UNIVERSITY OF MINNESOTA
Driven to Discover℠

EE 5561: Image Processing and Applications

Lecture 6

Mehmet Akçakaya

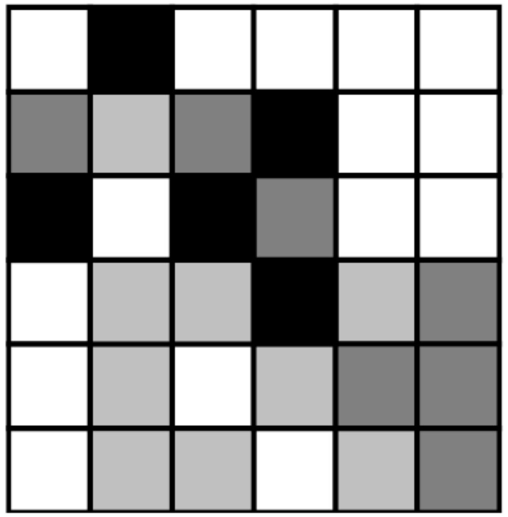
Recap of Last Lecture

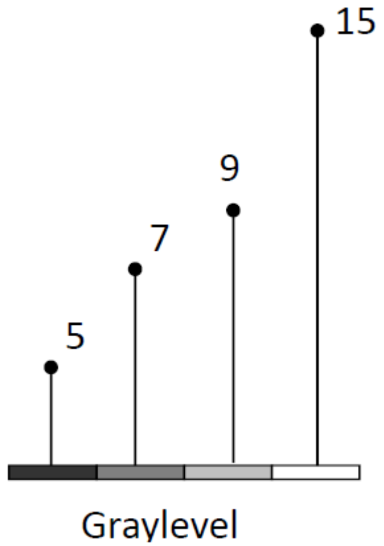
- Operations on images
- Concept of image enhancement
- Noise smoothing (uses low-pass filters)
- Image sharpening (uses high-pass filters)
- Edge detection (as very basic image segmentation)

- Today: More image enhancement

Histogram Processing

- Histogram ~ estimate of the probability density function (PDF) of underlying random variable
  - For images: grey level/intensities
  - e.g.



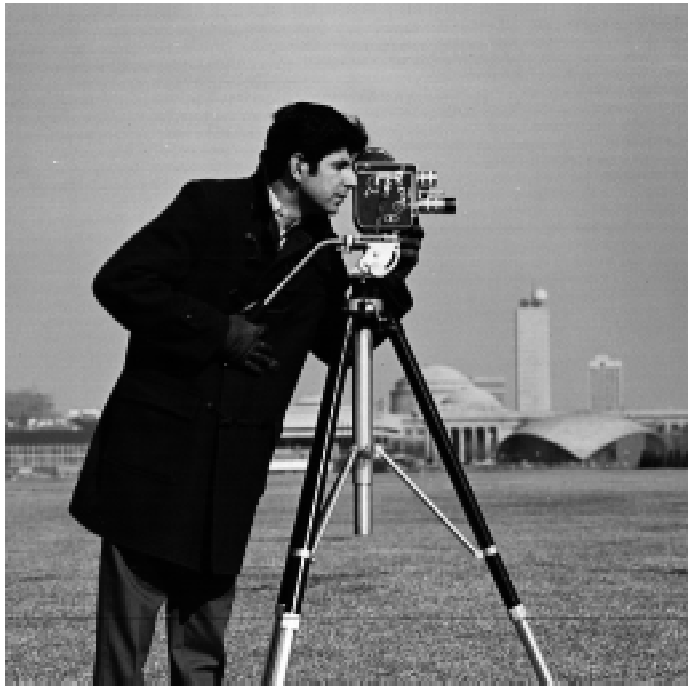


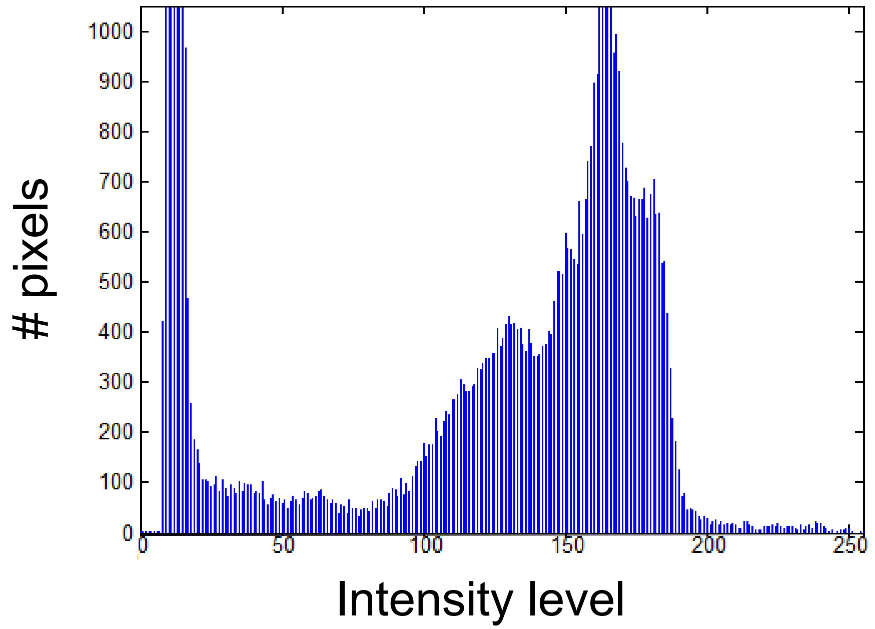
$\mathbb{P}(\text{pixel is black}) = 5/36$

  - Histogram is often normalized to better represent probabilities

Histogram

- We usually display the histogram as a bar plot showing probabilities that a random variable falls into each set of intervals/bins
- Some software use the unnormalized version (i.e. # pixels)





Histogram

- Histogram is a simple form of density estimation
- Let X be the random variable describing the intensities in an image
- Recall the cumulative distribution function (CDF) of X

$$F_X(x) = \mathbb{P}(X \leq x)$$

- The pdf is then given by (assuming intensities are continuous)

$$f_X(x) = \frac{dF_X(x)}{dx}$$

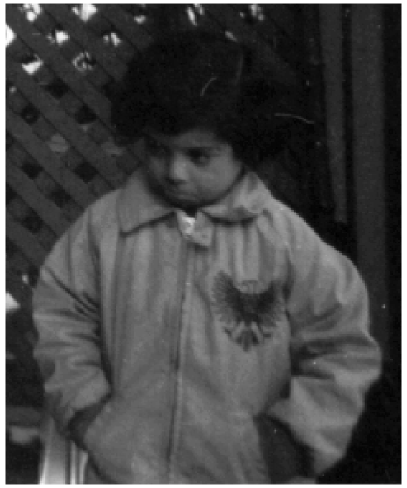
- Then the "binning" process corresponds to

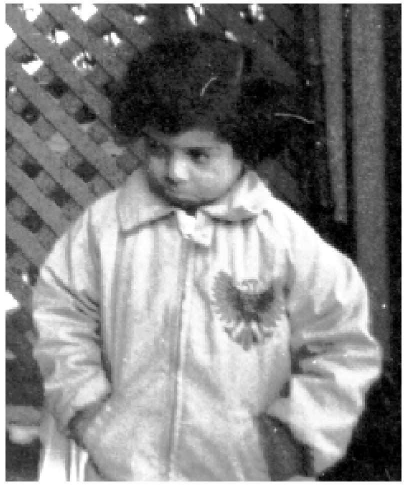
$$p_k = \mathbb{P}(t_{k-1} < X \leq t_k\} = \int_{t_{k-1}}^{t_k} f_X(x)dx$$

- The histogram instead estimates these as

$$\hat{p}_k = \frac{\#\text{ of X's} \in (t_{k-1}, t_k]}{n} \qquad k = 1, \ldots, K$$

Histogram Normalization

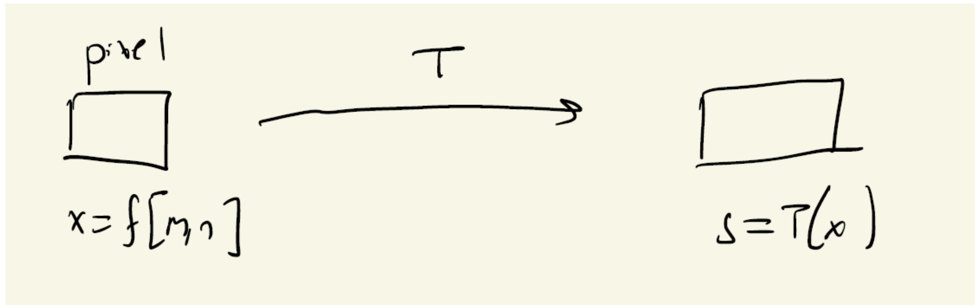




- Goal: Better distribution of intensities
  - Areas of lower local contrast changed to gain higher contrast
  - Without affecting global contrast
  - “Better” may be subjective

Histogram Normalization

- General idea: Map histogram of given image to a desired histogram by a non-linear single-pixel operation



- Since histogram ~ PDF → CDF/PDF transformations

Histogram Normalization

1. Treat histogram of image as a PDF, $f_X(x), \quad x \in [0,1]$. Thus
2. The mapping of interest is some $T : [0,1] \to [0,1]$
3. Let the desired PDF is f_d , with corresponding CDF F_d
4. Find $Z = T(X) \quad \text{such that} \quad Z \sim f_d$

We know from probability that we need

$$Z = F_d^{-1}(F_X(X))$$

Why?

$$F_Z(z) = \mathbb{P}(Z \leq z) = \mathbb{P}\left(F_d^{-1}(F_X(X)) \leq z\right) = \mathbb{P}\left(F_X(X) \leq F_d(z)\right)$$

