Computer Vision

Image data analysis: Mathematical Tools

Tien-Lam Pham

PIAS
VIỆN NGHIÊN CỨU
TIÊN TIẾN PHENIKAA

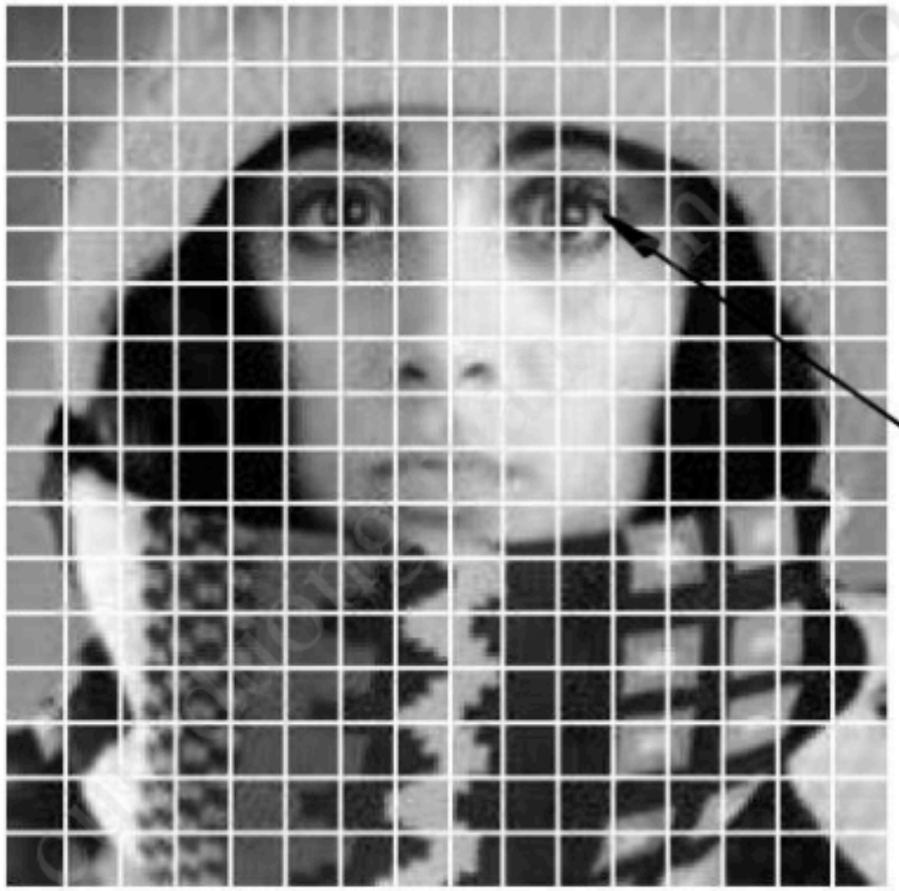
PHENIKAA
UNIVERSITY

Contents

- Image resolution: spatial and Intensity
- Image interpolation
- Linear vs. Non-linear operations
- Arithmetic Operations
- Spatial Operations
- Image Transforms

Digital image

- Mỗi pixel chứa một màu (hoặc mức xám)



$$f(x, y) = \begin{bmatrix} f(0,0) & f(0,1) & \cdots & f(0,N-1) \\ f(1,0) & f(1,1) & \cdots & f(1,N-1) \\ \vdots & \vdots & & \vdots \\ f(M-1,0) & f(M-1,1) & \cdots & f(M-1,N-1) \end{bmatrix}$$

