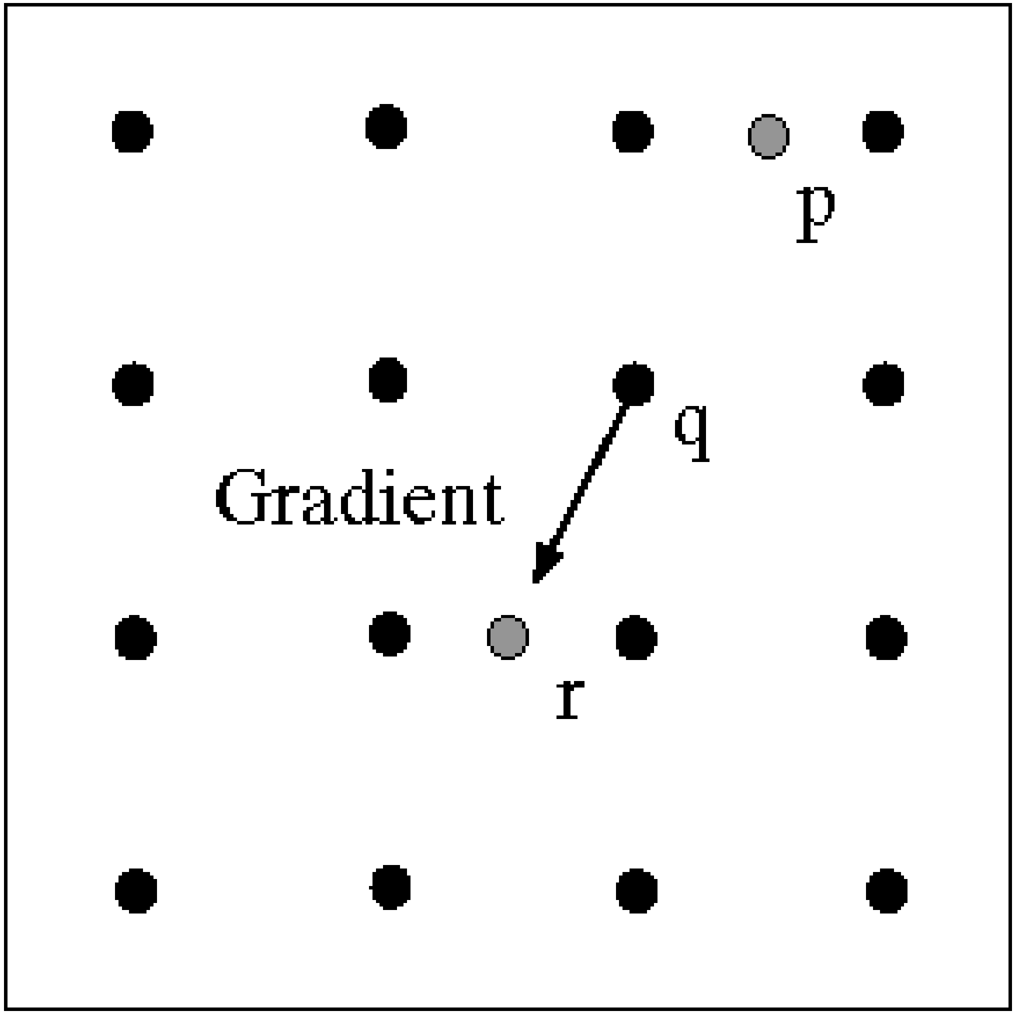
Upsampling and Interpolation

Nearest-neighbor interpolation

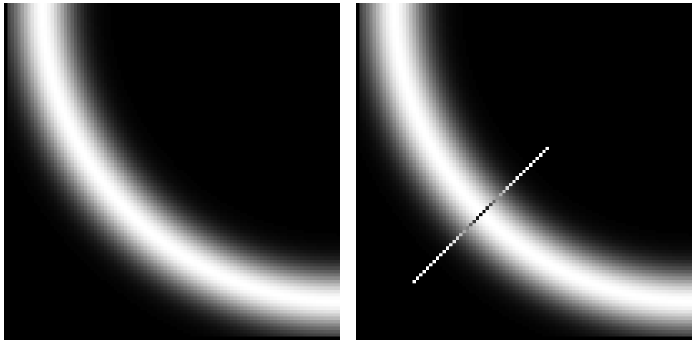
hq3x interpolation (ZSNES) http://en.wikipedia.org/wiki/Hqx

Many slides from
Steve Marschner, Alexei Efros, Noah Snavely

Last time: Non-Maximum Suppression



At q, we have a maximum if the value is larger than those at both p and at r. Interpolate to get these values.



