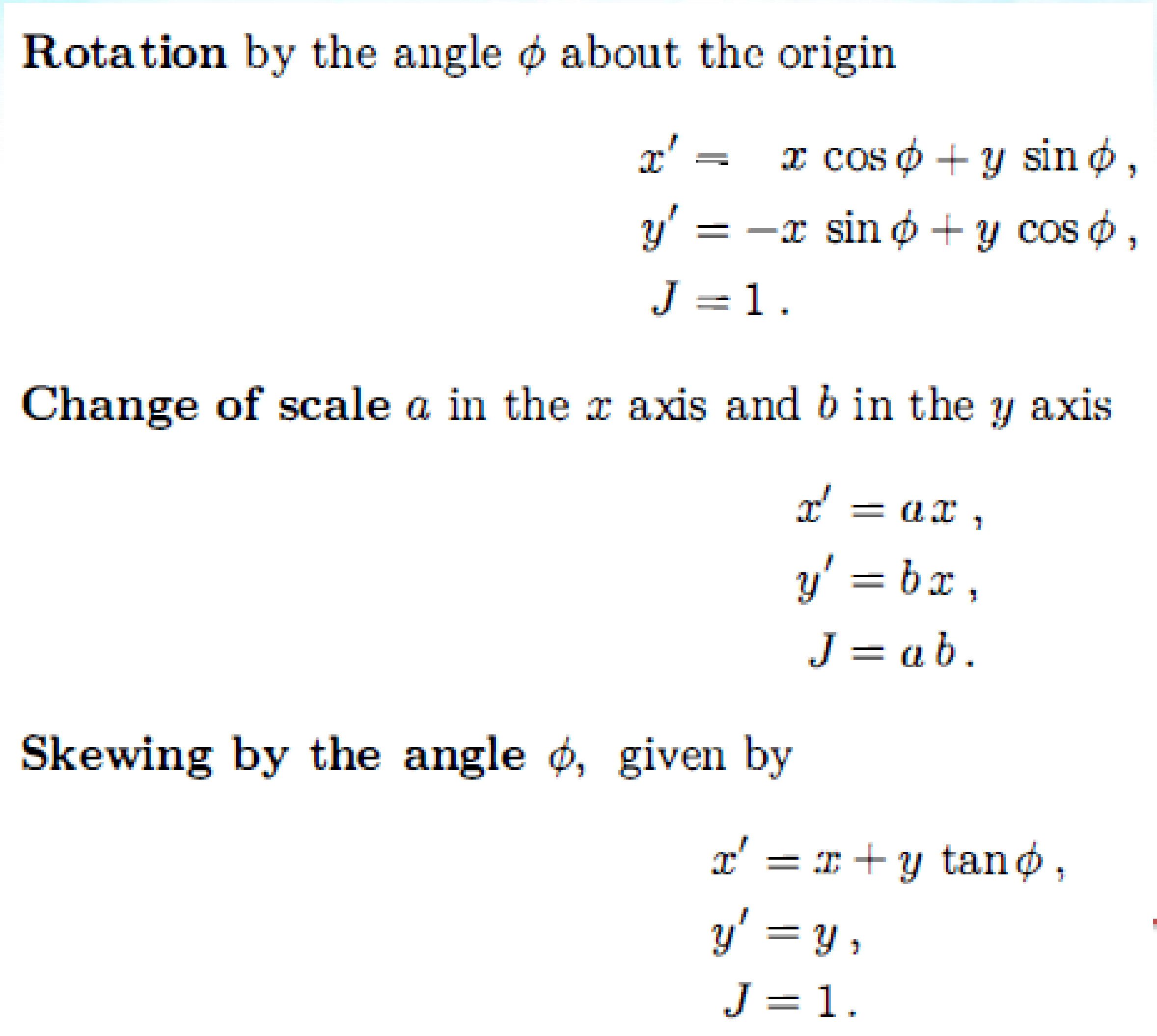
- If the transformation is singular (has no inverse) , then J = 0. If the area of the image is invariant under the transformation, then J = 1 .
- The Jacobian for the bilinear transform and the affine transformation are respectively:

$$J = a_1 b_2 - a_2 b_1 + (a_1 b_3 - a_3 b_1)x + (a_3 b_2 - a_2 b_3)y$$

$$J = a_1 b_2 - a_2 b_1$$

Pixel co-ordinate transformations

- Some important geometric transformations are



Rotation by the angle ϕ about the origin

$$x' = x \cos\phi + y \sin\phi \,,$$
$$y' = -x \sin\phi + y \cos\phi \,,$$
$$J = 1 \,.$$

Change of scale a in the x axis and b in the y axis

$$x' = a\,x \,,$$
$$y' = b\,x \,,$$
$$J = a\,b \,.$$

Skewing by the angle ϕ, given by

$$x' = x + y \tan\phi \,,$$
$$y' = y \,,$$
$$J = 1 \,.$$

Pixel co-ordinate transformations

- To **approximate complex geometric transformations** (distortion):
  - partitioning an image into smaller rectangular subimages;
  - for each subimage, a simple geometric transformation, such as the affine, is estimated using pairs of corresponding pixels.
  - The geometric transformation (distortion) is then repaired separately in each subimage.

Brightness interpolation

- After pixel co-ordinate transformations, new point co-ordinates (x', y') obtained. The collection of transformed points gives the samples of the output image with non-integer co-ordinates.
- Values on the integer grid are needed, and each pixel value in the output image raster can be obtained by brightness interpolation of some neighboring non-integer samples.
- Brightness interpolation influences image quality.
- The brightness interpolation problem is usually expressed in a dual way by determining the brightness of the original point in the input image that corresponds to the point in the output image lying on the discrete raster.

Brightness interpolation

- The brightness can be expressed by the convolution equation

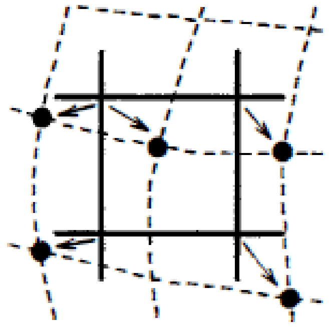
$$f_n(x,y) = \sum_{l=-\infty}^{\infty} \sum_{k=-\infty}^{\infty} g_s(l\,\Delta x, k\,\Delta y)\, h_n(x - l\,\Delta x, y - k\,\Delta y)\;.$$

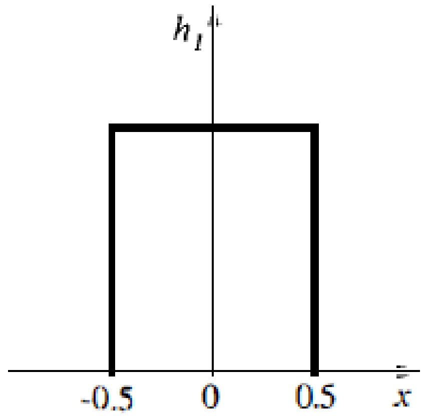
The function h_n is called the interpolation kernel. Usually, only a small neighborhood is used, outside which h_n is zero.

Brightness interpolation

- Nearest-neighborhood interpolation assigns to the point (x, y) the brightness value of the nearest point g in the discrete raster

$$f_1(x, y) = g_s\big(\mathrm{round}(x), \mathrm{round}(y)\big)$$





The discrete raster of the original image is depicted by the solid line.

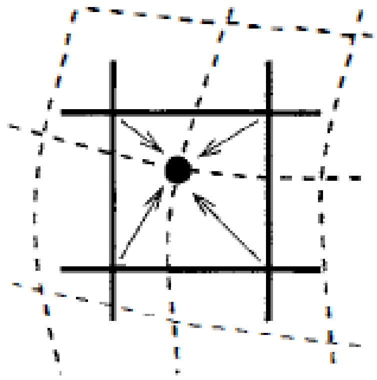
the interpolation kernel h1 in the 1D case

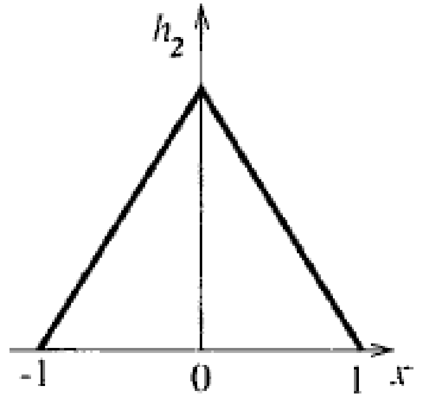
Brightness interpolation

- Linear interpolation explores four points neighboring the point (x,y), and assumes that the brightness function is linear in this neighborhood.

$$f_2(x,y) = (1-a)(1-b)\,g_s(l,k) + a\,(1-b)\,g_s(l+1,k) + b\,(1-a)\,g_s(l,k+1) + a\,b\,g_s(l+1,k+1)\,,$$

$$l = \mathrm{floor}(x)\,,\quad a = x - l\,,\quad k = \mathrm{floor}(y)\,,\quad b = y - k\,.$$





The discrete raster of the original image is depicted by the solid line.

4.2.3 Local pre-processing

Local pre-processing

- We shall consider methods that use a small neighborhood of a pixel in an input image to produce a new brightness value in the output image.
  - Also called filtration (or filtering) if signal processing terminology is used.
- Local pre-processing methods can be divided into two groups according to the goal of the processing.
  - smoothing aims to suppress noise or other small fluctuations in the image; smoothing also blurs all sharp edges that bear important information about the image.
  - gradient operators are based on local derivatives of the image function. Derivatives are bigger at locations of the image where the image function undergoes rapid changes, and the aim of gradient operators is to indicate such locations in the image. Noise is often high frequency in nature; if a gradient operator is applied to an image, the noise level increases simultaneously.

Local pre-processing

- Another classification of local pre-processing methods is according to the transformation properties
  - Linear transformations
  - Non-linear transformations
- Linear operations calculate the resulting value in the output image pixel g(i, j) as a linear combination of brightnesses in a local neighborhood O of the pixel f(i, j) in the input image.
- The contribution of the pixels in the neighborhood O is weighted by coefficients h:

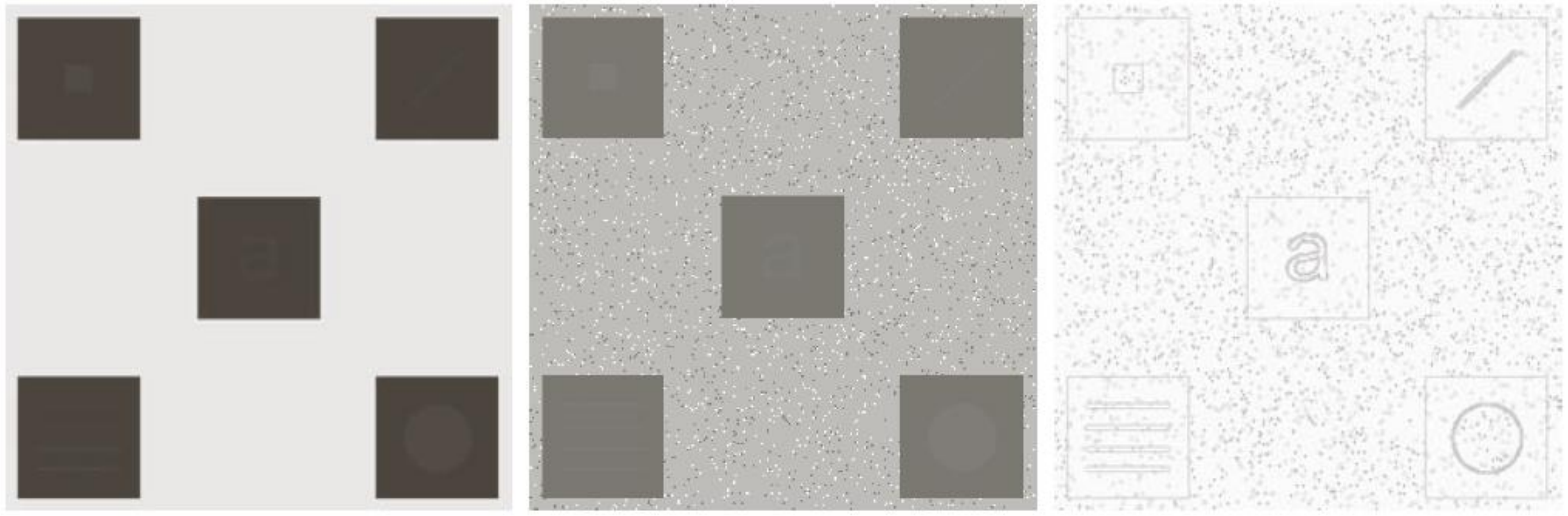
$$f(i,j) = \sum_{(m,n)\in O} \sum h(i-m, j-n)\, g(m,n)$$

the kernel h is called a convolution mask.

