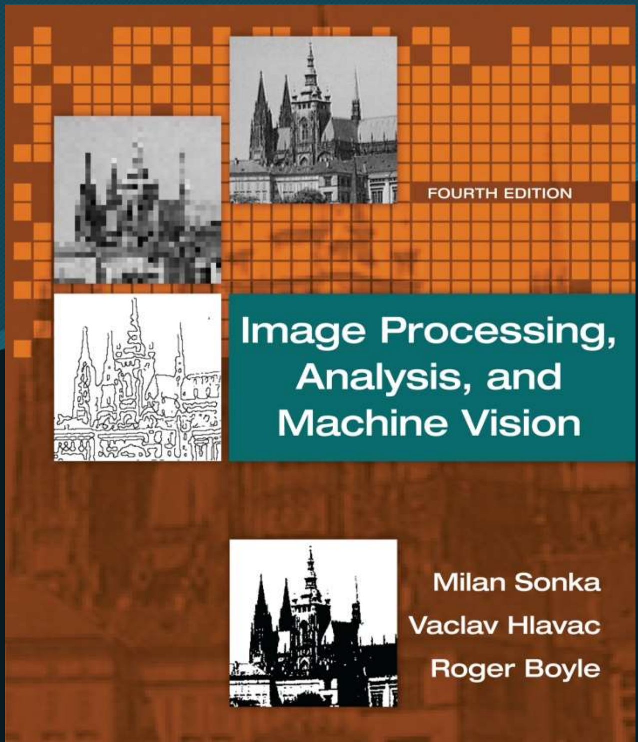
# Chapter 5

## Image pre-processing



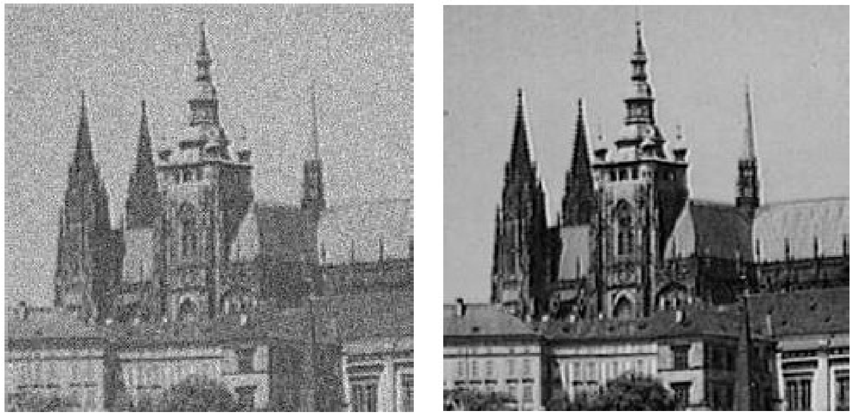

CENGAGE Learning

# Image pre-processing

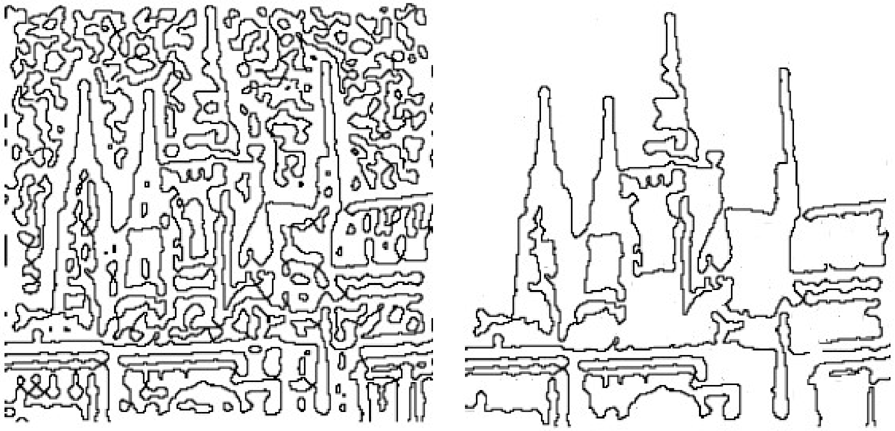
- Pixel brightness transformations
- Geometric transformations
- Local pre-processing
  - Image smoothing, edge detection, line detection, corner detection, and region detection.
- Image restoration

- **Objectives of image pre-processing**

(a) Enhancing information that is useful for later analysis



(b) Suppress image information that is not relevant to later work



# Pixel brightness transformations

- Position-dependent brightness correction
  - The sensitivity of image acquisition and digitization devises may depend on position in the image.
  - Uneven object illumination (不均勻的光源) is also a source of degradation (退化).
  - If degradation is systematic (有條理的), it can be suppressed (抑制) by brightness correction.
  - A multiplicative error coefficient $e(i, j)$ describes the change from the ideal.
  - Assume $g(i, j)$ is the original undegraded image and $f(i, j)$ is degraded version.

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

# Examples



http://portfolio.veronika-by.com.ua/raw-to-jpeg-with-brigthness-correction/

# Pixel brightness transformations

- Brightness correction
  - The error coefficient $e(i,j)$ can be obtained if a reference image $g(i,j)$ with known brightnesses is captures.
  - The simplest method to detect the error coefficient is to use an image $g'$ with constant brightness $c$.
  - The degraded result is the image $f_c(i,j)$.

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

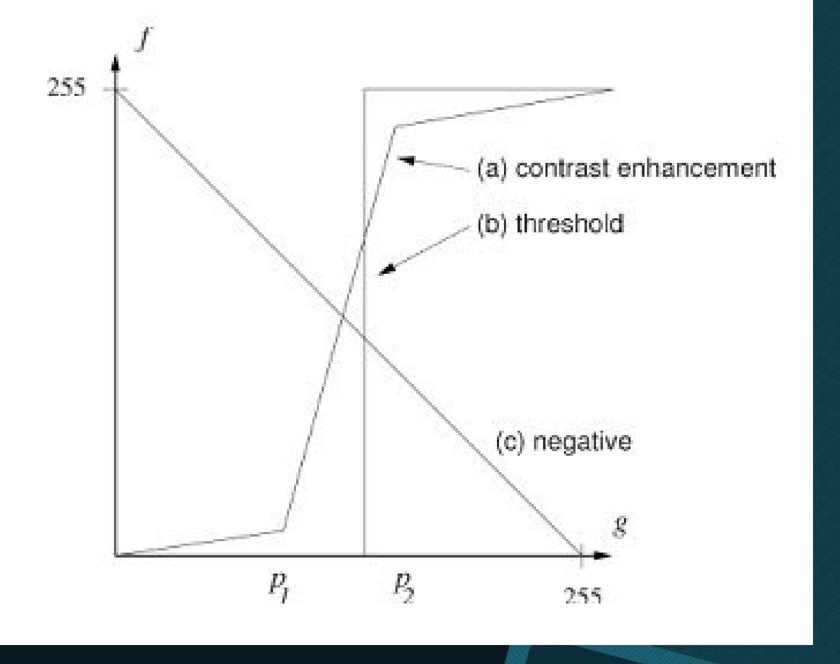
  - The systematic brightness errors can be suppressed by

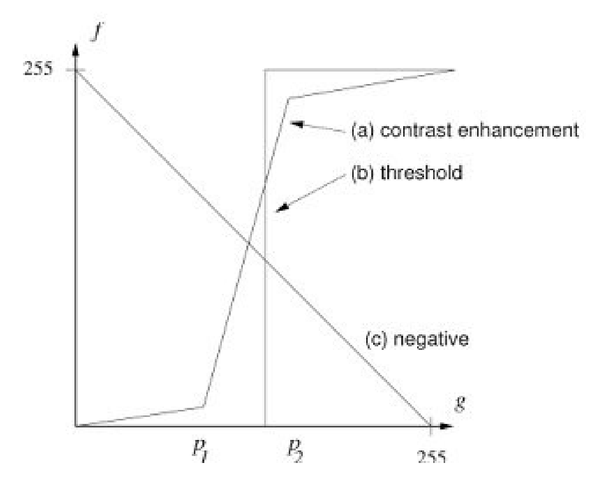
$$g(i,j) = \frac{f(i,j)}{e(i,j)} = \frac{cf(i,j)}{f_c(i,j)}$$

  - The method can be used only if the image degradation process is stable.