Image Processing

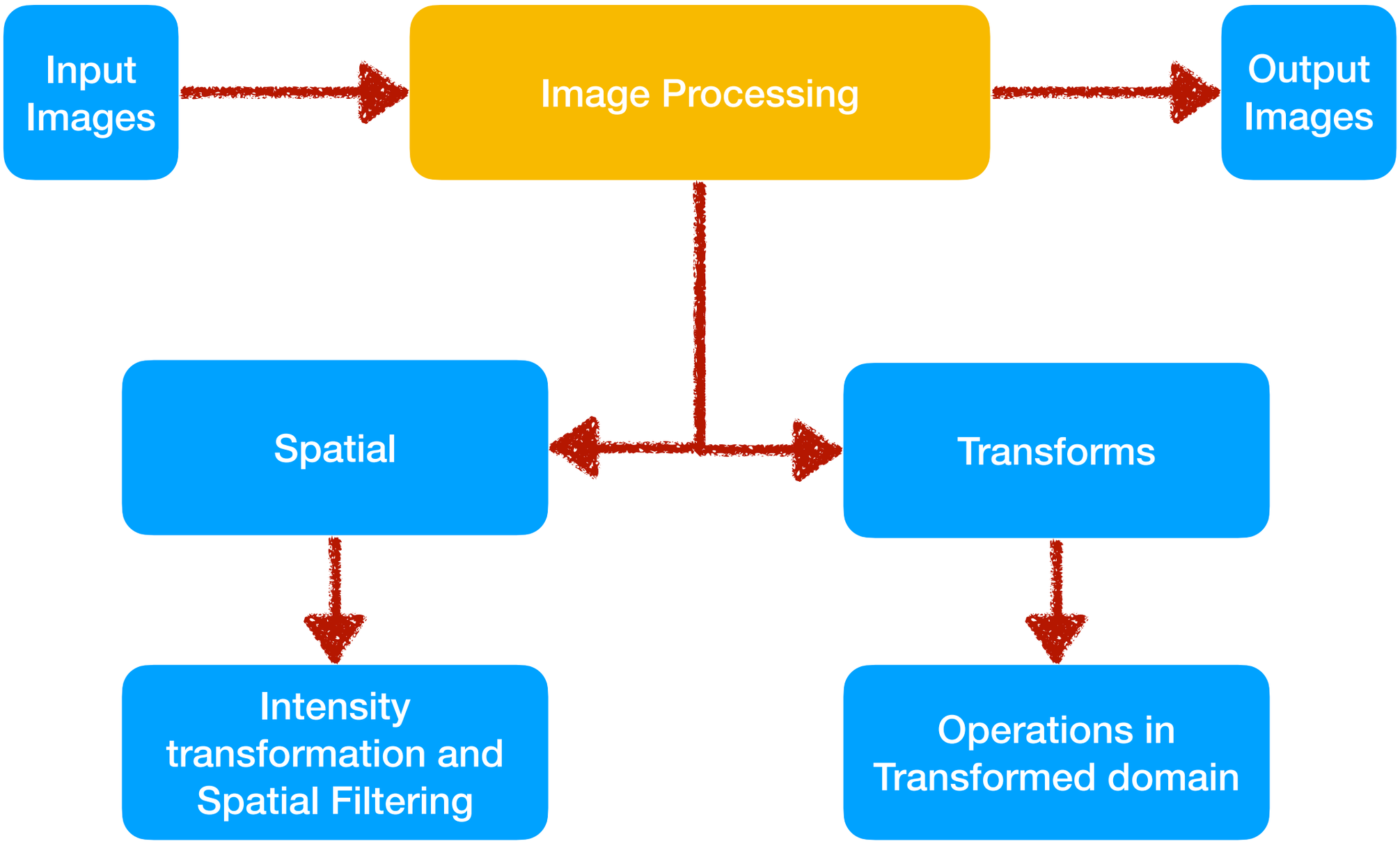


Image Interpolation

- How to zoom, shrink, rotate and correct images

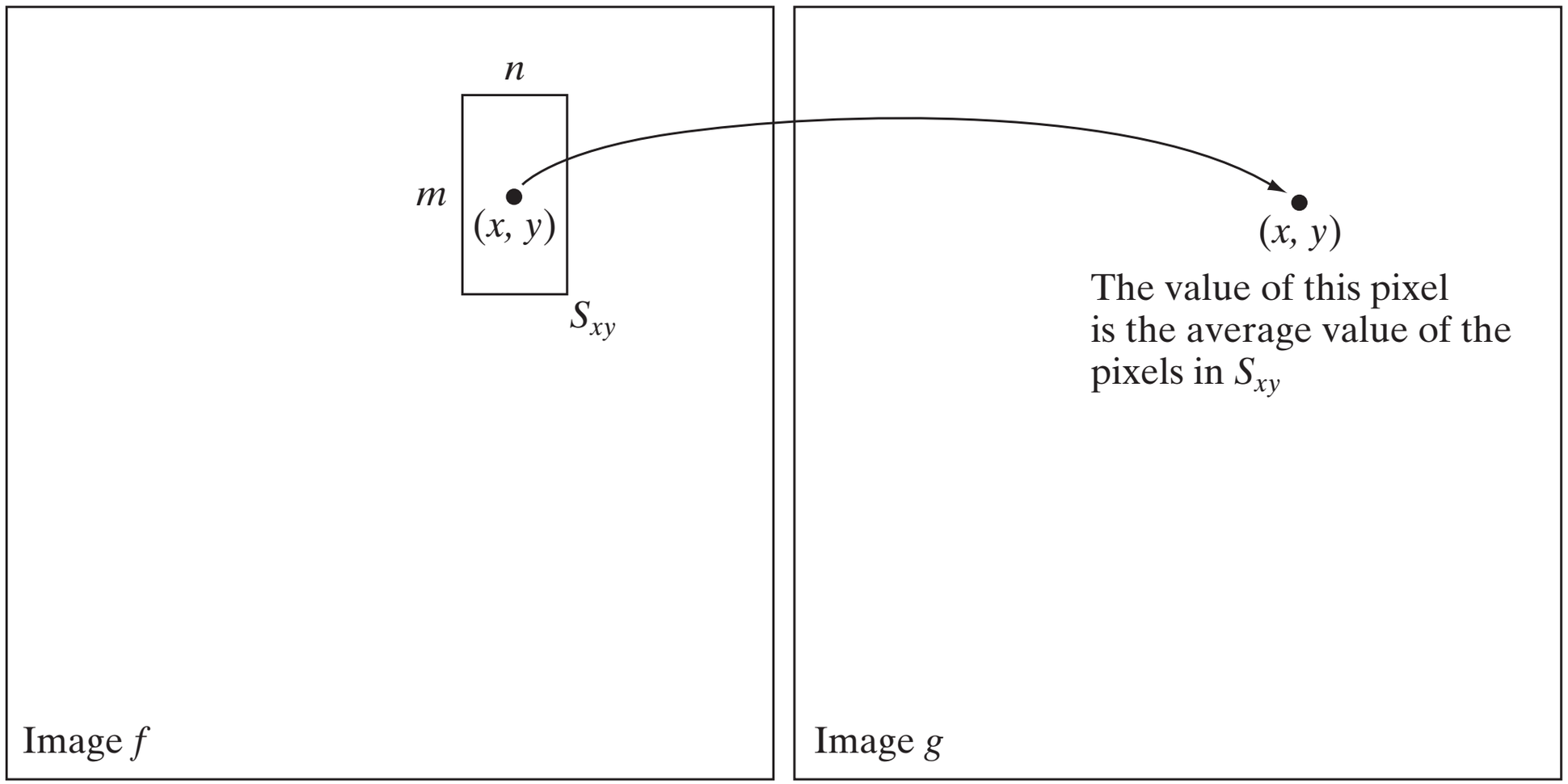


Image Zoom

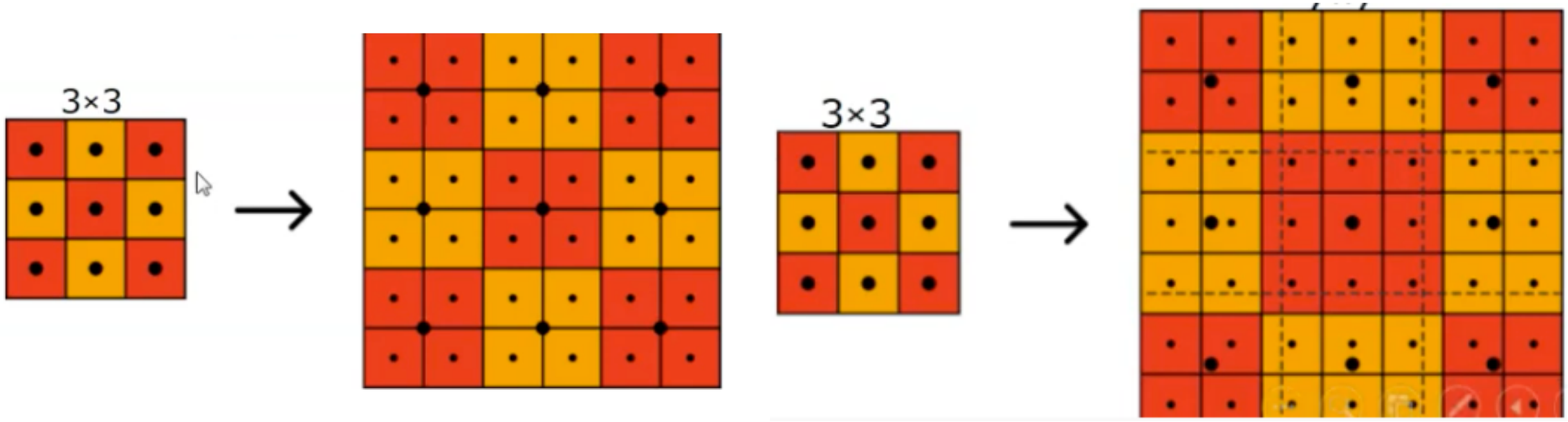


Image rotation

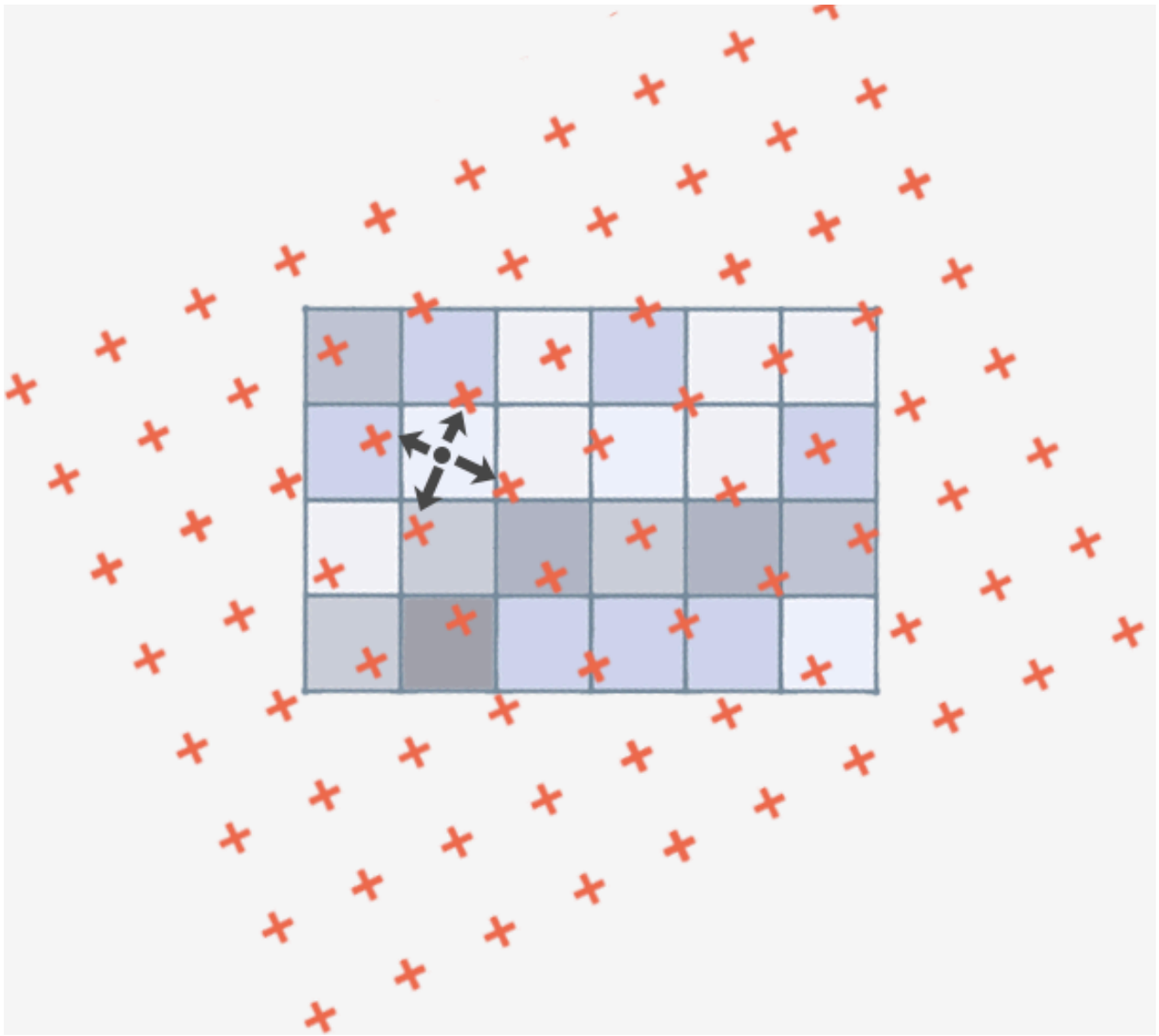


Image interpolation

- Nearest neighbor interpolation
- Bilinear interpolation

$$v(x, y) = ax + by + cxy + d$$

- Bicubic interpolation

$$v(x, y) = \sum_{i=0}^{3} \sum_{j=0}^{3} a_{ij} x^i y^j$$

```
cv2.resize(src, dsize[, dst[, fx[, fy[, interpolation]]]])
```

interpolation	[optional] flag that takes one of the following methods. INTER_NEAREST – a nearest-neighbor interpolation INTER_LINEAR – a bilinear interpolation (used by default) INTER_AREA – resampling using pixel area relation. It may be a preferred method for image decimation, as it gives moire'-free results. But when the image is zoomed, it is