EECS 556 – Image Processing– W 09

Interpolation

- Interpolation techniques
- B-splines

What is image processing?

Image processing is the application of 2D signal processing methods to images

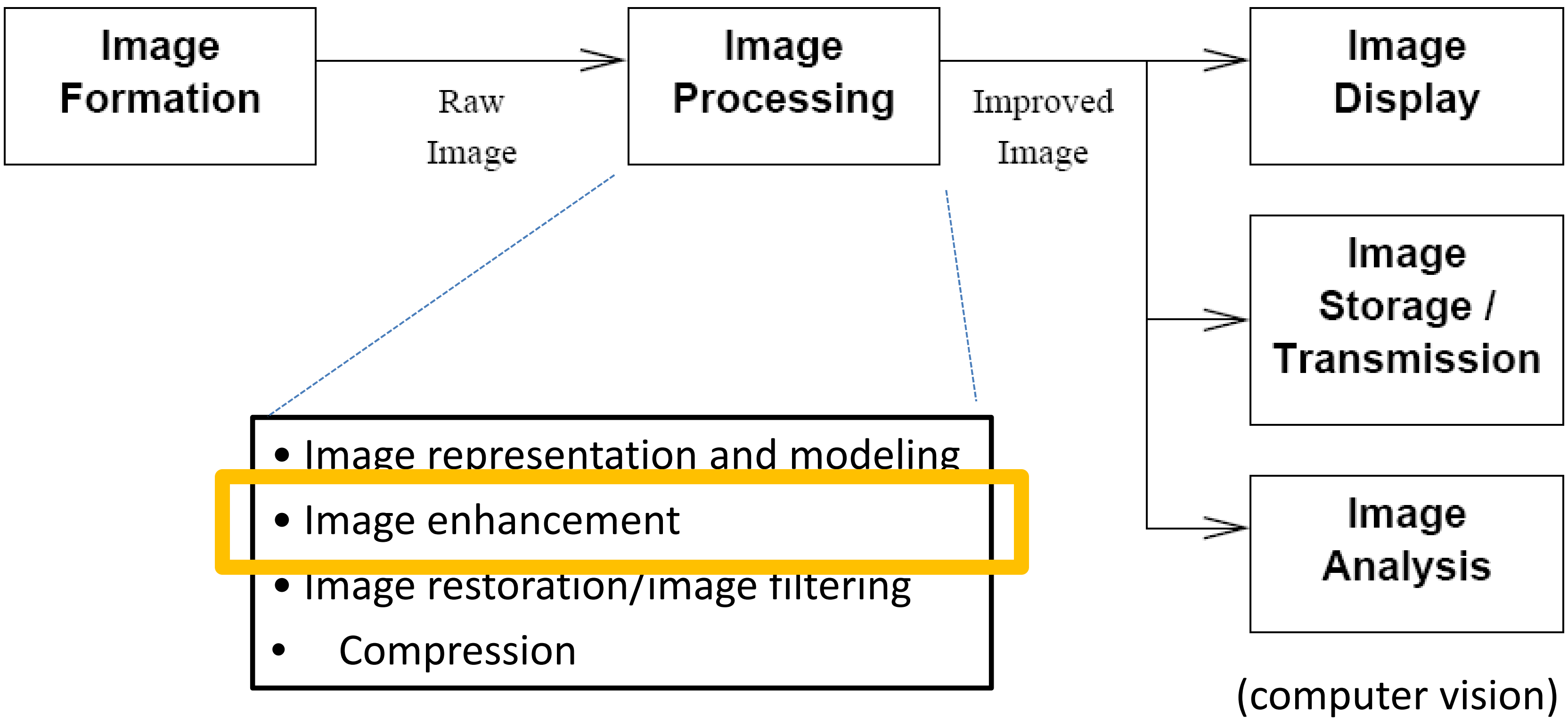


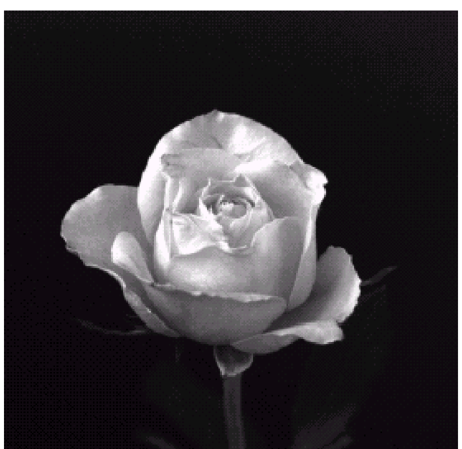
Image enhancement

- Accentuate certain desired features for subsequent analysis or display.
  - Contrast stretching / dynamic range adjustment
  - Color histogram normalization
  - Noise reduction
  - Sharpening
  - Edge detection
  - Corner detection
  - Image interpolation

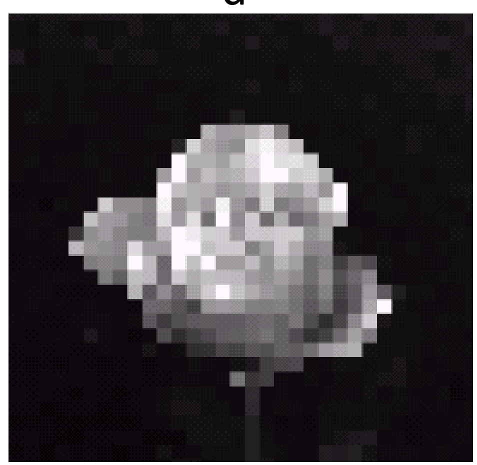
What is interpolation?

- given a discrete-space image $g_d[n,m]$
- This corresponds to samples of some continuous space image $g_a(x, y)$
- Compute values of the CS image $g_a(x, y)$ at (x,y) locations *other than the sample locations.*

$g_a(x, y)$



$g_d[n,m]$



$g'_d[n,m]$

What is interpolation?

- Interpolation is critical for:
  - Displaying
  - Image zooming
  - Warping
  - Coding
  - Motion estimation

Is perfect recovery always possible?

No, (never actually), unless some assumptions on $g_a(x,y)$ are made

There are uncountable infinite collection of functions $g_a(x, y)$ that agree with $g_d[n,m]$ at the sample points

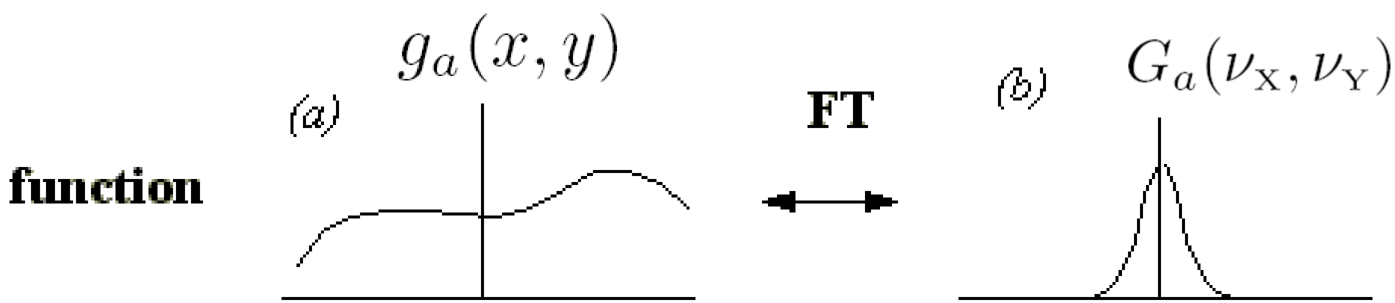
What are these assumptions?

- g_a must be band-limited
- Sampling rate must satisfy Nyquist theorem

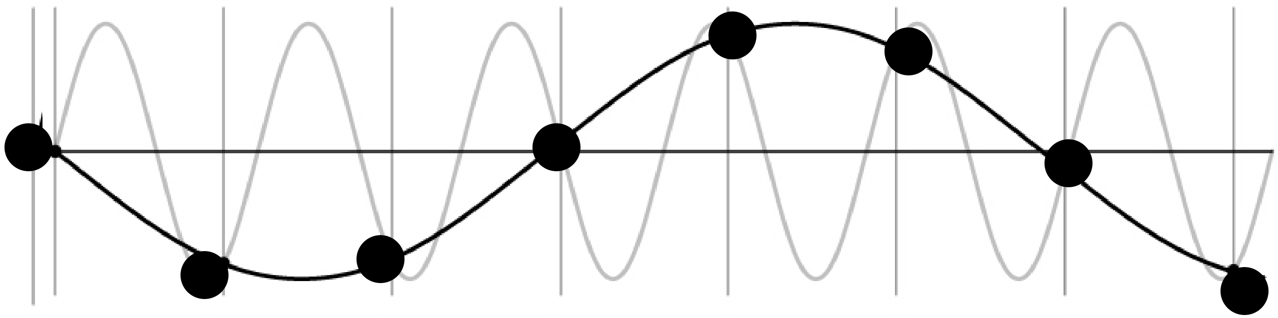
g_a must be band-limited

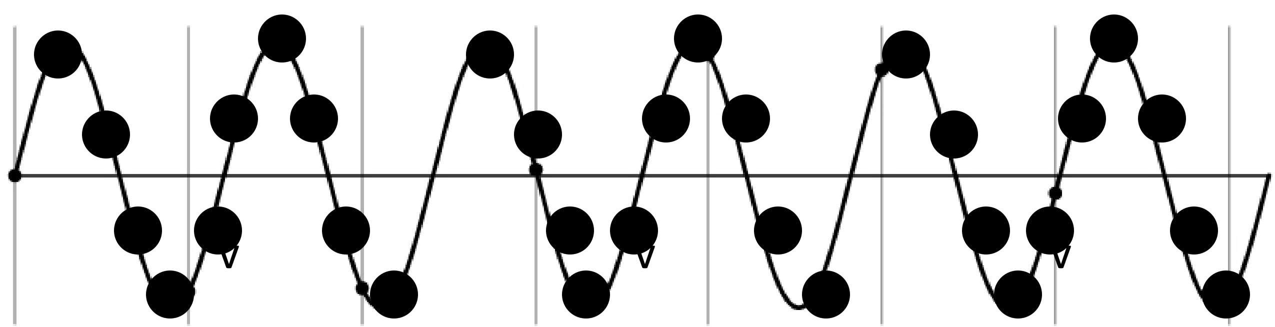
There exists $(\nu_X^{\max}, \nu_Y^{\max})$ such that

$$G_a(\nu_X, \nu_Y) = 0 \text{ for } |\nu_X| \geq \nu_X^{\max} \text{ or } |\nu_Y| \geq \nu_Y^{\max}$$



