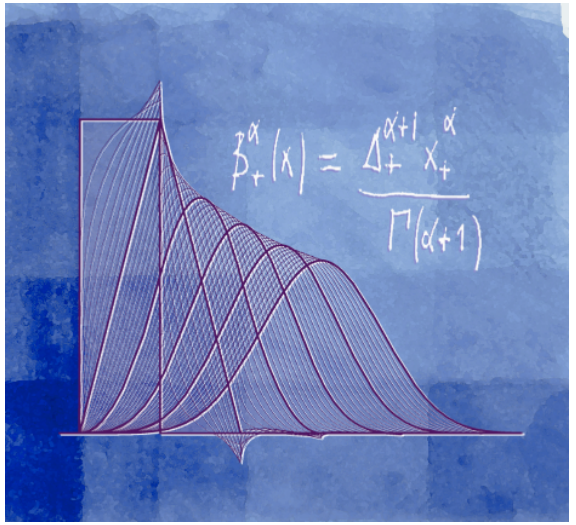
EPFL

Image Processing

Chapter 3

Characterization of discrete images and linear filtering

Prof. Michael Unser, LIB
Prof. Dimitri Van De Ville, MIPLAB



October-November 2025

CONTENT

3.1 Characterization of discrete images

- Discrete image representation
- Space of square-summable sequences
- Discrete-space Fourier transform
- Parseval relation
- Multidimensional z-transform
- z-transform properties

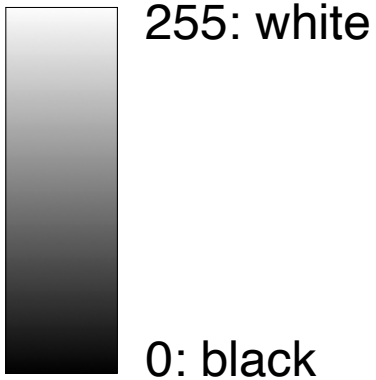
Discrete image representation

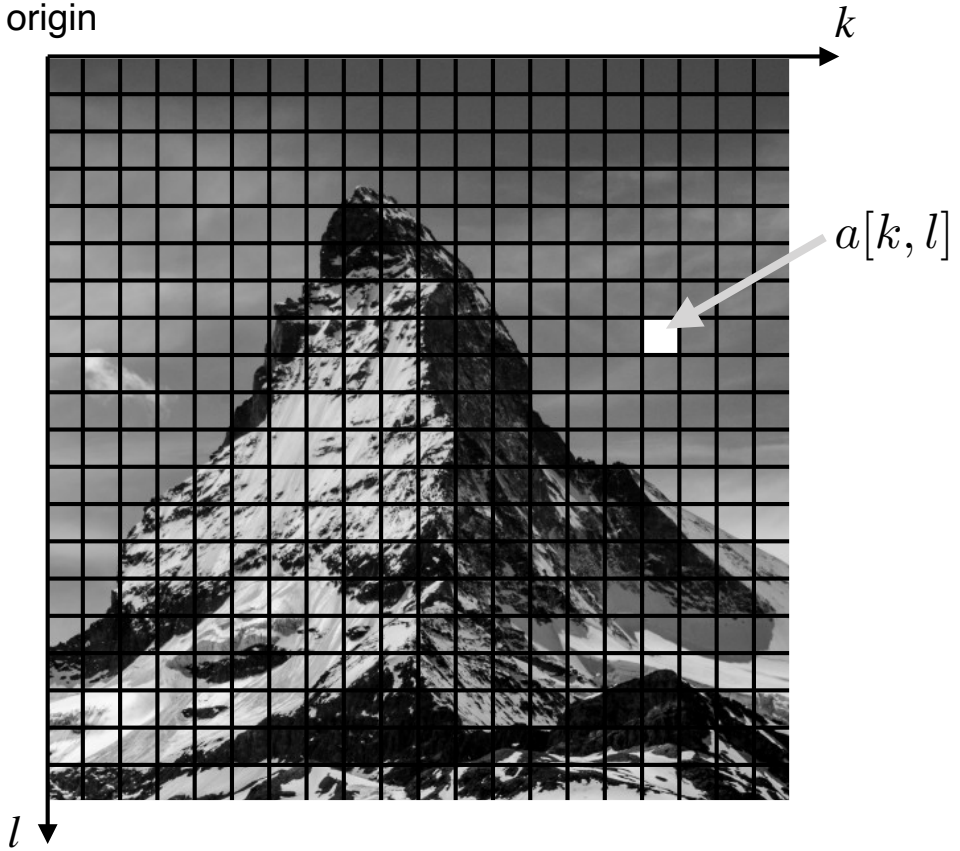
- Set of *pixels* (picture elements)

  $\{a[k,l]\}$ with $(k = 0, \cdots, K-1$ and $l = 0, \cdots, L-1)$

  K: number of columns

  L: number of rows (lines)





- Array of pixels of size $K \times L$

  Storage as a $L \times K$ Matlab matrix: $\mathbf{A} = [a_{i,j}]$ with $a_{i,j} = a[j-1, i-1]$

Space of square-summable images

- Images as 2D sequences of the space variables

  $a[k,l] \in \ell_2(\mathbb{Z}^2)$ or $a[\boldsymbol{k}]$ with $\boldsymbol{k} = (k,l) \in \mathbb{Z}^2$ (compact notation)

- 2D ℓ_2-inner product

  $$\langle a, b \rangle_{\ell_2} = \sum_{k \in \mathbb{Z}} \sum_{l \in \mathbb{Z}} a[k,l]\, b^*[k,l], \qquad \text{induced } \ell_2\text{-norm: } \|a\|_{\ell_2} = \sqrt{\langle a, a \rangle_{\ell_2}}$$

- Space of square-summable discrete images

  Hilbert space: $\ell_2(\mathbb{Z}^2) = \left\{ a[\boldsymbol{k}] : \boldsymbol{k} \in \mathbb{Z}^2, \|a\|_{\ell_2}^2 < +\infty \right\}$

- Extension to higher dimensions

  $a[\boldsymbol{k}]$ with $\boldsymbol{k} = (k_1, k_2, \cdots, k_d) \in \mathbb{Z}^d$

  $$\langle a, b \rangle_{\ell_2(\mathbb{Z}^d)} = \sum_{\boldsymbol{k} \in \mathbb{Z}^d} a[\boldsymbol{k}]\, b^*[\boldsymbol{k}], \qquad \ell_2(\mathbb{Z}^d) = \left\{ a[\boldsymbol{k}] : \boldsymbol{k} \in \mathbb{Z}^d, \|a\|_{\ell_2}^2 < +\infty \right\}$$

Discrete-space Fourier transform

- 2D discrete Fourier transform: definition

$$\hat{a}(\omega_1, \omega_2) = \sum_{k_1 \in \mathbb{Z}} \sum_{k_2 \in \mathbb{Z}} a[k_1, k_2]\, e^{-j(\omega_1 k_1 + \omega_2 k_2)}$$

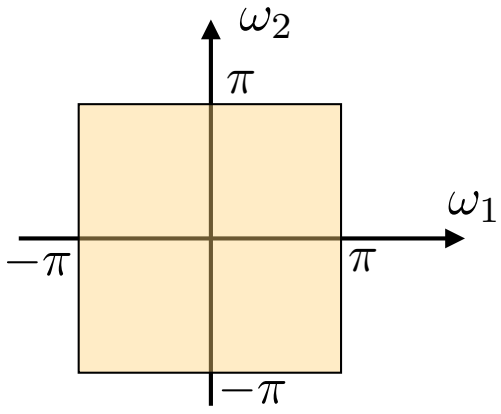
Sufficient condition for existence: $a \in \ell_1(\mathbb{Z}^2)$

Space of absolute-summable sequences

- $\ell_1(\mathbb{Z}^d) = \{a[\boldsymbol{k}] : \boldsymbol{k} \in \mathbb{Z}^d, \|a\|_{\ell_1} < +\infty\}$
- ℓ_1-norm: $\|a\|_{\ell_1} = \sum_{\boldsymbol{k} \in \mathbb{Z}^d} |a[\boldsymbol{k}]|$
- $\ell_1(\mathbb{Z}^d) \subset \ell_2(\mathbb{Z}^d)$

- 2π–periodicity

$$\hat{a}(\omega_1, \omega_2) = \hat{a}(\omega_1 + m2\pi, \omega_2 + n2\pi), \quad (m, n) \in \mathbb{Z}^2$$



Support of main Fourier period:

$$[-\pi, \pi]^2 = [-\pi, \pi] \times [-\pi, \pi]$$

- Inverse Fourier transform

$$a[k_1, k_2] = \frac{1}{(2\pi)^2} \int_{-\pi}^{+\pi} \int_{-\pi}^{+\pi} \hat{a}(\omega_1, \omega_2)\, e^{j(\omega_1 k_1 + \omega_2 k_2)} \mathrm{d}\omega_1 \mathrm{d}\omega_2$$

Multidimensional Fourier transform

- Multidimensional vector notation (d dimensions)
  - Spatial variables: $\boldsymbol{k} = (k_1, \ldots, k_d) \in \mathbb{Z}^d$
  - Frequency variables: $\boldsymbol{\omega} = (\omega_1, \ldots, \omega_d) \in \mathbb{R}^d$
  - Equivalent phase: $\langle \boldsymbol{\omega}, \boldsymbol{k} \rangle = \boldsymbol{\omega}^T \boldsymbol{k} = \omega_1 k_1 + \cdots + \omega_d k_d$

$$\hat{a}(\boldsymbol{\omega}) = \sum_{\boldsymbol{k} \in \mathbb{Z}^d} a[\boldsymbol{k}]\, e^{-j\langle \boldsymbol{\omega}, \boldsymbol{k} \rangle}$$

$$a[\boldsymbol{k}] = \frac{1}{(2\pi)^d} \int_{[-\pi,\pi]^d} \hat{a}(\boldsymbol{\omega})\, e^{j\langle \boldsymbol{\omega}, \boldsymbol{k} \rangle}\, \mathrm{d}\omega_1 \cdots \mathrm{d}\omega_d$$

$$[-\pi, \pi]^d = [-\pi, \pi] \times \cdots \times [-\pi, \pi]$$

- Classical framework

$a \in \ell_1(\mathbb{Z}^d) \quad \Rightarrow \quad \hat{a}(\boldsymbol{\omega})$ bounded, continuous

$$|\hat{a}(\boldsymbol{\omega})| \leq \sum_{\boldsymbol{k} \in \mathbb{Z}^d} \left| a[\boldsymbol{k}]\, e^{-j\langle \boldsymbol{\omega}, \boldsymbol{k} \rangle} \right| = \sum_{\boldsymbol{k} \in \mathbb{Z}^d} |a[\boldsymbol{k}]| = \|a\|_{\ell_1}$$

Parseval relation

- Discrete-space Fourier transform in ℓ_2

$$a \in \ell_2(\mathbb{Z}^d) \quad \Leftrightarrow \quad \hat{a}(\boldsymbol{\omega}) \in L_2([-\pi,\pi]^d)$$

Theorem: The complex sinusoids $\left\{e^{\pm j\langle \boldsymbol{\omega},\boldsymbol{k}\rangle}\right\}_{\boldsymbol{k}\in\mathbb{Z}^d}$ form an orthonormal basis of $L_2([-\pi,\pi]^d)$ with respect to the Hermitian inner-product: $\langle \hat{a},\hat{b}\rangle_{L_2([-\pi,\pi]^d)} = \dfrac{1}{(2\pi)^d}\displaystyle\int_{\boldsymbol{\omega}\in[-\pi,\pi]^d} \hat{a}(\boldsymbol{\omega})\,\hat{b}^*(\boldsymbol{\omega})\,\mathrm{d}\omega_1\cdots\mathrm{d}\omega_d.$

