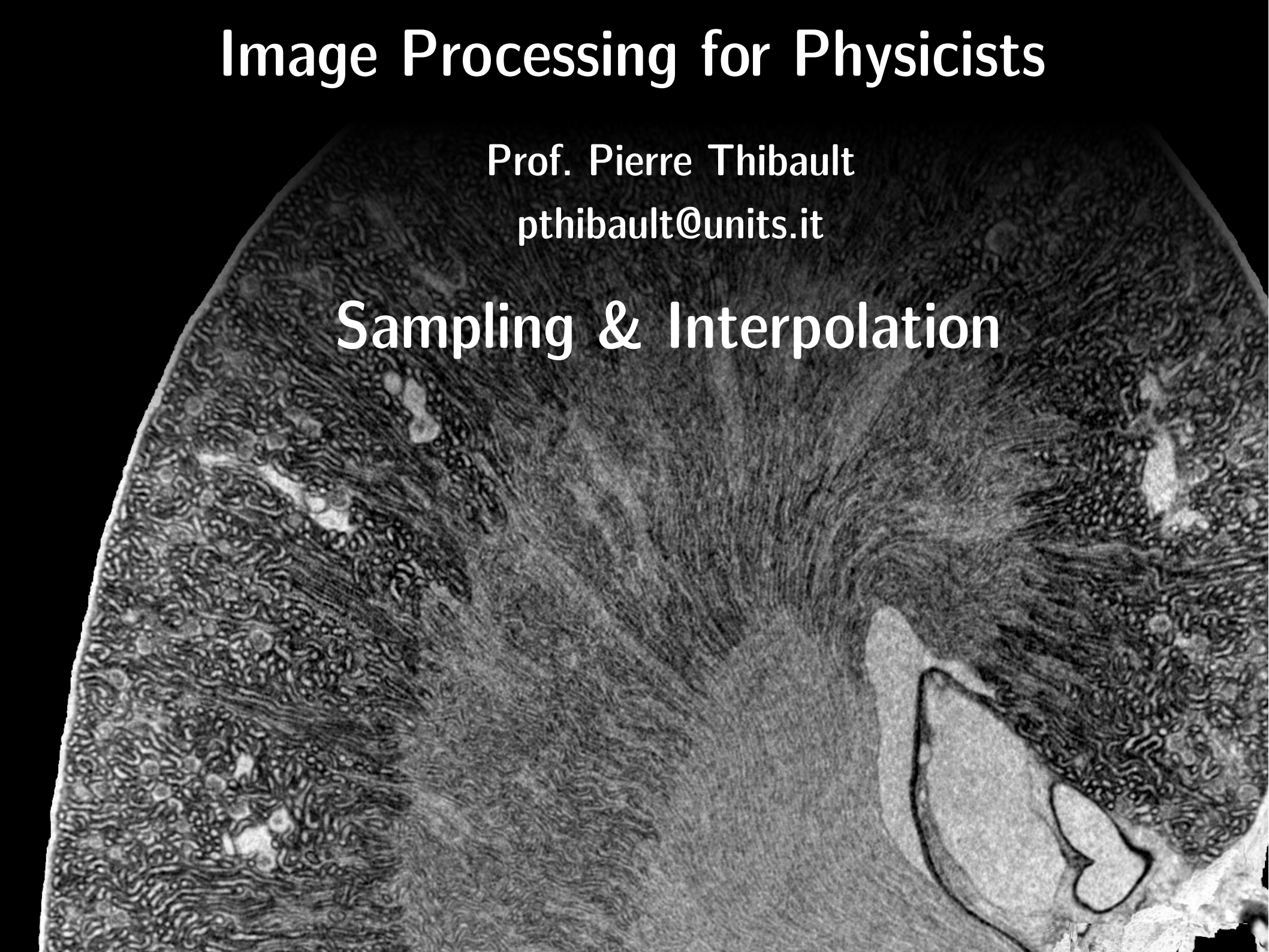
Image Processing for Physicists

Prof. Pierre Thibault

pthibault@units.it

Sampling & Interpolation



Overview

- Sampling
  - Nyquist theorem
  - Undersampling and Aliasing
- Interpolation (resampling)

Discrete Fourier Transform

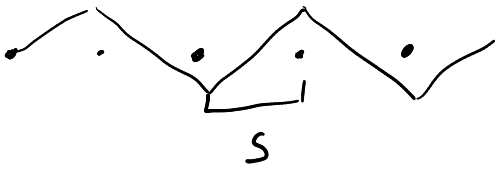
- A **periodic** function has a **discrete** spectrum in the Fourier domain;
- A function with **discrete** values in the spatial domain is **periodic** in the Fourier domain;

  $\Rightarrow$ A periodic and discrete function has a periodic and discrete Fourier transform.

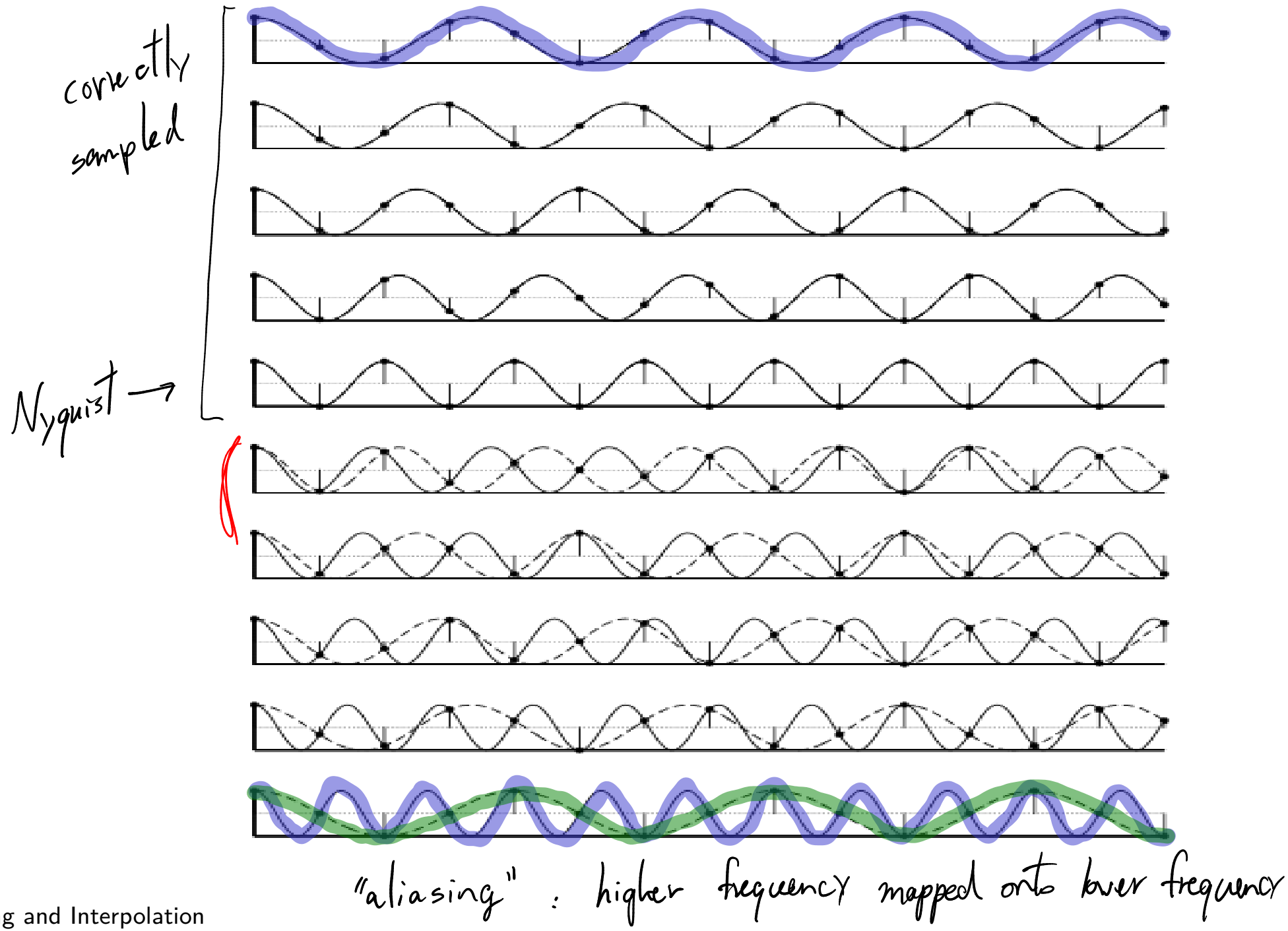
$$\tilde{F}_k = \sum_{n=0}^{N-1} f_n \underbrace{e^{-2\pi i n k / N}}_{1}$$

Sampling: Nyquist theorem

- The largest frequency that can be represented in a signal sampled at intervals s is 1/2s