Source: D. Forsyth

Interpolation

- See blackboard

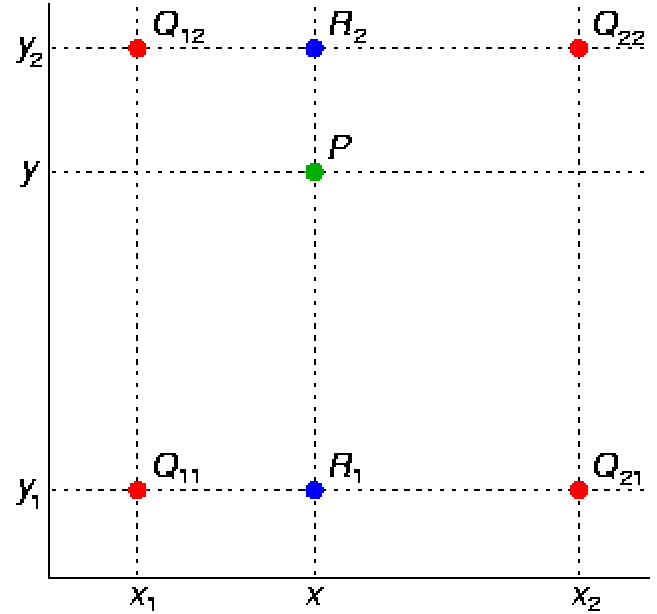
Bilinear Interpolation: Summary

$$f(R_1) \approx \frac{x_2 - x}{x_2 - x_1} f(Q_{11}) + \frac{x - x_1}{x_2 - x_1} f(Q_{21})$$

$$f(R_2) \approx \frac{x_2 - x}{x_2 - x_1} f(Q_{12}) + \frac{x - x_1}{x_2 - x_1} f(Q_{22})$$

$$f(P) \approx \frac{y_2 - y}{y_2 - y_1} f(R_1) + \frac{y - y_1}{y_2 - y_1} f(R_2).$$

$$\begin{aligned} f(x,y) \approx\ & \frac{f(Q_{11})}{(x_2-x_1)(y_2-y_1)}(x_2-x)(y_2-y) + \\ & \frac{f(Q_{21})}{(x_2-x_1)(y_2-y_1)}(x-x_1)(y_2-y) + \\ & \frac{f(Q_{12})}{(x_2-x_1)(y_2-y_1)}(x_2-x)(y-y_1) + \\ & \frac{f(Q_{22})}{(x_2-x_1)(y_2-y_1)}(x-x_1)(y-y_1) \\ =\ & \frac{1}{(x_2-x_1)(y_2-y_1)}\Big(f(Q_{11})(x_2-x)(y_2-y) + \\ & \quad f(Q_{21})(x-x_1)(y_2-y) + \\ & \quad f(Q_{12})(x_2-x)(y-y_1) + \\ & \quad f(Q_{22})(x-x_1)(y-y_1) \Big) \end{aligned}$$



