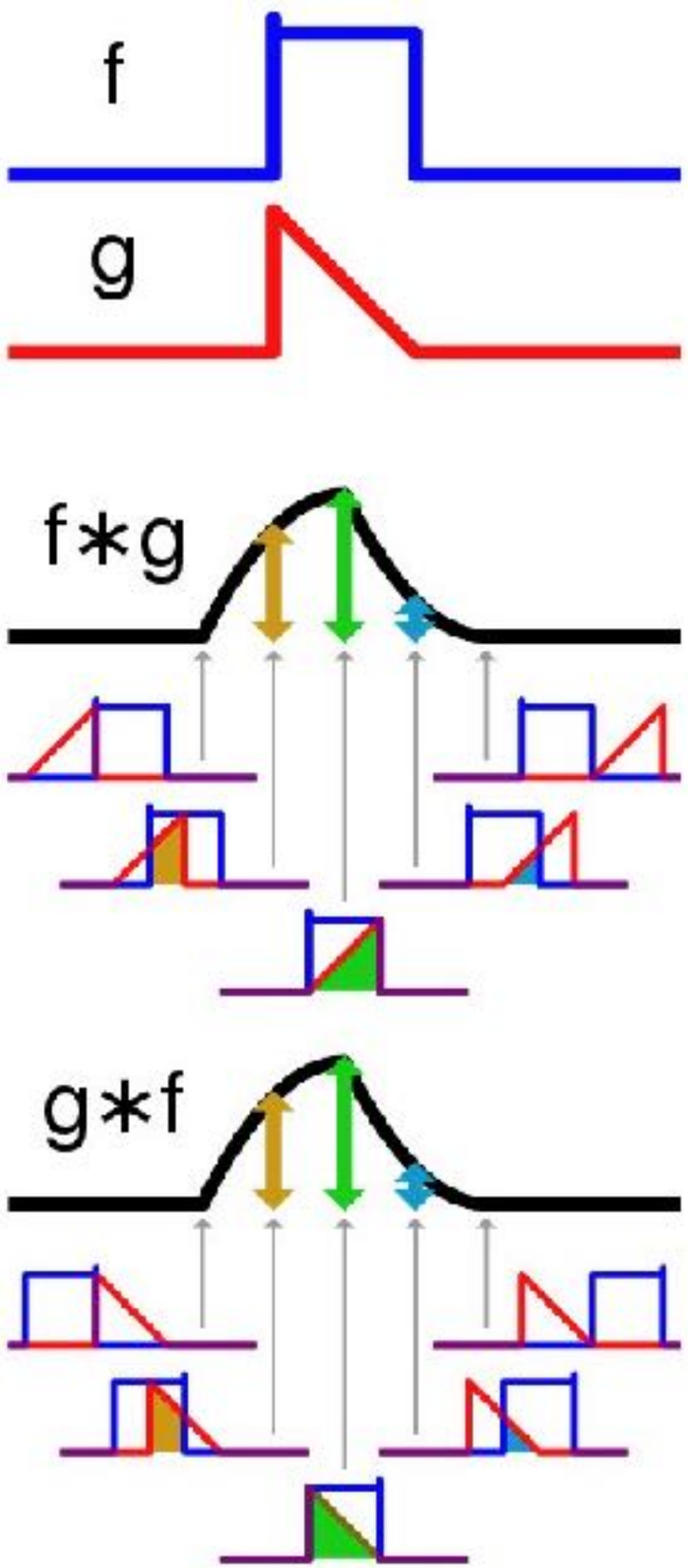
Image and Signal Processing

TechX CV — Lecture 2

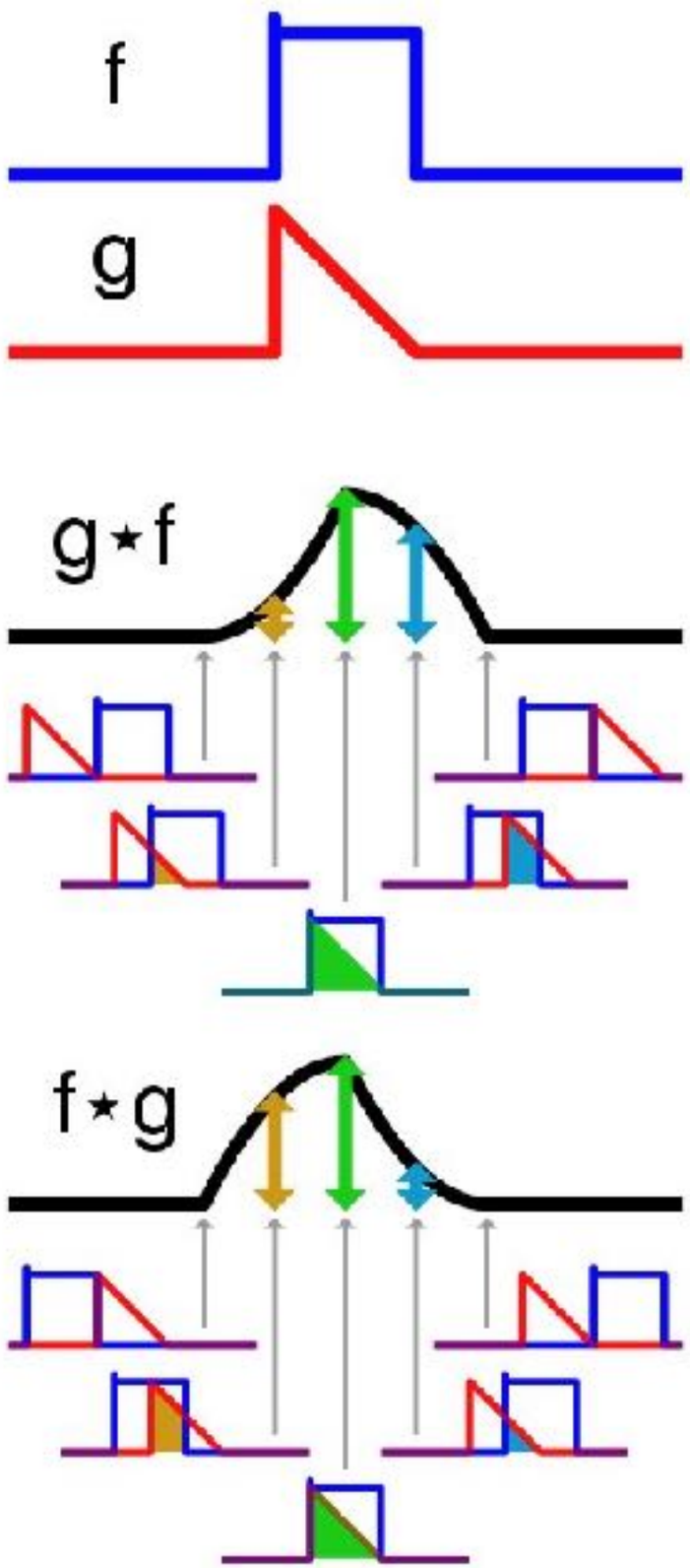
Cecilia Zhang

Recap

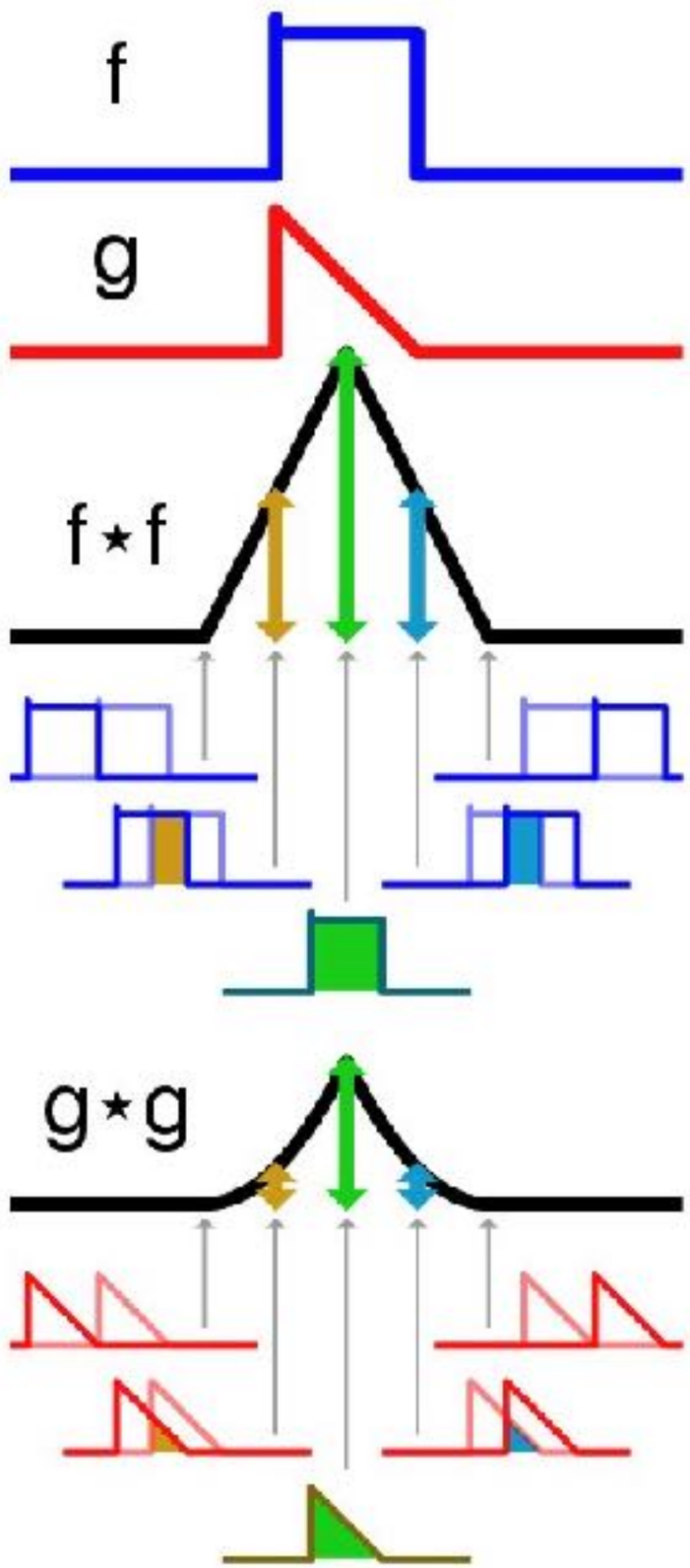
Convolution



Cross-correlation



Autocorrelation



Review Questions

1. Write down a 3x3 filter that returns a positive value if the average value of the 4-adjacent neighbors is less than the center and a negative value otherwise

2. Write down a filter that will compute the gradient in the x-direction:

```
gradx(y,x) = im(y,x+1)-im(y,x) for each x, y
```

Recap

- Filtering — Convolution
- Box filter
- Gaussian filter
- Impact of scale / width of smoothing filter
- Separable filters more efficient
- Template matching
- Normalized cross-correlation
- Pyramid — Gaussian and Laplacian
- Image blending

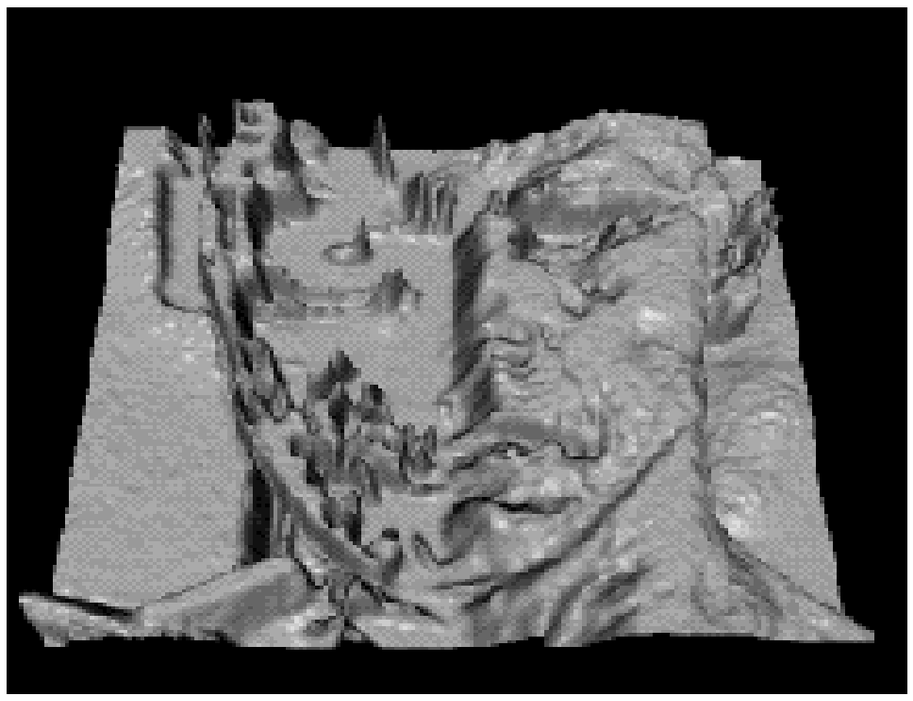
Image Representations

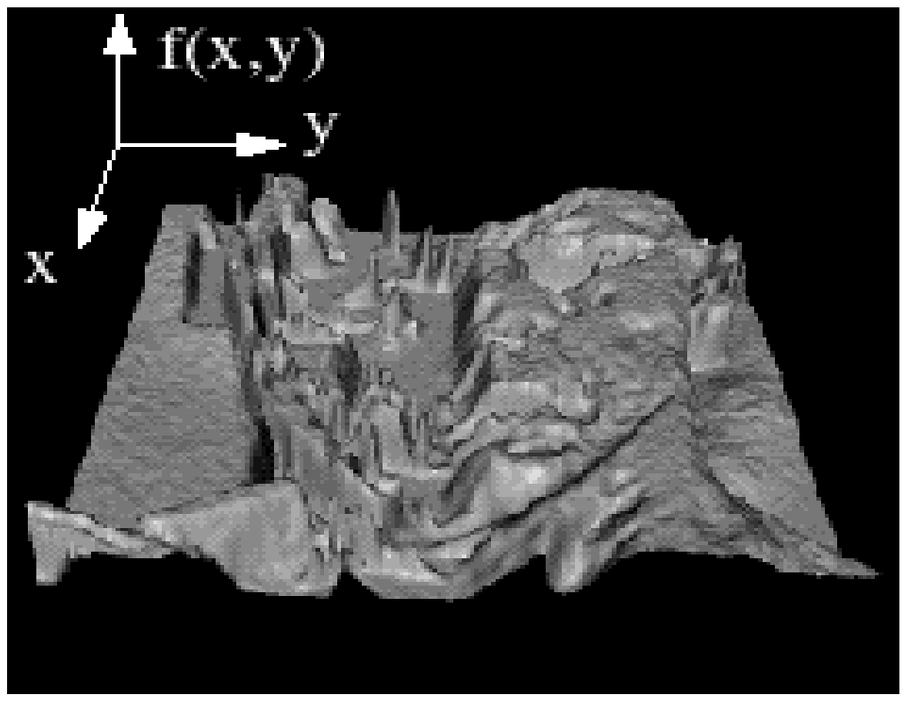
- **Pixels**: great for spatial resolution, poor access to frequency

- **Pyramids/filter banks**: balance between spatial and frequency information

- Today — **Fourier transform**: great for frequency, not for spatial info

High frequency





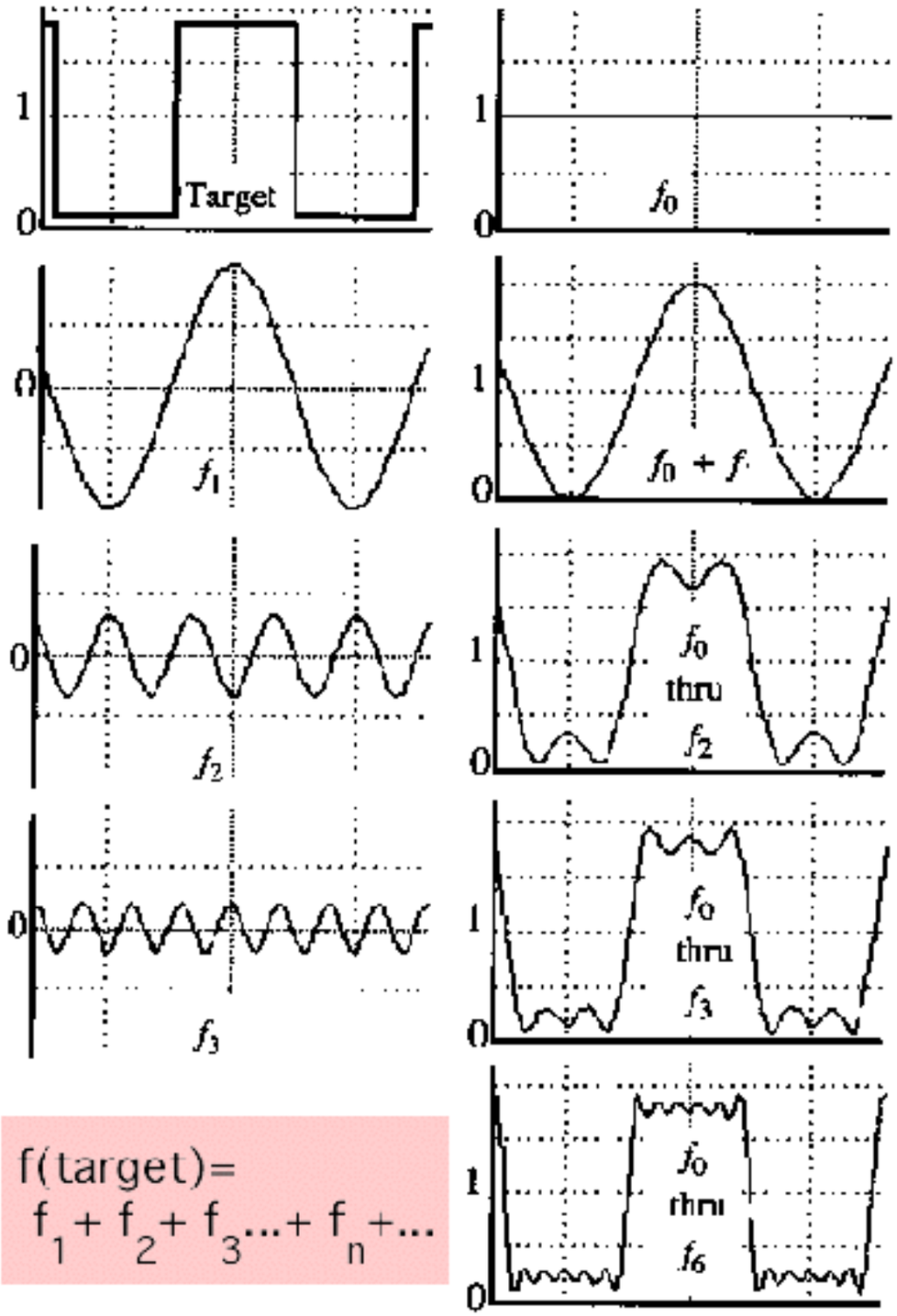
Think in terms of Frequency

A sum of sines

Our building block:

$$A\sin(\omega x + \phi)$$

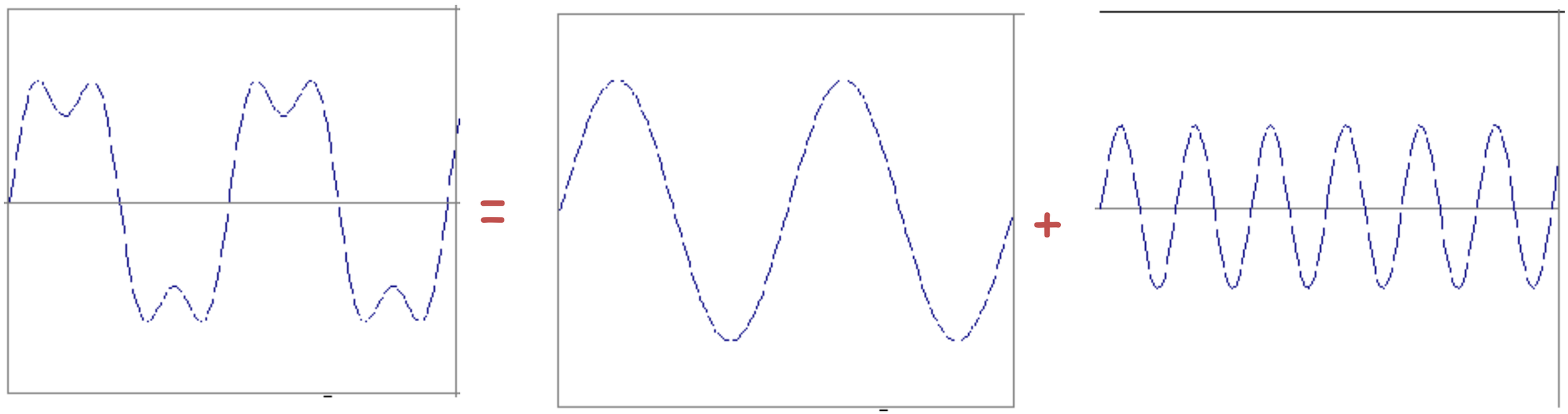
Add enough of them to get any signal f(x) you want!

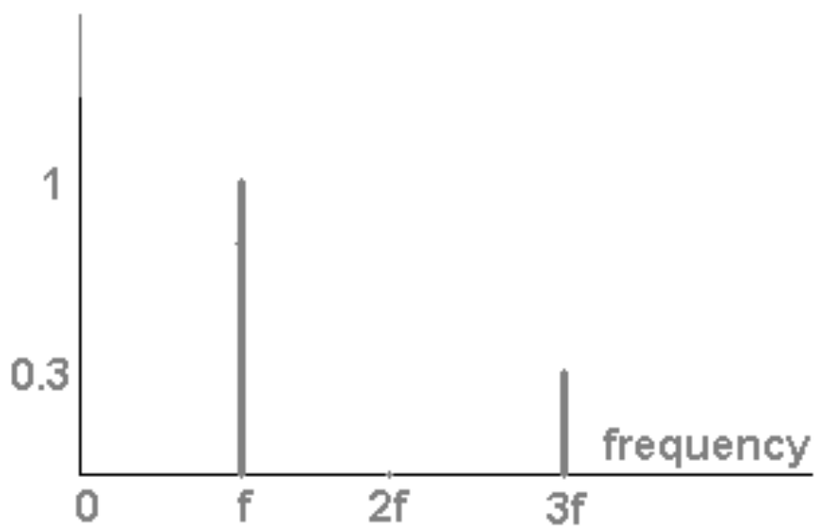


f(target)=
$f_1 + f_2 + f_3 ... + f_n + ...$

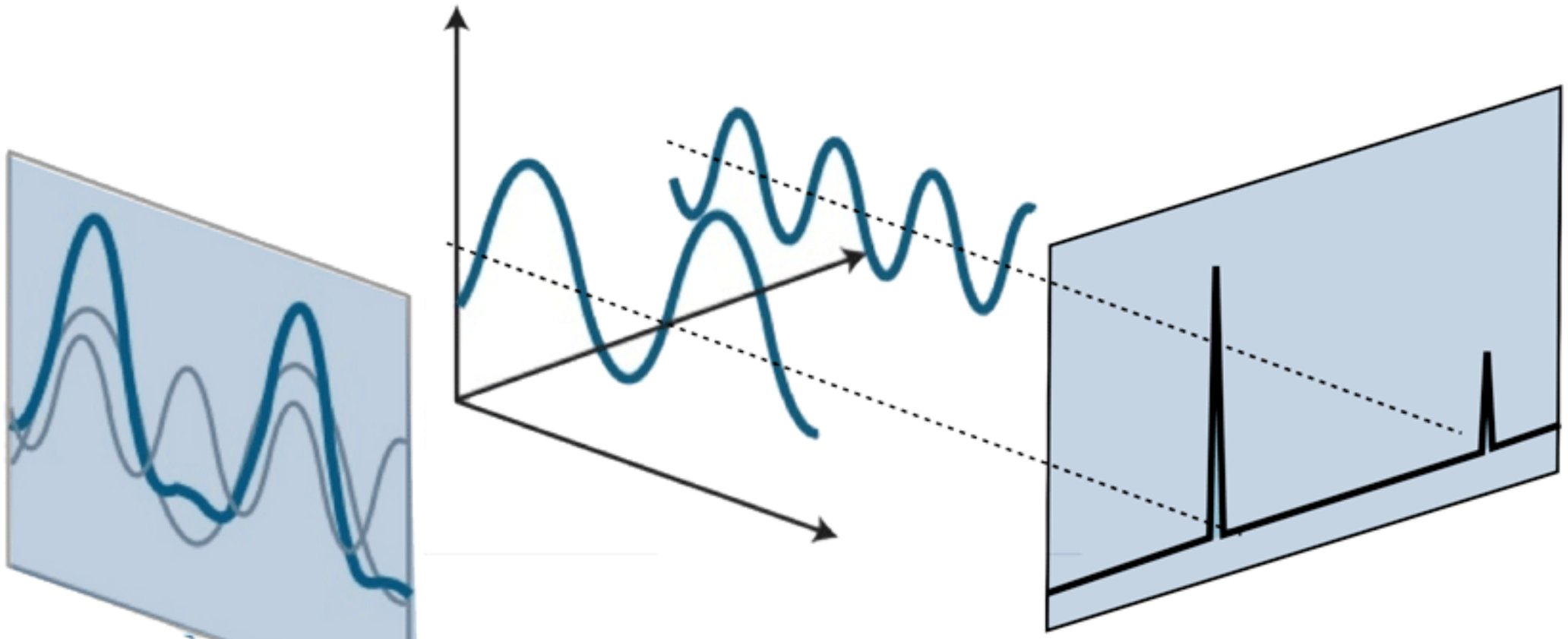
1D Frequency Spectra

- example : $g(t) = \sin(2\pi f t) + (1/3)\sin(2\pi(3f)\, t)$





1D Frequency Spectra

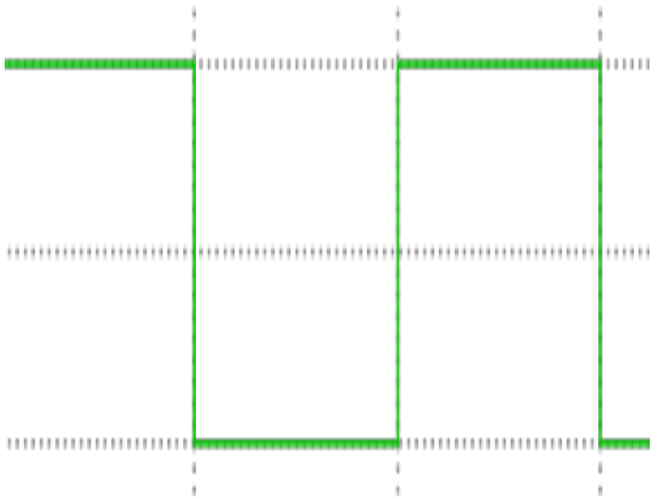


Time Domain
s(t)

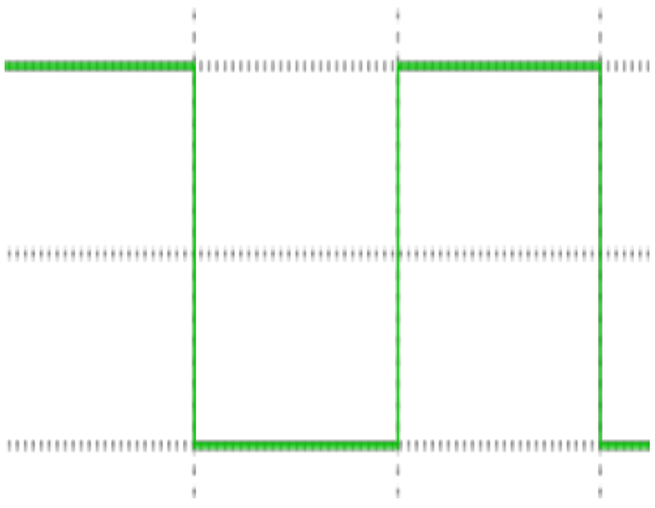
FT

Frequency Domain
S(ω)

