Linear Image Processing

- Digital image as vector
- Linear operator
- Impulse response
- Separable processing
- Shift-invariant systems
- Non-separable filtering

Digital Images Written as Vectors

$$\vec{f} = \begin{bmatrix} f(0,0) \\ f(1,0) \\ \vdots \\ f(N-1,0) \\ f(0,1) \\ \vdots \\ f(N-1,1) \\ \vdots \\ f(0,L-1) \\ f(N-1,L-1) \end{bmatrix} = \begin{bmatrix} f_{00} \\ f_{01} \\ \vdots \\ f_{0,N-1} \\ f_{10} \\ \vdots \\ f_{1,N-1} \\ \vdots \\ f_{L-1,0} \\ f_{L-1,N-1} \end{bmatrix}$$

Column vector of length L × N

Linear Image Processing

- Any linear image processing algorithm can be written as

$$\vec{g} = \mathbf{H}\vec{f}$$

Note: matrix **H** need not be square

- Definition of a linear operator O[•]

$$O\left[\alpha \cdot \vec{f} + \beta \cdot \vec{g}\right] = \alpha \cdot O\left[\vec{f}\right] + \beta \cdot O\left[\vec{g}\right]$$

for all scalars α , β

- Almost all image processing algorithms contain at least some linear operators.

Impulse Response

- Another way to represent any linear image processing scheme

$$g(\alpha, \beta) = \sum_{x=0}^{N-1} \sum_{y=0}^{L-1} f(x, y) \cdot h(x, \alpha, y, \beta)$$

impulse response
point spread function

- Unit impulse at pixel (a,b)

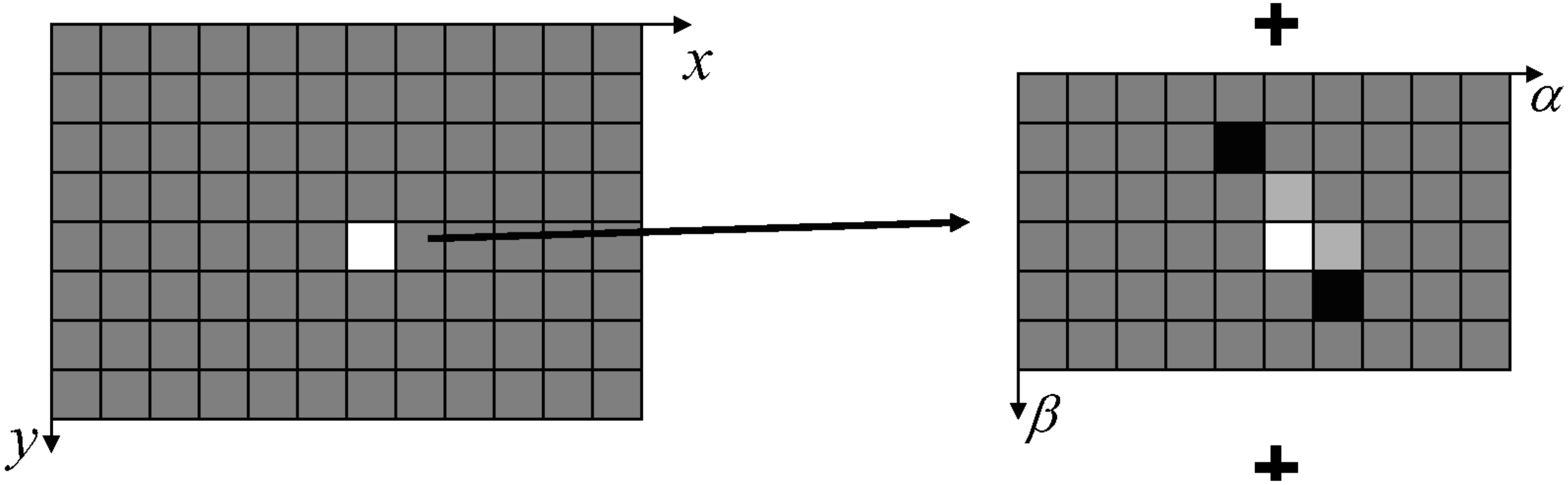
$$\delta(x-a, y-b) = \begin{cases} 1 & : \quad x = a \wedge y = b \\ 0 & : \quad \text{else} \end{cases}$$

