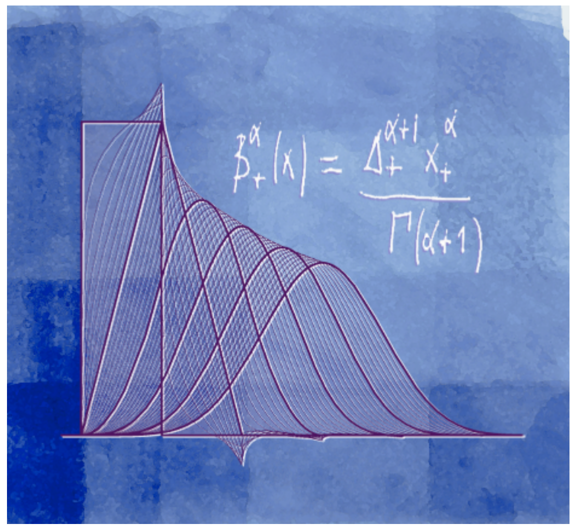
EPFL

# Image Processing

## Chapter 7

## Continuous/discrete image processing

Prof. Michael Unser, LIB
*Prof. Dimitri Van De Ville, MIP*
Dr. Daniel Sage, LIB




March 2025

# CONTENT

- **7.1 Continuous/discrete representation of images**
  - Classical image interpolation
  - Generalized image interpolation
  - Interpolation: filtering solution
- **7.2 Polynomial splines**
  - Splines: definition
  - B-spline basis functions
  - Interpolating splines
- **7.3 From splines to wavelets**
  - Haar transform revisited
  - Image pyramids
  - From pyramids to wavelets

# INTRODUCTION

*At the beginning there was a continuum.*
*Man made it discrete* !

| Continuous domain: $L_2(\mathbb{R}^d)$ | $\longleftrightarrow$ | Discrete domain: $\ell_2(\mathbb{Z}^d)$ |
|---|---|---|
| $f(\boldsymbol{x}),\ \boldsymbol{x} \in \mathbb{R}^d$ | | $f[\boldsymbol{k}],\ \boldsymbol{k} \in \mathbb{Z}^d$ |

- real-world objects
- images
- sensor input

- measurements
- algorithms
- image processing

- Sampling and image acquisition

- Continuous/discrete algorithm design

  Finding discrete solutions for problems formulated in the continuous domain

  - Interpolation
  - Spatial transformations, warping
  - Feature detection (edges)
  - Image registration
  - Tomography, etc. . .

- Multiresolution approaches

  - Coarse-to-fine and multigrid algorithms
  - Image pyramids and wavelets

# 7.1 CONTINUOUS/DISCRETE REPRESENTATIONS

- Classical image interpolation
- Generalized interpolation
- Interpolation: filtering solution
- Interpolating basis functions
- Spatial transformations

# Classical image interpolation

Discrete image data $\Longleftrightarrow$ Continuous image model

$f[\boldsymbol{k}],\ \boldsymbol{k} = (k_1, \ldots, k_d) \in \mathbb{Z}^d$ $\qquad$ $f(\boldsymbol{x}),\ \boldsymbol{x} = (x_1, \ldots, x_d) \in \mathbb{R}^d$

- Interpolation formula: $$f(\boldsymbol{x}) = \sum_{\boldsymbol{k} \in \mathbb{Z}^d} f[\boldsymbol{k}] \varphi_{\text{int}}(\boldsymbol{x} - \boldsymbol{k})$$

  $f[\boldsymbol{k}]$: pixel values at location $\boldsymbol{k}$

  $\varphi_{\text{int}}(\boldsymbol{x})$: continuous-space interpolation function

  $\varphi_{\text{int}}(\boldsymbol{x} - \boldsymbol{k})$: interpolation function translated to location $\boldsymbol{k}$

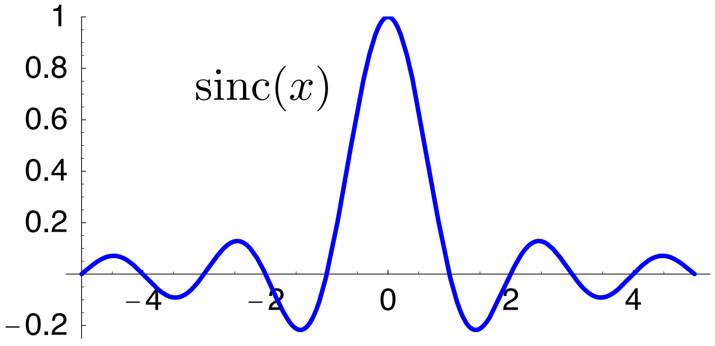
- Interpolation condition

  At the grid points $\boldsymbol{x} = \boldsymbol{k}_0$: $$f[\boldsymbol{k}_0] = \sum_{\boldsymbol{k} \in \mathbb{Z}^p} f[\boldsymbol{k}] \varphi_{\text{int}}(\boldsymbol{k}_0 - \boldsymbol{k})$$

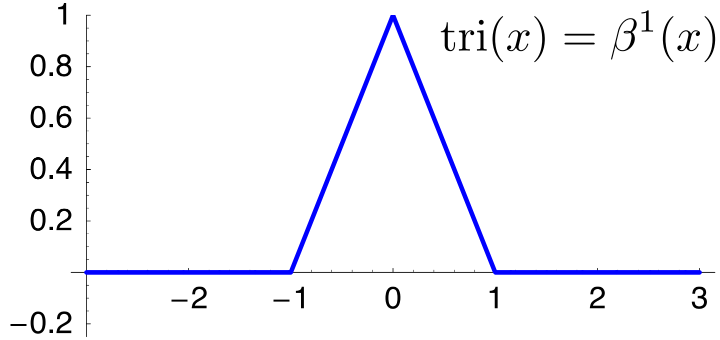
  Only possible $\forall f$ iff. $$\varphi_{\text{int}}(\boldsymbol{k}) = \begin{cases} 1, & \boldsymbol{k} = \boldsymbol{0} \\ 0, & \text{otherwise} \end{cases}$$

# Examples of popular interpolation functions

Interpolation condition: $\varphi_{\text{int}}(x)\big|_{x=k} = \delta[k]$




$$\operatorname{sinc}(x) = \frac{\sin(\pi x)}{\pi x}$$

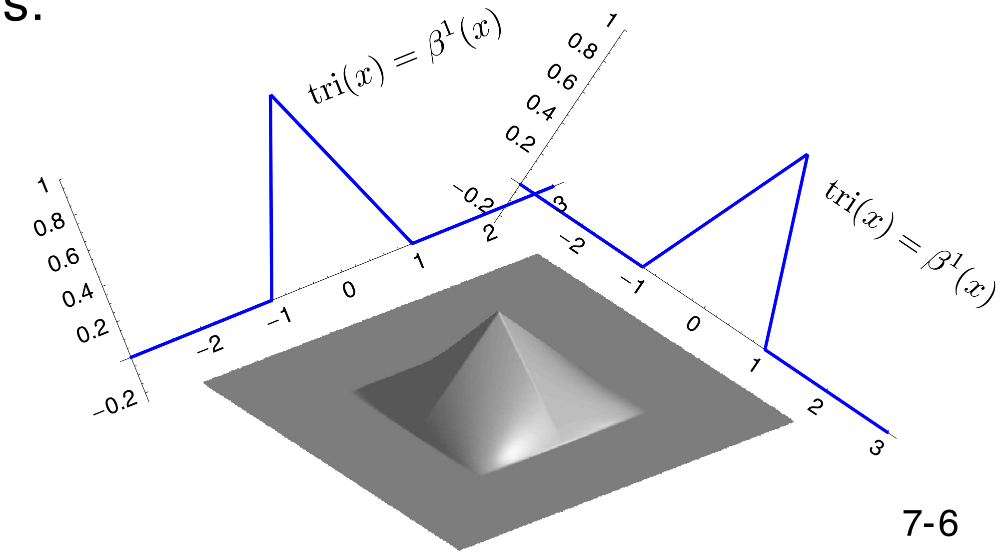



$$\operatorname{tri}(x) = \beta^1(x)$$

$$= \begin{cases} 1 - |x| & \text{if } |x| \leq 1 \\ 0 & \text{otherwise} \end{cases}$$

Separable extension to higher dimensions:

$$\varphi_{\text{int}}(\boldsymbol{x}) = \prod_{i=1}^{d} \varphi_{\text{int}}(x_i)$$



# Generalized image interpolation

- Desired features for the interpolation kernel
  - Short (to minimize computations)
  - Simple expression (e.g., polynomial)
  - Smooth (to avoid model discontinuities)
  - Good approximation properties: reproduction of polynomials

- Generalized interpolation formula:

$$f(\boldsymbol{x}) = \sum_{\boldsymbol{k} \in \mathbb{Z}^d} c[\boldsymbol{k}]\, \varphi(\boldsymbol{x} - \boldsymbol{k})$$

  - Simple integer-shift-invariant structure
  - Simple expression (e.g., polynomial)
  - $\varphi$ selected freely (not interpolating and much shorter)

$\Rightarrow$ Faster interpolation formulas!

... but one new difficulty: How to pre-compute the coefficients $c[\boldsymbol{k}]$ ?

- Separable basis functions: $\varphi(\boldsymbol{x}) = \varphi(x_1) \cdot \varphi(x_2) \cdots \varphi(x_d)$

$\Rightarrow$ Further acceleration

# Interpolation: filtering solution

Interpolation problem: Given the samples $\{f[\boldsymbol{k}]\}$, find the coefficients $\{c[\boldsymbol{k}]\}$

- Interpolation condition: $f(\boldsymbol{x})|_{\boldsymbol{x}=\boldsymbol{k}} = f[\boldsymbol{k}] = \sum_{\boldsymbol{k}_1 \in \mathbb{Z}^d} c[\boldsymbol{k}_1]\varphi(\boldsymbol{k} - \boldsymbol{k}_1)$

  Discrete convolution equation: $f[\boldsymbol{k}] = (b * c)[\boldsymbol{k}]$

  with $b[\boldsymbol{k}] \triangleq \varphi(\boldsymbol{k}) \quad \stackrel{z}{\longleftrightarrow} \quad B(\mathbf{z}) = \sum_{\boldsymbol{k} \in \mathbb{Z}^d} b[\boldsymbol{k}]\mathbf{z}^{-\boldsymbol{k}}$

- Inverse-filtering solution

$f[\boldsymbol{k}]$

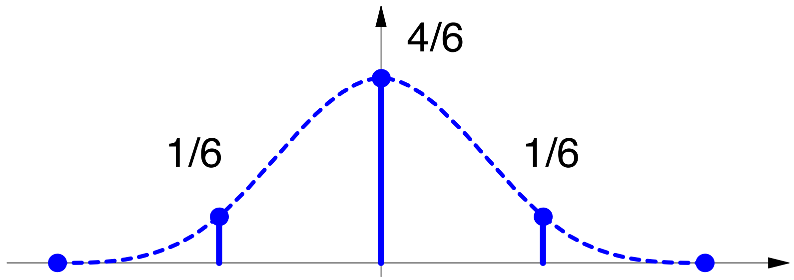

$c[\boldsymbol{k}] = (h_{\mathrm{int}} * f)[\boldsymbol{k}]$ with $H_{\mathrm{int}}(\mathbf{z}) = \dfrac{1}{B(\mathbf{z})} = \dfrac{1}{\sum_{\boldsymbol{k} \in \mathbb{Z}^d} \varphi(\boldsymbol{k})\mathbf{z}^{-\boldsymbol{k}}}$

Note: $\varphi(\boldsymbol{x})$ separable $\quad \Rightarrow \quad h_{\mathrm{int}}[\boldsymbol{k}]$ separable

# Example: cubic-spline interpolation

- Cubic B-spline

$$\varphi(x) = \beta^3(x) = \begin{cases} \frac{2}{3} - \frac{1}{2}|x|^2(2-|x|), & 0 \le |x| < 1 \\ \frac{1}{6}(2-|x|)^3, & 1 \le |x| < 2 \\ 0, & \text{otherwise} \end{cases}$$




- Discrete B-spline kernel: $B(z) = \dfrac{z+4+z^{-1}}{6}$

- Interpolation filter

$$\frac{6}{z+4+z^{-1}} = \frac{(1-\alpha)^2}{(1-\alpha z)(1-\alpha z^{-1})} \quad \overset{z}{\longleftrightarrow} \quad h_{\text{int}}[k] = \left(\frac{1-\alpha}{1+\alpha}\right)\alpha^{|k|}$$

$\alpha = -2 + \sqrt{3} = -0.171573$ Symmetric exponential (cf. Chap IP-3)

- Multidimensional interpolation

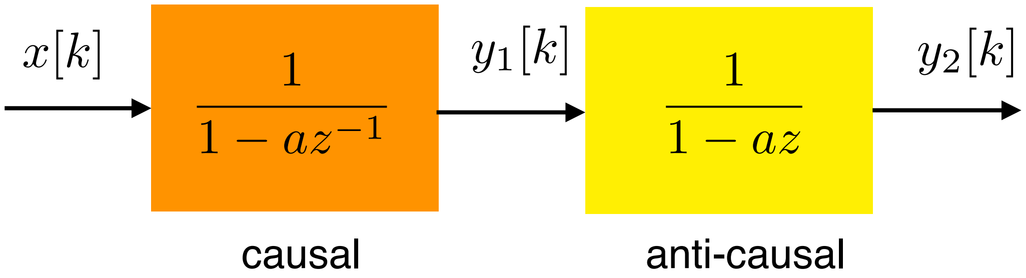
Separability $\Rightarrow$ successive 1D filtering along the dimensions of the data