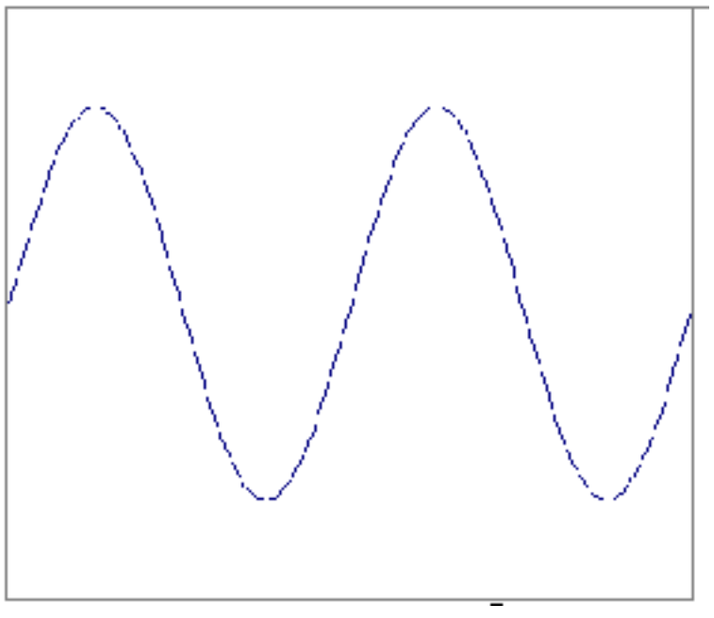
Frequency Spectra



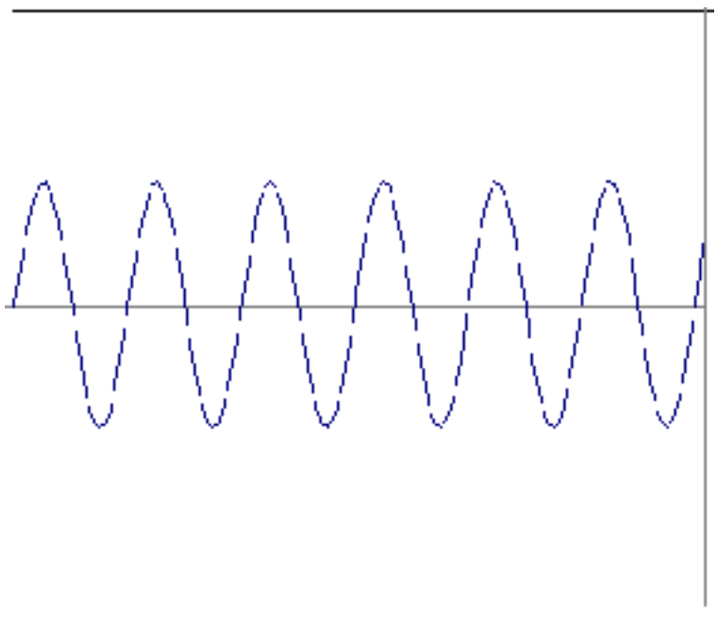
Frequency Spectra



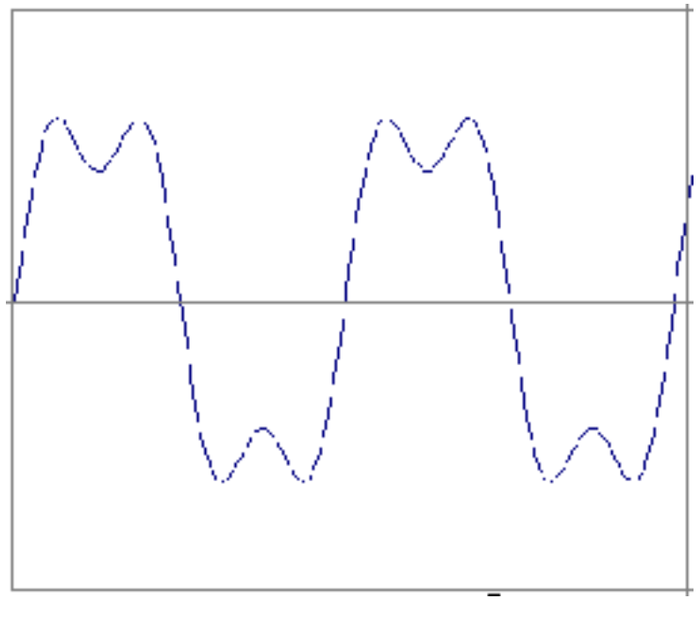
=



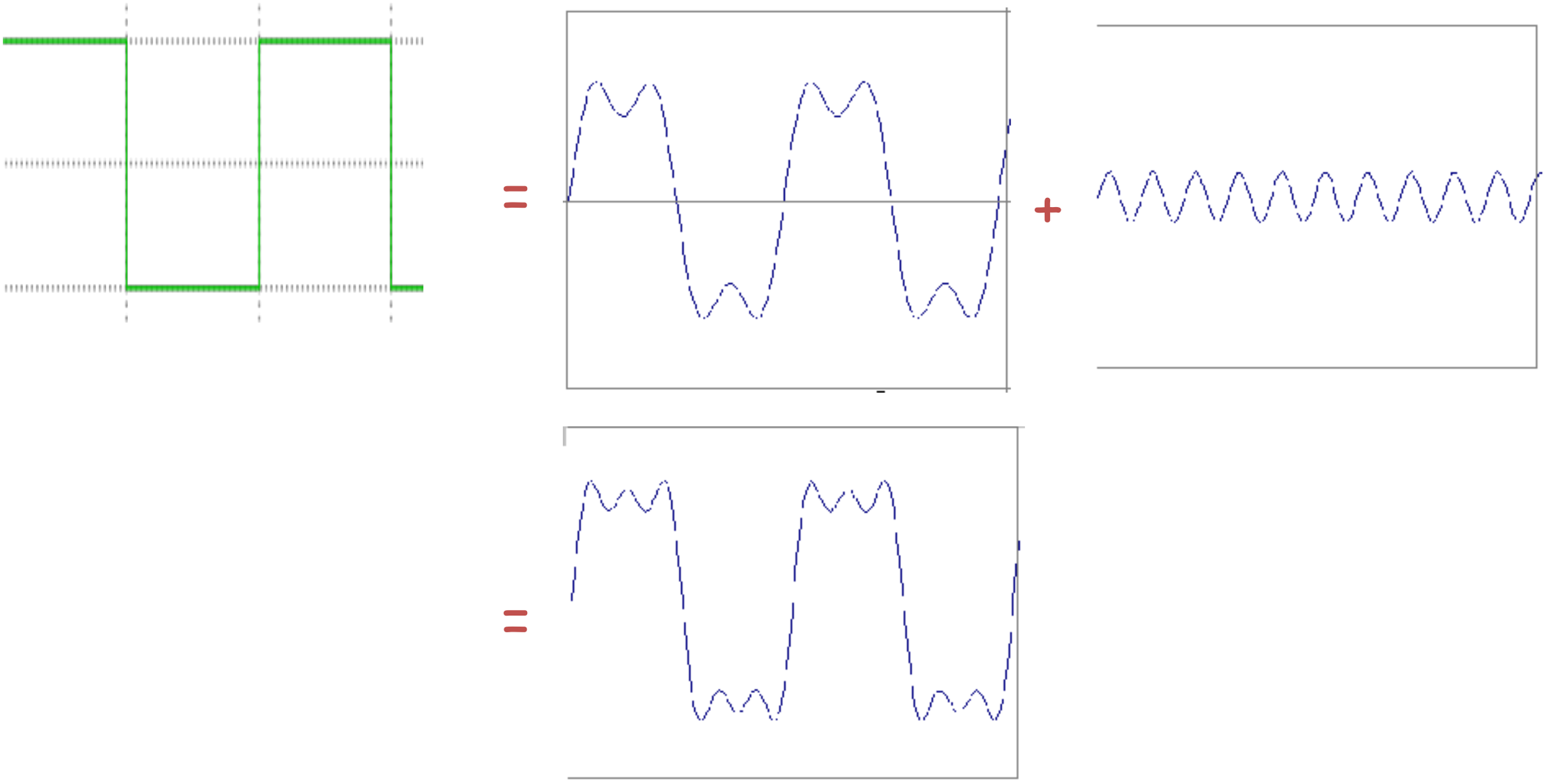
+



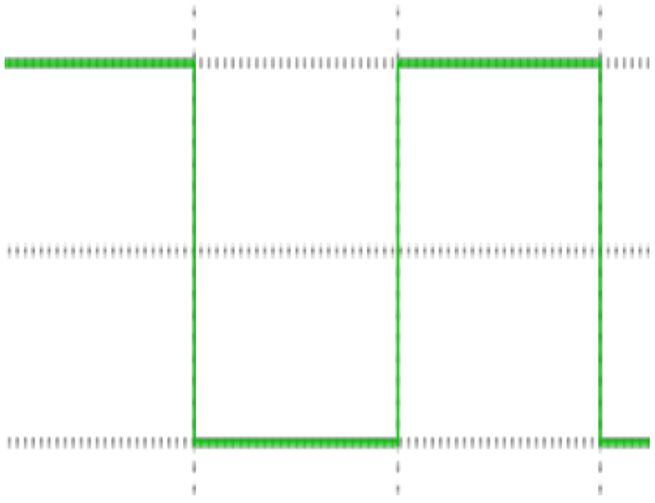
=



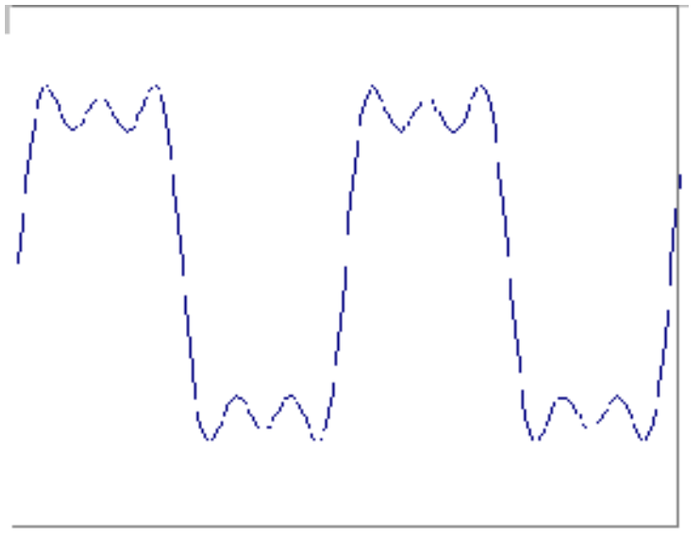
Frequency Spectra



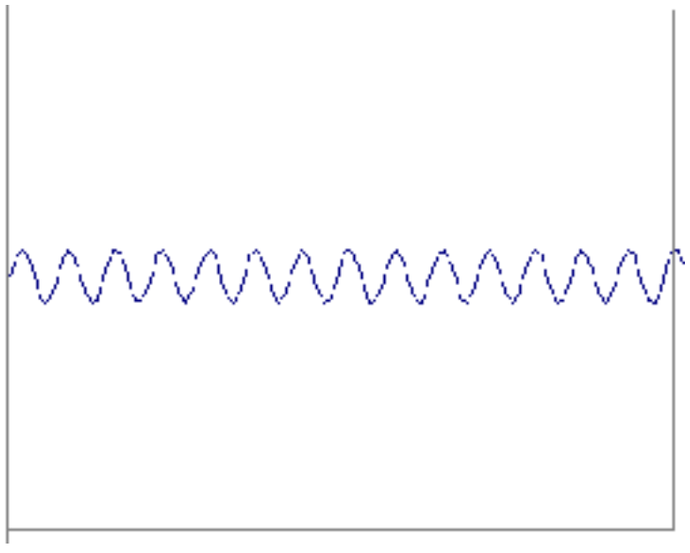
Frequency Spectra



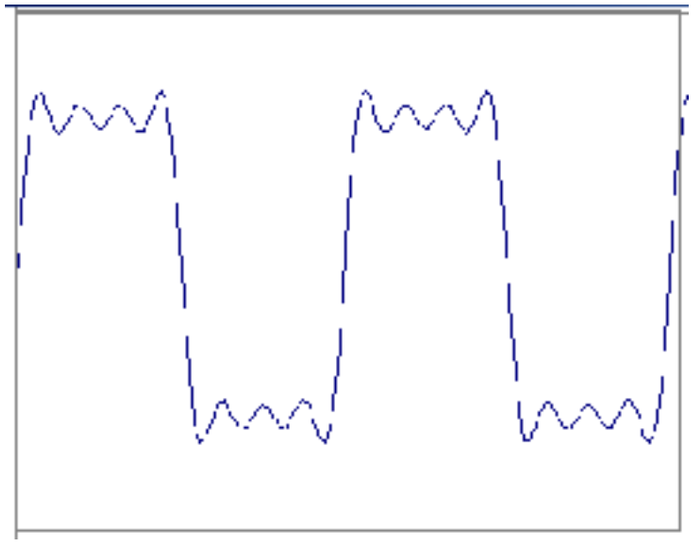
=



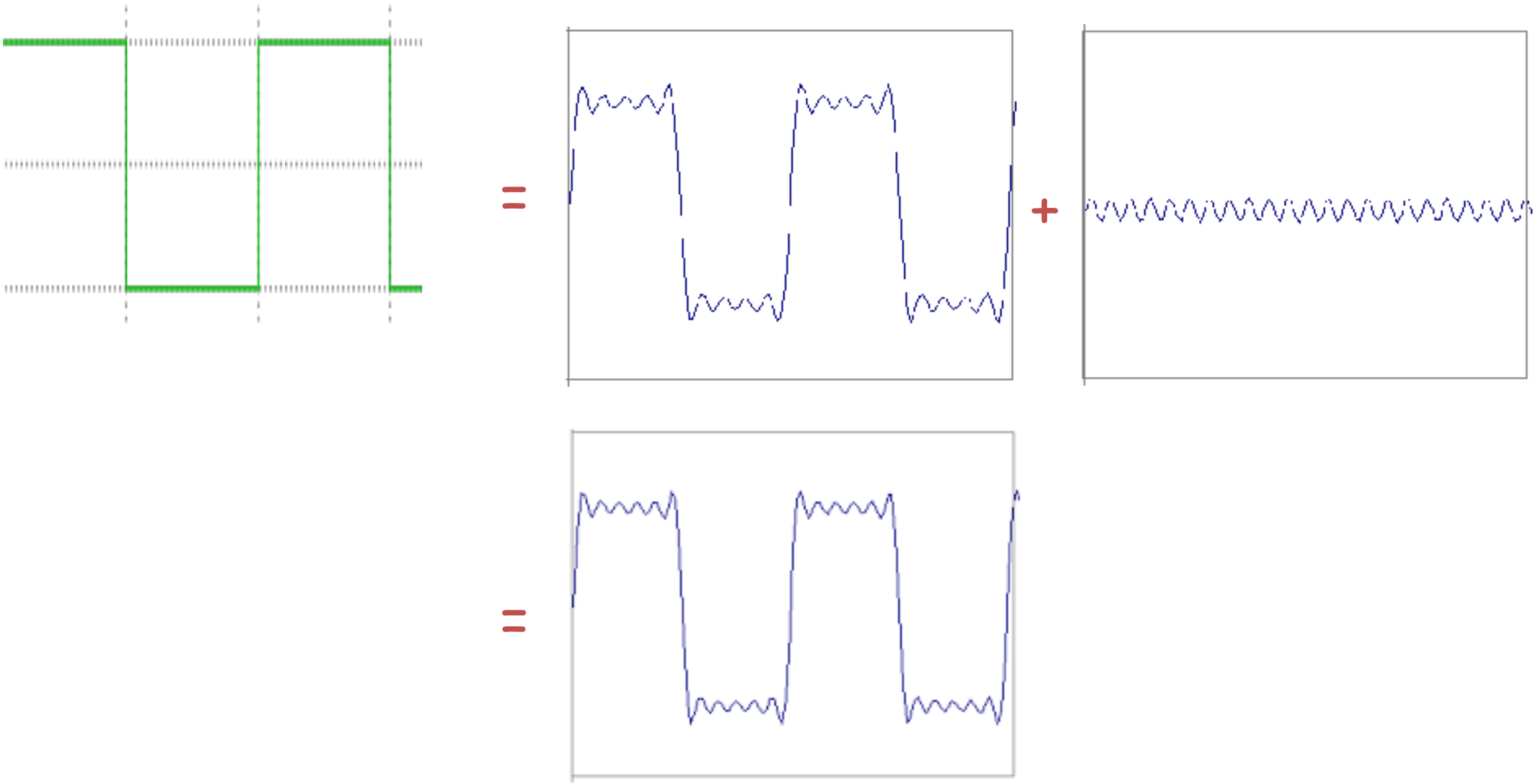
+



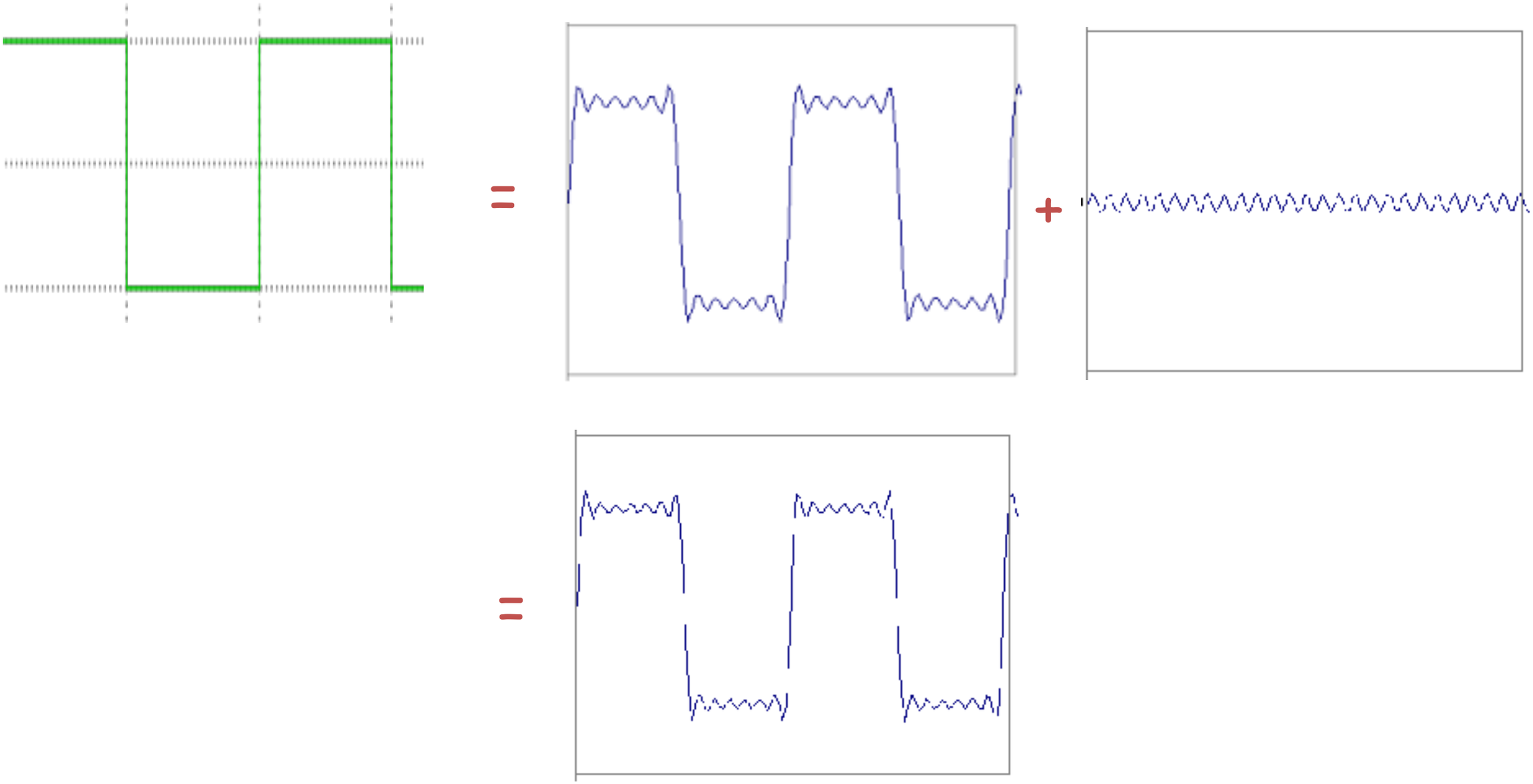
=



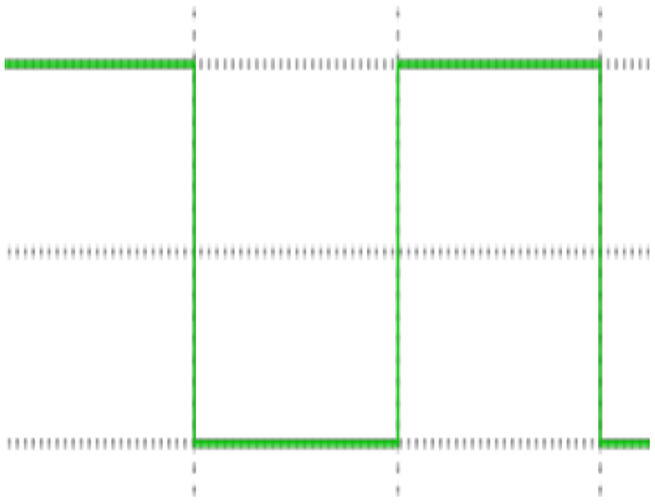
Frequency Spectra



Frequency Spectra

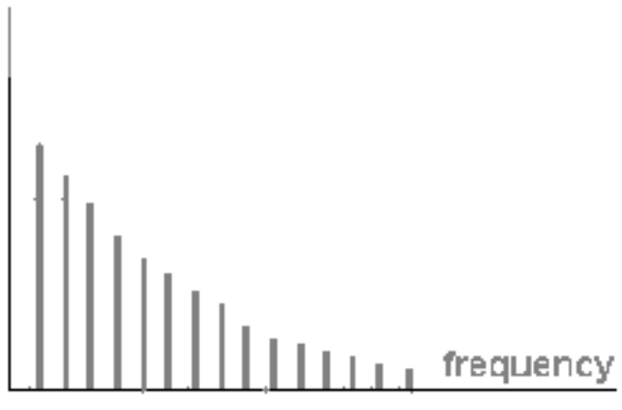


Frequency Spectra



=

$$A\sum_{k=1}^{\infty}\frac{1}{k}\sin(2\pi kt)$$



Formal definition of 1D Fourier transform

Forward mapping Frequency —> Time

$$F(s) = \int_{-\infty}^{\infty} f(t) e^{-j2\pi st} dt$$

Backward mapping Time —> Frequency: inverse FT

$$f(t) = \int_{-\infty}^{\infty} F(s) e^{+j2\pi st} ds$$

Fourier series (**Euler's identity**)

$$f(t) = \sum_{n=-\infty}^{\infty} c_n e^{j\frac{2\pi n}{T}t}$$

$$f(t) = a_0 + \sum_{n=1}^{\infty} a_n \sin\left(j2\pi\frac{n}{T}t\right) + \sum_{n=1}^{\infty} b_n \cos\left(j2\pi\frac{n}{T}t\right)$$

Digital, Continuous and Fast Fourier Transform

DFT **CFT** **FFT**

For now, you only need to know the big picture:

	Continuous Time	Discrete Time
Fourier Series	$x(t) = \sum_{k=-\infty}^{\infty} a_k e^{jk\omega_0 t}$ continuous and periodic in time $a_k = \frac{1}{T}\int_T x(t)e^{-jk\omega_0 t}dt$ discrete and aperiodic in frequency	$x[n] = \sum_{k=<N>} a_k e^{jk\frac{2\pi}{N}n}$ discrete and periodic in time $a_k = \frac{1}{N}\sum_{n=<N>} x[n]e^{-jk\frac{2\pi}{N}n}$ discrete and periodic in frequency
Fourier Transform	$x(t) = \frac{1}{2\pi}\int_{-\infty}^{\infty} X(j\omega)e^{j\omega t}d\omega$ continuous and aperiodic in time $X(j\omega) = \int_{-\infty}^{\infty} x(t)e^{-j\omega t}dt$ continuous and aperiodic in frequency	$x[n] = \frac{1}{2\pi}\int_{2\pi} X(e^{j\omega})e^{j\omega n}d\omega$ discrete and aperiodic in time $X(e^{j\omega}) = \sum_{n=-\infty}^{\infty} x[n]e^{-j\omega n}$ continuous and periodic in frequency

Take a signal processing course if you ever want to work with images!

Fourier Transform

- Fourier transform stores the magnitude and phase at each frequency
  - Magnitude encodes how much signal there is at a particular frequency
  - Phase encodes spatial information (indirectly)
  - For mathematical convenience, this is often notated in terms of real and complex numbers

Amplitude: $A = \pm\sqrt{R(\omega)^2 + I(\omega)^2}$

Phase: $\phi = \tan^{-1}\frac{I(\omega)}{R(\omega)}$

The Convolution Theorem

- The Fourier transform of the convolution of two functions is the product of their Fourier transforms

$$\mathrm{F}[g * h] = \mathrm{F}[g]\mathrm{F}[h]$$

- The inverse Fourier transform of the product of two Fourier transforms is the convolution of the two inverse Fourier transforms

$$\mathrm{F}^{-1}[gh] = \mathrm{F}^{-1}[g] * \mathrm{F}^{-1}[h]$$

- *Convolution in spatial domain is equivalent to multiplication in frequency domain!*

Properties of Fourier Transforms

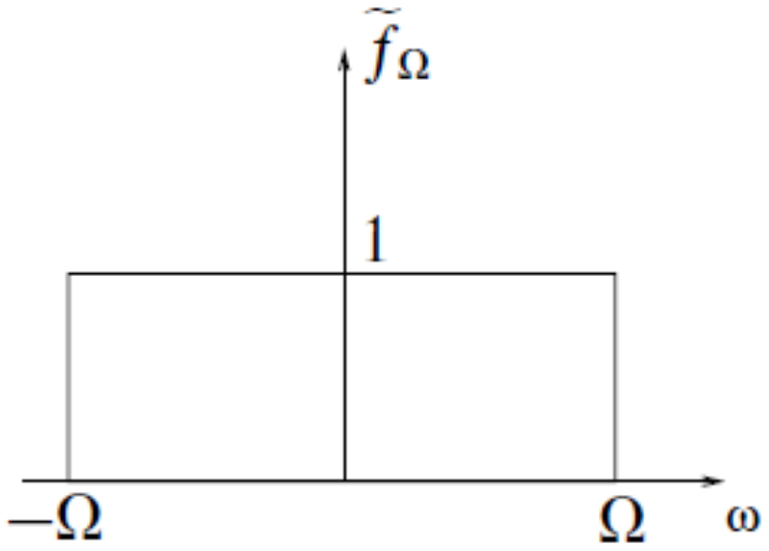
- Linearity

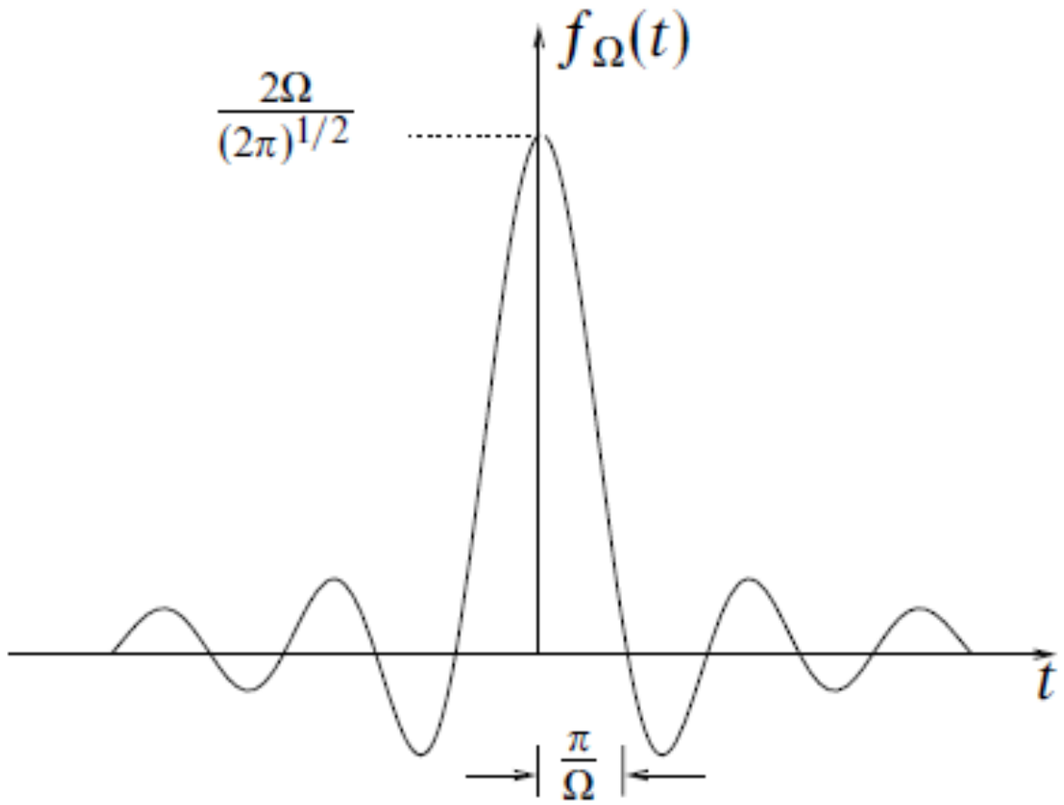
- Fourier transform of a real signal is symmetric about the origin

