Sampling and pixels

CS 178, Spring 2011

Begun 4/14/11. Finished 4/19.

Marc Levoy
Computer Science Department
Stanford University

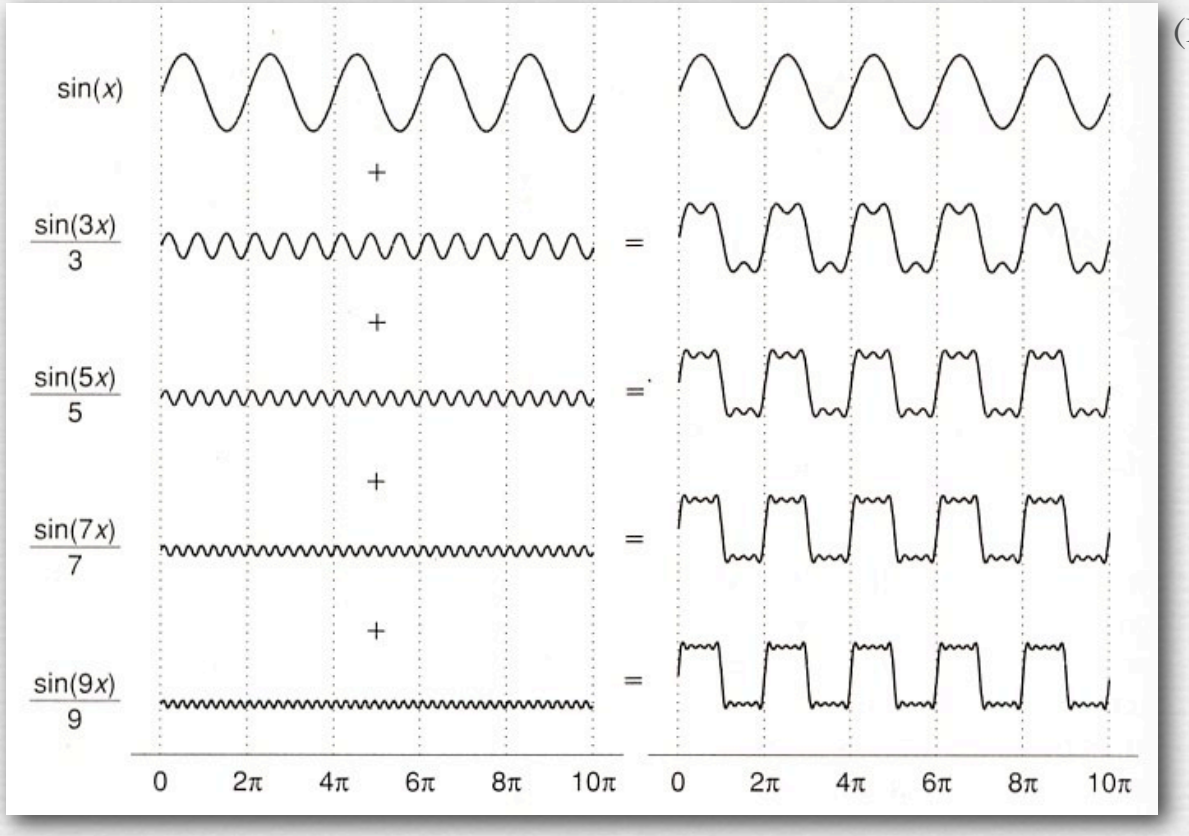
Why study sampling theory?

- Why do I sometimes get moiré artifacts in my images?
- What is an antialiasing filter?
- How many megapixels is enough?
- How do I compute circle of confusion for depth of field?
- Is Apple's "Retina Display" just hype?
- What do MTF curves in lens reviews mean?
- What does Photoshop do when you downsize/upsize?
- What's the difference between more pixels and more bits?

Outline

- frequency representations of images
  - filtering, blurring, sharpening
  - MTF as a measure of sharpness in images
- resolution and human perception
  - the spatial resolution of typical display media
  - the acuity of the human visual system
  - the right way to compute circle of confusion (*C*)
- sampling and aliasing
  - aliasing in space and time, 1D and 2D, images and audio
  - prefiltering using convolution to avoid aliasing
  - prefiltering and sampling in photography
- sampling versus quantization

Frequency representations



(Foley)

- a sum of sine waves, each of different wavelength (*frequency*) and height (*amplitude*), can approximate arbitrary functions
- to adjust horizontal position (*phase*), replace with cosine waves, or use a mixture of sine and cosine waves

Fourier analysis

- Fourier series: any continuous, integrable, periodic function can be represented as an infinite series of sines and cosines

$$f(x) = \frac{a_0}{2} + \sum_{n=1}^{\infty} \left[a_n \cos(nx) + b_n \sin(nx) \right]$$

- Discrete Fourier transform (DFT):

$$X_k = \sum_{n=0}^{N-1} x_n e^{-\frac{2\pi i}{N} k n} \quad , k = 0, \ldots, N-1$$

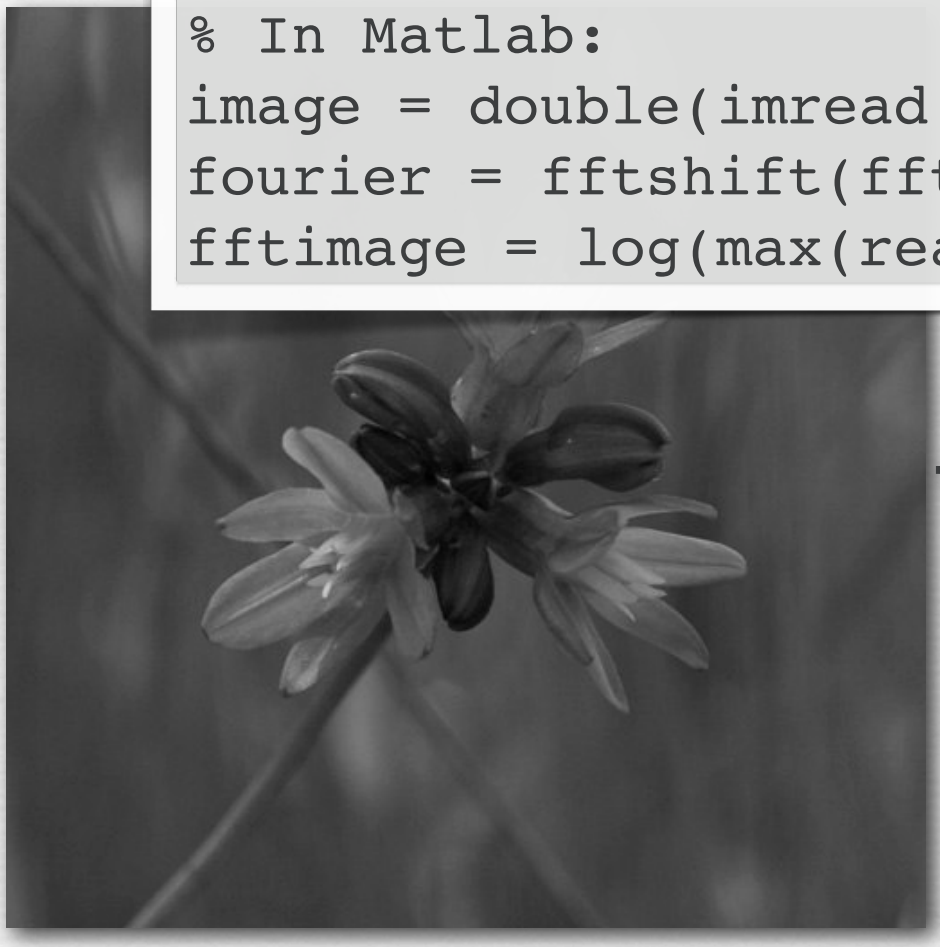
$$x_n = \sum_{k=0}^{N-1} X_k e^{-\frac{2\pi i}{N} k n} \quad , n = 0, \ldots, N-1$$

where $e^{i n x} = \cos(nx) + i \sin(nx)$

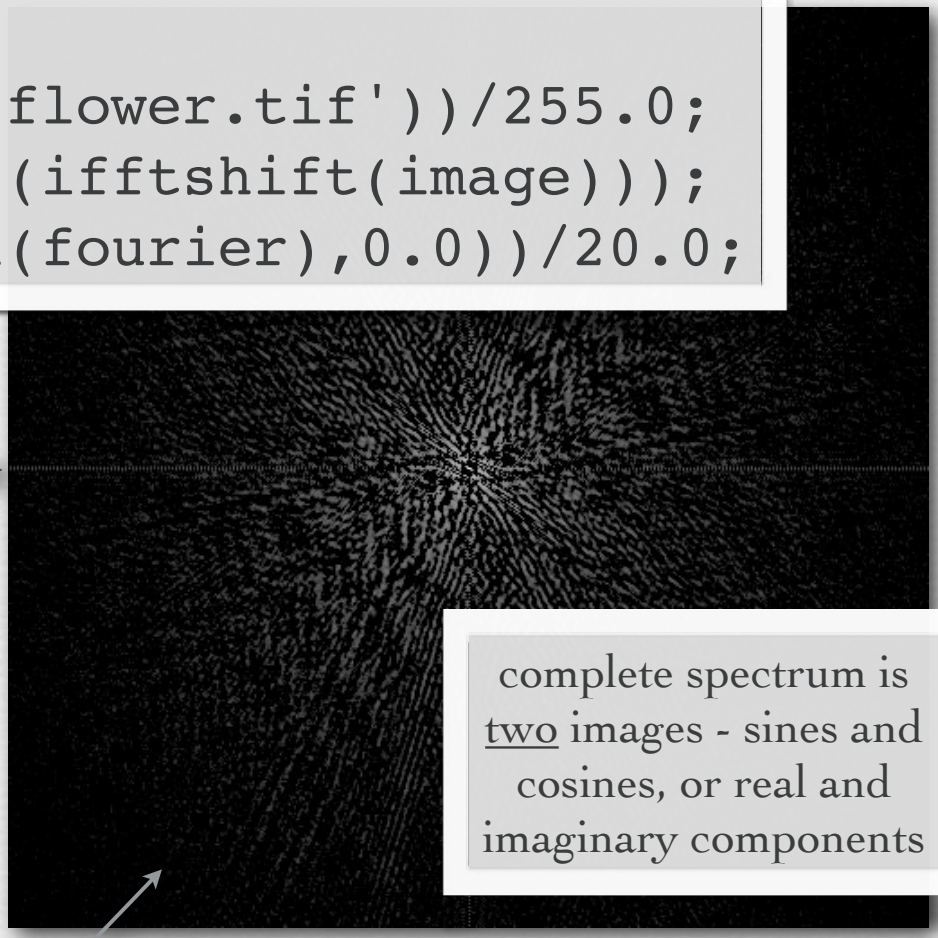
fast version is called the Fast Fourier Transform (FFT)

Fourier transforms of images

```
% In Matlab:
image = double(imread('flower.tif'))/255.0;
fourier = fftshift(fft2(ifftshift(image)));
fftimage = log(max(real(fourier),0.0))/20.0;
```



image



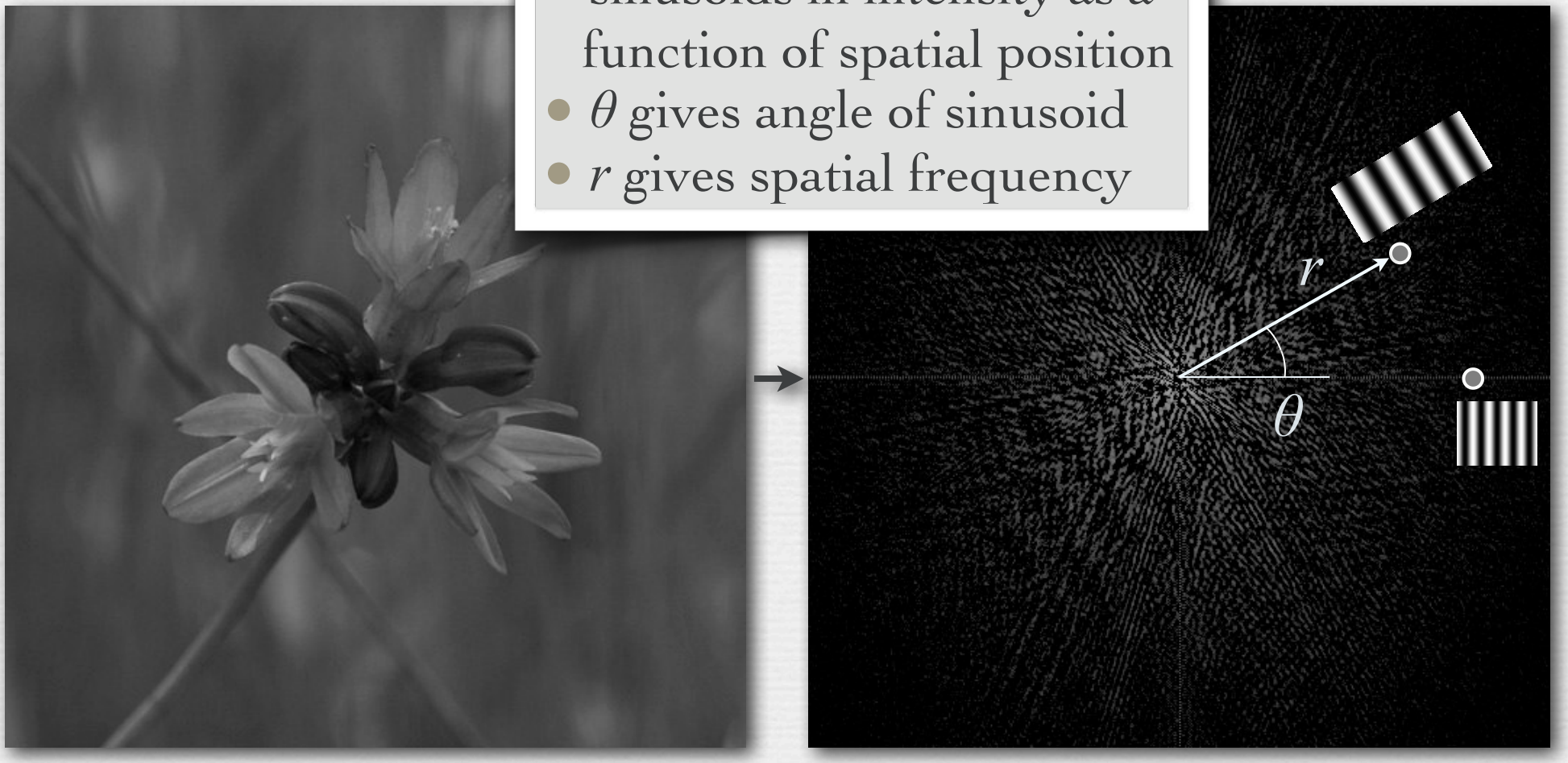
FFT(image)

often called a spectrum

complete spectrum is two images - sines and cosines, or real and imaginary components

Fourier transforms of images

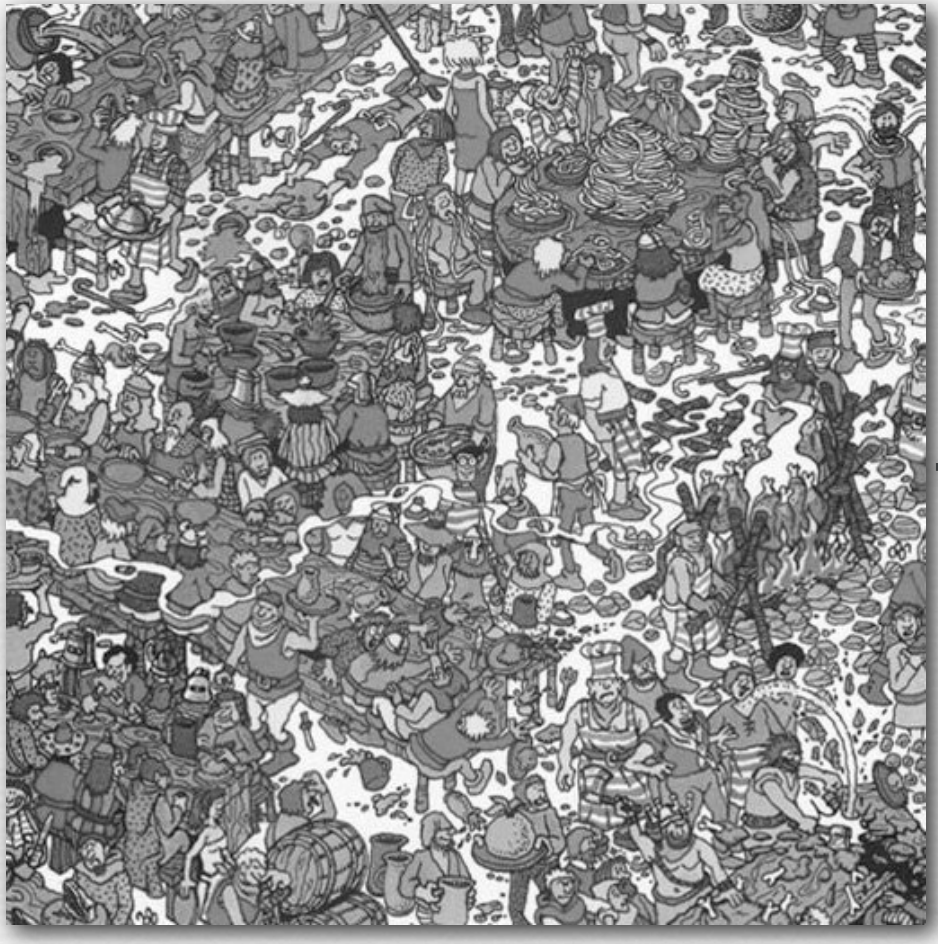
- sinusoids in intensity as a function of spatial position
- θ gives angle of sinusoid
- r gives spatial frequency