Dual Hilbert-space interpretation

- $\displaystyle\sum_{\boldsymbol{k}\in\mathbb{Z}^d} a[\boldsymbol{k}]e^{-j\langle\boldsymbol{\omega},\boldsymbol{k}\rangle}$: Fourier series expansion of the 2π-periodic function $\hat{a}(-\boldsymbol{\omega})$
- $a[\boldsymbol{k}] = \langle \hat{a}(\cdot), e^{-j\langle\cdot,\boldsymbol{k}\rangle}\rangle_{L_2([-\pi,\pi]^d)}$: standard formula for the Fourier coefficients

- Parseval's formula: $\langle a,b\rangle_{\ell_2} = \langle \hat{a},\hat{b}\rangle_{L_2([-\pi,\pi]^d)}$

$$\sum_{\boldsymbol{k}\in\mathbb{Z}^d} a[\boldsymbol{k}]\,b^*[\boldsymbol{k}] = \frac{1}{(2\pi)^d}\int_{\boldsymbol{\omega}\in[-\pi,\pi]^d} \hat{a}(\boldsymbol{\omega})\,\hat{b}^*(\boldsymbol{\omega})\,\mathrm{d}\omega_1\cdots\mathrm{d}\omega_d$$

- Preservation of the energy (isometry property)

$$\|a\|^2_{\ell_2(\mathbb{Z}^d)} = \|\hat{a}\|^2_{L_2([-\pi,\pi]^d)} = \frac{1}{(2\pi)^d}\int_{\boldsymbol{\omega}\in[-\pi,\pi]^d} |\hat{a}(\boldsymbol{\omega})|^2\,\mathrm{d}\omega_1\cdots\mathrm{d}\omega_d$$

Relation with continuous-space transform

- Shannon's representation of a bandlimited function

$$f_{\text{Shannon}}(\boldsymbol{x}) = \sum_{\boldsymbol{k}\in\mathbb{Z}^d} f[\boldsymbol{k}]\,\text{sinc}(\boldsymbol{x}-\boldsymbol{k})$$

Continuous-domain Fourier transform:

$$\begin{aligned}\mathcal{F}\{f_{\text{Shannon}}\}(\boldsymbol{\omega}) &= \sum_{\boldsymbol{k}\in\mathbb{Z}^d} f[\boldsymbol{k}]\,\mathcal{F}\{\text{sinc}(\cdot-\boldsymbol{k})\}(\boldsymbol{\omega}) && \text{(by linearity)}\\ &= \underbrace{\sum_{\boldsymbol{k}\in\mathbb{Z}^d} f[\boldsymbol{k}]e^{-j\langle\boldsymbol{\omega},\boldsymbol{k}\rangle}}\ \mathcal{F}\{\text{sinc}\}(\boldsymbol{\omega}) && \text{(shift property)}\end{aligned}$$

$$\hat{f}_{\text{Shannon}}(\boldsymbol{\omega}) = \hat{f}_{\text{discrete}}(\boldsymbol{\omega}) \times \text{rect}\left(\frac{\boldsymbol{\omega}}{2\pi}\right) = \begin{cases}\hat{f}_{\text{discrete}}(\boldsymbol{\omega}), & \text{for } \boldsymbol{\omega}\in[-\pi,\pi]^d\\ 0, & \text{otherwise.}\end{cases}$$

Note: this equivalence only holds when the function $f = f_{\text{Shannon}}$ is bandlimited.

Multidimensional z-transform

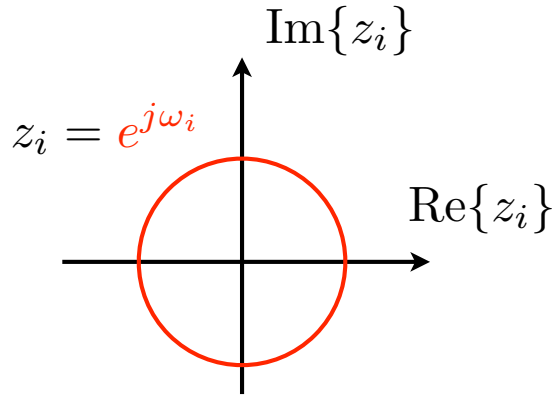
Complex variable: $\boldsymbol{z} = (z_1, \ldots, z_d) \in \mathbb{C}^d$, Space index: $\boldsymbol{k} = (k_1, \ldots, k_d) \in \mathbb{Z}^d$

- Definition
  - Multi-index exponent: $\boldsymbol{z}^{\boldsymbol{k}} \triangleq z_1^{k_1} z_2^{k_2} \cdots z_d^{k_d}$
  - z-transform : $A(\boldsymbol{z}) = \sum_{\boldsymbol{k} \in \mathbb{Z}^d} a[\boldsymbol{k}] \boldsymbol{z}^{-\boldsymbol{k}}$ for $\boldsymbol{z} \in$ ROC : Region of convergence

- Relation with the Fourier transform

$z_1 = e^{j\omega_1}, z_2 = e^{j\omega_2}, \ldots, z_d = e^{j\omega_d}$

unit circles:



Define: $e^{j\boldsymbol{\omega}} \triangleq (e^{j\omega_1}, \ldots, e^{j\omega_d})$

$$A(\boldsymbol{z})|_{\boldsymbol{z}=e^{j\boldsymbol{\omega}}} = \sum_{\boldsymbol{k} \in \mathbb{Z}^d} a[\boldsymbol{k}] \left(e^{j\boldsymbol{\omega}}\right)^{-\boldsymbol{k}} = \sum_{\boldsymbol{k} \in \mathbb{Z}^d} a[\boldsymbol{k}] e^{-j\omega_1 k_1} \cdots e^{-j\omega_d k_d}$$

$$\Rightarrow \quad A(e^{j\boldsymbol{\omega}}) = \hat{a}(\boldsymbol{\omega}) = \sum_{\boldsymbol{k} \in \mathbb{Z}^d} a[\boldsymbol{k}] e^{-j\langle \boldsymbol{\omega}, \boldsymbol{k} \rangle}$$

Region of convergence

ROC: region of $\mathbb{C}^d$ where $A(\boldsymbol{z}) = \sum_{\boldsymbol{k} \in \mathbb{Z}^d} a[\boldsymbol{k}] \boldsymbol{z}^{-\boldsymbol{k}}$ converges uniformly

- Practical constraint: convergent Fourier transform

  $\Rightarrow$ ROC must include the unit circles domain $z_1 = e^{j\omega_1}, \ldots, z_d = e^{j\omega_d}$

- Most cases of interest fall into these two categories:

  - $a[\boldsymbol{k}]$ is bounded and compactly supported (FIR)

    $\Rightarrow$ ROC $= \mathbb{C}^d \setminus \{\boldsymbol{0}\}$ (the complex hyperplane without the origin)

  - $a[\boldsymbol{k}] \in \ell_1(\mathbb{Z}^d)$

    $$\left|A(e^{j\boldsymbol{\omega}})\right| = \left|\sum_{\boldsymbol{k} \in \mathbb{Z}^d} a[\boldsymbol{k}] e^{-j\omega_1 k_1} \cdots e^{-j\omega_d k_d}\right| \leqslant \sum_{\boldsymbol{k} \in \mathbb{Z}^d} |a[\boldsymbol{k}]| = \|a\|_{\ell_1} < +\infty$$

    $\Rightarrow$ ROC includes the unit circles $z_1 = e^{j\omega_1}, \ldots, z_d = e^{j\omega_d}$

z-transform properties

Separability	$x[\boldsymbol{k}] = x_1[k_1] \times \cdots \times x_d[k_d]$	$\overset{z}{\longleftrightarrow}$	$X(\boldsymbol{z}) = X_1(z_1) \times \cdots \times X_d(z_d)$
Delay	$x[\boldsymbol{k} - \boldsymbol{k}_0]$	$\overset{z}{\longleftrightarrow}$	$\boldsymbol{z}^{-\boldsymbol{k}_0} X(\boldsymbol{z})$
Reflection	$x^T[\boldsymbol{k}] = x[-\boldsymbol{k}]$	$\overset{z}{\longleftrightarrow}$	$X(z_1^{-1}, \ldots, z_d^{-1})$
Convolution	$(h * x)[\boldsymbol{k}] = \sum_{\boldsymbol{k}_1 \in \mathbb{Z}^d} h[\boldsymbol{k}_1] x[\boldsymbol{k} - \boldsymbol{k}_1]$	$\overset{z}{\longleftrightarrow}$	$Y(\boldsymbol{z}) = H(\boldsymbol{z}) X(\boldsymbol{z})$

Sketch of proof:

$$Y(\boldsymbol{z}) = \sum_{\boldsymbol{k} \in \mathbb{Z}^d} \sum_{\boldsymbol{k}_1 \in \mathbb{Z}^d} h[\boldsymbol{k}_1] x[\boldsymbol{k} - \boldsymbol{k}_1] \boldsymbol{z}^{-\boldsymbol{k}}$$

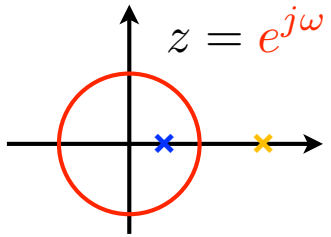
$\Updownarrow$ Change of variable $\boldsymbol{k} - \boldsymbol{k}_1 = \boldsymbol{k}_2$

$$Y(\boldsymbol{z}) = \sum_{\boldsymbol{k}_1 \in \mathbb{Z}^d} \sum_{\boldsymbol{k}_2 \in \mathbb{Z}^d} h[\boldsymbol{k}_1] x[\boldsymbol{k}_2] \boldsymbol{z}^{-(\boldsymbol{k}_1 + \boldsymbol{k}_2)}$$

$\Updownarrow$

$$Y(\boldsymbol{z}) = \sum_{\boldsymbol{k}_1 \in \mathbb{Z}^d} h[\boldsymbol{k}_1] \boldsymbol{z}^{-\boldsymbol{k}_1} \sum_{\boldsymbol{k}_2 \in \mathbb{Z}^d} x[\boldsymbol{k}_2] \boldsymbol{z}^{-\boldsymbol{k}_2} = H(\boldsymbol{z}) X(\boldsymbol{z})$$

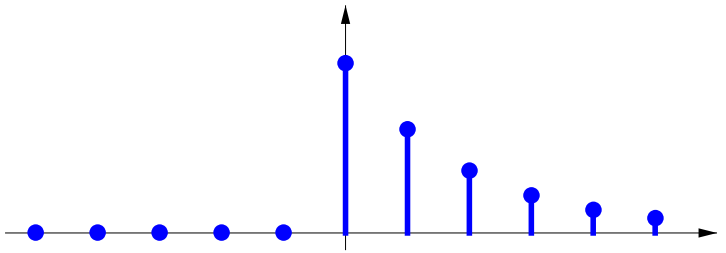
z-transform examples in 1D



- Definition: $X(z) = \sum_{k \in \mathbb{Z}} x[k] z^{-k}$

- Causal exponential

$$x_+[k] = \begin{cases} a^k, & k \geqslant 0 \\ 0, & \text{otherwise} \end{cases} \qquad \text{with } 0 < |a| < 1$$