- Result of linear image processing operator

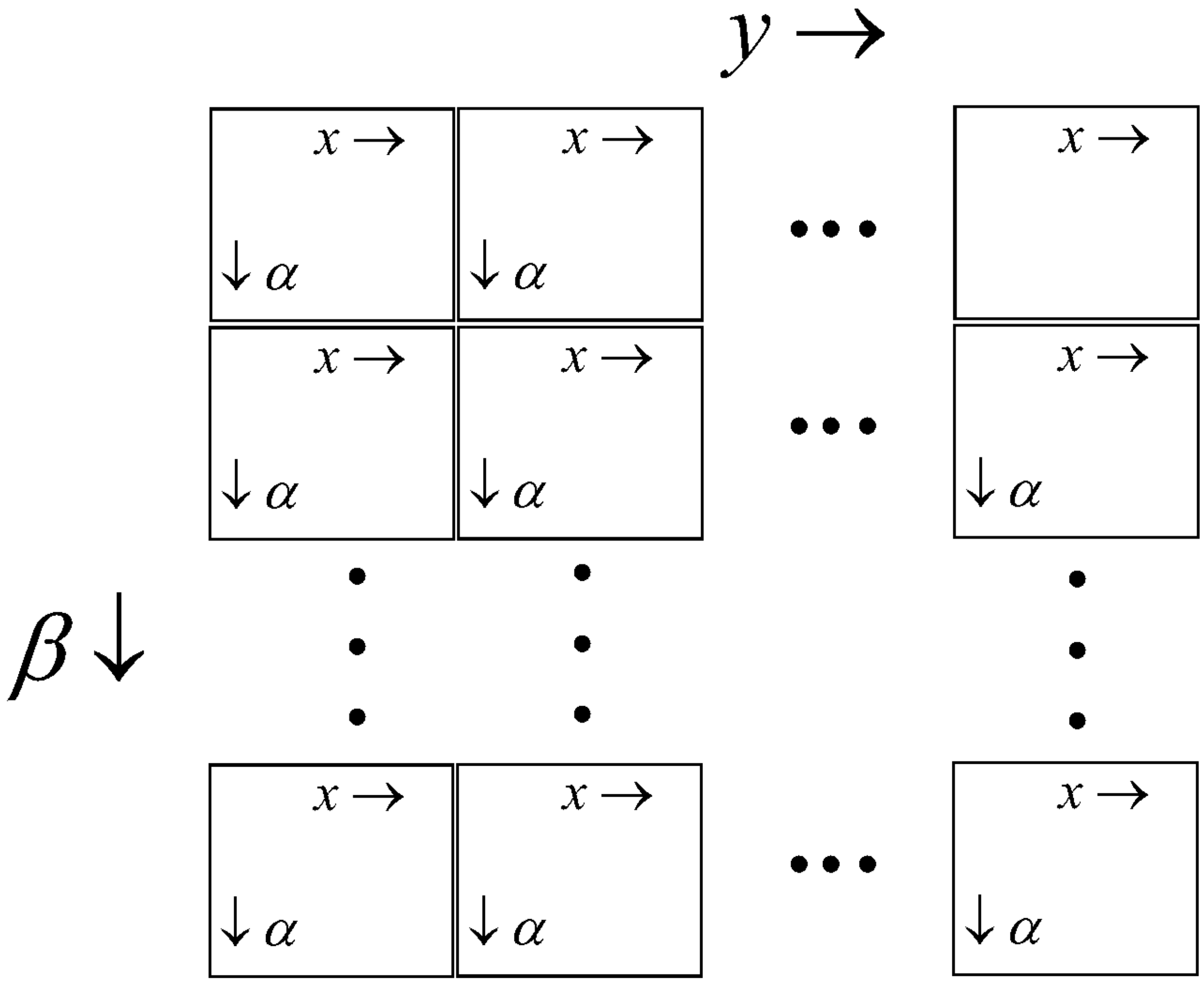
$$\sum_{x=0}^{N-1} \sum_{y=0}^{L-1} \delta(x-a, y-b) \cdot h(x, \alpha, y, \beta) = h(a, \alpha, b, \beta)$$

Superposition of Impulse Responses

$$f(x,y) \qquad\qquad g(\alpha,\beta) = \cdots + f(x,y)\cdot h(x,\alpha,y,\beta) + \cdots$$