- The energy of the signal is the same as the energy of its Fourier transform

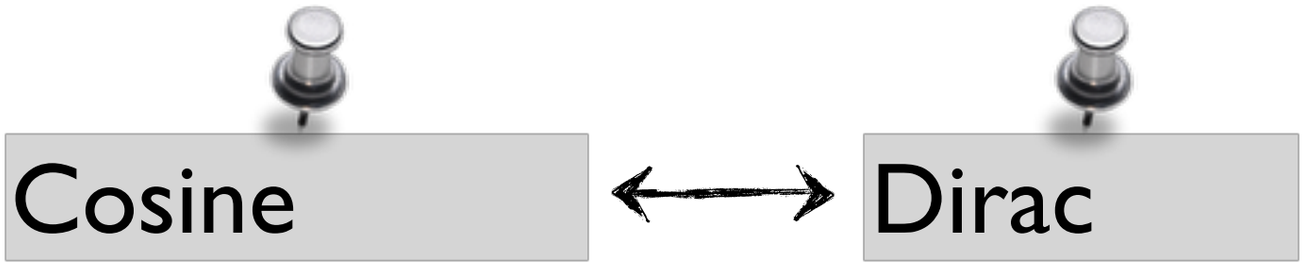
Common Transform pairs

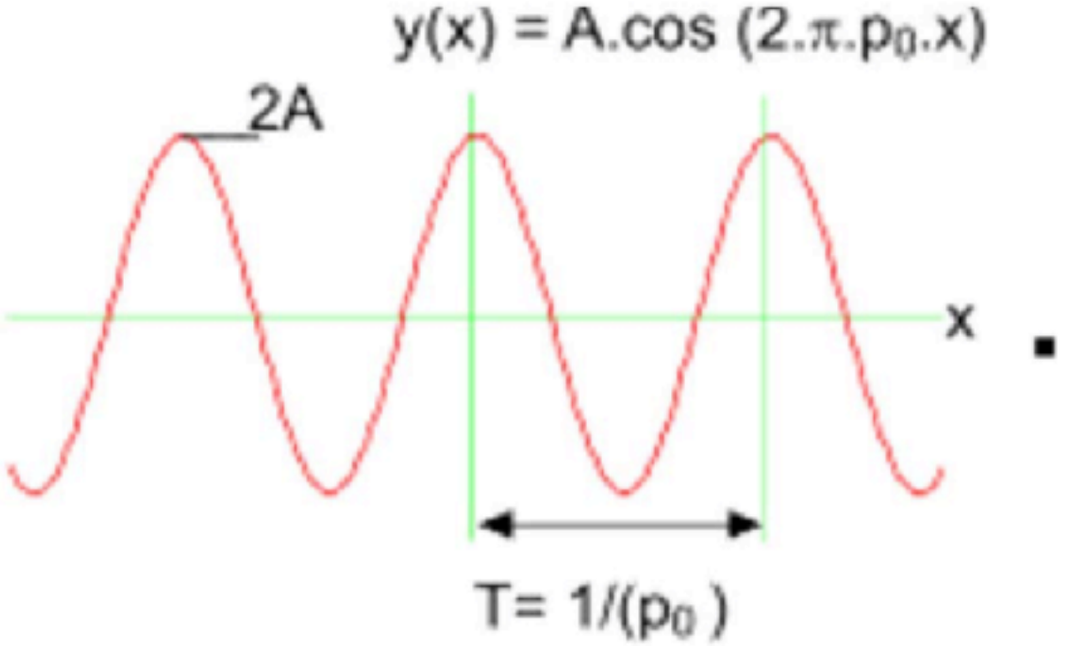
Rectangle ⟷ Sinc

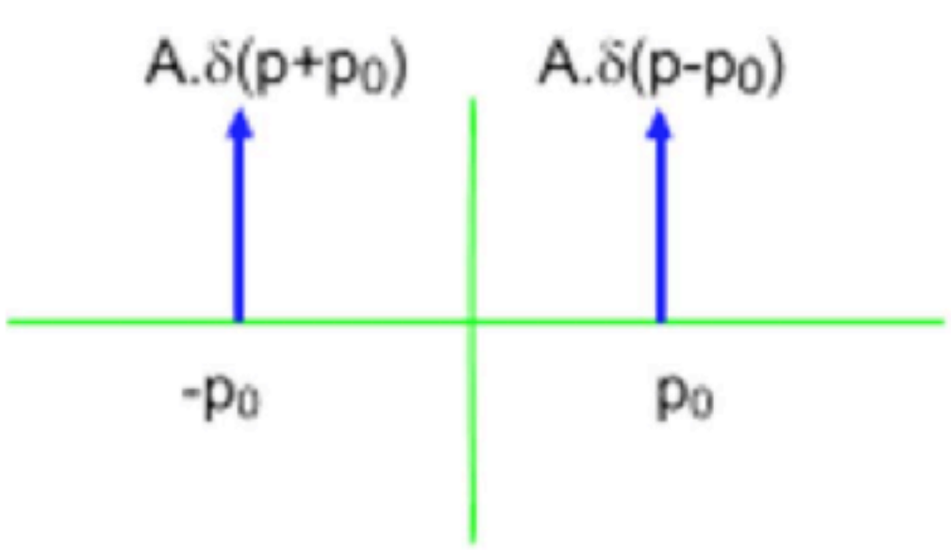




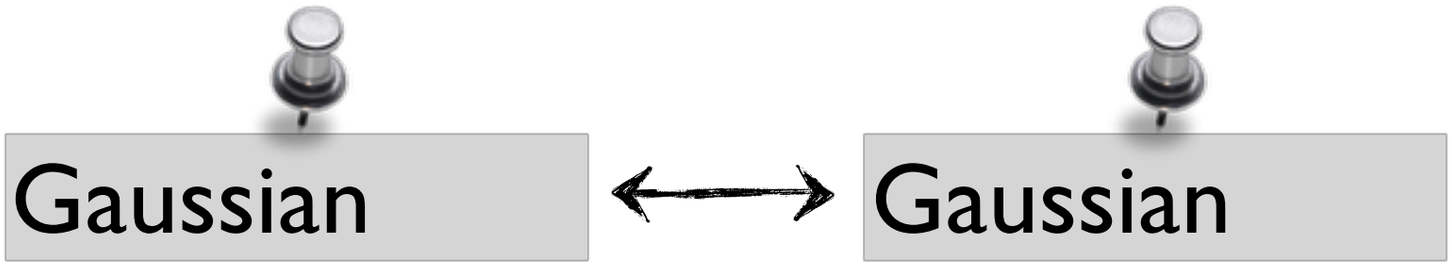
Common Transform pairs

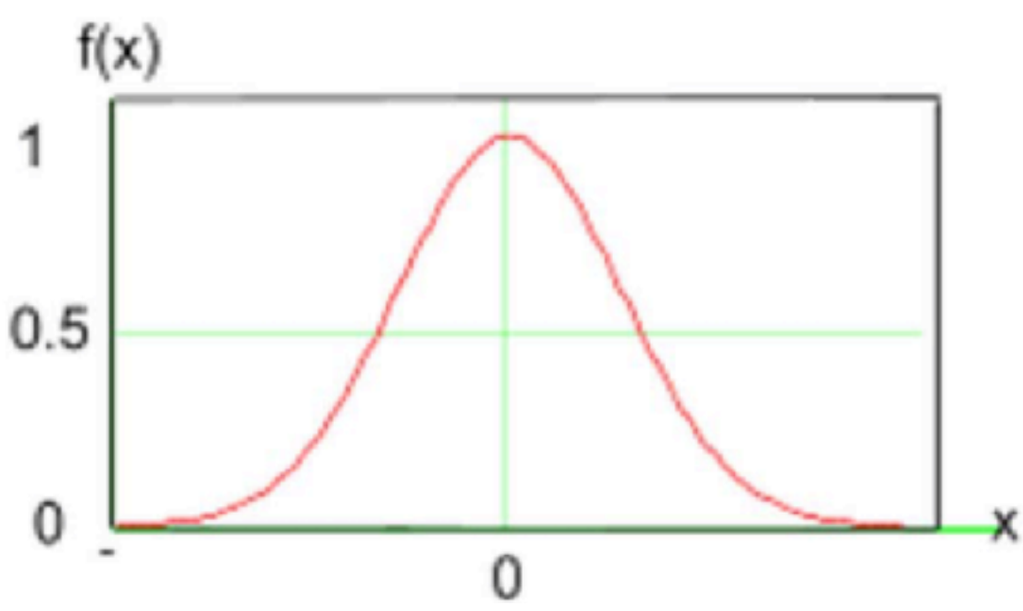


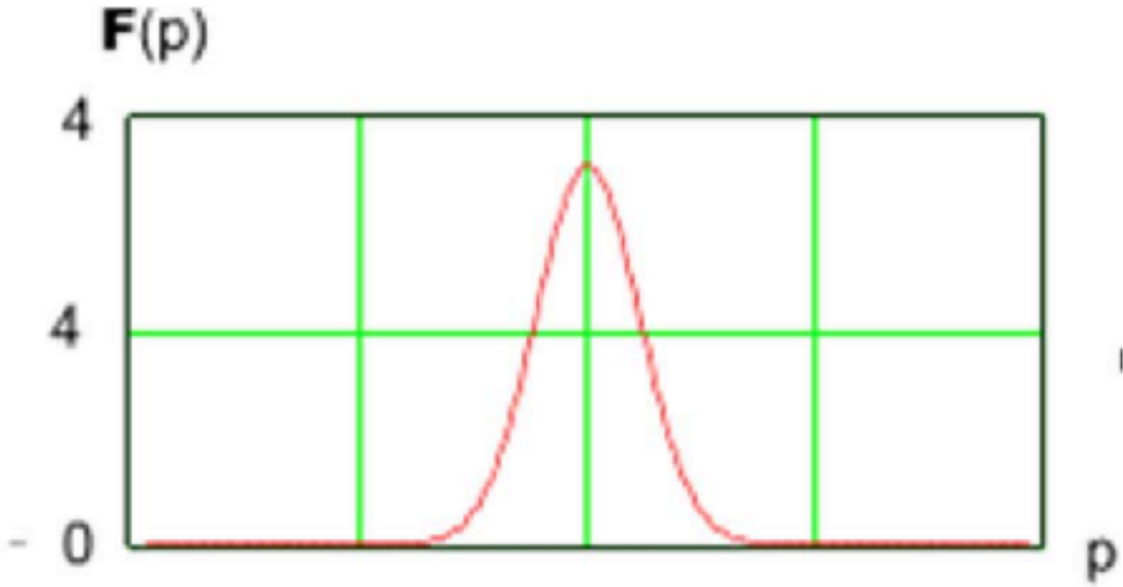




Common Transform pairs

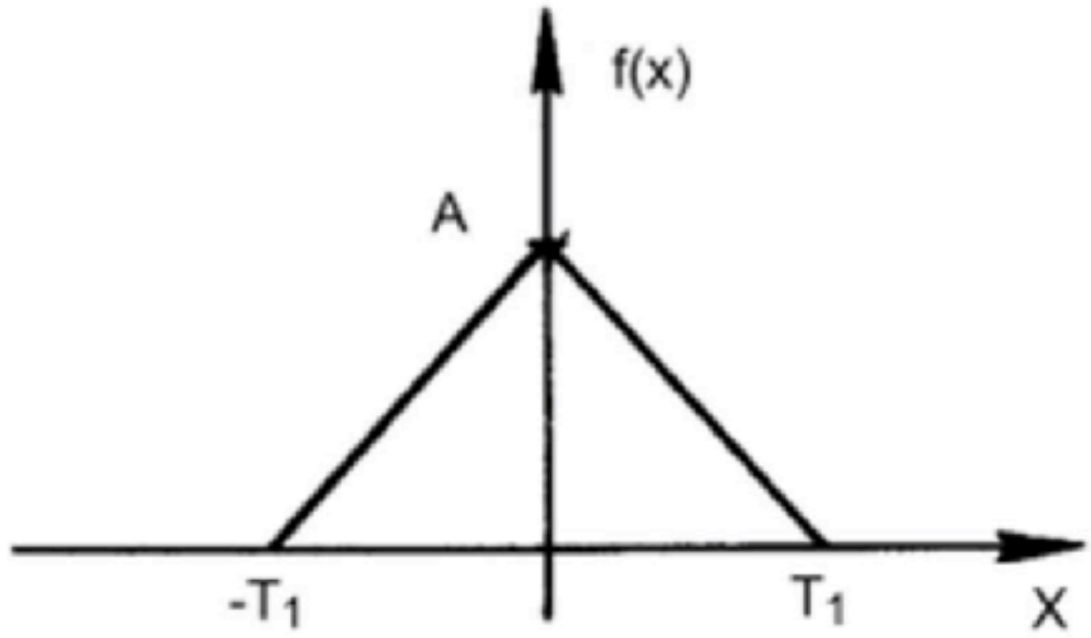






Your turn

What is the FT of a triangle function?



Hint: how do you get triangle function from the functions shown so far?

Convolution and Multiplication

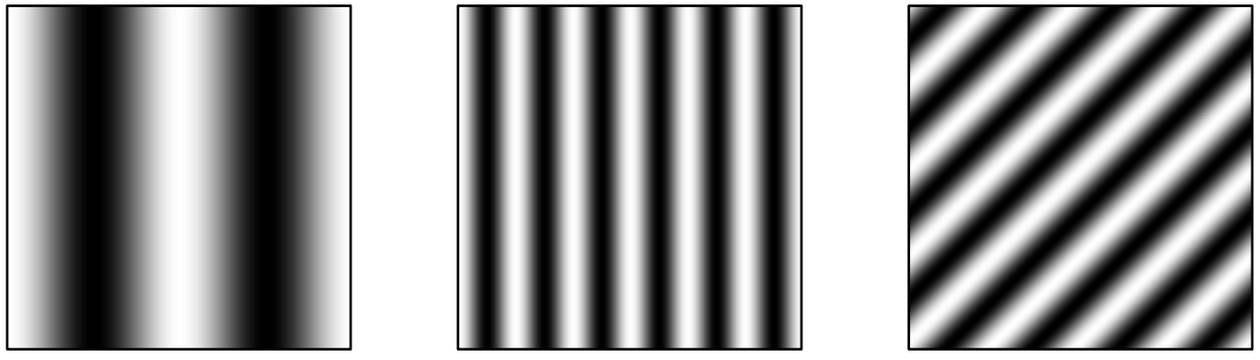
Triangle = box convolved with box

So its FT is sinc x sinc

1D —> 2D

2D Fourier analysis in images

Intensity Image
(spatial)



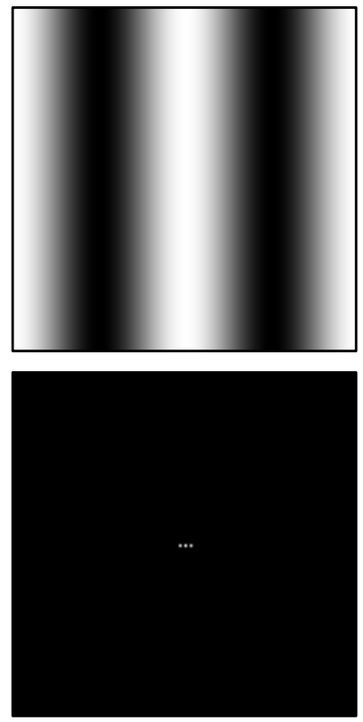
Fourier Image
(Frequency)

http://sharp.bu.edu/~slehar/fourier/fourier.html#filtering

2D Fourier transform Plot

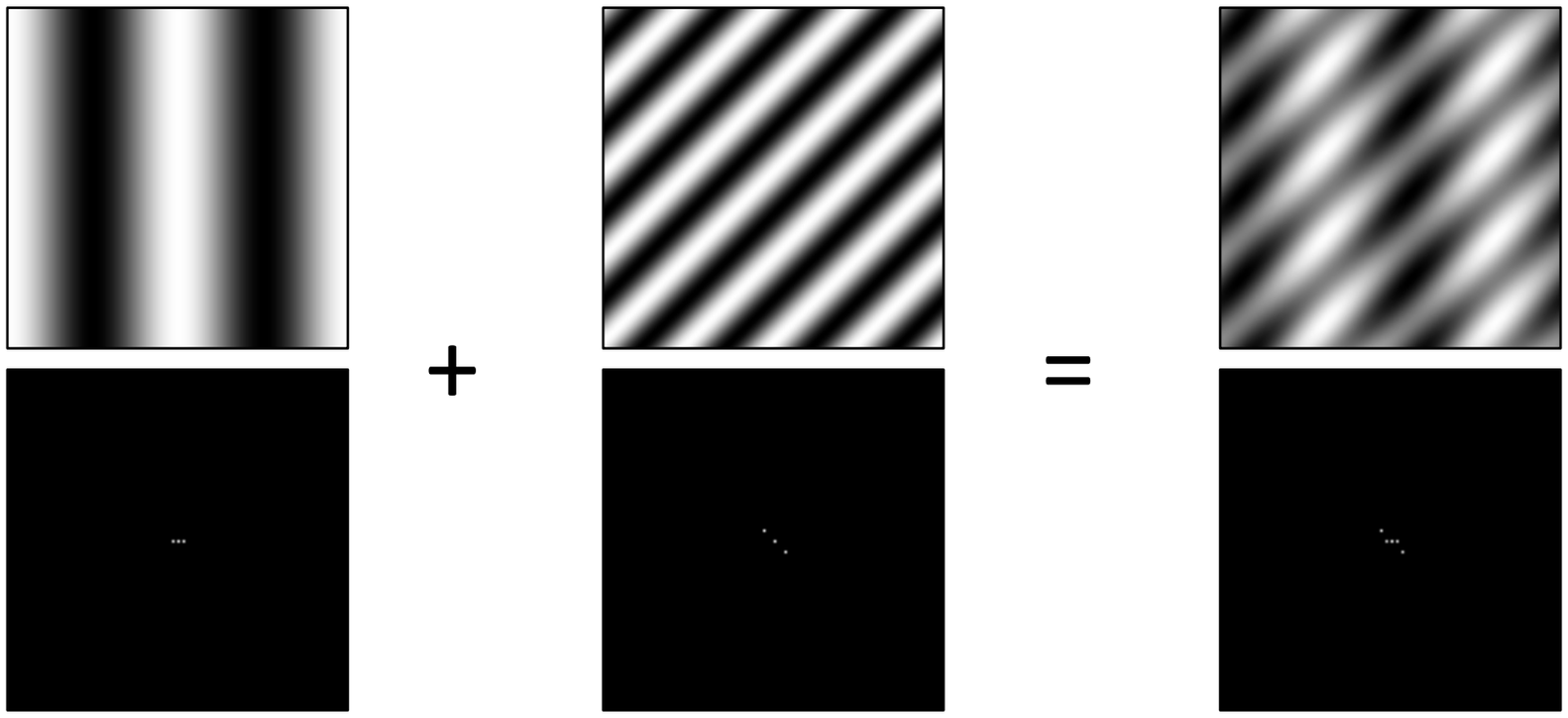
```
import cv2
import numpy as np
from matplotlib import pyplot as plt

img = cv2.imread('messi5.jpg',0)
f = np.fft.fft2(img)
fshift = np.fft.fftshift(f)
magnitude_spectrum = 20*np.log(np.abs(fshift))
```



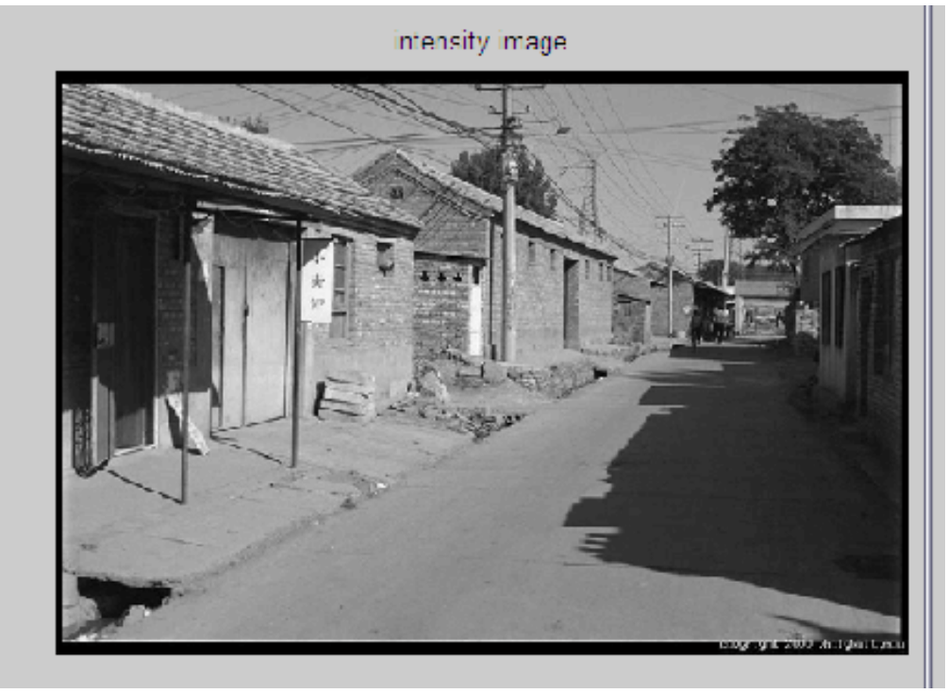
- Only plot magnitude for visualization
    - Need phase for reconstruction
- Image center —> zero frequency
    - fftshift
    - log scale

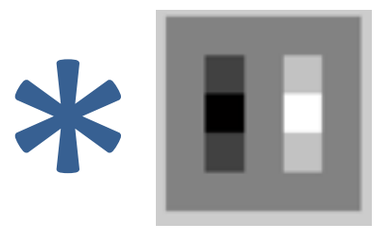
Signals can be (de)composed

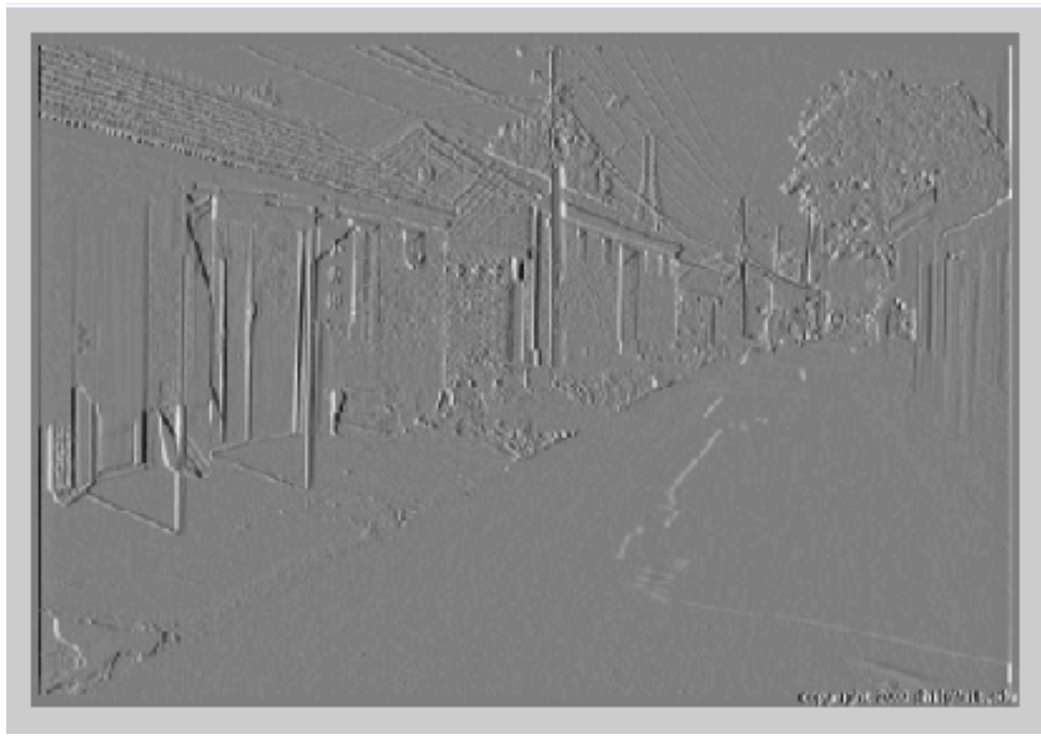


http://sharp.bu.edu/~slehar/fourier/fourier.html#filtering
More: http://www.cs.unm.edu/~brayer/vision/fourier.html

Remember?
Filtering in spatial domain

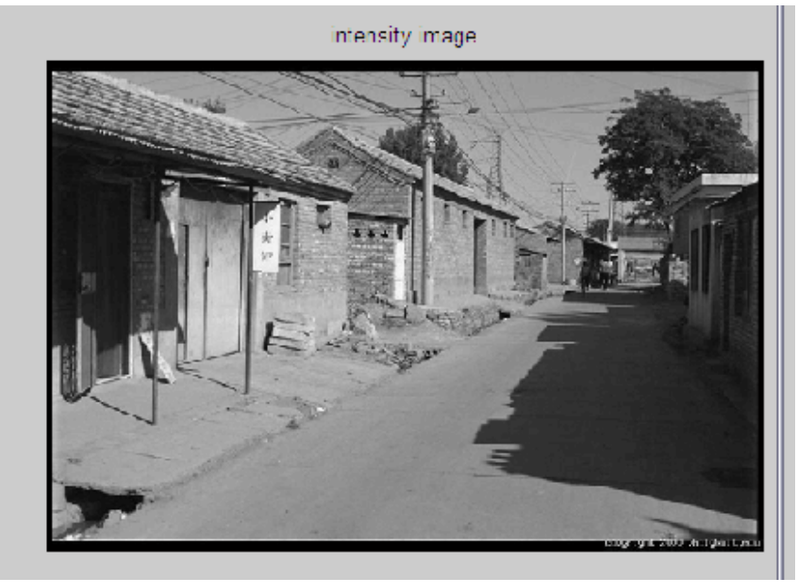




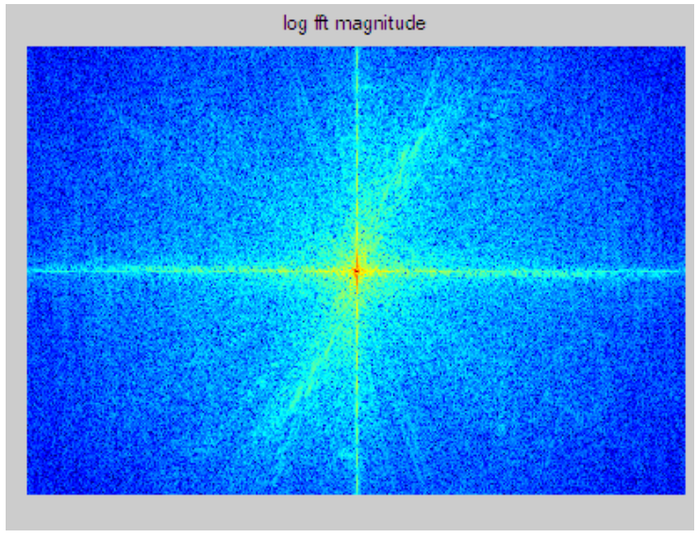


1	0	-1
2	0	-2
1	0	-1

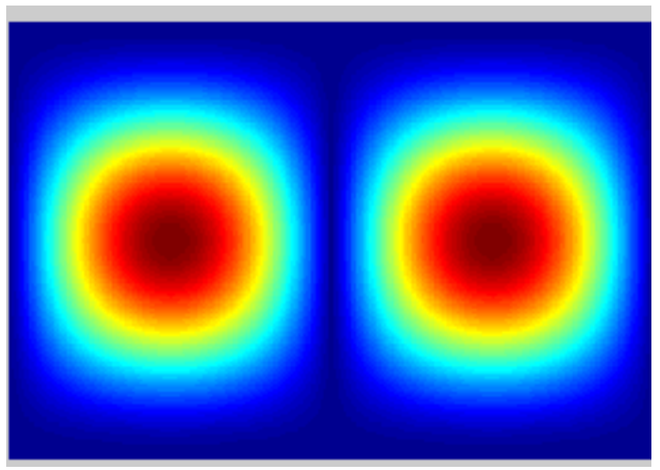
Filtering in frequency domain



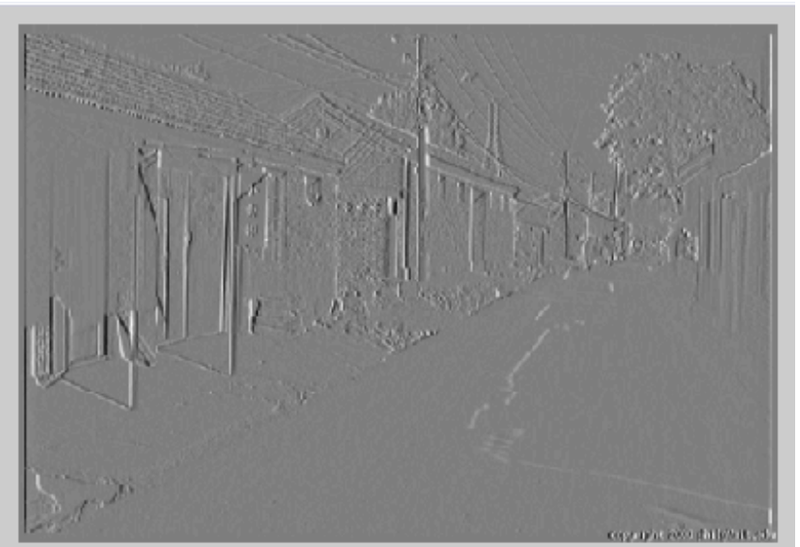
FFT



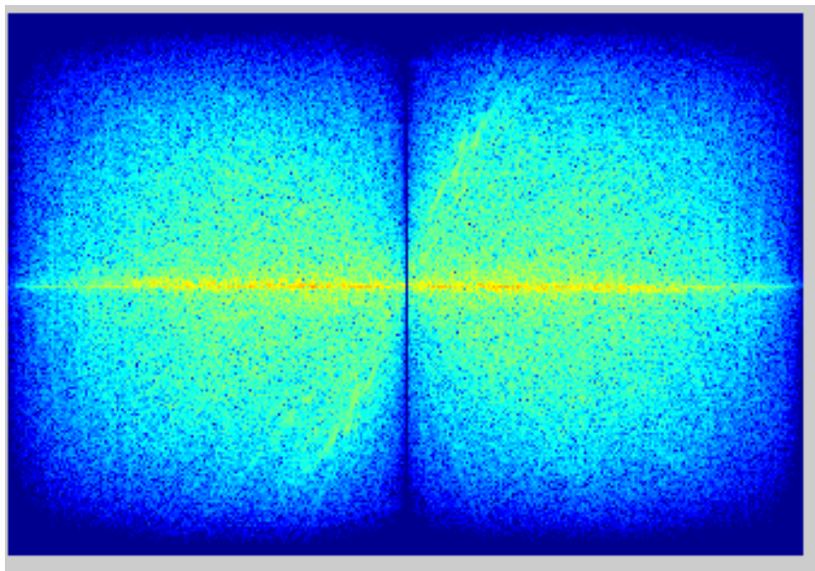
FFT



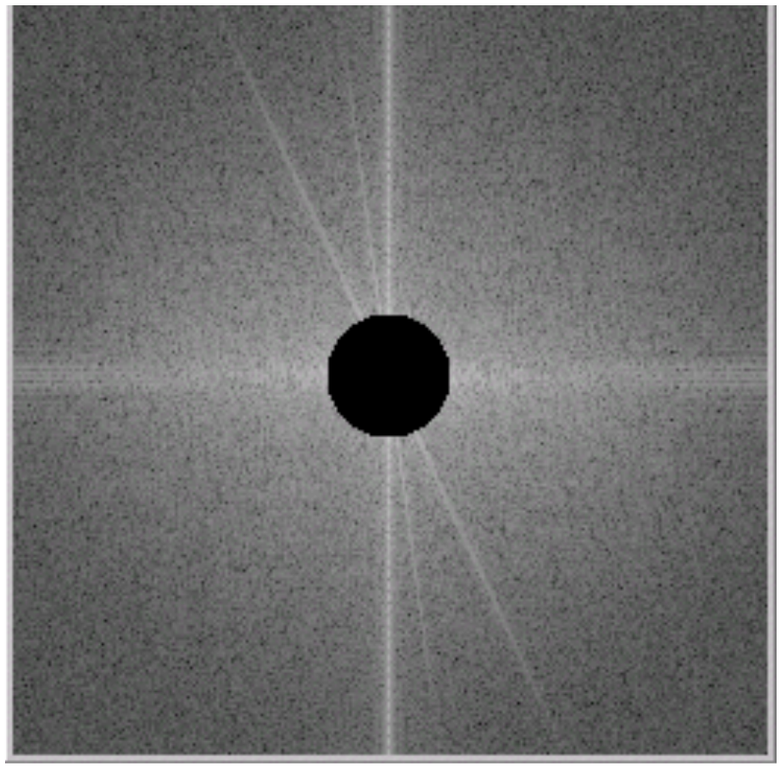
=



Inverse FFT



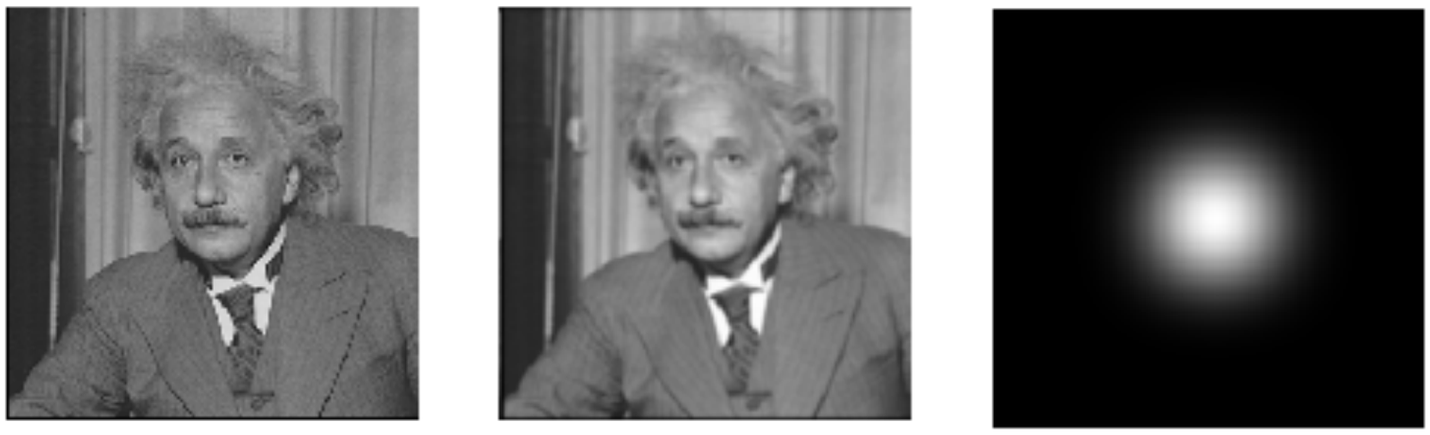
Edges in images



High-pass filter

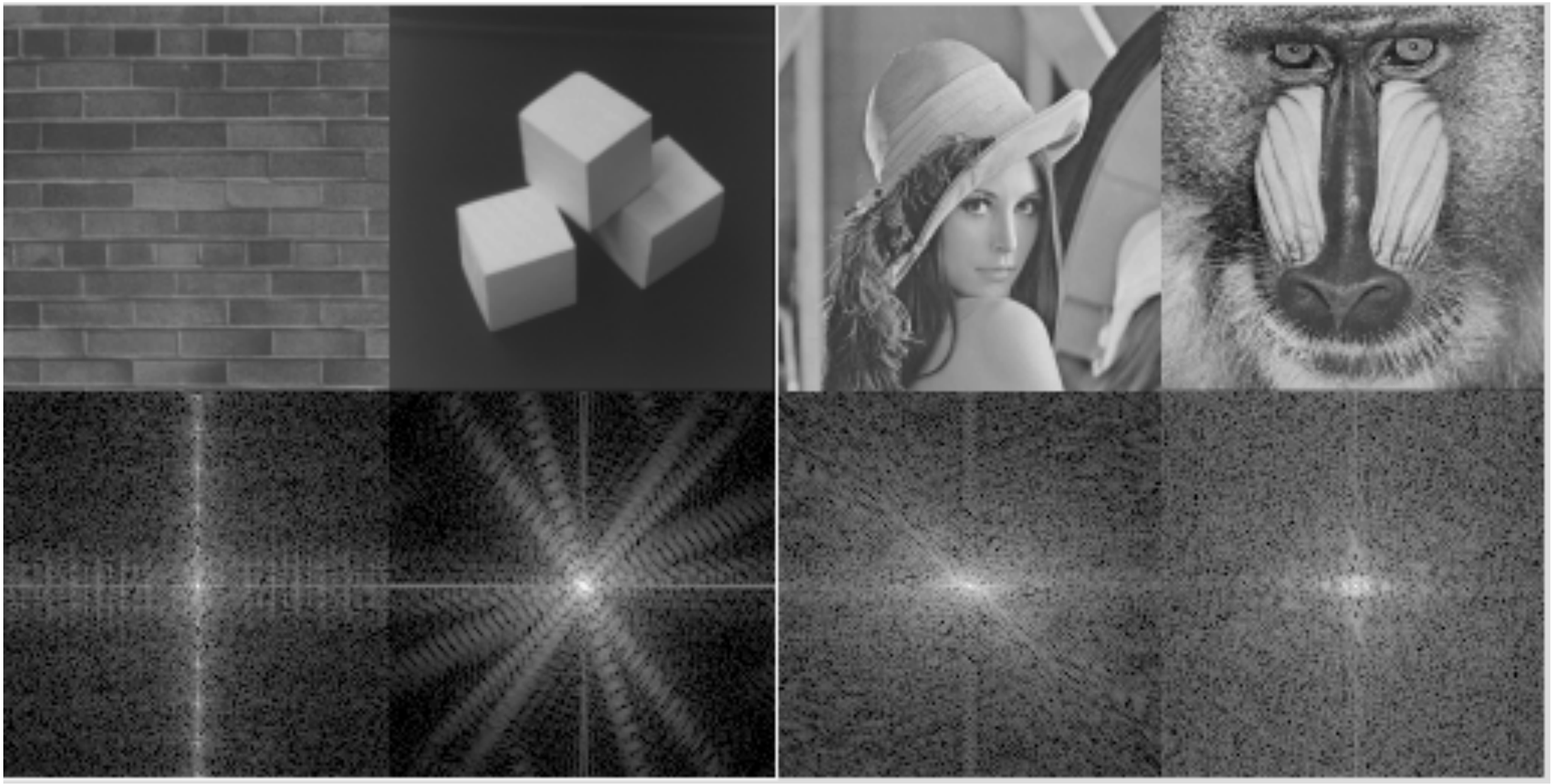
Low-pass, Band-pass, High-pass filters

Low-pass:



High-pass / band-pass:

2D Fourier transform examples



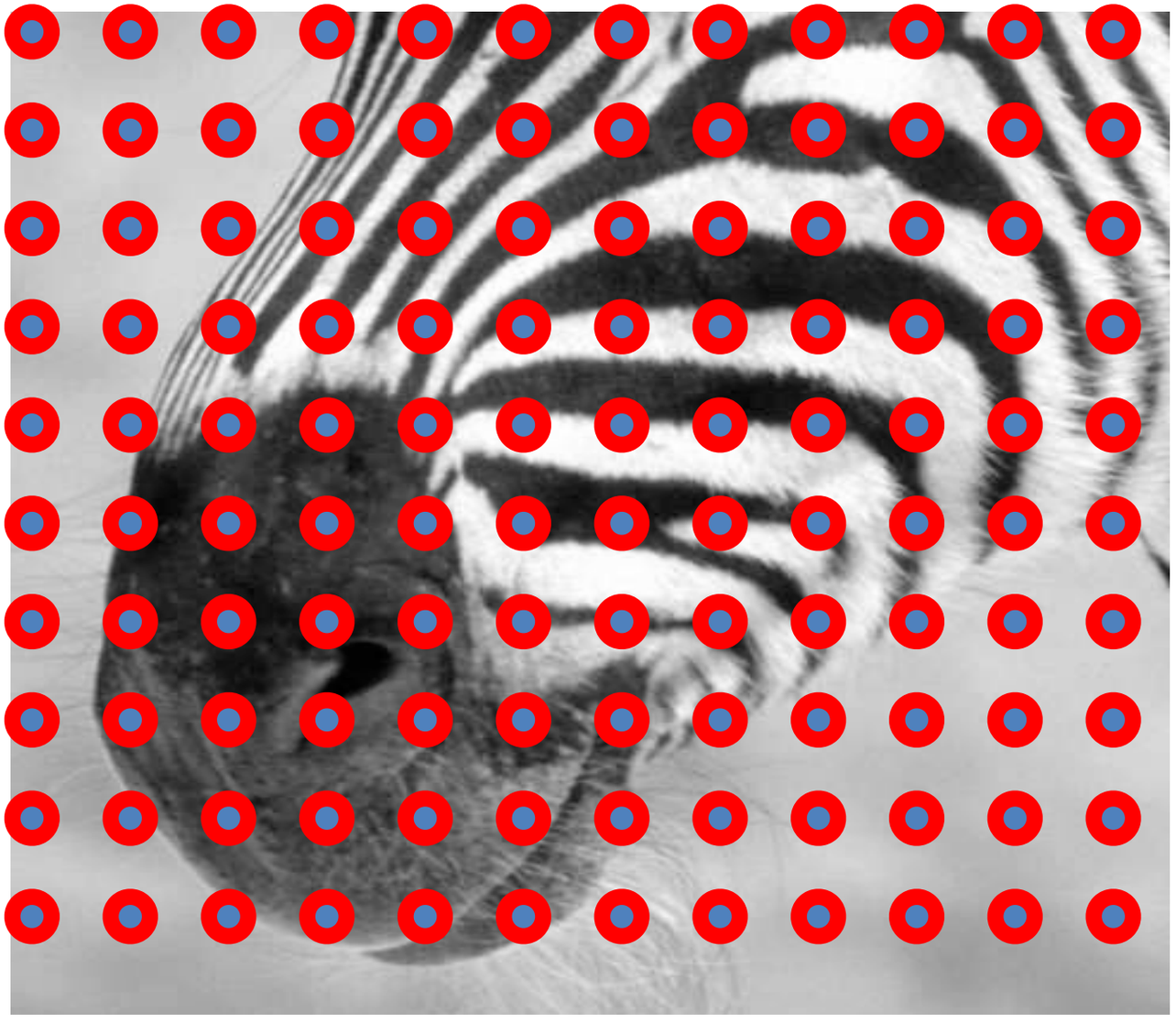
Sampling

Sampling

Why does a lower resolution image still make sense to us? What do we lose?