# Exponential filtering: implementation

REMINDER

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

Cascade of first-order recursive filters

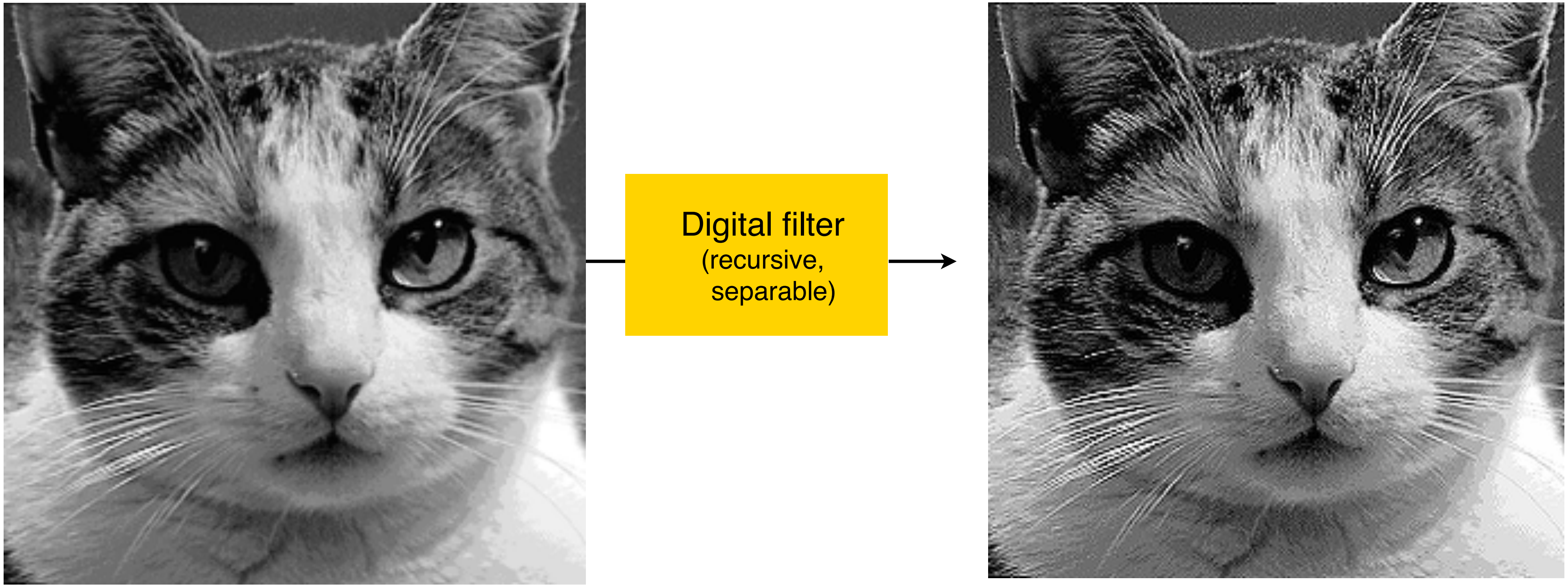



$$Y_1(z) = \frac{X(z)}{1 - az^{-1}} \quad \Rightarrow \quad Y_1(z) = X(z) + az^{-1}Y_1(z)$$

- Recursive-filtering algorithm

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

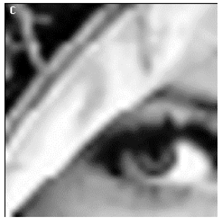
# Cubic-spline coefficients in 2D




Pixel values $f[k,l]$

B-spline coefficients $c[k,l]$

# One-to-one continuous/discrete representation



Continuously-defined signal

$$f(\boldsymbol{x}) = \sum_{\boldsymbol{k}\in\mathbb{Z}^p} c[\boldsymbol{k}]\varphi(\boldsymbol{x}-\boldsymbol{k})$$

Riesz-basis property

Expansion coefficients

$c[\boldsymbol{k}]$

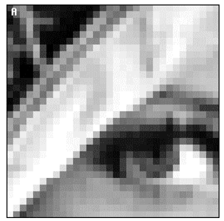
Digital filtering

$* \, b$ (FIR)

$* \, h_{\mathrm{int}}$ (IIR)

Sampling: $f(\boldsymbol{x})|_{\boldsymbol{x}=\boldsymbol{k}}$

$f[\boldsymbol{k}]$



Discrete signal

# Interpolating basis function

- Equivalent interpretation of generalized interpolation

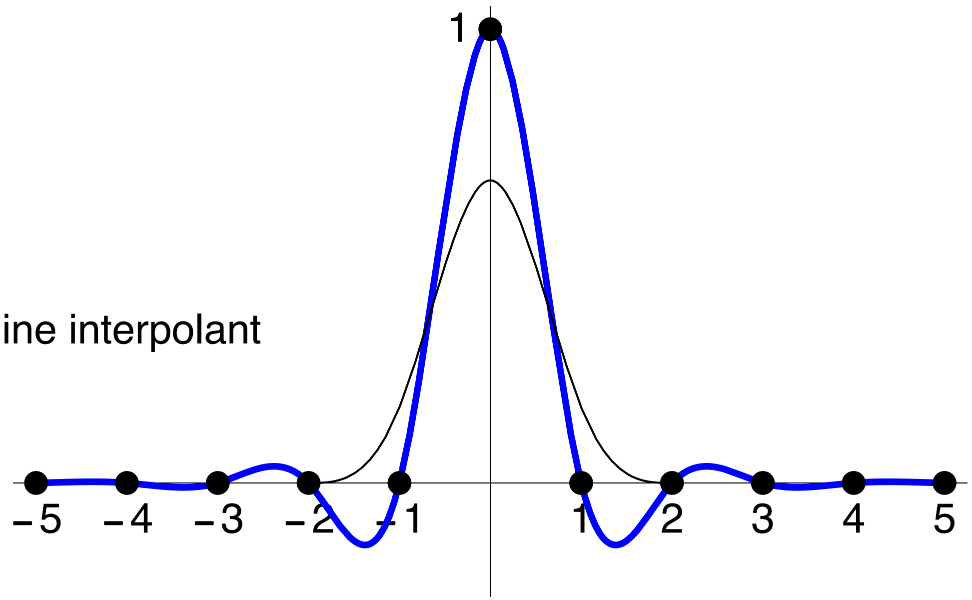
$$f(x) = \sum_{k\in\mathbb{Z}} c[k]\varphi(x-k) = \sum_{k\in\mathbb{Z}} \left((f * h_{\mathrm{int}})[k]\right)\varphi(x-k)$$

$$= \sum_{k\in\mathbb{Z}} f[k]\,\varphi_{\mathrm{int}}(x-k)$$

- Interpolation basis function

$$\varphi_{\mathrm{int}}(x) = \sum_{k\in\mathbb{Z}} h_{\mathrm{int}}[k]\,\varphi(x-k)$$

Example: cubic-spline interpolant




Finite-cost implementation of an infinite impulse response interpolator!

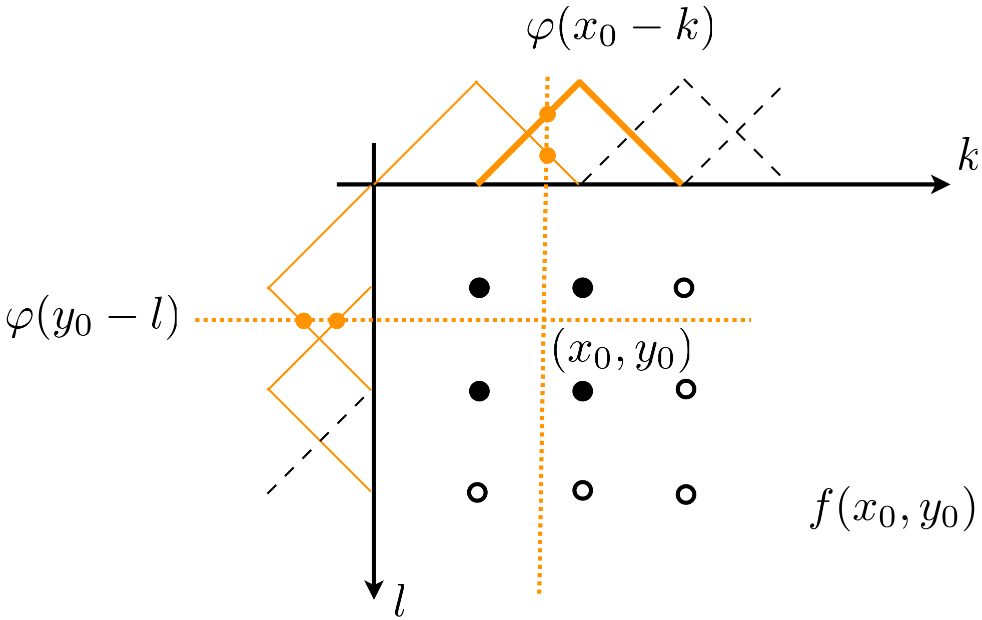
# Spatial transformation

- One-to-one coordinate mapping $T : \mathbb{R}^d \to \mathbb{R}^d$

$$\boldsymbol{x} = (x_1, \ldots, x_d) \quad \to \quad \boldsymbol{\xi} = T(\boldsymbol{x})$$
$$\boldsymbol{\xi} = (\xi_1, \ldots, \xi_d) \quad \to \quad \boldsymbol{x} = T^{-1}(\boldsymbol{\xi})$$

(*e.g.*, affine transformation $= \boldsymbol{A}\boldsymbol{x} + \boldsymbol{b}$)

- Image transformation by re-sampling: $f_T(\boldsymbol{\xi}_0) = f\Big(\underbrace{T^{-1}\{\boldsymbol{\xi}_0\}}_{\boldsymbol{x}_0}\Big) = \sum_{\boldsymbol{k}\in\mathbb{Z}^d} c[\boldsymbol{k}]\varphi\big(\boldsymbol{x}_0 - \boldsymbol{k}\big)$

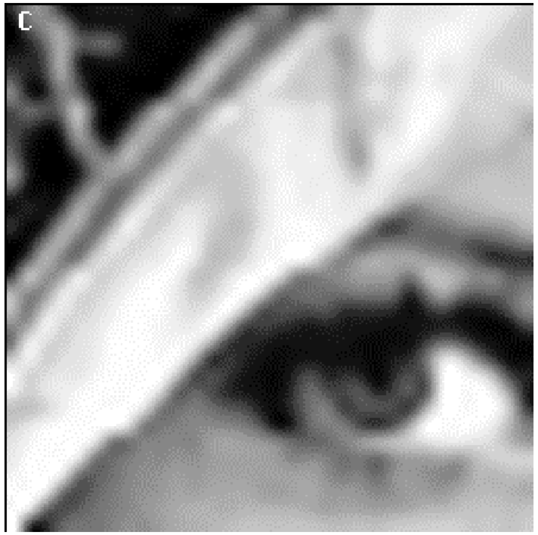



$$f(x_0, y_0) = \sum_{k=k_0}^{k_1} \sum_{l=l_0}^{l_1} c[k,l]\varphi(x_0 - k)\varphi(y_0 - l)$$

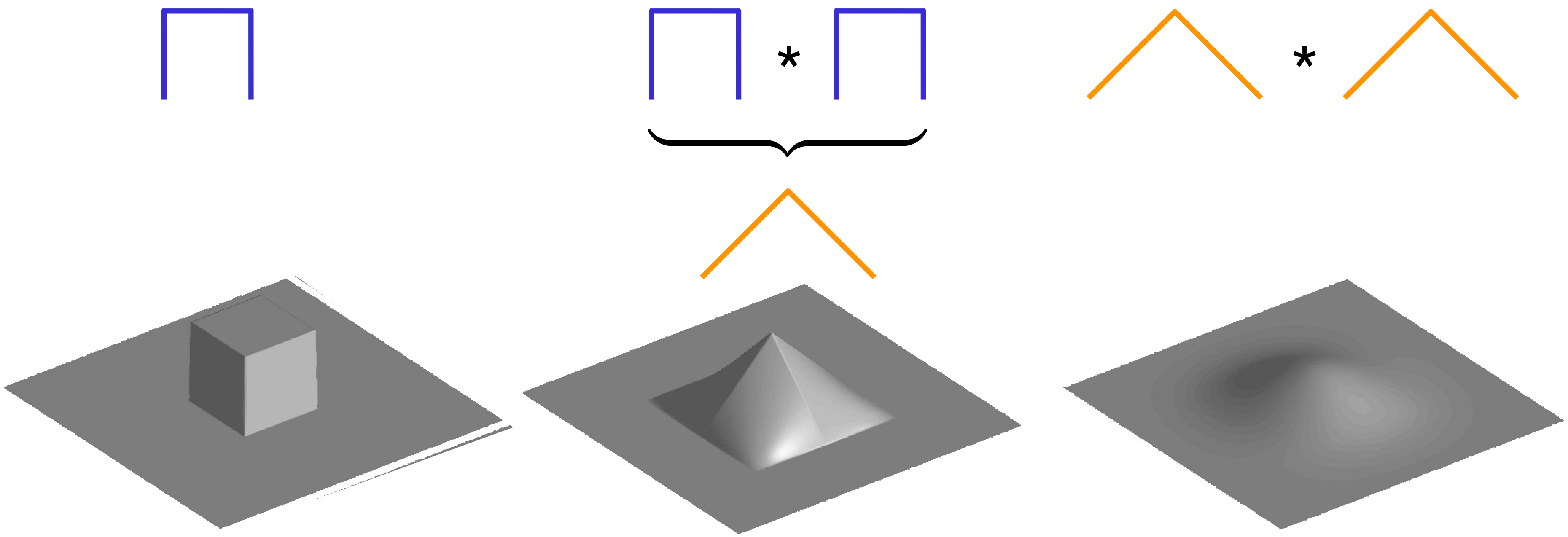
# Image zooming


Piecewise constant


Bilinear




Cubic spline



# Interpolation benchmark

Cumulative rotation experiment: the best algorithm wins!