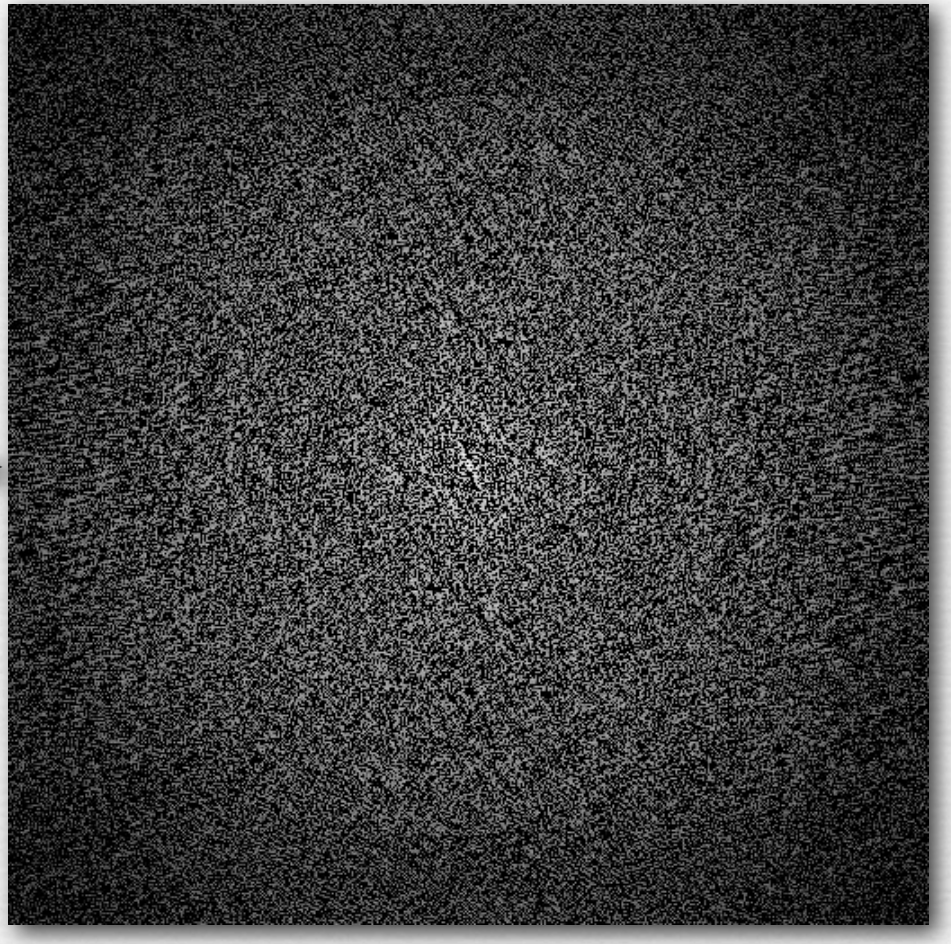
image

FFT(image)

Fourier transforms of images

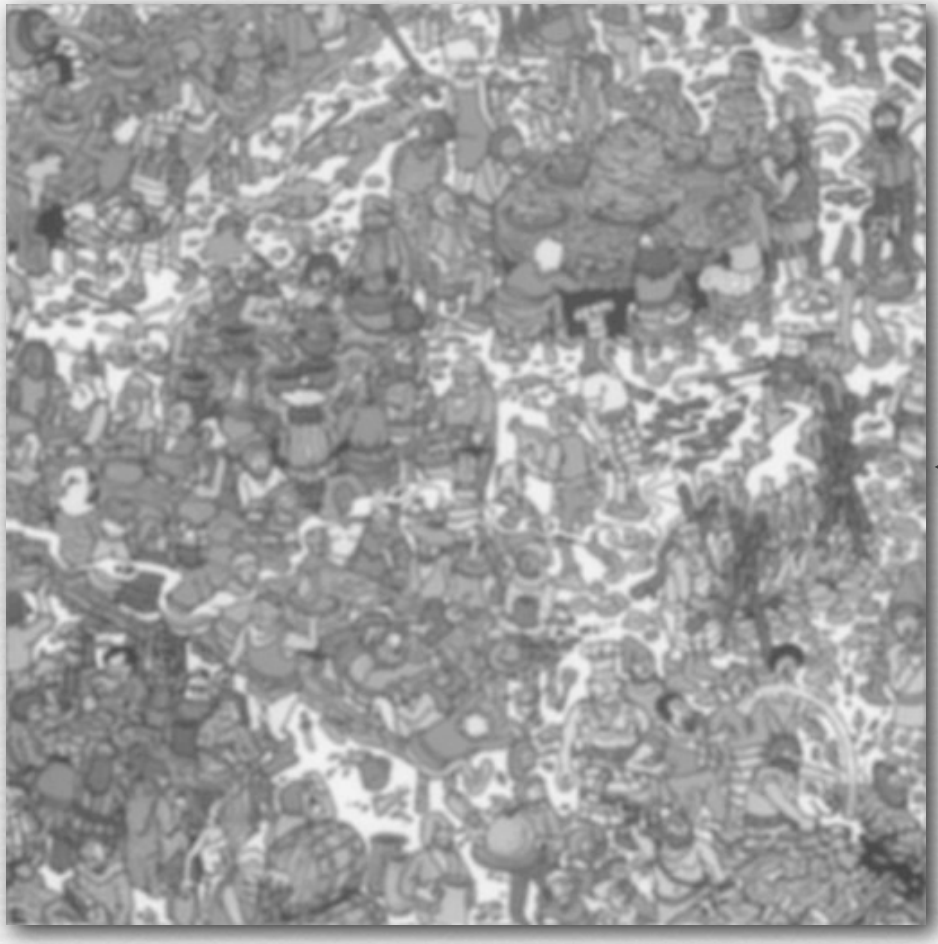


image

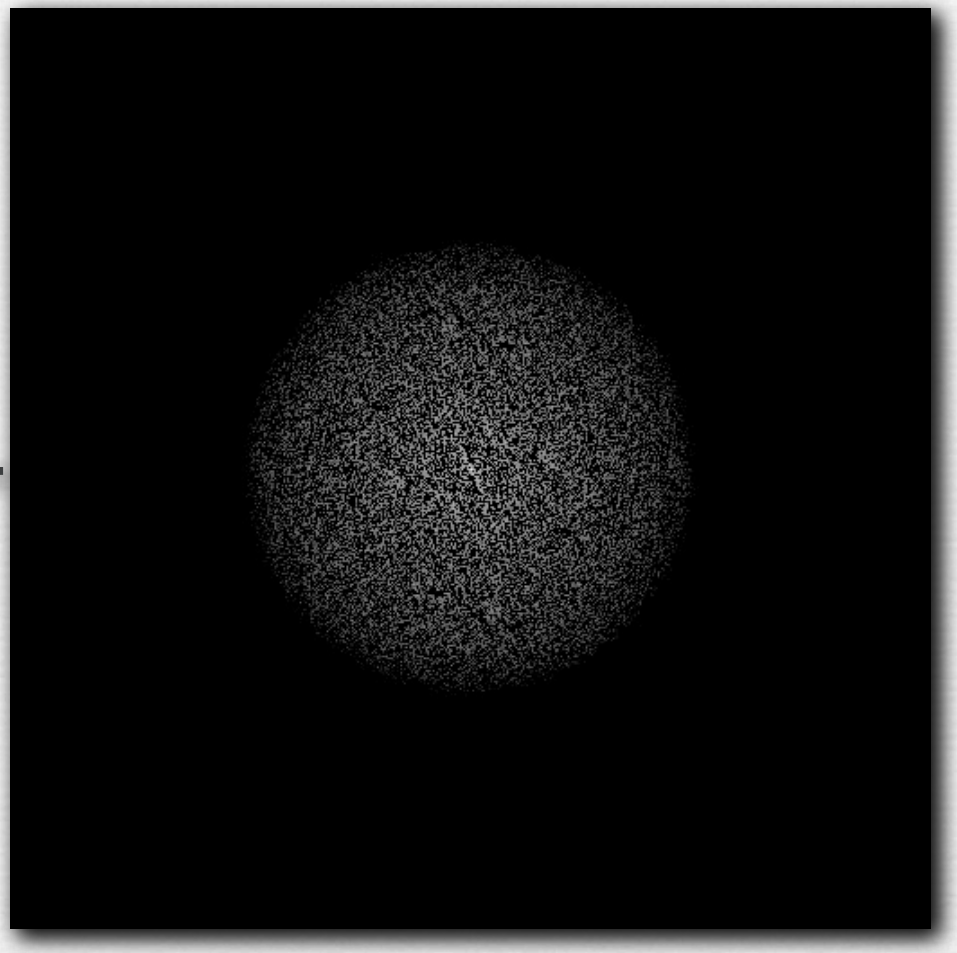


FFT(image)

Blurring in the Fourier domain



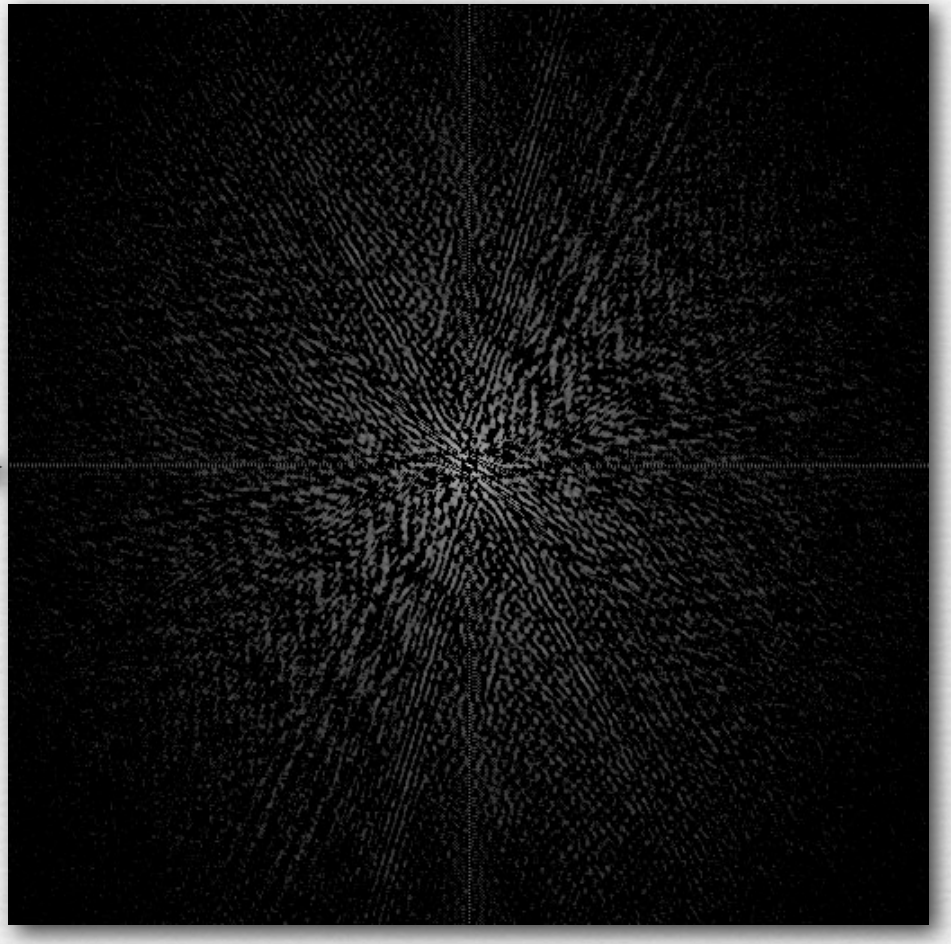
image



FFT(image)

Sharpening in the Fourier domain

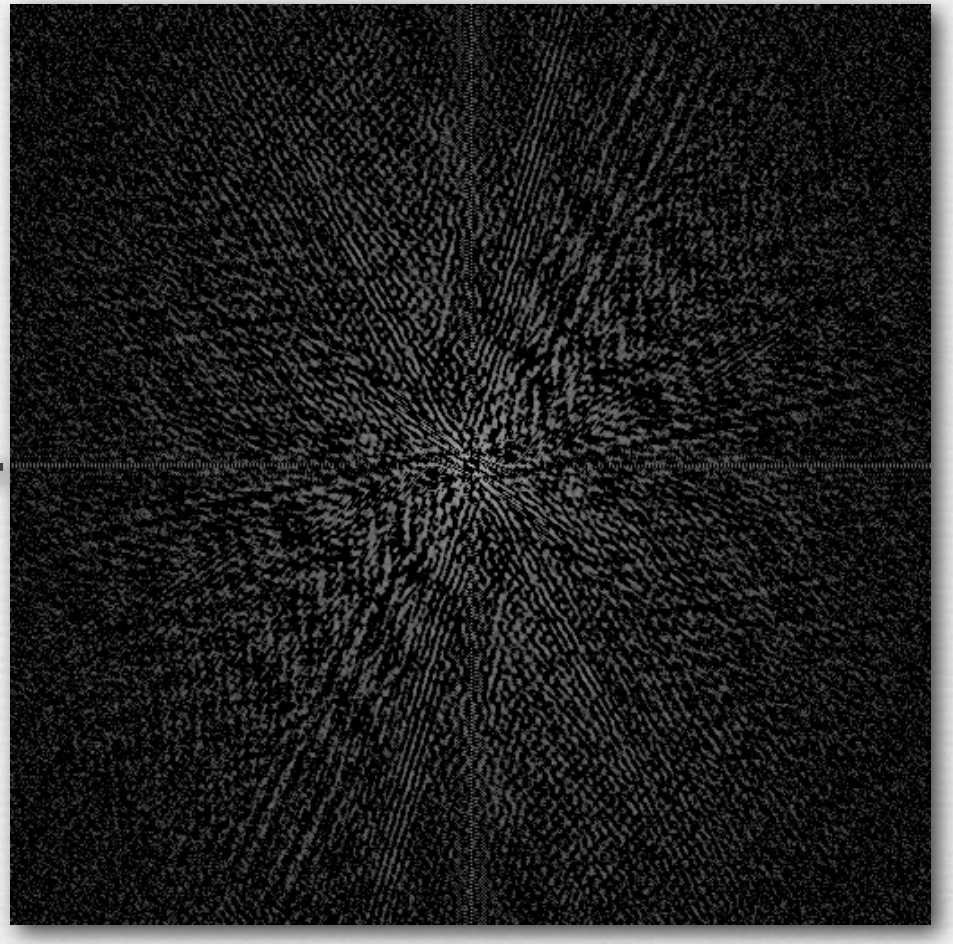
image



FFT(image)