local histogram processing

- problem: global spatial processing not always desirable
- solution: apply point-operations to a pixel neighborhood with a sliding window

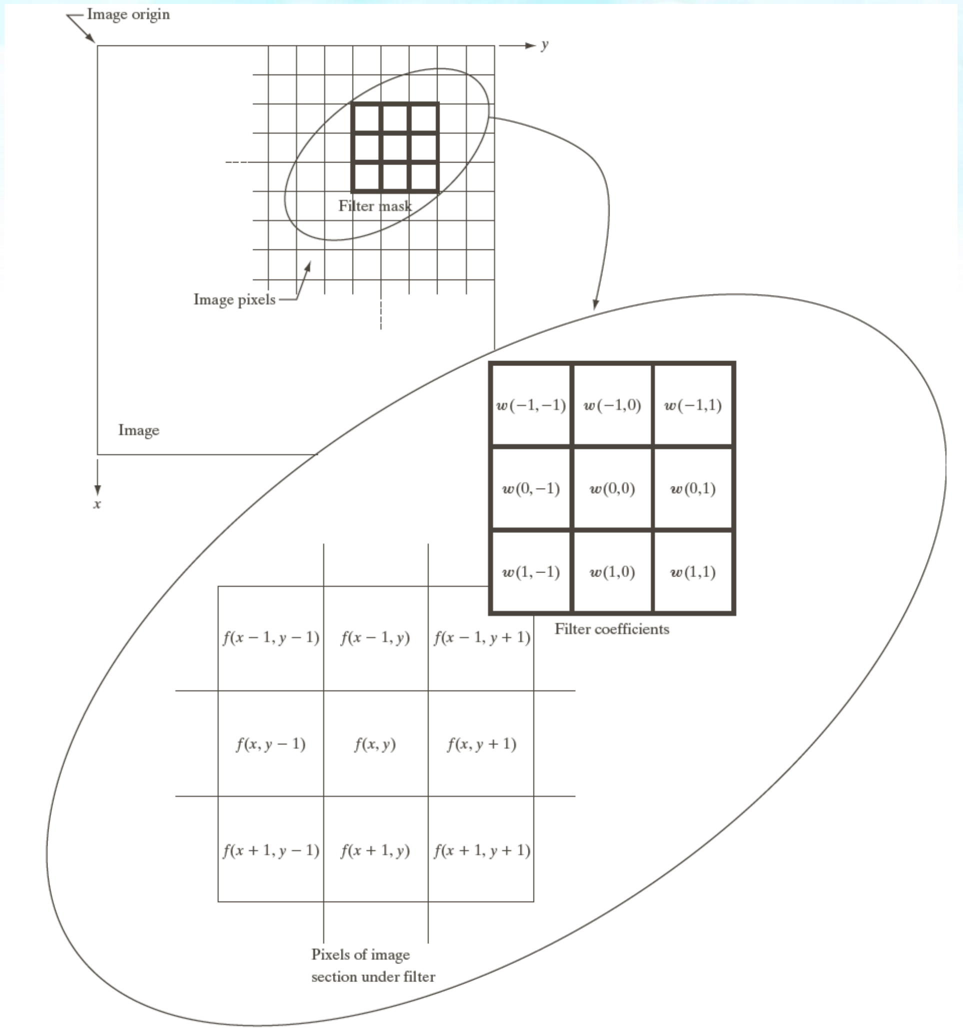
w_1	w_2	w_3
w_4	w_5	w_6
w_7	w_8	w_9



a b c

FIGURE 3.26 (a) Original image. (b) Result of global histogram equalization. (c) Result of local histogram equalization applied to (a), using a neighborhood of size 3×3.

spatial filtering in image neighborhoods

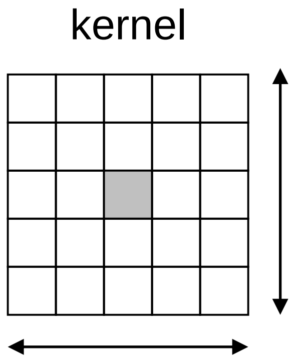


kernel operator / filter masks

$$T_N(.) = w(.)$$

$$f \longrightarrow \boxed{\text{Spatial Filtering}} \longrightarrow g$$

$$g(m,n) = \sum_{i=-a}^{a} \sum_{j=-b}^{b} w(i,j)\, f(m+i, n+j)$$

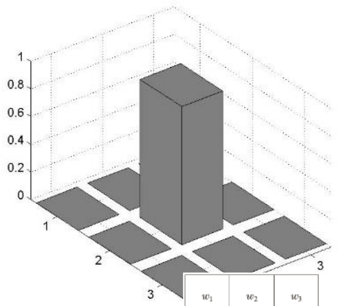


$1 \le m \le M$

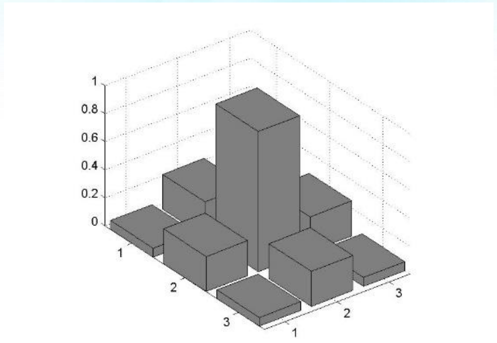
$1 \le n \le N$

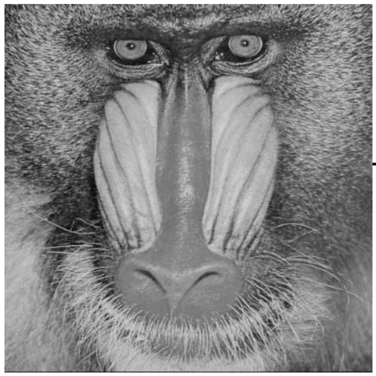
Smoothing: Image Averaging

$$\frac{1}{9} \times \begin{array}{|c|c|c|}\hline 1 & 1 & 1 \\ \hline 1 & 1 & 1 \\ \hline 1 & 1 & 1 \\ \hline \end{array} \qquad \frac{1}{16} \times \begin{array}{|c|c|c|}\hline 1 & 2 & 1 \\ \hline 2 & 4 & 2 \\ \hline 1 & 2 & 1 \\ \hline \end{array}$$

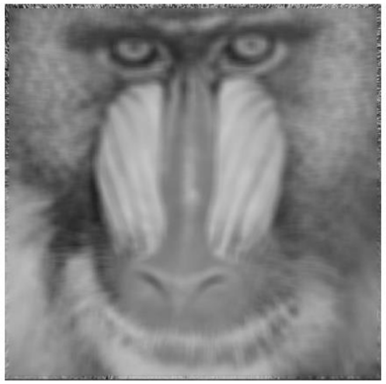


w_1	w_2	w_3
w_4	w_5	w_6
w_7	w_8	w_9





$\longrightarrow$ smoothing operator $\longrightarrow$



Low-pass filter, leads to softened edges

spatial averaging can suppress noise

- image with iid noise $y(m,n) = x(m,n) + N(m,n)$
- averaging
  $v(m,n) = (1/N_w) \Sigma\, x(m-k, n-l) + (1/N_w) \Sigma\, N(m-k, n-l)$
    - N_w: number of pixels in the averaging window
  - Noise variance reduced by a factor of N_w
  - SNR improved by a factor of N_w
  - Window size is limited to avoid excessive blurring

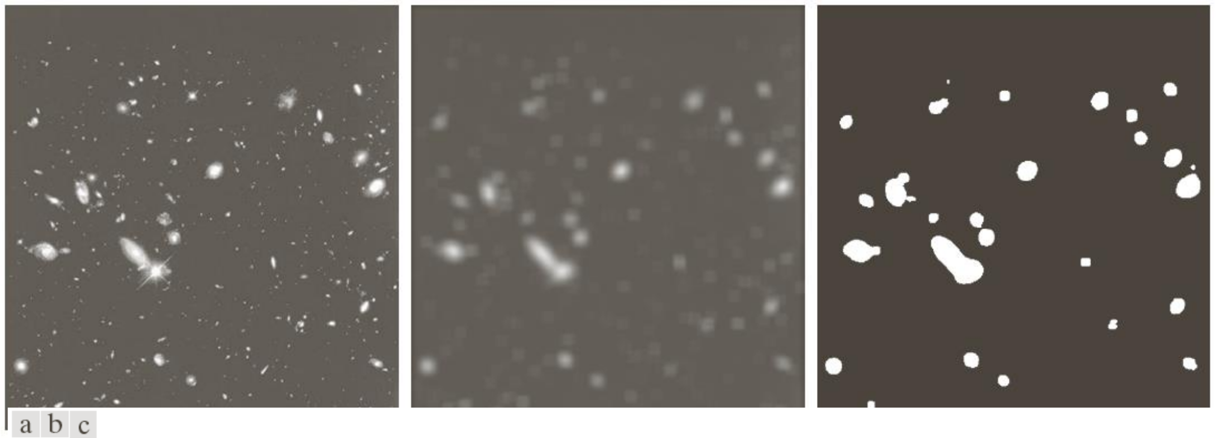
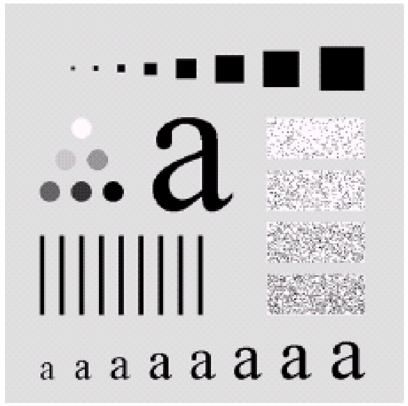
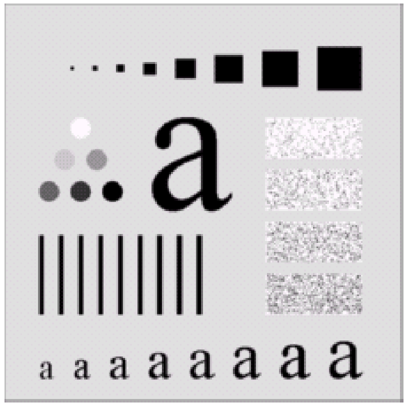
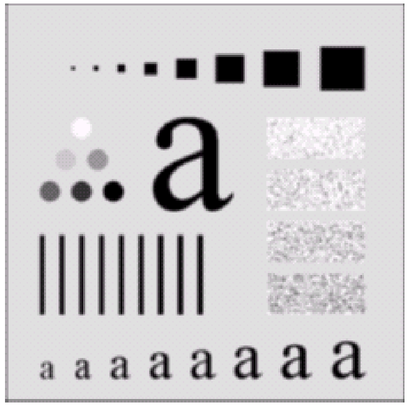


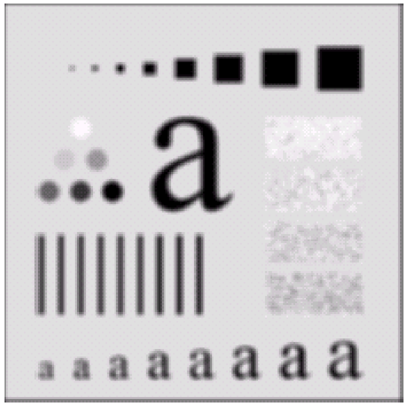
FIGURE 3.34 (a) Image of size 528 × 485 pixels from the Hubble Space Telescope. (b) Image filtered with a 15 × 15 averaging mask. (c) Result of thresholding (b). (Original image courtesy of NASA.)