Relationship of **H** and h(x,α,y,β)



Separable Linear Image Processing

- Impulse response is separable in (x,α) and (y,β), i.e., can be written as

$$g(\alpha,\beta) = \sum_{x=0}^{N-1} \sum_{y=0}^{L-1} f(x,y) \cdot h_x(x,\alpha) h_y(y,\beta)$$

- Processing can be carried out
  - Row by row, then column by column

$$g(\alpha,\beta) = \sum_{y=0}^{L-1} h_y(y,\beta) \sum_{x=0}^{N-1} f(x,y) \cdot h_x(x,\alpha)$$

  - Column by column, then row by row

$$g(\alpha,\beta) = \sum_{x=0}^{N-1} h_x(x,\alpha) \sum_{y=0}^{L-1} f(x,y) \cdot h_y(y,\beta)$$

Separable Linear Image Processing

- If the digital input and output images are written as matrices **f** and **g**, we can conveniently write

$$\mathbf{g} = \mathbf{H_y}^T \cdot \mathbf{f} \cdot \mathbf{H_x}$$

$$\mathbf{H_y} = \begin{bmatrix} h_y(0,0) & h_y(0,1) & \cdots & h_y(0,L_g-1) \\ h_y(1,0) & h_y(1,1) & \cdots & h_y(1,L_g-1) \\ \vdots & \vdots & \ddots & \vdots \\ h_y(L-1,0) & h_y(L-1,1) & \cdots & h_y(L-1,L_g-1) \end{bmatrix}$$

$$\mathbf{H_x} = \begin{bmatrix} h_x(0,0) & h_x(0,1) & \cdots & h_x(0,N_g-1) \\ h_x(1,0) & h_x(1,1) & \cdots & h_x(1,N_g-1) \\ \vdots & \vdots & \ddots & \vdots \\ h_x(N-1,0) & h_x(N-1,1) & \cdots & h_x(N-1,N_g-1) \end{bmatrix}$$

- Output image **g** has size $L_g \times N_g$
- If the operator does not change image size, $\mathbf{H_x}$ and $\mathbf{H_y}$ are square matrices

Example: Subsampling

- Image subsampling 2:1 horizontally and vertically
- Small input image of size 8x8, output image size 4x4

$$\mathbf{H_x} = \mathbf{H_y} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \end{bmatrix}$$

- Nice for unified treatment, but NOT recommended for implementation

Example: Subsampling

- A somewhat better technique for 2:1 image size reduction

$$\mathbf{H}_\mathrm{x} = \mathbf{H}_\mathrm{y} = \begin{bmatrix} 0.5 & 0 & 0 & 0 \\ 0.5 & 0 & 0 & 0 \\ 0 & 0.5 & 0 & 0 \\ 0 & 0.5 & 0 & 0 \\ 0 & 0 & 0.5 & 0 \\ 0 & 0 & 0.5 & 0 \\ 0 & 0 & 0 & 0.5 \\ 0 & 0 & 0 & 0.5 \end{bmatrix}$$

- Can you figure out what this does to the image?
- Why is this a better technique than the previous one?

Example: Filtering

- Each pixel is replaced by the average of two horizontally (vertically) neighboring pixels

$$\mathbf{H_x} = \mathbf{H_y} = \begin{bmatrix} 0.5 & 0 & & \cdots & & & & 0 \\ 0.5 & 0.5 & & & & & & \\ 0 & 0.5 & 0.5 & & & & & \\ & & 0.5 & 0.5 & \ddots & & & \vdots \\ & & & 0.5 & 0.5 & & & \\ \vdots & & & \ddots & 0.5 & 0.5 & & \\ & & & & & 0.5 & 0.5 & 0 \\ 0 & & & \cdots & & 0 & 0.5 & 0.5 \end{bmatrix}$$

