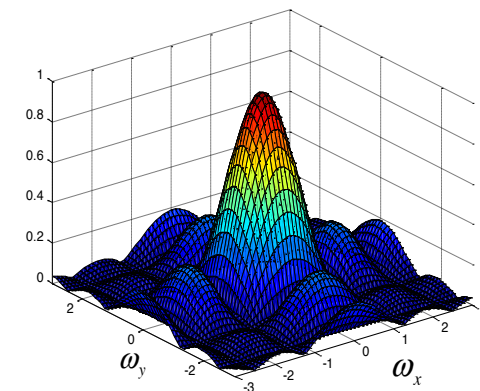
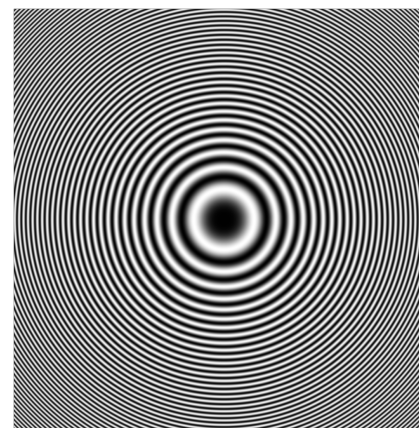


Linear Image Processing and Filtering

- Math of 2-d linear systems
- Separability
- Shift-invariant linear systems, 2-d convolution
- Integral image for fast box-filtering
- 2-d frequency response
- Blurring and sharpening
- Zoneplate
- Image sampling/aliasing
- Wiener filtering
- Nonlinear noise reduction/sharpening



Linear image processing

- Image processing system $S(\cdot)$ is linear, iff superposition principle holds:

$$S\left(\alpha \cdot f[x, y] + \beta \cdot g[x, y]\right) = \alpha \cdot S\left(f[x, y]\right) + \beta \cdot S\left(g[x, y]\right) \quad \text{for all } \alpha, \beta \in \mathbb{R}$$

- Any linear image processing system can be written as

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

Note: matrix H need not be square.

by sorting pixels into a column vector

$$\vec{f} = \left(f[0,0] \quad f[1,0] \quad \cdots \quad f[N-1,0] \quad f[0,1] \quad \cdots \quad f[N-1,1] \quad \cdots \quad \cdots \quad f[0,L-1] \quad \cdots \quad f[N-1,L-1] \right)^T$$

Impulse response

- Another way to represent any linear image processing scheme

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

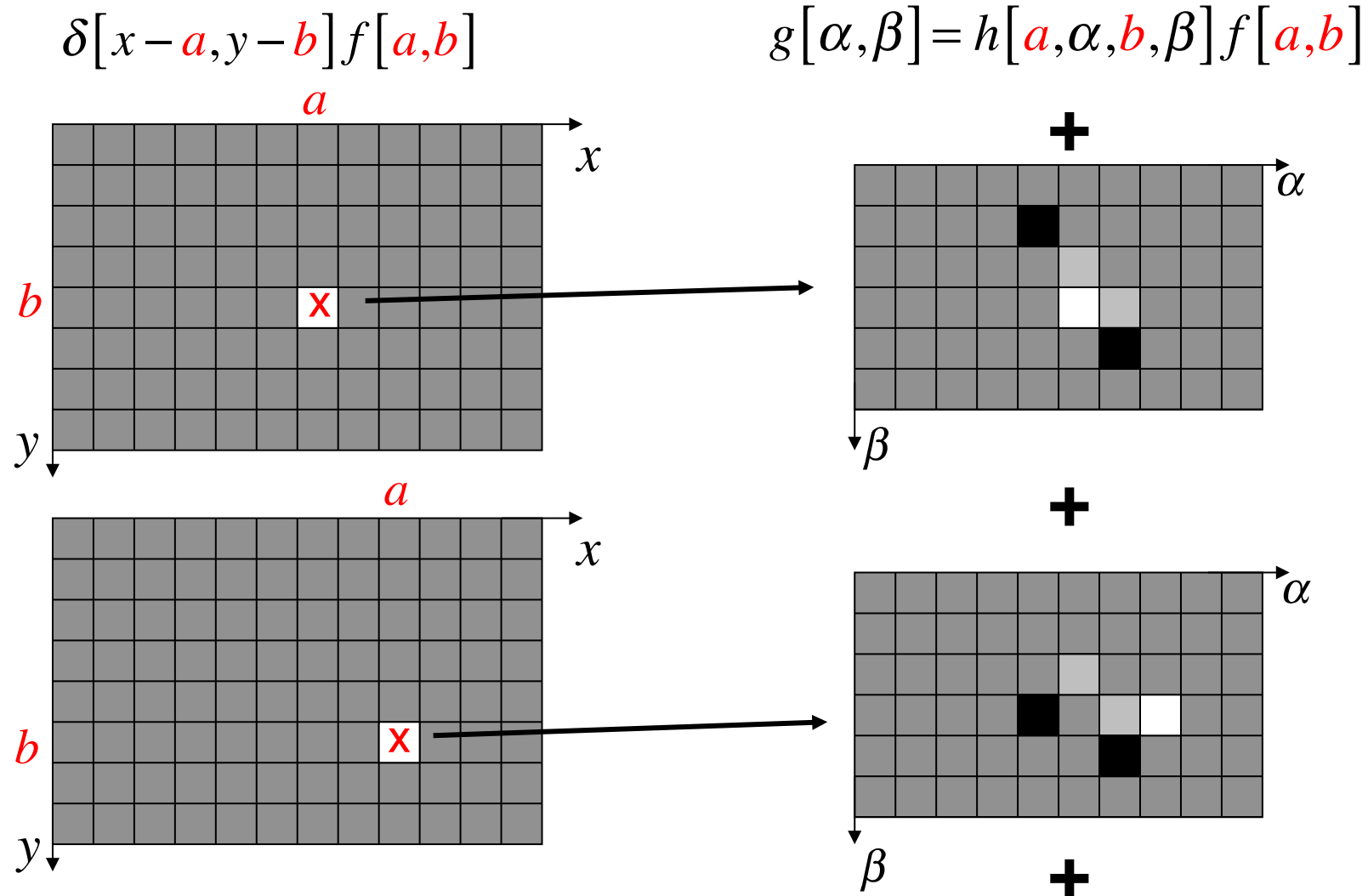
- Input: unit impulse at pixel $[a, b]$

impulse response

- Output: impulse response $f[x, y] = \delta[x - a, y - b] = \begin{cases} 1 & x = a \wedge y = b \\ 0 & \text{else} \end{cases}$

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

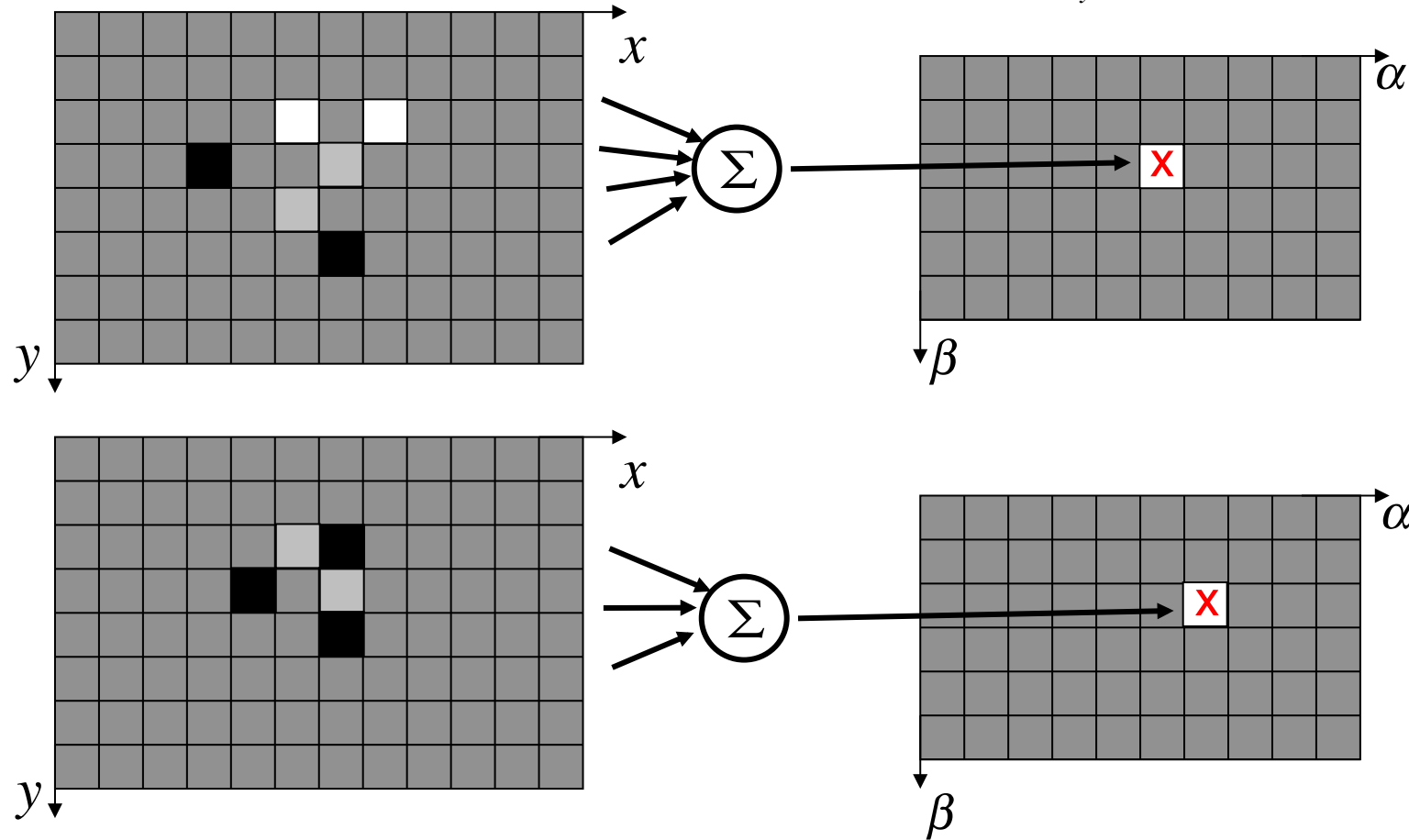
Interpretation #1: superposition of impulse responses



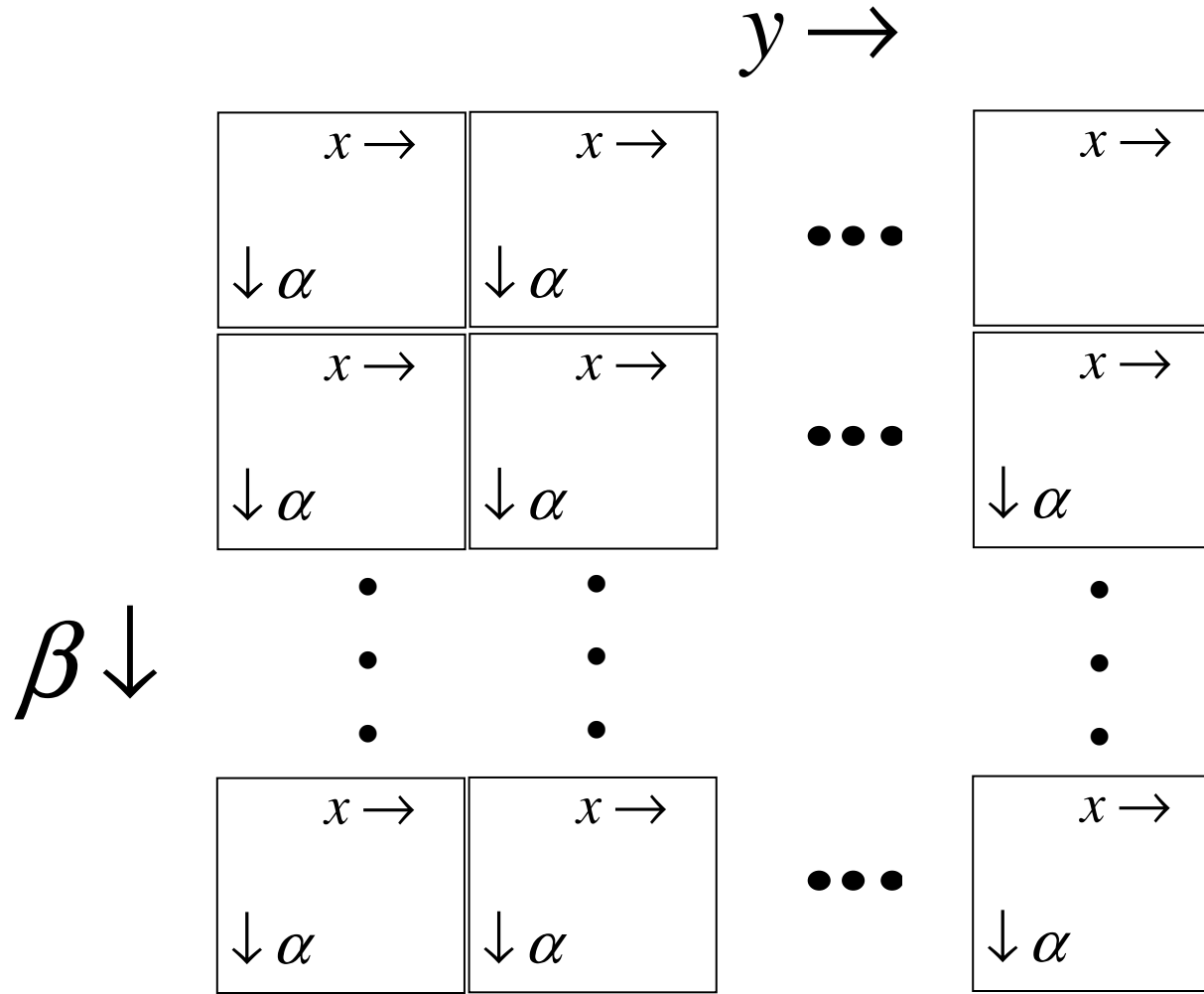
Interpretation #2: linear combination of input values

$$f[x,y] \cdot h[x,\alpha,y,\beta]$$

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



Relationship of H and $h[x,\alpha,y,\beta]$



Separable linear image processing

- Impulse response is separable in $[x,\alpha]$ and $[y,\beta]$, 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 (cont.)

- If the digital input and output images are written as a matrices $\mathbf{f}$ and $\mathbf{g}$, we can conveniently write

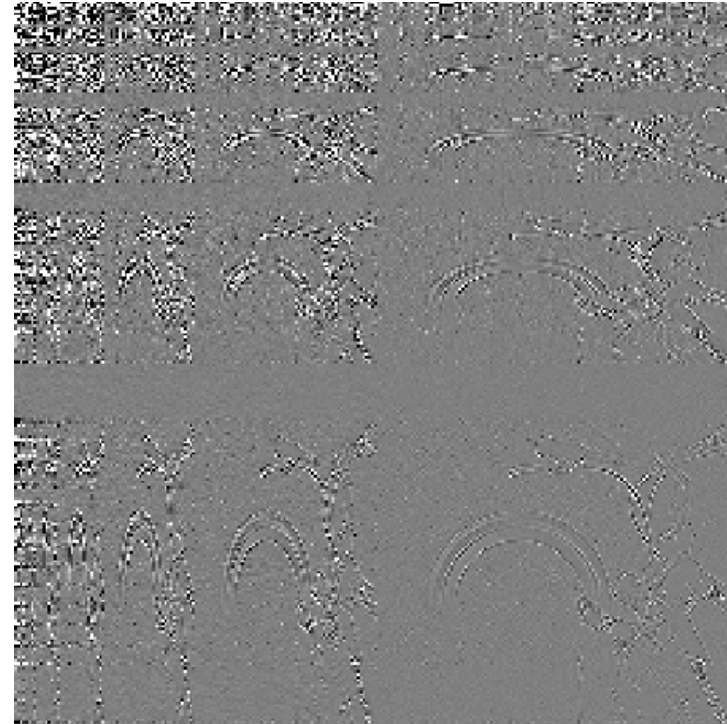
$$\mathbf{g} = \mathbf{H}_y^T \cdot \mathbf{f} \cdot \mathbf{H}_x$$

$$\mathbf{H}_y = \begin{bmatrix} h_y[0,0] & h_y[0,1] & \cdots & h_y[0,L_g-1] \\ h_y[1,0] & h_y[1,1] & \cdots & h_y[1,L_g-1] \\ \vdots & \vdots & & \vdots \\ h_y[L-1,0] & h_y[L-1,1] & \cdots & h_y[L-1,L_g-1] \end{bmatrix}$$
$$\mathbf{H}_x = \begin{bmatrix} h_x[0,0] & h_x[0,1] & \cdots & h_x[0,N_g-1] \\ h_x[1,0] & h_x[1,1] & \cdots & h_x[1,N_g-1] \\ \vdots & \vdots & & \vdots \\ h_x[N-1,0] & h_x[N-1,1] & \cdots & h_x[N-1,N_g-1] \end{bmatrix}$$

- Output image $\mathbf{g}$ 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: Separable Haar transform

Original *Bike*
256x256



256x256
Haar transform
 $H_x = H_y = H_{r_{256}}$



Haar transform

- Haar transform matrix for sizes $N=2,4,8$

$$\mathbf{Hr}_2 = \frac{1}{\sqrt{2}} \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix}$$

$$\mathbf{Hr}_4 = \frac{1}{\sqrt{4}} \begin{pmatrix} 1 & 1 & \sqrt{2} & 0 \\ 1 & 1 & -\sqrt{2} & 0 \\ 1 & -1 & 0 & \sqrt{2} \\ 1 & -1 & 0 & -\sqrt{2} \end{pmatrix}$$

$$\mathbf{Hr}_8 = \frac{1}{\sqrt{8}} \begin{pmatrix} 1 & 1 & \sqrt{2} & 0 & 2 & 0 & 0 & 0 \\ 1 & 1 & \sqrt{2} & 0 & -2 & 0 & 0 & 0 \\ 1 & 1 & -\sqrt{2} & 0 & 0 & 2 & 0 & 0 \\ 1 & 1 & -\sqrt{2} & 0 & 0 & -2 & 0 & 0 \\ 1 & -1 & 0 & \sqrt{2} & 0 & 0 & 2 & 0 \\ 1 & -1 & 0 & \sqrt{2} & 0 & 0 & -2 & 0 \\ 1 & -1 & 0 & -\sqrt{2} & 0 & 0 & 0 & 2 \\ 1 & -1 & 0 & -\sqrt{2} & 0 & 0 & 0 & -2 \end{pmatrix}$$

- Can be computed by taking sums and differences
- Fast algorithms by recursively applying $\mathbf{Hr}_2$

Example: Subsampling

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

$$\mathbf{H}_x = \mathbf{H}_y = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \end{bmatrix}$$



Example: Subsampling (cont.)

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

$$\mathbf{H}_x = \mathbf{H}_y = \begin{bmatrix} 0.5 & 0 & 0 & 0 \\ 0.5 & 0 & 0 & 0 \\ 0 & 0.5 & 0 & 0 \\ 0 & 0.5 & 0 & 0 \\ 0 & 0 & 0.5 & 0 \\ 0 & 0 & 0.5 & 0 \\ 0 & 0 & 0 & 0.5 \\ 0 & 0 & 0 & 0.5 \end{bmatrix}$$



Side by side comparison

$$\mathbf{H}_x = \mathbf{H}_y =$$

$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \end{bmatrix}$$



$$\mathbf{H}_x = \mathbf{H}_y =$$

$$\begin{bmatrix} 0.5 & 0 & 0 & 0 \\ 0.5 & 0 & 0 & 0 \\ 0 & 0.5 & 0 & 0 \\ 0 & 0.5 & 0 & 0 \\ 0 & 0 & 0.5 & 0 \\ 0 & 0 & 0.5 & 0 \\ 0 & 0 & 0 & 0.5 \\ 0 & 0 & 0 & 0.5 \end{bmatrix}$$



Another example: filtering

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

$$\mathbf{H}_x = \mathbf{H}_y = \begin{bmatrix} 0.5 & 0 & & \dots & & & 0 \\ 0.5 & 0.5 & & & & & \\ 0 & 0.5 & 0.5 & & & & \\ & & 0.5 & 0.5 & \ddots & & \vdots \\ & & & 0.5 & 0.5 & & \\ \vdots & & & \ddots & 0.5 & 0.5 & \\ & & & & 0.5 & 0.5 & 0 \\ 0 & & \dots & & 0 & 0.5 & 0.5 \end{bmatrix}$$

- Shift-invariant operation (except for image boundary)

Shift-invariant systems and Toeplitz matrices

- For a separable, shift-invariant, linear system

$$h_x[x, \alpha] = h_{siv/x}[\alpha - x] \quad \text{and} \quad h_y[y, \beta] = h_{siv/y}[\beta - y]$$

- Matrices $\mathbf{H}_x$ and $\mathbf{H}_y$ are square, and Toeplitz matrices, e.g.,

$$\mathbf{H}_x = \begin{bmatrix} h_{siv/x}[0] & h_{siv/x}[1] & \cdots & h_{siv/x}[N-1] \\ h_{siv/x}[-1] & h_{siv/x}[0] & \cdots & h_{siv/x}[N-2] \\ \vdots & \vdots & & \vdots \\ h_{siv/x}[1-N] & h_{siv/x}[2-N] & \cdots & h_{siv/x}[0] \end{bmatrix}$$

- 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 2-d convolution

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

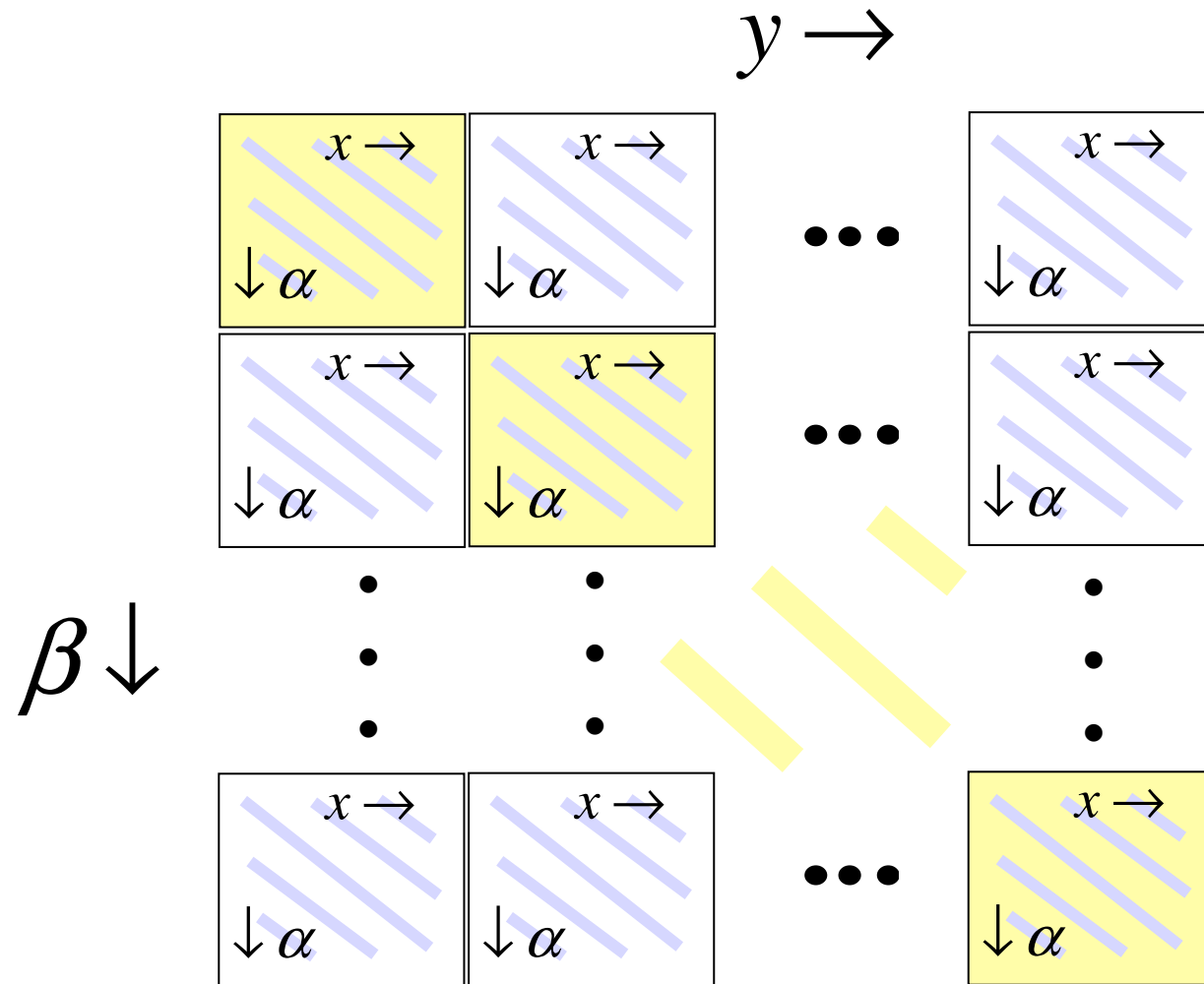
$$g[\alpha, \beta] = \sum_{x=0}^{N-1} \sum_{y=0}^{L-1} f[x, y] \cdot h_{\text{ziv}}[\alpha - x, \beta - y]$$

- Viewed as a matrix operation . . .

