ECE 172A: Introduction to Image Processing
Discrete Images and Filtering: Part II

Rahul Parhi
Assistant Professor, ECE, UCSD

Winter 2025

Outline

- Characterization of Discrete Images ✓
  - Discrete Image Representation
  - Discrete-Space Fourier Transform
  - Two-Dimensional $\boldsymbol{z}$-transform $(= (z_1, z_2)$-transform)
- Discrete (Digital?) Filtering
  - Filtering With 2D Masks ✓
  - Equivalent Filter Characterizations ✓
  - Separability
- Filtering Images: Practical Considerations

Filtering Examples: Revisited

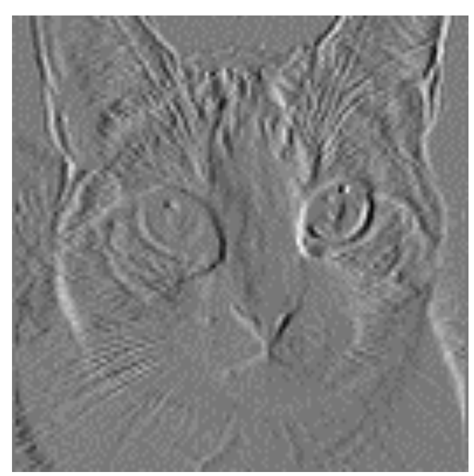
Mask: $\boldsymbol{w}$

– Local (3×3)-average

$$\boldsymbol{w}_{\text{ave}} = \frac{1}{9}\begin{bmatrix} 1 & 1 & 1 \\ 1 & \boxed{1} & 1 \\ 1 & 1 & 1 \end{bmatrix}$$

– Horizontal-edge enhancement

$$\boldsymbol{w}_{\text{hor}} = \begin{bmatrix} -1 & -2 & -1 \\ 0 & \boxed{0} & 0 \\ 1 & 2 & 1 \end{bmatrix}$$



– Vertical-edge enhancement

$$\boldsymbol{w}_{\text{vert}} = \begin{bmatrix} -1 & 0 & 1 \\ -2 & \boxed{0} & 2 \\ -1 & 0 & 1 \end{bmatrix}$$

Local Average

- Mask:

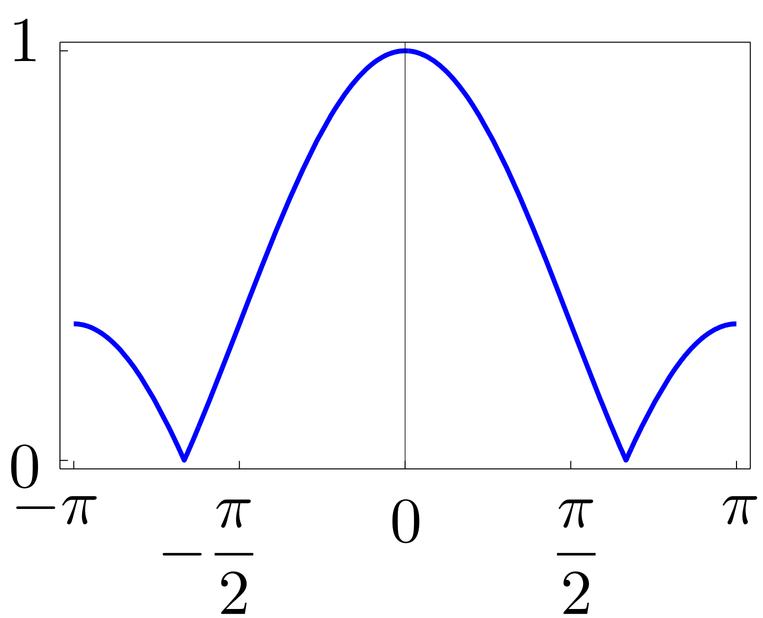
$$\boldsymbol{w}_{\text{ave}} = \frac{1}{9}\begin{bmatrix} 1 & 1 & 1 \\ 1 & \boxed{1} & 1 \\ 1 & 1 & 1 \end{bmatrix} \quad \Longrightarrow \quad \boldsymbol{h}_{\text{ave}} = \frac{1}{9}\begin{bmatrix} 1 & 1 & 1 \\ 1 & \boxed{1} & 1 \\ 1 & 1 & 1 \end{bmatrix}$$

- Transfer function: $H(z_1, z_2) = \frac{1}{3}(z_1 + 1 + z_1^{-1}) \cdot \frac{1}{3}(z_2 + 1 + z_2^{-1})$

- Frequency response: $H(\mathrm{e}^{\mathrm{j}\omega_1}, \mathrm{e}^{\mathrm{j}\omega_2}) = \left(\frac{1 + 2\cos\omega_1}{3}\right)\left(\frac{1 + 2\cos\omega_2}{3}\right)$

$$= H_{\text{ave}}(\mathrm{e}^{\mathrm{j}\omega_1}) H_{\text{ave}}(\mathrm{e}^{\mathrm{j}\omega_2})$$

$|H_{\text{ave}}(\mathrm{e}^{\mathrm{j}\omega})|$



low-pass behavior

Vertical-Edge Enhancement

- Mask:

$$\boldsymbol{w}_{\text{vert}} = \begin{bmatrix} -1 & 0 & 1 \\ -2 & \boxed{0} & 2 \\ -1 & 0 & 1 \end{bmatrix} \quad \Longrightarrow \quad \boldsymbol{h}_{\text{vert}} = \begin{bmatrix} 1 & 0 & -1 \\ 2 & \boxed{0} & -2 \\ 1 & 0 & -1 \end{bmatrix}$$

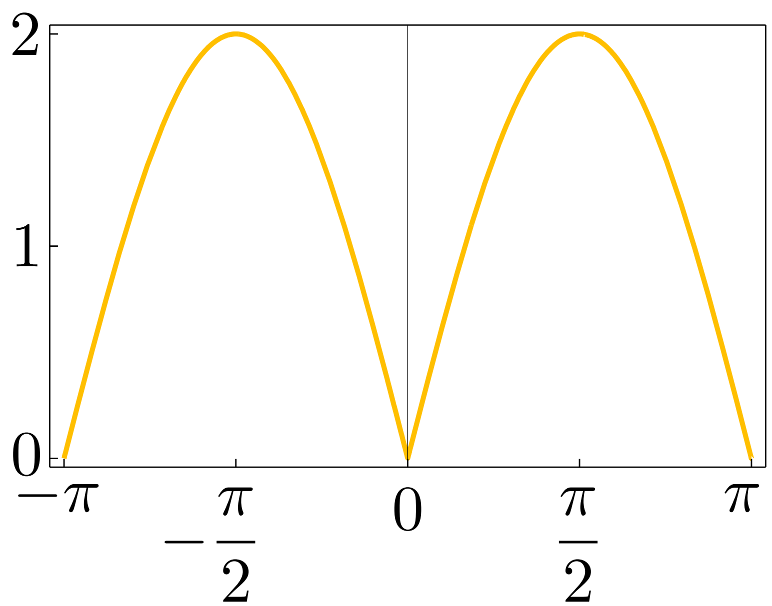
"correlation" "convolution"

- Transfer function: $H(z_1, z_2) = (z_1 - z_1^{-1})(z_2 + 2 + z_2^{-1})$

- Frequency response: $H(\mathrm{e}^{\mathrm{j}\omega_1}, \mathrm{e}^{\mathrm{j}\omega_2}) = (\mathrm{j}2\sin\omega_1)(2 + 2\cos\omega_2)$

$$= H_1(\mathrm{e}^{\mathrm{j}\omega_1}) H_{\text{low}}(\mathrm{e}^{\mathrm{j}\omega_2})$$