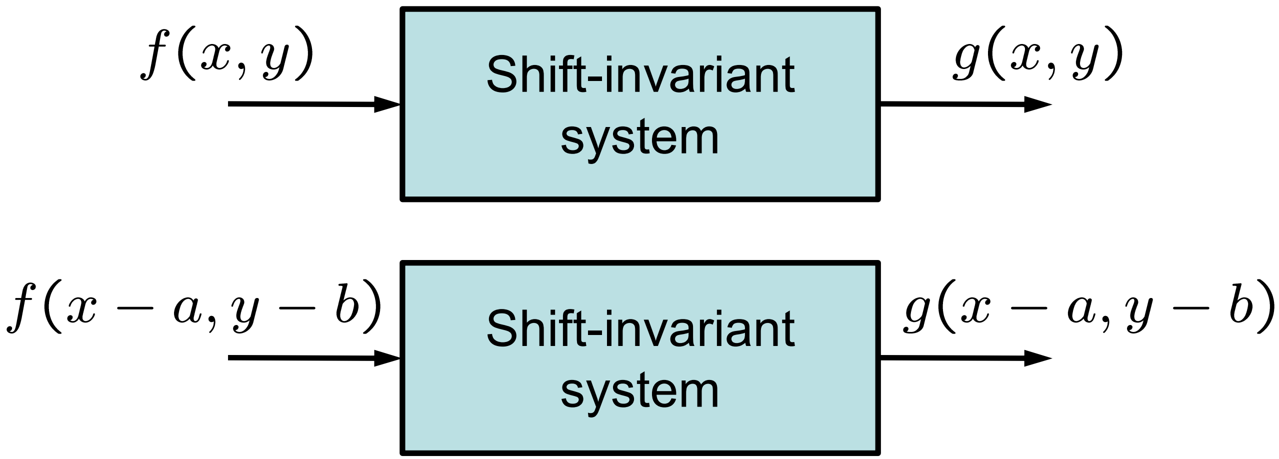
- Shift-invariant operation (except for image boundary)

Shift-Invariant Systems

- Assume that digital images f(x,y) and g(x,y) have infinite support

$$(x,y) \in \{\cdots,-2,-1,0,1,2,\cdots\}\times\{\cdots,-2,-1,0,1,2,\cdots\}$$

. . . then, for all integers a and b



- Shift-invariance does **not** imply linearity (or vice versa).

Shift-Invariant Systems

- Why is shift-invariance desirable for image processing systems?
- Often, image processing results should not depend on the choice of the coordinate system origin, e.g.,
  - all parts of an image should be equally sharpened,
  - the ZIP code of a letter should be extracted, regardless of where exactly the address is written on the envelope,
  - the quality of an image coder should not fluctuate, if the same image is presented, but shifted by few pixels.
- Many more complicated image processing systems are not exactly shift-invariant.
  - They might use blockwise transforms.
  - They might use processing on coarser sampling grids.

Shift-Invariant Systems and Toeplitz Matrices

- For a separable, shift-invariant, linear system
  $h_x(x,\alpha) = h_{siv,x}(\alpha - x)$ and $h_y(y,\beta) = h_{siv,y}(\beta - y)$
- Matrices $\mathbf{H_x}$ and $\mathbf{H_y}$ are square and Toeplitz, e.g.,

$$\mathbf{H_x} = \begin{bmatrix} h_{siv,x}(0) & h_{siv,x}(1) & \cdots & h_{siv,x}(N-1) \\ h_{siv,x}(-1) & h_{siv,x}(0) & \cdots & h_{siv,x}(N-2) \\ \vdots & \vdots & \ddots & \vdots \\ h_{siv,x}(1-N) & h_{siv,x}(2-N) & \cdots & h_{siv,x}(0) \end{bmatrix}$$

- Operation is a 2-d separable convolution (“filtering”)

$$g(\alpha,\beta) = \sum_{x=0}^{N-1} \sum_{y=0}^{L-1} f(x,y) \cdot h_{siv,x}(\alpha - x) h_{siv,y}(\beta - y)$$

Non-Separable Filtering

- Convolution kernel of linear shift-invariant system ("filter") can also be non-separable