Image: http://www.flickr.com/photos/igorms/136916757/

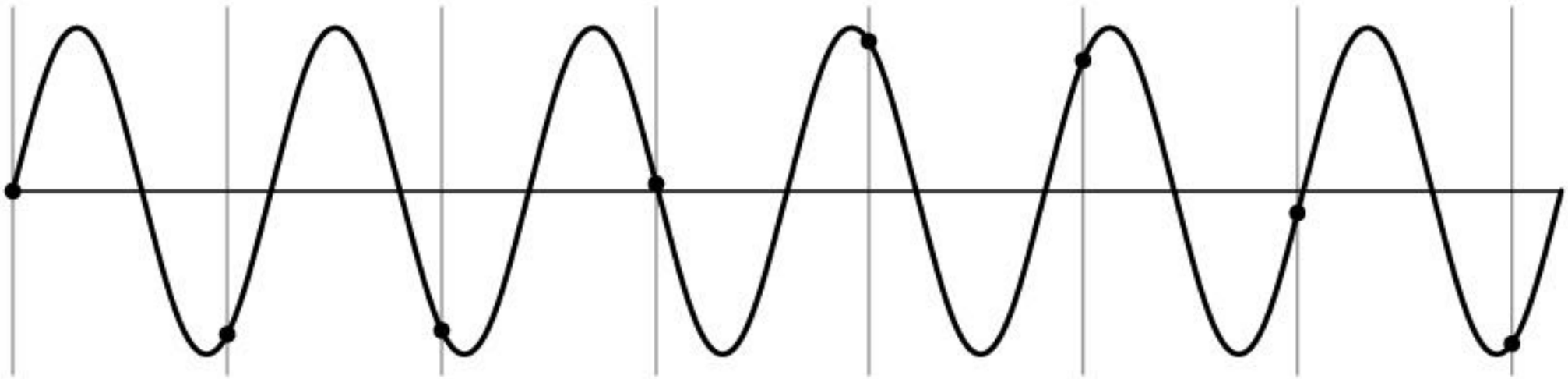
Subsampling by a factor of 2



Throw away every other row and column
to create a 1/2 size image

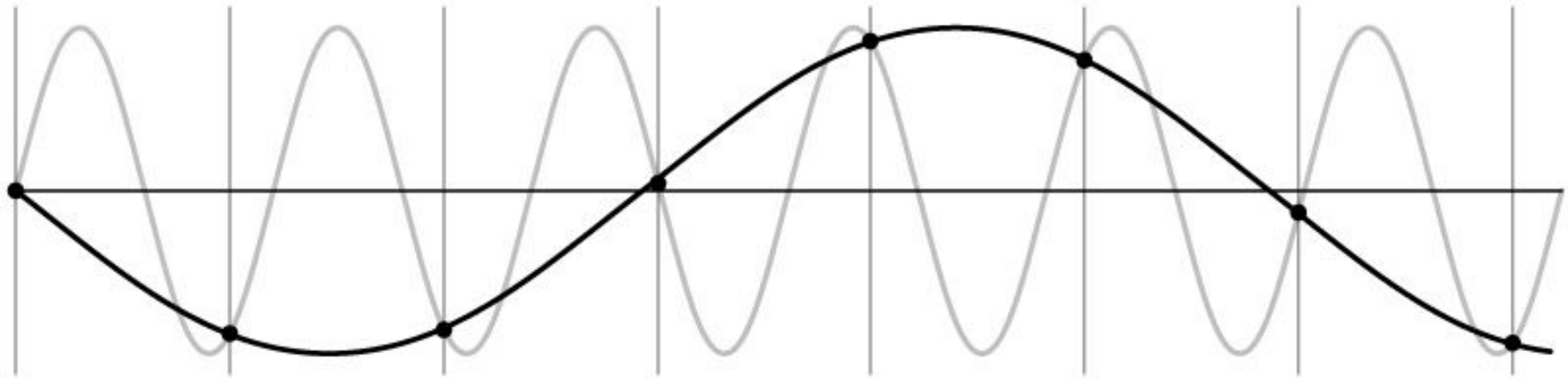
Aliasing problem

- 1D example (sinewave):



Aliasing problem

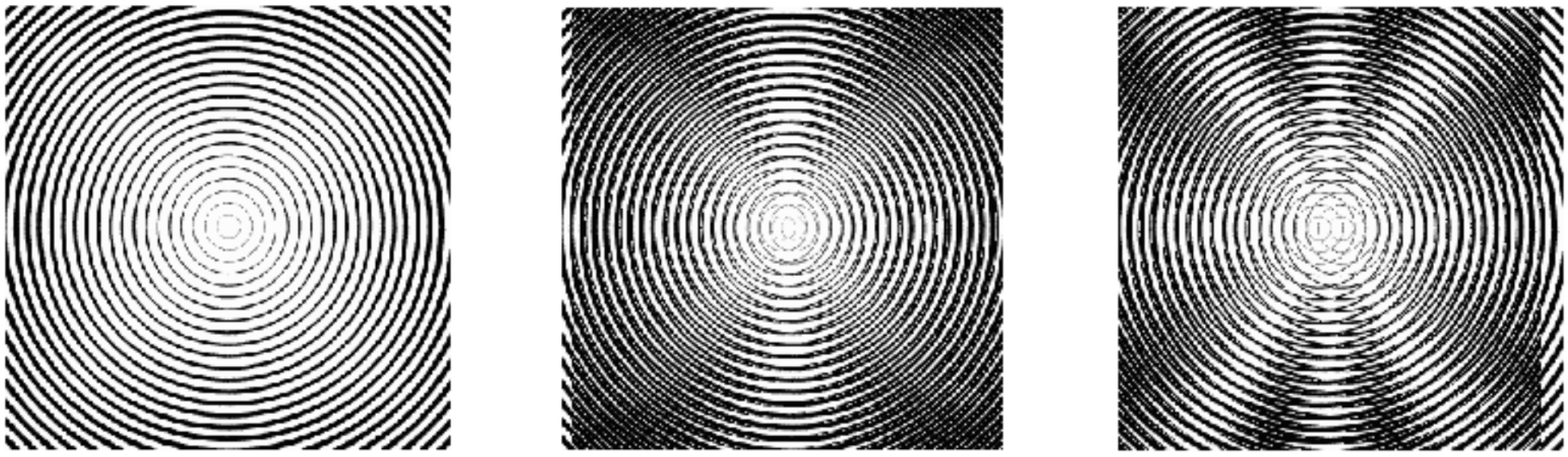
- 1D example (sinewave):



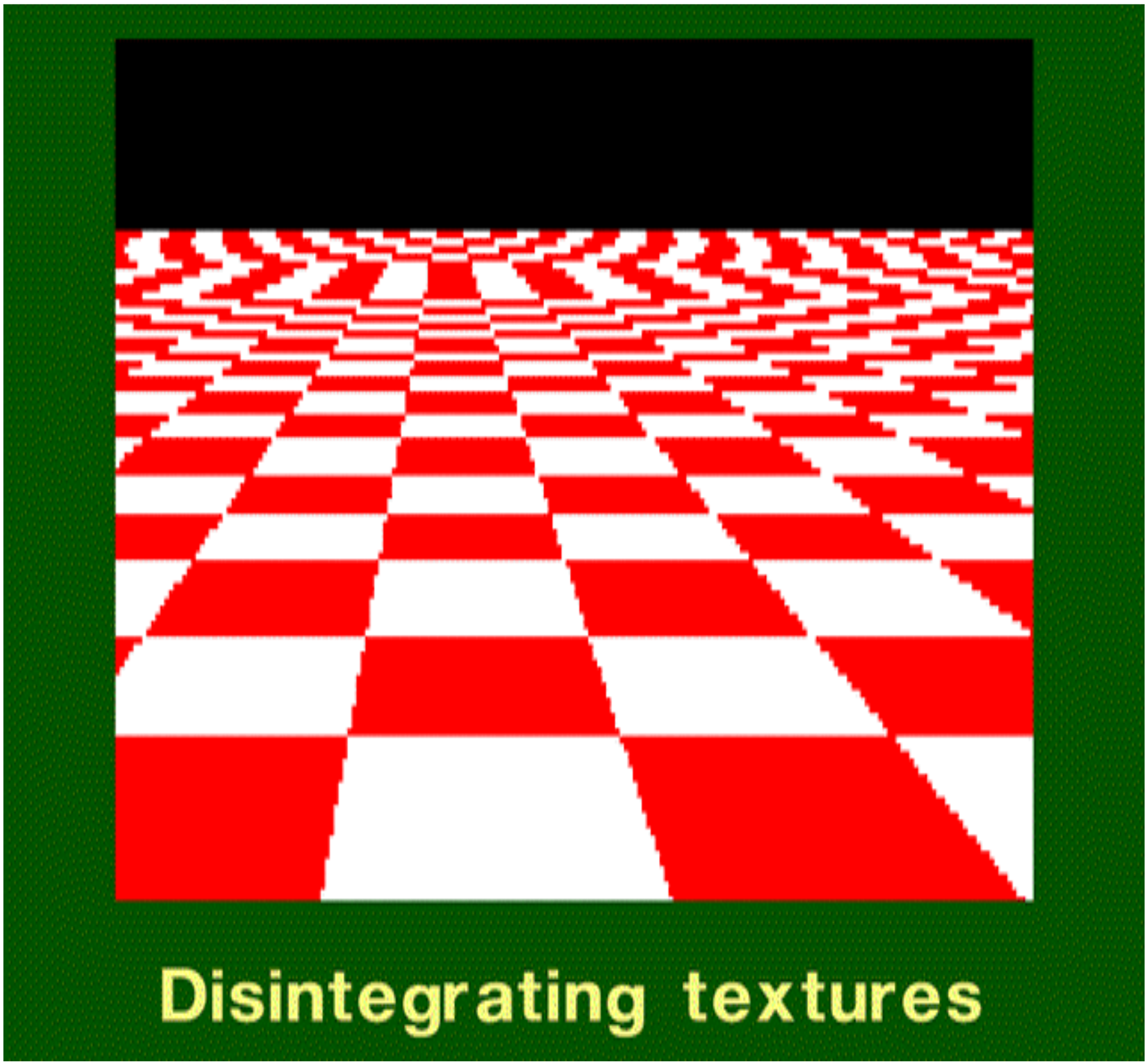
Source: S. Marschner

Aliasing happens when choosing the wrong numbers to fill in during sampling

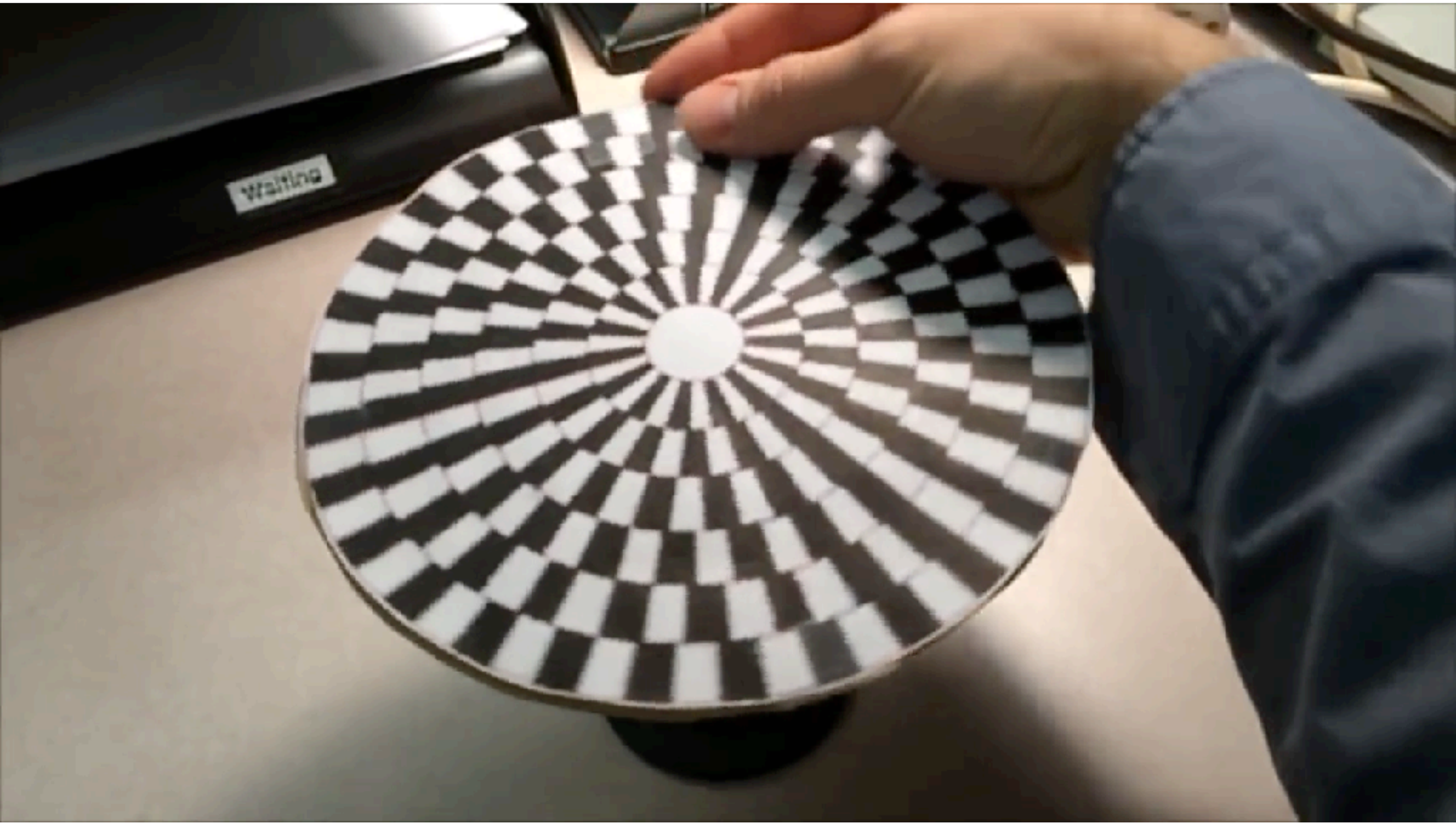
Moire pattern



Aliasing in texture mapping



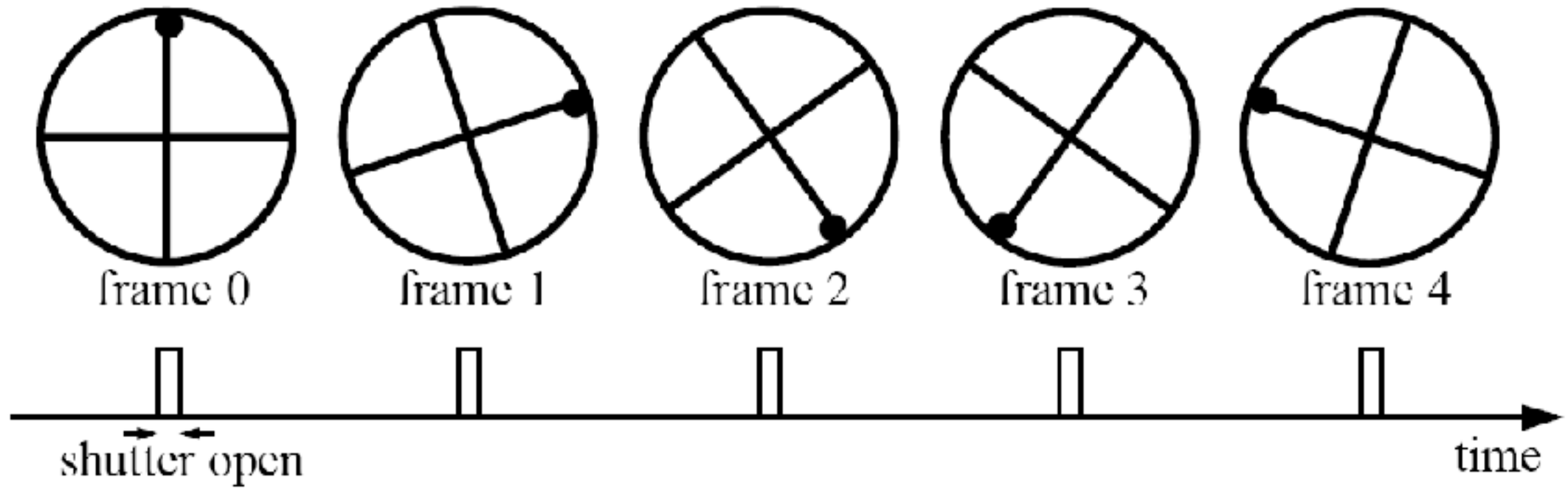
Aliasing in a video — wagon wheel effect



Aliasing in a video

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

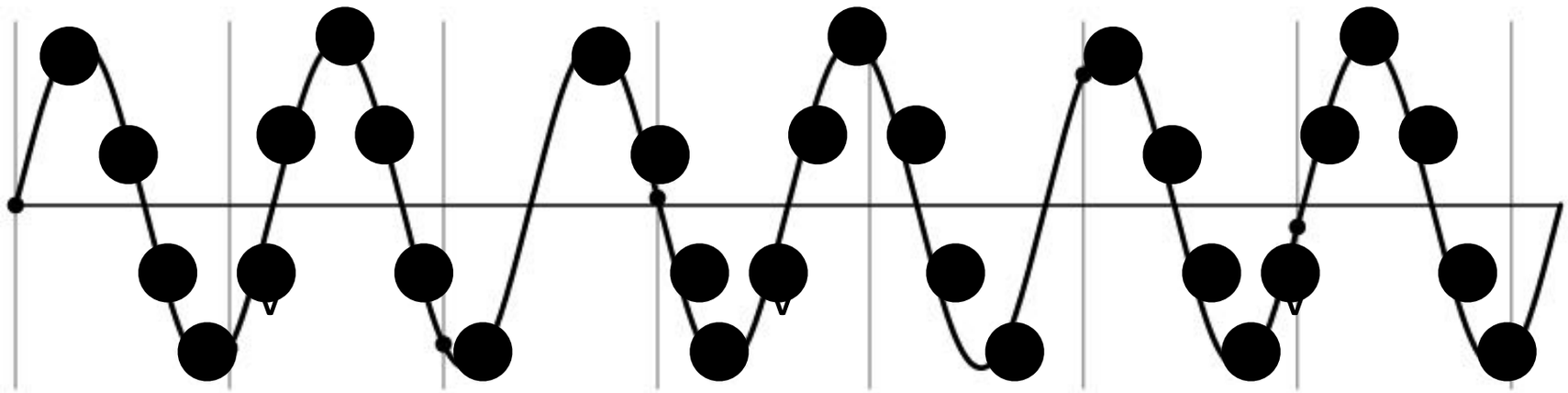
If camera shutter is only open for a fraction of a frame time (frame time = 1/30 sec. for video, 1/24 sec. for film):



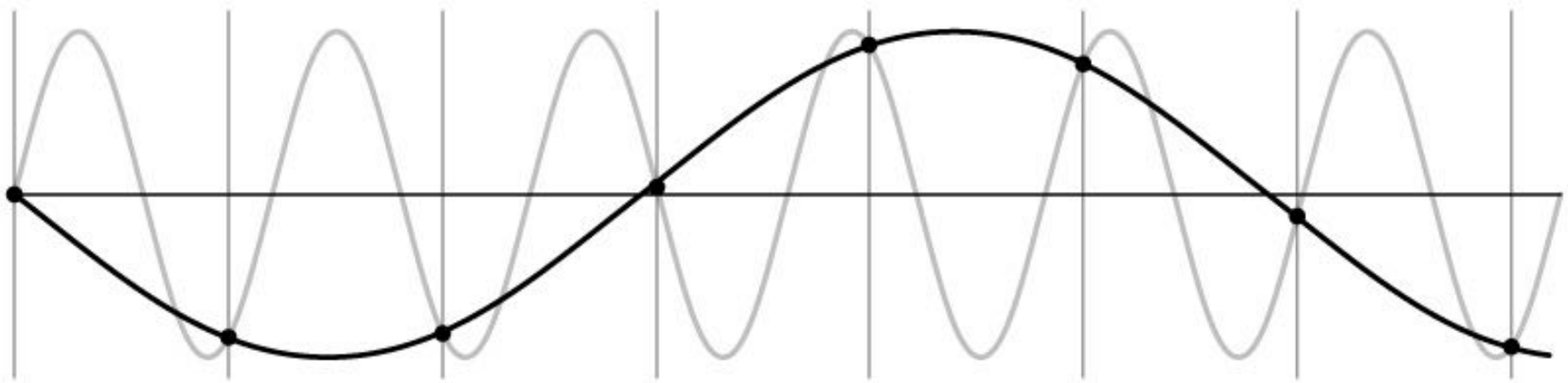
Without dot, wheel appears to be rotating slowly backwards!
(counterclockwise)

Nyquist-Shannon Sampling Theorem

- When sampling a signal at discrete intervals, the sampling frequency must be $\geq 2 \times f_{max}$
- f_{max} = max frequency of the input signal
- This will allows to reconstruct the original perfectly from the sampled version



good



bad!!

Anti-aliasing

Two Solutions:

- Sample more often
- Get rid of all frequencies that are greater than half the new sampling frequency
  - Will lose information
  - But it's better than aliasing
  - Apply a smoothing filter

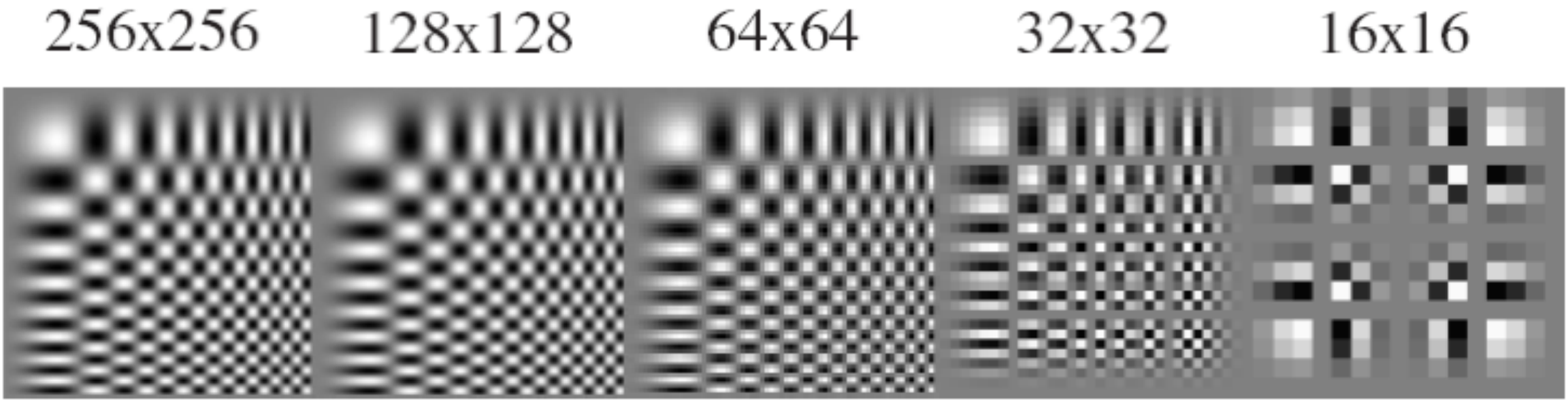
Algorithm for downsampling by factor of 2

1. Start with image(h, w)
2. Apply low-pass filter

   im_blur = imfilter(image, fspecial('gaussian', 7, 1))
3. Sample every other pixel

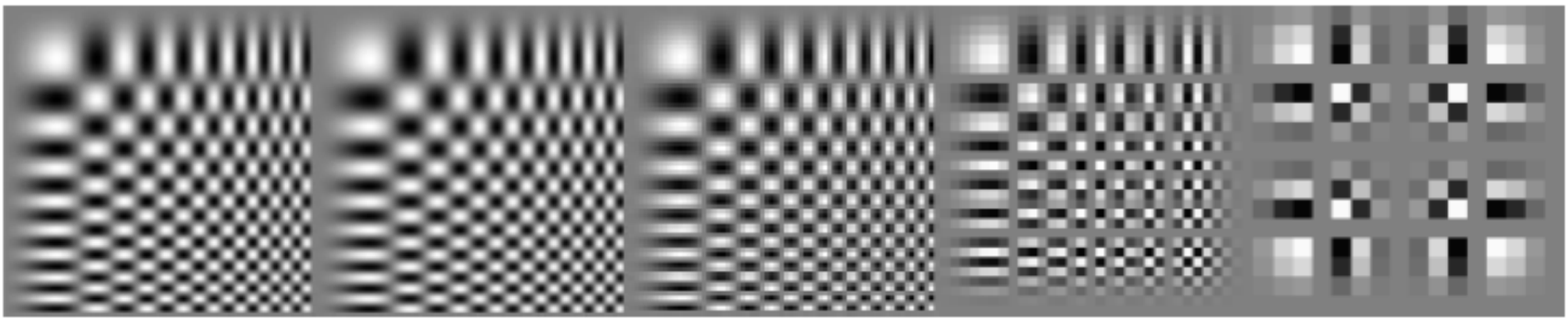
   im_small = im_blur(1:2:end, 1:2:end);