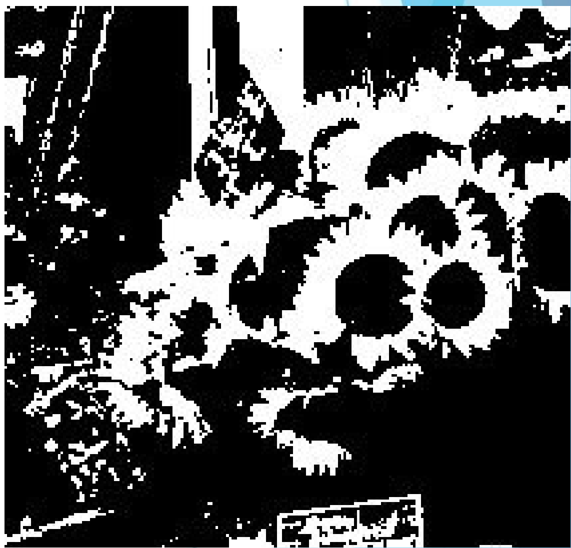
# Pixel brightness transformations

- Gray-scale transformation
  - Gray-scale transformations do not depend on the position of the pixel in the image.
  - A transformation $\mathcal{T}$ of original brightness $p$ from scale $[p_0, p_k]$ into brightness $q$ from a new scale $[q_0, q_k]$ is given by
    $$q = \mathcal{T}(p)$$
  - The figure shows the most common gray-scale transformations.



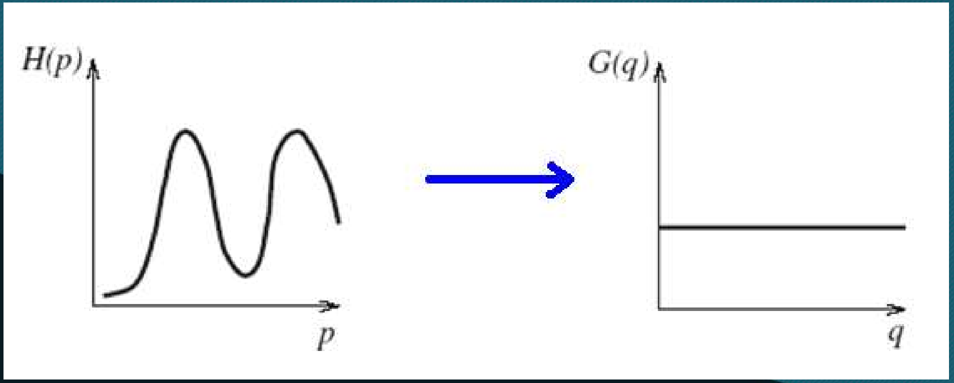


f
255
(a) contrast enhancement
(b) threshold
(c) negative
g
p1
p2
255

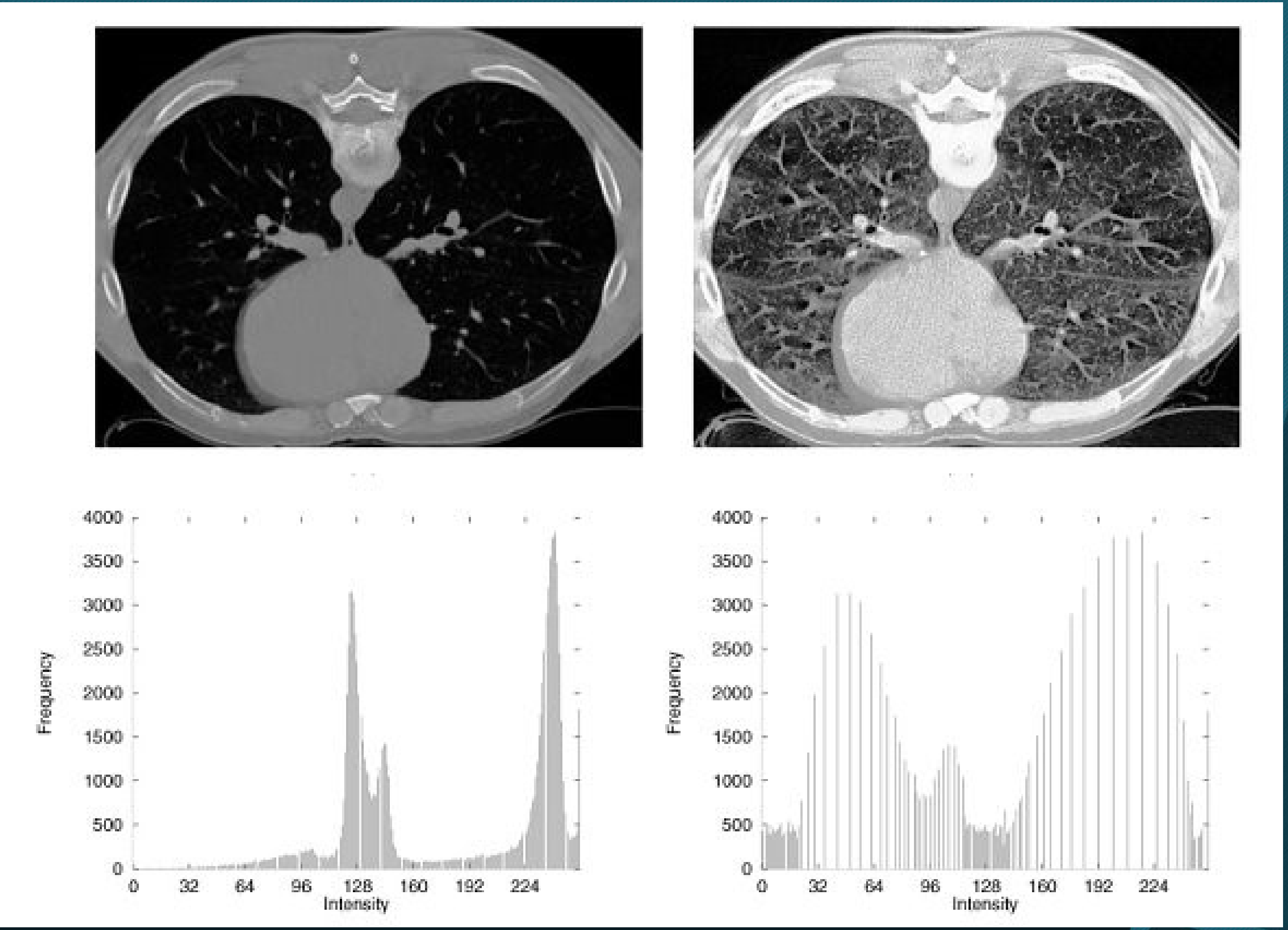



# Gray-scale transformation

- Histogram equalization
  - 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.



# Histogram equalization



# Histogram equalization

- Let $H(p)$ be the input histogram, its gray-scale is $[p_0, p_k]$,
  $G(q)$ be the desired output histogram, its gray-scale is $[q_0, q_k]$.

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

- The monotonic property of the transform $\mathcal{T}$ implies

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

- The equalized histogram $G(q)$ corresponds to the uniform probability density function $f$ whose function value is constant

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

for an $N \times N$ image.