Sharpening in the Fourier domain

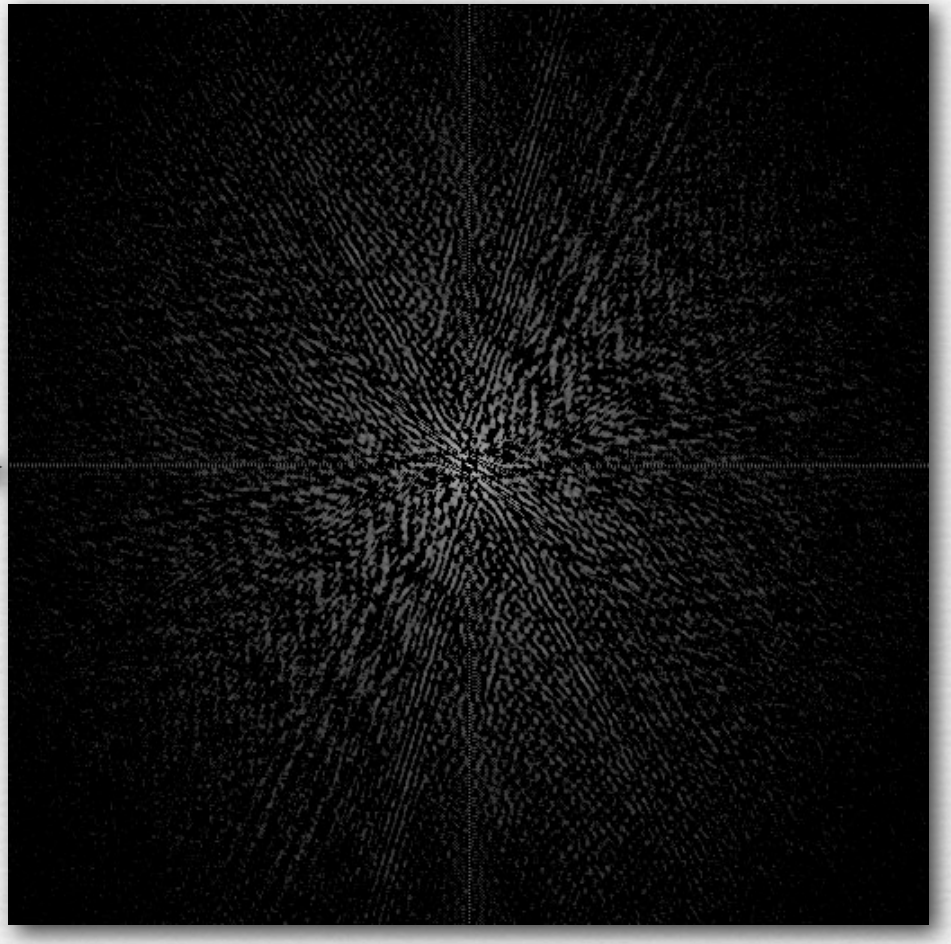
image



FFT(image)

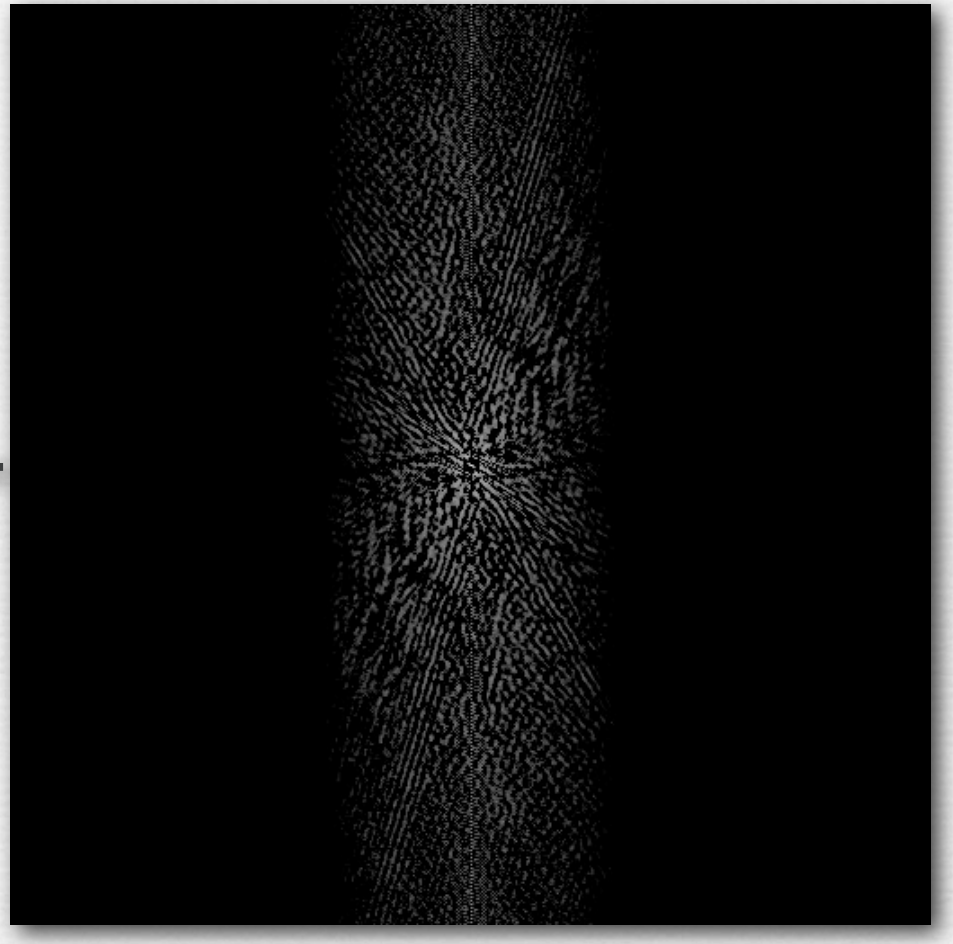
Q. What does this filtering operation do?

image



FFT(image)

Blurring in *x*, sharpening in *y*

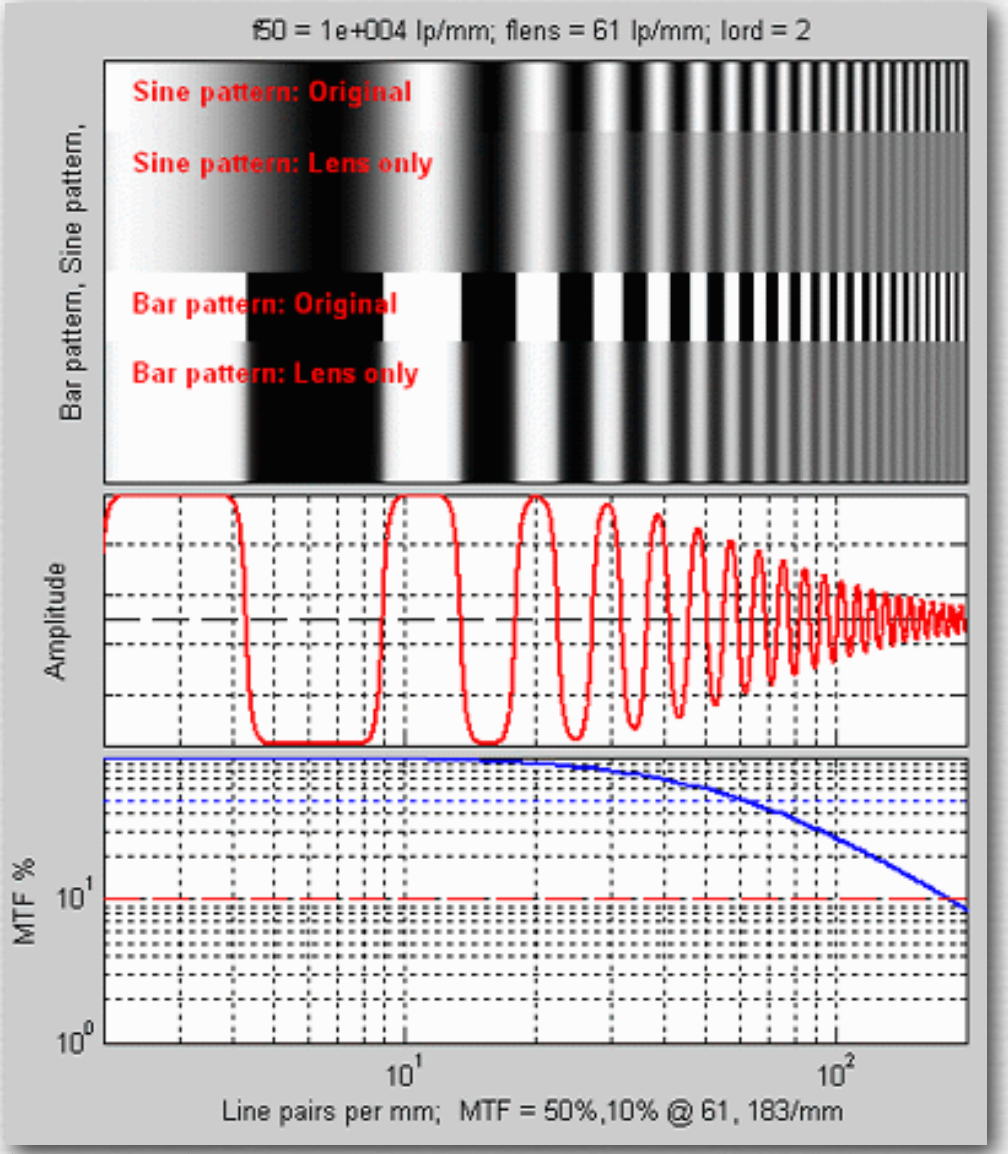


image

FFT(image)

argh, astigmatism!

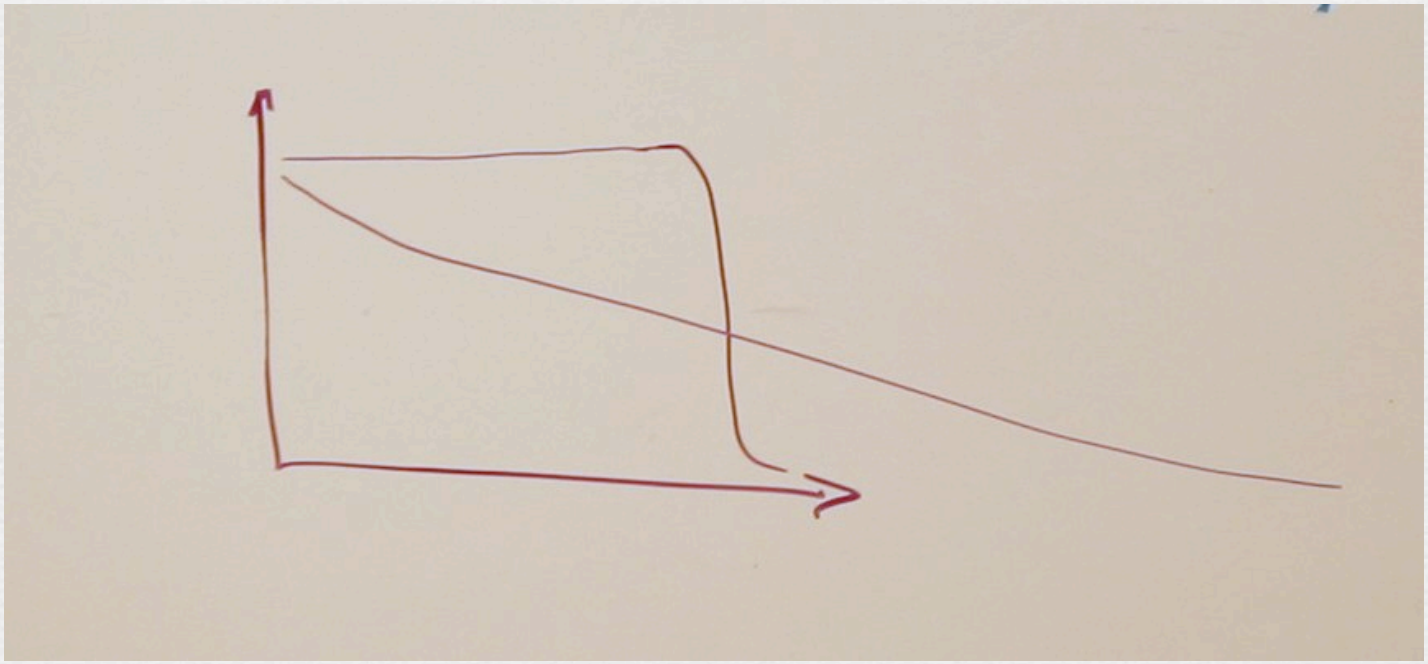
Describing sharpness in images: the modulation transfer function (MTF)



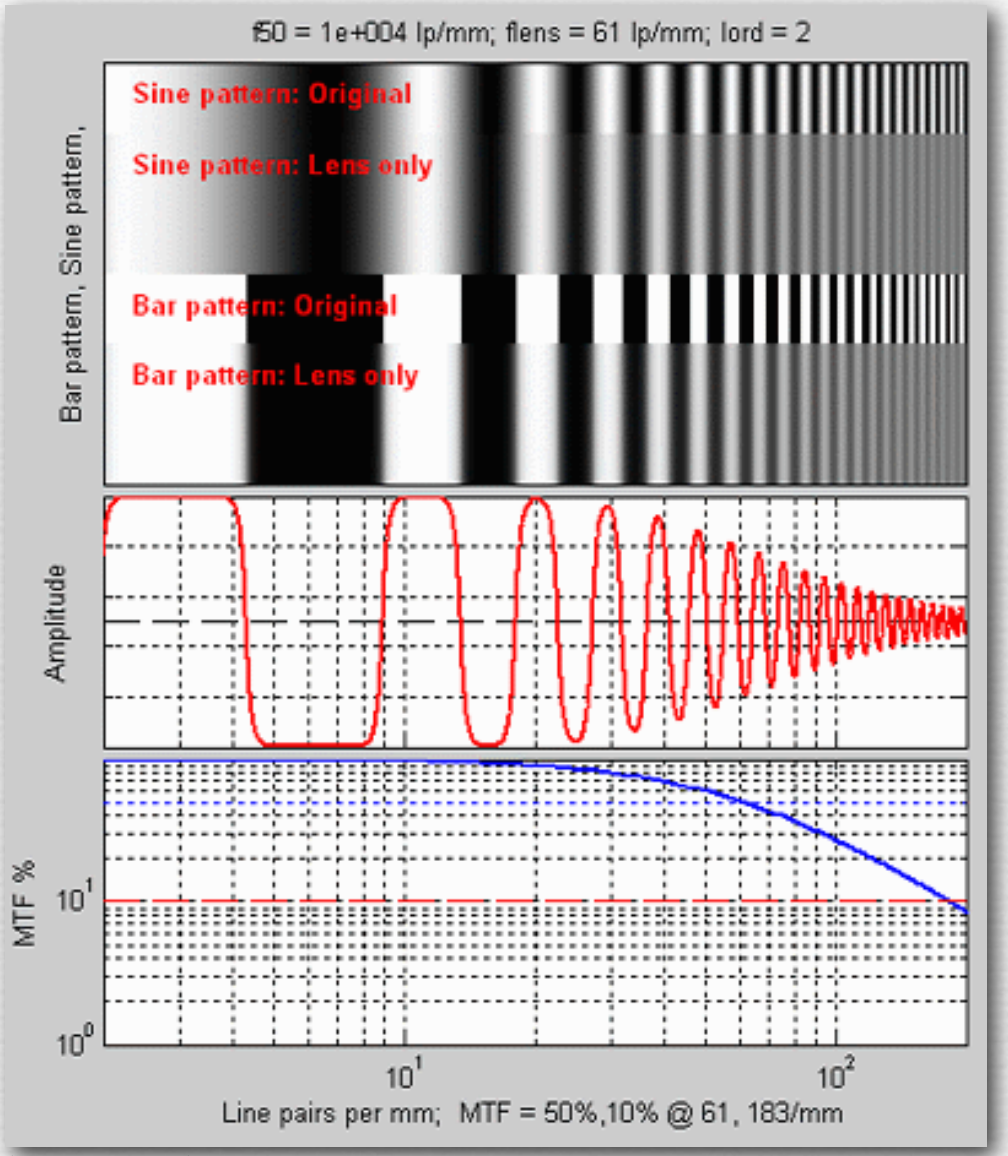
- the amount of each spatial frequency that can be reproduced by an optical system