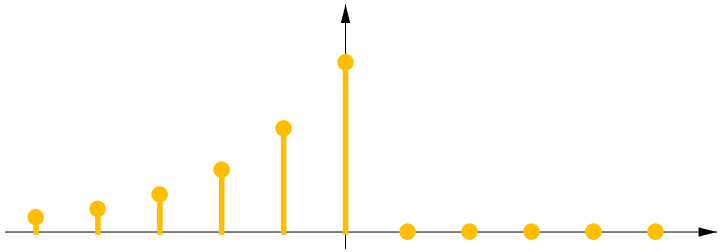


Geometric series

$$\Rightarrow \quad X_+(z) = \sum_{k=0}^{+\infty} (a/z)^k = \lim_{K \to +\infty} \left(\frac{1 - (a/z)^K}{1 - (a/z)} \right) = \frac{1}{1 - az^{-1}} = \frac{z}{z - a}, \qquad |z| > |a|$$

- Anti-causal exponential

$$x_-[k] = \begin{cases} a^{|k|}, & k \leqslant 0 \\ 0, & \text{otherwise} \end{cases} \qquad \text{with } 0 < |a| < 1$$

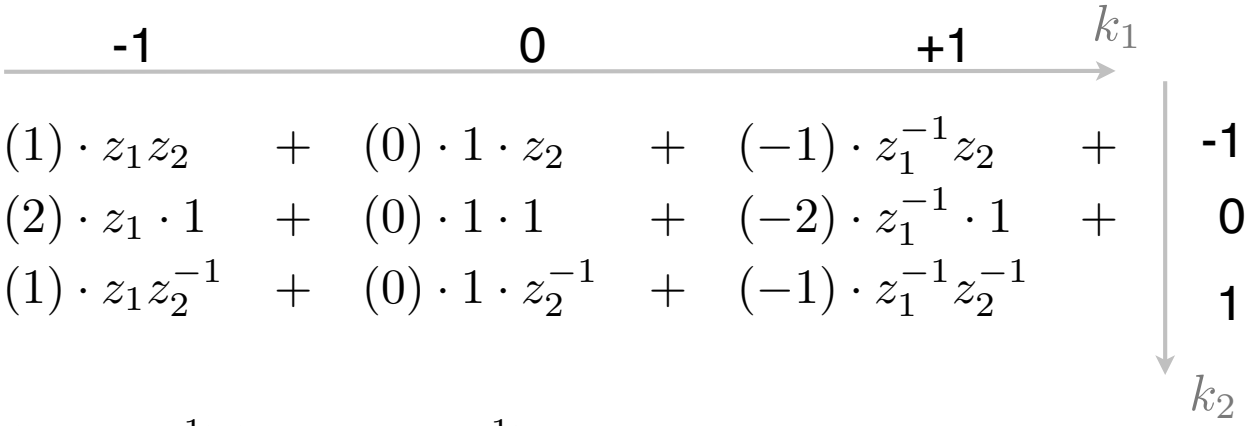


$$X_-(z) = X_+(z^{-1}) \quad \Rightarrow \quad X_-(z) = \frac{1}{1 - az} = -\frac{a^{-1}}{z - a^{-1}}, \qquad |z| < |a|^{-1}$$

z-transform example in 2D

- Basic formula: $X(z_1, z_2) = \sum_{k_1 \in \mathbb{Z}} \sum_{k_2 \in \mathbb{Z}} x[k_1, k_2] z_1^{-k_1} z_2^{-k_2}$

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



$$X(z_1, z_2) = \begin{array}{lllll} (1) \cdot z_1 z_2 & + & (0) \cdot 1 \cdot z_2 & + & (-1) \cdot z_1^{-1} z_2 \\ (2) \cdot z_1 \cdot 1 & + & (0) \cdot 1 \cdot 1 & + & (-2) \cdot z_1^{-1} \cdot 1 \\ (1) \cdot z_1 z_2^{-1} & + & (0) \cdot 1 \cdot z_2^{-1} & + & (-1) \cdot z_1^{-1} z_2^{-1} \end{array} +$$

$$= (z_1 - z_1^{-1})(z_2 + 2 + z_2^{-1}) \quad \Rightarrow \quad \text{Separable!}$$

$x[\boldsymbol{k}]$ bounded and compactly supported

$\Rightarrow$ ROC $= \mathbb{C}^2 \setminus \{\mathbf{0}\}$ (the complex hyperplane without the origin)

Inverse z-transform

- Identify the coefficients of the Laurent polynomial

$$X(z_1, z_2) = \sum_{k_1 \in \mathbb{Z}} \sum_{k_2 \in \mathbb{Z}} x[k_1, k_2] z_1^{-k_1} z_2^{-k_2}$$

- Take advantage of separability when it is present: $X(z_1, z_2) = X_1(z_1) \cdot X_2(z_2)$

- Reminder of 1D methods

  - Cauchy integral theorem: $x[k] = \frac{1}{2\pi j} \oint_\Gamma X(z) z^{k-1} \, \mathrm{d}z$

    Γ: any contour that encloses the origin

  - Use of tables and/or partial-fraction decomposition (linearity)

Example: $\frac{-3}{2z^{-1} - 5 + 2z} = \frac{3/4}{(1 - \frac{1}{2}z^{-1})(1 - \frac{1}{2}z)} = \frac{1}{1 - \frac{1}{2}z^{-1}} + \frac{1}{1 - \frac{1}{2}z} - 1$

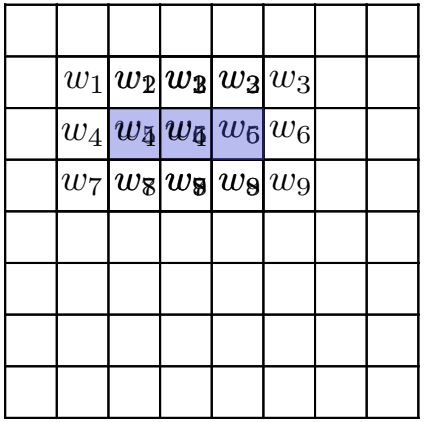
$$\Rightarrow \quad x[k] = \left(\frac{1}{2}\right)^{|k|} = u[k] \left(\frac{1}{2}\right)^{k} + u[-k] \left(\frac{1}{2}\right)^{-k} - \delta[k]$$

3.2 DIGITAL FILTERING

- Filtering with 2D masks
- Linearity and shift-invariance
- Impulse response and discrete convolution
- Equivalent filter characterizations
- Examples of transfer functions
- Separability
- z-transform and recursive filtering

Filtering with 2D masks

- Mask or local operator formulation



  - Filtering mask (weights) $(2M+1) \times (2N+1)$

$$\boldsymbol{w} = \begin{bmatrix} w[-M,-N] & \cdots & w[M,-N] \\ \vdots & \boxed{w[0,0]} & \vdots \\ w[-M,N] & \cdots & w[M,N] \end{bmatrix}$$

  - Local neighborhood vector

$$\boldsymbol{f}[k,l] = \begin{bmatrix} f[k-M,l-N] & \cdots & f[k+M,l-N] \\ \vdots & f[k,l] & \vdots \\ f[k-M,l+N] & \cdots & f[k+M,l+N] \end{bmatrix}$$

- Filtering: matrix formulation

$$g[\boldsymbol{k}] = \langle \boldsymbol{f}[\boldsymbol{k}], \boldsymbol{w} \rangle = \sum_i \sum_j [\boldsymbol{f}[\boldsymbol{k}]]_{i,j} [\boldsymbol{w}]_{i,j} \qquad \text{(term-by-term product)}$$

CAUTION: "correlation" formula

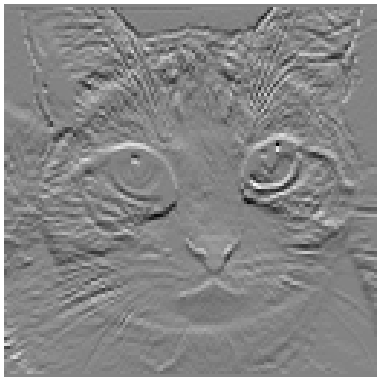
Filter examples

digital filter

Mask: $\boldsymbol{w}$

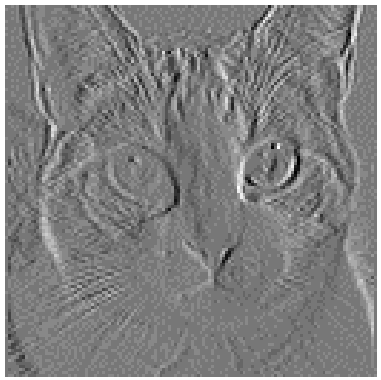
- Local 3×3 average

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



- Horizontal-edge enhancement

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



- Vertical-edge enhancement

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

Digital filtering: implementation

- Pseudo code (JAVA)

```
Input image : f (size K × L)
Output image : g (size K × L)
Local neighborhood array :  v (size M × M)
Mask array :  w (size M × M) (e.g., {{1,1,1}, {0,0,0}, {-1,-1,-1}} )

for (x=0 ; x<K ; x++) {
    for  (y=0 ; y<L ; y++) {
      v=f.getNeighborhood(x,y) ;
      outpix=0.0 ;
      for ( i=0 ; i< M ; i++) {
        for  (j= 0; j<M ; j++) {
          outpix=outpix+ v[i,j]*w[i,j] ;
         }
       }
       g.putPixel(x,y,outpix) ;
    }
}
```

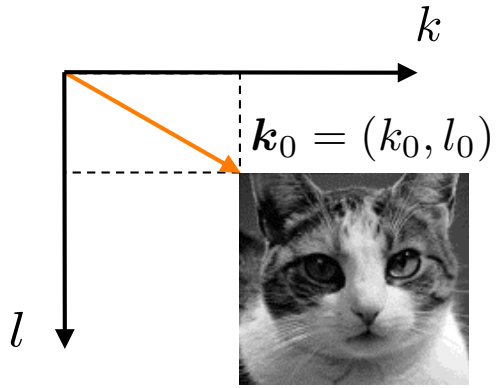
Linearity and shift-invariance

- Linearity

$a_1 f_1[\boldsymbol{k}] + a_2 f_2[\boldsymbol{k}]$ → $T_{\text{lin}}\{\}$ → $T_{\text{lin}}\{a_1 f_1 + a_2 f_2\}[\boldsymbol{k}] = a_1 T_{\text{lin}}\{f_1\}[\boldsymbol{k}] + a_2 T_{\text{lin}}\{f_2\}[\boldsymbol{k}]$

- Shift operator

$f[\boldsymbol{k}]$ → $S^{\boldsymbol{k}_0}\{\}$ → $f[\boldsymbol{k} - \boldsymbol{k}_0]$



- Shift-invariant filter

$f[\boldsymbol{k}]$ → $T_{\text{SI}}\{\}$ → $g[\boldsymbol{k}]$

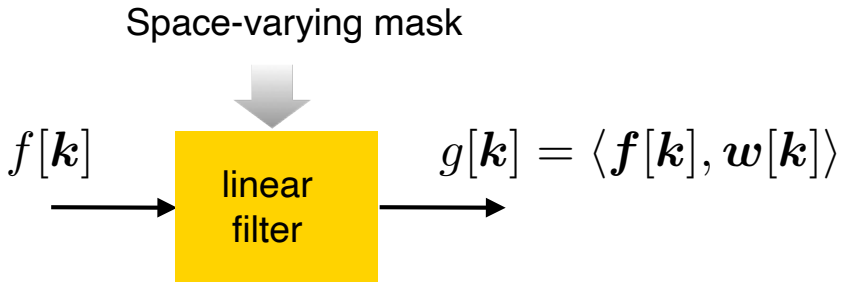
$f[\boldsymbol{k} - \boldsymbol{k}_0]$ → $T_{\text{SI}}\{\}$ → $g[\boldsymbol{k} - \boldsymbol{k}_0]$

