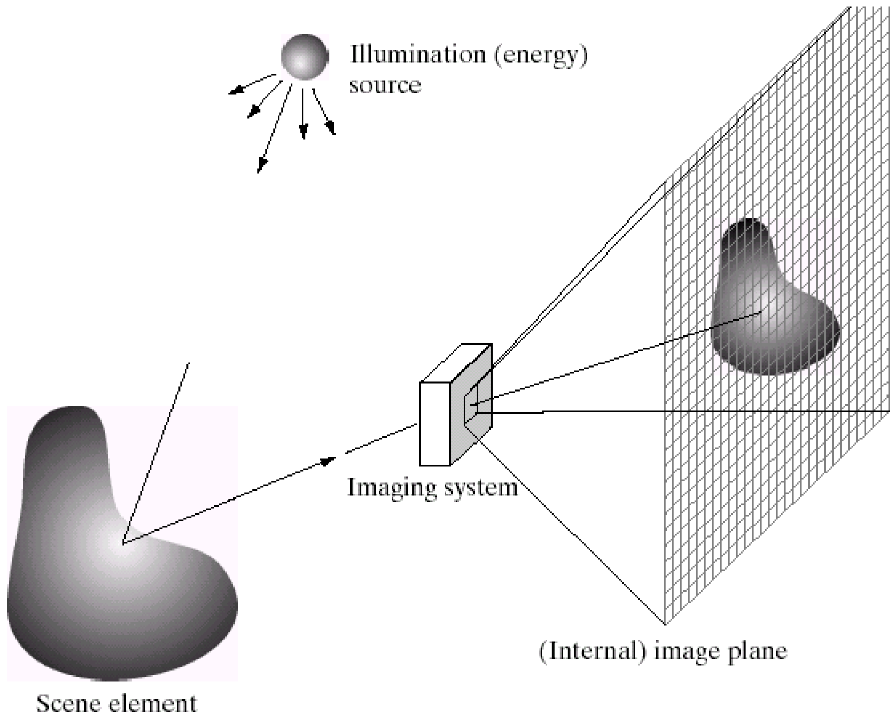
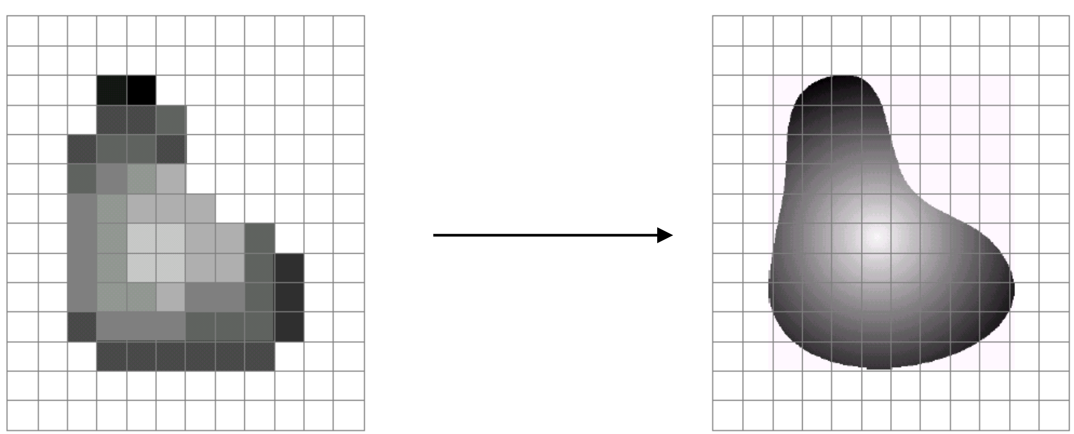
Sampling and Reconstruction

15-463: Computational Photography
Alexei Efros, CMU, Fall 2010

Many slides from
Steve Marschner

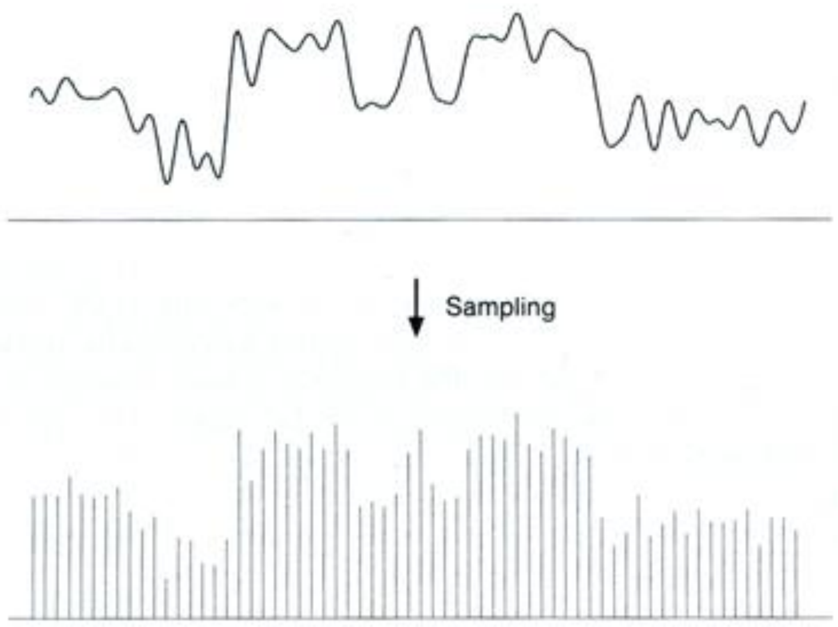
Sampling and Reconstruction





Sampled representations

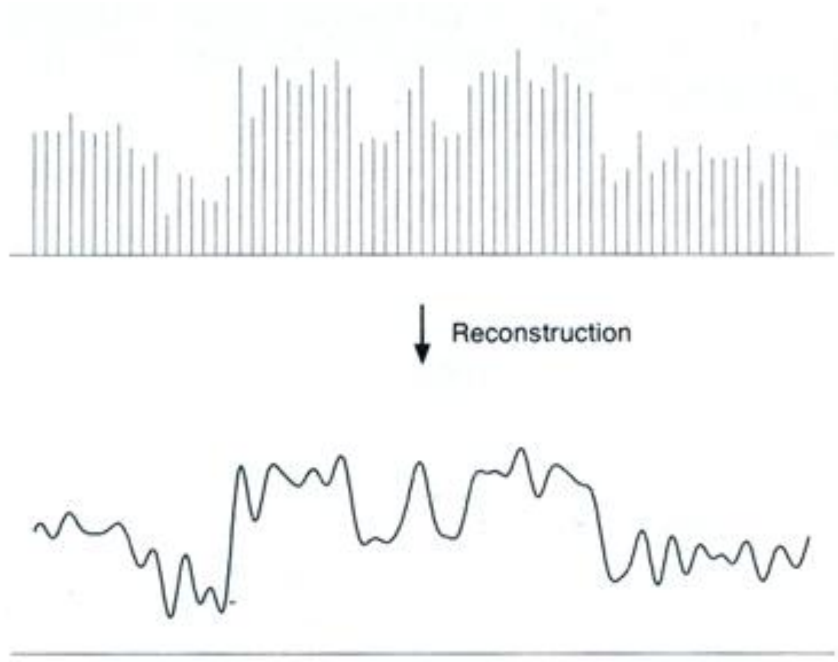
- How to store and compute with continuous functions?
- Common scheme for representation: samples
  - write down the function's values at many points



[FvDFH fig.14.14b / Wolberg]

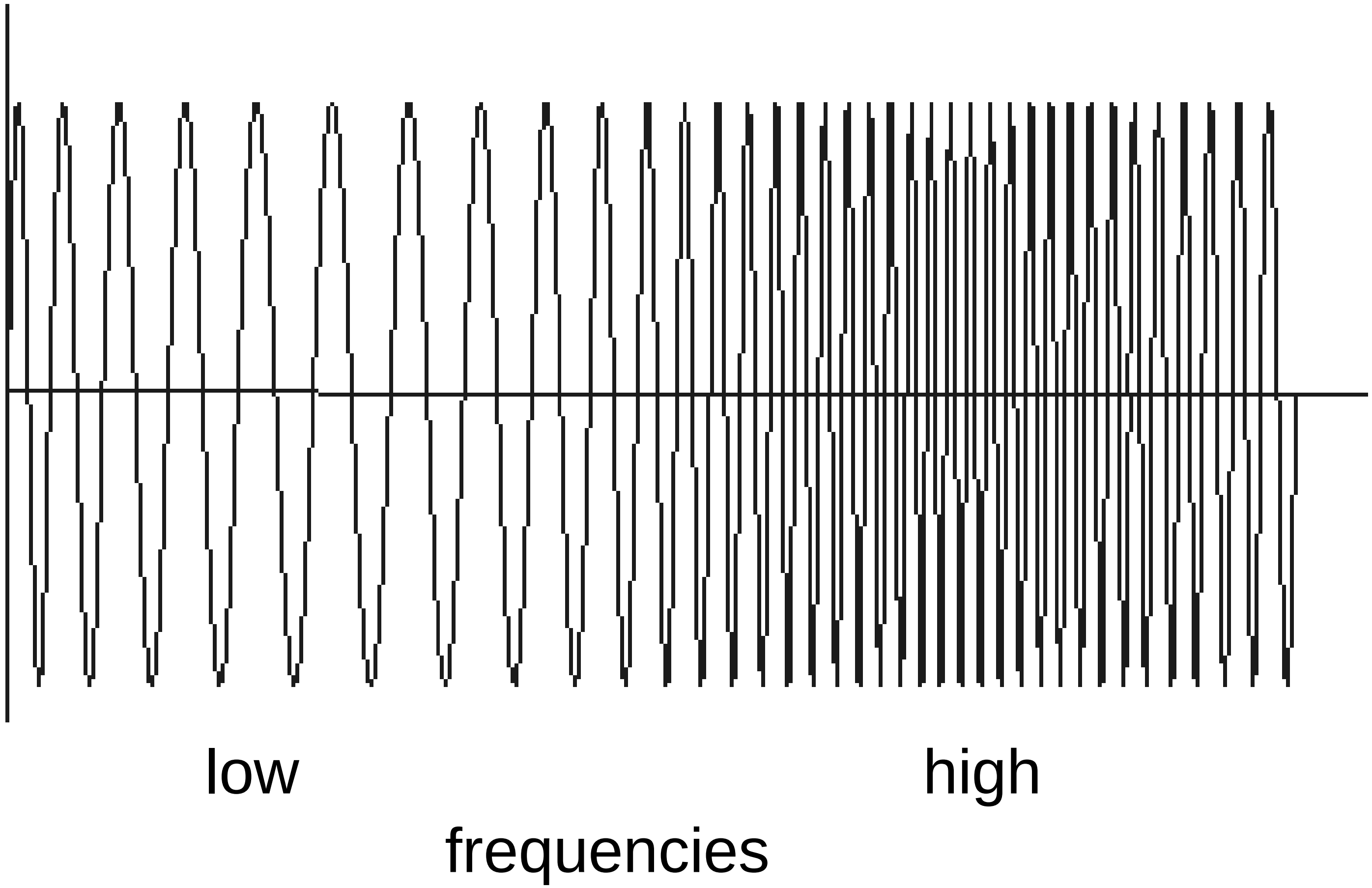
Reconstruction

- Making samples back into a continuous function
  - for output (need realizable method)
  - for analysis or processing (need mathematical method)
  - amounts to “guessing” what the function did in between



