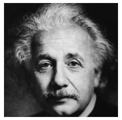
# Filtering (in Frequency Domain)

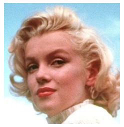
CS 543 / ECE 549 – Saurabh Gupta

Many slides from Derek Hoiem.

While we wait,

Who do you see in this picture? 

A. Albert Einstein 

B. Marilyn Monroe 

C. None of these

# Today’s Class

- Fourier transforms
- Filtering in frequency domain
- Sampling
- Image Pyramids

# Why does the Gaussian give a nice smooth image, but the square filter give edgy artifacts?

Gaussian

Box filter

# Thinking in terms of frequency

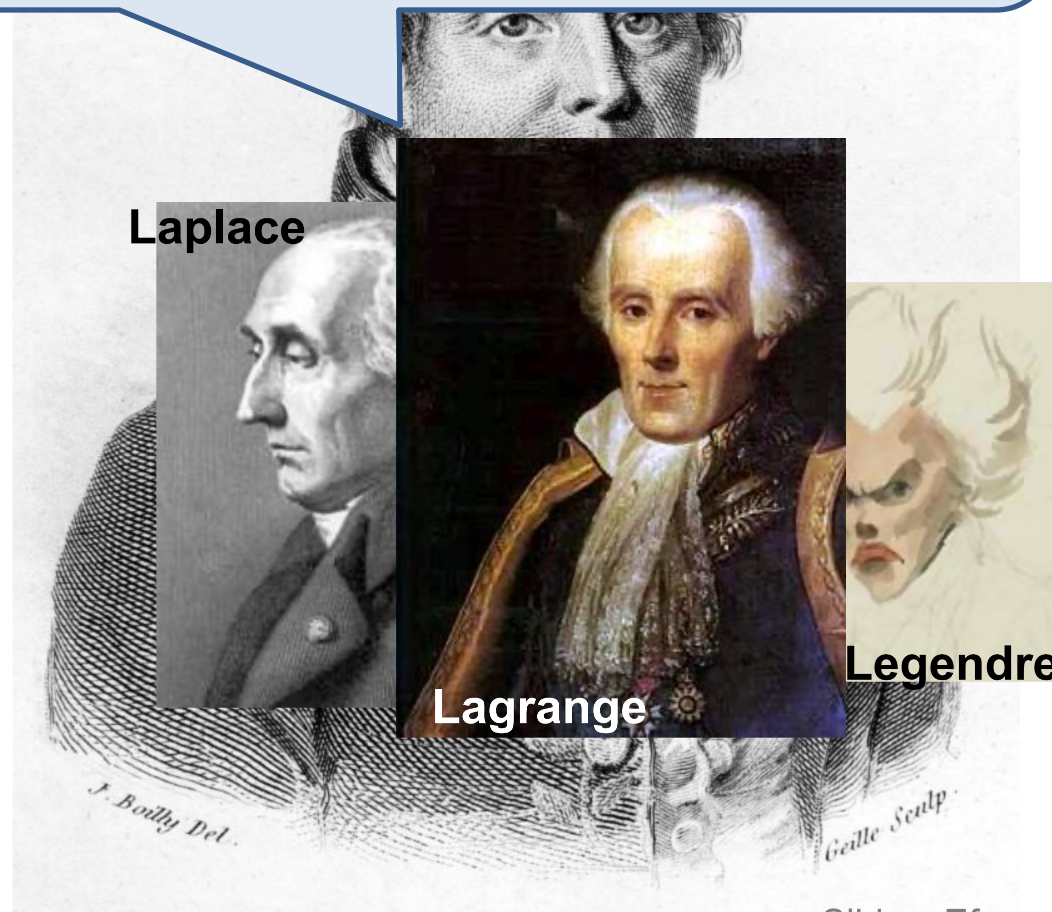
# Jean Baptiste Joseph Fourier (1768-1830)

had crazy idea (1807):

*__Any__ univariate function can*
*rewritten as a weighted sum*
*sines and cosines of different*
*frequencies.*

> *...the manner in which the author arrives at these equations is not exempt of difficulties and...his analysis to integrate them still leaves something to be desired on the score of generality and even rigour.*

- Don't believe it?
  - Neither did Lagrange, Laplace, Poisson and other big wigs
  - Not translated into English until 1878!
- But it's (mostly) true!
  - called Fourier Series
  - there are some subtle restrictions



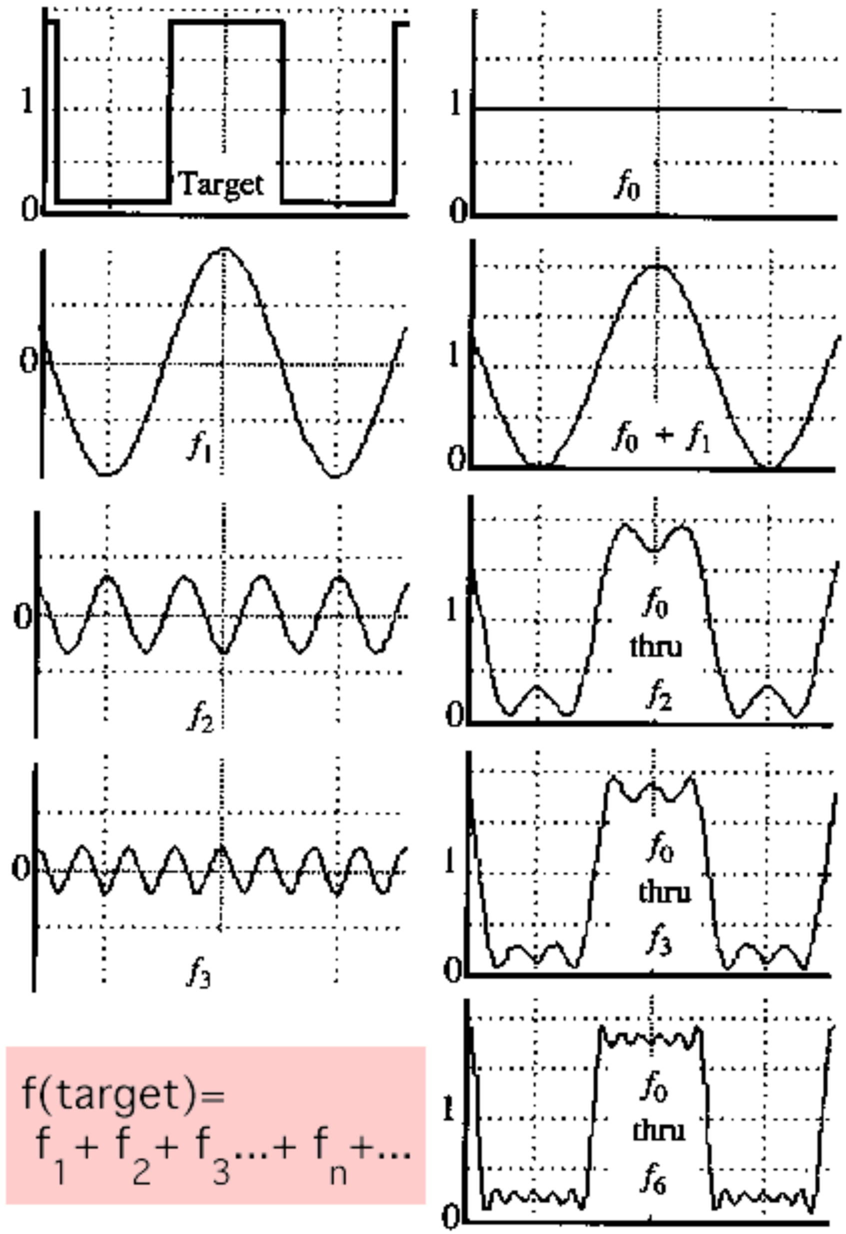

Slides: Efros

# A sum of sines

Our building block:

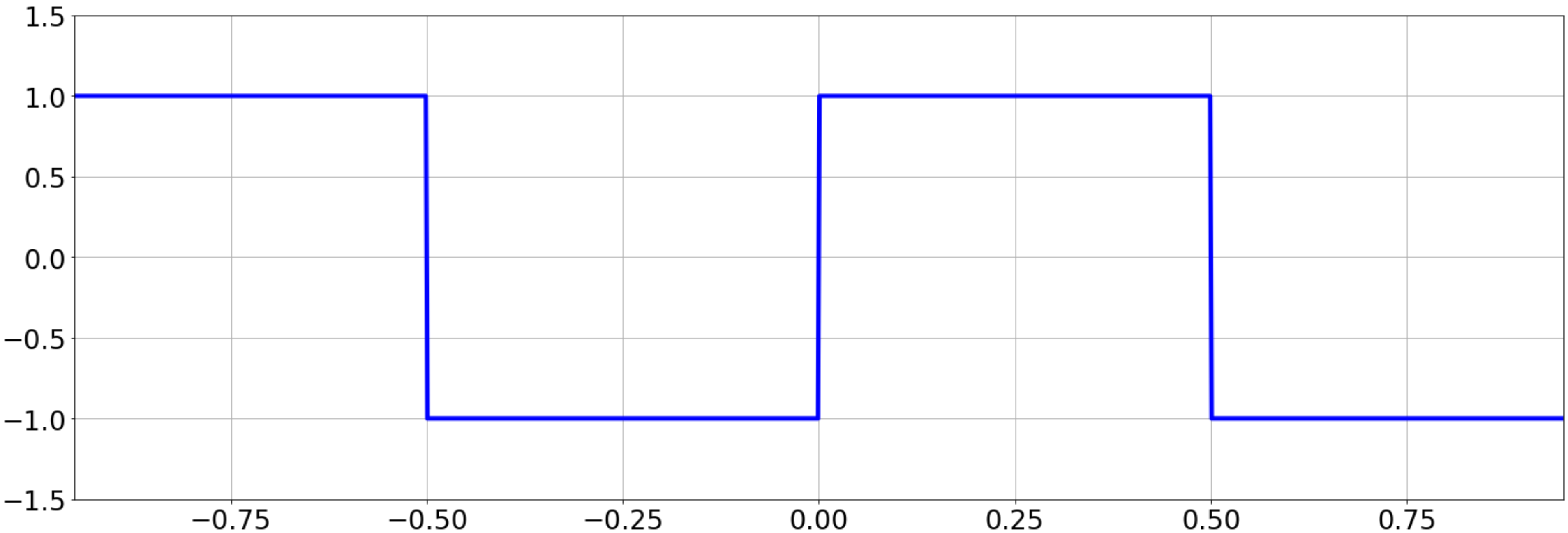
$$A\sin(\omega x+\phi)$$

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




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

# Frequency Spectra

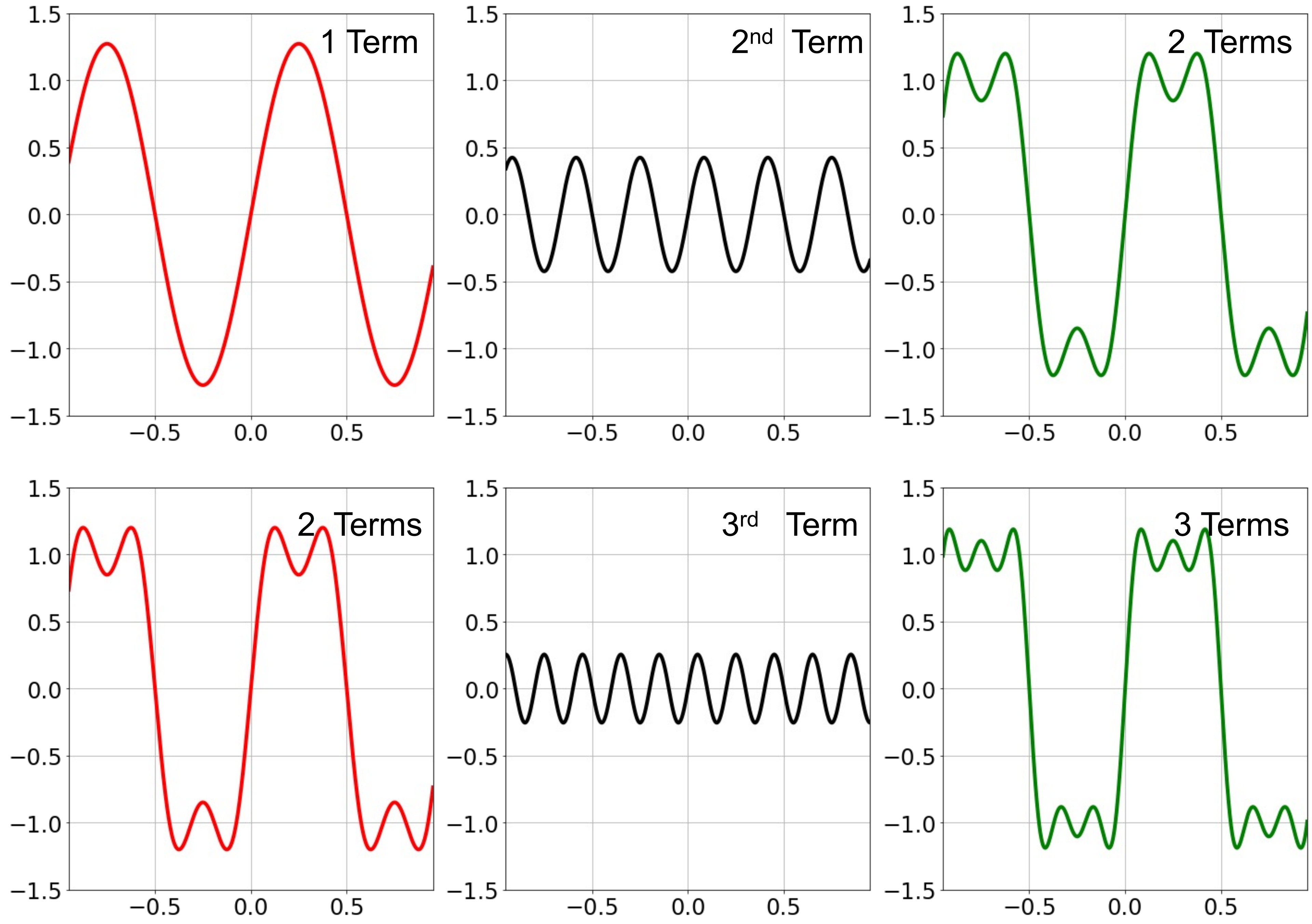


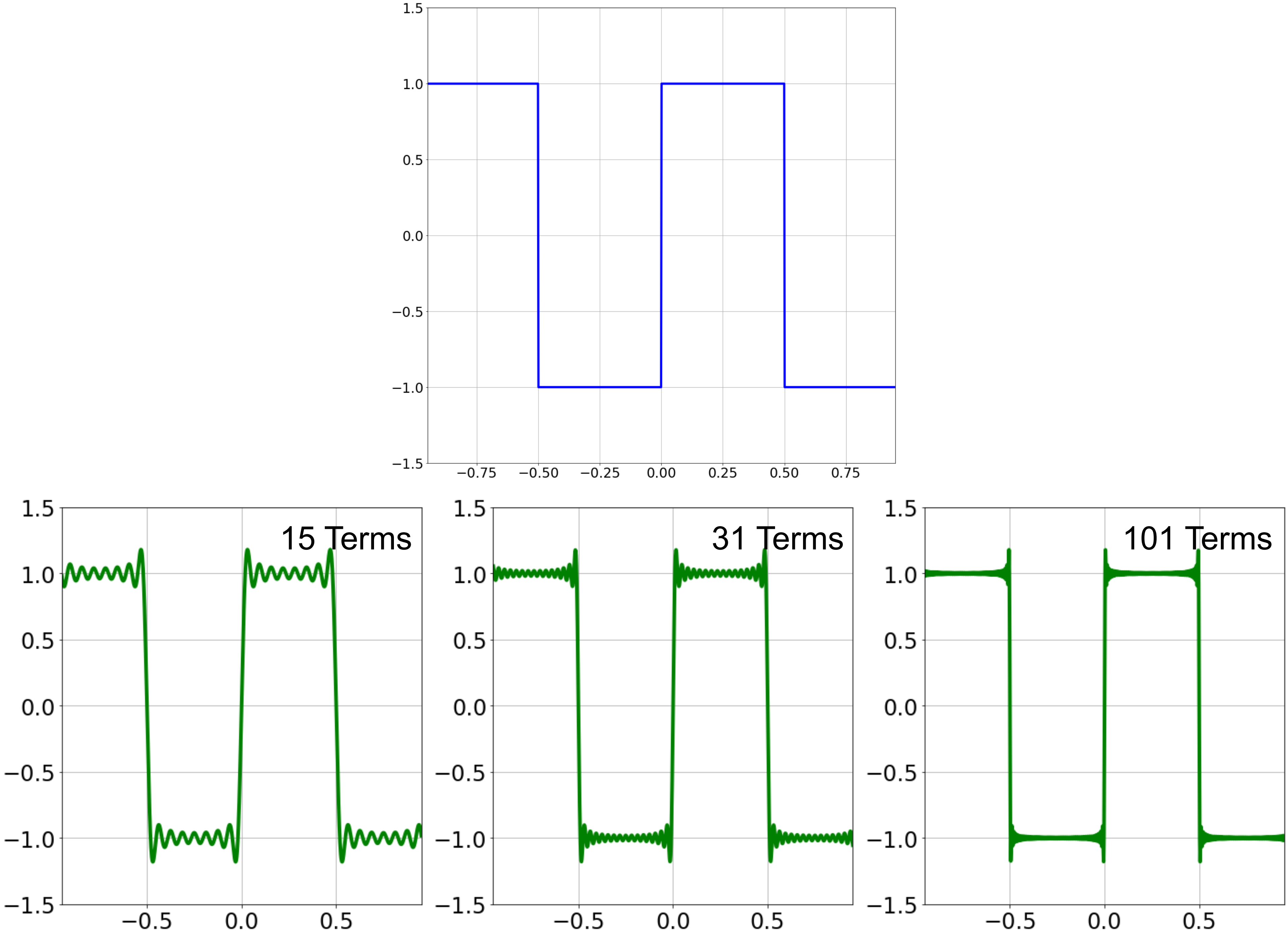

Square Wave:

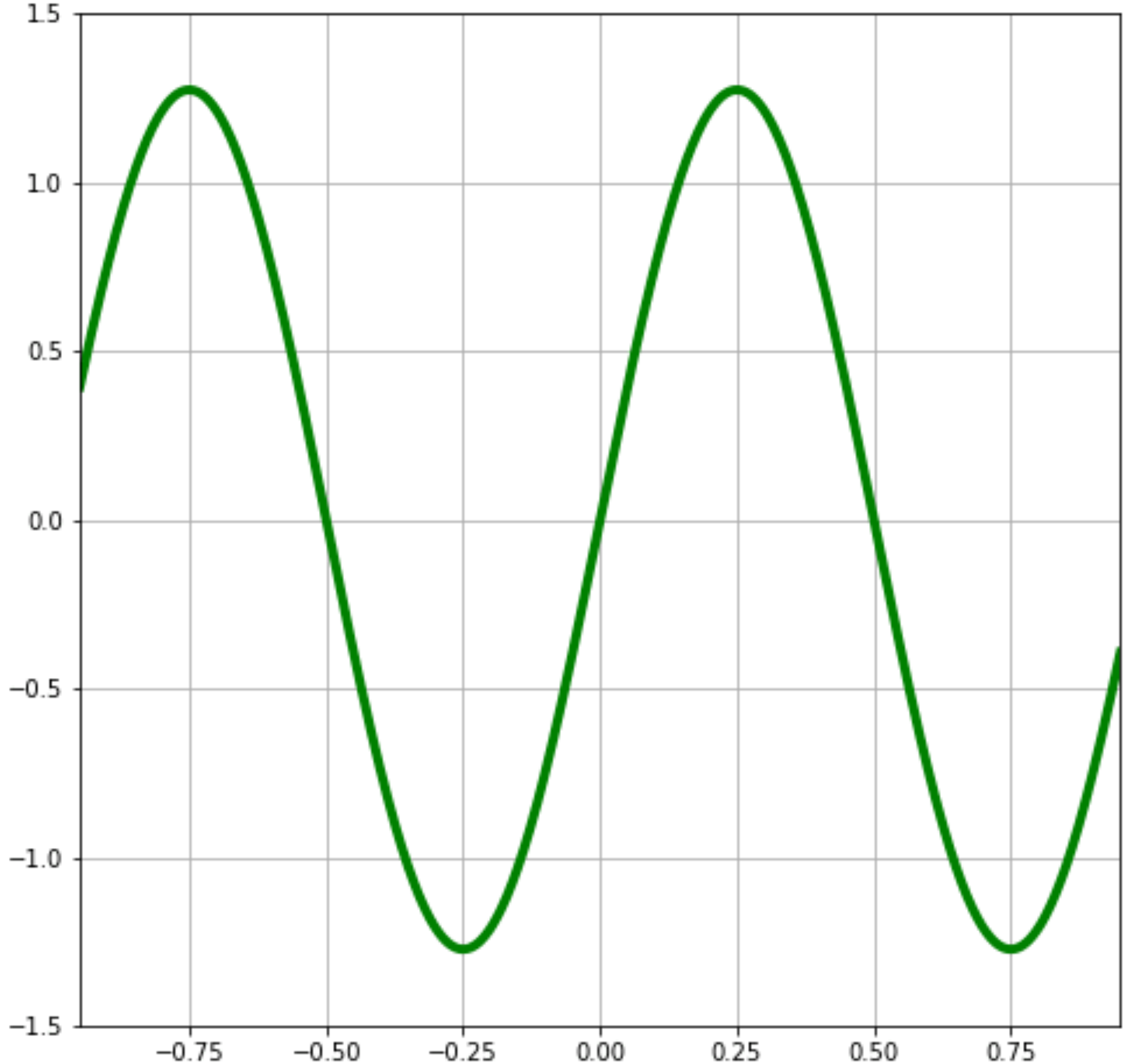
$$f(x) = \begin{cases} 1, & \text{if frac}(x) < 0.5 \\ -1, & \text{otherwise} \end{cases}$$