Sampling and Aliasing

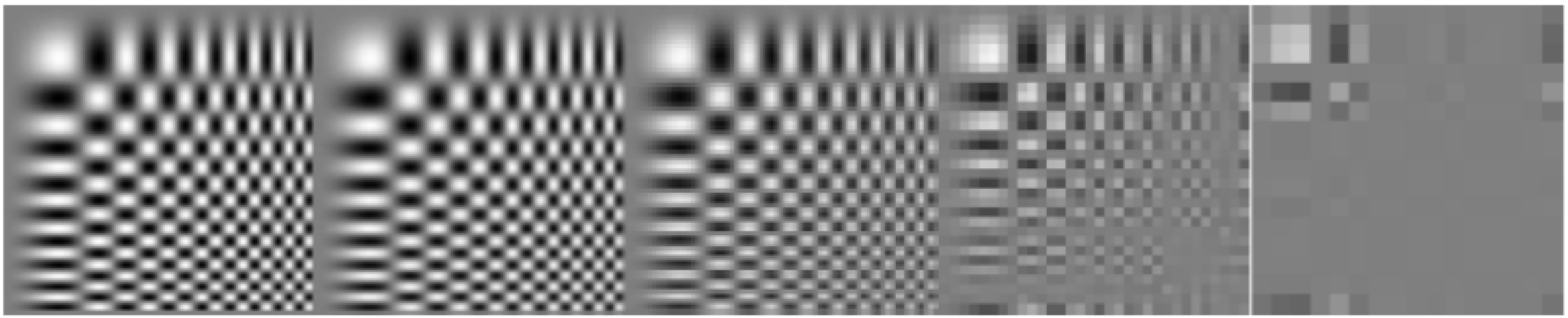


Anti-aliasing

256x256 128x128 64x64 32x32 16x16

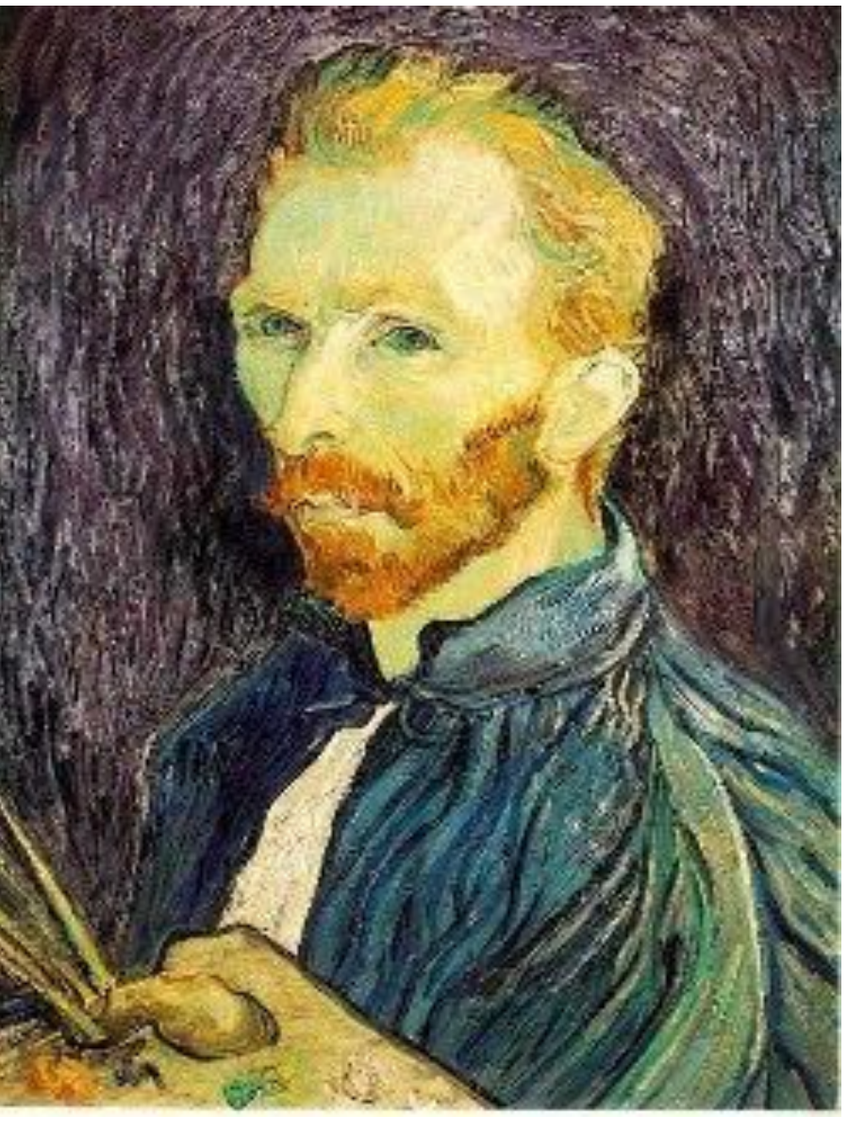


256x256 128x128 64x64 32x32 16x16

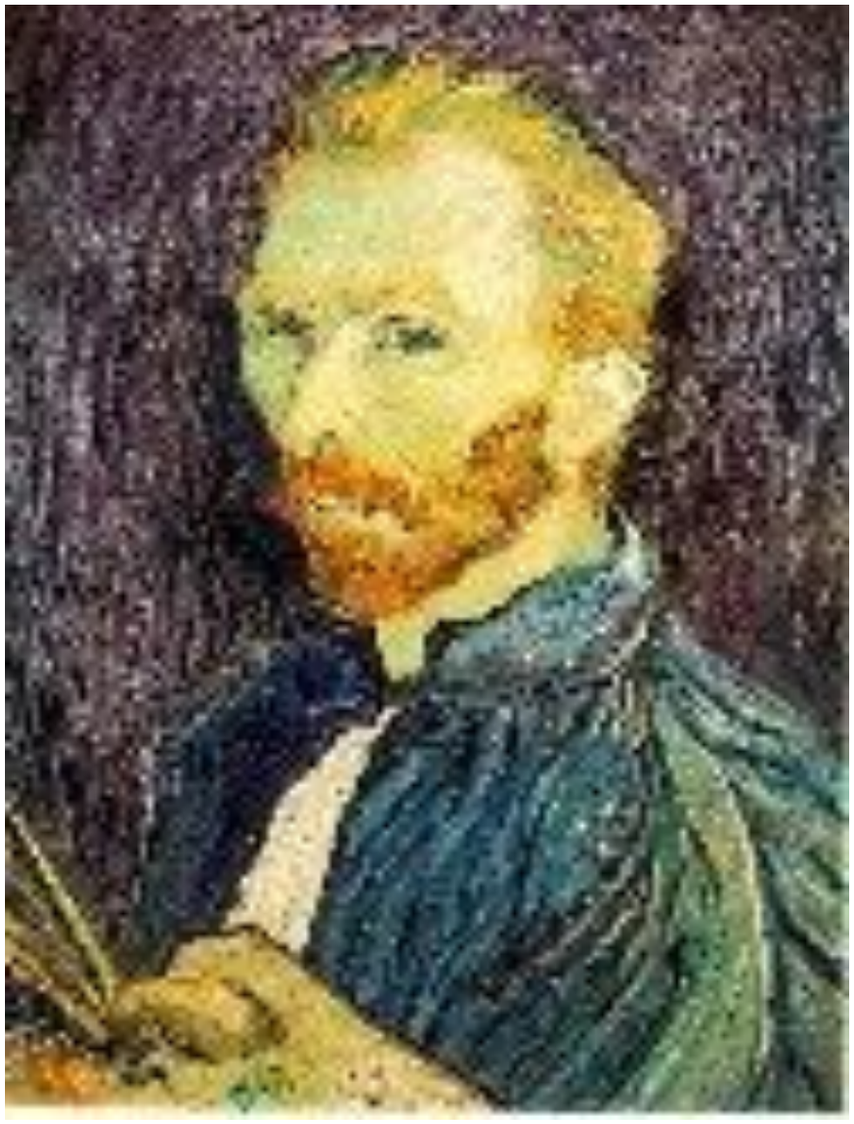


Forsyth and Ponce 2002

Subsampling without pre-filtering



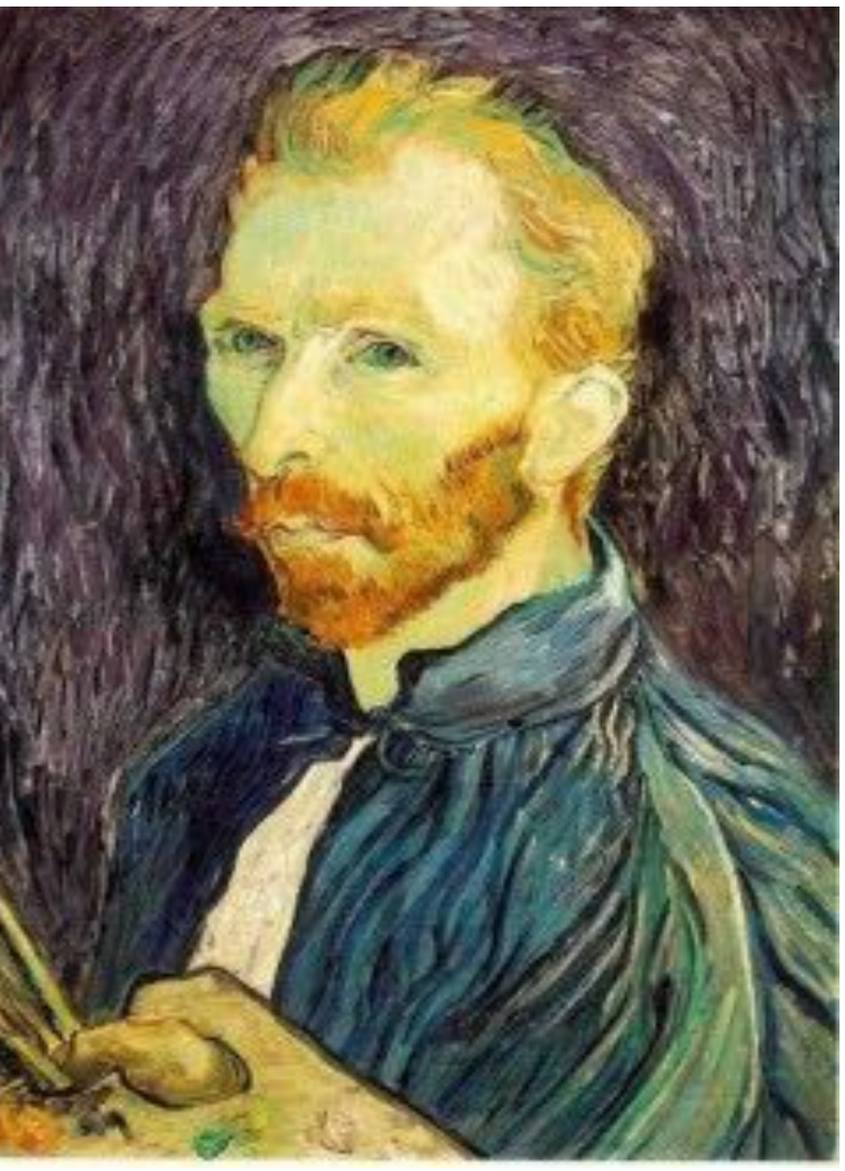
1/2



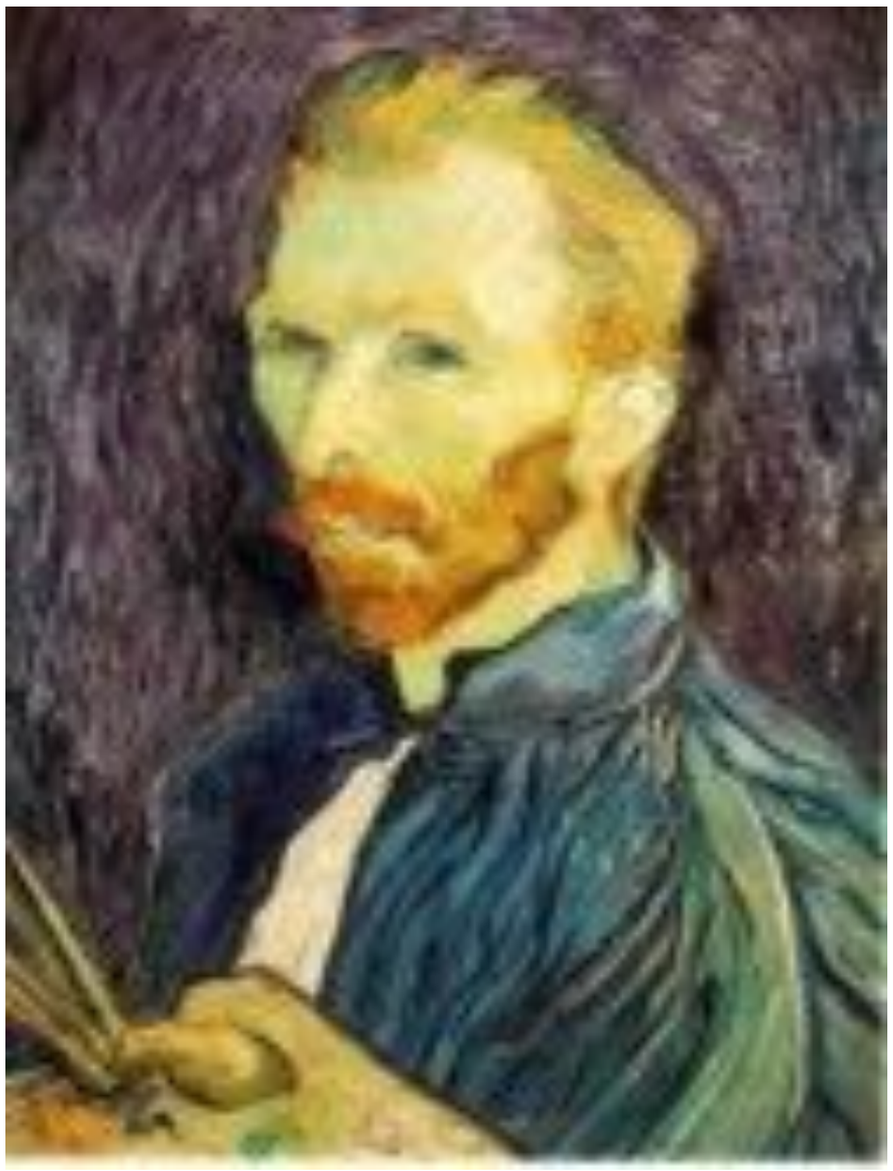
1/4 (2x zoom)

1/8 (4x zoom)

Subsampling with Gaussian pre-filtering



Gaussian 1/2



G 1/4

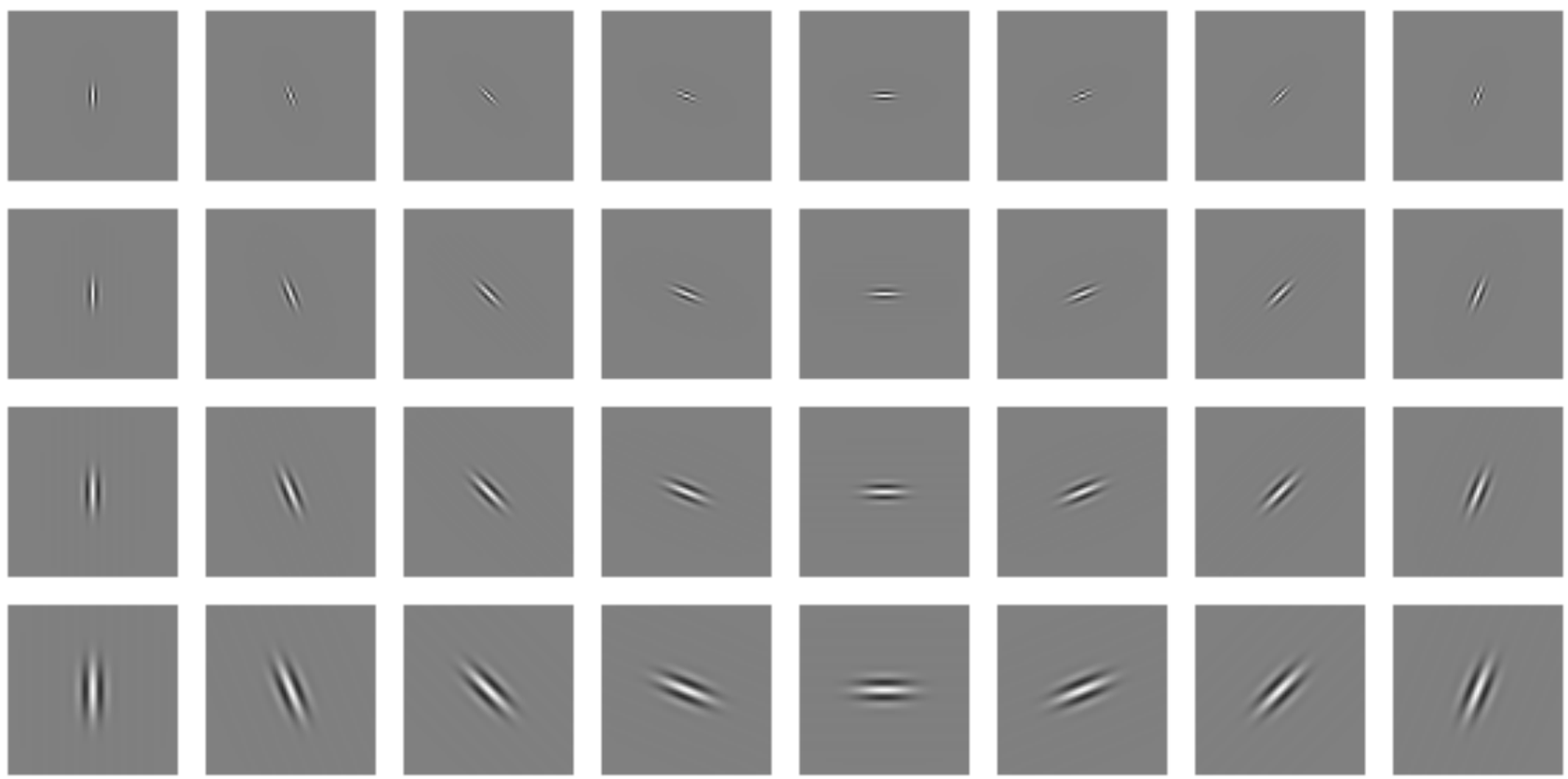
G 1/8

Slide by Steve Seitz

Always filtering and then sampling!

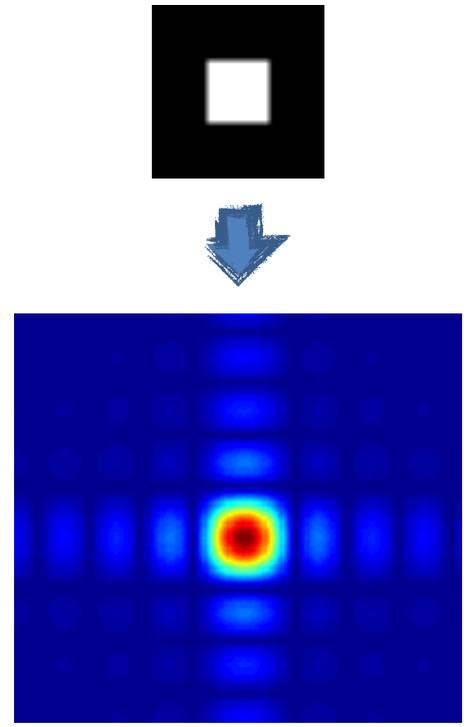
Subsampling with Gaussian pre-filtering

- Early processing in humans filters for various orientations and scales of frequency
- Perceptual cues in the mid-high frequencies dominate perception
- When we see an image from far away, we are effectively subsampling it



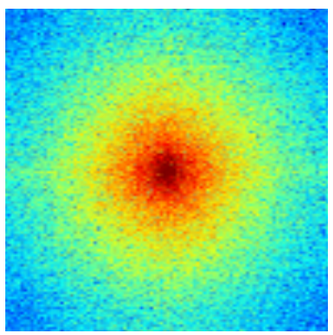
Early Visual Processing: Multi-scale edge and blob filters

Things to Remember



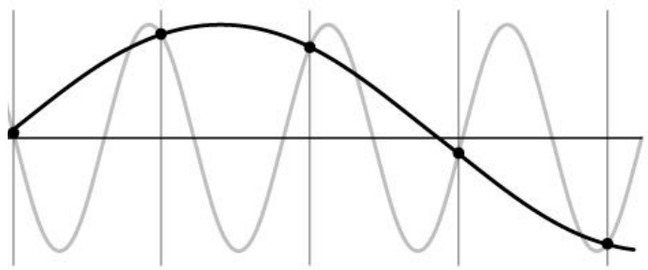
- Sometimes it makes sense to think of images and filtering in the frequency domain
  - Fourier analysis

- Can be faster to filter using FFT for large images (N logN vs. N^2)



- Images are mostly smooth
  - Basis for compression

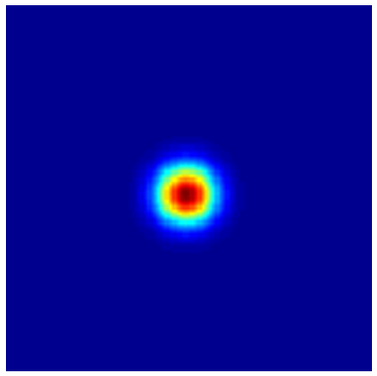
- Remember to low-pass before sampling



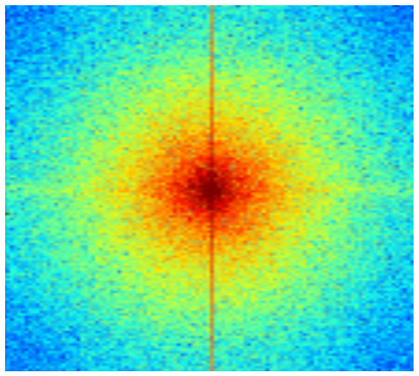
Practice question

Match the spatial domain image to the Fourier magnitude image

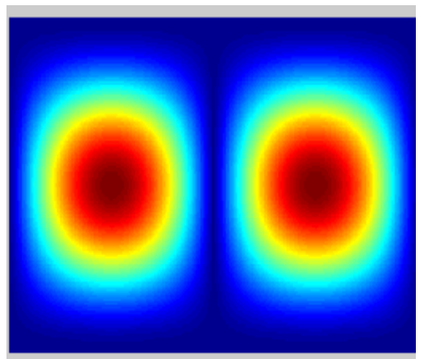
1



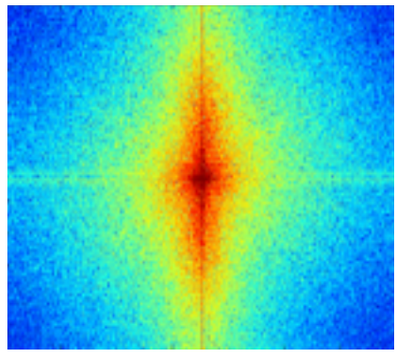
2



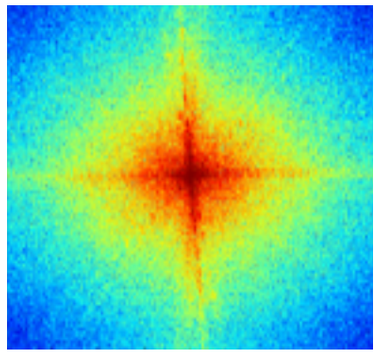
3



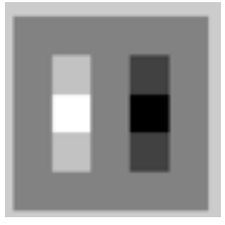
4



5



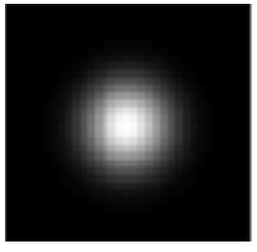
A



B

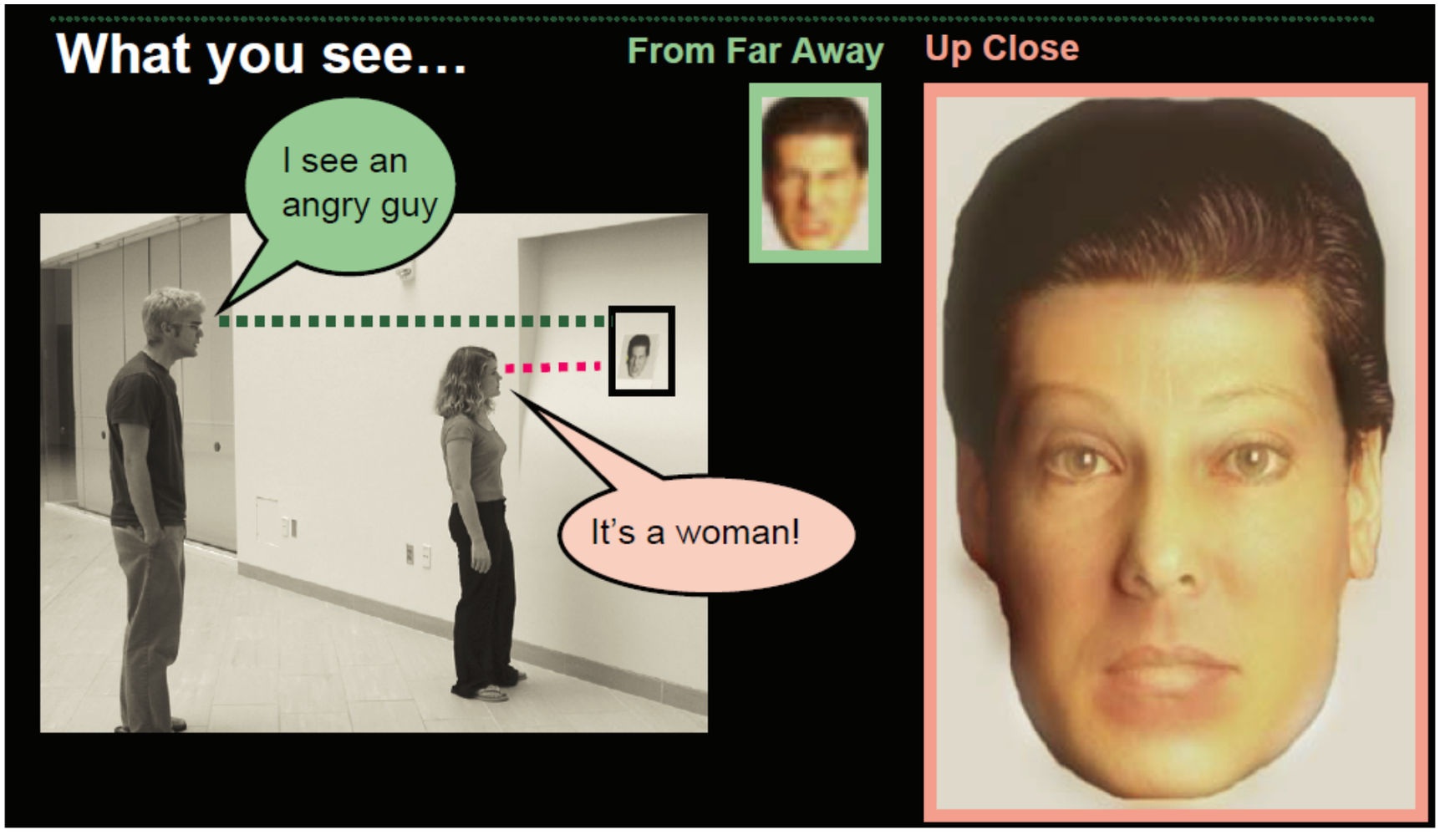
C

D



E

Application: hybrid image

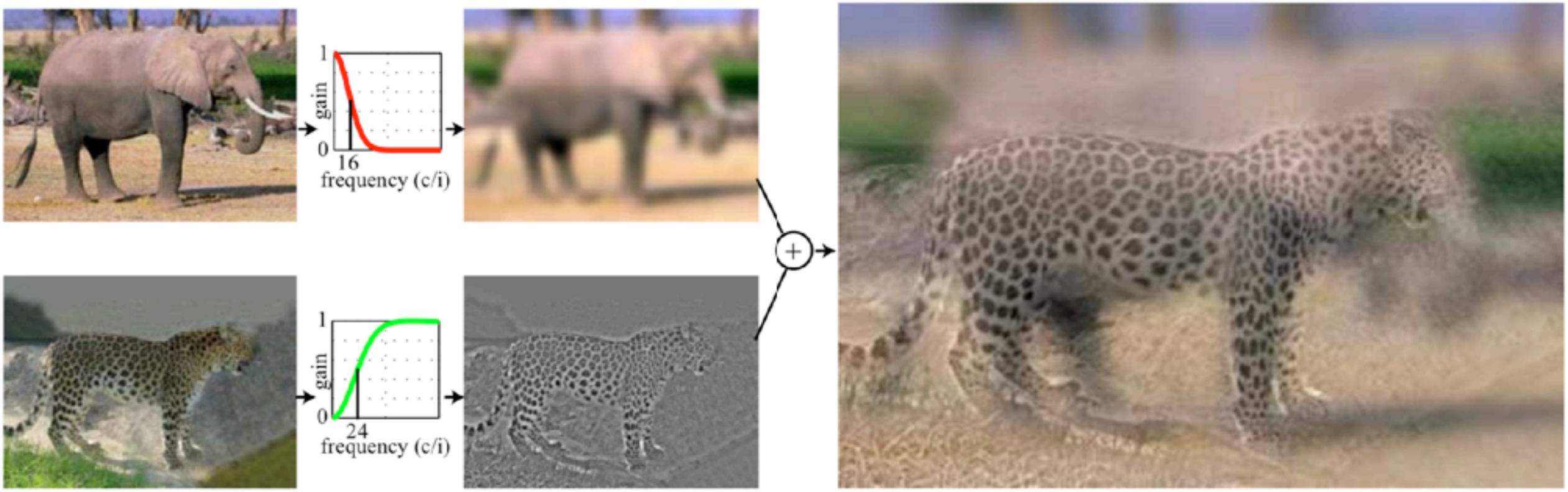


Oliva et al [SIGGRAPH 2016]

http://cvcl.mit.edu/publications/OlivaTorralb_Hybrid_Siggraph06.pdf

Hybrid image: main idea

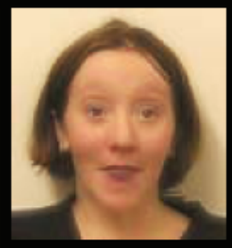
- Generated by superimposing two images at two different spatial scales
- The low-spatial scale is obtained by filtering one image with a low-pass filter
- The high spatial scale is obtained by filtering a second image with a high-pass filter
- The final hybrid image is composed by adding these two filtered images



Changing expression

SIGGRAPH2006

Sad ⟷ Surprised



Edge Detection

Edge detection

- Goal: map image from 2d array of pixels to a set of curves or line segments or contours.
- Why?

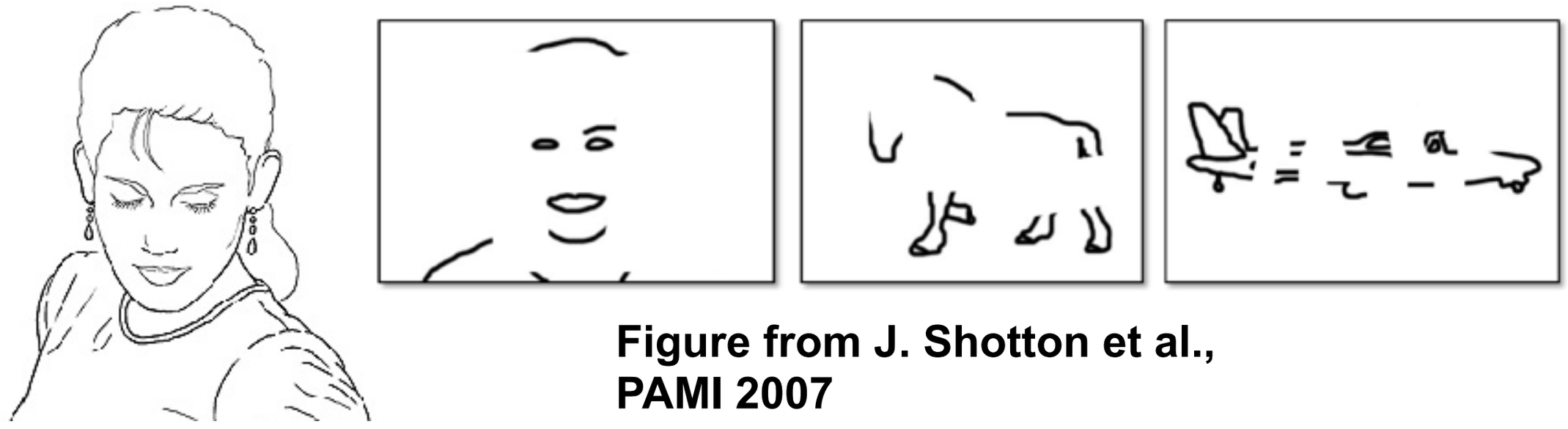


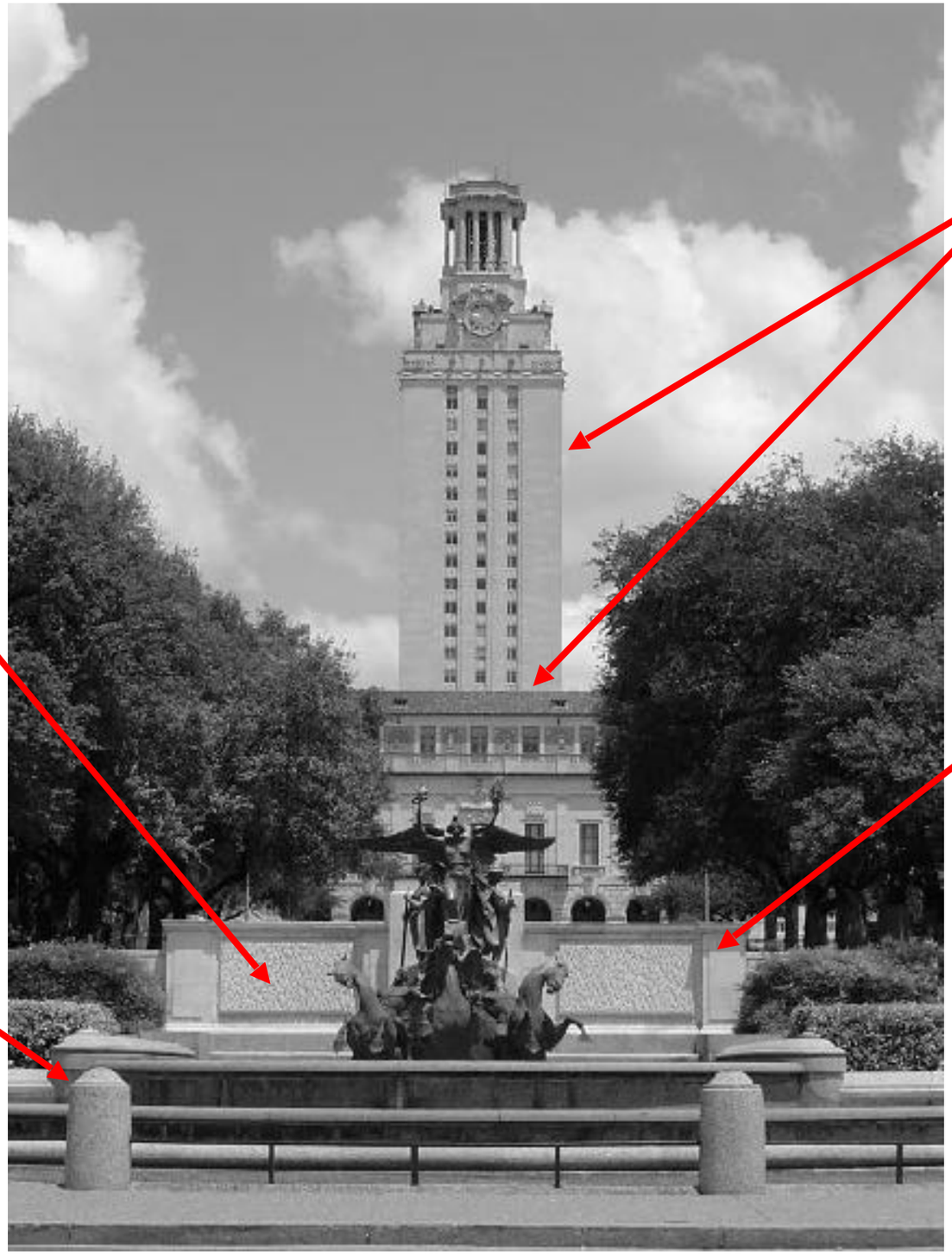
Figure from J. Shotton et al., PAMI 2007

- Main idea: look for strong gradients, post-process

What causes an edge?