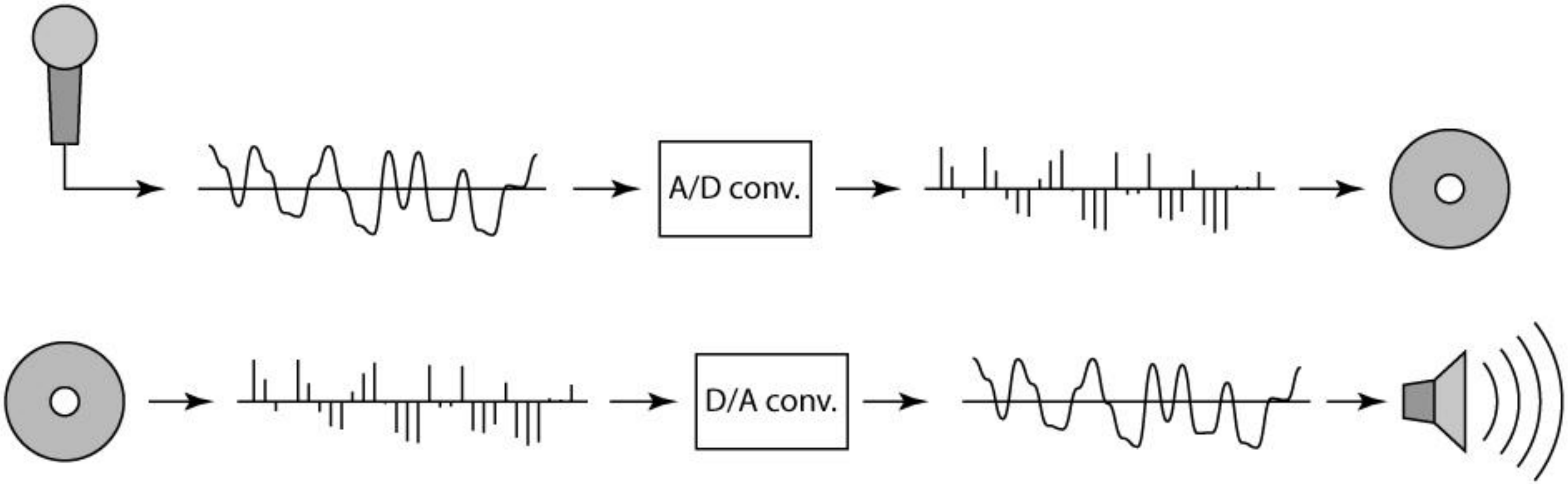
[FvDFH fig.14.14b / Wolberg]

1D Example: Audio



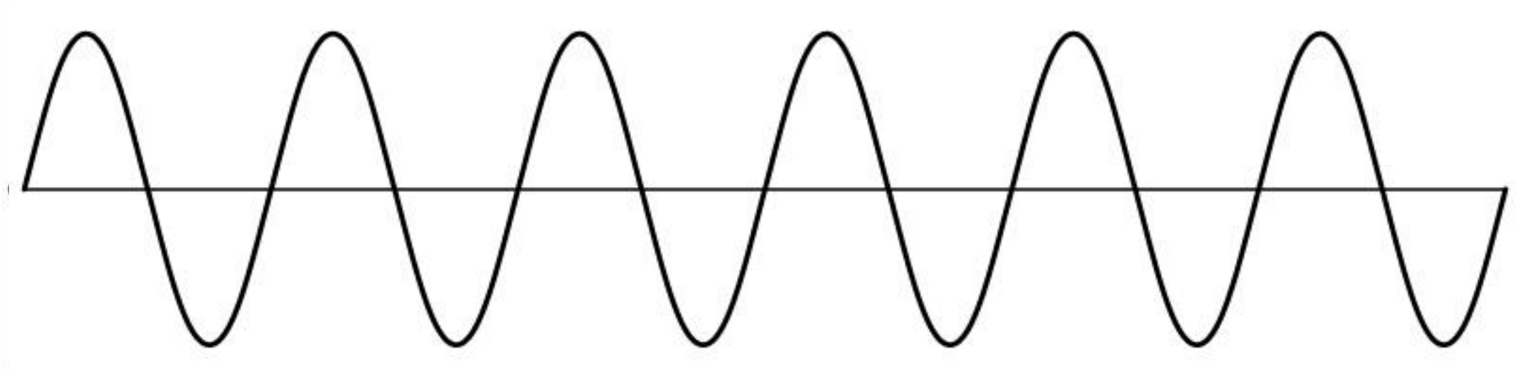
Sampling in digital audio

- Recording: sound to analog to samples to disc
- Playback: disc to samples to analog to sound again
  - how can we be sure we are filling in the gaps correctly?



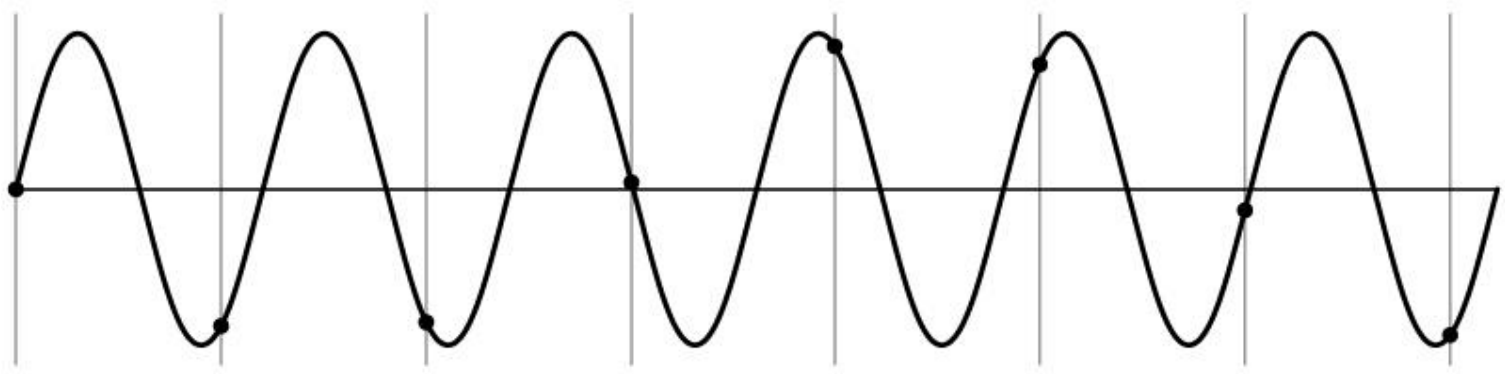
Sampling and Reconstruction

- Simple example: a sign wave



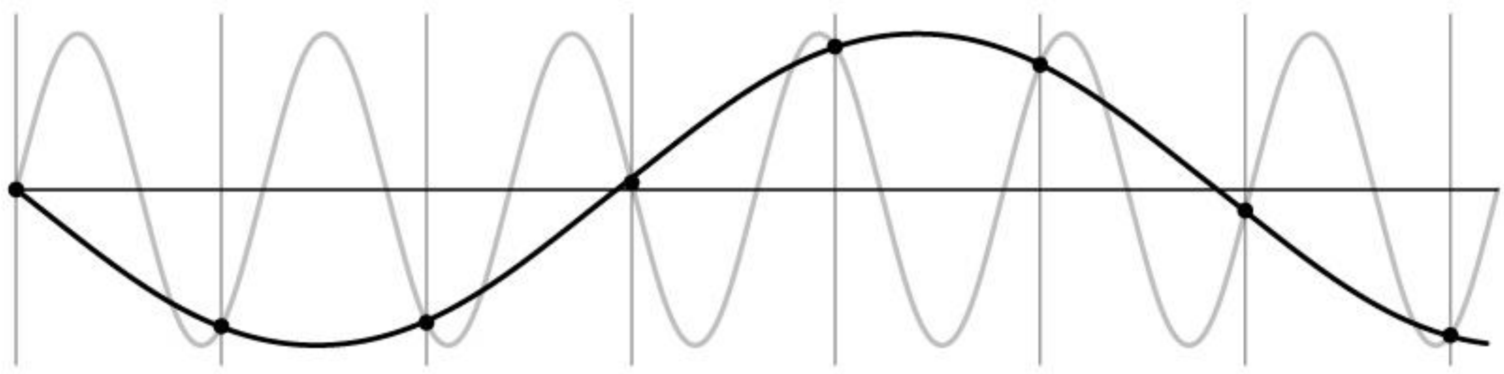
Undersampling

- What if we “missed” things between the samples?
- Simple example: undersampling a sine wave
  - unsurprising result: information is lost



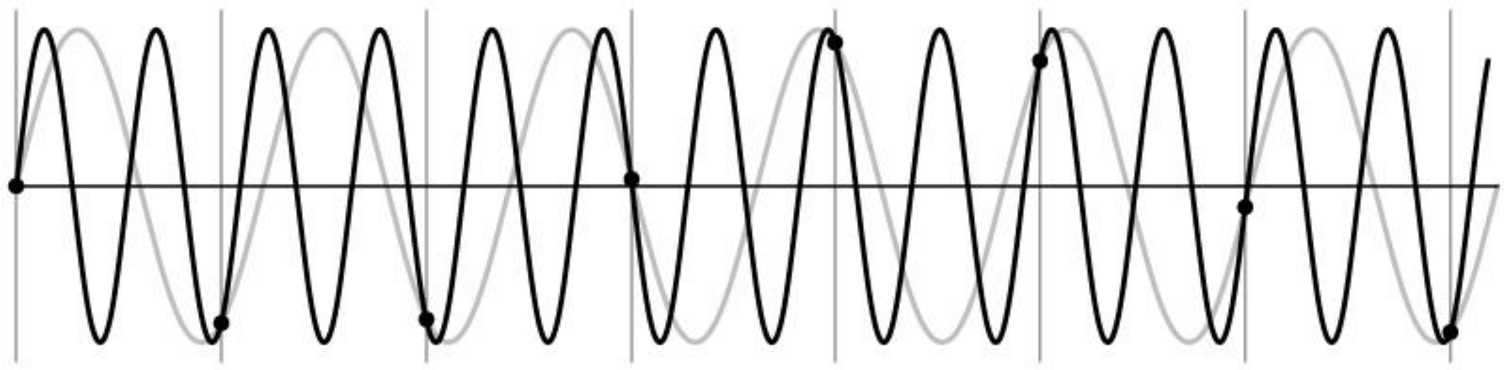
Undersampling

- What if we "missed" things between the samples?
- Simple example: undersampling a sine wave
  - unsurprising result: information is lost
  - surprising result: indistinguishable from lower frequency



Undersampling

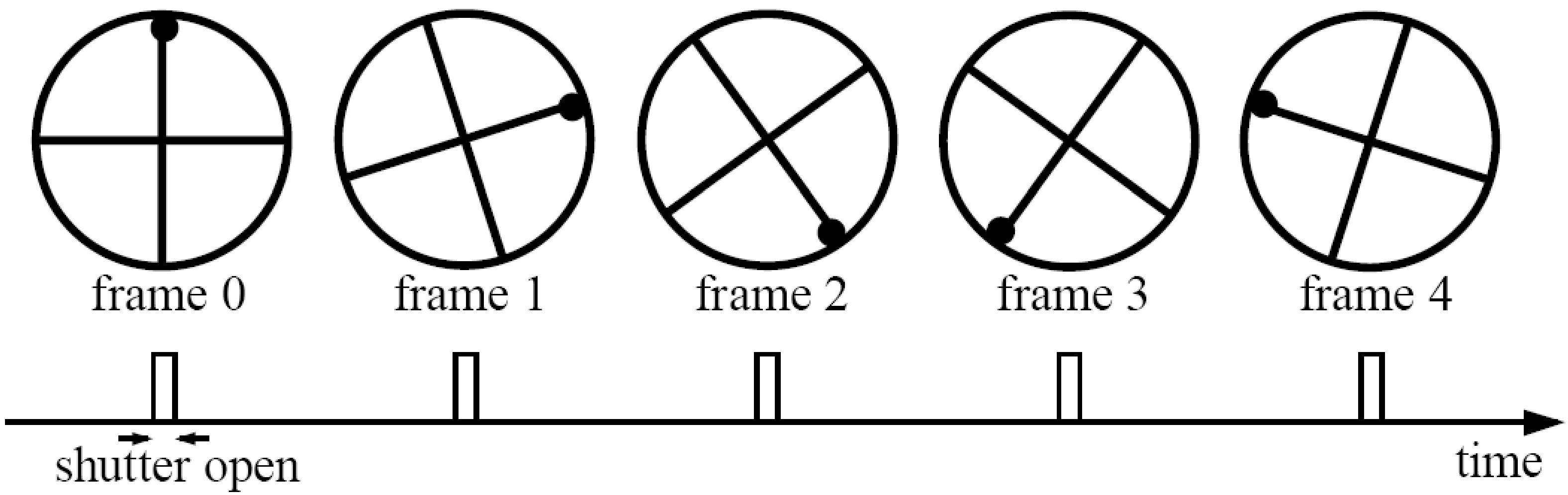
- What if we "missed" things between the samples?
- Simple example: undersampling a sine wave
  - unsurprising result: information is lost
  - surprising result: indistinguishable from lower frequency
  - also was always indistinguishable from higher frequencies
  - *aliasing*: signals "traveling in disguise" as other frequencies



Aliasing in video

Imagine a spoked wheel moving to the right (rotating clockwise). Mark wheel with dot so we can see what's happening.