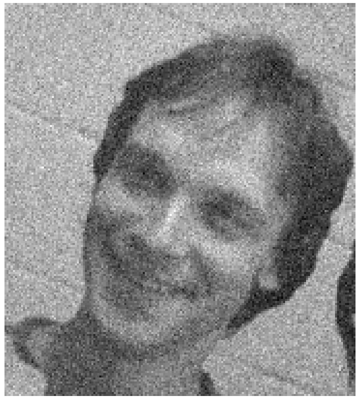
Bilinear

Windowed-sinc

Cubic spline

# Image-registration problem

Source: $f_{\mathrm{S}}(\boldsymbol{x})$



Target: $f_{\mathrm{T}}(\boldsymbol{x})$

Find a spatial transformation: $\boldsymbol{x} \rightarrow \mathbf{g}(\boldsymbol{x})$ such that $f_{\mathrm{S}}(\mathbf{g}(\boldsymbol{x})) \approx f_{\mathrm{T}}(\boldsymbol{x})$

- Basic ingredients of a registration algorithm
  - A metric for comparing images
  - A class of admissible transformations
  - A resampling/interpolation mechanism
  - An optimization procedure

**Splines**

# High-quality rigid-body registration

- Registration as a non-linear least-squares estimation problem

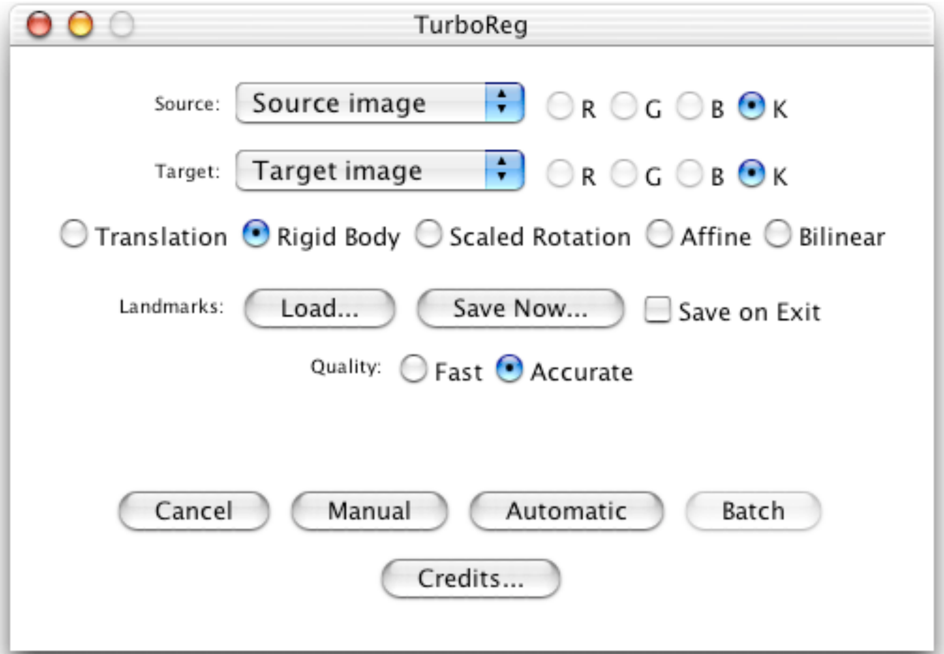
$$\min_{\mathbf{A},\boldsymbol{x}_0} \left\{ \sum_{\boldsymbol{k}} \left| f_{\mathrm{S}}\left(\mathbf{A}(\boldsymbol{k}-\boldsymbol{x}_0)\right) - f_{\mathrm{T}}[\boldsymbol{k}] \right|^2 \right\}$$

$f_{\mathrm{S}}(\boldsymbol{x})$: source image (continuously-defined)

$f_{\mathrm{T}}[\boldsymbol{k}]$: target image (discrete)

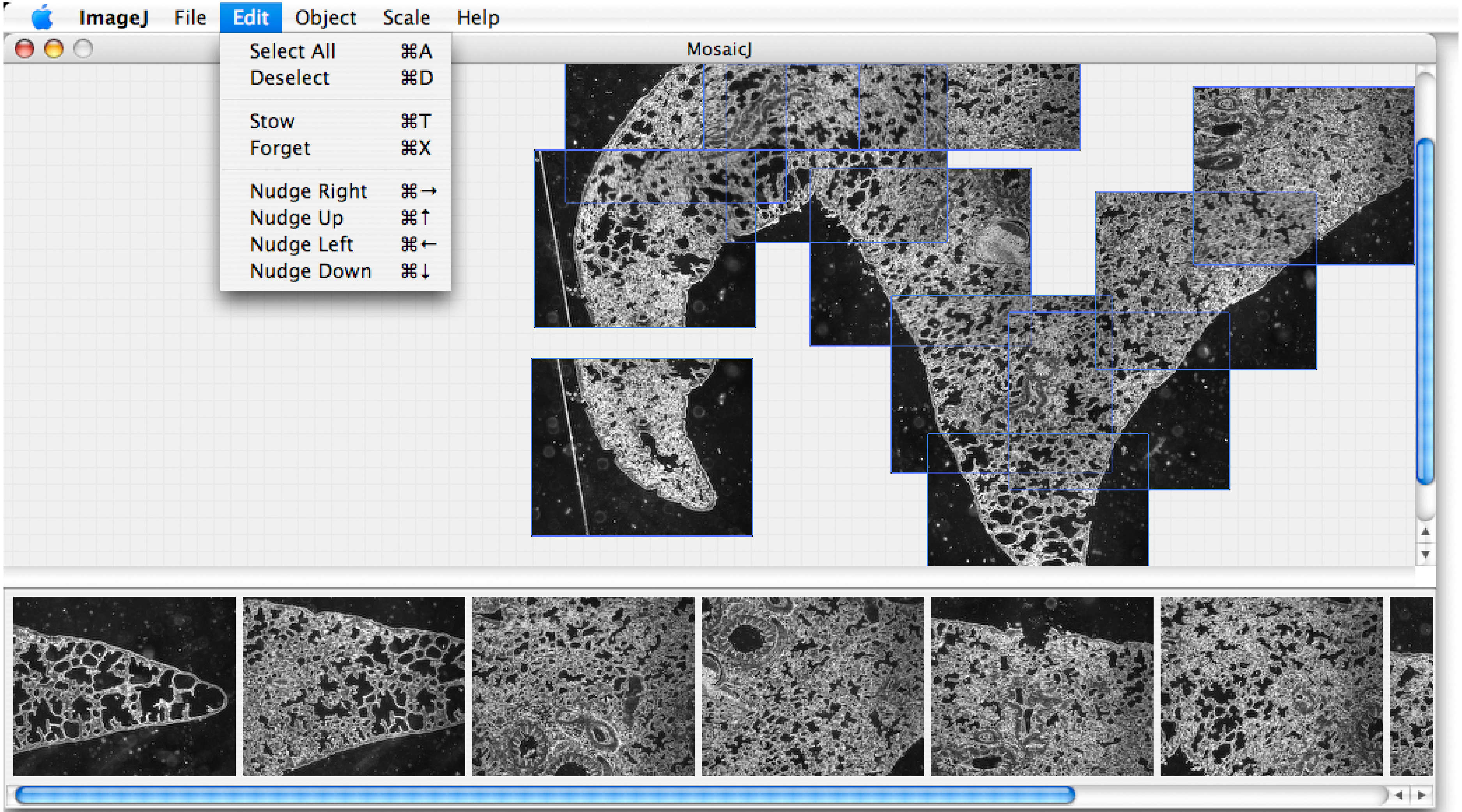
- Proposed spline-based algorithm
  - Cubic-spline interpolation of $f_{\mathrm{S}}(\boldsymbol{x})$
  - Coarse-to-fine strategy
    - cubic-spline pyramids
  - Marquardt-Levenberg optimizer
  - Consistent implementation
    - least-squares approximation
    - exact gradient

## Plugin for ImageJ

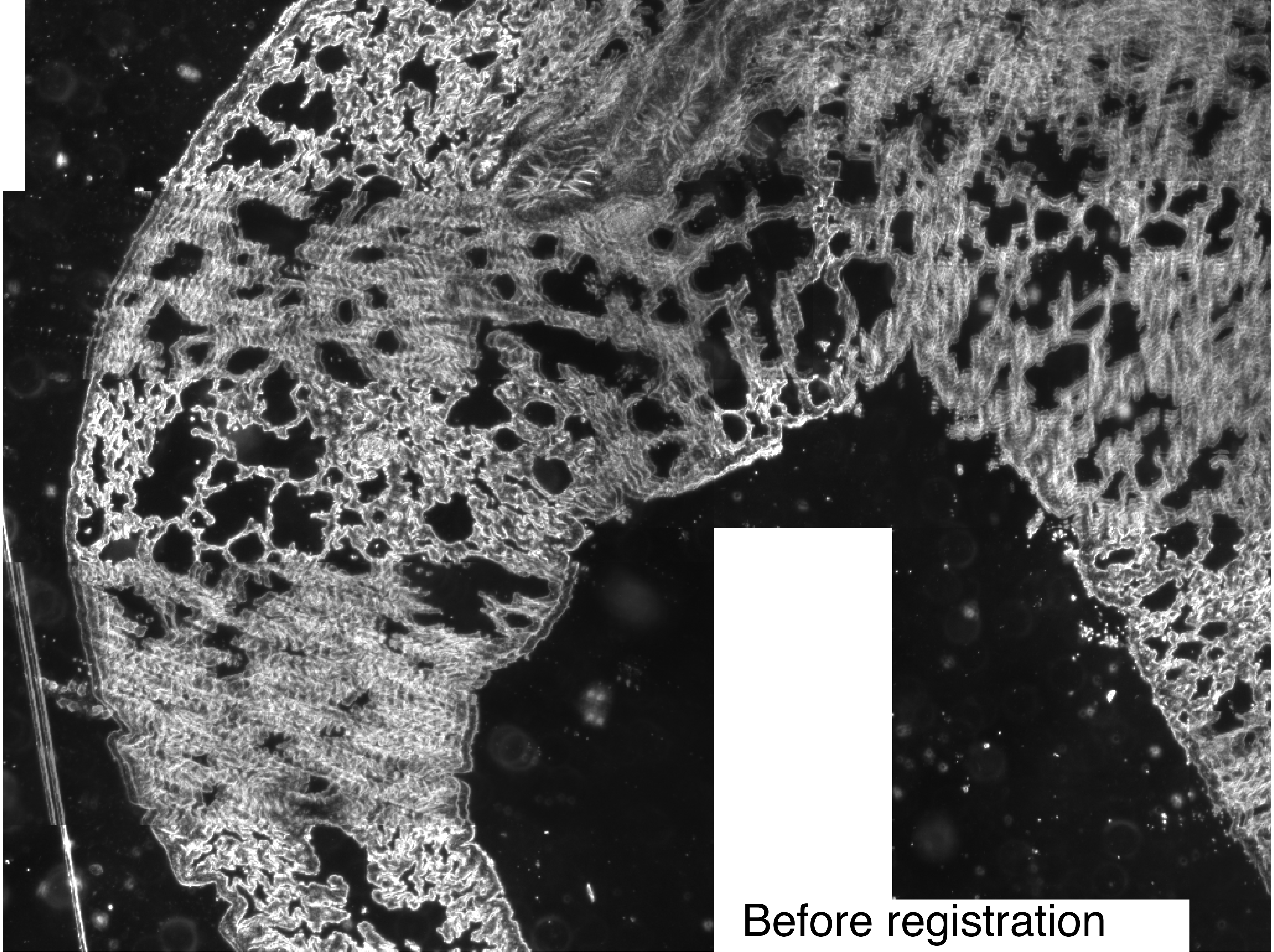


# MosaicJ: A stitching tool for microscopy

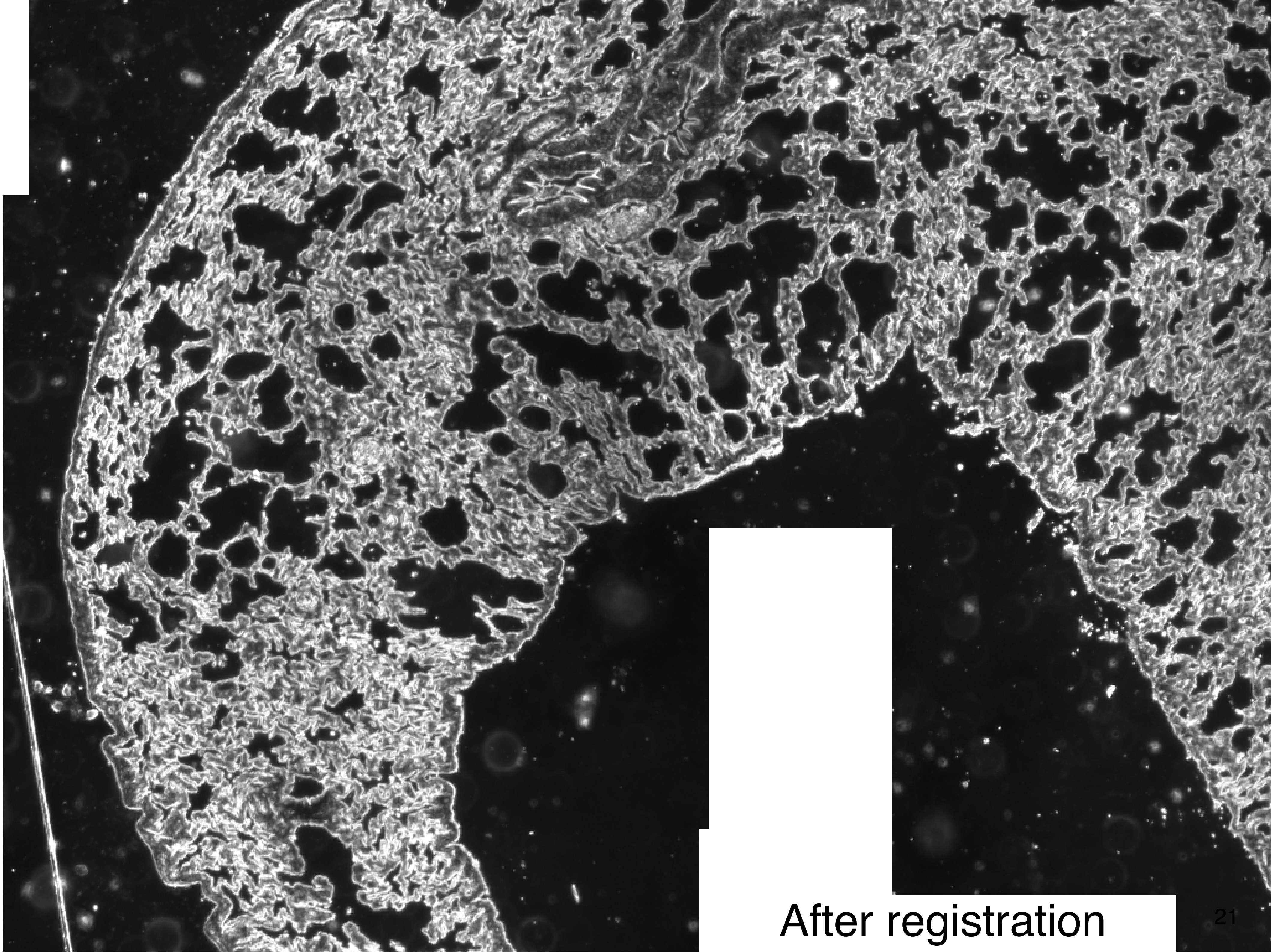
## User-friendly interface (within ImageJ)