Fourier Transform:

$$\frac{4}{\pi}\left(\frac{\sin(2\pi.1.x)}{1} + \frac{\sin(2\pi.3.x)}{3} + \frac{\sin(2\pi.5.x)}{5} + \ \ldots\right)$$


1 Term
2nd Term
2 Terms
2 Terms
3rd Term
3 Terms


15 Terms
31 Terms
101 Terms



1.5
1.0
0.5
0.0
−0.5
−1.0
−1.5
−0.75
−0.50
−0.25
0.00
0.25
0.50
0.75

# 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 complex numbers
- Amplitude: $A = \sqrt{R(\omega)^2 + I(\omega)^2}$
- Phase: $\phi = tan^{-1} \frac{I(\omega)}{R(\omega)}$

# Computing the Fourier Transform

- $H(\omega) = \mathcal{F}[h(x)]$

- Continuous:
  - $H(\omega) = \int_{-\infty}^{\infty} h(x) e^{-j\omega x} dx$

- Discrete:
  - $H(k) = \frac{1}{N} \sum_{x=0}^{N-1} h(x) e^{-j2\pi kx/N}$

- Euler's Formula:
  - $e^{jnx} = \cos(nx) + j \sin(nx)$

# Properties of Fourier Transforms

- Linearity:
  - $\mathcal{F}[ax(t) + by(t)] = a\mathcal{F}[x(t)] + b\mathcal{F}[y(t)]$

- 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

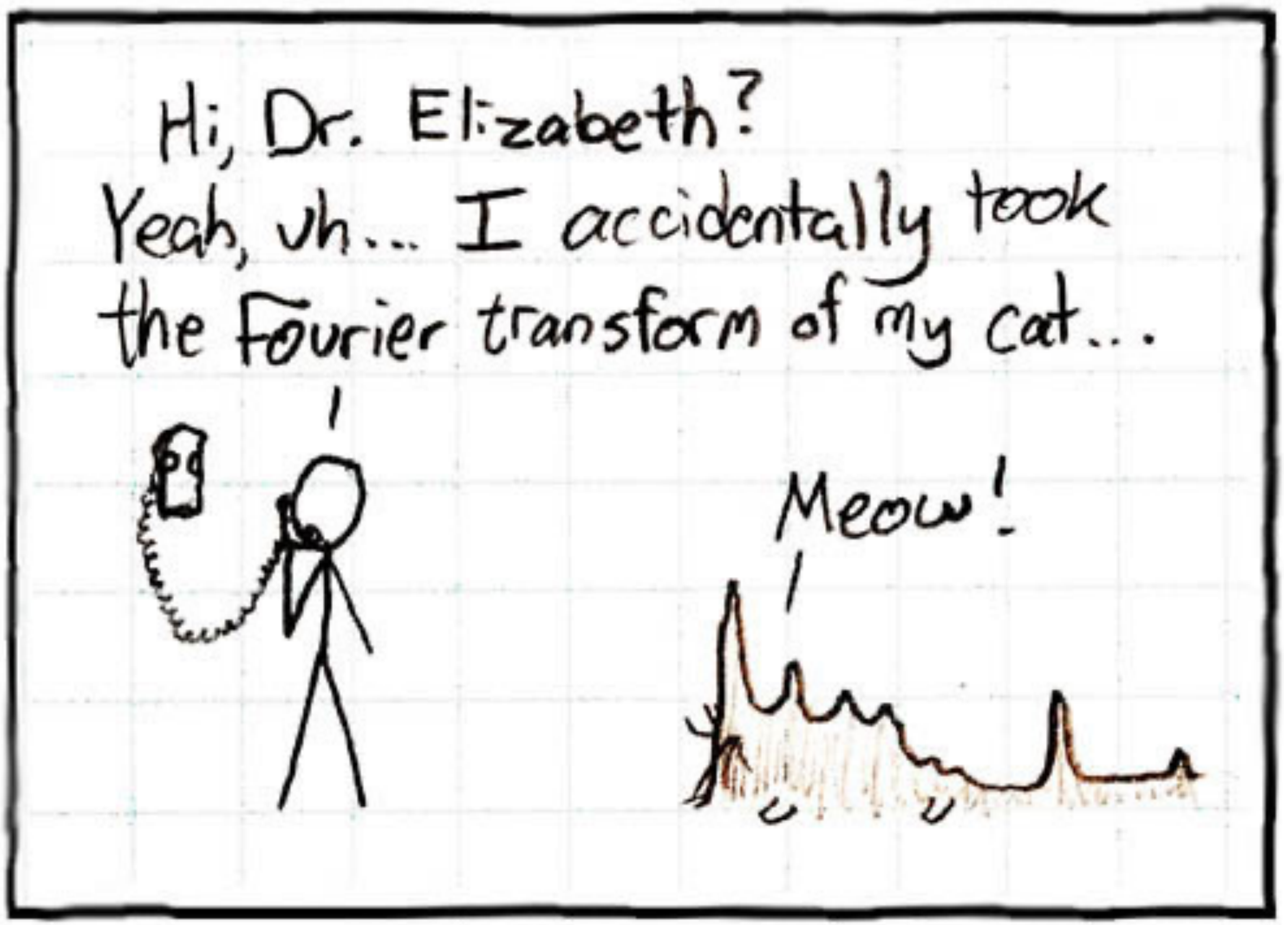
See Szeliski Book (3.4)

# The Convolution Theorem

- The Fourier transform of the convolution of two functions is the product of their Fourier transforms
  - $\mathcal{F}[g * h] = \mathcal{F}[g]\mathcal{F}[h]$
- The inverse Fourier transform of the product of two Fourier transforms is the convolution of the two inverse Fourier transforms
  - $\mathcal{F}^{-1}[gh] = \mathcal{F}^{-1}[g] * \mathcal{F}^{-1}[h]$
- **Convolution** in spatial domain is equivalent to **multiplication** in frequency domain!

# Other signals

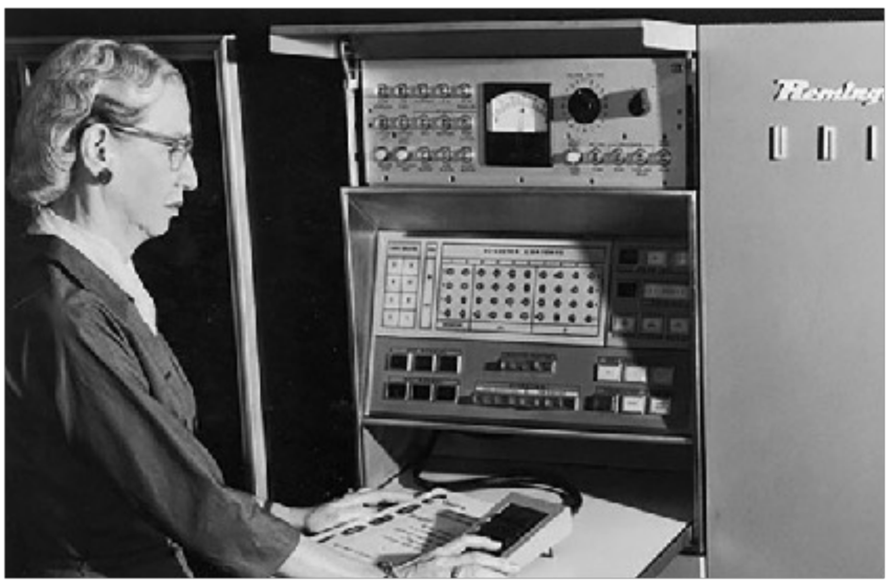
- We can also think of all kinds of other signals the same way

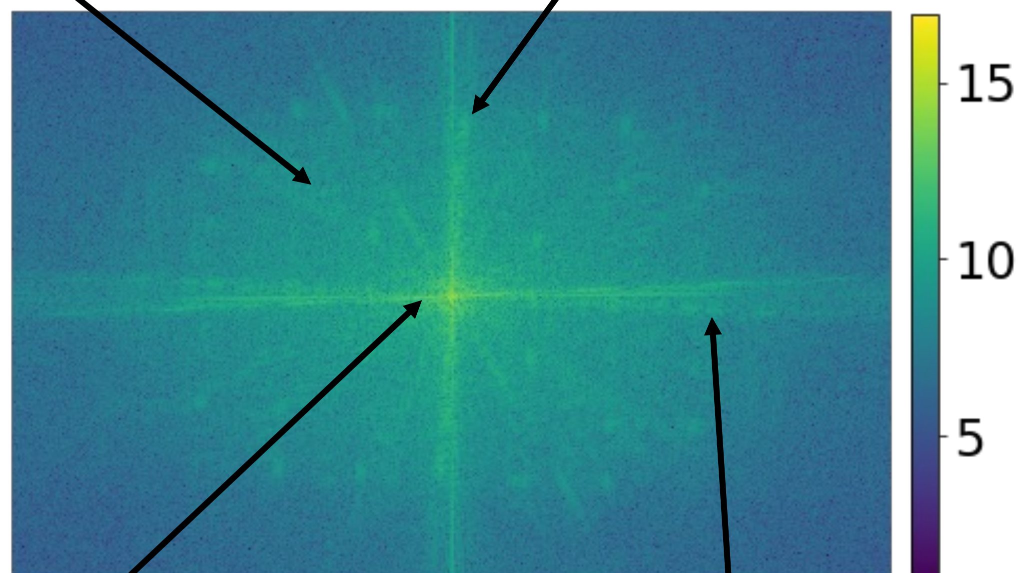



xkcd.com

# Images

$$H(k,l) = \frac{1}{MN}\sum_{m=0}^{M-1}\sum_{n=0}^{N-1} h(m,n)e^{-j2\pi\left(\frac{k}{M}m+\frac{l}{N}n\right)}$$

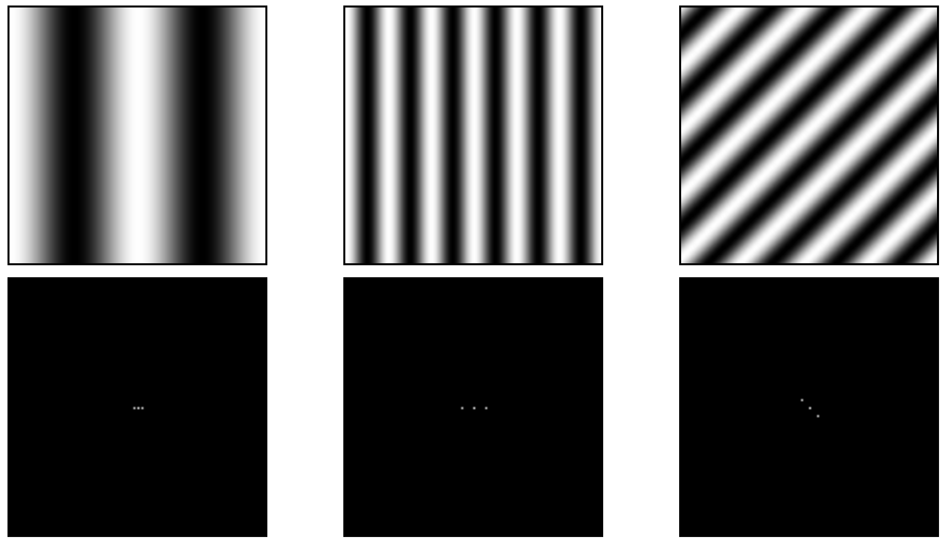




# Fourier analysis in images

Intensity Image

Fourier Image

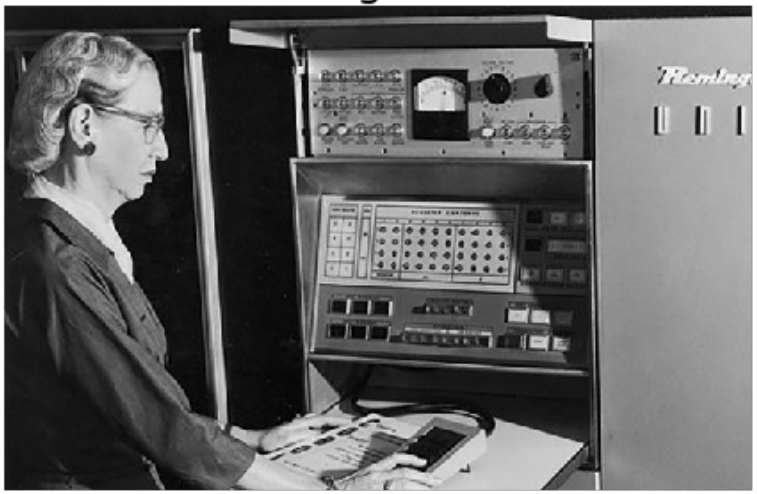


http://sharp.bu.edu/~slehar/fourier/fourier.html#filtering

# Filtering in spatial domain

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

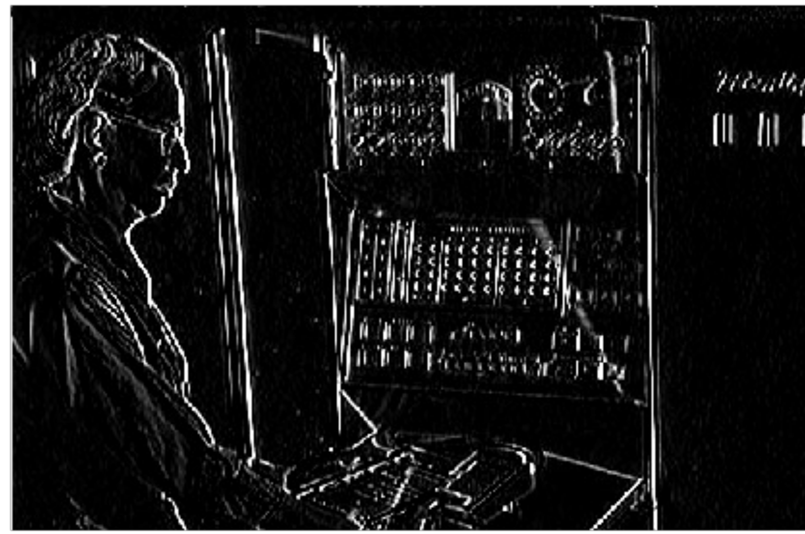
Original



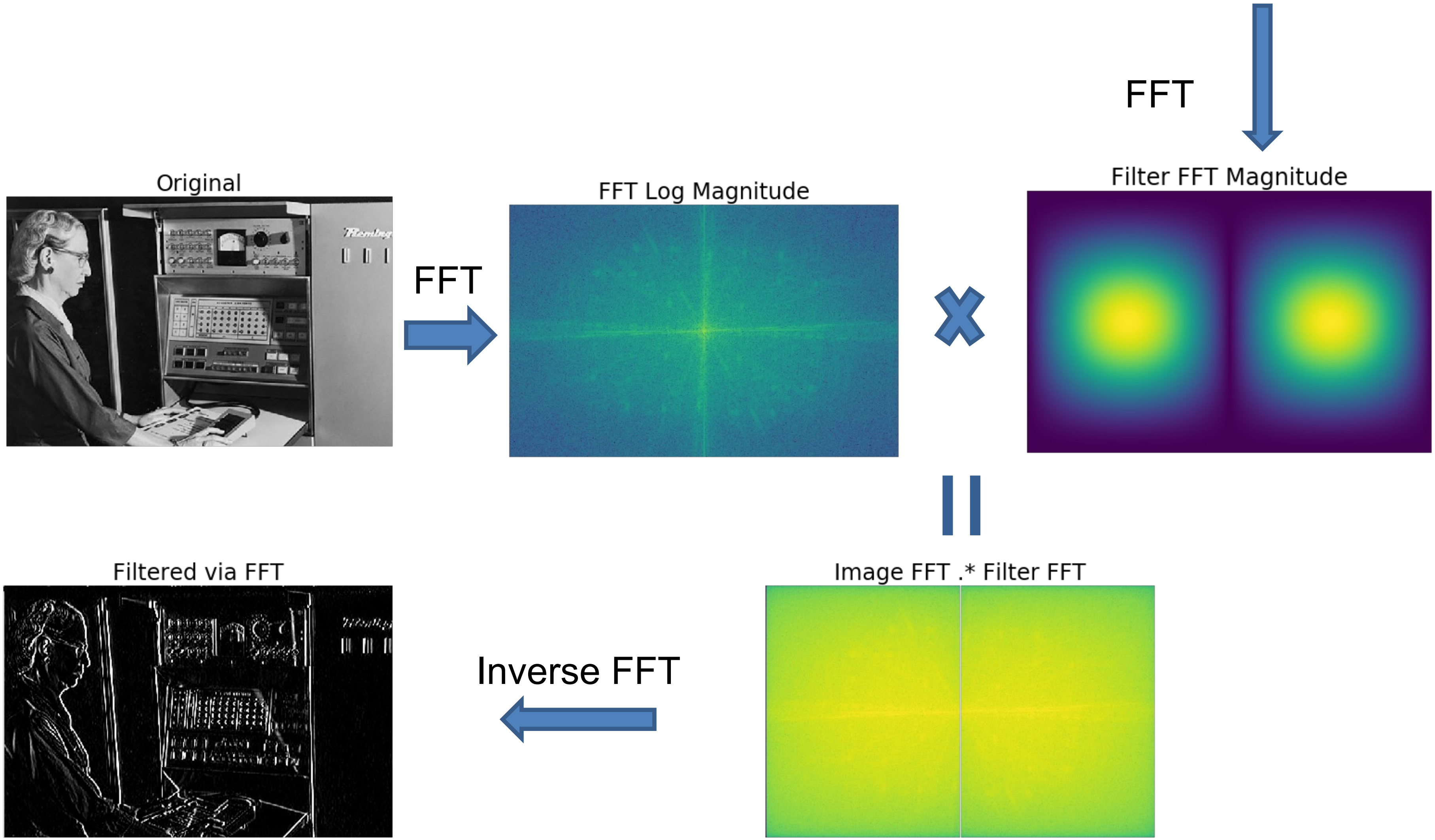
*

=

Filtered via Convolution

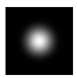


# Filtering in frequency domain



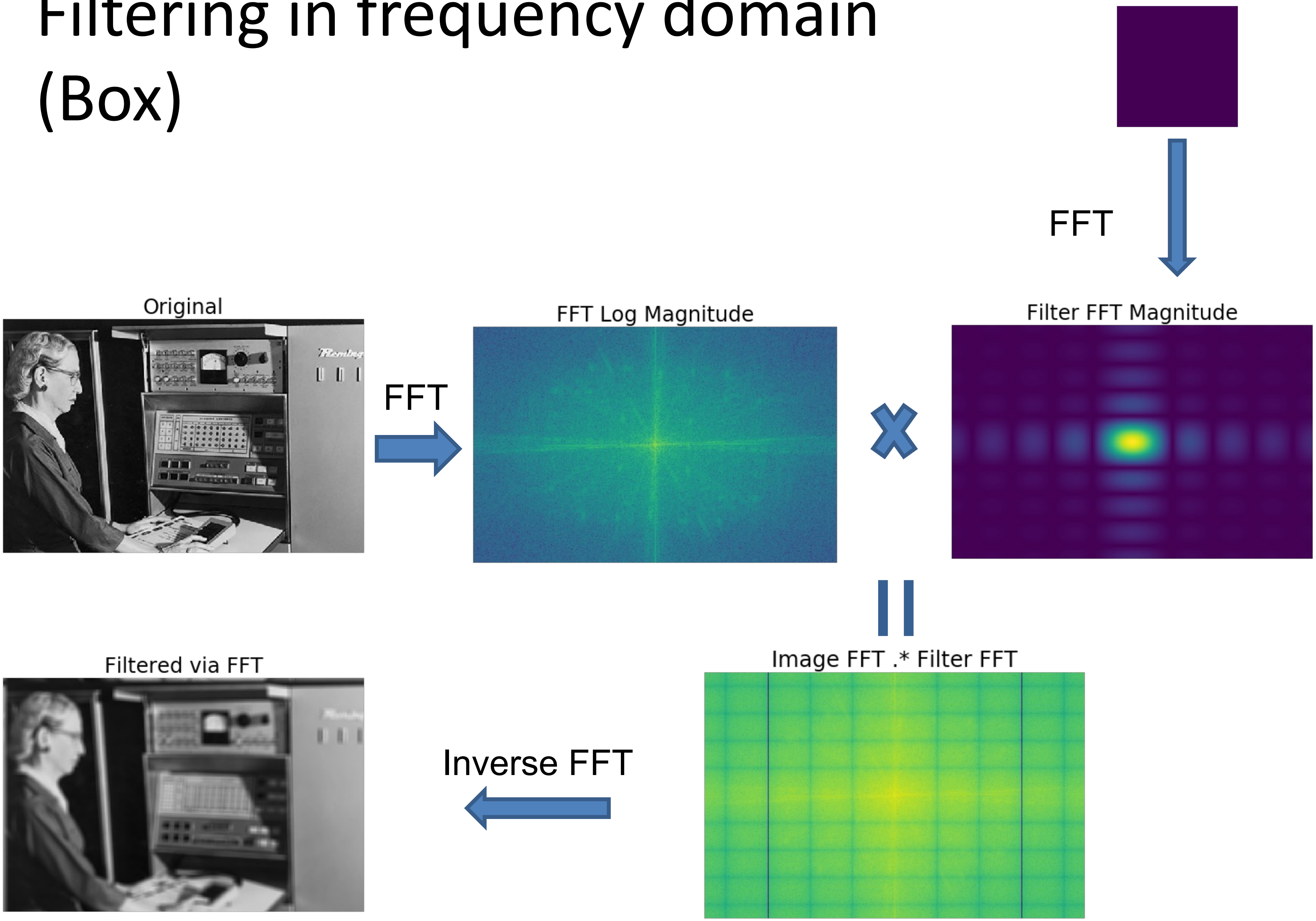
# Why does the Gaussian give a nice smooth image, but the square filter give edgy artifacts?