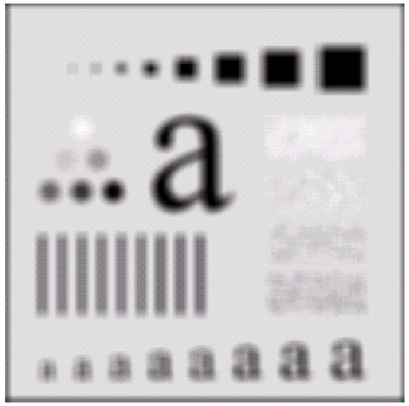
smoothing operator of different sizes

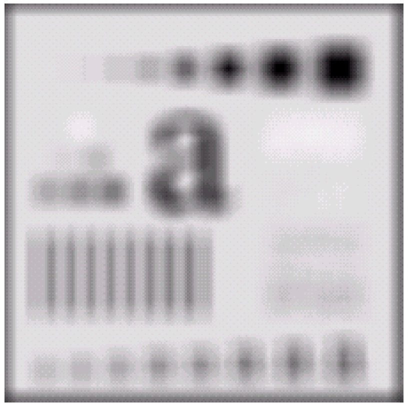

original


3x3


5x5

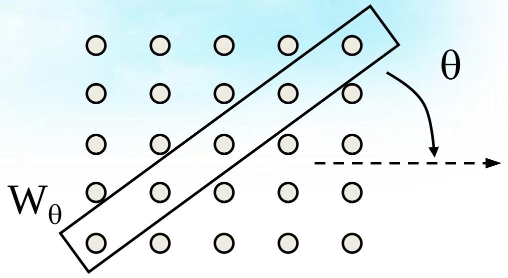

9x9


15x15


35x35

directional smoothing

- Problems with simple spatial averaging mask
  - Edges get blurred
- Improvement
  - Restrict smoothing to along edge direction
  - Avoid filtering across edges



- Directional smoothing
  - Compute spatial average along several directions
  - Take the result from the direction giving the smallest changes before & after filtering
- Other solutions
  - Use more explicit edge detection and adapt filtering accordingly

non-linear smoothing operator

- Median filtering
  - median value ξ over a small window of size N_w

$$\tilde{x} = sort(x);\ \ \xi = \tilde{x}[\frac{N_w + 1}{2}]$$

  - nonlinear
    - median{ x(m) + y(m) } $\neq$ median{x(m)} + median{y(m)}
  - odd window size is commonly used
    - 3x3, 5x5, 7x7
    - 5-pixel "+"-shaped window
  - for even-sized windows take the average of two middle values as output
- Other order statistics: min, max, x-percentile …

median filter example

- Median filtering
  - resilient to statistical outliers
  - incurs less blurring
  - simple to implement

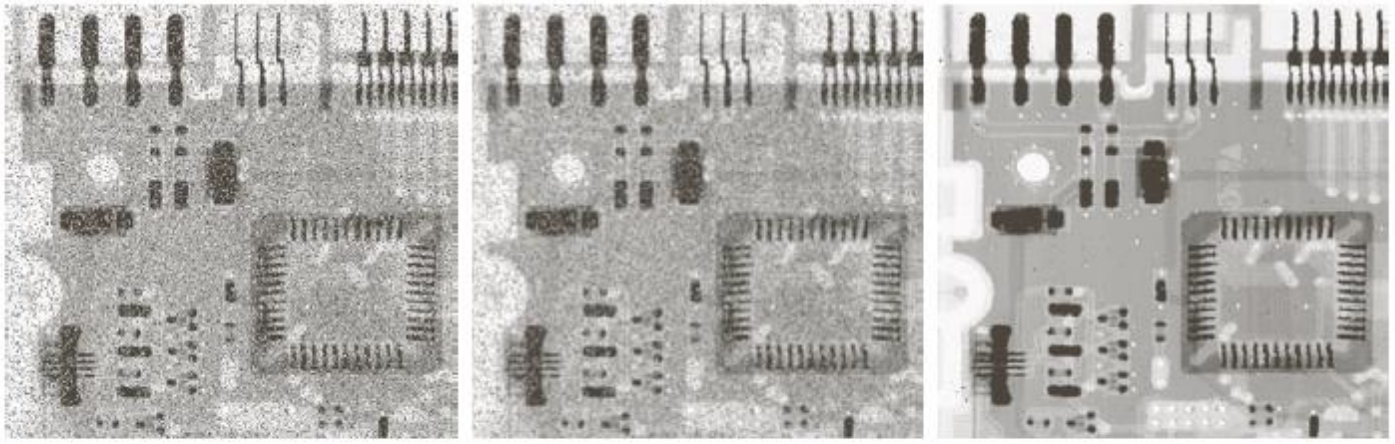
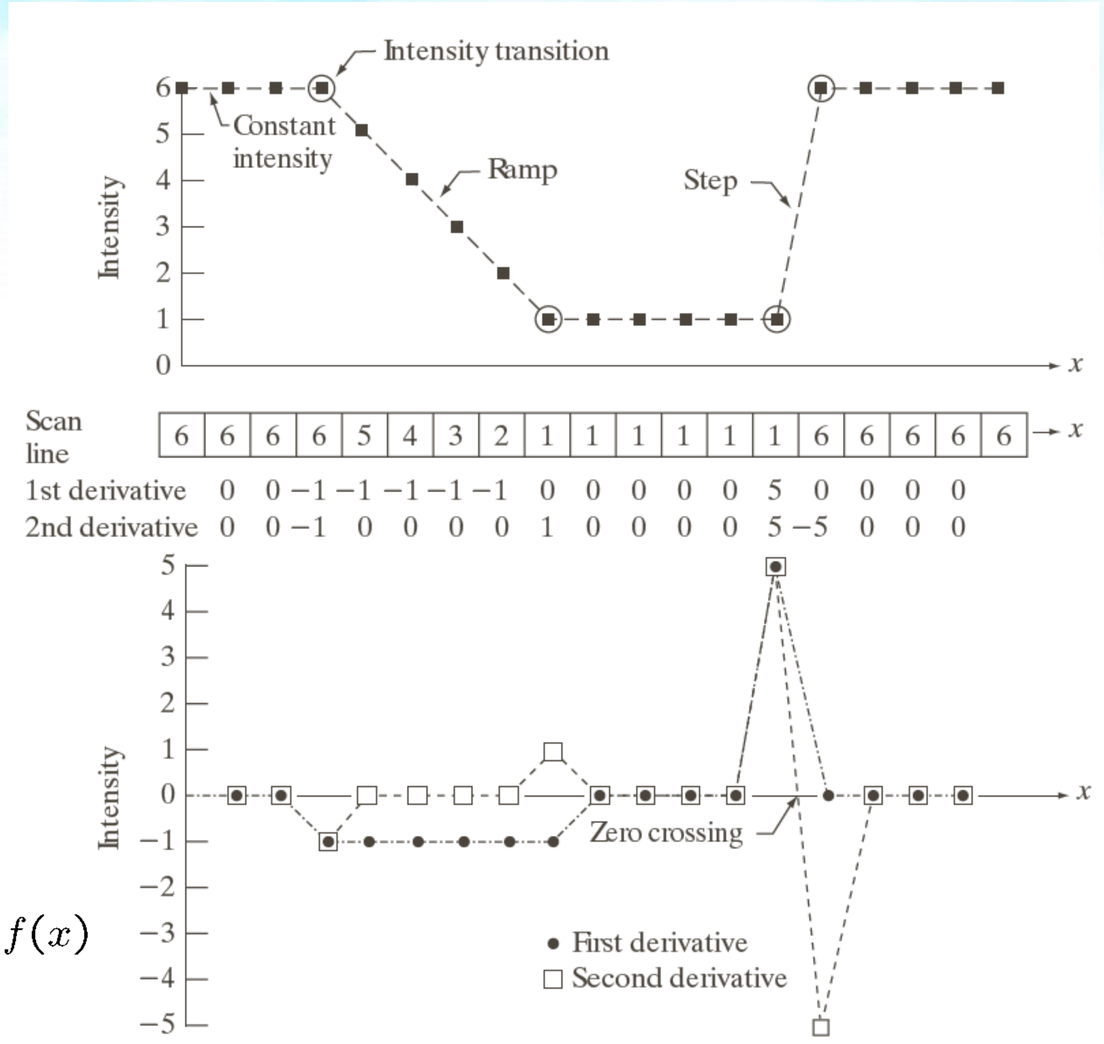


FIGURE 3.35 (a) X-ray image of circuit board corrupted by salt-and-pepper noise. (b) Noise reduction with a 3 × 3 averaging mask. (c) Noise reduction with a 3 × 3 median filter. (Original image courtesy of Mr. Joseph E. Pascente, Lixi, Inc.)

image derivative and sharpening

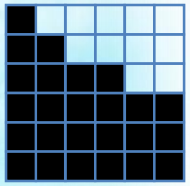
$$f'(x) = \frac{\partial f}{\partial x} = f(x+1) - f(x)$$

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

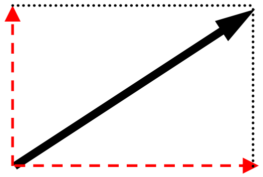


Scan line	6	6	6	6	5	4	3	2	1	1	1	1	1	1	6	6	6	6	6
1st derivative		0	0	−1	−1	−1	−1	−1	0	0	0	0	0	5	0	0	0	0	
2nd derivative		0	0	−1	0	0	0	0	1	0	0	0	0	5	−5	0	0	0	

edge and the first derivative



- Edge: pixel locations of abrupt luminance change



- Spatial luminance gradient vector
  - a vector consists of partial derivatives along two orthogonal directions
  - gradient gives the direction with highest rate of luminance changes
- Representing edge: edge intensity + directions
- Detection Methods
  - prepare edge examples (templates) of different intensities and directions, then find the best match
  - measure transitions along 2 orthogonal directions