If camera shutter is only open for a fraction of a frame time (frame time = 1/30 sec. for video, 1/24 sec. for film):

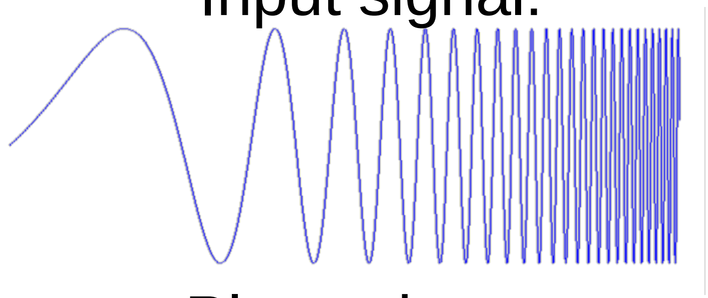


Without dot, wheel appears to be rotating slowly backwards! (counterclockwise)

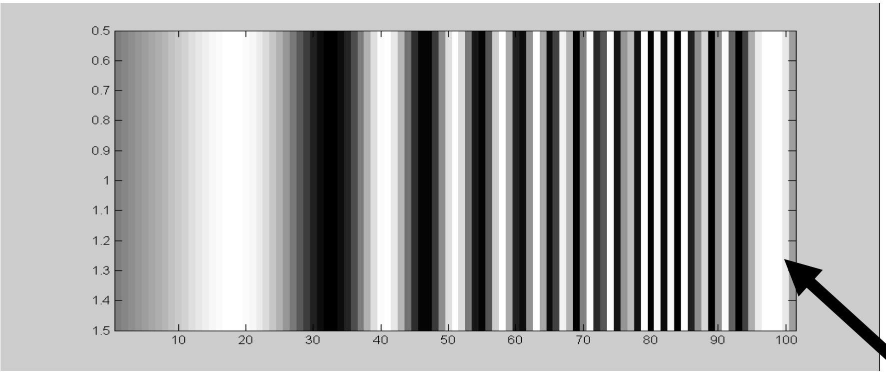
Aliasing in images

What’s happening?

Input signal:

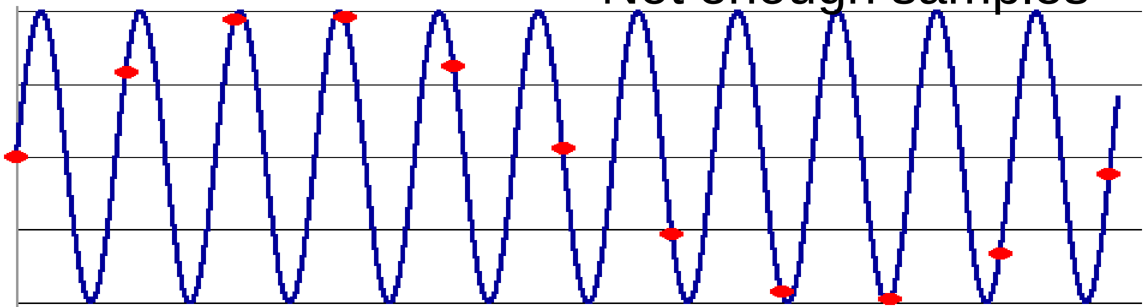


Plot as image:



x = 0:.05:5; imagesc(sin((2.^x).*x))

Alias!
Not enough samples



Antialiasing

What can we do about aliasing?

Sample more often

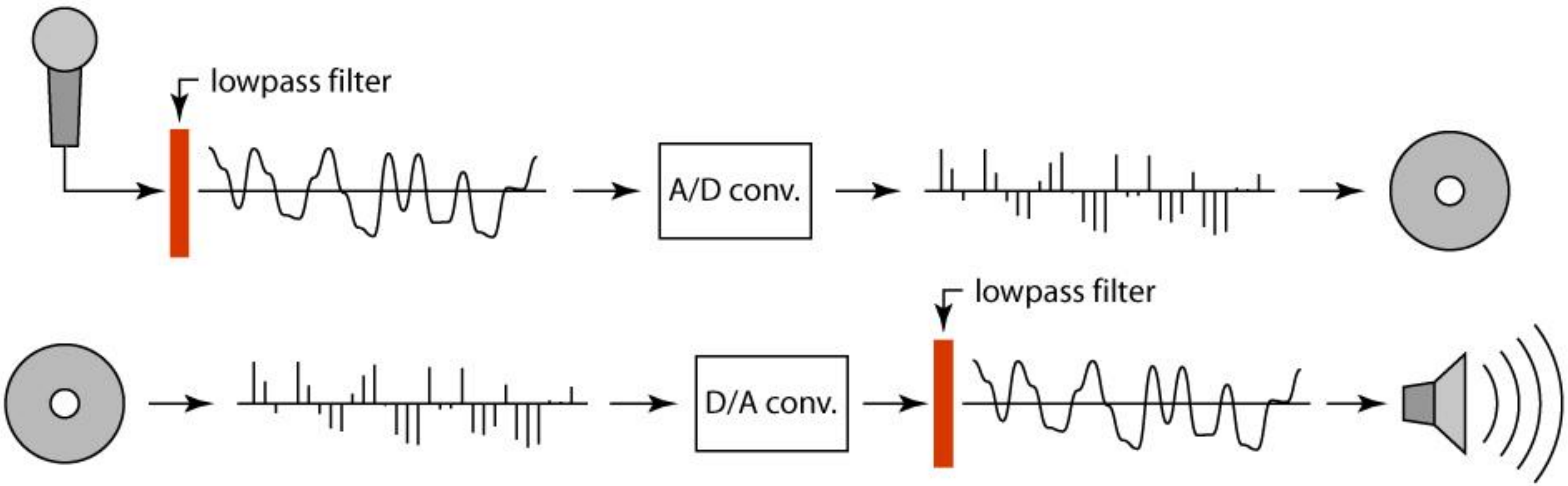
- Join the Mega-Pixel craze of the photo industry
- But this can't go on forever

Make the signal less "wiggly"

- Get rid of some high frequencies
- Will loose information
- But it's better than aliasing

Preventing aliasing

- Introduce lowpass filters:
  - remove high frequencies leaving only safe, low frequencies
  - choose lowest frequency in reconstruction (disambiguate)

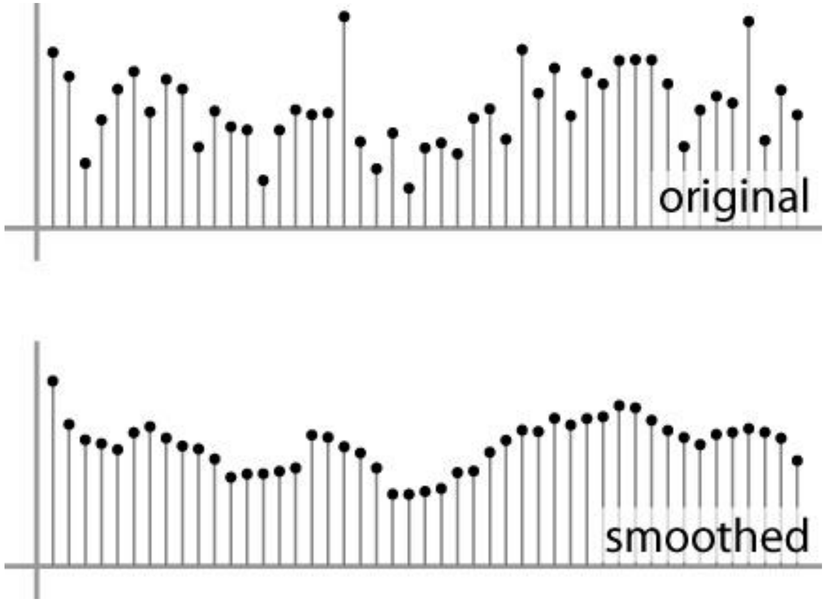


Linear filtering: a key idea

- Transformations on signals; e.g.:
  - bass/treble controls on stereo
  - blurring/sharpening operations in image editing
  - smoothing/noise reduction in tracking
- Key properties
  - linearity: filter(f + g) = filter(f) + filter(g)
  - shift invariance: behavior invariant to shifting the input
    - delaying an audio signal
    - sliding an image around
- Can be modeled mathematically by *convolution*

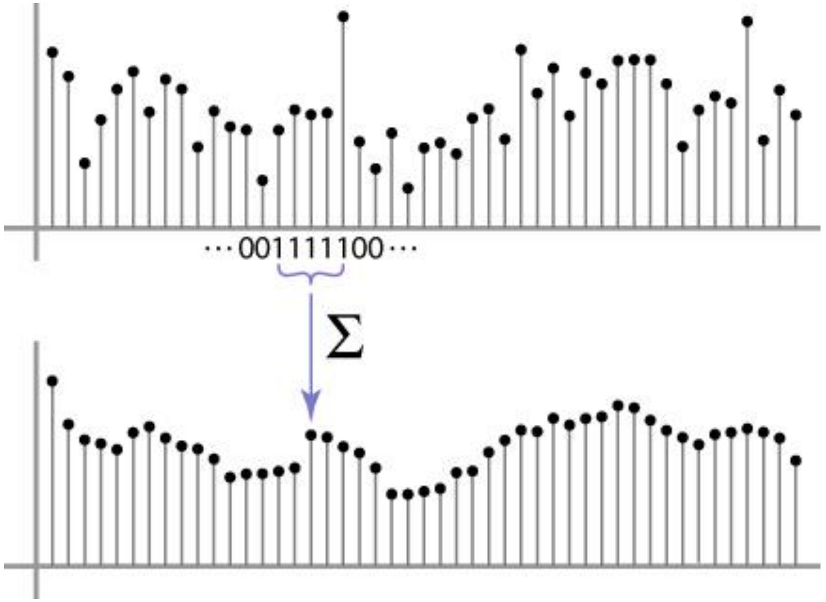
Moving Average

- basic idea: define a new function by averaging over a sliding window
- a simple example to start off: smoothing



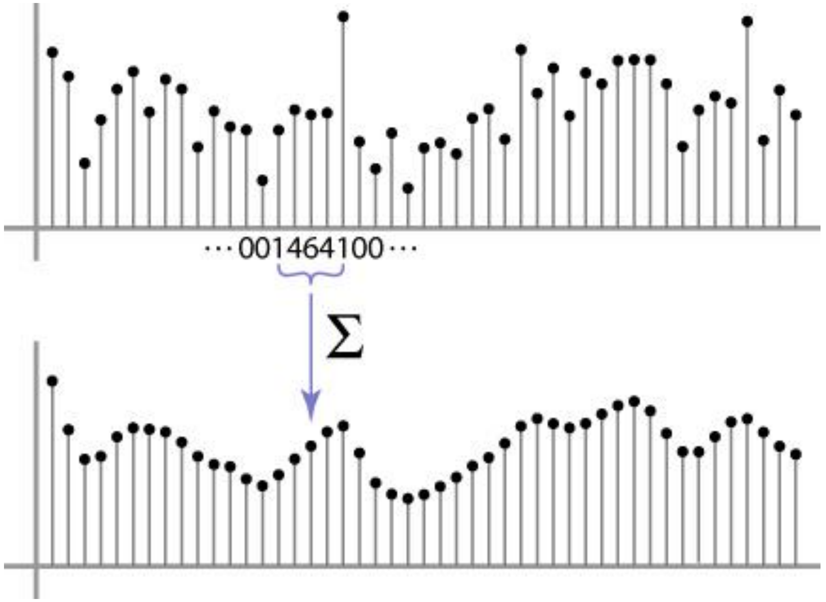
Weighted Moving Average

- Can add weights to our moving average
- *Weights* [..., 0, 1, 1, 1, 1, 1, 0, ...] / 5



Weighted Moving Average

- bell curve (gaussian-like) weights [..., 1, 4, 6, 4, 1, ...]