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

. . . H is a block Toeplitz matrix

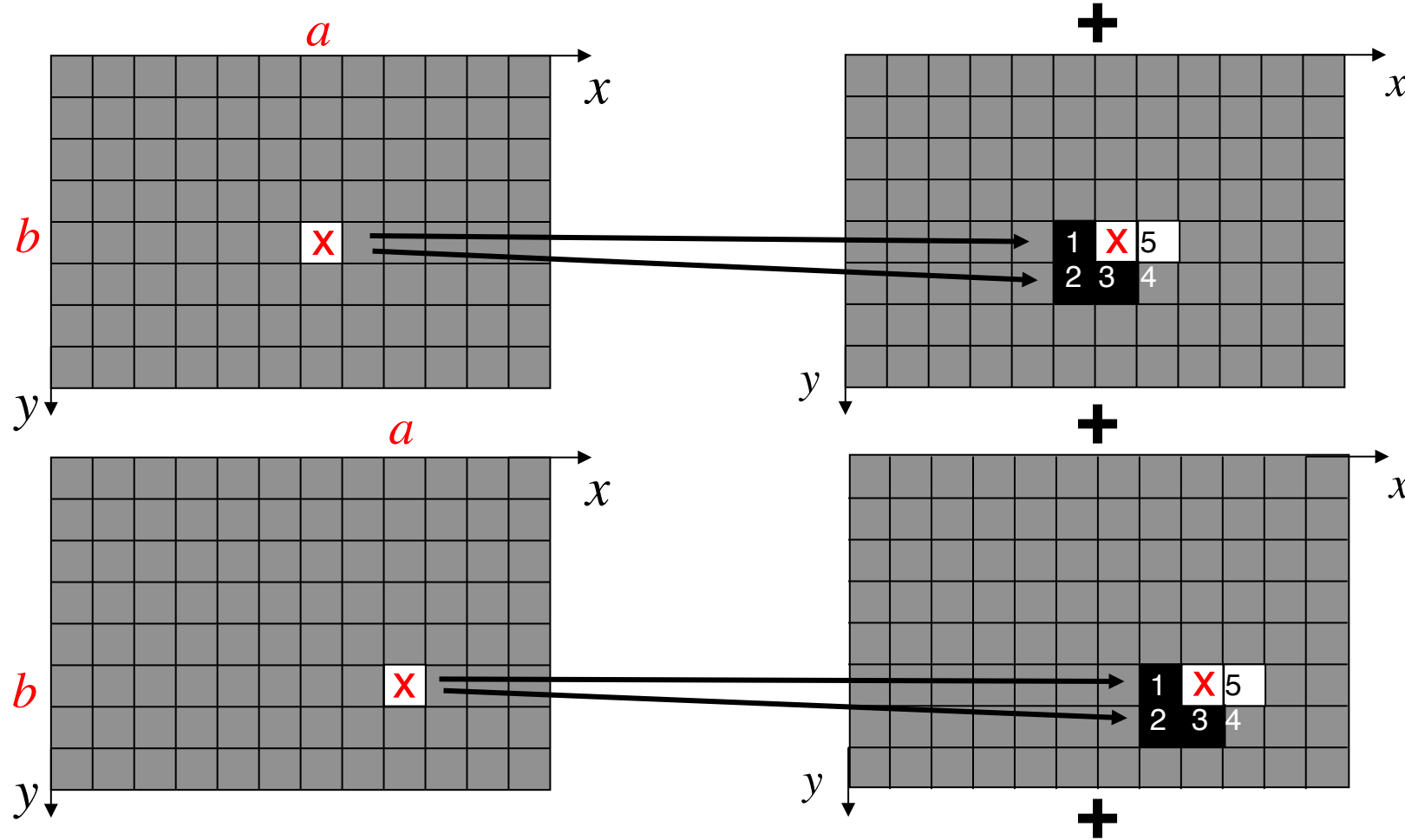
Structure of H for non-separable convolution



Convolution: superposition of impulse responses

$$f[a,b]\delta[x-a,y-b]$$

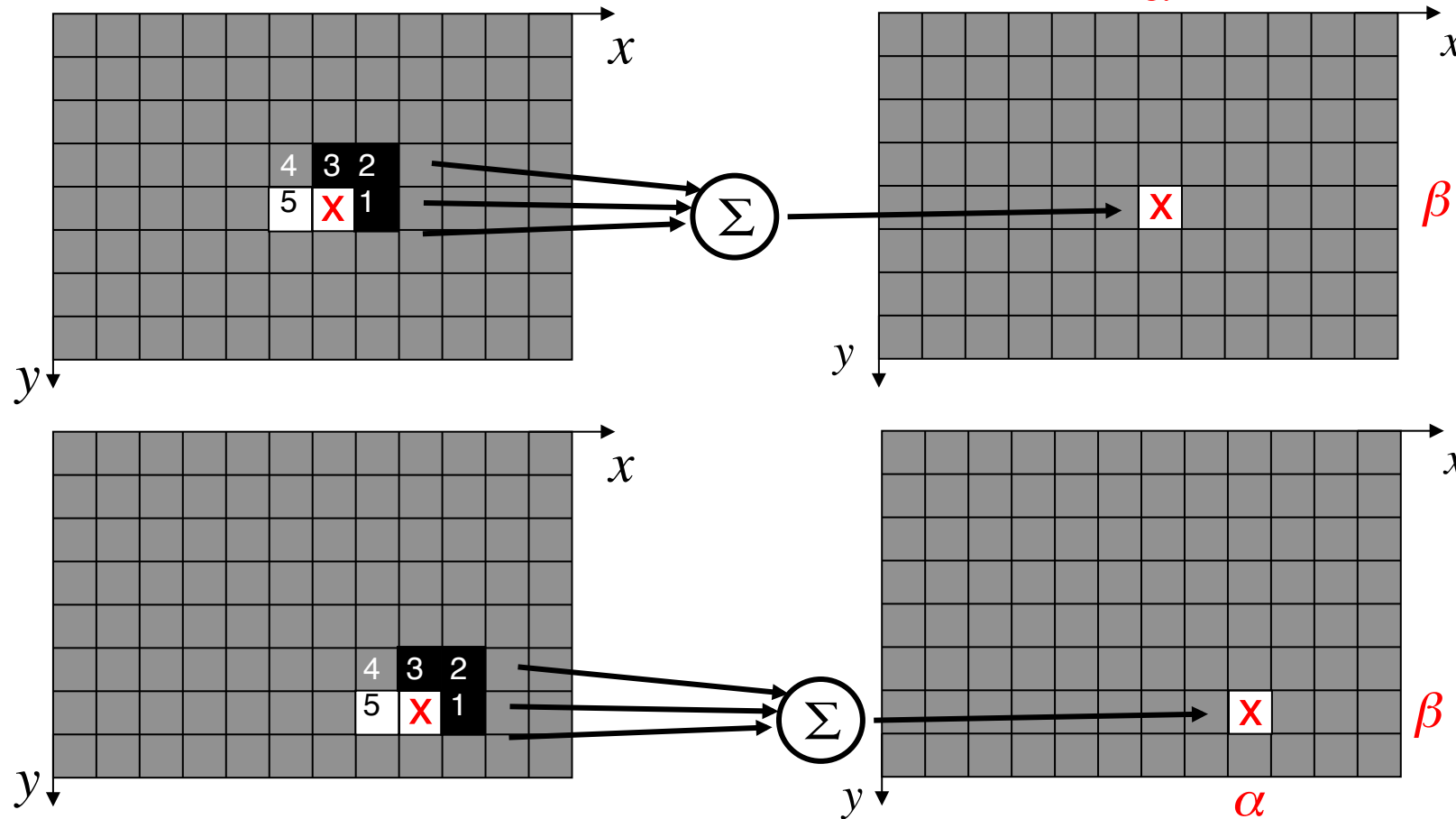
$$g[x,y] = h_{siv}[x-a,y-b]f[a,b]$$



Convolution: linear combination of neighboring pixel values

$$f[x, y] \cdot h_{siv}[\alpha - x, \beta - y]$$

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



Convolution examples



Original
Bike



Bike blurred by convolution
Impulse response „box filter“

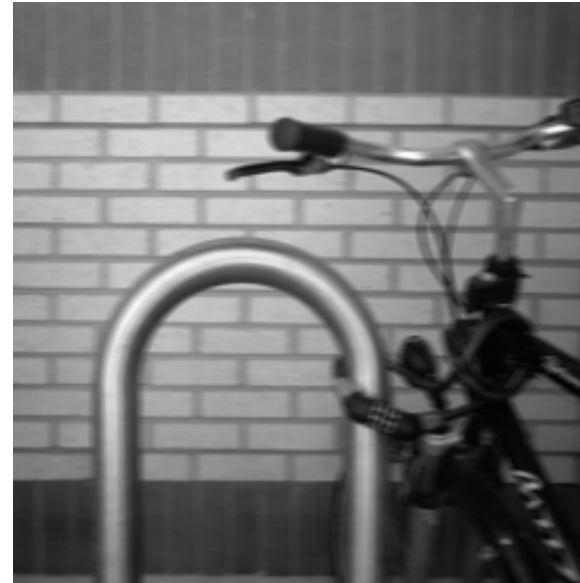
$$\frac{1}{25} \begin{pmatrix} 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & [1] & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \end{pmatrix}$$



Convolution examples



Original
Bike



Bike blurred horizontally
Filter impulse response

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



Convolution examples



Original
Bike

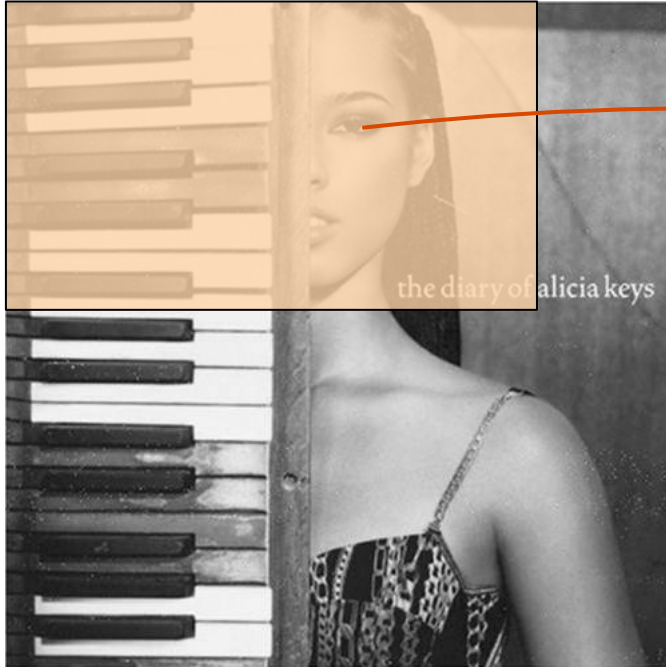


Bike blurred vertically
Filter impulse response

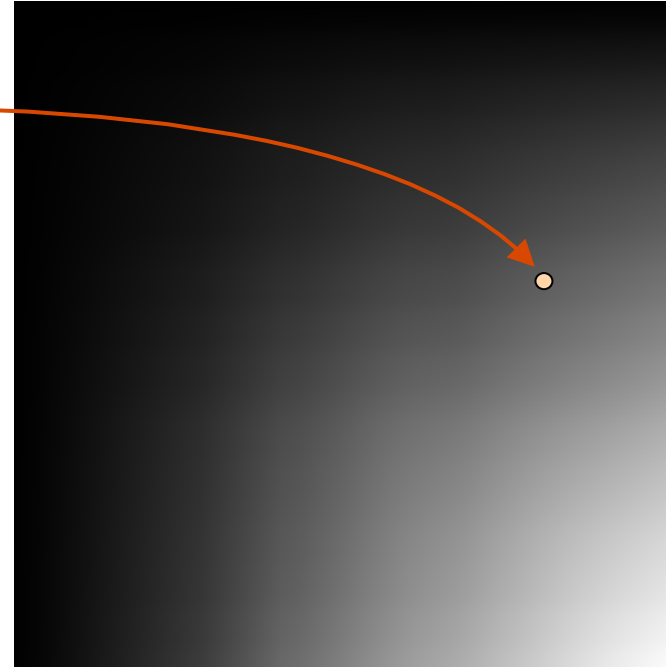
$$\frac{1}{5} \begin{pmatrix} 1 \\ 1 \\ [1] \\ 1 \\ 1 \end{pmatrix}$$



Integral image



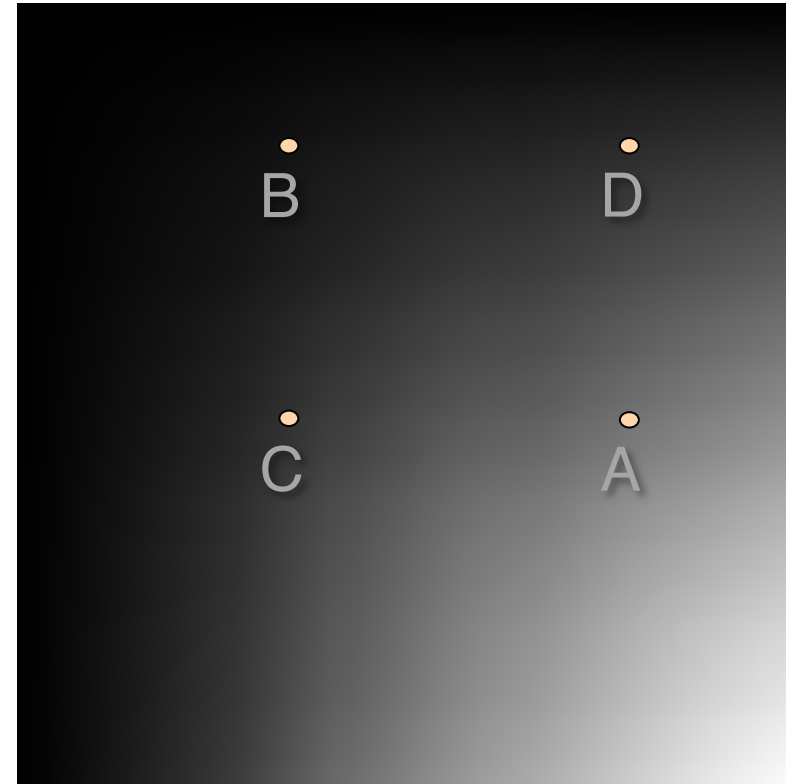
$$f[x, y]$$



$$i[x, y] = \sum_{u=0}^x \sum_{v=0}^y f[u, v]$$

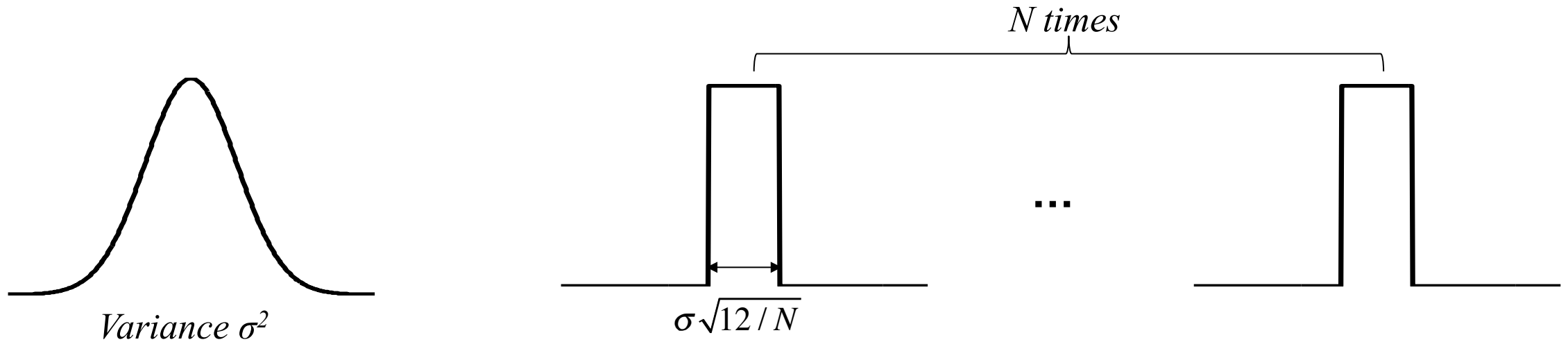
$$\begin{aligned} j[x, y] &= j[x, y-1] + f[x, y] \\ i[x, y] &= i[x-1, y] + j[x, y] \end{aligned}$$

Box filtering with integral image



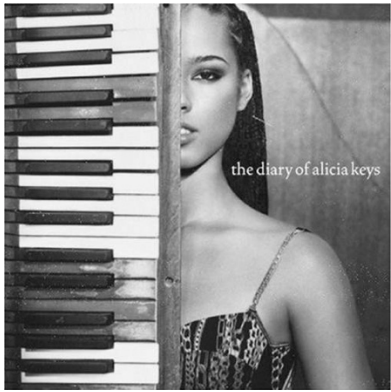
$$\text{Sum} = A + B - C - D$$

Gaussian filtering by repeated box filtering

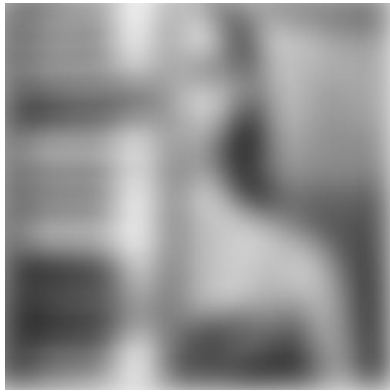


- Approximate convolution with Gaussian kernel by convolution with box kernel, repeated N times
- Convolution with box kernel can be computed efficiently using integral image

Gaussian filtering by repeated box filtering



Original image
500x500

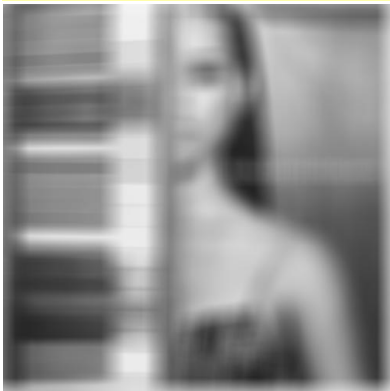


Gaussian filtered
 $\sigma = 20$, 81x81
kernel

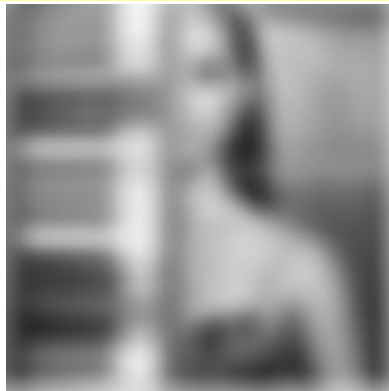
Direct implementation:
6561 multiplications and
6560 additions per pixel

x-y separable filtering:
162 multiplications and
160 additions per pixel

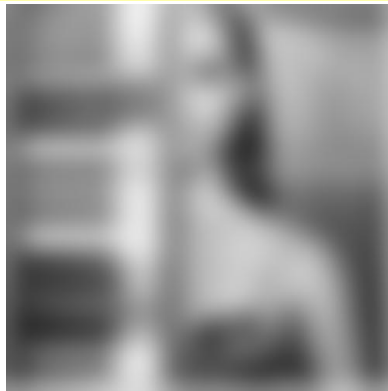
20 additions or subtractions per pixel



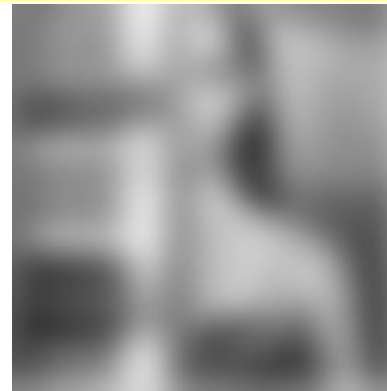
Box filtered
after $N = 1$



Box filtered
after $N = 2$



Box filtered
after $N = 3$



Box filtered
after $N = 4$



1-d discrete-time Fourier transform

- Given a 1-d sequence $s[k]$, $k \in \mathbf{Z} = \{\dots, -1, 0, 1, 2, 3, \dots\}$
- Fourier transform

$$S(e^{j\omega}) = \sum_{k=-\infty}^{\infty} s[k] e^{-j\omega k} \quad \omega \in \mathfrak{R}$$

- Fourier transform is periodic with 2π
- Inverse Fourier transform

$$s[k] = \frac{1}{2\pi} \int_{-\pi}^{\pi} S(e^{j\omega}) e^{j\omega k} d\omega$$

2-d discrete-space Fourier transform

- Given a 2-d array of image samples

$$s[m, n], \quad m, n \in \mathbf{Z}^2$$

- Fourier transform

$$S(e^{j\omega_x}, e^{j\omega_y}) = \sum_{m=-\infty}^{\infty} \sum_{n=-\infty}^{\infty} s[m, n] e^{-j\omega_x m - j\omega_y n} \quad \omega_x, \omega_y \in \mathbb{R}^2$$

- Fourier transform is 2π -periodic both in ω_x and ω_y
- Inverse Fourier transform

$$s[m, n] = \frac{1}{4\pi^2} \int_{-\pi}^{\pi} \int_{-\pi}^{\pi} S(e^{j\omega_x}, e^{j\omega_y}) e^{j\omega_x m + j\omega_y n} d\omega_x d\omega_y$$

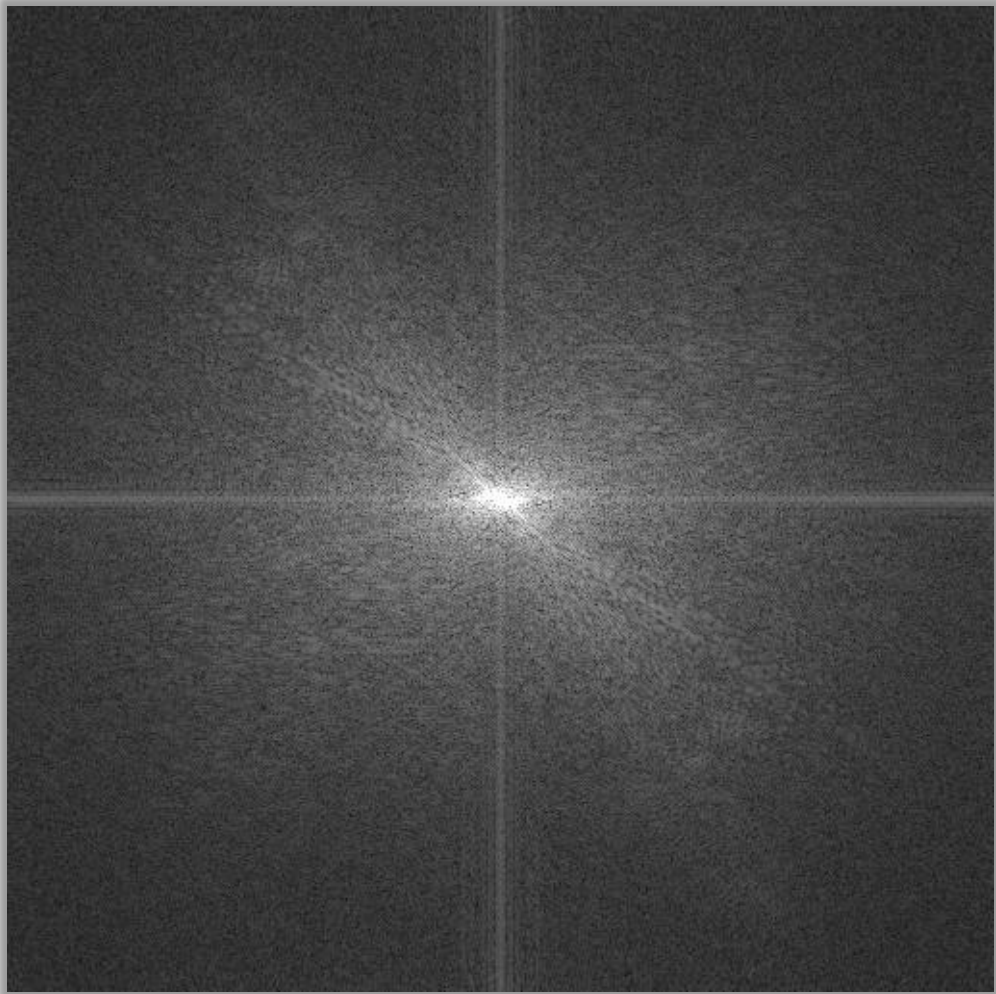
Fourier transform – interesting properties

- DFT matrix is orthonormal
- conceptually write as DFT-matrix vector multiplication $O(N^2)$ but in practice use FFT $O(N \log N)$
- FT of symmetric function is always real; FT of real function is always symmetric
- convolution theorem is super useful:

$$x * g = F^{-1} \{ F \{ x \} \cdot F \{ g \} \}$$

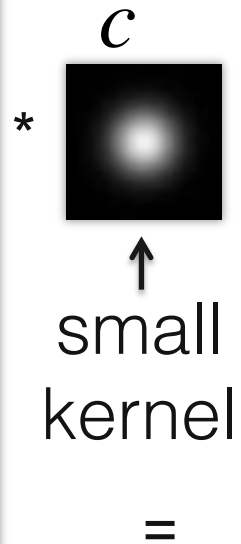
Discrete Fourier Transform

- What is this?