Two different MTF curves

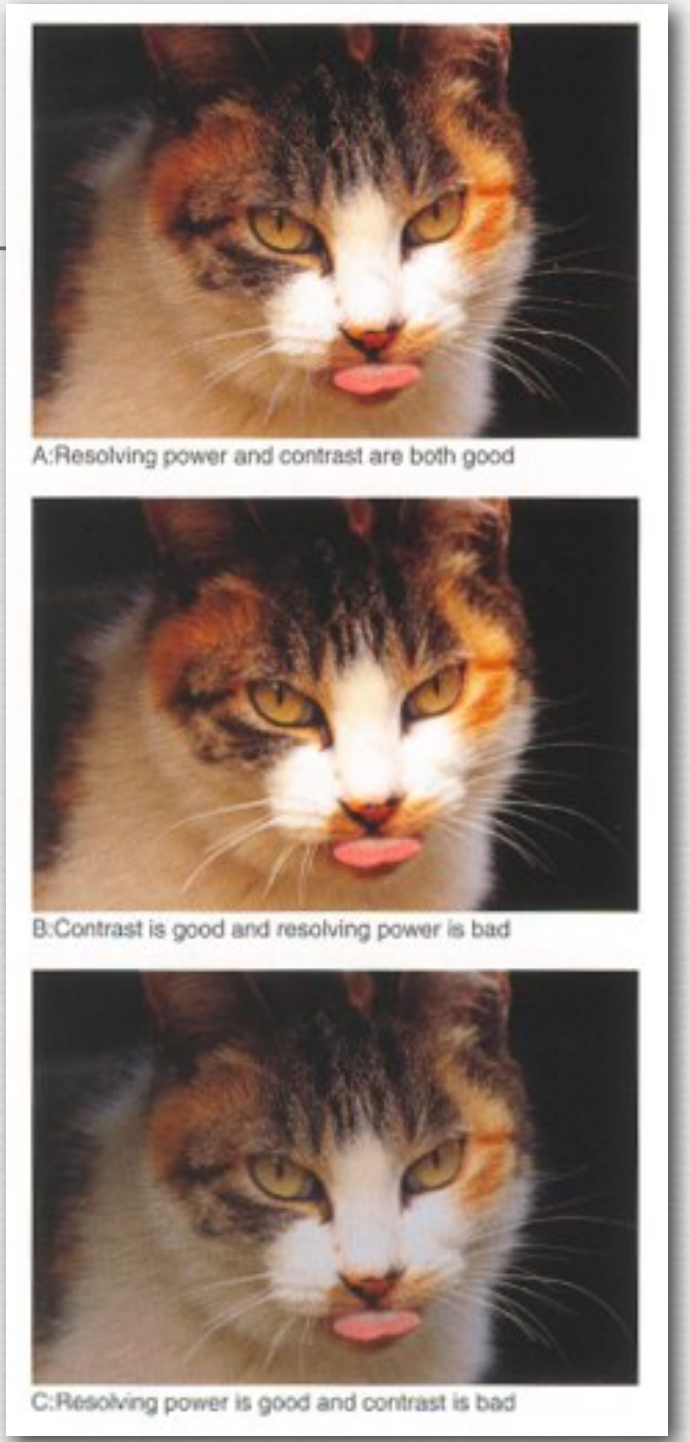
- in one curve, contrast stays high, but drops off at a relatively low resolution
- in the other curve, higher-resolution features are preserved, but contrast is low throughout



Sharpness versus contrast



(imatest.com)



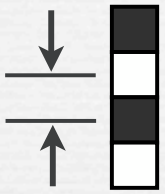
(Canon)

Recap

- any image can be equivalently represented by its Fourier transform, a.k.a. frequency or spectral representation
  - weighted sum of sine and cosine component images
  - each having a frequency, intensity, and orientation in the plane
- filtering, for example blurring or sharpening, can be implemented by amplifying or attenuating selected frequencies
  - i.e. brightening or darkening selected sine or cosine components relative to others, while maintaining same average over all components
  - attenuating high frequencies ≈ *low-pass-filtering* ≈ blurring
  - attenuating low frequencies ≈ *high-pass filtering* ≈ sharpening
  - filtering this way is slow
- MTF measures preservation of frequencies by an optical system
  - subjective image quality depends on both sharpness and contrast

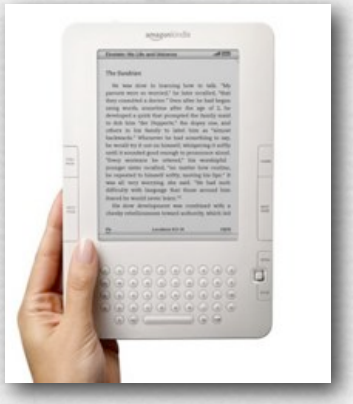
Questions?

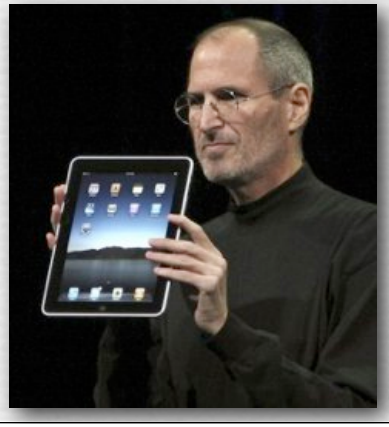
Spatial resolution of display media

pitch = Δx  density = $1/\Delta x$

- Example #1: Macbook Pro (laptop)
  - 900 pixels on 8" high display
  - Δx = 8"/ 900 pixels = 0.0089"/pixel
  - $1/\Delta x$ = 112 dpi (dots per inch)
- Example #2: Kindle 2
  - 800 pixels on 4.8" high display
  - $1/\Delta x$ = 167 dpi
- Example #3: iPad
  - 768 pixels on 5.8" high display
  - $1/\Delta x$ = 132 dpi

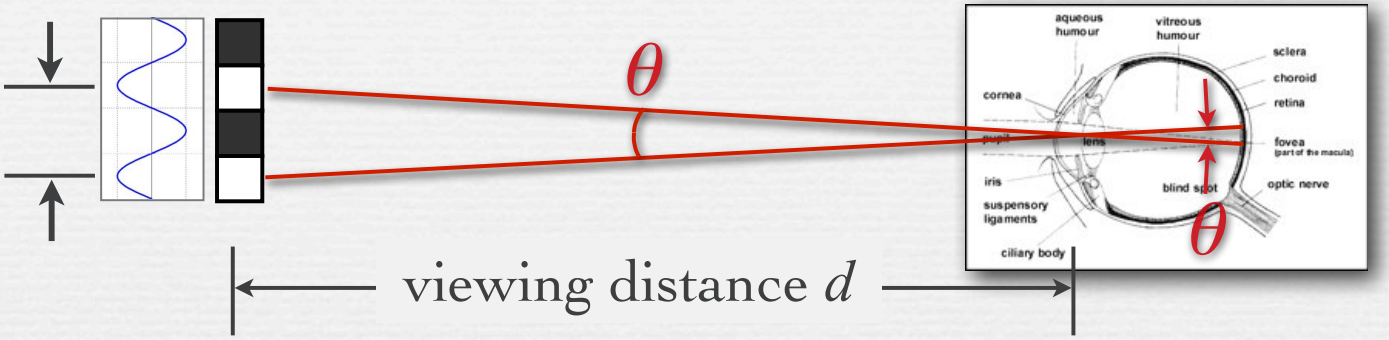
Line printers are 300 dpi. This is why we don't like reading on laptops.





Spatial frequency on the retina

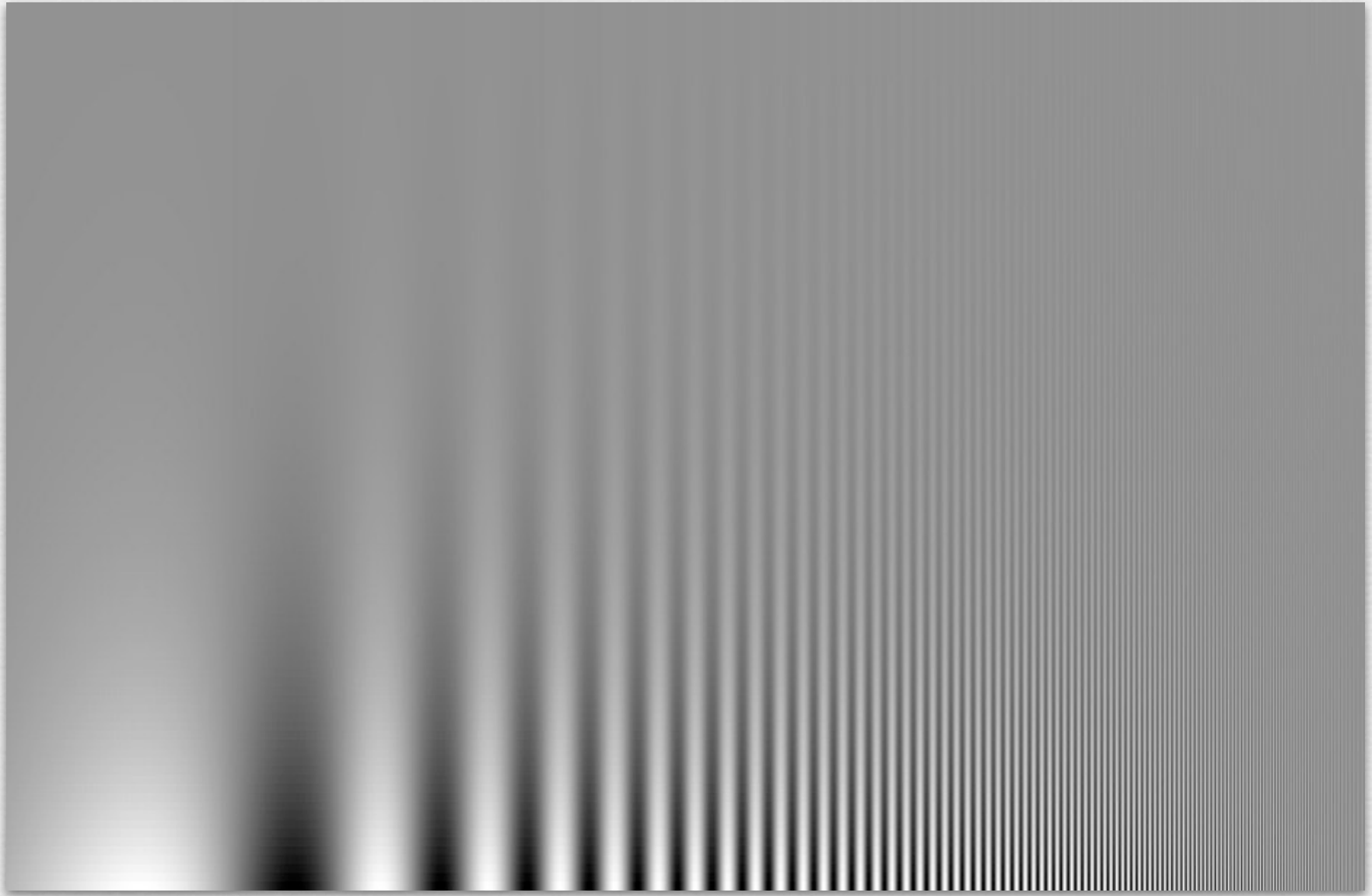
assume the minimum period p of a sine wave is a black-white pixel pair



- Example #1: Macbook Pro viewed at $d = 18$”
  - 900 pixels on 8” high display, $p = 2 \times 0.0089$”
  - retinal arc $\theta = 2 \arctan(p / 2d) = 0.057^o$
  - spatial frequency on retina $1/\theta = 17.6$ cycles per degree

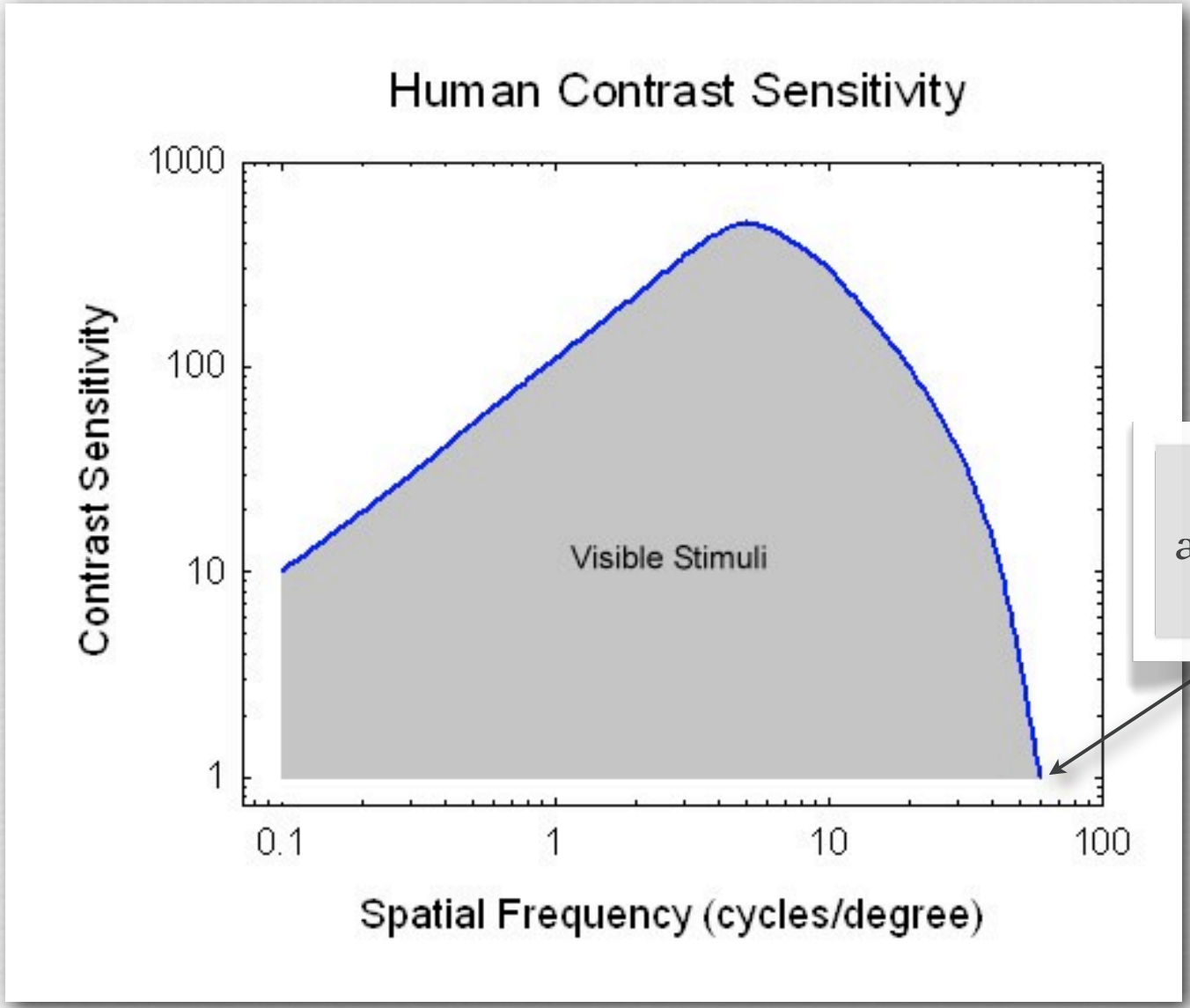
Q. What is the acuity of the human visual system?

Human spatial sensitivity
(Campbell-Robson Chart)



(neurovision.berkeley.edu)

Human spatial sensitivity



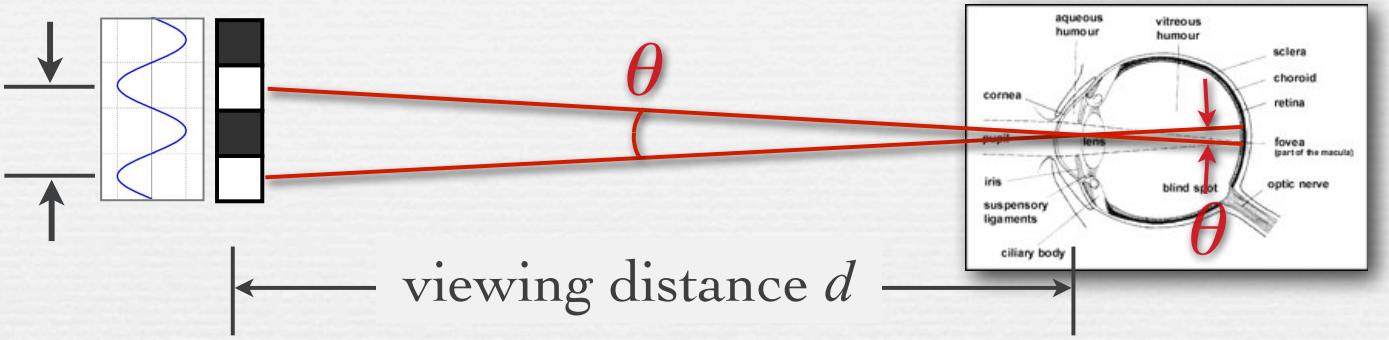
(horizontal axis not comparable to image on previous slide)

cutoff is at about 50 cycles per degree

(psych.ndsu.nodak.edu)

Spatial frequency on the retina

assume the minimum period *p* of a sine wave is a black-white pixel pair



- Example #1: Macbook Pro viewed at *d = 18”*
  - 900 pixels on 8” high display, so *p = 2 × 0.0089”*
  - retinal arc *θ = 2 arctan (p / 2d) = 0.057°*
  - spatial frequency on retina *1/θ = 17.6* cycles per degree

not nearly as high as human acuity

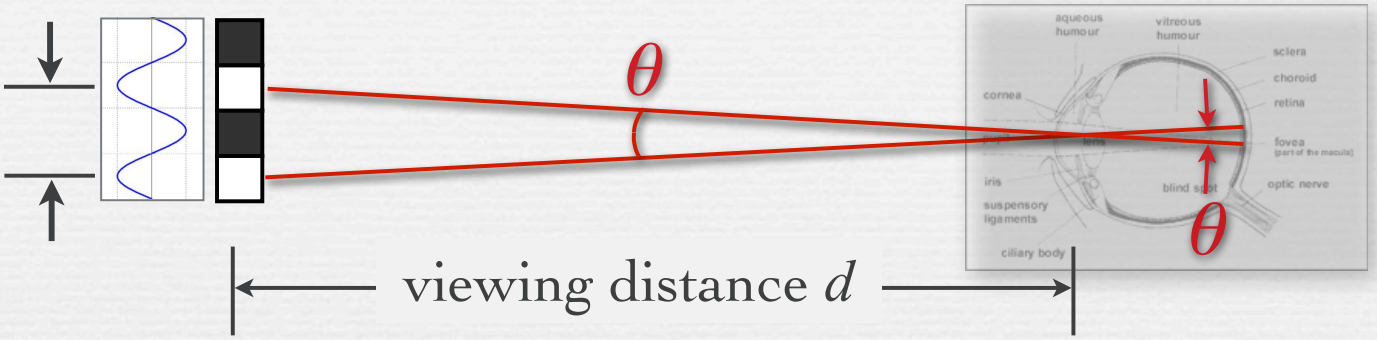
(Graham Flint)