Sampling rate must satisfy Nyquist theorem





Ideal Uniform Rectilinear Sampling

- 2D uniformly sampled grid

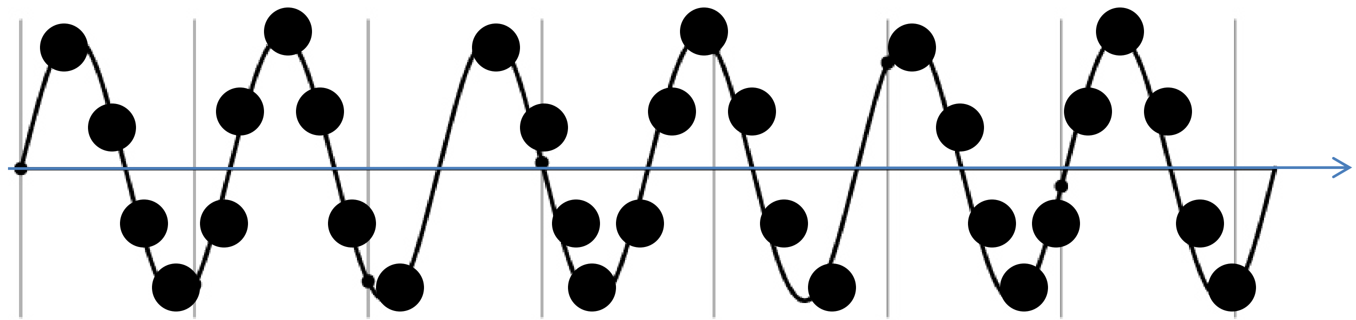
$$\boxed{g_d[n, m] = g_a(n\Delta_{\mathrm{X}}, m\Delta_{\mathrm{Y}})\,.}$$

Δ_{X} and Δ_{Y} be the **sampling intervals**

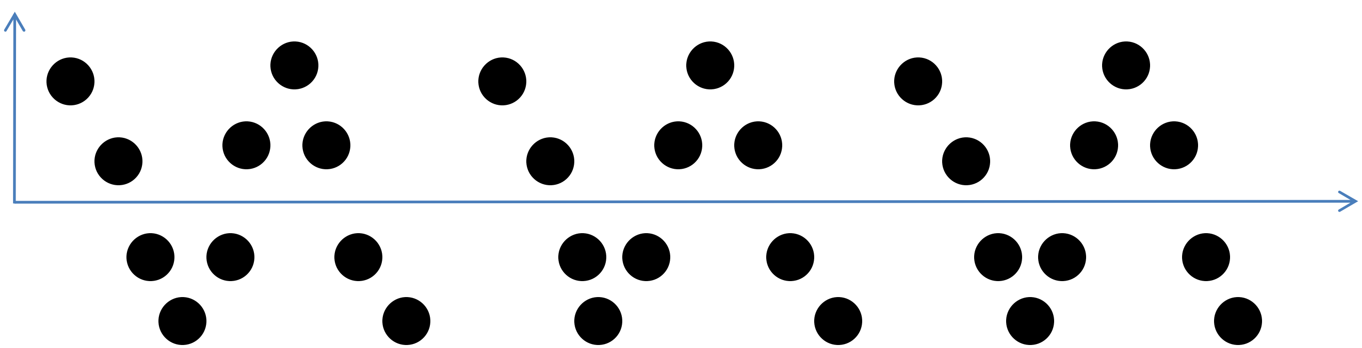
$1/\Delta_{\mathrm{X}}$ and $1/\Delta_{\mathrm{Y}}$ are called the **sampling rates**

Ideal Uniform Rectilinear Sampling

$g_a(x,y)$



$g_s(x,y)$

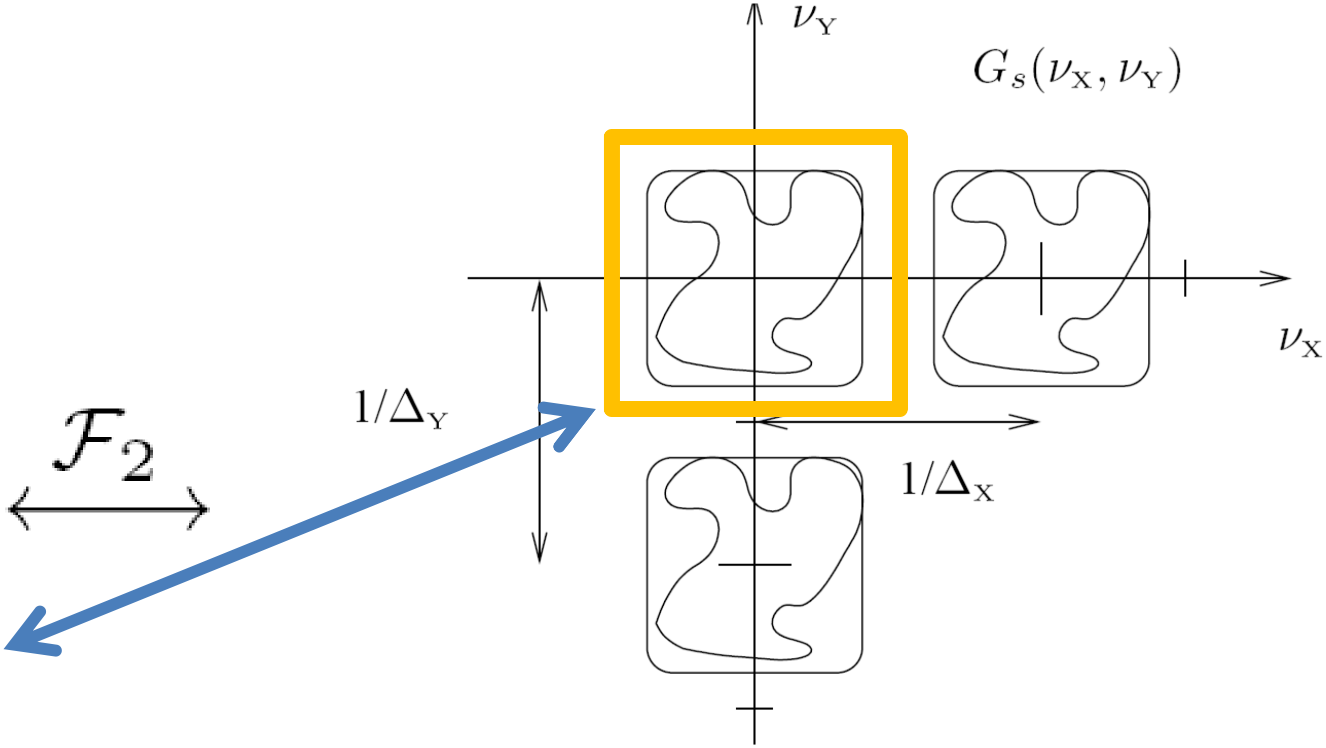


$$g_s(x,y) \quad = \quad \sum_{n=-\infty}^{\infty} \sum_{m=-\infty}^{\infty} g_d[n,m]\, \Delta_X \Delta_Y\, \delta_2(x - n\Delta_X, y - m\Delta_Y)$$

How to recover g_a ?

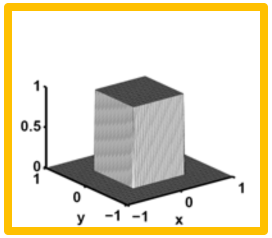
$$g_s(x,y) \overset{\mathcal{F}_2}{\longleftrightarrow} G_s(\nu_X, \nu_Y)$$

$$= \boxed{\sum_{k=-\infty}^{\infty} \sum_{l=-\infty}^{\infty} G_a(\nu_X - k/\Delta_X, \nu_Y - l/\Delta_Y),}$$

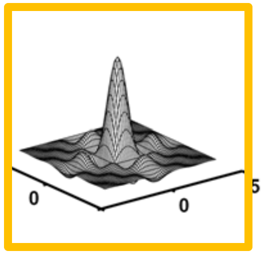


$g_a(x,y,)$

How to recover g_a ?

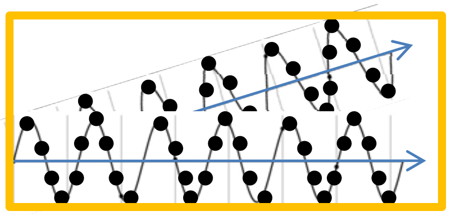


$$\boxed{G_a(\nu_X, \nu_Y) = G_s(\nu_X, \nu_Y)\left[\mathrm{rect}_2(\nu_X \Delta_X, \nu_Y \Delta_Y)\right].}$$



$$g_a(x, y) \quad = \quad g_s(x, y) ** \left[\frac{1}{\Delta_X \Delta_Y} \mathrm{sinc}_2\left(\frac{x}{\Delta_X}, \frac{y}{\Delta_Y}\right)\right]$$

$$= \quad \left[\sum_{n=-\infty}^{\infty} \sum_{m=-\infty}^{\infty} g_d[n, m]\, \Delta_X \Delta_Y\, \delta_2(x - n\Delta_X, y - m\Delta_Y)\right]$$



$$** \left[\frac{1}{\Delta_X \Delta_Y} \mathrm{sinc}_2\left(\frac{x}{\Delta_X}, \frac{y}{\Delta_Y}\right)\right]$$

Sinc interpolation

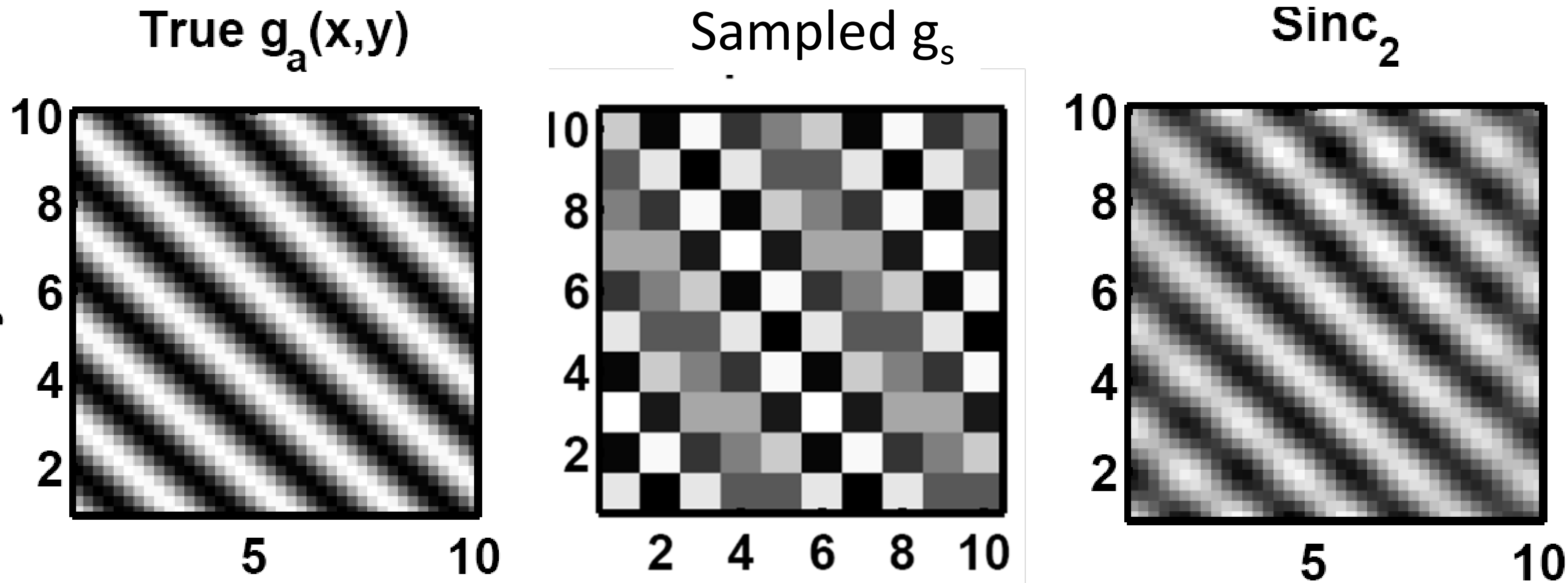
We can recover $g_a(x, y)$ by interpolating the samples $g_d[n,m]$ using sinc functions

$$\boxed{g_a(x,y) = \left[\sum_{n=-\infty}^{\infty}\sum_{m=-\infty}^{\infty} g_d[n,m]\,\Delta_X\Delta_Y\,\delta_2(x-n\Delta_X, y-m\Delta_Y)\right] ** \left[\frac{1}{\Delta_X\Delta_Y}\operatorname{sinc}_2\left(\frac{x}{\Delta_X},\frac{y}{\Delta_Y}\right)\right]}$$

$$= \boxed{\sum_{n=-\infty}^{\infty}\sum_{m=-\infty}^{\infty} g_d[n,m]\operatorname{sinc}_2\left(\frac{x-n\Delta_X}{\Delta_X},\frac{y-m\Delta_Y}{\Delta_Y}\right).}$$

$$\operatorname{sinc}_2\left(\frac{x}{\Delta_X},\frac{y}{\Delta_Y}\right) = \operatorname{sinc}\left(\frac{x}{\Delta_X}\right)\operatorname{sinc}\left(\frac{y}{\Delta_Y}\right)$$

Interpolation



MATLAB's interp2

What’s the problem with sinc interpolation?

