Lecture 9:
Image Interpolation

Contents

Motivation

What is Interpolation?

- recovery of continuous data from *exact* discrete data *inside* the data interval
- inverse step to sampling, where discrete data are created from continuous ones (see Lectures 1 and 5)
- always involves assumptions on the data, e.g. smoothness or band limitation (although these assumptions are not always stated explicitly)
- must be distinguished from
  - *extrapolation:* uses a model *outside* the given data interval; example: weather forecast
  - *approximation:* model does not reproduce the given discrete data exactly; example: regression curve through noisy data
- important topic that is not treated adequately in most text books

Motivation (2)

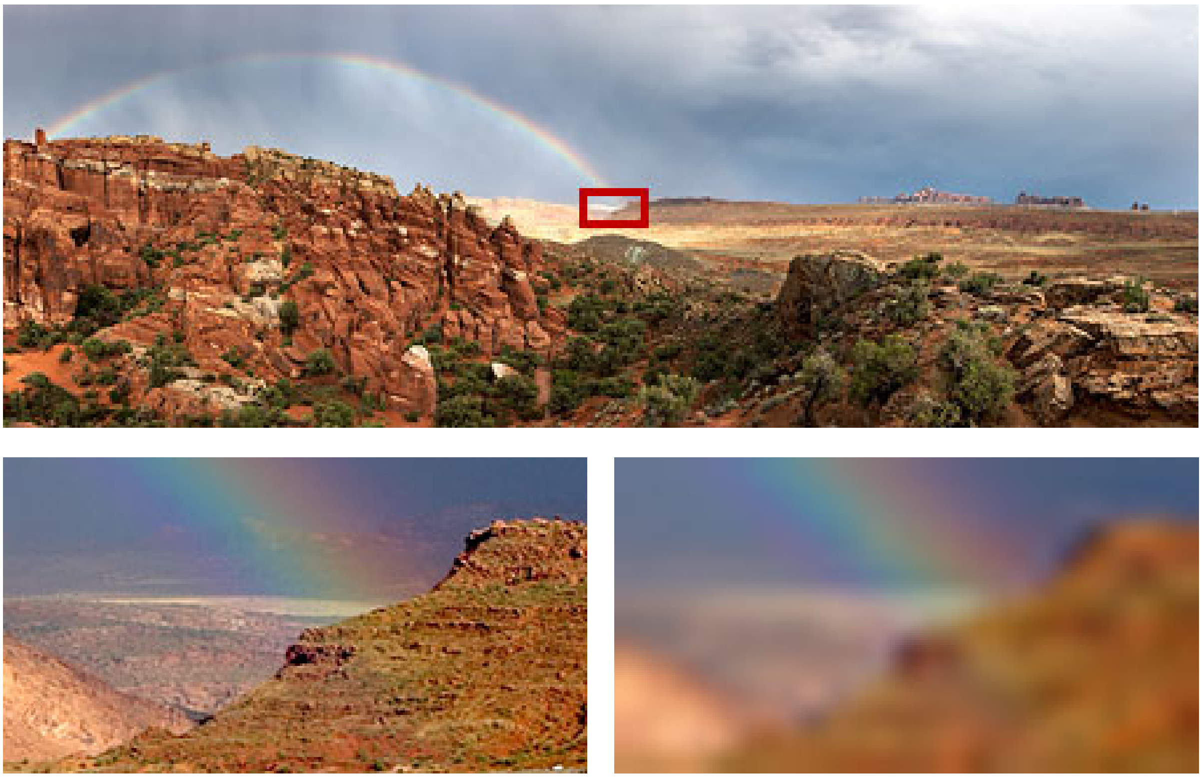
M I A
1 2
3 4
5 6
7 8
9 10
11 12
13 14
15 16
17 18
19 20
21 22
23 24
25 26
27 28
29 30

Where is Interpolation Necessary ?