similar to the INTER_NEAREST method. INTER_CUBIC – a bicubic interpolation over 4×4 pixel neighborhood INTER_LANCZOS4 – a Lanczos interpolation over 8×8 pixel neighborhood

```
resized = cv2.resize(img, dim, interpolation = cv2.INTER_AREA)
```

Operations

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

Linear Operator

$$H\Big[a_i f_i(x, y) + a_j f_j(x, y)\Big] = a_i H\Big[f_i(x, y)\Big] + a_j H\Big[f_j(x, y)\Big]$$

$$= a_i g_i(x, y) + a_j g_j(x, y)$$

Arithmetic Operations

$$s(x, y) = f(x, y) + g(x, y)$$

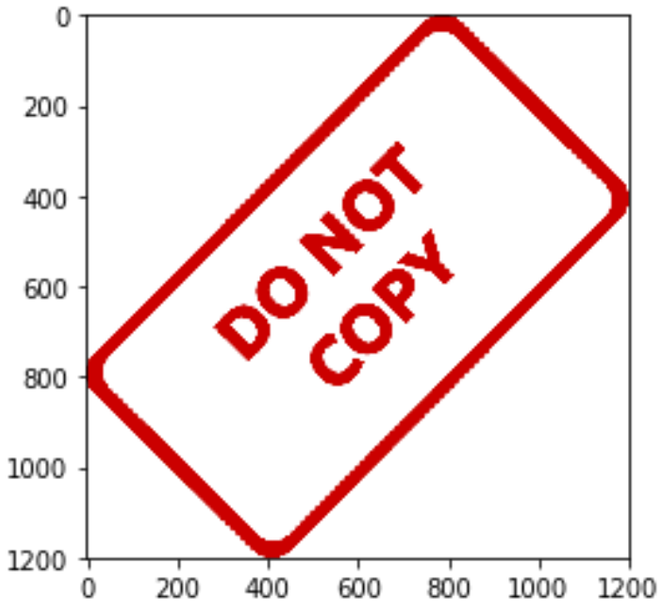
$$d(x, y) = f(x, y) - g(x, y)$$

$$p(x, y) = f(x, y) \times g(x, y)$$

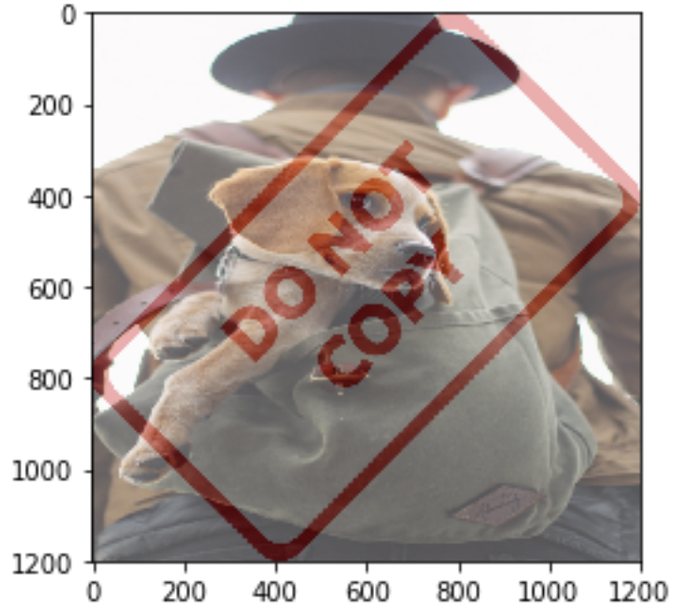
$$v(x, y) = f(x, y) \div g(x, y)$$

Examples

(+)

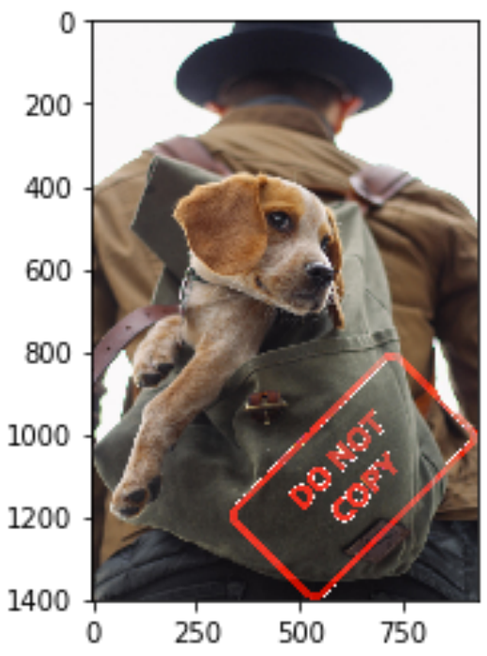


=



$$img1 * \alpha + img2 * \beta + \gamma$$

```
blended = cv2.addWeighted(src1=img1,alpha=0.7,src2=img2,beta=0.3,gamma=0)
```