Gaussian

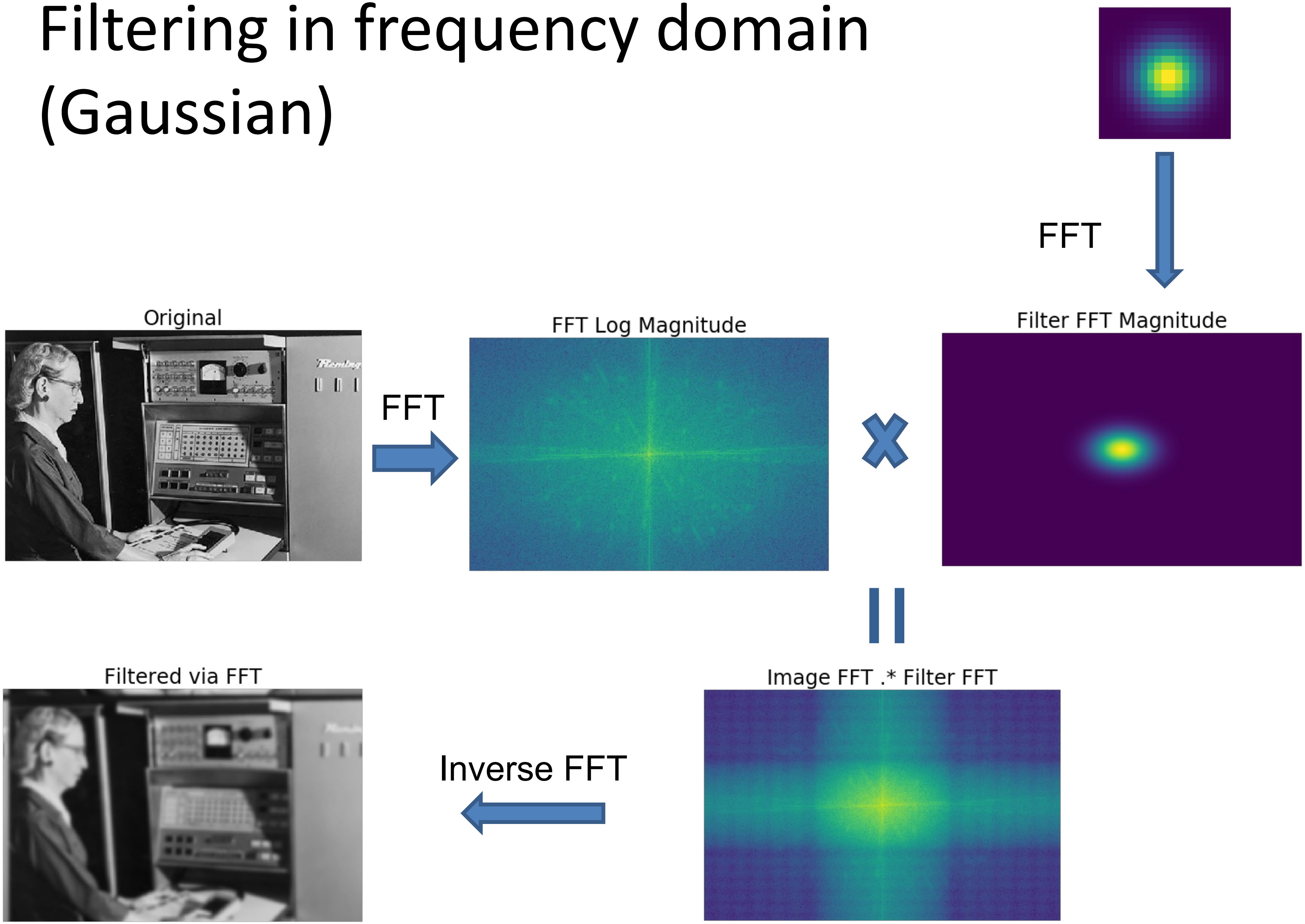


Box filter

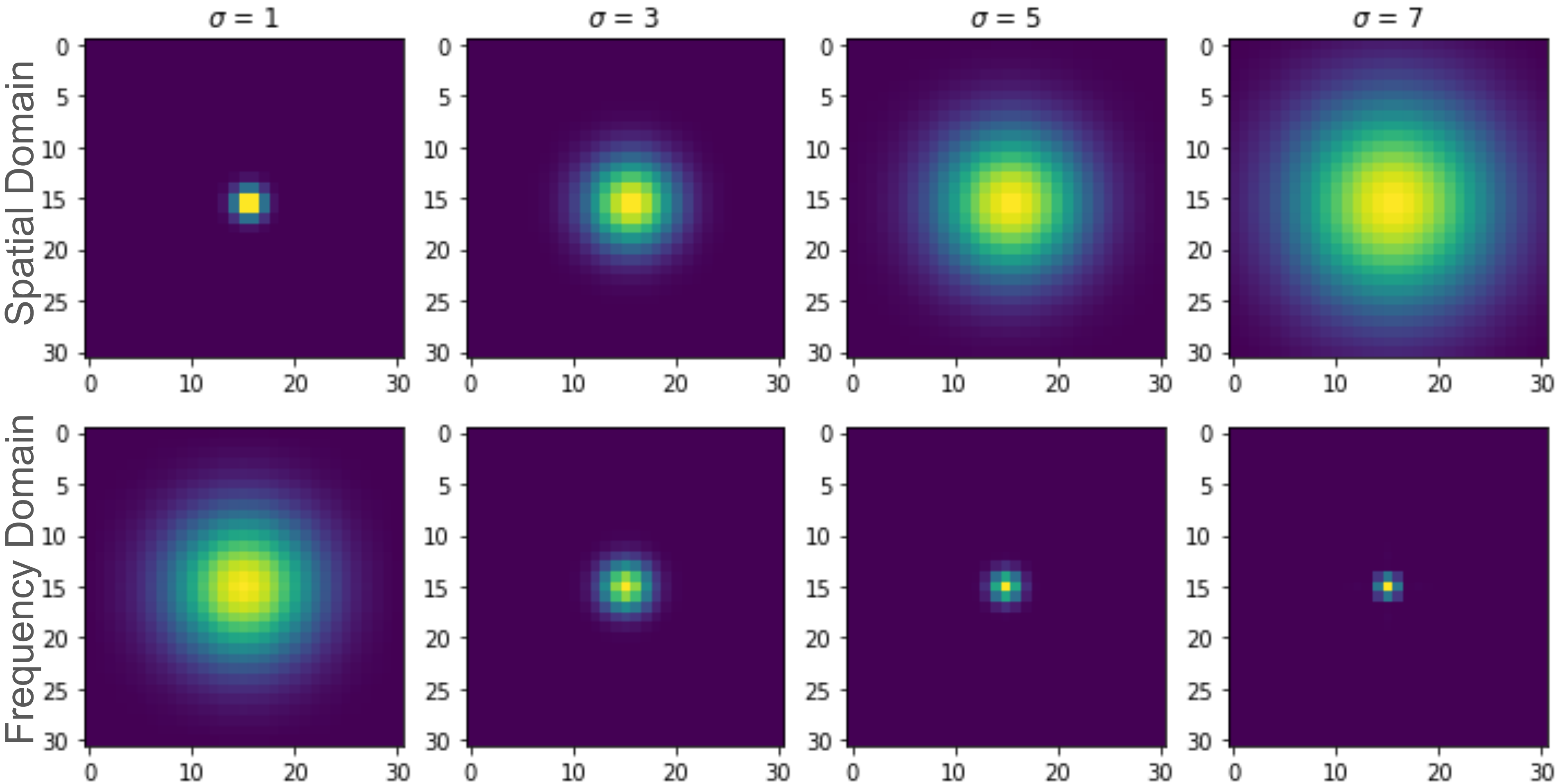
# Filtering in frequency domain (Box)



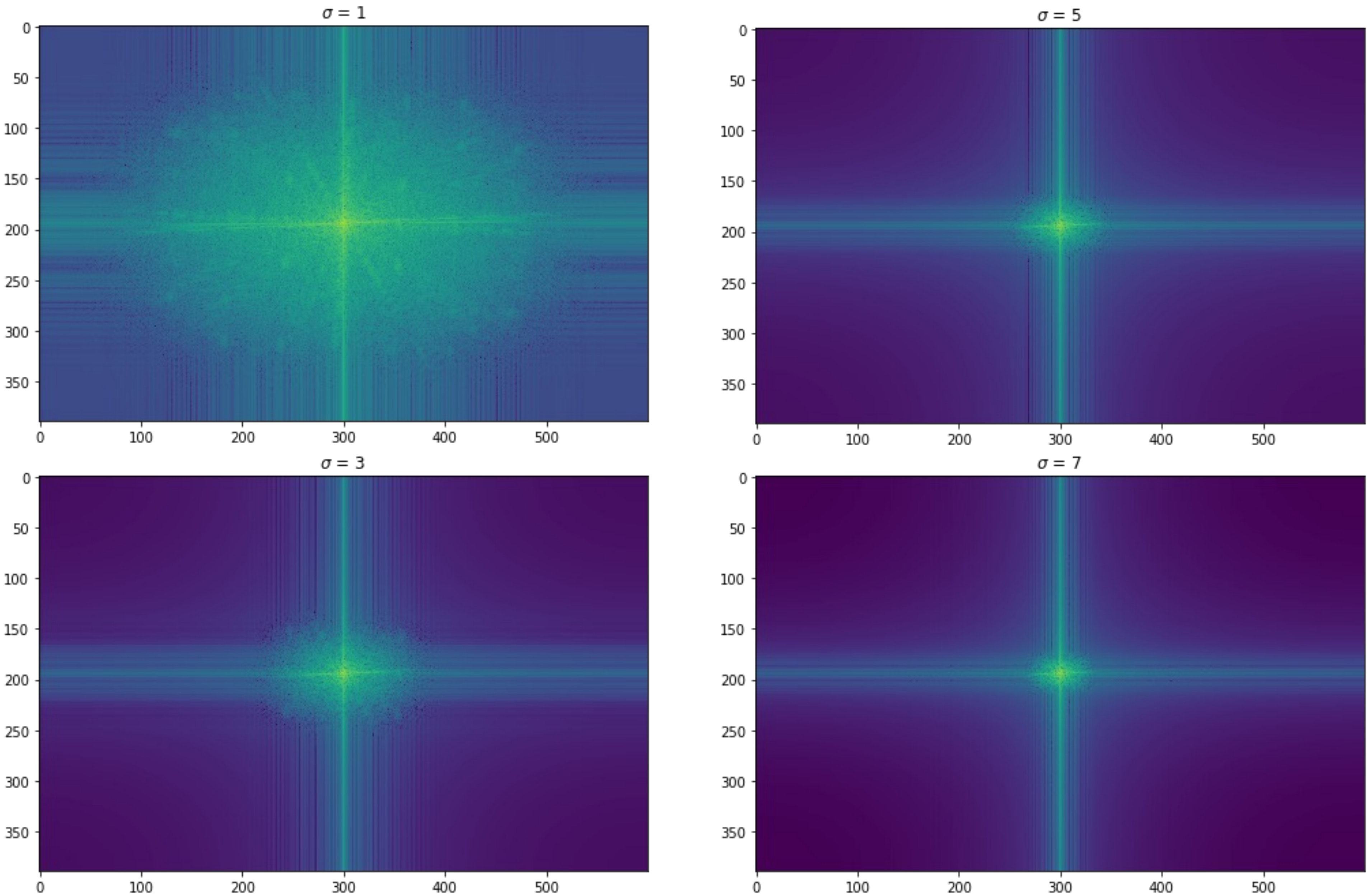
# Filtering in frequency domain (Gaussian)



# Filtering in frequency domain (Gaussian)



# Filtering in frequency domain (Gaussian)



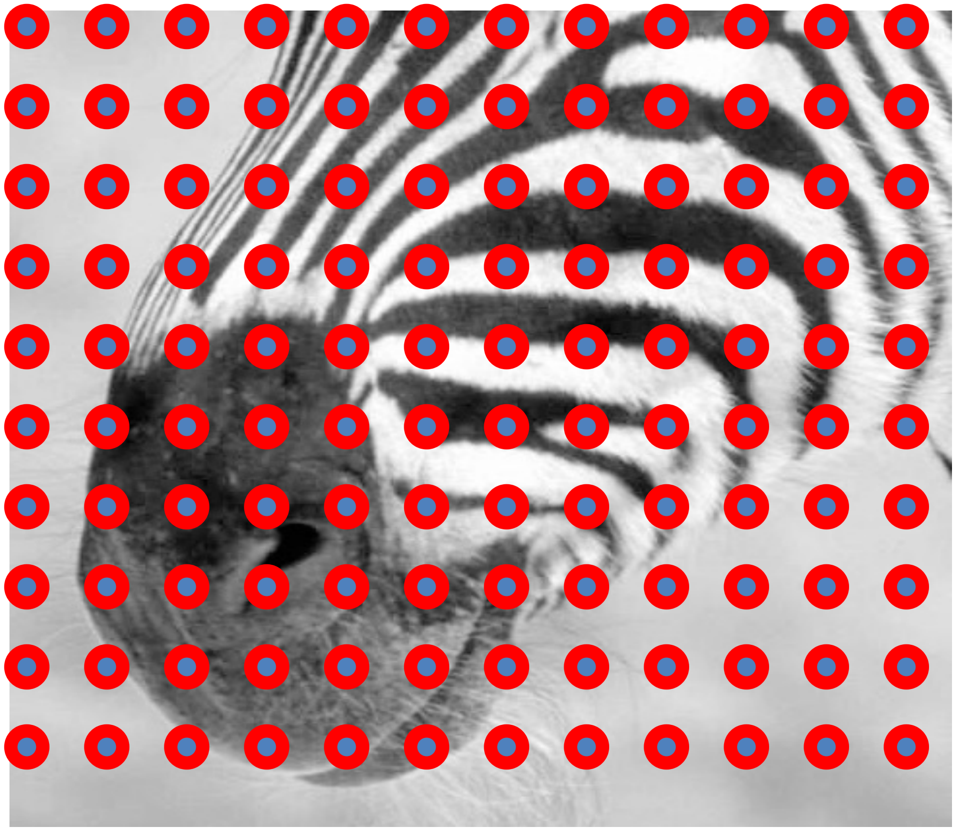
# Filtering in frequency domain (Gaussian)

# Sampling

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

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

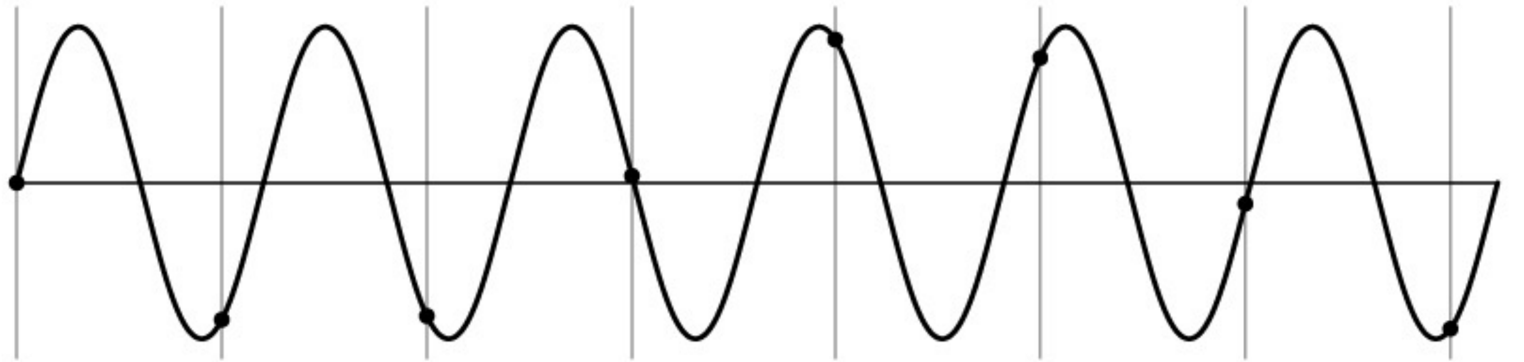
# Subsampling by a factor of 2



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

# Aliasing problem

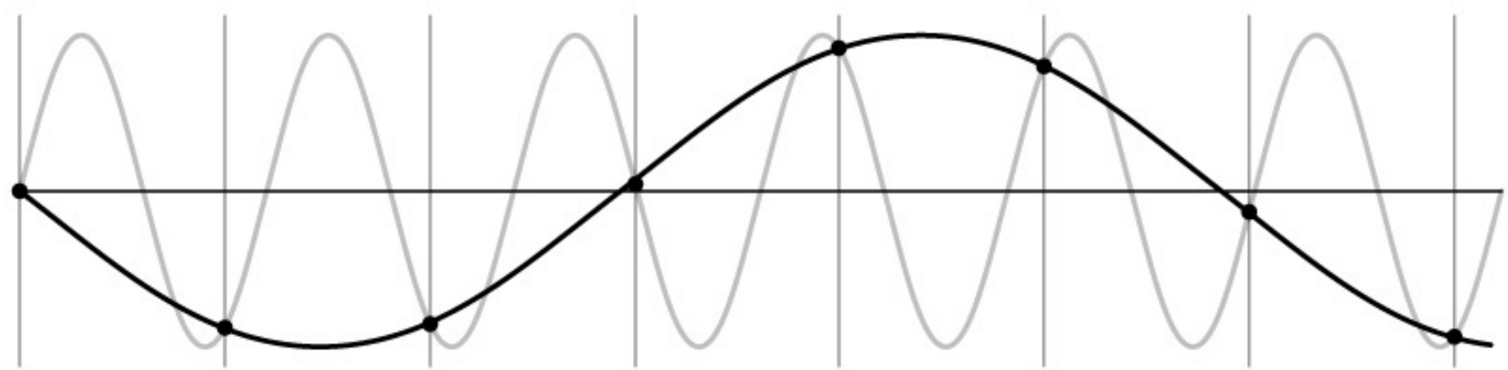
- 1D example (sinewave):



Source: S. Marschner

# Aliasing problem

- 1D example (sinewave):



Source: S. Marschner

# Aliasing problem

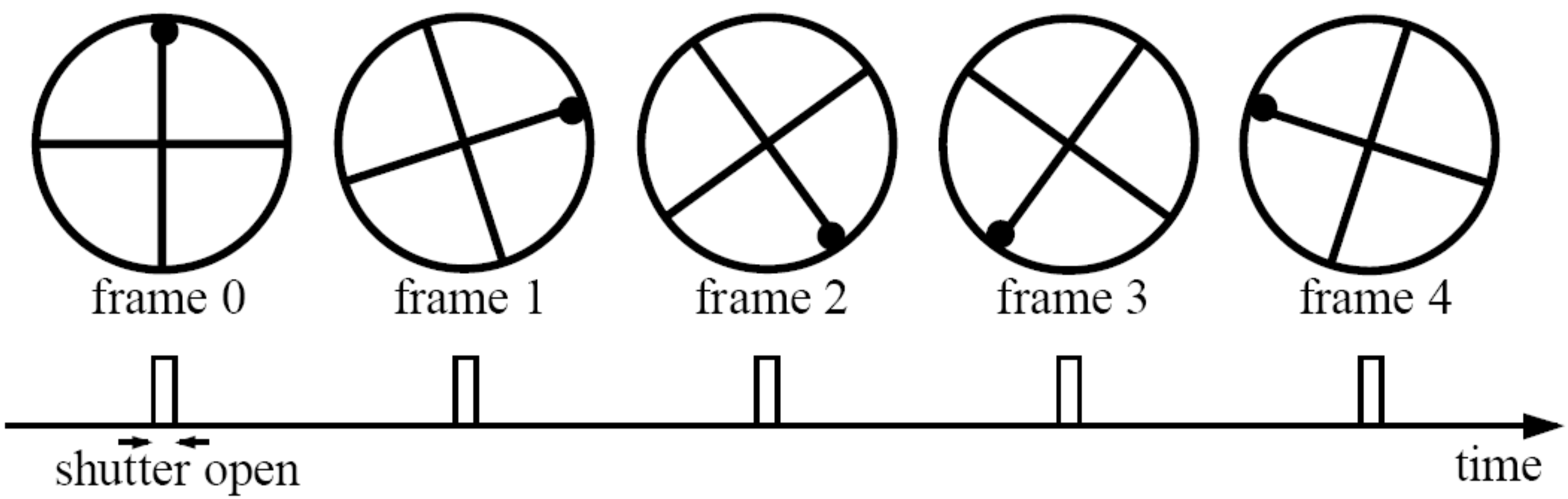
- Sub-sampling may be dangerous….
- Characteristic errors may appear:
  - “Wagon wheels rolling the wrong way in movies” See
  - “Checkerboards disintegrate in ray tracing”
  - “Striped shirts look funny on color television”