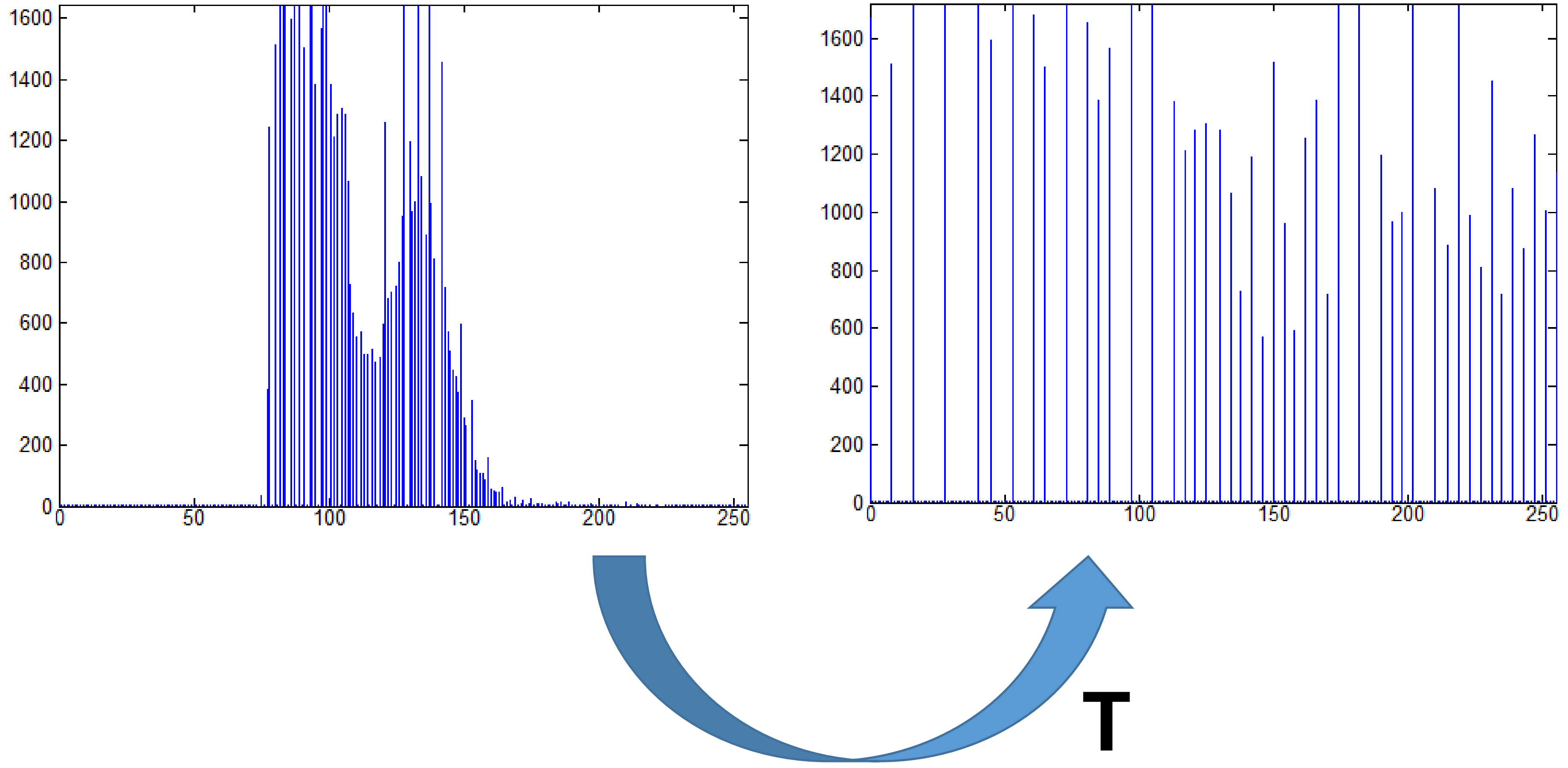
$$= \underbrace{\mathbb{P}\left(X \leq F_X^{-1}(F_d(z))\right)}_{\text{CDF of } X \text{ evaluated at } F_X^{-1}(F_d(z))} = F_X\left(F_X^{-1}(F_d(z))\right) = F_d(z)$$

Thus Z has the desired CDF

Histogram Equalization

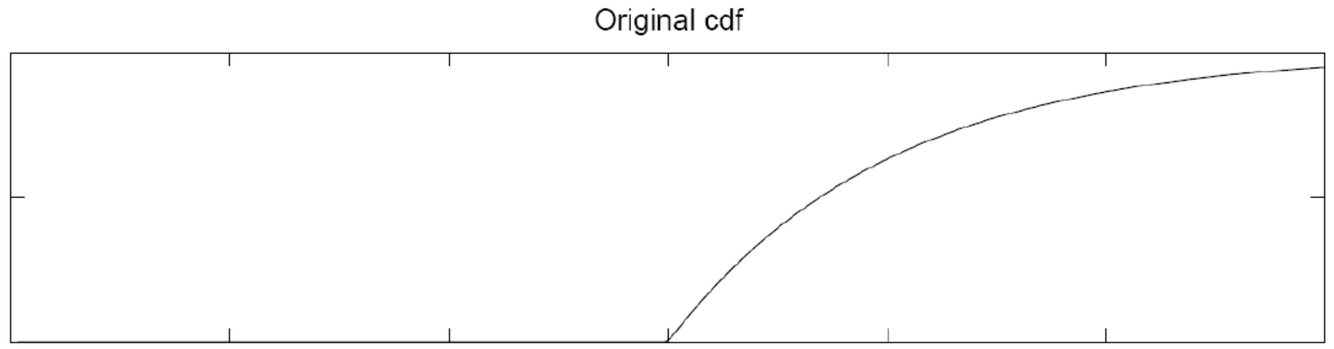
- What’s the desired F_d?
  - One option: Design *T* such that the “normalized” CDF is linearized across the intensity value range (e.g. [0, 255])
  - Why?
    - Flat PDF
  - Another criterion: T must be monotone increasing
    - Keep black & white consistent in the transformed image

Histogram Equalization

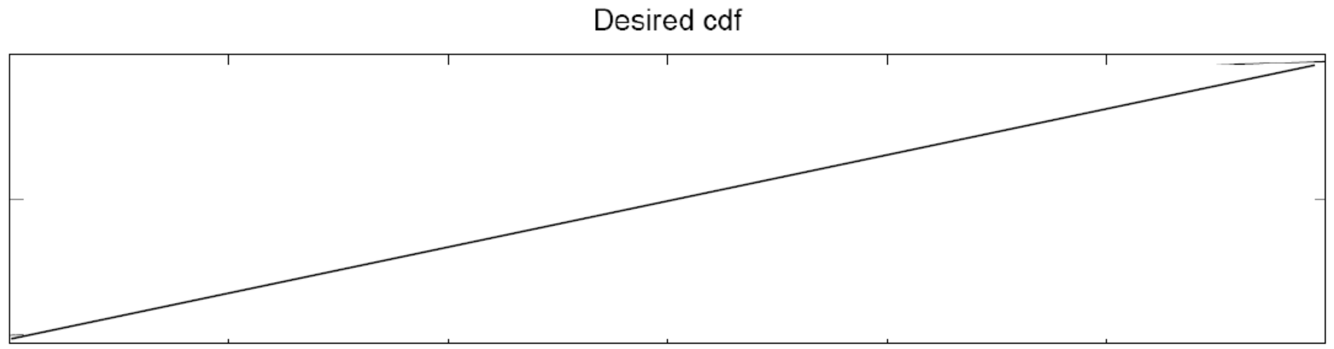


Histogram Equalization

$F(x)$



$F_d = K\,x$



Fessler

Histogram Equalization

- Pseudo-code
  - Find histogram hist[i], i = 0, …, 255

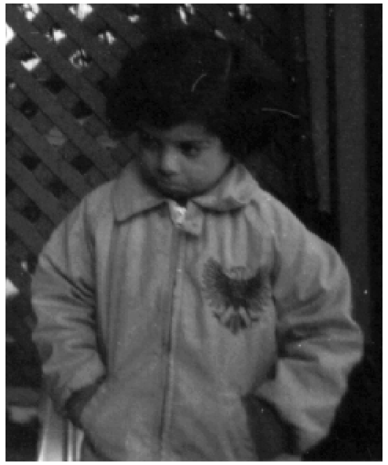
```
sum = 0
for i = 0:255
      sum = sum + hist[i]
      lookup[i] = sum * 255 + 0.5
end
y[i,j] = lookup(x[i,j])
```