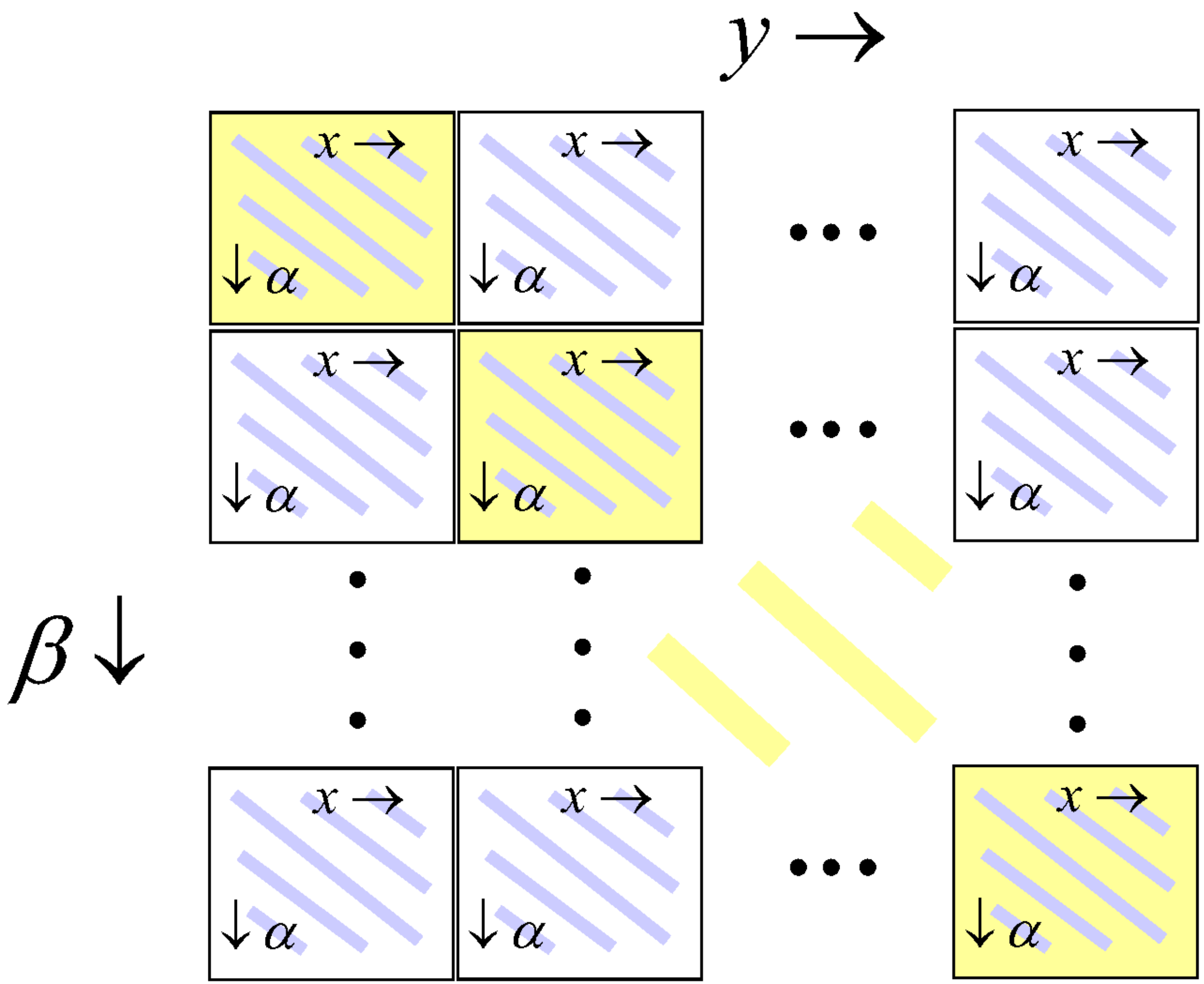
$$g(\alpha,\beta) = \sum_{x=0}^{N-1} \sum_{y=0}^{L-1} f(x,y) \cdot h_{siv}(\alpha - x, \beta - y)$$

- Viewed as a matrix operation . . .

