The Frequency Domain

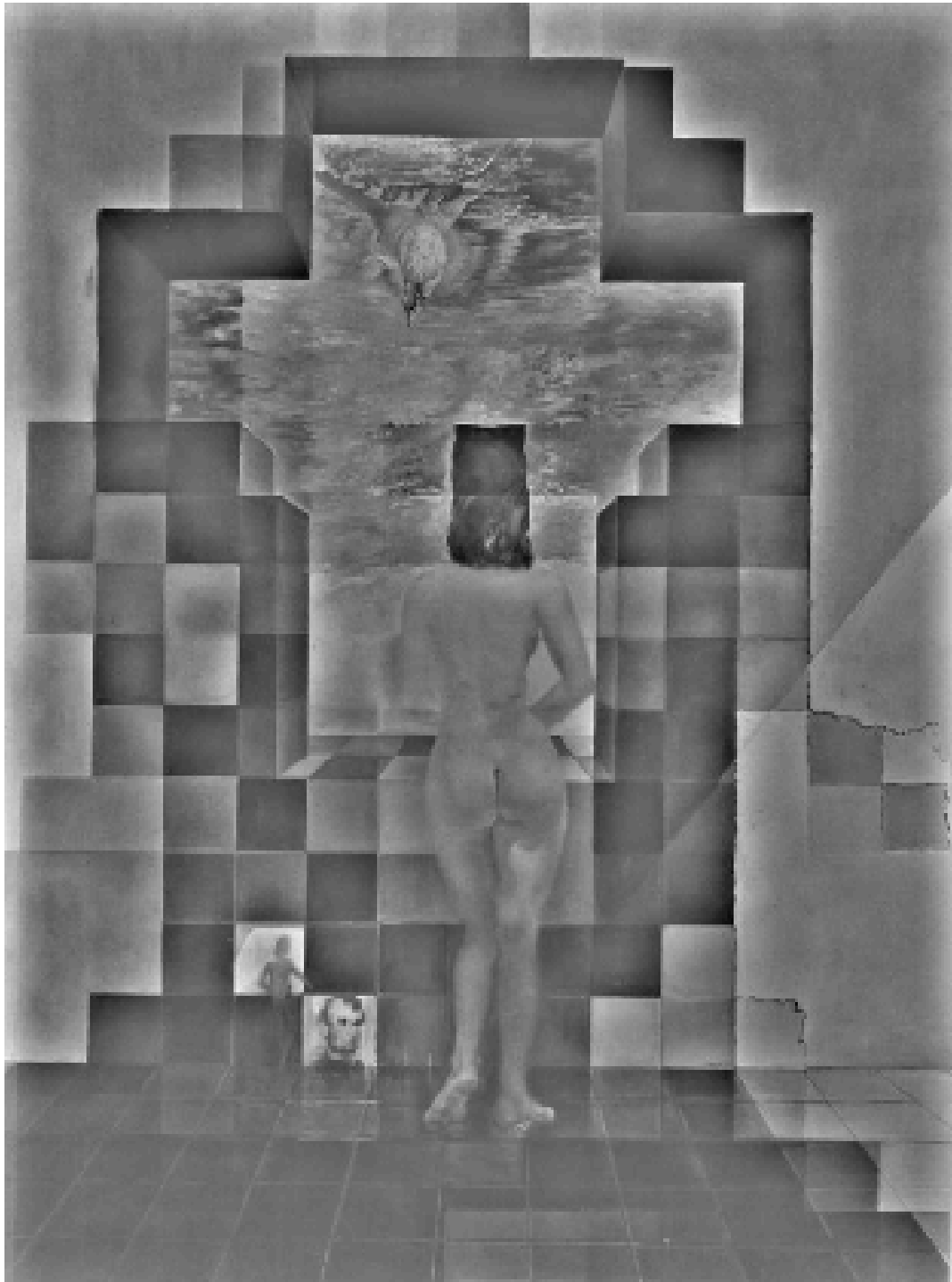
Somewhere in Cinque Terre, May 2005

CS194: Image Manipulation & Computational Photography

Alexei Efros, UC Berkeley, Fall 2014

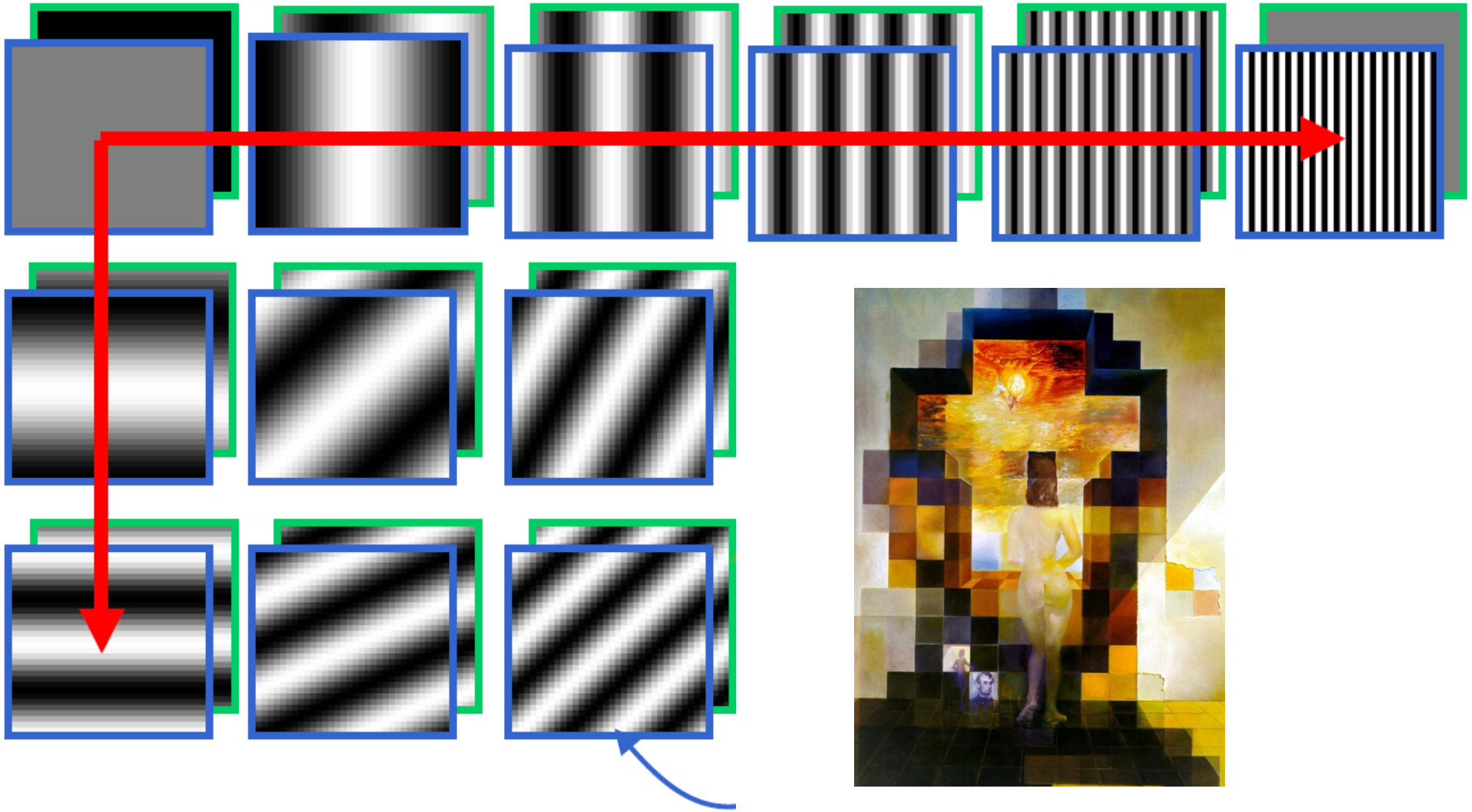
Many slides borrowed from Steve Seitz

Salvador Dali
“Gala Contemplating the Mediterranean Sea, which at 30 meters becomes the portrait of Abraham Lincoln”, 1976



A nice set of basis

Teases away fast vs. slow changes in the image.