$|H_1(\mathrm{e}^{\mathrm{j}\omega})|$



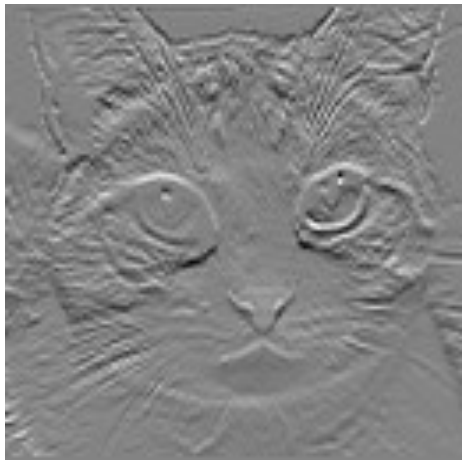
band-pass behavior

Horizontal-edge enhancement is just the "transpose"

Filtering Examples: Separability

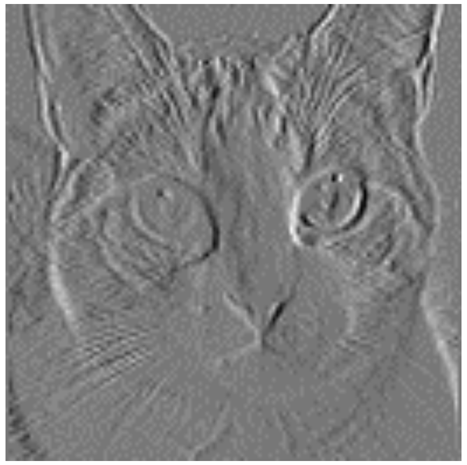
– Local (3×3)-average

$$\boldsymbol{h}_{\text{ave}} = \frac{1}{9}\begin{bmatrix} 1 & 1 & 1 \\ 1 & \boxed{1} & 1 \\ 1 & 1 & 1 \end{bmatrix} = \frac{1}{3}\begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix} \cdot \frac{1}{3}\begin{bmatrix} 1 & 1 & 1 \end{bmatrix}$$



– Horizontal-edge enhancement

$$\boldsymbol{h}_{\text{hor}} = \begin{bmatrix} 1 & 2 & 1 \\ 0 & \boxed{0} & 0 \\ -1 & -2 & -1 \end{bmatrix} = \begin{bmatrix} 1 \\ 0 \\ -1 \end{bmatrix} \cdot \begin{bmatrix} 1 & 2 & 1 \end{bmatrix}$$



– Vertical-edge enhancement

$$\boldsymbol{h}_{\text{vert}} = \begin{bmatrix} 1 & 0 & -1 \\ 2 & \boxed{0} & -2 \\ 1 & 0 & -1 \end{bmatrix} = \begin{bmatrix} 1 \\ 2 \\ 1 \end{bmatrix} \cdot \begin{bmatrix} 1 & 0 & -1 \end{bmatrix}$$

Separability

Most useful image-processing filters are separable...

which brings us back to a 1D problem

- Definition(s) of separability

$$h[k_1, k_2] = h_1[k_1]h_2[k_2]$$

$$\Updownarrow$$

$$H(z_1, z_2) = H_1(z_1)H_2(z_2)$$

$$\Updownarrow$$

$$H(\mathrm{e}^{\mathrm{j}\omega_1}, \mathrm{e}^{\mathrm{j}\omega_2}) = H_1(\mathrm{e}^{\mathrm{j}\omega_1})H_2(\mathrm{e}^{\mathrm{j}\omega_2})$$

$$\Updownarrow$$

$$\boldsymbol{h} = \boldsymbol{h}_2\,\boldsymbol{h}_1^{\mathsf{T}}$$

Separability

- Multiplication-table perspective

$$\boldsymbol{b}\,\boldsymbol{a}^\mathsf{T} = \begin{bmatrix} a_1 b_1 & a_2 b_1 & \cdots & a_M b_1 \\ a_1 b_2 & a_2 b_2 & \cdots & a_M b_2 \\ \vdots & \vdots & \ddots & \vdots \\ a_1 b_N & a_2 b_N & \cdots & a_M b_N \end{bmatrix}$$

	1	0	-1
1	1	0	-1
2	2	0	-2
1	1	0	-1

Example: $\boldsymbol{a} = (1, 0, -1)$ and $\boldsymbol{b} = (1, 2, 1)$
Vertical-edge enhancer

A filter is separable if and only if it can be factored as the **outer product** of two vectors

A filter is separable if and only if it is rank 1

Example: Smoother

$$\boldsymbol{h} = \frac{1}{16}\begin{bmatrix} 1 & 2 & 1 \\ 2 & \boxed{4} & 2 \\ 1 & 2 & 1 \end{bmatrix}$$

Exercise: (i) show that this filter is separable
(ii) Determine the 1D filters that comprise this 2D filter
(iii) Determine the transfer function of this filter

$$\boldsymbol{h} = \frac{1}{16}\begin{bmatrix} 1 & 2 & 1 \\ 2 & \boxed{4} & 2 \\ 1 & 2 & 1 \end{bmatrix} = \frac{1}{4}\begin{bmatrix} 1 \\ \boxed{2} \\ 1 \end{bmatrix} \cdot \frac{1}{4}\begin{bmatrix} 1 & \boxed{2} & 1 \end{bmatrix}$$

$$h[k_1, k_2] = h_1[k_1]h_1[k_2] \quad \text{where} \quad h_1[k] = \begin{cases} 1/4, & k = \pm 1 \\ 1/2, & k = 0 \\ 0, & \text{else} \end{cases}$$

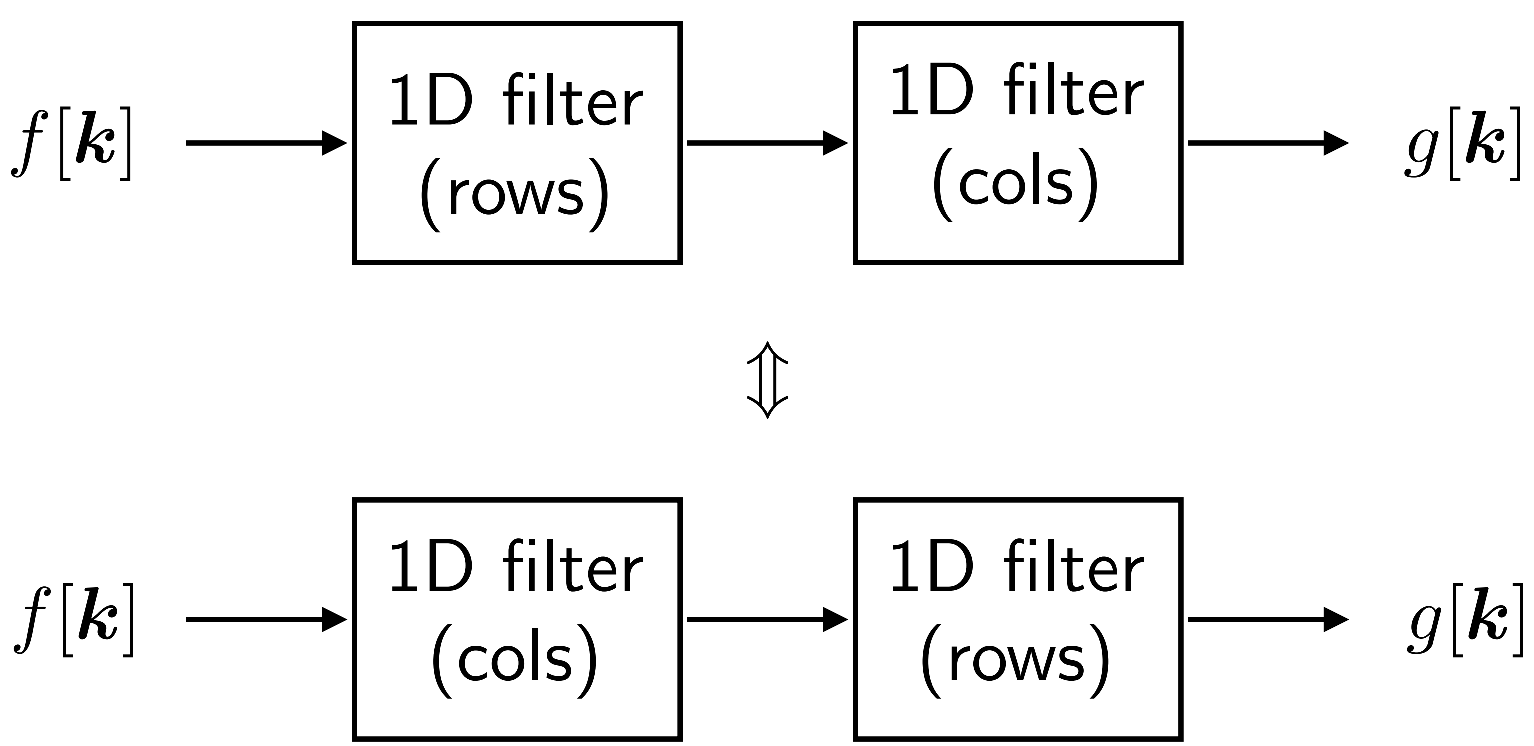
$$H(z_1, z_2) = \frac{1}{16}(z_1 + 2 + z_1^{-1})(z_2 + 2 + z_2^{-1})$$