lookup table

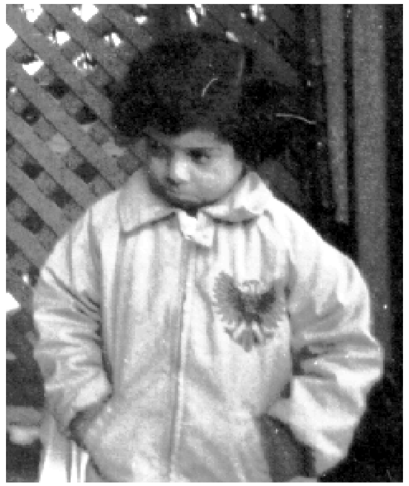
image mapping

Histogram Equalization

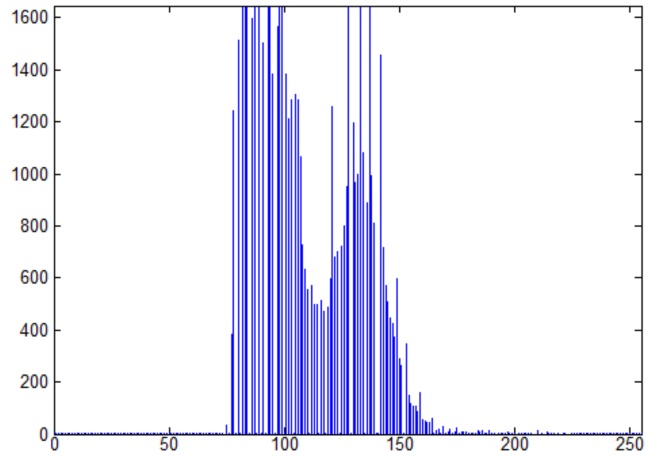
pout - original



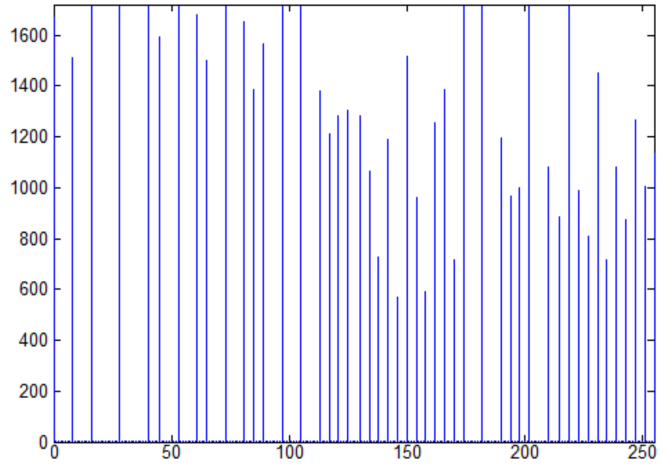
pout – after histeq



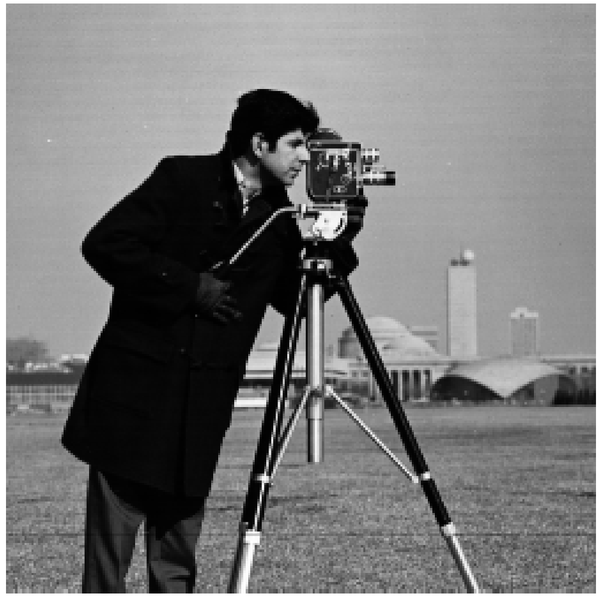
Histogram of pout - original

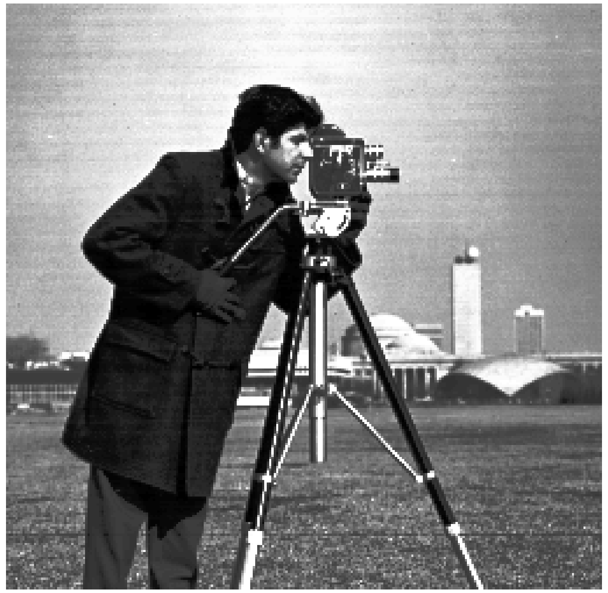


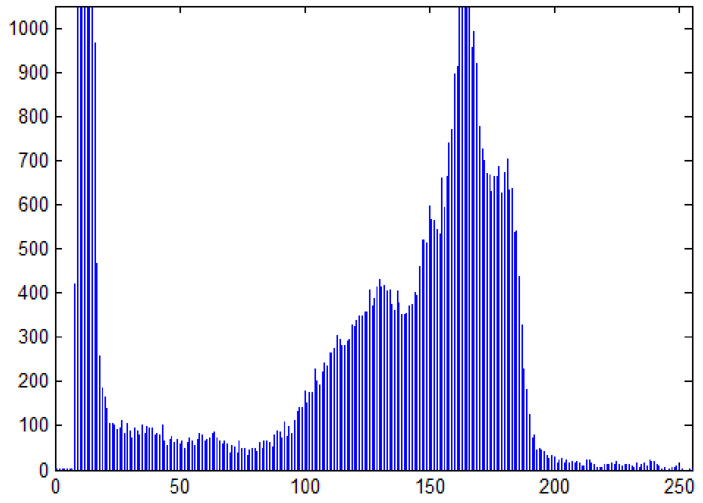
Histogram of pout – after histeq

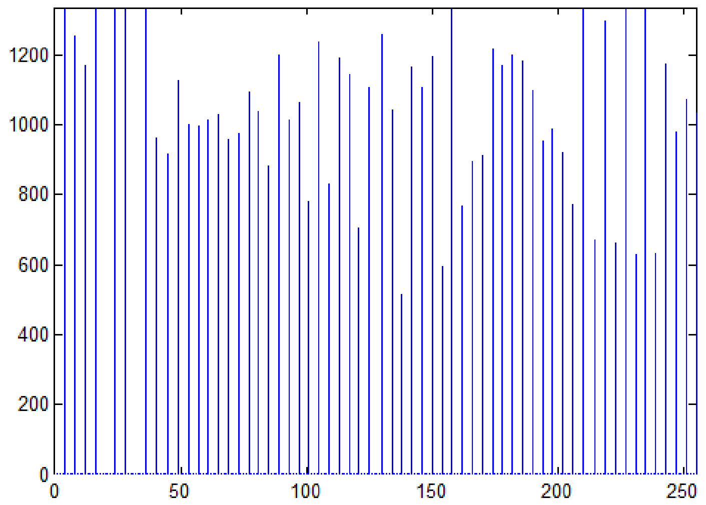


Histogram Equalization





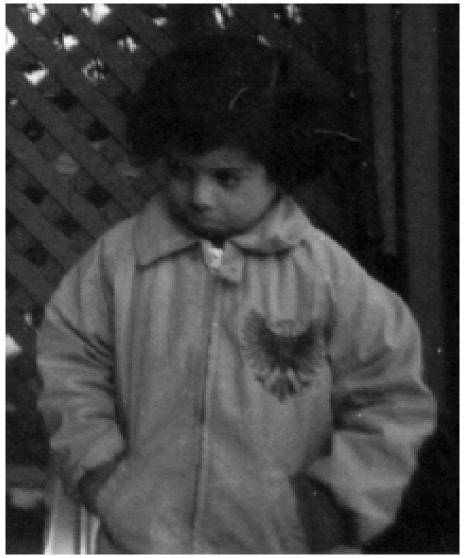


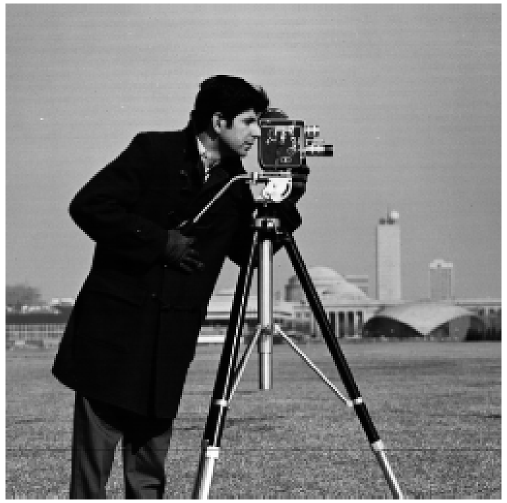


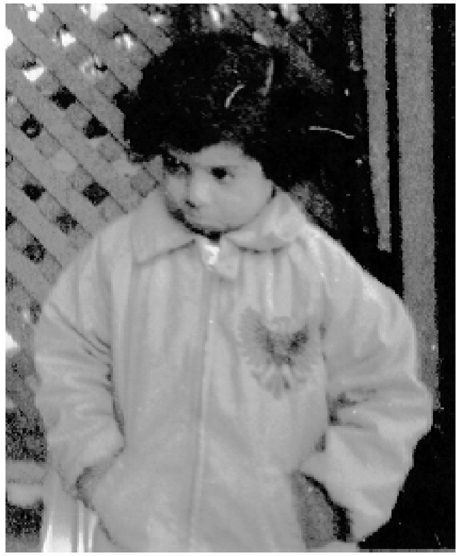
Histogram Matching

- Generalization of histogram equalization
  - Instead of “flattening” the histogram, make histogram of A look like histogram of B
  - Procedurally:
    - First flatten A and B with transformation T and U
    - Then desired transformation is $U^{-1}(T(\mathrm{A}))$

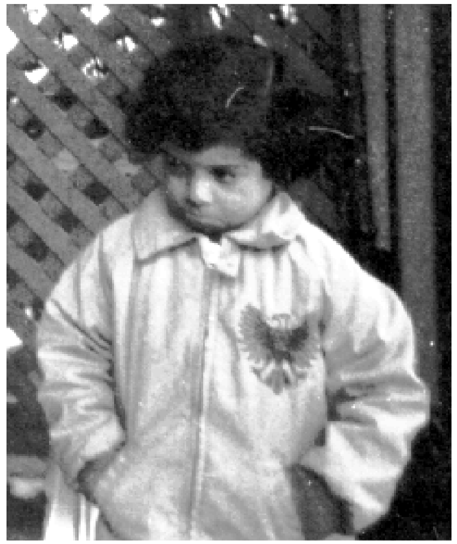
Histogram Matching



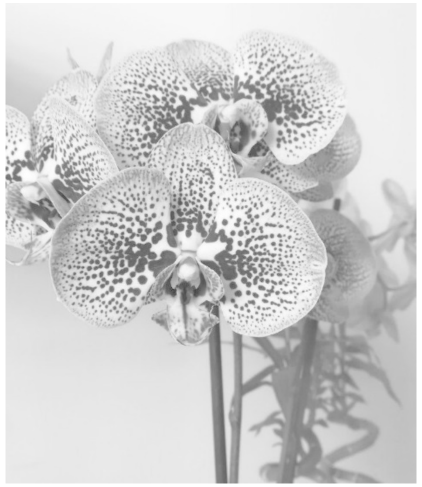


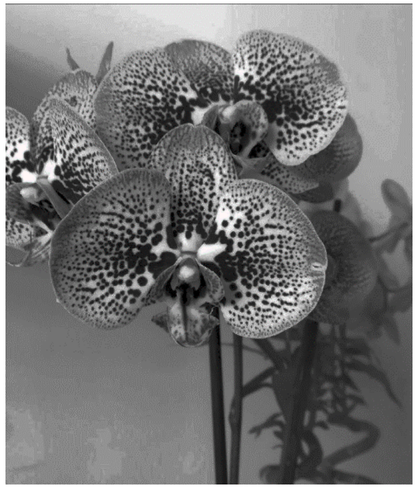


instead of (equalized)