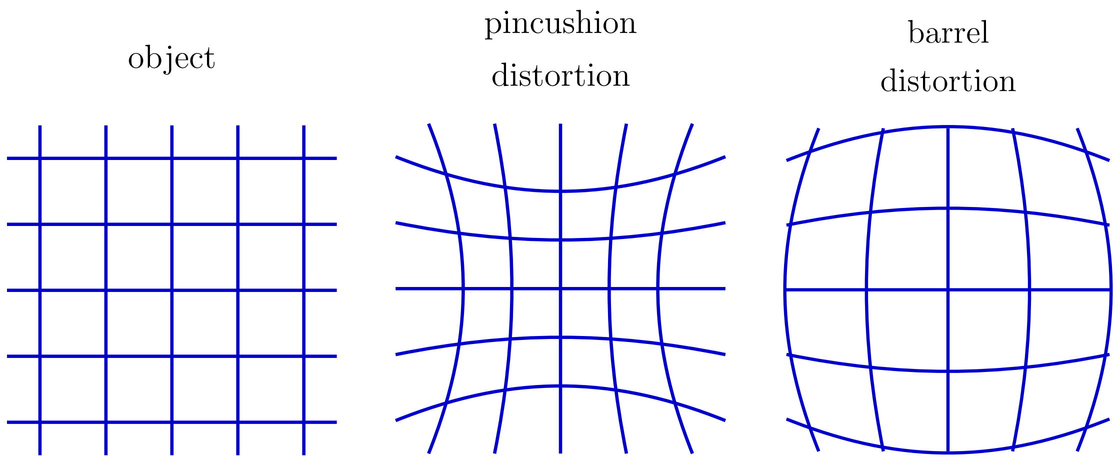
- rescaling, zooming:
  - increase the apparent resolution of images
  - examples: "digital zooms" in cameras, video upscaling
- reslicing:
  - display a (medical) 3-D data set along arbitrary planes that do not coincide with the measured planes along the axis directions
- all kinds of geometric transformations, e.g.
  - image translation with subpixel precision
  - image rotation
  - warping in order to compensate for motion in image sequences
  - registration of medical images
  - ultrasound scan conversion from polar to cartesian coordinates
  - compensation of pincushion and barrel lens distortions (kissen- und tonnenförmige Verzeichnungen)

Motivation (3)

M I A
1 2
3 4
5 6
7 8
9 10
11 12
13 14
15 16
17 18
19 20
21 22
23 24
25 26
27 28
29 30



Top: Original image. **Bottom left:** Optical zoom with a factor 10. **Bottom right:** "Digital zoom". Source: `http://www.cambridgeincolour.com/tutorials/image-interpolation.htm`.



Most optical lens systems – in particular so-called "super zooms" – suffer from pincushion or barrel distortions. Compensating these distortions in digital images requires interpolation methods. Modern cameras can correct this automatically with their interpolation firmware. Author: M. Mainberger.

Basic Structure of Classical Interpolation

Classical Interpolation

- Consider an infinitely extended (e.g. by reflection) m-dimensional discrete signal $(f_{\boldsymbol{k}})$ on an equidistant grid with $\boldsymbol{k} = (k_1, k_2, ..., k_m)^\top \in \mathbb{Z}^m$ (grid size 1).

  Example for $m = 2$: image $(f_{i,j})$ with $(i,j)^\top \in \mathbb{Z}^2$.

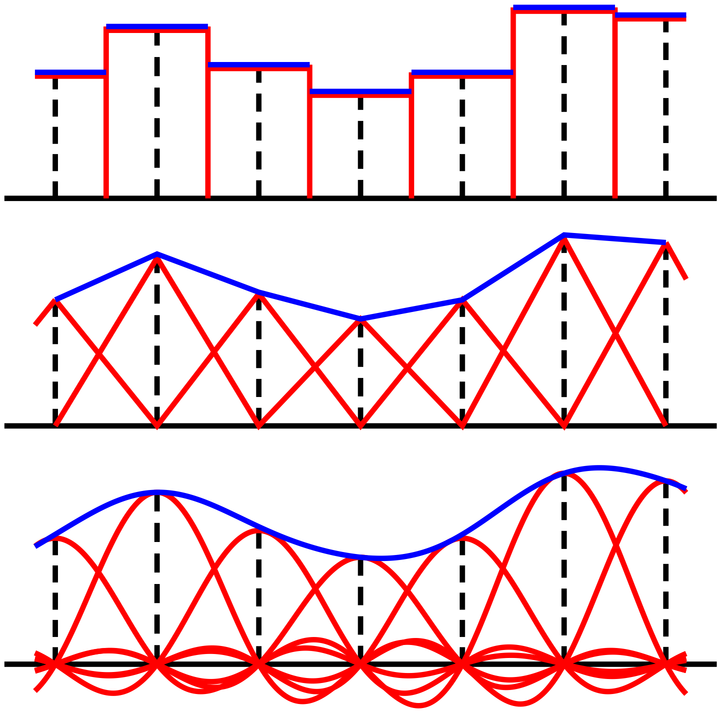
- Compute a continuous signal $u(\boldsymbol{x})$ as a weighted average of the discrete samples $\{f_{\boldsymbol{k}} \,|\, \boldsymbol{k} \in \mathbb{Z}^m\}$:

$$u(\boldsymbol{x}) = \sum_{\boldsymbol{k} \in \mathbb{Z}^m} f_{\boldsymbol{k}} \, \varphi_{\text{int}}(\boldsymbol{x} - \boldsymbol{k})$$

  for all $\boldsymbol{x} = (x_1, x_2, ..., x_m)^\top \in \mathbb{R}^m$.

- The only remaining freedom lies in the *synthesis function* $\varphi_{\text{int}} : \mathbb{R}^m \to \mathbb{R}$. It determines the weights.

Basic Structure of Classical Interpolation (2)



Different synthesis functions (red) and their interpolant (blue). The discrete data are drawn in black. **Top:** Nearest neighbour interpolation uses box functions as synthesis functions. **Middle:** Linear interpolation uses hat functions. **Bottom:** Interpolation with sinc functions. Author: M. Mainberger.