# Histogram equalization

- For the 'idealized' continuous probability density, from

$$\sum_{i=0}^{k} G(q_i) = \sum_{i=0}^{k} H(p_i) \qquad \text{and} \qquad f = \frac{N^2}{q_k - q_0}$$

we can drive

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

from which $\mathcal{T}$ can be derived as

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

The integral in this equation is called the cumulative histogram.

## Algorithm 5.1 Histogram equalization

1. For an $N \times M$ image of $G$ gray-levels, initialize an array $H$ of length $G$ to 0.
2. From the image histogram: Scan every pixel $p$ – if it has intensity $g_p$, perform

$$H[g_p] = H[g_p] + 1$$

   Then let $g_{min}$ be the minimum $g$ for which $H[g] > 0$.

3. Form the cumulative image histogram $H_c$:

$$H_c[0] = H[0]$$
$$H_c[g] = H_c[g-1] + H[g], \qquad g = 1,2,\dots,G-1$$

   Let $H_{min} = H_c[g_{min}]$.

4. Set

$$T[g] = \text{round}\left(\frac{H_c[g] - H_{min}}{MN - H_{min}}(G-1)\right)$$

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

$$g_q = T[g_p]$$

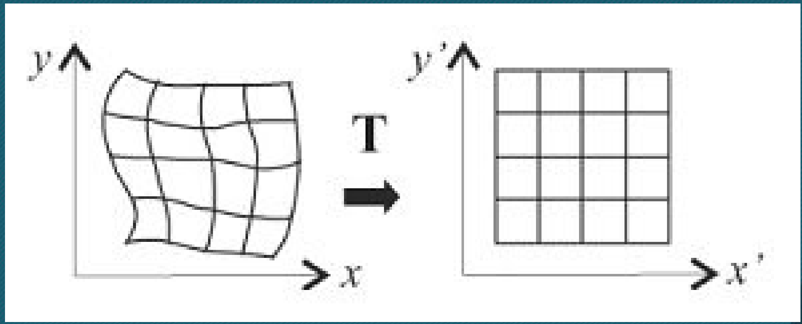
# Geometric transformations

- Geometric transformations permit elimination of the geometric distortion that occurs when an image is captured.
- A geometric transform is a vector function T that maps the pixel $(x, y)$ to a new position $(x', y')$.

$$\mathrm{T} = (T_x, T_y)$$

$$x' = T_x(x, y)$$

$$y' = T_y(x, y)$$

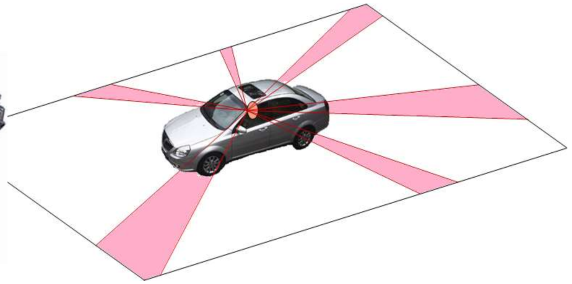



- Two basic steps of a geometric transform
  - First: pixel co-ordinate transformation
  - Second: brightness interpolation

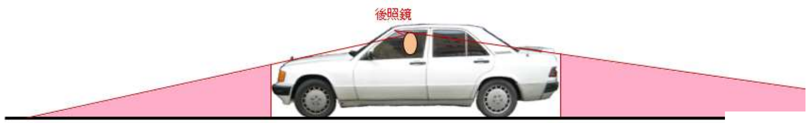
# Application: Bird's-View Image Generation

Blind areas around a vehicle

Window pillars

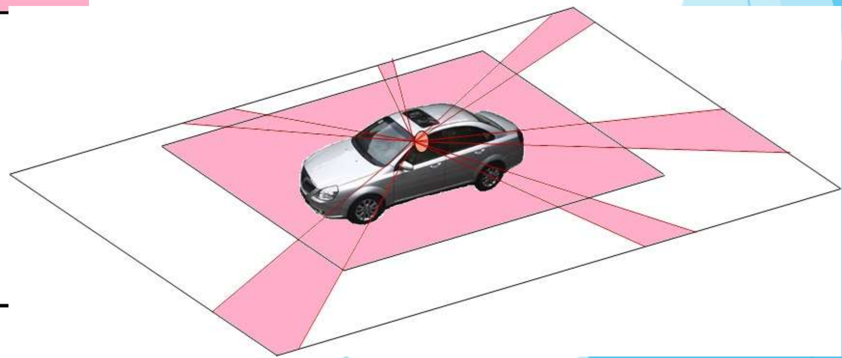



Height of vehicle

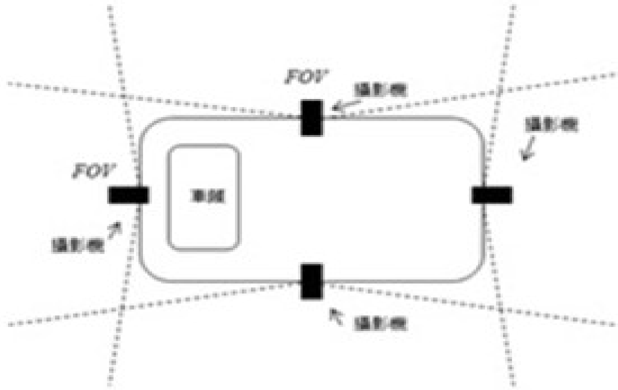



Summary of blind areas

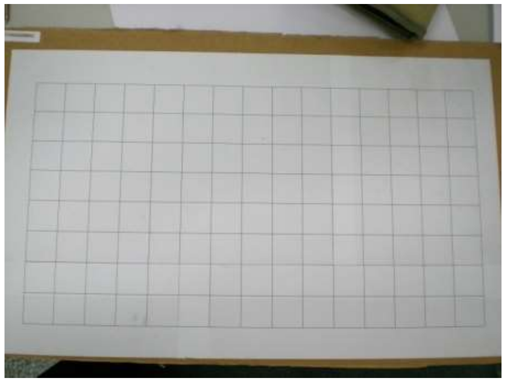
Driver's position



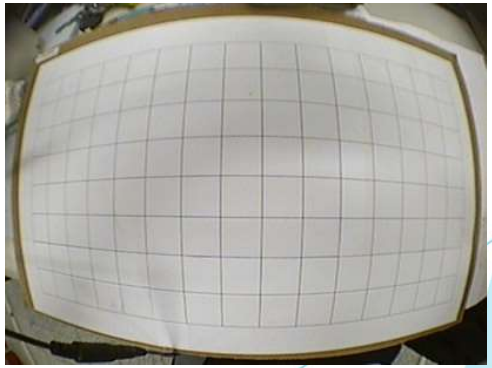
# System Configuration




Fish-eye camera:



Scene



Image


Front
Left
video
Back
Right

# Pixel co-ordinate transformation

- Equations $x' = T_x(x, y), y' = T_y(x, y)$ can be approximated by a polynomial equations.

$$x' = T_x(x, y) = \sum_{r=0}^{m} \sum_{k=0}^{m-r} a_{rk} x^r y^k$$

$$y' = T_y(x, y) = \sum_{r=0}^{m} \sum_{k=0}^{m-r} b_{rk} x^r y^k$$

  - If pairs of corresponding points $(x, y), (x', y')$ in both images are known, it is possible to determine $a_{rk}, b_{rk}$ by solving a set of linear equations.

# Pixel co-ordinate transformation

- Bilinear transform

$$x' = a_0 + a_1 x + a_2 y + a_3 xy$$
$$y' = b_0 + b_1 x + b_2 y + b_3 xy$$

  - The geometric transform can be approximated by a linear transform.