Source: D. Forsyth

# Aliasing in video

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

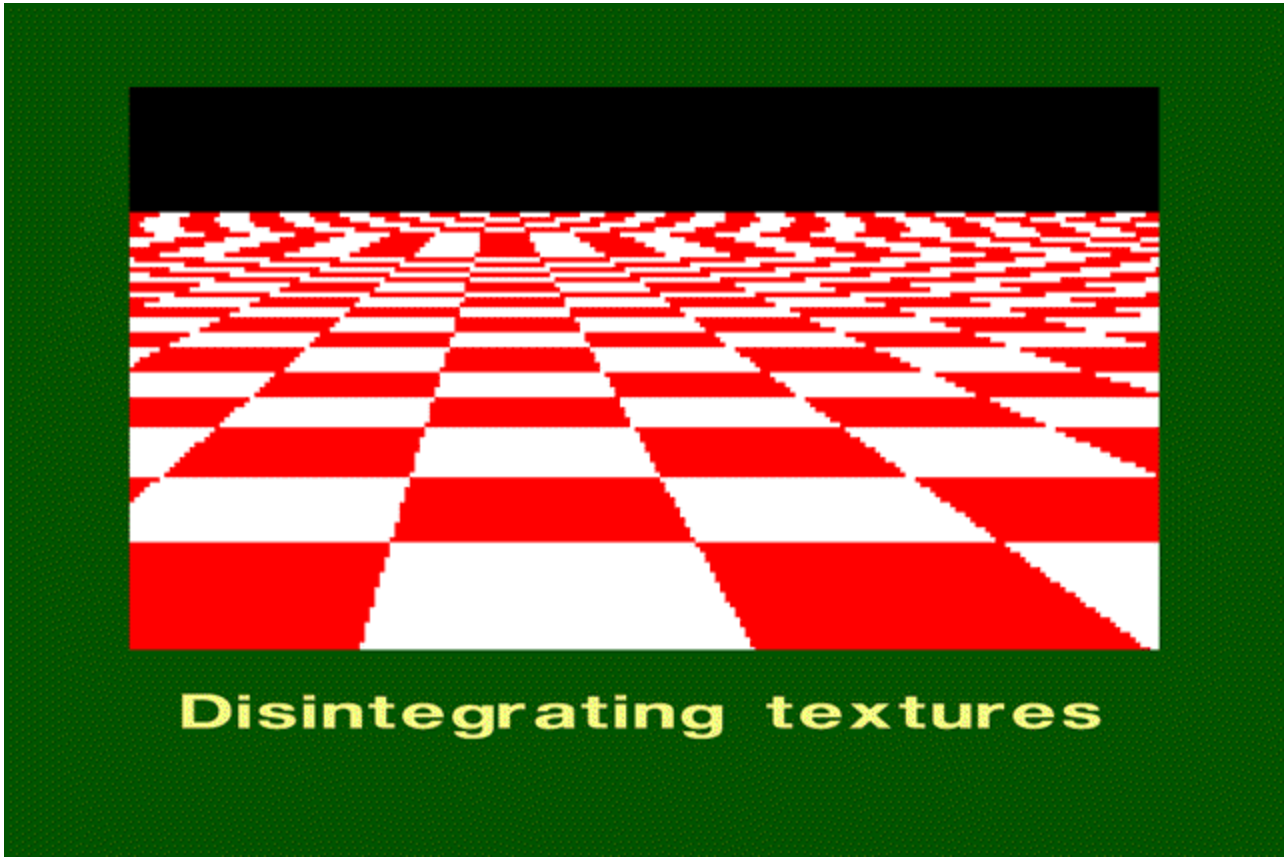
If camera shutter is only open for a fraction of a frame time (frame time = 1/30 sec. for video, 1/24 sec. for film):




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

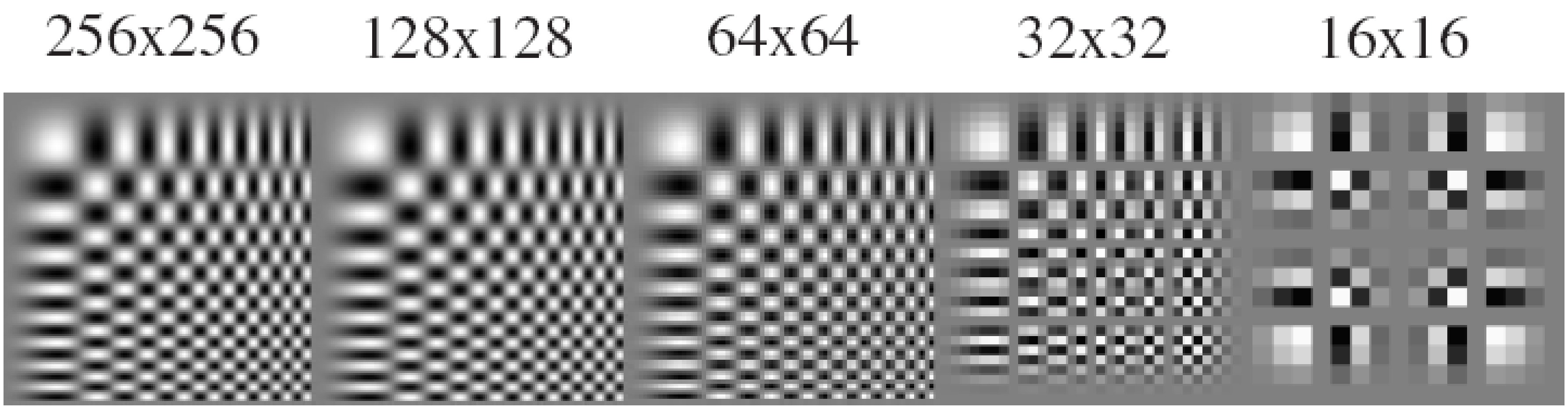
Slide by Steve Seitz

# Aliasing in graphics




Source: A. Efros

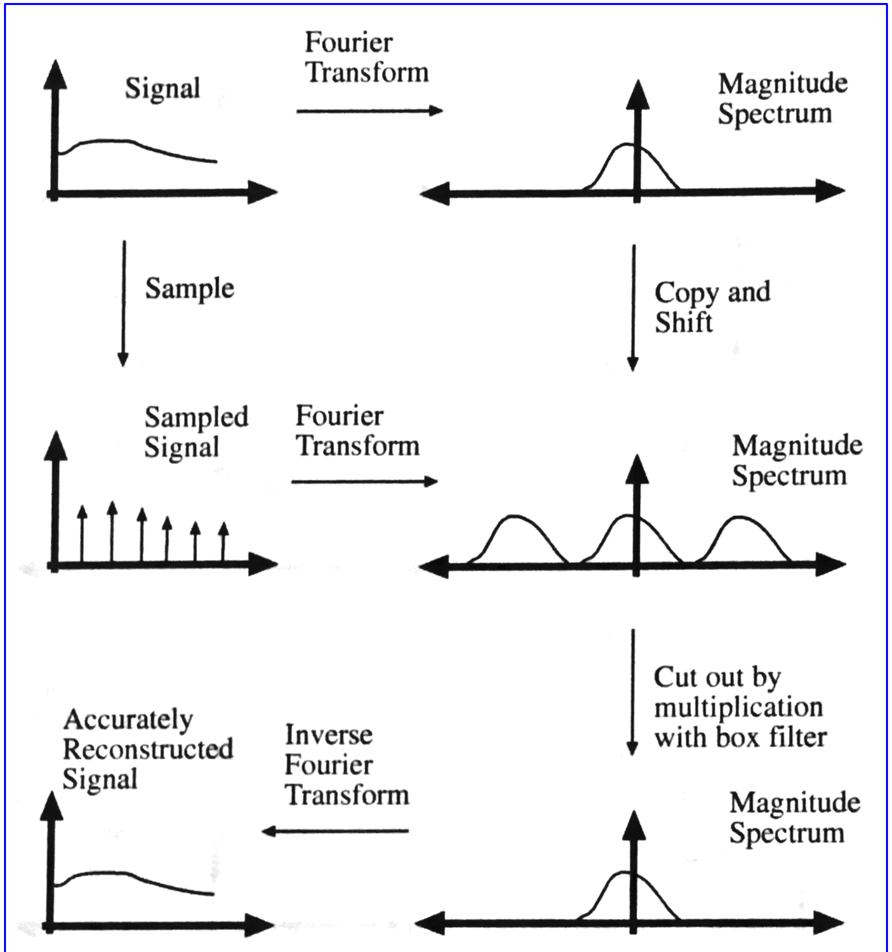
# Sampling and aliasing



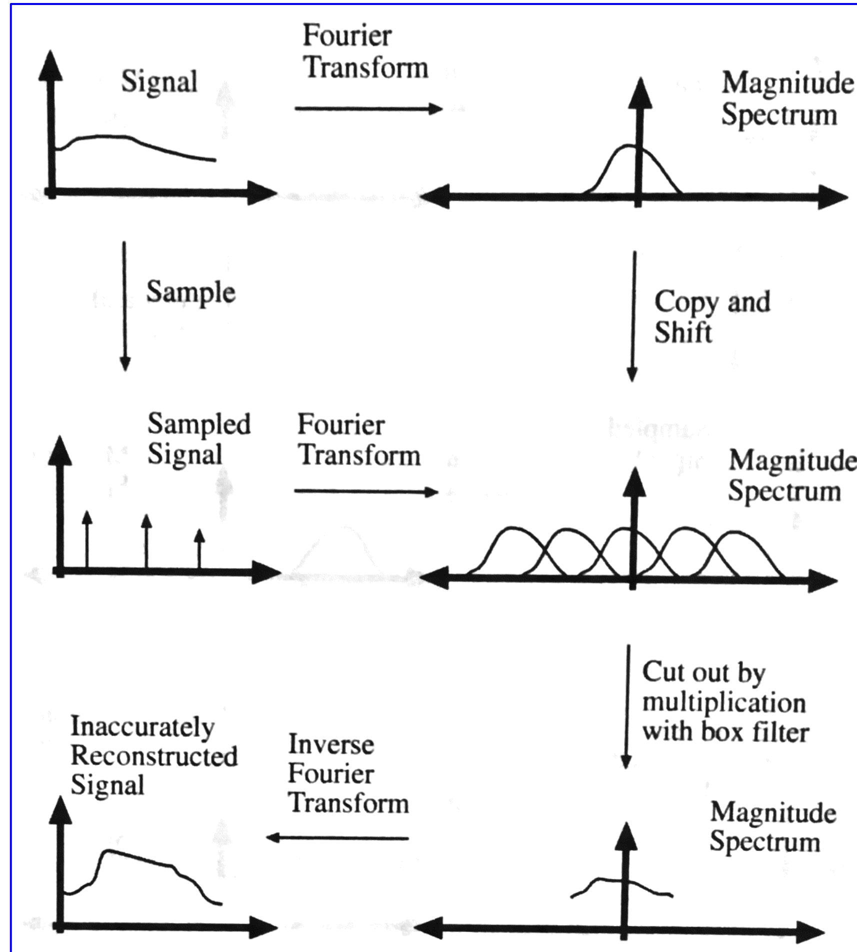

Slide from Derek Hoiem.

# Aliasing in Frequency Domain

No Aliasing



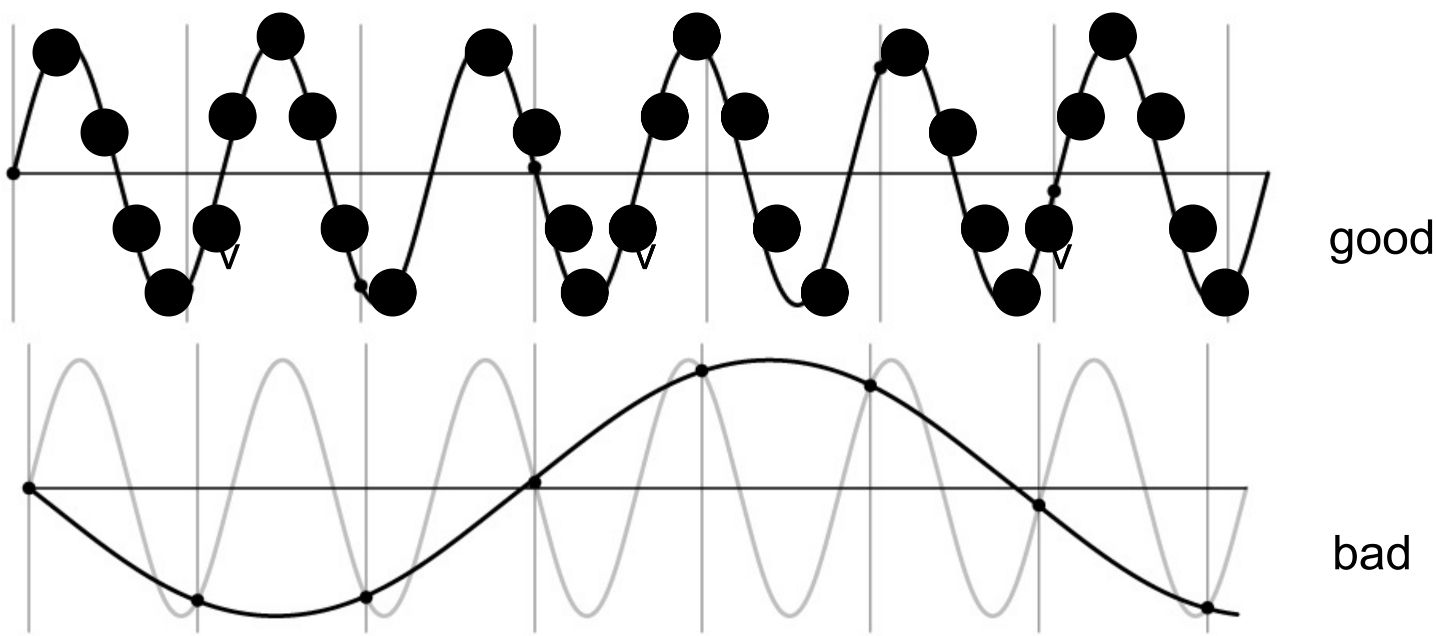

Aliasing




Source: Forsyth and Ponce

# 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




Slide from Derek Hoiem.

# Anti-aliasing

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

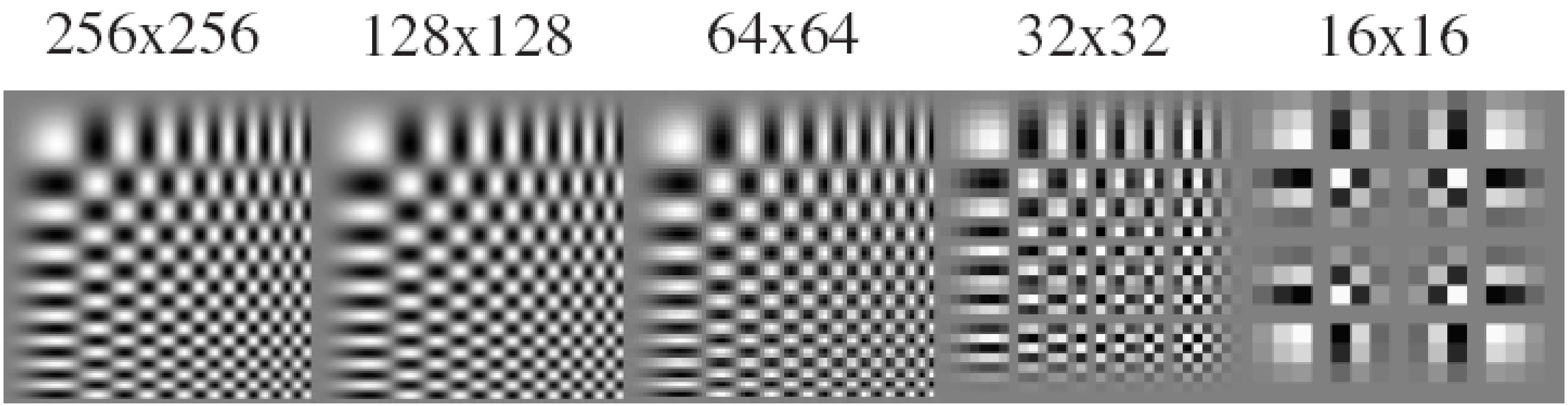
Slide from Derek Hoiem.

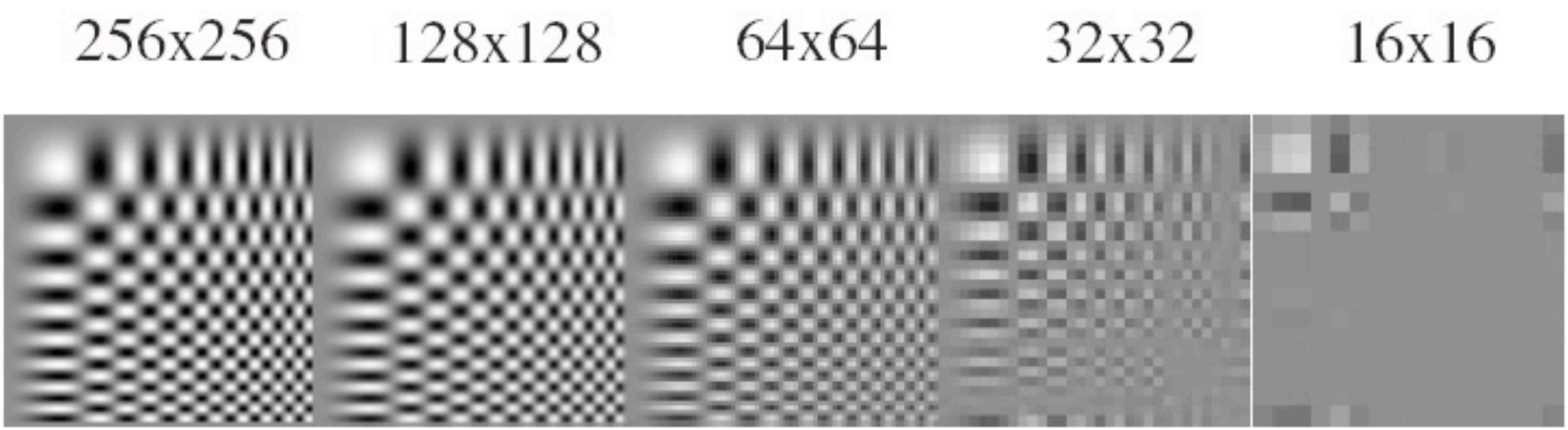
# Algorithm for downsampling by factor of 2

1. Start with image(h, w)
2. Apply low-pass filter
3. Sample every other pixel

Slide from Derek Hoiem.