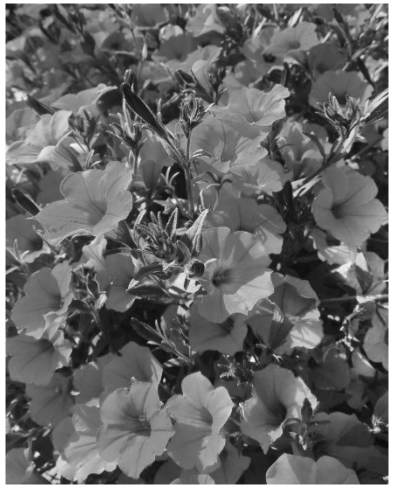


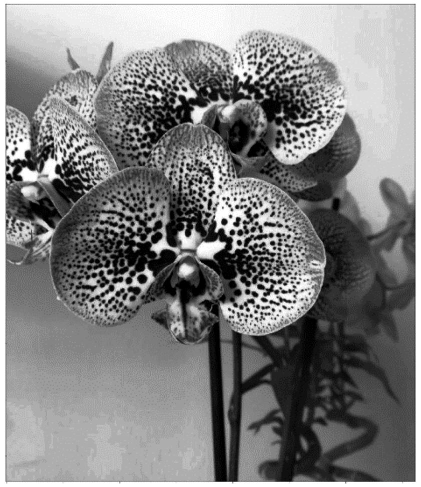
Histogram Matching





instead of (equalized)

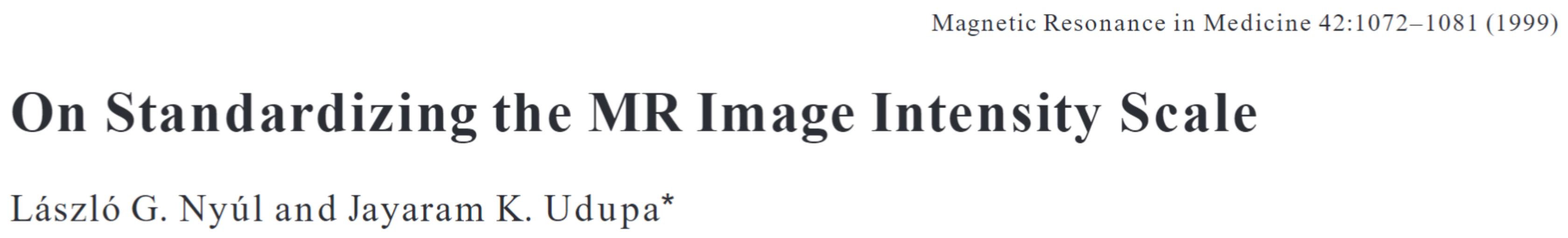




https://towardsdatascience.com/histogram-matching-ee3a67b4cbc1

Histogram processing applications

- A related concept is crucial as a pre-processing step for analysis of MRI data
  - Images in MRI are acquired in arbitrary units (not well-defined scientific units)
  - Thus, variations exist across different scanner vendors, subjects and visits (even with the same acquisition protocol)
  - It affects things as simple as thresholding
  - Intensity normalization brings the intensities to a common scale



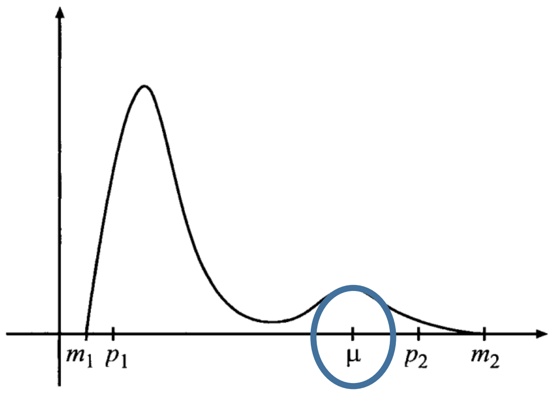
Magnetic Resonance in Medicine 42:1072–1081 (1999)

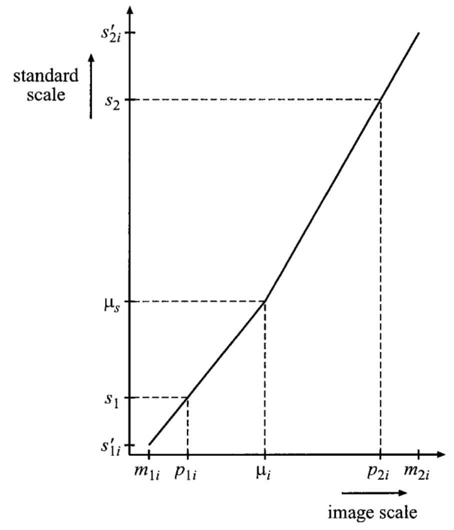
On Standardizing the MR Image Intensity Scale

László G. Nyúl and Jayaram K. Udupa*

Histogram processing applications

- Nyul's method has two steps:
  - Use training data (images from a population of subjects) to find landmarks of a standard histogram

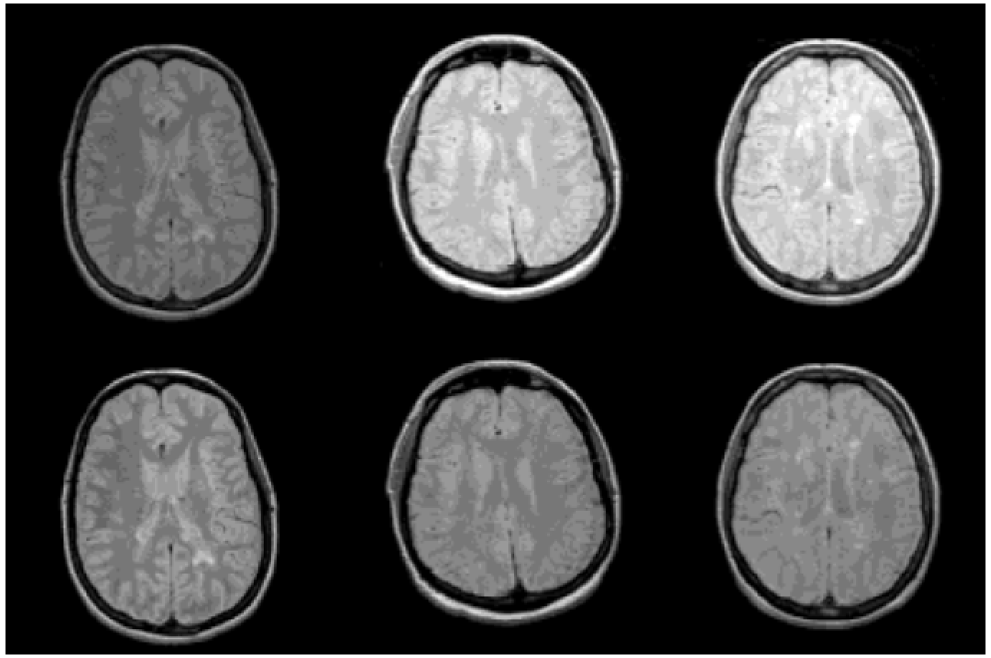


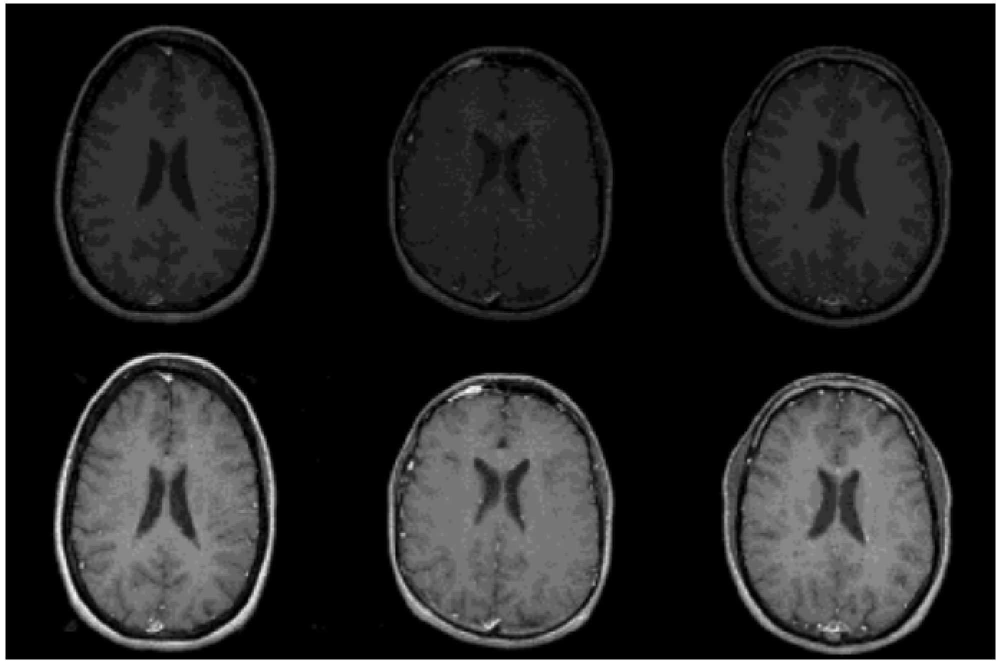


  - New data is transformed to match the standard histogram

Histogram processing applications

- Nyul’s method (top: acquired, bottom: normalized):





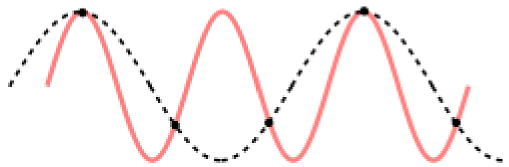
- More complicated methods exist for intensity normalization