Separable Filtering

Are there any limitations to considering separable filters?

- Orientation-sensitive filters are in general non-separable
  - Often used in **texture analysis**

- Separable filters have an efficient implementation



Recursive Filtering

Recursive filtering provides an efficient way to implement IIR filters

- Rational transfer function and difference equations

$$H(z) = \frac{G(z)}{F(z)} = \frac{\sum_{m=0}^{M-1} b_m z^{-m}}{\sum_{n=0}^{N-1} a_n z^{-n}} \quad \Longleftrightarrow \quad \sum_{n=0}^{N-1} a_n g[k-n] = \sum_{m=0}^{M-1} b_m f[k-m]$$

recursive-filter implementation

  - Example: Causal exponential

$$G(z) = \left(\frac{1}{1 + z^{-1} a_1}\right) F(z) \quad \Longleftrightarrow \quad g[k] = f[k] - a_1 g[k-1]$$

- Stability of rational filters
  - poles **inside** the unit circle

When is this system stable?

Recursive Filtering

Recursive filtering provides an efficient way to implement IIR filters

- Rational transfer function and difference equations
  - Example: Causal exponential

pole at $z = a_1$

$$G(z) = \left(\frac{1}{1 - z^{-1}a_1}\right) F(z) \quad \Longleftrightarrow \quad g[k] = f[k] + a_1 g[k-1]$$

  - Example: Anti-causal exponential

pole at $z = a_1^{-1}$

$$G(z) = \left(\frac{1}{1 - z a_1}\right) F(z) \quad \Longleftrightarrow \quad g[k] = f[k] + a_1 g[k+1]$$

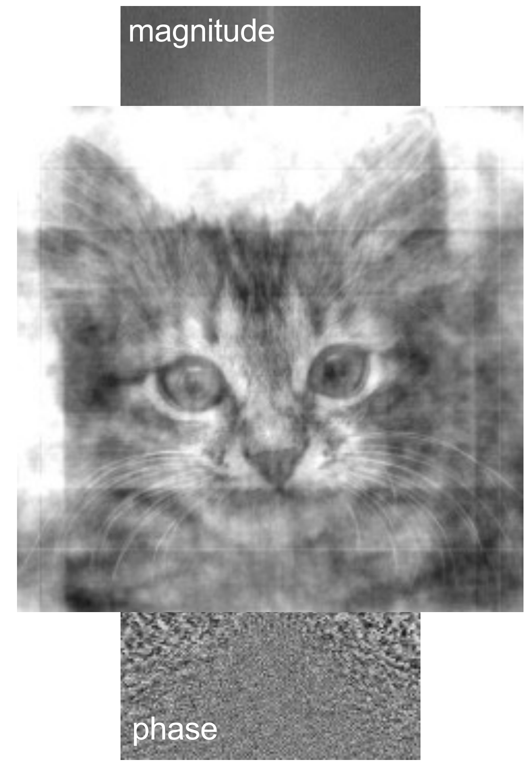
Can both of these filters be simultaneously stable?

What makes $H(z) = \left(\dfrac{1}{1 - z a_1}\right)$ not causal?

Filtering Images: Practical Considerations

- Filter Design for Image Processing
- Boundary Conditions
- Fourier-domain versus spatial-domain implementations

Filter Design for Image Processing

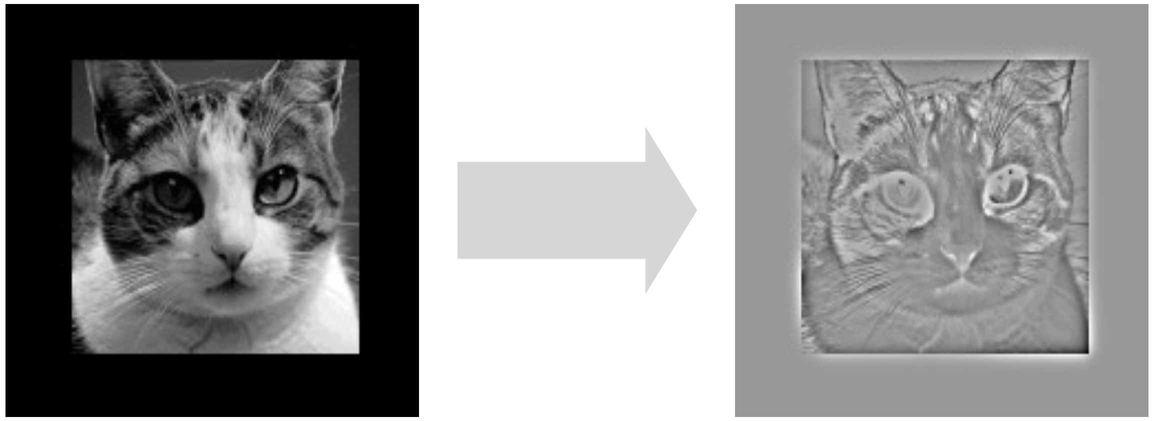


- Semantic information (edges, contours, etc.) is stored in the phase of the Fourier transform
  - Use linear-phase filters (i.e., symmetric or antisymmetric)
- Exact shape of the frequency response is not so important
  - Go for the simplest and fastest

Boundary Conditions

- 60s-80s: Lazy handling (IP filters are short anyways...)
- 90s: people started to care about the boundaries (splines, wavelets)
  - Input image: $K \times L$ array: $\{f[k,l]\}_{k=0,\ldots,K-1,l=0,\ldots,L-1}$
  - Extended image: $\{f_{\text{ext}}[k,l]\}_{(k,l)\in\mathbb{Z}^2}$
  - Filtered image: $g[k,l] = \sum_{(m,n)\in\mathbb{Z}^2} h[m,n] f_{\text{ext}}[k-m,l-n]$
- Lazy solution: Zero padding

$$f_{\text{ext}}[k,l] = 0 \quad \text{for} \quad (k,l) \notin [0,\ldots,K-1] \times [0,\ldots,L-1]$$