$$\vec{g} = \mathbf{H}\vec{f}$$

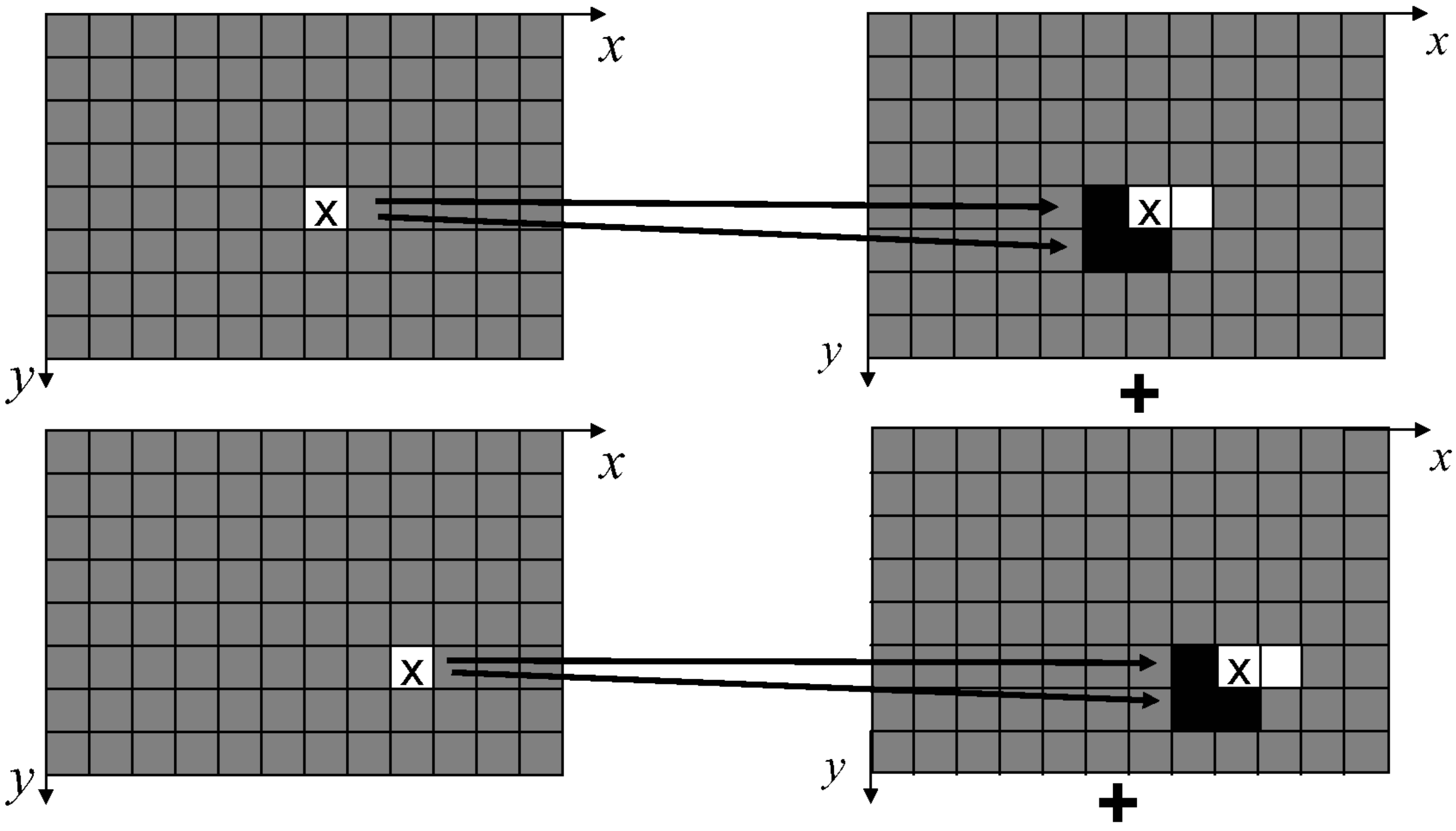
. . . **H** is a block Toeplitz matrix

Structure of H for Non-Separable Filtering



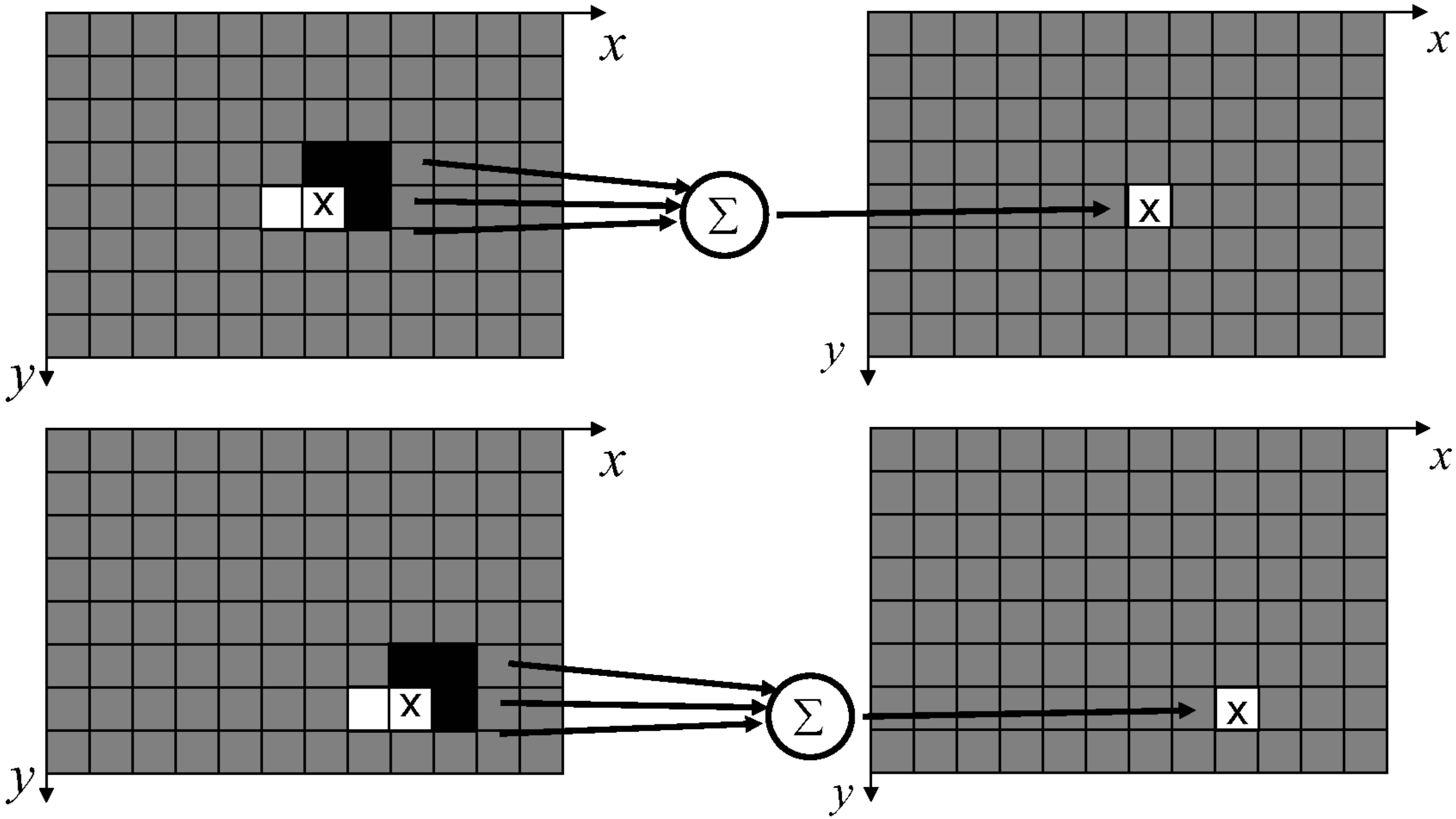
Superposition of Impulse Responses

$$f(x,y) \qquad g(x,y) = \cdots + f(a,b) \cdot h_{siv}(x-a, y-b) + \cdots$$



Linear Combination of Neighboring Pixel Values

$$\sum_{a=0}^{N-1} \sum_{b=0}^{L-1} f(a,b) \cdot h_{siv}(x-a, y-b) = g(x,y)$$



Filtering Examples

Cameraman
original

Cameraman
blurred by convolution
Filter impulse response:

$$\frac{1}{25}\begin{bmatrix} 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & [1] & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \end{bmatrix}$$

Filtering Examples

Cameraman
original

Cameraman
Blurred horizontally
Filter impulse response:

$$\frac{1}{5}\begin{bmatrix} 1 & 1 & [1] & 1 & 1 \end{bmatrix}$$

Filtering Examples

Cameraman
original

Cameraman
blurred vertically
Filter impulse response:

$$\frac{1}{5}\begin{bmatrix} 1 \\ 1 \\ [1] \\ 1 \\ 1 \end{bmatrix}$$

Filtering Examples

Cameraman
original

Cameraman
sharpened
Filter impulse response:

$$\frac{1}{4}\begin{bmatrix} 0 & -1 & 0 \\ -1 & [8] & -1 \\ 0 & -1 & 0 \end{bmatrix}$$

Filtering Examples

Cameraman
original

Cameraman
sharpened
Filter impulse response:

$$\begin{bmatrix} 0 & -1 & 0 \\ -1 & [5] & -1 \\ 0 & -1 & 0 \end{bmatrix}$$