Interpolation

- As discussed earlier
  - We work on discrete-space image $f_d[m,n]$
  - These are typically samples of continuous-space image $f_a(x,y)$
- Sometimes we want to compute values of $f_a(x,y)$ at locations other than the sampling locations
  - Why?
    - Display/zoom
    - Registration/morphing
    - Motion correction

Interpolation

- Is perfect recovery possible?
  - Not unless some assumptions are made on $f_a(x,y)$
  - Otherwise uncountably many $f_a(x,y)$ that agree with $f_d[m,n]$



  - For sampling we saw
    - For band-limited signals & sampling rate satisfying Nyquist criterion

$$f_a(x,y) = \sum_{m,n} f_d[m,n] \underbrace{h\Big(x - m\Delta_x, y - n\Delta_y\Big)}_{h(x,y)=\mathrm{sinc}\left(\frac{x}{\Delta_x}\right)\mathrm{sinc}\left(\frac{y}{\Delta_y}\right)}$$

Interpolation

- So we're done?
- Not quiet…
  - sinc is unbounded
  - summations require infinitely many samples
  - Images need not be band-limited (though we pre-filter usually)

- We want new $h(x, y)$ $\longleftarrow$ interpolation kernel

$$f_r(x,y) = \sum_{m,n} f_d[m,n] h\Big(x - m\Delta_x, y - n\Delta_y\Big)$$

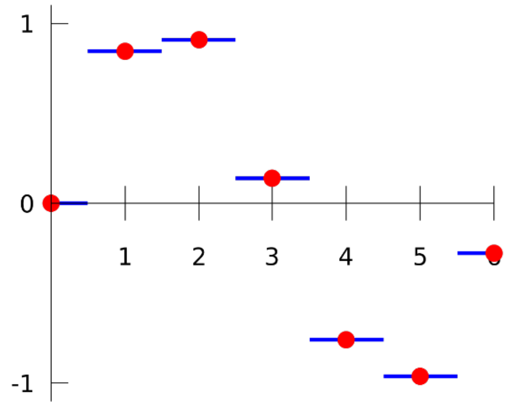
- Note this process of interpolation is "linear" (as we defined earlier)
  - Output is a linear combination of the samples
  - Linear in quotes since "linear interpolation" usually refers to something specific

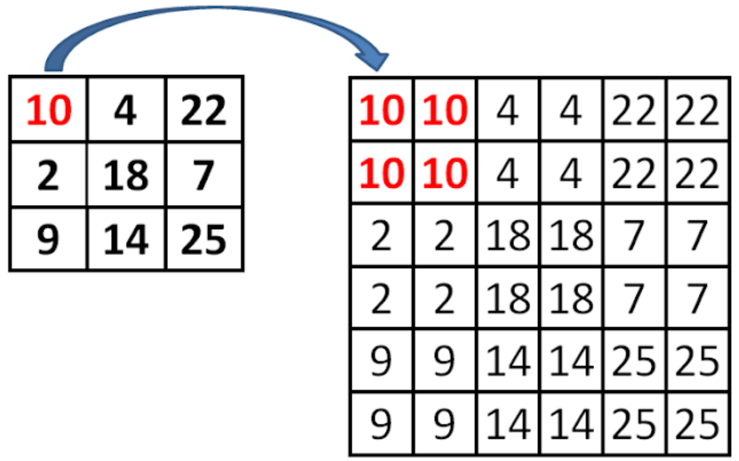
Interpolation

- What do we want for $h(x, y)$?
  - Bounded support
  - Finitely many samples
  - Computationally easy

Nearest Neighbor

- Simplest: Nearest neighbor interpolation (or zero-order hold)





10	**4**	**22**
2	**18**	**7**
9	**14**	**25**

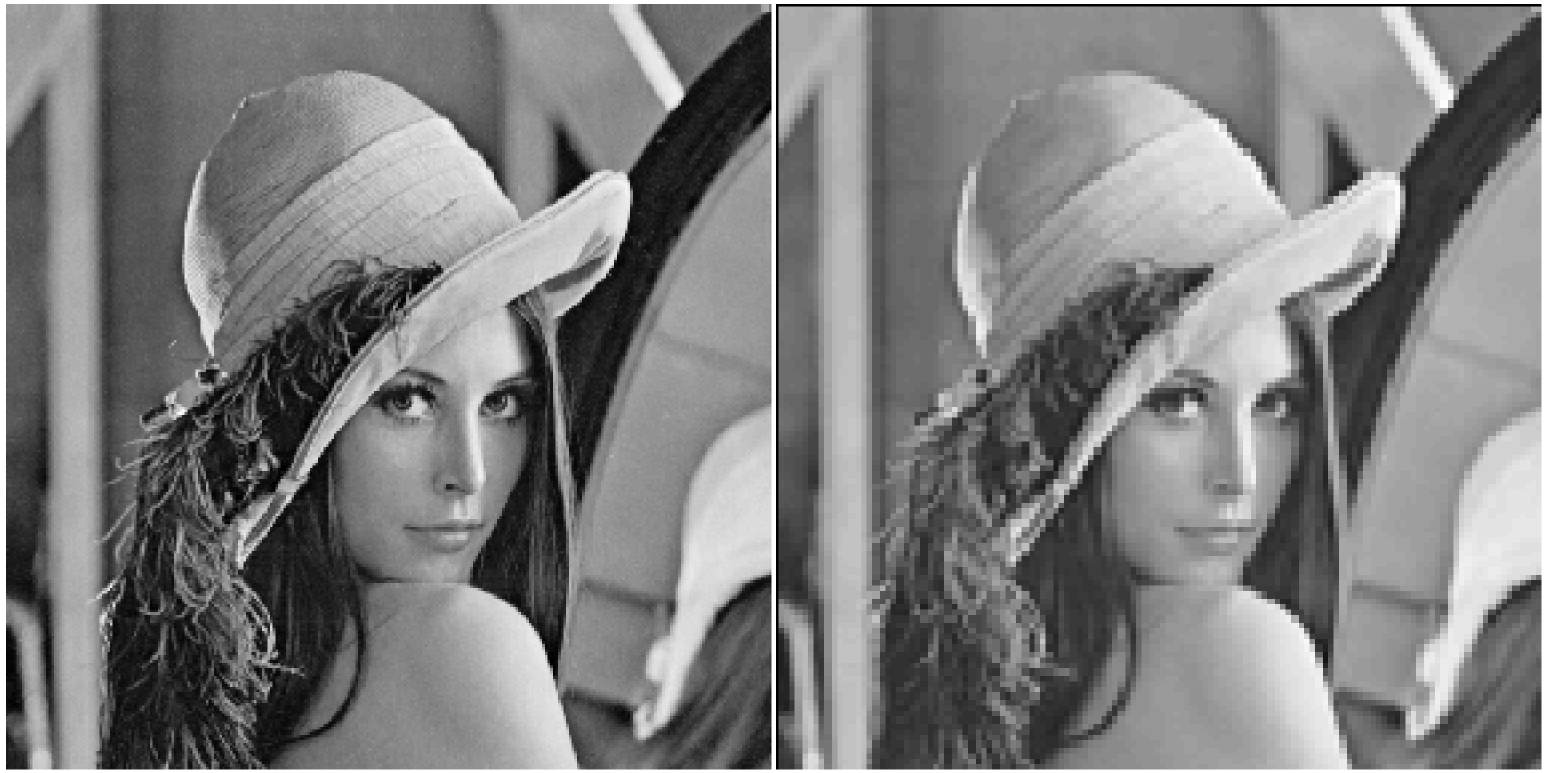
10	10	4	4	22	22
10	10	4	4	22	22
2	2	18	18	7	7
2	2	18	18	7	7
9	9	14	14	25	25
9	9	14	14	25	25

  - Interpolation kernel:

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

$$h(x, y) = \mathrm{rect}\left(\frac{x}{\Delta_x}, \frac{y}{\Delta_y}\right)$$

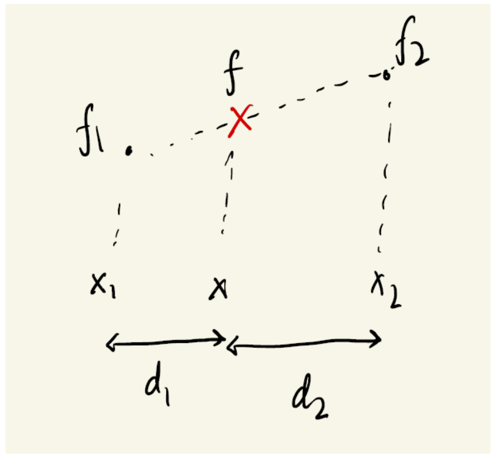
Nearest Neighbor



Downsample & resize

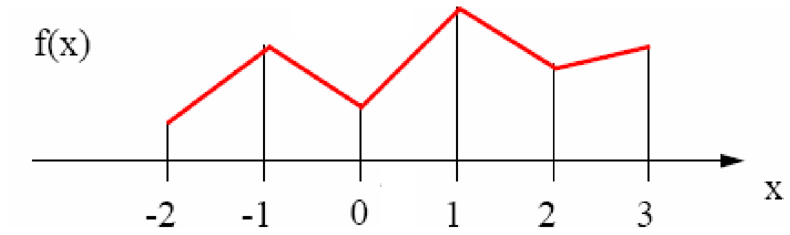
Bilinear Interpolation

- Linear/bilinear interpolation
  - Second meaning to linear
  - 1D case



$$\underbrace{\frac{f - f_1}{x - x_1}}_{d_1} = \underbrace{\frac{f_2 - f}{x_2 - x}}_{d_2}$$

$$f = \frac{d_1 f_2 + d_2 f_1}{d_1 + d_2}$$

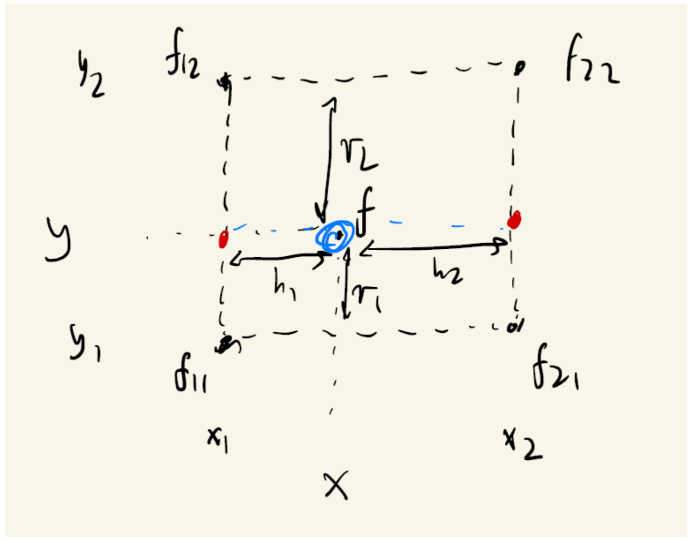


If $[x_1, x_2] = [0, 1]$ then

$$f = f_2 x + (1 - x) f_1 \text{ for } 0 < x < 1$$

Bilinear Interpolation

- Linear/bilinear interpolation
  - 2D case (bilinear)