Filter commutes with shift operator:

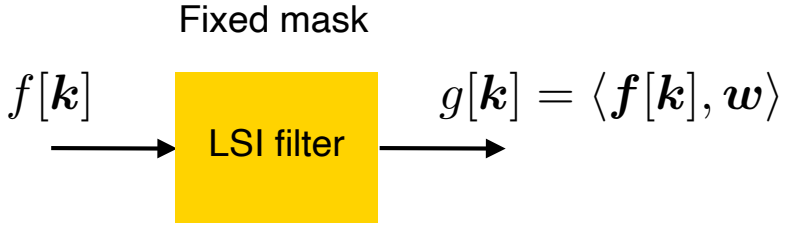
$S^{\boldsymbol{k}_0}\{\}$ → $T_{\text{SI}}\{\}$ ⇕ $T_{\text{SI}}\{\}$ → $S^{\boldsymbol{k}_0}\{\}$

Linear and shift-invariant filters

- Characterization of linear filters



- Characterization of linear, shift-invariant filters

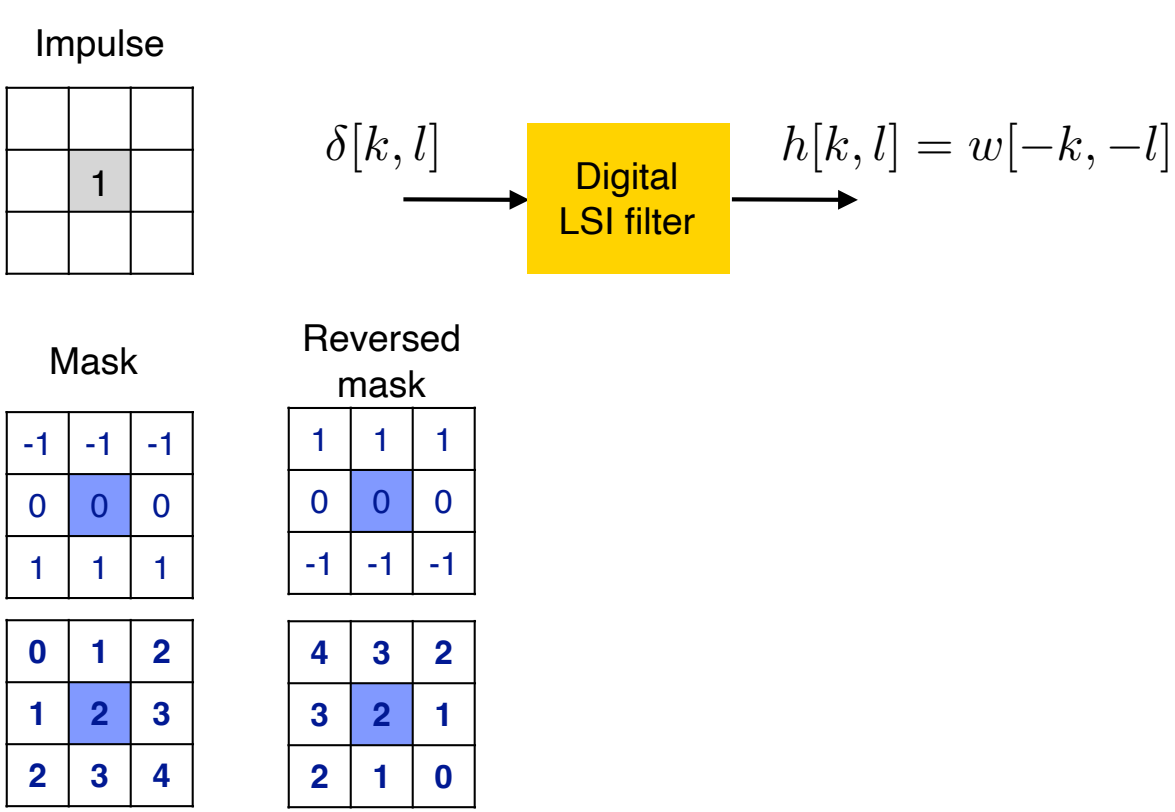


Example: $w = \begin{bmatrix} -1 & -1 & -1 \\ 0 & \boxed{0} & 0 \\ 1 & 1 & 1 \end{bmatrix}$

Converting a mask into an impulse response

- Filter implementation using a mask

$$g[k,l] = \sum_i \sum_j w[i,j]\, f[k+i, l+j] \qquad \text{(correlation)}$$



Impulse response = **space-reversed** version of the mask

Impulse response and discrete convolution

- Impulse response (e.g., discrete point-spread function)

Unit impulse (Kronecker delta) → Digital LSI filter → Impulse response

$$\delta[\boldsymbol{k}] \quad \longrightarrow \quad \text{Digital LSI filter} \quad \longrightarrow \quad h[\boldsymbol{k}] = \mathrm{T}_{\mathrm{LSI}}\{\delta[\cdot]\}[\boldsymbol{k}]$$

Property : A digital LSI filter is uniquely characterized by its impulse response, which is the **space-reversed** version of its mask : $h[\boldsymbol{k}] = w[-\boldsymbol{k}]$

- Equivalent convolution operator

$$f[k,l] \quad \longrightarrow \quad \text{Digital LSI filter} \quad \longrightarrow \quad g[k,l] = (h * f)[k,l] = \sum_{m\in\mathbb{Z}} \sum_{n\in\mathbb{Z}} f[m,n]h[k-m,l-n]$$

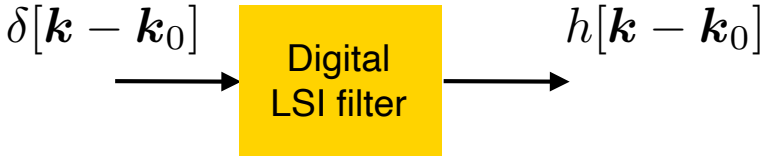
Discrete convolution

Convolution: breaking it into steps

- Unit impulse (pixel) at the origin

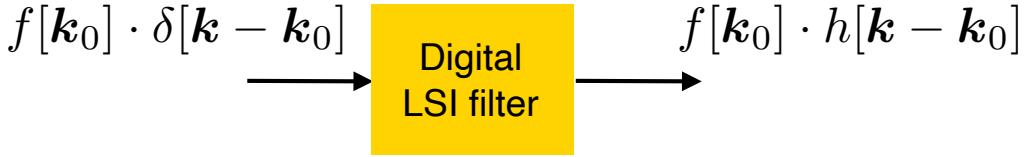
Definition of impulse response

- Unit impulse (pixel) at $\boldsymbol{k}_0$



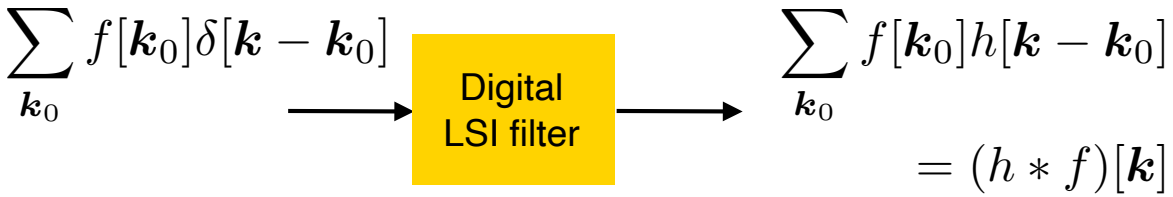
Shift-invariance

- Pixel at $\boldsymbol{k}_0$ with value $f[\boldsymbol{k}_0]$



Linearity

- Input image = sum of pixels



Superposition

Equivalent filter characterizations

Digital LSI filter = Discrete convolution operator

$$T_{\mathrm{LSI}}\{f\}[\boldsymbol{k}] = (h * f)[\boldsymbol{k}] = \sum_{\boldsymbol{n}\in\mathbb{Z}^d} h[\boldsymbol{n}] f[\boldsymbol{k}-\boldsymbol{n}] \quad \overset{z}{\longleftrightarrow} \quad G(\boldsymbol{z}) = H(\boldsymbol{z})F(\boldsymbol{z})$$

- Impulse response: $T_{\mathrm{LSI}}\left\{\delta[\cdot]\right\}[\boldsymbol{k}] = h[\boldsymbol{k}]$

- Transfer function: $H(z_1, z_2) = \sum_{k_1\in\mathbb{Z}} \sum_{k_2\in\mathbb{Z}} h[k_1, k_2] z_1^{-k_1} z_2^{-k_2}$

- Frequency response: $H(e^{j\omega_1}, e^{j\omega_2}) = \sum_{k_1\in\mathbb{Z}} \sum_{k_2\in\mathbb{Z}} h[k_1, k_2] e^{-jk_1\omega_1} e^{-jk_2\omega_2}$

Response to a complex sinusoid: $T_{\mathrm{LSI}}\left\{e^{j\langle\boldsymbol{\omega},\boldsymbol{k}'\rangle}\right\}[\boldsymbol{k}] = e^{j\langle\boldsymbol{\omega},\boldsymbol{k}\rangle} \cdot H(e^{j\boldsymbol{\omega}})$

Proof: $\left(h * e^{j\langle\boldsymbol{\omega},\cdot\rangle}\right)[\boldsymbol{k}] = \sum_{\boldsymbol{n}\in\mathbb{Z}^d} h[\boldsymbol{n}] e^{j\langle\boldsymbol{\omega},\boldsymbol{k}-\boldsymbol{n}\rangle} = e^{j\langle\boldsymbol{\omega},\boldsymbol{k}\rangle} \sum_{\boldsymbol{n}\in\mathbb{Z}^d} h[\boldsymbol{n}] e^{-j\langle\boldsymbol{\omega},\boldsymbol{n}\rangle}$