Basic Structure of Classical Interpolation (3)

A Popular Condition for Synthesis Functions

- **Interpolation Condition**

  With our interpolation formula

$$u(\boldsymbol{x}) = \sum_{\boldsymbol{k}\in\mathbb{Z}^m} f_{\boldsymbol{k}}\, \varphi_{\text{int}}(\boldsymbol{x}-\boldsymbol{k})$$

  the interpolation condition $u(\boldsymbol{n}) = f_{\boldsymbol{n}}$ for all $\boldsymbol{n} \in \mathbb{Z}^m$ is trivial to guarantee if

$$\varphi_{\text{int}}(\boldsymbol{x}) = \begin{cases} 1 & \text{for } \boldsymbol{x} = (0,...,0)^\top, \\ 0 & \text{for } \boldsymbol{x} \in \mathbb{Z}^m \setminus \{(0,...,0)^\top\}. \end{cases}$$

  This condition fixes φ_{int} at integer arguments, but not inbetween.
  For non-integer arguments, $\varphi_{\text{int}}(\boldsymbol{x})$ can be tuned to specific applications.

- This condition is sufficient for interpolation, but not necessary.

- Synthesis functions satisfying this interpolation condition are called *interpolating synthesis functions.*

Desirable Properties of the Synthesis Function

Separability

$$\varphi_{\text{int}}(\boldsymbol{x}) = \prod_{i=1}^{m} \tilde{\varphi}_{\text{int}}(x_i) \qquad \text{for all } \boldsymbol{x} = (x_1, x_2, ..., x_m)^\top \in \mathbb{R}^m$$

- allows simple and efficient m-D implementations using 1-D interpolations
- equal treatment of all axes

Symmetry

$$\varphi_{\text{int}}(\boldsymbol{x}) = \varphi_{\text{int}}(-\boldsymbol{x}) \qquad \text{for all } \boldsymbol{x} \in \mathbb{R}^m$$

- equal treatment of opposite sides
- weaker requirement than rotation invariance

Partition of Unity

$$\sum_{\boldsymbol{k} \in \mathbb{Z}^m} \varphi_{\text{int}}(\boldsymbol{x} - \boldsymbol{k}) = 1 \qquad \text{for all } \boldsymbol{x} \in \mathbb{R}^m$$

- guarantees that a constant signal is reproduced exactly

M I A
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30

How Can One Judge the Quality of a Good Interpolant?

Small Support Region of the Synthesis Function

- size of the region where φ_{int} does not vanish *(support region)* should be small
- allows fast computation involving only a few neighbours

High Approximation Order

- Let us interpolate samples of a smooth function f. Let the grid size h go to 0. If the error between the interpolant u and f is $\mathcal{O}(h^p)$, then p is called *approximation order*.
- Larger p indicate better approximation qualities (faster error decay).
- equivalent to the exact reproduction of all polynomials of degree $\leq p-1$

High Regularity

- The interpolant should be as smooth as possible, i.e. it should belong to the set C^k of k-times continuously differentiable functions with a large k.
- allows to compute higher order derivatives analytically, e.g. for feature detection

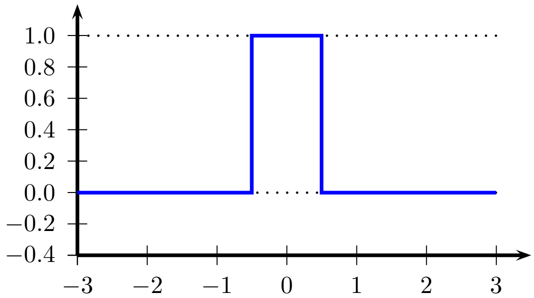
Synthesis Functions for Classical Interpolation (1)

Synthesis Functions for Classical Interpolation (in 1-D)

- **Nearest Neighbour Interpolation**

$$\varphi_{\text{int}}(x) = \begin{cases} 1 & \text{for } |x| < \frac{1}{2}, \\ \frac{1}{2} & \text{for } |x| = \frac{1}{2}, \\ 0 & \text{else.} \end{cases}$$

  - very small support interval: $[-\frac{1}{2}, \frac{1}{2}]$
  - lowest approximation order: $p = 1$ (reproduces constant functions)
  - leads to a discontinuous interpolant



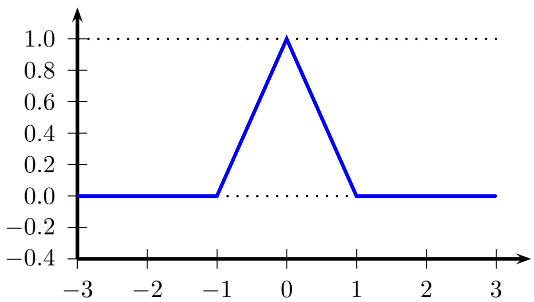
Synthesis function for nearest neighbour interpolation. Author: M. Mainberger.