This change of basis has a special name…

Jean Baptiste Joseph Fourier (1768-1830)

had crazy idea (1807

Any *univariate function ca*
be rewritten as a weighted
sum of sines and cosines
different frequencies.

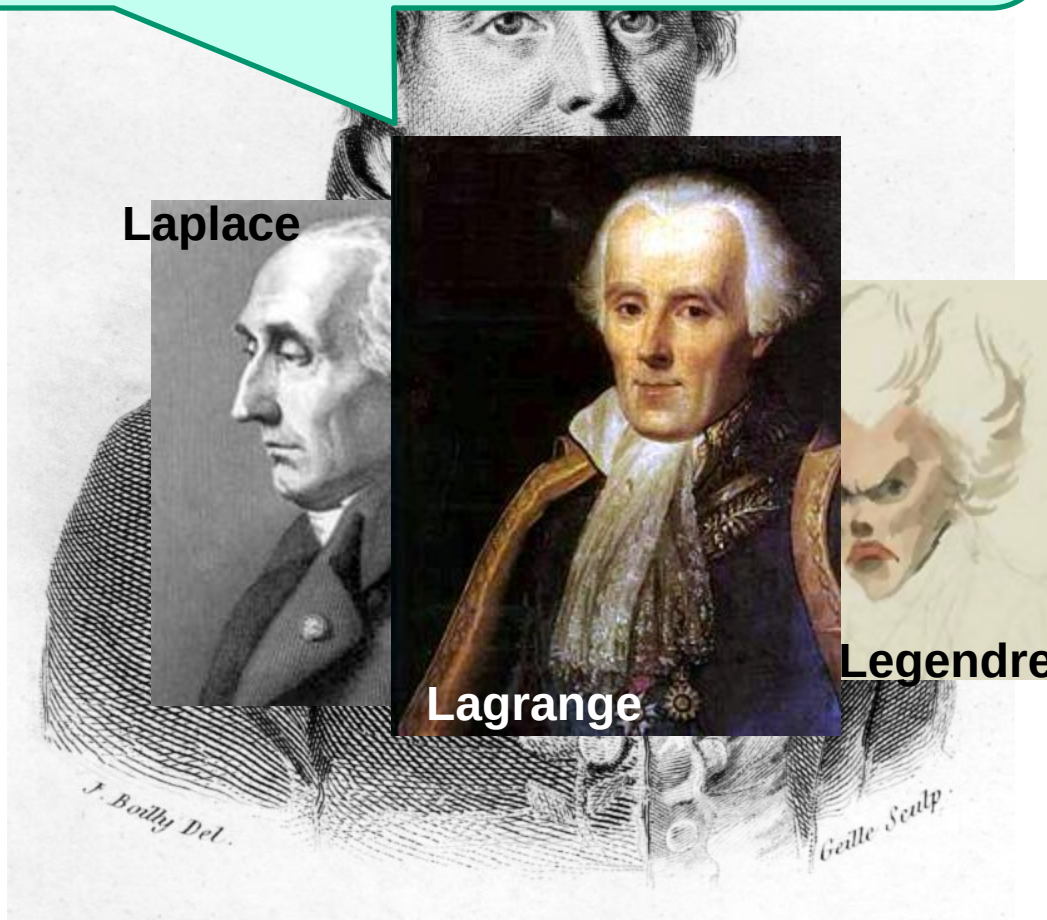
Don't believe it?

- 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

...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.