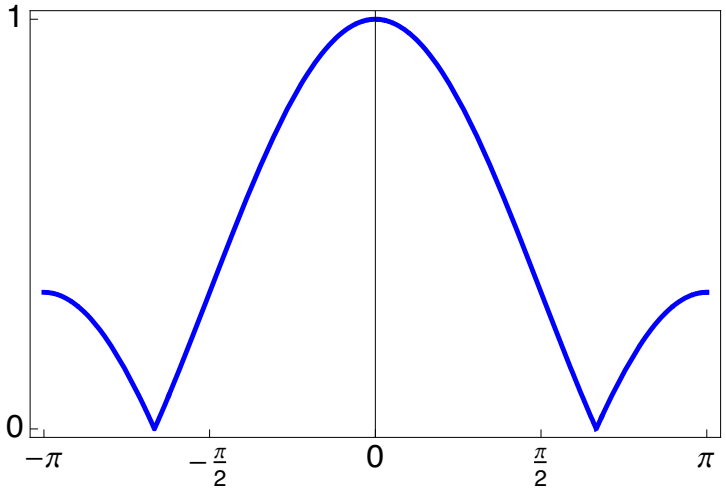
Example 1: local 3 × 3 average

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

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

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

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



lowpass behavior

Example 2: vertical-edge enhancer

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

impulse response

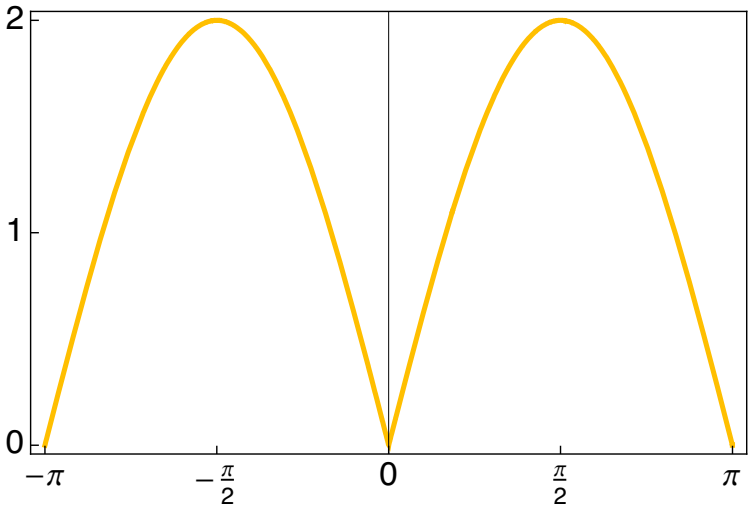
1	0	-1
1	0	-1
1	0	-1

"correlation"

"convolution"

- Transfer function: $H_{\text{vert}}(z_1, z_2) = (z_1 - z_1^{-1})(z_2 + 1 + z_2^{-1})$

- Frequency response: $H_{\text{vert}}(e^{j\omega_1}, e^{j\omega_2}) = \underbrace{(2j \sin \omega_1)}_{\text{bandpass}} \underbrace{(1 + 2\cos \omega_2)}_{\text{lowpass}}$



$\left|H_{\text{band}}(e^{j\omega_1})\right|$

Separability

Most useful image processing filters are separable...
which brings us back to a 1D problem

- Definition of separability

$$h[k_1, k_2] = h_1[k_1] \cdot h_2[k_2] \quad \rightarrow \quad h[\boldsymbol{k}] = \prod_{i=1}^{d} h_i[k_i]$$

$$\Updownarrow$$

$$H(z_1, z_2) = H_1(z_1) \cdot H_2(z_2) \quad \rightarrow \quad H(\boldsymbol{z}) = \prod_{i=1}^{d} H_i(z_i)$$

$$\Updownarrow$$

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

$$\Updownarrow$$

$$\boldsymbol{h} = \boldsymbol{h}_1^T \otimes \boldsymbol{h}_2 = \left[h_1[-k_0] \cdots h_1[-k_0 + M - 1]\right] \otimes \begin{bmatrix} h_2[-l_0] \\ \vdots \\ h_2[-l_0 + N - 1] \end{bmatrix}$$

Separability and direct (or tensor) products

- Direct vector product = $M \times N$ multiplication table

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

×	1	0	-1
1	1	0	-1
2	2	0	-2
1	1	0	-1

Example: $\boldsymbol{a} = (1, 0, -1), \quad \boldsymbol{b} = (1, 2, 1)$

- Definition of direct matrix product

$\boldsymbol{A}$: $P \times Q$ matrix ; $\boldsymbol{B}$: $M \times N$ matrix

$$\boldsymbol{C} = \boldsymbol{A} \otimes \boldsymbol{B} = \begin{bmatrix} \boldsymbol{A} b_{11} & \boldsymbol{A} b_{12} & \cdots & \boldsymbol{A} b_{1N} \\ \boldsymbol{A} b_{21} & & \cdots & \boldsymbol{A} b_{2N} \\ \vdots & & & \vdots \\ \boldsymbol{A} b_{M1} & & & \boldsymbol{A} b_{MN} \end{bmatrix} \qquad PM \times QN \text{ matrix}$$

Example: 3 × 3 smoother

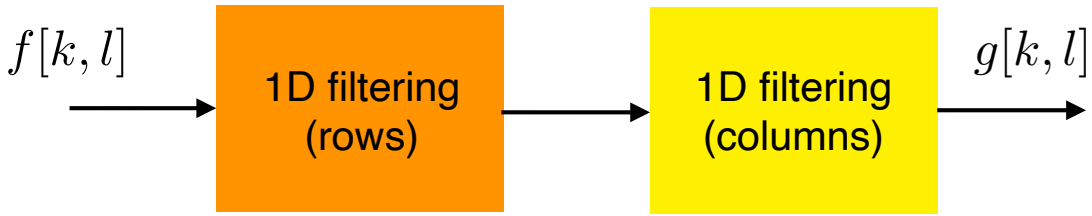
$$\boldsymbol{h}_{\mathrm{s}} = \frac{1}{16}\begin{bmatrix} 1 & 2 & 1 \\ 2 & \boxed{4} & 2 \\ 1 & 2 & 1 \end{bmatrix} = \frac{1}{4}\begin{bmatrix} 1 & \boxed{2} & 1 \end{bmatrix} \otimes \frac{1}{4}\begin{bmatrix} 1 \\ \boxed{2} \\ 1 \end{bmatrix}$$

$$h[k,l] = h_1[k] \cdot h_1[l] \qquad \text{where } h_1[k] = \begin{cases} 1/4, & k = \pm 1 \\ 1/2, & k = 0 \\ 0, & \text{otherwise} \end{cases}$$

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

- Limitation of separability
  - Orientation-sensitive filters are in general non-separable

Separable filtering: implementation



- Design: 1D filtering routine

  input: $u[n]$ with $(n = 0, \ldots, N-1)$

  output: $v[n]$

  ```
  function  v = filter1d(u,N)
  ```

- Data handling

  array-to-line conversion

  `getrow`, `putrow`

  `getcolumn`, `putcolumn`

- Generic separable filtering algorithm

  input: $f[k,l]$ with $(k = 0, \ldots, K-1,\ l = 0, \ldots, L-1)$

  output: $g[k,l]$

```
for (int j=0; j<L; j++) {
   u=getrow(f,j);
   v=filter1d(u,K);
   putrow(g,j,v);
}

for (int i=0; i<K; i++) {
   u=getcolumn(g,i);
   v=filter1d(u,L);
   putcolumn(g,i,v);
}
```

  etc... (for 3D or more)

z-transform and recursive filtering

- Rational transfer functions and difference equations

$$H(z) = \frac{Y(z)}{X(z)} = \frac{\sum_{m=0}^{M-1} b_m z^{-m}}{\sum_{n=0}^{N-1} a_n z^{-n}} \qquad \Leftrightarrow \qquad \sum_{n=0}^{N-1} a_n y[k-n] = \sum_{m=0}^{M-1} b_m x[k-m]$$

recursive-filter implementation

- Example: causal exponential

$$Y(z) = \left(\frac{1}{1 - z^{-1} a_1} \right) X(z) \qquad \Leftrightarrow \qquad y[k] = x[k] + a_1 y[k-1]$$

- Stability of rational filters

No poles z_i on the unit circle $z = e^{j\omega}$

Causal part: $|z_i| < 1$

Anti-causal part: $|z_i| > 1$

Recursive filtering and stability

- Recursive filter implementation

  - Example: causal exponential ($|z| > |a_1|$, pole $z = a_1$)

$$Y(z) = \left(\frac{1}{1 - z^{-1} a_1} \right) X(z) \qquad \Leftrightarrow \qquad y[k] = x[k] + a_1 y[k-1]$$

  - Example: anti-causal exponential ($|z| < |a_1|^{-1}$, pole $z = a_1^{-1}$)

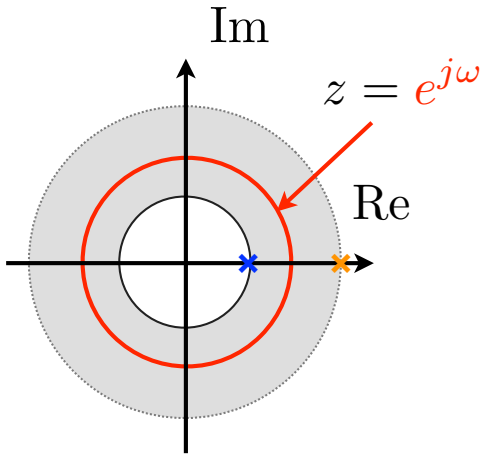
$$Y(z) = \left(\frac{1}{1 - z a_1} \right) X(z) \qquad \Leftrightarrow \qquad y[k] = x[k] + a_1 y[k+1]$$

- Stability of rational filters

  No poles z_i on the unit circle $z = e^{j\omega}$

  Causal part: $|z_i| < 1$

  Anti-causal part: $|z_i| > 1$

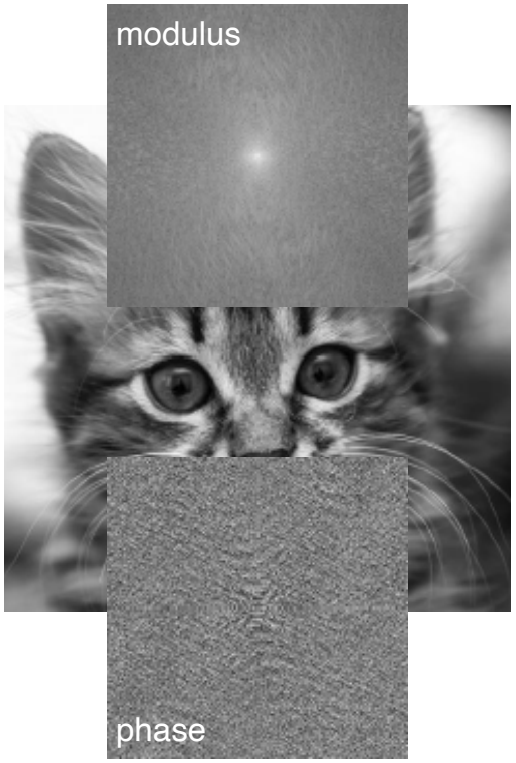


3.3 Filtering: practical considerations

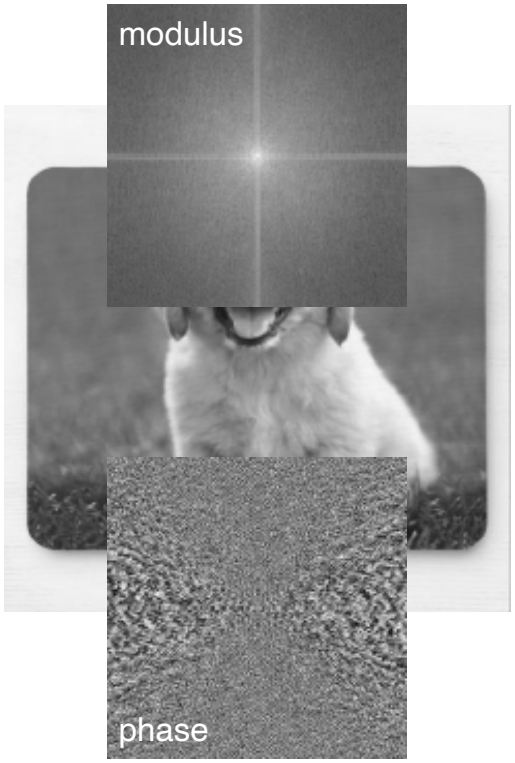
- About filter design for image processing
- Boundary conditions
- Fourier versus space-domain implementation