Synthesis Functions for Classical Interpolation (2)

- **Linear Interpolation**

$$\varphi_{\text{int}}(x) = \begin{cases} 1 - |x| & \text{for } |x| < 1, \\ 0 & \text{else} \end{cases}$$

  - small support interval: $[-1, 1]$
  - approximation order $p = 2$ (reproduces linear functions)
  - gives a continuous (C^0) interpolant



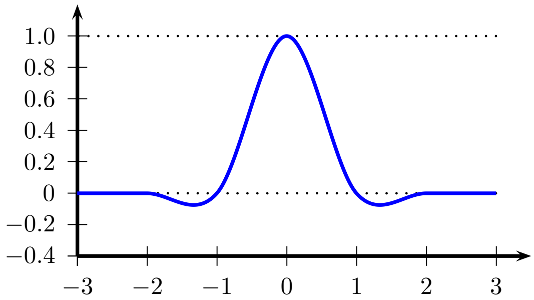
Synthesis function for linear interpolation. Author: M. Mainberger.

Synthesis Functions for Classical Interpolation (3)

- **Keys Interpolation (with $a = -\frac{1}{2}$)**

$$\varphi_{\text{int}}(x) = \begin{cases} \frac{3}{2}|x|^3 - \frac{5}{2}x^2 + 1 & \text{for } |x| < 1, \\ -\frac{1}{2}|x|^3 + \frac{5}{2}x^2 - 4|x| + 2 & \text{for } 1 \leq |x| < 2, \\ 0 & \text{else.} \end{cases}$$

  - support interval $[-2, 2]$
  - approximation order $p = 3$ (reproduces quadratic functions)
  - creates a continuously differentiable (C^1) interpolant



Synthesis function for Keys interpolation. Author: M. Mainberger.

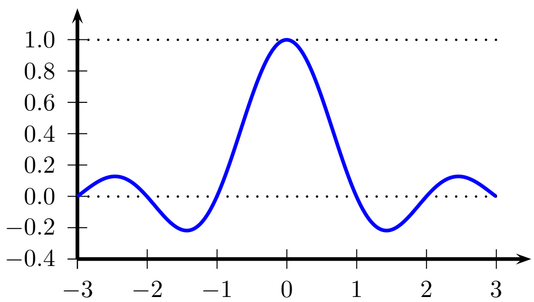
Synthesis Functions for Classical Interpolation (4)

M I A
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30

- **Sinc Interpolation**

$$\varphi_{\text{int}}(x) \;=\; \operatorname{sinc}(\pi x) \;=\; \frac{\sin(\pi x)}{\pi x}$$

  - infinite (!) support interval $(-\infty, \infty)$
  - gives the exact (!) reconstruction for a bandlimited signal that has been sampled according to the sampling theorem (cf. Lecture 5)
  - yields an infinitely times differentiable (C^∞) interpolant
  - difficult to implement exactly due to its infinite support and its slow decay: usually approximated by truncated or windowed sinc functions



Synthesis function for sinc interpolation. Author: M. Mainberger.

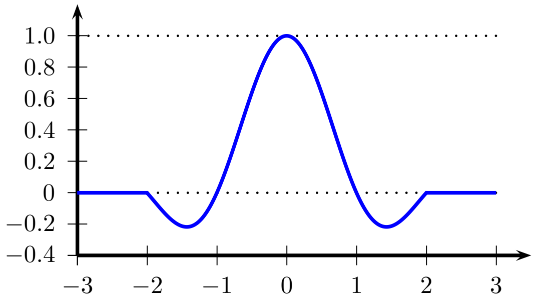
Synthesis Functions for Classical Interpolation (5)

Truncated Sinc Interpolation

- truncates the sinc function outside some interval $[-n, n]$, $n \in \mathbb{N}$
- gives the so-called *Dirichlet apodisation*

$$\varphi_{\text{int}}(x) = \begin{cases} \frac{\sin(\pi x)}{\pi x} & \text{for } |x| \leq n, \\ 0 & \text{else.} \end{cases}$$

- Unfortunately, this leads only to a C^0 interpolant.
- This does not even satisfy the partition of unity!
  Thus, it may create severe shifts in the average grey level of the image.



Synthesis function for Dirichlet apodisation $(n = 2)$. Author: M. Mainberger.

Synthesis Functions for Classical Interpolation (6)

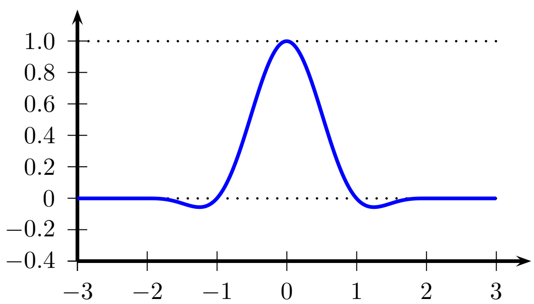
M I A 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30

Windowed Sinc Interpolation

- multiplies the sinc function with a smooth window of support $[-n, n]$, $n \in \mathbb{N}$
- using e.g. a so-called *Hanning window* gives the *Hanning apodisation*

$$\varphi_{\text{int}}(x) = \begin{cases} \frac{\sin(\pi x)}{\pi x} \left(\frac{1}{2} + \frac{1}{2}\cos\left(\frac{\pi x}{n}\right)\right) & \text{for } |x| \leq n, \\ 0 & \text{else.} \end{cases}$$

- creates a C^1 interpolant
- violates partition of unity as well, leading to greyscale shifts



Synthesis function for Hanning apodisation $(n = 2)$. Author: M. Mainberger.