Moving Average In 2D

What are the weights H?

0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	0	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	0	0	0	0	0	0	0
0	0	90	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0

$F[x, y]$

$H[u, v]$

Slide by Steve Seitz

Cross-correlation filtering

- Let's write this down as an equation. Assume the averaging window is (2k+1)x(2k+1):

$$G[i,j] = \frac{1}{(2k+1)^2} \sum_{u=-k}^{k} \sum_{v=-k}^{k} F[i+u, j+v]$$

  - We can generalize this idea by allowing different weights for different neighboring pixels:

$$G[i,j] = \sum_{u=-k}^{k} \sum_{v=-k}^{k} H[u,v] F[i+u, j+v]$$

  - This is called a **cross-correlation** operation and written:

$$G = H \otimes F$$

  - H is called the "filter," "kernel," or "mask."

Slide by Steve Seitz

Gaussian filtering

A Gaussian kernel gives less weight to pixels further from the center of the window

0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	0	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	0	0	0	0	0	0	0
0	0	90	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0

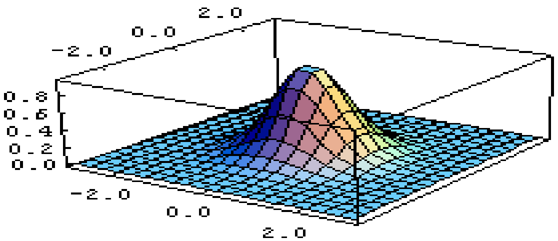
$F[x, y]$

$\frac{1}{16}$

1	2	1
2	4	2
1	2	1

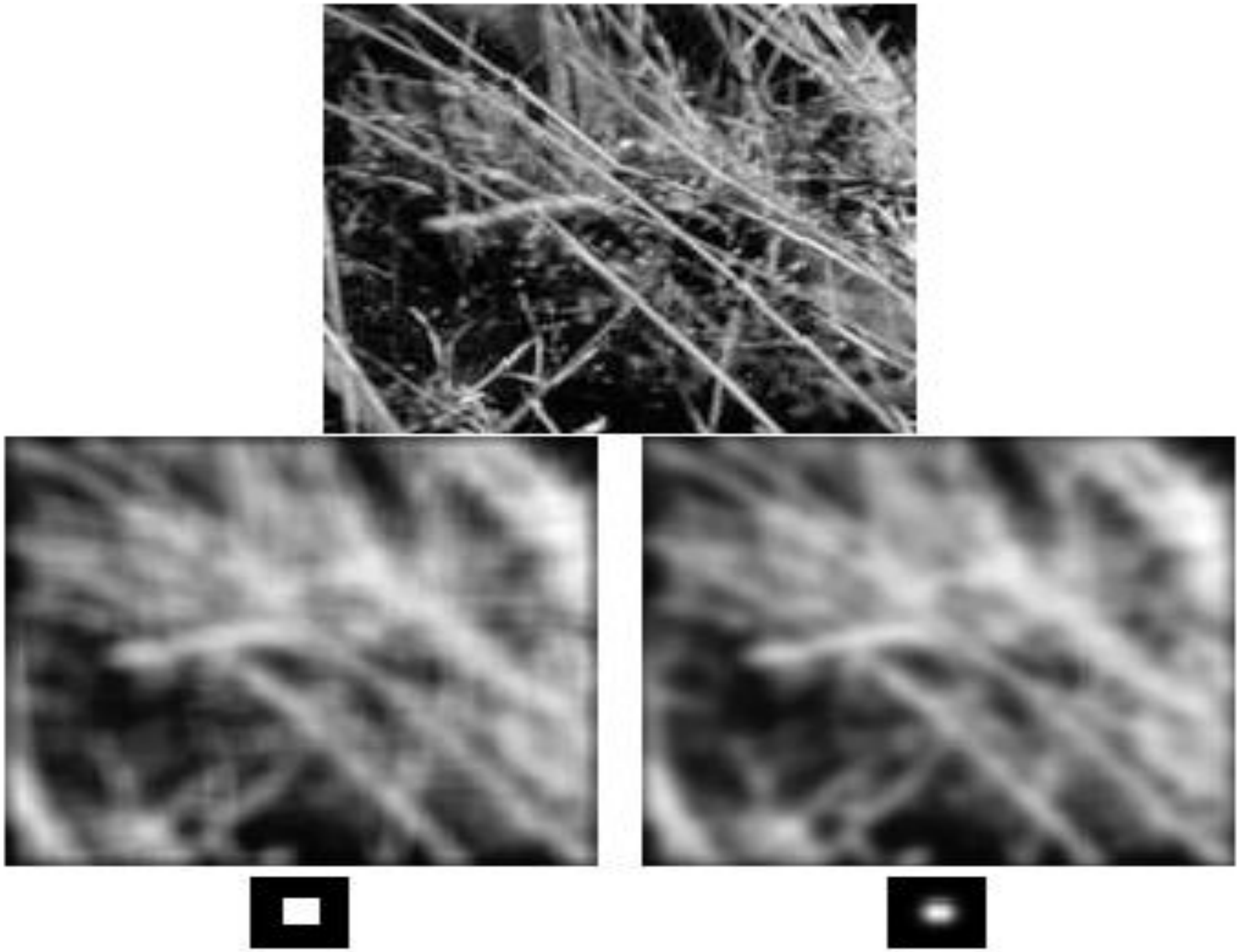
$H[u, v]$

$$h(u, v) = \frac{1}{2\pi\sigma^2} e^{-\frac{u^2+v^2}{\sigma^2}}$$



Slide by Steve Seitz

Mean vs. Gaussian filtering



Slide by Steve Seitz

Convolution

cross-correlation: $G = H \otimes F$

$$G[i,j] = \sum_{u=-k}^{k} \sum_{v=-k}^{k} H[u,v]F[i+u,j+v]$$

A **convolution** operation is a cross-correlation where the filter is flipped both horizontally and vertically before being applied to the image:

$$G[i,j] = \sum_{u=-k}^{k} \sum_{v=-k}^{k} H[u,v]F[i-u,j-v]$$

It is written:

$$G = H \star F$$

Suppose H is a Gaussian or mean kernel. How does convolution differ from cross-correlation?

Slide by Steve Seitz

Convolution is nice!

- Notation: $b = c \star a$
- Convolution is a multiplication-like operation
  - commutative $a \star b = b \star a$
  - associative $a \star (b \star c) = (a \star b) \star c$
  - distributes over addition $a \star (b + c) = a \star b + a \star c$
  - scalars factor out $\alpha a \star b = a \star \alpha b = \alpha(a \star b)$
  - identity: unit impulse *e* = […, 0, 0, 1, 0, 0, …]

    $a \star e = a$
- Conceptually no distinction between filter and signal
- Usefulness of associativity
  - often apply several filters one after another: $(((a * b_1) * b_2) * b_3)$
  - this is equivalent to applying one filter: $a * (b_1 * b_2 * b_3)$

Tricks with convolutions

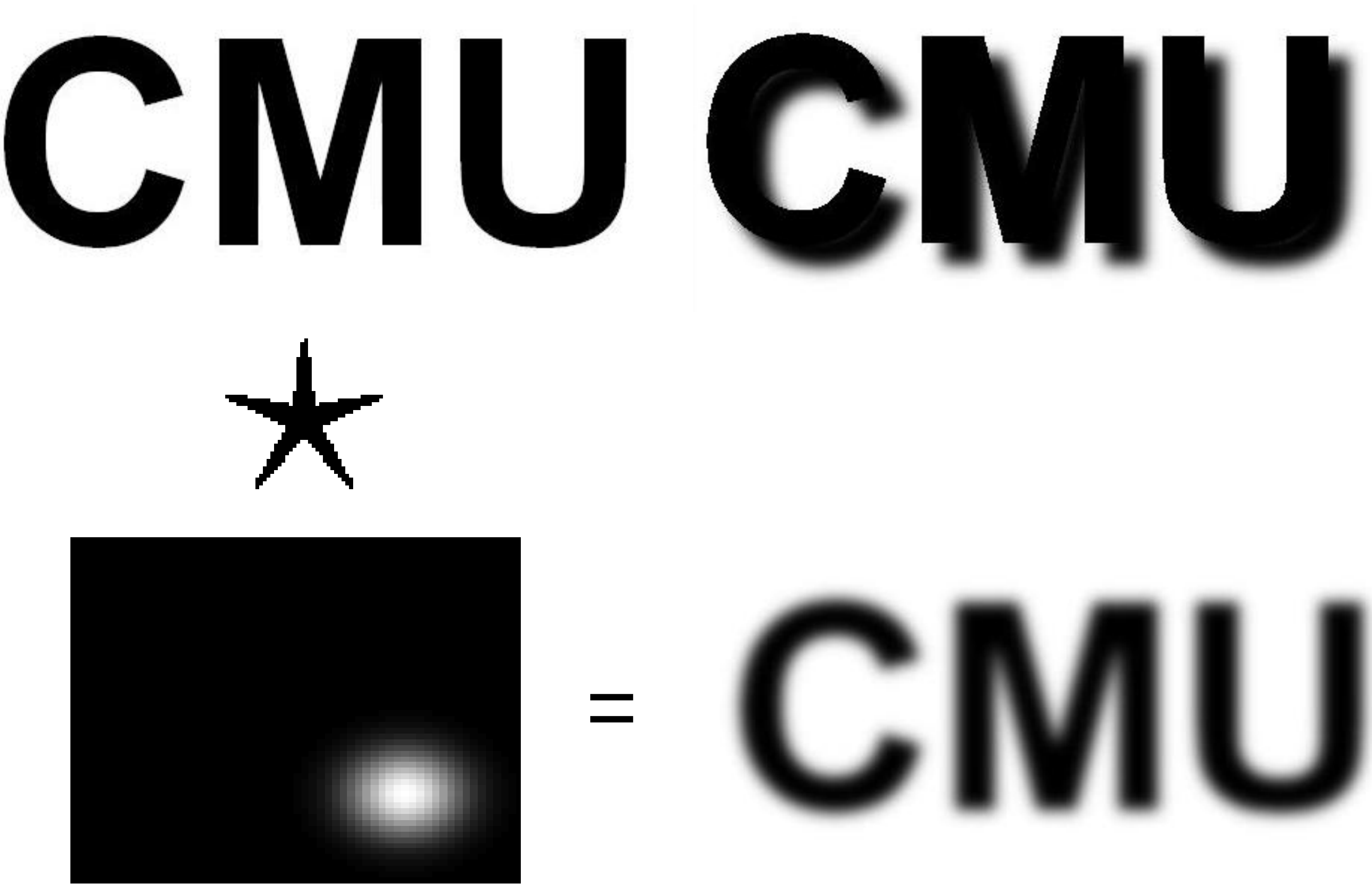


Image half-sizing

This image is too big to fit on the screen. How can we reduce it?

How to generate a half-sized version?