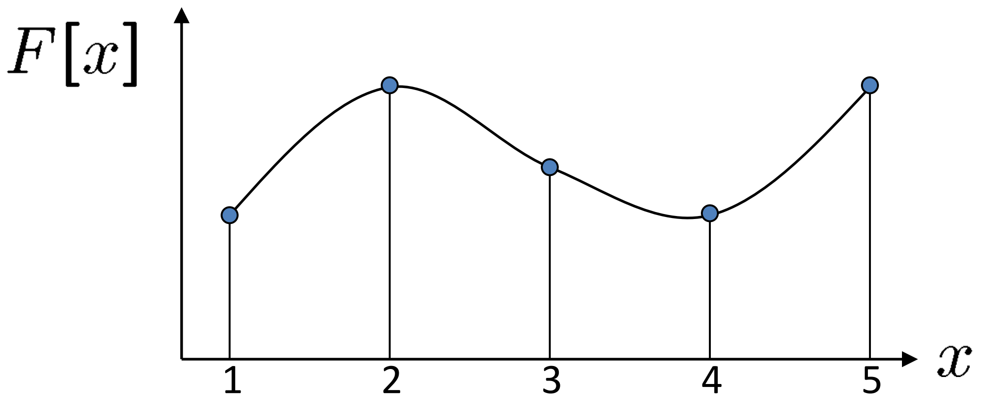
Bilinear Interpolation

- Not actually linear
  - If you fix *x* it's linear in *y.* If you fix *y*, it's linear in *x*.

Upsampling

- This image is too small for this screen: 
- How can we make it 10 times as big?
- Simplest approach:
  repeat each row
  and column 10 times
- ("Nearest neighbor
  interpolation")

Image interpolation



d = 1 in this example

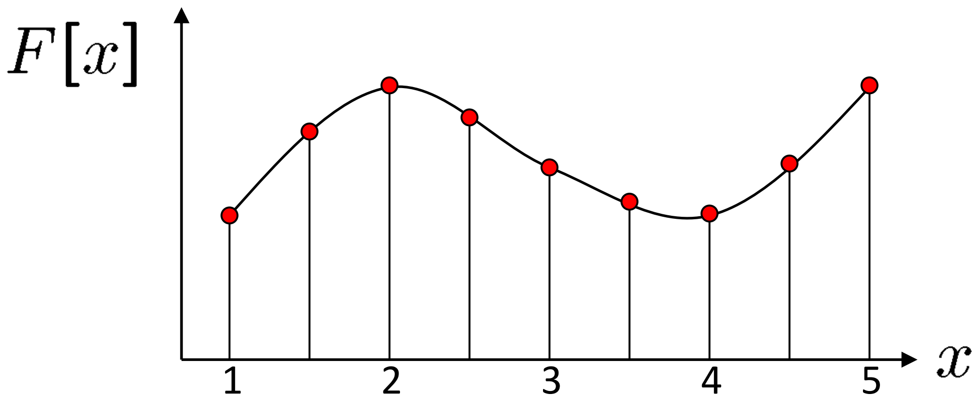
Recall how a digital image is formed

$$F[x, y] = \text{quantize}\{f(xd, yd)\}$$

- It is a discrete point-sampling of a continuous function
- If we could somehow reconstruct the original function, any new image could be generated, at any resolution and scale

Adapted from: S. Seitz

Image interpolation



d = 1 in this example

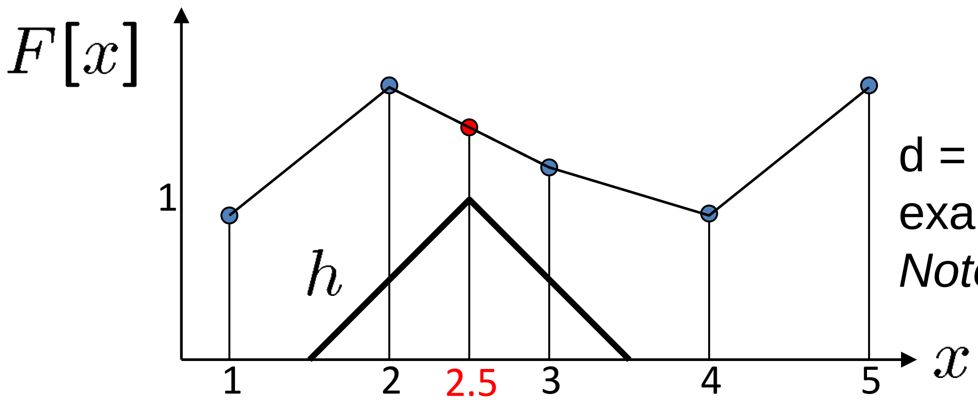
Recall how a digital image is formed

$$F[x, y] = \text{quantize}\{f(xd, yd)\}$$

- It is a discrete point-sampling of a continuous function
- If we could somehow reconstruct the original function, any new image could be generated, at any resolution and scale

Adapted from: S. Seitz

Image interpolation



d = 1 in this example.
Note: this is linear interpolation!