- Real world images need not be exactly band-limited.
- Unbounded support
- Summations require infinitely many samples
- Computationally very expensive

Linear interpolation

- sinc interpolation formula

$$g_a(x, y \quad = \quad \boxed{\sum_{n=-\infty}^{\infty} \sum_{m=-\infty}^{\infty} g_d[n, m] \, \mathrm{sinc}_2\left(\frac{x - n\Delta_X}{\Delta_X}, \frac{y - m\Delta_Y}{\Delta_Y} \right).}$$

- Linear interpolation:

$$g_a(x, y) = \sum_{n=-\infty}^{\infty} \sum_{m=-\infty}^{\infty} g_d[n, m] \, h(x - n\Delta_X, y - m\Delta_Y)$$

h(x, y) = interpolation kernel

Why this is a linear interpolation? Linear function of the samples $g_d[n,m]$

Linear interpolation

- Consider alternative linear interpolation schemes that address issues with:
  - Unbounded support
  - Summations require infinitely many samples
  - Computationally very expensive

Basic polynomial interpolation

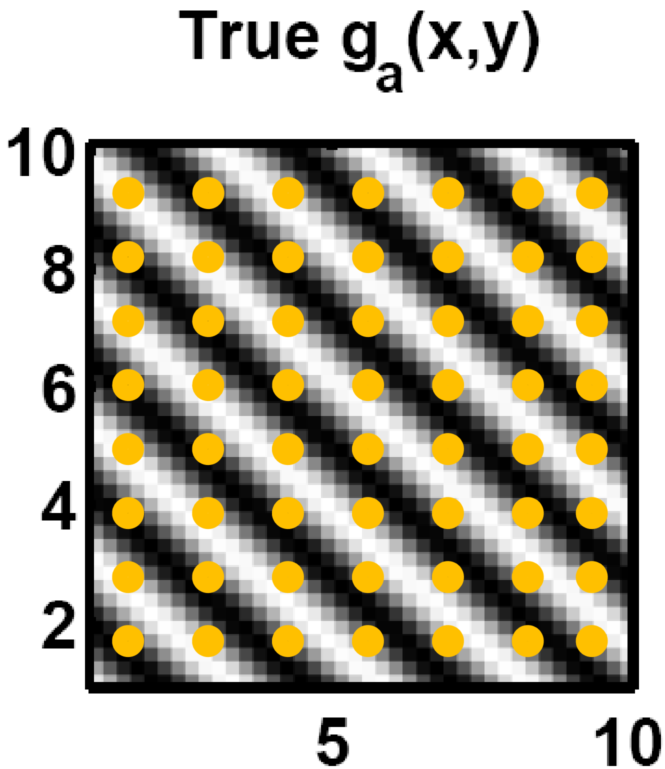
Here kernels that are piecewise polynomials

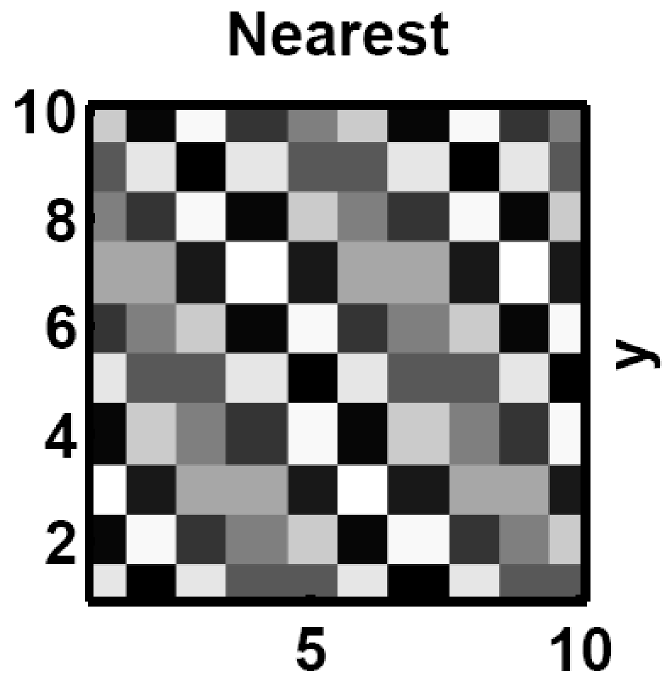
- **Zero-order or nearest neighbor interpolation**
  - Use the value at the nearest sample location
  - Kernels are zero order polynomials

$$h(x) = \mathrm{rect}(x)$$

$$h(x, y) = \mathrm{rect}_2\left(\frac{x}{\Delta_X}, \frac{y}{\Delta_Y}\right) = \mathrm{rect}\left(\frac{x}{\Delta_X}\right)\mathrm{rect}\left(\frac{y}{\Delta_Y}\right)$$

Nearest neighbor interpolation

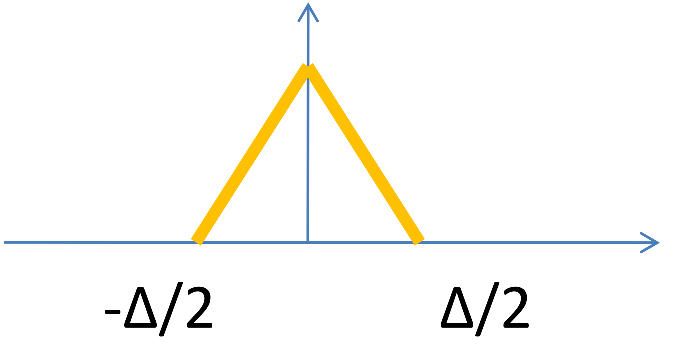




Linear or bilinear interpolation

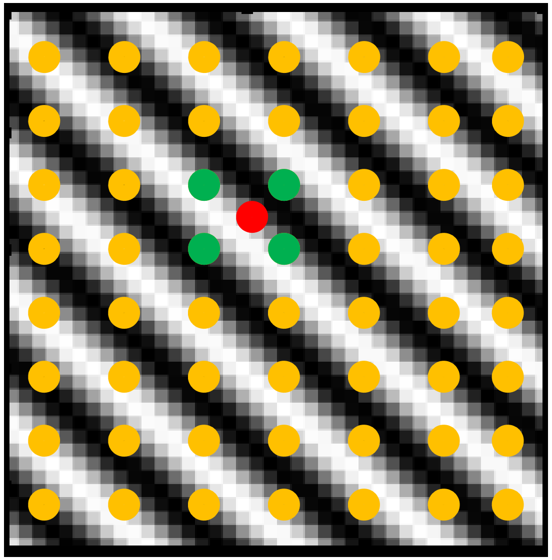
– ***tri*** function is a piecewise linear function of its (spatial) arguments

$$h(x, y) = \mathrm{tri}\left(\frac{x}{\Delta_X}\right)\mathrm{tri}\left(\frac{y}{\Delta_Y}\right)$$



Linear or bilinear interpolation

- For any point (x, y), we find the four nearest sample point $g_d[0,0], g_d[1,0], g_d[0,1], \; g_d[1,1]$
- fit a polynomial of the form $\alpha_0 + \alpha_1 x + \alpha_2 y + \alpha_3 xy$
- Estimate alphas from system of 4 equations in 4 unknowns:



$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 \\ 1 & 1 & 1 & 1 \end{bmatrix} \begin{bmatrix} \alpha_0 \\ \alpha_1 \\ \alpha_2 \\ \alpha_3 \end{bmatrix} = \begin{bmatrix} g_d[0,0] \\ g_d[1,0] \\ g_d[0,1] \\ g_d[1,1] \end{bmatrix}$$

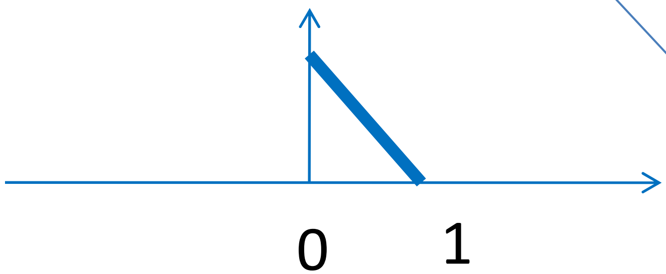
Interpolating formula ($x \in [0, 1] \times [0, 1]$):

$$g_d[0,0](1-x)(1-y) \; + g_d[1,0]\, x(1-y)$$

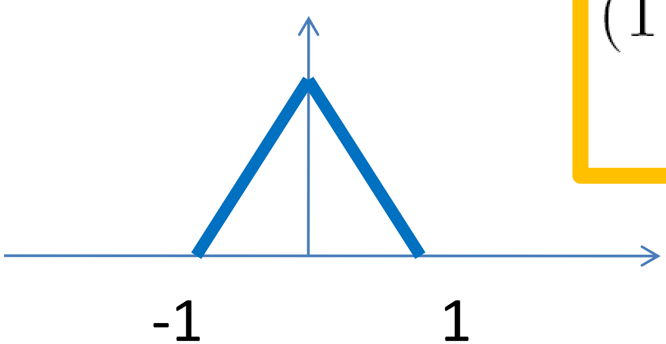
$$+ \, g_d[0,1](1-x)y \; + g_d[1,1]\, xy$$