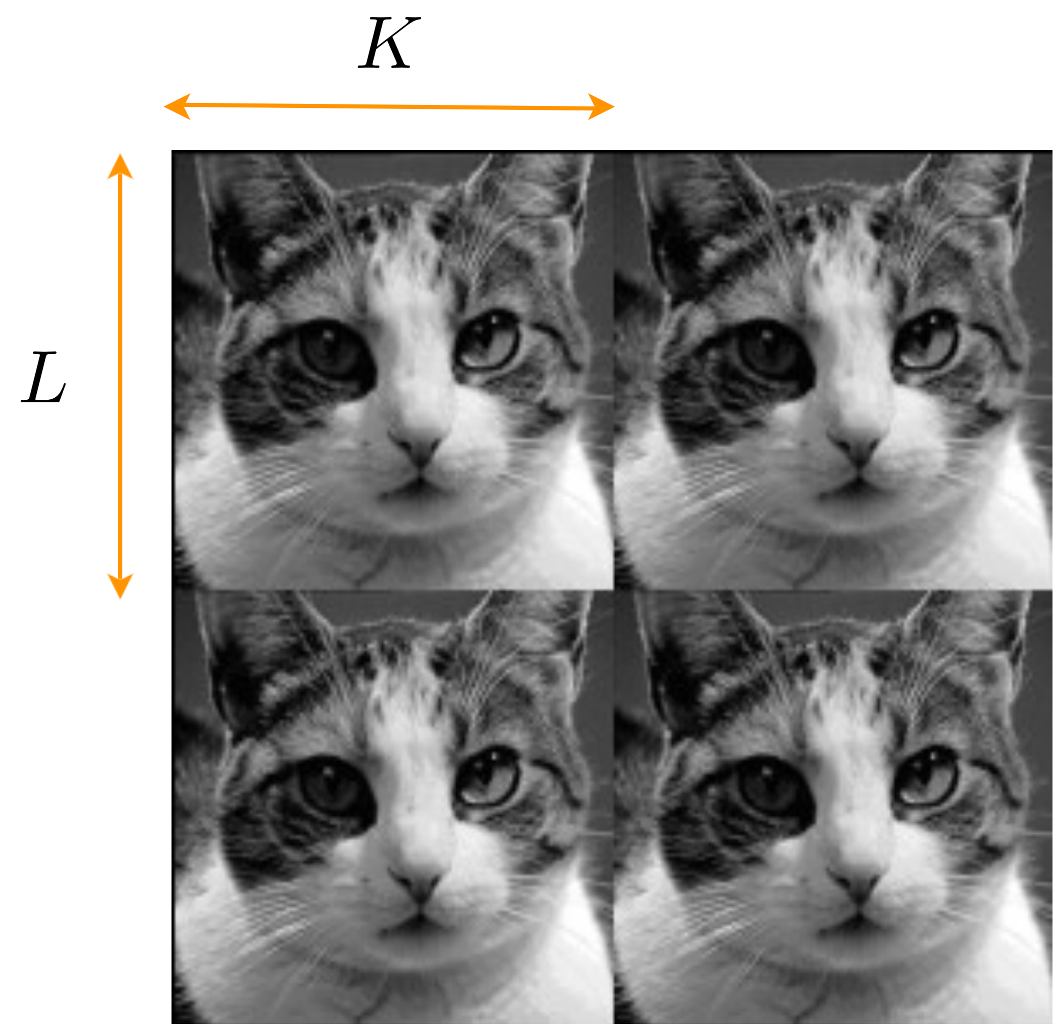


CAUTION: Lack of consistency;
i.e., filtered version of a zero-padded image is no longer zero at the boundaries.

Boundary Conditions (cont'd)

- Periodization

  $$f_{\text{ext}}[k,l] = f[k \bmod K, l \bmod L]$$



- Advantages
  - Simple to implement
  - Consistent: filtering a periodic image produces a periodic image
  - Periodization is implicit if filtering is performed with FFTs
- Disadvantage
  - Produces boundary artifacts

Boundary Conditions (cont'd)

- Symmetrization / mirror folding

$$\forall \boldsymbol{k} \in \mathbb{Z}^2, \quad f_{\text{ext}}[\boldsymbol{k}] = f_{\text{ext}}[-\boldsymbol{k}] \quad \text{and} \quad f_{\text{ext}}[\boldsymbol{k}_0 + \boldsymbol{k}] = f_{\text{ext}}[\boldsymbol{k}_0 - \boldsymbol{k}]$$

$$\boldsymbol{k}_0 = (K-1, L-1)$$

  - Image extension is $2\boldsymbol{k}_0$-periodic

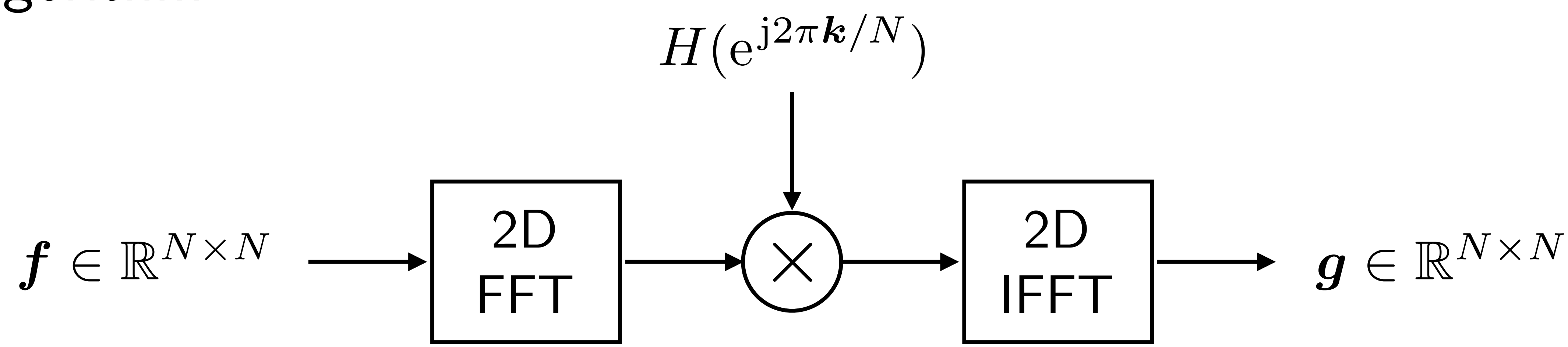
- Advantages
  - Simple to implement
  - Consistent: symmetric filtering a folded image produces a folded image; antisymmetric filtering yields an antisymmetric folded image
  - No boundary artifacts

Periodic Extensions and FFT-Based Filtering

Periodic convolution $\Leftrightarrow$ convolution with periodic extension

equivalent to FFT-based filtering

- Algorithm



What is the complexity of this algorithm?

1. 2D FFT of $N \times N$ image $O(N^2 \log N)$
2. Multiply with the FFT of the filter impulse response $O(N^2)$
3. Take the inverse FFT of the result $O(N^2 \log N)$

Total procedure complexity is $O(N^2 \log N)$

Fourier-Domain vs. Spatial-Domain Filtering

Long filters should be implemented in the Fourier domain!

Rule of thumb:

> FFT filtering starts paying off when the number of taps is greater than $8 \log_2 N$ in 1D, and $16 \log_2 N$ in 2D

However:

- Most usual image-processing filters are short (e.g., 3×3)

  $\Longrightarrow$ They are implemented most efficiently in the spatial domain

- Some classes of large filters can also be implemented efficiently in the spatial domain using recursive and/or multiscale algorithms

- Boundary conditions are handled best in the spatial domain

- Spatial-domain implementations gives more flexibility: spatially-adaptive filters, non-linear filtering, etc.