- What if we don't know f ?
  - Guess an approximation: $\tilde{f}$
  - Can be done in a principled way: filtering
  - Convert F to a continuous function:

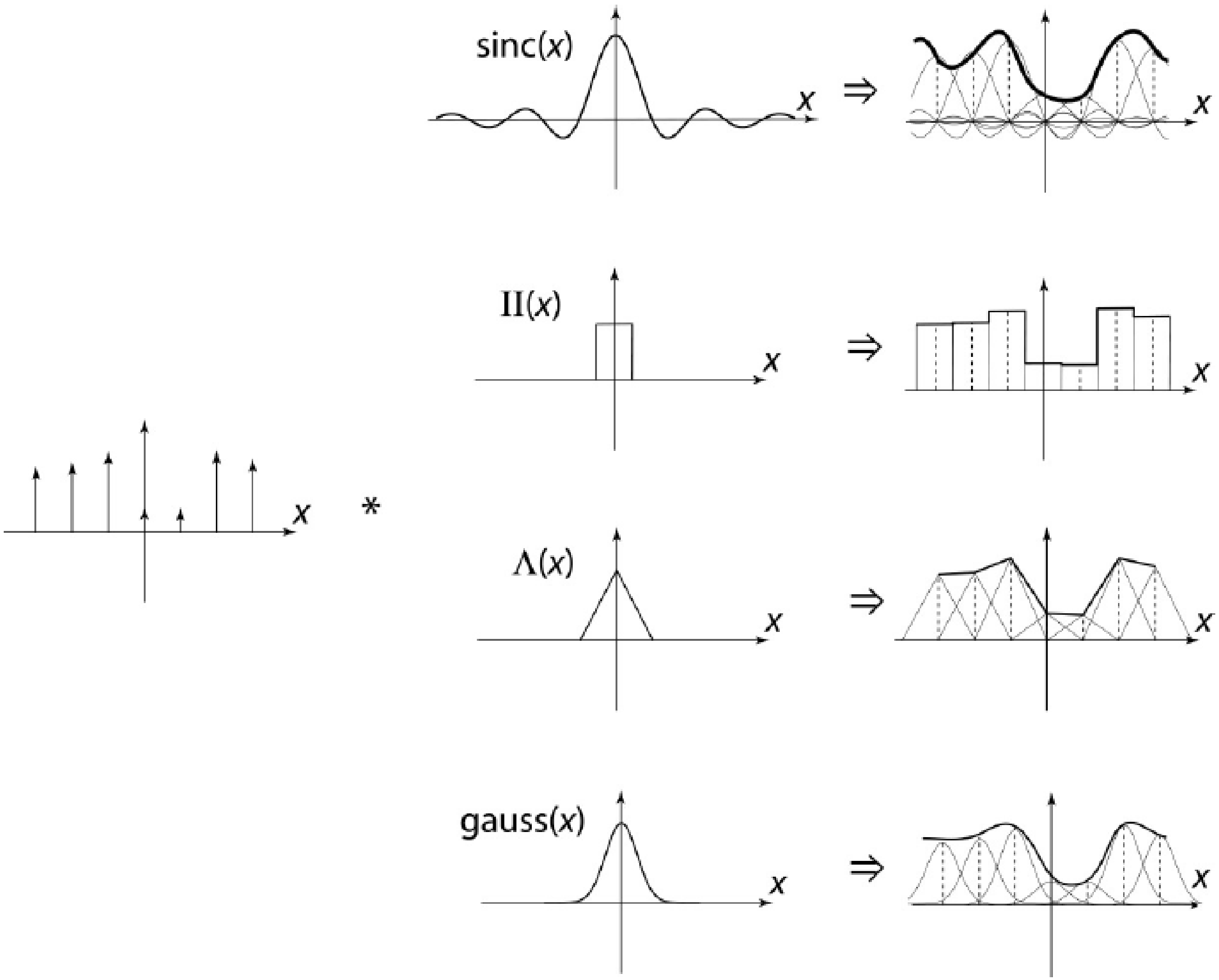
$$f_F(x) = F(\tfrac{x}{d}) \text{ when } \tfrac{x}{d} \text{ is an integer, 0 otherwise}$$