Balboa Park, San Diego

(original is 40K × 20K pixels, Gates Hall print is 72” × 36”)

Spatial frequency on the retina

assume the minimum period p of a sine wave is a black-white pixel pair



- Example #1: Macbook Pro viewed at $d = 18$”
  - 900 pixels on 8” high display, $p = 2 \times 0.0089$”
  - retinal arc $\theta = 2 \arctan(p / 2d) = 0.057°$
  - spatial frequency on retina $1/\theta = 17.6$ cycles per degree

- Example #2: gigapixel photo viewed at $d = 48$”
  - 20,000 pixels on 36” high print, $p = 2 \times 0.0018$”
  - spatial frequency on retina $1/\theta = 232$ cycles per degree

way beyond human acuity

Human acuity & circle of confusion

- the maximum allowable circle of confusion (*C*) in a photograph can be computed from human spatial acuity projected onto the intended display medium

- Example: photographic print from viewed at 12”
  - max human acuity on retina $1/\theta \approx 50$ cycles per degree
  - minimum detectable retinal arc $\theta \approx 0.02^{o}$
  - minimum feature size $p = 2 \times 12” \times \tan(\theta / 2) = 0.0043”$ *(0.1mm)*

- assume 5” × 7” print and Canon 5D II (5616 × 3744 pixels)
  - 5” / 3744 pixels = 0.0017”/pixel *(0.04mm)*
  - therefore, circle of confusion can be 2.5 pixels wide before it’s blurry
  - $C = 6.4\mu$ per pixel × 2.5 pixels = 16μ

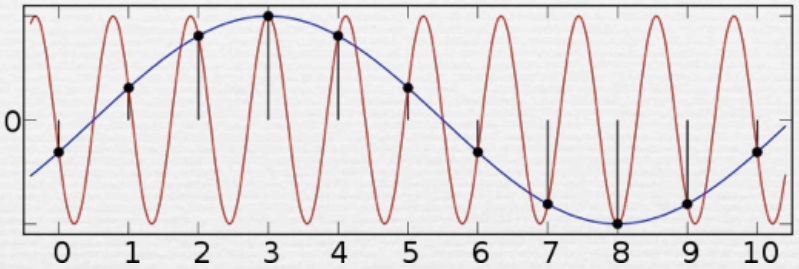
Recap

- spatial resolution of display media is measured by
  - pitch (distance between dots or pixels) or density (dots per inch)
- effect on human observers is measured by
  - retinal angle (degrees of arc) or frequency (cycles per degree)
  - depends on viewing distance
- human spatial acuity is about 50 cycles per degree
  - depends on contrast
  - convert back to pitch to obtain circle of confusion for depth of field

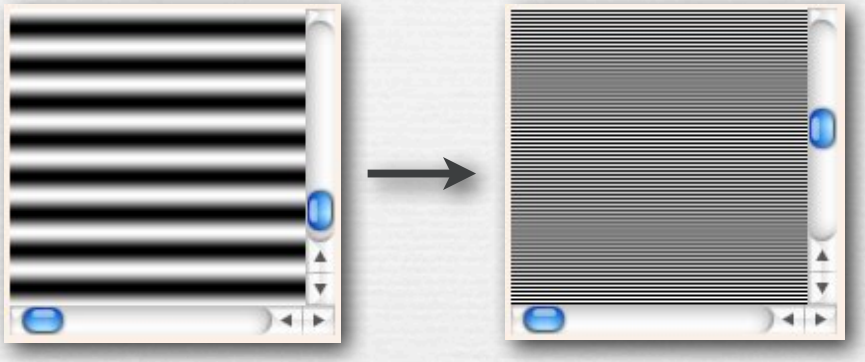
Questions?

Sampling and aliasing

abstract function



spatial aliasing in images

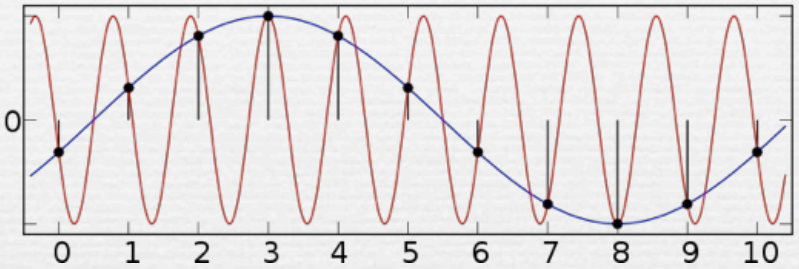


(http://ptolemy.eecs.berkeley.edu/eecs20/week13/moire.html)

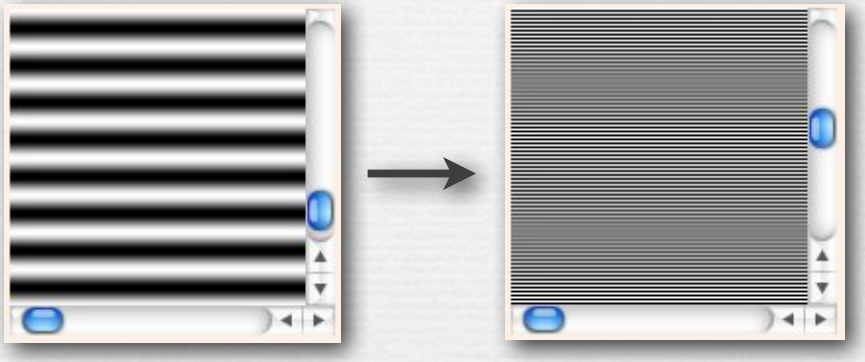
- aliasing is high frequencies masquerading as low frequencies due to insufficiently closely spaced samples

Sampling and aliasing

abstract function

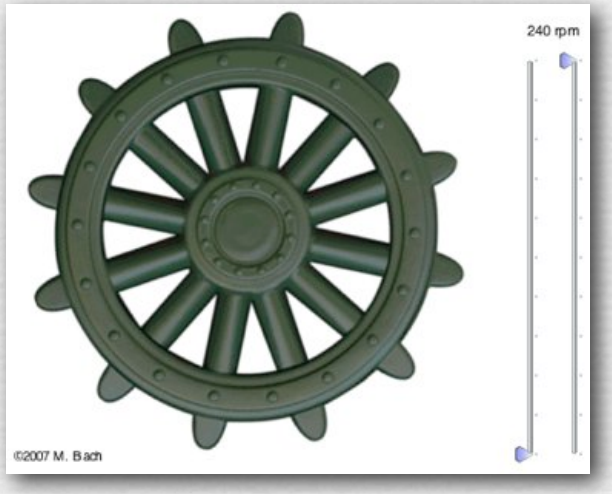


spatial aliasing in images



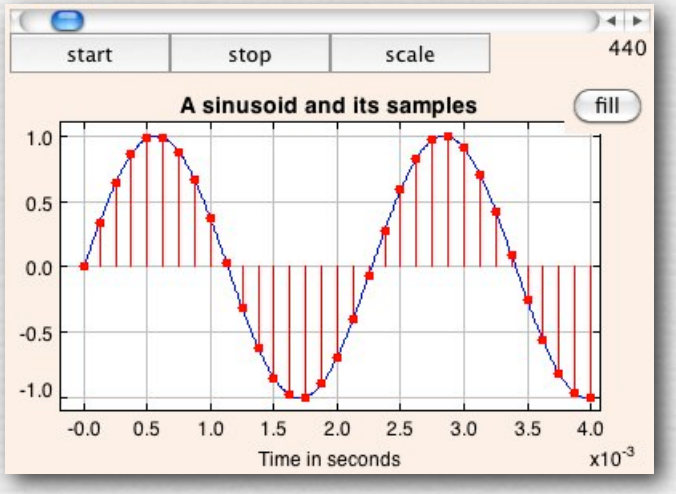
(http://ptolemy.eecs.berkeley.edu/eecs20/week13/moire.html)

temporal aliasing



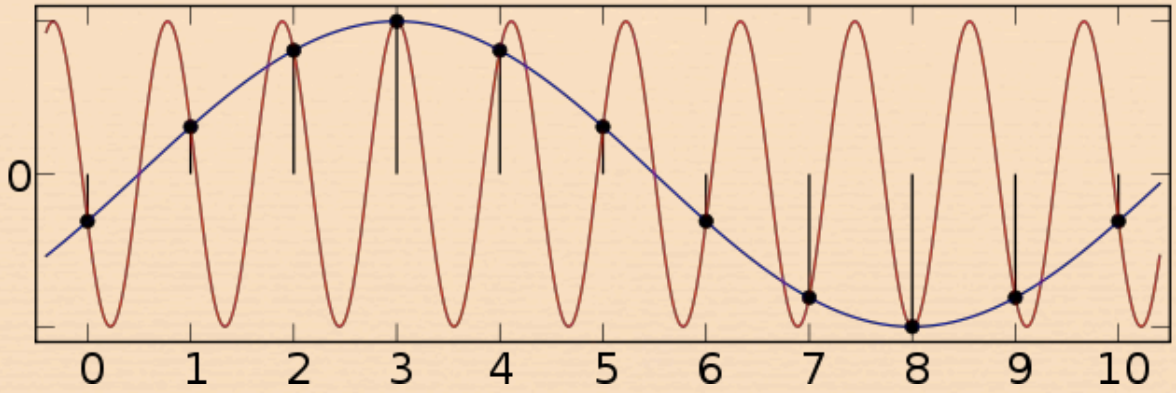
(http://www.michaelbach.de/ot/mot_wagonWheel/index.html)

temporal aliasing in audio



(http://ptolemy.eecs.berkeley.edu/eecs20/week13/aliasing.html)

Fourier analysis of aliasing



- Nyquist-Shannon sampling theorem: a function having frequencies no higher than *n* can be completely determined by samples spaced *1 / 2n* apart

$$f_{sampling} > 2 \times f_{cutoff}$$

Retinal sampling rate

- the human retina consists of discrete sensing cells
- therefore, the retina performs sampling
- sampling theory says $f_{sampling} > 2 \times f_{cutoff}$
- if observed human cutoff is 50 cycles per degree, then its sampling rate must be > 100 samples per degree
- this agrees with observed retinal cell spacing!

(Cornsweet)

spacing between L,M cone cells is 1μ ≈ 30 arc-seconds (1/120º)

Retinal sampling rate

- the human retina consists of discrete sensing cells
- therefore, the retina performs sampling
- sampling theory says $f_{sampling} > 2 \times f_{cutoff}$
- if observed human cutoff is 50 cycles per degree, then its sampling rate must be > 100 samples per degree
- this agrees with observed retinal cell spacing!

yes, almost equal to human acuity

- Example #3: iPhone 4 "Retina Display" viewed at 12" inches
  - 960 pixels on 2.94" high display
  - $1/\Delta x$ = 326 dpi
  - spatial frequency on retina = *34* cycles per degree

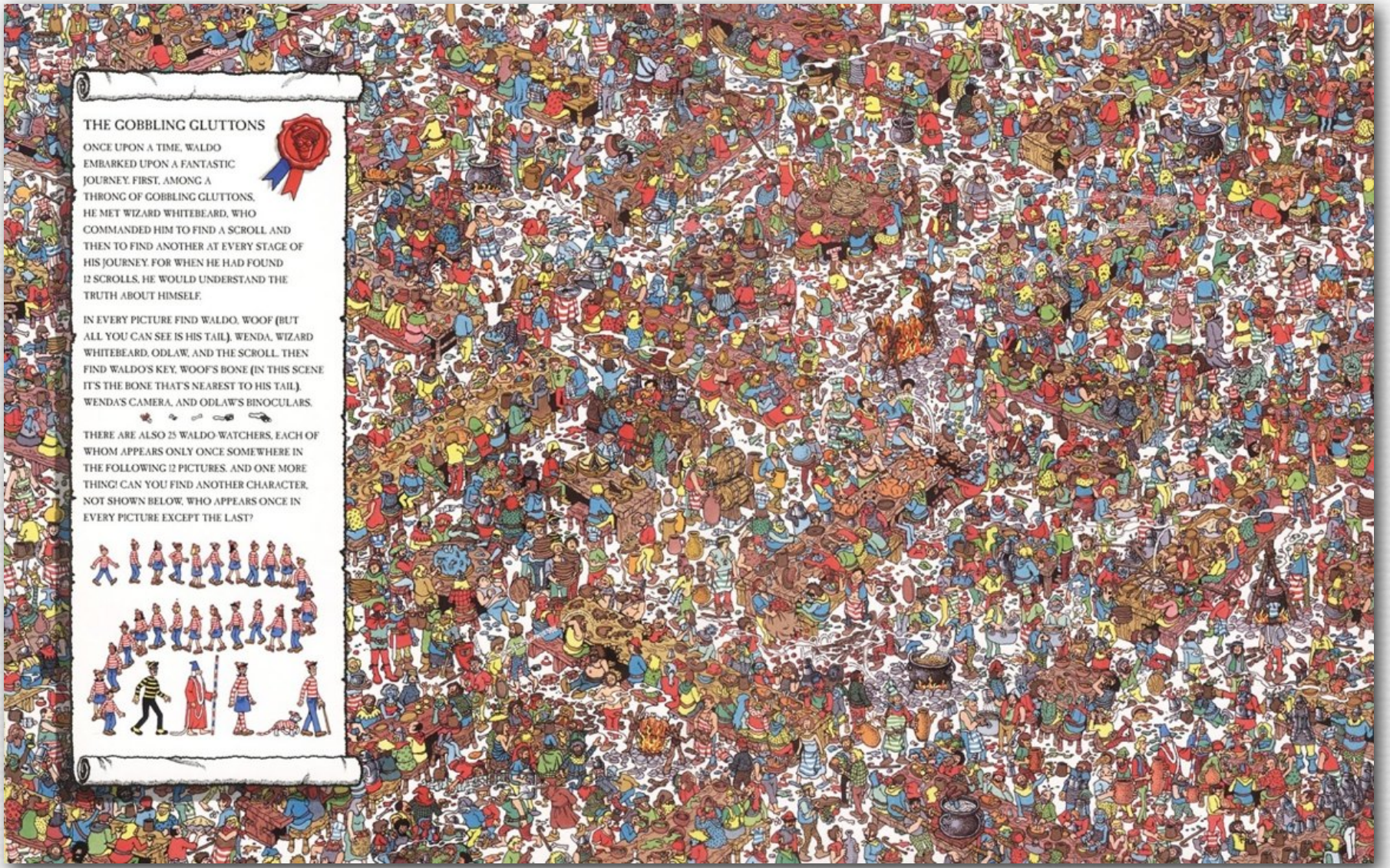
Vernier acuity (a.k.a. hyperacuity)

- we can detect jaggies as small as 5 seconds of retinal arc (1/720º)
  - even though our cells are spaced 30 seconds apart
  - to make such jaggies invisible, the iPhone display would need to be 5,760 pixels vertically !!