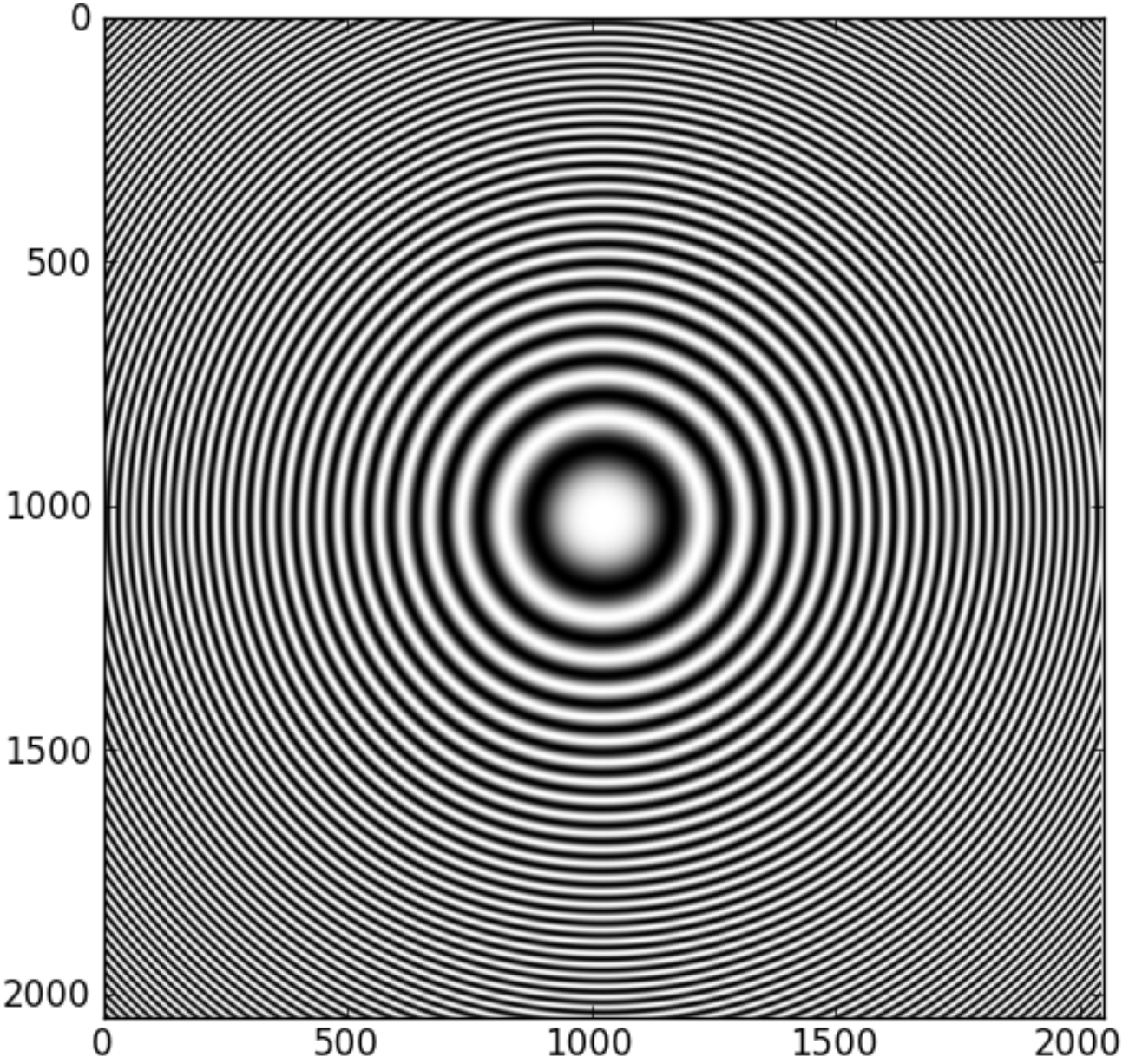


Undersampling



Undersampling

"Fresnel zone" test pattern: radial linear increase in spatial frequency

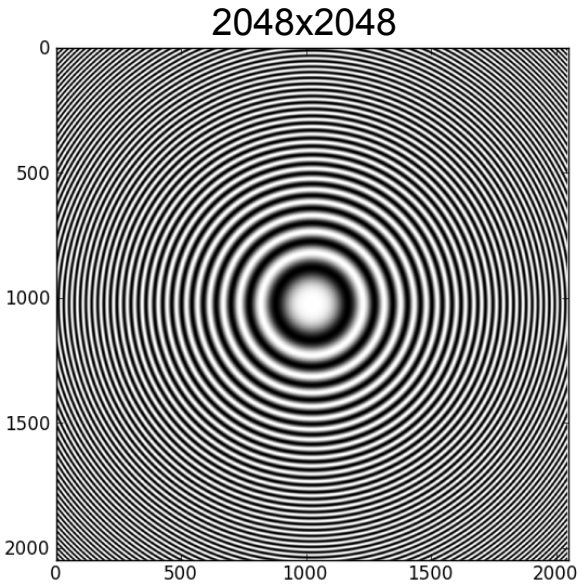


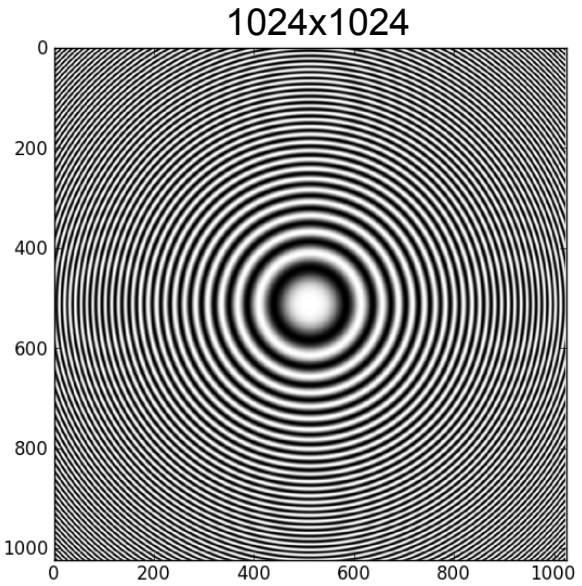
$\cos(r^2 \cdot a)$

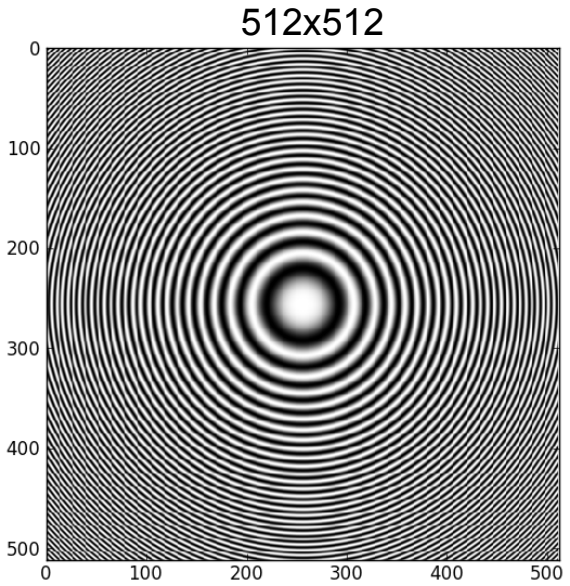
$\cos(r\ (ra))$

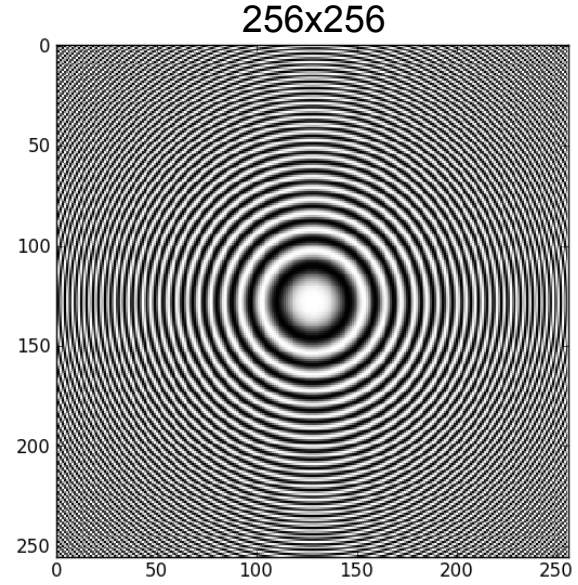
frequency increases radially, linearly with r

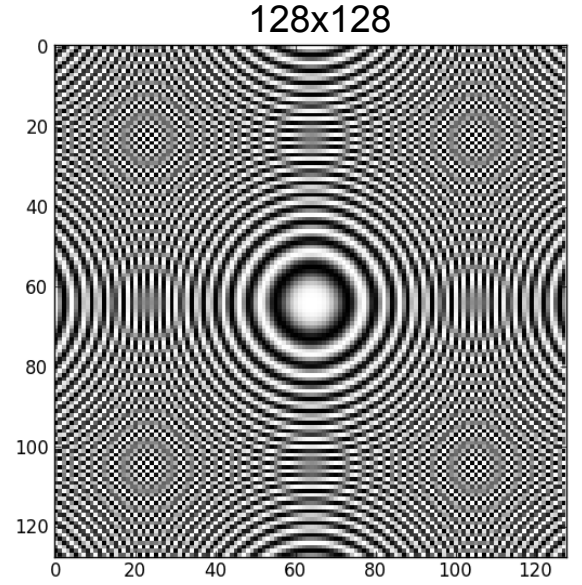
Undersampling & aliasing

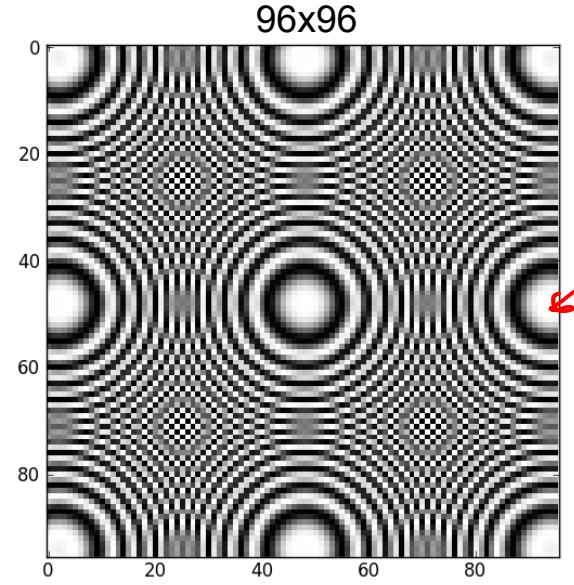




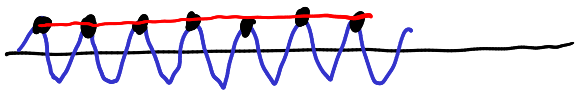






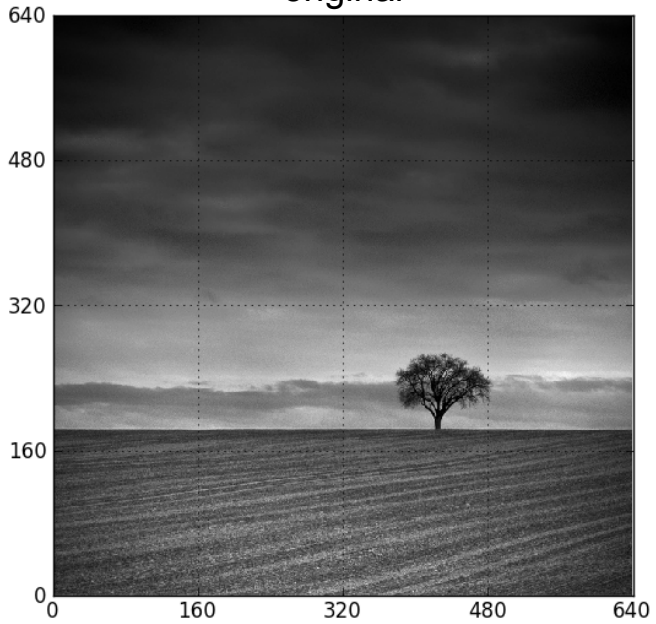


2x Nyquist



Fourier space translation

original



amplitude of Fourier spectrum

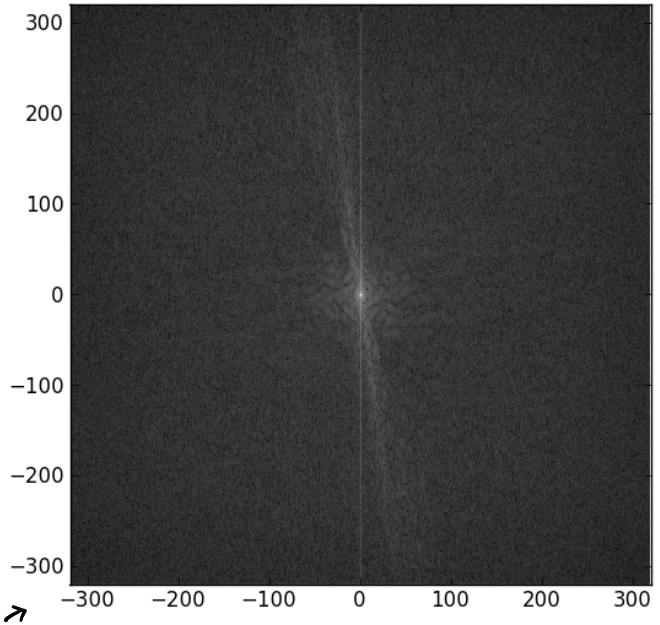
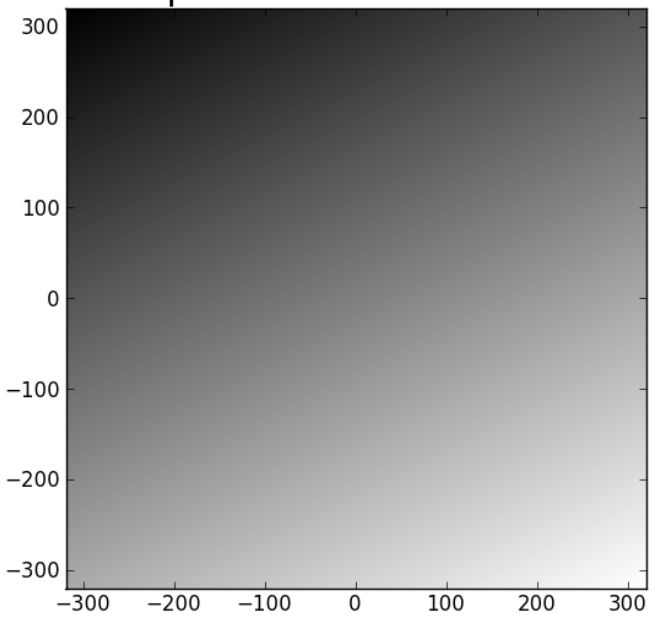