edge detection operators

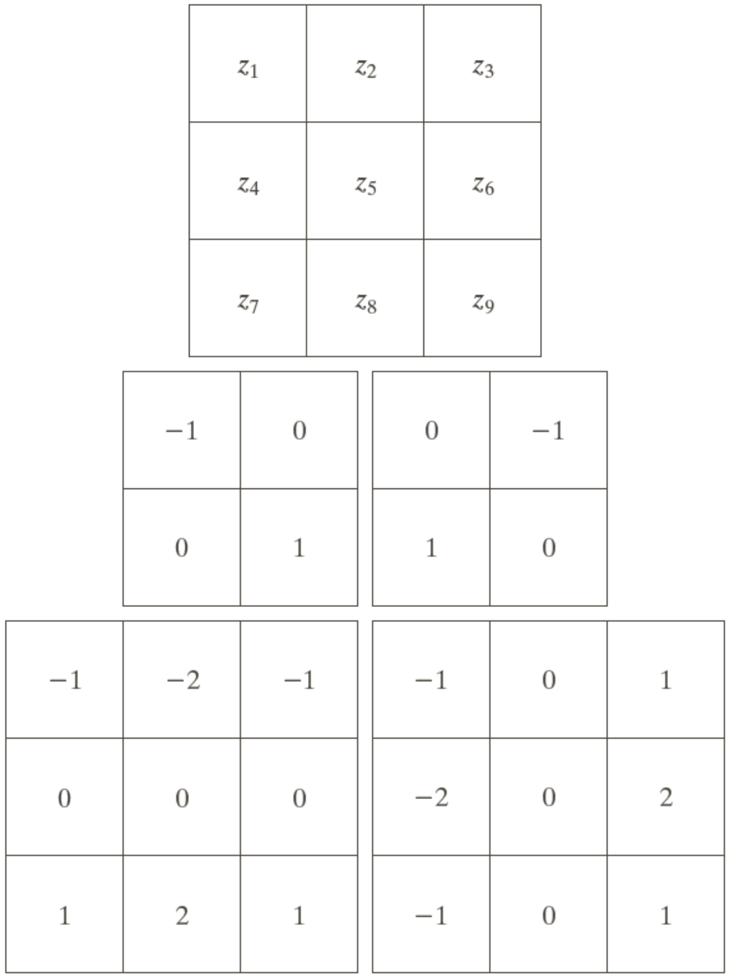
Image gradient:

$$\nabla f = \begin{bmatrix} G_x \\ G_y \end{bmatrix} = \begin{bmatrix} \frac{\partial f}{\partial x} \\ \frac{\partial f}{\partial y} \end{bmatrix}$$

$$\|\nabla f\| \approx |G_x| + |G_y|$$

Robert's operator

Sobel's operator



z_1	z_2	z_3
z_4	z_5	z_6
z_7	z_8	z_9

−1	0
0	1

0	−1
1	0

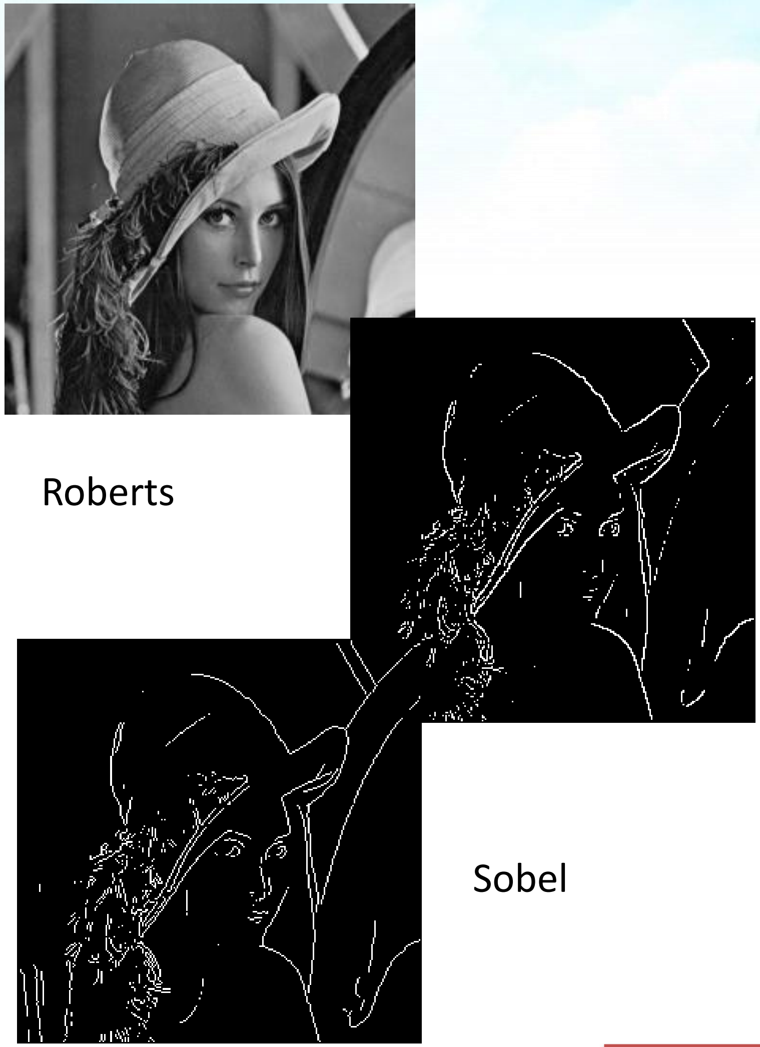
−1	−2	−1
0	0	0
1	2	1

−1	0	1
−2	0	2
−1	0	1

a
b c
d e

FIGURE 3.41 A 3 × 3 region of an image (the *z*s are intensity values). (b)–(c) Roberts cross gradient operators. (d)–(e) Sobel operators. All the mask coefficients sum to zero, as expected of a derivative operator.

edge detection example



Roberts

Sobel

http://flickr.com/photos/reneemarie11/97326485/

second derivative in 2D

Image Laplacian: $$\nabla^2 f = \frac{\partial^2 f}{\partial x^2} + \frac{\partial^2 f}{\partial y^2}$$

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

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

强调的是图像中灰度的突变。

0	1	0
1	–4	1
0	1	0

1	1	1
1	–8	1
1	1	1

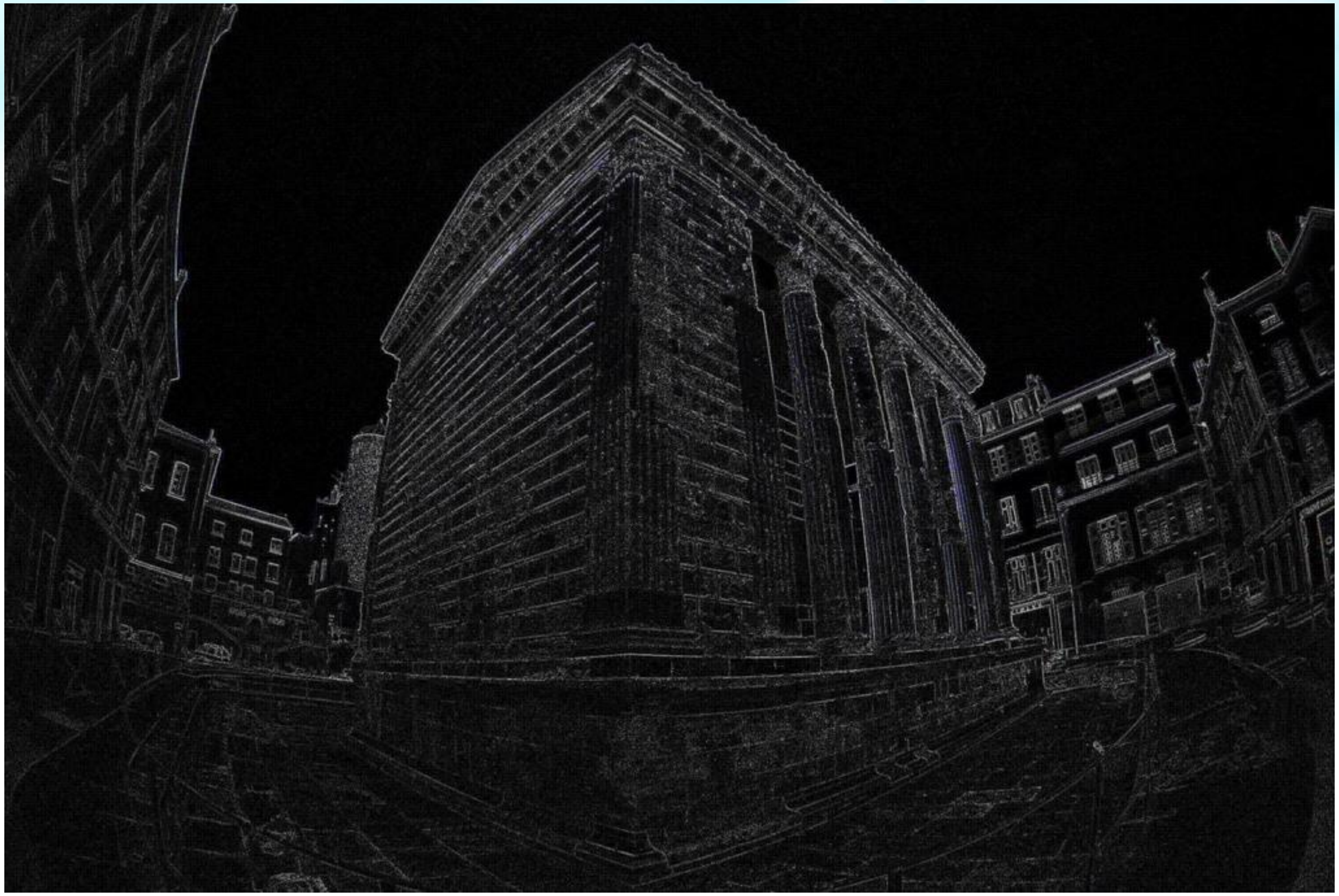
0	–1	0
–1	4	–1
0	–1	0

–1	–1	–1
–1	8	–1
–1	–1	–1

a b
c d

FIGURE 3.39 (a) Filter mask used to implement the digital Laplacian, as defined in Eq. (3.7-4). (b) Mask used to implement an extension of this equation that includes the diagonal neighbors. (c) and (d) Two other implementations of the Laplacian.

laplacian of roman ruins



http://flickr.com/photos/starfish235/388557119/

unsharp masking

- **Unsharp masking** is an image manipulation technique for increasing the apparent sharpness of photographic images.
- The "unsharp" of the name derives from the fact that the technique uses a blurred, or "unsharp", positive to create a "mask" of the original image. The unsharped mask is then combined with the negative, creating a resulting image sharper than the original.