Reflectance change: appearance information, texture

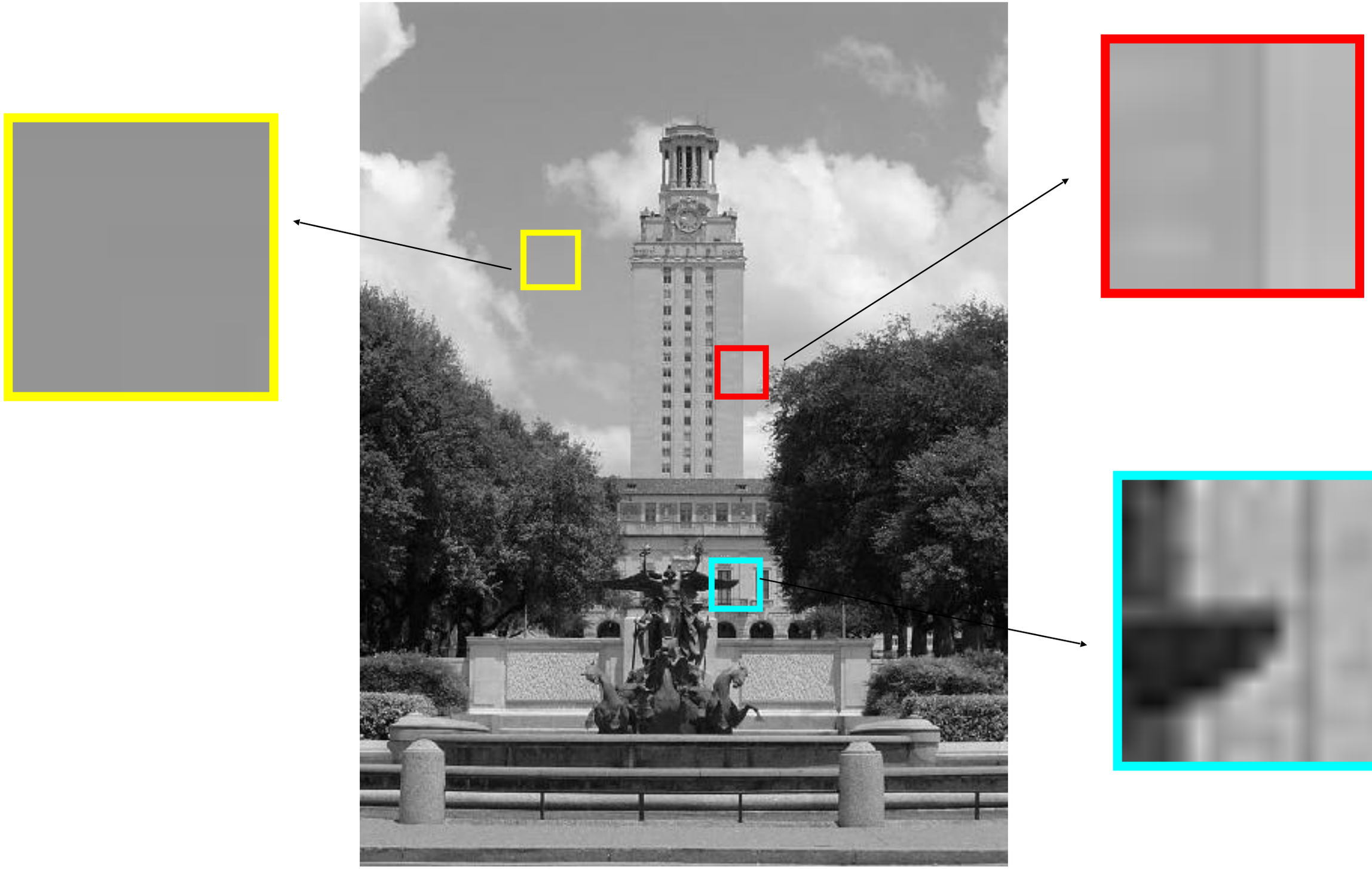
Depth discontinuity: object boundary



Change in surface orientation: shape

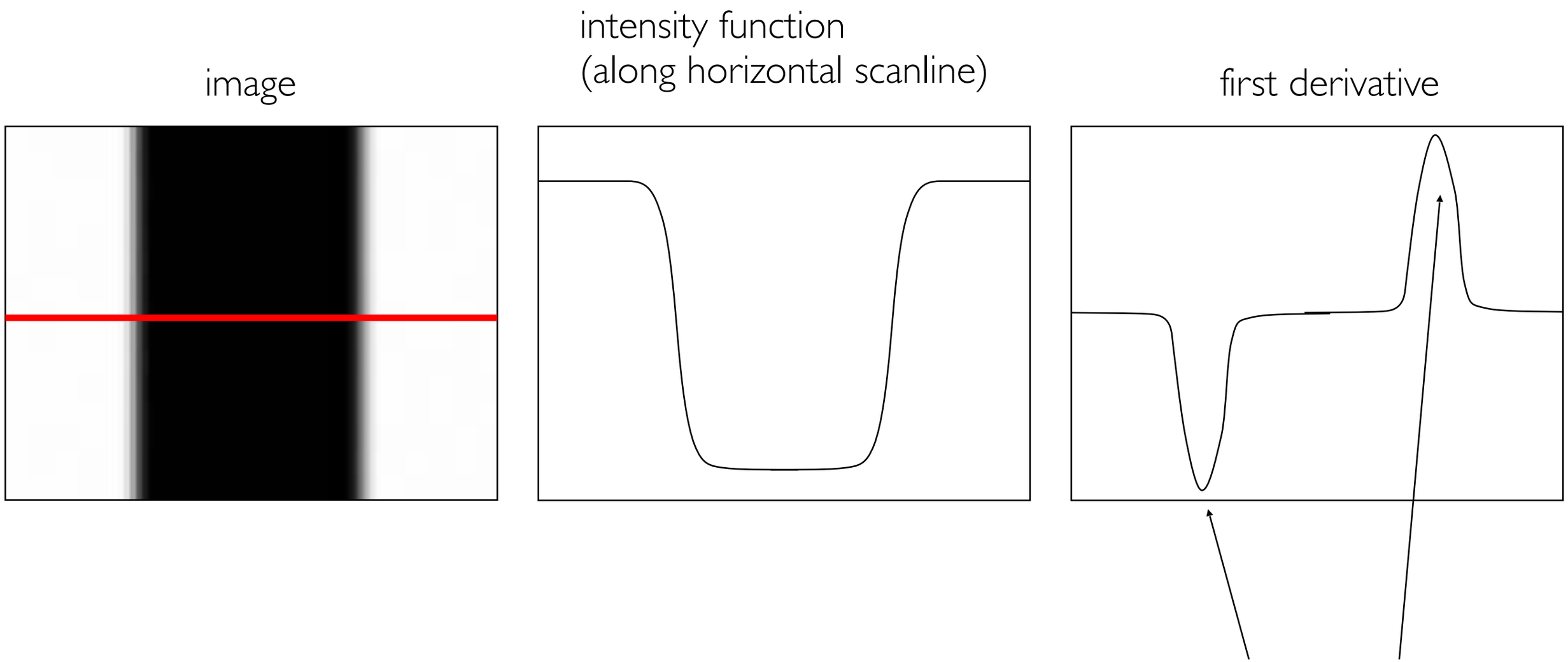
Cast shadows

Edges/gradients and invariance



Derivatives and edges

An edge is a place of rapid change in the image intensity function.



Derivatives with convolution

For 2D function, f(x,y), the partial derivative is:

$$\frac{\partial f(x,y)}{\partial x} = \lim_{\varepsilon \to 0} \frac{f(x+\varepsilon, y) - f(x,y)}{\varepsilon}$$

For discrete data, we can approximate using finite differences:

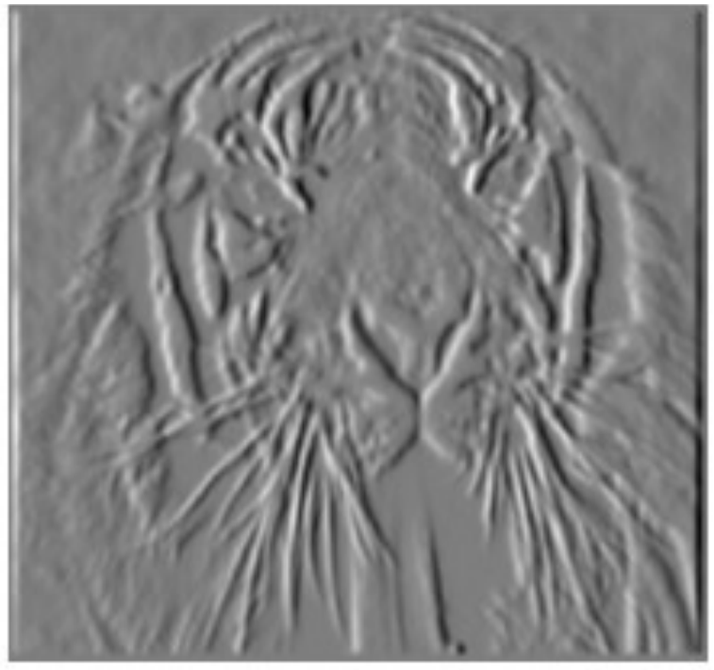
$$\frac{\partial f(x,y)}{\partial x} \approx \frac{f(x+1,y) - f(x,y)}{1}$$

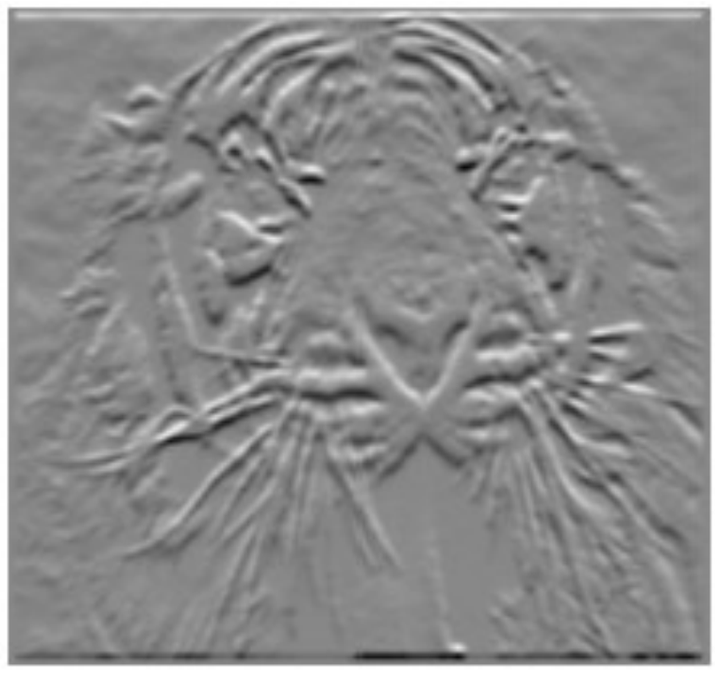
To implement above as convolution, what would be the associated filter?

Partial derivatives of an image

$\frac{\partial f(x,y)}{\partial x}$

-1	1





$\frac{\partial f(x,y)}{\partial y}$

-1
1

? **or**

1
-1

Which shows changes with respect to x?

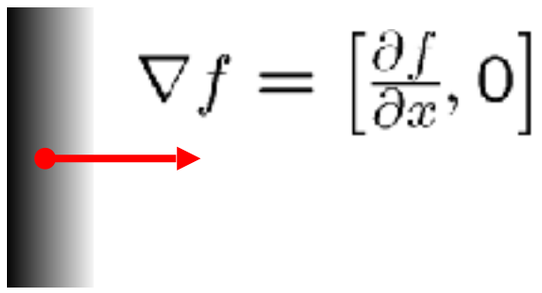
(showing filters for correlation)

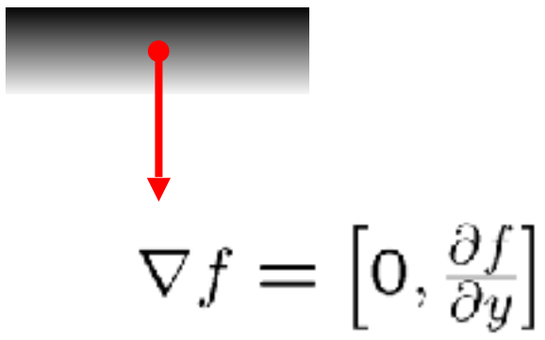
Image gradient

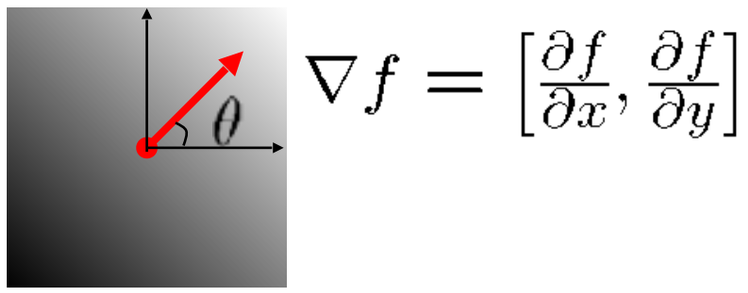
The gradient of an image:

$$\nabla f = \left[\frac{\partial f}{\partial x}, \frac{\partial f}{\partial y}\right]$$

The gradient points in the direction of most rapid change in intensity





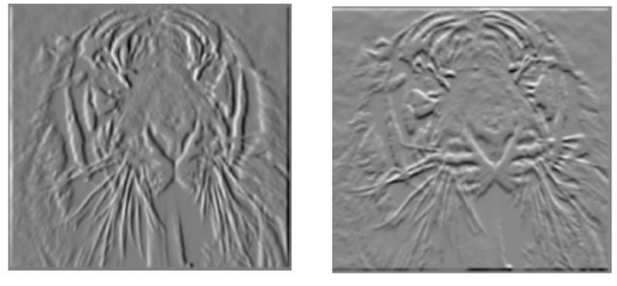


The gradient direction (orientation of edge normal) is given by:

$$\theta = \tan^{-1}\left(\frac{\partial f}{\partial y} / \frac{\partial f}{\partial x}\right)$$

The edge strength is given by the gradient magnitude

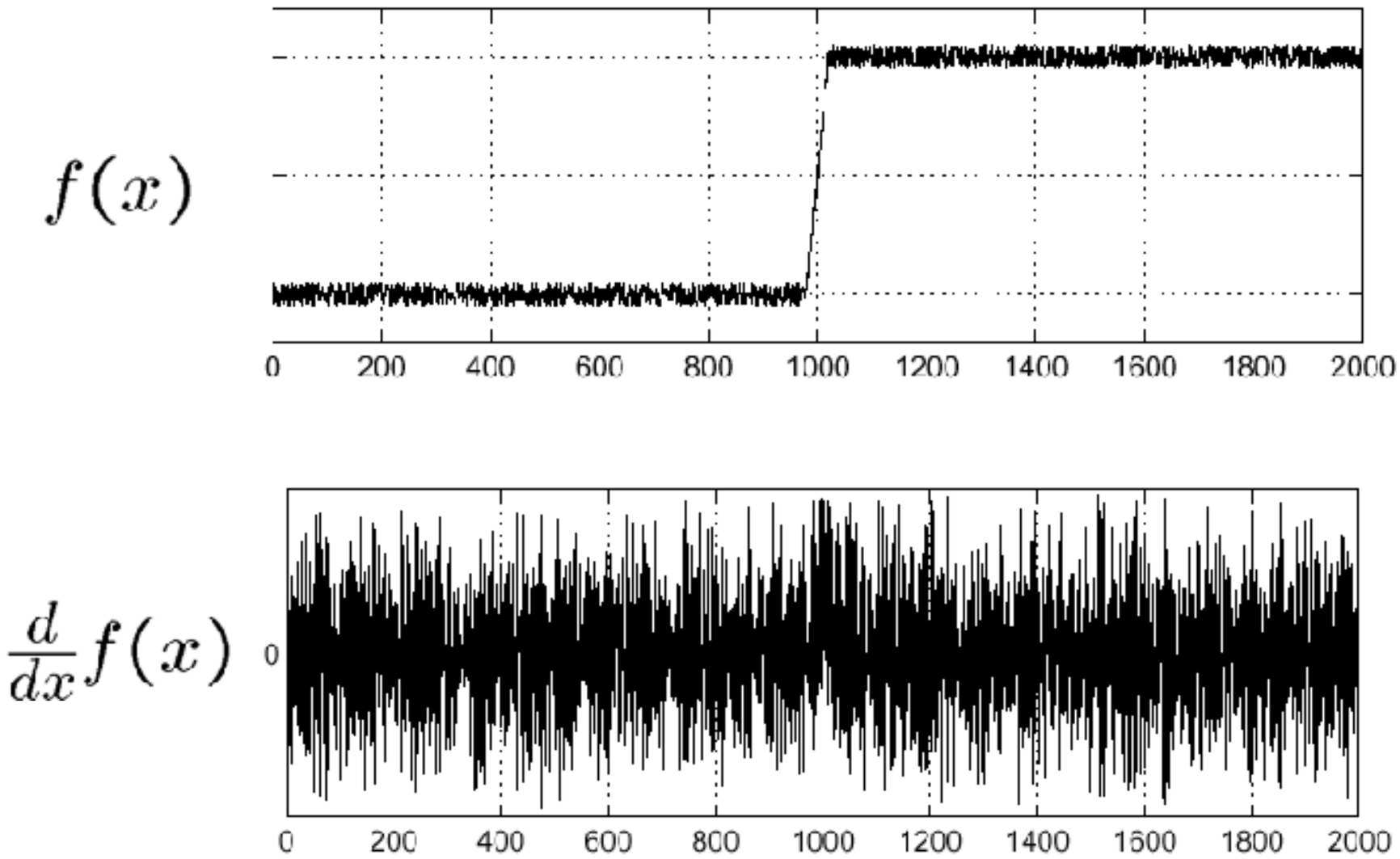
$$\|\nabla f\| = \sqrt{\left(\frac{\partial f}{\partial x}\right)^2 + \left(\frac{\partial f}{\partial y}\right)^2}$$



Effects of noise

Consider a single row or column of the image

- Plotting intensity as a function of position gives a signal

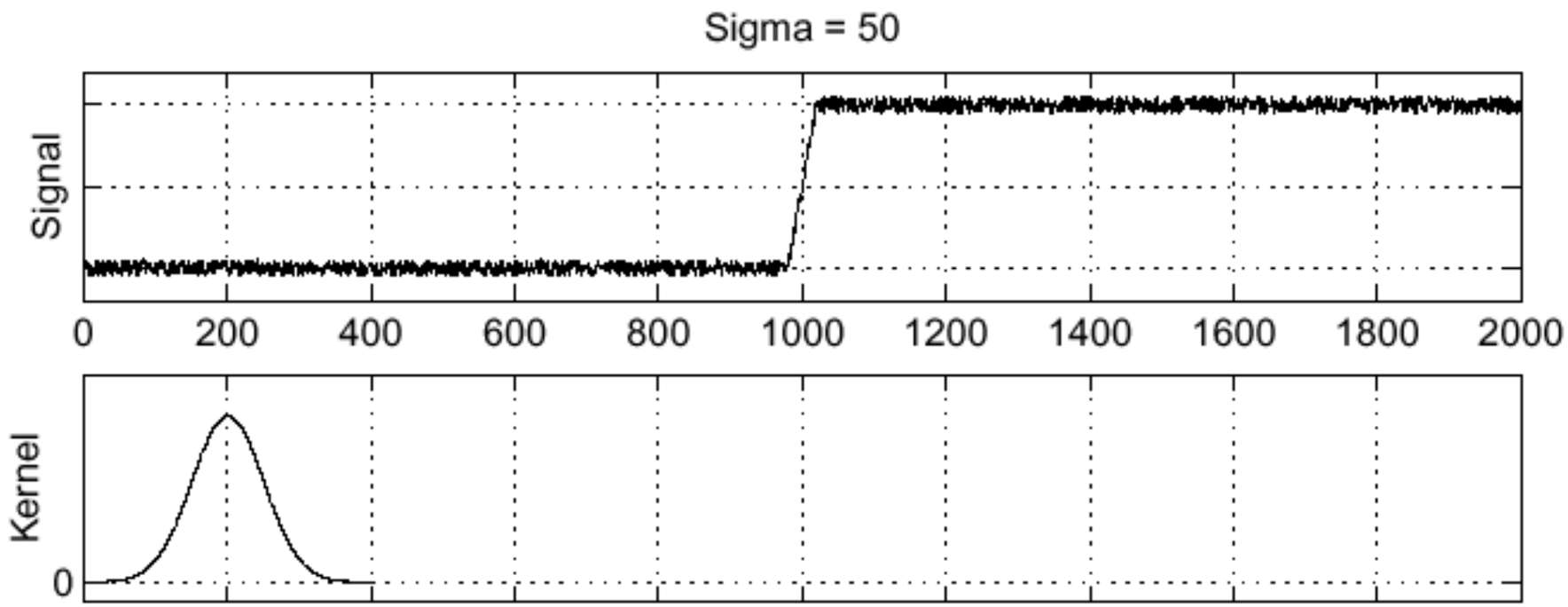


Where is the edge?

Solution: smooth first!

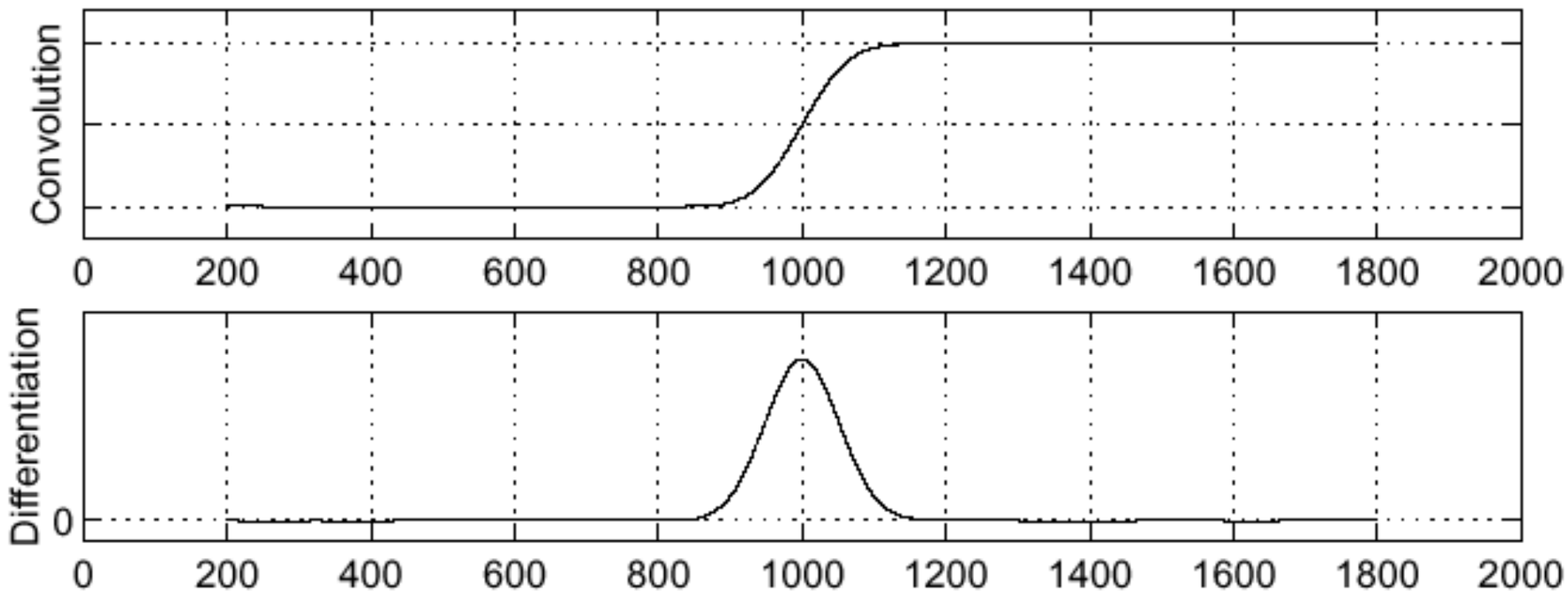
f

h



$h \star f$

$\frac{\partial}{\partial x}(h \star f)$



Where is the edge?

Look for peaks in $\frac{\partial}{\partial x}(h \star f)$

Derivative theorem of convolution

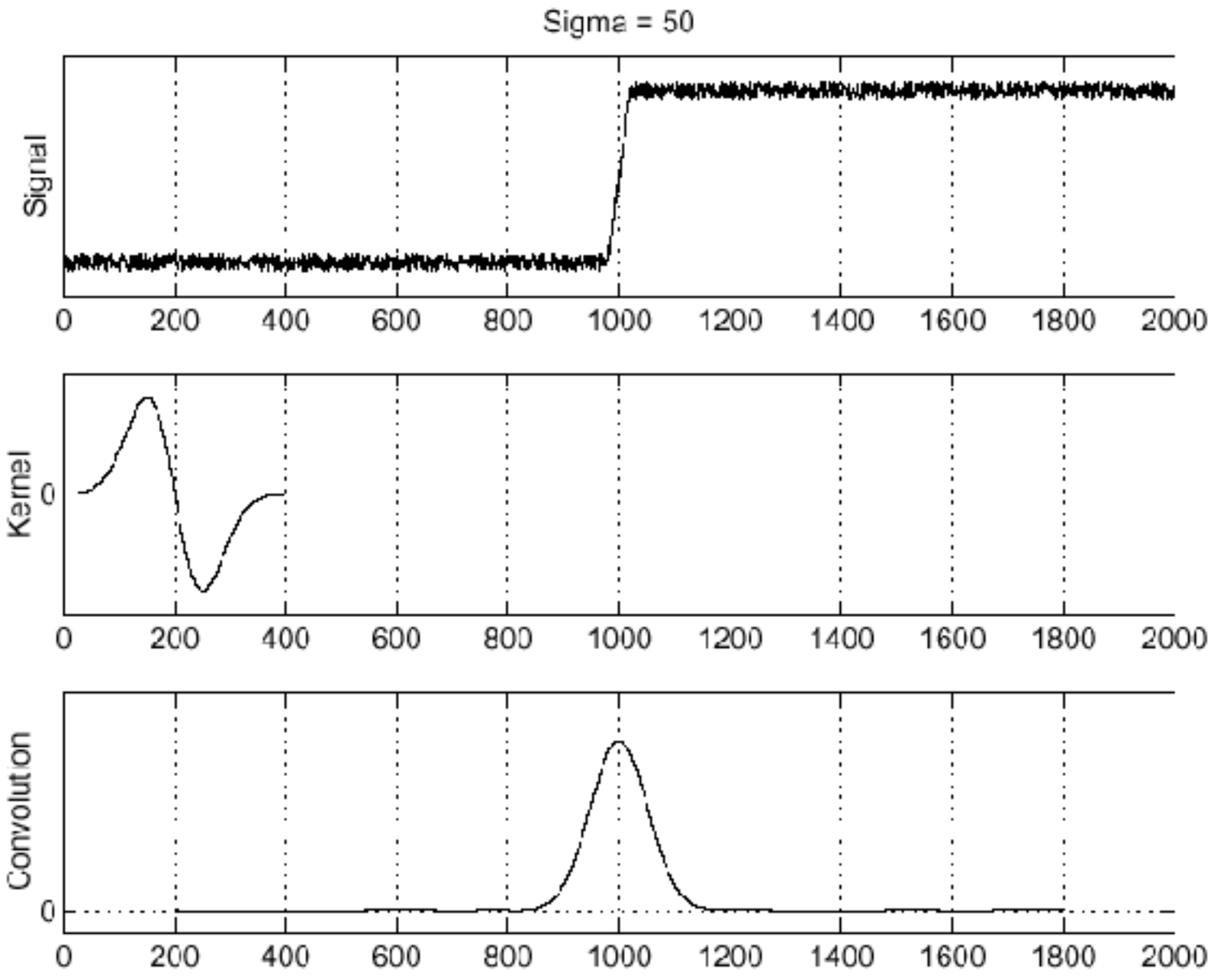
$$\frac{\partial}{\partial x}(h \star f) = (\frac{\partial}{\partial x}h) \star f$$

Differentiation property of convolution

f

$\frac{\partial}{\partial x}h$

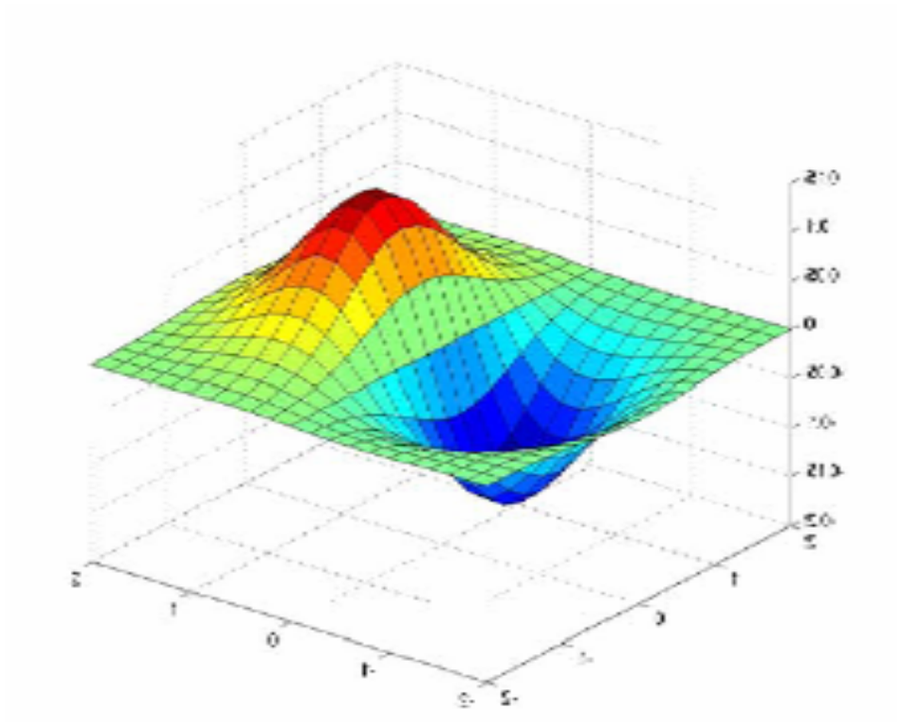
$(\frac{\partial}{\partial x}h) \star f$