Aliasing in photography

- a lens creates a focused image on the sensor
- suppose the sensor measured this image at points on a 2D grid, but ignored the imagery between points?
  - a.k.a. *point sampling*

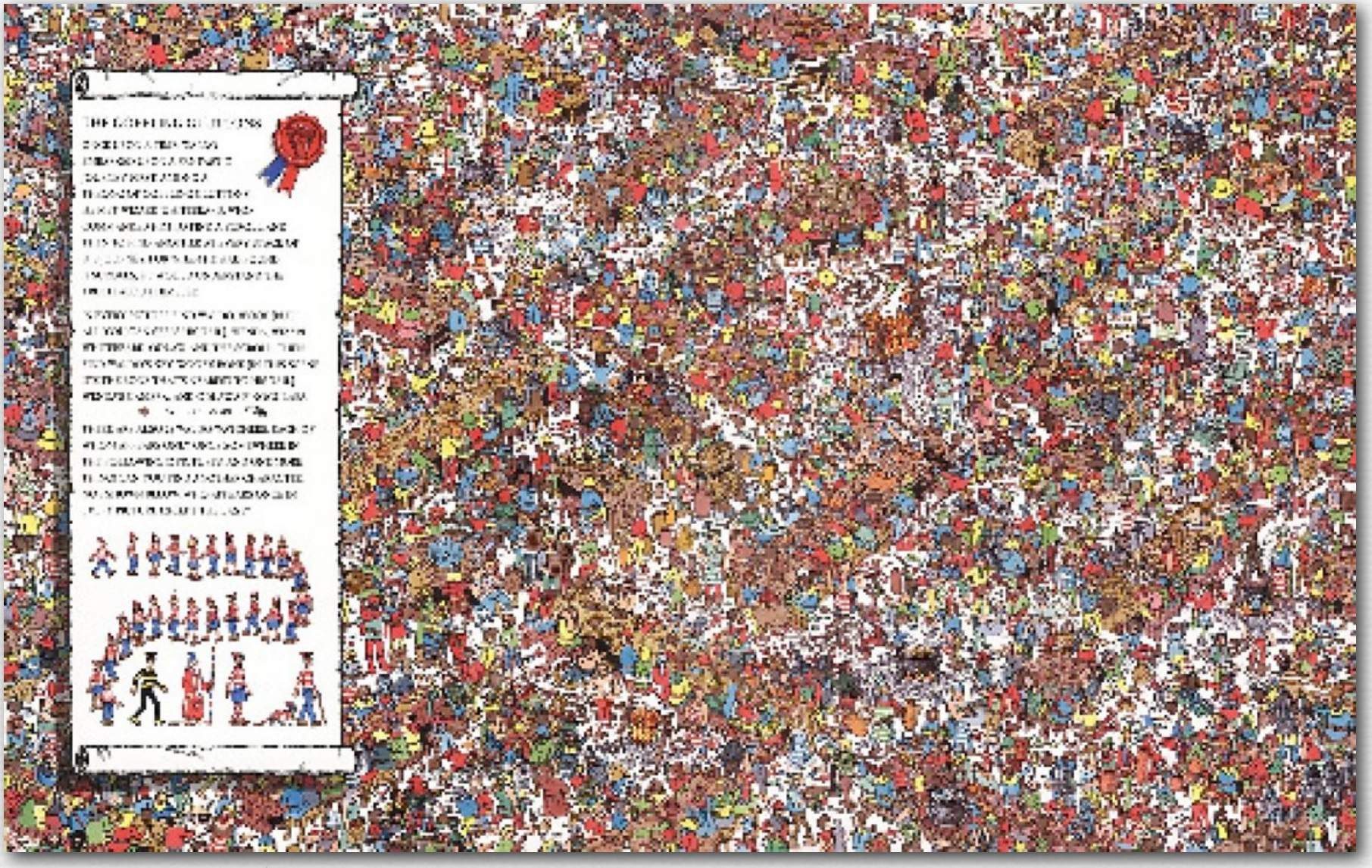
Simulation of point sampling



(Classic Media)

digital image, 1976 x 1240 pixels

Simulation of point sampling



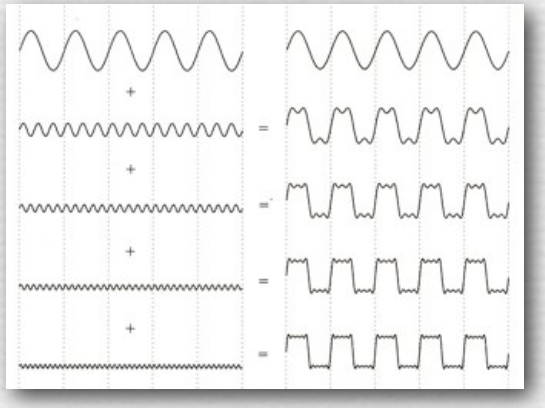
every 4th pixel in *x* and *y*, then upsized using pixel replication

Prefiltering to avoid aliasing

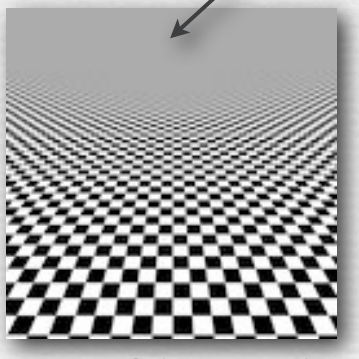
- before sampling, remove (or at least attenuate) sine waves of frequency greater than half the sampling rate

$$f_{cutoff} < \frac{1}{2} f_{sampling}$$

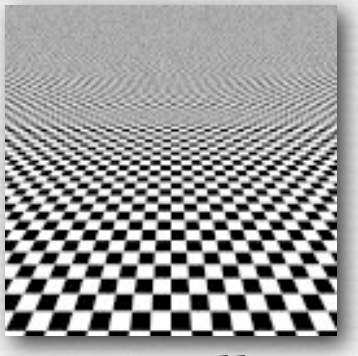
replace removed waves with their average intensity (gray in this case)



unfiltered



prefiltered



partially pre-filtered

Methods for prefiltering

- method #1: frequency domain
  1. convert image to frequency domain
  2. *multiply* by low-pass filter (removes frequencies above f_{cutoff})
  3. convert back to spatial domain
  4. perform point sampling as before

  - conversions are slow
  - not clear how to apply this method to images in a camera

- method #2: spatial domain
  1. *convolve* image by low-pass filter
  2. perform point sampling as before

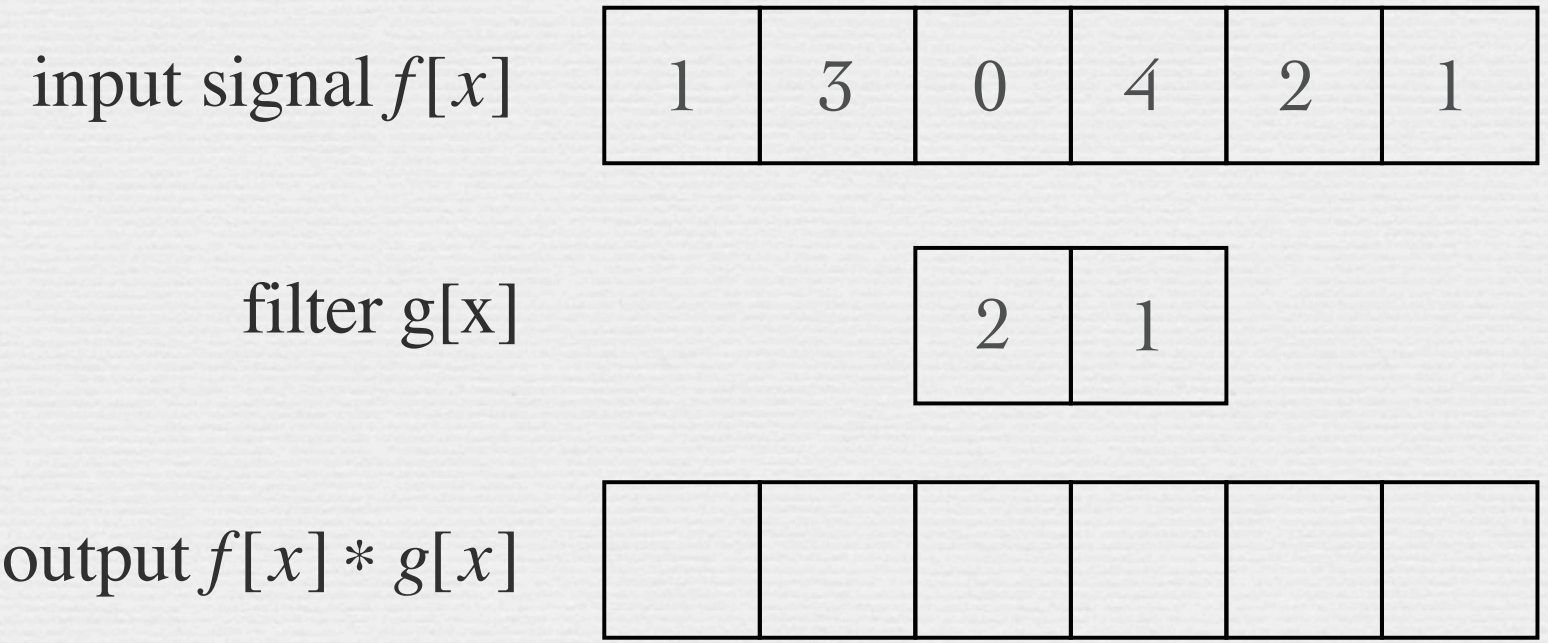
  - direct and faster
  - equivalent to method #1 (proof is beyond scope of this course)

filters for use in frequency versus spatial domains are related but are not the same

Discrete convolution in 1D

- replace each input value with a weighted sum of itself and its neighbors, with weights given by a filter function

$$f[x] * g[x] \quad = \quad \sum_{k=-\infty}^{\infty} f[k] \cdot g[x-k]$$



Discrete convolution in 1D

- replace each input value with a weighted sum of itself and its neighbors, with weights given by a filter function

$$f[x] * g[x] = \sum_{k=-\infty}^{\infty} f[k] \cdot g[x-k]$$

input signal $f[x]$

1	3	0	4	2	1

1	2

notice that the filter gets flipped when applied

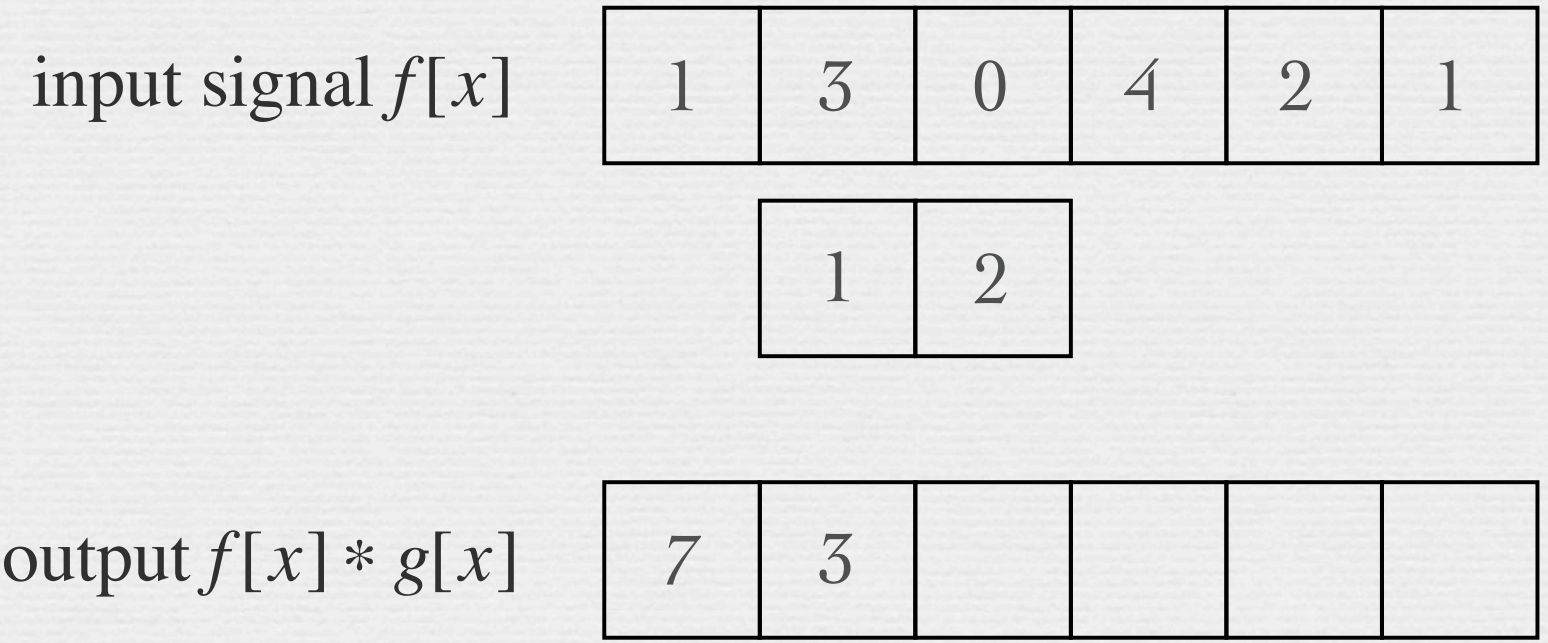
output $f[x] * g[x]$

7					

Discrete convolution in 1D

- replace each input value with a weighted sum of itself and its neighbors, with weights given by a filter function

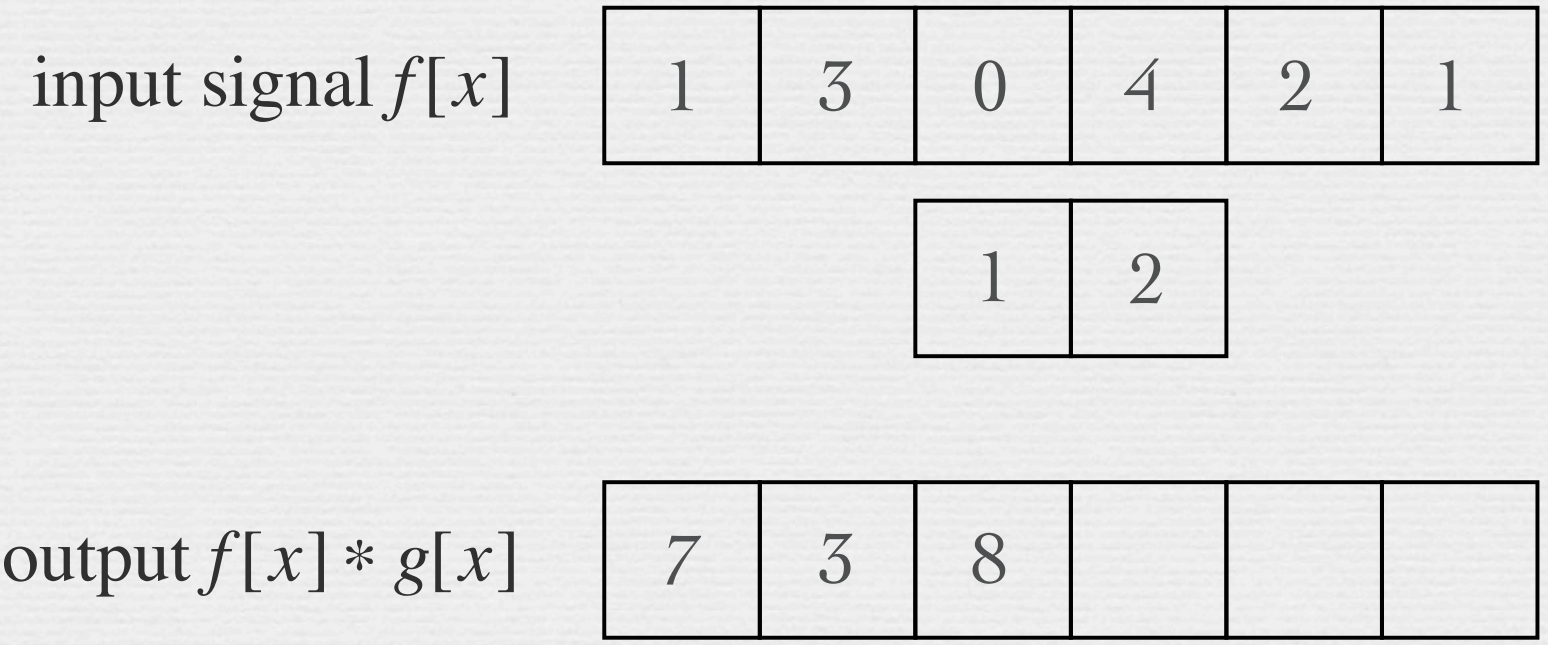
$$f[x] * g[x] \quad = \quad \sum_{k=-\infty}^{\infty} f[k] \cdot g[x-k]$$