Linear or bilinear interpolation

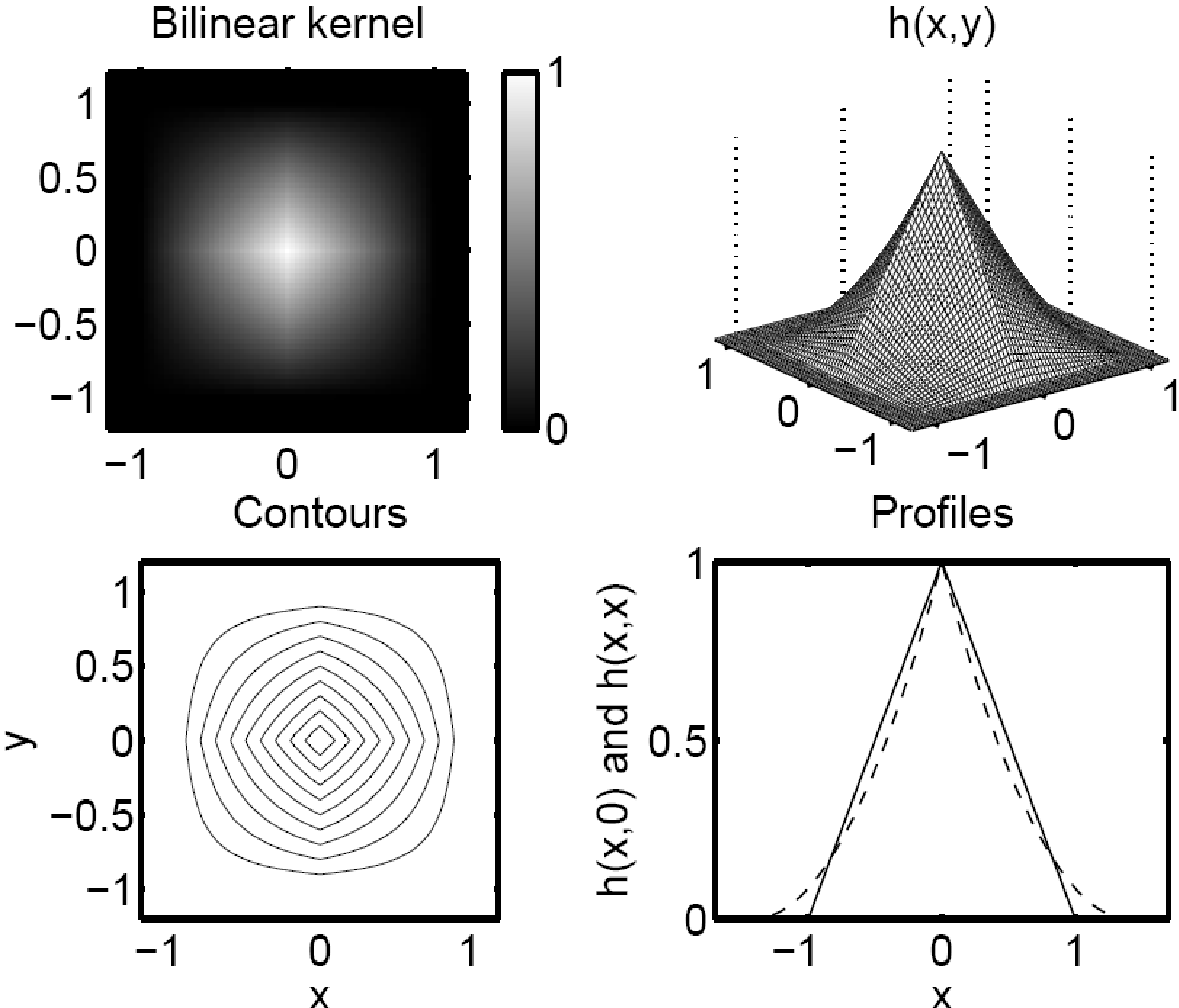
$$g_d[0,0]\boxed{(1-x)(1-y)} + g_d[1,0]\,x(1-y) + g_d[0,1](1-x)y + g_d[1,1]\,xy$$



If x∈ [-1, 1] × [-1, 1]:

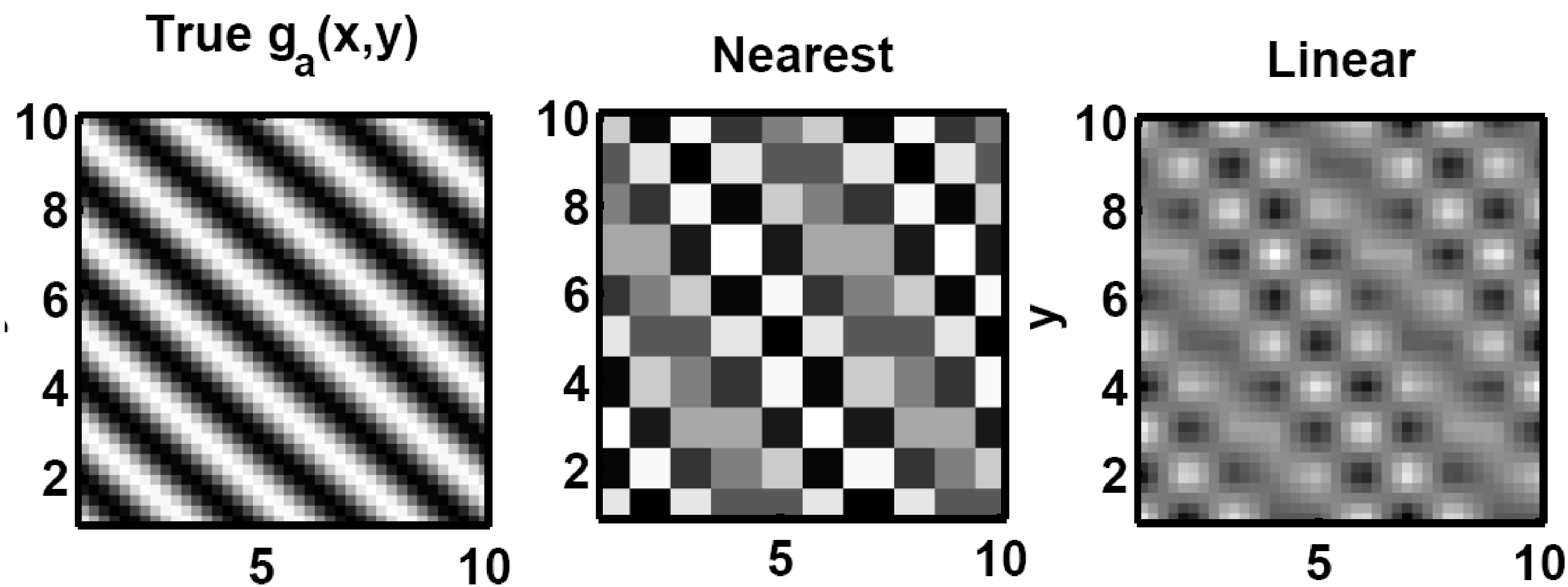


$$\boxed{\begin{aligned}&(1-|x|)(1-|y|)\,\mathrm{rect}(x/2)\,\mathrm{rect}(y/2)\\&= \mathrm{tri}(x)\,\mathrm{tri}(y)\end{aligned}}$$

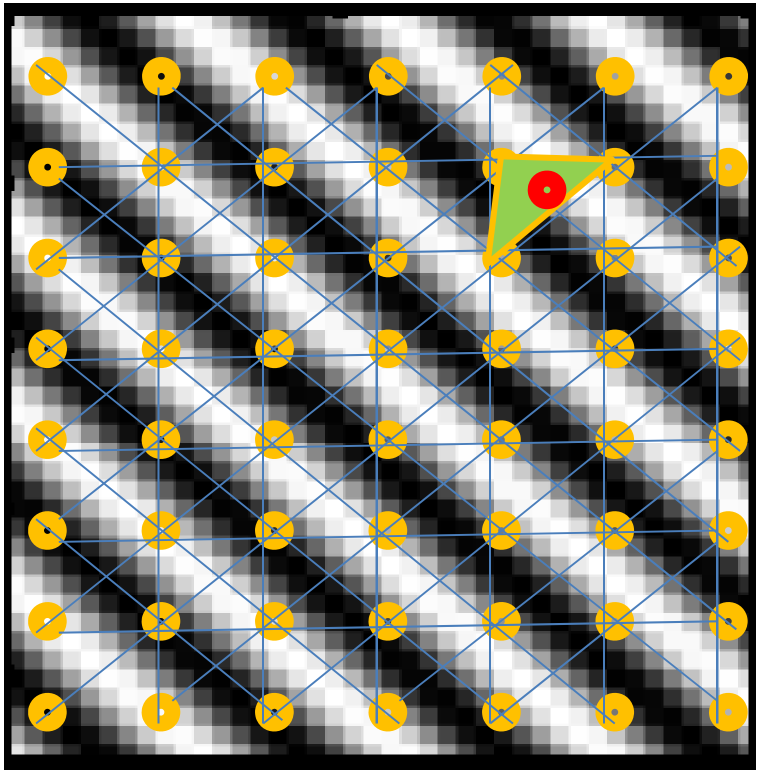


Bilinear kernel
h(x,y)
Contours
Profiles
1
0.5
0
−0.5
−1
−1
0
1
1
0
x
y
h(x,0) and h(x,x)