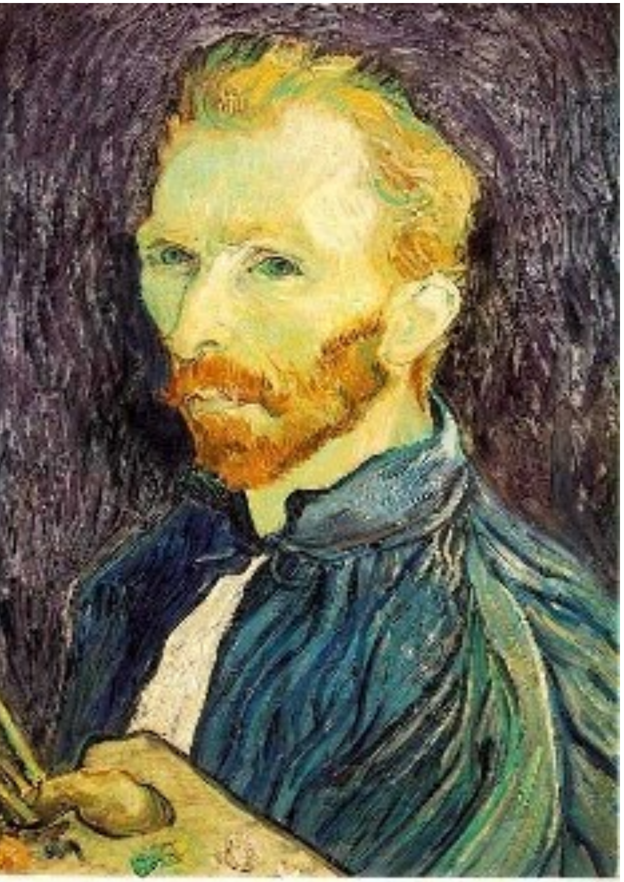
# Anti-aliasing




Forsyth and Ponce 2002

# Subsampling without pre-filtering



1/2

1/4 (2x zoom)

1/8 (4x zoom)

Slide by Steve Seitz

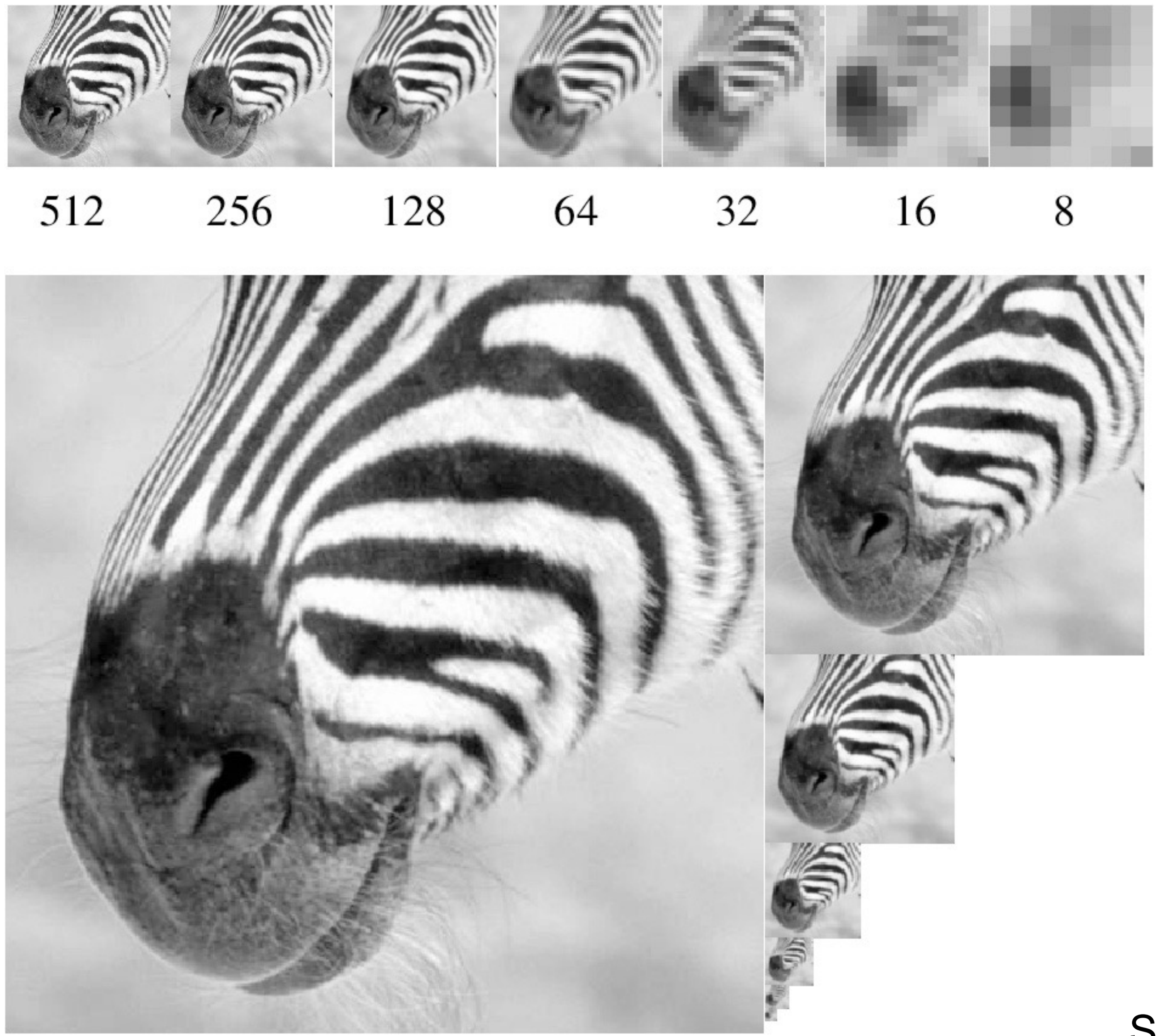
# Subsampling with Gaussian pre-filtering

Gaussian 1/2

G 1/4

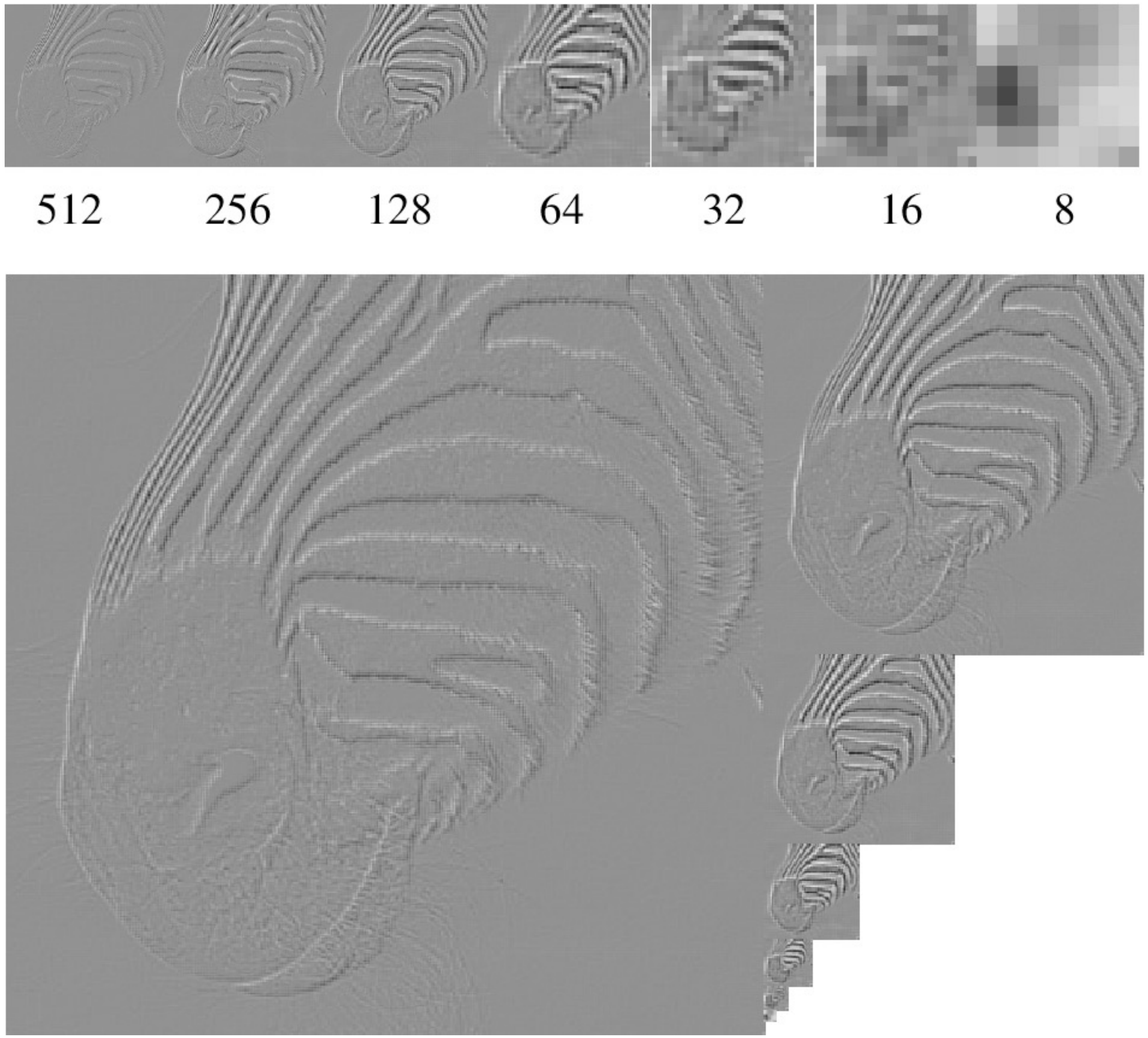
G 1/8

# Gaussian pyramid (Repeated blurring and sampling)



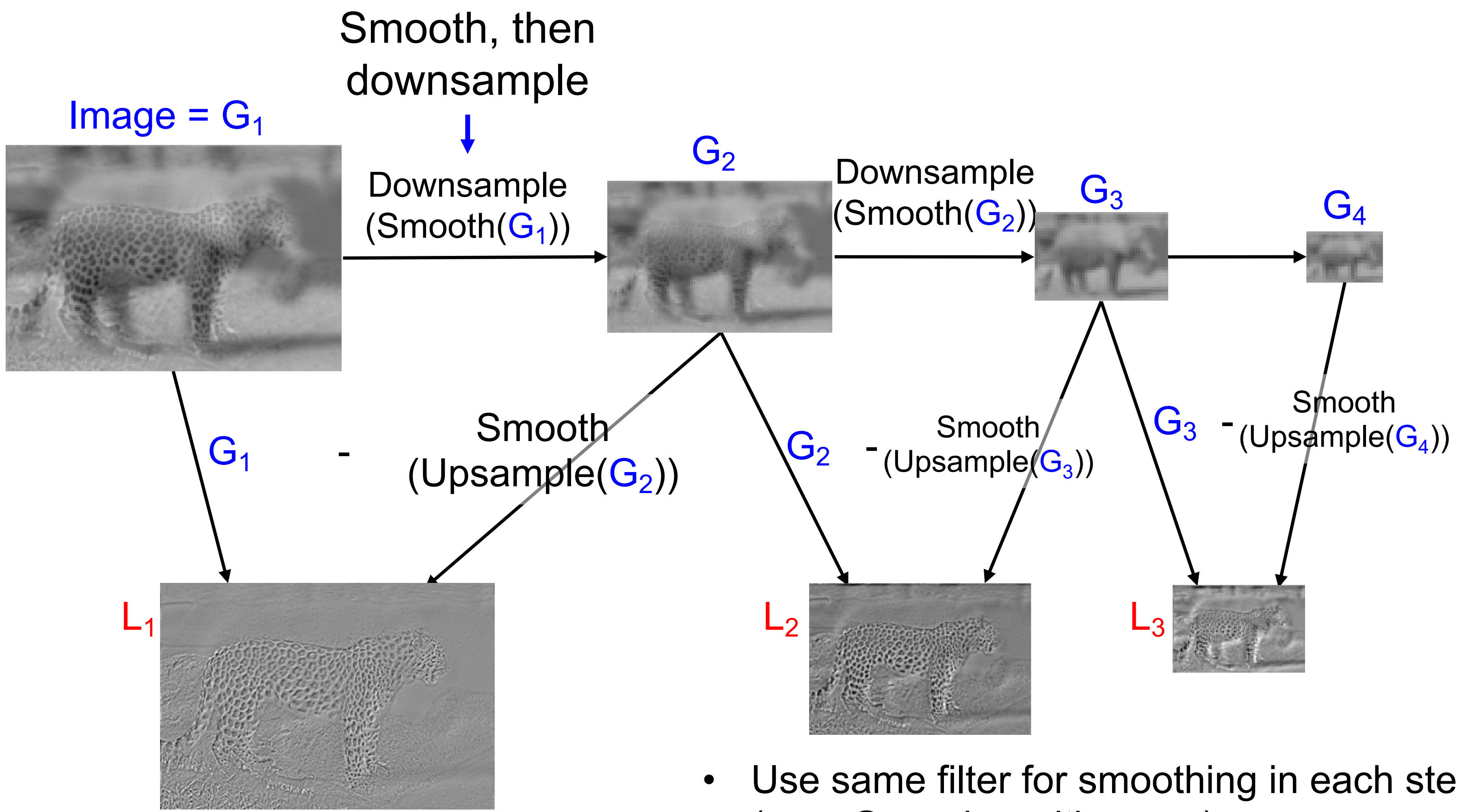

Source: Forsyth

# Laplacian pyramid




Source: Forsyth

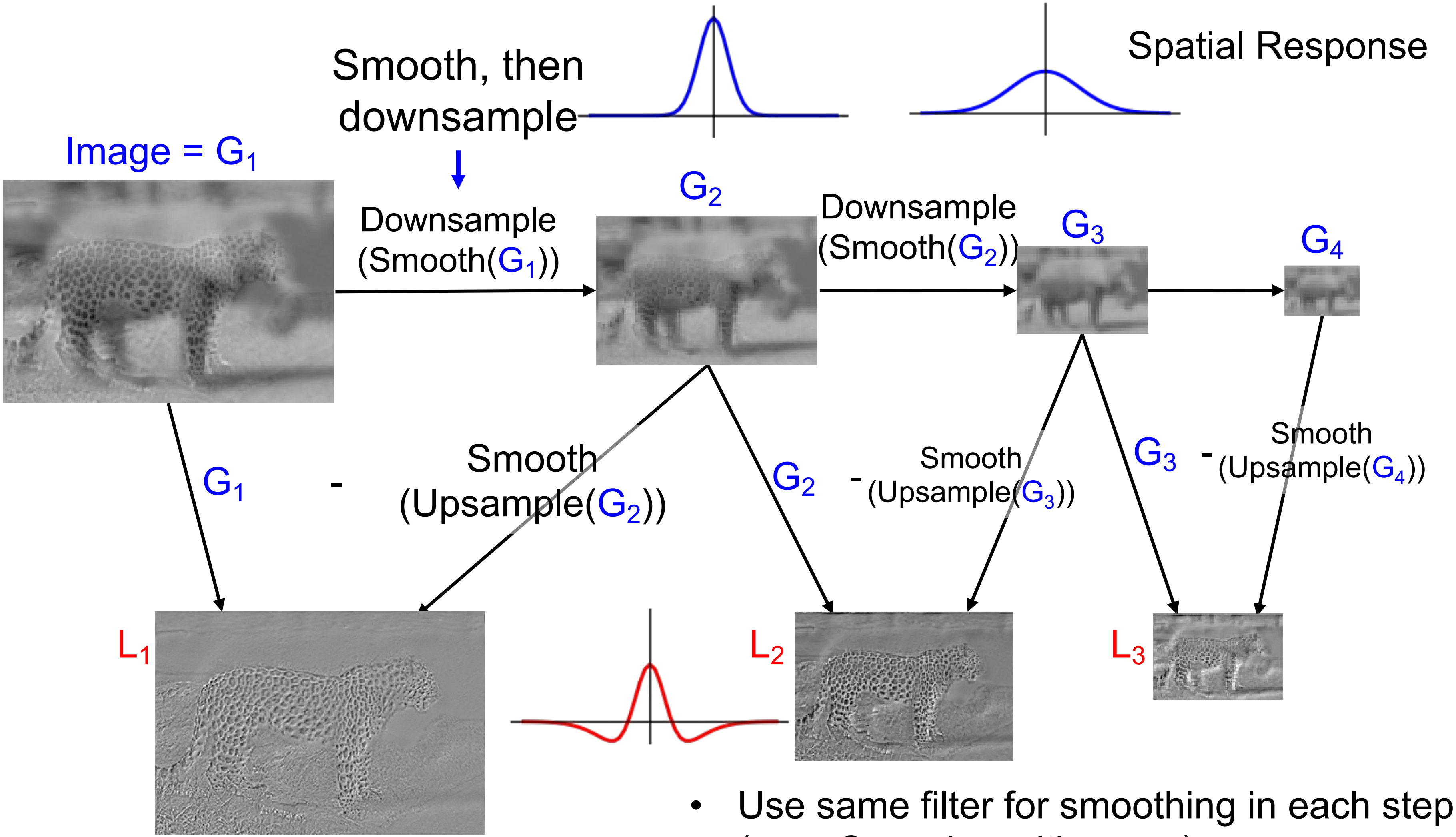
# Creating the Difference of Gaussian Pyramid




- Use same filter for smoothing in each step (e.g., Gaussian with $\sigma = 2$)
- Downsample/upsample with "nearest" interpolation

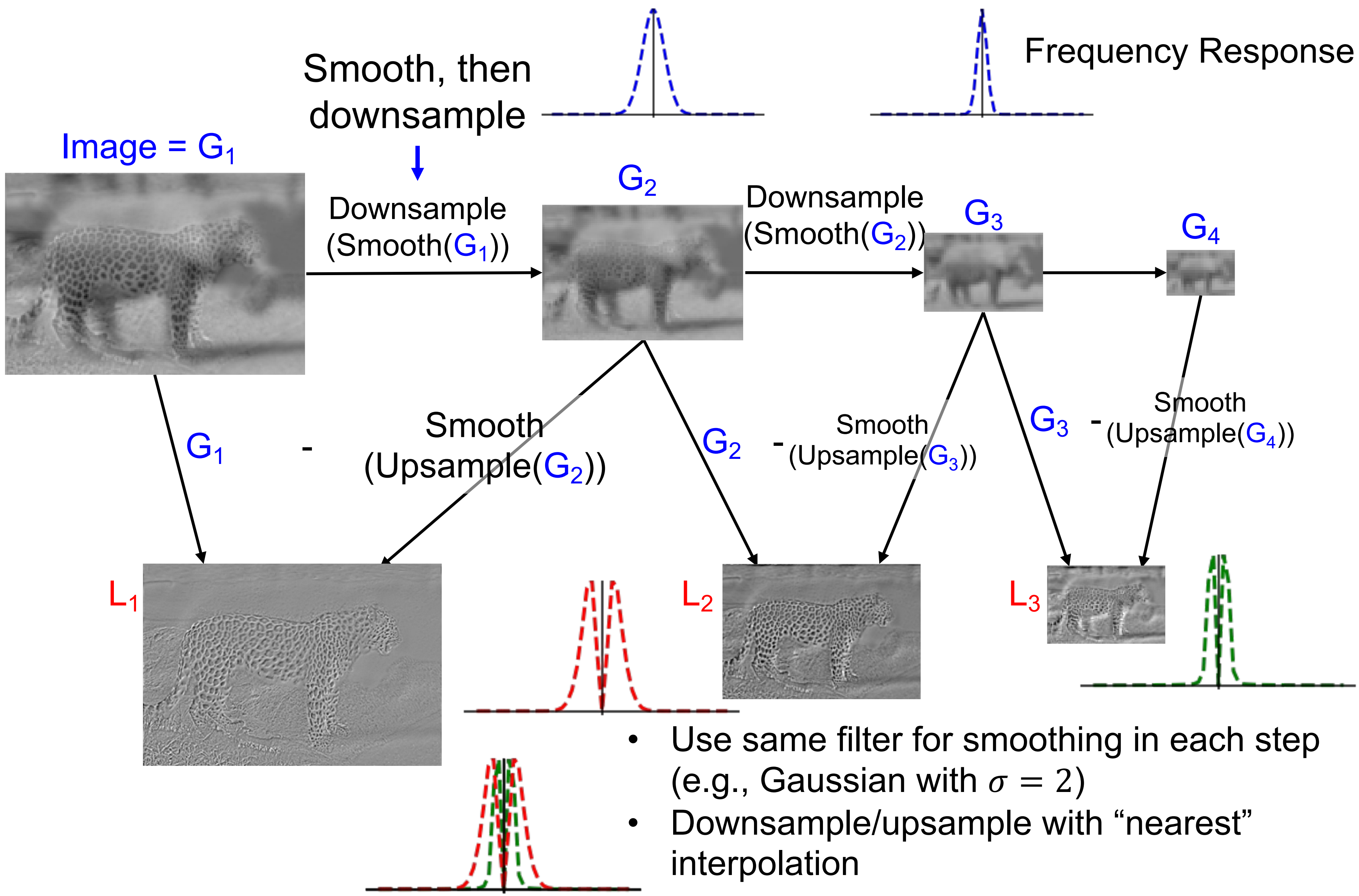
Leopard, Elephant image from Olivia and Torralba

# Creating the Difference of Gaussian Pyramid



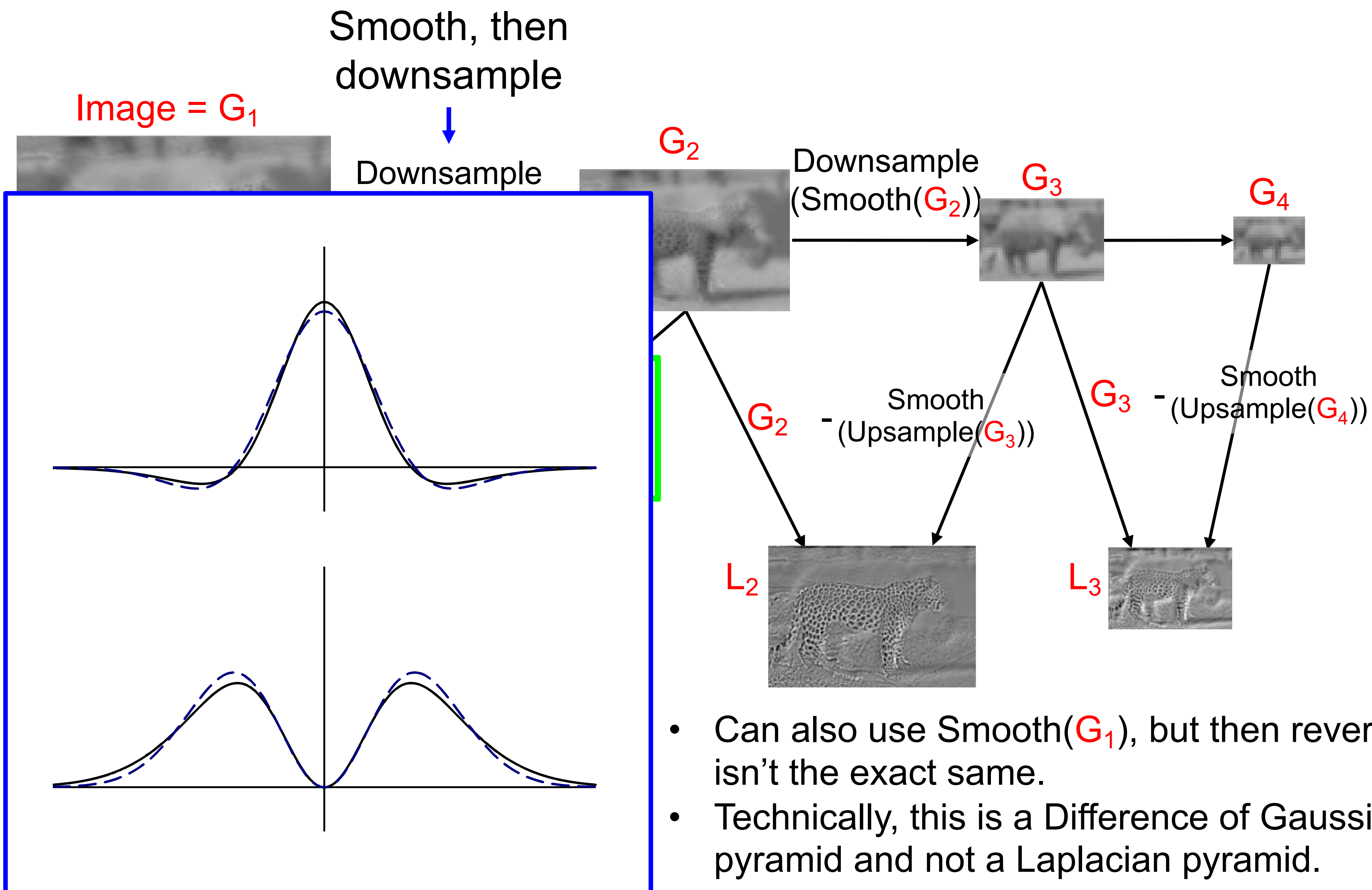

- Use same filter for smoothing in each step (e.g., Gaussian with $\sigma = 2$)
- Downsample/upsample with "nearest" interpolation