  - Reconstruct by convolution with a *reconstruction filter, h*

$$\tilde{f} = h * f_F$$

Adapted from: S. Seitz

Image interpolation



"Ideal" reconstruction

Nearest-neighbor interpolation

Linear interpolation

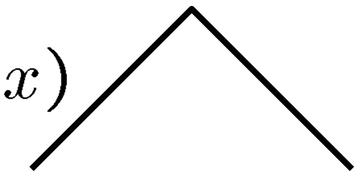
Gaussian reconstruction

Source: B. Curless

Reconstruction filters

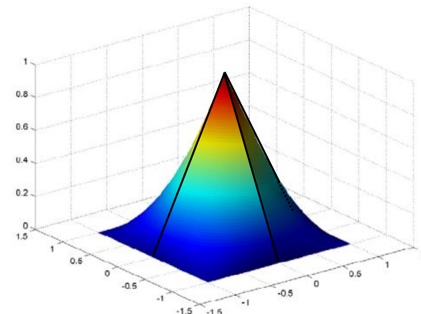
- What does the 2D version of this hat function look like?

$h(x)$

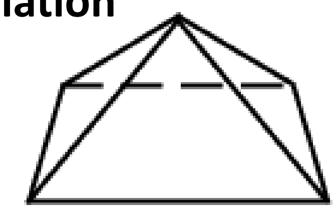


performs
linear interpolation

$h(x, y)$

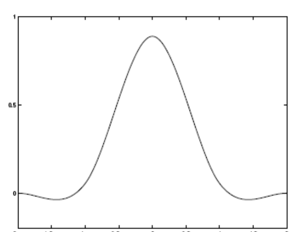


(tent function) performs
bilinear interpolation



Better filters give better resampled images

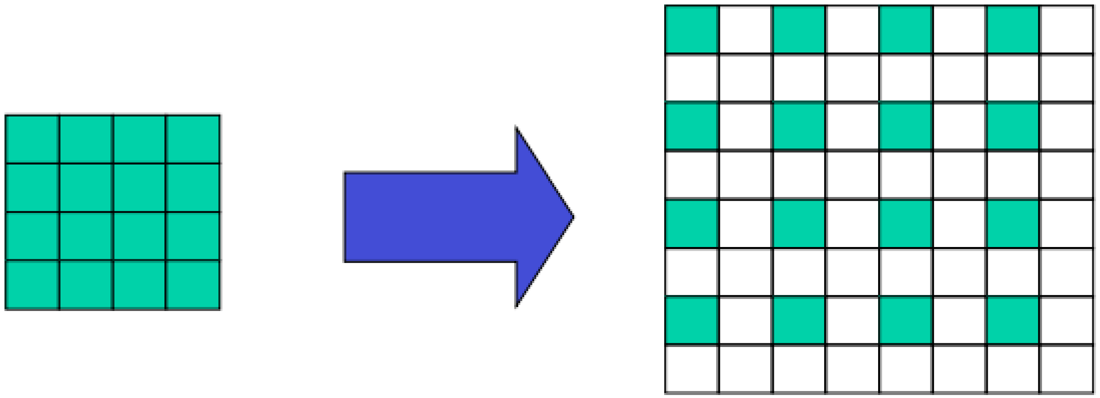
- **Bicubic** is common choice



$$r(x) = \frac{1}{6}\begin{cases} (12-9B-6C)|x|^3 + (-18+12B+6C)|x|^2 + (6-2B) & |x| < 1 \\ ((-B-6C)|x|^3 + (6B+30C)|x|^2 + (-12B-48C)|x| + (8B+24C) & 1 \leq |x| < 2 \\ 0 & \textit{otherwise} \end{cases}$$

Cubic reconstruction filter

Upsampling



- The empty pixels are initially set to 0
- Convolve with a (Gaussian, or another) filter
- If the filter sums to 1, multiply the result by 4
  - ¾ of the new image was initially 0

Image interpolation

Original image:  x 10

Nearest-neighbor interpolation

Bilinear interpolation

Bicubic interpolation

Image interpolation

Also used for *resampling*