M I
A
1 2
3 4
5 6
7 8
9 10
11 12
13 14
15 16
17 18
19 20
21 22
23 24
25 26
27 28
29 30

Generalised Interpolation

Motivation

- So far we have considered classical interpolation methods, where a weighted average of the *function values* $\{f_{\boldsymbol{k}} \,|\, \boldsymbol{k} \in \mathbb{Z}^m\}$ is computed:

$$u(\boldsymbol{x}) = \sum_{\boldsymbol{k} \in \mathbb{Z}^m} f_{\boldsymbol{k}}\, \varphi_{\text{int}}(\boldsymbol{x} - \boldsymbol{k}).$$

- In practice, this restricts the choice of suitable synthesis functions: We used interpolating synthesis functions that satisfy the *interpolation condition*

$$\varphi_{\text{int}}(\boldsymbol{x}) = \begin{cases} 1 & \text{for } \boldsymbol{x} = (0, ..., 0)^\top, \\ 0 & \text{for } \boldsymbol{x} \in \mathbb{Z}^m \setminus \{(0, ..., 0)^\top\}. \end{cases}$$

- It is possible to obtain novel, better methods when we
  - permit coefficients that do not coincide with the function values
  - and do not insist on the interpolation condition ?

M I
A
1 2
3 4
5 6
7 8
9 10
11 12
13 14
15 16
17 18
19 20
21 22
23 24
25 26
27 28
29 30

Basic Idea Behind Generalised Interpolation

- Consider the *generalised interpolation* formula

$$u(\boldsymbol{x}) = \sum_{\boldsymbol{k} \in \mathbb{Z}^m} c_{\boldsymbol{k}}\, \varphi(\boldsymbol{x} - \boldsymbol{k}).$$

  - The unknown coefficients $\{c_{\boldsymbol{k}} \,|\, \boldsymbol{k} \in \mathbb{Z}^m\}$ must be computed (details later): They depend on the function values $\{f_{\boldsymbol{k}} \,|\, \boldsymbol{k} \in \mathbb{Z}^m\}$ and the synthesis function φ.
  - The synthesis function φ can be *noninterpolating:* It does not have to fullfil the interpolation condition (we write φ instead of φ_{int}).

- Generalised interpolation proceeds in two steps:
  - compute the coefficients $\{c_{\boldsymbol{k}} \,|\, \boldsymbol{k} \in \mathbb{Z}^m\}$
  - interpolate with the preceding formula

- This two-step procedure can pay off in terms of interpolation quality: Noninterpolating synthesis functions with a finite support can be equivalent to interpolating synthesis functions with an infinite support.

Generalised Interpolation (3)

Desirable Properties of Noninterpolating Synthesis Functions

- Just as for interpolating synthesis functions, separability and symmetry are useful.
- Partition of unity must be modified: One can show that

$$\sum_{\boldsymbol{k}\in\mathbb{Z}^m} \varphi(\boldsymbol{x}-\boldsymbol{k}) \;=\; \frac{1}{c_0} \qquad \text{for all } \boldsymbol{x}\in\mathbb{R}^m$$

  guarantees that a constant signal is reproduced exactly.

Quality Criteria of a Good Generalised Interpolant

- We have the same criteria as for a classical interpolant:
  - small support region of the synthesis function
  - high approximation order
  - high regularity
- Moreover, it should be easy to compute the coefficents $\{c_{\boldsymbol{k}} \,|\, \boldsymbol{k}\in\mathbb{Z}^m\}$.

Generalised Interpolation (4)

How Are the Coefficients Computed ?

- For the sake of simplicity, consider the 1-D case with *finitely* many equidistant interpolation data f_1, f_2,..., f_N at the points $x_1\!=\!1$, $x_2\!=\!2$, ..., $x_N\!=\!N$.
- For a given synthesis function φ, the conditions $u(i) = f_i$ for $i = 1,\ldots,N$ yield

$$\sum_{k=1}^{N} c_k\,\varphi(i-k) \;=\; f_i \qquad \text{for } i=1,\ldots,N.$$

- This describes a linear system of N equations for the N unknowns c_1, c_2, ..., c_N:

$$\begin{pmatrix} \varphi(0) & \varphi(1) & \ldots & \varphi(N-1) \\ \varphi(1) & \varphi(0) & \ldots & \varphi(N-2) \\ \vdots & \vdots & \ldots & \vdots \\ \varphi(N-1) & \varphi(N-2) & \ldots & \varphi(0) \end{pmatrix} \begin{pmatrix} c_1 \\ c_2 \\ \vdots \\ c_N \end{pmatrix} = \begin{pmatrix} f_1 \\ f_2 \\ \vdots \\ f_N \end{pmatrix}$$

  where we have used the symmetry condition $\varphi(x) = \varphi(-x)$.
- If φ has a small support, then we have a band matrix with a small band.