Linear or bilinear interpolation

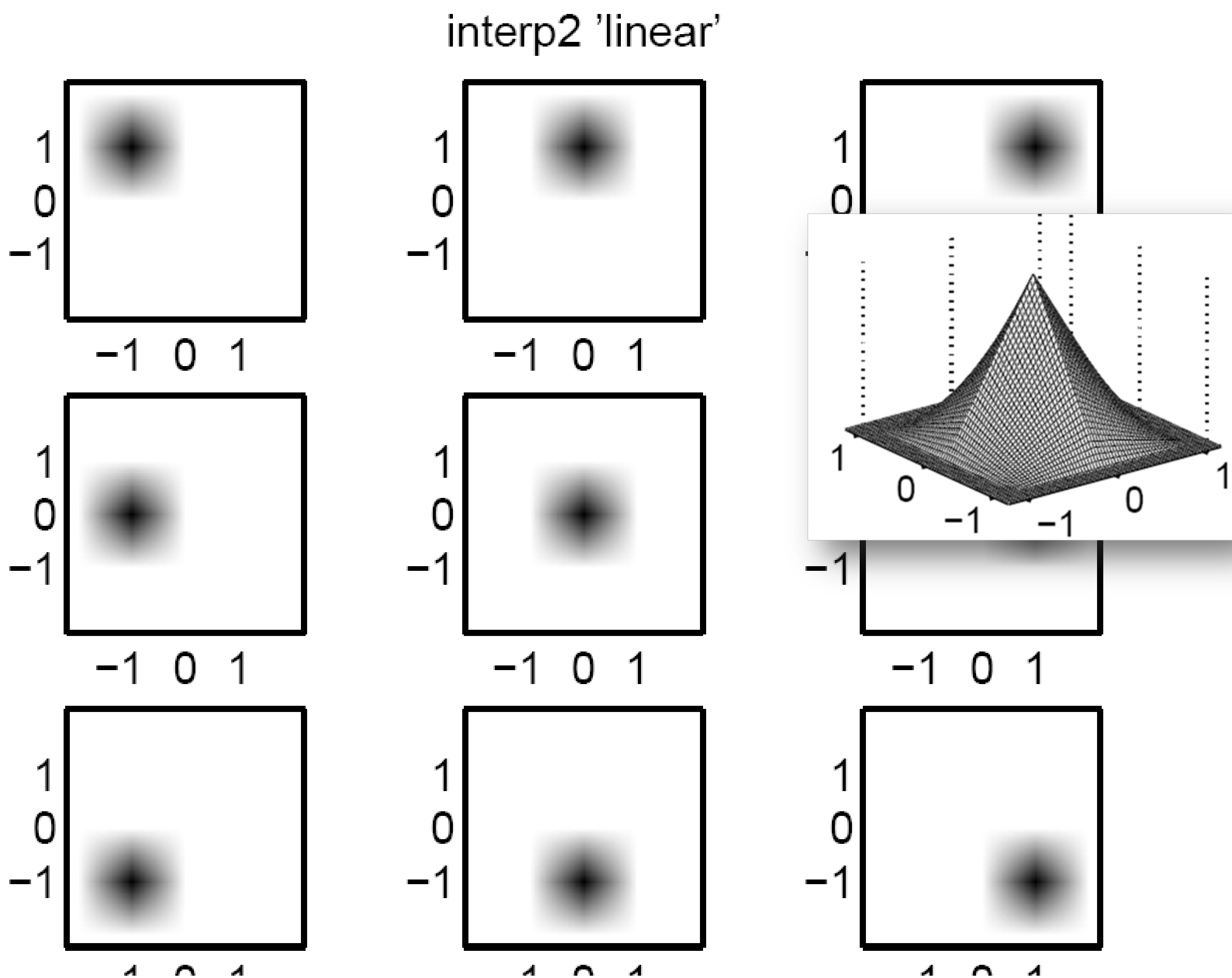


Linear interpolation by Delaunay triangulation

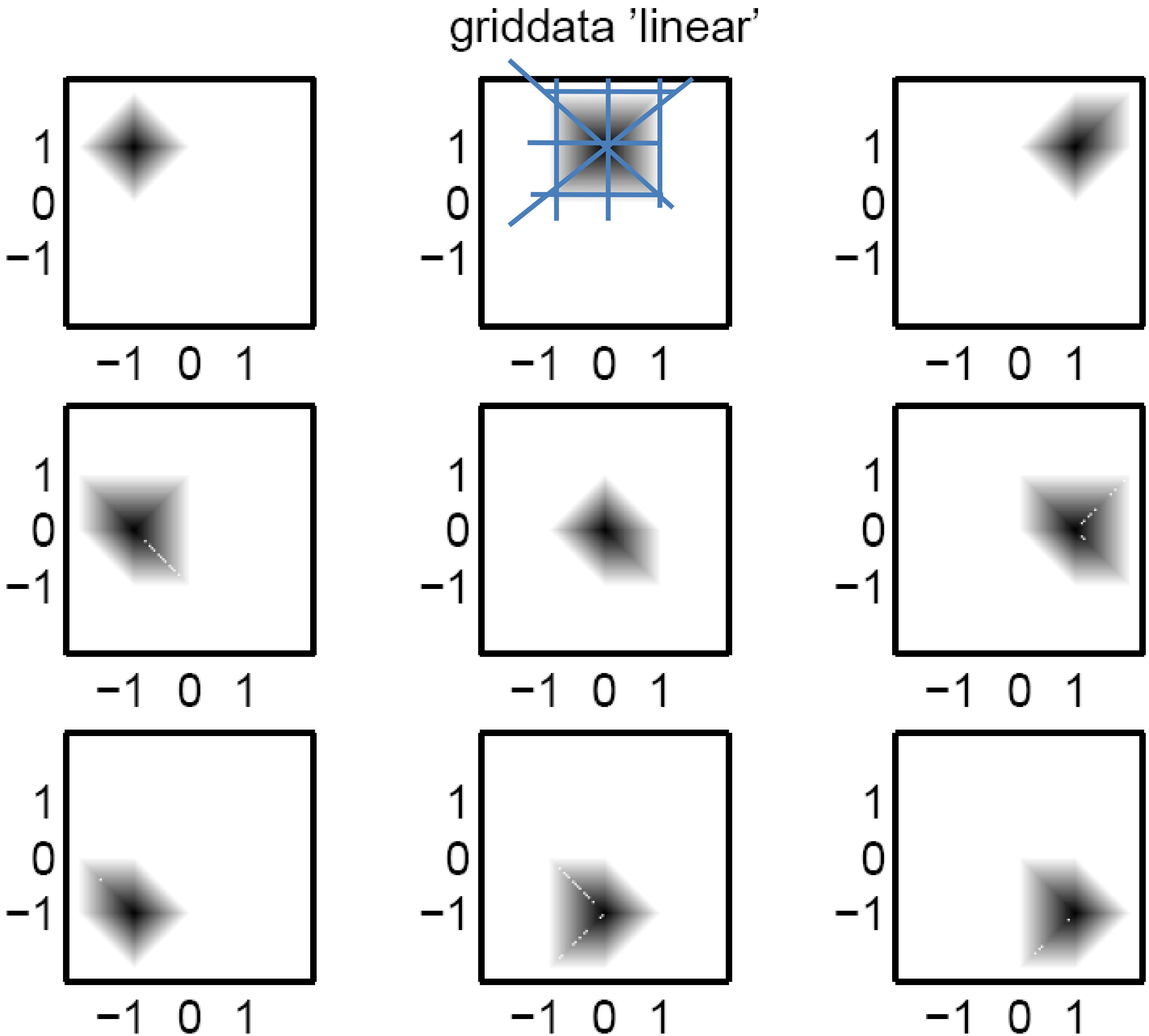


Use a linear function within each triangle (three points determine a plane).

MATLAB's `griddata` with the 'linear' opt



interp2 'linear'
1
0
−1
−1 0 1


griddata 'linear'
1
0
−1
−1 0 1

What's the problem zero-order or bilinear interpolation?

- neither the zero-order or bilinear interpolator are differentiable.
- They are not continuous and smooth

Desirable properties of interpolators

- "self consistency": $g_a(x)\Big|_{x=n\Delta} = g_a(n\Delta) = g_d[n]$

- Continuous and smooth (differentiable):

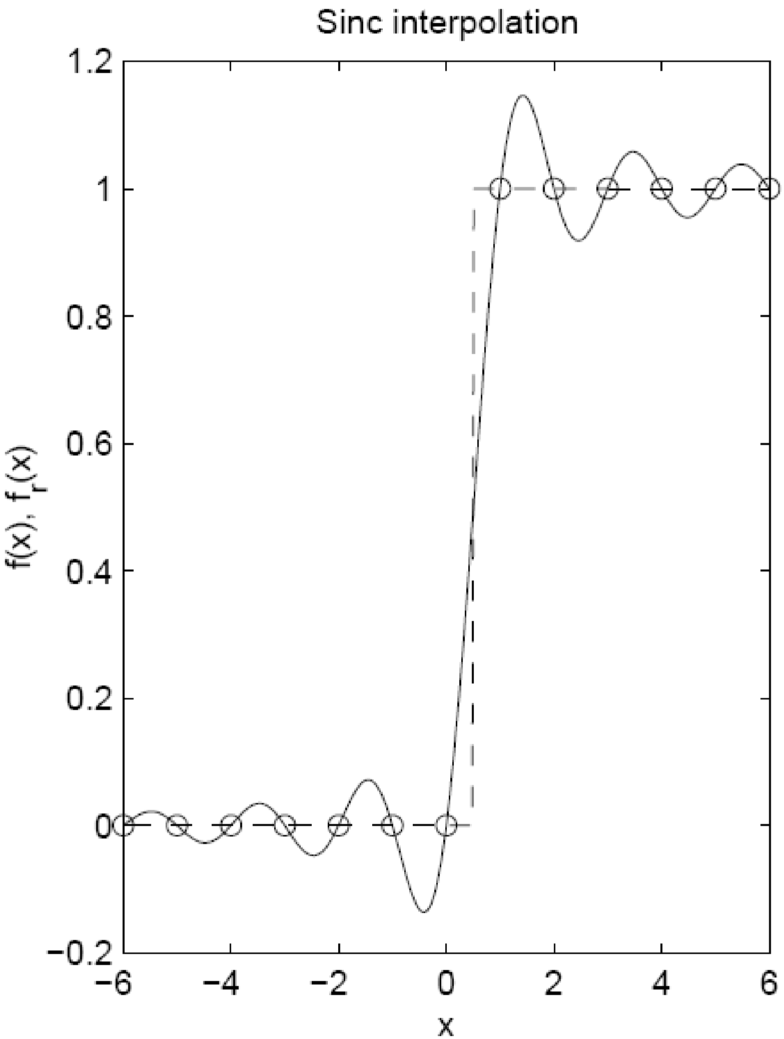
$$\frac{d}{dx}\, g_a(x) = \sum_{n=-\infty}^{\infty} g_d[n]\, \frac{d}{dx}\, h(x - n\Delta)$$

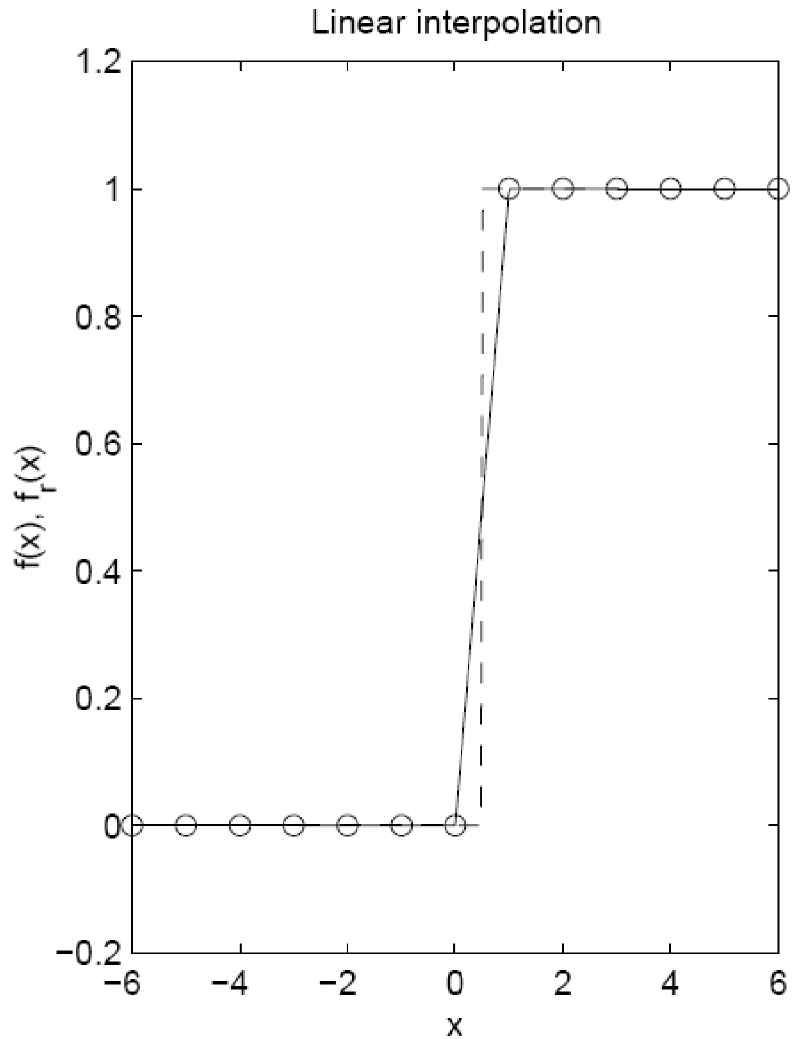
- Short spatial extent to minimize computation

$$g_a(x) = \sum_{n=-\infty}^{\infty} g_d[n]\, h(x - n\Delta) = \sum_{\{n \in \mathbb{Z}\, :\, x - n\Delta \in \mathcal{S}\}} g_d[n]\, h(x - n\Delta)$$