Computational engine: TurboReg

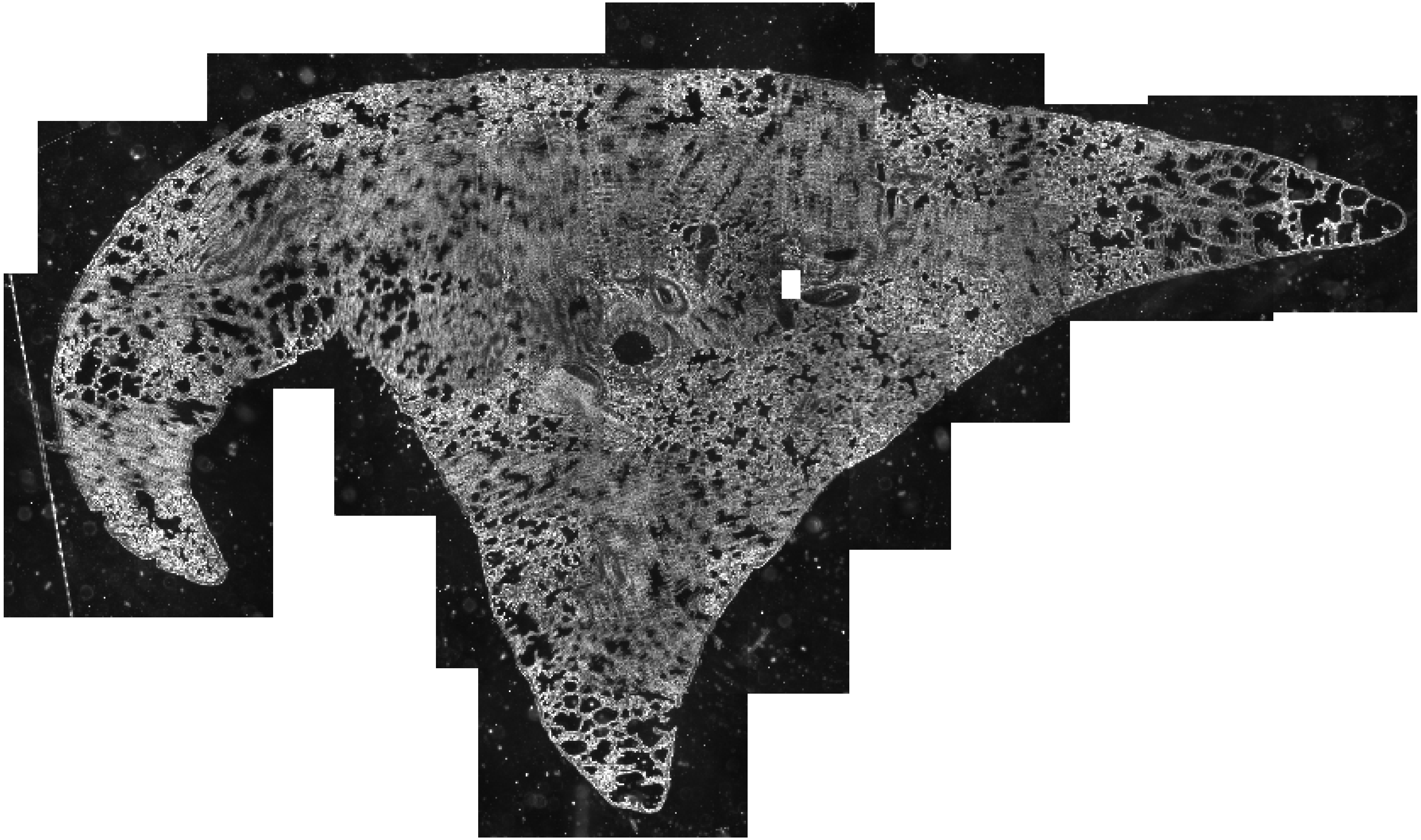


Before registration

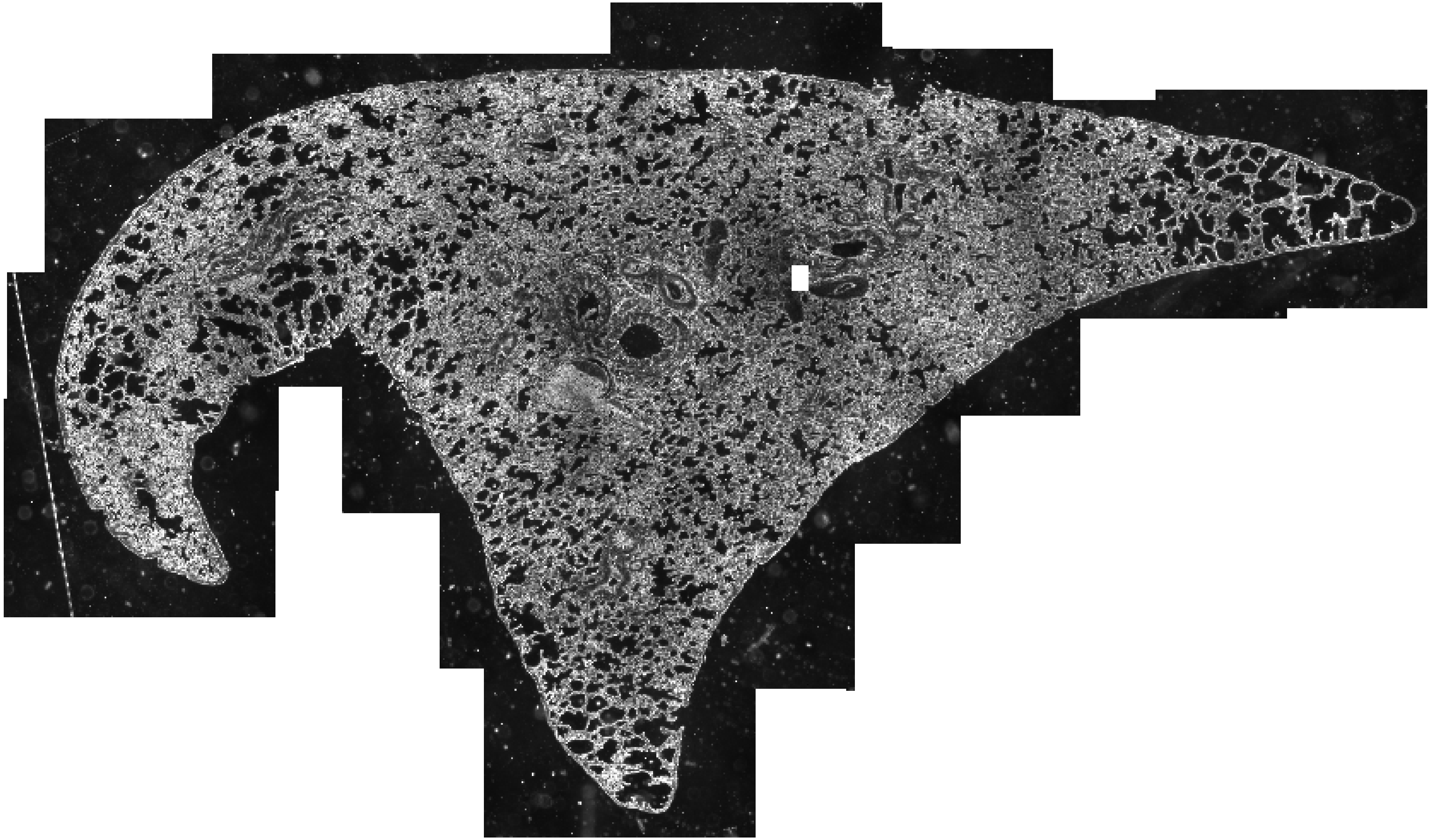


After registration

# **Before:** Twenty tiles (636 x 512)



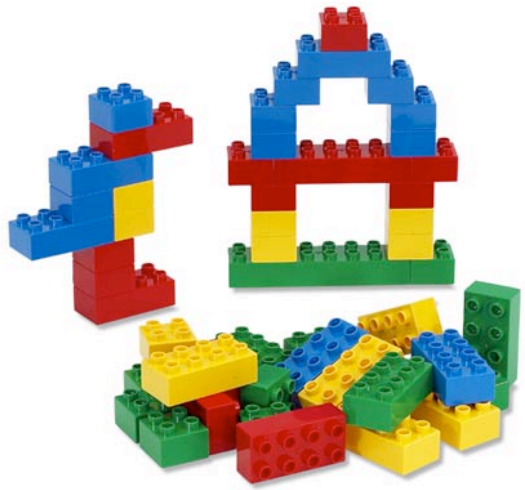
# **After:** One mosaic (3’332 x 1’957)



P. Thévenaz, M. Unser, *Microscopy Research and Technique*, 70(2), pp. 135-146, 2007

# 7.2 POLYNOMIAL SPLINES REPRESENTATIONS

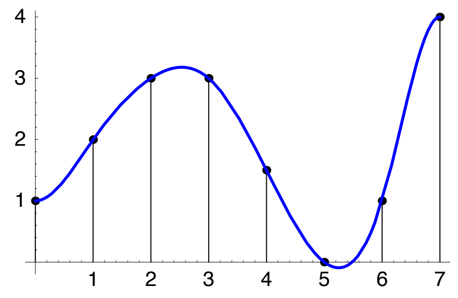
- Splines: definition
- Basic atoms: B-splines
- B-spline properties
- B-spline interpolation
- Why B-splines?



# Splines: Definition

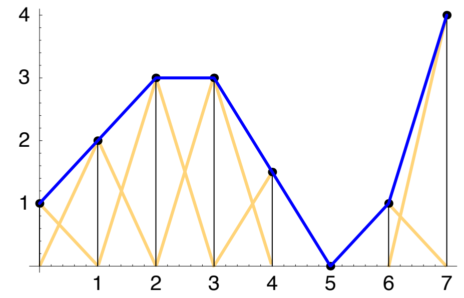
**Definition**: A function $s(x)$ is a polynomial spline of degree $n$ with knots $\cdots < x_k < x_{k+1} < \cdots$ iff. it satisfies the following two properties:

- Piecewise polynomial: $s(x)$ is a polynomial of degree $n$ within each interval $[x_k, x_{k+1})$;
- Higher-order continuity: $s(x), s^{(1)}(x), \cdots, s^{(n-1)}(x)$ are continuous at the knots $x_k$.




- Effective degrees of freedom per segment:

$$\underbrace{(n+1)}_{\text{(polynomial coefficients)}} - \underbrace{n}_{\text{(constraints)}} = 1$$




- **Cardinal splines** = unit spacing and infinite number of knots

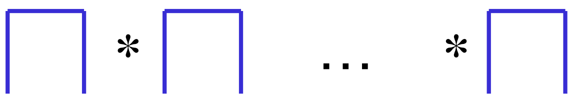
⇒ The right framework for signal processing

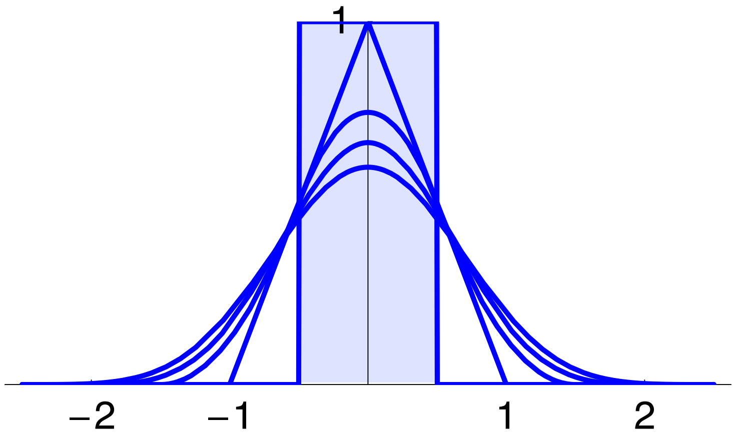
# B-spline basis functions

- Centered B-spline of degree $n$

$$\beta^0(x) = \begin{cases} 1, & |x| < \frac{1}{2} \\ 0, & |x| > \frac{1}{2} \end{cases}$$

$$\beta^n(x) = \underbrace{(\beta^0 * \beta^0 * \cdots * \beta^0)}_{(n+1)\ \text{times}}(x)$$




- Fourier-domain formula: $\hat{\beta}^n(\omega) = \left(\dfrac{\sin(\omega/2)}{\omega/2}\right)^{n+1}$

# B-spline representation

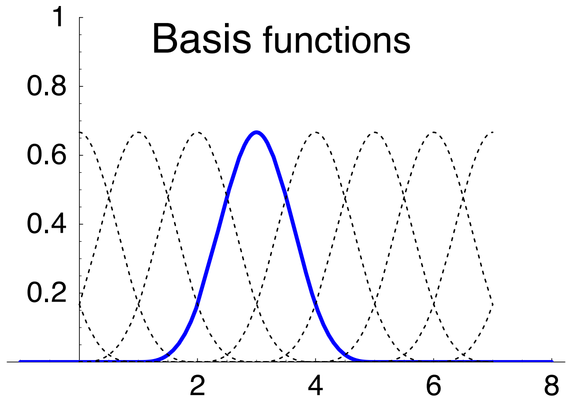
**Theorem** (Schoenberg, 1946)

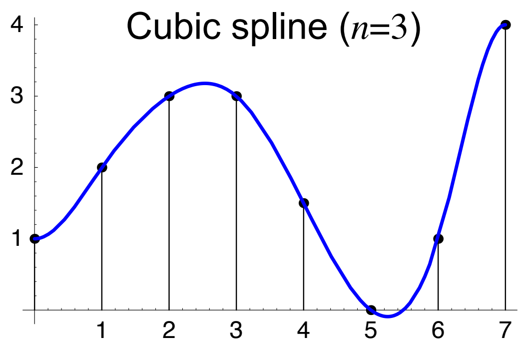
Every cardinal polynomial spline $s(x)$ has a unique and stable representation in terms of its B-spline expansion

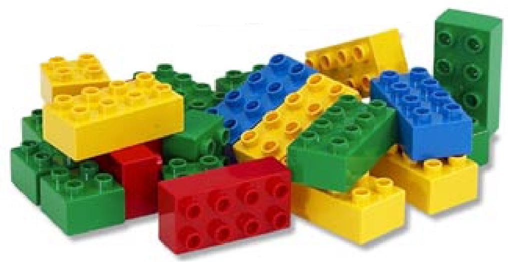
$$\boxed{s(x)} = \sum_{k\in\mathbb{Z}} \boxed{c[k]}\, \beta^n(x-k)$$

analog signal

discrete signal
(B-spline coefficients)



In modern terminology: $\{\beta^n(x-k)\}_{k\in\mathbb{Z}}$ forms a Riesz basis

# Differential operators

- Derivative

$$\mathrm{D} f(x) = \frac{\mathrm{d} f(x)}{\mathrm{d} x} \qquad \mathrm{D} \quad \stackrel{\mathcal{F}}{\longleftrightarrow} \quad j\omega$$

- Multiple differentiation

$$\mathrm{D}^n f(x) = \frac{\mathrm{d}^n f(x)}{\mathrm{d} x^n} \qquad \mathrm{D}^n \quad \stackrel{\mathcal{F}}{\longleftrightarrow} \quad (j\omega)^n$$

$$\mathrm{D}^{n_1} \mathrm{D}^{n_2} f(x) = \mathrm{D}^{n_1+n_2} f(x)$$

- Integrator

$$\mathrm{D}^{-1} f(x) = \int_{-\infty}^{x} f(t) \, \mathrm{d}t \qquad \mathrm{D}^{-1} \quad \stackrel{\mathcal{F}}{\longleftrightarrow} \quad \frac{1}{j\omega} + \pi\delta(\omega)$$

- Finite difference

$$\Delta f(x) = f\left(x + \tfrac{1}{2}\right) - f\left(x - \tfrac{1}{2}\right) \qquad \Delta \quad \stackrel{\mathcal{F}}{\longleftrightarrow} \quad e^{j\omega/2} - e^{-j\omega/2}$$