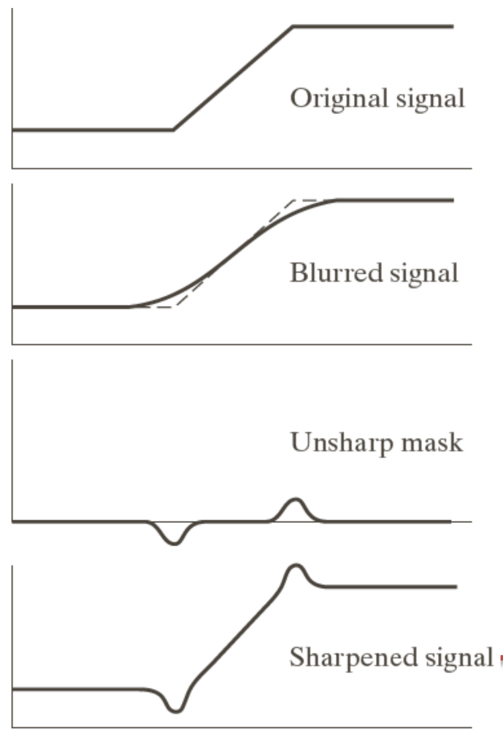
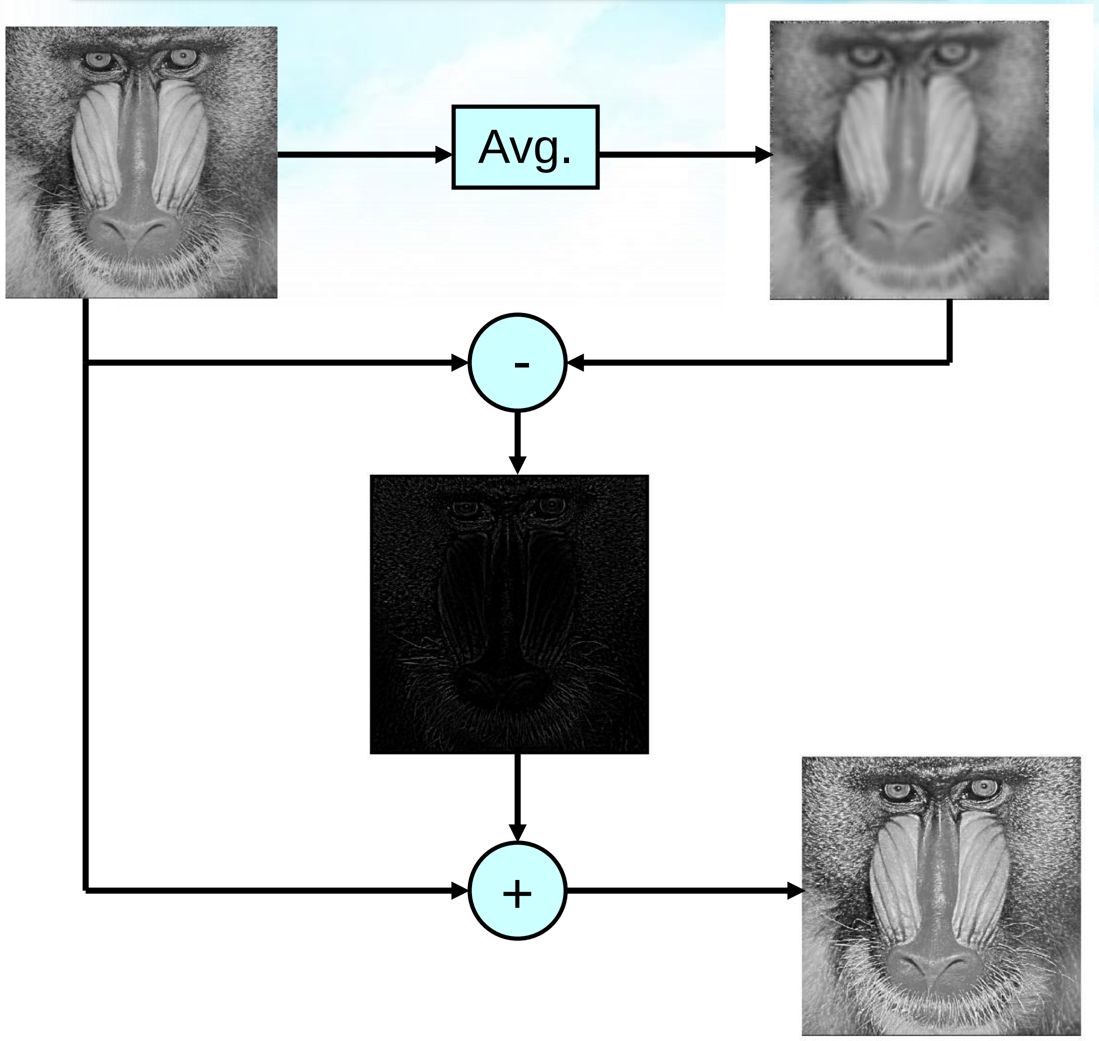


FIGURE 3.39 1-D illustration of the mechanics of unsharp masking. (a) Original signal. (b) Blurred signal with original shown dashed for reference. (c) Unsharp mask. (d) Sharpened signal, obtained by adding (c) to (a).

- Steps
  - Blur the image
  - Subtract the blurred version from the original (this is called the *mask*)
  - Add the "mask" to the original

high-boost filtering



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

$$g(x,y) = f(x,y) + k * g_{mask}(x,y)$$

$$f(x,y) - \mathit{original\ image},$$

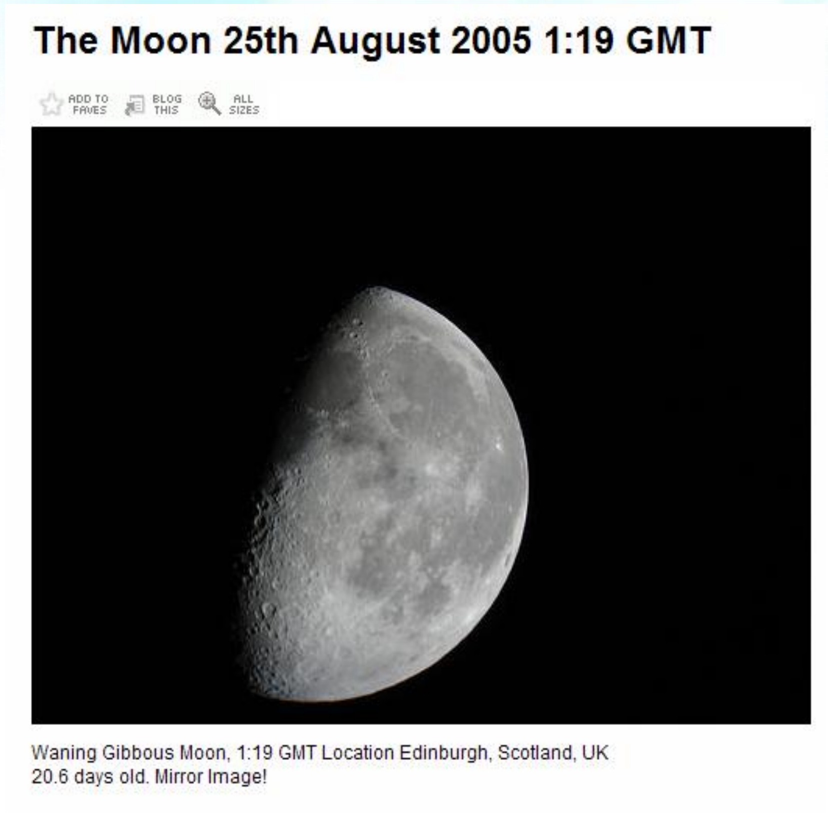
$$\bar{f}(x,y) - \mathit{Blurred\ image},$$

$$g_{mask}(x,y) - \mathit{unsharp\ mask}.$$

0	−1	0
−1	$A+4$	−1
0	−1	0

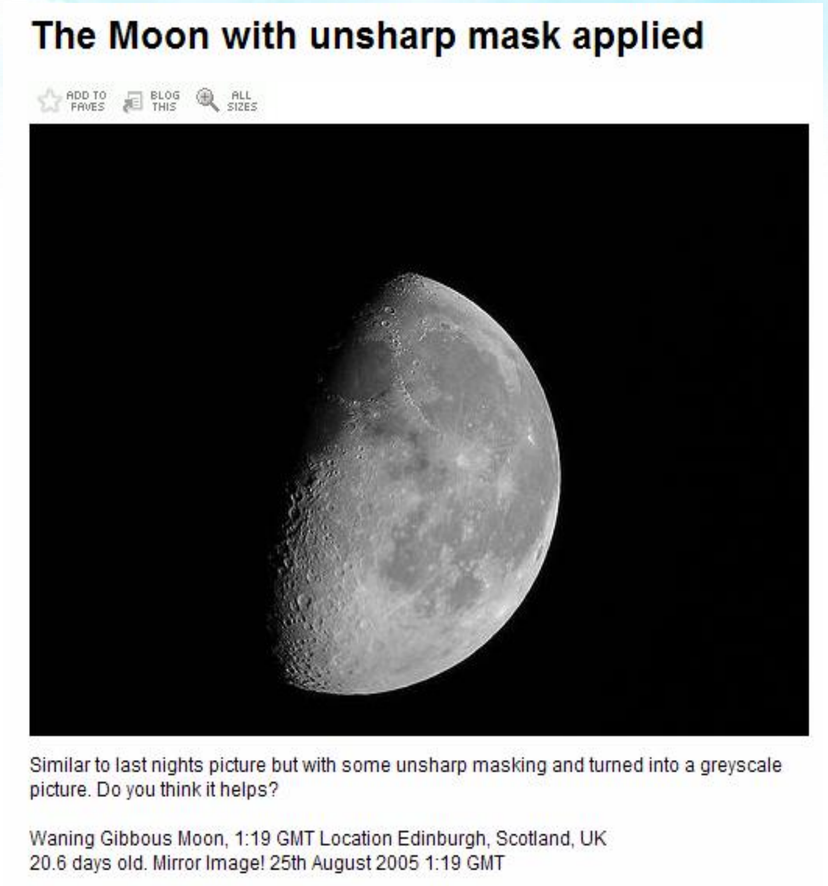
−1	−1	−1
−1	$A+8$	−1
−1	−1	−1

unsharp mask example



The Moon 25th August 2005 1:19 GMT

Waning Gibbous Moon, 1:19 GMT Location Edinburgh, Scotland, UK
20.6 days old. Mirror Image!



The Moon with unsharp mask applied

Similar to last nights picture but with some unsharp masking and turned into a greyscale picture. Do you think it helps?

Waning Gibbous Moon, 1:19 GMT Location Edinburgh, Scotland, UK
20.6 days old. Mirror Image! 25th August 2005 1:19 GMT

END

Image smoothing

- Image smoothing is the set of local pre-processing methods whose predominant use is the suppression of image noise
- Calculation of the new value is based on the averaging of brightness values in some neighborhood O.
- Smoothing poses the problem of blurring sharp edges in the image, and so we shall concentrate on smoothing methods which are edge preserving.

Averaging, statistical principles of noise suppression

- Assume that the noise value v at each pixel is an independent random variable with zero mean and standard deviation σ
- capture the same static scene under the same conditions n times.
- An estimate of the correct value can be obtained as an average of these values, with corresponding noise values v1 , . . . , vn

$$\frac{g_1 + \ldots + g_n}{n} + \frac{\nu_1 + \ldots + \nu_n}{n}$$

- The second term here describes the effect of the noise, which is again a random value with zero mean and standard deviation $\sigma/\sqrt{n}$.

Averaging, statistical principles of noise suppression

- if n images of the same scene are available, smoothing can be accomplished without blurring the image by