Image shifting using shifting property of FT

$e^{iR \cdot v}$

phase of filter function



shifted image

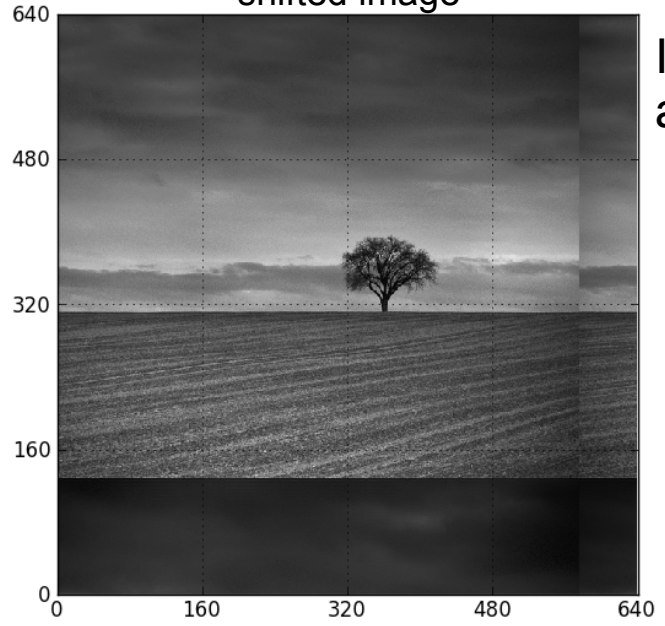
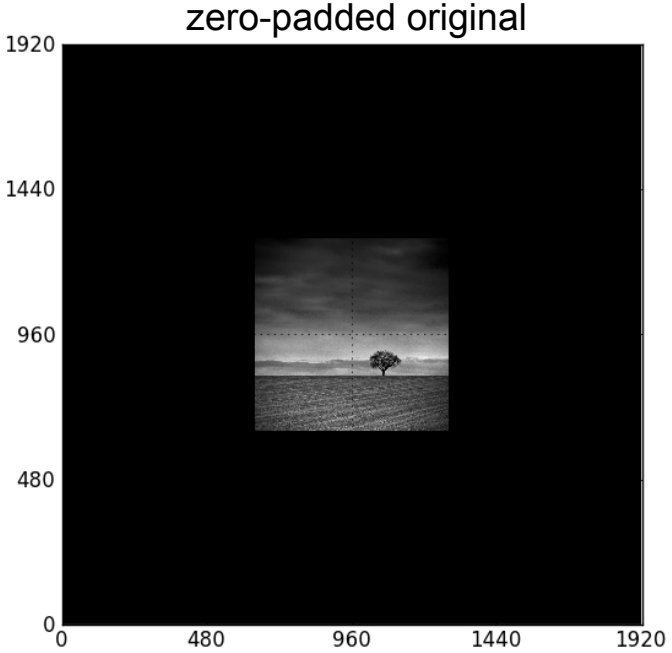
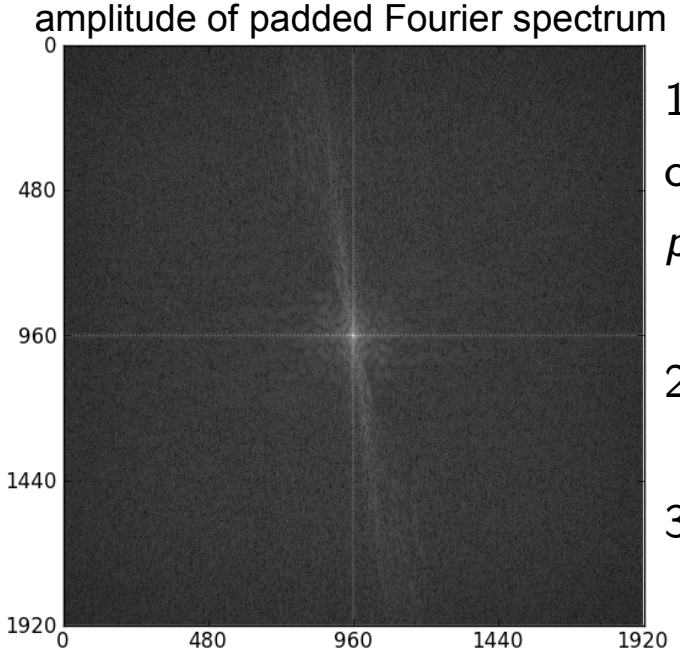