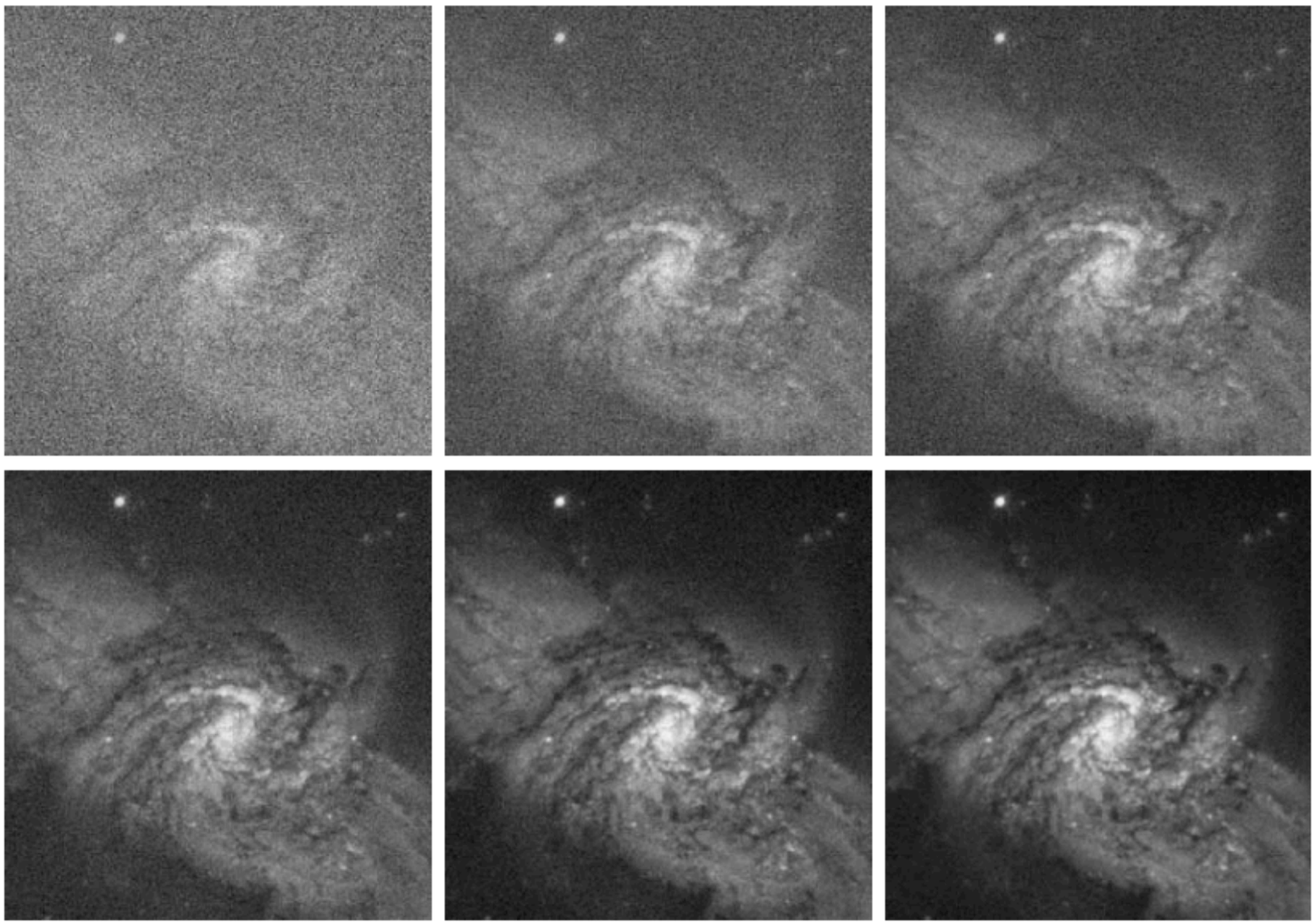
Your turn!

Examples

$$g(x, y) = f(x, y) + \eta(x, y)$$

$$\bar{g}(x, y) = \frac{1}{K} \sum_{i=1}^{K} g_i(x, y)$$

$$\sigma_{\bar{g}(x,y)} = \frac{1}{\sqrt{K}} \sigma_{\eta(x,y)}$$