M I A
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30

Which Synthesis Functions Are Used ?

- The most popular synthesis functions are derived from so-called *B-splines.*
- We start with the box function for nearest neighbour interpolation.
  We define it as synthesis function β_0:

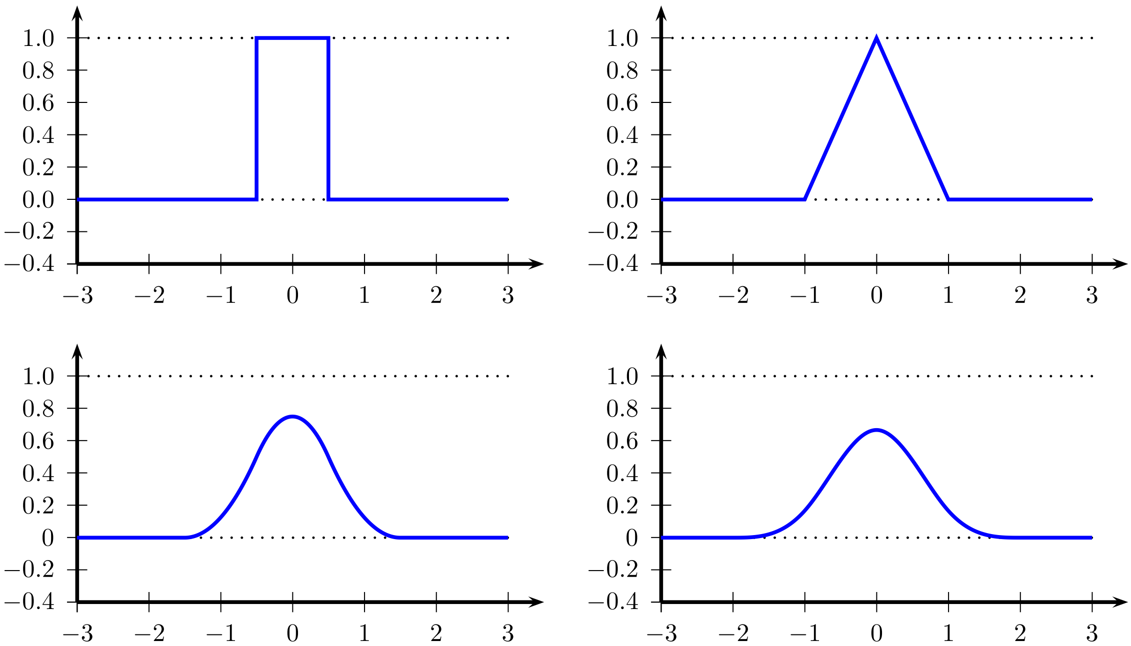
$$\beta_0(x) = \begin{cases} 1 & \text{for } |x| < \frac{1}{2}, \\ \frac{1}{2} & \text{for } |x| = \frac{1}{2}, \\ 0 & \text{else.} \end{cases}$$

- The synthesis functions β_n with $n = 1, 2, \ldots$ are derived iteratively by convolution with β_0 (see also Assignment H1, Problem 2):

$$\begin{aligned} \beta_1 &= \beta_0 * \beta_0\,, \\ \beta_2 &= \beta_1 * \beta_0\,, \\ \beta_3 &= \beta_2 * \beta_0\,. \end{aligned}$$

- For $n \geq 1$, the synthesis function β_n is a piecewise polynomial of degree n.
  It is symmetric in 0, and $n-1$ times continuously differentiable.

M I A
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30



The first four synthesis functions derived from B-splines. **Top left:** β_0. **Top right:** β_1. **Bottom left:** β_2. **Bottom right:** β_3. Note that β_0 and β_1 satisfy the interpolation condition, while β_2 and β_3 are noninterpolating synthesis functions with $\beta_i(0) \neq 1$. Author: M. Mainberger.