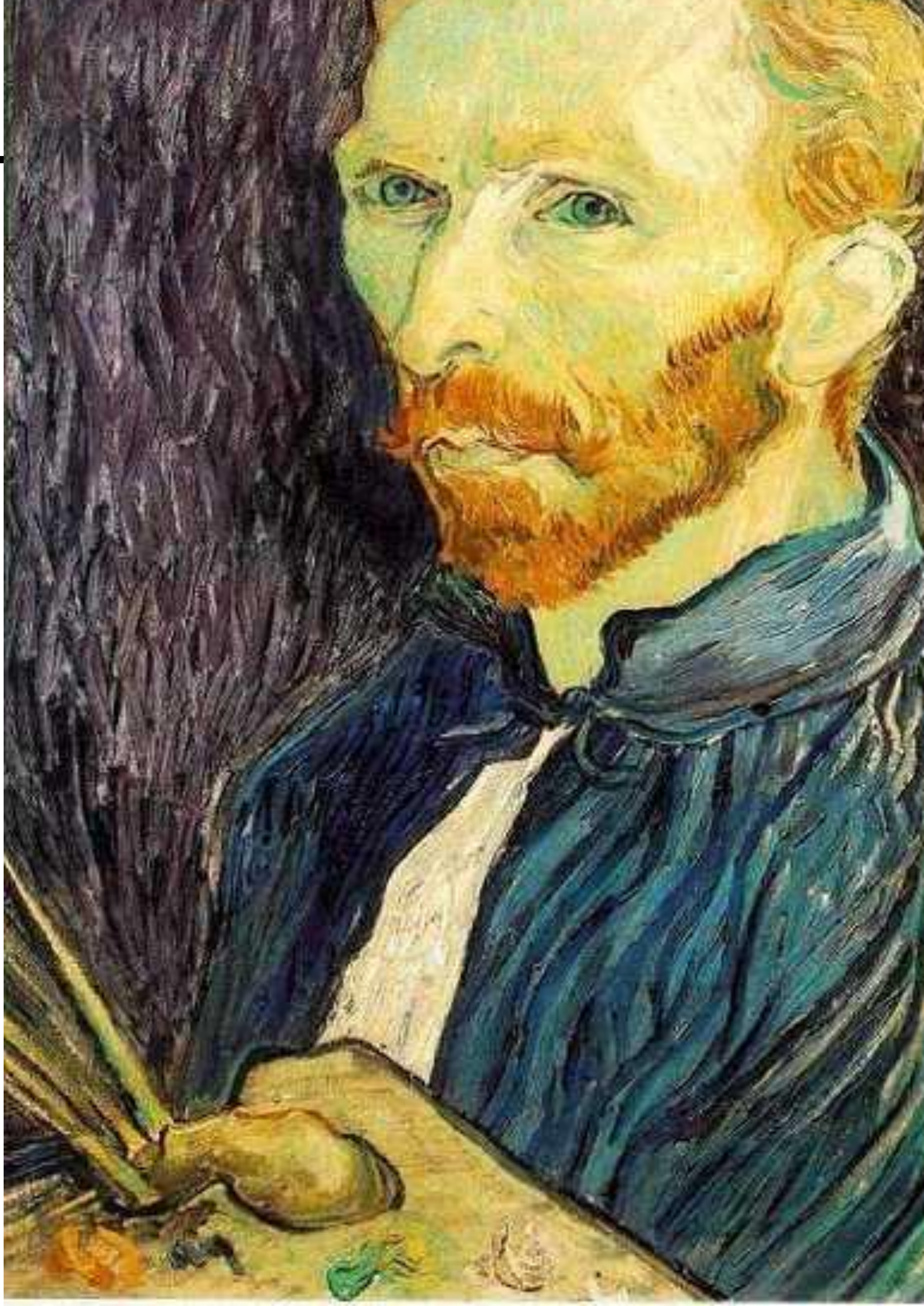
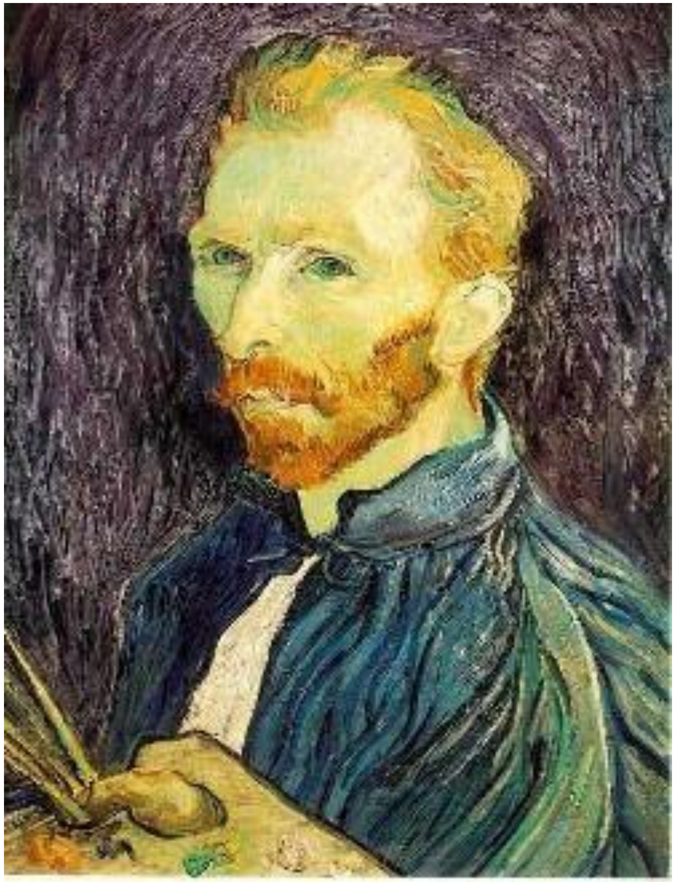


Image sub-sampling



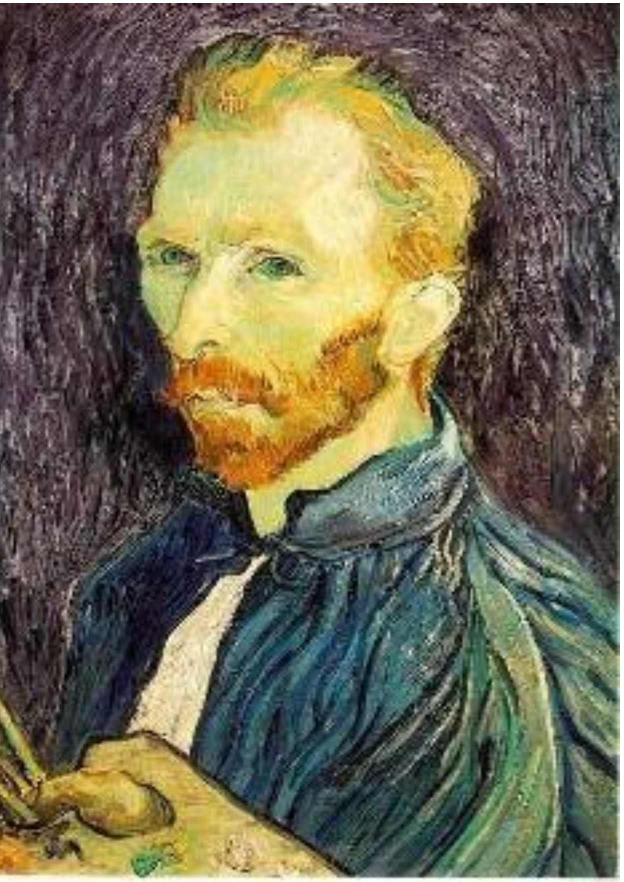
1/4

1/8

Throw away every other row and column to create a 1/2 size image
- called *image sub-sampling*

Slide by Steve Seitz

Image sub-sampling



1/2

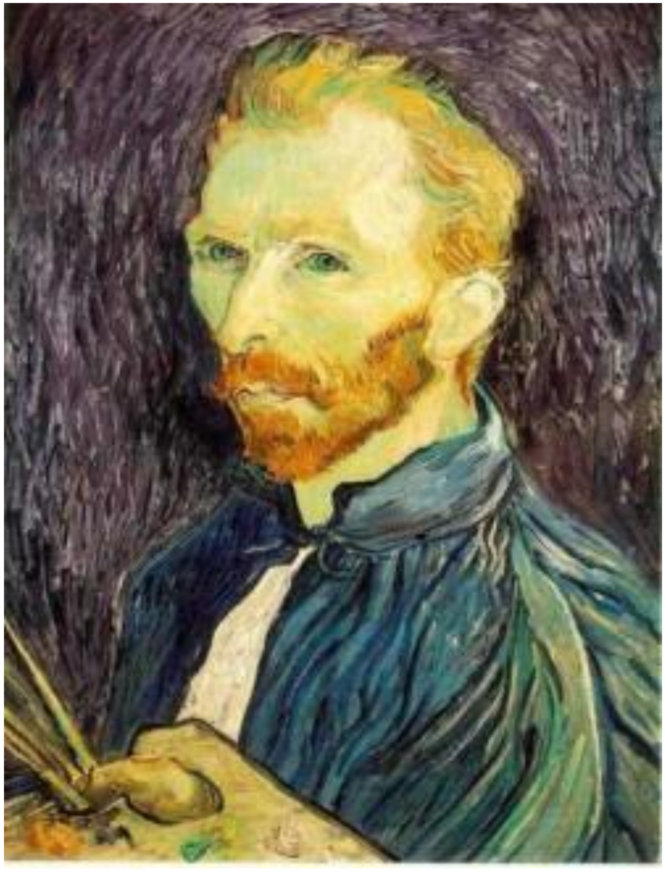
1/4 (2x zoom)

1/8 (4x zoom)

Aliasing! What do we do?

Slide by Steve Seitz

Gaussian (lowpass) pre-filtering



Gaussian 1/2

G 1/4

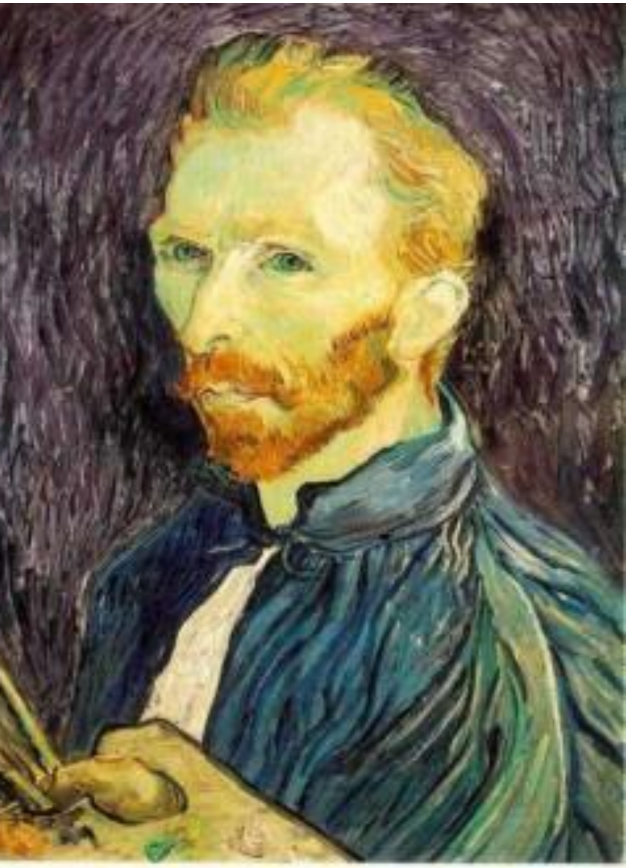
G 1/8

Solution: filter the image, *then* subsample

- Filter size should double for each ½ size reduction. Why?