About filter design for image processing

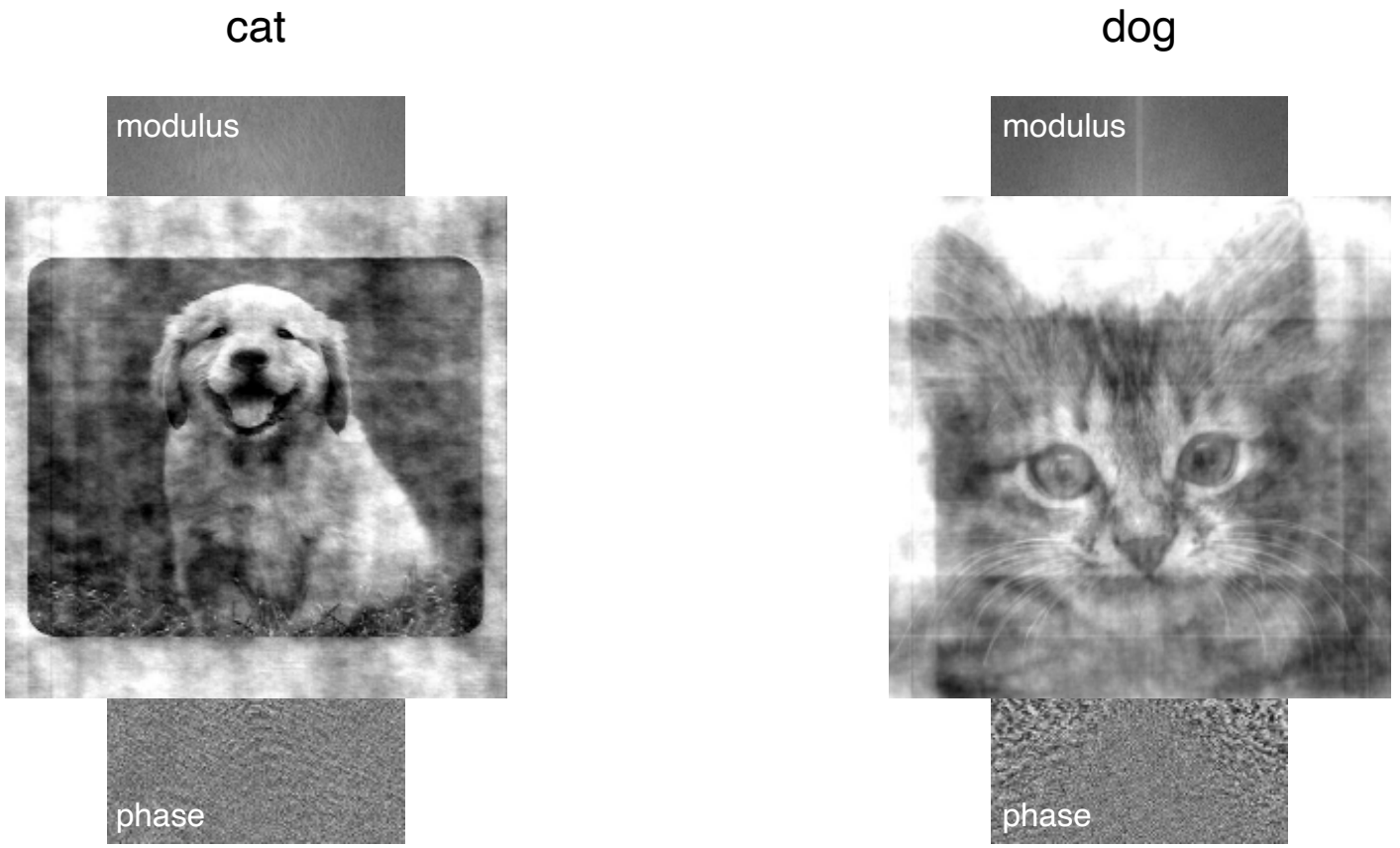
cat



dog



About filter design for image processing



- Semantic information (edges, contours) is contained in the phase of the Fourier transform

  $\Rightarrow$ Use linear-phase filters; i.e., symmetric or antisymmetric

- Exact shape of the frequency response is not so important

  $\Rightarrow$ Go for the simplest and fastest. . .

Boundary conditions

"Thou should not neglect what happens at the boundaries."

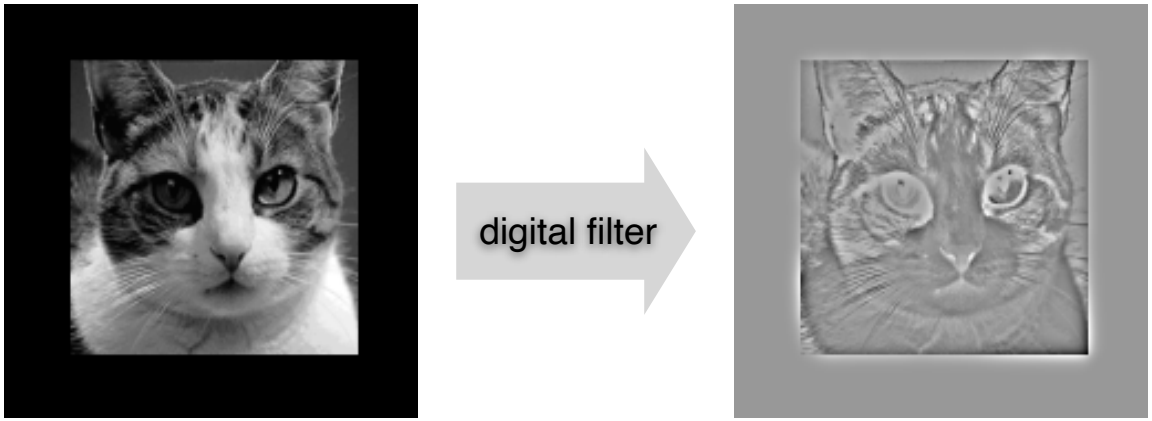
60's-80's: lazy handling (IP filters are short anyway...)

90's: consistent handling becomes an important issue (e.g., splines, wavelets)

- Input image: $K \times L$ array $\quad \{f[k,l]\}_{k=0,\ldots,K-1,l=0,\ldots,L-1}$
- Extended image: $\quad \{f_{\text{ext}}[k,l]\}_{(k,l)\in\mathbb{Z}^2}$
- Filtered output: $\quad g[k,l] = \sum_{(m,n)\in\mathbb{Z}^2} h[m,n] f_{\text{ext}}[k-m,l-n]$

- Lazy solution: zero padding

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



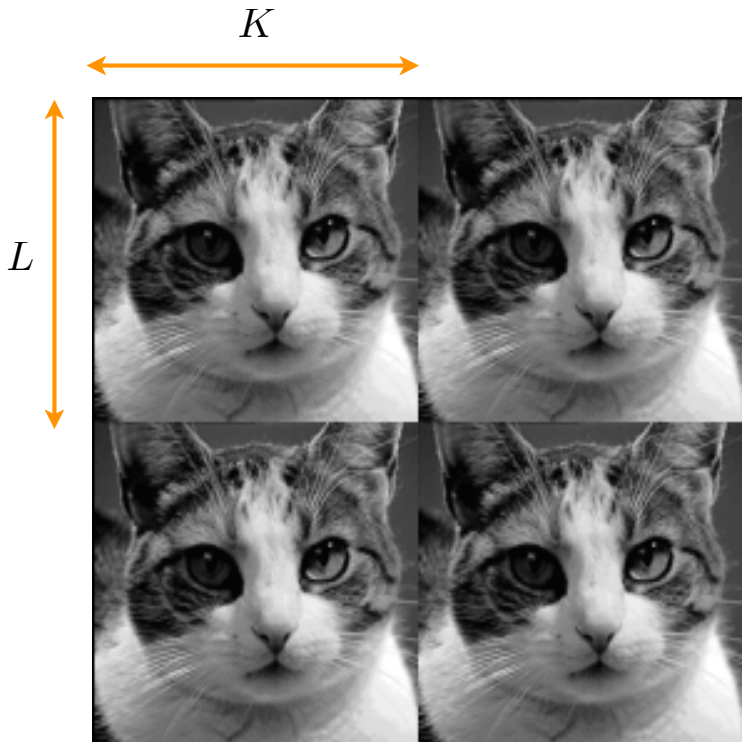
CAUTION: Lack of consistency;
i.e., filtered version of a zero-padded signal is no longer zero at the boundaries.

Boundary conditions (Cont'd)

- Periodization

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

```
function p=getpixel(f,k,l);  {
   i=k mod K;
   j=l mod L;
   p=f(i,j);
}
```



- Advantages
  - Simple to implement
  - Consistent: the filtering of a periodic signal produces a periodic signal
  - Periodization is implicit if filtering is performed in the Fourier domain (FFT algorithm)
- Disadvantage
  - Produces boundary artifacts

Boundary conditions (Cont'd)

- Symmetrization / mirror folding

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

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

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

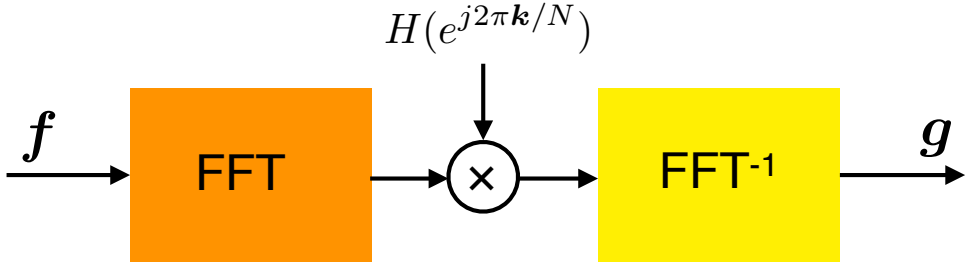
```
function p=getpixel(f,k,l);  {
   i=k; j=l;
   if (i<0) i=-i;
   if (i⩾K) i=2K-2-(i mod (2K-2));
   if (j<0) j=-j;
   if (j⩾L) j=2L-2-(j mod (2L-2));
   p=f(i,j);
}
```

- Advantages
  - Consistent: the symmetric filtering of a folded signal produces a folded signal; antisymmetric filtering yields an antisymmetric signal extension
  - No boundary artifacts

Fourier-domain filtering

Periodic convolution corresponds to a product in the Fourier domain

- Algorithm



Images of size $N \times N$

1. Take discrete 2D Fourier transform of f (FFT algorithm) $\Rightarrow$ $O(2N^2 \log_2 N)$ operations
2. Multiply with the transfer function of the filter $\Rightarrow$ $O(N^2)$ operations
3. Take inverse discrete Fourier transform (FFT algorithm) $\Rightarrow$ $O(2N^2 \log_2 N)$ operations

- Interpretation

The discrete Fourier transform $\mathbf{F}$ diagonalizes the periodic convolution matrix $\mathbf{T}$

$$\boldsymbol{g} = \mathbf{T}\boldsymbol{f}$$

$$\mathbf{F}\boldsymbol{g} = \mathbf{F}\mathbf{T}\overbrace{\mathbf{F}^{-1}\mathbf{F}}^{\mathbf{I}}\boldsymbol{f} = \mathbf{F}\mathbf{T}\mathbf{F}^{-1} \cdot \mathbf{F}\boldsymbol{f} = \mathrm{diag}(\lambda_{(0,0)}, \ldots, \lambda_{(N-1,N-1)}) \cdot \mathbf{F}\boldsymbol{f}$$

with $\lambda_{\boldsymbol{k}} = H(e^{j2\pi \boldsymbol{k}/N})$ (Transfer function)

Fourier versus space-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 (typ. 3×3): they are implemented most efficiently in the space domain
- Some classes of large filters can also be implemented efficiently in the space domain using recursive and/or multiscale algorithms
- Boundary conditions are handled best in the spatial domain
- Spatial implementation gives much more flexibility: adaptive algorithms, non-linear filtering, etc. . .