Useful Filters for Image Processing

- Smoothing
- Moving Average
- Symmetric Exponential Filter
- Gaussian Filter

Smoothing: The Universal Tool

- Spatial smoothing
  - Simulate a sampling aperature
  - Adjustable resolution
  - Flexibility?

How can we design other kinds of filters with smoothing?

- Primary applications
  - Image simplification
  - Noise reduction
  - Image enhancement
  - Feature extraction (image analysis)


original image


smoothed image
(low-pass filtering)


original − smoothed
(high-pass filtering)


$\text{smoothed}_1 - \text{smoothed}_2$
(band-pass filtering)

Smoothing (cont'd)

- Desirable features
  - Computational efficiency (fast)
  - Simplicity
  - Adjustable size
  - Symmetry

    (sensitivity of human-visual-system to phase distortion)
  - Shape of frequency response is not so important
  - Best to avoid sharp frequency cut-offs (Gibbs oscillations)

Efficient + Adjustable size $\Longrightarrow$ Separable + Recursive implementation

- Smoothing-filter requirements (1D)

Positivity: $h[k] \geq 0, \forall k \in \mathbb{Z}$

Unit gain: $\sum_{k \in \mathbb{Z}} h[k] = 1 \quad \Leftrightarrow \quad H(z)|_{z=1} = 1$

Symmetry: $h[k] = h[-k] \quad \Rightarrow \quad \sum_{k \in \mathbb{Z}} k\, h[k] = 0$

(centered at the origin)

"Window size": $\sigma^2 = \sum_{k \in \mathbb{Z}} k^2\, h[k]$

Example: Moving Average Filter

- $L_1 \times L_2$ moving average

$$g[k_1, k_2] = \frac{1}{L_1 L_2} \sum_{m=-\lfloor L_1/2 \rfloor}^{\lfloor L_1/2 \rfloor} \sum_{n=-\lfloor L_2/2 \rfloor}^{\lfloor L_2/2 \rfloor} f[k_1 - m, k_2 - n]$$

L_1 and L_2 are the horizontal and vertical window sizes (must be odd)

- Example: 3×3 moving average

$$\boldsymbol{h} = \frac{1}{9} \begin{bmatrix} 1 & 1 & 1 \\ 1 & \boxed{1} & 1 \\ 1 & 1 & 1 \end{bmatrix} = \frac{1}{3} \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix} \cdot \frac{1}{3} \begin{bmatrix} 1 & 1 & 1 \end{bmatrix} \qquad \text{(separable filter)}$$

- General case: separable transfer function

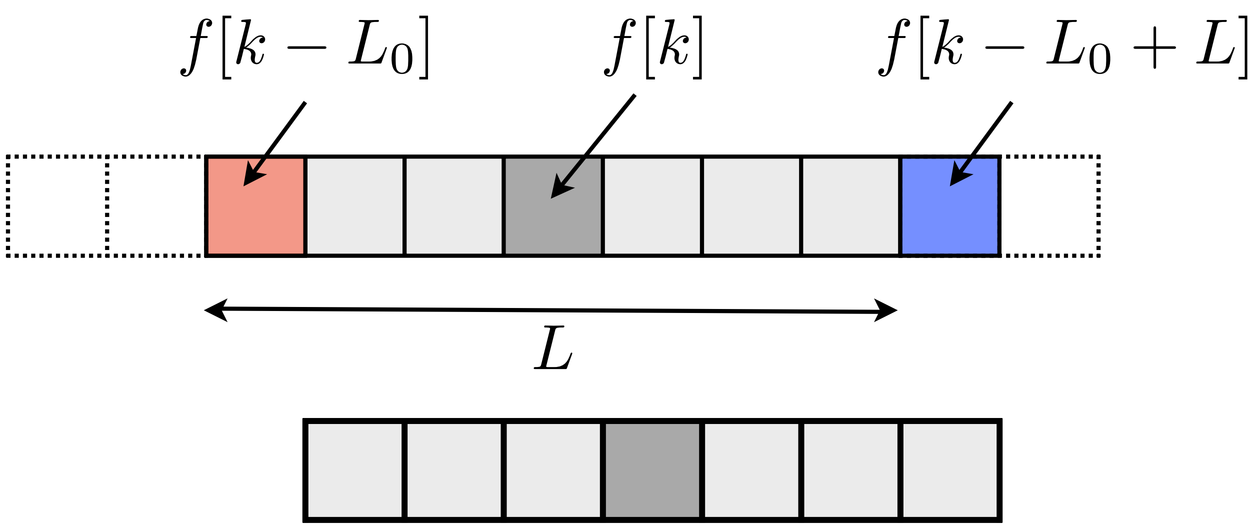
$$H(z_1, z_2) = H_{L_1}(z_1) H_{L_2}(z_2)$$

where $H_L(z) = \frac{1}{L} \sum_{k=-L_0}^{L_0} 1 \cdot z^{-k} = \frac{z^{-L_0}}{L} \sum_{k=0}^{L-1} z^k$ with $L_0 = \lfloor L/2 \rfloor$

Implementation by successive filter along rows and columns!

Moving Average Implementation

- Recursive implementation in 1D



$$L_0 = \lfloor L/2 \rfloor$$

$$g[k] = \frac{1}{L} \sum_{l=0}^{L-1} f[k - L_0 + l]$$

$$g[k+1] = ???$$

$$g[k+1] = g[k] + \frac{1}{L} \left(f[k - L_0 + L] - f[k - L_0] \right)$$

2 additions and 1 multiplication per sample, **irrespective** of L!

z-Transform: $zG(z) = G(z) + \frac{1}{L} F(z)(z^{L-L_0} - z^{-L_0})$

$$\Rightarrow H_L(z) = \frac{G(z)}{F(z)} = \frac{z^{-L_0}}{L} \left(\frac{z^L - 1}{z - 1} \right)$$

Moving Average: Transfer Function

- z-Transform

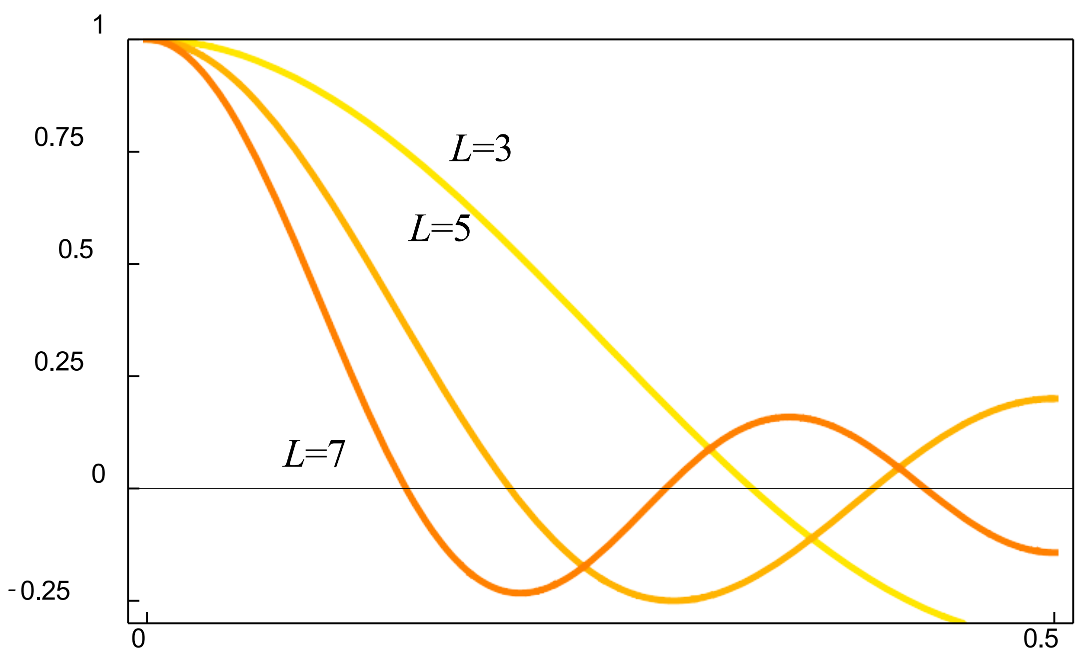
$$H_L(z) = \frac{G(z)}{F(z)} = \frac{z^{-L_0}}{L}\left(\frac{z^L - 1}{z - 1}\right)$$

Exercise: Compute the Fourier transform $H_L(\mathrm{e}^{\mathrm{j}\omega})$.