Generalised Interpolation (7)

Explicit Formulas for the B-Spline Synthesis Functions

$$\beta_0(x) = \begin{cases} 1 & \text{for } |x| < \frac{1}{2}, \\ \frac{1}{2} & \text{for } |x| = \frac{1}{2}, \\ 0 & \text{else,} \end{cases}$$

$$\beta_1(x) = \begin{cases} 1 - |x| & \text{for } |x| < 1, \\ 0 & \text{else,} \end{cases}$$

$$\beta_2(x) = \begin{cases} \frac{3}{4} - x^2 & \text{for } |x| < \frac{1}{2}, \\ \frac{1}{2}(\frac{3}{2} - |x|)^2 & \text{for } \frac{1}{2} \le |x| < \frac{3}{2}, \\ 0 & \text{else,} \end{cases}$$

$$\beta_3(x) = \begin{cases} \frac{2}{3} - x^2 + \frac{1}{2}|x|^3 & \text{for } |x| < 1, \\ \frac{1}{6}(2 - |x|)^3 & \text{for } 1 \le |x| < 2, \\ 0 & \text{else.} \end{cases}$$

Generalised Interpolation (8)

Cubic B-Spline Interpolation

- most popular generalised interpolation
- uses synthesis function β_3
- support interval $[-2, 2]$.
- can be shown to have approximation order $p = 4$ (reproduces cubic functions)
- creates a twice continuously differentiable (C^2) interpolant
- one order better than Keys interpolation which has the same support interval
- offers good cost–performance ratio

M I A
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30

Finding the Coefficients for Cubic B-Spline Interpolation

- synthesis function β_3 at integer values gives

$$\beta_3(k) \;=\; \begin{cases} \frac{2}{3} & \text{for } k = 0, \\ \frac{1}{6} & \text{for } k = \pm 1, \\ 0 & \text{for other integer values } k. \end{cases}$$

- leads to the following tridiagonal system for the interpolation coefficients:

$$\begin{pmatrix} \frac{2}{3} & \frac{1}{6} & 0 & \dots & \dots & 0 \\ \frac{1}{6} & \frac{2}{3} & \frac{1}{6} & 0 & \dots & 0 \\ & \ddots & \ddots & \ddots & & \\ & & \ddots & \ddots & \ddots & \\ 0 & \dots & 0 & \frac{1}{6} & \frac{2}{3} & \frac{1}{6} \\ 0 & \dots & \dots & 0 & \frac{1}{6} & \frac{2}{3} \end{pmatrix} \begin{pmatrix} c_1 \\ c_2 \\ \vdots \\ \vdots \\ c_{N-1} \\ c_N \end{pmatrix} = \begin{pmatrix} f_1 \\ f_2 \\ \vdots \\ \vdots \\ f_{N-1} \\ f_N \end{pmatrix}.$$

- can be solved efficiently with the so-called Thomas algorithm (Lecture 17)

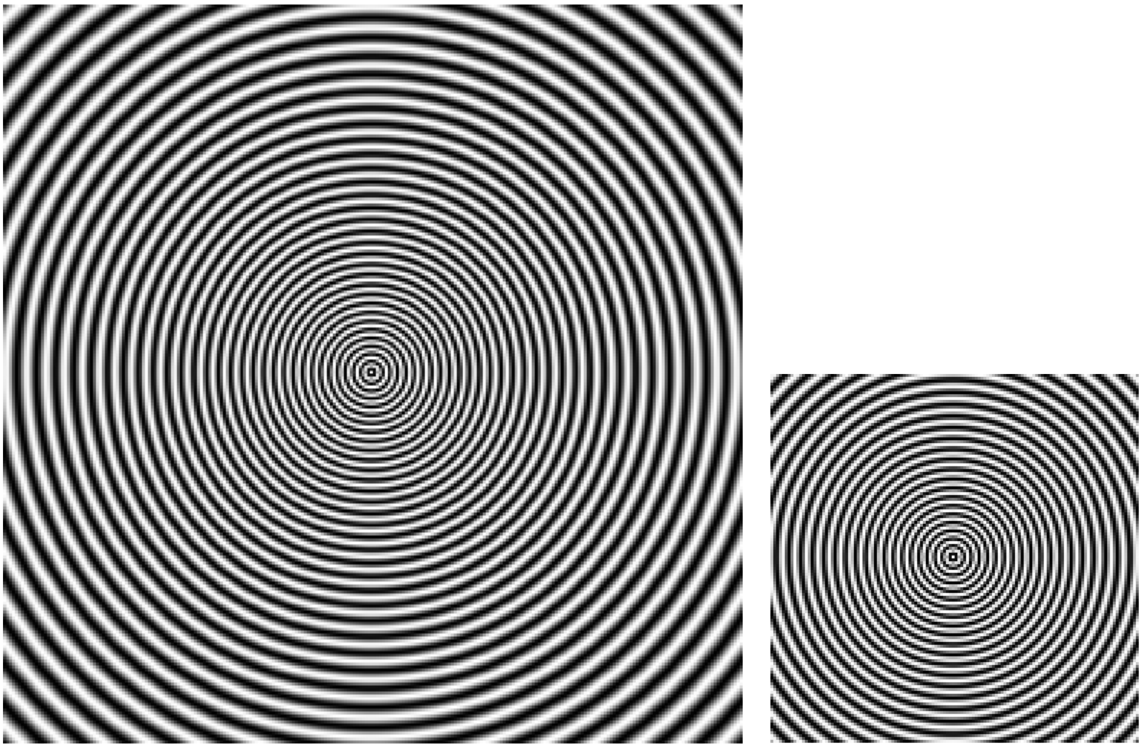
Experiments

- Consider a rotationally invariant test image.