Image gets wrapped around

Zero-padding

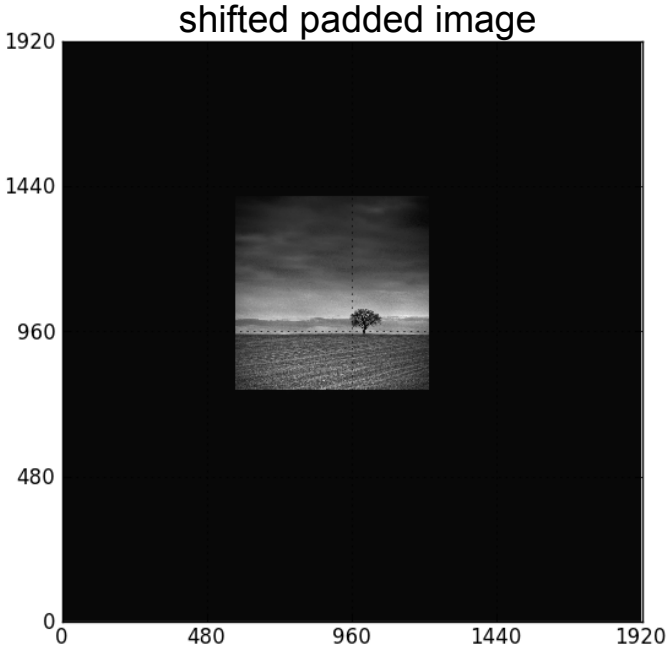


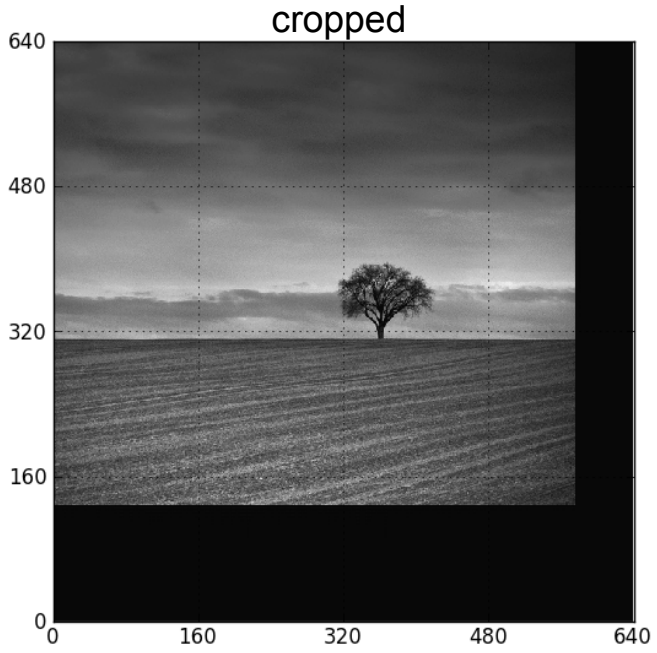


1. Add zeros around original image (*zero-padding*)

2. Shift using FT

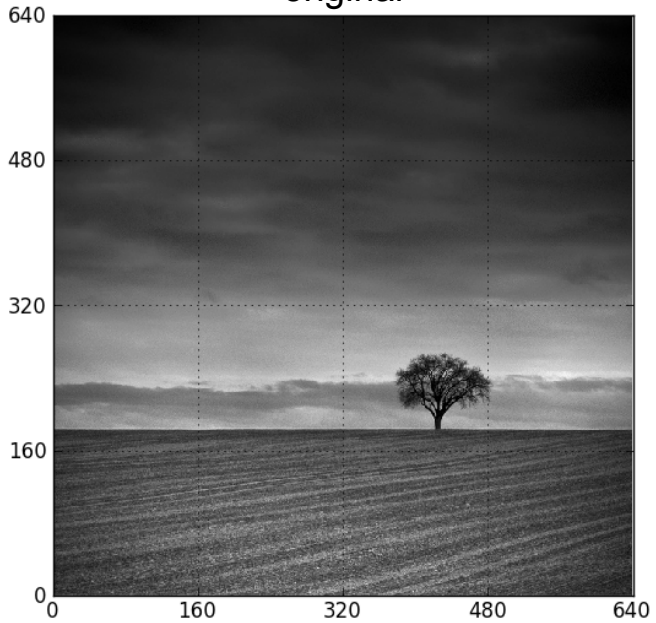
3. Crop result



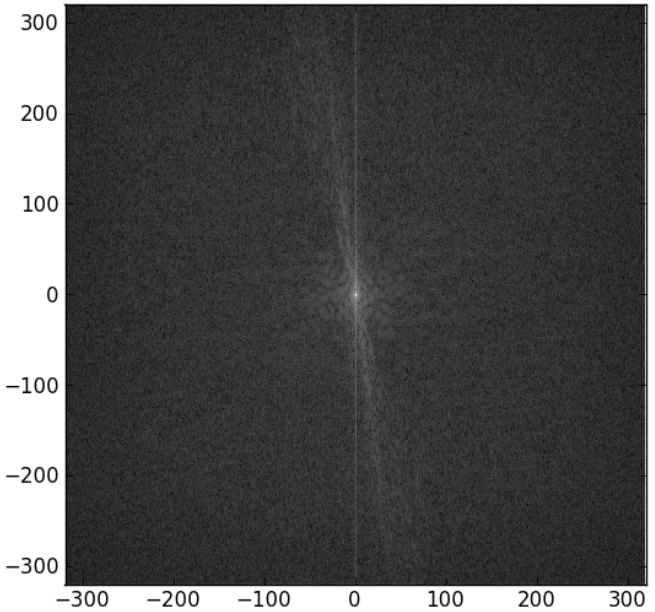


Zero-padding in Fourier space

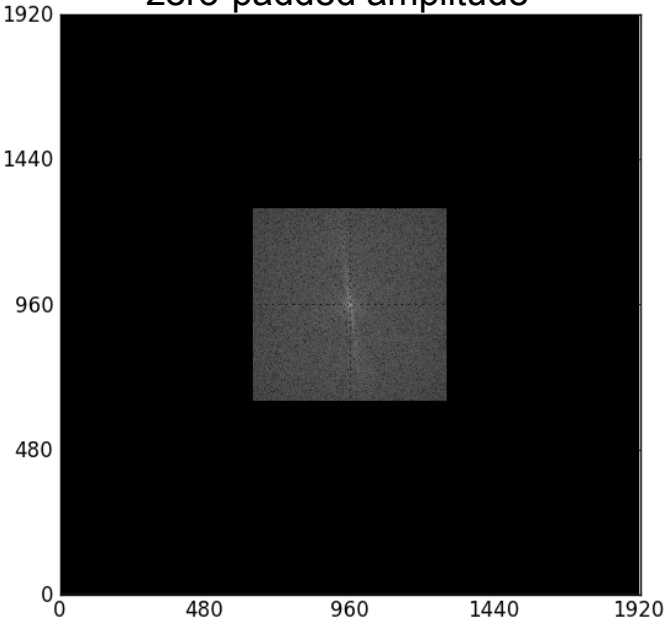
original



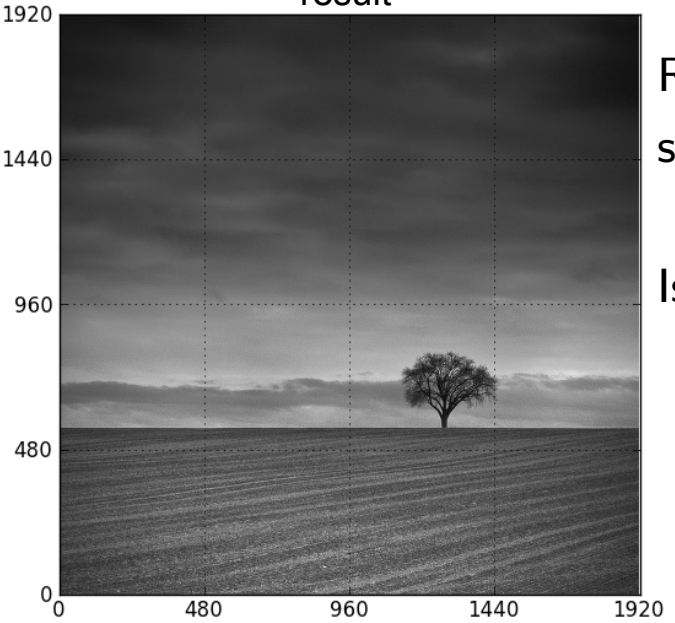
amplitude of Fourier spectrum



zero-padded amplitude



result

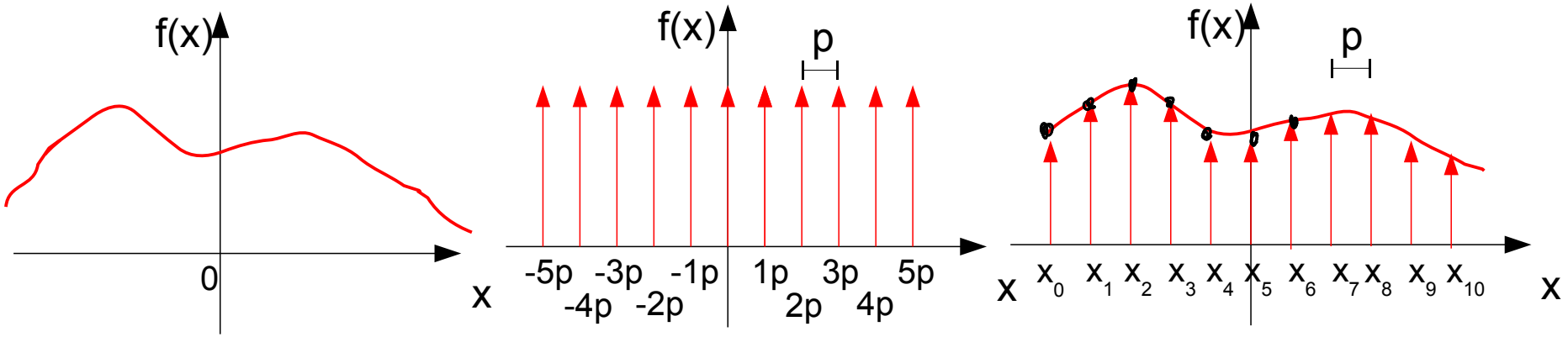


Result: increased sampling!

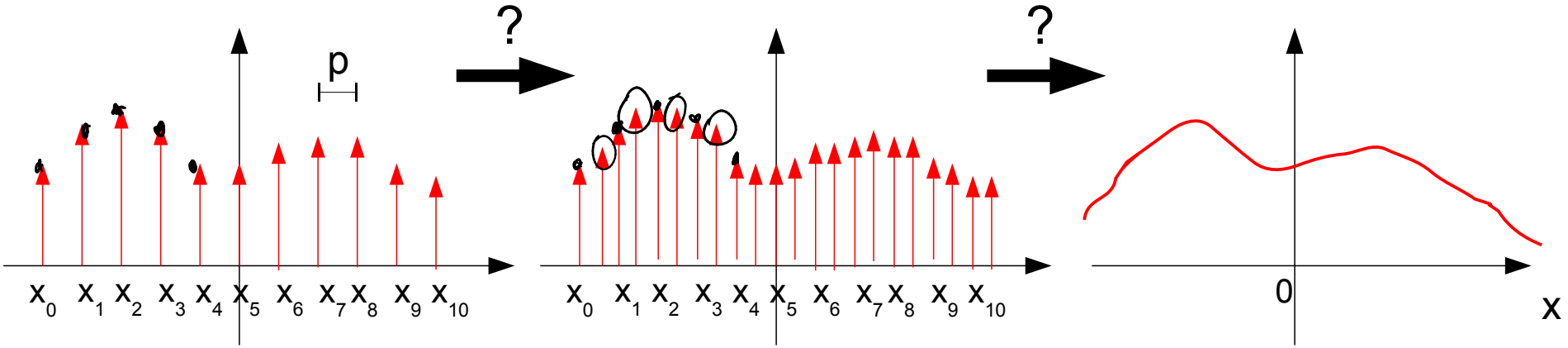
Is it magic?

Interpolation

- Discrete sampling of a continuous function



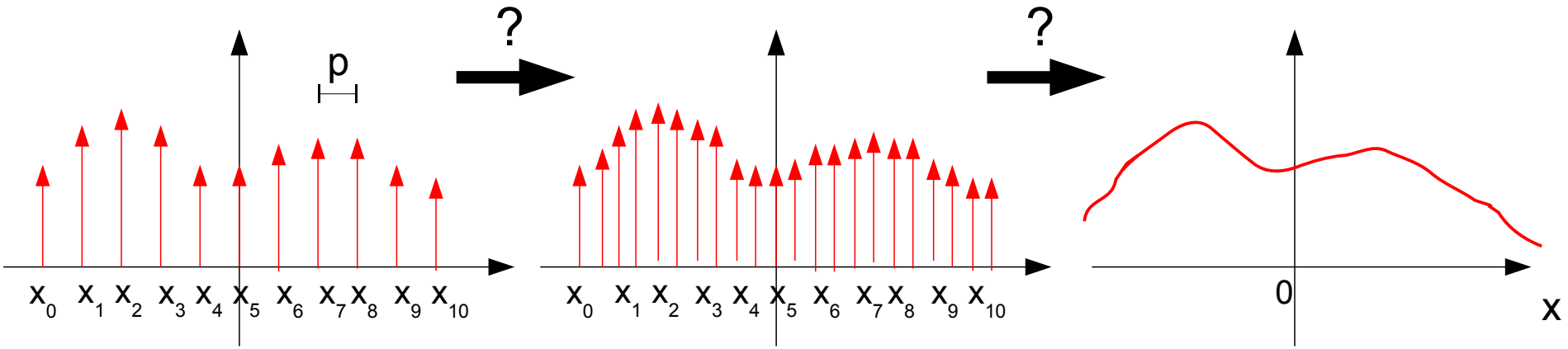
- Reconstruct original function from sampled data?



Interpolation

Finding unknown points between known ones

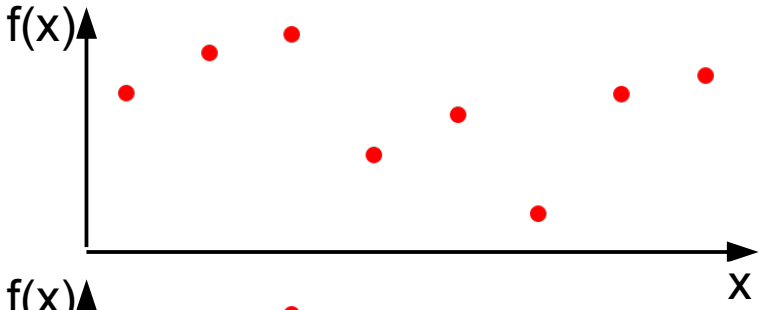
- wide field, many different approaches
- closely related to approximation theory and curve fitting



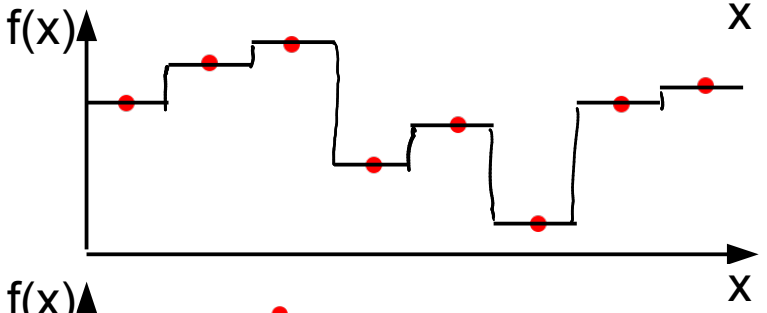
Interpolation

Various "classical" interpolation methods available

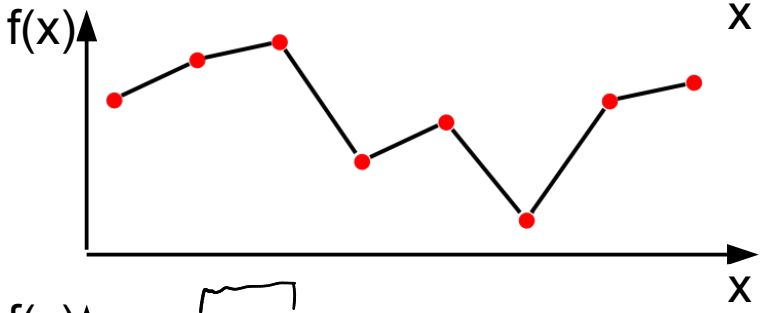
sampled points



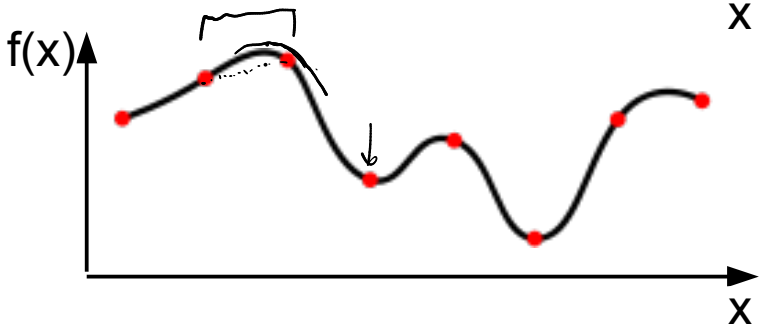
nearest neighbor



linear

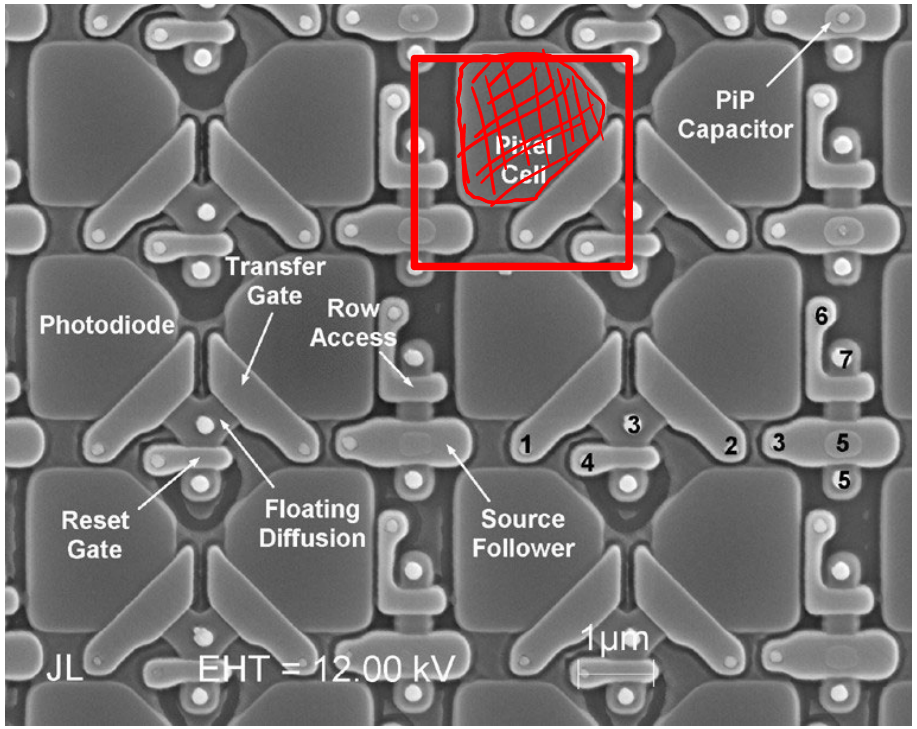


cubic spline



Pixels

- distinguish between detector pixels, image pixels and screen pixels
  - detector pixels are rarely square
  - image pixels are commonly, but not necessarily square
  - screen pixels are rarely square



CMOS ?