Hint: $L = 2L_0 + 1$

$$H_L(\mathrm{e}^{\mathrm{j}\omega}) = \frac{1}{L}\left(\frac{\mathrm{e}^{\mathrm{j}\omega(L_0+1)} - \mathrm{e}^{-\mathrm{j}\omega L_0}}{\mathrm{e}^{\mathrm{j}\omega} - 1}\right) = \frac{1}{L}\left(\frac{\mathrm{e}^{\mathrm{j}\omega L/2} - \mathrm{e}^{-\mathrm{j}\omega L/2}}{\mathrm{e}^{\mathrm{j}\omega/2} - \mathrm{e}^{-\mathrm{j}\omega/2}}\right)$$

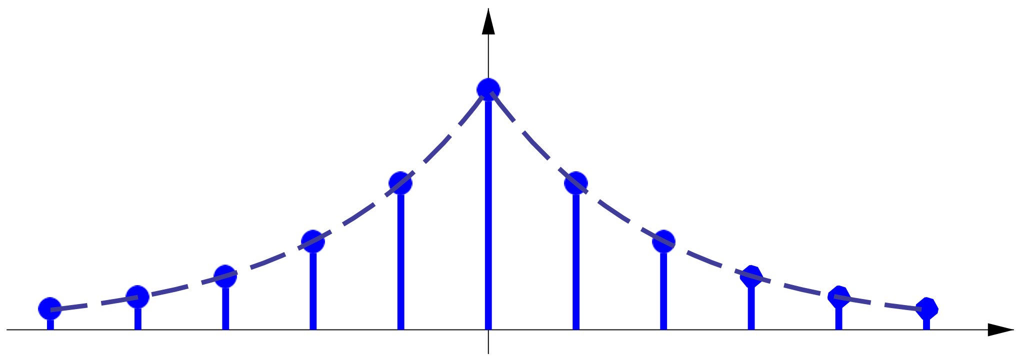
$$= \frac{1}{L}\frac{\sin(\omega L/2)}{\sin(\omega/2)}$$



As L increases, the filter becomes more low-pass

Symmetric Exponential Filter

- Impulse response



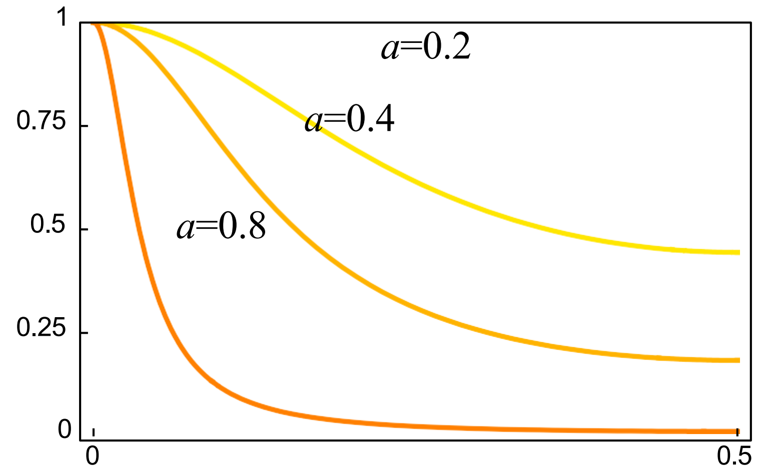
$$h[k_1, k_2] = C a_1^{|k_1|} a_2^{|k_2|}$$

$$C \text{ such that } \sum_{\boldsymbol{k} \in \mathbb{Z}^2} h[\boldsymbol{k}] = 1$$

- Separable transfer function

$$H(z_1, z_2) = H_{a_1}(z_1) H_{a_2}(z_2) \quad \text{where} \quad H_a(z) = \frac{C_a}{(1 - a z^{-1})(1 - a z)}$$

Implementation by successive filter along rows and columns!



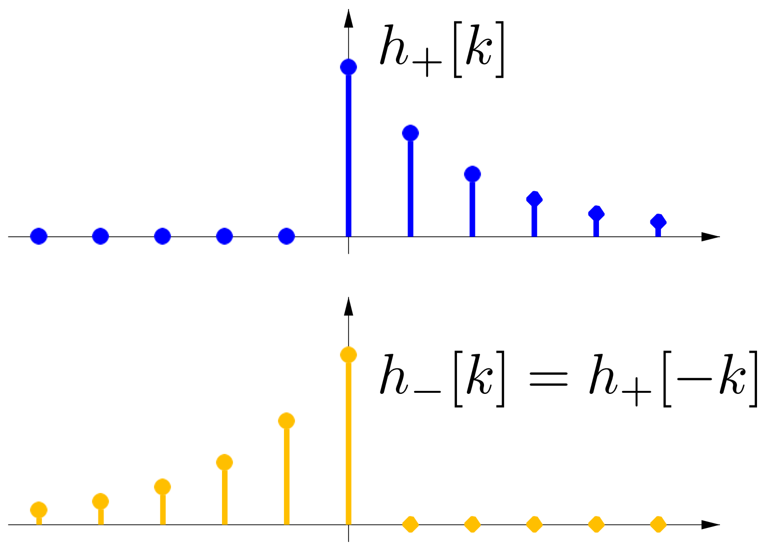
As a increases, the filter becomes more low-pass

Symmetric Exponential Construction (1D)

- Construction of a symmetric exponential

$$a^{|k|} = h_+[k] + h_+[-k] - \delta[k], \quad 0 < a < 1$$

$$h_+[k] = \begin{cases} a^k, & k \geq 0 \\ 0, & \text{else} \end{cases} \implies H_+(z) = \frac{1}{1 - az^{-1}}$$



- Transfer function

$$H_+(z) + H_+(z^{-1}) - 1 = \frac{1}{1 - az^{-1}} + \frac{1}{1 - az} - 1 = \frac{1 - a^2}{(1 - az^{-1})(1 - az)}$$