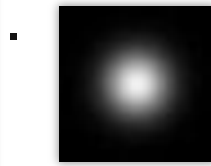
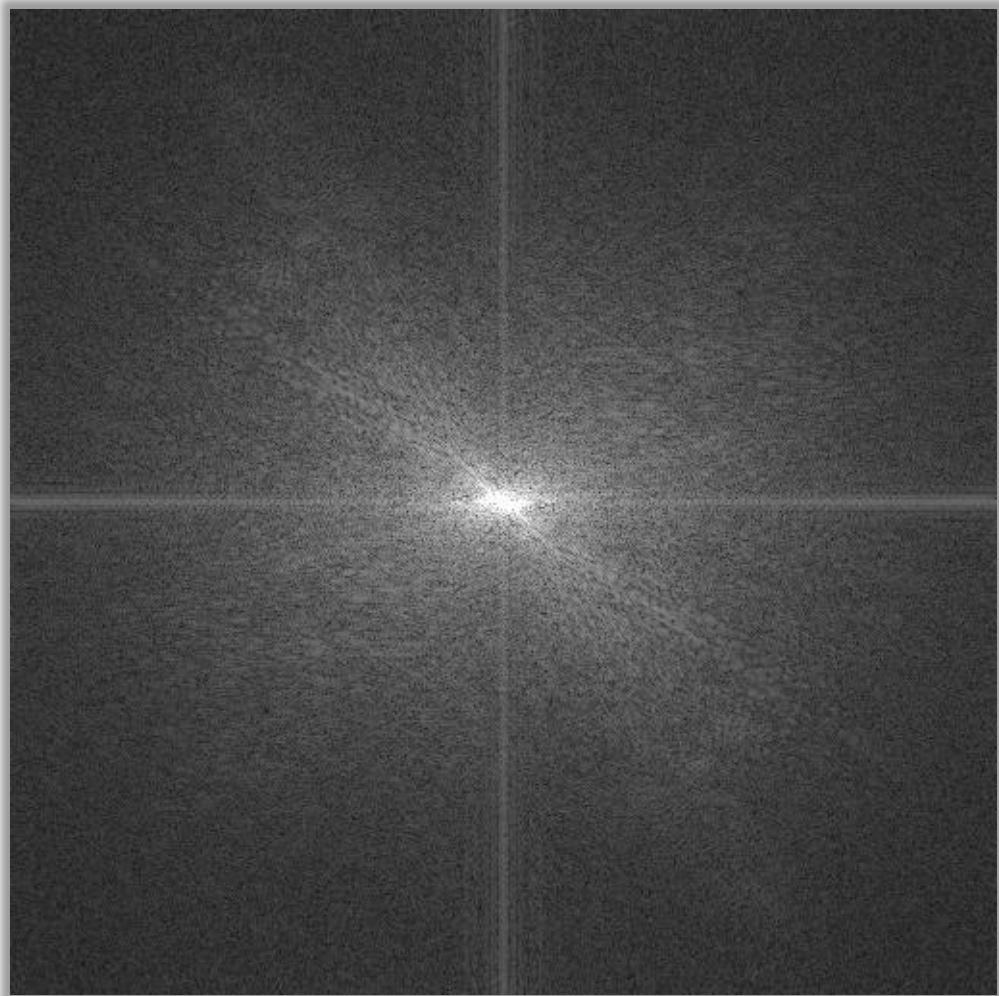
Filtering – Low-pass Filter

- low-pass filter: convolution in primal domain $b = x * c$



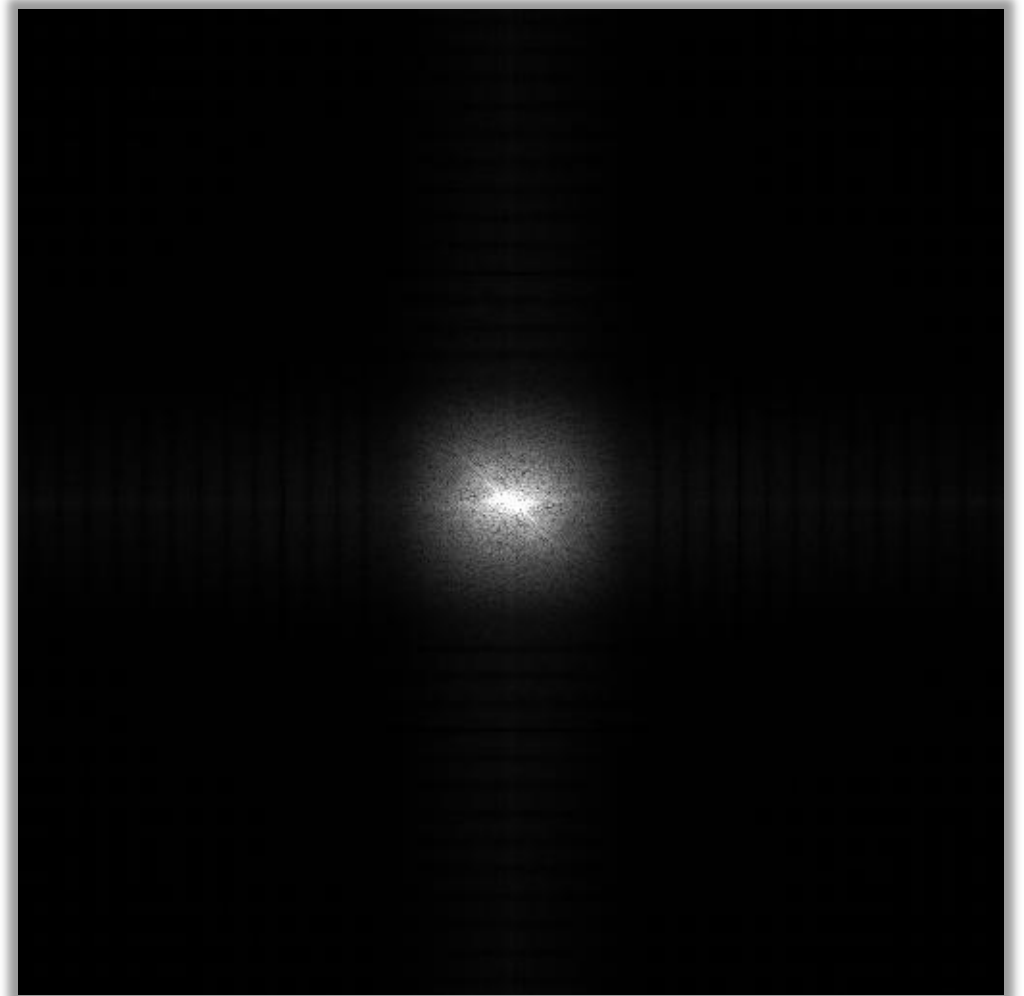
Filtering – Low-pass Filter

- low-pass filter: multiplication in frequency domain $F\{b\} = F\{x\} \cdot F\{c\}$



↑
big

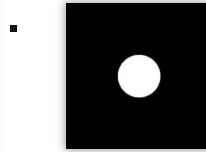
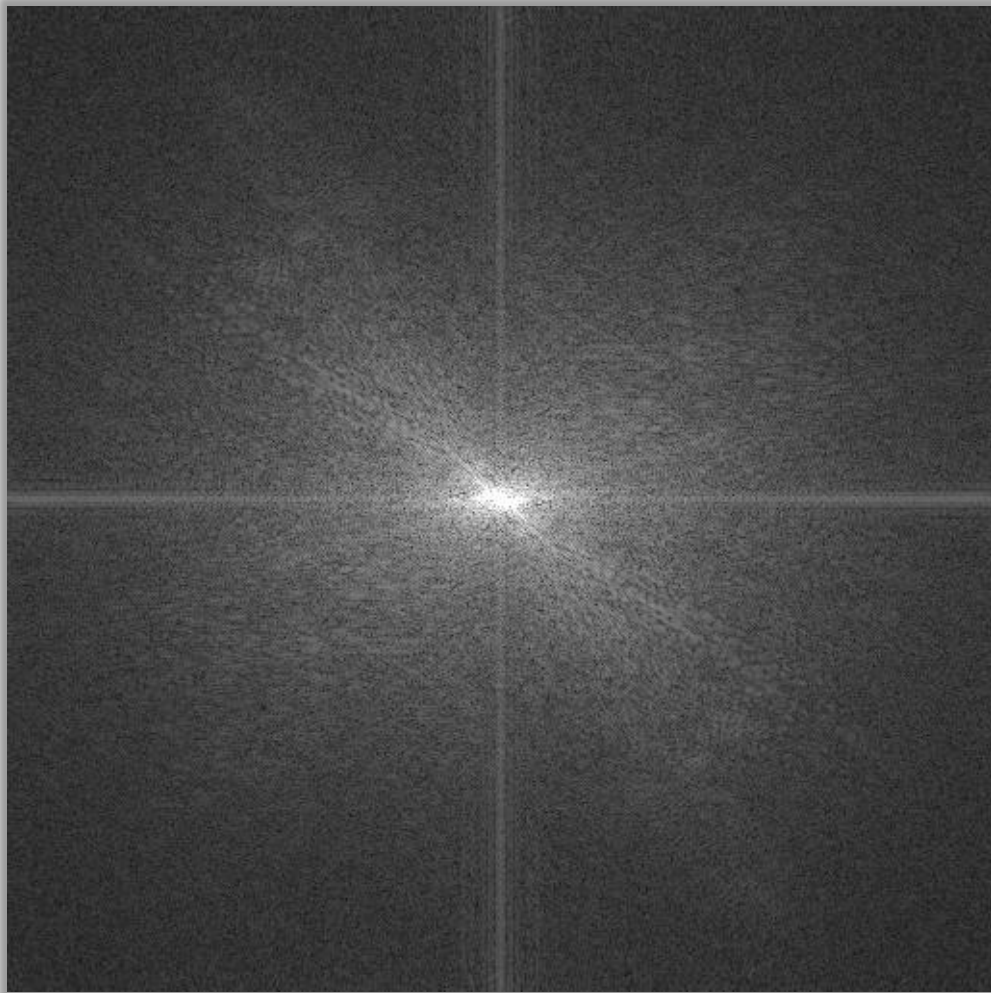
=



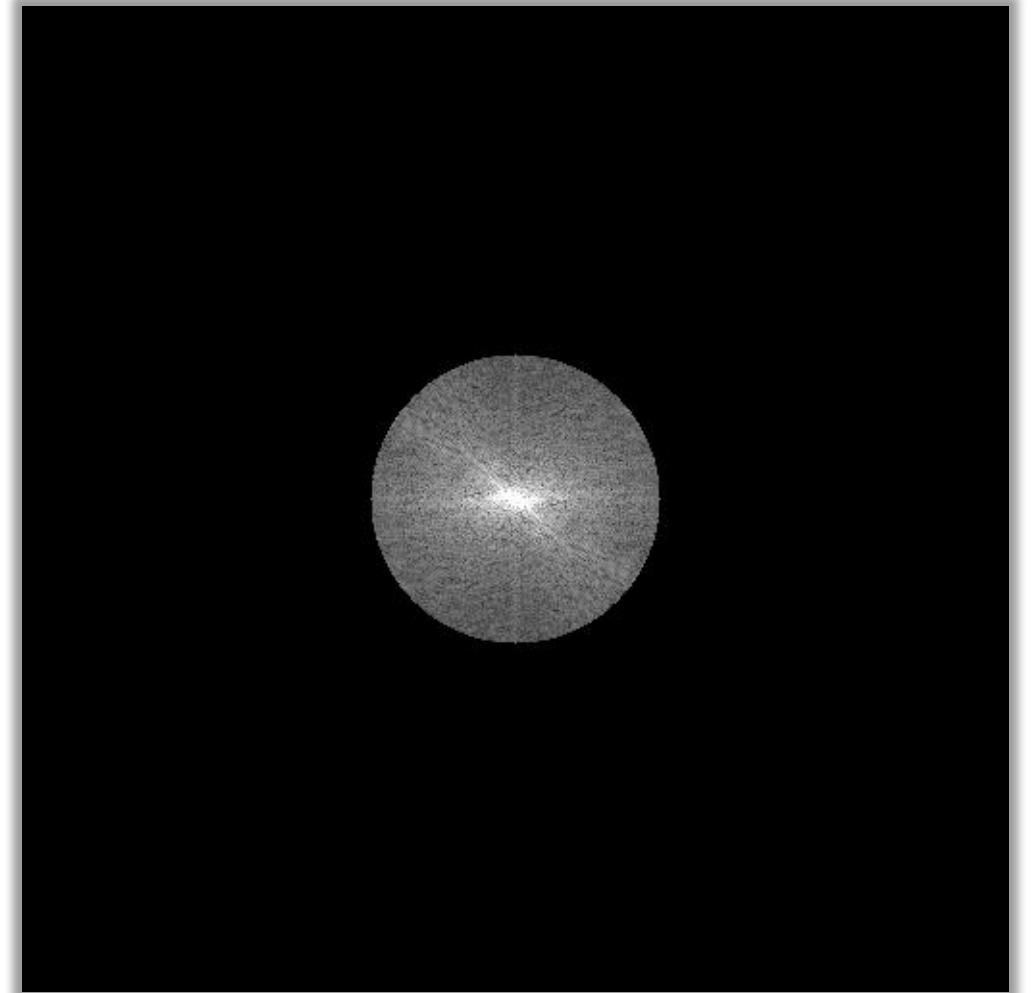
Filtering – Low-pass Filter

$$F\{b\} = F\{x\} \cdot F\{c\}$$

- low-pass filter: hard cutoff



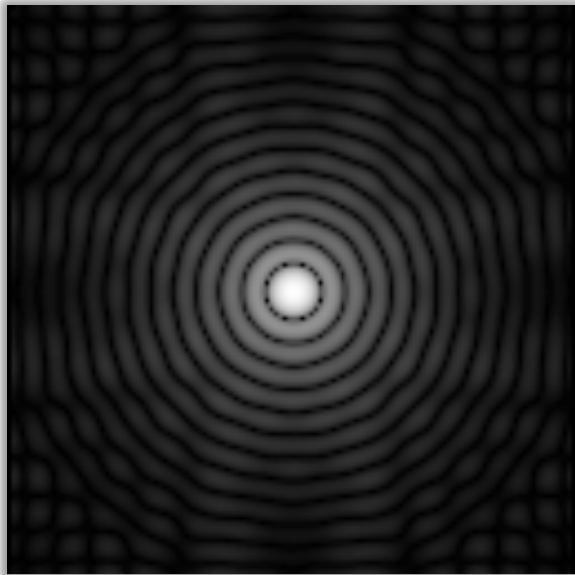
=



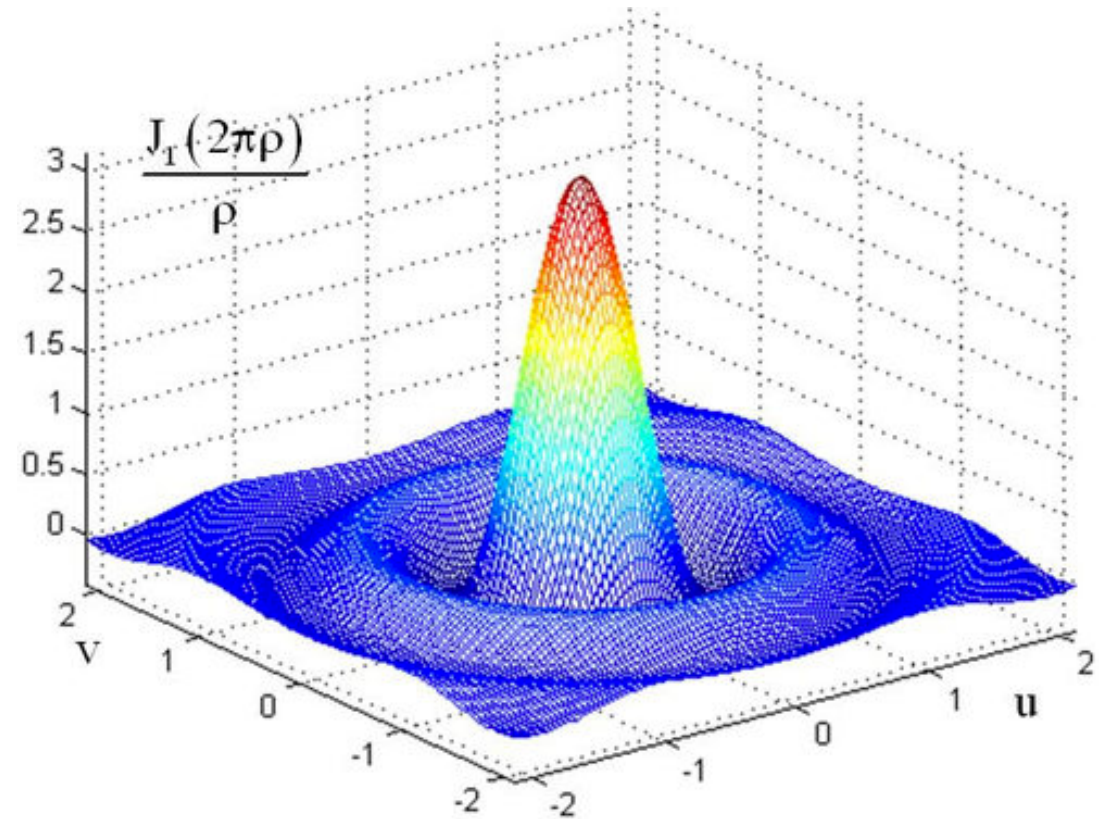
Filtering – Low-pass Filter

- Bessel function of the first kind or “jinc”

$$F^{-1} \left\{ \text{[Black square with white circle]} \right\}$$

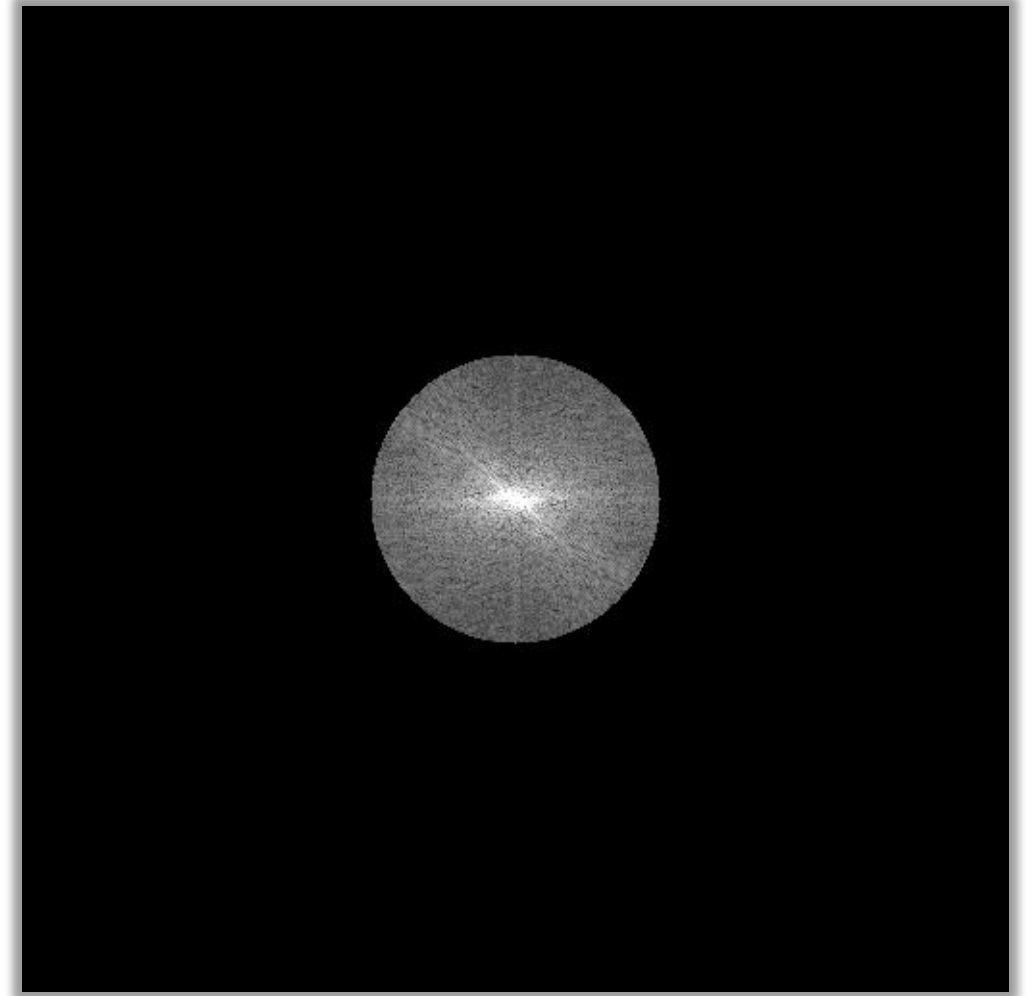


imagemagick.org



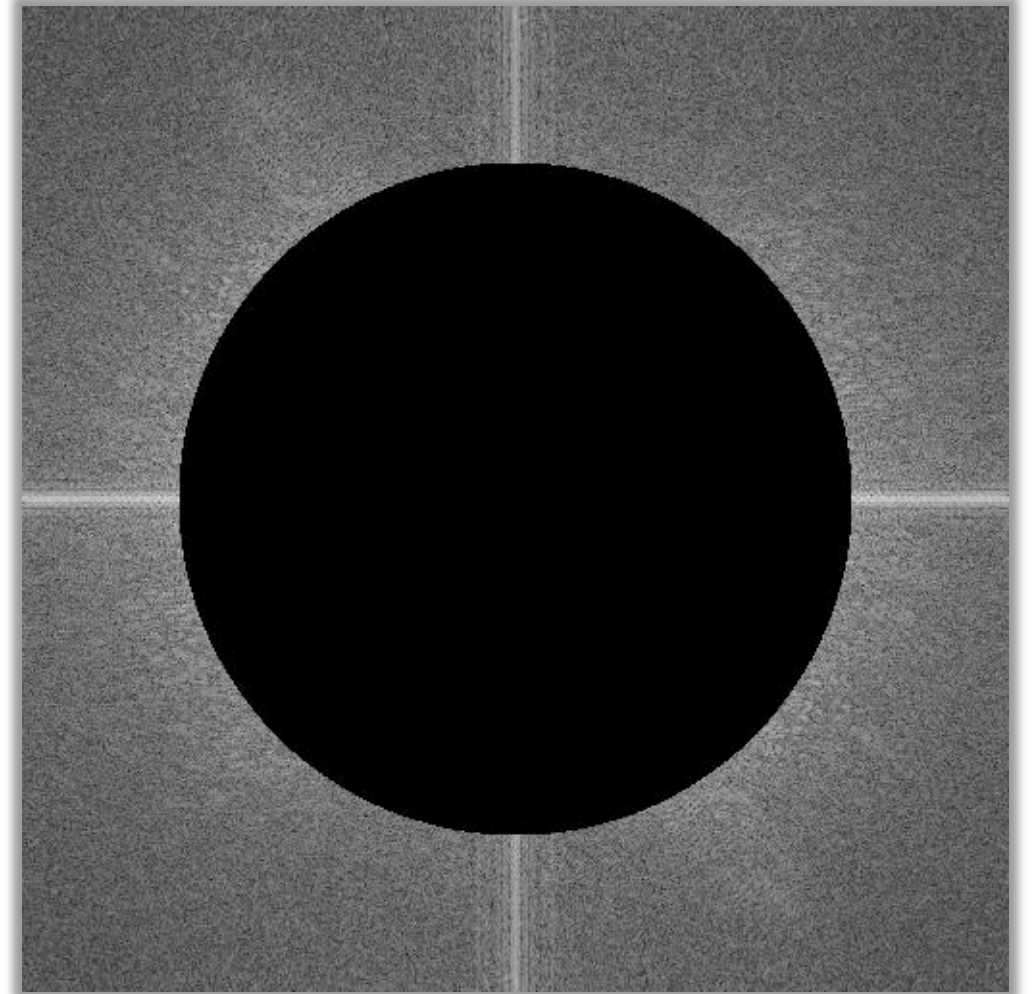
Filtering – Low-pass Filter

- hard frequency filters often introduce ringing



Filtering – High-pass Filter

- sharpening (possibly with ringing, but don't see any here)