# Creating the Difference of Gaussian Pyramid



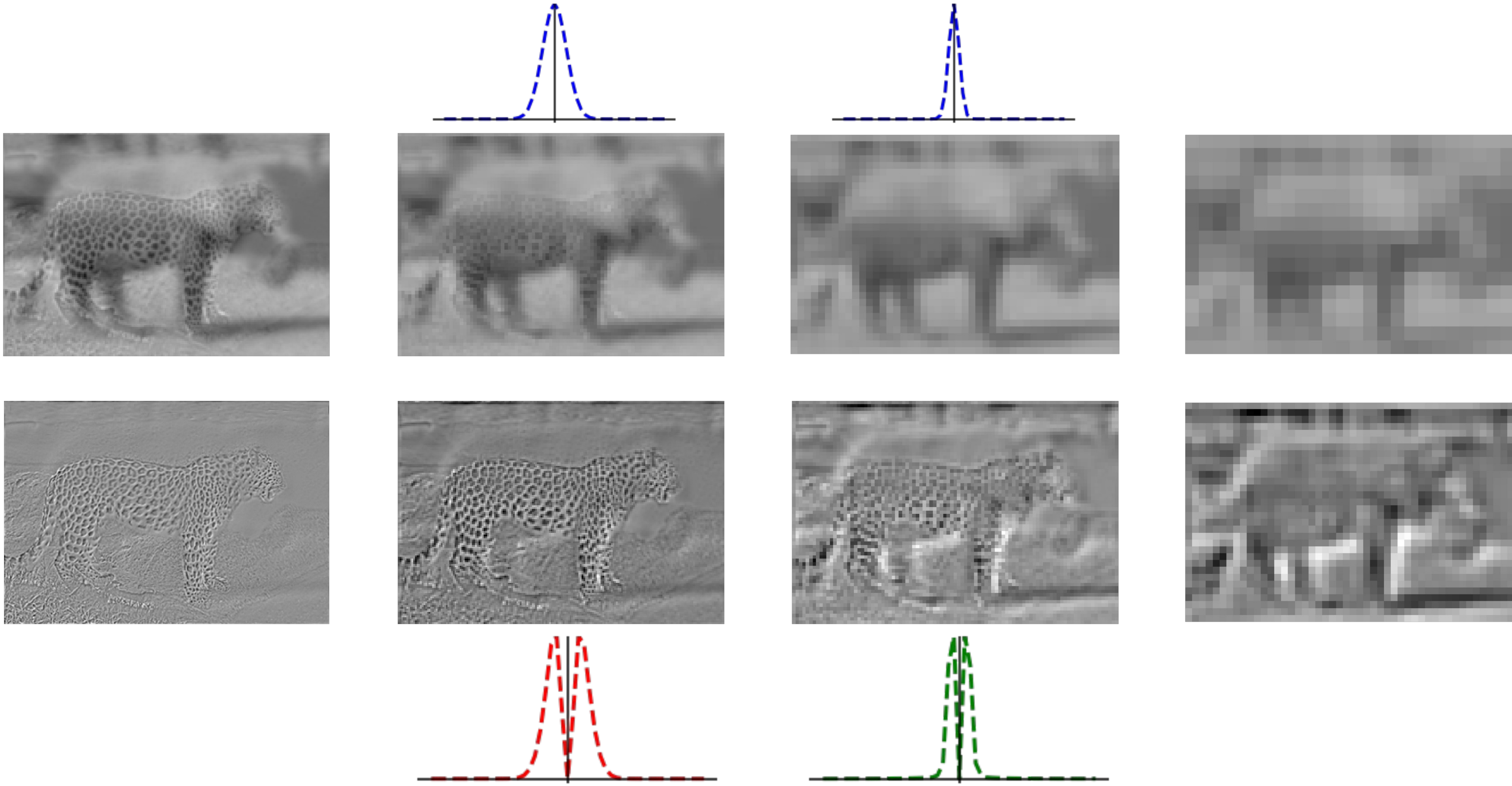

- Use same filter for smoothing in each step (e.g., Gaussian with $\sigma = 2$)
- Downsample/upsample with "nearest" interpolation

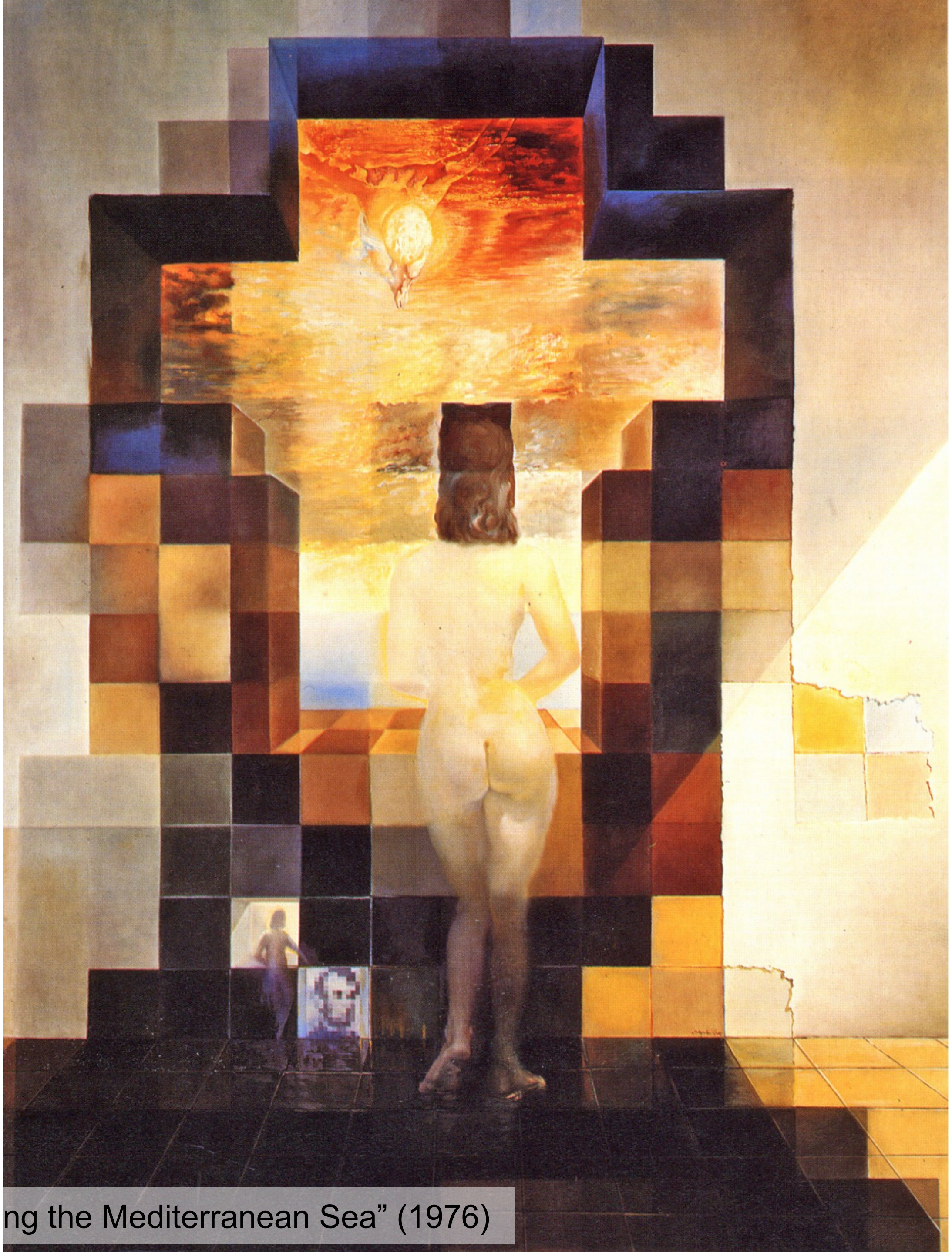
# Creating the Difference of Gaussian Pyramid




- Can also use Smooth($G_1$), but then reverse isn't the exact same.
- Technically, this is a Difference of Gaussian pyramid and not a Laplacian pyramid.

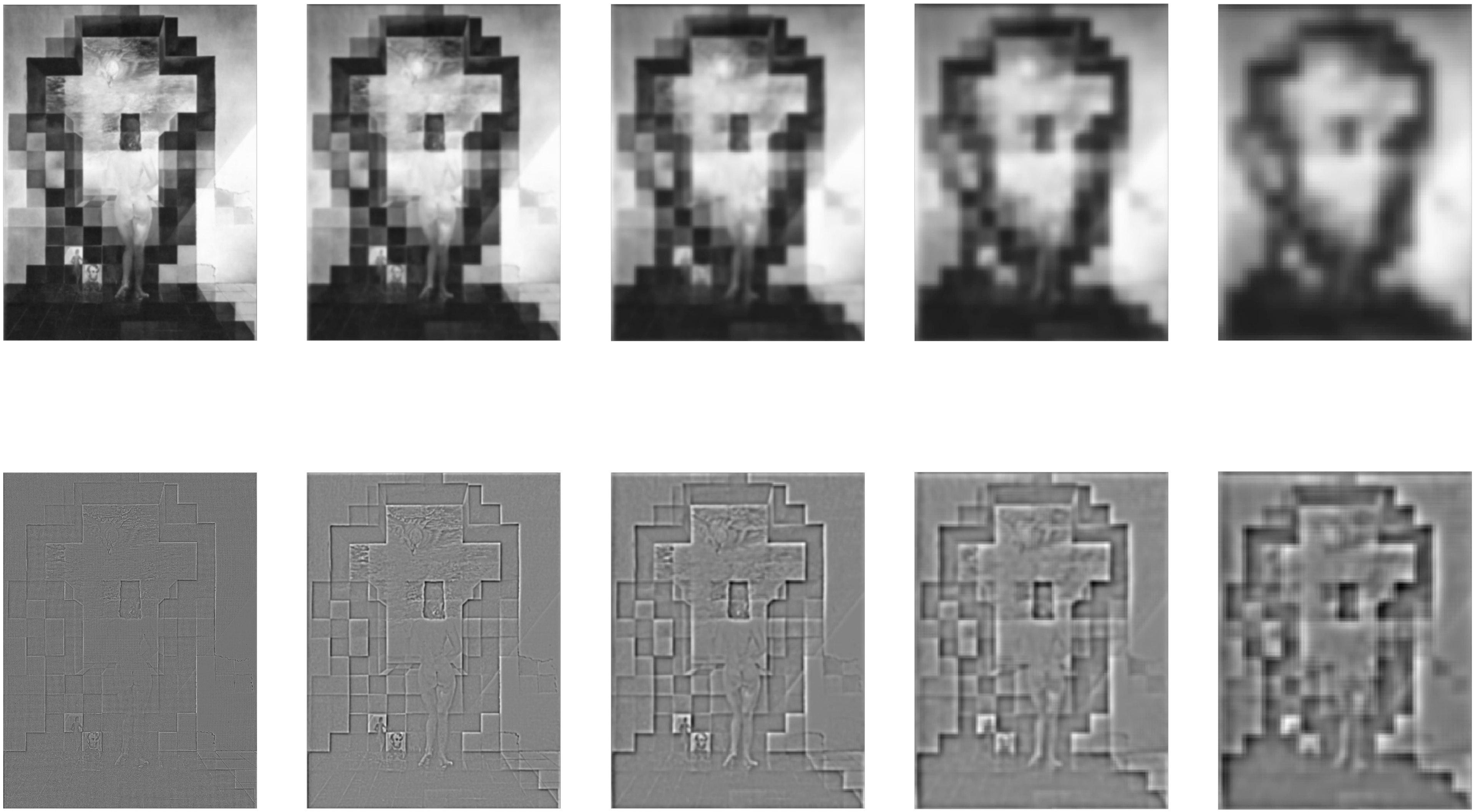
# Images in a Difference of Gaussian Pyramid





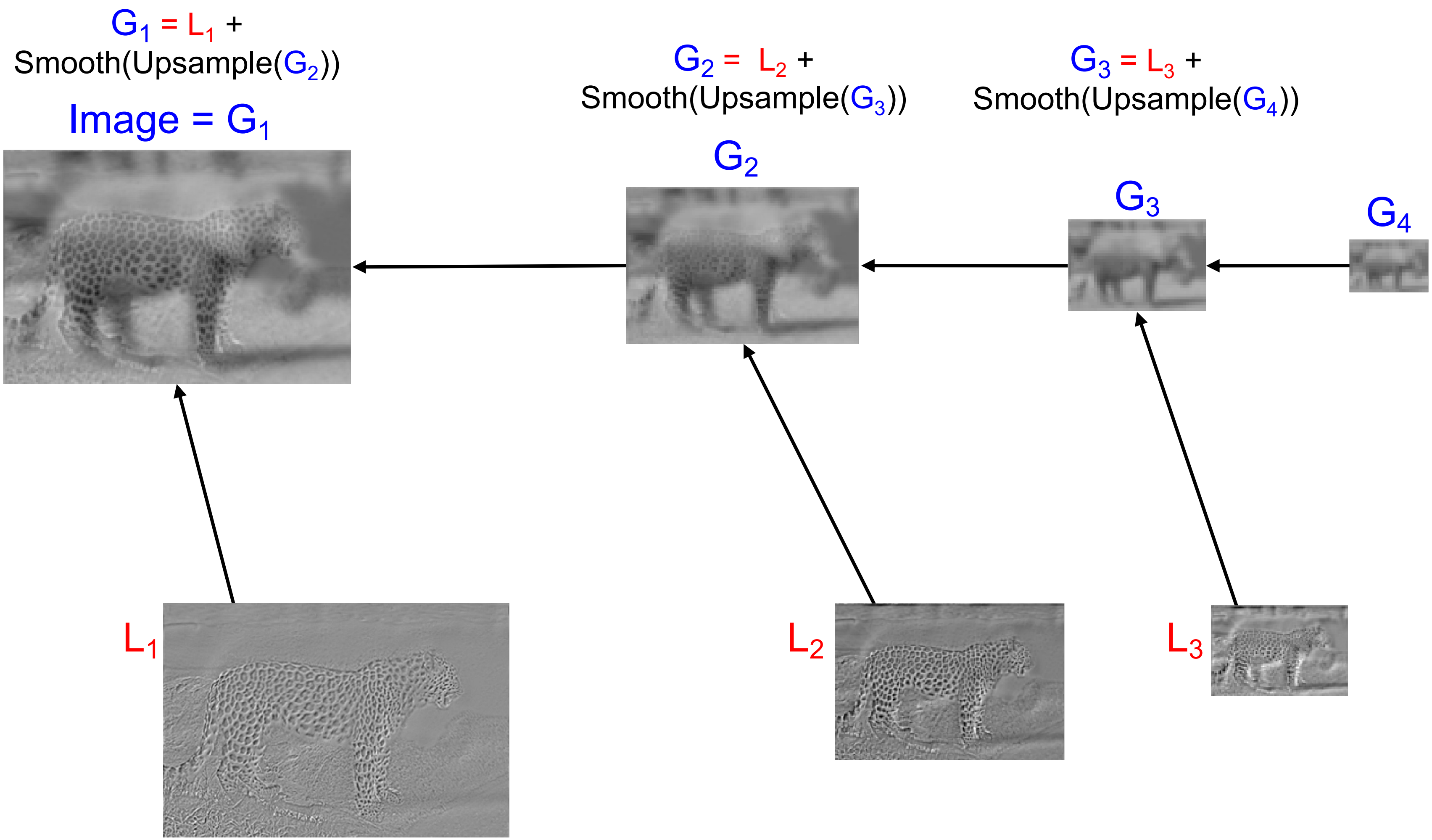
Dali: “Gala Contemplating the Mediterranean Sea” (1976)

# Images in a Difference of Gaussian Pyramid



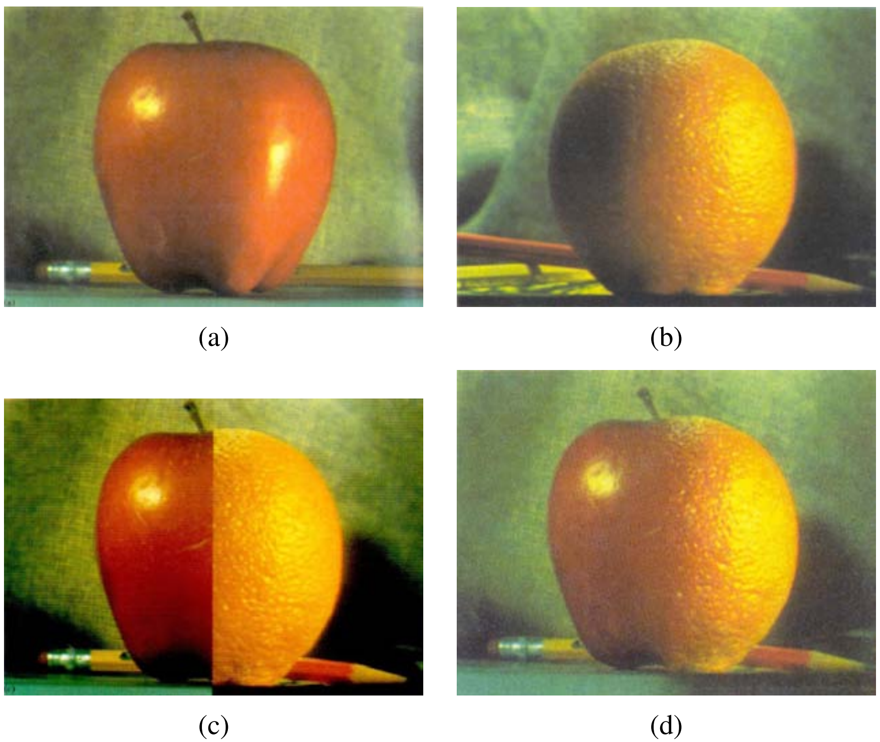
Dali: “Gala Contemplating the Mediterranean Sea” (1976)

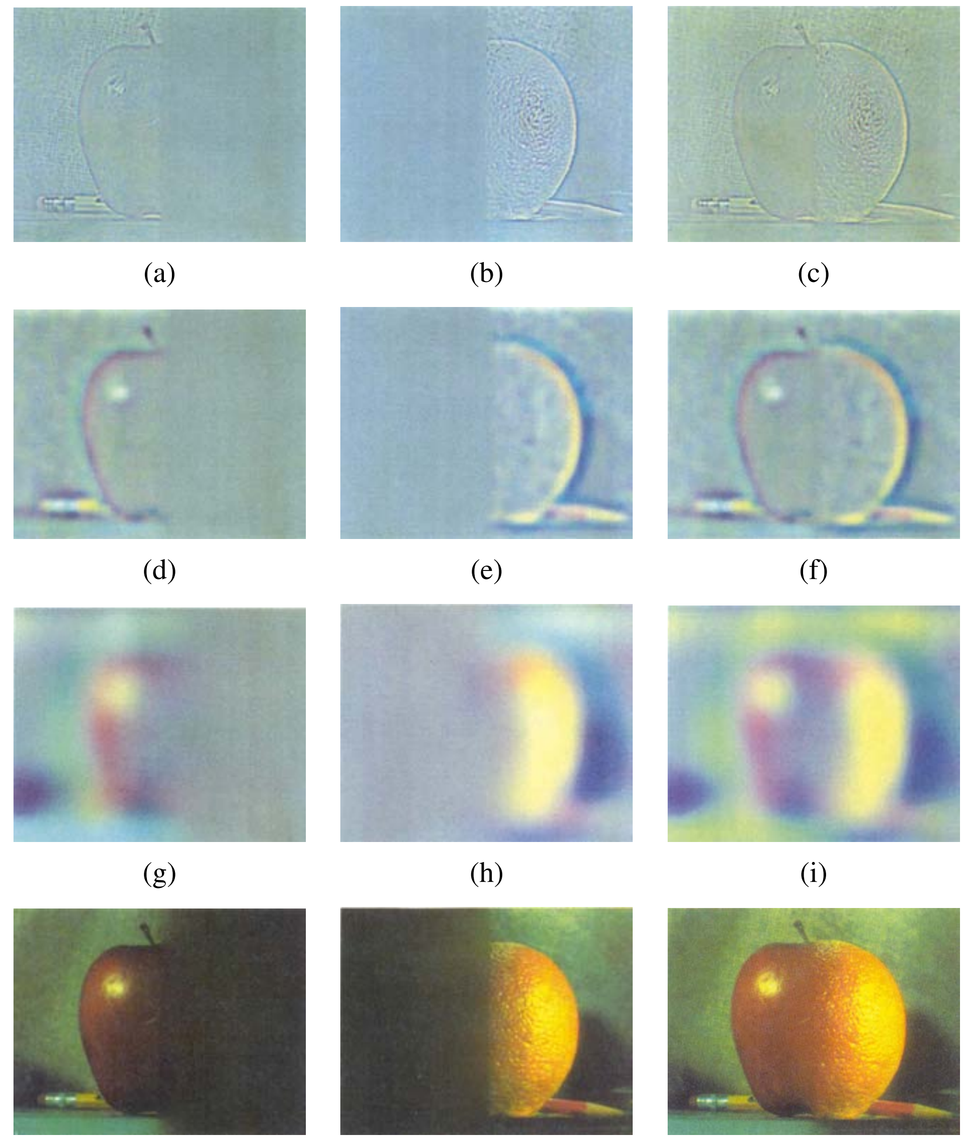
# Reconstructing from Diff of Gauss Pyramid