Discrete convolution in 1D

- replace each input value with a weighted sum of itself and its neighbors, with weights given by a filter function

$$f[x] * g[x] = \sum_{k=-\infty}^{\infty} f[k] \cdot g[x-k]$$



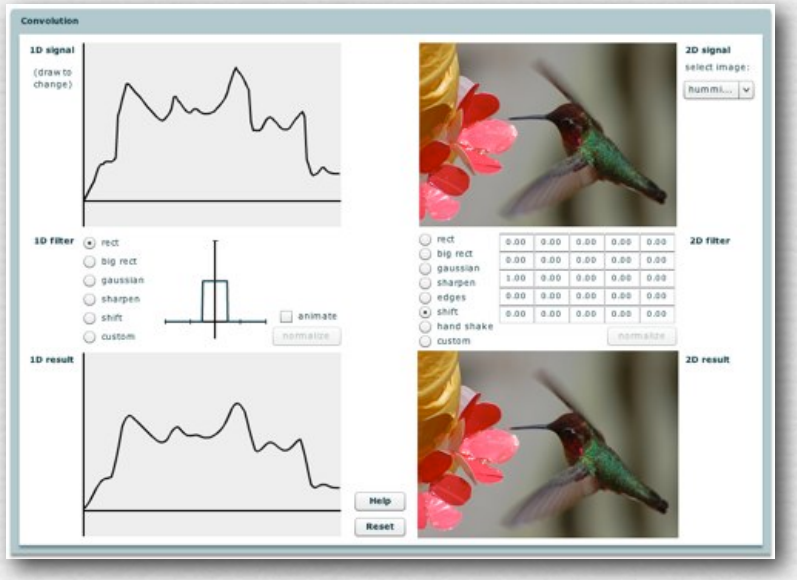
More convolution formulae

- 1D discrete: defined only on the integers

$$f[x] * g[x] = \sum_{k=-\infty}^{\infty} f[k] \cdot g[x-k]$$

- 1D continuous: defined on the real line

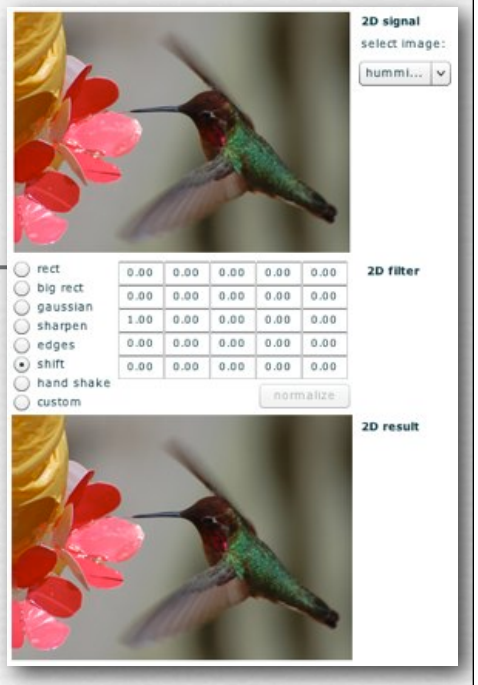
$$f(x) * g(x) = \int_{-\infty}^{\infty} f(\tau) \cdot g(x-\tau)\, d\tau$$



(FLASH DEMO)

http://graphics.stanford.edu/courses/cs178/applets/convolution.html

More convolution formulae



- 1D discrete: defined only on the integers

$$f[x] * g[x] = \sum_{k=-\infty}^{\infty} f[k] \cdot g[x-k]$$

- 1D continuous: defined on the real line

$$f(x) * g(x) = \int_{-\infty}^{\infty} f(\tau) \cdot g(x-\tau)\, d\tau$$

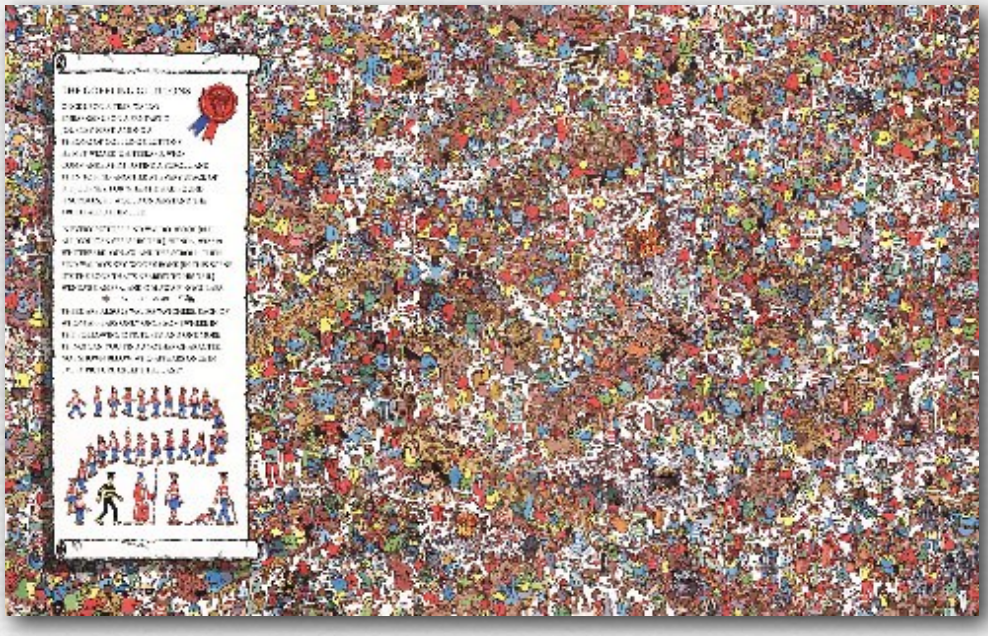
- 2D discrete: defined on the x, y integer grid

$$f[x,y] * g[x,y] = \sum_{i=-\infty}^{\infty} \sum_{j=-\infty}^{\infty} f[i,j] \cdot g[x-i, y-j]$$

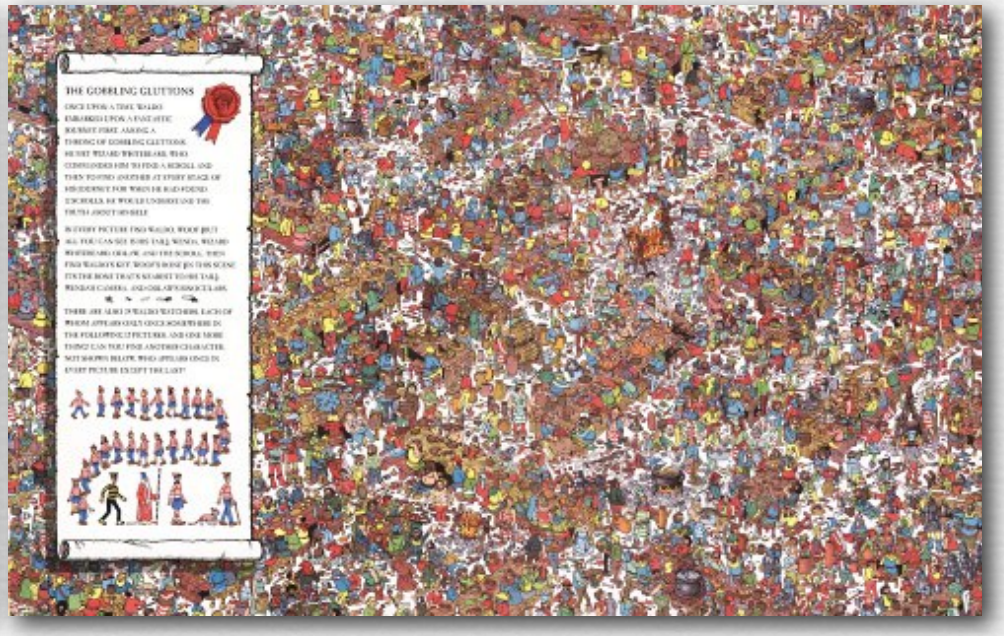
- 2D continuous: defined on the x,y plane

$$f(x,y) * g(x,y) = \int_{\tau_1=-\infty}^{\infty} \int_{\tau_2=-\infty}^{\infty} f(\tau_1,\tau_2) \cdot g(x-\tau_1, y-\tau_2)\, d\tau_1\, d\tau_2$$

Prefiltering reduces aliasing



every 4th pixel in x and y

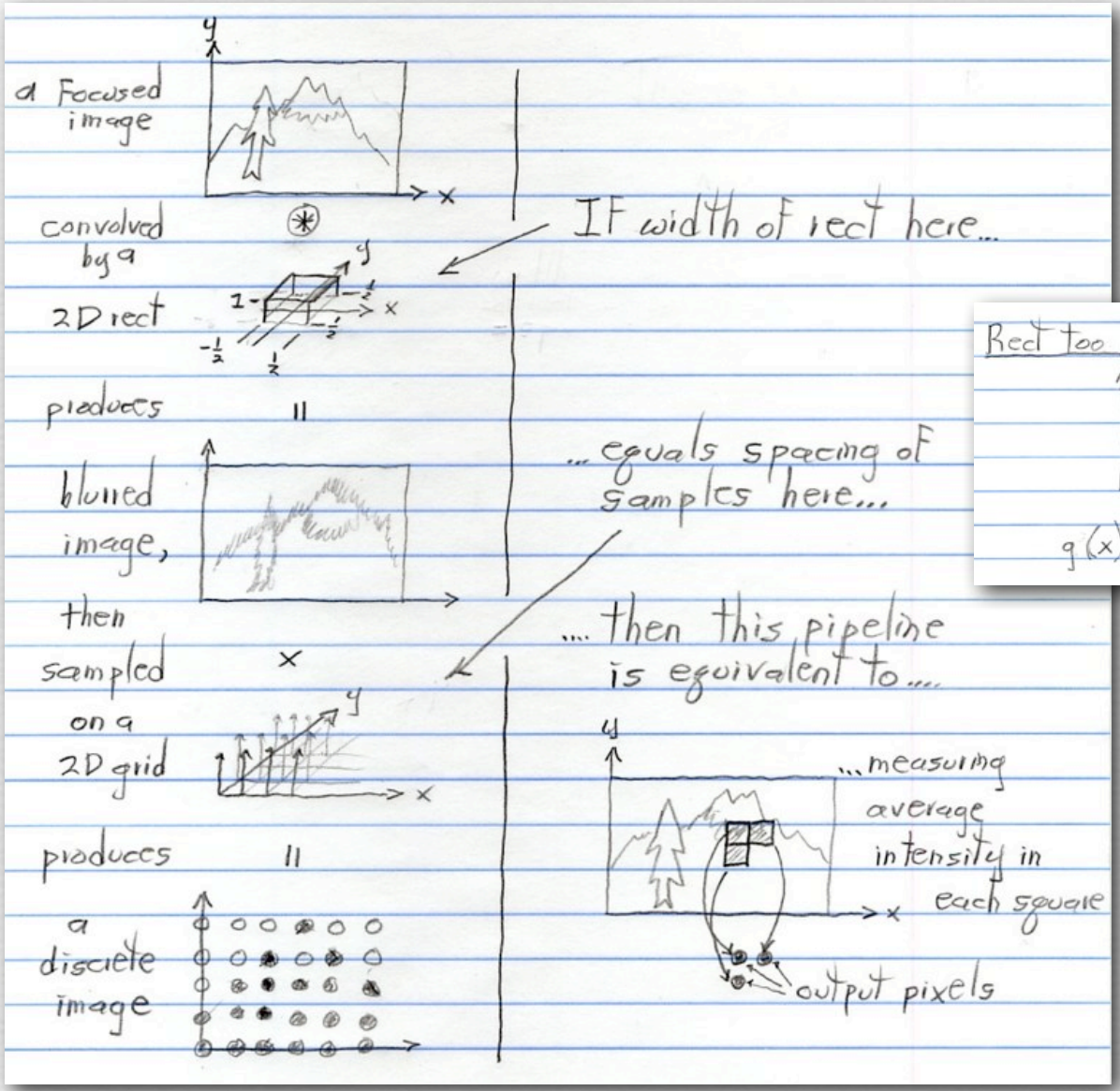


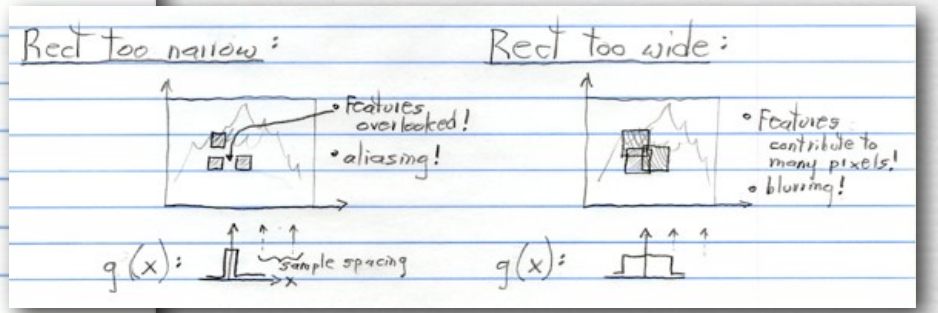
convolved by 4×4 pixel rect,
then sampled every 4th pixel

Prefiltering & sampling in photography

- photography consists of convolving the focused image by a 2D rect filter, then sampling on a 2D grid
  - each point on this grid is called a *pixel*
- if convolution is followed by sampling, you only need to compute the convolution at the sample positions
  - for a rect filter of width equal to the sample spacing, this is equivalent to measuring the average intensity of the focused image in a grid of abutting squares
  - this is exactly what a digital camera does
- the width of the rect is typically equal to the spacing between sample positions
  - narrower leaves aliasing; wider produces excessive blur

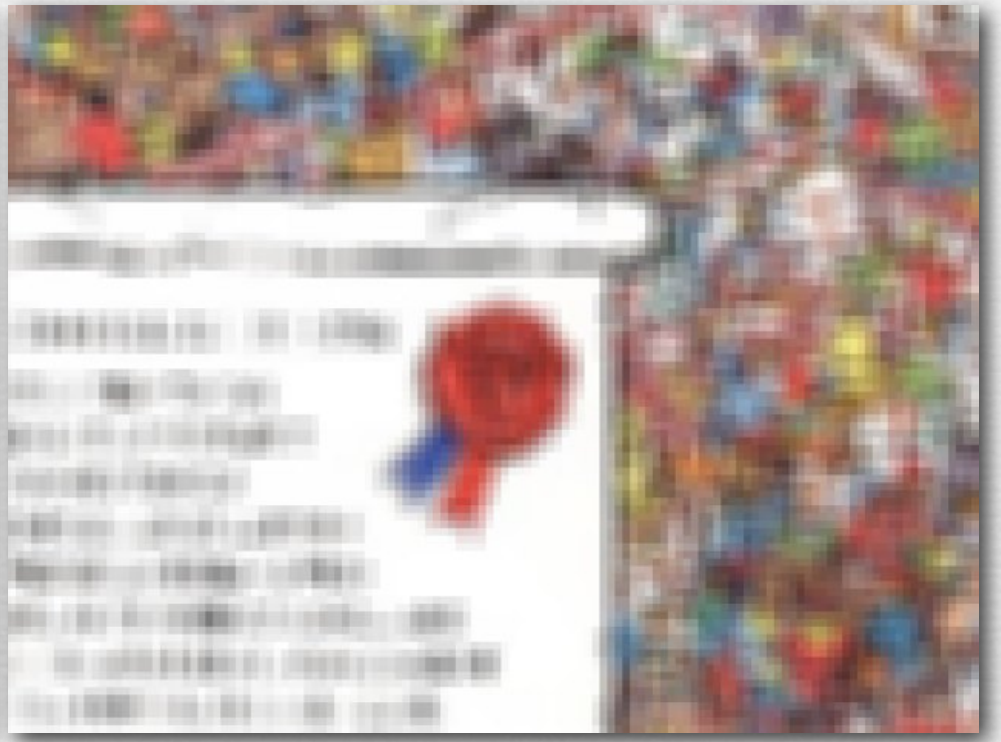
Prefiltering & sampling in photography (contents of whiteboard)





Prefiltering versus postfiltering

convolved by 4×4 pixel rect,
then sampled every 4th pixel



point sampled every 4th pixel,
then convolved by 4×4 pixel rect

- aliasing is high frequencies masquerading as low frequencies;
  it cannot be removed by postfiltering without destroying the image

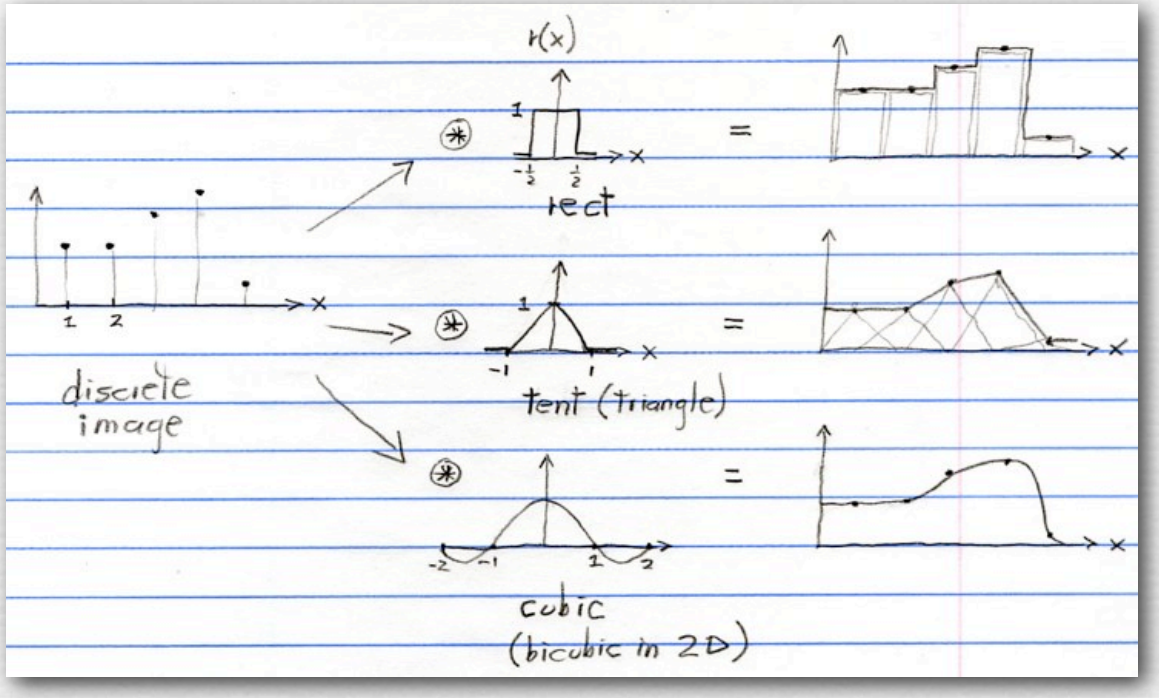
Upsizing/downsizing in Photoshop

- *reconstruction* is the conversion of a discrete (i.e. sampled) image into a continuous image by *interpolation* between the available samples
  - interpolation can be modeled as (another) convolution