Filtering – Unsharp Masking

- sharpening (without ringing): unsharp masking, e.g. in Photoshop

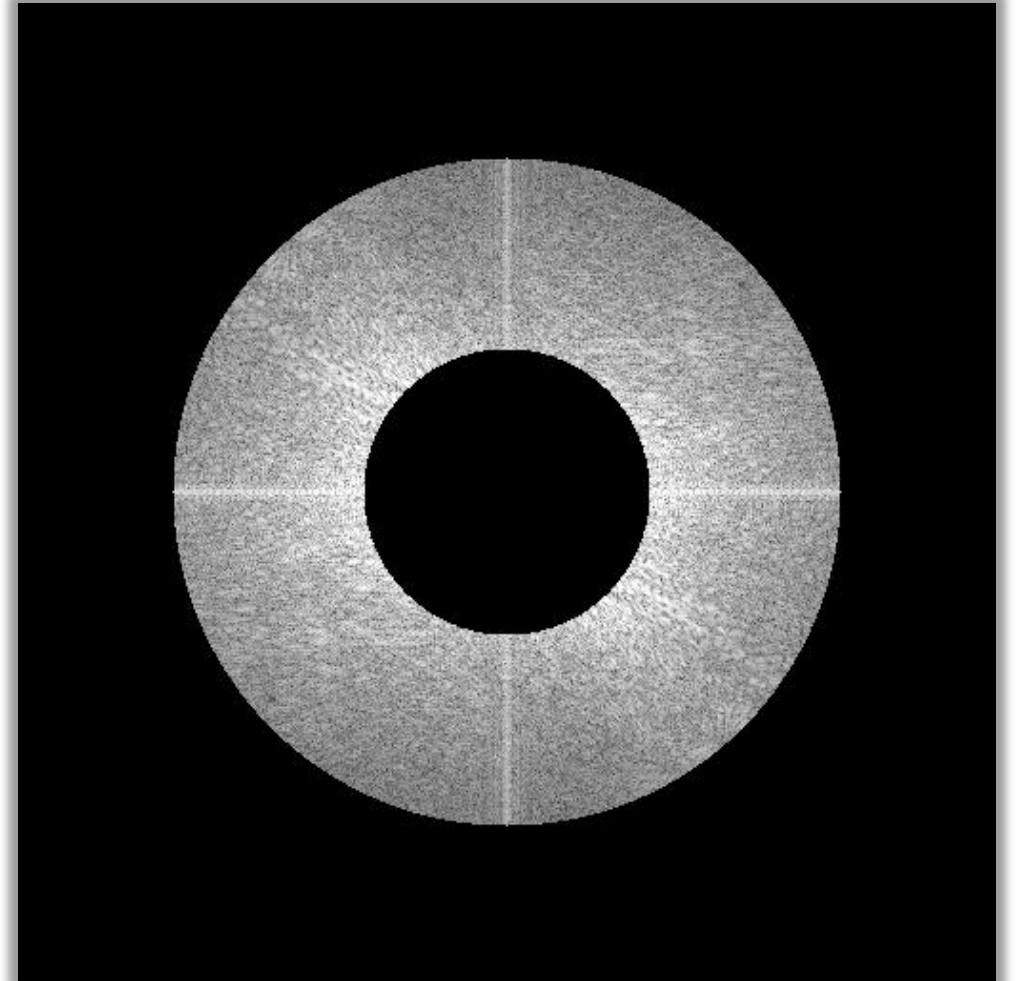


$$b = x * (\delta - c_{\text{lowpass_gauss}}) = x - x * c_{\text{lowpass_gauss}}$$

or

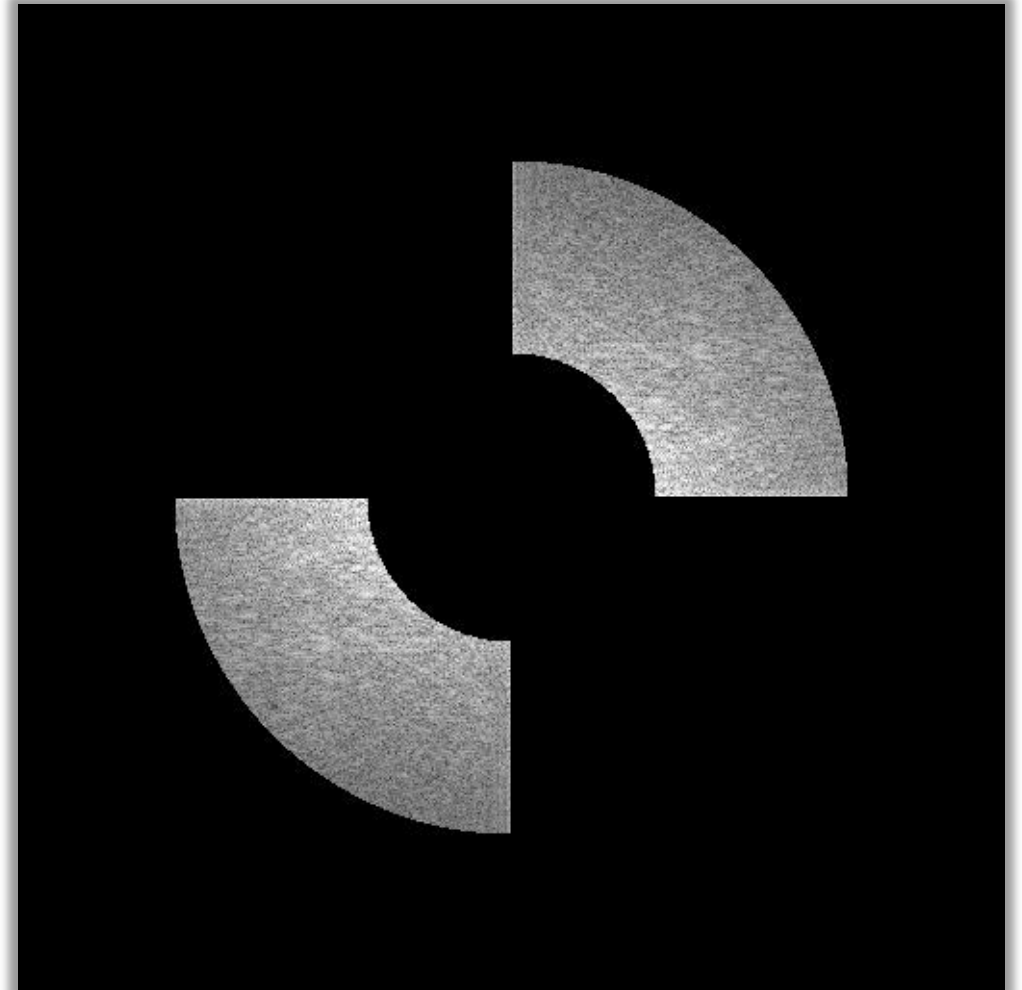
$$b = x * (\delta + c_{\text{highpass}}) = x + x * c_{\text{highpass}}$$

Filtering – Band-pass Filter



Filtering – Oriented Band-pass Filter

- edges with specific orientation (e.g., hat) are gone!



Optical Filtering with Fourier Optics

- can do all of this optically (with coherent light)!
- Fourier optics – not part of this course

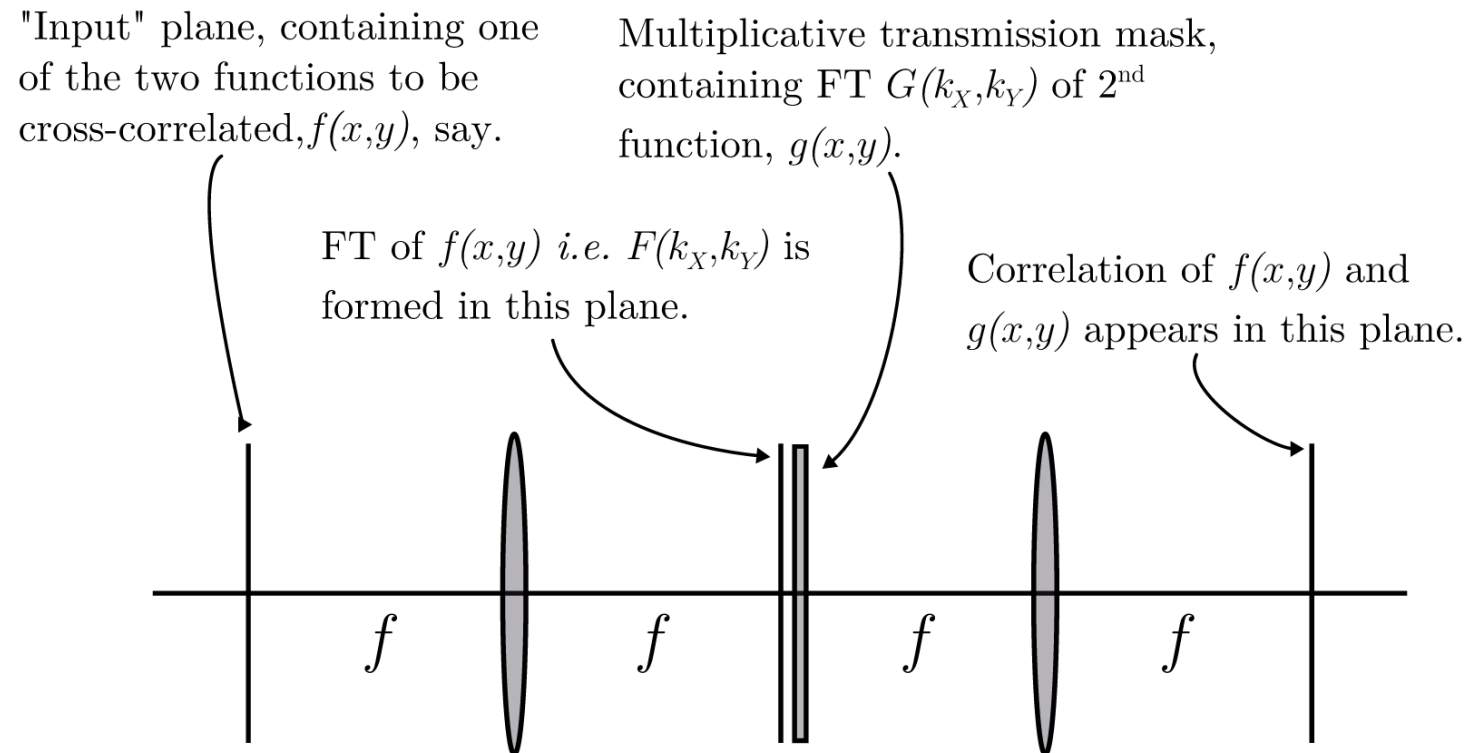


Image Downsampling (& Upsampling)

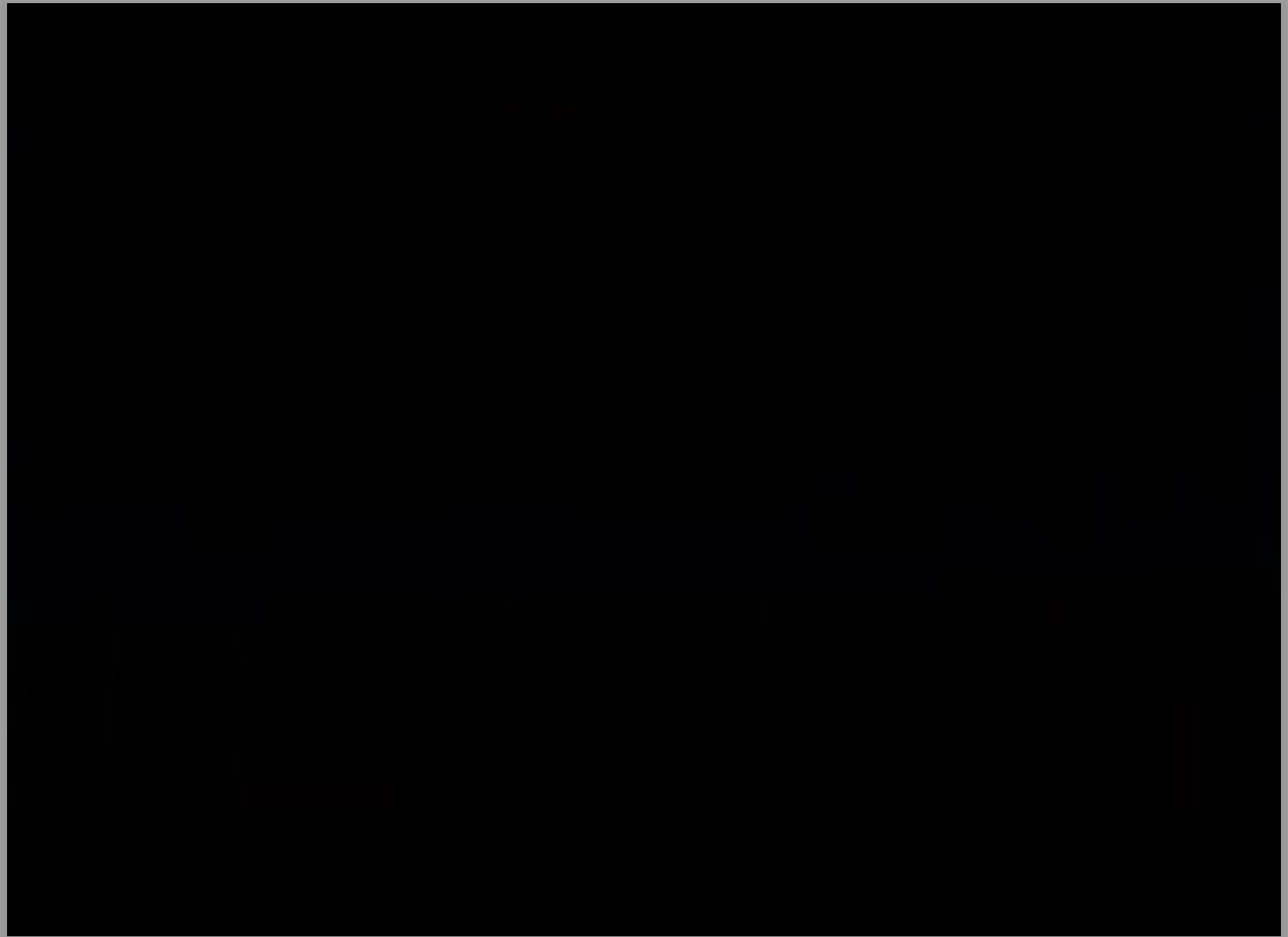
- “anti-aliasing” → before re-sampling, apply appropriate filter!
- how much filtering? Nyquist-Shannon sampling theorem:

$$f_s \geq 2f_{\max}$$

Other Forms of Aliasing

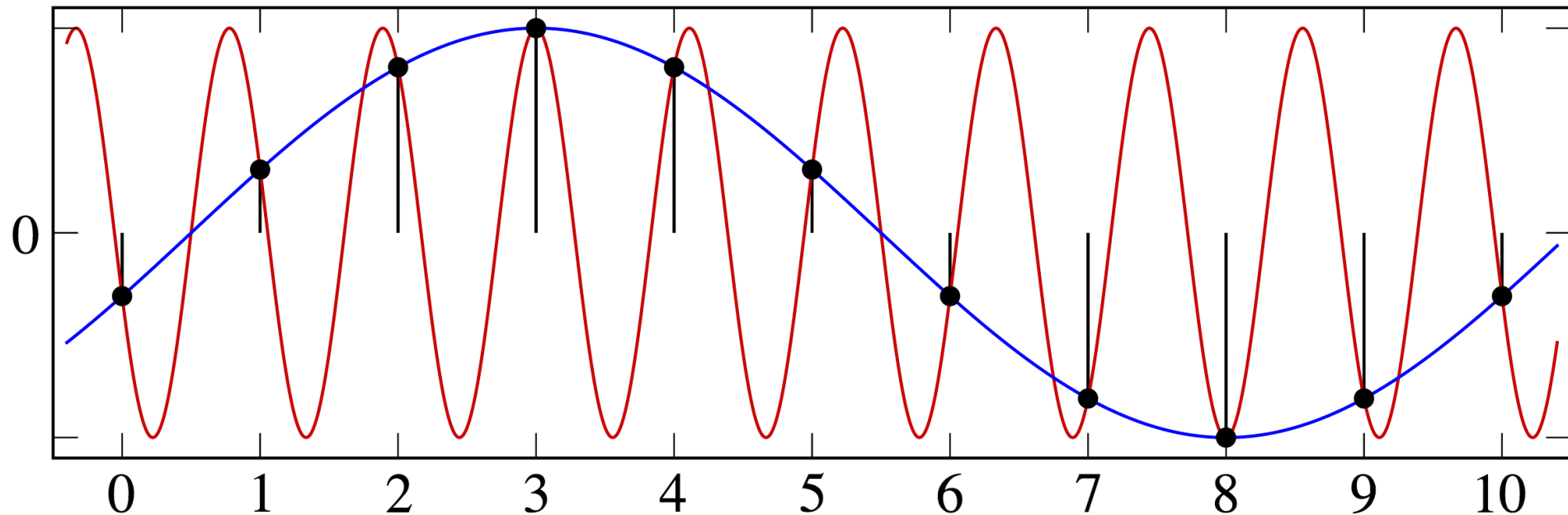
- wagon wheel effect
(temporal aliasing):

[youtube.com/watch?v=jHS9JGkEOmA](https://www.youtube.com/watch?v=jHS9JGkEOmA)



Other Forms of Aliasing

- wagon wheel effect (temporal aliasing)
- sampling frequency was lower than $2f_{\max}$

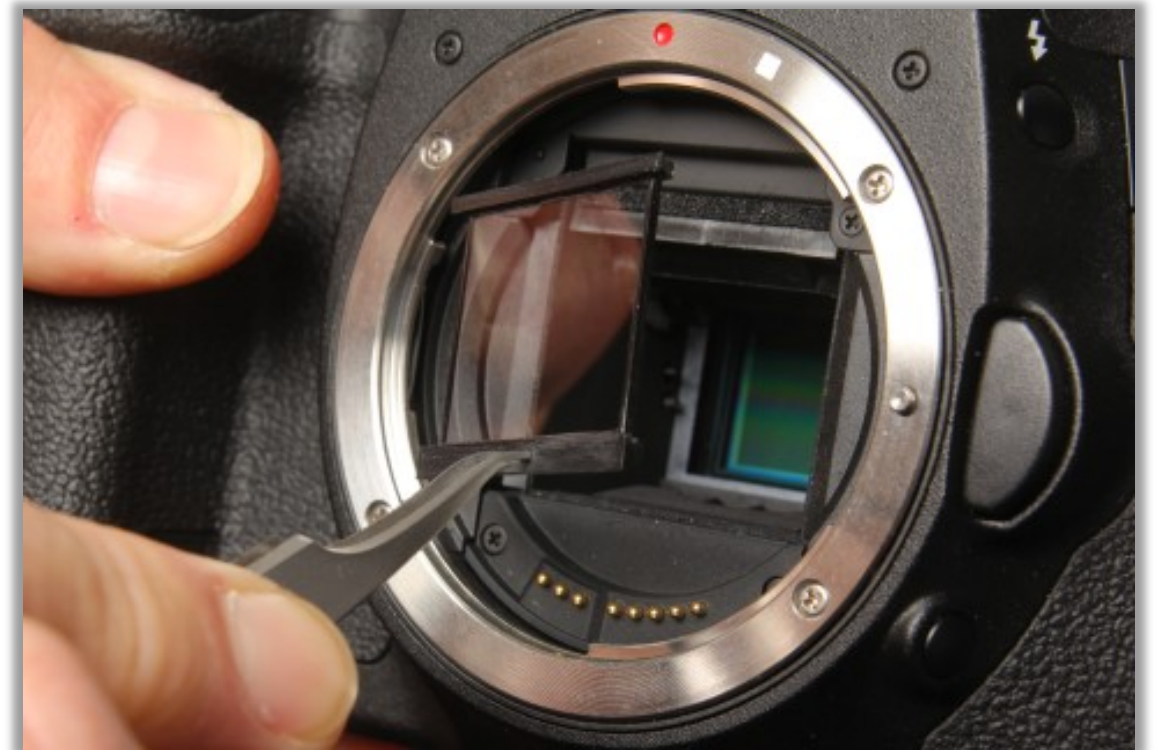


Other Forms of Aliasing

- photography – optical AA filter removed (“hot rodding” camera)



John Shafer



mosaicengineering.com

Other Forms of Aliasing

- photography – optical AA filter removed (“hot rodding” camera)

without AA filter



with AA filter (standard)



Other Forms of Aliasing

- photography – optical AA filter removed (“hot rodding” camera)

without AA filter



with AA filter (standard)



5x5 box filter revisited



Original
Bike

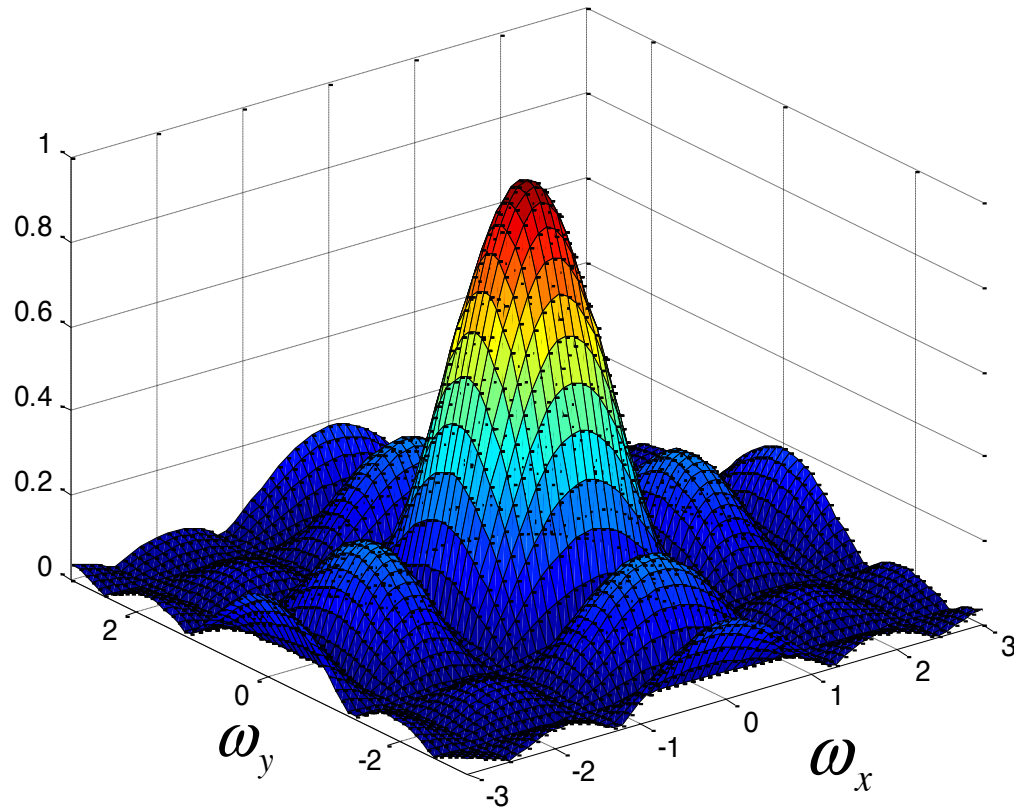


Bike blurred by convolution
Impulse response „box filter“

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



Frequency response of 5x5 lowpass filter



$$\begin{aligned} H(e^{j\omega_x}, e^{j\omega_y}) &= \sum_{m=-\infty}^{\infty} \sum_{n=-\infty}^{\infty} h[m, n] e^{-j\omega_x m - j\omega_y n} \\ &= \frac{1}{25} \sum_{m=-2}^2 \sum_{n=-2}^2 e^{-j\omega_x m - j\omega_y n} = \frac{1}{25} \sum_{m=-2}^2 e^{-j\omega_x m} \sum_{n=-2}^2 e^{-j\omega_y n} \\ &= \frac{1}{25} (1 + 2\cos\omega_x + 2\cos(2\omega_x)) (1 + 2\cos\omega_y + 2\cos(2\omega_y)) \end{aligned}$$

Separable filter:
1-d frequency responses
are multiplied



Horizontal lowpass filter

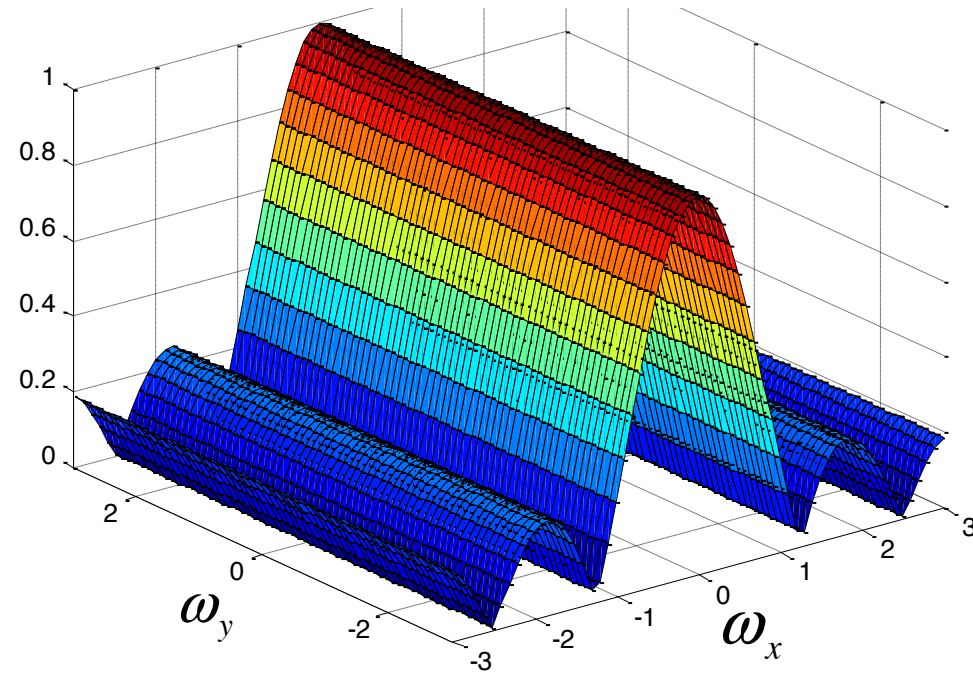


Bike blurred horizontally

Filter impulse response

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

$$H(e^{j\omega_x}, e^{j\omega_y}) = \frac{1}{5} (1 + 2\cos\omega_x + 2\cos(2\omega_x))$$



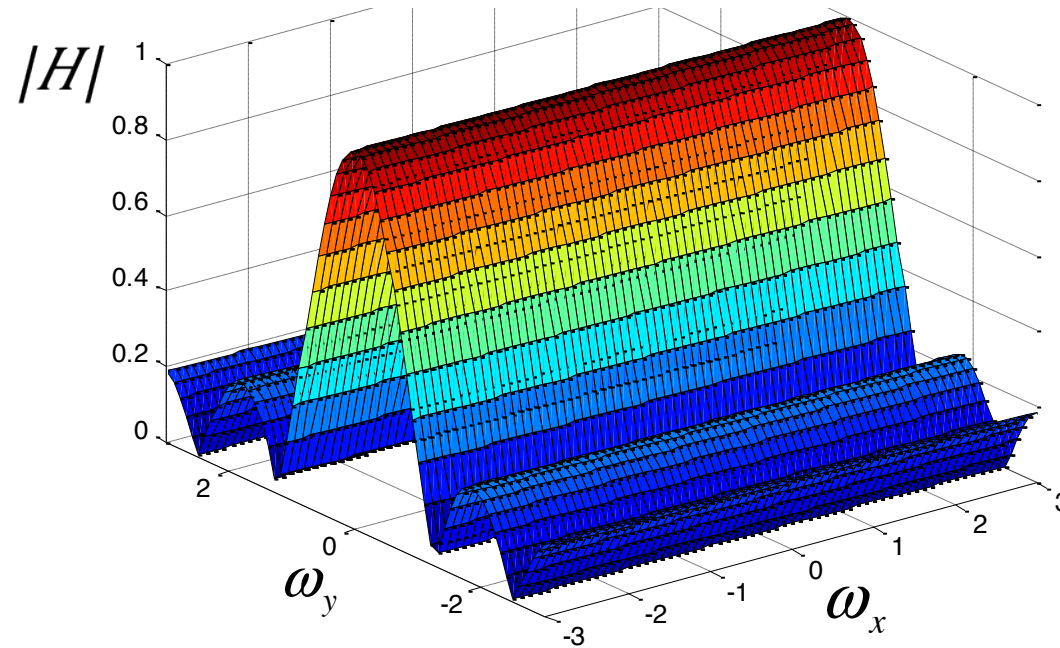
Vertical lowpass filter



Bike blurred vertically
Filter impulse response

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

$$H(e^{j\omega_x}, e^{j\omega_y}) = \frac{1}{5} (1 + 2\cos\omega_y + 2\cos(2\omega_y))$$