- Higher-order finite difference

$$\Delta^n f(x) = \sum_{k=0}^{n} \binom{n}{k} (-1)^k f\left(x - k + \frac{n}{2}\right) \qquad \Delta^n \quad \stackrel{\mathcal{F}}{\longleftrightarrow} \quad \left(e^{j\omega/2} - e^{-j\omega/2}\right)^n$$

$$\Delta^{n_1} \Delta^{n_2} f(x) = \Delta^{n_1+n_2} f(x)$$

# B-splines and derivatives

Derivative operator: $\mathrm{D} = \frac{\mathrm{d}}{\mathrm{d}x} \quad \stackrel{\mathcal{F}}{\longleftrightarrow} \quad \hat{D}(\omega) = j\omega$

Finite difference operator: $\Delta \quad \stackrel{\mathcal{F}}{\longleftrightarrow} \quad \hat{\Delta}(\omega) = e^{j\omega/2} - e^{-j\omega/2} = j\omega + \mathcal{O}(\omega^3)$

- Fourier transform of a B-spline revisited

$$\hat{\beta}^0(\omega) = \frac{\sin(\omega/2)}{\omega/2} = \frac{e^{j\omega/2} - e^{-j\omega/2}}{j\omega} = \frac{\hat{\Delta}(\omega)}{\hat{D}(\omega)}$$

$$\hat{\beta}^n(\omega) = \left(\frac{\sin(\omega/2)}{\omega/2}\right)^{n+1} = \frac{\hat{\Delta}^{n+1}(\omega)}{\hat{D}^{n+1}(\omega)}$$

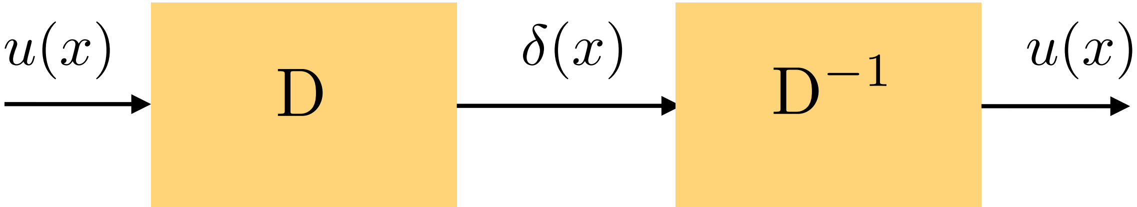
Discrete-space operator

Continuous-space operator

- Explicit derivative formula

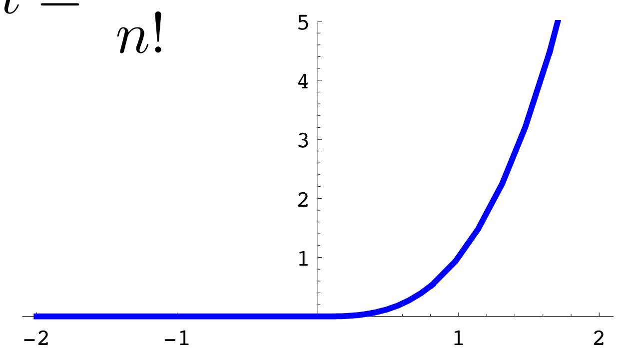
$$\mathrm{D}^m \beta^n(x) = \Delta^m \beta^{n-m}(x) \quad \stackrel{\mathcal{F}}{\longleftrightarrow} \quad \hat{D}^m(\omega) \underbrace{\left(\frac{\hat{\Delta}^{n+1}(\omega)}{\hat{D}^{n+1}(\omega)}\right)}_{\hat{\beta}^n(\omega)} = \hat{\Delta}^m(\omega) \underbrace{\left(\frac{\hat{\Delta}^{n+1-m}(\omega)}{\hat{D}^{n+1-m}(\omega)}\right)}_{\hat{\beta}^{n-m}(\omega)}$$

# Refresher: impulse response of $n$-fold integrator




$$
\begin{aligned}
\mathrm{D}^{-1}\delta(x) &= u(x) \\
\mathrm{D}^{-2}\delta(x) &= (u * u)(x) = \int_{-\infty}^{x} u(t)\,\mathrm{d}t = \frac{x_+}{1!} \\
\mathrm{D}^{-3}\delta(x) &= (u * u * u)(x) = \int_{0}^{x} t\,\mathrm{d}t = \frac{x_+^2}{2!} \\
&\;\vdots \qquad\qquad\qquad\qquad \vdots \\
\mathrm{D}^{-(n+1)}\delta(x) &= \underbrace{(u * \cdots * u)}_{(n+1)\ \text{times}}(x) = \int_{0}^{x} \frac{t^{n-1}}{(n-1)!}\,\mathrm{d}t = \frac{x_+^n}{n!}
\end{aligned}
$$

One-sided power function: $x_+^n = \begin{cases} x^n, & x \geqslant 0 \\ 0, & x < 0 \end{cases}$



# Explicit space-domain B-spline formula

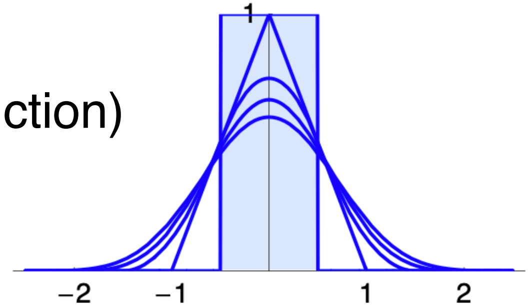
- B-spline of degree 0: $\beta^0(x) = u\left(x+\frac{1}{2}\right) - u\left(x-\frac{1}{2}\right) = \Delta u(x) = \Delta D^{-1}\delta(x)$


$$\boxed{\Delta D^{-1}\delta(x) \stackrel{\mathcal{F}}{\longleftrightarrow} \frac{\hat{\Delta}(\omega)}{\hat{D}(\omega)}}$$

- Higher-order generalization

$$\beta^n(x) = \frac{\Delta^{n+1} x_+^n}{n!} = \frac{1}{n!}\sum_{k=0}^{n+1} \binom{n+1}{k} (-1)^k \left(x - k + \frac{n+1}{2}\right)_+^n$$

where $(x)_+^n = \begin{cases} x^n, & x \geqslant 0 \\ 0, & x < 0 \end{cases}$ (one-sided power function)




Proof: 
$$\begin{aligned} \beta^n(x) &= \underbrace{(\beta^0 * \beta^0 * \cdots * \beta_0)}_{(n+1)\text{ times}}(x) = \underbrace{(\Delta u * \cdots * \Delta u)}_{(n+1)\text{ times}}(x) \\ &= \Delta^{n+1}\underbrace{(u * \cdots * u)}_{(n+1)\text{ times}}(x) = \Delta^{n+1} D^{-(n+1)}\delta(x) = \Delta^{n+1}\frac{x_+^n}{n!} \end{aligned}$$

# B-spline properties

- Symmetry, nonnegativity: $\beta^n(x) = \beta^n(-x)$ and $\beta^n(x) \geqslant 0$

- Piecewise polynomial: $\beta^n(x) = \dfrac{\Delta^{n+1} x_+^n}{n!}$

- Compact support: $\left[-\frac{n+1}{2}, \frac{n+1}{2}\right]$
  $\Rightarrow$ shortest polynomial spline of degree $n$

- Explicit differentiation formulas

$$\mathrm{D}^1 \beta^n(x) = \Delta \beta^{n-1}(x) = \beta^{n-1}\left(x + \frac{1}{2}\right) - \beta^{n-1}\left(x - \frac{1}{2}\right)$$

$$\mathrm{D}^m \beta^n(x) = \Delta^m \beta^{n-m}(x)$$

- Controlled smoothness: Hölder-continuous of order $n$
  $\Rightarrow$ bounded derivative of order $n$

# Efficient B-spline interpolation

- Discrete B-spline kernels: $b_1^n[k] = \beta^n(x)\big|_{x=k} \quad \stackrel{z}{\longleftrightarrow} \quad B_1^n(z) = \sum_{k=-\lfloor n/2 \rfloor}^{\lfloor n/2 \rfloor} \beta^n[k] z^{-k}$

- B-spline interpolation: filtering solution

$$f(x) = \sum_{l \in \mathbb{Z}} c[l] \beta^n(x - l)$$

$$f[k] = (b_1^n * c)[k] \quad \Rightarrow \quad c[k] = (h_{\text{int}}^n * f)[k]$$

- Efficient recursive solution

$$h_{\text{int}}^n[k] \quad \stackrel{z}{\longleftrightarrow} \quad \frac{1}{B_1^n(z)} = c_0 \prod_{i=1}^{\lfloor n/2 \rfloor} \left( \frac{-\alpha_i}{(1 - \alpha_i z)(1 - \alpha_i z^{-1})} \right)$$

⇒ Cascade of symmetric exponential filters (cf. Chap IP-3)

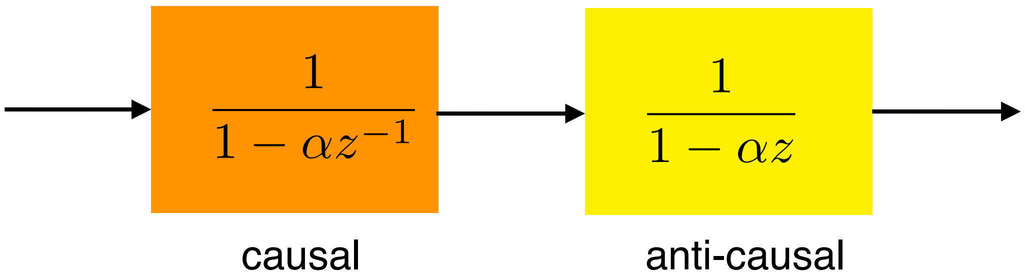


TABLE I

TRANSFER FUNCTIONS AND POLES OF DIRECT B-SPLINE FILTERS FOR $n = 0$ TO 7

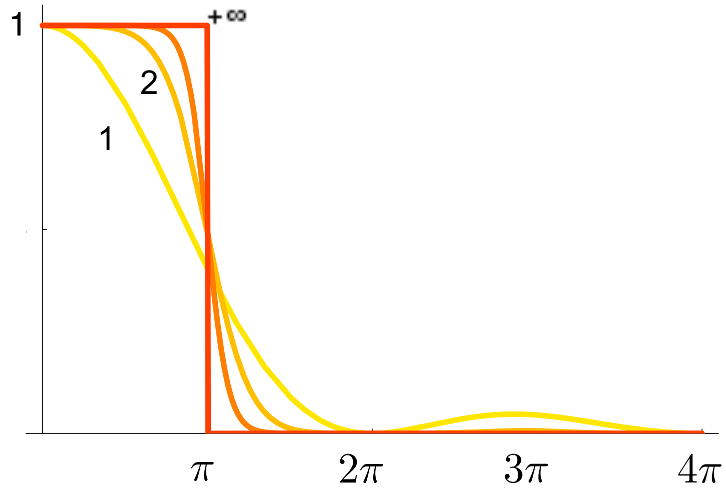
| $n$ | Direct B-Spline Filter: $B_1^n(z)^{-1}$ | $c_0$ | Poles: $\{\lvert z_i\rvert < 1, i = 1, \cdots, n\}$ |
|---|---|---|---|
| 0 | $1$ | 1 | — |
| 1 | $1$ | 1 | — |
| 2 | $\dfrac{8}{z + 6 + z^{-1}}$ | 8 | $z_1 = -3 + 2\sqrt{2} = -0.171573$ |
| 3 | $\dfrac{6}{z + 4 + z^{-1}}$ | 6 | $z_1 = -2 + \sqrt{3} = -0.267949$ |
| 4 | $\dfrac{384}{z^2 + 76z + 230 + 76z^{-1} + z^{-2}}$ | 384 | $z_1 = -0.361341$<br>$z_2 = -0.0137254$ |
| 5 | $\dfrac{120}{z^2 + 26z + 66 + 26z^{-1} + z^{-2}}$ | 120 | $z_1 = -0.430575$<br>$z_2 = -0.0430963$ |
| 6 | $\dfrac{46080}{z^3 + 722z^2 + 10543z + 23548 + 10543z^{-1} + 722z^{-2} + z^{-3}}$ | 46080 | $z_1 = -0.488295$<br>$z_2 = -0.0816793$<br>$z_3 = -0.00141415$ |
| 7 | $\dfrac{5040}{z^3 + 120z^2 + 1191z + 2416 + 1191z^{-1} + 120z^{-2} + z^{-3}}$ | 5040 | $z_1 = -0.53528$<br>$z_2 = -0.122555$<br>$z_3 = -0.00914869$ |

# Limiting behavior (splines)

- Spline interpolator

Impulse response / Frequency response

$$\varphi_{\mathrm{int}}^{n}(x) \quad \stackrel{\mathcal{F}}{\longleftrightarrow} \quad \hat{\varphi}_{\mathrm{int}}^{n}(\omega) = \left(\frac{\sin(\omega/2)}{\omega/2}\right)^{n+1} H_{\mathrm{int}}^{n}(e^{j\omega})$$




- Asymptotic property

The cardinal spline interpolators converge to the sinc-interpolator (ideal filter) as the degree goes to infinity:

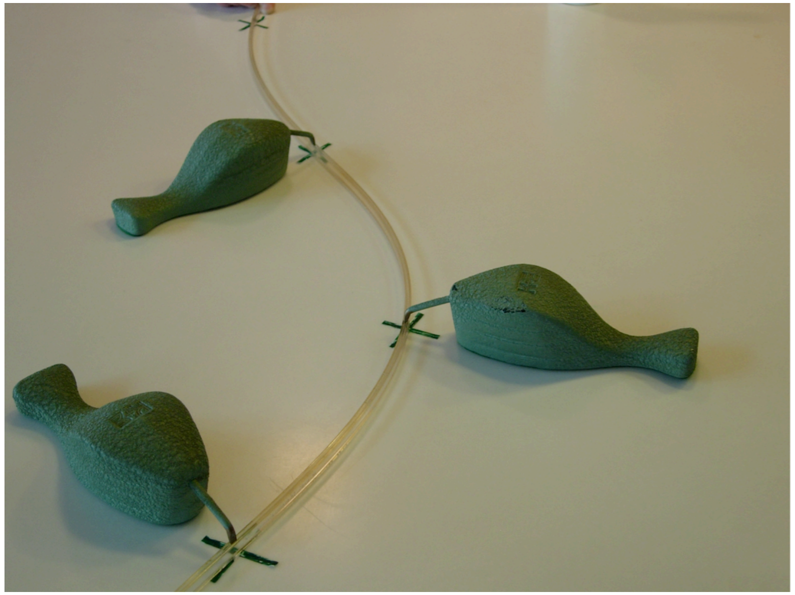
$$\lim_{n\to\infty} \varphi_{\mathrm{int}}^{n}(x) = \mathrm{sinc}(x), \qquad \lim_{n\to\infty} \hat{\varphi}_{\mathrm{int}}^{n}(\omega) = \mathrm{rect}\left(\frac{\omega}{2\pi}\right)$$

(in all $L_p$-norms)

(Aldroubi et al., *Sig. Proc.*, 1992)

Includes Shannon's theory as a particular case !