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.)

Single-pixel operations

$$s = T(z)$$

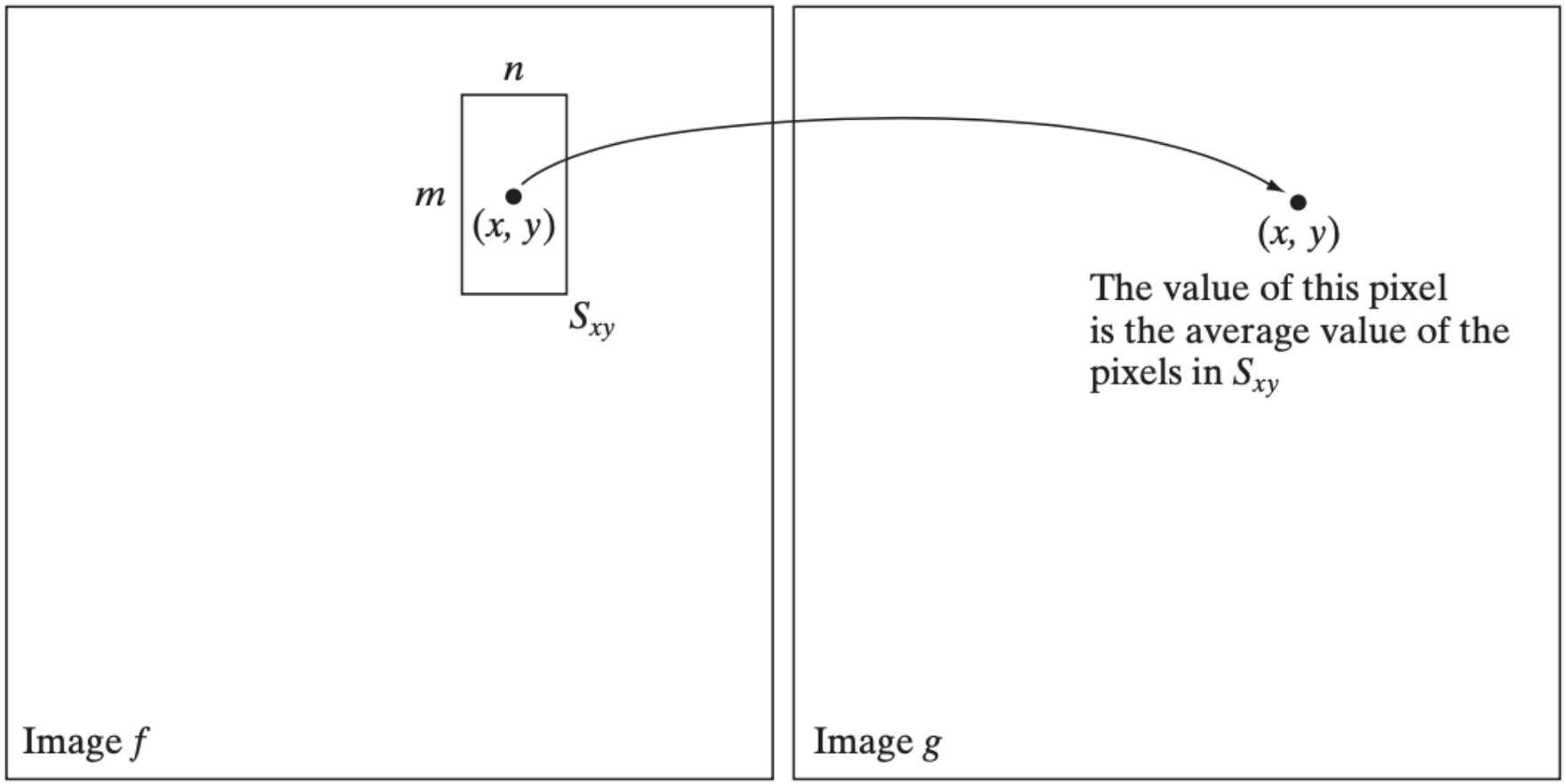
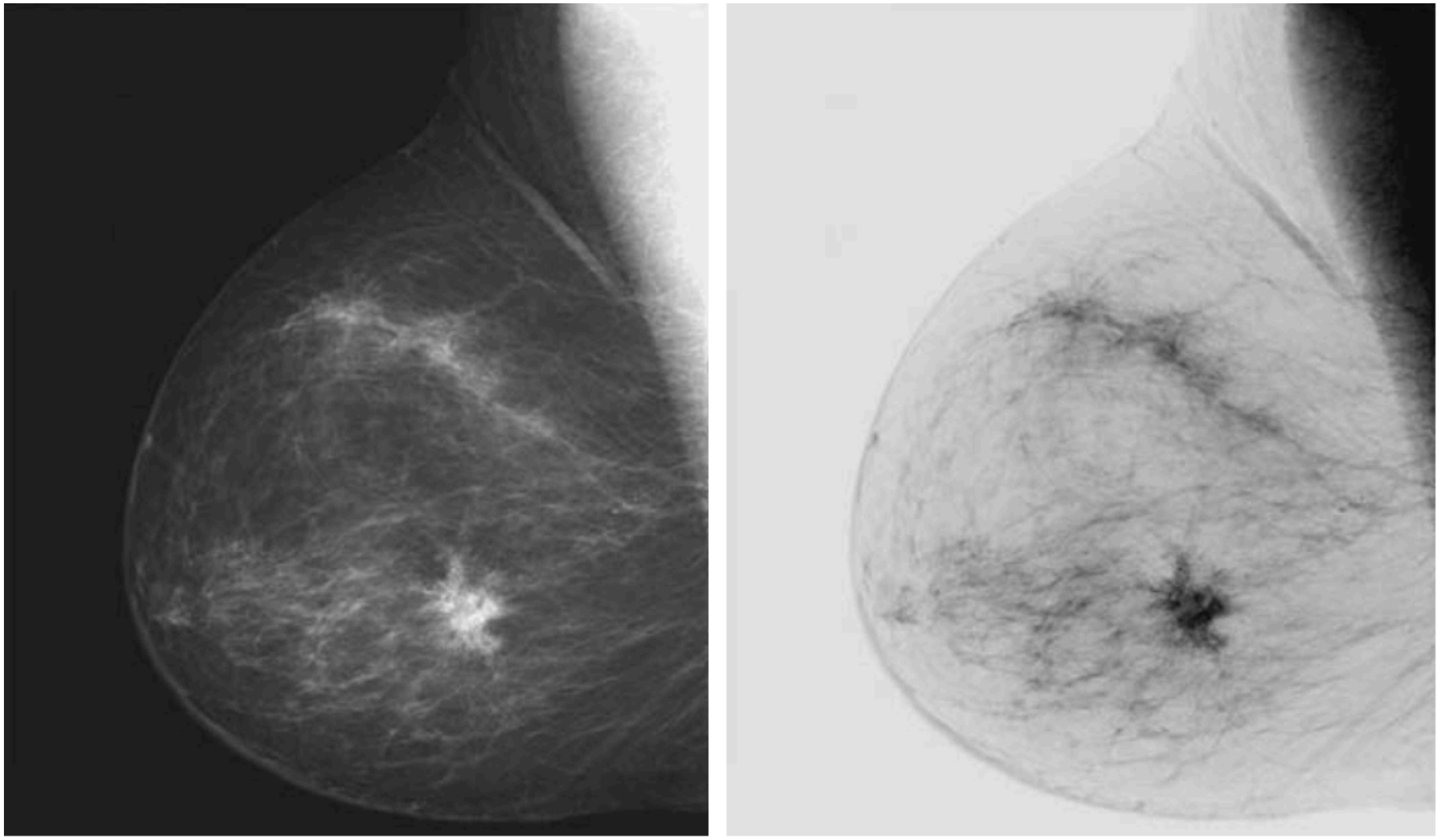