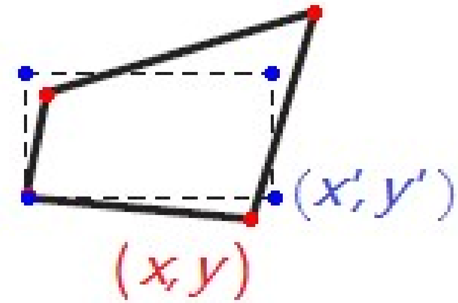
- Affine transform

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

  - The affine transformation includes typical geometric transformations such as rotation, translation, scaling, and skewing.

Example: Bilinear transform

$$x' = a_0 + a_1 x + a_2 y + a_3 xy$$
$$y' = b_0 + b_1 x + b_2 y + b_3 xy$$




Needs at least 4 pairs of corresponding points to determine the parameters $a_0, a_1, a_2, a_3, b_0, b_1, b_2, b_3$

$$((x_1, y_1), (x_1', y_1')), \quad ((x_2, y_2), (x_2', y_2'))$$
$$((x_3, y_3), (x_3', y_3')), \quad ((x_4, y_4), (x_4', y_4'))$$

$$x_1' = a_0 + a_1 x_1 + a_2 y_1 + a_3 x_1 y_1, \; y_1' = b_0 + b_1 x_1 + b_2 y_1 + b_3 x_1 y_1$$
$$x_2' = a_0 + a_1 x_2 + a_2 y_2 + a_3 x_2 y_2, \; y_2' = b_0 + b_2 x_2 + b_2 y_2 + b_3 x_2 y_2$$
$$x_3' = a_0 + a_1 x_3 + a_2 y_3 + a_3 x_3 y_3, \; y_3' = b_0 + b_1 x_3 + b_2 y_3 + b_3 x_3 y_3$$
$$x_4' = a_0 + a_1 x_4 + a_2 y_4 + a_3 x_4 y_4, \; y_4' = b_0 + b_1 x_4 + b_2 y_4 + b_3 x_4 y_4$$

$$
\begin{bmatrix}
1 & x_1 & y_1 & x_1 y_1 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 1 & x_1 & y_1 & x_1 y_1 \\
1 & x_2 & y_2 & x_2 y_2 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 1 & x_2 & y_2 & x_2 y_2 \\
1 & x_3 & y_3 & x_3 y_3 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 1 & x_3 & y_3 & x_3 y_3 \\
1 & x_4 & y_4 & x_4 y_4 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 1 & x_4 & y_4 & x_4 y_4
\end{bmatrix}
\begin{bmatrix} a_0 \\ a_1 \\ a_2 \\ a_3 \\ b_0 \\ b_1 \\ b_2 \\ b_3 \end{bmatrix}
=
\begin{bmatrix} x_1' \\ y_1' \\ x_2' \\ y_2' \\ x_3' \\ y_3' \\ x_4' \\ y_4' \end{bmatrix}
$$

$$A\boldsymbol{x} = \boldsymbol{b}$$

Solve $\boldsymbol{x}$ by the least square error method.

# Pixel co-ordinate transformation

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

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

  - If the transformation is singular (has no inverse), then $J = 0$.
  - If the area of the image is invariant under the transformation, then $J = 1$.

# Pixel co-ordinate transformation

- The Jacobia determinant for the bilinear transform

$$x' = a_0 + a_1 x + a_2 y + a_3 xy$$
$$y' = b_0 + b_1 x + b_2 y + b_3 xy$$
$$J = \begin{vmatrix} \frac{\partial x'}{\partial x} & \frac{\partial x'}{\partial y} \\ \frac{\partial y'}{\partial x} & \frac{\partial y'}{\partial y} \end{vmatrix} = \begin{vmatrix} a_1 + a_3 y & a_2 + a_3 x \\ b_1 + b_3 y & b_2 + b_3 x \end{vmatrix}$$
$$= (a_1 + a_3 y)(b_2 + b_3 x) - (b_1 + b_3 y)(a_2 + a_3 x)$$
$$= a_1 b_2 - a_2 b_1 + (a_1 b_3 - a_3 b_1)x + (a_3 b_2 - a_2 b_3)y$$

- The Jacobia determinant for the affine transform

$$J = a_1 b_2 - a_2 b_1$$

# Pixel co-ordinate transformation

- Some important geometric transformations are:
  - Rotation by the angle $\phi$ about the origin:

$$x' = \quad x\cos\phi + y\sin\phi$$

$$y' = -x\sin\phi + y\cos\phi$$

$$J = 1$$

  - Change of scale $a$ in the $x$ axis and $b$ in the $y$ axis:

$$x' = ax$$

$$y' = by$$

$$J = ab$$

  - Skewing by the angle $\phi$, given by:

$$x' = x + y\tan\phi$$

$$y' = y$$

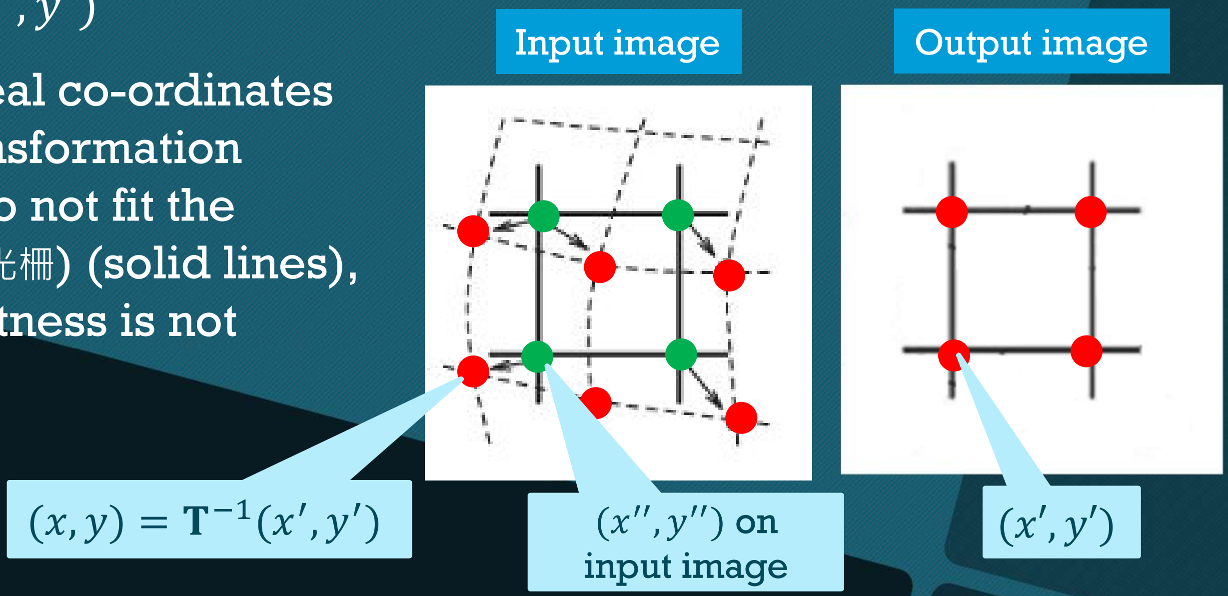
$$J = 1$$

# Brightness interpolation

- After applying a pixel co-ordinate transformation
  - Assume we wish to compute the brightness values of the pixel $(x', y')$ in the output image (integer numbers, illustrated by solid lines in figures).
  - The co-ordinates of the point $(x, y)$ in the original image can be obtained by inverting the planar transformation.

$$(x, y) = \mathbf{T}^{-1}(x', y')$$

  - In general, the real co-ordinates after inverse transformation (dashed lines) do not fit the discrete raster (光柵) (solid lines), and so the brightness is not known.