$\Rightarrow$ Image processing is mostly performed in the space domain

3.4 Useful filters for image processing

- Smoothing: the universal tool
- Moving average
- Symmetric exponential filter
- Gaussian filter

Smoothing: the universal tool

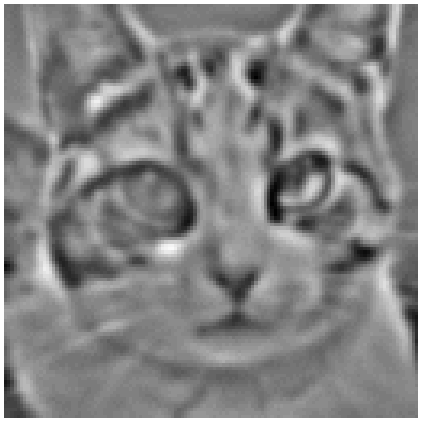
- Spatial smoothing
  - Simulates sampling aperture
  - Adjustable resolution
  - Flexibility

Original = Identity

Smoothing = *Lowpass* filtering

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

Highpass = Identity - *Lowpass*



Bandpass = $Lowpass_1$ - $Lowpass_2$

Smoothing (Cont'd)

- Desirable features
  - Computational efficiency (fast)
  - Simplicity
  - Adjustable size
  - Symmetry
    (sensitivity of HVS to phase distortion)

On the other hand:

- Shape of frequency response is not important
- Best to avoid sharp frequency cut-offs
  (Gibbs oscillation)

Efficient + Adjustable size $\Rightarrow$ SEPARABLE + RECURSIVE implementation

- Smoothing-filter requirements (1D)

Positivity: $h[k] \geqslant 0$

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

Symmetry: $\Rightarrow \quad \sum_{k \in \mathbb{Z}} k \cdot h[k] = 0$ (centered on the origin)

Equivalent window size: $\sigma_{\text{eq}}^2 = \sum_{k \in \mathbb{Z}} k^2 h[k]$

Moving average filter

- Centered $L_1 \times L_2$ moving average

$$y[k,l] = \frac{1}{L_1 L_2} \sum_{m=-\lfloor L_1/2 \rfloor}^{\lfloor L_1/2 \rfloor} \sum_{n=-\lfloor L_2/2 \rfloor}^{\lfloor L_2/2 \rfloor} x[k-m, l-n]$$

L_1, L_2: horizontal and vertical window sizes (must be odd)

- Example: 3×3 moving average

$$\boldsymbol{h} = \frac{1}{9} \begin{bmatrix} 1 & 1 & 1 \\ 1 & \boxed{1} & 1 \\ 1 & 1 & 1 \end{bmatrix} = \frac{1}{3} \begin{bmatrix} 1 & \boxed{1} & 1 \end{bmatrix} \otimes \frac{1}{3} \begin{bmatrix} 1 \\ \boxed{1} \\ 1 \end{bmatrix}$$

- Separable transfer function:

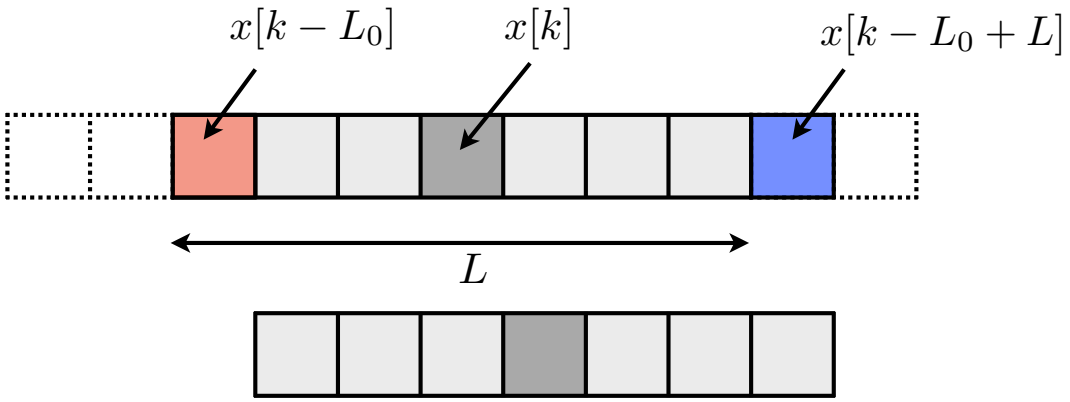
$$H(z_1, z_2) = H_{L_1}(z_1) \cdot H_{L_2}(z_2)$$

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 $L_0 = \lfloor L/2 \rfloor$

$\Rightarrow$ successive filtering along the rows and columns!

Moving average: implementation

- Recursive implementation in 1D



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

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

$$y[k+1] = y[k] + \frac{1}{L}\left(x[k-L_0+L] - x[k-L_0]\right) \qquad \text{with} \qquad L_0 = \lfloor L/2 \rfloor$$

$\Rightarrow$ 2 adds and 1 mult per sample irrespective of L!

z-transform: $zY(z) = Y(z) + \frac{1}{L}X(z)\left(z^{L-L_0} - z^{-L_0}\right)$

$$\Rightarrow \quad H_L(z) = \frac{Y(z)}{X(z)} = \frac{z^{-L_0}}{L}\left(\frac{z^L - 1}{z-1}\right)$$

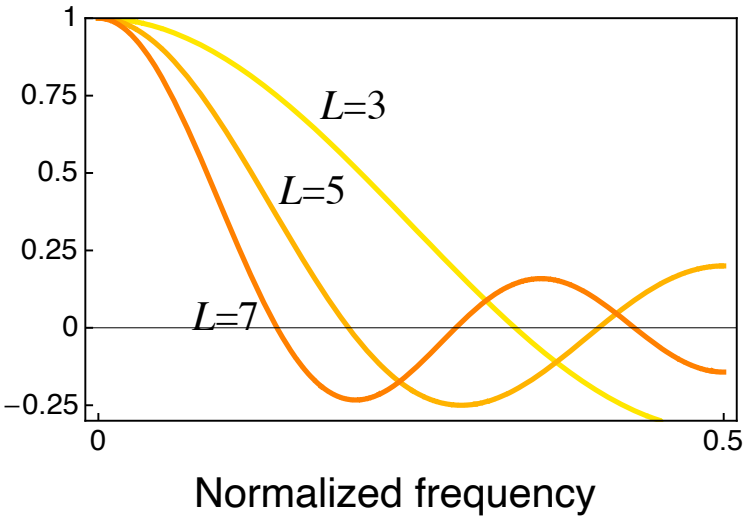
Moving average: transfer function

- z-transform

$$H_L(z) = \frac{z^{-L_0}}{L} \sum_{k=0}^{L-1} z^k = \frac{z^{-L_0}}{L} \left(\frac{z^L - 1}{z - 1} \right)$$

- Fourier transform: $L = 2L_0 + 1$ (odd)

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

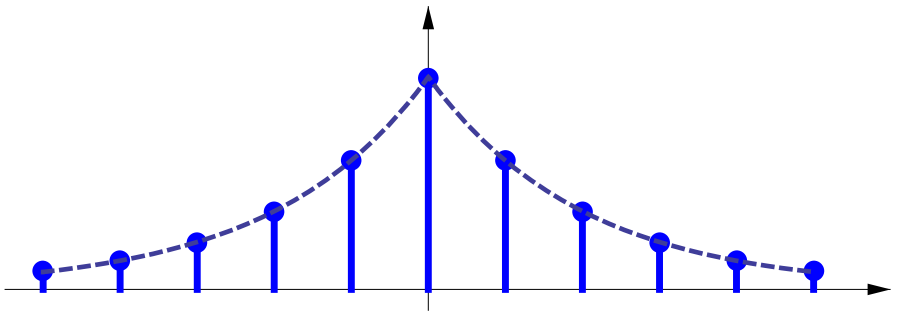


Symmetric exponential filter

- Impulse response

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

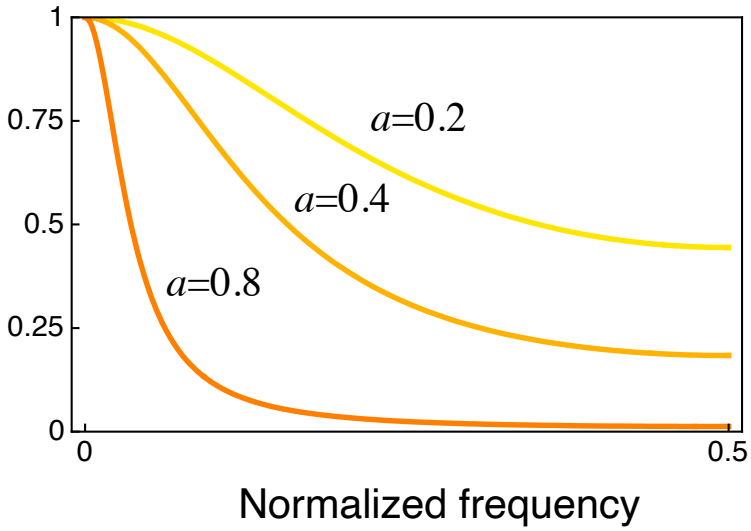
C such that $\sum_{(k,l)\in\mathbb{Z}^2} h[k,l] = 1$



- Separable transfer function

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

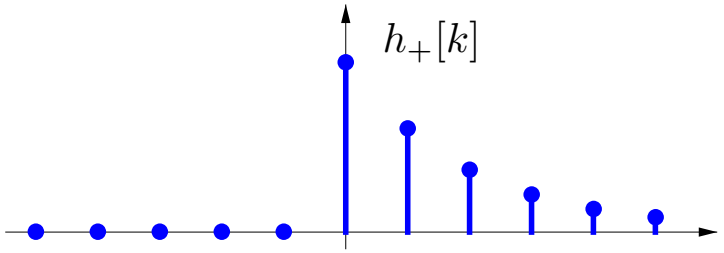
$\Rightarrow$ successive filtering along the rows and columns!