# Splines: variational properties



- Spline [American Heritage Dictionary]
  - *"A flexible piece of wood, hard rubber, or metal used in drawing curves."*
  - *"A wooden or metal strip; a slat."*

- Mathematical arguments
  - First integral equation: $\forall f \in W_2^m, \quad \|\mathrm{D}^m f\|_{L_2}^2 = \|\mathrm{D}^m s_{\mathrm{int}}\|_{L_2}^2 + \|\mathrm{D}^m (f - s_{\mathrm{int}})\|_{L_2}^2$

    $f$: any function that is $m$ times differentiable ($L_2$-sense)

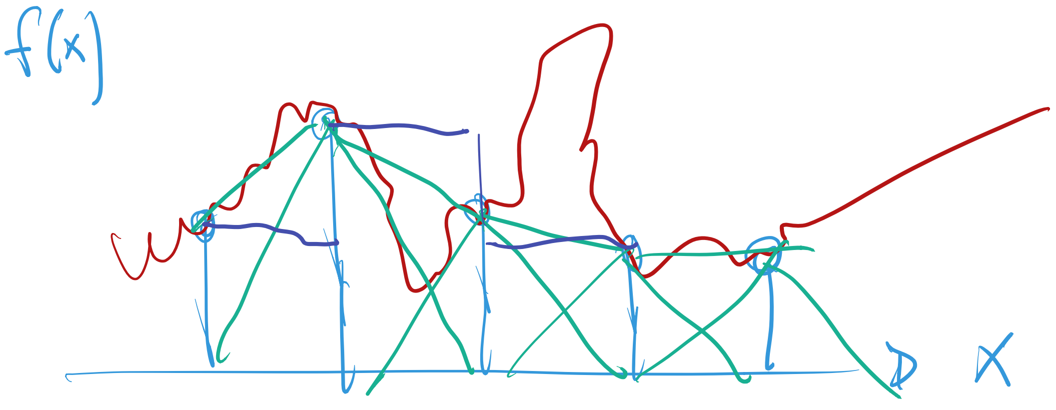
    $s_{\mathrm{int}}$: spline interpolant of odd degree $n = 2m - 1$ such that $f[k] = s_{\mathrm{int}}(k)$

  - Optimal interpolant [Ahlberg-Nilson, 1964]

$$f(x) = s_{\mathrm{int}}(x) \qquad \Rightarrow \qquad \int_{-\infty}^{+\infty} \left| f^{(m)}(x) \right|^2 \mathrm{d}x \quad \text{minimum}$$

$\Rightarrow$ minimum-curvature property of cubic spline ($m = 2$)

# Splines unify multiple perspectives




Engineer

$$f(x) = \sum f(k)\,\varphi(x-k)$$

Computer Scientist

Physicist

$$f(x) = \sum_k f(k)\,\mathrm{tri}(x-k)$$

Sig Proc Theoretician

$$f(x) = \sum f(k)\,\mathrm{sinc}(x-k)$$

$$f(x) = \sum f(k)\,\mathrm{rect}(x-k)$$

Mathematician

s.t. $f(k) = f[k]$

$\min \|D^2 f\|$

$\rightarrow$ prove solution exist and is unique!

# Spline-based gradient operator

QUESTION: How to compute $\nabla f(x,y) = \big(f_x(x,y), f_y(x,y)\big)$?

ANSWER: Exact computation using spline interpolation model: $f(\boldsymbol{x}) = \sum_{\boldsymbol{k}\in\mathbb{Z}^d} c[\boldsymbol{k}]\varphi(\boldsymbol{x}-\boldsymbol{k})$

- Spline model: $f(x) = \sum_{l\in\mathbb{Z}} c[l]\beta^n(x-l)$

  (1D formulation since both B-splines and derivative operators are separable)

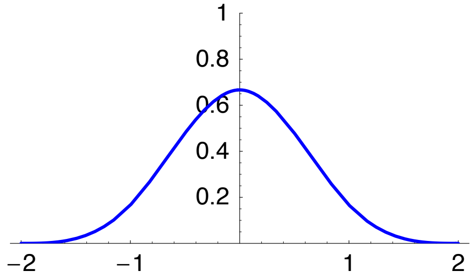
- Differentiation:

$$f_x(x) = \mathrm{D}f(x) = \sum_{l\in\mathbb{Z}} c[l]\,\mathrm{D}\beta^n(x-l) = \sum_{l\in\mathbb{Z}} c[l]\,\Delta\beta^{n-1}(x-l)$$

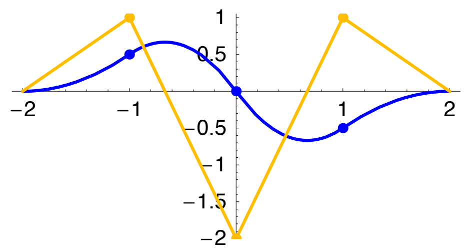
- Discretization:

$$f_x(k) = \sum_{l\in\mathbb{Z}} c[l]\,\Delta\beta^{n-1}(k-l) = (c * d_1^n)[k] = \left(f * h_{\text{int}}^n * d_1^n\right)[k]$$

- Derivative kernels: $d_1^n[k] = \Delta\beta^{n-1}(k)$



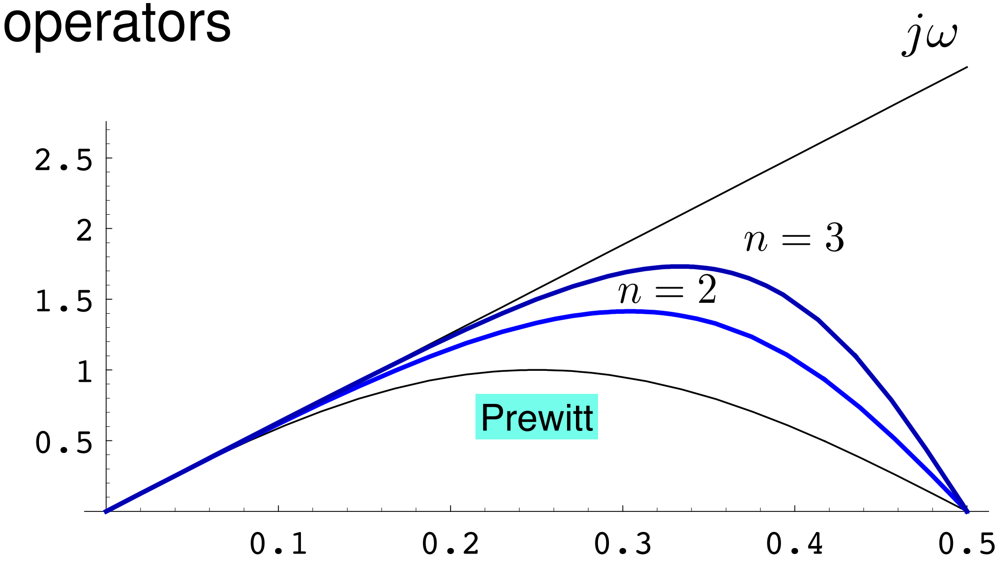

Example: cubic B-spline



# Differentiation filters

| | First derivative | Second derivative |
|---|---|---|
| Generic | $\dfrac{\sum_{k\in\mathbb{Z}} \Delta\beta^{n-1}(k)z^{-k}}{B_1^n(z)}$ | $\dfrac{\sum_{k\in\mathbb{Z}} \Delta^2\beta^{n-2}(k)z^{-k}}{B_1^n(z)}$ |
| $n=1$ | $(z-1)$ | N/A |
| $n=2$ | $\left(\dfrac{8}{z+6+z^{-1}}\right)\left(\dfrac{z-z^{-1}}{2}\right)$ | $\left(\dfrac{8}{z+6+z^{-1}}\right)(z-2+z^{-1})$ |
| $n=3$ | $\left(\dfrac{6}{z+4+z^{-1}}\right)\left(\dfrac{z-z^{-1}}{2}\right)$ | $\left(\dfrac{6}{z+4+z^{-1}}\right)(z-2+z^{-1})$ |

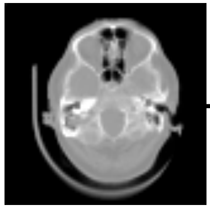
- Comparison of differentiation operators



# Example: *Cubic*-spline image differentials (n=3)

## Convolution-based implementation

$f(k,l)$



→ 2D filtering (separable) → $c[k,l]$ → Differential mask →

$$\frac{\partial^{p+q}}{\partial x^p \partial y^q} f(k,l)$$

$$\left.\frac{\partial^{p+q}}{\partial x^p \partial y^q}\varphi(x,y)\right|_{x=k,y=l}$$

- Laplacian

$$\frac{\partial^2}{\partial x^2}+\frac{\partial^2}{\partial y^2}: \quad \frac{1}{3}\begin{bmatrix} 1 & 1 & 1 \\ 1 & -8 & 1 \\ 1 & 1 & 1 \end{bmatrix}$$

- Hessian masks

$$\frac{\partial^2}{\partial x^2}: \frac{1}{6}\begin{bmatrix} 1 & -2 & 1 \\ 4 & -8 & 4 \\ 1 & -2 & 1 \end{bmatrix} \qquad \frac{\partial^2}{\partial x \partial y}: \frac{1}{2\cdot 2}\begin{bmatrix} 1 & 0 & -1 \\ 0 & 0 & 0 \\ -1 & 0 & 1 \end{bmatrix}$$

$$\frac{\partial^2}{\partial y^2}: \quad \frac{1}{6}\begin{bmatrix} 1 & 4 & 1 \\ -2 & -8 & -2 \\ 1 & 4 & 1 \end{bmatrix}$$

- Gradient masks

$$\frac{\partial}{\partial x}: \quad \frac{1}{6\cdot 2}\begin{bmatrix} -1 & 0 & 1 \\ -4 & 0 & 4 \\ -1 & 0 & 1 \end{bmatrix}$$

$$\frac{\partial}{\partial y}: \quad \frac{1}{6\cdot 2}\begin{bmatrix} -1 & -4 & -1 \\ 0 & 0 & 0 \\ 1 & 4 & 1 \end{bmatrix}$$

# Why B-splines ?

- Symmetry; compact support; explicit piecewise-polynomial form; positivity
- Efficient interpolation algorithms (recursive filtering)
- Well-suited for explicit computation of differential operators

  Differentiation $\Rightarrow$ degree reduction & finite differences

- Explicit control of image smoothness
- Generality: transition from piecewise-constant $(n = 0)$ to bandlimited model $(n \rightarrow +\infty)$
- Shortest functions that reproduce polynomials of degree $n$:

$$\exists a_m[k], \quad x^m = \sum_{k\in\mathbb{Z}} a_m[k]\beta^n(x-k), \quad (m = 0, \dots, n)$$

  In particular: $$\sum_{k\in\mathbb{Z}} \beta^n(x-k) = 1$$ (partition of unity)

- Good approximation power

$\Rightarrow$ **Best interpolation quality for a given computational budget!**

# 7.3 FROM SPLINES TO WAVELETS

- Multiresolution: motivation
- Notation and conventions
- Haar transform revisited
- Image pyramids
- Error (or Laplacian) pyramid
- From pyramids to wavelets

# Multiresolution: motivation

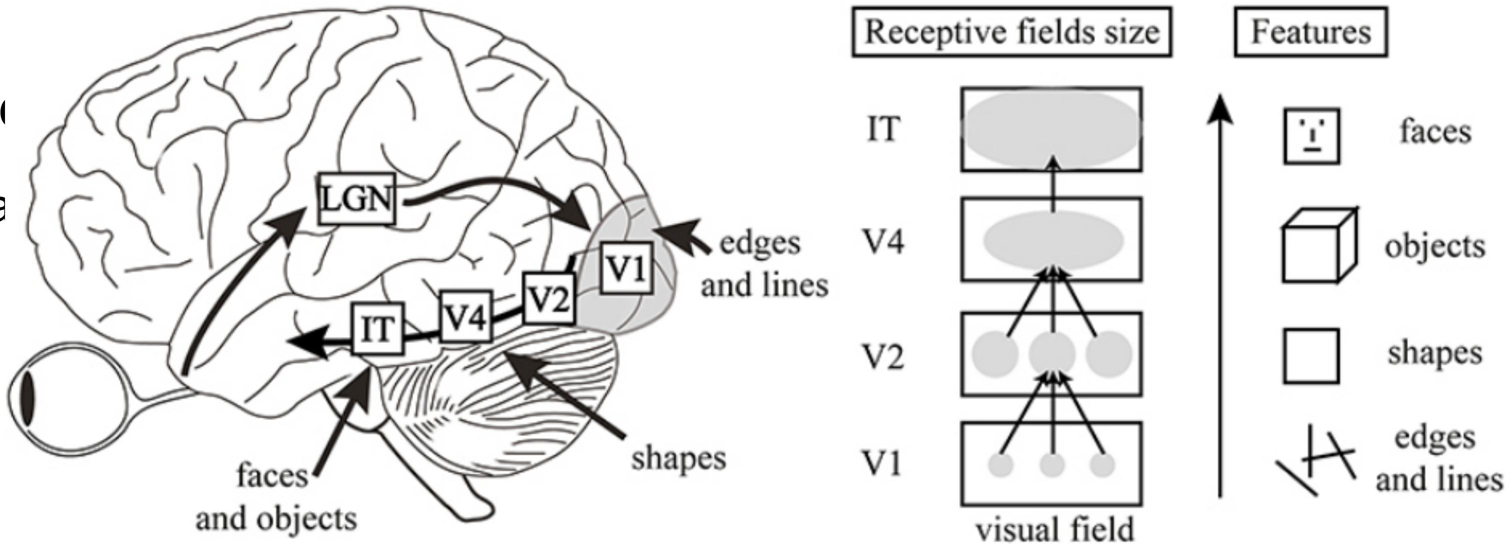
Why should one be stuck with a *fixed-resolution* image format?