Sharpening filter



Original
Bike

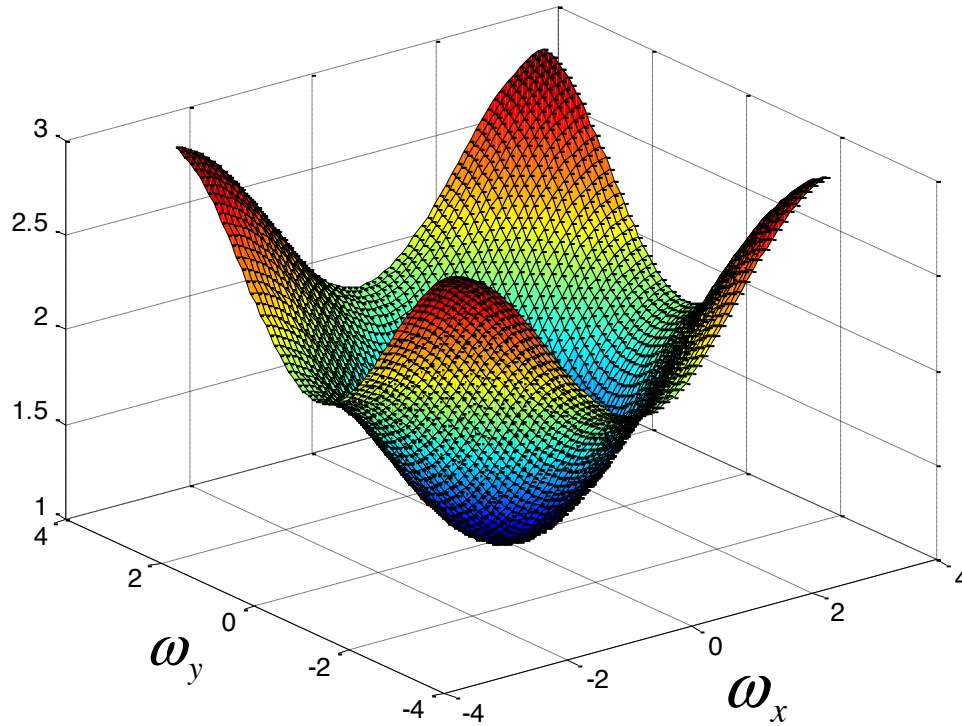


Bike sharpened
Filter impulse response

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



Frequency response of sharpening filter



$$\begin{aligned} H(e^{j\omega_x}, e^{j\omega_y}) &= \sum_{m=-\infty}^{\infty} \sum_{n=-\infty}^{\infty} h[m, n] e^{-j\omega_x m - j\omega_y n} \\ &= \frac{1}{4} \left(8 - e^{-j\omega_x} - e^{j\omega_x} - e^{-j\omega_y} - e^{j\omega_y} \right) \\ &= 2 - \frac{1}{2} \cos \omega_x - \frac{1}{2} \cos \omega_y \end{aligned}$$



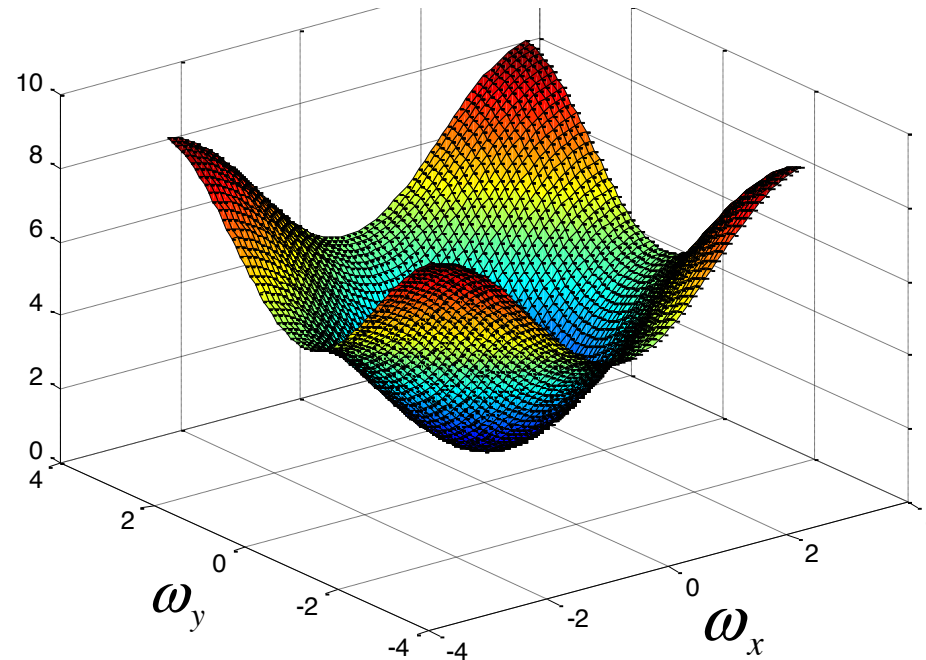
More aggressive sharpening



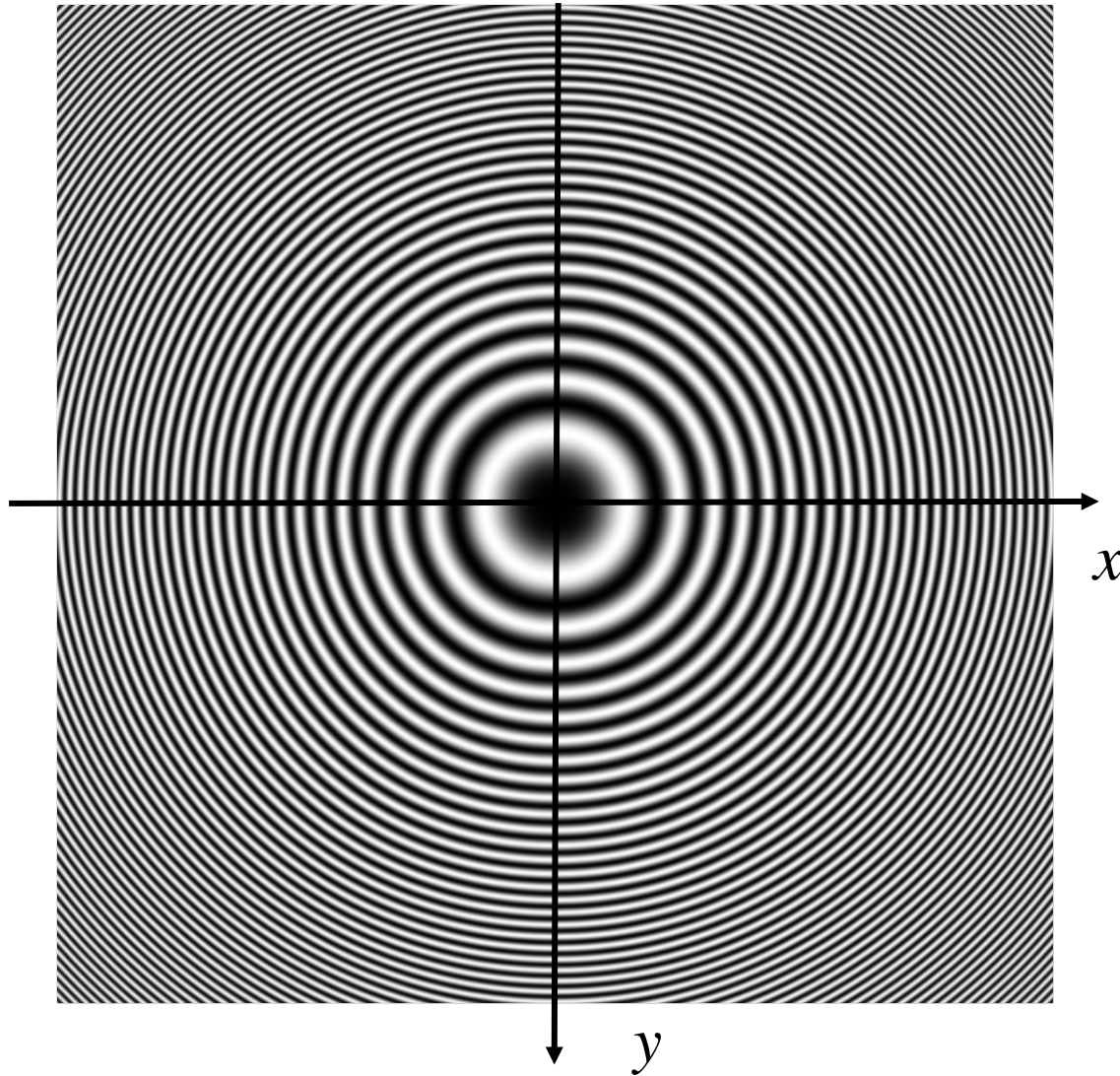
Bike sharpened
Filter impulse response

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

$$H(e^{j\omega_x}, e^{j\omega_y}) = 5 - 2\cos\omega_x - 2\cos\omega_y$$



Zoneplate pattern to visualize frequency plane



Equation to generate pattern:

$$s(x, y) = \hat{s} \cos(a_x x^2 + a_y y^2) + s_0$$

Local frequency at (x, y)

$$\frac{\partial}{\partial x}(a_x x^2 + a_y y^2) = 2a_x x$$

$$\frac{\partial}{\partial y}(a_x x^2 + a_y y^2) = 2a_y y$$

$$s[x, y] \xrightleftharpoons[\text{sampling}]{\text{interpolation}} s(x, y)$$

Sampling interpretation of 2-d discrete-space Fourier transform

- How is the Fourier transform of $s[m,n]$ related to the Fourier transform of the continuous signal

$$s(x, y) = \sum_{m=-\infty}^{\infty} \sum_{n=-\infty}^{\infty} s[m, n] \delta_2(x - m, y - n)$$

continuous

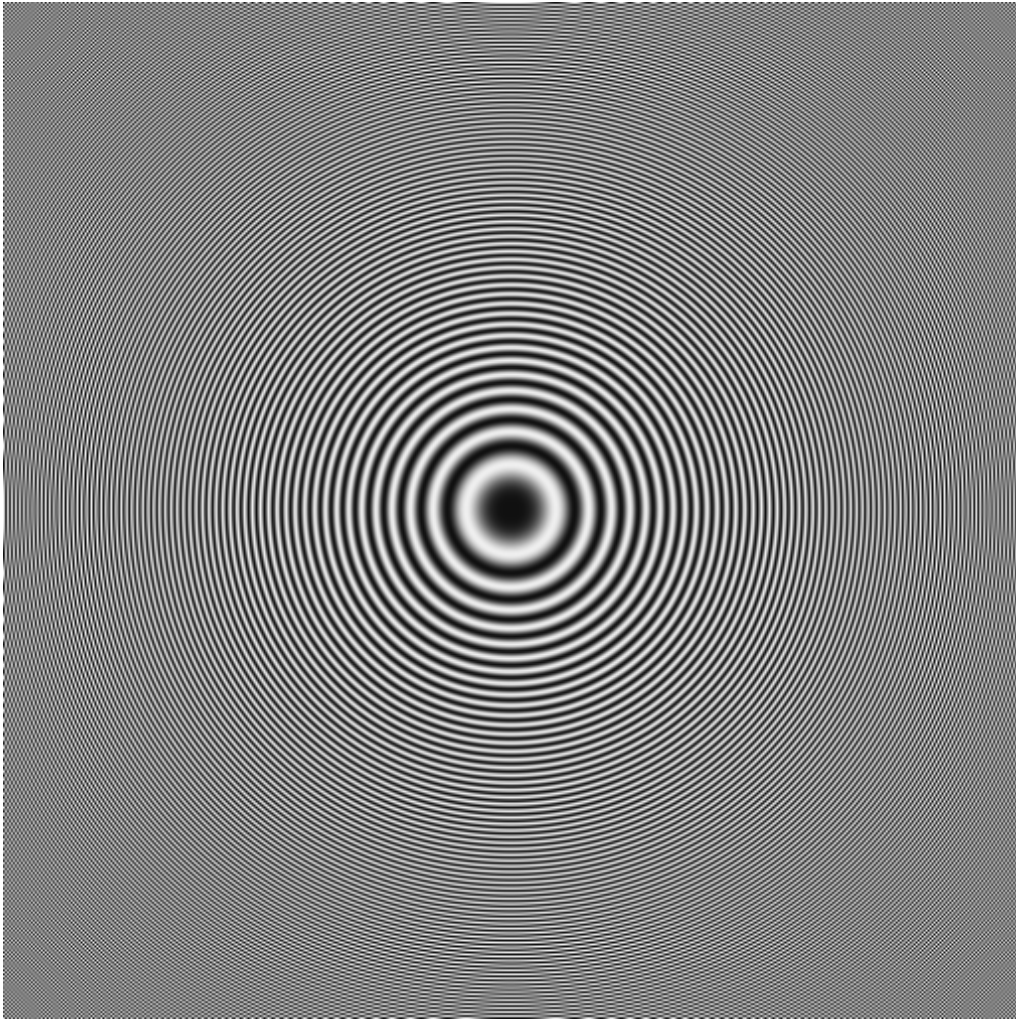
discrete

„continuous“
delta function

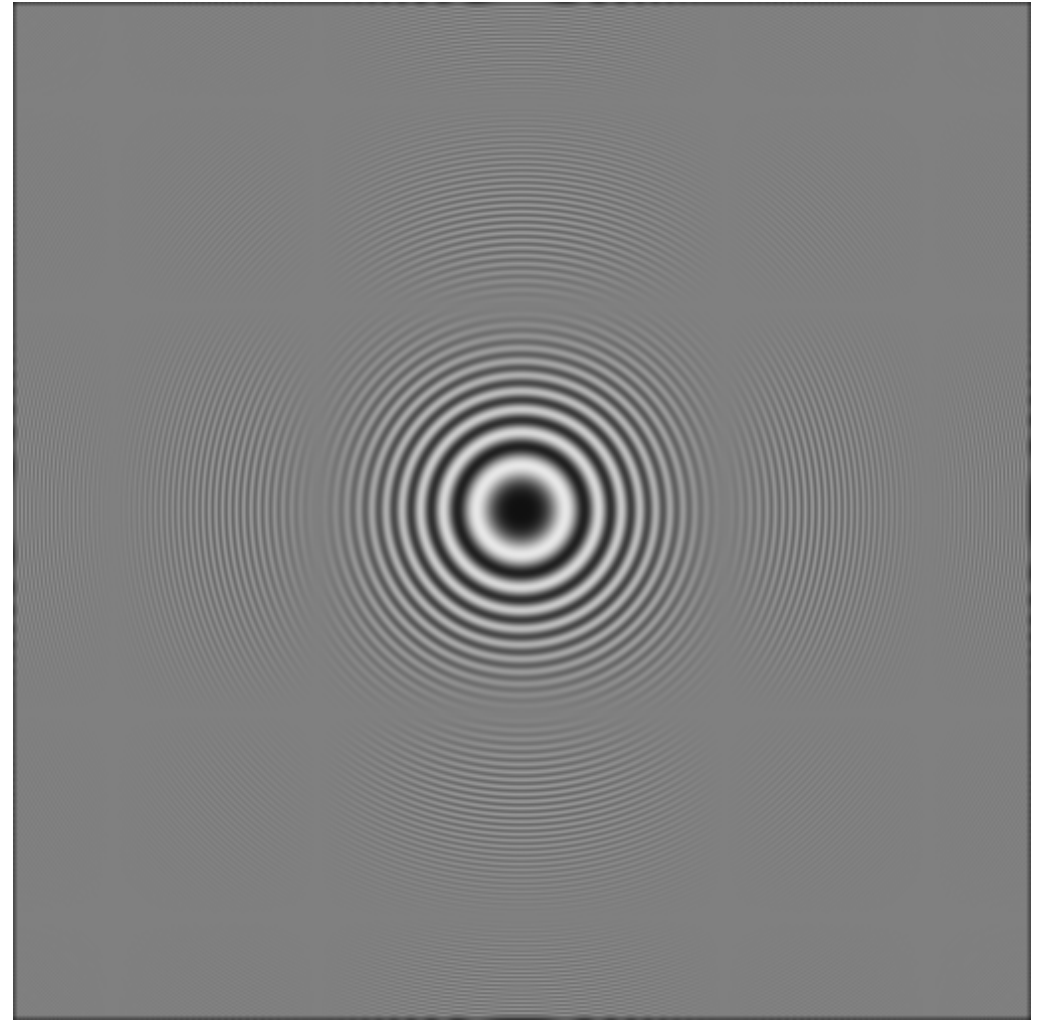
- Continuous-space 2-d Fourier transform

$$\begin{aligned} \hat{S}(\omega_x, \omega_y) &= \int \int \sum_{m=-\infty}^{\infty} \sum_{n=-\infty}^{\infty} s[m, n] \delta_2(x - m, y - n) e^{-j\omega_x x - j\omega_y y} dx dy \\ &= \sum_{m=-\infty}^{\infty} \sum_{n=-\infty}^{\infty} s[m, n] e^{-j\omega_x m - j\omega_y n} = S(e^{j\omega_x}, e^{j\omega_y}) \end{aligned}$$

Lowpass filtering

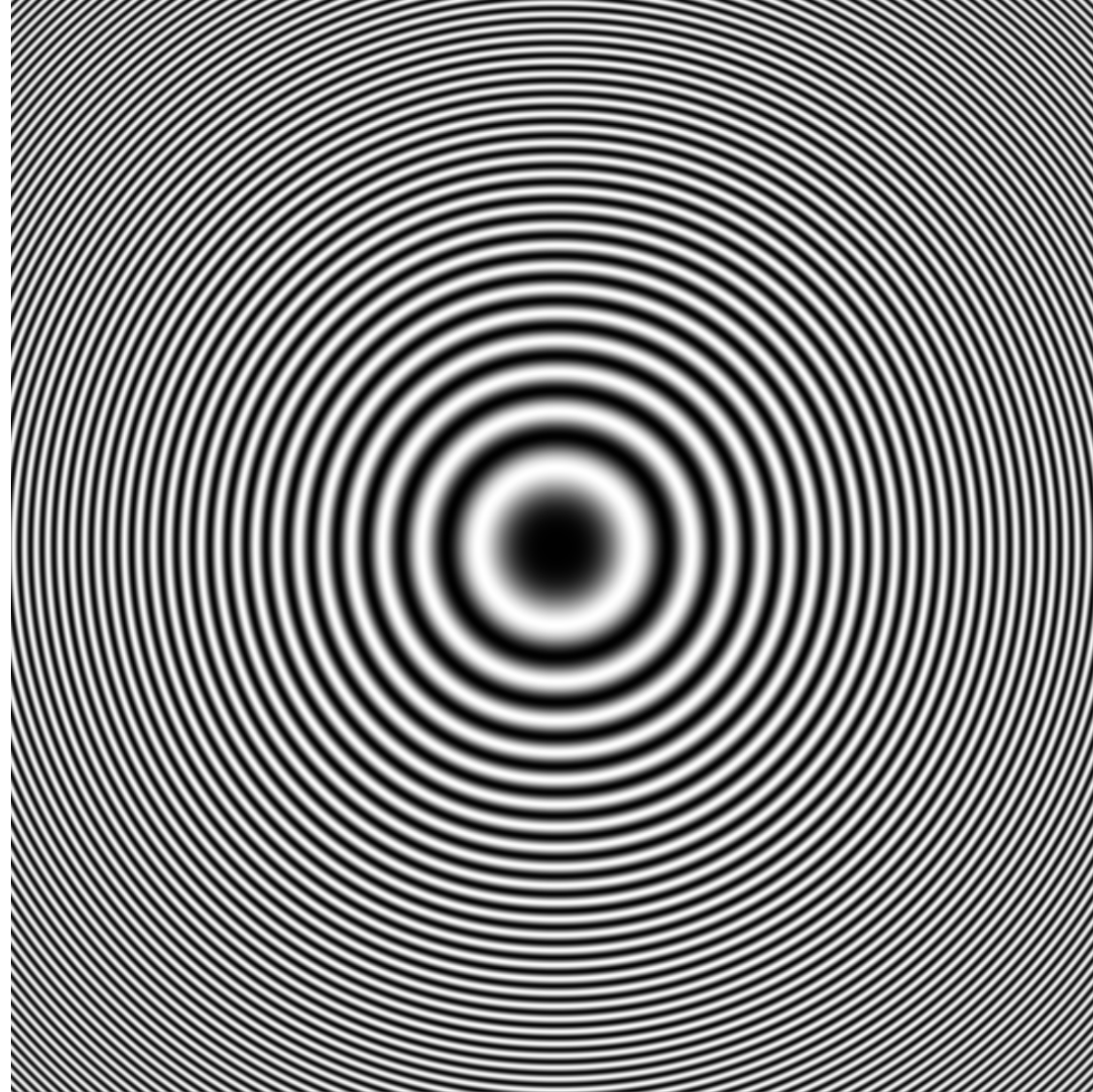


Original *Zoneplate* (512x512)

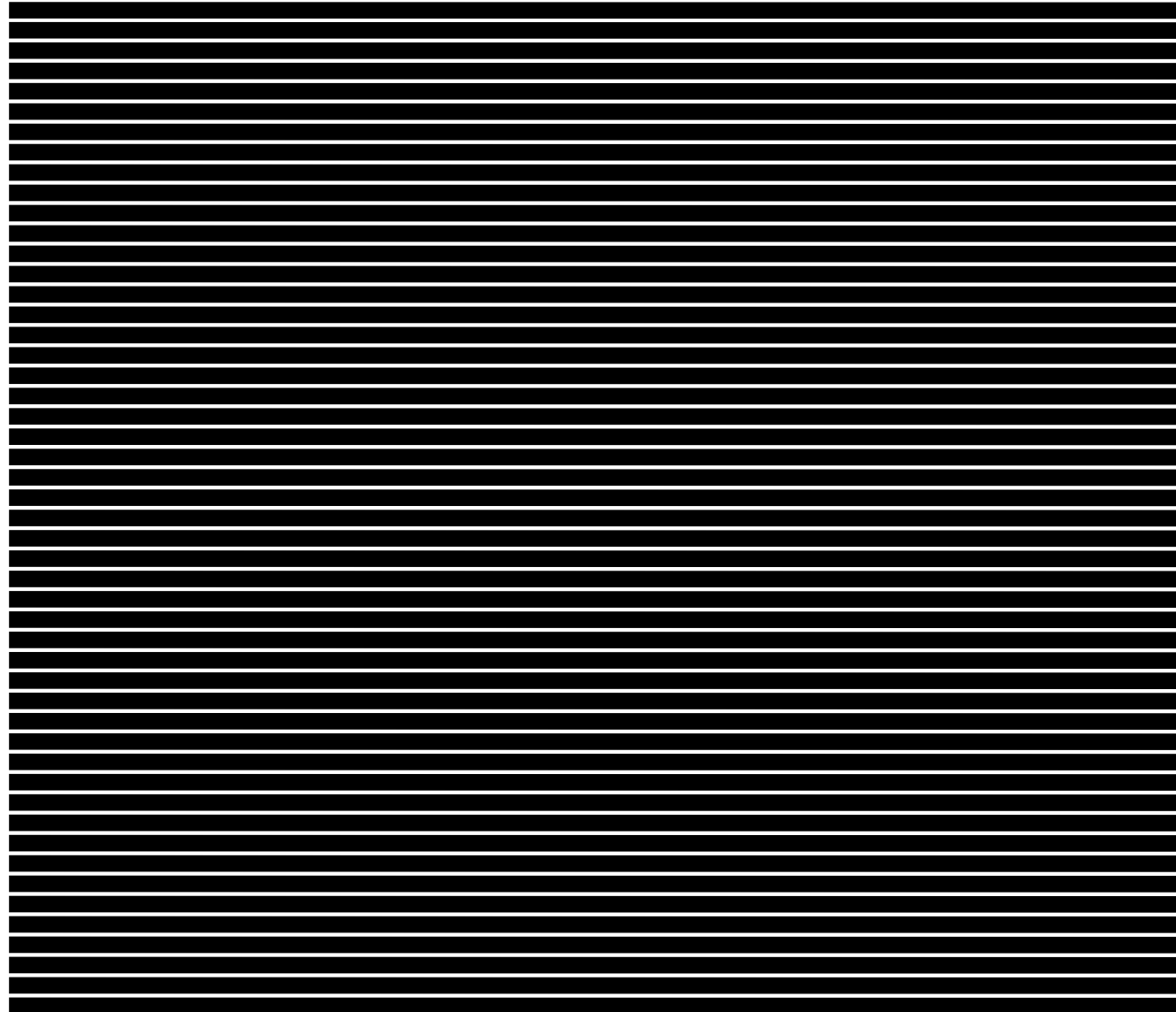


Lowpass filtered with 5x5 box filter

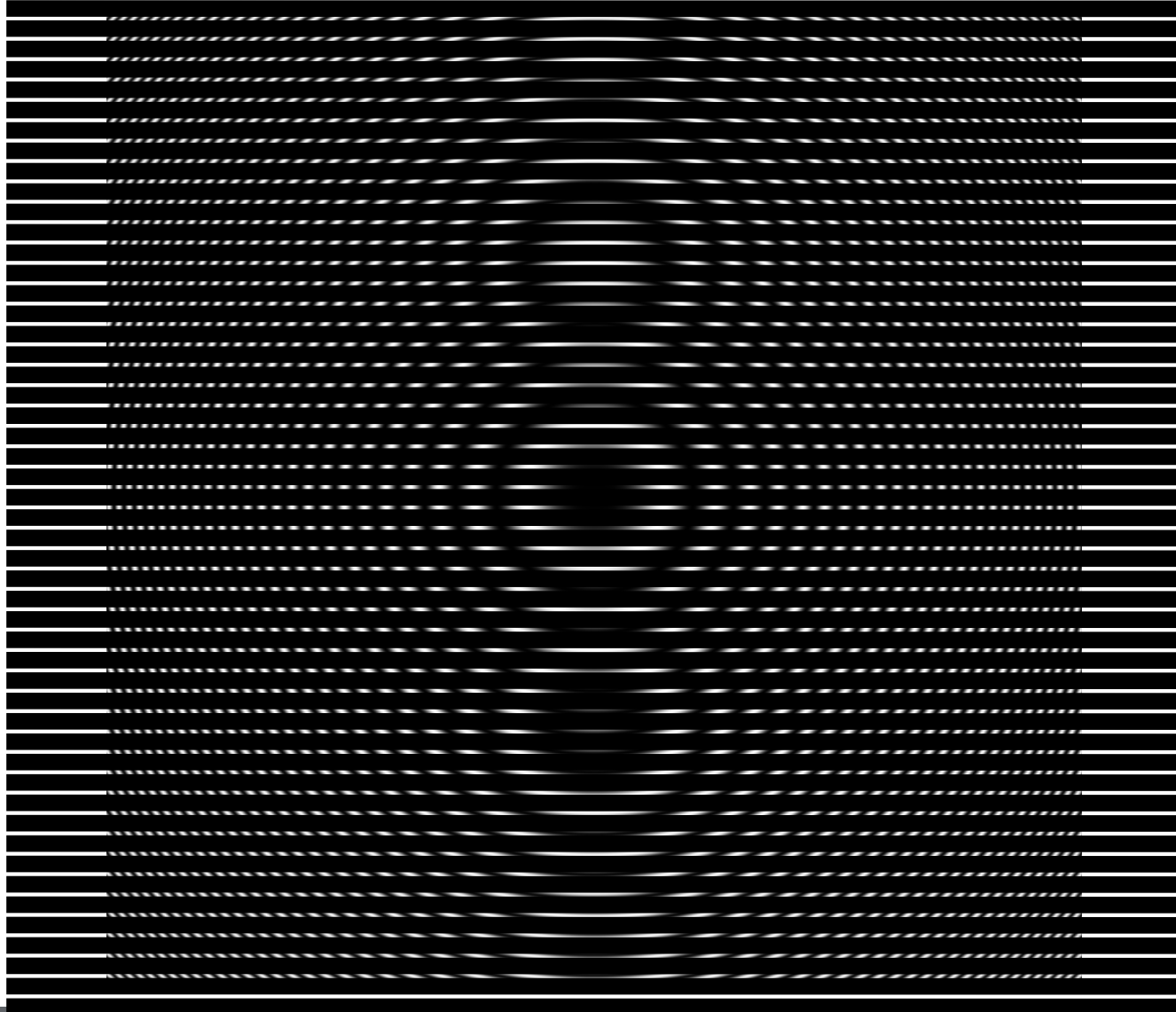
Zoneplate demonstration: 1-d sampling of 2-d signals