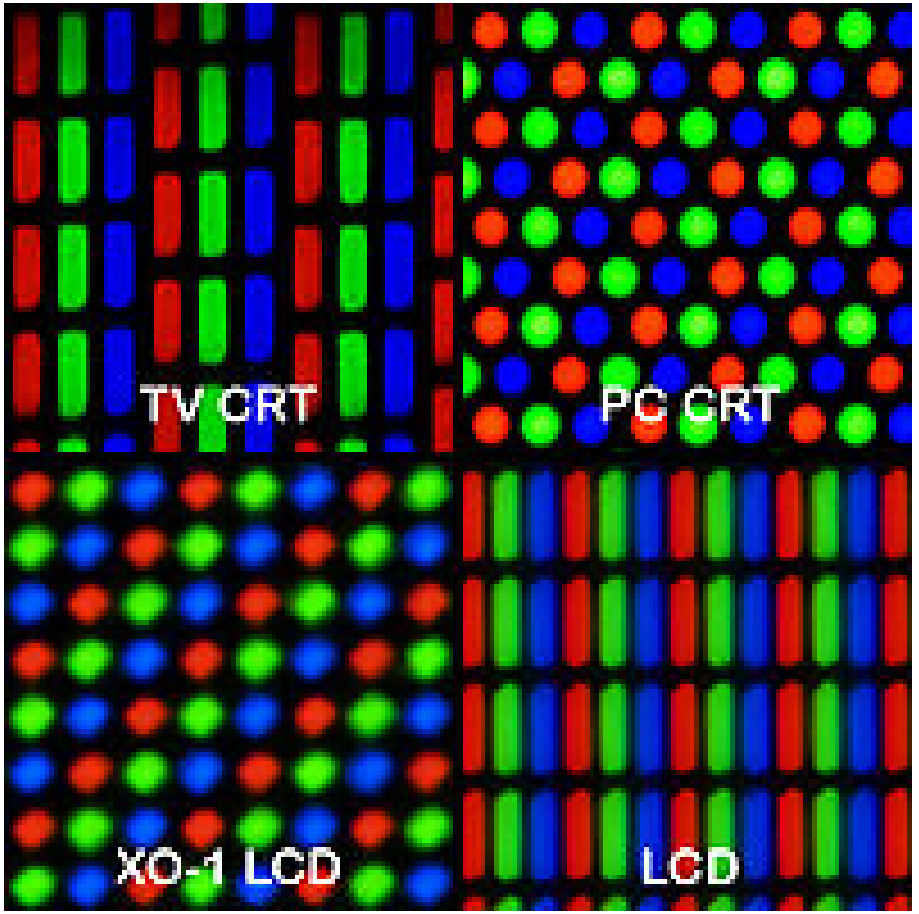
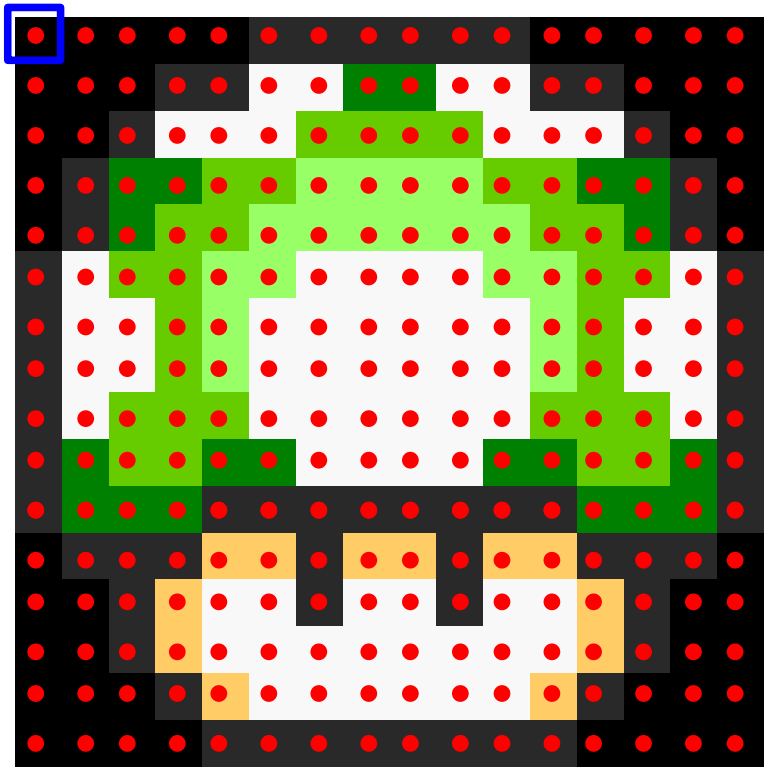


Image pixels

Images are discrete samples of a continuous function

- ...with coordinates
- ...and values (voltage at coordinate, integral over pixel area, ...)
- ...represented by pixel basis functions on a sampling grid



Linear interpolation

- Interpolation as an operator

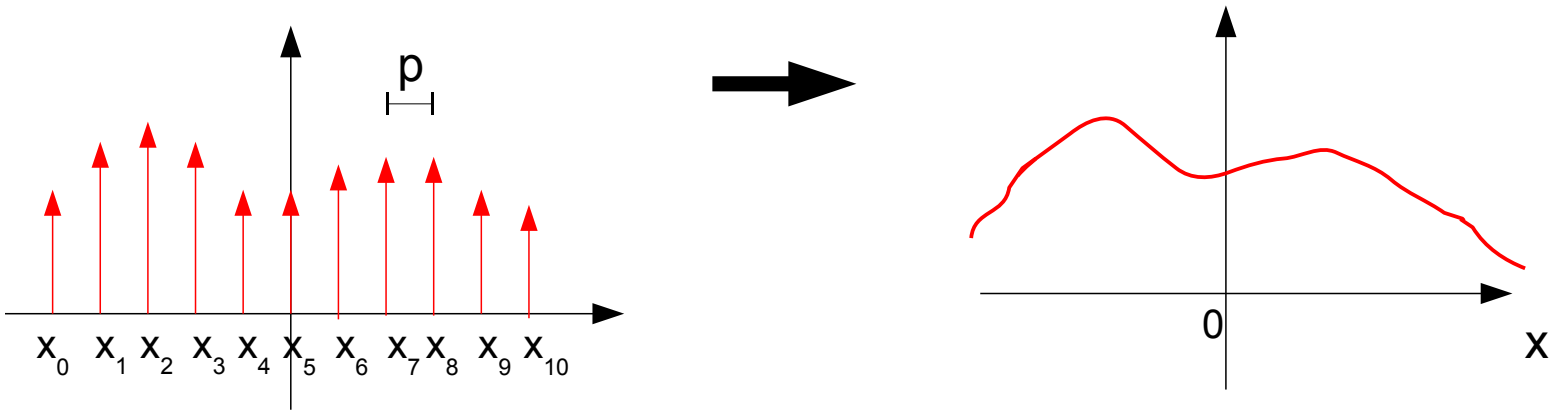
$$f(x) = \mathcal{L}\{f_n\}$$

- Linearity of interpolation

$$\mathcal{L}\{f_n + g_n\} = \mathcal{L}\{f_n\} + \mathcal{L}\{g_n\}$$

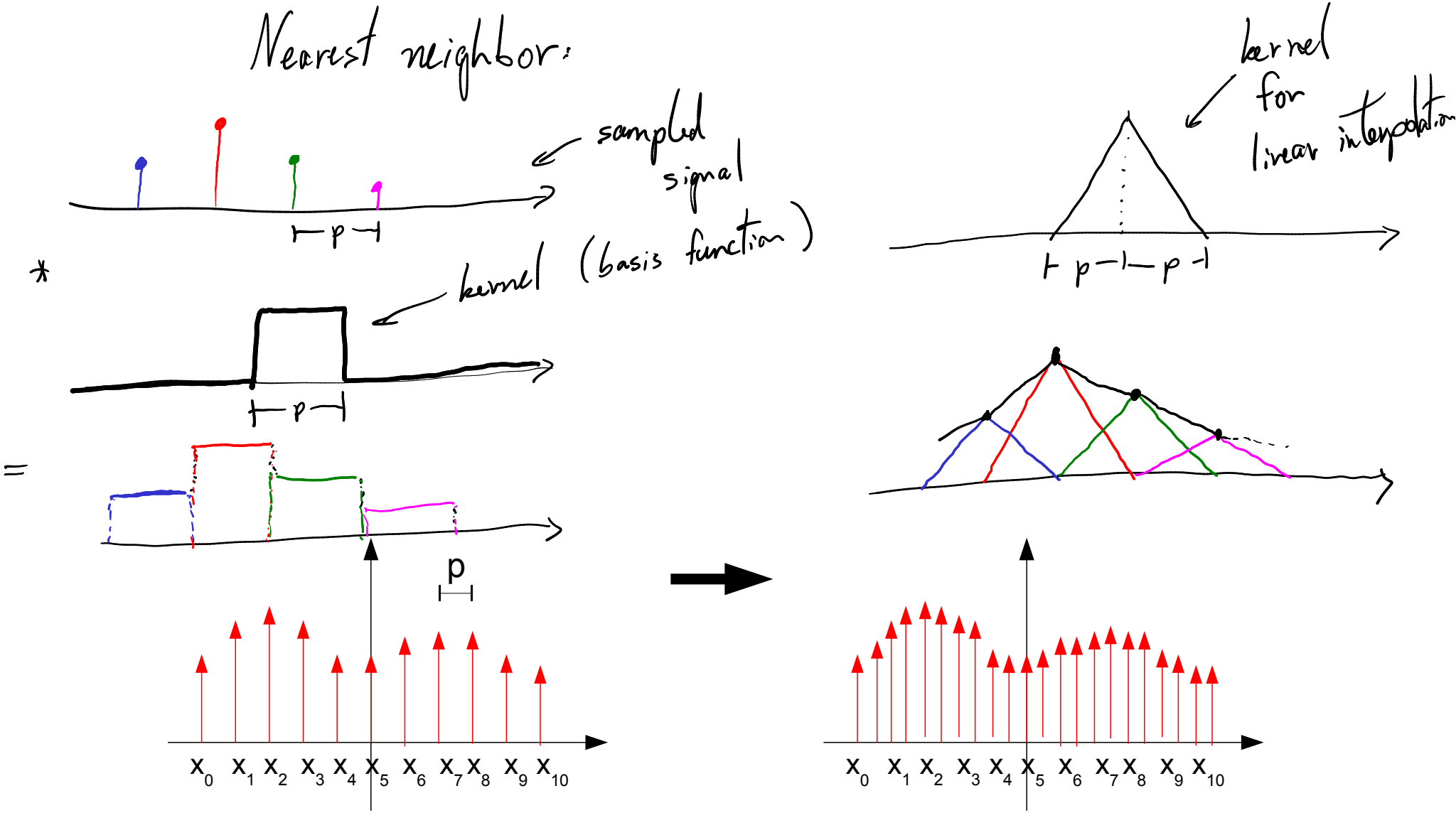
- Shift invariance

- Kernel



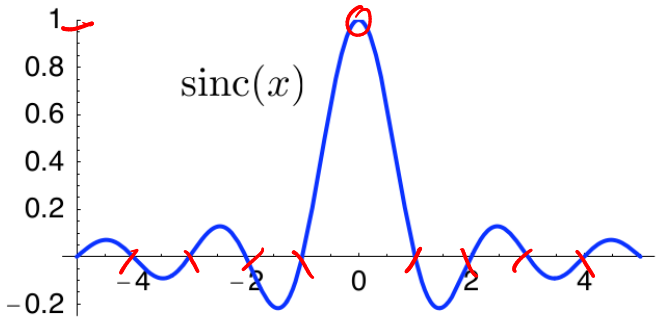
Linear interpolation

- Linear interpolation can be written as a convolution with a kernel (e.g. a basis function)