Slide by Steve Seitz

Subsampling with Gaussian pre-filtering



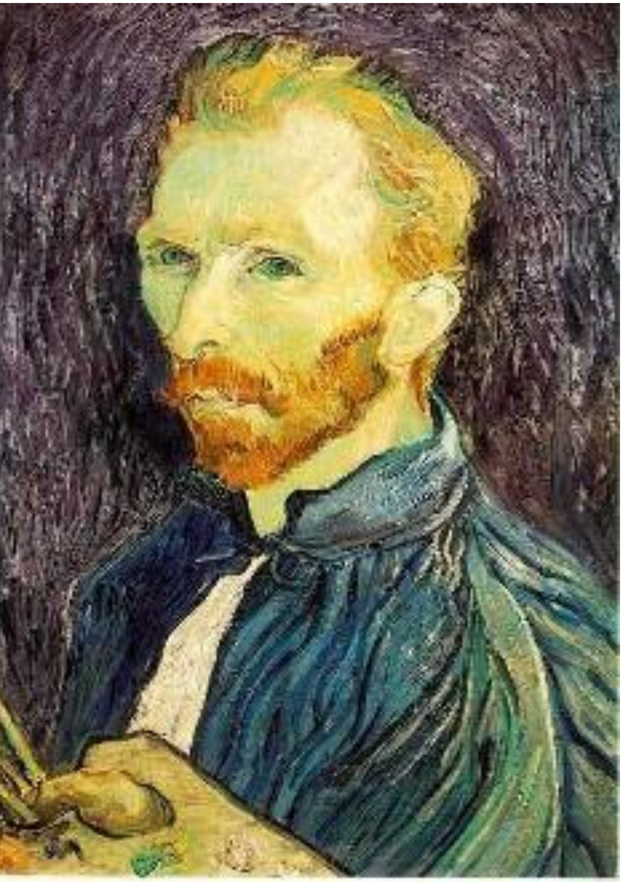
Gaussian 1/2

G 1/4

G 1/8

Slide by Steve Seitz

Compare with...



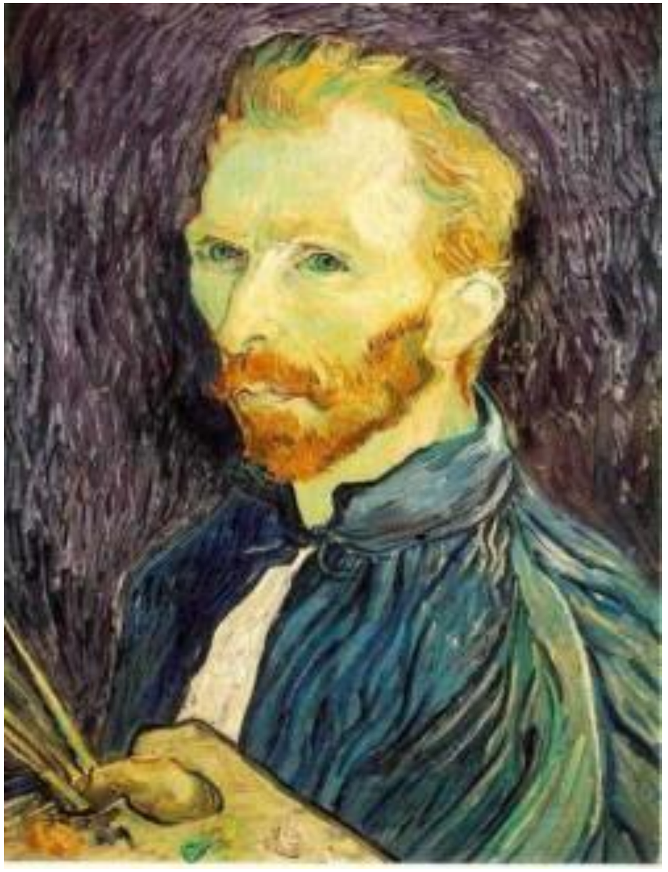
1/2

1/4 (2x zoom)

1/8 (4x zoom)

Slide by Steve Seitz

Gaussian (lowpass) pre-filtering



Gaussian 1/2

G 1/4

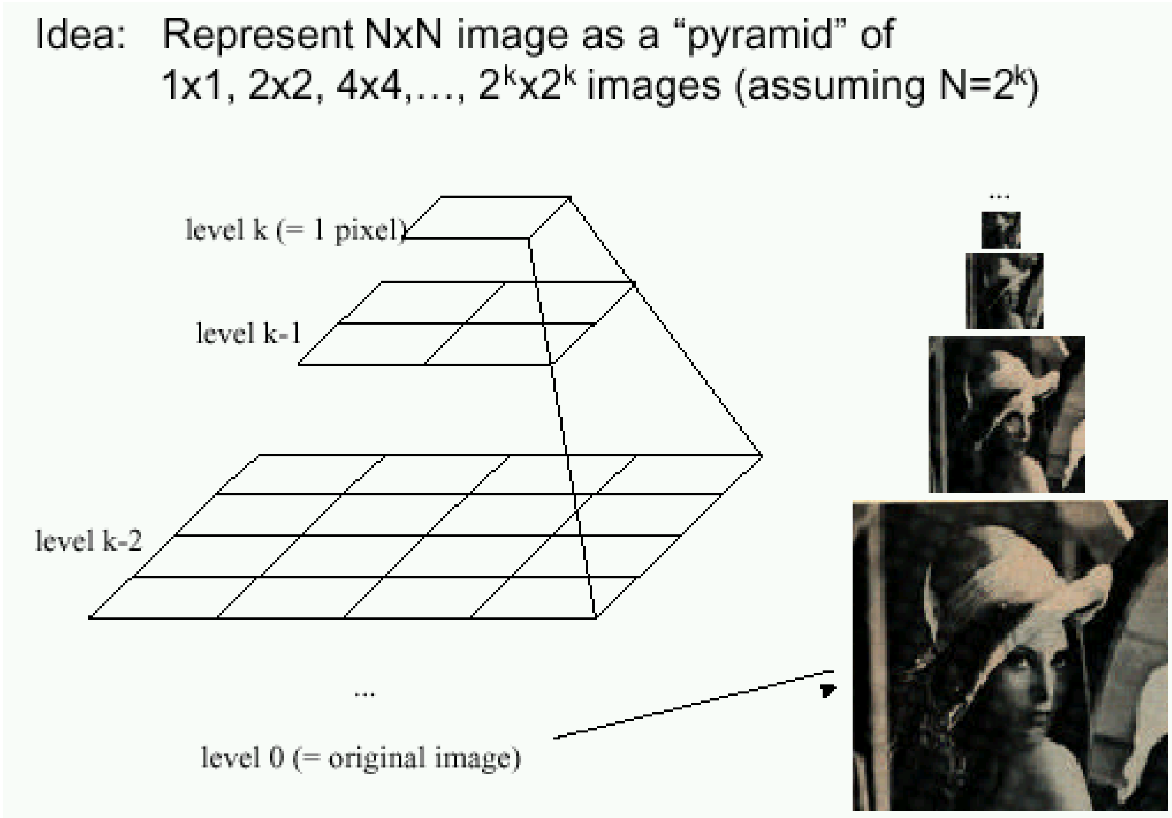
G 1/8

Solution: filter the image, *then* subsample

- Filter size should double for each ½ size reduction. Why?
- How can we speed this up?

Slide by Steve Seitz

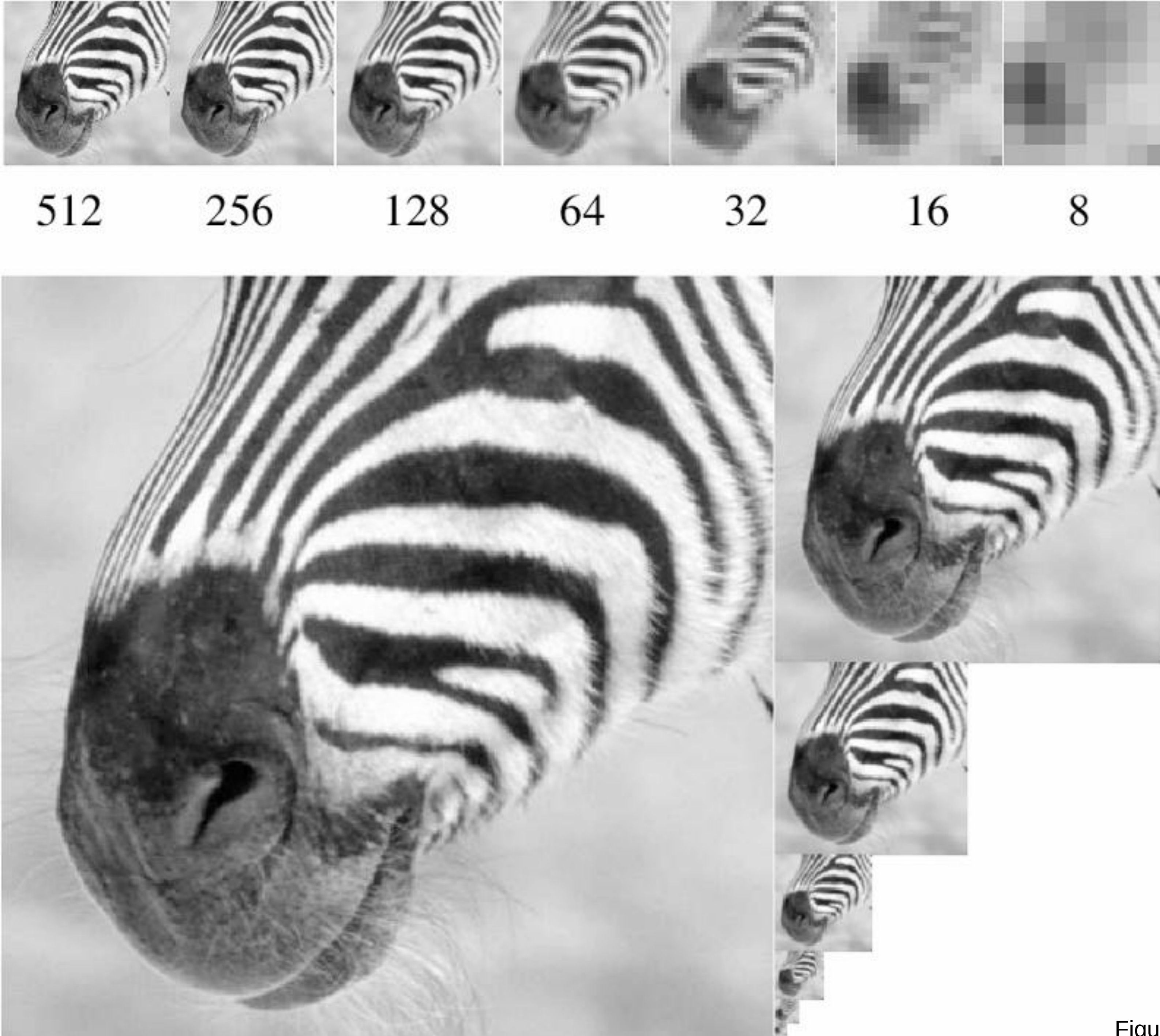
Image Pyramids



Known as a **Gaussian Pyramid** [Burt and Adelson, 1983]

- In computer graphics, a *mip map* [Williams, 1983]
- A precursor to *wavelet transform*

Slide by Steve Seitz



A bar in the big images is a hair on the zebra’s nose; in smaller images, a stripe; in the smallest, the animal’s nose

Figure from David Forsyth

What are they good for?