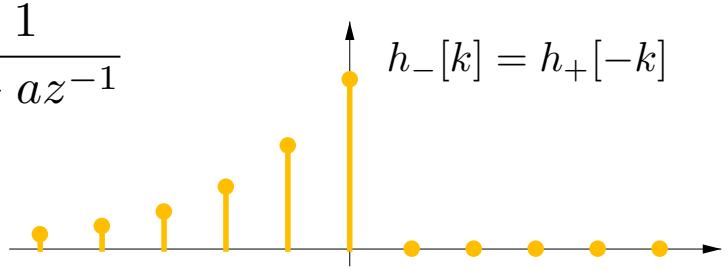
Symmetric exponential (1D)

- Construction of a symmetric exponential

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



$$h_+[k] = \begin{cases} a^k, & k \geqslant 0 \\ 0, & \text{otherwise} \end{cases} \quad \overset{z}{\longleftrightarrow} \quad H_+(z) = \frac{1}{1 - az^{-1}}$$



- 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

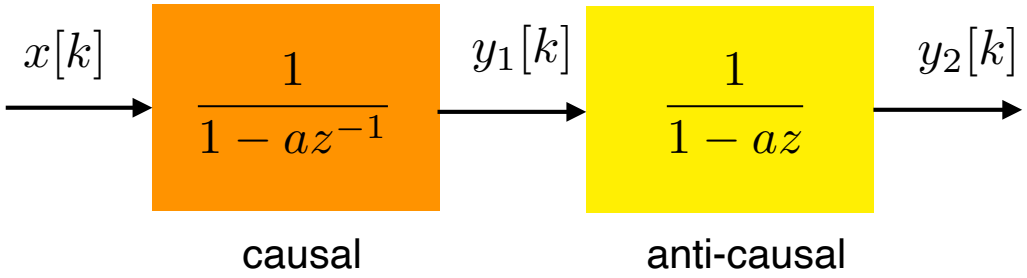
$$H_a(z) = \frac{C_a}{(1 - az^{-1})(1 - az)} \quad \text{such that} \quad \sum_{k \in \mathbb{Z}} h_a[k] = H_a(1) = 1 \quad \Rightarrow \quad C_a = (1 - a)^2$$

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

Exponential filtering: implementation

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

  ➡ Cascade of first-order recursive filters



$$Y_1(z) = \frac{X(z)}{1 - az^{-1}} \quad \Rightarrow \quad Y_1(z) = X(z) + az^{-1}Y_1(z)$$

- Recursive-filtering algorithm

  1. Causal filtering: $y_1[k] = x[k] + ay_1[k-1]$, for $(k = 0, \ldots, N-1)$
  2. Anti-causal filtering: $y_2[k] = y_1[k] + ay_2[k+1]$, for $(k = N-1, \ldots, 0)$
  3. Normalization: $y[k] = C_a \cdot y_2[k]$

Gaussian filter

- 2D Gaussian impulse response

$$h_\sigma[k,l] = \frac{1}{2\pi\sigma^2}\exp\left(-\frac{(k^2+l^2)}{2\sigma^2}\right) = \mathrm{gauss}(k;\sigma)\times\mathrm{gauss}(l;\sigma)$$

where $\mathrm{gauss}(x;\sigma) = \frac{1}{\sigma\sqrt{2\pi}}\exp\left(-\frac{x^2}{2\sigma^2}\right)$

- Motivation
  - Only filter that is both circular-symmetric and separable

    $\Rightarrow$ successive filtering along the rows and columns!
  - Optimal space-frequency localization
  - Linear scale space

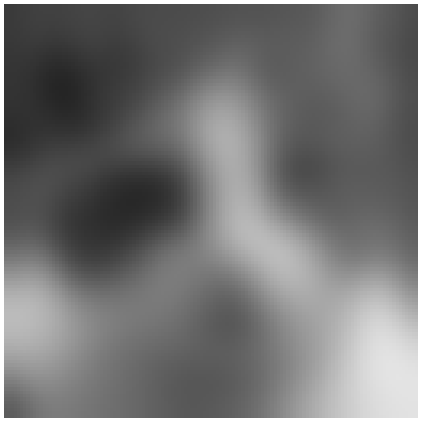
Linear scale-space & the melting cat

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

$\sigma = 1$

$\sigma = 2$

$\sigma = 4$



$\sigma = 8$

- Heat-flow interpretation

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

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

Refresher: Central-limit theorem

- Probability density function (PDF)

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

- Moments: mean and variance

$$\mu = \mathrm{E}\{x\} = \int_{-\infty}^{+\infty} x \cdot p(x)\,\mathrm{d}x$$

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

- Sum of two independent random variables

$$\mathrm{Var}\{x_1 + x_2\} = \mathrm{Var}\{x_1\} + \mathrm{Var}\{x_2\}$$

PDF: $p_{1+2}(x) = (p_1 * p_2)(x)$

- Sum of N independent, identically-distributed random variables

$$\mathrm{Var}\left\{\sum_{i=1}^{N} x_i\right\} = \sum_{i=1}^{N} \mathrm{Var}\{x_i\} = N\sigma^2$$

PDF: N-fold convolution of $p(x)$

Central-limit theorem: $p_{\mathrm{sum}}(x) = \underbrace{(p * p * \cdots p)}_{N \text{ times}}(x) \quad \rightarrow \quad \frac{1}{\sqrt{2\pi N\sigma^2}} \exp\left(\frac{-(x-N\mu)^2}{2N\sigma^2}\right)$

Efficient Gaussian filtering

via a judicious use of elementary operators...

- Convolution interpretation of the Central-limit theorem

  "The N-fold iteration of any lowpass filter converges to a Gaussian"

- Gaussian filtering by repeated moving average of size $2L_0 + 1$ (odd)

$$\sigma^2_{\text{rect}} = \frac{1}{2L_0+1} \sum_{k=-L_0}^{L_0} k^2 = \frac{L_0 + L_0^2}{3} \qquad \left(= \sum_{k\in\mathbb{Z}} k^2 h[k] \approx \int_{-\infty}^{\infty} x^2 p(x)\,\mathrm{d}x \right)$$

N iterations $\quad \Rightarrow \quad \sigma^2_{\text{eq}} = N\sigma^2_{\text{rect}}$

- Gaussian filtering by repeated exponential filtering

$$\sigma^2_{\text{exp}} = \frac{2a}{(1-a)^2} \qquad \left(= \sum_{k\in\mathbb{Z}} k^2 h_a[k] = \left.\frac{\mathrm{d}^2 H_a(z)}{\mathrm{d}z^2}\right|_{z=1} \right)$$

N iterations $\quad \Rightarrow \quad \sigma^2_{\text{eq}} = \dfrac{2Na}{(1-a)^2}$

Determination of exponential parameter for a desired σ and N

$$\sigma^2 = \frac{2Na}{(1-a)^2} \qquad \Rightarrow \qquad a = 1 + \frac{N}{\sigma^2} - \frac{\sqrt{N^2 + 2N\sigma^2}}{\sigma^2}$$

Side note

- Identity to retrieve equivalent window size using z-transform

  We have $H_a(z) = \sum_k h_a[k] z^{-k}$

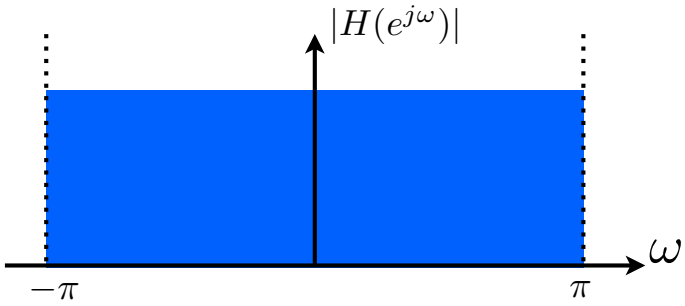
  so also $\frac{dH_a(z)}{dz} = \sum_k h_a[k](-k) z^{-k-1}$

  and $\frac{d^2 H_a(z)}{dz^2} = \sum_k h_a[k](k+1) k z^{-k-2}$

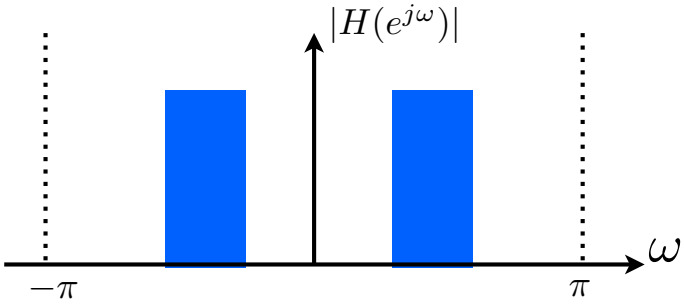
  Thus, for $z = 1$ we obtain $\sum_k h_a[k] k^2 + \underbrace{\sum_k h_a[k] k}_{=0 \text{ (centered)}}$

Nomenclature of prototypical filters

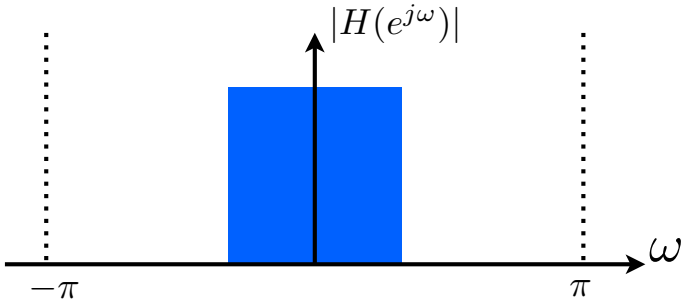
all-pass



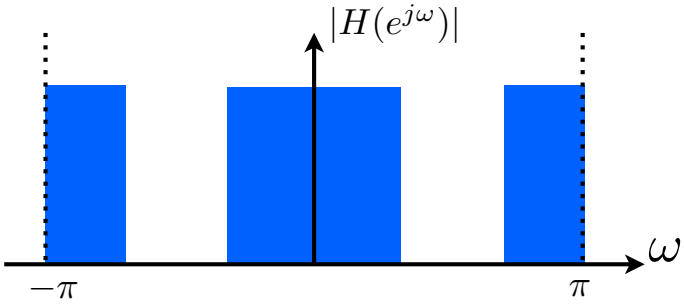
band-pass



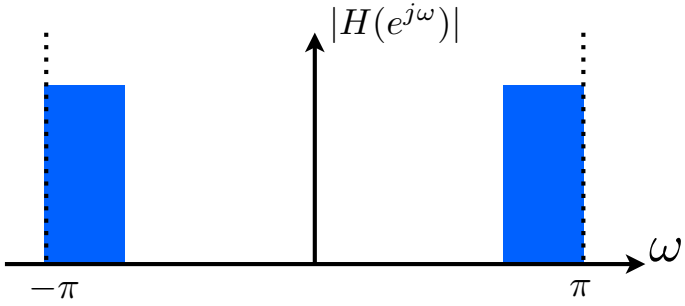
low-pass



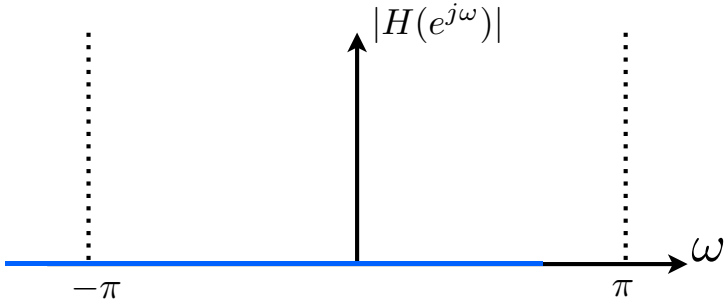
band-cut



high-pass



all-cut



3.5 SUMMARY

- Discrete images are sequences indexed by two (or more) spatial integer variables. When they have finite energy, they can be viewed as points in the Hilbert space $\ell_2(\mathbb{Z}^d)$.
- A discrete image is characterized by its 2D Fourier transform which is 2π-periodic.
- The 2D z-transform provides an often more convenient characterization. It is a direct vector generalization of the 1D transform. Thus, it has essentially the same properties.
- Digital filtering can be described as a local masking operation (running inner-product), or as a discrete convolution. A 2D digital 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 exponential, and the Gaussian filter. They can all be implemented recursively with a complexity independent of the window size.