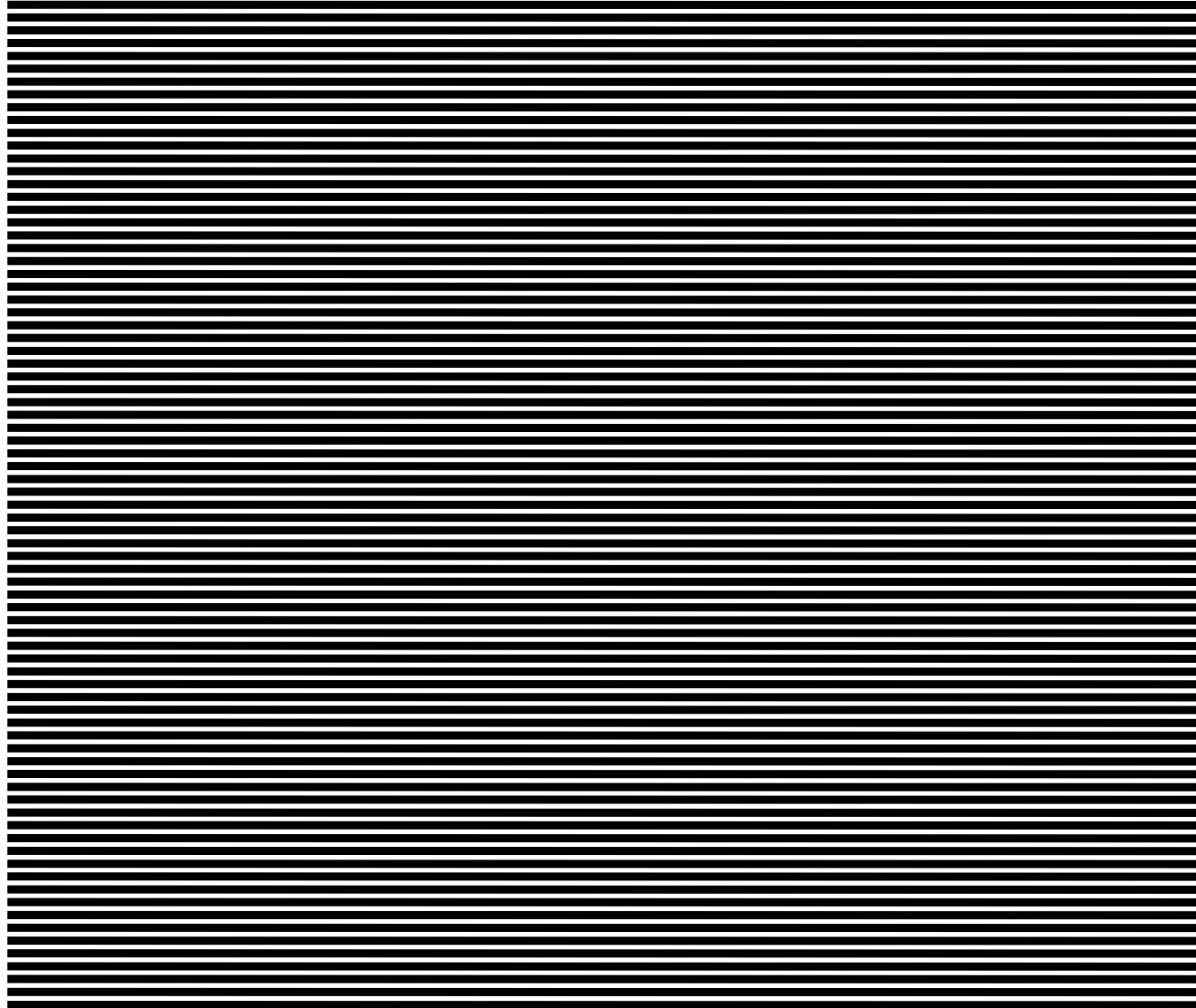
Zoneplate demonstration: 1-d sampling of 2-d signals



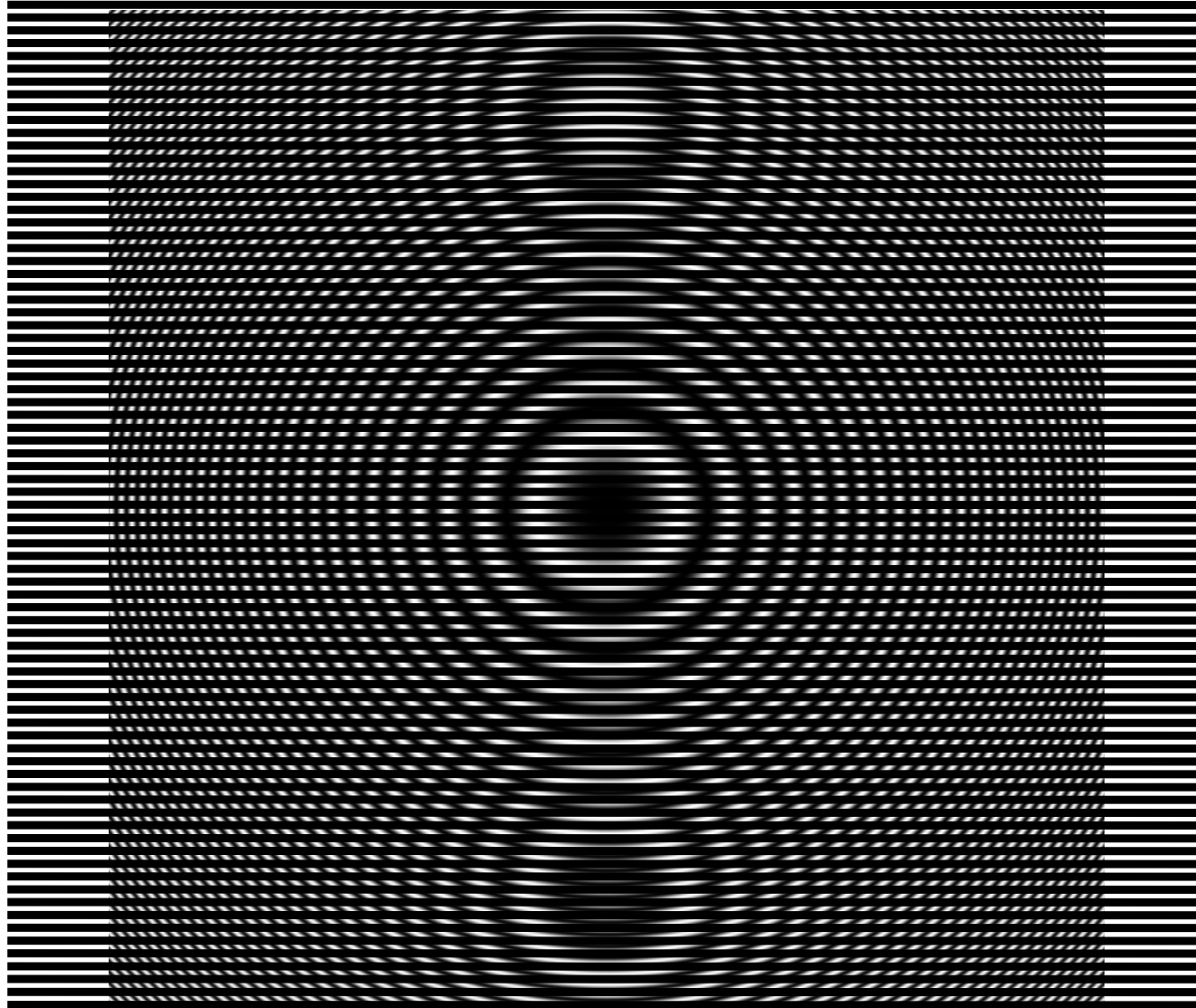
Zoneplate demonstration: 1-d sampling of 2-d signals



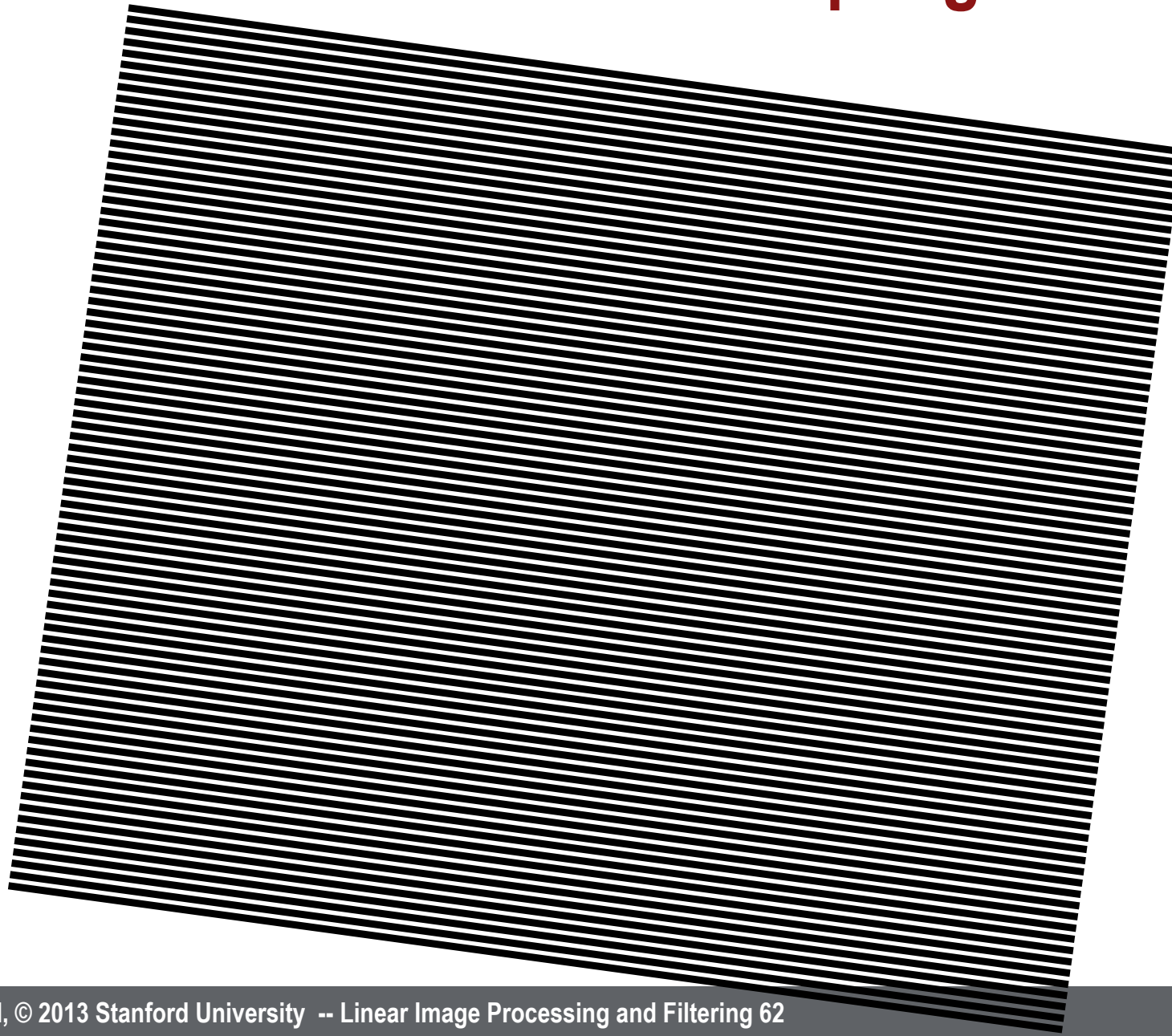
Zoneplate demonstration: 1-d sampling of 2-d signals



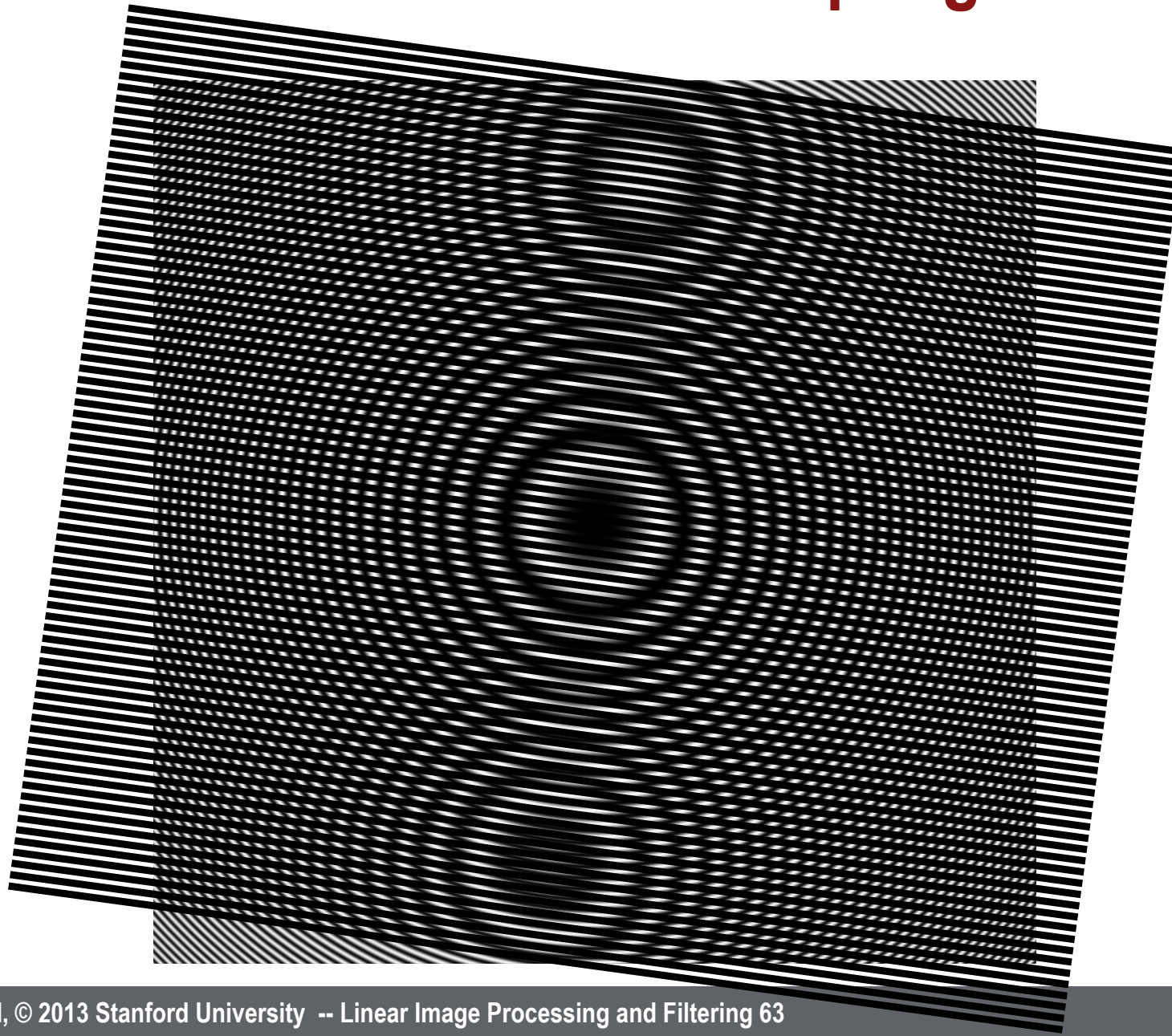
Zoneplate demonstration: 1-d sampling of 2-d signals



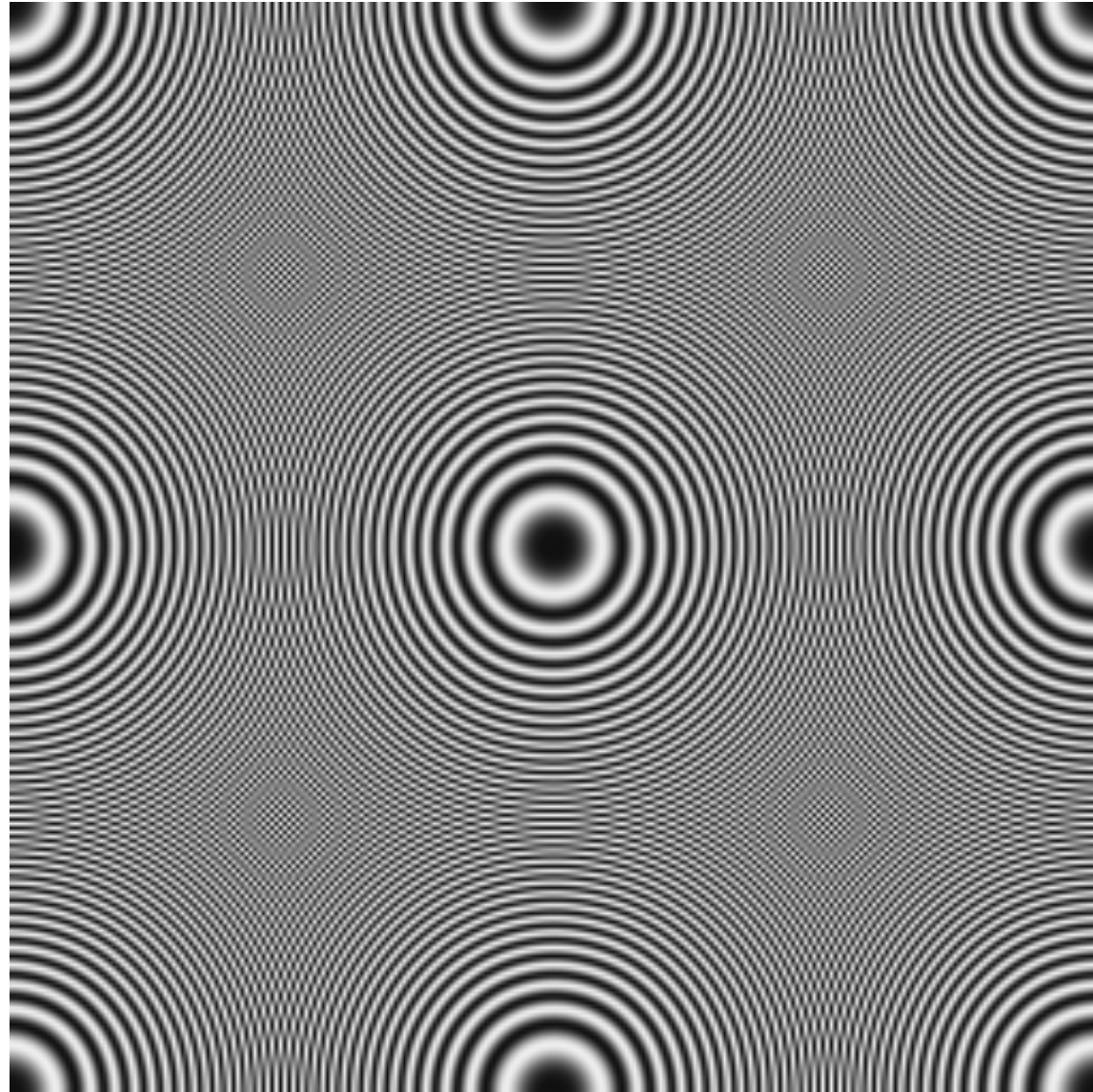
Zoneplate demonstration: 1-d sampling of 2-d signals



Zoneplate demonstration: 1-d sampling of 2-d signals



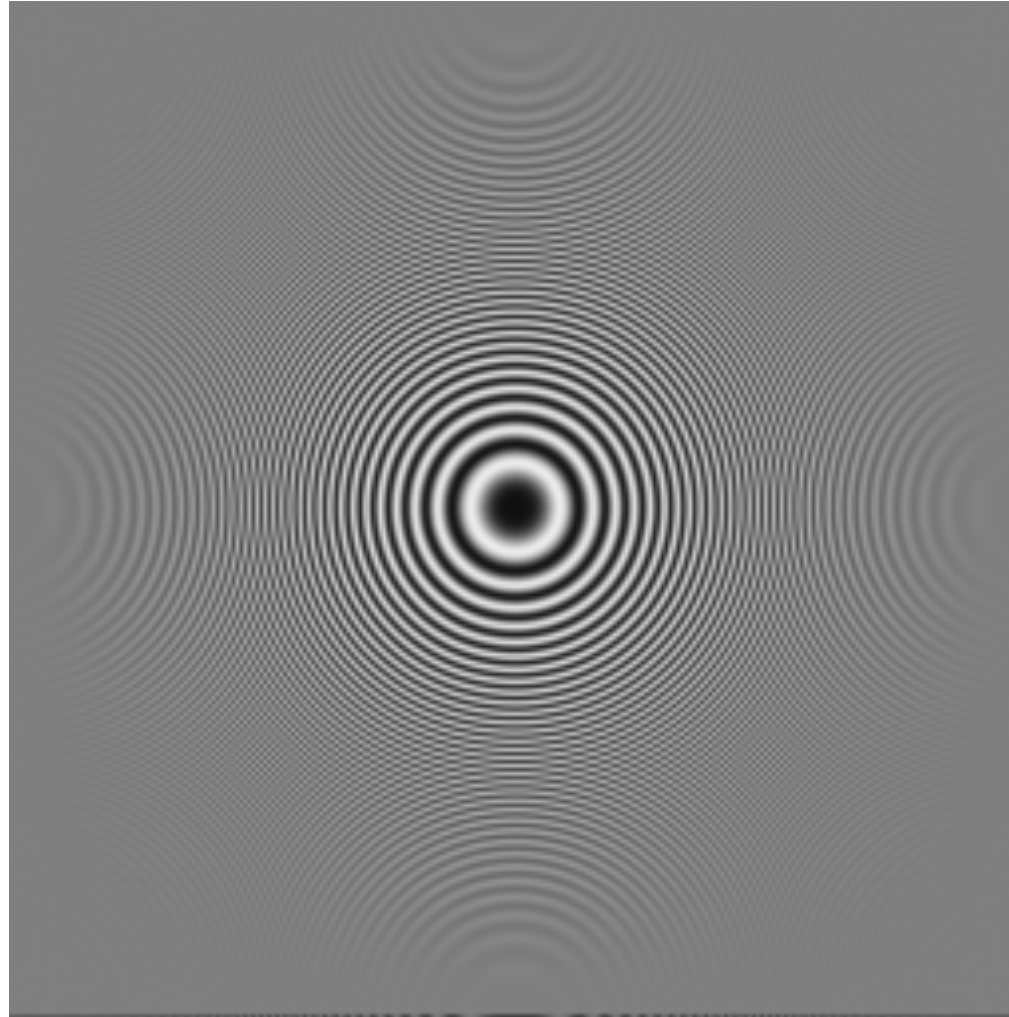
Horizontal/vertical 2:1 subsampling without prefiltering



Horizontal/vertical 2:1 subsampling with prefiltering

2d impulse response

$$\begin{pmatrix} \frac{1}{16} & \frac{1}{8} & \frac{1}{16} \\ \frac{1}{8} & \frac{1}{4} & \frac{1}{8} \\ \frac{1}{16} & \frac{1}{8} & \frac{1}{16} \end{pmatrix}$$