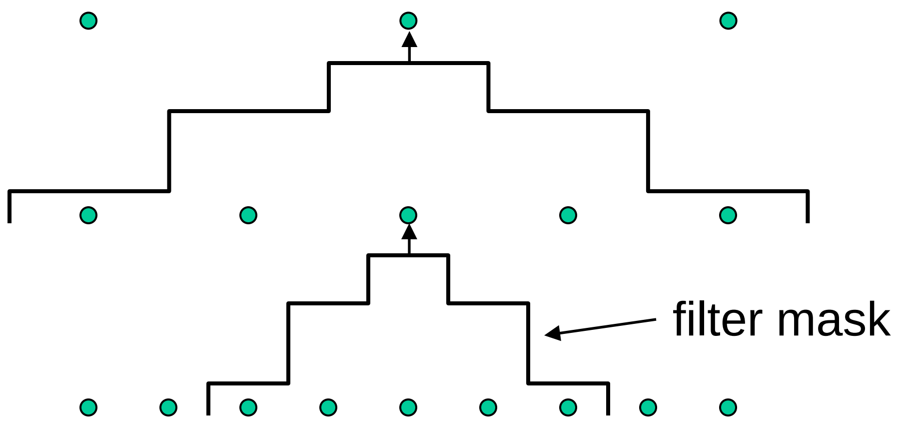
Improve Search

- Search over translations
  - Like project 1
  - Classic coarse-to-fine strategy
- Search over scale
  - Template matching
  - E.g. find a face at different scales

Pre-computation

- Need to access image at different blur levels
- Useful for texture mapping at different resolutions (called mip-mapping)

Gaussian pyramid construction



Repeat
- Filter
- Subsample

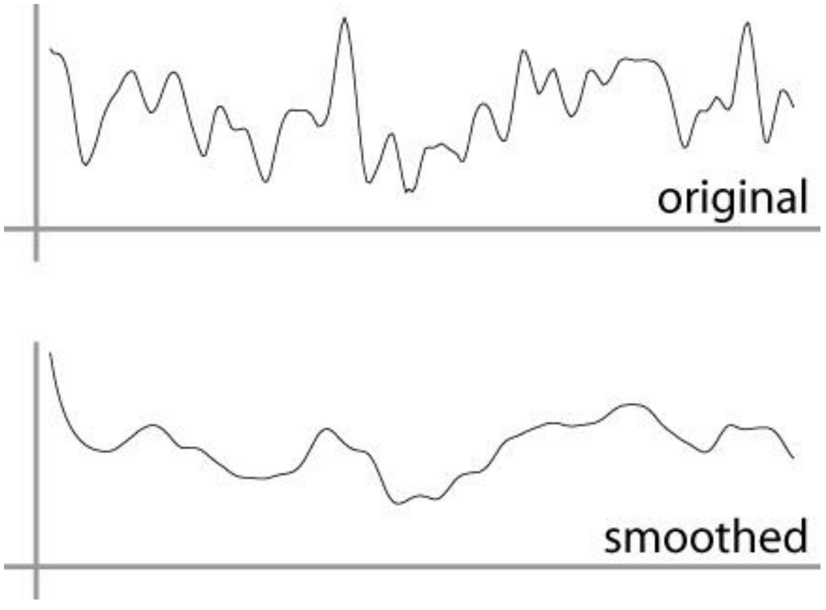
Until minimum resolution reached
- can specify desired number of levels (e.g., 3-level pyramid)

The whole pyramid is only 4/3 the size of the original image!

Slide by Steve Seitz

Continuous convolution: warm-up

- Can apply sliding-window average to a continuous function just as well
  - output is continuous
  - integration replaces summation



Continuous convolution

- Sliding average expressed mathematically:

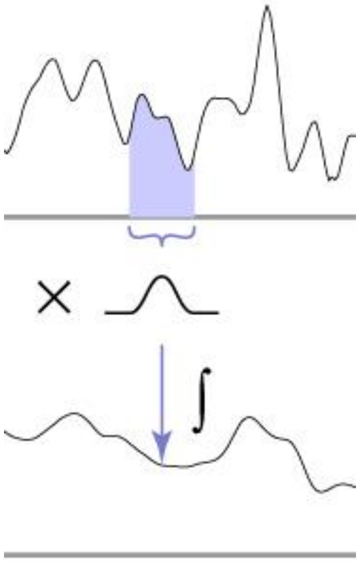
$$g_{\text{smooth}}(x) = \frac{1}{2r}\int_{x-r}^{x+r} g(t)dt$$

  – note difference in normalization (only for box)

- Convolution just adds weights

$$(f \star g)(x) = \int_{-\infty}^{\infty} f(t)g(x-t)dt$$

  – weighting is now by a function
  – weighted integral is like weighted average
  – again bounds are set by support of *f(x)*



One more convolution

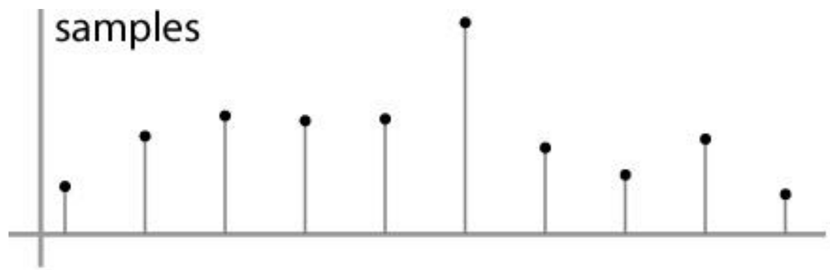
- Continuous–discrete convolution

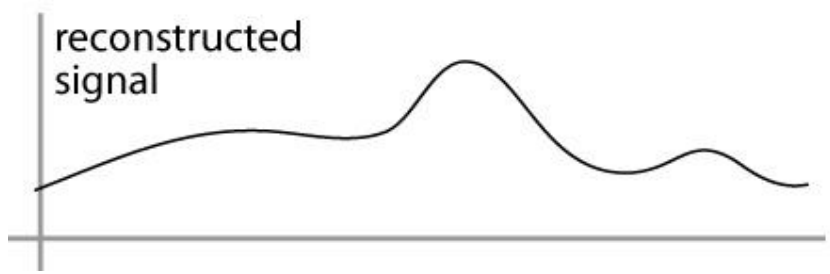
$$(a \star f)(x) = \sum_i a[i] f(x - i)$$

$$(a \star f)(x, y) = \sum_{i,j} a[i, j] f(x - i, y - j)$$

  - used for reconstruction and resampling

Reconstruction



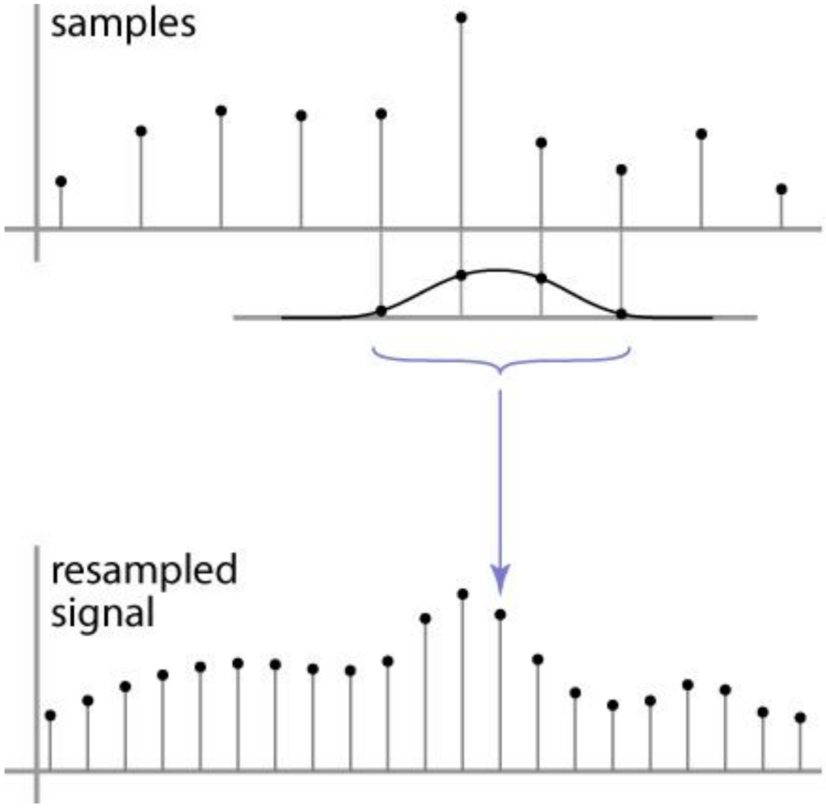


Resampling

- Changing the sample rate
  - in images, this is enlarging and reducing
- Creating more samples:
  - increasing the sample rate
  - “upsampling”
  - “enlarging”
- Ending up with fewer samples:
  - decreasing the sample rate
  - “downsampling”
  - “reducing”

Resampling

- Reconstruction creates a continuous function
  - forget its origins, go ahead and sample it

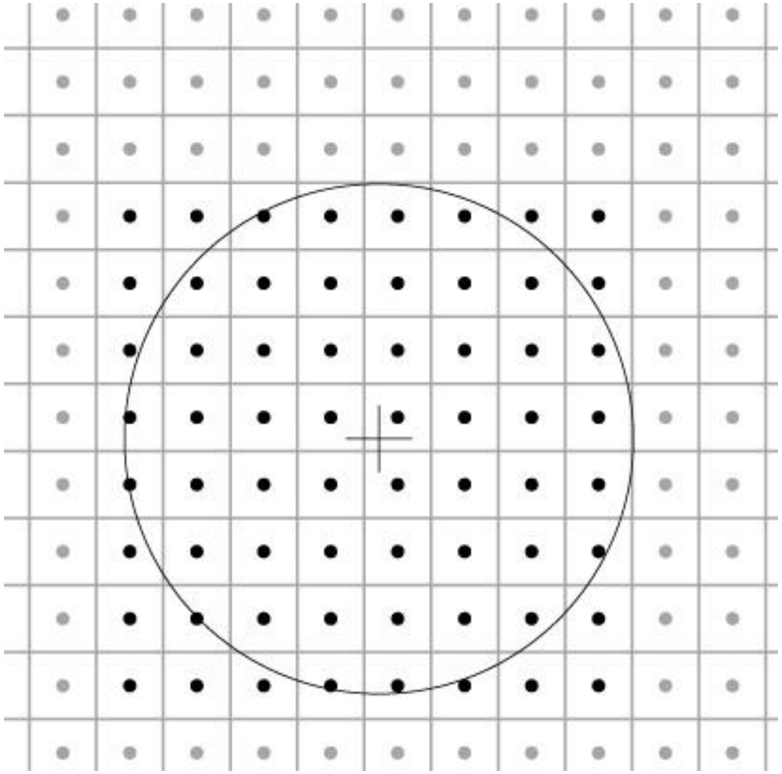


Cont.–disc. convolution in 2D

- same convolution—just two variables now

$$(a \star f)(x, y) = \sum_{i,j} a[i, j] f(x - i, y - j)$$

  - loop over nearby pixels, average using filter weight
  - looks like discrete filter, but offsets are not integers and filter is continuous
  - remember placement of filter relative to grid is variable



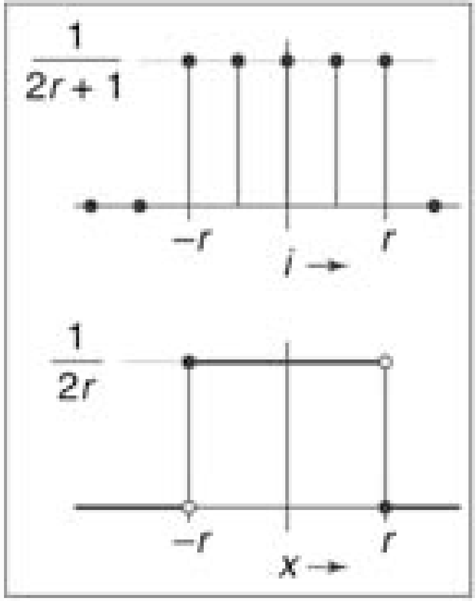
A gallery of filters

- Box filter
  - Simple and cheap
- Tent filter
  - Linear interpolation
- Gaussian filter
  - Very smooth antialiasing filter
- B-spline cubic
  - Very smooth