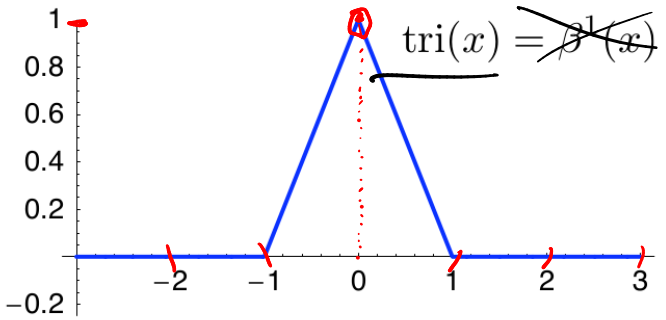


Linear interpolation

- Bandlimited



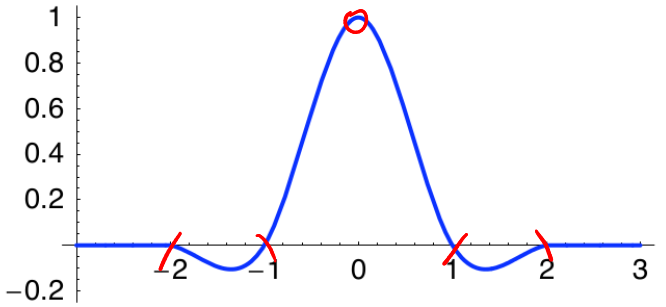
- Piecewise linear



Interpolation condition:

$$\varphi_{\text{int}}(k) = \delta_k = \begin{cases} 1, & k = 0 \\ 0, & \text{otherwise} \end{cases}$$

- Cubic convolution



[Keys, 1981; Karup-King 1899]

source: http://bigwww.epfl.ch/tutorials/unser_isbi_06_part1

Interpolation via convolution

$$f(x) = K(x) * \left[\sum f_n \, \delta(x-n) \right]$$

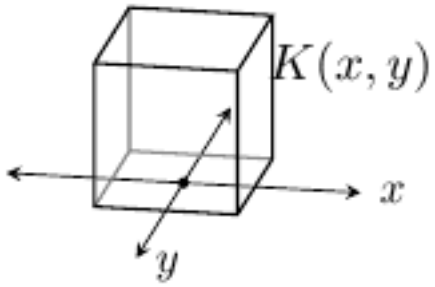
$$= \sum f_n \, K(x-n)$$

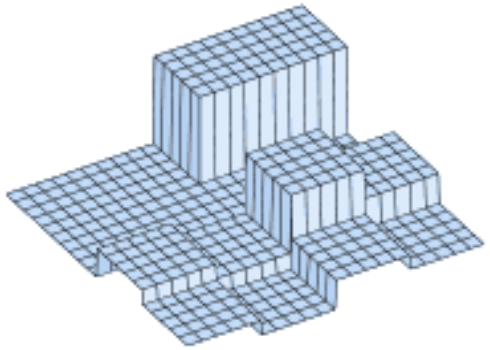
2D interpolation

- Make 2D interpolation linear in each variable

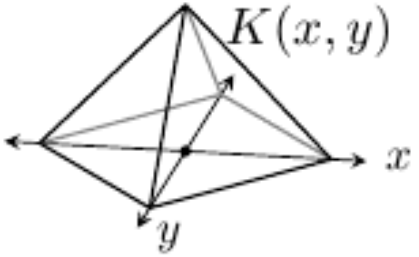
$$K(x,y) = K(x)\,K(y)$$

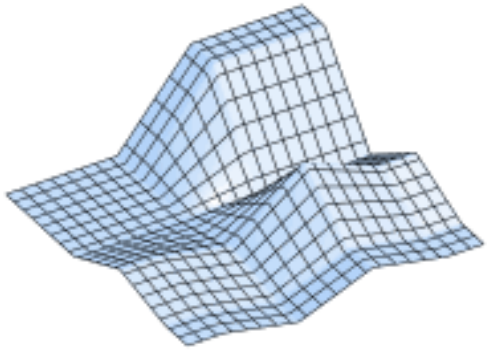
nearest neighbor



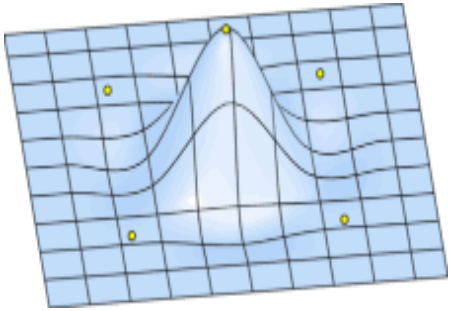


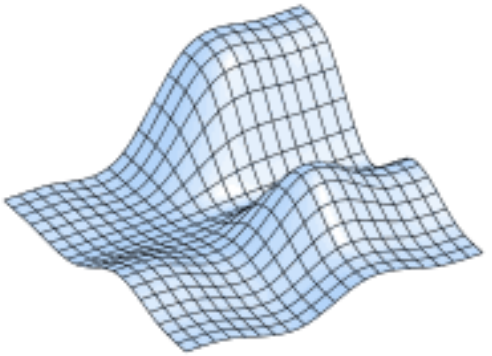
bilinear





bicubic

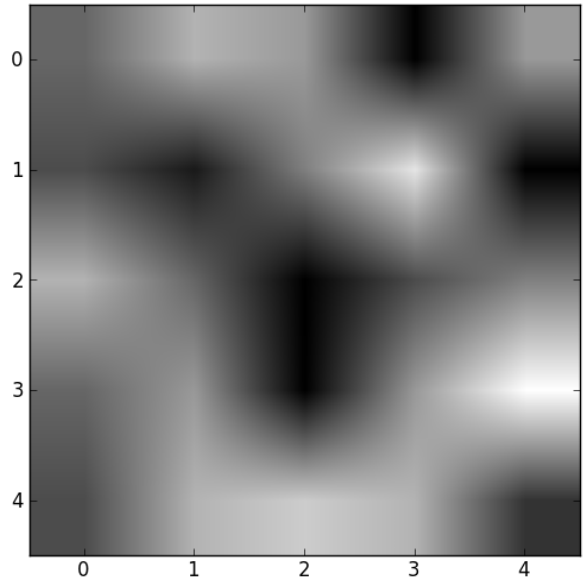




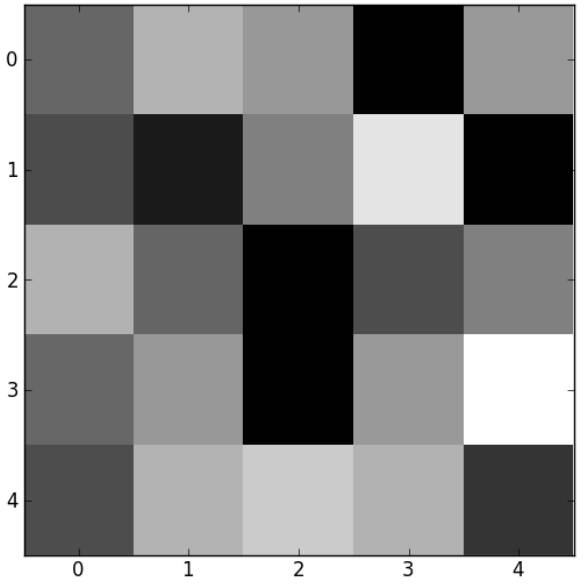
source: http://www.ipol.im/pub/art/2011/g_lmii/

Python plotting

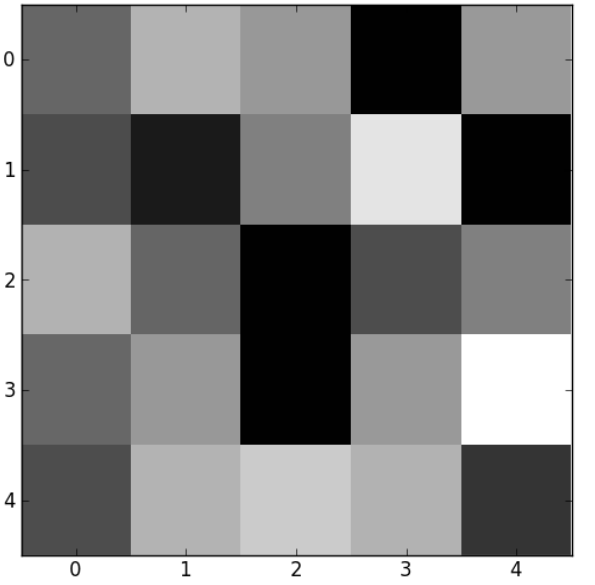
plt.imshow(im)



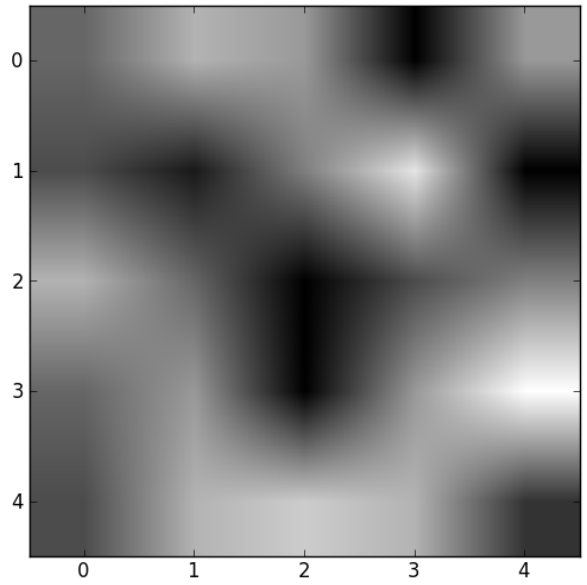
plt.imshow(im, interpolation='none')



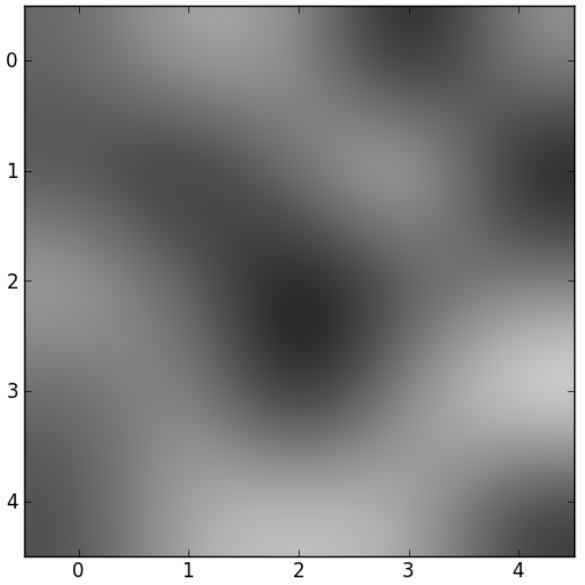
plt.imshow(im, interpolation='nearest')



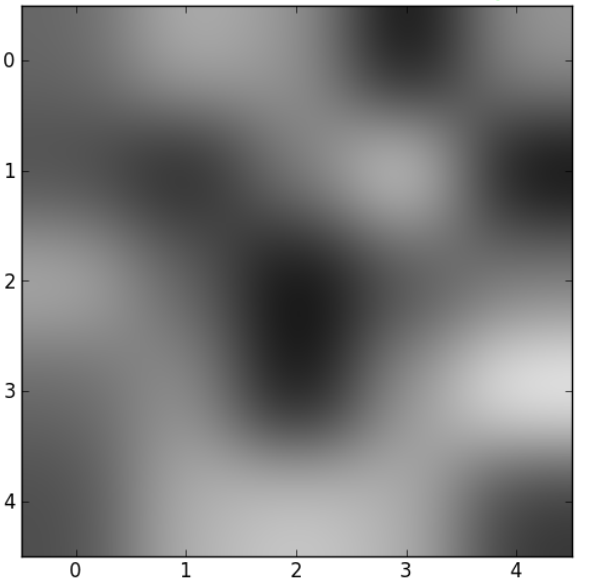
plt.imshow(im, interpolation='bilinear')



plt.imshow(im, interpolation='bicubic')