Desirable properties of interpolators

- Frequency response approximately rect().
- symmetric
- Shift invariant
- Minimum sidelobes to avoid ringing “artifacts”

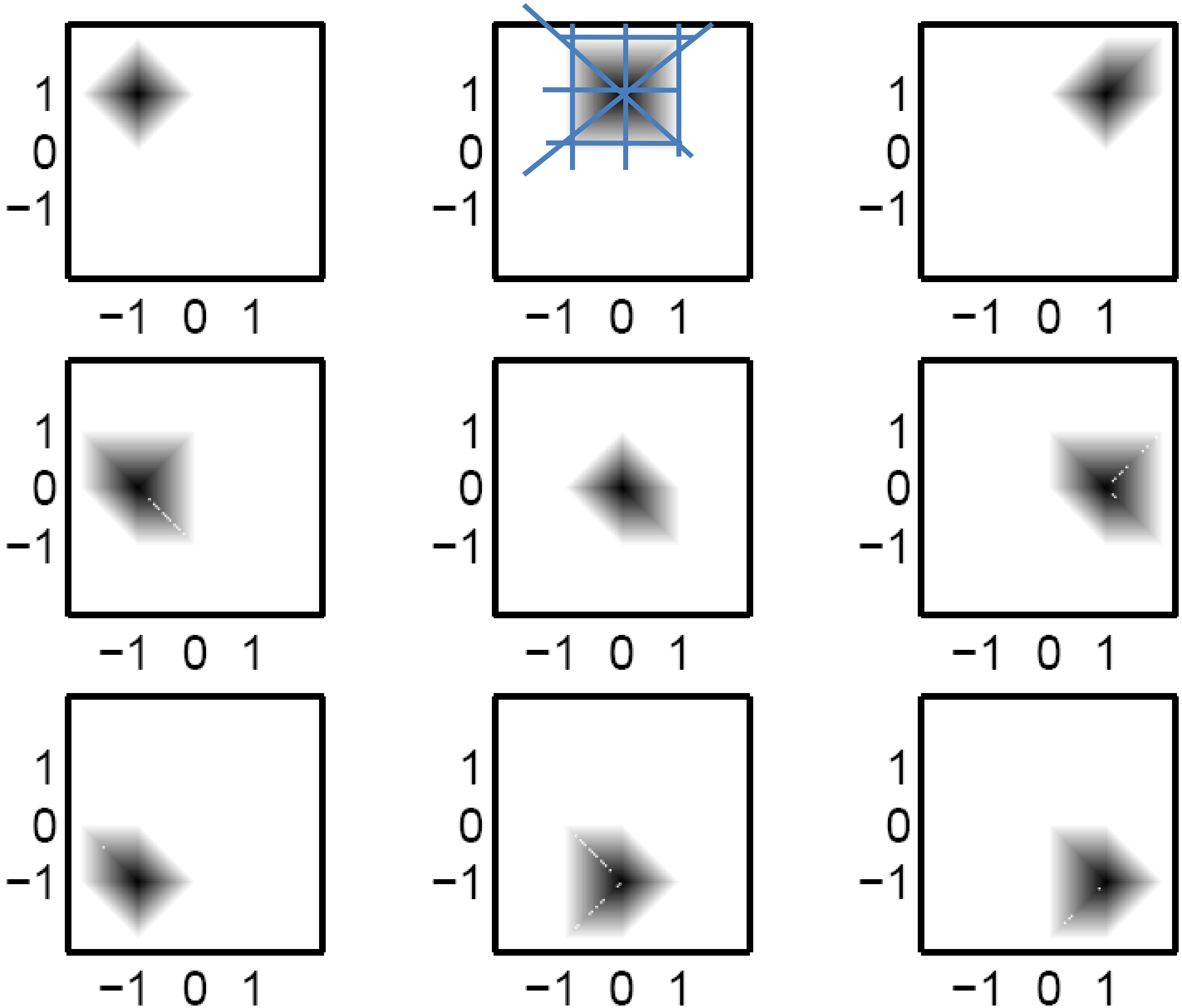




Continuous and smooth
Large sidelobes

Non smooth
Small sidelobes

griddata 'linear'



Non smooth; non shift invariant

Polynomial interpolation

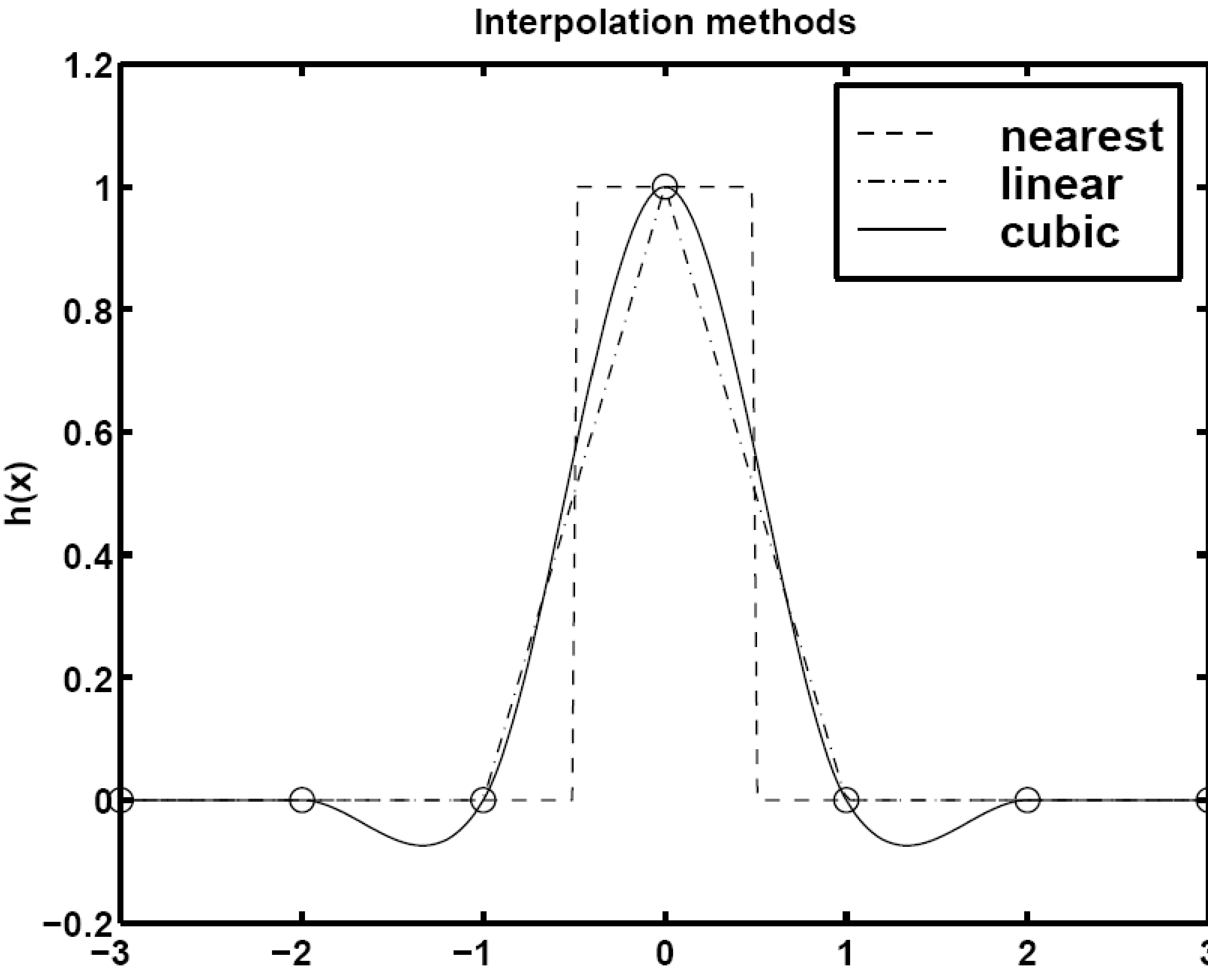
– Cubic

interpolation kernel:

$$h(x) = \begin{cases} 1 - \frac{5}{2}\left|x\right|^2 + \frac{3}{2}\left|x\right|^3, & \left|x\right| \leq 1 \\ 2 - 4\left|x\right| + \frac{5}{2}\left|x\right|^2 - \frac{1}{2}\left|x\right|^3, & 1 < \left|x\right| < 2 \\ 0, & \text{otherwise.} \end{cases}$$

– Quadratic

– B-splines

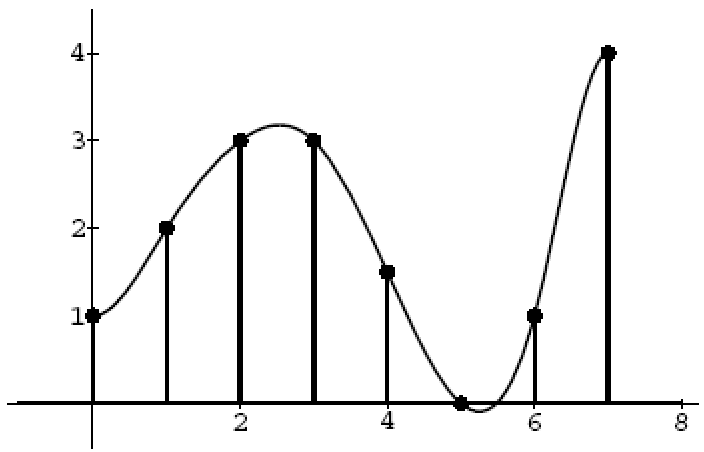

Interpolation methods
nearest
linear
cubic
h(x)
1.2
1
0.8
0.6
0.4
0.2
0
−0.2
−3
−2
−1
0
1
2
3

B-spline interpolation

- Motivation:
  - n-order differentiable
  - Short spatial extent to minimize computation

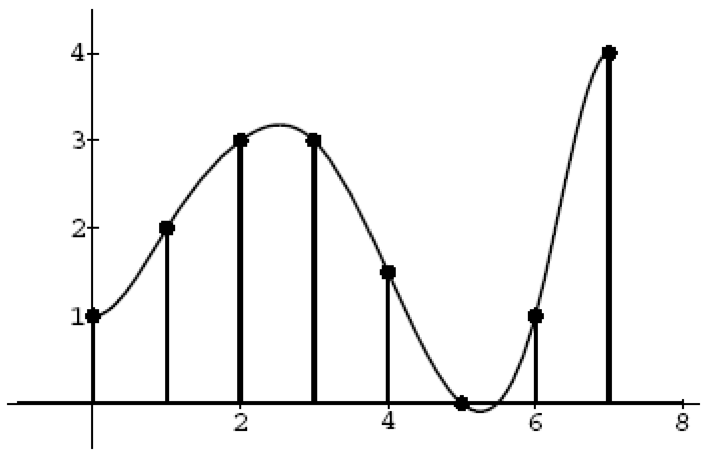
Background

- Splines are piecewise polynomials with pieces that are smoothly connected together
- The joining points of the polynomials are called ***knots***



Background

- For a spline of degree *n, each segment is a* polynomial of degree *n*
  - Do i need n+1 coefficient to describe each piece?



- Additional smoothness constraint imposes the continuity of the spline and its derivatives up to order (n-1) at the knots
- only one degree of freedom per segment!