- Multiscale processing
  - Adapting resolution: coarse-to-fine or multigrid iteration strategies
  - Speed: less computation + faster convergence
  - Robustness
  - Inspired by the human visual system
  - Old idea (70's, early 80's)      [Rosenfeld, Burt & Adelson]

$\Rightarrow$ Multiscale implementation of most iterative IP algorithms

- Multir

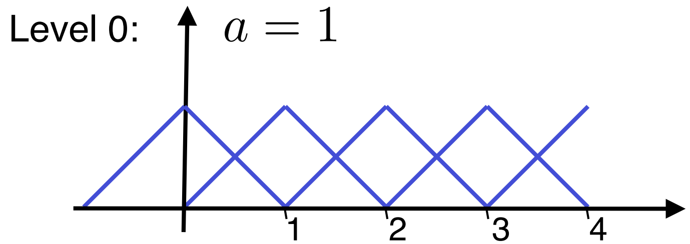
Compa … sing)



# Notation and conventions

- Basic signal space

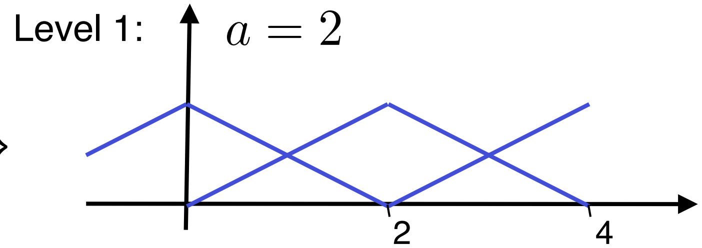
$$V_0 = \left\{ s(x) = \sum_{k\in\mathbb{Z}} c[k]\varphi(x-k),\ c \in l_2 \right\}$$




- Fine-to-coarse sequence of subspaces

$$V_0 \rightarrow V_1 \rightarrow \quad \dots \quad \rightarrow V_i$$

$$V_i = \left\{ s_i(x) = \sum_{k\in\mathbb{Z}} c_i[k]\varphi\left(\frac{x-2^i k}{2^i}\right),\ c_i \in \ell_2 \right\}$$



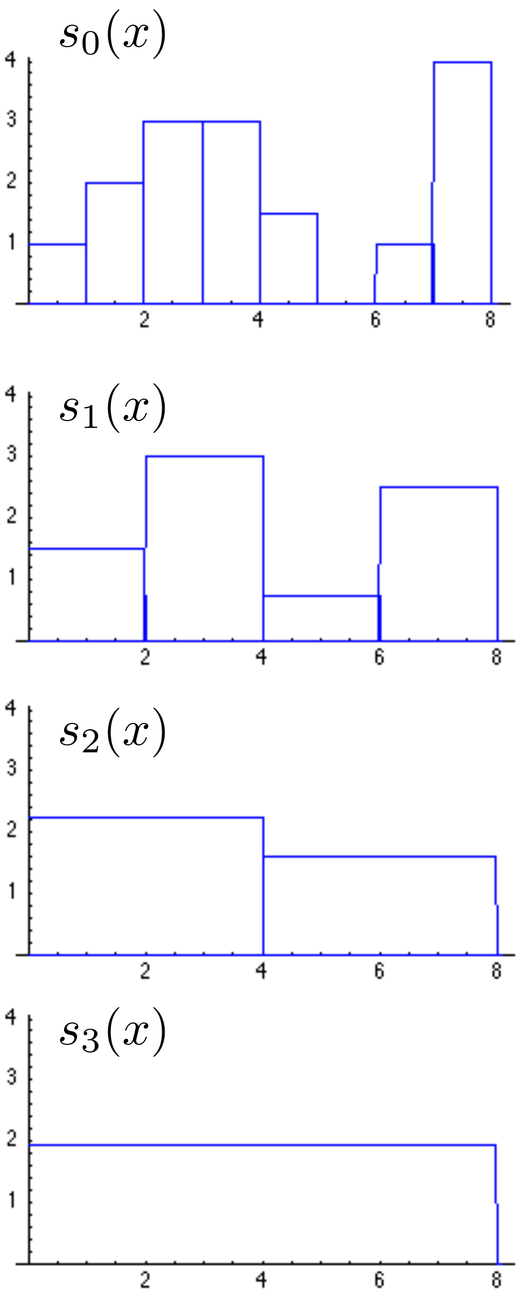

- Basis functions at resolution $a = 2^i$

$$\varphi_{i,k} = \varphi(x/2^i - k)$$

Scale index $i$ $\Rightarrow$ dilation by $a = 2^i$ (powers of two!)

Translation index $k$ $\Rightarrow$ shift by $b = 2^i \cdot k$

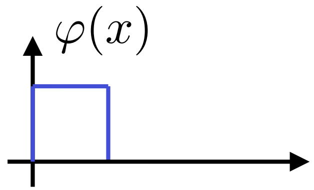
# Wavelets: Haar transform revisited




Signal representation

$$s_0(x) = \sum_{k\in\mathbb{Z}} c[k]\varphi(x-k)$$

Scaling function



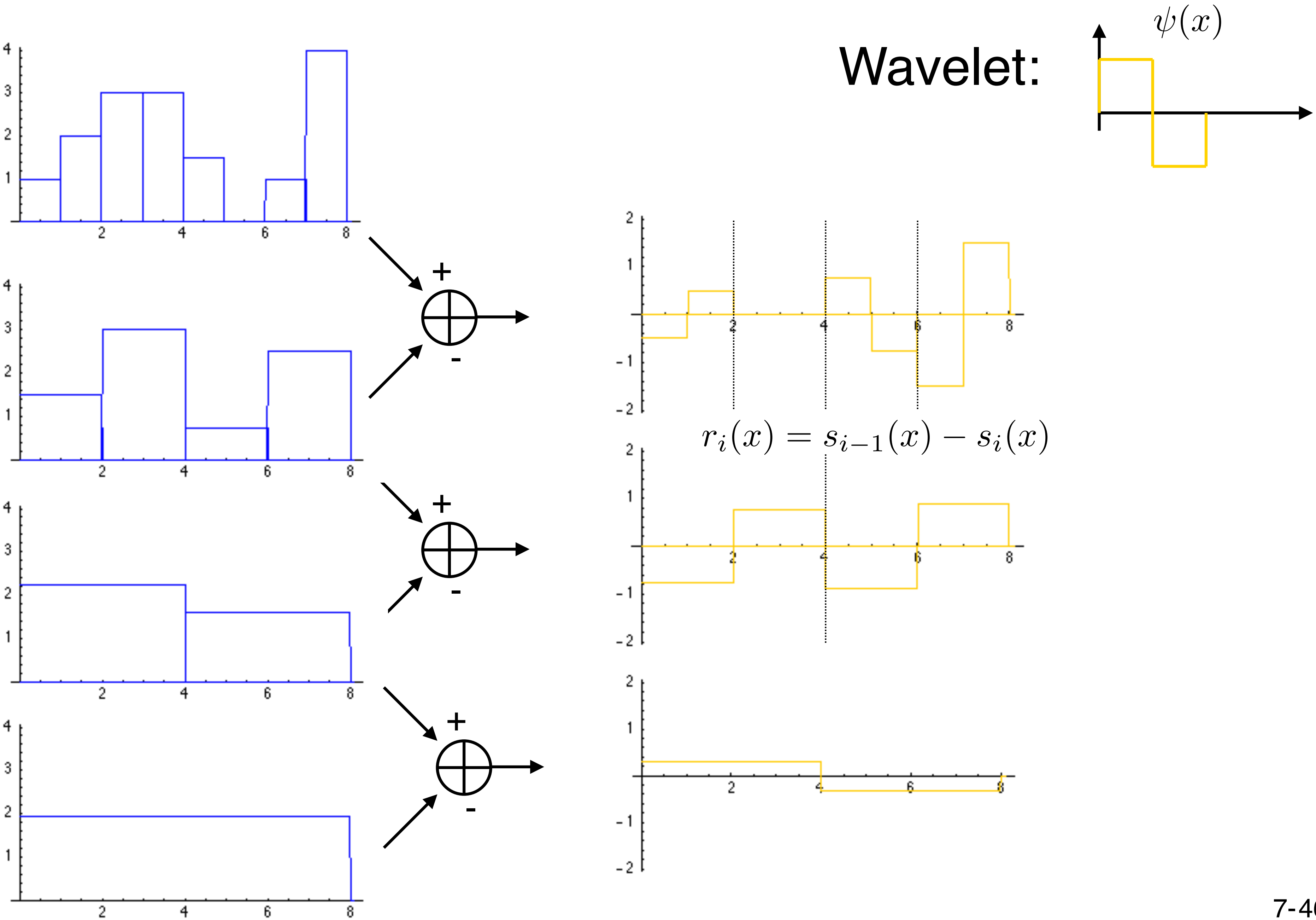

Multiscale signal representation

$$s_i(x) = \sum_{k\in\mathbb{Z}} c_i[k]\varphi_{i,k}(x)$$

Multiscale basis functions

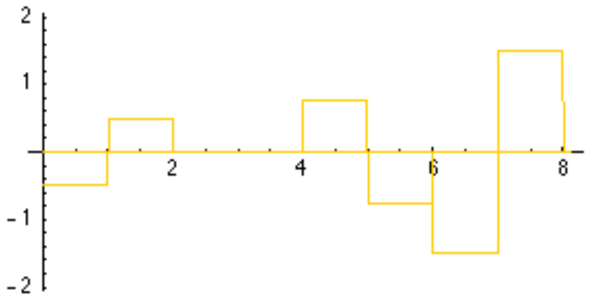
$$\varphi_{i,k}(x) = \varphi\left(\frac{x-2^i k}{2^i}\right)$$

# Wavelets: Haar transform revisited



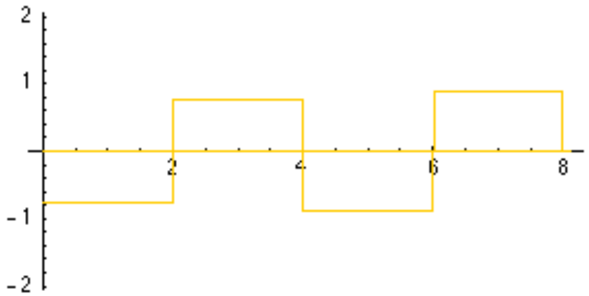
# Wavelets: Haar transform revisited

$$r_1(x) = \sum_k d_1[k]\psi_{1,k}$$



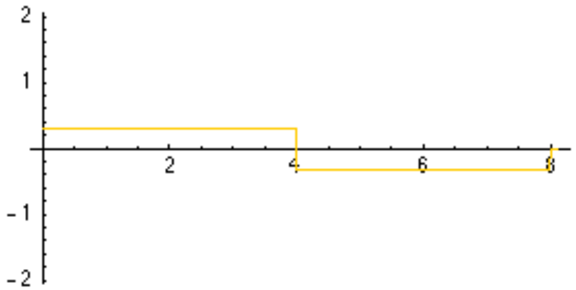
+

$$r_2(x) = \sum_k d_2[k]\psi_{2,k}$$



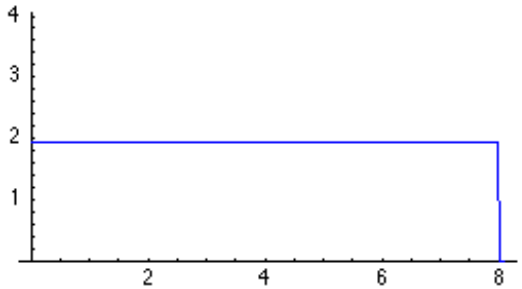
+

$$r_3(x) = \sum_k d_3[k]\psi_{3,k}$$



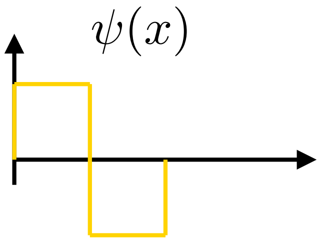
+

$$s_3(x) = \sum_k c_3[k]\varphi_{3,k}$$

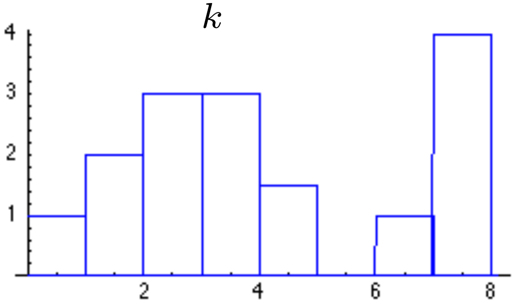


=

Wavelet:




$$s(x) = \sum_k c[k]\varphi(x-k)$$



# Basic ingredients

- Generic representation: $s_i = \sum_k c_i[k]\varphi_{i,k}$

  $\Rightarrow$ vector space $V_i$ (integer shift-invariant)

- Two-scale relation $\Rightarrow$ sequence of *nested* subspaces

  $$L_2(\mathbb{R}) \cdots \supset V_0 \supset V_1 \supset \cdots \supset V_i \supset \cdots \supset \{0\}$$

- Sequence of minimum-error approximations: $s_0 \longrightarrow s_1 \cdots \longrightarrow s_i$

  $\Rightarrow$ $s_i$ is the *orthogonal projection* of $s_0$ onto $V_i$

- Decoupling between error and approximation

  $\Rightarrow$ *orthogonality* of the residual spaces (i.e., $\forall k \in \mathbb{Z},\ \langle \varphi(\cdot), \psi(\cdot - k)\rangle = 0$)

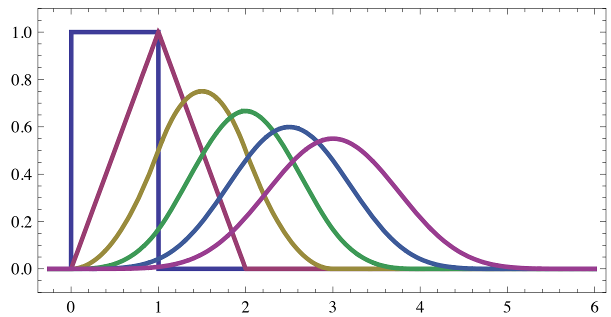
- Wavelet decomposition: compact representation of the residues

  $$r_i = s_{i-1} - s_i = \sum_{k\in\mathbb{Z}} d_i[k]\psi_{i,k} \in W_i$$

# m-scale relation

- Causal B-spline of degree $n$

$$\beta_+^n(x) = \beta^n(x - \tfrac{n+1}{2})$$

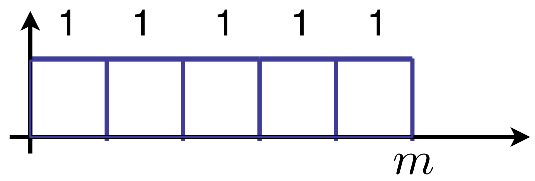



- B-spline dilated by an integer factor $m$

$$\beta_+^n(x/m) = \sum_{k\in\mathbb{Z}} h_m^n[k]\beta_+^n(x-k) \quad \text{with} \quad H_m^n(z) = \frac{1}{m^n}\left(\sum_{k=0}^{m-1} z^{-k}\right)^{n+1}$$