- Use same filter for smoothing as in deconstruction
- Upsample with “nearest” interpolation
- Reconstruction will be nearly lossless

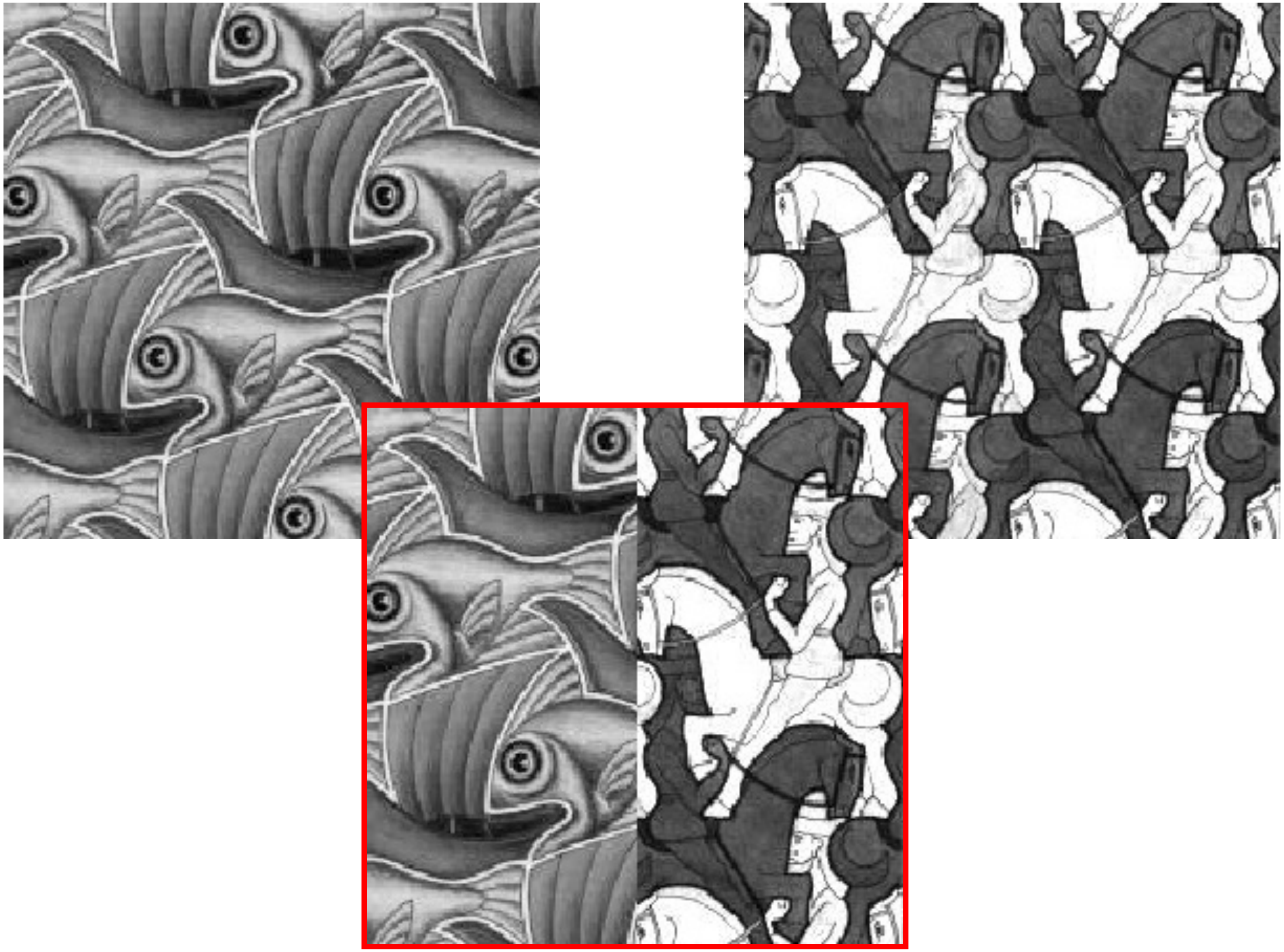
# Application: Image Blending



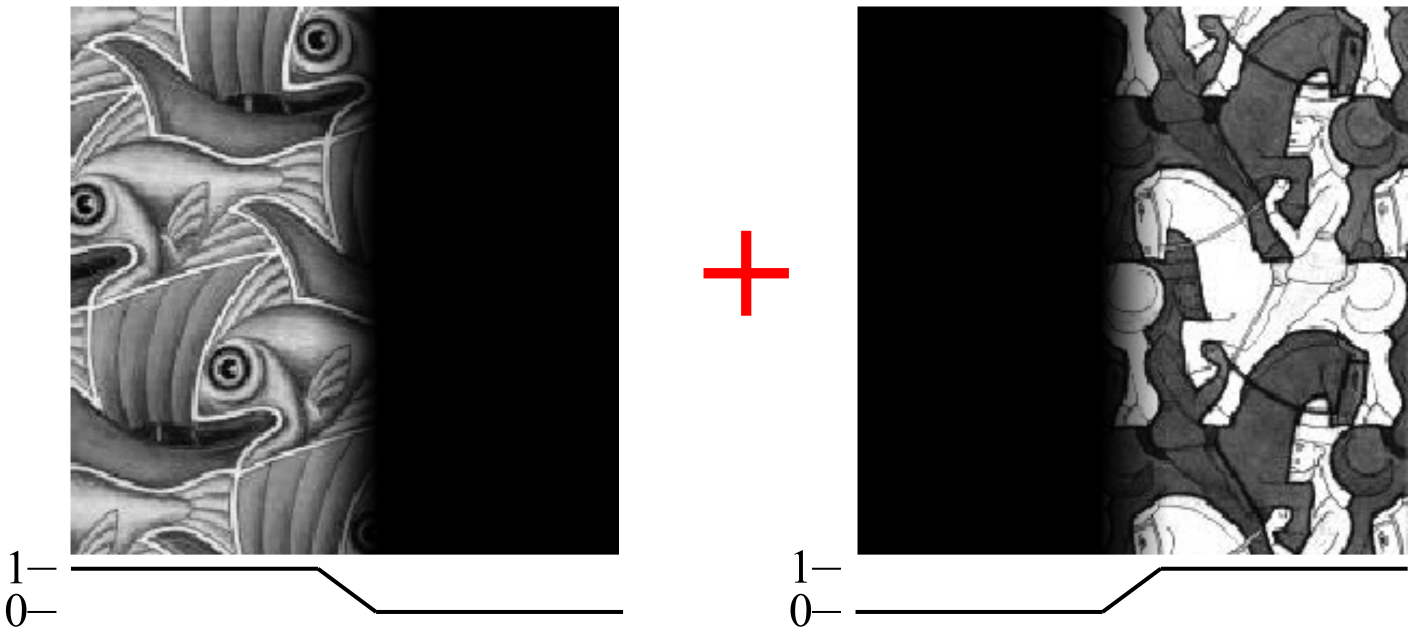



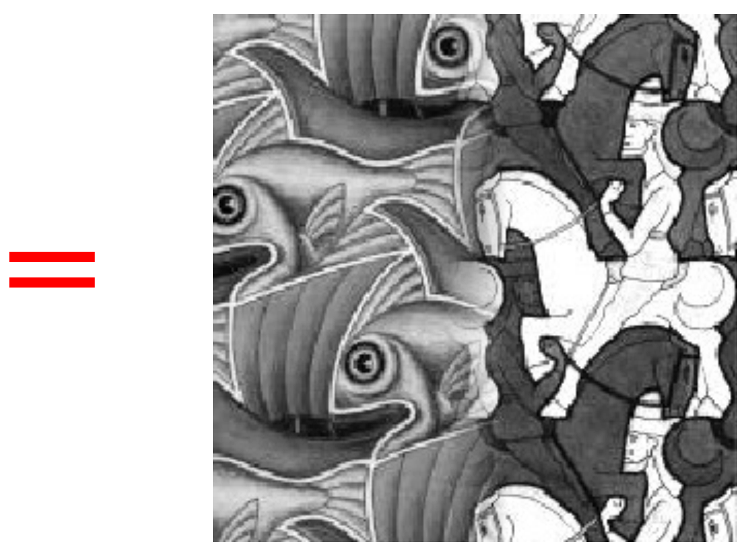

Laplacian pyramid blending (Burt and Adelson 1983b)

# Blending



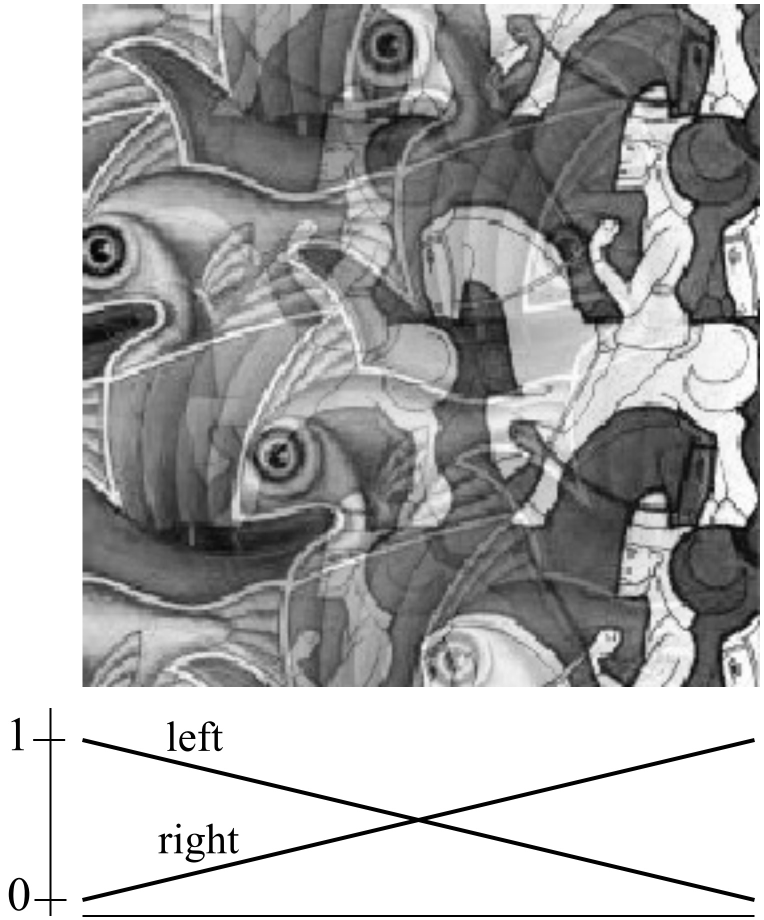
# Alpha Blending / Feathering

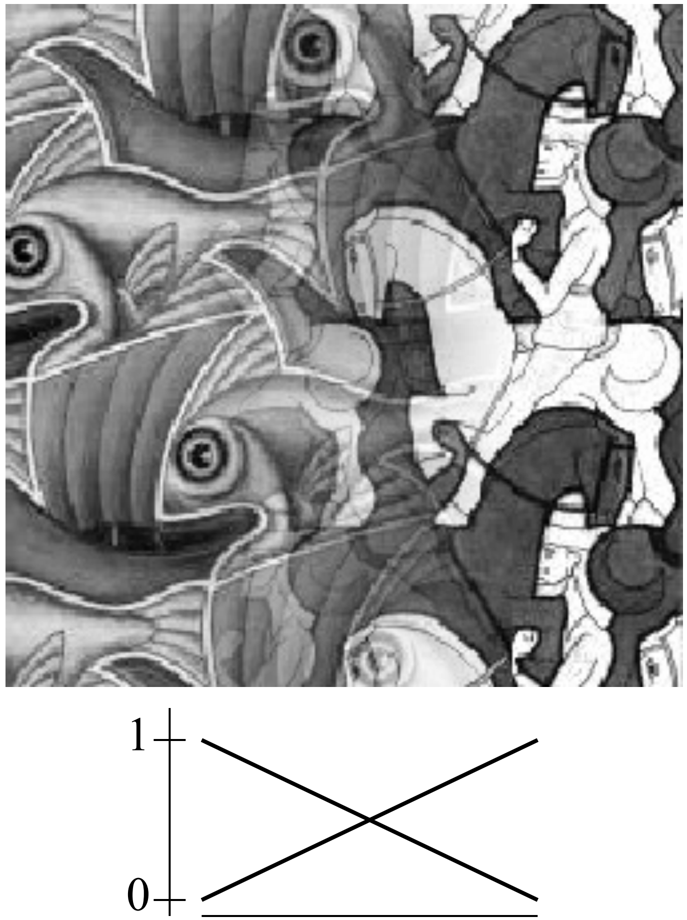


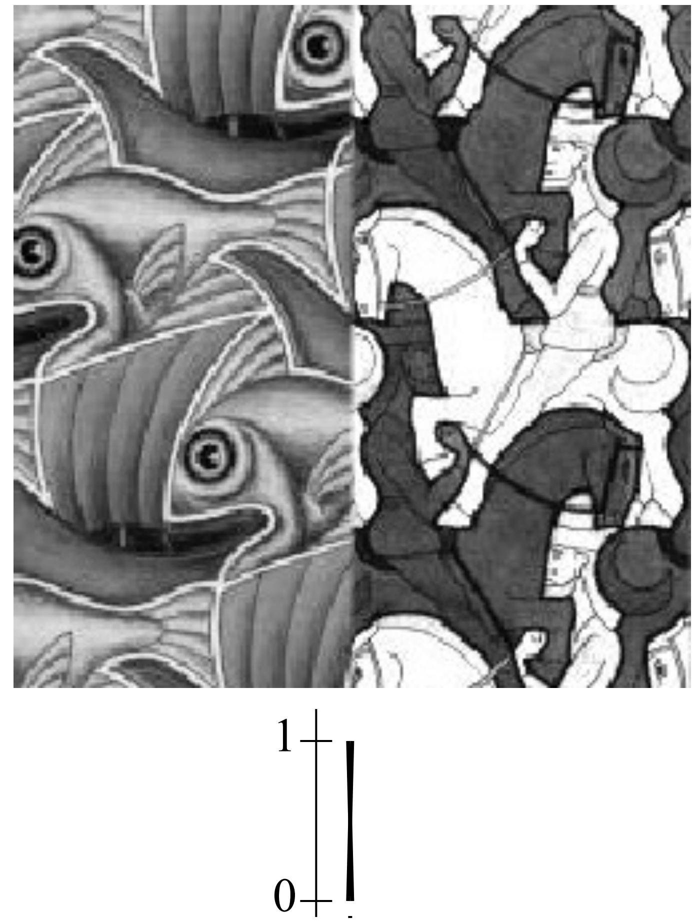
$$I_{blend} = \alpha I_{left} + (1-\alpha) I_{right}$$

# Affect of Window Size

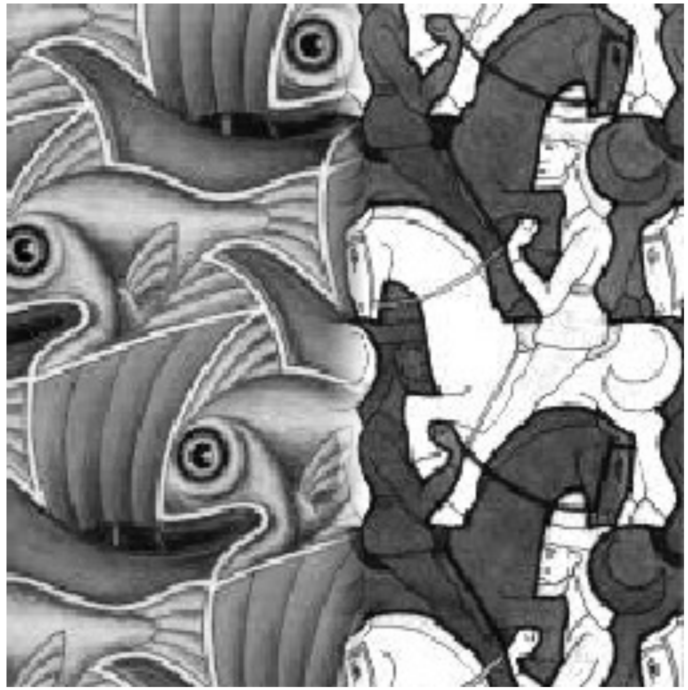


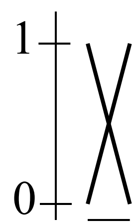



# Affect of Window Size

# Good Window Size





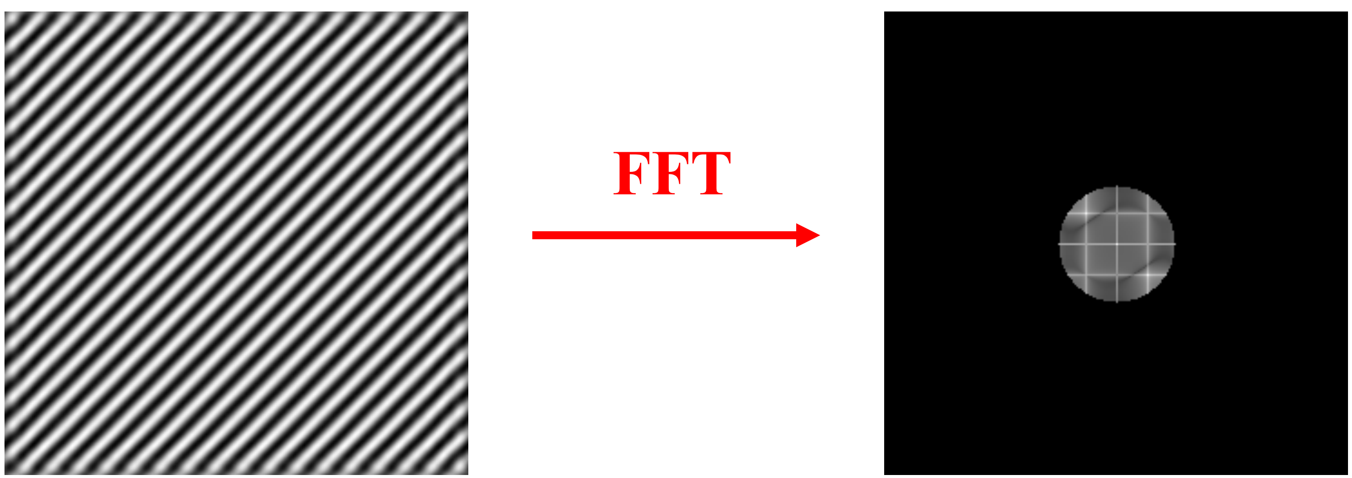

“Optimal” Window: smooth but not ghosted

# What is the Optimal Window?

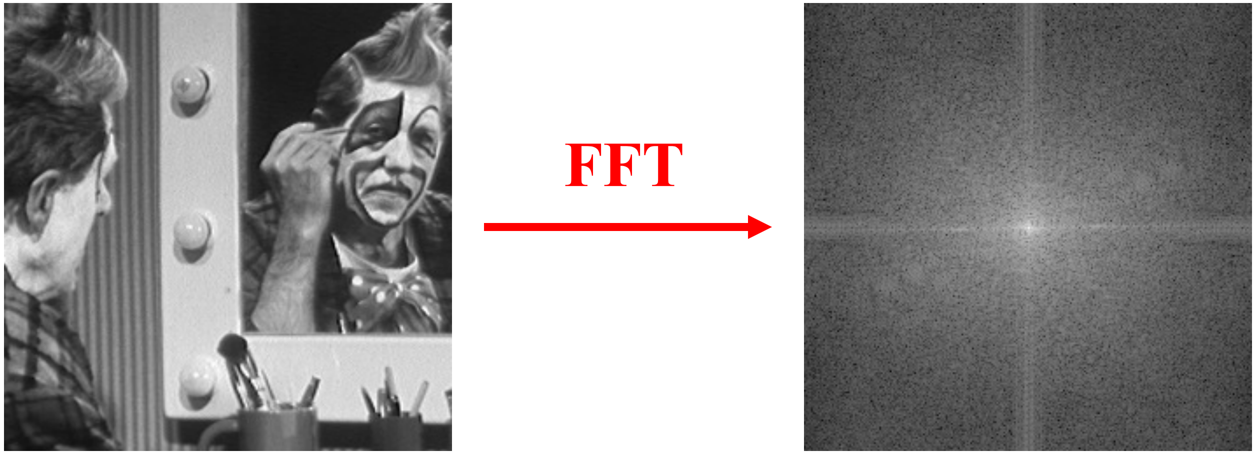
- To avoid seams
  - window = size of largest prominent feature
- To avoid ghosting
  - window <= 2*size of smallest prominent feature

## Natural to cast this in the *Fourier domain*

- largest frequency <= 2*size of smallest frequency
- image frequency content should occupy one “octave” (power of two)