# Brightness interpolation

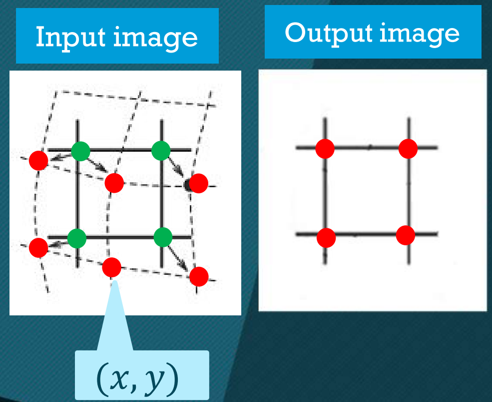
- How to calculate the brightness value of $(x, y)$?
  - Denote the result of the brightness interpolation by $f_n(x, y)$, where $n$ distinguished different interpolation methods.
  - The brightness can be expressed by the convolution equation

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

$h_n$: the interpolation kernel function

$g_s(l\Delta x, k\Delta y)$: brightness value, the sampled version of the originally continuous image function $f(x, y)$

let $\Delta x = \Delta y = 1$ for simplification

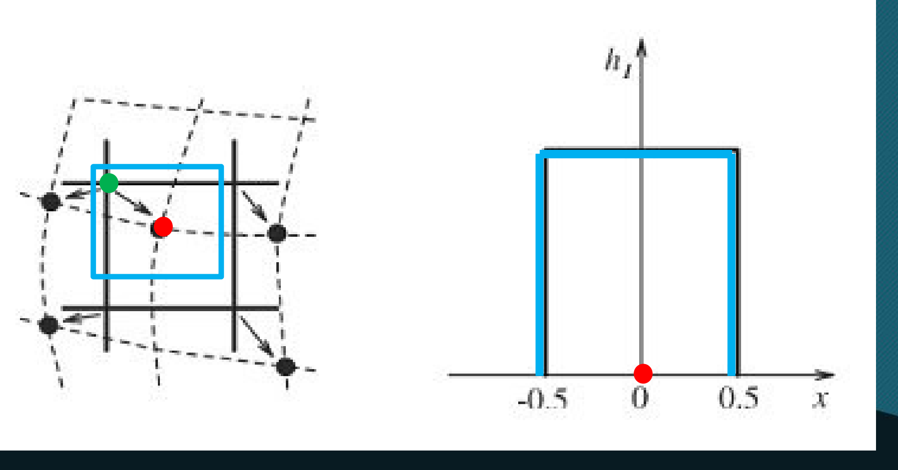


# Brightness interpolation

- Three most common brightness interpolation methods are **nearest neighbor, linear** and **bi-cubic.**
- **Nearest-neighborhood interpolation**

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

  - The position error of nearest-neighborhood interpolation is at most half a pixel.



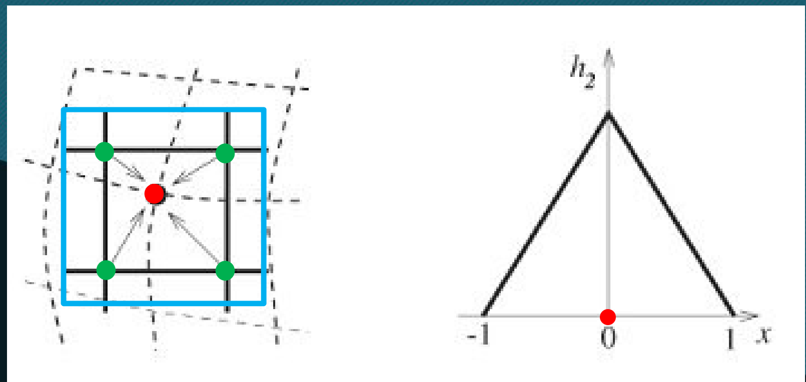
# Brightness interpolation

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

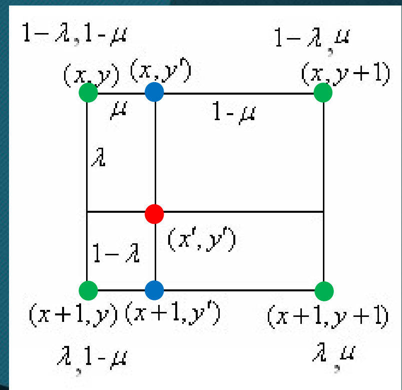
$$f_2(x,y) = (1-a)(1-b)g_s(l,k) + a(1-b)g_s(l+1,k)$$
$$+b(1-a)g_s(l,k+1) + ab\,g_s(l+1,k+1)$$

where

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



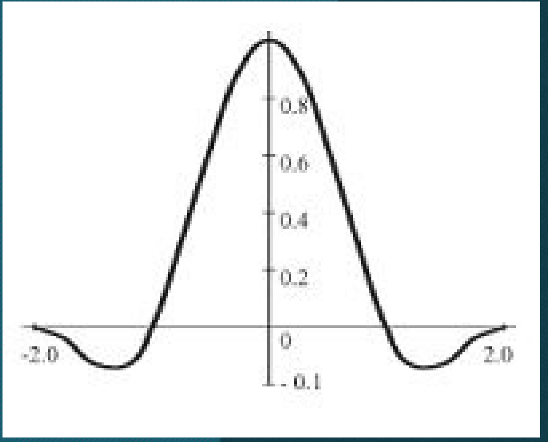

https://en.wikipedia.org/wiki/Bilinear_interpolation



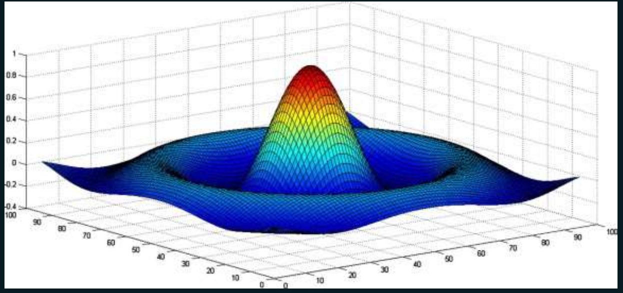
# Brightness interpolation

- Bi-cubic interpolation
  - Bi-cubic interpolation improves the model of the brightness function by approximating it locally by a bi-cubic polynomial surface, 16 neighboring points are used for interpolation.
  - The 1D interpolation kernel (Mexican hat) is given by

$$h_3 = \begin{cases} 1 - 2|x|^2 + |x|^3 & \text{for } 0 \le |x| < 1 \\ 4 - 8|x| + 5|x|^2 - |x|^3 & \text{for } 1 \le |x| < 2 \\ 0 & \text{otherwise} \end{cases}$$




  - Bi-cubic interpolation preserves fine details in the image very well.



https://www.researchgate.net/figure/262569642_fig1_An-example-of-a-Mexican-hat-wavelet-function

# Local pre-processing

- Local pre-processing methods are divided into two groups according to the goal of the processing.
  - Smoothing
    - To suppress noise or other small fluctuations (變動) in the image
  - Gradient operators
    - To indicate the locations with bigger local derivatives (導數) in the image

PS: 導數: 一個函數在某一點的導數描述了這個函數在這一點附近的變化率。
https://zh.wikipedia.org/wiki/%E5%AF%BC%E6%95%B0

- Another classification of local pre-processing methods is according to the transformation properties
  - Linear transformations
  - Non-linear transformations

# Local pre-processing

- Linear transformations
  - Linear operations calculate the resulting value in the output image pixel $f(i,j)$ as a linear combination of brightness values in a local neighborhood $\mathcal{O}$ of the pixel $g(i,j)$ in the input image.
  - The contribution of the pixels in the neighborhood $\mathcal{O}$ is weighted by coefficients $h$.

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

  - The kernel $h$ is called a convolution mask.