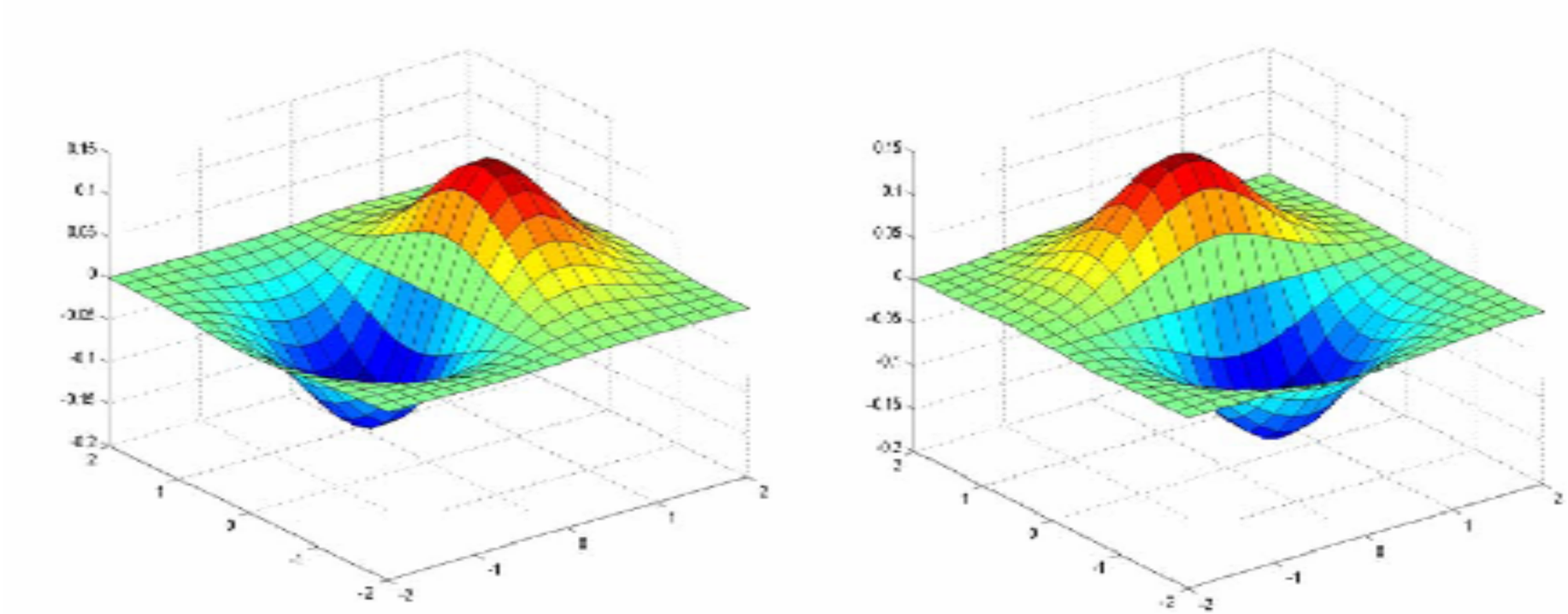
Derivative of Gaussian filters

$$(I \otimes g) \otimes h = I \otimes (g \otimes h)$$

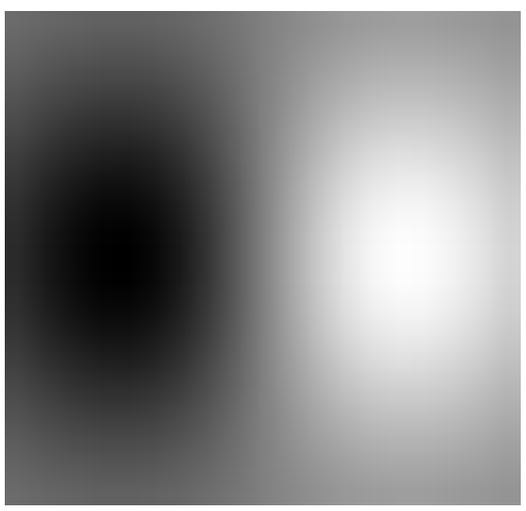
$$\begin{bmatrix} 0.0030 & 0.0133 & 0.0219 & 0.0133 & 0.0030 \\ 0.0133 & 0.0596 & 0.0983 & 0.0596 & 0.0133 \\ 0.0219 & 0.0983 & 0.1621 & 0.0983 & 0.0219 \\ 0.0133 & 0.0596 & 0.0983 & 0.0596 & 0.0133 \\ 0.0030 & 0.0133 & 0.0219 & 0.0133 & 0.0030 \end{bmatrix} \otimes \begin{bmatrix} 1 & -1 \end{bmatrix}$$



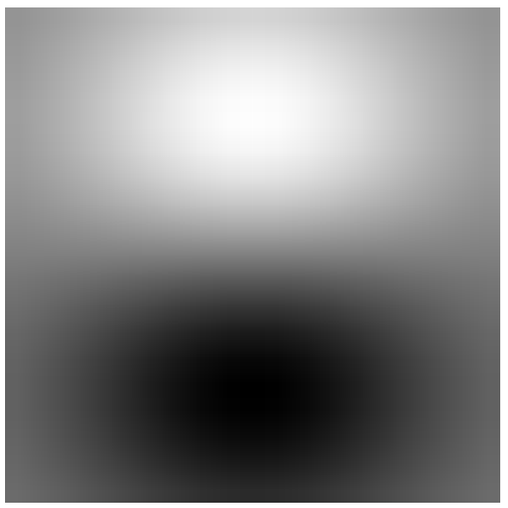
Derivative of Gaussian filters



***x*-direction**

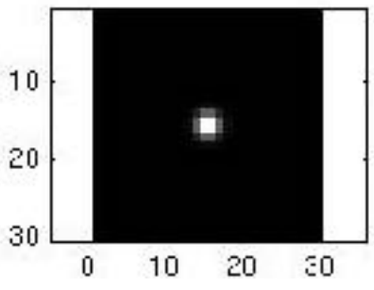


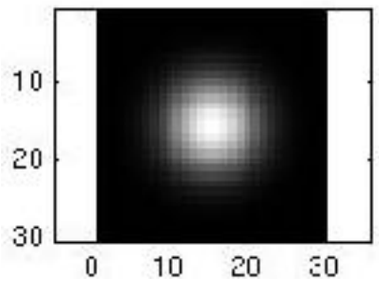
***y*-direction**

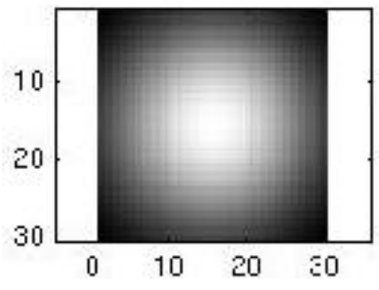


Smoothing with a Gaussian

Recall: parameter σ is the "scale" / "width" / "spread" of the Gaussian kernel, and controls the amount of smoothing.







Effect of σ on derivatives

σ = 1 pixel

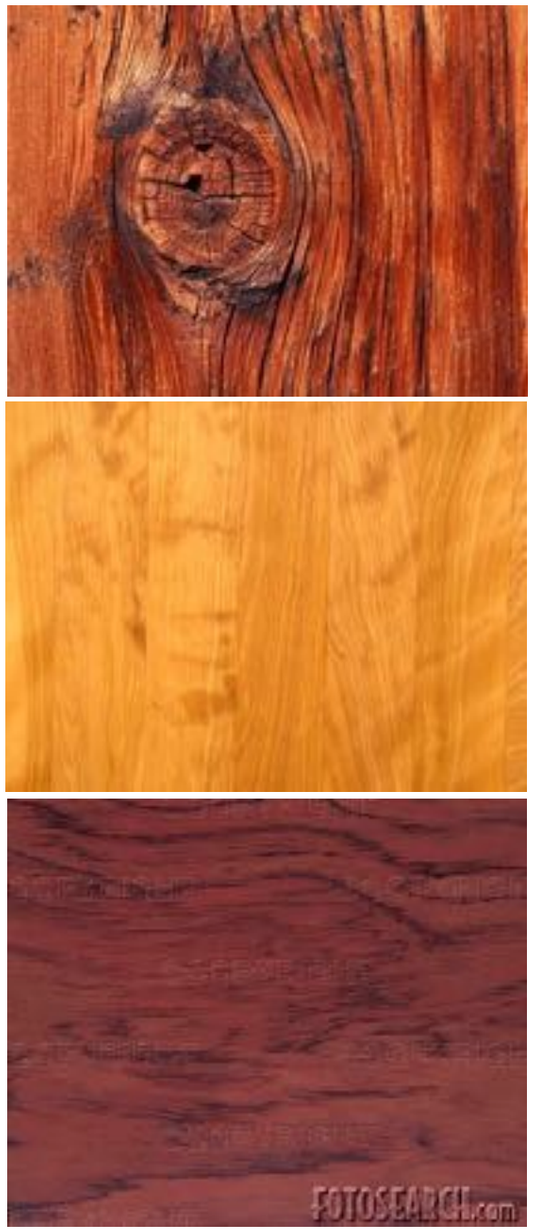
σ = 3 pixels

The apparent structures differ depending on Gaussian's scale parameter.

Larger values: larger scale edges detected
Smaller values: finer features detected

So, what scale to choose?

It depends on what we're looking for!



Gradients —> Edges

Primary edge detection steps:

1. Smoothing: suppress noise
2. Edge enhancement: filter for contrast
3. Edge localization

   Determine which local maxima from filter output are actually edges vs. noise

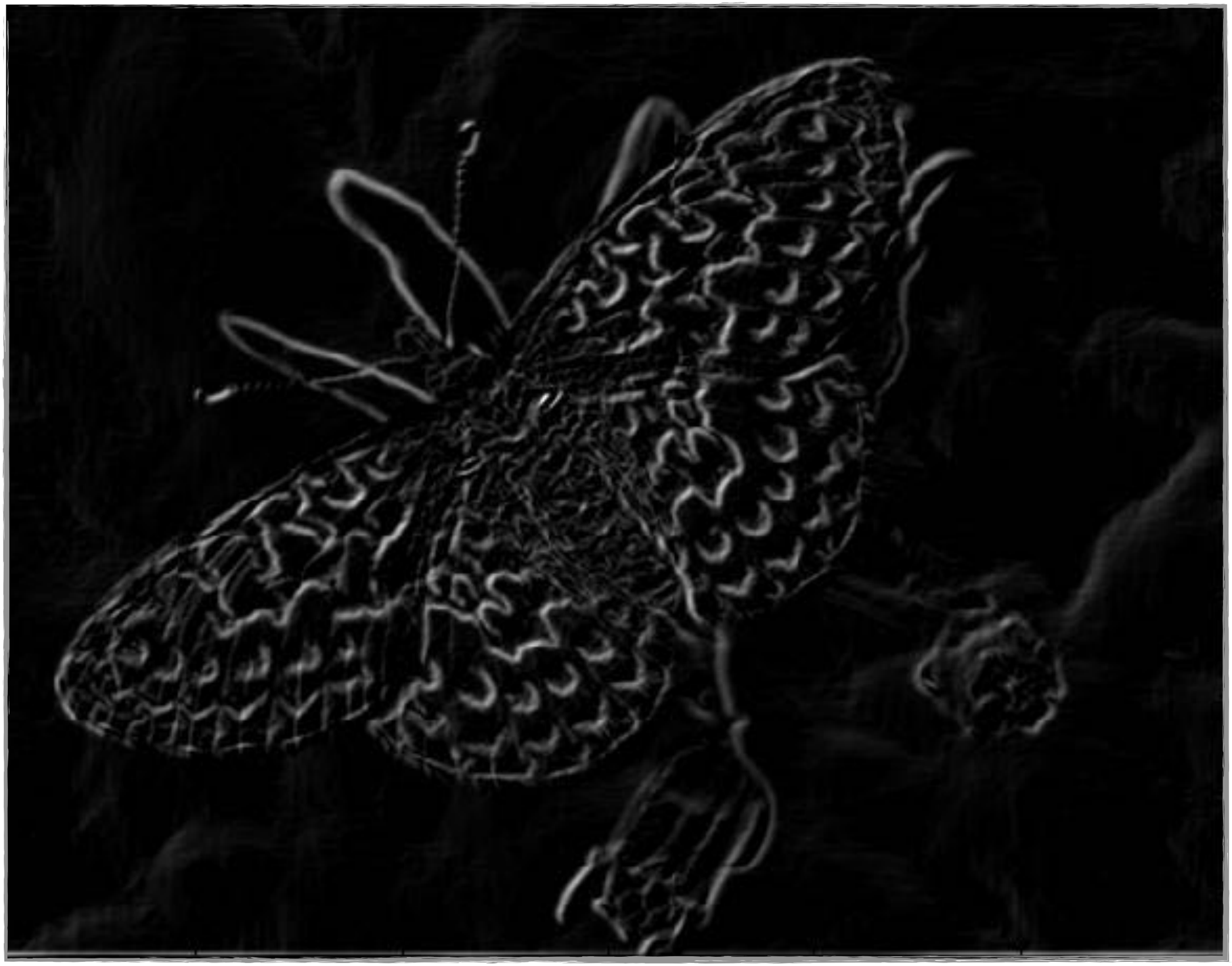
   - Threshold, Thin

Thresholding

- Choose a threshold value t
- Set any pixels less than t to zero (off)
- Set any pixels greater than or equal to t to one (on)

Original image

Gradient magnitude image



Thresholding gradient with a lower threshold

Thresholding gradient with a higher threshold

Canny edge detector

- Filter image with derivative of Gaussian
- Find magnitude and orientation of gradient
- Non-maximum suppression:
  - Thin wide "ridges" down to single pixel width
- Linking and thresholding (hysteresis):
  - Define two thresholds: low and high
  - Use the high threshold to start edge curves and the low threshold to continue them

The Canny edge detector

original image (Lena)

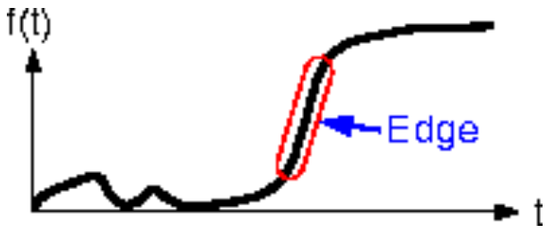
The Canny edge detector

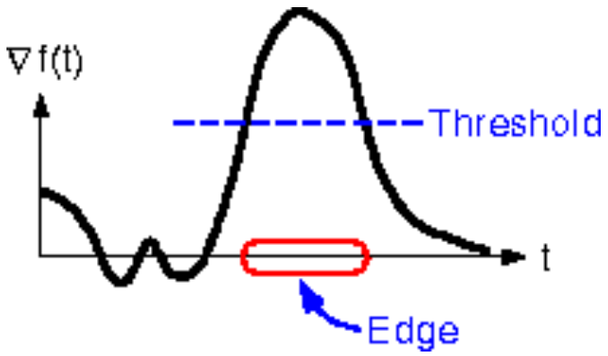
norm of the gradient

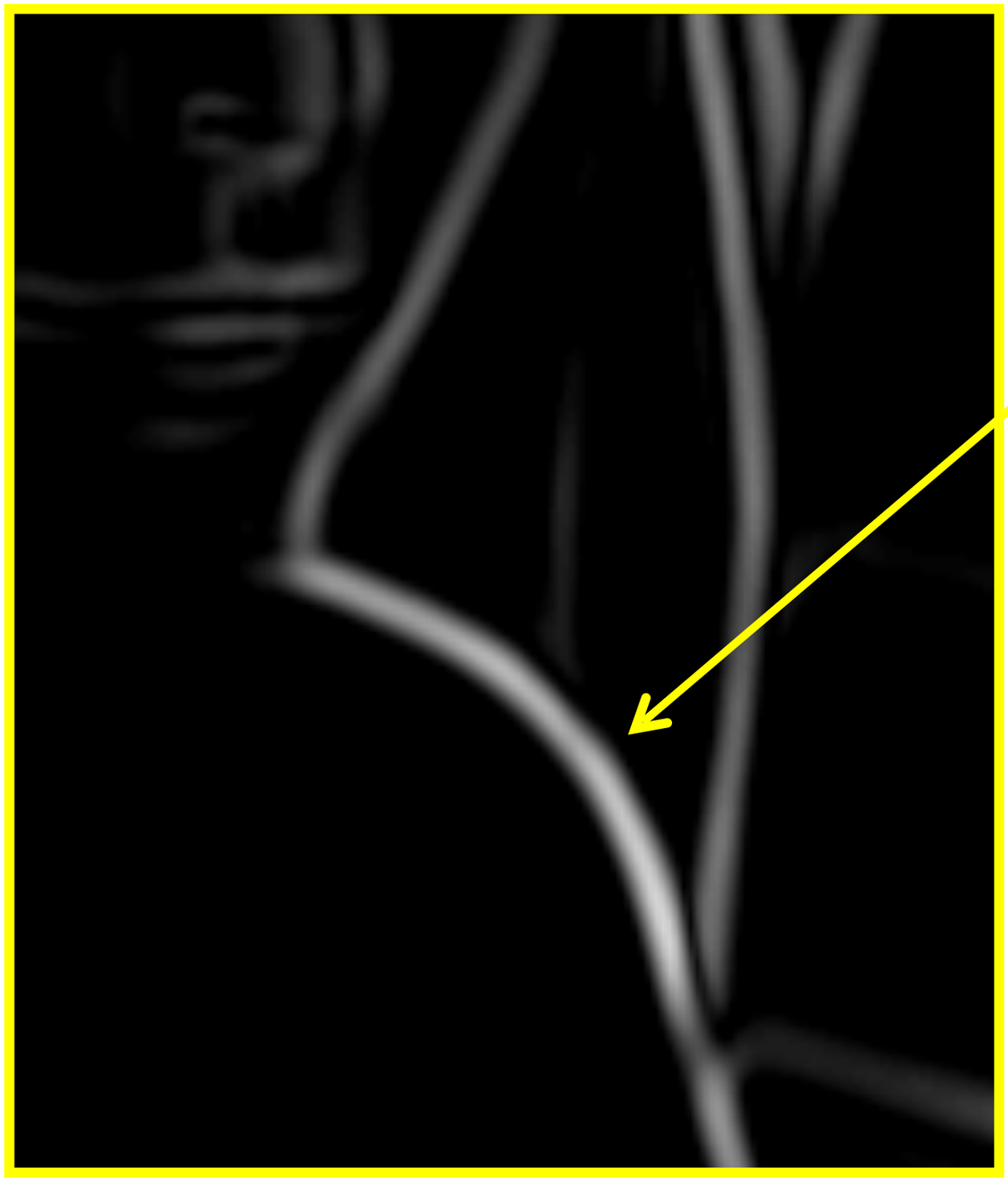
The Canny edge detector

thresholding

The Canny edge detector

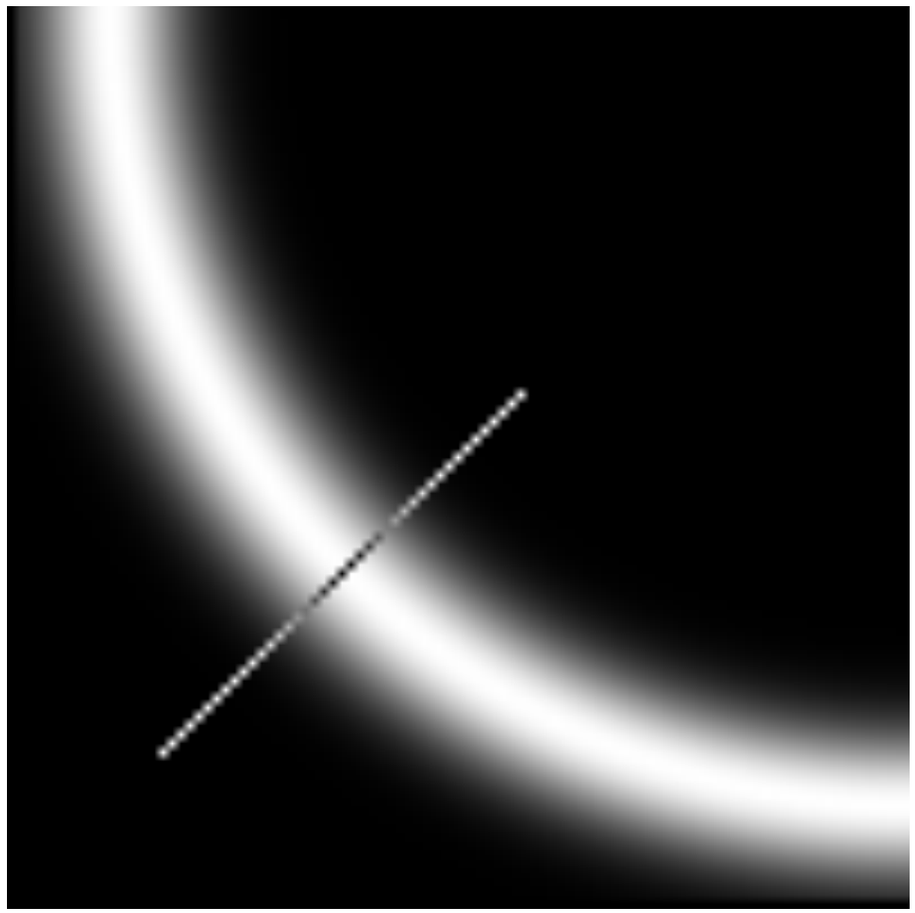


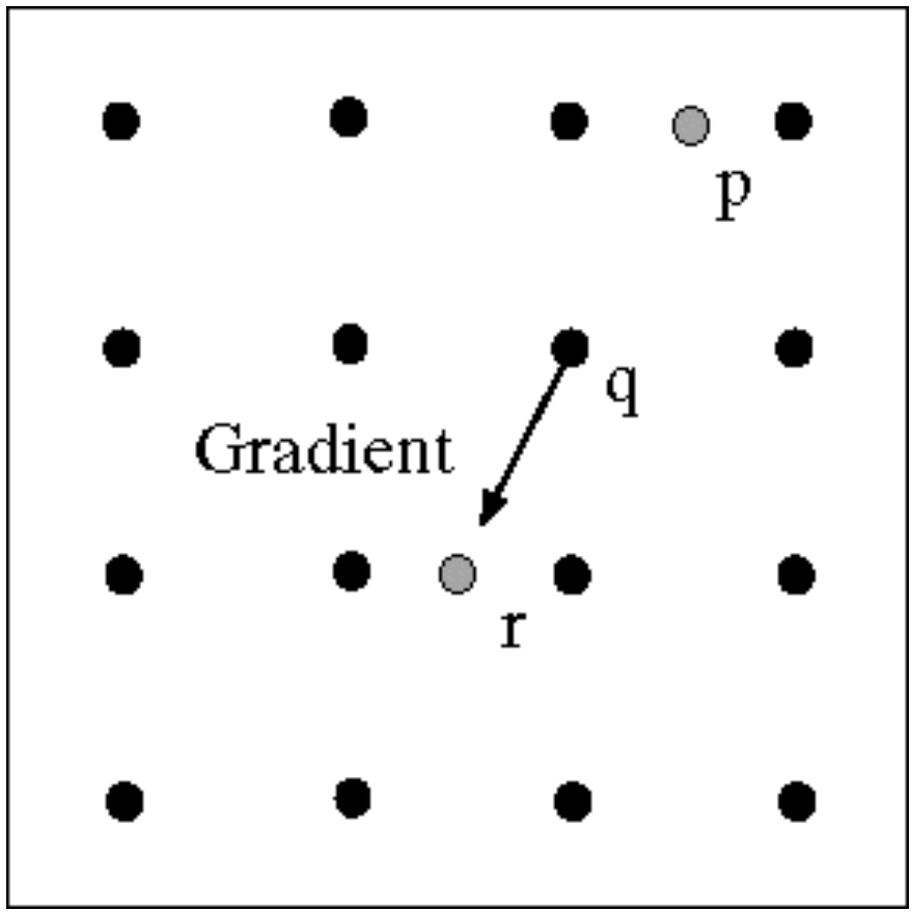




How to turn these thick regions of the gradient into curves?

Non-maximum suppression





Check if pixel is local maximum along gradient direction, select single max across width of the edge

- requires checking interpolated pixels p and r

The Canny edge detector

Problem: pixels along this edge didn't survive the thresholding

thinning

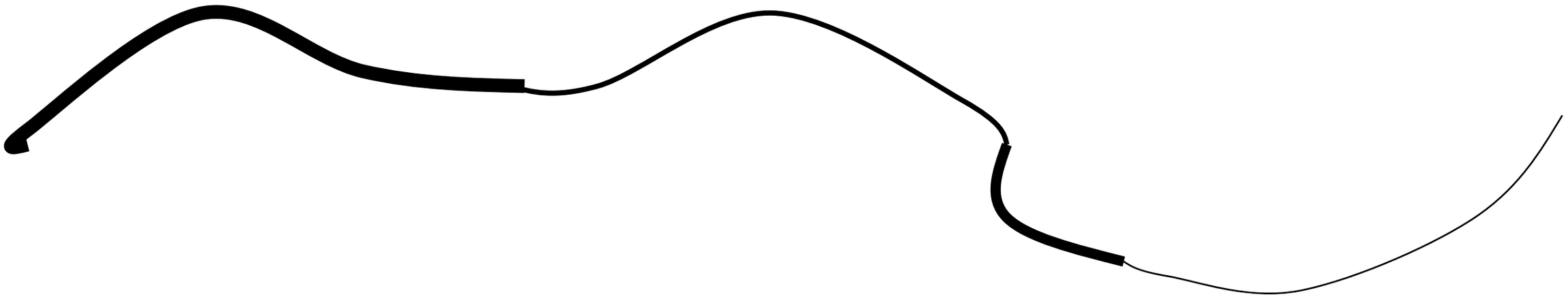
(non-maximum suppression)

Hysteresis thresholding

- Threshold at low/high levels to get weak/strong edge pixels
- Do connected components, starting from strong edge pixels

Hysteresis thresholding

- Use a high threshold to start edge curves, and a low threshold to continue them.

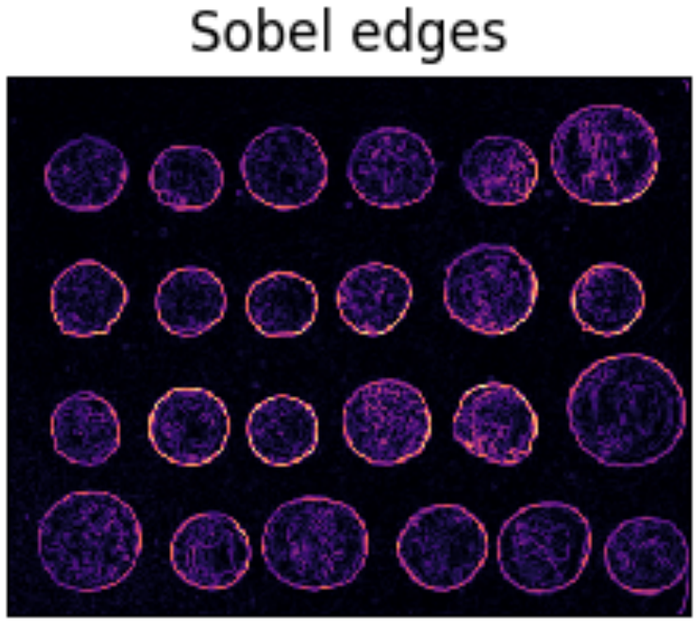


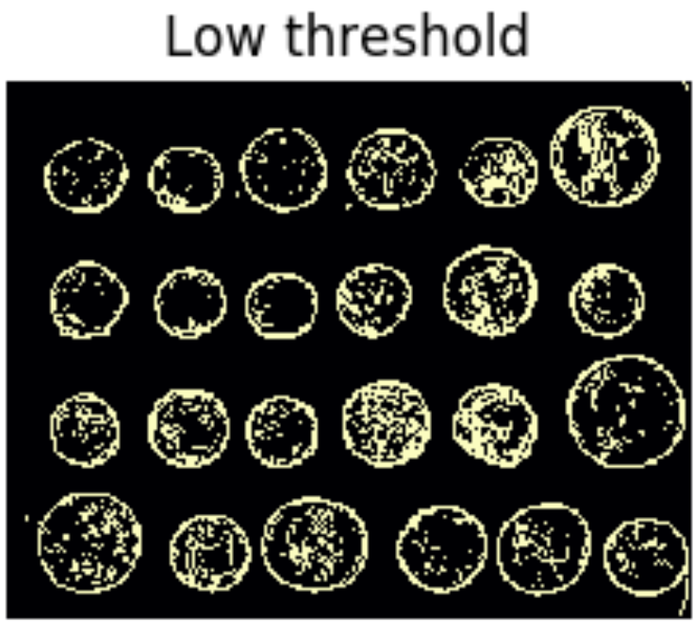
Canny edge output

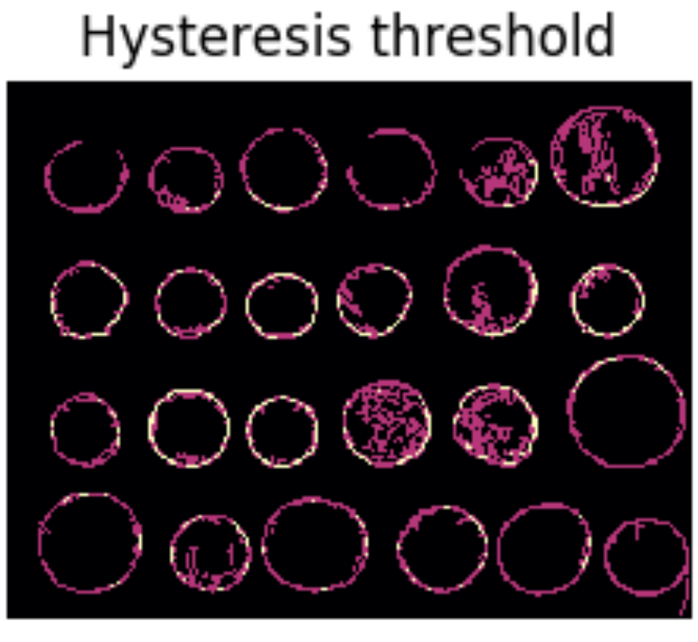
Credit: James Hays

Try it yourself?