Frequency response

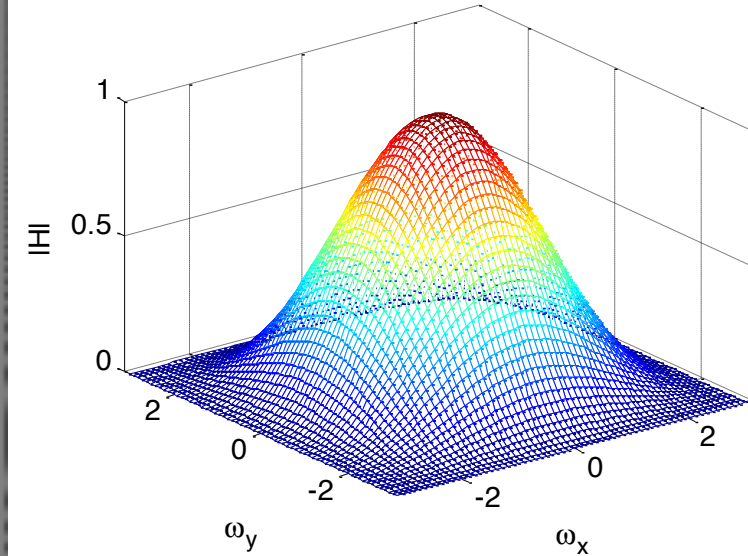
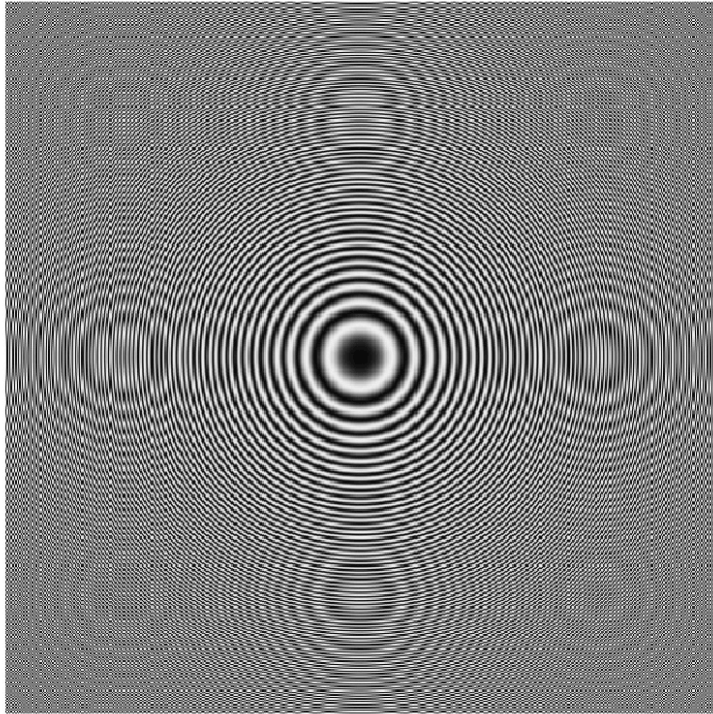
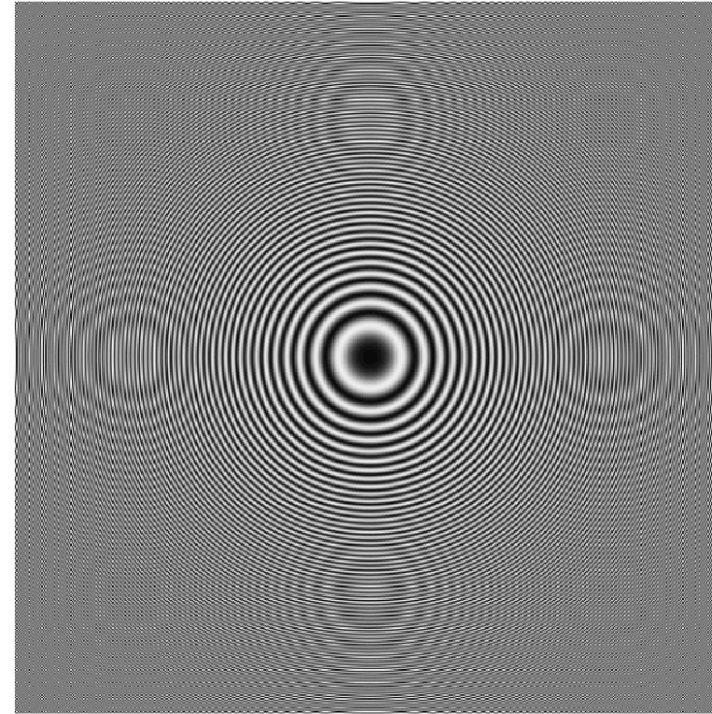


Image magnification (10%)



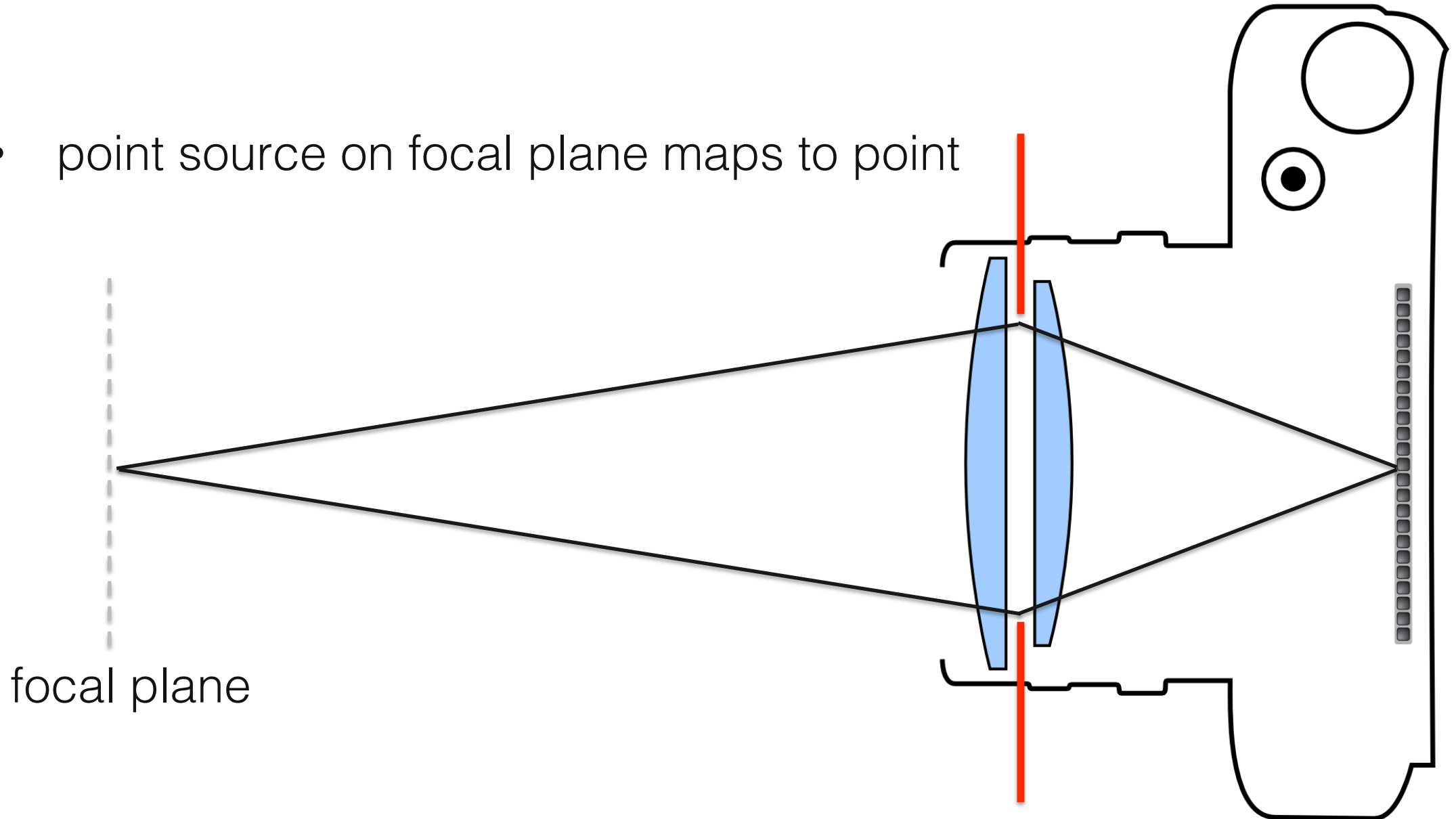
Nearest neighbor interpolation



Bilinear interpolation

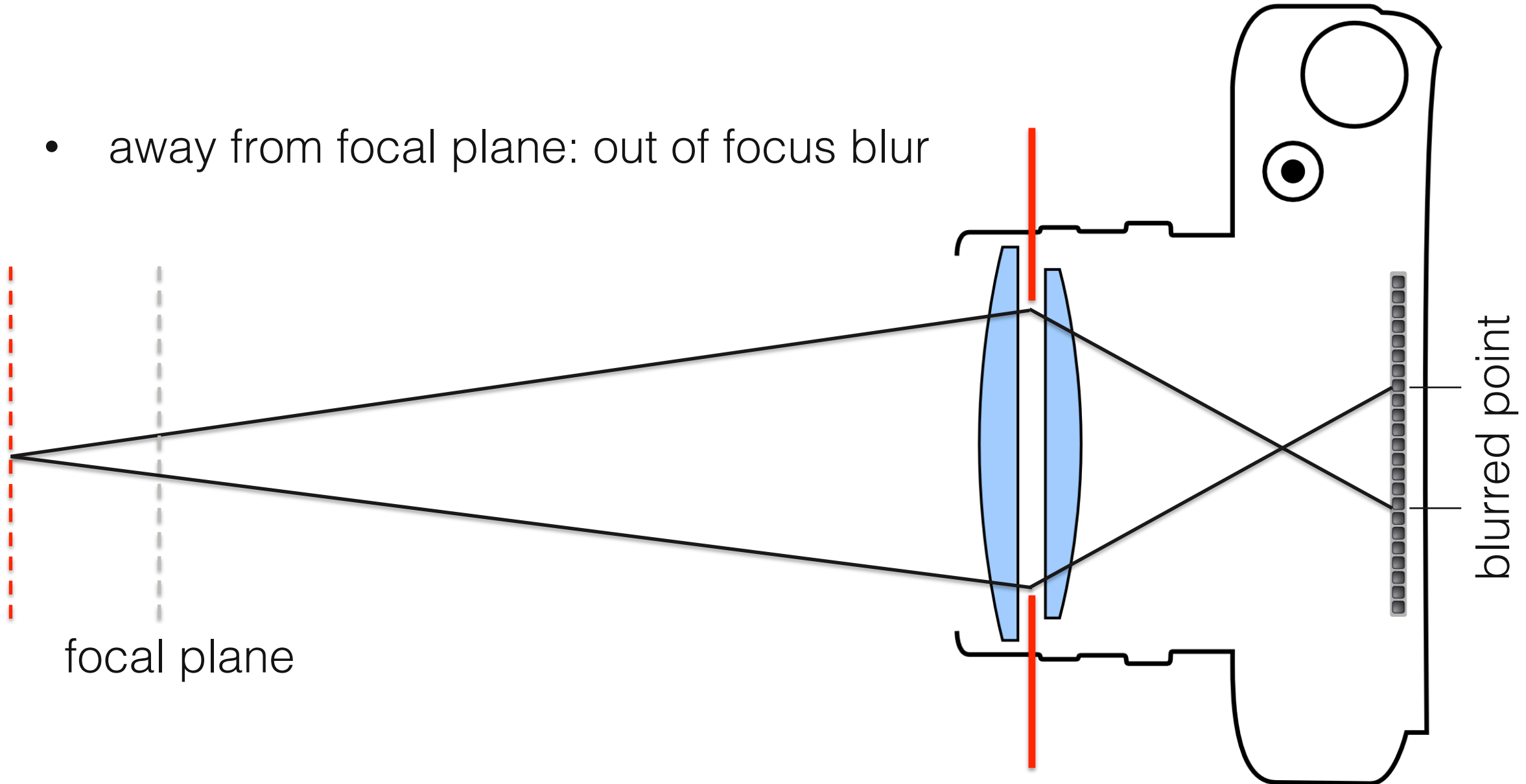
Lens as Optical Low-pass Filter

- point source on focal plane maps to point



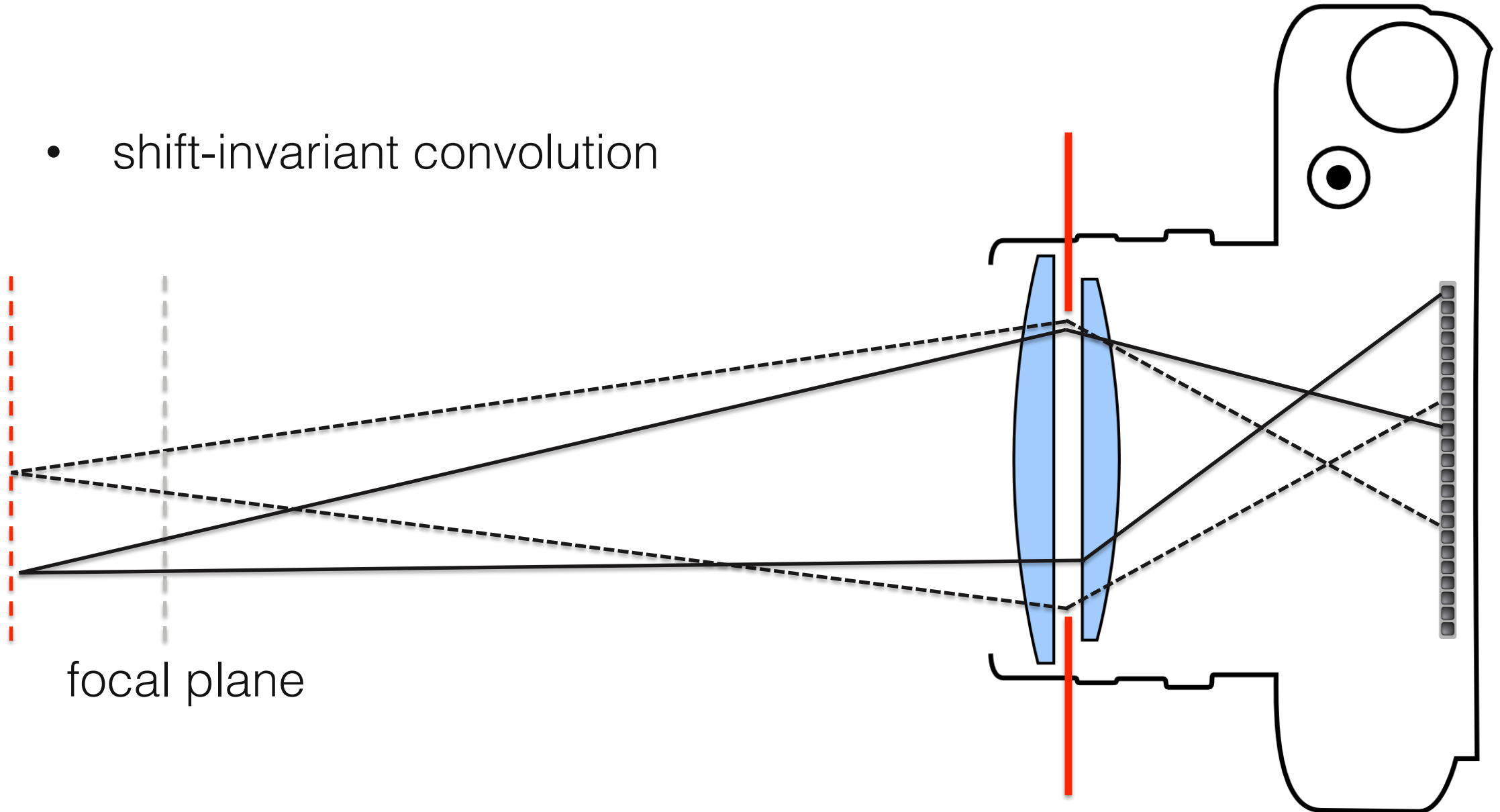
Lens as Optical Low-pass Filter

- away from focal plane: out of focus blur



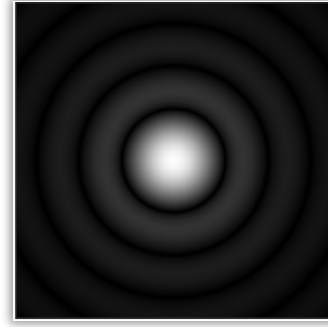
Lens as Optical Low-pass Filter

- shift-invariant convolution

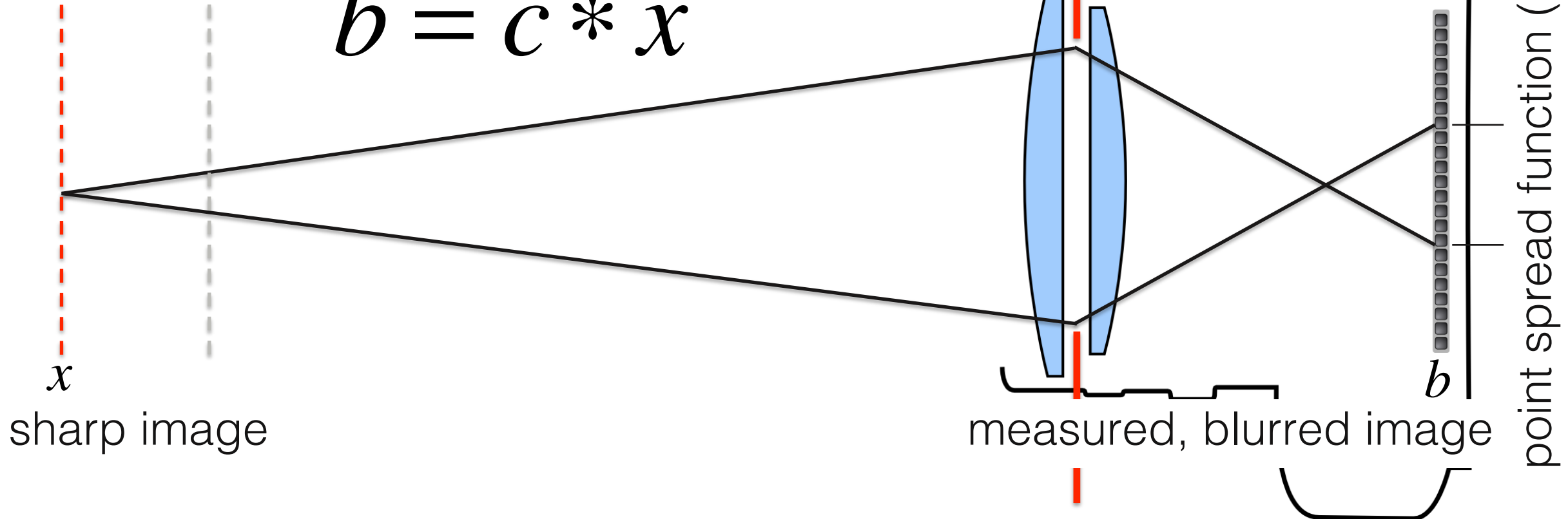


Lens as Optical Low-pass Filter

diffraction-limited PSF of circular aperture (aka “Airy” pattern):

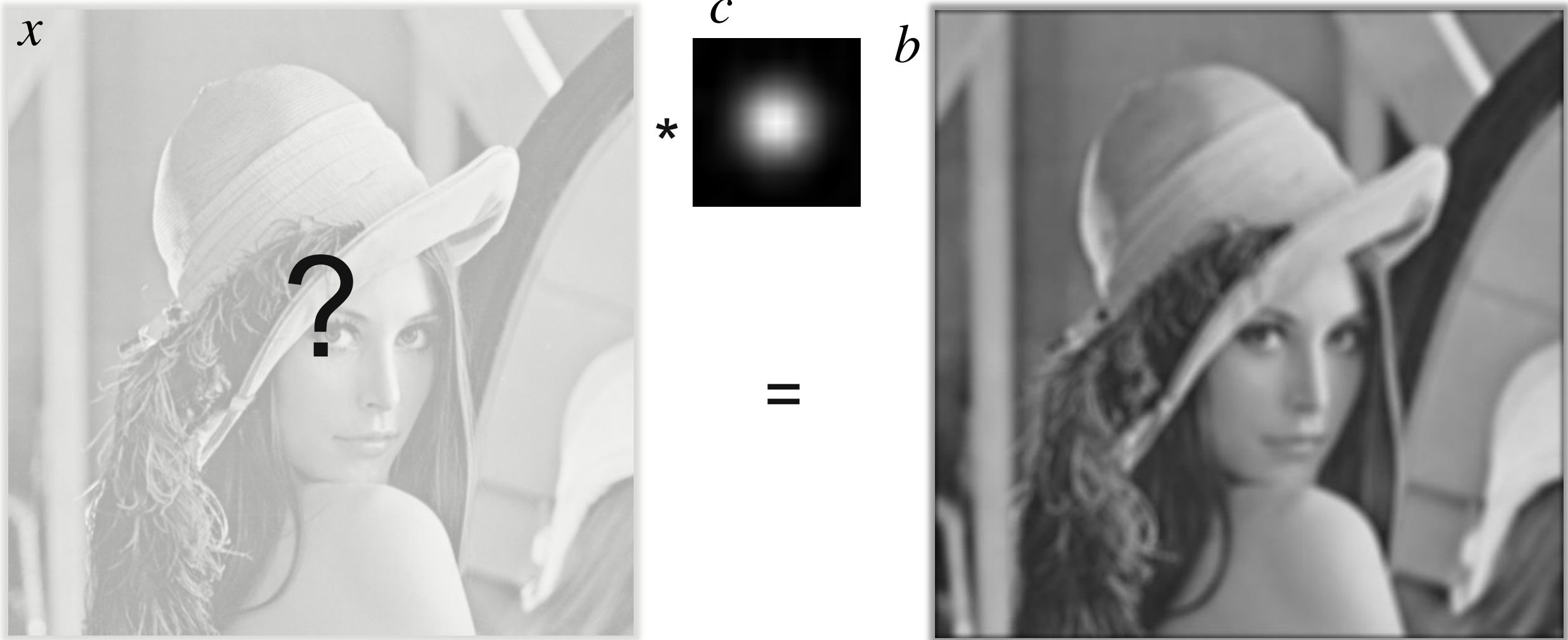


$$b = c * x$$



Deconvolution

- given measurements b and convolution kernel c , what is x ?



The diagram illustrates the deconvolution problem. It shows three main components arranged horizontally, separated by an asterisk and an equals sign. On the left is a grayscale image of a woman wearing a hat, labeled with the variable x in the top-left corner. A large question mark is superimposed over the center of this image. In the middle is a small, square, grayscale convolution kernel labeled with the variable c in the top-left corner. To the right of the kernel is a grayscale image of the same woman, labeled with the variable b in the top-right corner. Below the kernel and between the two images is an equals sign (=).