$$f(i,j) = \frac{1}{n}\sum_{k=1}^{n} g_k(i,j)$$

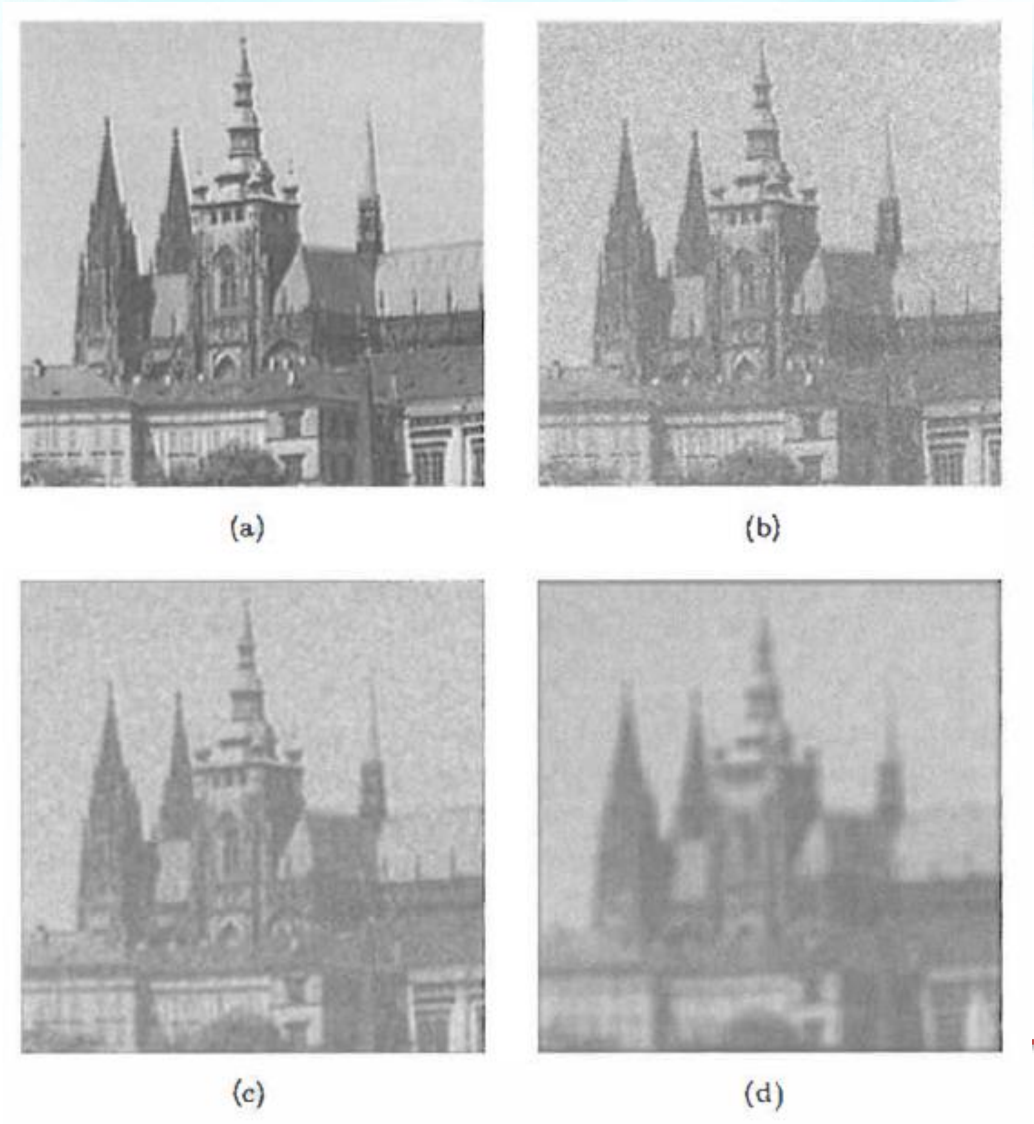
- Averaging is a special case of discrete convolution. For a 3 x 3 neighborhood, the convolution mask h is

$$h = \frac{1}{9}\begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}$$

- The significance of the pixel in the center of the convolution mask h or its 4-neighbors is sometimes increased

$$h = \frac{1}{10}\begin{bmatrix} 1 & 1 & 1 \\ 1 & 2 & 1 \\ 1 & 1 & 1 \end{bmatrix} \qquad h = \frac{1}{16}\begin{bmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{bmatrix}$$

Noise with Gaussian distribution and averaging filters

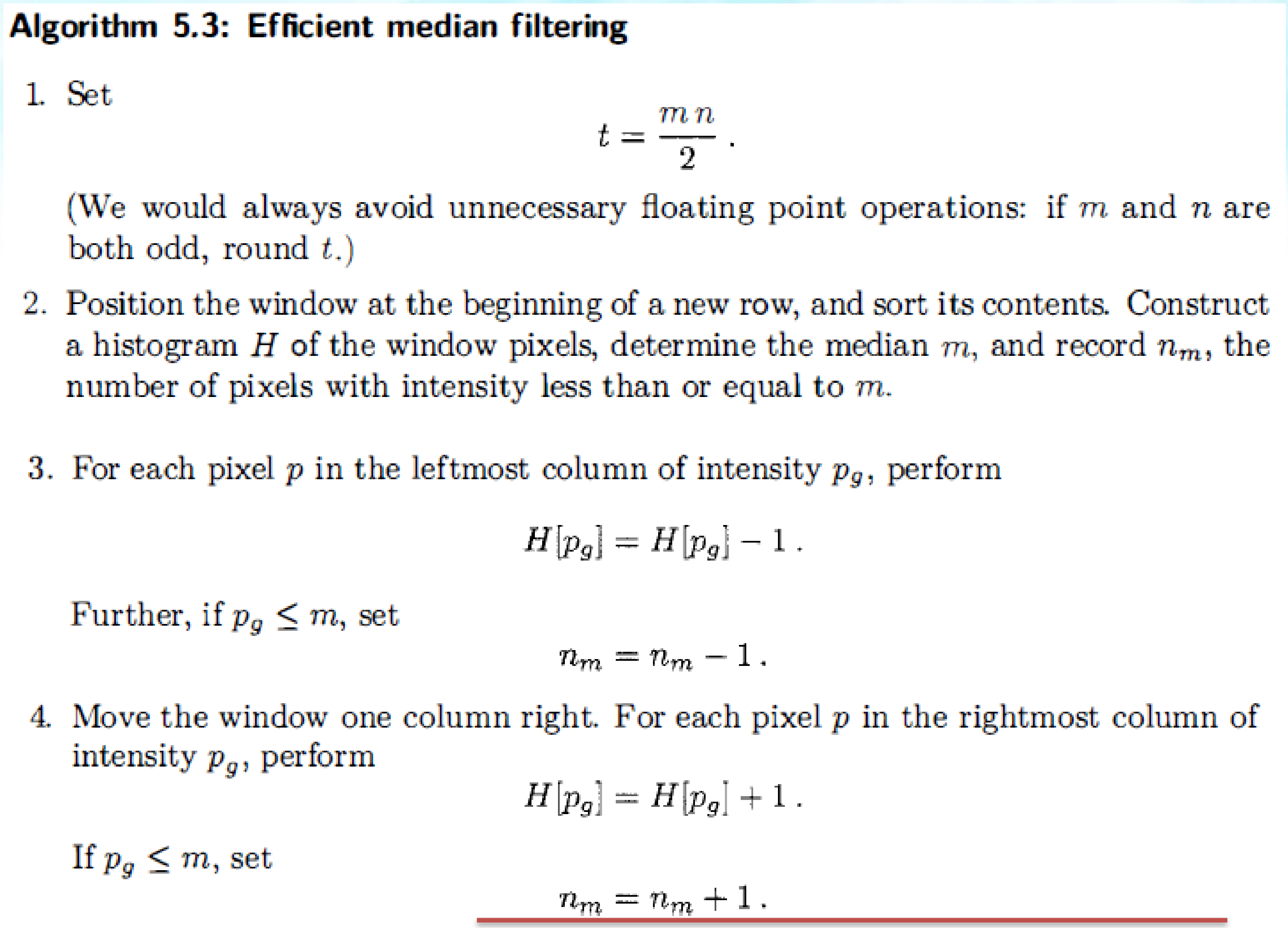


(a) Original image.
(b) Superimposed
noise (random Gaussian noise characterized by zero mean and standard deviation
equal to one-half of the gray-level standard deviation of the original image).
(c) 3 x 3 averaging.
(d) 7 x 7 averaging.

Median filtering

- The median of a finite list of real numbers can be found by ordering the value and selecting the middle one.
- Median filtering is a non-linear smoothing method that reduces the blurring of edges
- the idea is to replace the current point in the image by the median of the brightnesses in its neighborhood.
- The median of the brightnesses in the neighborhood is not affected by individual noise spikes and so median smoothing eliminates impulse noise quite well.

*Efficient median filtering **



Algorithm 5.3: Efficient median filtering

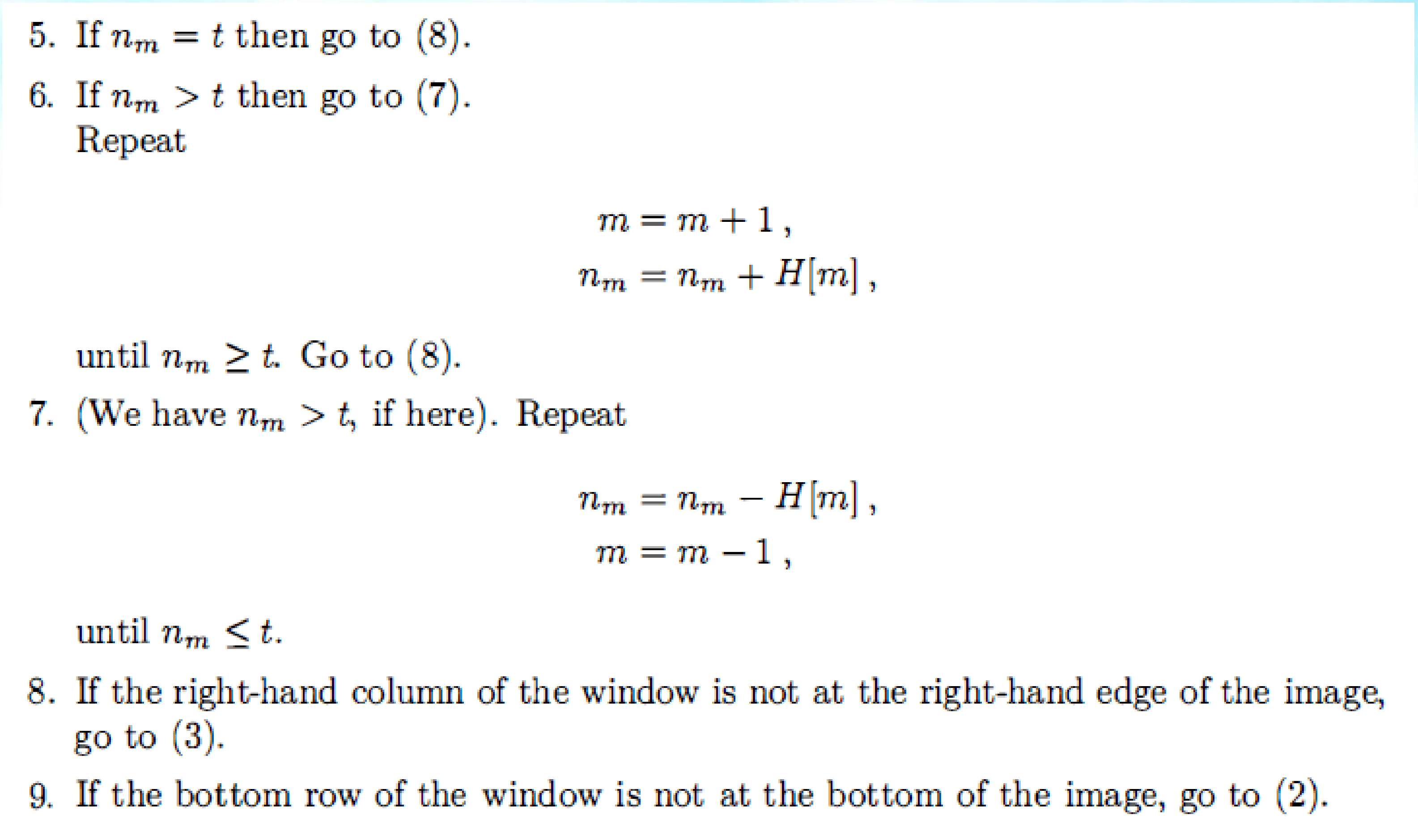
1. Set
$$t = \frac{m\,n}{2}\,.$$
(We would always avoid unnecessary floating point operations: if m and n are both odd, round t.)

2. Position the window at the beginning of a new row, and sort its contents. Construct a histogram H of the window pixels, determine the median m, and record n_m, the number of pixels with intensity less than or equal to m.