# Image smoothing

- Image smoothing
  - Image smoothing uses redundancy (多餘;重複) in image data to suppress noise, usually by some form of averaging of brightness values in some neighborhood $\mathcal{O}$.
  - Goal: suppressing noise
  - Problem: blurring sharp edges
  - Solution: edge preserving
    - Usually using non-linear methods
    - For example: computing the average only from points in the neighborhood which have similar properties to the point being processed.

# Image smoothing

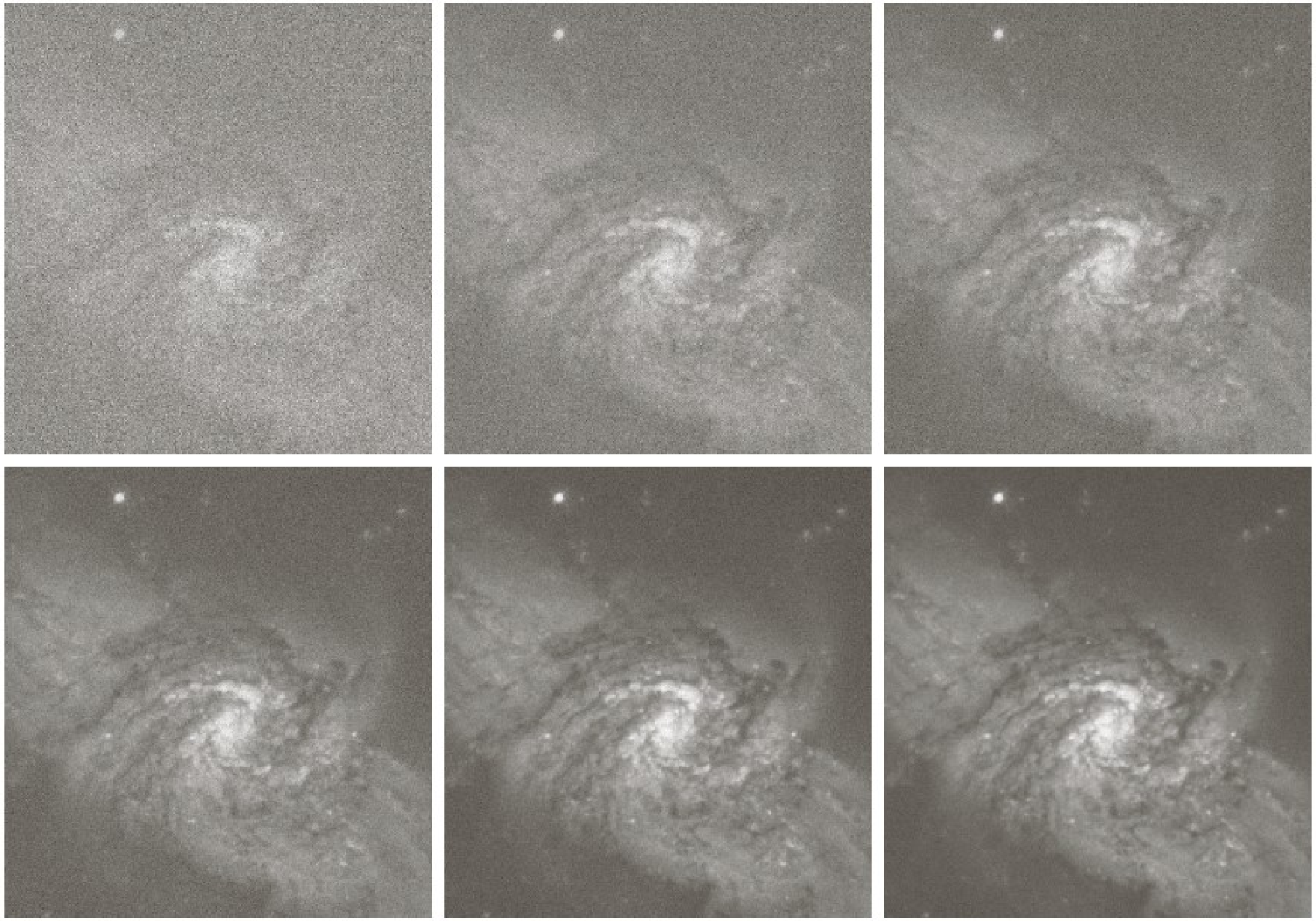
- Averaging, statistical principles of noise suppression
  - Assume that the noise value $\nu$ at each pixel is an independent random variable with zero mean and standard deviation $\sigma$.
  - We might capture the same static scene under the same conditions $n$ times and from each captured image a particular pixel value $g_i, i = 1, \dots, n$ is selected.
  - An estimate of the correct value can be obtained as an average of theses values.

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

    - The second term describes the noise with zero mean and standard deviation $\frac{\sigma}{\sqrt{n}}$.
  - Thus, if $n$ images of the same scene are available,

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

# Noise reduction (for addition noise)



a b c
d e f

**FIGURE 2.26** (a) Image of Galaxy Pair NGC 3314 corrupted by additive Gaussian noise. (b)–(f) Results of averaging 5, 10, 20, 50, and 100 noisy images, respectively. (Original image courtesy of NASA.)

# Image smoothing

- Averaging, statistical principles of noise suppression
  - Usually, only one noise corrupted is available, and averaging is then performed in a local neighborhood.
  - Averaging is a special case of discrete convolution.
    - For a $3 \times 3$ neighborhood, the convolution mask $h$ is

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

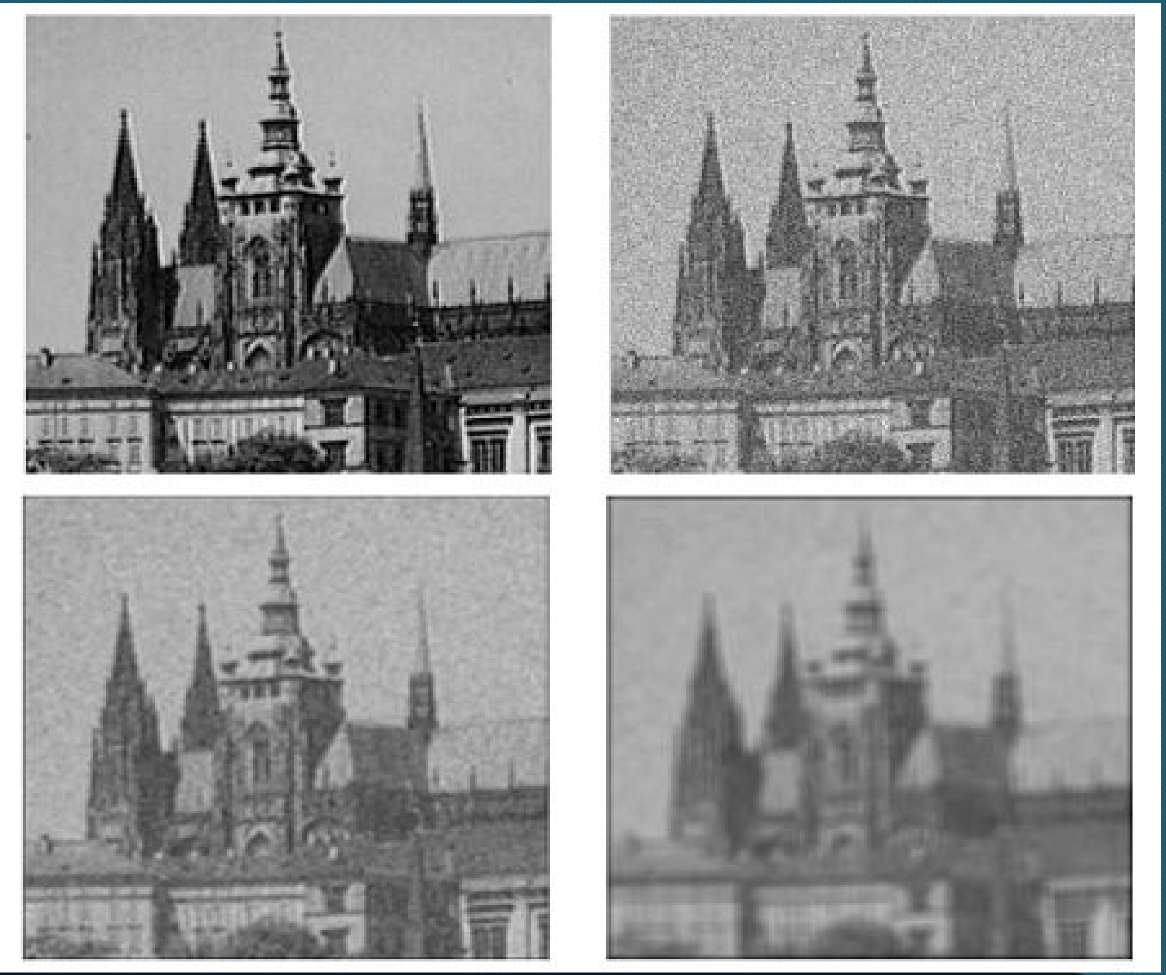
  - A Gaussian probability distribution: better approximations