Box filter

$$a_{\text{box},r}[i] = \begin{cases} 1/(2r+1) & |i| \le r, \\ 0 & \text{otherwise.} \end{cases}$$

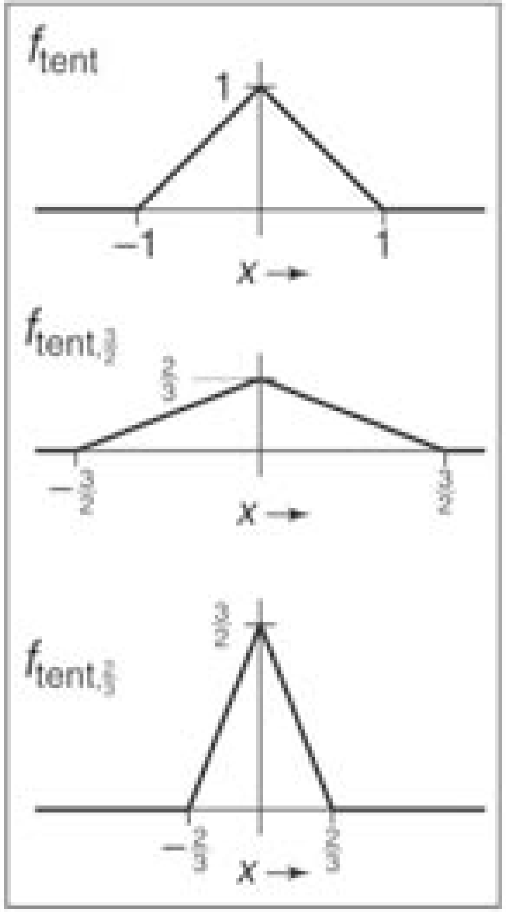
$$f_{\text{box},r}(x) = \begin{cases} 1/(2r) & -r \le x < r, \\ 0 & \text{otherwise.} \end{cases}$$



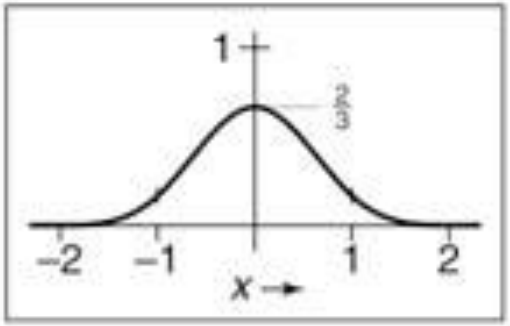
Tent filter

$$f_{\text{tent}}(x) = \begin{cases} 1 - |x| & |x| < 1, \\ 0 & \text{otherwise;} \end{cases}$$

$$f_{\text{tent},r}(x) = \frac{f_{\text{tent}}(x/r)}{r}.$$

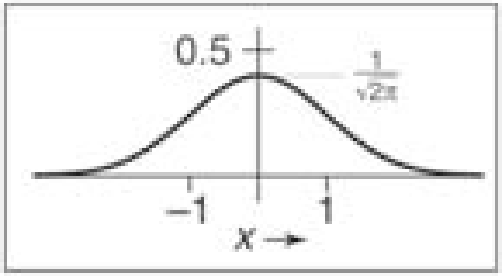


B-Spline cubic



$$f_B(x) = \frac{1}{6} \begin{cases} -3(1-|t|)^3 + 3(1-|t|)^2 + 3(1-|t|) + 1 & -1 \le t \le 1, \\ (2-|t|)^3 & 1 \le |t| \le 2, \\ 0 & \text{otherwise.} \end{cases}$$

Gaussian filter



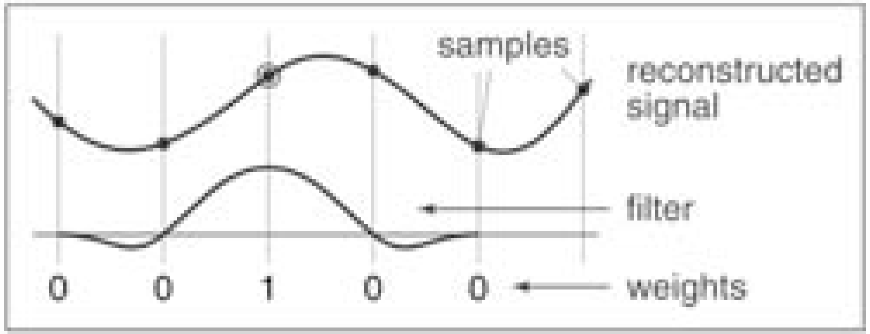
$$f_g(x) = \frac{1}{\sqrt{2\pi}} e^{-x^2/2}.$$

Effects of reconstruction filters

- For some filters, the reconstruction process winds up implementing a simple algorithm
- Box filter (radius 0.5): nearest neighbor sampling
  - box always catches exactly one input point
  - it is the input point nearest the output point
  - so output[i, j] = input[round(x(i)), round(y(j))]
    x(i) computes the position of the output coordinate i on the input grid
- Tent filter (radius 1): linear interpolation
  - tent catches exactly 2 input points
  - weights are a and (1 – a)
  - result is straight-line interpolation from one point to the next

Properties of filters

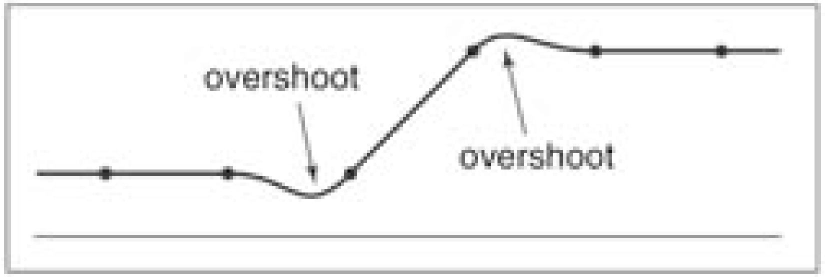
- Degree of continuity
- Impulse response
- Interpolating or no
- Ringing, or overshoot

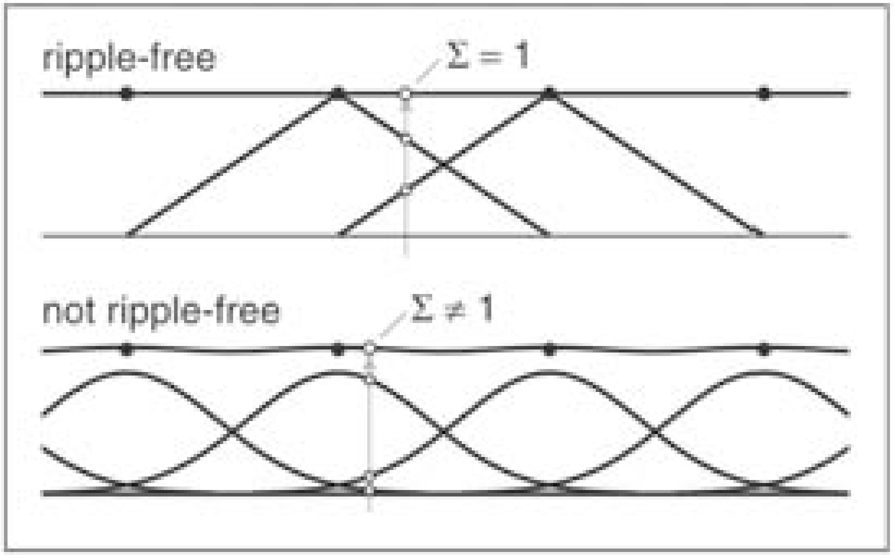


interpolating filter used for reconstruction

Ringing, overshoot, ripples

- Overshoot
  - caused by negative filter values
- Ripples
  - constant in, non-const. out
  - ripple free when:





Yucky details

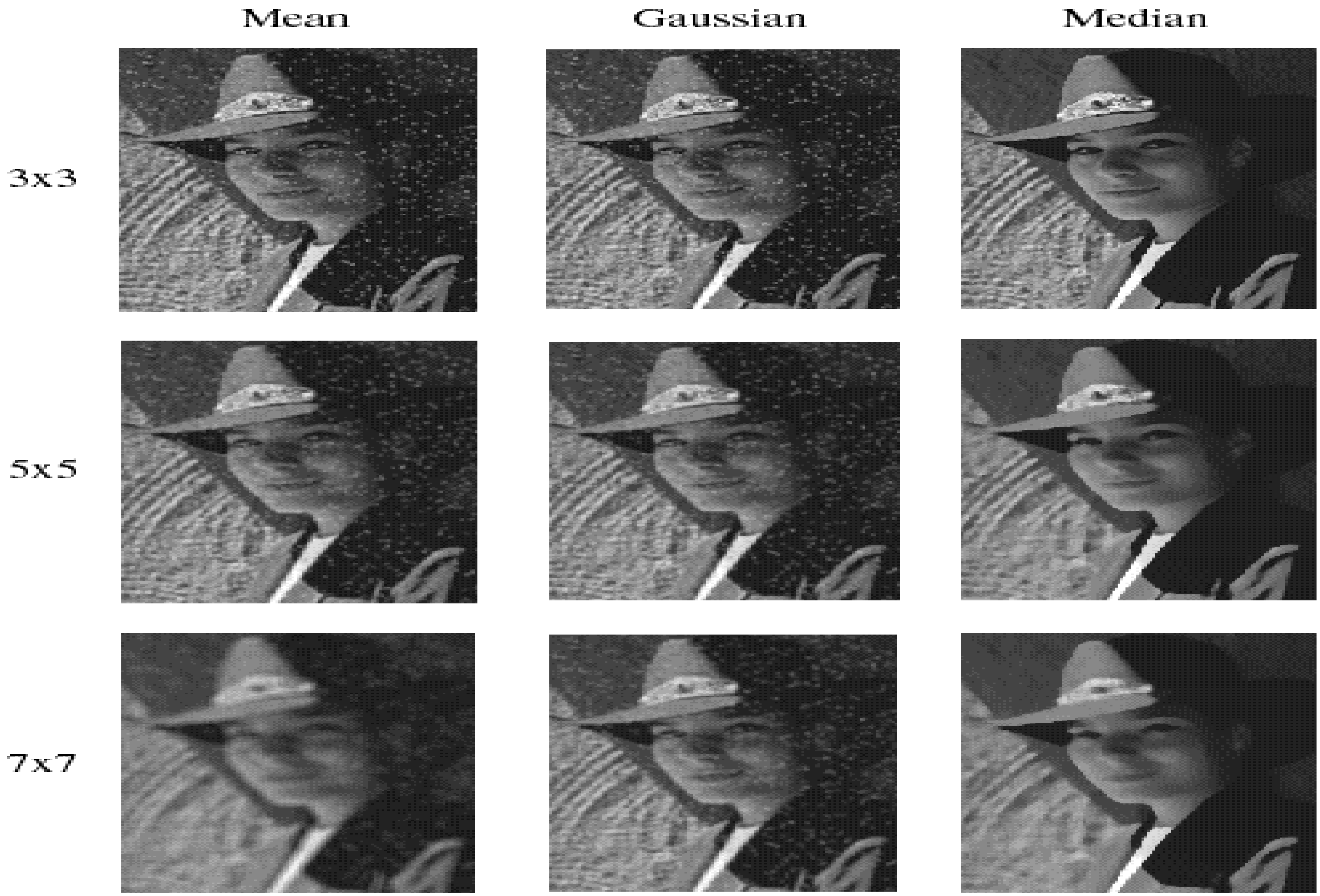
- What about near the edge?
  - the filter window falls off the edge of the image
  - need to extrapolate
  - methods:
    - clip filter (black)
    - wrap around
    - copy edge
    - reflect across edge
    - vary filter near edge

Median filters

- A **Median Filter** operates over a window by selecting the median intensity in the window.
- What advantage does a median filter have over a mean filter?
- Is a median filter a kind of convolution?

Slide by Steve Seitz

Comparison: salt and pepper noise



Slide by Steve Seitz