Hysteresis detects ‘real’ edges







Low-level edges vs. perceived contours

Background

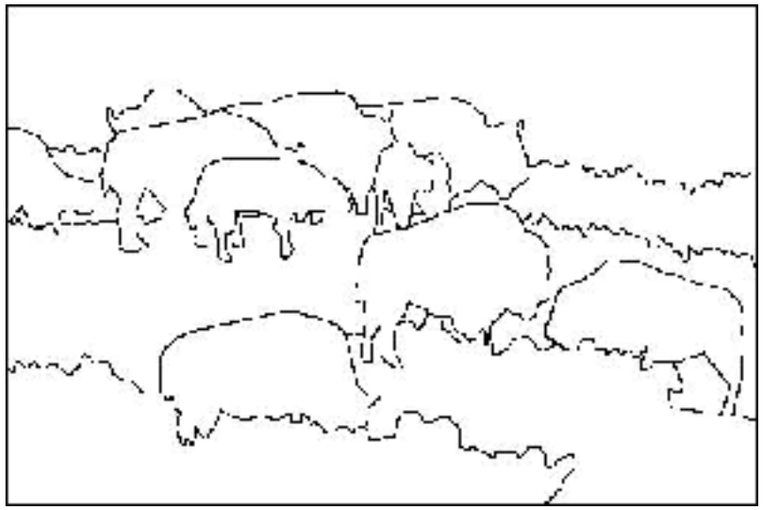
Texture

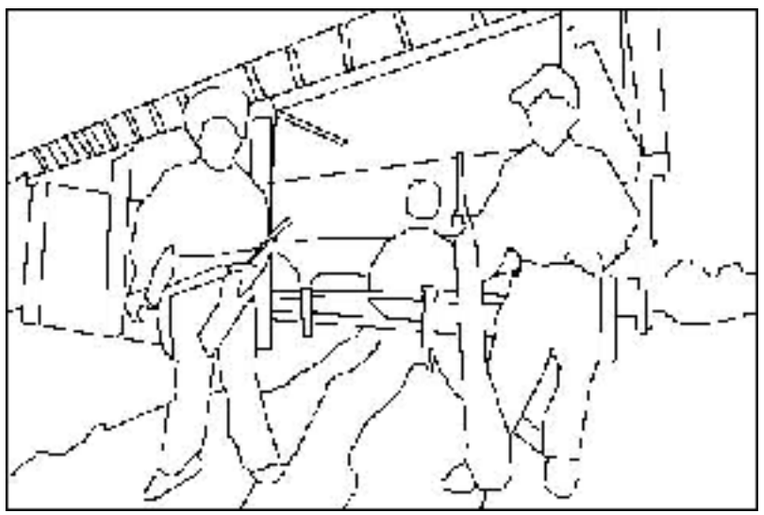
Shadows

Low-level edges vs. perceived contours

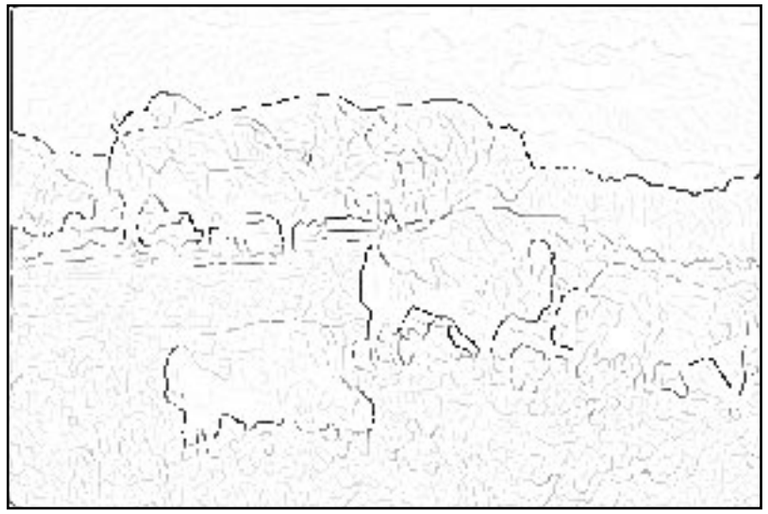
image

human segmentation





gradient magnitude



Seam carving: main idea

[Shai & Avidan, SIGGRAPH 2007]

Seam carving: main idea

Content-aware resizing

Traditional resizing

[Shai & Avidan, SIGGRAPH 2007]

Slide credit: Kristen Grauman

Seam carving: main idea

Content-aware resizing

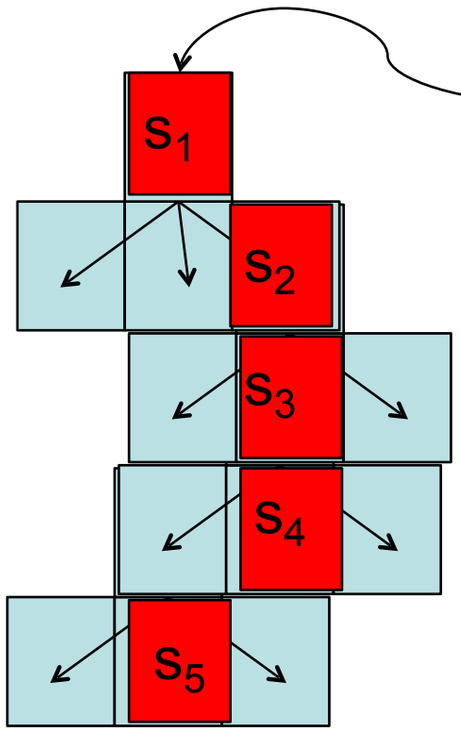
Intuition:

- Preserve the most "interesting" content
  - Prefer to remove pixels with low gradient energy
- To reduce or increase size in one dimension, remove irregularly shaped "seams"
  - Optimal solution via dynamic programming.

Seam carving: main idea

$$Energy(f) = \sqrt{(\frac{\partial f}{\partial x})^2 + (\frac{\partial f}{\partial y})^2}$$

Seam carving: algorithm



$$Energy(f) = \sqrt{\left(\frac{\partial f}{\partial x}\right)^2 + \left(\frac{\partial f}{\partial y}\right)^2}$$

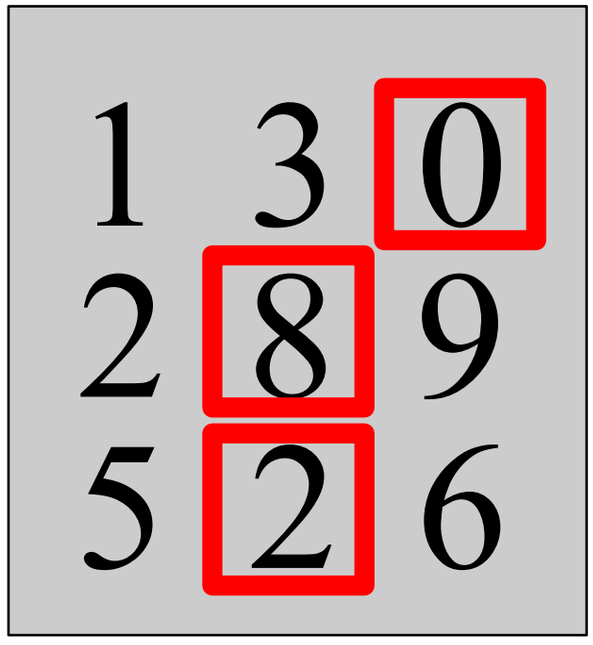
$$Cost(\mathbf{s}) = \sum_{i=1}^{h} Energy(f(s_i))$$

$$\mathbf{s}^* = \min_{\mathbf{s}} Cost(\mathbf{s})$$

Slide credit: Kristen Grauman

How to identify the minimum cost seam?

- First, consider a greedy approach:



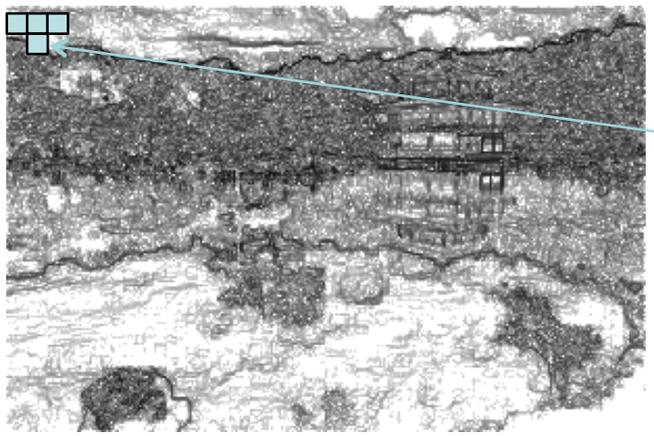
Energy matrix
(gradient magnitude)

Slide credit: Kristen Grauman

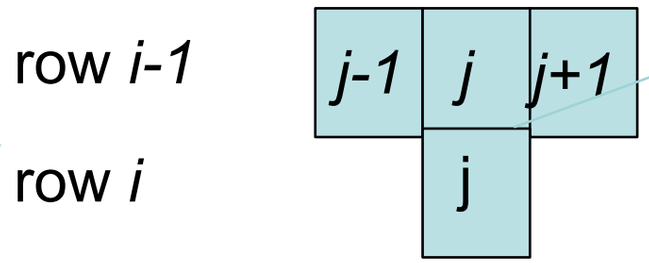
Seam carving: algorithm

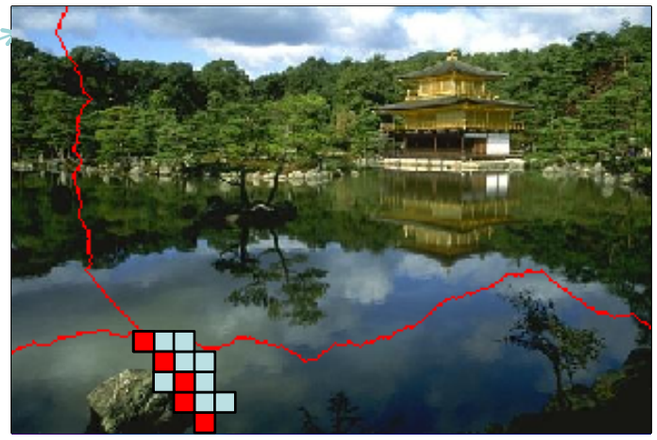
- Compute the cumulative minimum energy for all possible connected seams at each entry (i, j):

$$\mathbf{M}(i,j) = Energy(i,j) + \min\left(\mathbf{M}(i-1,j-1), \mathbf{M}(i-1,j), \mathbf{M}(i-1,j+1)\right)$$



Energy matrix
(gradient magnitude)





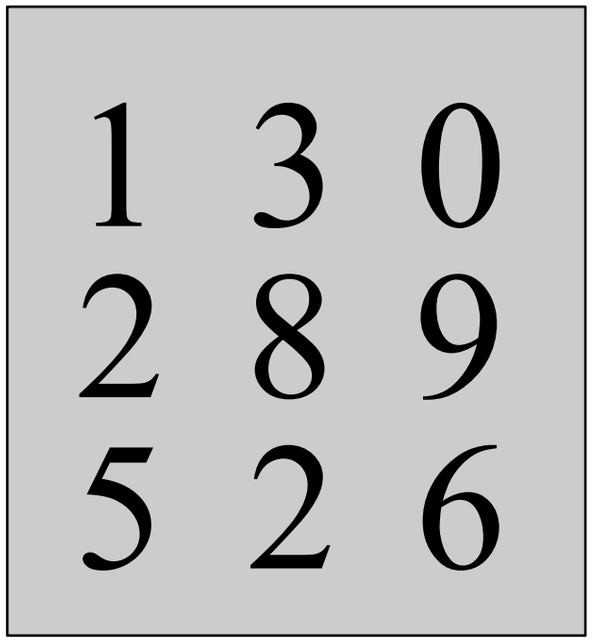
M matrix:
cumulative min energy
(for vertical seams)

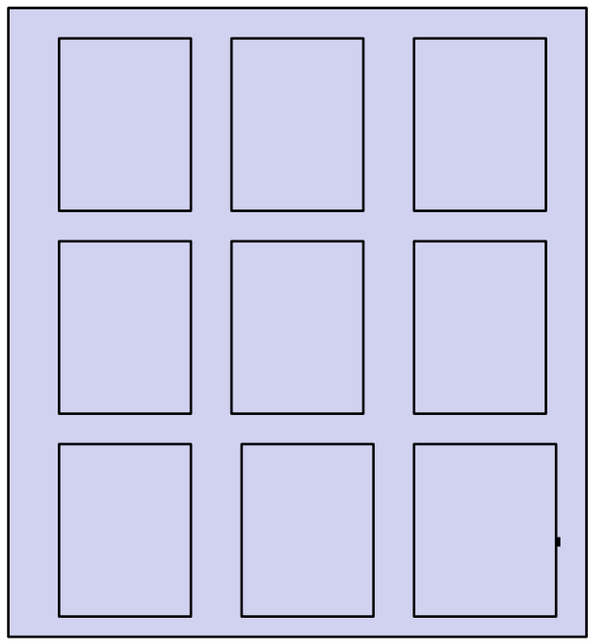
- Then, min value in last row of M indicates end of the minimal connected vertical seam.
- Backtrack up from there, selecting min of 3 above in M.

Slide credit: Kristen Grauman

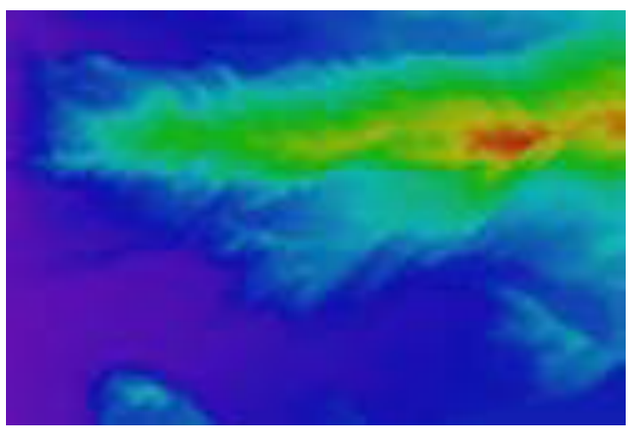
Example

$$\mathbf{M}(i,j) = Energy(i,j) + \min\left(\mathbf{M}(i-1,j-1), \mathbf{M}(i-1,j), \mathbf{M}(i-1,j+1)\right)$$





Energy matrix
(gradient magnitude)

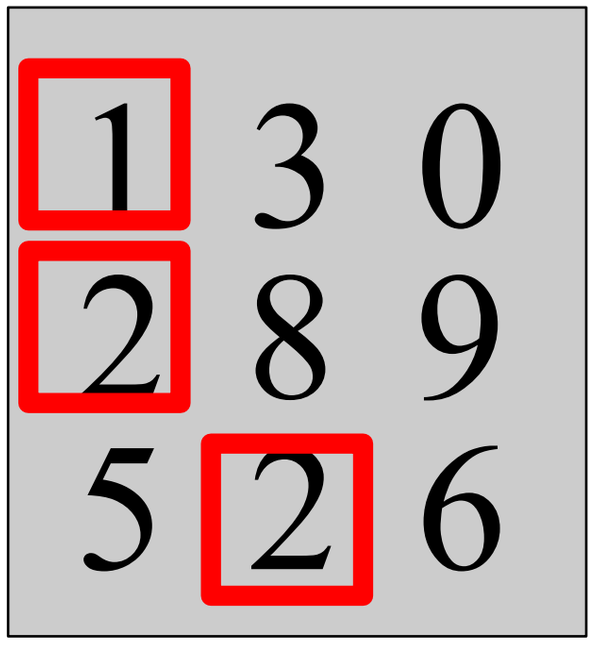


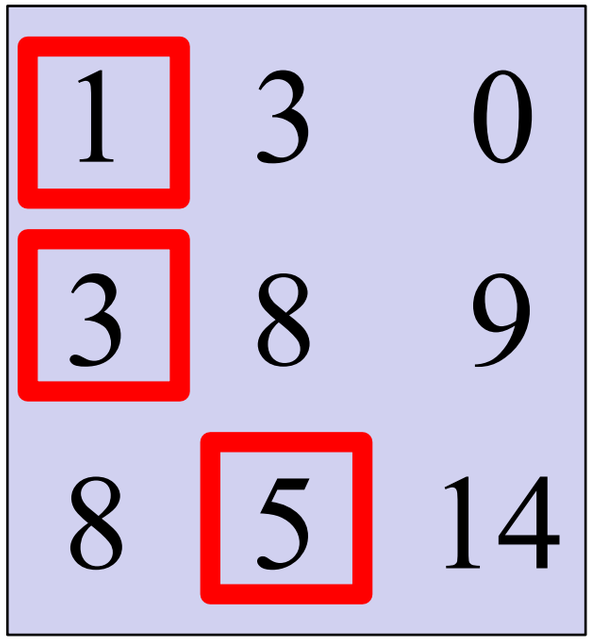
M matrix
(for vertical seams)

Slide credit: Kristen Grauman

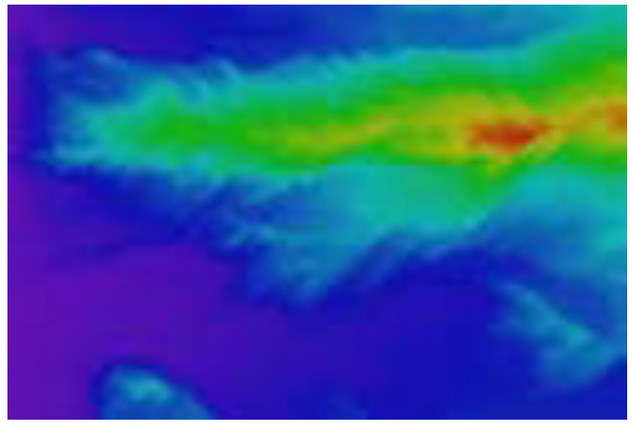
Example

$$\mathbf{M}(i,j) = Energy(i,j) + \min\left(\mathbf{M}(i-1,j-1), \mathbf{M}(i-1,j), \mathbf{M}(i-1,j+1)\right)$$





Energy matrix
(gradient magnitude)



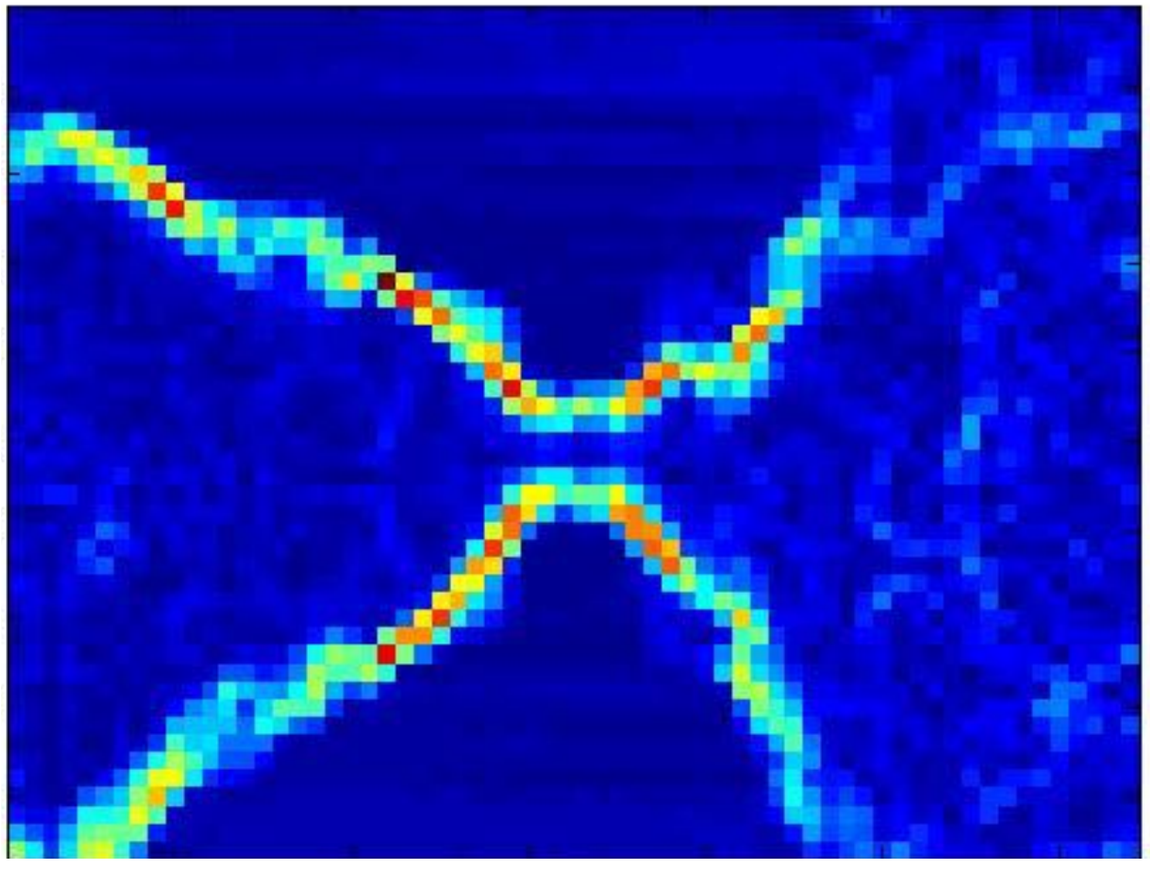
M matrix
(for vertical seams)

Slide credit: Kristen Grauman

Real image example

Original Image

Energy Map



Blue = low energy
Red = high energy

Slide credit: Kristen Grauman

Other notes on seam carving

- Analogous procedure for horizontal seams
- Can also insert seams to increase size of image in either dimension
  - Duplicate optimal seam, averaged with neighbors
- Other energy functions may be plugged in
  - E.g., color-based, interactive,…
- Can use combination of vertical and horizontal seams

One more thing

How to debug
image processing code?

Common bugs

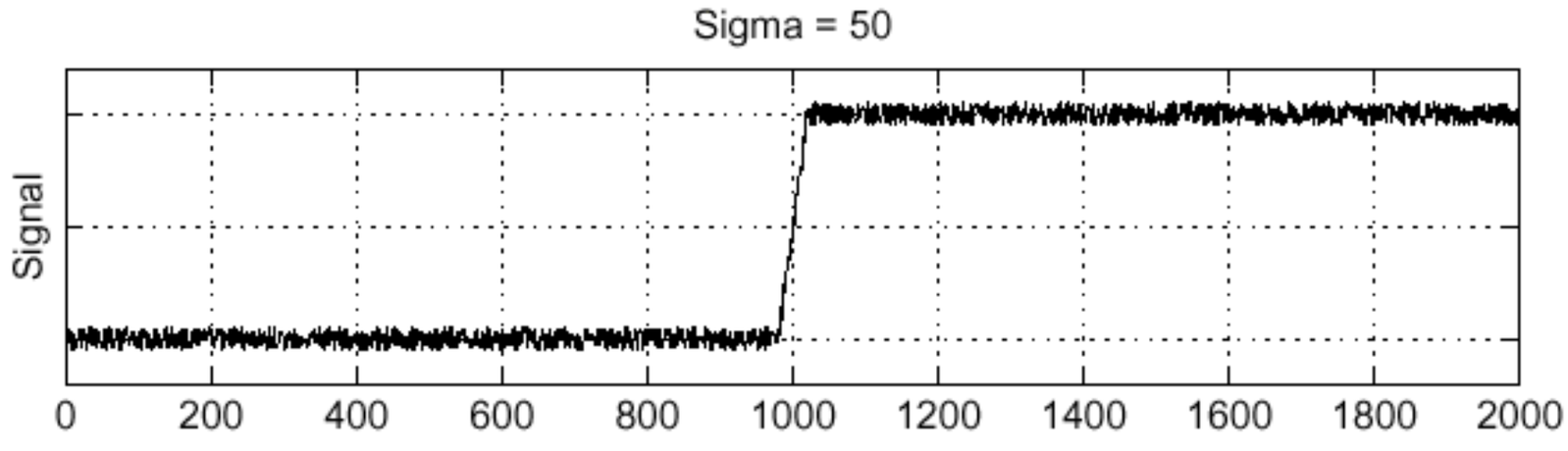
- Color channel order
  - Different libraries handle order differently
  - RGB, BGR, etc.
- Pixel scale
  - uint8: 255 (unsigned integer, default when read in)
  - float: [0, 1]
- Row and column indexing
  - (row, col) or (col, row)
  - Need to be careful when ‘imresize’
- Linear or gamma-corrected color space
  - Gamma curve

Jupyter live demo

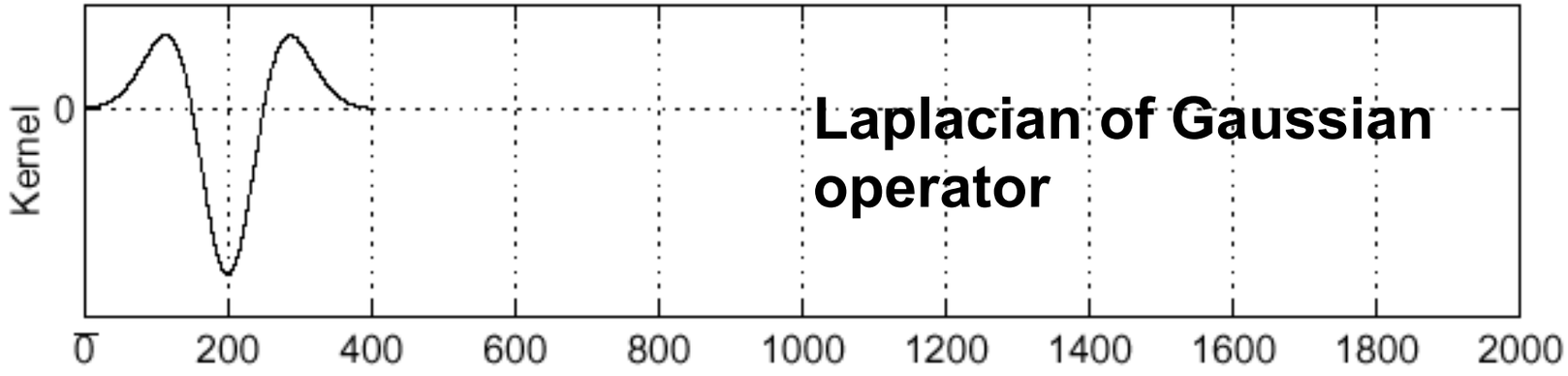
Laplacian of Gaussian

$$\frac{\partial^2}{\partial x^2}(h \star f)$$

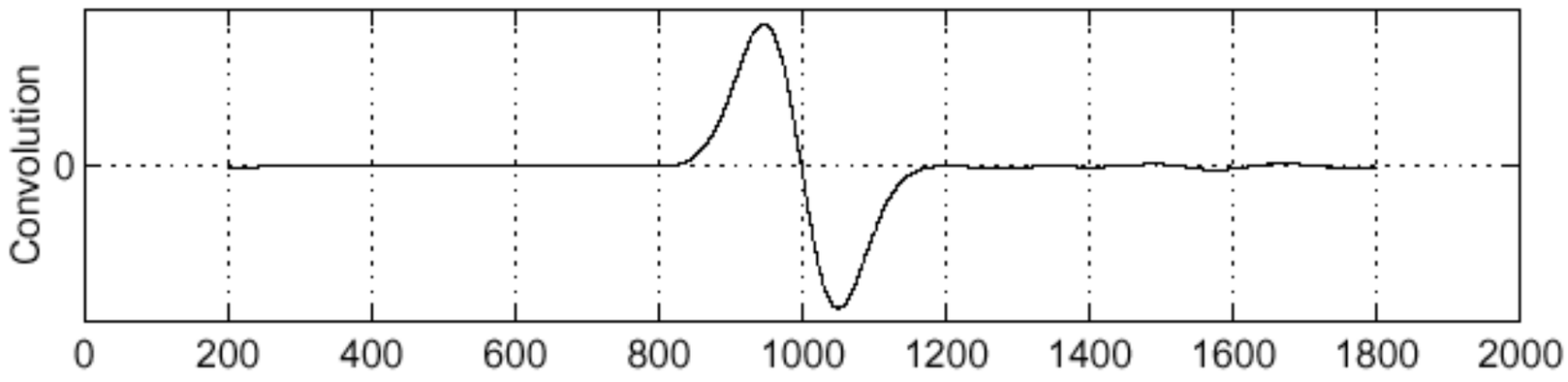
$$f$$



$$\frac{\partial^2}{\partial x^2}h$$



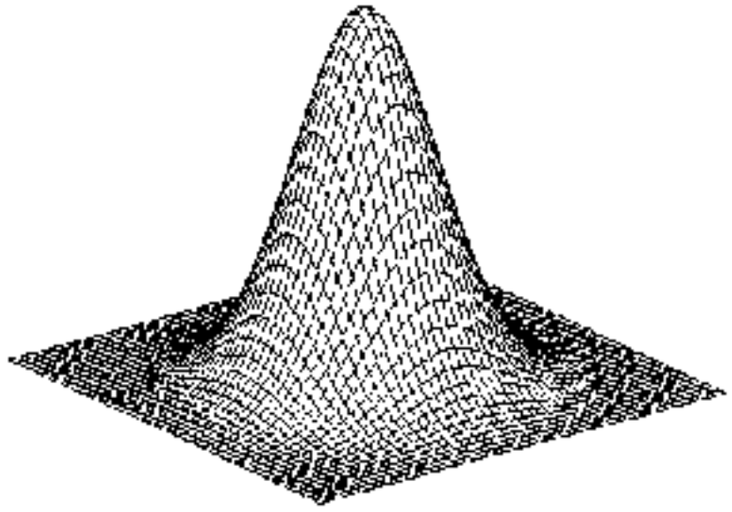
$$(\frac{\partial^2}{\partial x^2}h) \star f$$



Where is the edge?

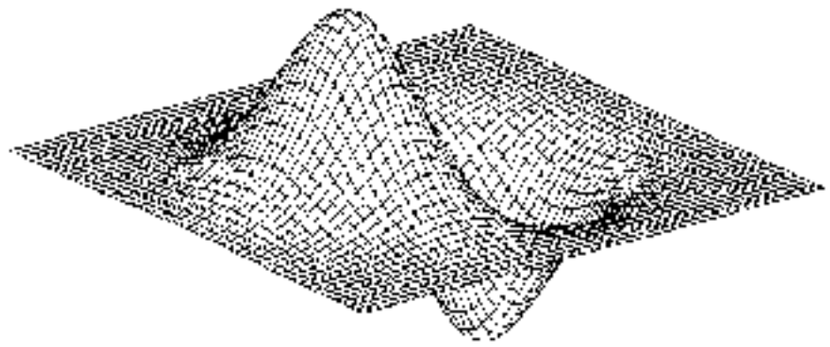
Zero-crossings of bottom graph

2D edge detection filters



Gaussian

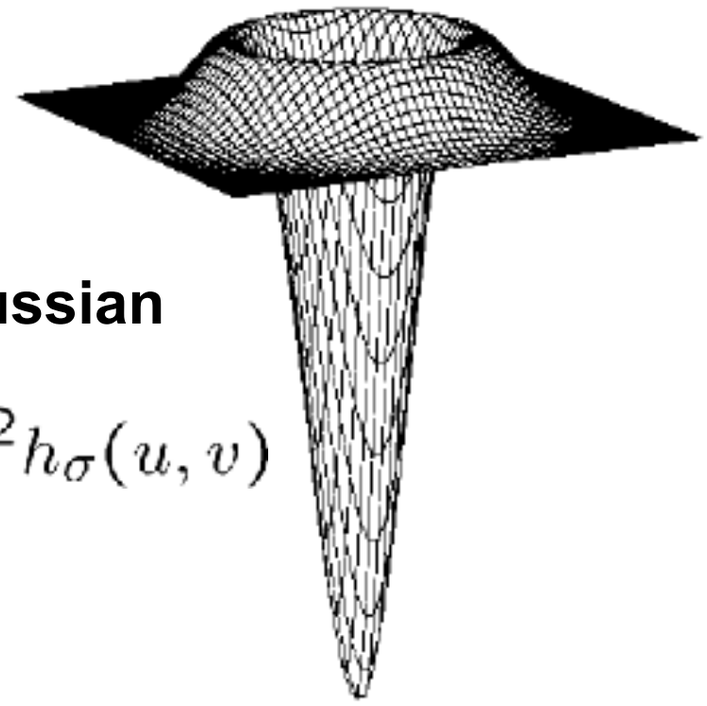
$$h_\sigma(u,v) = \frac{1}{2\pi\sigma^2} e^{-\frac{u^2+v^2}{2\sigma^2}}$$



derivative of Gaussian

$$\frac{\partial}{\partial x} h_\sigma(u,v)$$

Laplacian of G



$$\nabla^2 h_\sigma(u,v)$$

- ∇^2 is the Laplacian operator:

$$\nabla^2 f = \frac{\partial^2 f}{\partial x^2} + \frac{\partial^2 f}{\partial y^2}$$

Slide credit: Steve Seitz