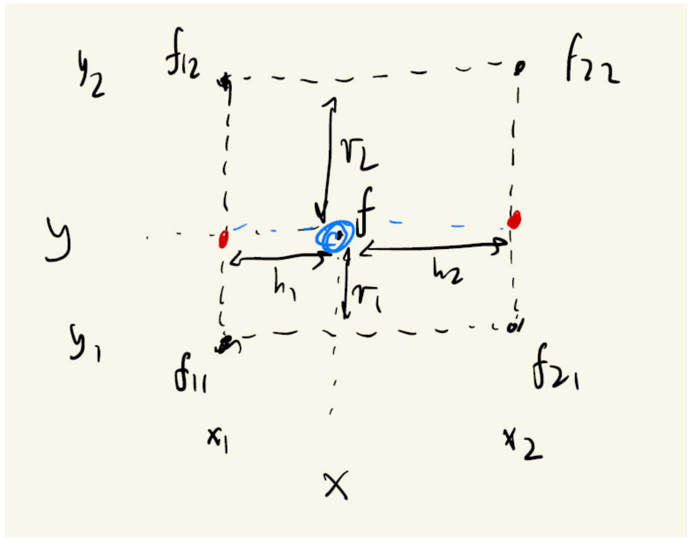
$$f(x_1, y) = \frac{v_1 f(x_1, y_2) + v_2 f(x_1, y_1)}{v_1 + v_2}$$

$$f(x_2, y) = \frac{v_1 f(x_2, y_2) + v_2 f(x_2, y_1)}{v_1 + v_2}$$

$$f(x, y) = \frac{h_1 f(x_2, y) + h_2 f(x_1, y)}{h_1 + h_2} = \ldots$$
$$= \frac{h_1 v_1 f_{22} + h_1 v_2 f_{21} + h_2 v_1 f_{12} + h_2 v_2 f_{11}}{(h_1 + h_2)(v_1 + v_2)}$$

Bilinear Interpolation

- Linear/bilinear interpolation
  - 2D case (bilinear)

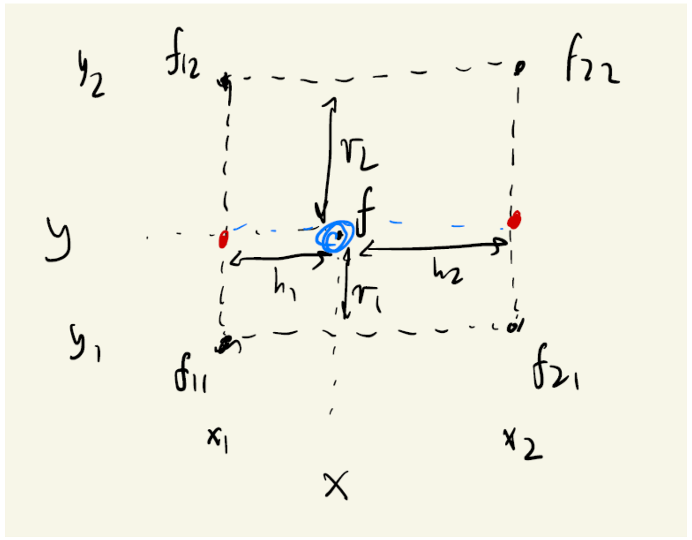


Again it will help to evaluate these on the unit square

$$\begin{aligned} f(x,y) &= (1-x)(1-y)f[0,0] + (1-x)yf[0,1] \\ &\quad + x(1-y)f[1,0] + xyf[1,1] \\ &= \begin{bmatrix} 1-x & x \end{bmatrix} \begin{bmatrix} f[0,0] & f[0,1] \\ f[1,0] & f[1,1] \end{bmatrix} \begin{bmatrix} 1-y \\ y \end{bmatrix} \end{aligned}$$

Bilinear Interpolation

- Linear/bilinear interpolation
  - 2D case (bilinear)



Alternatively

$$f(x,y) = a_0 + a_1 x + a_2 y + a_3 xy$$

Solve for the coefficients using

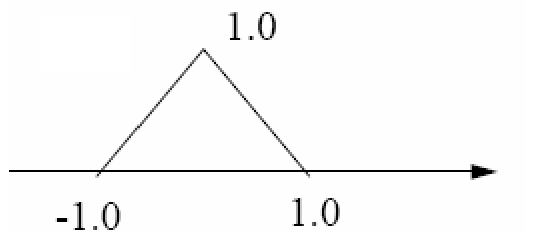
$$\begin{bmatrix} 1 & x_1 & y_1 & x_1 y_1 \\ 1 & x_1 & y_2 & x_1 y_2 \\ 1 & x_2 & y_1 & x_2 y_1 \\ 1 & x_2 & y_2 & x_2 y_2 \end{bmatrix} \begin{bmatrix} a_0 \\ a_1 \\ a_2 \\ a_3 \end{bmatrix} = \begin{bmatrix} f_{11} \\ f_{12} \\ f_{21} \\ f_{22} \end{bmatrix}$$

Bilinear Interpolation

- Interpolation kernel:

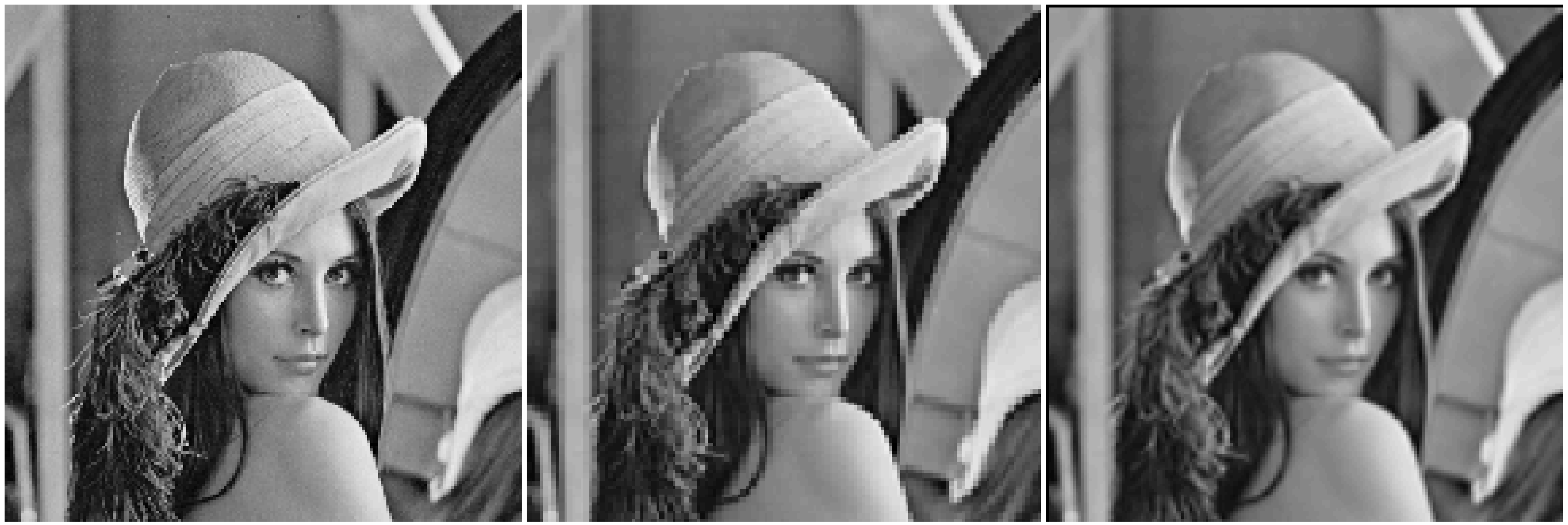
$$h(x) = 1 - |x| \triangleq \mathrm{tri}(x)$$

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



- Frequency response is sinc^2
  - Since triangle function is the convolution of rectangle function with itself

Bilinear Interpolation



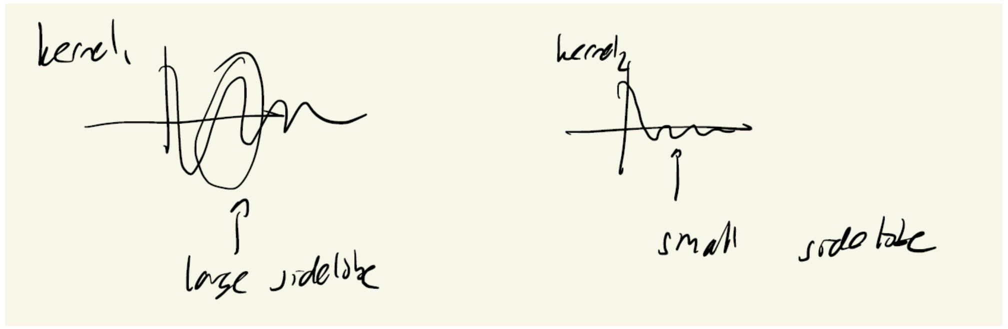
Downsample & resize

Other Interpolation

- Done?
  - Not smooth, not differentiable
- What do we want from interpolation?
  - Self-consistent $$f_r(x)\Big|_{x=n\Delta_x} = f_d[n]$$
    at samples
  - Continuous & differentiable $$\frac{df_r(x)}{dx} = \sum_n f_d[n] \frac{dh(x-n\Delta_x)}{dx}$$
  - Small support → fast