- Perform 15 rotations by $\frac{360}{15}$ degrees.
  The output of any step is used as input of the next step.
  Each rotation requires interpolation.

- For an ideal interpolant, the resulting image should be identical to the original one.

Experiments (2)

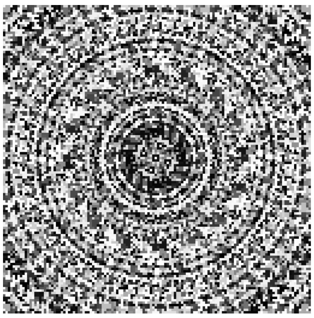


Left: Test image. **Right:** Central square. Authors: P. Thévenaz, T. Blu, M. Unser.

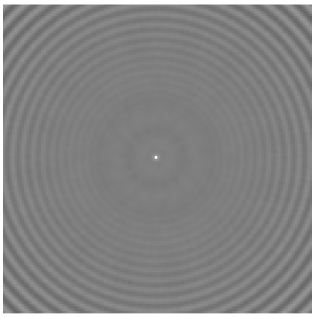
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30

Experiments (3)

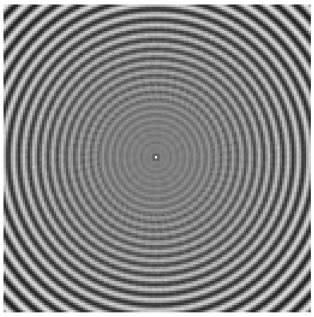
nearest neighbour

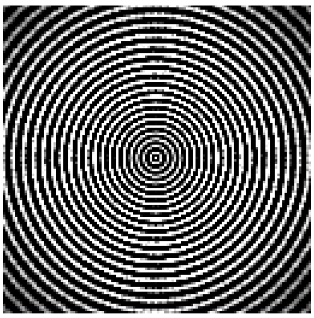


linear

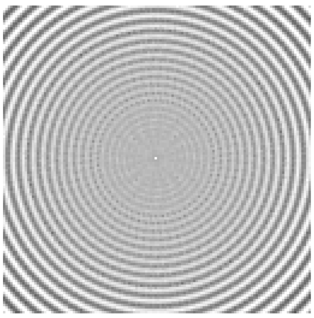


Keys ($a = -\frac{1}{2}$)

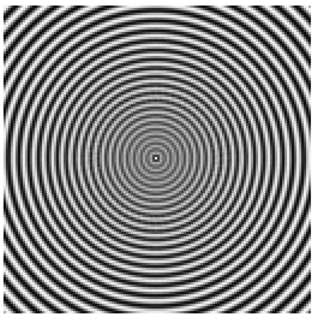




truncated sinc ($n = 2$)



windowed sinc ($n = 2$)



cubic B-spline

Comparison of different interpolants after 15 rotations with $\frac{360}{15}$ degrees. Note also the grey level shifts for truncated and windowed sinc interpolation. Authors: P. Thévenaz, T. Blu, M. Unser.

1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30

Summary

- Interpolation recovers continuous data from discrete samples.
- Classical interpolation involves
  - the function values,
  - a synthesis function that satisfies the interpolation condition.
- Examples: nearest neighbour, linear, Keys, sinc interpolation
- Generalised interpolation renounces the interpolation condition.
  It requires to compute the weight coefficients.
- This extra effort can be rewarded by better quality.
- Example: Cubic spline interpolation.
  It offers a favourable cost–performance ratio.

References

When it comes to interpolation, you better forget about text books and consult the following articles instead:

- P. Thévenaz, T. Blu, M. Unser: Image interpolation and resampling. In I. N. Bankman (Ed.): *Medical Imaging, Processing and Analysis*. Academic Press, San Diego, pp. 393–420, 2000. (`http://bigwww.epfl.ch/publications/thevenaz9901.pdf`)
  (excellent paper that formed the basis of the current lecture)
- H. S. Hou, H. C. Andrews: Cubic splines for image interpolation and digital filtering. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, Vol. 26, No. 6, pp. 508–517, Dec. 1978.
  (one of the first papers on spline interpolation in image analysis)
- T. M. Lehmann, C. Gönner, K. Spitzer: Survey: Interpolation methods in medical image processing. *IEEE Transactions on Medical Imaging*, Vol. 18, No. 11, pp. 1049–1075,Nov. 1999.
  (experimental evaluation of numerous methods)
- E. H. W. Meijering, W. J. Niessen, M. A. Viergever: Quantitative evaluation of convolution-based methods for medical image interpolation. *Medical Image Analysis*, Vol. 5, No. 2, pp. 111–126, 2001.
  (another detailed experimental evaluation)
- E. Meijering: A chronology of interpolation: From ancient astronomy to modern signal and image processing. *Proceedings of the IEEE*, Vol. 90, No. 3, pp. 319–342, March 2002.
  (provides interesting historical facts)

M I A
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30