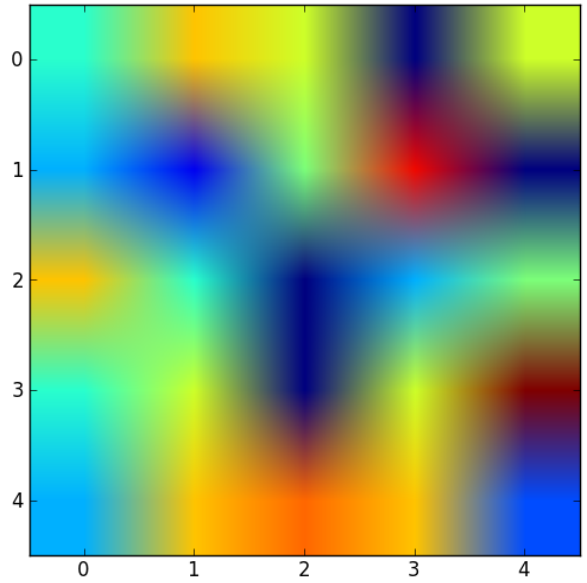


plt.imshow(im, interpolation='gaussian')

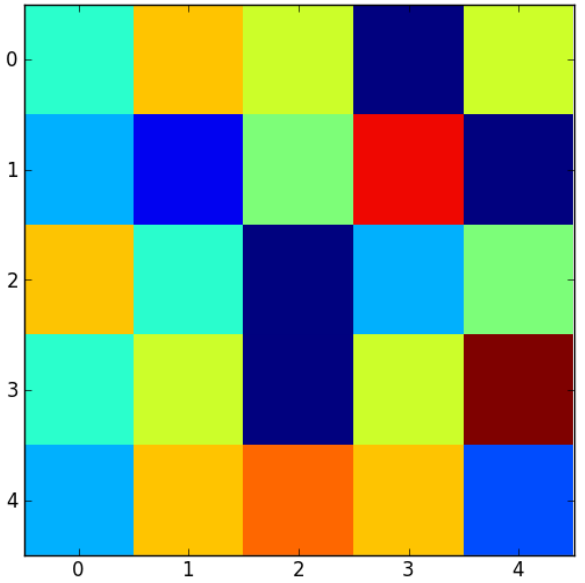


Python plotting

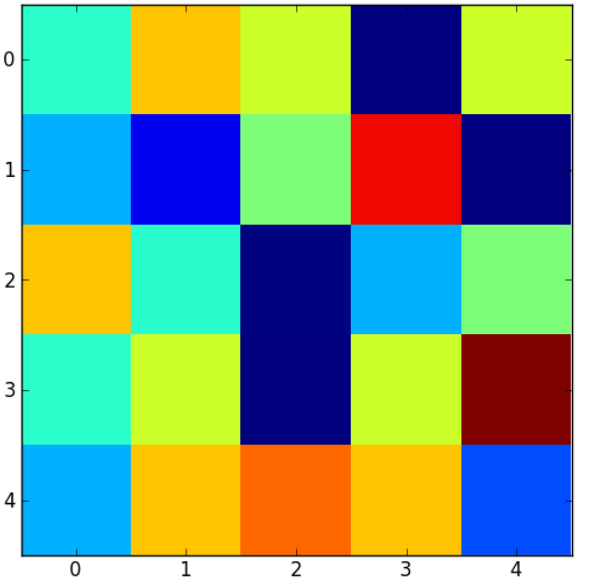
plt.imshow(im)



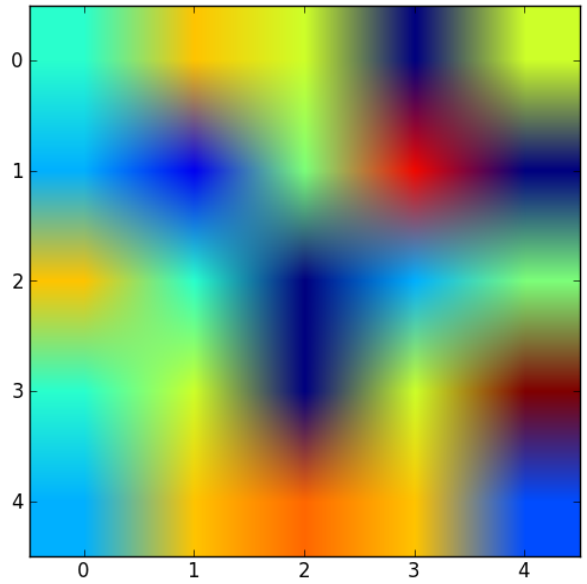
plt.imshow(im, interpolation='none')



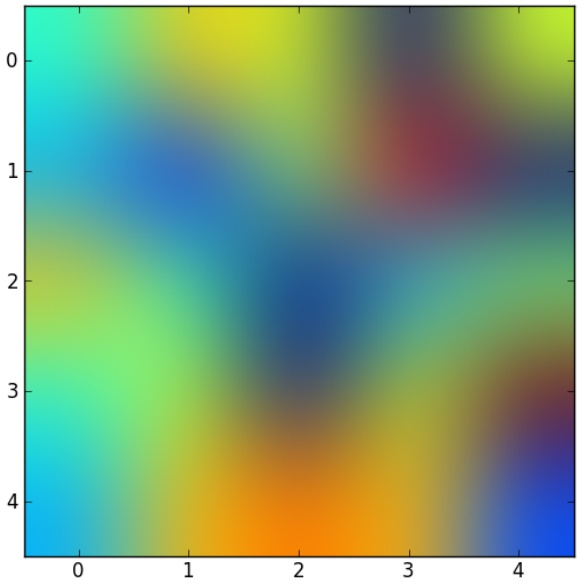
plt.imshow(im, interpolation='nearest')



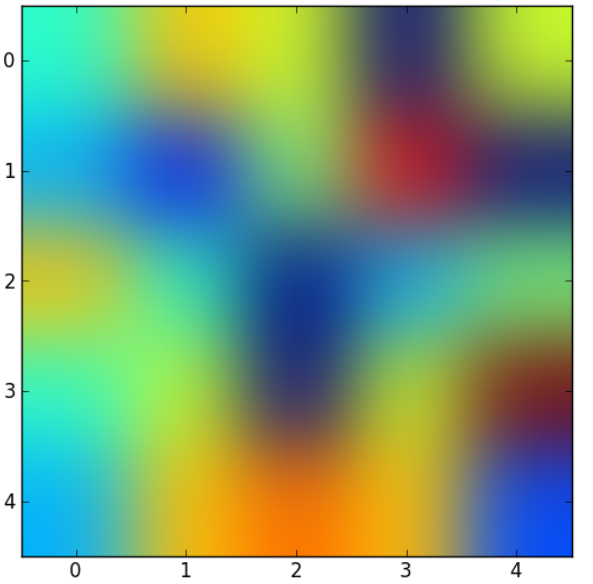
plt.imshow(im, interpolation='bilinear')



plt.imshow(im, interpolation='bicubic')



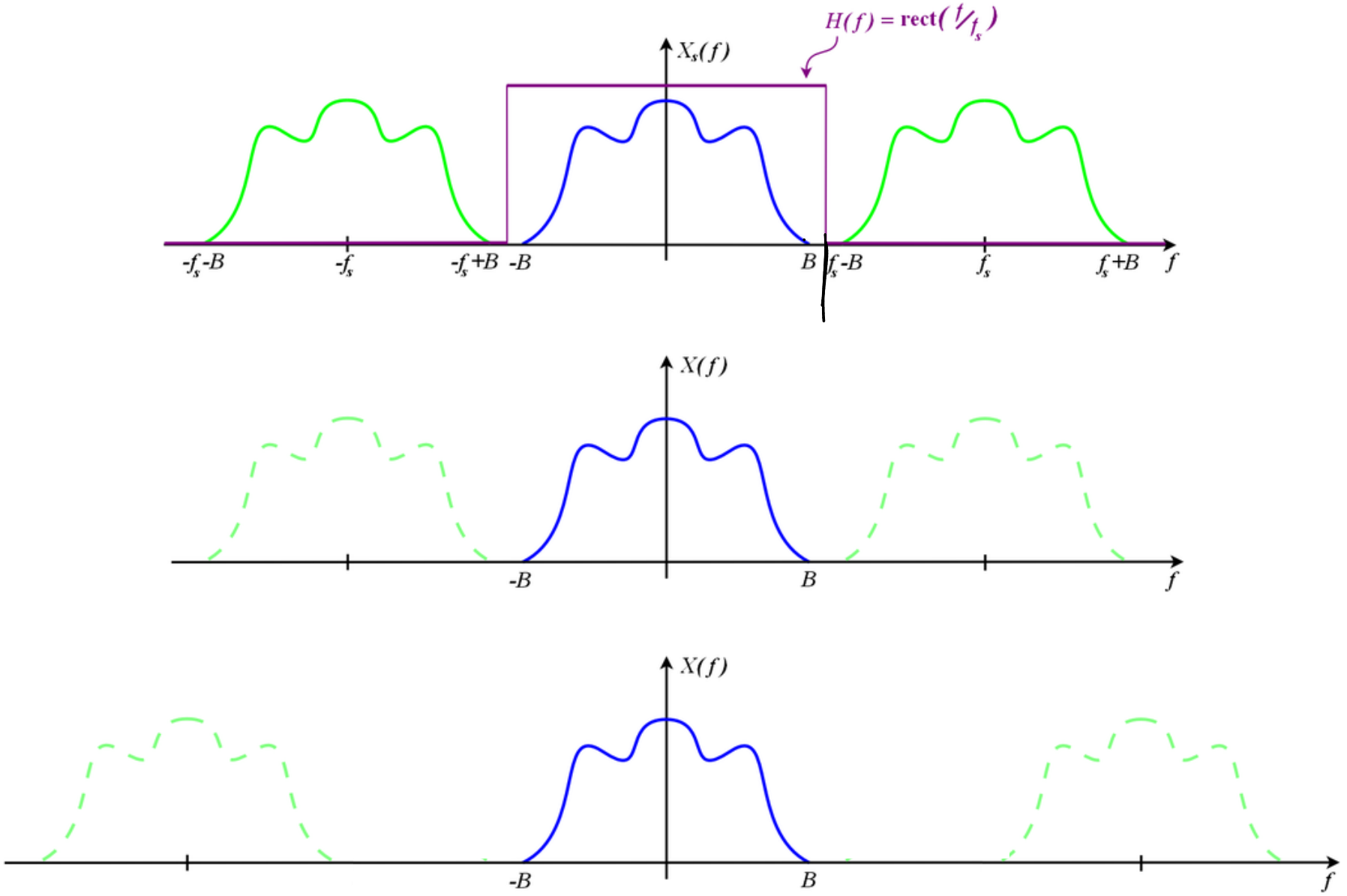
plt.imshow(im, interpolation='gaussian')