Other Interpolation

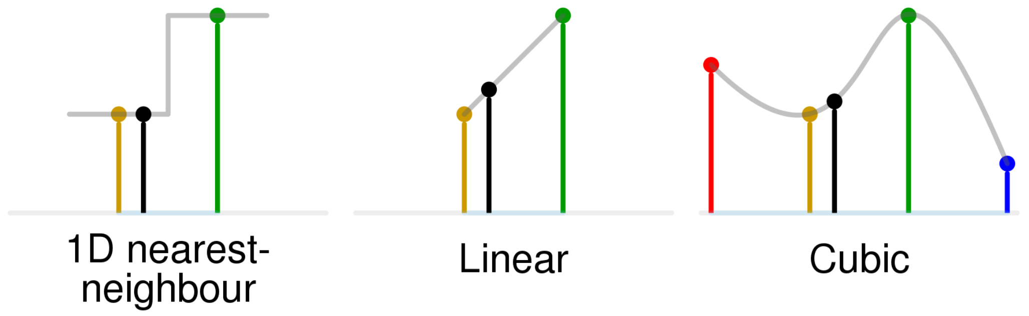
- Done?
  - Not smooth, not differentiable
- What do we want from interpolation?
  - Preferably
    - Frequency response ~ rect (ideal case)
    - Shift-invariant process (ensured by the convolutional setup above)
    - Avoid ringing → small sidelobes



Other Interpolation

- No clear solution, but several alternatives
  - Polynomial interpolation
    - e.g. bicubic

$$f(x,y) = a_0 + a_1 x + a_2 y + a_3 xy + a_4 x^2 y + \cdots + a_{14} x^3 + a_{15} y^3$$



    - Why did we skip quadratic?
      - Both have continuous derivatives, but cubic has derivable derivatives → smoother
      - Quadratics tend to have bigger sidelobes

https://en.wikipedia.org/wiki/Bicubic_interpolation

Other Interpolation

- No clear solution, but several alternatives
  - B-spline (basis spline) interpolation
    - *n*-order differentiable
    - small support
    - spline of degree *n* → each segment is a polynomial of degree *n*
  - So need *n*+1 coefficients to describe each segment?
  - We add a smoothness constraint
    - Continuity of splines & its derivatives up to order (*n*-1) at the endpoints (i.e. samples)
    - 1 degree of freedom per support!

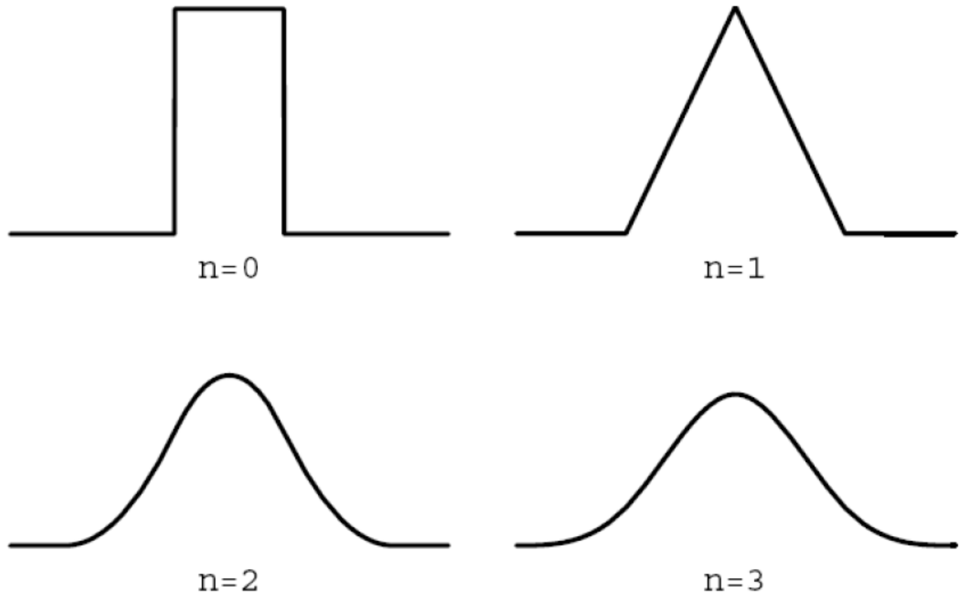
Other Interpolation

– B-spline (basis spline) interpolation

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

$$\beta^{(0)}(x) = \begin{cases} 1 & 1/2 \leq x < 1/2 \\ 0 & \text{otherwise} \end{cases}$$

i.e. rect



Other Interpolation

– B-spline (basis spline) interpolation

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

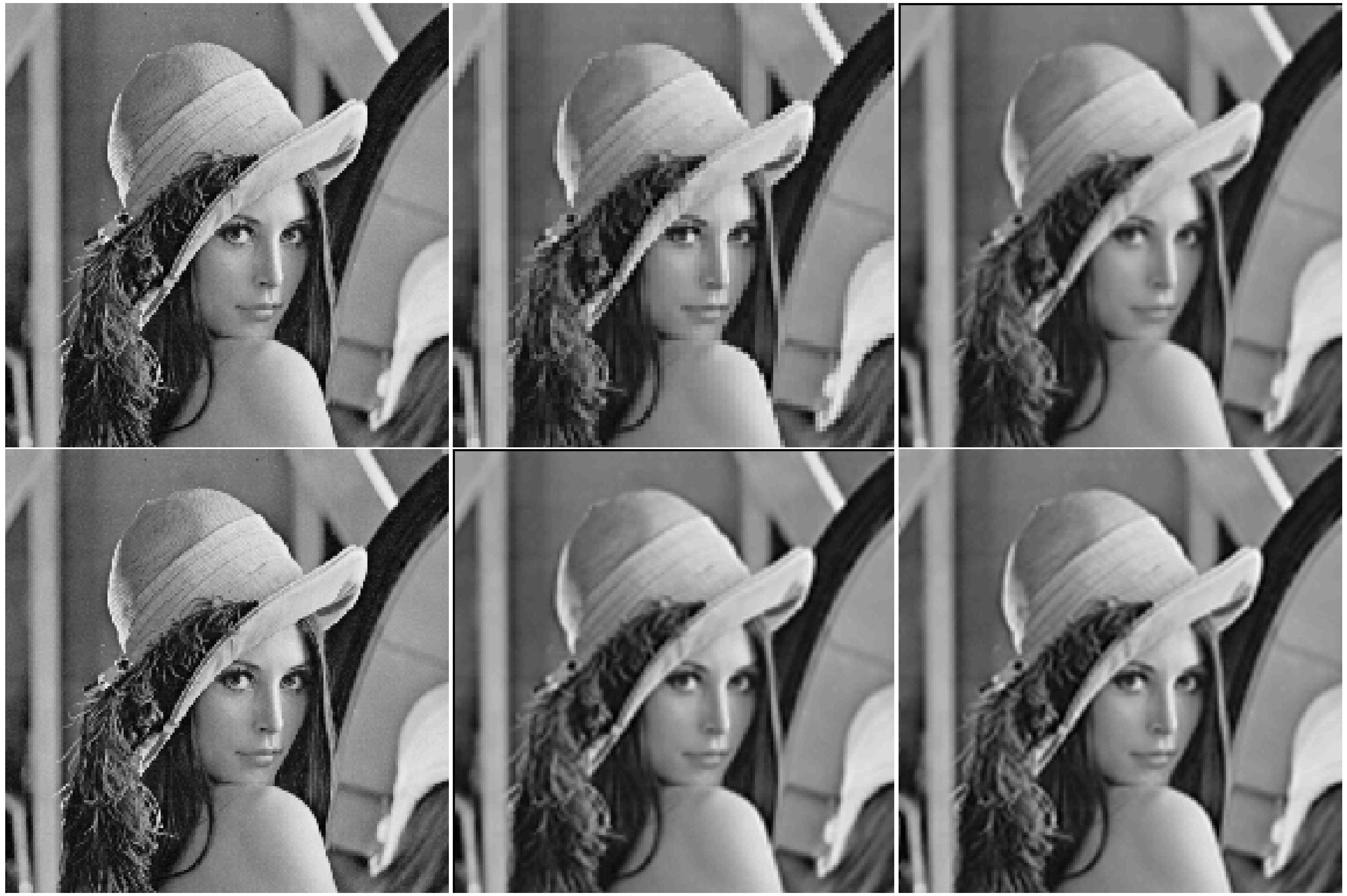
$$\beta^{(0)}(x) = \begin{cases} 1 & 1/2 \leq x < 1/2 \\ 0 & \text{otherwise} \end{cases}$$

i.e. rect

- Fourier transform is $\operatorname{sinc}(u)^{n+1}$

- Cubic b-splines are popular
- Cubic spline vs. cubic interpolation (1D)
  - Cubic interpolation: Use 4 points to fit one cubic polynomial (no constraint on derivative)
  - Cubic spline interpolation: Enforces smoothness constraint
  - i.e. For 4 points, we calculate a different cubic polynomial per segment and enforce smoothness at the endpoints