B-spline expansion

- Splines are uniquely characterized in terms of a B-spline expansion

$$s(x) = \sum_{k \in Z} c(k)\beta^n(x-k)$$

integer shifts of the central B-spline of degree *n*

$\beta^n(x)$ = central B-spline = basis (B) spline

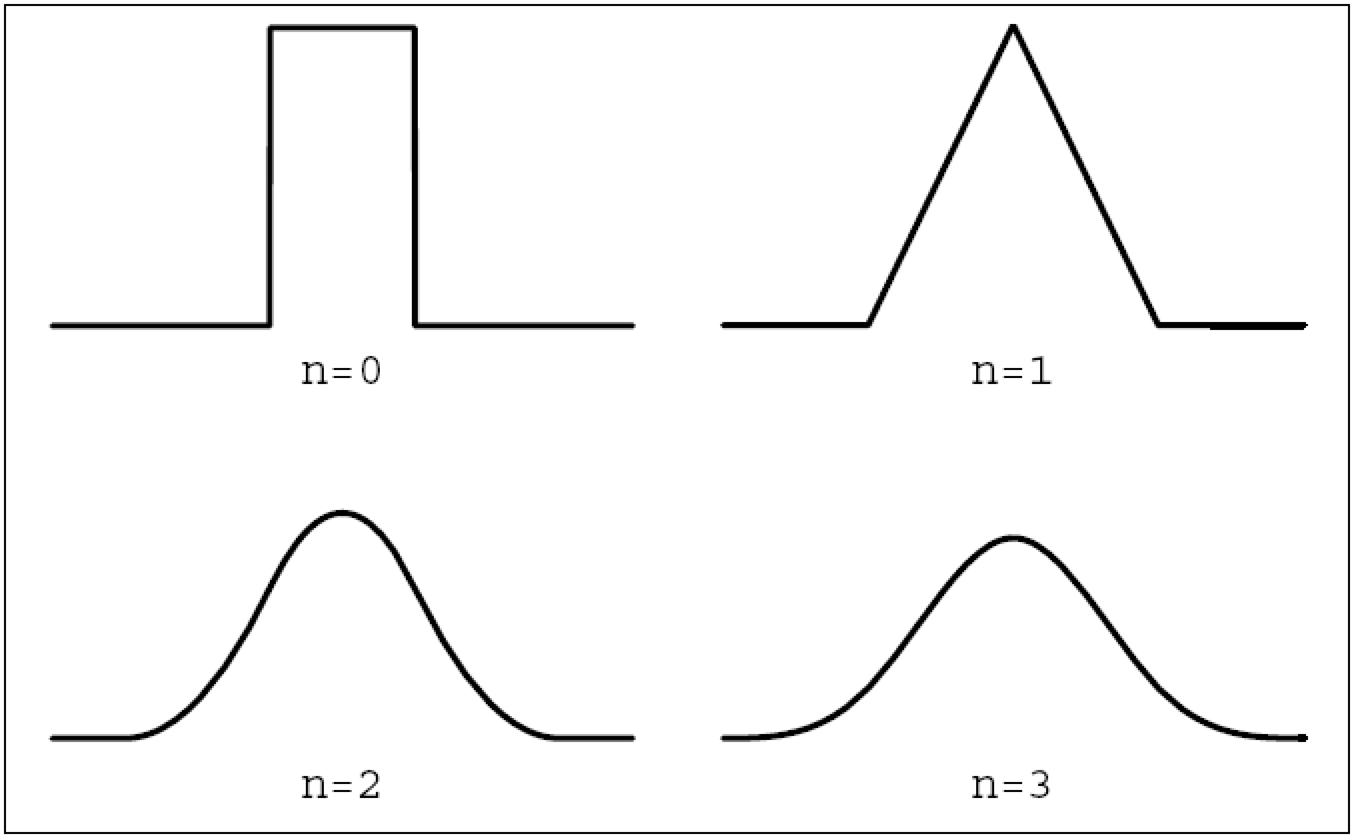
B-splines

- Symmetrical
- bell-shaped functions
- constructed from the *(n+1)-fold convolution*

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

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

B-splines



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

B-splines

- n-order B-spline:

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

$$(x)_+^n = \begin{cases} x^n, & x \geq 0 \\ 0, & x < 0 \end{cases}$$

- Derived using:

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

B-spline

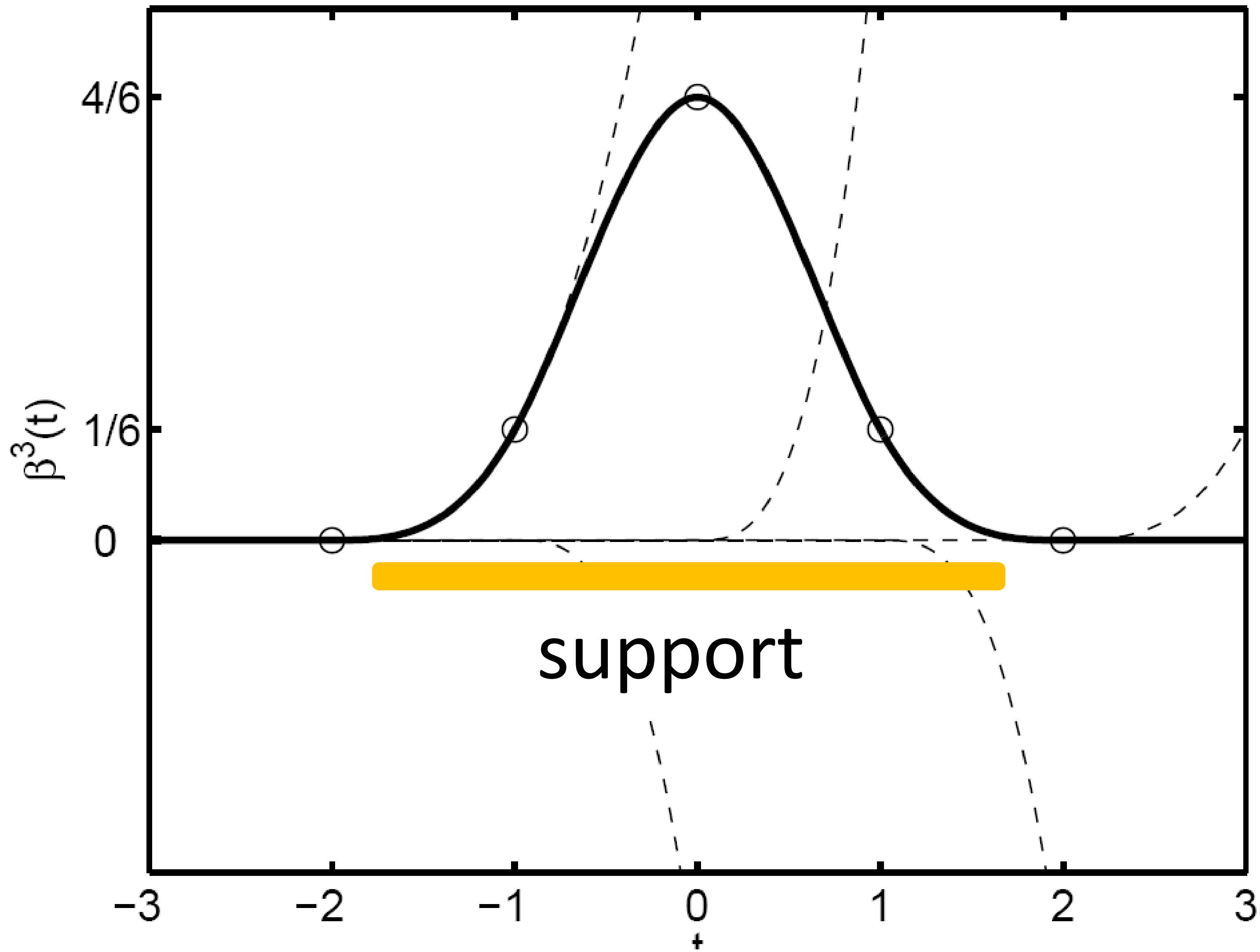
- Using cubic B-splines is a popular choice:

$$\beta^{(3)}(x) = \frac{1}{6}\left[(x+2)_+^3 - 4(x+1)_+^3 + 6(x)_+^3 - 4(x-1)_+^3 + (x-2)_+^3\right]$$

$$\beta^3(x) = \begin{cases} 2/3 - |x|^2 + |x|^3/2, & 0 \le |x| < 1 \\ (2-|x|)^3/6, & 1 \le |x| < 2 \\ 0, & 2 \le |x|. \end{cases}$$

- Important: compactly supported!

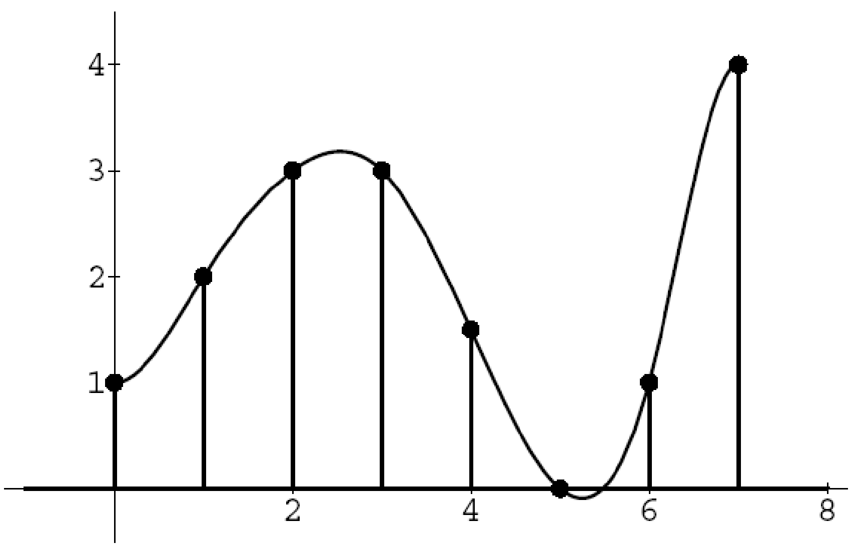
Cubic B–spline and its underlying components

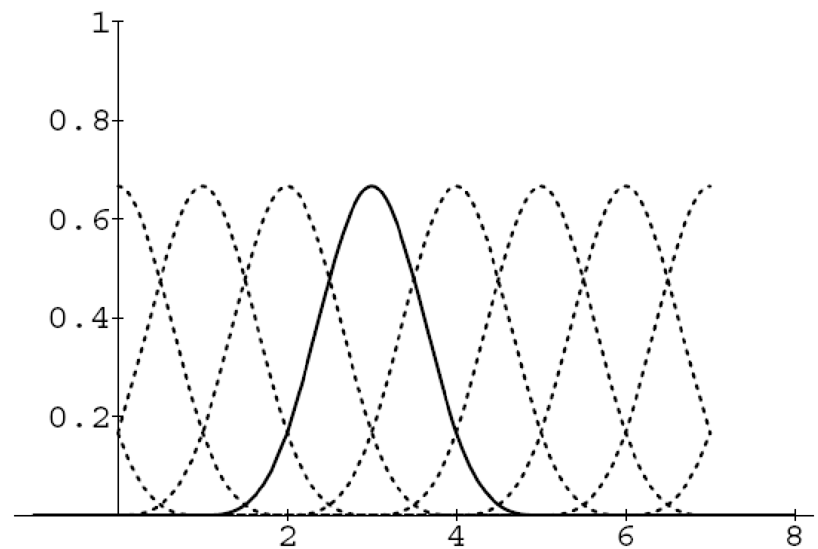


B-spline expansion

$$s(x) = \sum_{k \in Z} c(k) \beta^n (x - k)$$

Each spline is unambiguously characterized by its sequence of B-spline coefficients *c(k)*





Properties of B-spline expansion

- Discrete signal representation! *(even* though the underlying model is a continuous representation).