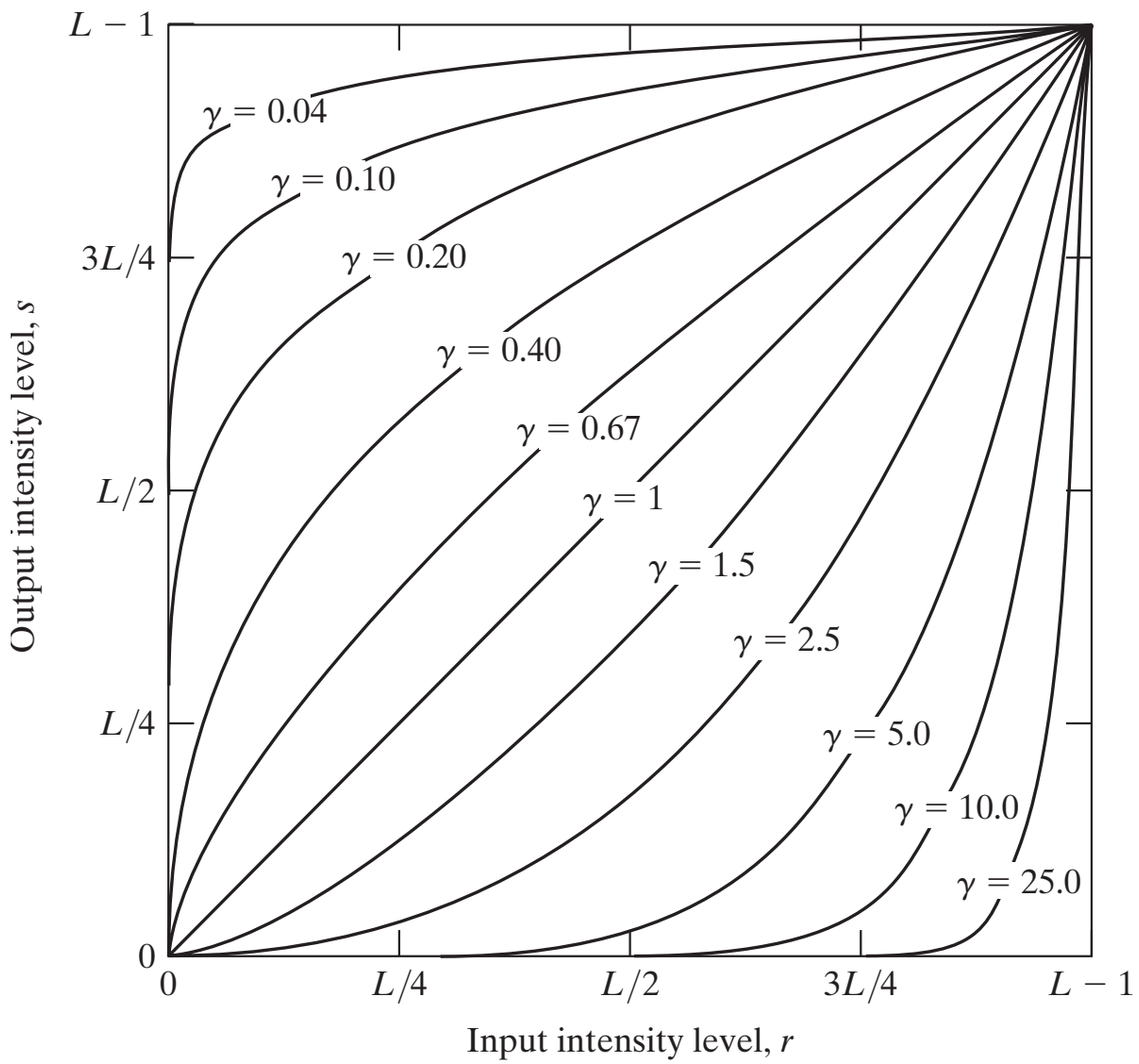
Image negatives

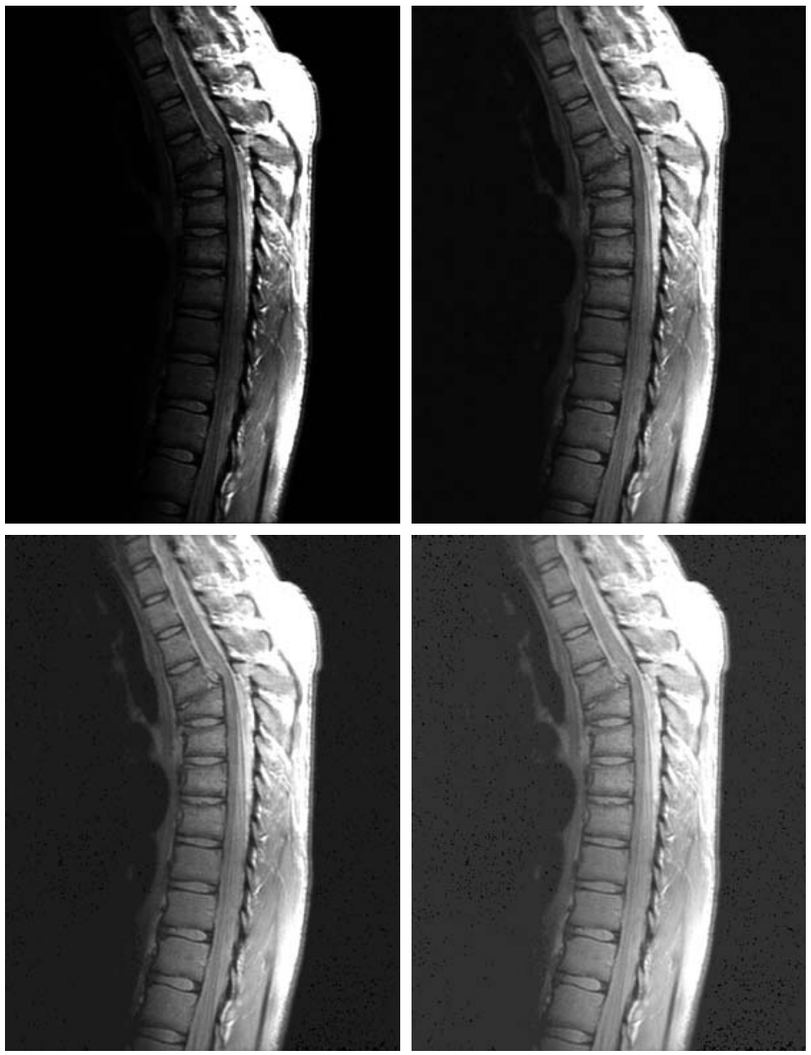
$$s = L - 1 - r$$



Gama Correction

$$s = cr^{\gamma}$$





a b
c d

FIGURE 3.8
(a) Magnetic resonance image (MRI) of a fractured human spine.
(b)–(d) Results of applying the transformation in Eq. (3.2-3) with $c = 1$ and $\gamma = 0.6$, 0.4, and 0.3, respectively. (Original image courtesy of Dr. David R. Pickens, Department of Radiology and Radiological Sciences, Vanderbilt University Medical Center.)

Brightness and Contrast

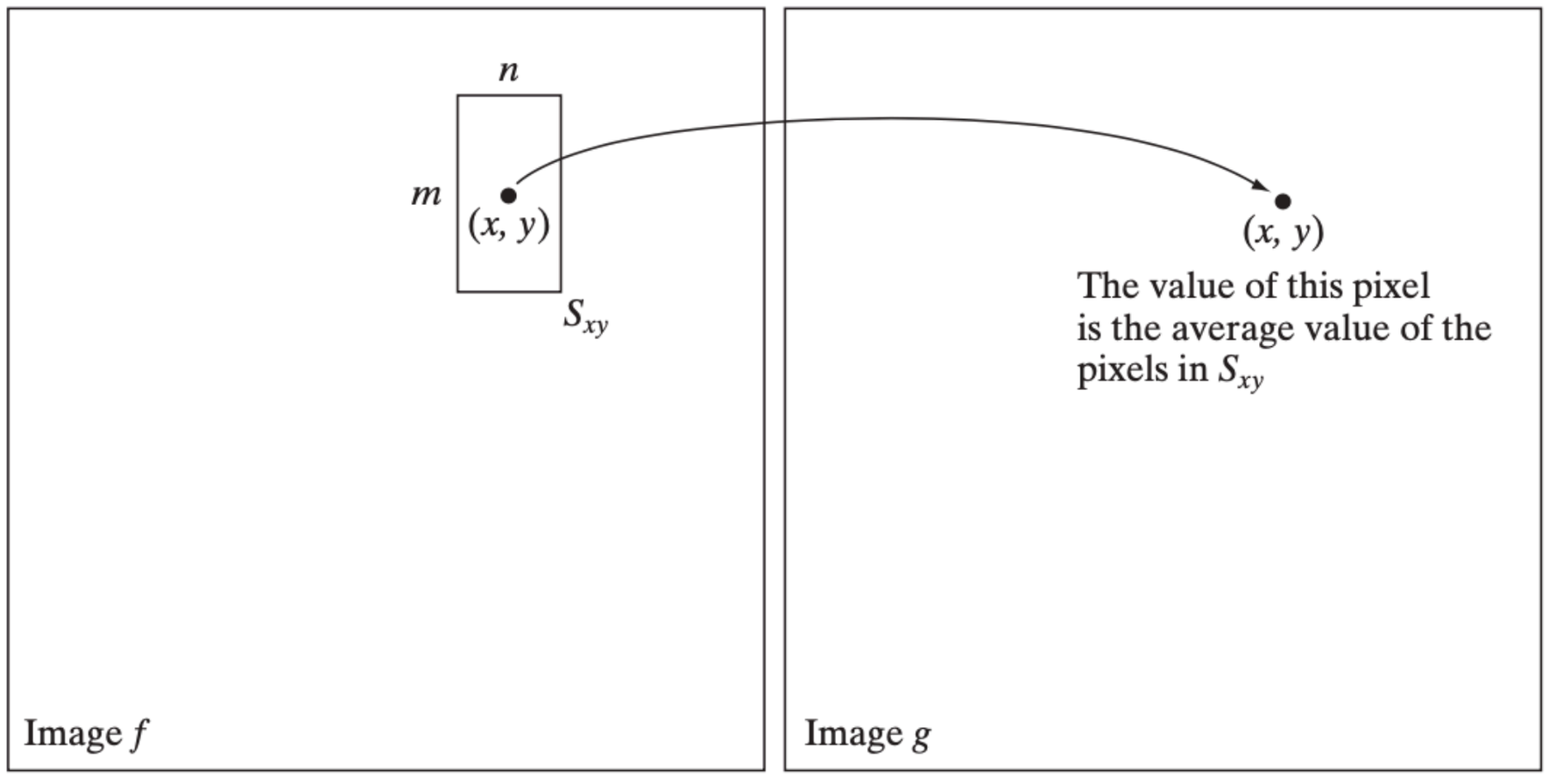
$$s = \alpha \times r + \beta$$

The following image has been corrected with: $\alpha = 1.3$ and $\beta = 40$.