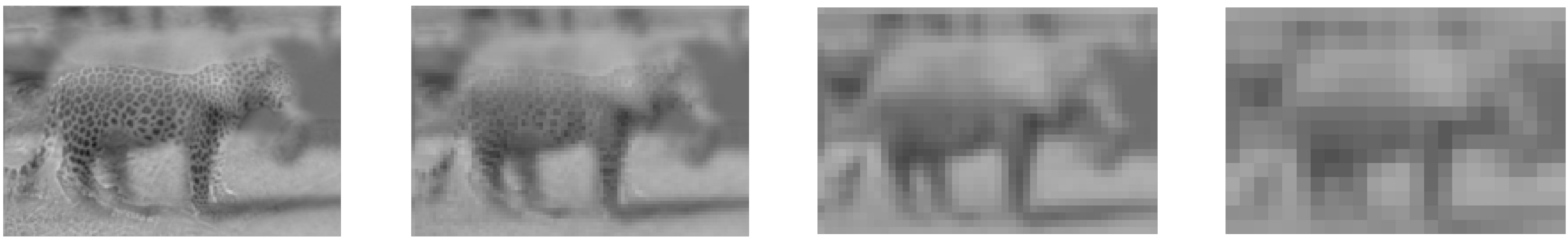
# What if the Frequency Spread is Wide

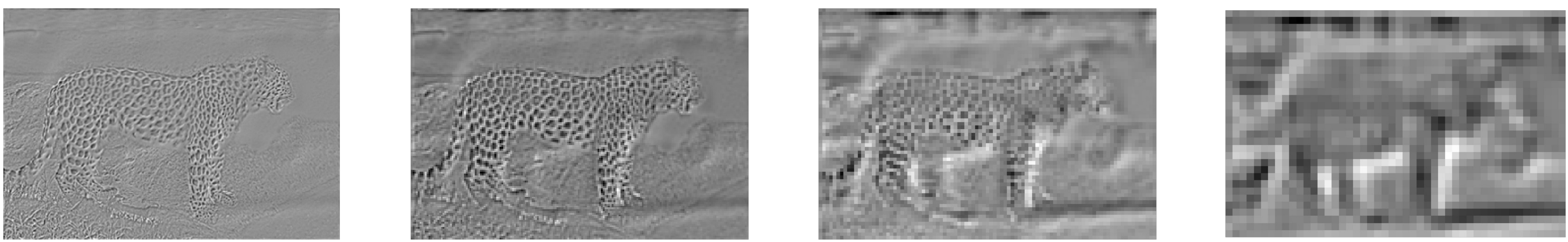



- Idea (Burt and Adelson)
  - Compute $F_{left} = FFT(I_{left})$, $F_{right} = FFT(I_{right})$
  - Decompose Fourier image into octaves (bands)
    - $F_{left} = F_{left}^1 + F_{left}^2 + \ldots$
  - Feather corresponding octaves $F_{left}^i$ with $F_{right}^i$
    - Can compute inverse FFT and feather in spatial domain
  - Sum feathered octave images in frequency domain
- Better implemented in *spatial domain*

# Octaves in the Spatial Domain

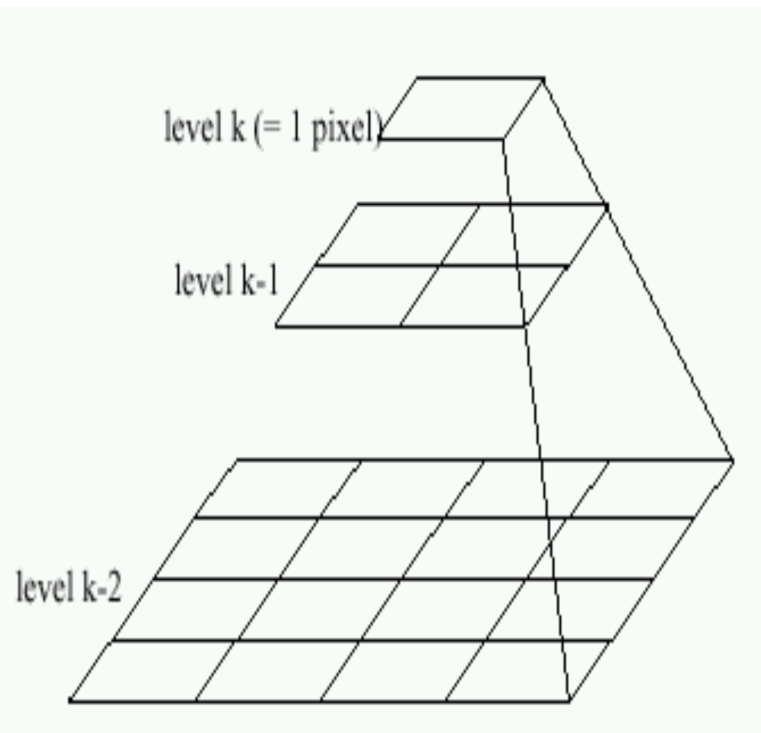
Lowpass Images



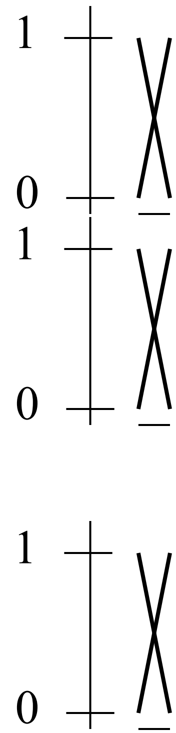


- Bandpass Images

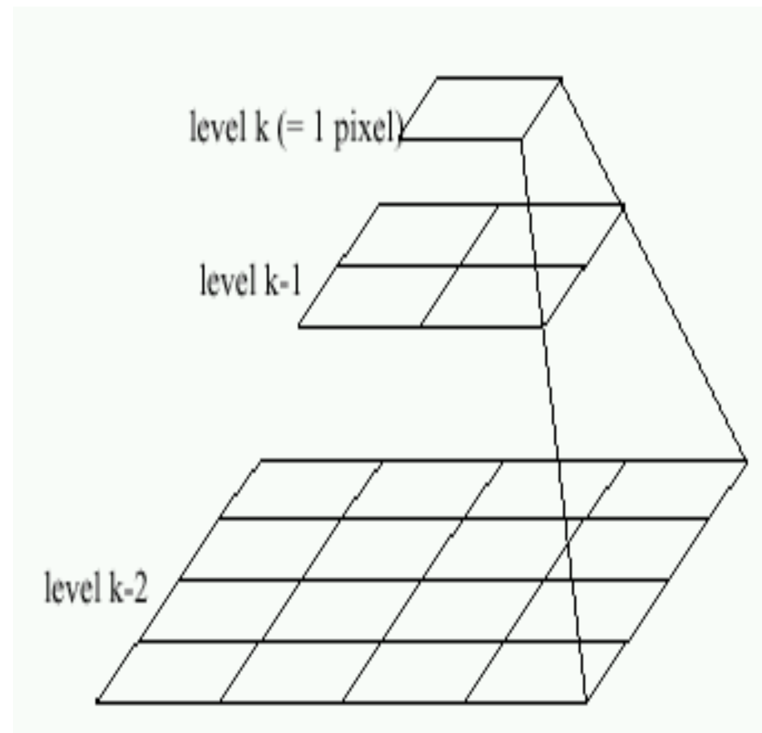
# Pyramid Blending




Left pyramid

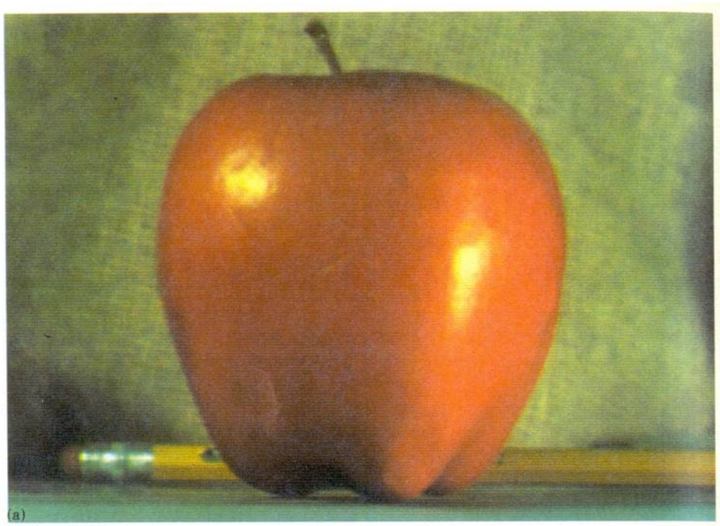


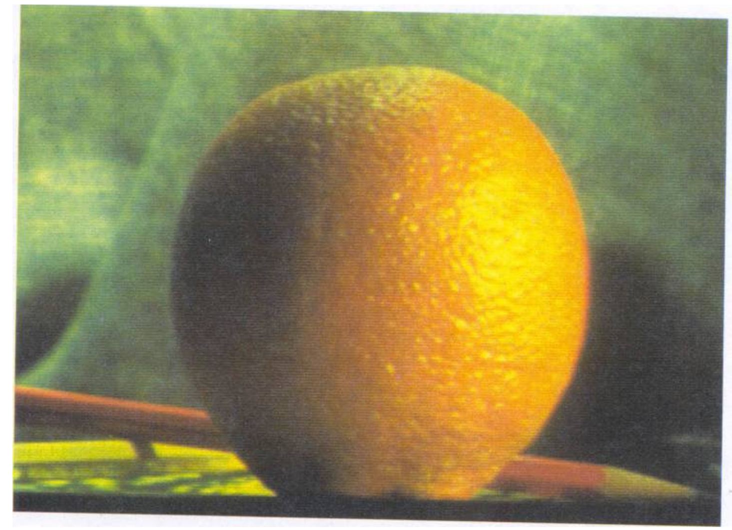

blend

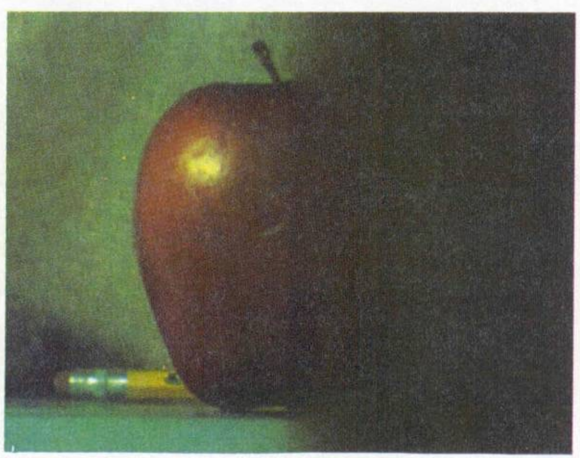


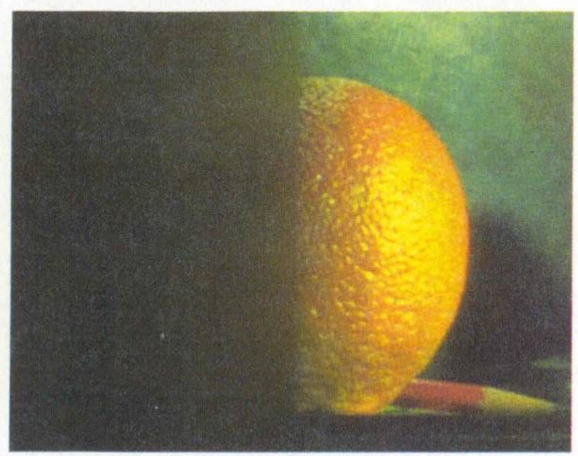

Right pyramid

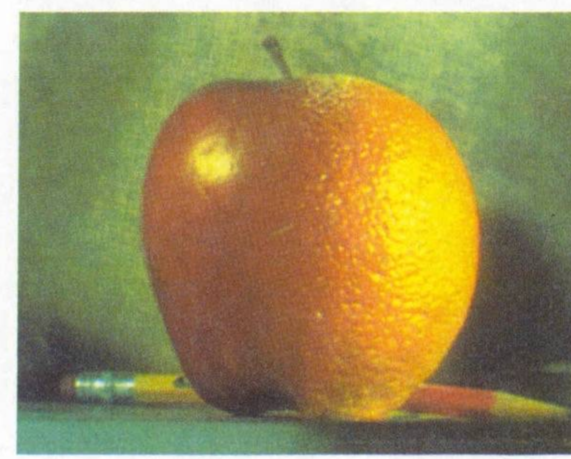
# Pyramid Blending

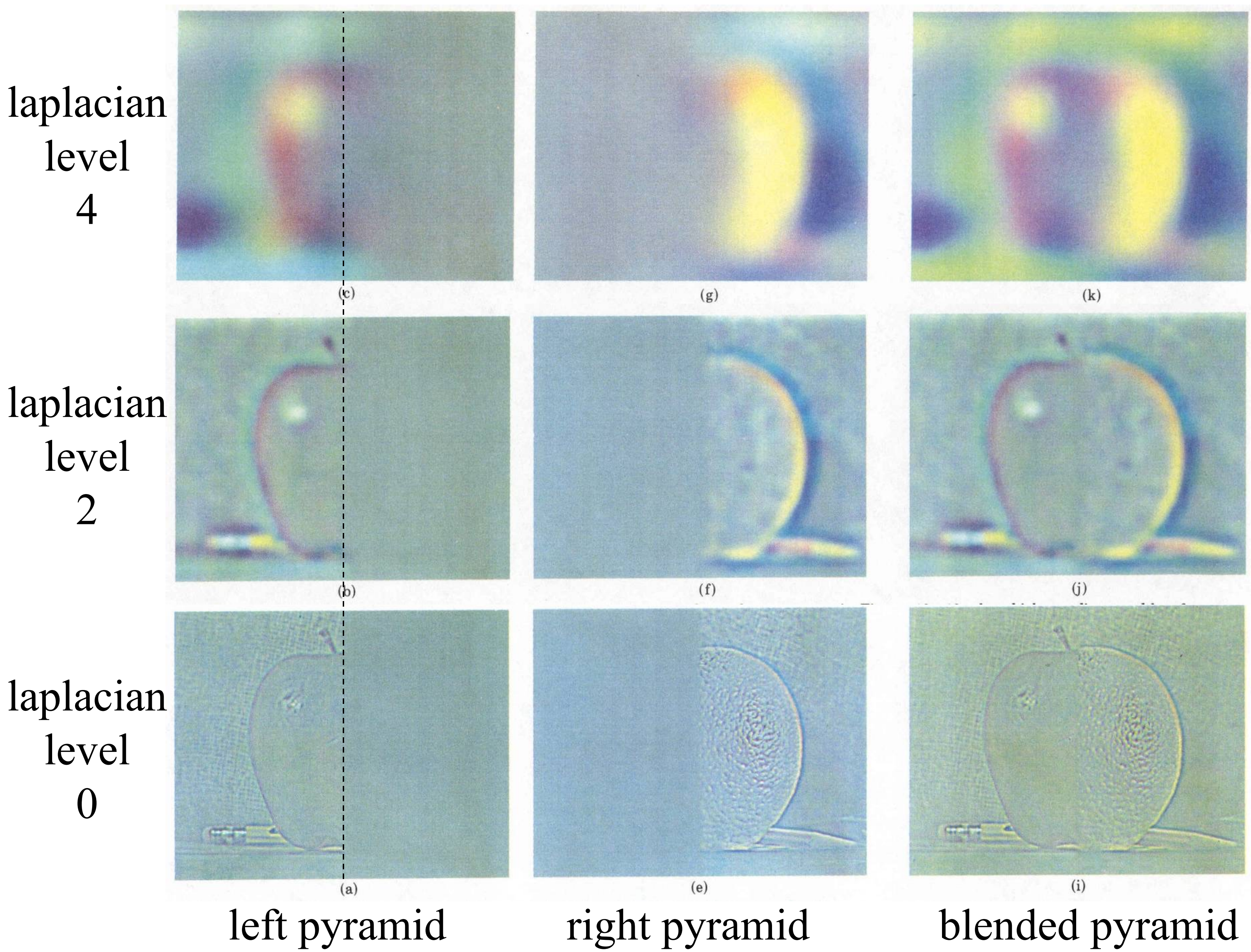



(d)

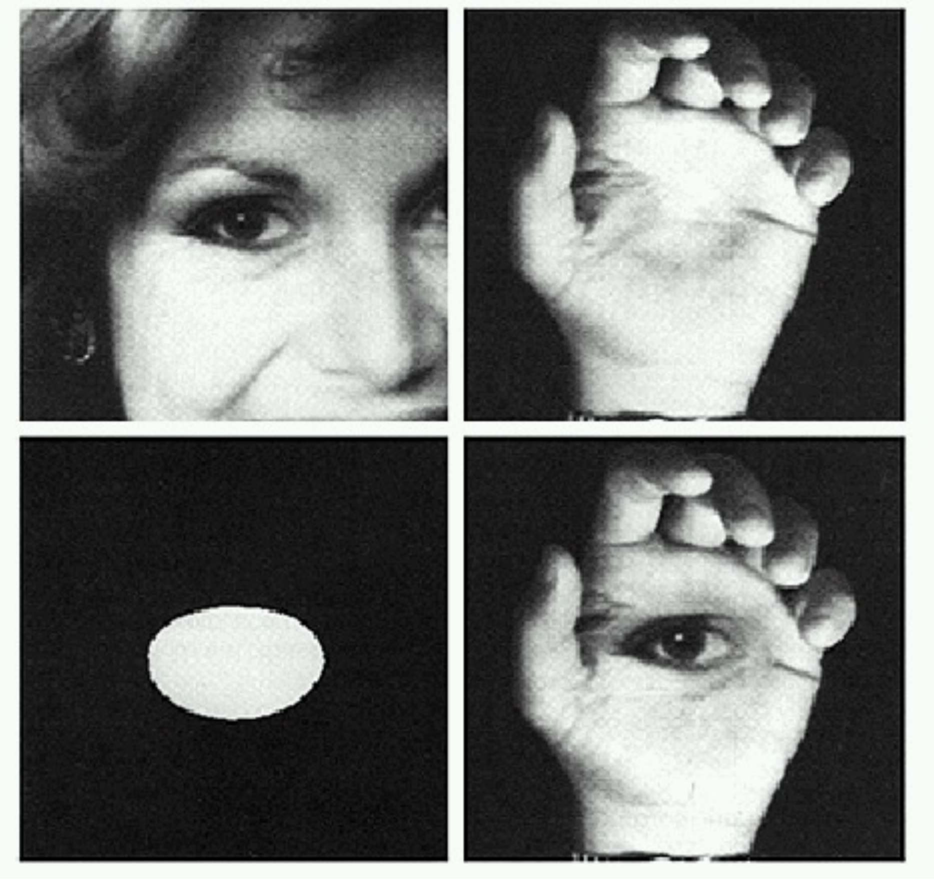

(h)


(l)



laplacian level 4
(c)
(g)
(k)
laplacian level 2
(b)
(f)
(j)
laplacian level 0
(a)
(e)
(i)
left pyramid
right pyramid
blended pyramid

# Blending Regions



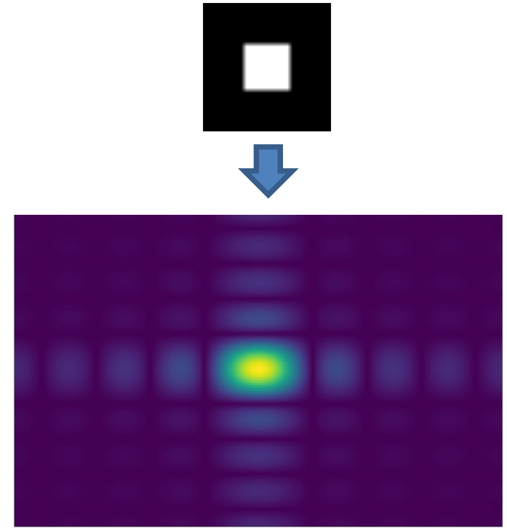
# Laplacian Pyramid: Blending

- General Approach:
  1. Build Laplacian pyramids *LA* and *LB* from images *A* and *B*
  2. Build a Gaussian pyramid *GR* from selected region *R*
  3. Form a combined pyramid *LS* from *LA* and *LB* using nodes of *GR* as weights:
     - $LS(i,j) = GR(I,j,)*LA(I,j) + (1-GR(I,j))*LB(I,j)$
  4. Collapse the *LS* pyramid to get the final blended image

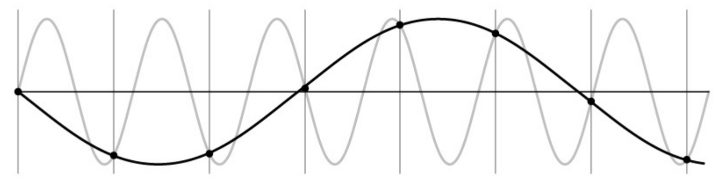
# Major uses of image pyramids

- Compression
- Object detection
  - Scale search
  - Features
- Detecting stable interest points
- Registration
  - Course-to-fine

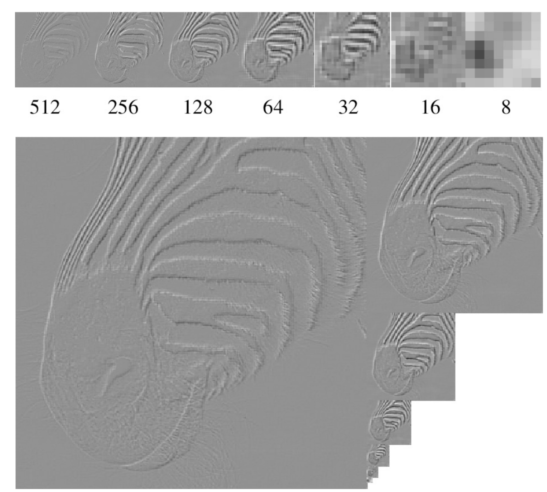
# Recap



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



- Sampling and Aliasing




- Image Pyramids