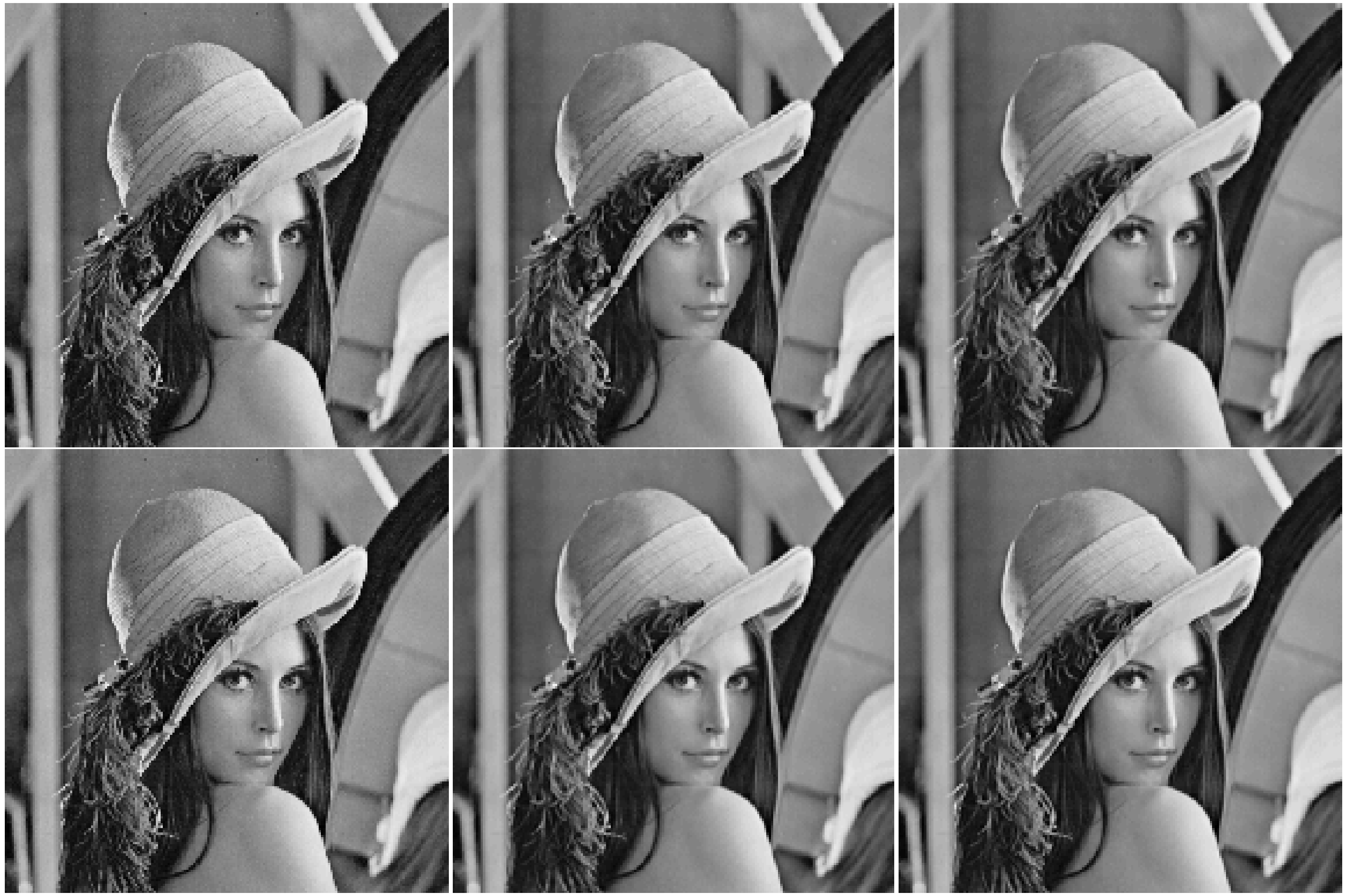
Comparison of Different Methods



Top: Original, Nearest Neighbor, Bilinear
Bottom: Original, Bicubic, Spline

Downsample & resize

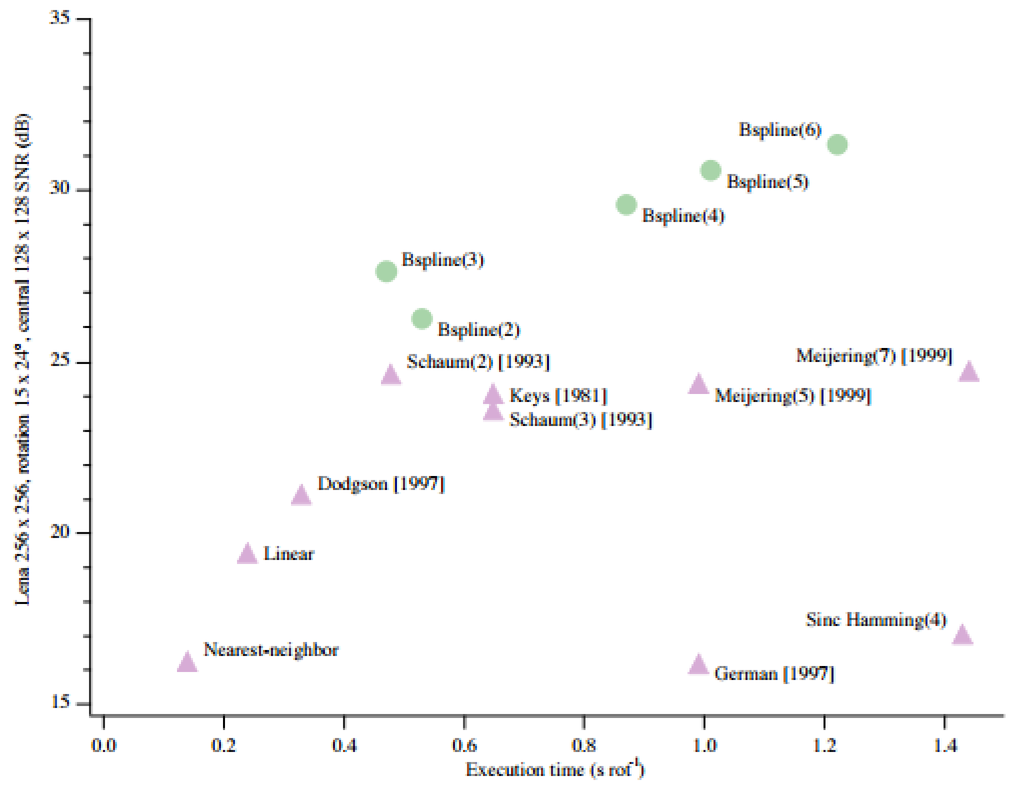
Comparison of Different Methods



Top: Original, Nearest Neighbor, Bilinear
Bottom: Original, Bicubic, Spline

Downsample less & resize

Comparison of Different Methods

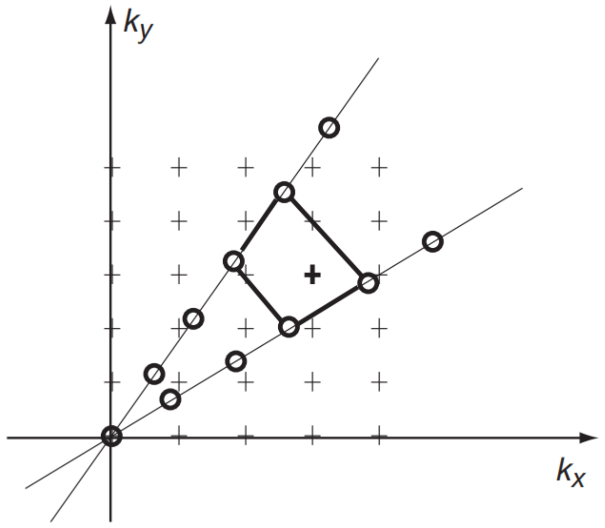


Thévenaz et al., *Handbook of Medical Image Processing*, 2000

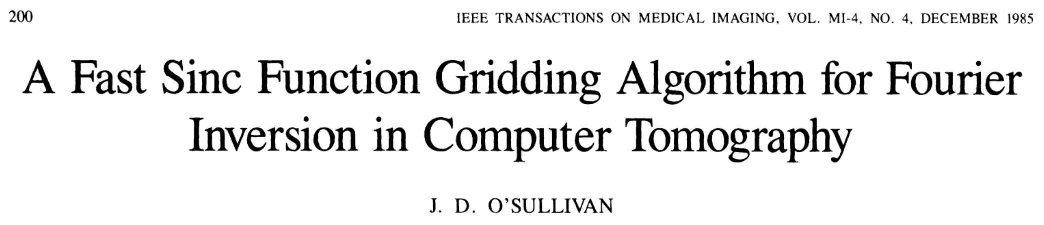
Interpolation applications

- Routinely used for display purposes
  - For instance, data is acquired at one resolution (e.g. 1.5mm) but displayed at a higher resolution (e.g. 0.75mm) → Uses bilinear interpolation in practice

- Also used in taking “off-grid” points to a Cartesian grid
  - Sometimes samples lie on a non-Cartesian grid (non-uniform samples)
  - May be easier to put them on a Cartesian grid & run FFT than to try to do a non-uniform DFT (called “gridding”)



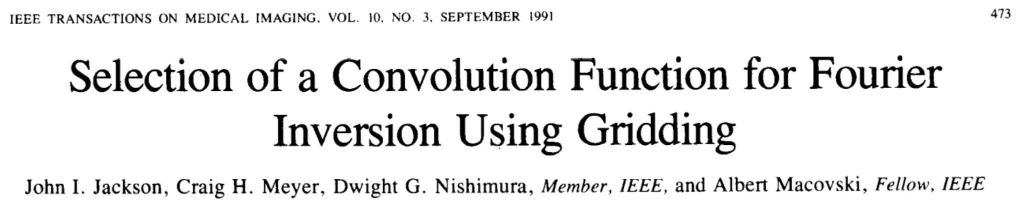
Interpolation applications



200 IEEE TRANSACTIONS ON MEDICAL IMAGING, VOL. MI-4, NO. 4, DECEMBER 1985

A Fast Sinc Function Gridding Algorithm for Fourier Inversion in Computer Tomography

J. D. O'SULLIVAN



IEEE TRANSACTIONS ON MEDICAL IMAGING, VOL. 10, NO. 3, SEPTEMBER 1991 473

Selection of a Convolution Function for Fourier Inversion Using Gridding

John I. Jackson, Craig H. Meyer, Dwight G. Nishimura, *Member, IEEE*, and Albert Macovski, *Fellow, IEEE*

- First paper shows:
  - Sinc is ideal (as we saw), but infinite length
  - Best finite extent can be chosen based on some optimality criterion, which is to maximize the energy in the main lobe of the window relative to the total energy (leads to prolate spheroidal wavefunction), but is hard to compute
  - Kaiser-Bessel window (with appropriate hyperparameter) gives a good approximation
  - Still used in non-Cartesian MRI

Recap

- Histogram processing
- Interpolation methods

- Next class: Start on statistical image processing