Sampling and Interpolation

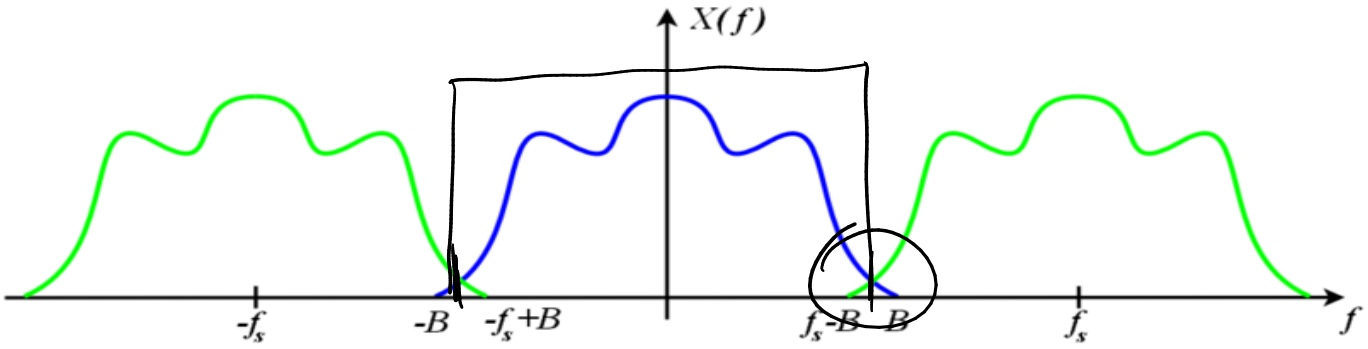
Sinc interpolation and zero-padding

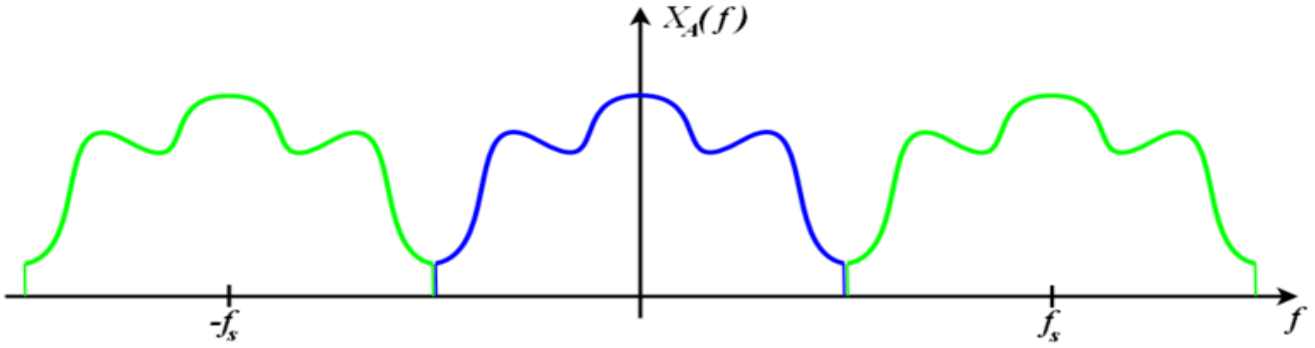
Also known as "Whittaker–Shannon interpolation"

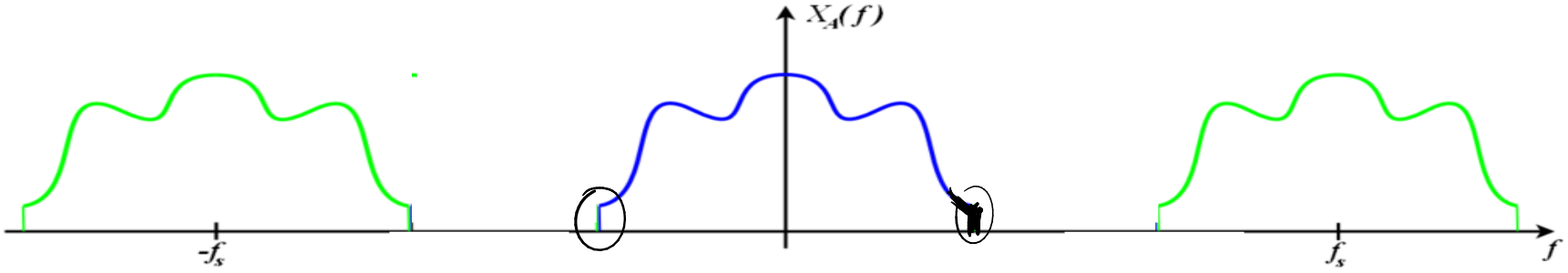


Sinc interpolation and zero-padding

Also known as "Whittaker–Shannon interpolation"







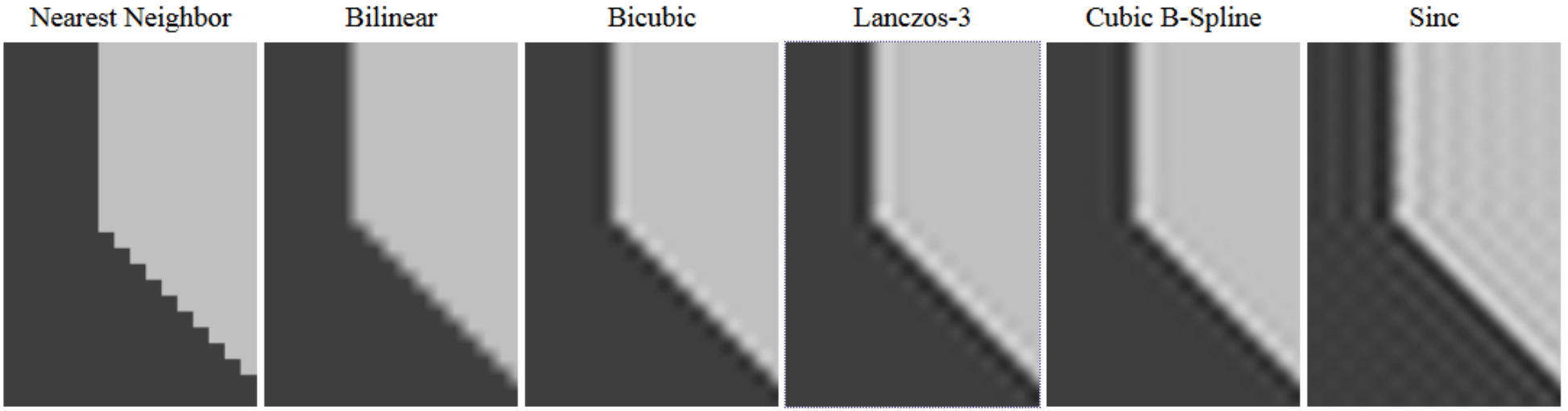
Reconstruction from samples

- Sinc interpolation can perfectly reconstruct a function from its samples if
  - sampled at a rate higher than Nyquist rate
  - bandlimited up to Nyquist frequency
  - no aliasing

equivalent

- Sinc interpolation introduces ringing otherwise, due to leakage of aliased frequencies

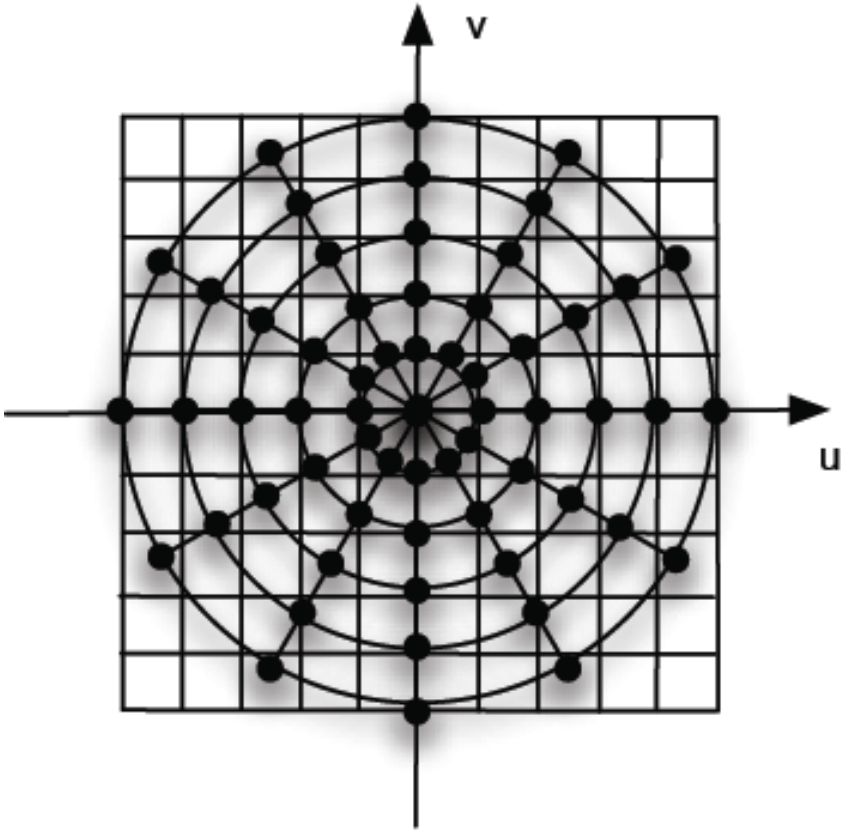
good compromise for visualization



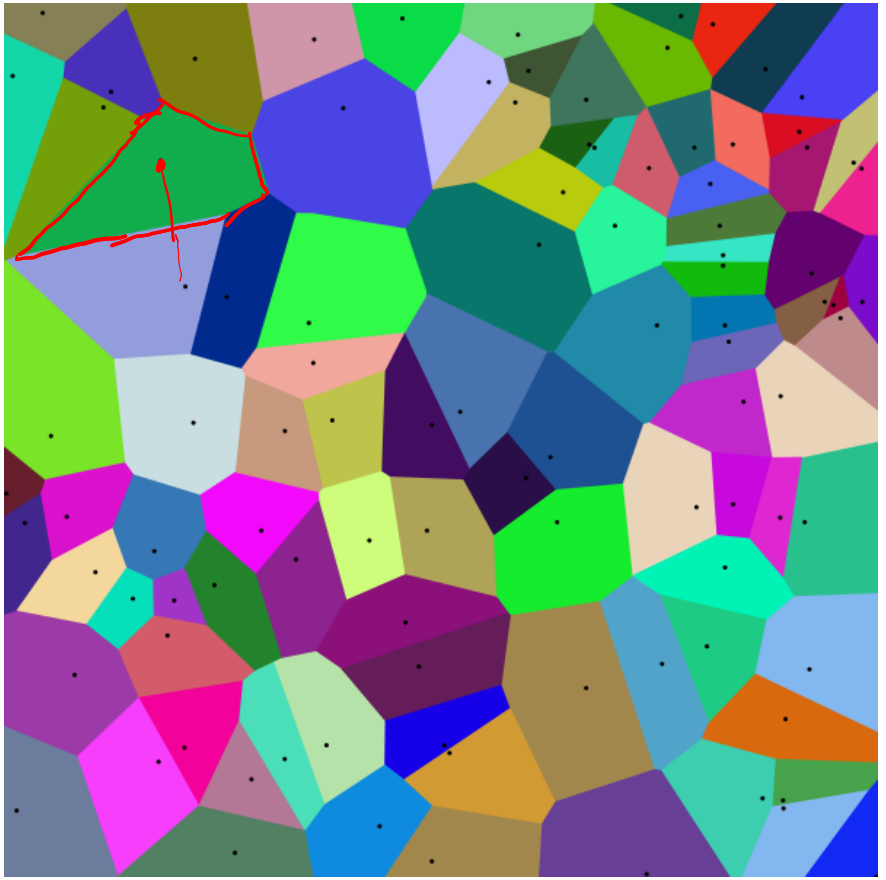
Linear interpolation of a step edge: a balance between staircase artifacts and ripples.

Other Interpolation

- Change from polar to cartesian grid
- Linear, but not translation invariant



polar vs. cartesian sampling



irregular sampling

Summary

- Images can be represented as a sampling grid and pixel basis functions
- Need for interpolation arises when changing the grid
- Linear and translation invariant interpolation can be written as a convolution with an interpolation kernel function
- Typical interpolation kernels include nearest neighbor, linear, cubic and higher B-spline interpolation
- Zero-padding in one domain equals sinc interpolation in the other
- "ideal" sinc interpolation may lead to ringing artifacts