- Normalized exponential

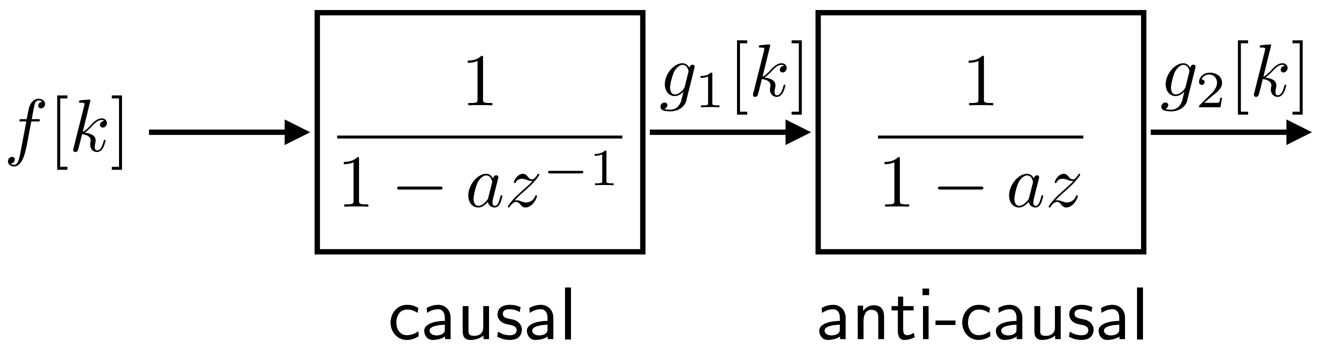
$$H_a(z) = \frac{C_a}{(1 - az^{-1})(1 - az)} \quad \text{such that} \quad \sum_{k \in \mathbb{Z}} h_a[k] = H_a(1) = 1 \ \Rightarrow C_a = \frac{1 - a}{1 + a}$$

$$h_a[k] = \left(\frac{1 - a}{1 + a}\right) a^{|k|} \quad \xleftrightarrow{z} \quad H_a(z) = \frac{(1 - a)^2}{(1 - az^{-1})(1 - az)}$$

Exponential Filtering: Implementation

- Exponential filter: $H_a(z) = \dfrac{C_a}{(1 - az^{-1})(1 - az)}$

Cascade of first-order recursive filters



$$G_1(z) = \frac{F(z)}{1 - az^{-1}} \quad \Rightarrow \quad G_1(z) = F(z) + az^{-1}G_1(z)$$

- Recursive-filtering algorithm

1. Causal filtering: $g_1[k] = f[k] + ag_1[k-1]$, for $k = 0, \ldots, N-1$
2. Anti-causal filtering: $g_2[k] = g_1[k] + ag_2[k-1]$, for $k = N-1, \ldots, 0$
3. Normalization: $g[k] = C_a\, g_2[k]$

Gaussian Filter

- 2D Gaussian impulse response

$$h_\sigma[k_1, k_2] = \frac{1}{2\pi\sigma^2} \exp\left(-\frac{(k_1^2 + k_2^2)}{2\sigma^2}\right) = \text{gaussian}(k_1; \sigma) \cdot \text{gaussian}(k_2; \sigma)$$

where $\text{gaussian}(k; \sigma) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{k^2}{2\sigma^2}\right)$

- Motivation for Gaussian filters
  - Only filter that is both circularly symmetric and separable

    $\Longrightarrow$ Implementation by successive filter along rows and columns!

  - Optimal space-frequency localization (uncertainty principle)

  - Linear scale space

Linear Scale Space and the Melting Cat

$u(x, y; t = 0)$

$\sigma = 1$

$\sigma = 2$

$\sigma = 4$

$\sigma = 8$

- Heat-flow interpretation

Diffusion equation (isotropic): $$\frac{\partial u(x, y; t)}{\partial t} = \Delta u(x, y; t)$$

General solution: $u(x, y; t) = u(x, y; t = 0) * \text{gaussian}(x, y; \sigma = \sqrt{2t})$

Central Limit Theorem (CLT)

- Probability density function (p.d.f.)

$$p(x) \geq 0 \qquad \int_{-\infty}^{\infty} p(x)\,\mathrm{d}x = 1$$

- Moments: mean and variance

$$X \sim p(x)$$

$$\mu = \mathbf{E}[X] = \int_{-\infty}^{\infty} x\,p(x)\,\mathrm{d}x$$

$$\sigma^2 = \mathrm{var}(X) = \int_{-\infty}^{\infty} (x-\mu)^2\,p(x)\,\mathrm{d}x$$

- Sum of two independent random variables (X_1 and X_2 i.i.d. from $p(x)$)

$$\mathrm{var}(X_1 + X_2) = \mathrm{var}(X_1) + \mathrm{var}(X_2) = 2\sigma^2$$

p.d.f. of the sum $p_{X_1+X_2}(x)$

$$= (p * p)(x)$$

- Sum of N i.i.d. random variables from $p(x)$

$$\mathrm{var}\left(\sum_{i=1}^{N} X_i\right) = \sum_{i=1}^{N} \mathrm{var}(X_i) = N\sigma^2$$

p.d.f. of the sum $p_{\mathrm{sum}}(x)$

$$= \underbrace{(p * p * \cdots * p)}_{N \text{ times}}(x)$$

CLT: $(p * p * \cdots * p)(x) \xrightarrow[N\to\infty]{} \dfrac{1}{\sqrt{2\pi N\sigma^2}} \exp\left(-\dfrac{(x-N\mu)^2}{2N\sigma^2}\right)$

Efficient Gaussian Filtering

- Convolution interpretation of the CLT

"The N-fold convolution of **any** low-pass filter converges to a Gaussian"

- Gaussian filtering by repeated moving average $(L = 2L_0 + 1)$

"window size" $\sigma^2 = \sum_{k \in \mathbb{Z}} k^2 h[k]$

$$\sigma_{\text{ave}}^2 = \frac{1}{2L_0 + 1} \sum_{k=-L_0}^{L_0} k^2 = \frac{L_0 + L_0^2}{3}$$

What does this mean?

$\Longrightarrow$ N-fold convolution is approximately Gaussian with $\sigma^2 = N\left(\frac{L_0 + L_0^2}{3}\right)$

Choose N and L_0 to adjust the approximate Gaussian filter

Efficient Gaussian Filtering

- Gaussian filtering by repeated low-pass filtering $H(z)$

How do we determine the window size from $H(z)$?

$$H(z) = \sum_{k \in \mathbb{Z}} h[k] z^{-k}$$

$$\frac{\mathrm{d}H(z)}{\mathrm{d}z} = \sum_{k \in \mathbb{Z}} h[k](-k) z^{-k-1}$$

$$\frac{\mathrm{d}^2 H(z)}{\mathrm{d}^2 z} = \sum_{k \in \mathbb{Z}} h[k](k+1) k z^{-k-2}$$

$$\left.\frac{\mathrm{d}^2 H(z)}{\mathrm{d}^2 z}\right|_{z=1} = \sum_{k \in \mathbb{Z}} k^2 \, h[k]$$

$\left(\sum_{k \in \mathbb{Z}} k \, h[k] = 0 \text{ by assumption} \right)$

Efficient Gaussian Filtering

- Gaussian filtering by repeated exponential filtering

$$H_a(z) = \frac{(1-a)^2}{(1-az^{-1})(1-az)}$$

Exercise: (i) Determine the window size
(ii) Determine the equivalent Gaussian variance after N iterations
(iii) Determine exponential parameter a for a desired σ and N