https://docs.opencv.org/3.4/d3/dc1/tutorial_basic_linear_transform.html

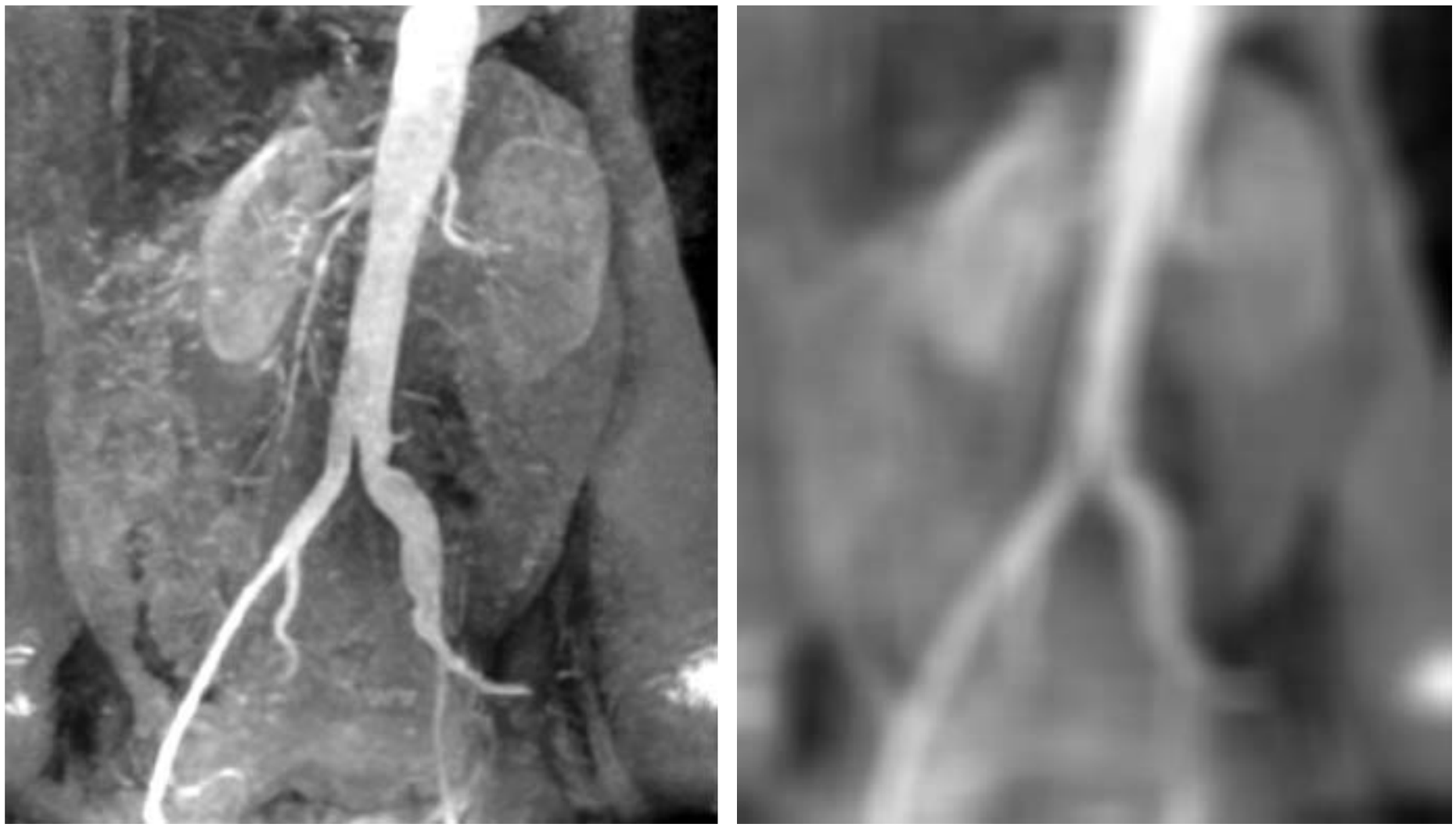
Neighborhood Operations

$$g(x, y) = \frac{1}{mn} \sum_{(r,c) \in S_{xy}} f(r, c)$$



Neighborhood Operations

$$g(x, y) = \frac{1}{mn} \sum_{(r,c) \in S_{xy}} f(r, c)$$



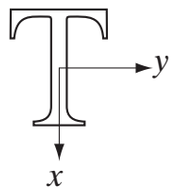
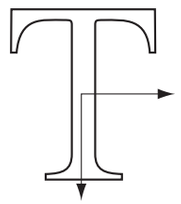
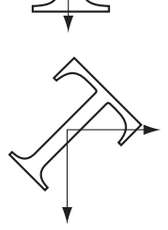
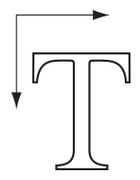
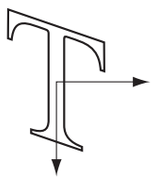
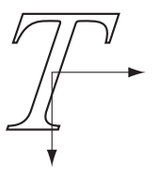
Spatial Transformation

$$(x, y) = T\{(v, w)\}$$

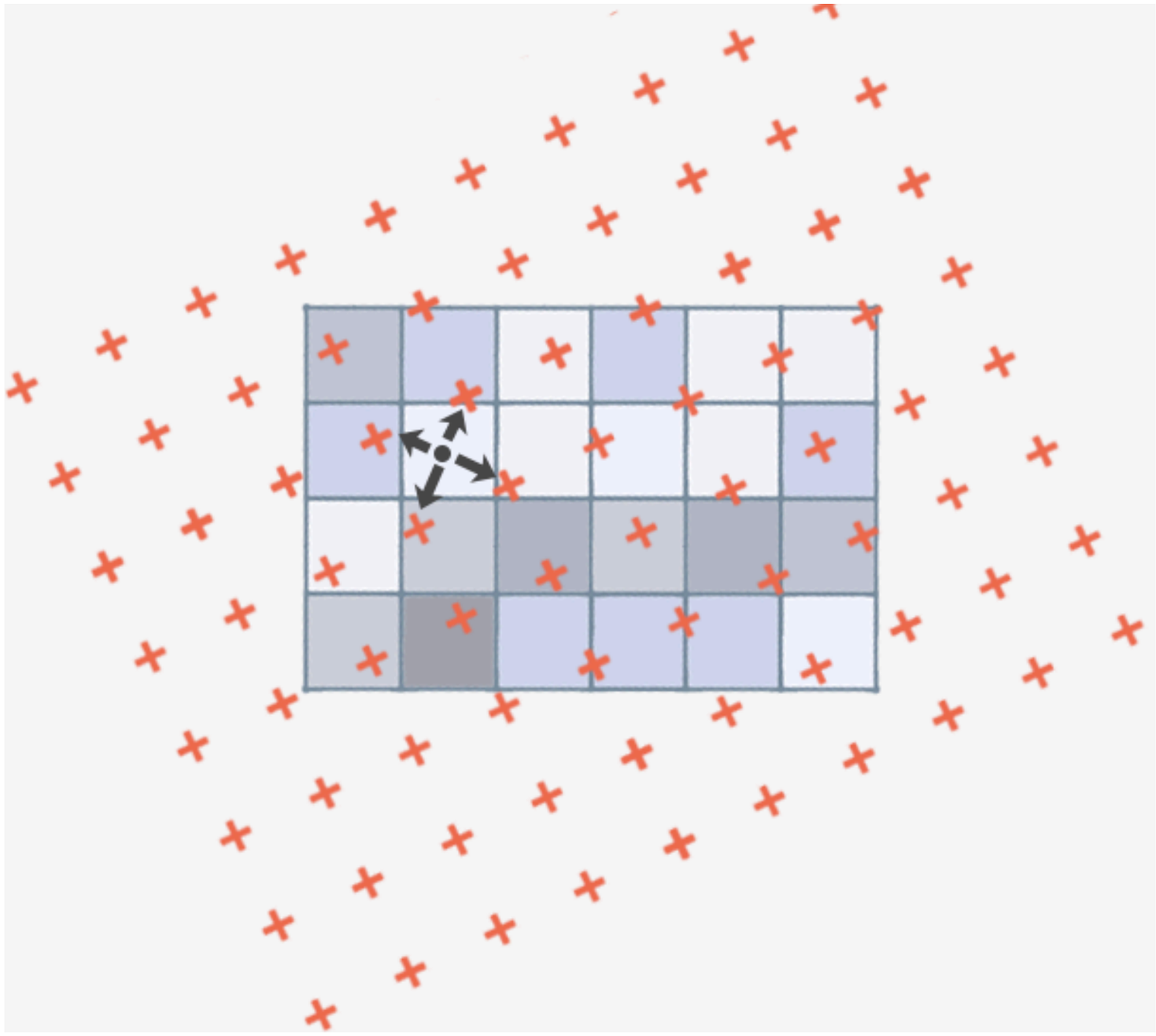
Affine Transformation

$$[x \;\; y \;\; 1] = [v \;\; w \;\; 1]\,\mathbf{T} = [v \;\; w \;\; 1] \begin{bmatrix} t_{11} & t_{12} & 0 \\ t_{21} & t_{22} & 0 \\ t_{31} & t_{32} & 1 \end{bmatrix}$$