Deconvolution with Inverse Filtering

- naive solution: apply inverse kernel $\tilde{x} = c^{-1} * b = F^{-1} \left\{ \frac{F\{b\}}{F\{c\}} \right\}$

x



$\tilde{x}$



Deconvolution with Inverse Filtering & Noise

- naive solution: apply inverse kernel $\tilde{x} = c^{-1} * b = F^{-1} \left\{ \frac{F\{b\}}{F\{c\}} \right\}$

- Gaussian noise, $\sigma = 0.05$

$\tilde{x}$

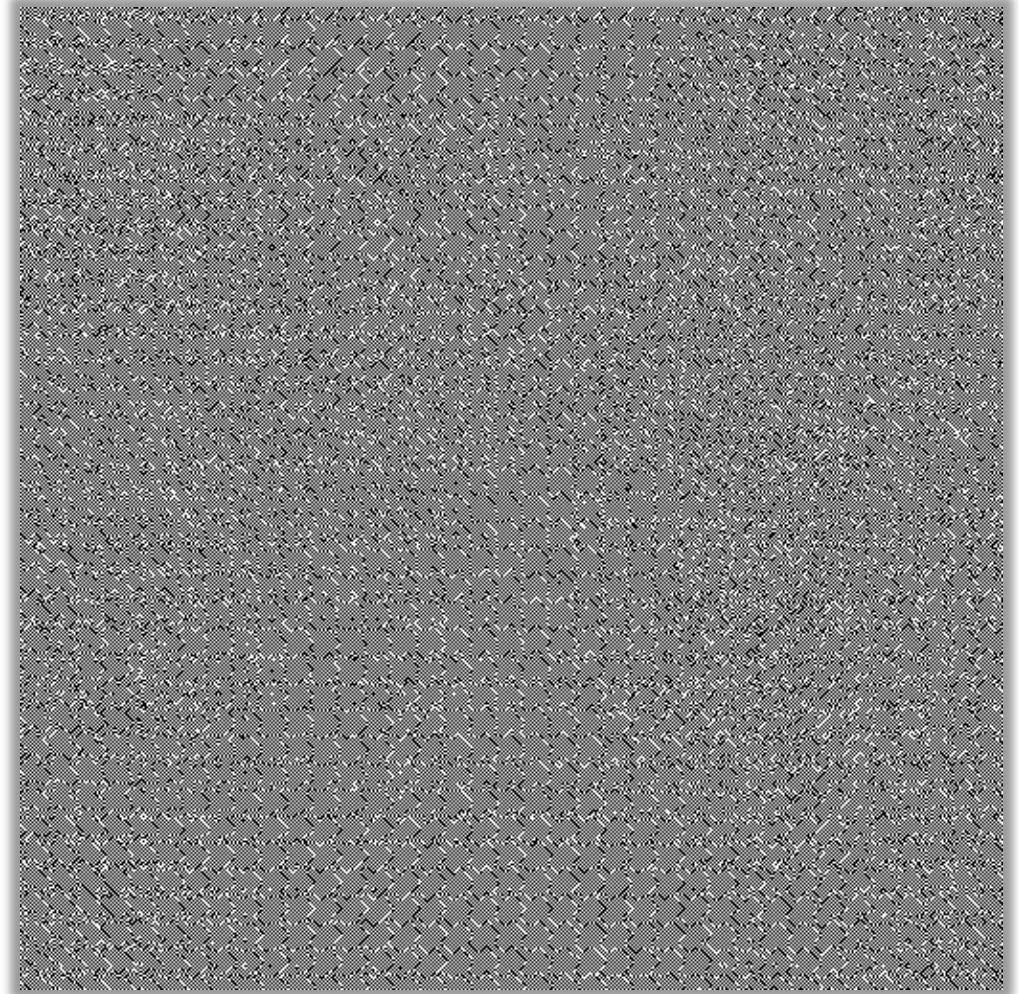
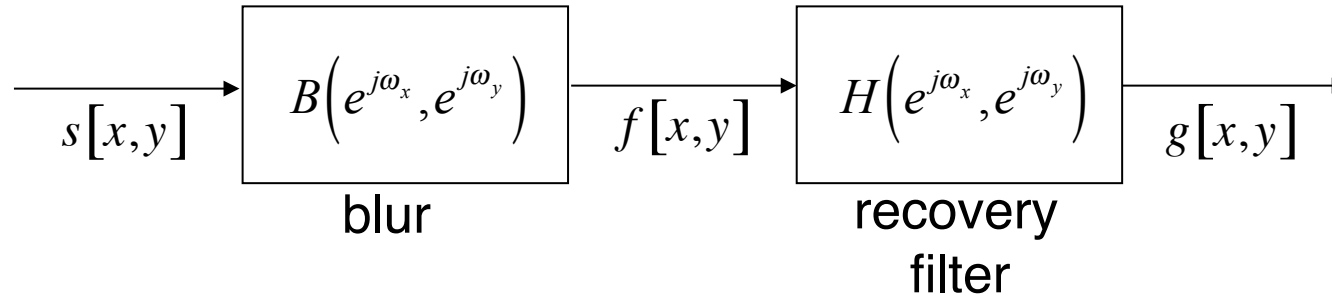


Image deconvolution

- Given an image $f[x,y]$ that is a blurred version of original image $s[x,y]$, recover the original image.
- Assume linear shift-invariant blur, transfer function $B(e^{j\omega_x}, e^{j\omega_y})$



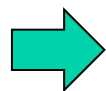
- Naive solution: inverse filter

$$H(e^{j\omega_x}, e^{j\omega_y}) = \frac{1}{B(e^{j\omega_x}, e^{j\omega_y})}$$

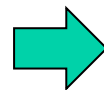
- Problem: $B(e^{j\omega_x}, e^{j\omega_y})$ might be zero, noise amplification

Naïve deconvolution

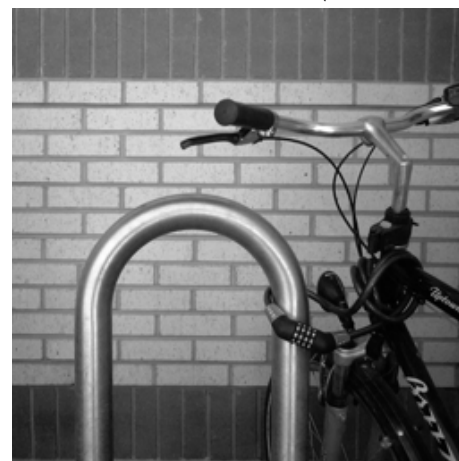
Original



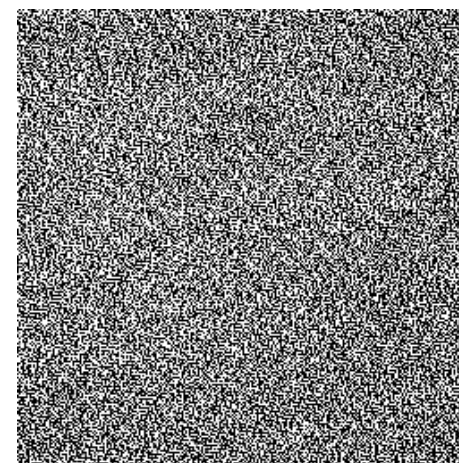
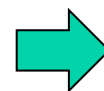
Blurred w/ $B(e^{j\omega_x}, e^{j\omega_y})$



Deblurred w/ $B^{-1}(e^{j\omega_x}, e^{j\omega_y})$



+ white noise  $rms = 2.5$



Deconvolution with Inverse Filtering & Noise

- results: terrible!
- why? this is an ill-posed problem (division by (close to) zero in frequency domain) → noise is drastically amplified!
- need to include prior(s) on images to make up for lost data
 - for example: noise statistics (signal to noise ratio)

Deconvolution with Wiener Filtering

- apply inverse kernel and don't divide by 0

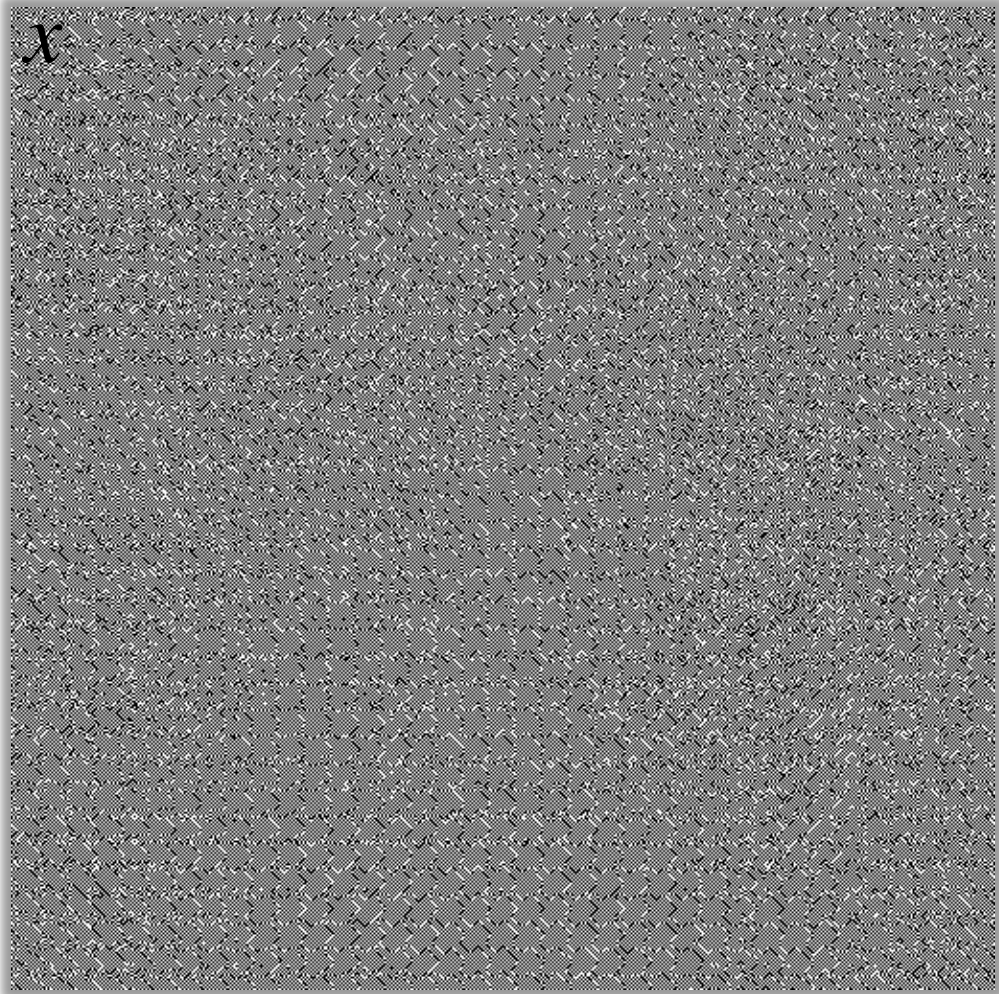
$$\tilde{x} = F^{-1} \left\{ \frac{|F\{c\}|^2}{|F\{c\}|^2 + 1/SNR} \cdot \frac{F\{b\}}{F\{c\}} \right\}$$

amplitude-dependent
damping factor!

$$SNR = \frac{\text{mean signal} \approx 0.5}{\text{noise std} = \sigma}$$

Deconvolution with Wiener Filtering

naive



Wiener



Deconvolution with Wiener Filtering



$\sigma = 0.01$



$\sigma = 0.05$



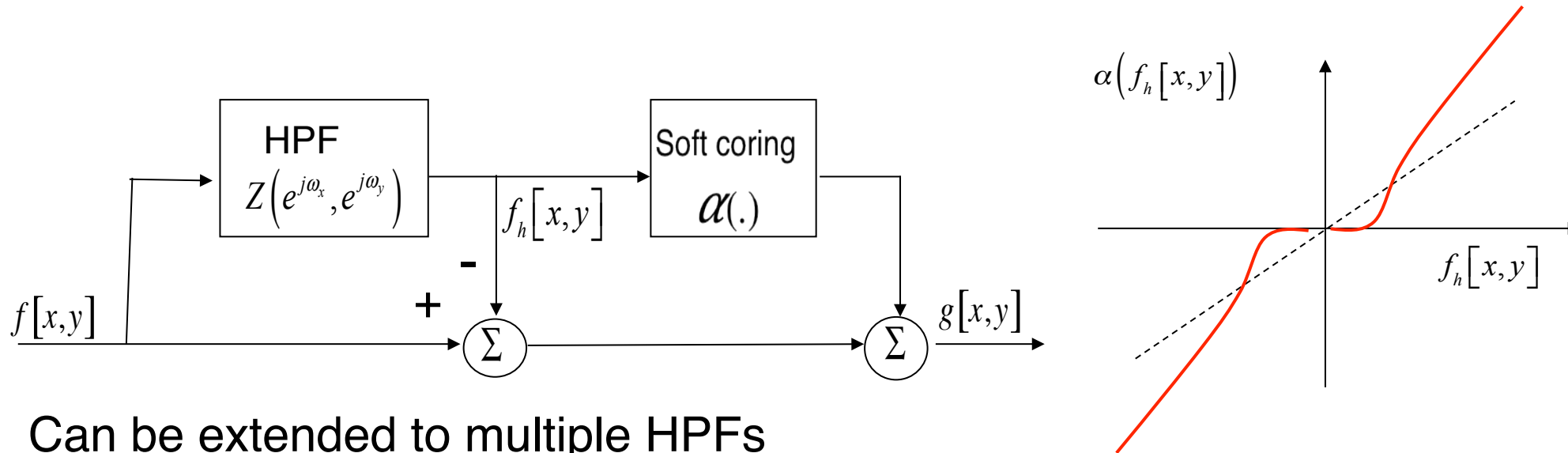
$\sigma = 0.1$

Deconvolution with Wiener Filtering

- results: not too bad, but noisy
- this is a heuristic → dampen noise amplification
- take EE 367 “Computational Imaging and Display” for more advanced deconvolution

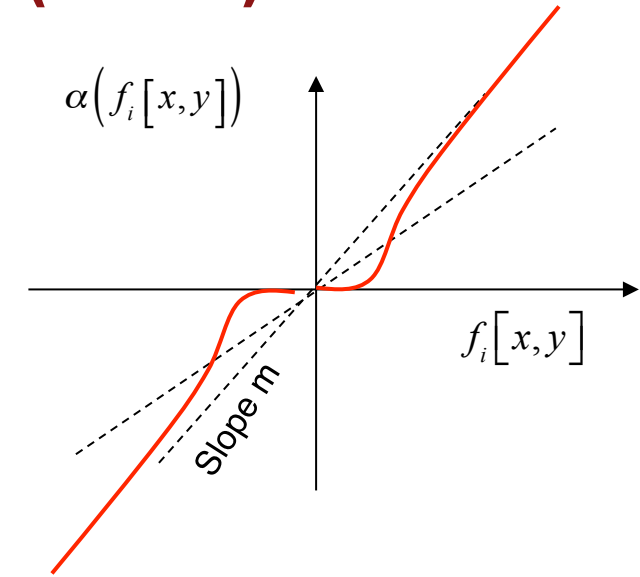
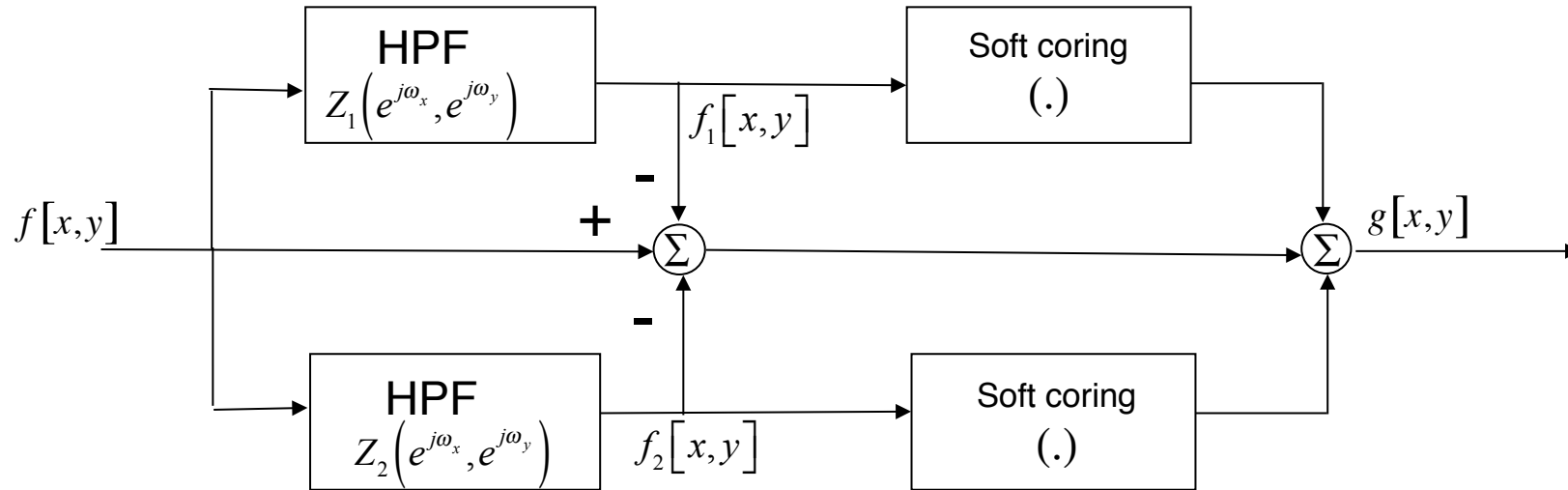
Nonlinear noise reduction/sharpening

- Noise reduction: smooth the image, lowpass filtering
- Deblurring: sharpen edges, highpass filtering
- How can both be achieved simultaneously?
- Key insight: large amplitude of highpass filtered image indicates presence of edge



- Can be extended to multiple HPFs

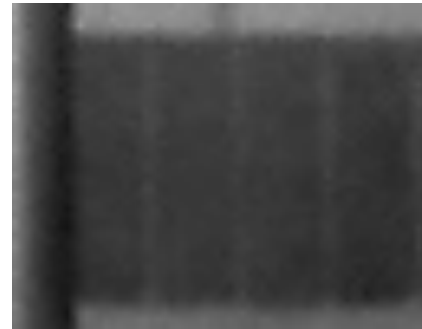
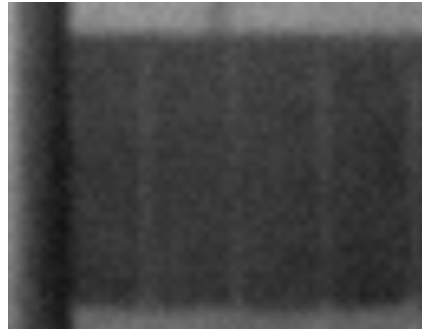
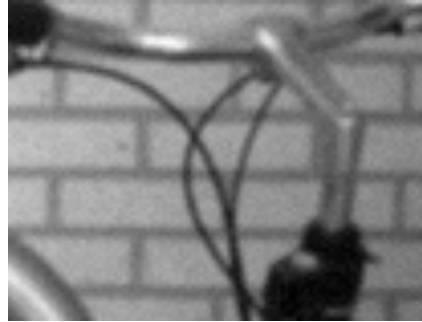
Nonlinear noise reduction/sharpening (cont.)



Nonlinear noise reduction/sharpening example



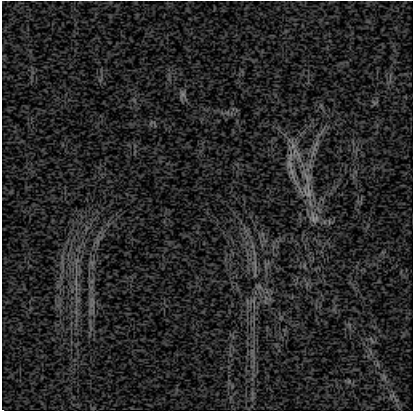
blurred, noisy image



noise-reduced
and sharpened

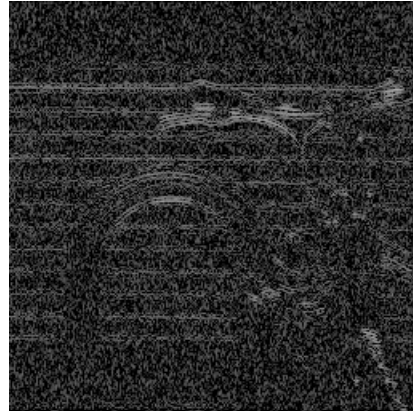


Highpass filtered images



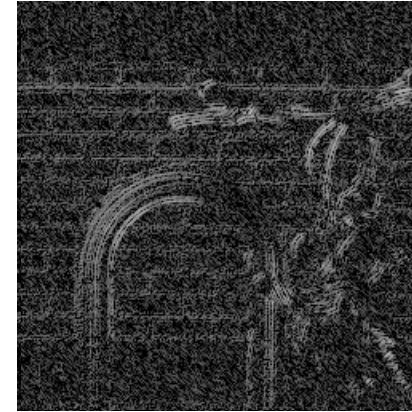
log magnitude of
image filtered with

$$\begin{pmatrix} 0 & 0 & 0 \\ -0.5 & [1] & -0.5 \\ 0 & 0 & 0 \end{pmatrix}$$



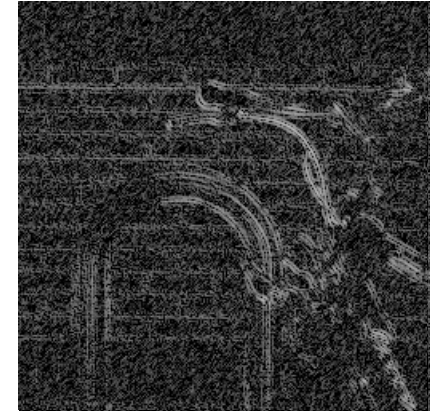
log magnitude of
image filtered with

$$\begin{pmatrix} 0 & -0.5 & 0 \\ 0 & [1] & 0 \\ 0 & -0.5 & 0 \end{pmatrix}$$



log magnitude of
image filtered with

$$\begin{pmatrix} -0.5 & 0 & 0 \\ 0 & [1] & 0 \\ 0 & 0 & -0.5 \end{pmatrix}$$



log magnitude of
image filtered with

$$\begin{pmatrix} 0 & 0 & -0.5 \\ 0 & [1] & 0 \\ -0.5 & 0 & 0 \end{pmatrix}$$



Soft coring function

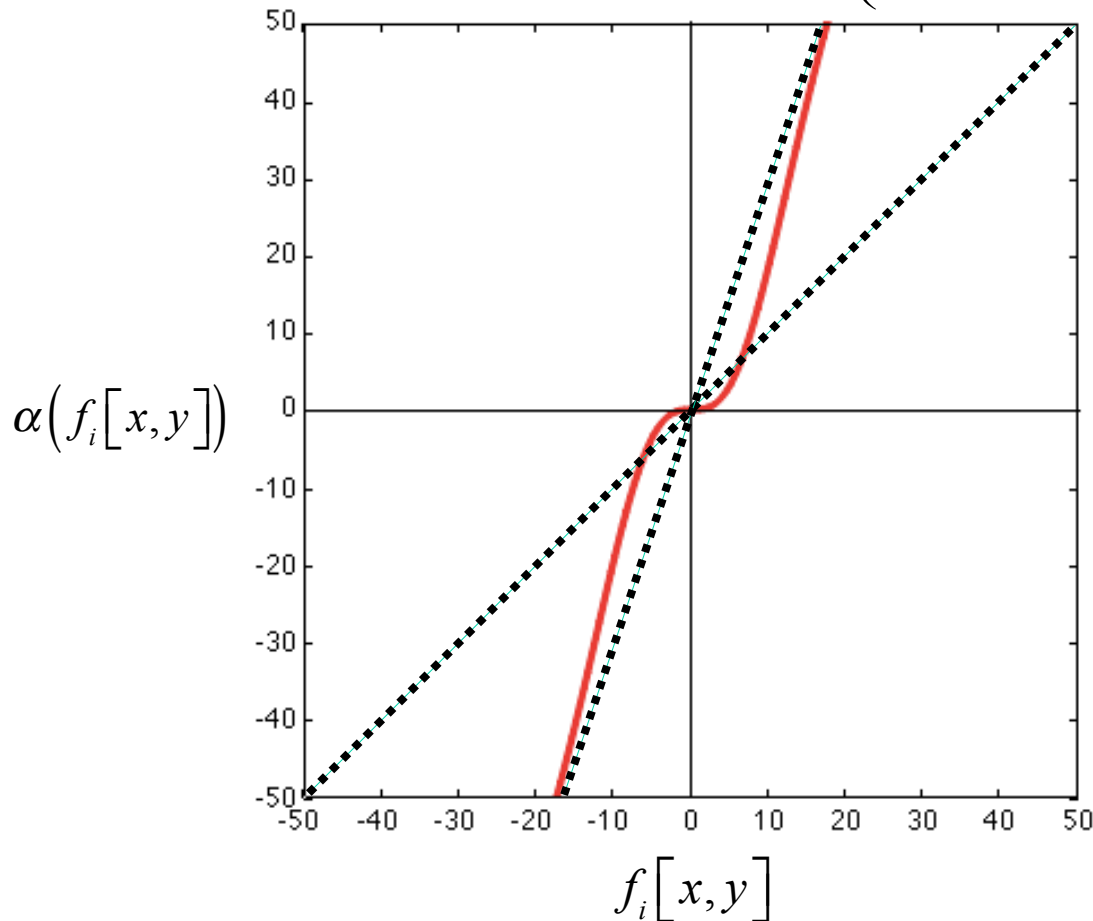
$$\alpha(f_i[x, y]) = m \cdot f_i[x, y] \cdot \left(1 - e^{-\left| \frac{f_i[x, y]}{\tau} \right|^\gamma} \right)$$

Example:

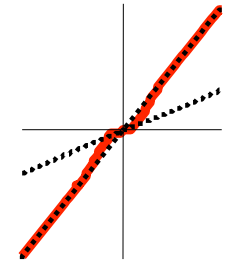
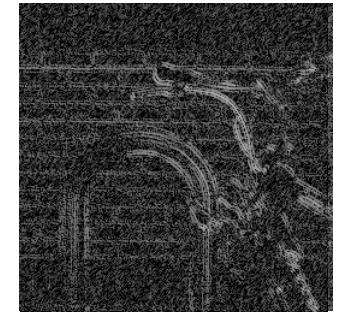
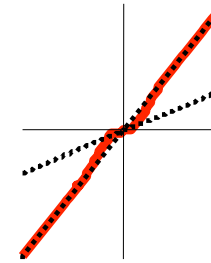
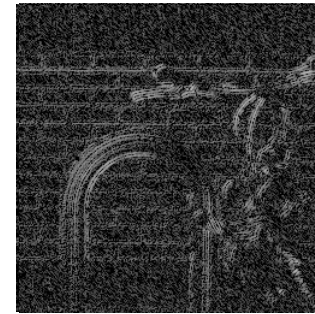
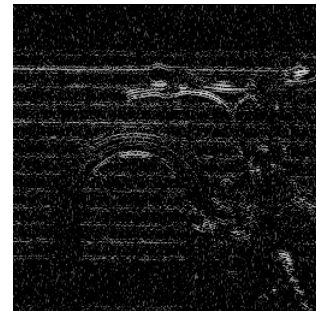
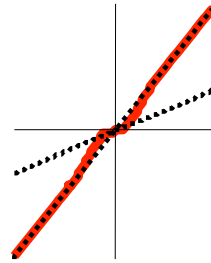
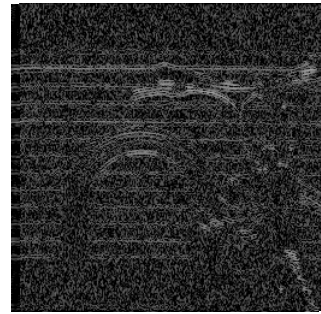
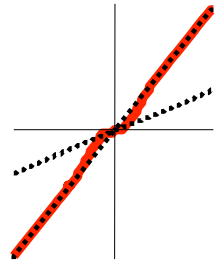
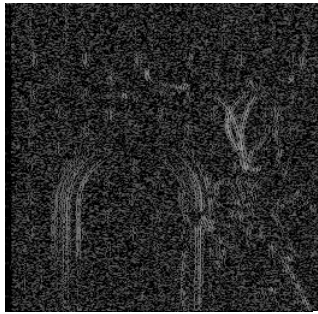
$m = 3$ slope of linear part

$\gamma = 2$ transition between suppression and linear

$\tau = 10$ width of suppressed area



Soft coring of highpass filtered images



Linear vs. nonlinear noise reduction/sharpening



Noise reduction by
lowpass filter (linear)



Sharpening by highpass
filter (linear)



Combined noise reduction and
sharpening (nonlinear)