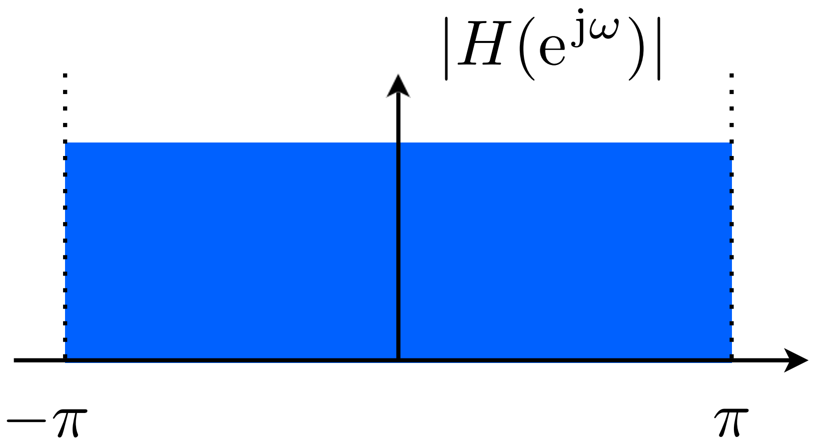
$$\sigma_{\mathrm{exp}}^2 = \frac{2a}{(1-a)^2}$$

$$N \text{ iterations} \implies \sigma^2 = \frac{2Na}{(1-a)^2}$$

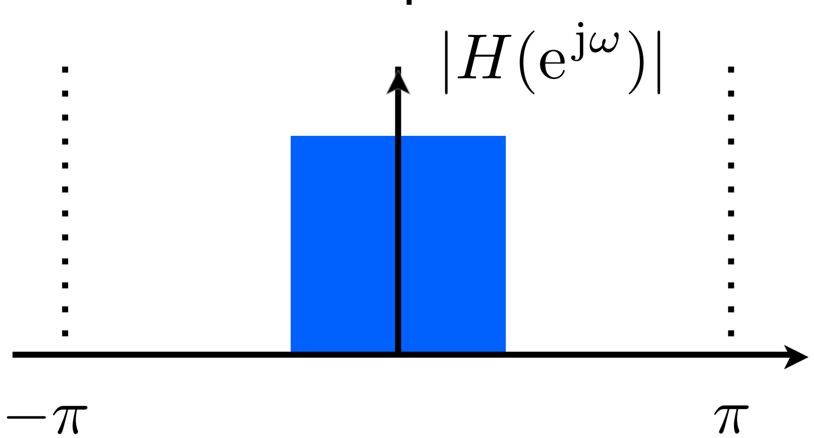
$$a = 1 + \frac{N}{\sigma^2} - \frac{\sqrt{N^2 + 2N\sigma^2}}{\sigma^2}$$

Nomenclature of Prototypical Filters

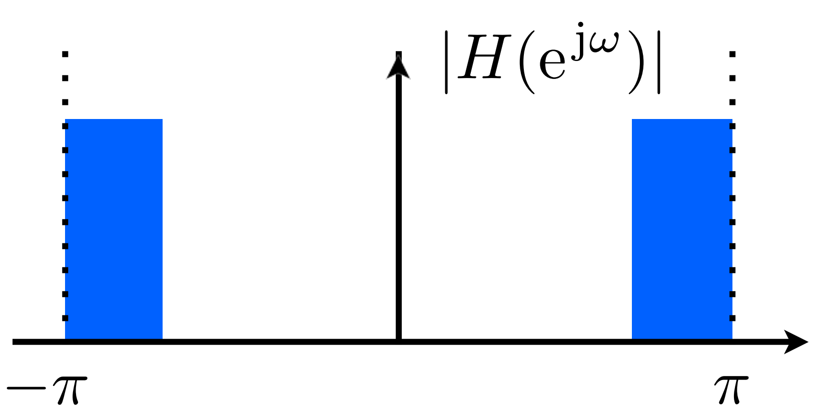
all-pass



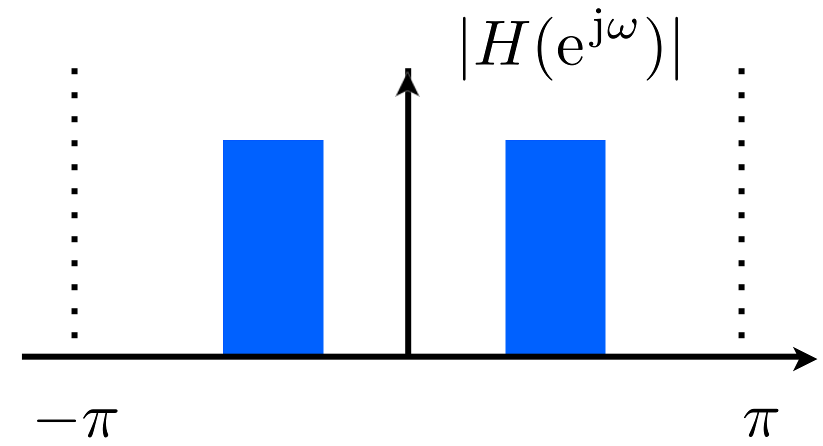
low-pass



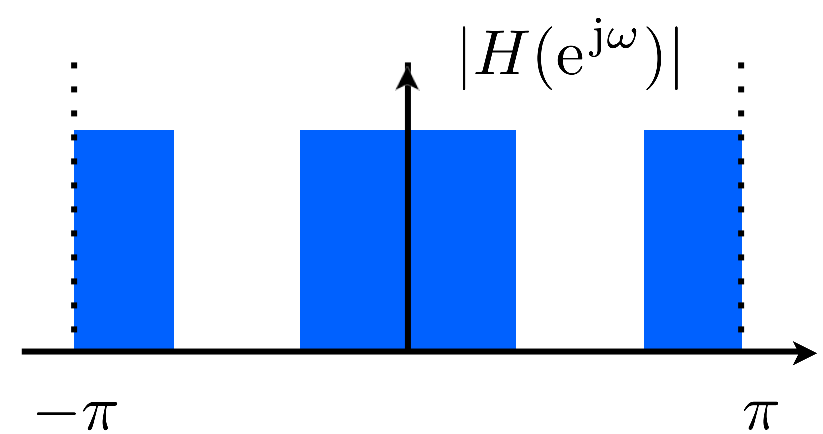
high-pass



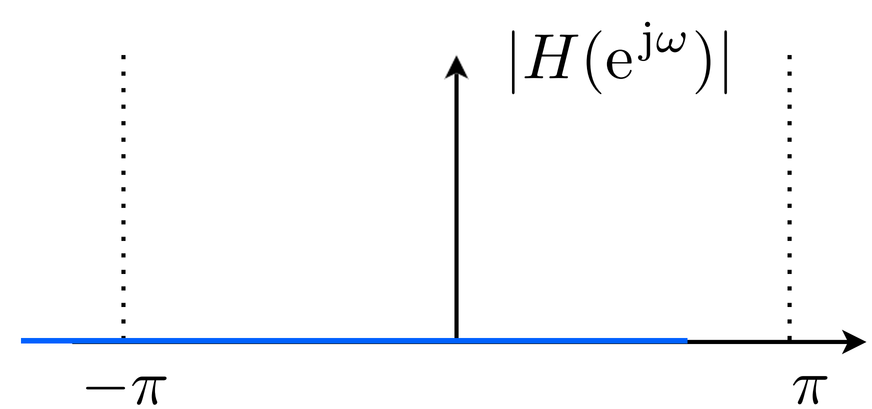
band-pass



band-cut



all-cut



Summary

- Discrete images are sequences indexed by two spatial integer variables. When they have finite energy, they can be viewed as points in the vector space $\ell^2(\mathbb{Z}^2)$.
- A discrete image is characterized by its 2D Fourier transform which is $(2\pi \times 2\pi)$-periodic.
- The 2D z-transform often provides a more convenient characterization. It is a direct vector generalization of the 1D transform. Thus, it has essentially the same properties.
- Discrete filtering can be described as a local masking operation, or as a discrete convolution. A 2D discrete filter is either described by a mask (which displays the reversed version of the impulse response), its transfer function, or its frequency response.
- When processing images, special care has to be taken to handle the boundaries (periodization or mirror folding).
- Many popular image-processing filters are short and separable. The computations are therefore usually performed in the spatial domain by successive filtering along the rows and columns.
- Very useful, low-complexity spatial smoothers are the moving average, the symmetric expo- nential, and the Gaussian filter. They can all be implemented recursively with a complexity independent of the window size.