3. For each pixel p in the leftmost column of intensity p_g, perform
$$H[p_g] = H[p_g] - 1\,.$$
Further, if $p_g \leq m$, set
$$n_m = n_m - 1\,.$$

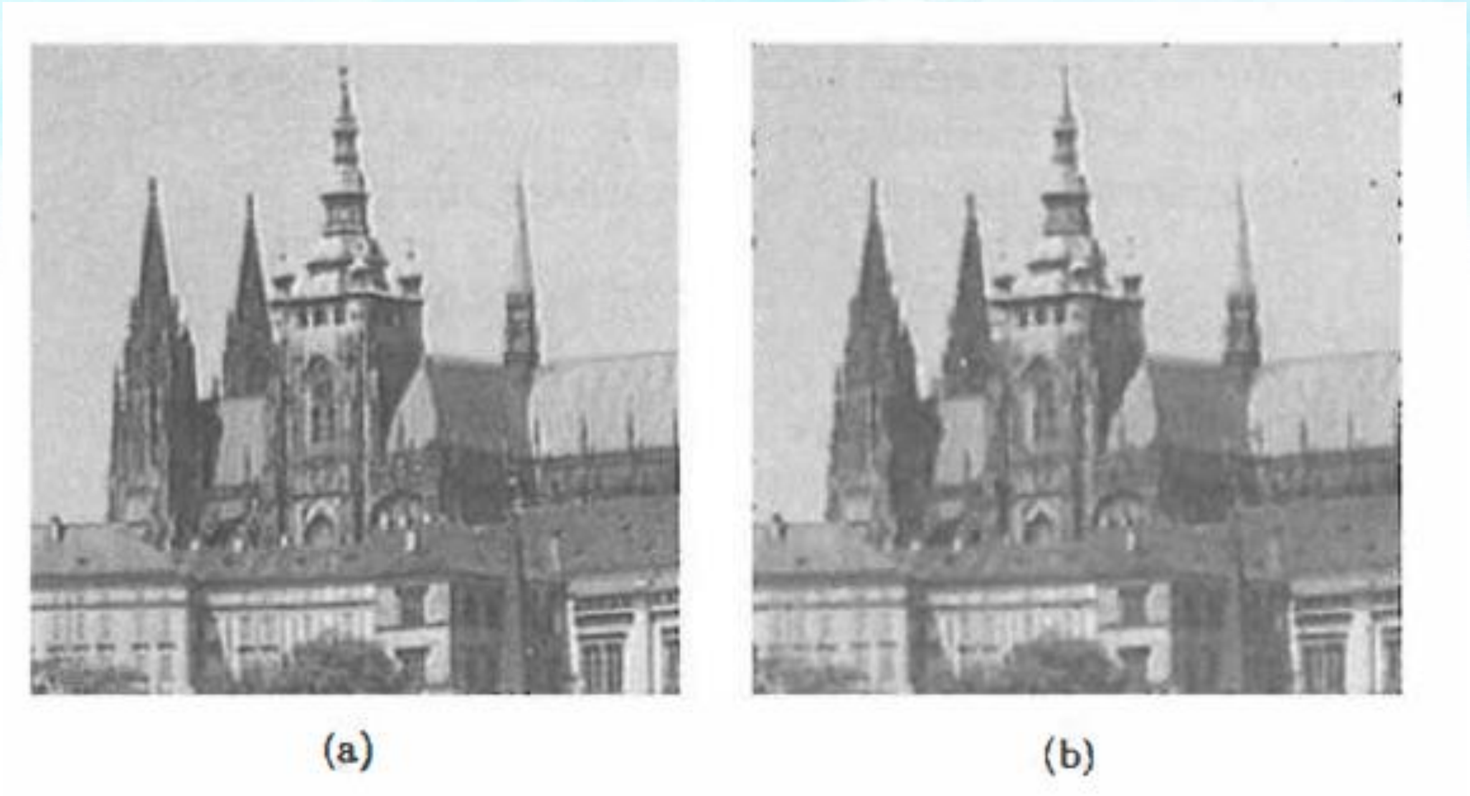
4. Move the window one column right. For each pixel p in the rightmost column of intensity p_g, perform
$$H[p_g] = H[p_g] + 1\,.$$
If $p_g \leq m$, set
$$n_m = n_m + 1\,.$$

Efficient median filtering *



5. If $n_m = t$ then go to (8).
6. If $n_m > t$ then go to (7).
   Repeat
   $$m = m + 1\,,$$
   $$n_m = n_m + H[m]\,,$$
   until $n_m \geq t$. Go to (8).
7. (We have $n_m > t$, if here). Repeat
   $$n_m = n_m - H[m]\,,$$
   $$m = m - 1\,,$$
   until $n_m \leq t$.
8. If the right-hand column of the window is not at the right-hand edge of the image, go to (3).
9. If the bottom row of the window is not at the bottom of the image, go to (2).

Median filtering



(a) Image corrupted with impulse noise (14% of image area covered with bright and dark dots). (b) Result of 3 x 3 median filtering.

Edge detectors

- Edge detectors are used to locate changes in the intensity function; edges are pixels where this function (brightness) changes abruptly.

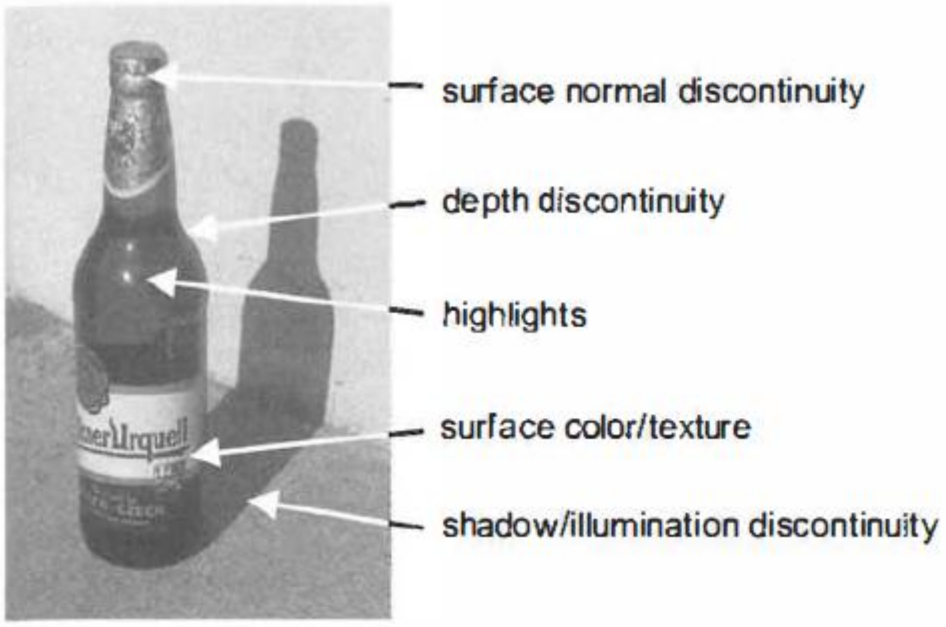

surface normal discontinuity
depth discontinuity
highlights
surface color/texture
shadow/illumination discontinuity

Figure 5.16: Origin of edges, i.e., physical phenomena in the image formation process which lead to edges in images.

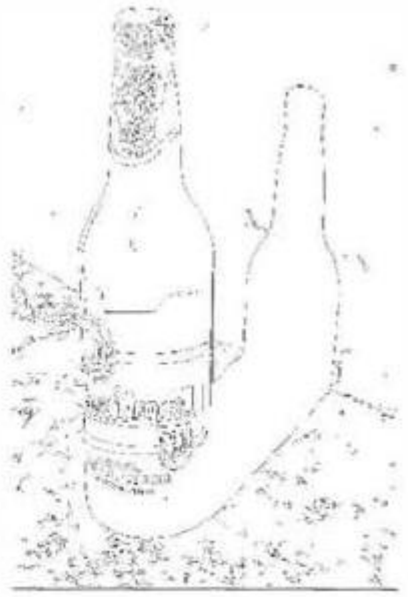


Figure 5.17: Detected edge elements.

Edge detectors

- An edge is a property attached to an individual pixel and is calculated from the image function behavior in a neighborhood of that pixel. It is a vector variable with two components, magnitude and direction.
- The edge magnitude is the magnitude of the gradient, and the edge direction ϕ is rotated with respect to the gradient direction ψ by - $90°$. The gradient direction gives the direction of maximum growth of the function.

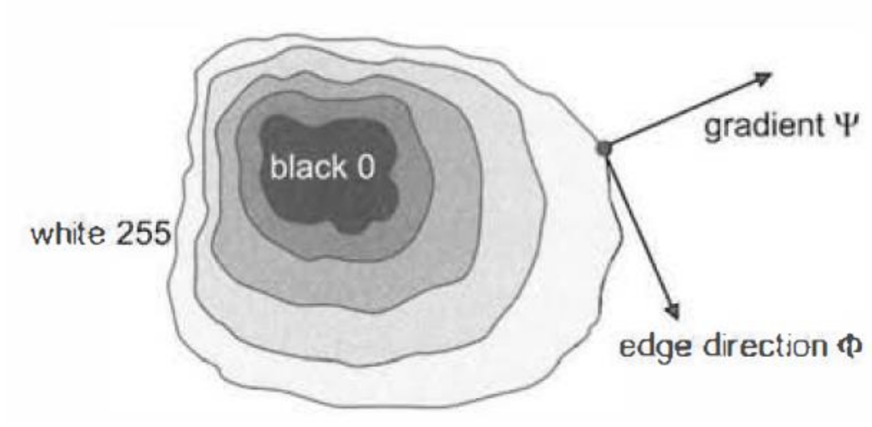


Figure 5.18: Gradient direction and edge direction.

Edge detectors

- The edge profile in the gradient direction (perpendicular to the edge direction) is typical for edges. Examples of several standard edge profiles:

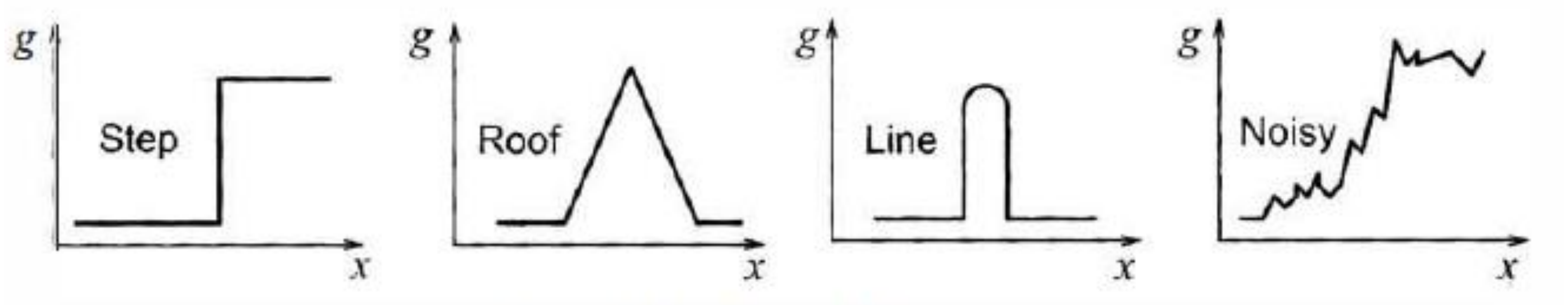


Figure 5.19: Typical edge profiles.

Edge detectors

- The gradient magnitude |grad g{x,y)| and gradient direction ψ are continuous image functions calculated as

$$|\operatorname{grad} g(x,y)| = \sqrt{\left(\frac{\partial g}{\partial x}\right)^2 + \left(\frac{\partial g}{\partial y}\right)^2}$$

$$\psi = \arg\left(\frac{\partial g}{\partial x}, \frac{\partial g}{\partial y}\right)$$

where arg{x,y) is the angle (in radians) from the x axis to the point (x,y).

- Sometimes we are interested only in edge magnitudes without regard to their orientations-a linear differential operator called the Laplacian may then be used.