- Easy to manipulate;
- E.g., derivatives:

$$\frac{d\beta^n(x)}{dx} = \beta^{n-1}\left(x + \tfrac{1}{2}\right) - \beta^{n-1}\left(x - \tfrac{1}{2}\right)$$

B-spline interpolation

- So far: B-spline model of a given input signal s(x)

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

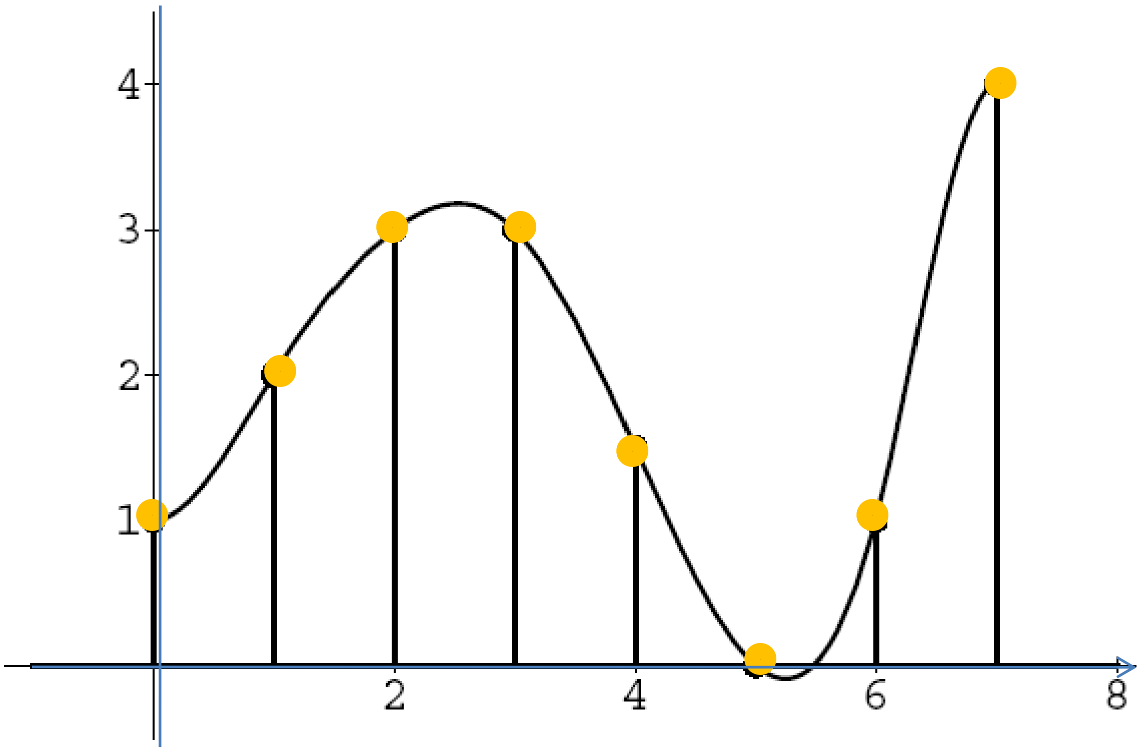
$\boxed{s_d[n]}$

Interpolation **problem**: find coefficients c(k) such that spline function goes through the data points exactly

That is, reconstruct signal using a spline representation!

Reconstruct signal using a spline representation

$$s(x) = \sum_{k \in Z} s_d[n] \beta^n (x - k)$$



B-spline interpolation

$$s(x) = \sum_{k \in Z} c(k) \beta^n (x - k)$$

- Relationship between coefficients and samples

$$c(k) = (b_1^n)^{-1} * s_d[n]$$

$$b_m^n(k) = \beta^n(x/m)\Big|_{x=k}$$ = discrete B-spline kernel

Obtained by sampling the B-spline of degree *n expanded by a* factor of *m* (m=1)

Cardinal B-spline interpolation

$$s(x) = \sum_{k \in Z} c(k) \beta^n (x - k)$$

$$s(x) = \sum_{k \in Z} \left((b_1^n)^{-1} * s \right)(k) \beta^n (x - k)$$

$$= \sum_{k \in Z} s(k) \sum_{l \in Z} (b_1^n)^{-1}(l) \beta^n (x - l - k)$$

$$= \sum_{k \in Z} s(k) \boxed{\eta^n (x - k)}$$

Cardinal splines

2D splines in images

- tensor-product basis functions

$$f(x,y) = \sum_{k=k_1}^{(k_1+K-1)} \sum_{l=l_1}^{(l_1+K-1)} c(k,l)\beta^n(x-k)\beta^n(y-l)$$

B-spline interpolation

- **Conclusion:**
  - n-order differentiable
  - Short spatial extent to minimize computation

General image spatial transformations

- transform the spatial coordinates of an image $f(x, y)$ so that (after being transformed), it better "matches" another image
  - Ex: warping brain images into a standard coordinate system to facilitate automatic image analysis

image spatial transformations

General image spatial transformations

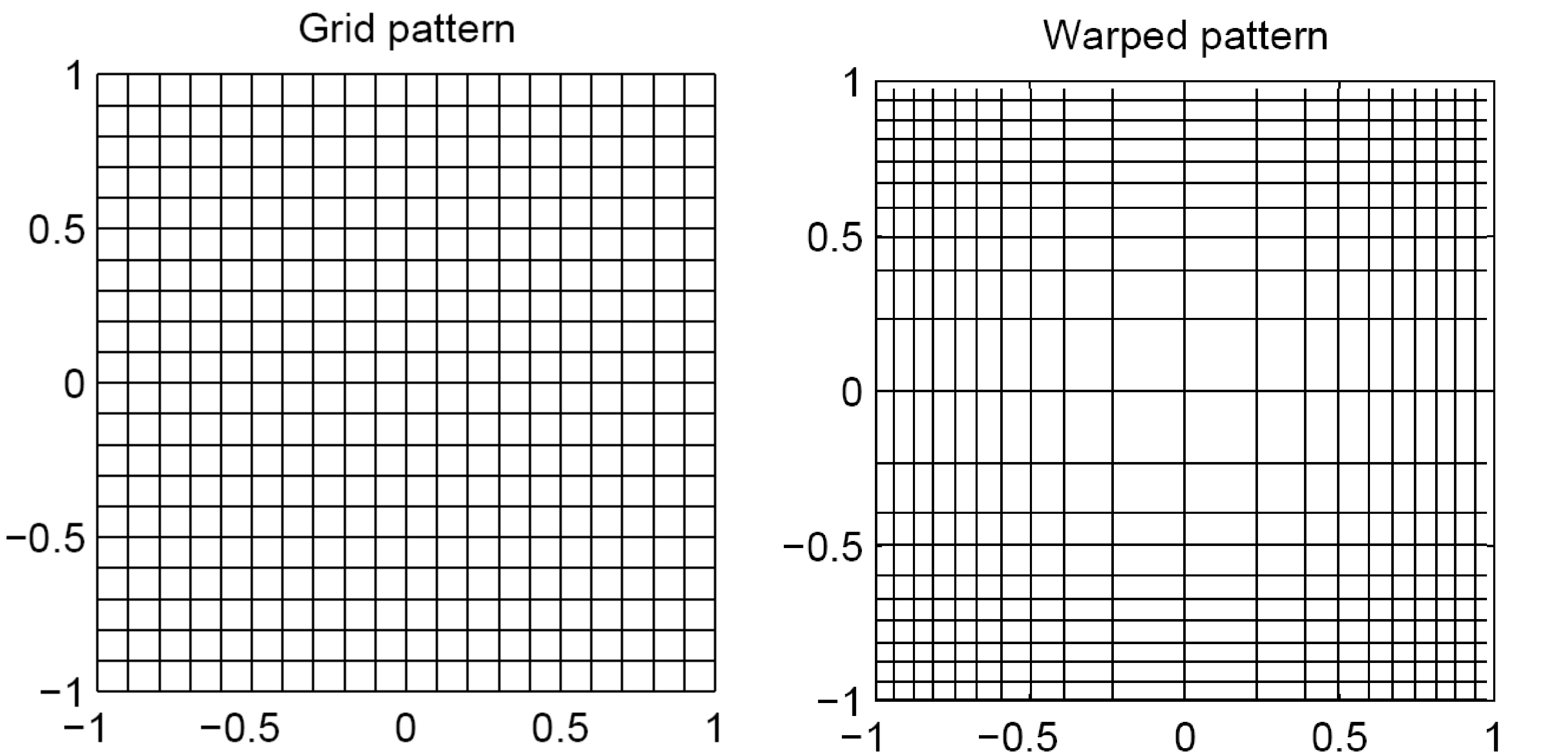
Form of transformations:

- Shift only $g(x, y) = f(x - x_0, y - y_0)$
- Affine $g(x, y) = f(ax + by, cx + dy)$
- **thin-plate splines** (describes smooth warpings using control points)
- General **spatial transformation** (*e.g., morphing or warping an image)*

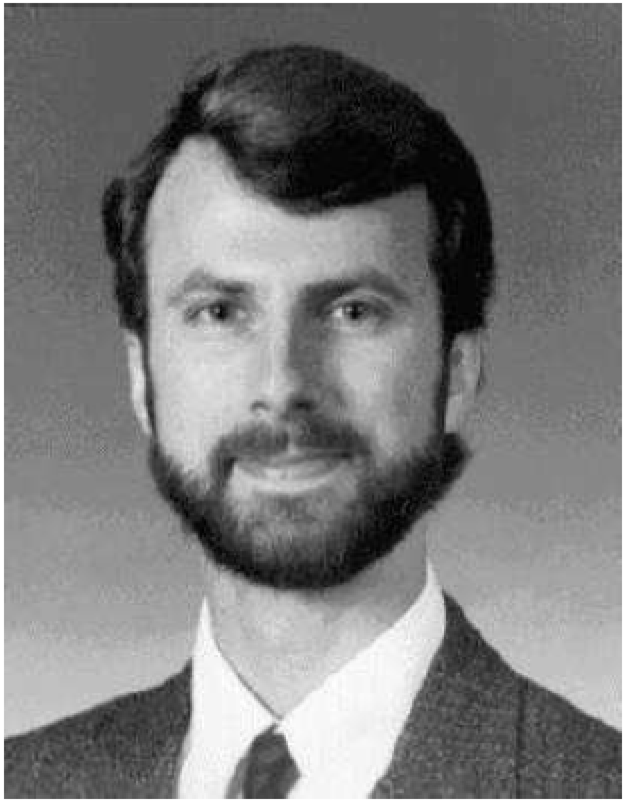
$$g(x, y) = f(T_{\mathrm{X}}(x, y), T_{\mathrm{Y}}(x, y))$$

MATLAB's interp2 or griddata functions

Interpolation is critical!



A professor



A warped professor