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

# Image smoothing

Original image

Gaussian noise



3 × 3 averaging

7 × 7 averaging

# Image smoothing

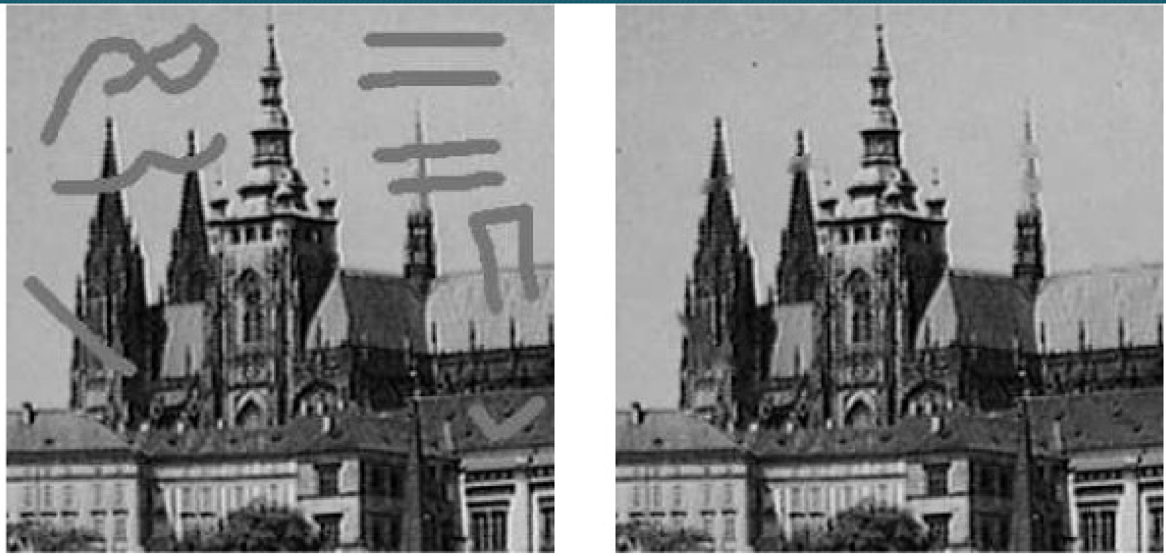
- Averaging with limited data validity
  - Try to avoid blurring by averaging
  - A simple criterion is to define a brightness interval of invalid data $[\min, \max]$ (typically corresponding to noise of known image faults), and apply image averaging only to pixels in that interval.
    - For a point $(m, n)$, the convolution mask is calculated in the neighborhood $\mathcal{O}$ by the non-linear formula

$$h(i,j) = \begin{cases} 1 & \text{for } g(m+i, n+j) \notin [\min, \max] \\ 0 & \text{otherwise} \end{cases}$$

where $(i, j)$ specify the mask element.

# Image smoothing

- An example of averaging with limited data validity
  - apply image averaging only to pixels on the noises



(a) (b)

**Figure 5.10**: Averaging with limited data validity. (a) Original corrupted image. (b) Result of corruption removal. © *Cengage Learning 2015.*

# Image smoothing

- Averaging according to inverse gradient
  - Within a convolution mask of odd size, the inverse gradient $\delta$ of a point $(i, j)$ with respect to the central pixel $(m, n)$ is defined as

$$\delta(i, j) = \frac{1}{|g(m, n) - g(i, j)|}$$

    - If $g(m, n) = g(i, j)$, then we define $\delta(i, j) = 2$.
    - So $\delta(i, j) \in (0,2]$ and is smaller at the edge than in the interior of a homogeneous region.
    - The kernel function $h$

$$h(i, j) = 0.5 \frac{\delta(i, j)}{\sum_{(m,n)\in\mathcal{O}} \delta(m, n)}$$

Moreover, the mask coefficient corresponding to the central pixel is defined as $h(i, j) = 0.5$.

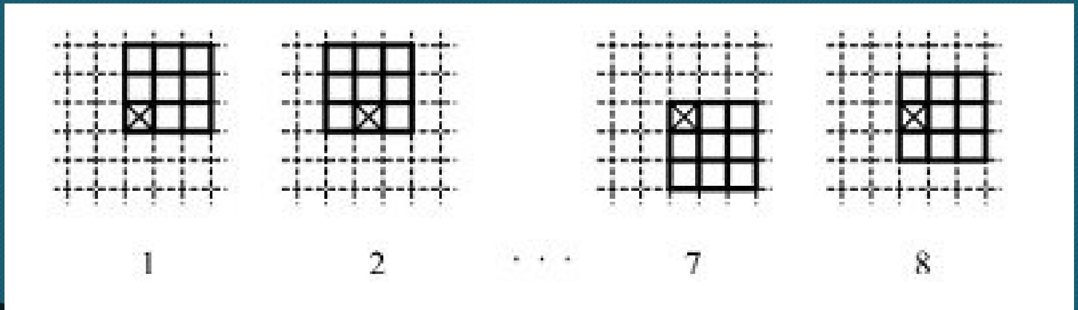
# Image smoothing

- Averaging using a rotating mask
  - A non-linear method that avoids edge blurring
  - The neighborhood of the current pixel is divided into two subsets
    - One set consists of all pixels neighboring the current pixel which satisfy the homogeneity criterion (具同質性)
    - The second set is the complement (不具同質性)
  - The brightness average is calculated only within homogeneity region which is measured by a brightness dispersion $\sigma^2$.
  - Let $n$ be a number of pixels in a region $R$ and $g$ be the input image.

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

# Image smoothing

- Averaging using a rotating mask
  - The shape and size of masks to compute region homogeneity
    - For example, eight possible rotated $3 \times 3$ makes that cover a $5 \times 5$ neighborhood of a current pixel.