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

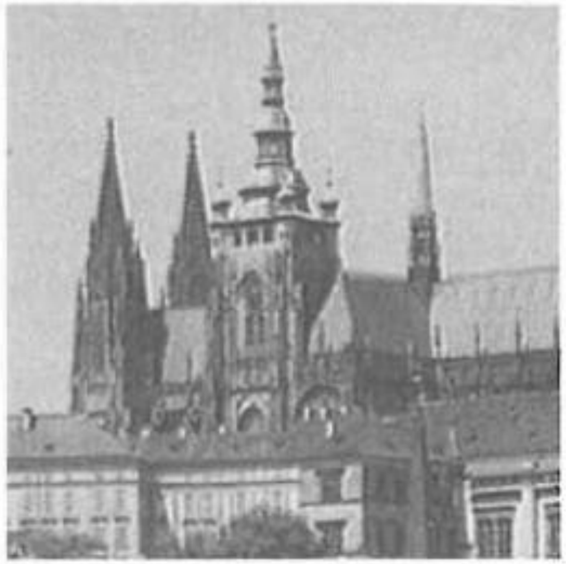
Edge detectors

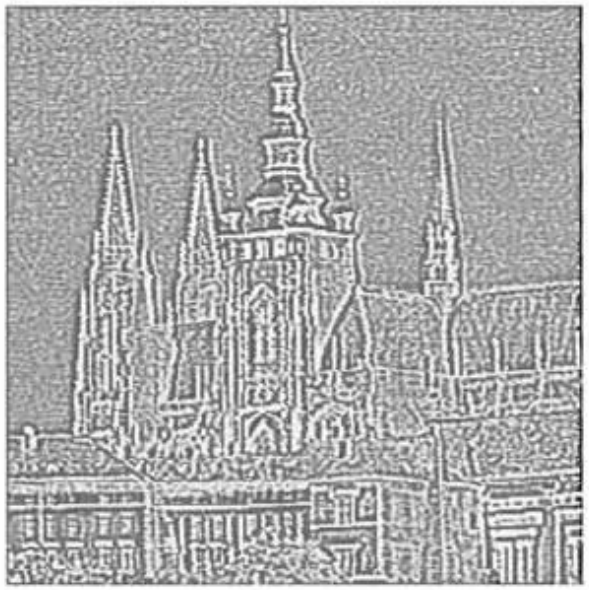
- Image sharpening has the objective of making edges steeper- the sharpened image is intended to be observed by a human. The sharpened output image f is obtained from the input image g as

$$f(i,j) = g(i,j) - C\,S(i,j)$$

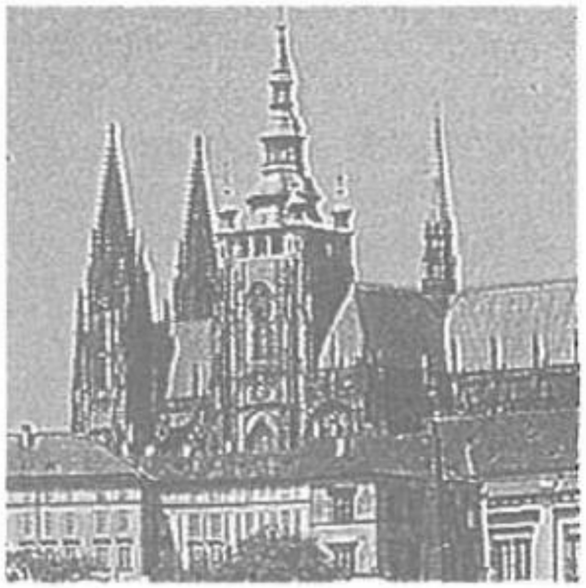
where C is a positive coefficient which gives the strength of sharpening and S(i.j) is a measure of the image function sheerness, calculated using a gradient operator (e.g., Laplacian).

Laplace gradient operator





(a)



(b)

Original image

Laplace gradient operator. (a) Laplace edge image using the 8-connectivity mask. (b) Sharpening using the Laplace operator (C = 0.7).

Edge detectors

- A digital image is discrete in nature, so gradient magnitude and gradient direction must be approximated by differences.
- The first difference of the image g in the vertical direction (for fixed i) and in the horizontal direction (for fixed j) are given by

$$\Delta_i\, g(i,j) = g(i,j) - g(i-n,j)$$
$$\Delta_j\, g(i,j) = g(i,j) - g(i,j-n)$$

where n is a small integer, usually 1 .

Edge detectors

- The Laplace operator ∇^2 is a very popular operator approximating the second derivative which gives the gradient magnitude only.
- The Laplacian, equation is approximated in digital images by a convolution sum. A 3 x 3 mask h is often used; for 4-neighborhoods and 8-neighborhoods it is defined as

$$h = \begin{bmatrix} 0 & 1 & 0 \\ 1 & -4 & 1 \\ 0 & 1 & 0 \end{bmatrix}, \qquad h = \begin{bmatrix} 1 & 1 & 1 \\ 1 & -8 & 1 \\ 1 & 1 & 1 \end{bmatrix}$$

- A Laplacian operator with stressed significance of the central pixel or its neighborhood is sometimes used.

$$h = \begin{bmatrix} 2 & -1 & 2 \\ -1 & -4 & -1 \\ 2 & -1 & 2 \end{bmatrix}, \qquad h = \begin{bmatrix} -1 & 2 & -1 \\ 2 & -4 & 2 \\ -1 & 2 & -1 \end{bmatrix}$$

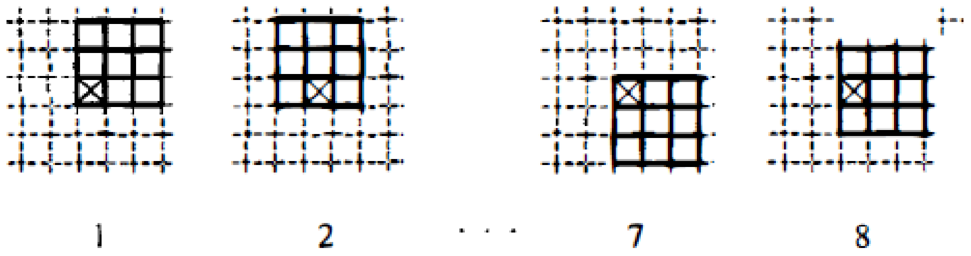
Averaging using a rotating mask

- Averaging using a rotating mask is a non-linear smoothing method that avoids edge blurring by searching for the homogeneous part of the current pixel neighborhood and the resulting image is in fact sharpened
- The brightness average is calculated only within this region
- A brightness dispersion σ^2 is used as the region homogeneity measure.
- Let n be the number of pixels in a region R and g be the input image. Dispersion σ^2 is calculated as

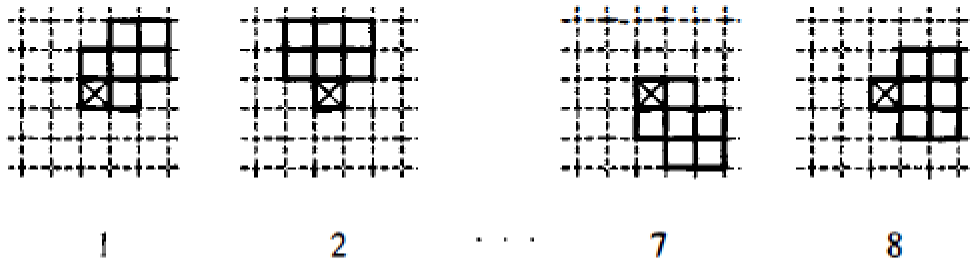
$$\sigma^2 = \frac{1}{n} \sum_{(i,j)\in R} \left(g(i,j) - \frac{1}{n} \sum_{(i,j)\in R} g(i,j) \right)^2 \qquad (5.29)$$

Averaging using a rotating mask

- Having computed region homogeneity, we consider its shape and size.
  - The eight possible 3 x 3 masks that cover a 5 x 5 neighborhood of a current pixel (marked by the small cross). The ninth mask is the 3 x 3 neighborhood of the current pixel itself.

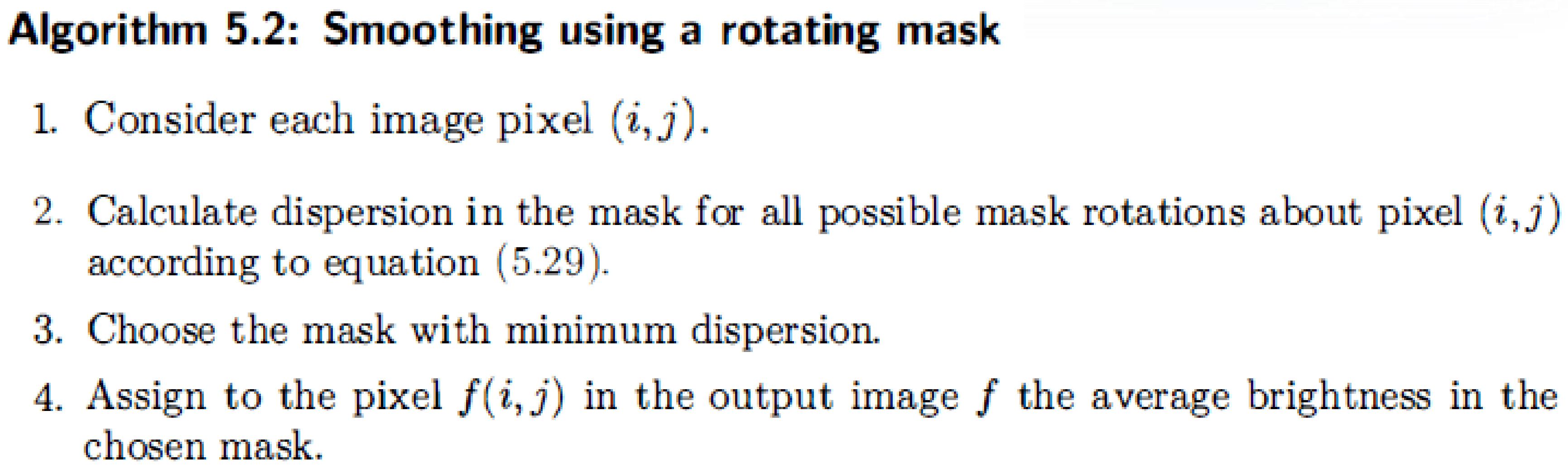


  - Another set of eight masks covering a 5 x 5 neighborhood of the current pixel. The ninth mask is the 3 x 3 neighborhood of the current pixel itself.



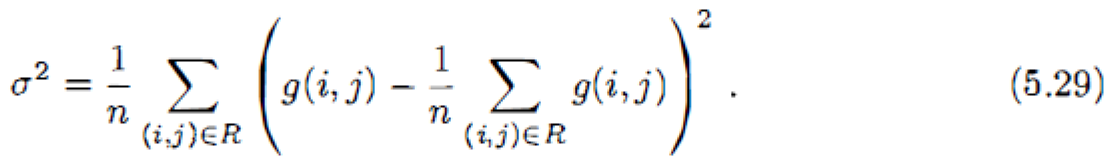
Averaging using a rotating mask

- Image smoothing using the rotating mask technique uses the following algorithm.



Algorithm 5.2: Smoothing using a rotating mask

1. Consider each image pixel (i, j).
2. Calculate dispersion in the mask for all possible mask rotations about pixel (i, j) according to equation (5.29).
3. Choose the mask with minimum dispersion.
4. Assign to the pixel $f(i, j)$ in the output image f the average brightness in the chosen mask.



$$\sigma^2 = \frac{1}{n} \sum_{(i,j)\in R} \left(g(i,j) - \frac{1}{n} \sum_{(i,j)\in R} g(i,j) \right)^2 . \qquad (5.29)$$