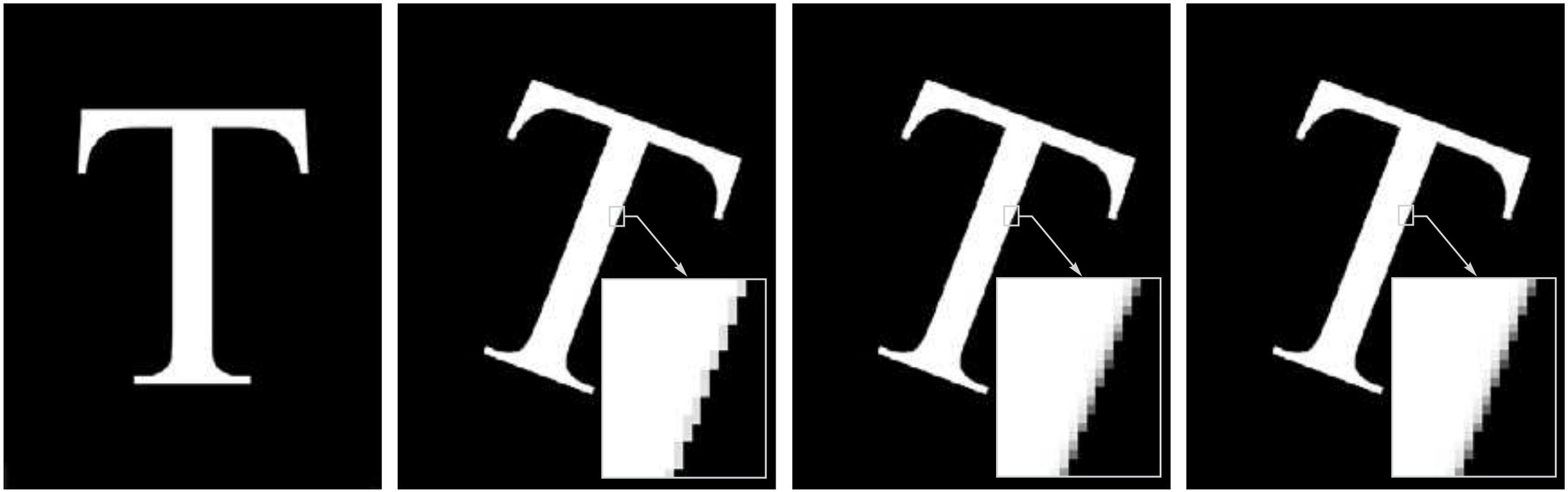
Affine Transformation

Transformation Name	Affine Matrix, T	Coordinate Equations	Example
Identity	$\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$	$x = v$ $y = w$	
Scaling	$\begin{bmatrix} c_x & 0 & 0 \\ 0 & c_y & 0 \\ 0 & 0 & 1 \end{bmatrix}$	$x = c_x v$ $y = c_y w$	
Rotation	$\begin{bmatrix} \cos\theta & \sin\theta & 0 \\ -\sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix}$	$x = v\cos\theta - w\sin\theta$ $y = v\cos\theta + w\sin\theta$	
Translation	$\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ t_x & t_y & 1 \end{bmatrix}$	$x = v + t_x$ $y = w + t_y$	
Shear (vertical)	$\begin{bmatrix} 1 & 0 & 0 \\ s_v & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$	$x = v + s_v w$ $y = w$	
Shear (horizontal)	$\begin{bmatrix} 1 & s_h & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$	$x = v$ $y = s_h v + w$	

Interpolation and Affine Transformation



Interpolation and Affine Transformation



a b c d

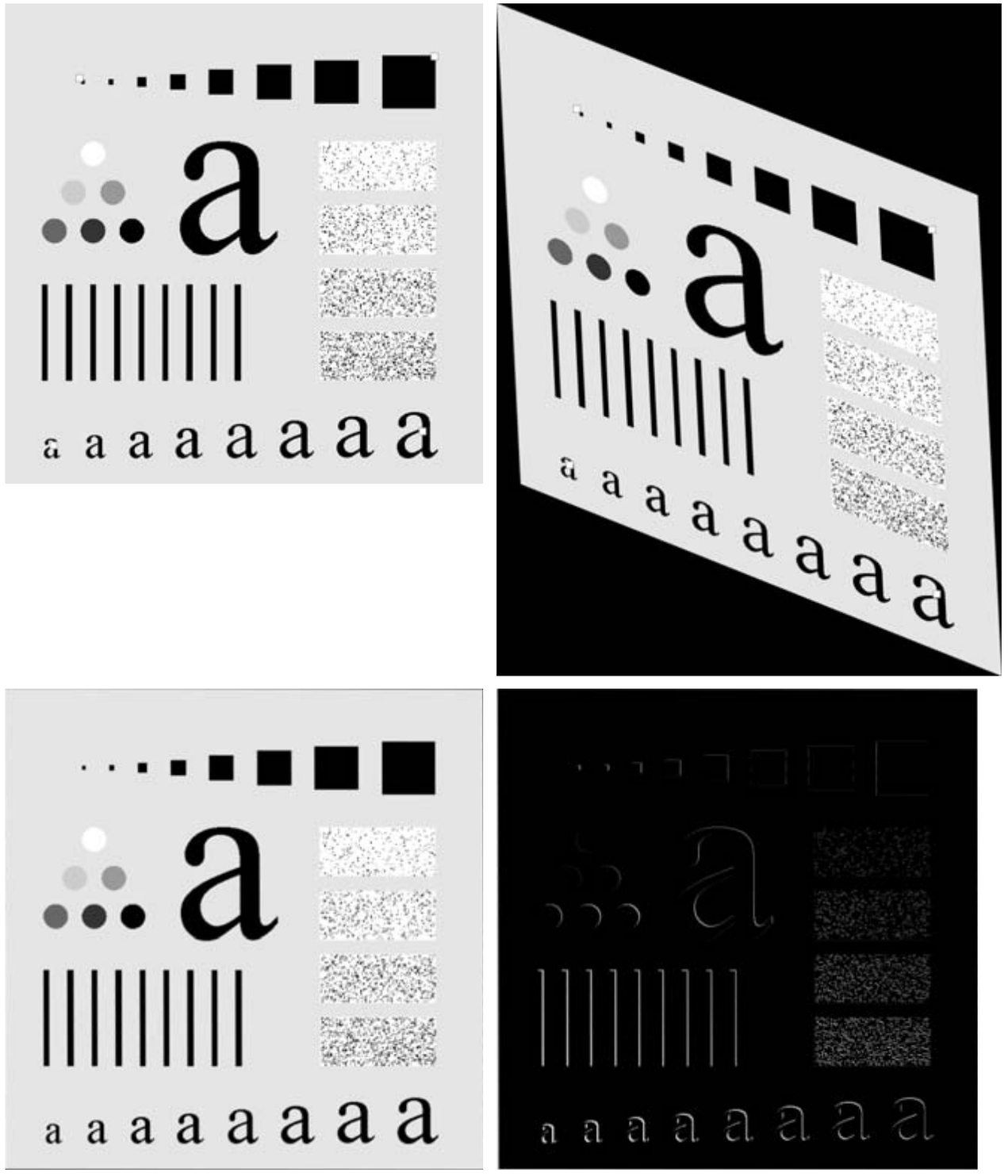
FIGURE 2.36 (a) A 300 dpi image of the letter T. (b) Image rotated 21° using nearest neighbor interpolation to assign intensity values to the spatially transformed pixels. (c) Image rotated 21° using bilinear interpolation. (d) Image rotated 21° using bicubic interpolation. The enlarged sections show edge detail for the three interpolation approaches.

Image Registration

- Align an image to a reference image
- How to?
  - ✓ Find **tie points** (control points)
  - ✓ Solve transformation Eq.

$$x = c_1 v + c_2 w + c_3 v w + c_4$$
$$y = c_5 v + c_6 w + c_7 v w + c_8$$

Image Registration



a b
c d

FIGURE 2.37
Image registration. (a) Reference image. (b) Input (geometrically distorted image). Corresponding tie points are shown as small white squares near the corners. (c) Registered image (note the errors in the border). (d) Difference between (a) and (c), showing more registration errors.

Image Transforms

Forward transform

$$T(u, v) = \sum_{x=0}^{M-1} \sum_{y=0}^{N-1} f(x, y) r(x, y, u, v)$$

Inverse transform

$$f(x, y) = \sum_{u=0}^{M-1} \sum_{v=0}^{N-1} T(u, v) s(x, y, u, v)$$

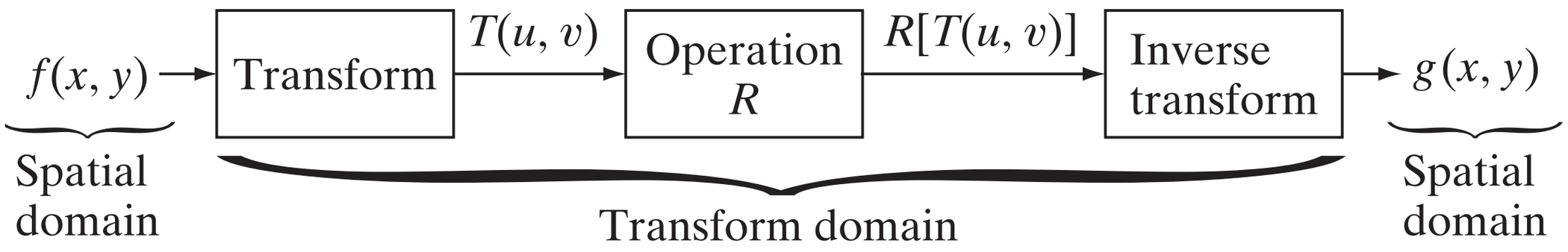


Image Transforms