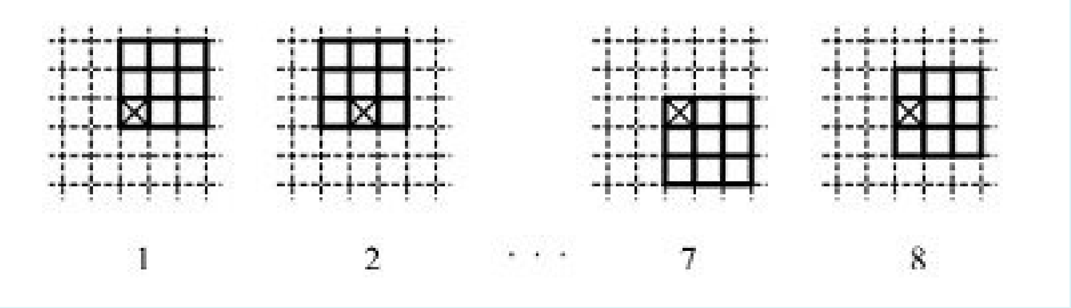


## Algorithm 5.2 Smoothing using a rotating mask

1. Consider each image pixel $(i, j)$.

2. Calculate dispersion for all possible mask rotations about pixel $(i, j)$ according to

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




3. Choose the mask with minimum dispersion (分散).

4. Assign to the pixel $f(i, j)$ in the output image $f$ the average brightness in the chosen mask.

# Image smoothing

- Median filtering
  - In probability theory, the median divides the higher half of a probability distribution from the lower half.
  - Median filter is a non-linear smoothing method.

Algorithm 5.3 Efficient median filtering

1. Set $t = \frac{mn}{2}$

   ($m, n$ are the numbers of rows and columns of the median window and 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 $p_g < m$, set $n_m = n_m - 1$.

## Algorithm 5.3 Efficient median filtering (con.)

4. Move the window one column right. For each pixel $p$ in the rightmost column of intensity $p_g$, perform

$$H[p_g] = H[p_g] + 1$$

Further, if $p_g < m$, set $n_m = n_m + 1$.

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).

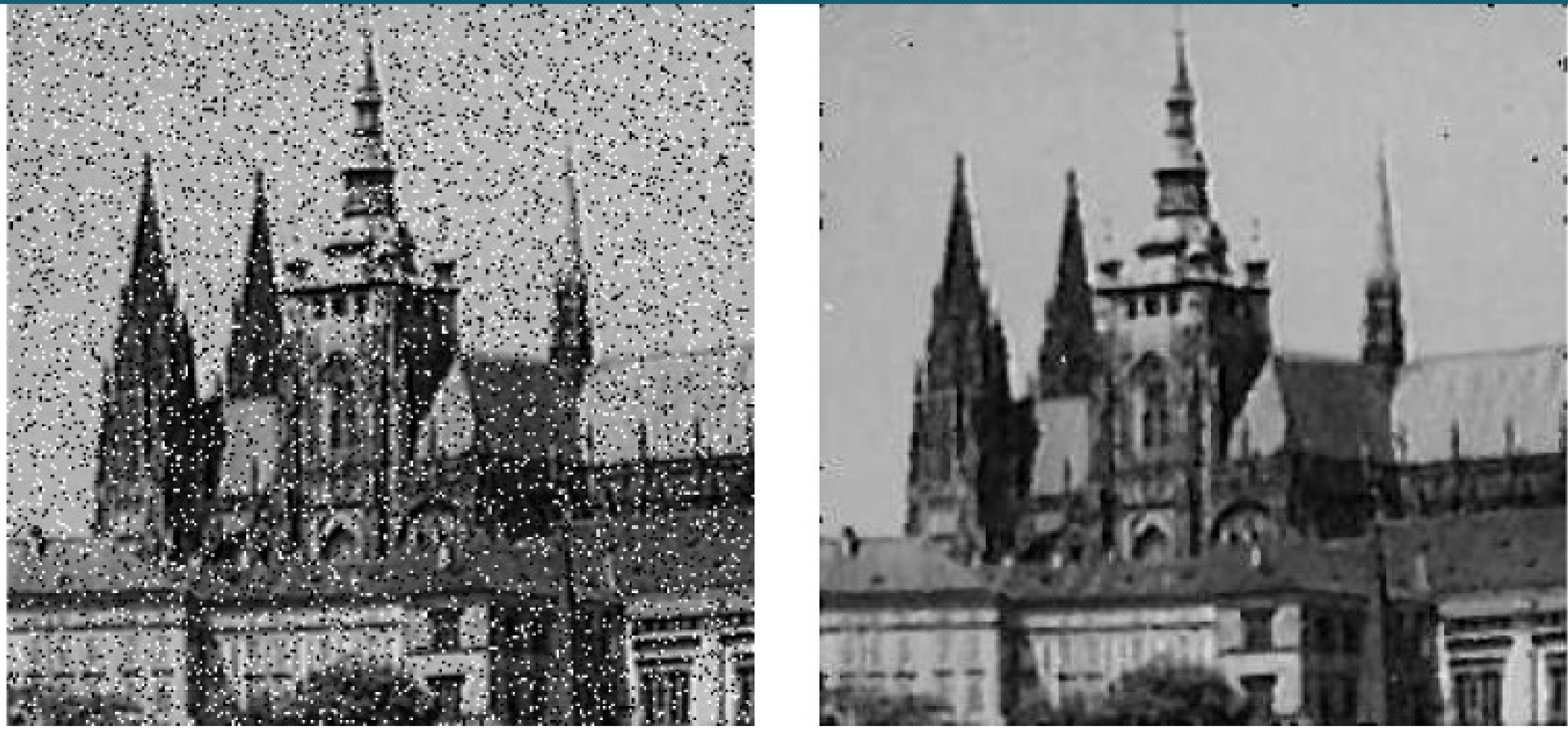
7. (We have $n_m > t$, if here)

   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).

# Image smoothing

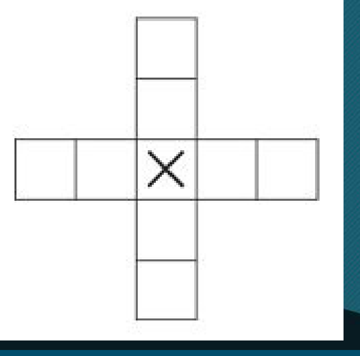


(a) (b)

**Figure 5.12:** Median filtering. (a) Image corrupted with impulse noise (14% of image area covered with bright and dark dots). (b) Result of $3 \times 3$ median filtering. © *Cengage Learning 2015.*

# Image smoothing

- Median filtering
  - The main disadvantage of median filtering in a rectangular neighborhood is its damaging of the thin lines and sharp corners.
  - This can be avoided if another shape of neighborhood is used.
  - For example, if horizontal/vertical lines need preserving, a neighborhood such as that in the figure can be used.



Horizontal/vertical line preserving neighborhood for median filtering

# Image smoothing

- Non-linear mean filter
  - A generalization of averaging technique

$$f(m,n) = u^{-1}\left(\frac{\sum_{(i,j)\in\mathcal{O}} a(i,j)u(g(i,j))}{\sum_{(i,j)\in\mathcal{O}} a(i,j)}\right)$$

where $f(m,n)$: the result of the filtering

$g(i,j)$: the pixel in the input image

$\mathcal{O}$: a local neighborhood of the current pixel $(m,n)$

$u^{-1}$: an inverse function of one variable function $u$

$a(i,j)$: weight coefficient

  - If the weights $a(i,j)$ are constant, the filter is called homomorphic.

# Image smoothing

- Non-linear mean filter
  - Some homomorphic filters
    - Arithmetic mean

$$u(g) = g$$

    - Harmonic mean

$$u(g) = \frac{1}{g}$$

    - Geometric mean

$$u(g) = \log g$$

$$u(g) = g; \qquad u^{-1}(g) = g$$

$$f(m,n) = u^{-1}\left(\frac{\sum_{(i,j)\in\mathcal{O}} a(i,j)u\big(g(i,j)\big)}{\sum_{(i,j)\in\mathcal{O}} a(i,j)}\right) = \frac{\sum_{(i,j)\in\mathcal{O}} a(i,j)u\big(g(i,j)\big)}{\sum_{(i,j)\in\mathcal{O}} a(i,j)} = \frac{\sum_{(i,j)\in\mathcal{O}} a(i,j)\, g(i,j)}{\sum_{(i,j)\in\mathcal{O}} a(i,j)}$$