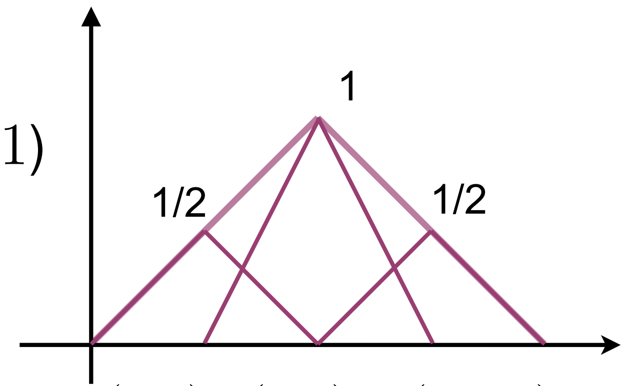
- Example 1: Piecewise-constant case ($n = 0, m$)

$$H_m^0(z) = 1 + z^{-1} + \cdots + z^{-(m-1)} \text{ (moving-sum filter)}$$




- Example 2: piecewise-linear splines ($m = 2, n = 1$)

$$H_2^1(z) = \frac{1}{2}(z + 2 + z^{-1})$$




$$(1,1) * (1,1) = (1,2,1)$$

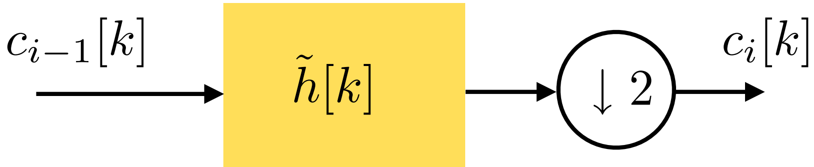
# Image pyramids

- Successive approximations at dyadic scales

$$V_i = \left\{ s_i(\boldsymbol{x}) = \sum_{\boldsymbol{k}\in\mathbb{Z}^d} c_i[\boldsymbol{k}]\varphi_{2^i}(\boldsymbol{x} - 2^i\boldsymbol{k}) : c_i[\boldsymbol{k}] \in \ell_2(\mathbb{Z}^d) \right\}$$

Rescaled basis function: $\varphi_{2^i}(\boldsymbol{x}) \triangleq \prod_{k=1}^{d} \beta^n\left(\frac{x_k}{2^i}\right)$

- Repeated, separable application of REDUCE operator




- Optimal prefilter for minimum $L_2$-norm approximation

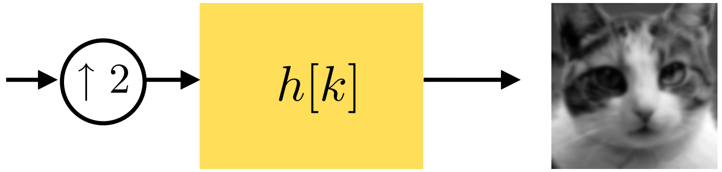
$\tilde{h}$: separable, uniquely specified given the scaling function $\varphi(x)$

Haar: 2 point average; otherwise typically IIR

# Error (or Laplacian) pyramid

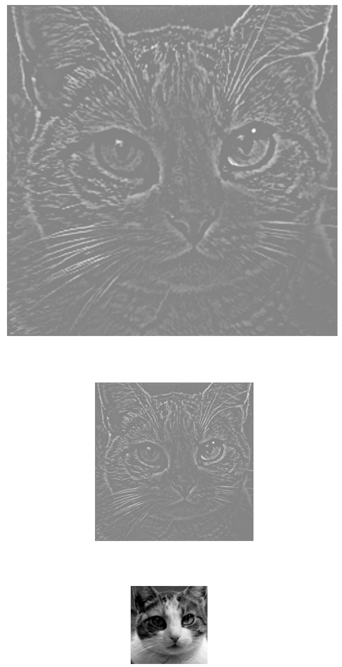
REDUCE


EXPAND

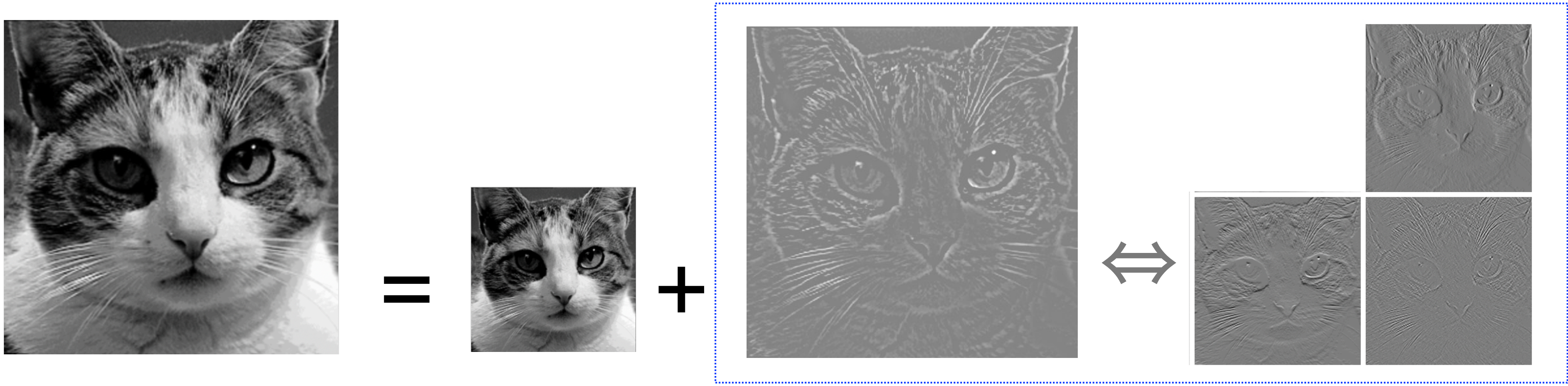



## Least-squares pyramid

## Error pyramid

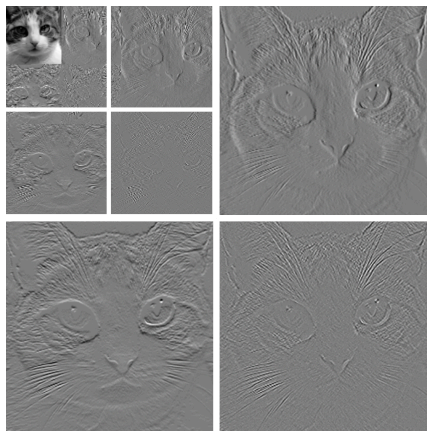


# And the connection with wavelets



and iterate.... you get the wavelet transform:

=

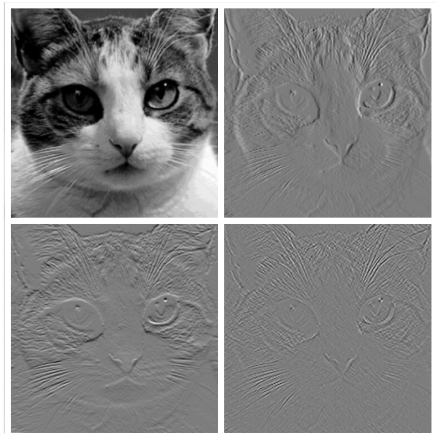


S.G. Mallat, "A theory of multiresolution signal decomposition: the wavelet representation," *IEEE Trans. Pattern Anal. Machine Intell.*, 11 (7), pp. 674-693, 1989

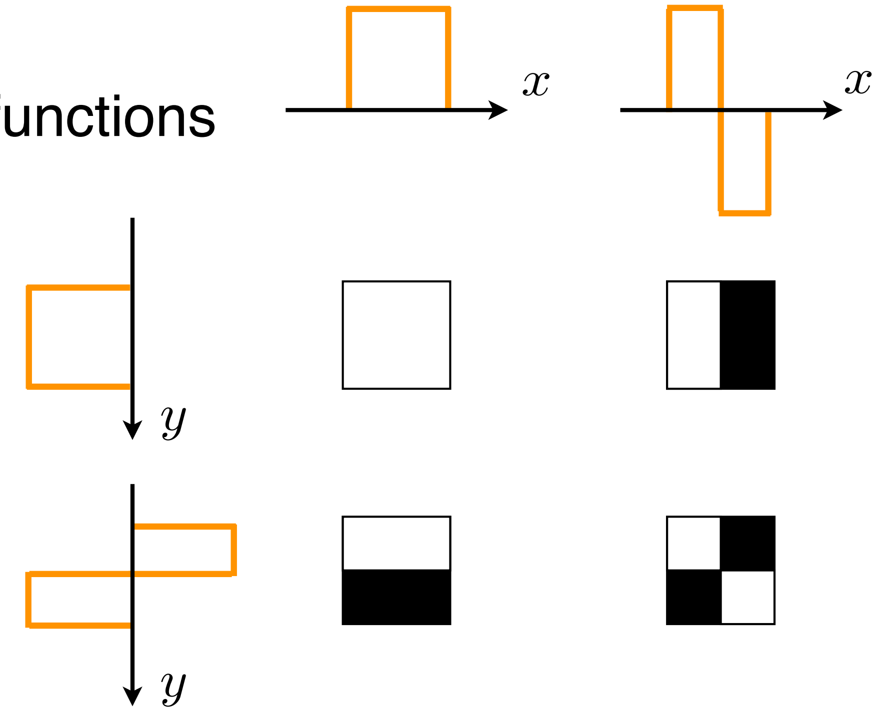
# 2D basis functions and Haar expansion

$$f(x,y) = \sum_{i,\boldsymbol{k}} w_{i,\boldsymbol{k}} \, \psi_{i,\boldsymbol{k}}(x,y)$$

Expansion coefficients

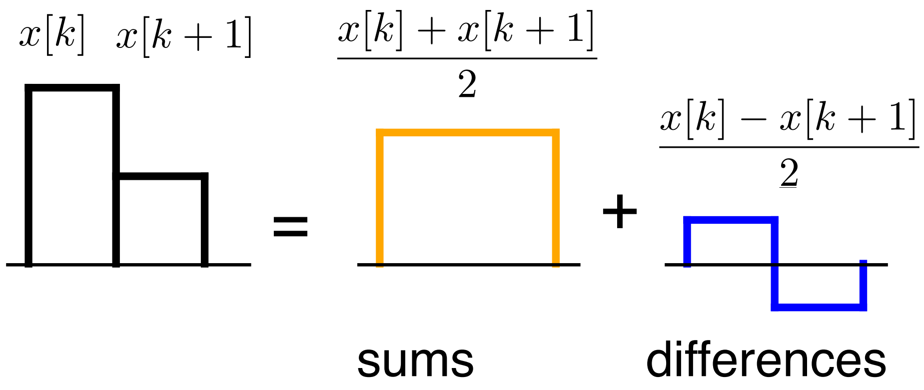


Tensor-product basis functions

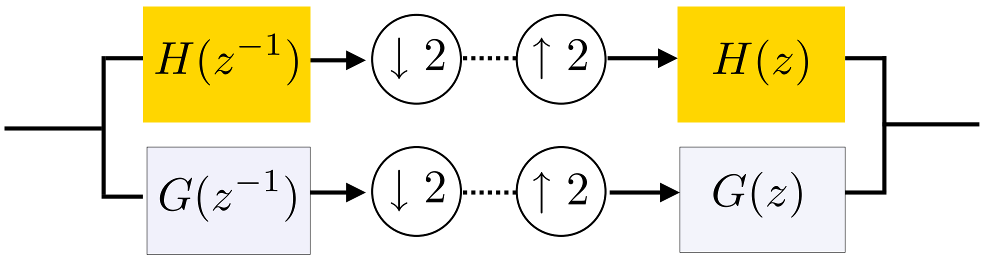


# Haar: filterbank formulation

- Basic principle



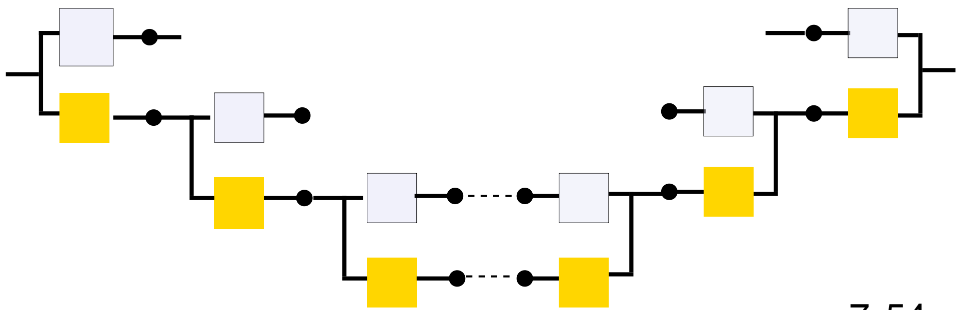

- Perfect reconstruction filterbank




Lowpass filter: $H(z) = \frac{1}{\sqrt{2}}(1 + z^{-1})$

Highpass filter: $G(z) = \frac{1}{\sqrt{2}}(1 - z^{-1})$

- Tree-structured filterbank algorithm



# 7.4 SUMMARY

- A continuous/discrete image representation involves shifted basis functions $\varphi(\boldsymbol{x} - \boldsymbol{k})$ centered on the pixels. There is exactly one coefficient $c[\boldsymbol{k}]$ per pixel location.
- The generating function $\varphi(\boldsymbol{x})$ may—or may not—have the interpolation property; in the latter case, image interpolation involves a digital prefiltering step.
- Interpolation is required for performing geometric transformations such as rotation, scaling or warping.
- The B-splines are a very useful family of generating functions. They are easy to manipulate and have many optimal properties (short support, etc...).
- A multiscale image representation (or pyramid) is a series of fine-to-coarse approximations using basis functions of increasing sizes (dyadic scale progression).
- The pyramid is constructed simply by iterative lowpass filtering and down-sampling.
- The residues in a LS pyramid are orthogonal to the next coarser image approximation. They can be represented concisely using wavelets (one-to-one representation).

# References


- S.G. Mallat, “A theory of multiresolution signal decomposition: The wavelet representation, *IEEE Trans. Pattern Anal. Machine Intell.*, vol. 11, no. 7, pp. 674-693, 1989.
- S. Mallat, *A wavelet tour of signal processing.* San Diego: Academic Press, 1998.
- M. Unser, A. Aldroubi and M. Eden, “B-spline signal processing: Part I—Theory,” *IEEE Trans. Signal Processing*, vol. 41, no. 2, pp. 821-833, 1993.
- M. Unser, A. Aldroubi and M. Eden, “B-spline signal processing: Part II—Efficient design and applications,” *IEEE Trans. Signal Processing*, vol. 41, no. 2, pp. 834-848, 1993.
- M. Unser, “Splines: A perfect fit for signal and image processing,” *IEEE Signal Processing Magazine*, vol. 16, no. 6, pp. 22-38, November 1999.