- *resampling* is the conversion of a discrete image into a second discrete image having more or fewer samples
  1. interpolate between samples to create a continuous image
  2. prefilter to remove frequencies that would otherwise alias
  3. point sample at the new rate

  - these steps can be simplified into a single discrete convolution

Interpolation via convolution (contents of whiteboard)



- if the input is a discrete (i.e. sampled) function, then convolution can be treated as placing an vertically-scaled copy of the filter *r(x)* at each sample position as shown, summing the results, and dividing by the area under the filter (1.0 in the cases shown)
- the effect is to interpolate between the samples, hence reconstructing a continuous function from the discrete function

Recap

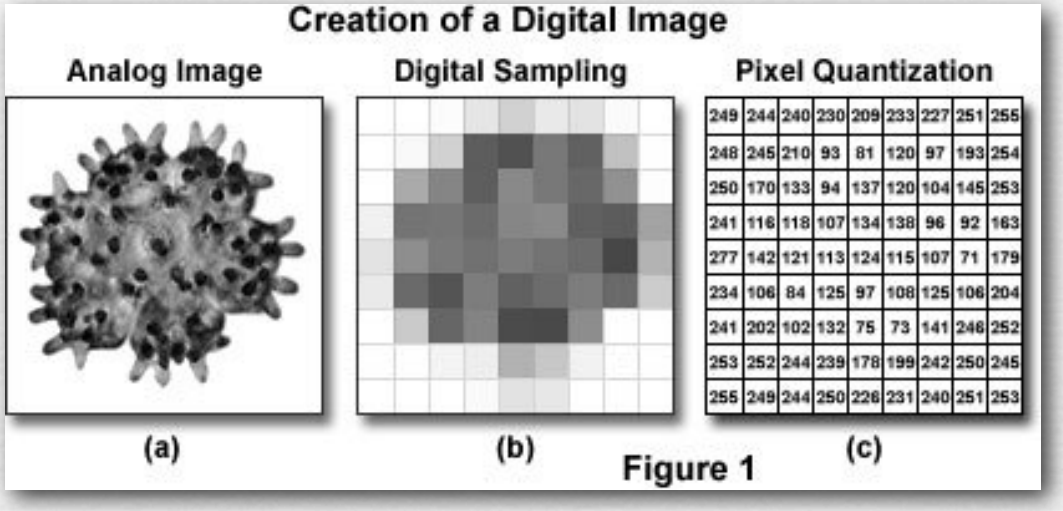
- *aliasing* is high frequencies masquerading as low frequencies due to insufficiently closely spaced samples

- reduce aliasing by prefiltering the input before sampling
  - implement by multiplication in the frequency domain
  - or convolution in the spatial domain
  - in the spatial domain, the prefilter is denoted *g(x)*
  - cannot remove aliasing by postfiltering, i.e. filtering after sampling

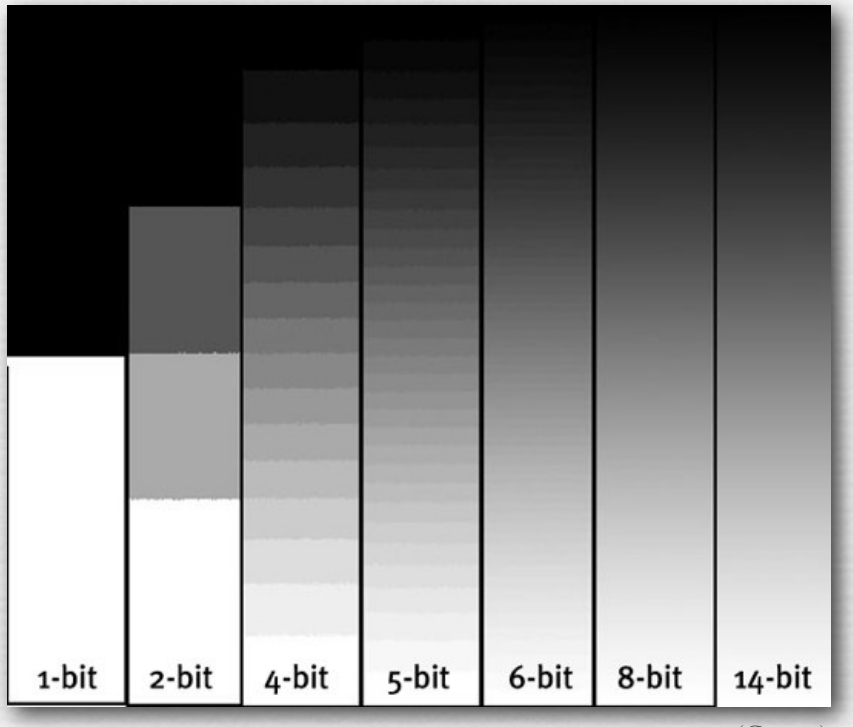
- in digital photography:
  - *g(x)* is a pixel-sized rect, thus averaging intensity over areas
  - if the rect is too small, aliasing occurs; solve with antialiasing filter

Questions?

Sampling versus quantization



(http://learn.hamamatsu.com/articles/digitalimagebasics.html)



(Canon)

- an image is a function $\vec{f}(\vec{x})$
  - typically $(\vec{x}) = (x,y)$ and $\vec{f} = (R,G,B)$
- we sample the domain $(\vec{x})$ of this function as pixels
- we quantize the range $\vec{f}$ of this function as intensity levels

Example

8 bits × R,G,B =
24 bits per pixel

Canon 1D III,
300mm, f/3.2

Example

8 bits × R,G,B =
24 bits per pixel

Example

6 bits × R,G,B =
18 bits per pixel

Example

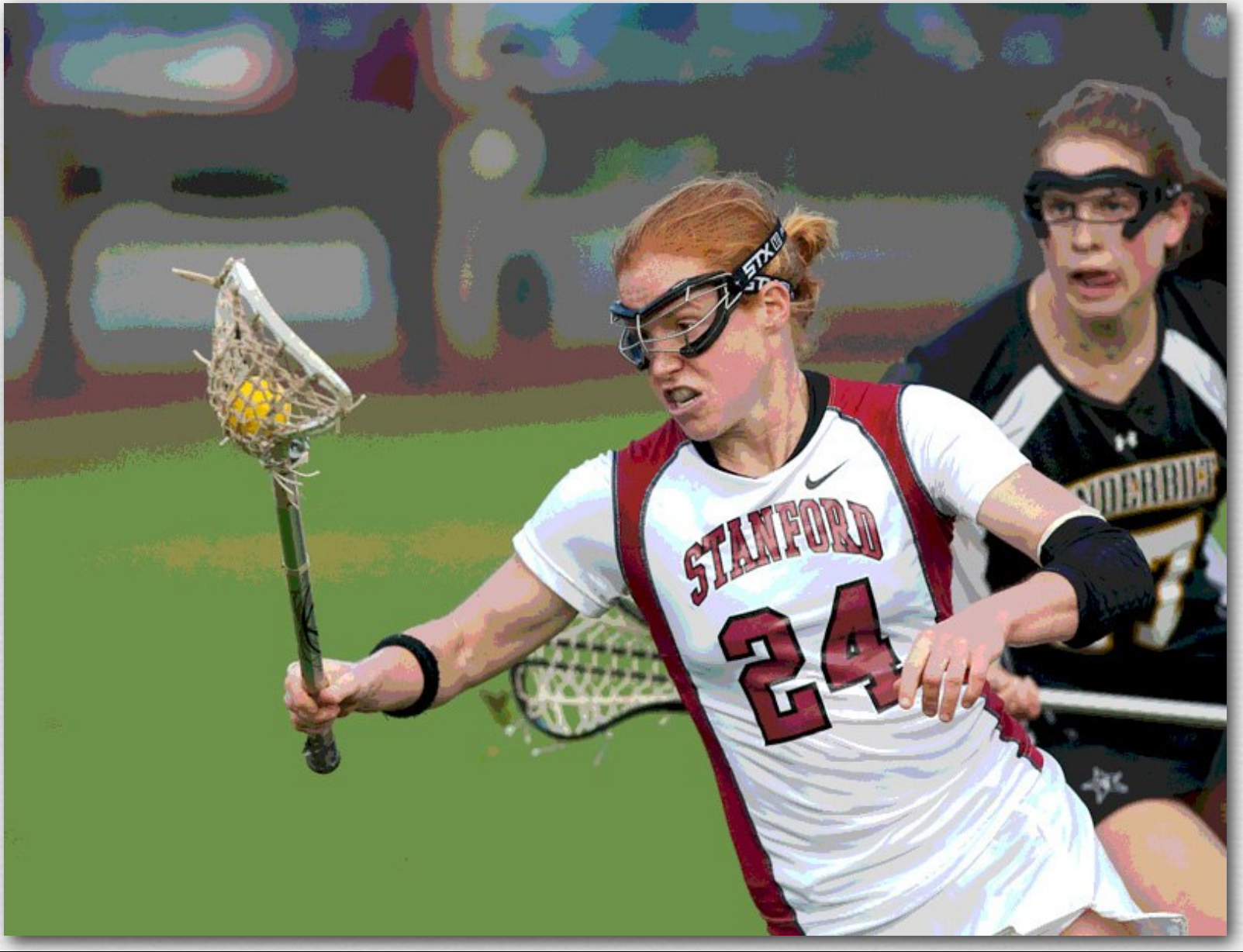
5 bits × R,G,B =
15 bits per pixel

Example

4 bits × R,G,B =
12 bits per pixel

Example

3 bits × R,G,B =
9 bits per pixel



Example

256 colors (8 bits) uniformly distributed across RGB cube, patterned dithering in Photoshop

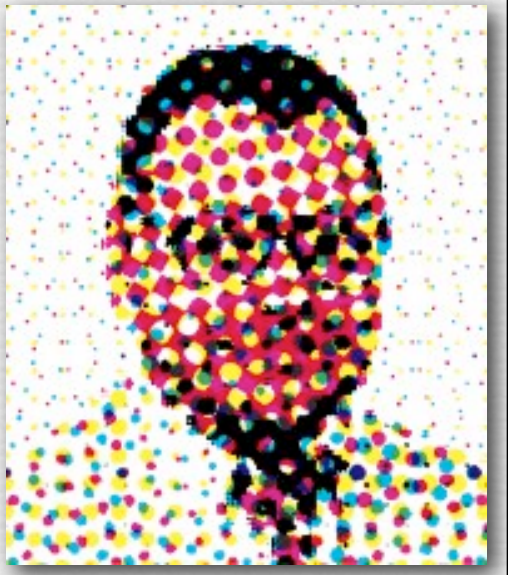
Example

256 colors (8 bits) adaptively distributed across RGB cube, patterned dithering in Photoshop

Dithering versus halftoning

- *dithering* for display (on a screen)
  - palette of a few hundred colors (uniform or adaptive)
  - flip some pixels in each neighborhood to the next available color in the palette to approximate intermediate colors when viewed from a distance

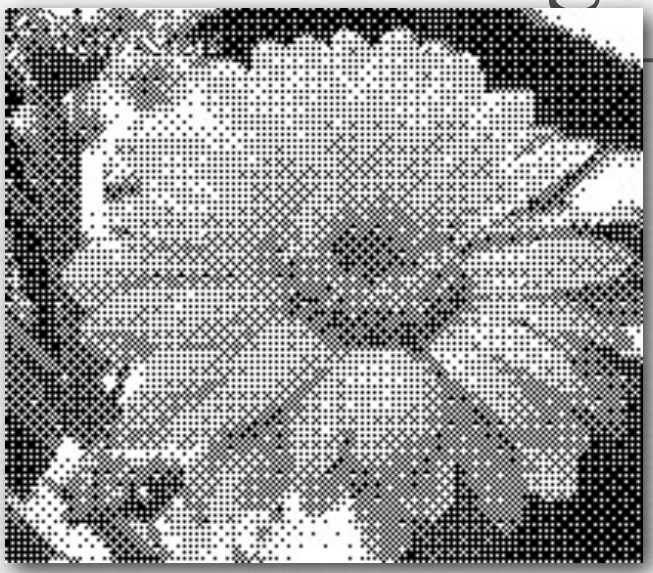
- *halftoning* for printing (on paper)
  - palette of only 3 or 4 primary colors
  - print each primary as a grid of dots, superimposed but slightly offset from the other primaries, and vary dot size locally to approximate intermediate colors



(wikimedia)

- both techniques are applicable to full-color or black and white imagery
- both trade off spatial resolution to obtain more colors, hence to avoid quantization (contouring)

Dithering versus halftoning

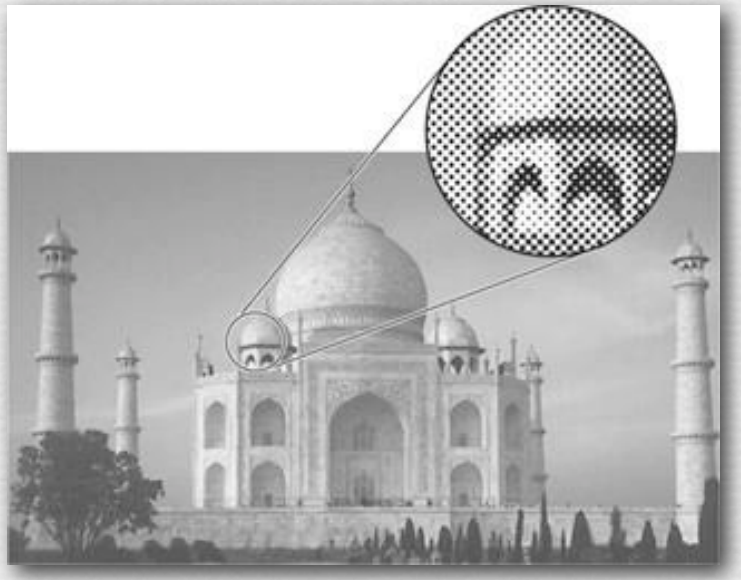


binary dithering

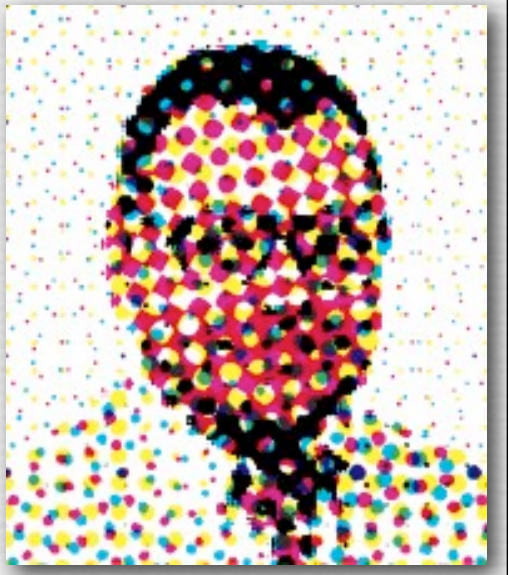
grayscale dithering

color dithering

(see http://bisqwit.iki.fi/jutut/colorquant/ for more examples)



grayscale halftoning



color halftoning

Recap

- *sampling* describes where in its domain you measure a function
  - for uniformly spaced samples, you can specify a *sampling rate*
  - if the sampling rate is too low, you might suffer from *aliasing*
  - you can reduce aliasing by *prefiltering*

- *quantization* describes how you represent these measurements
  - for uniformly spaced levels, you can specify a *bit depth*
  - if the bit depth is too low, you might suffer from *contouring*
  - you can reduce contouring by *dithering* (if displaying the image on a screen) or *halftoning* (if printing it on paper)

Questions?

Slide credits

- Pat Hanrahan

- Cornsweet, T.N., *Visual Perception,* Kluwer Academic Press, 1970.