A sum of sines

Our building block:

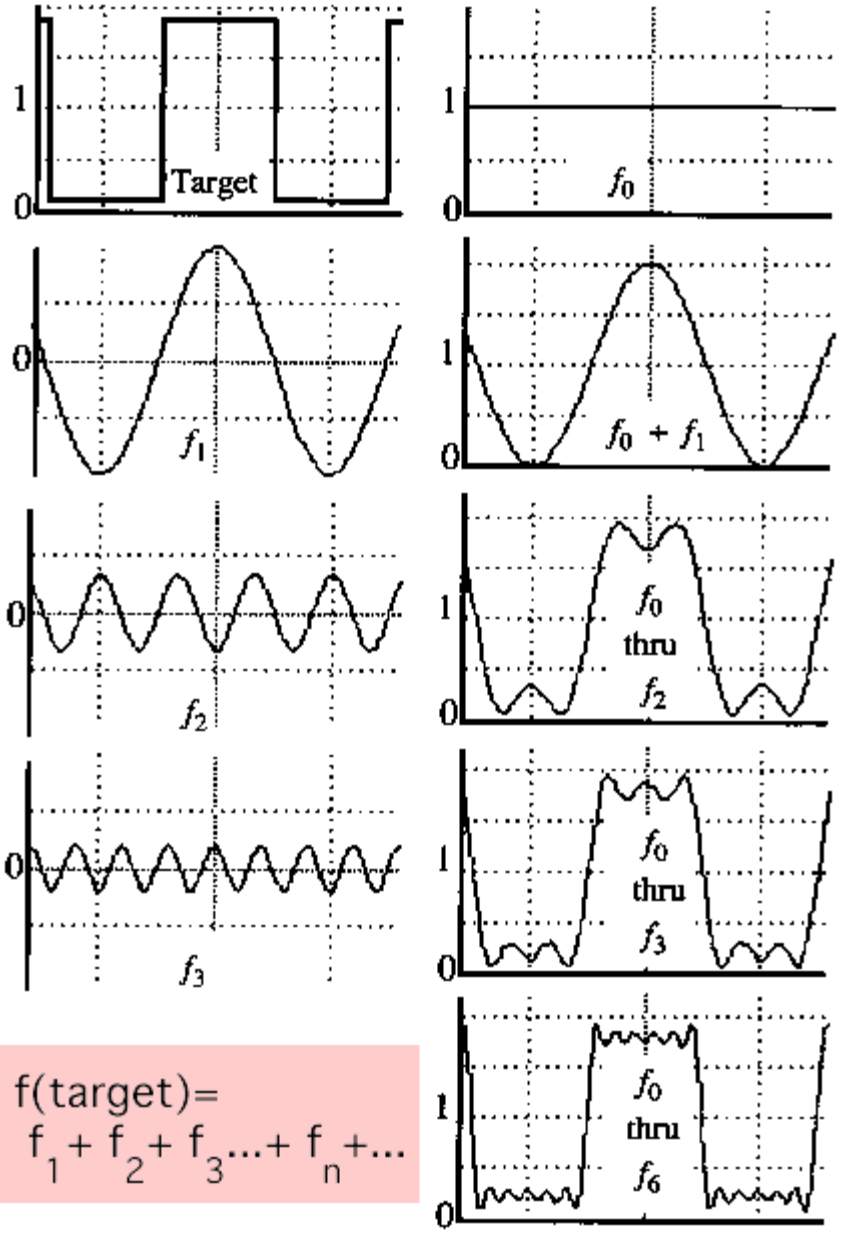
$$A\sin(\omega x + \phi)$$

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

How many degrees of freedom?

What does each control?

Which one encodes the coarse vs. fine structure of the signal?



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

Fourier Transform

We want to understand the frequency ω of our signal. So, let's reparametrize the signal by ω instead of *x*:

For every ω from 0 to inf, ***F(ω)*** holds the amplitude *A* and phase ϕ of the corresponding sine $A\sin(\omega x + \phi)$

- How can *F* hold both?

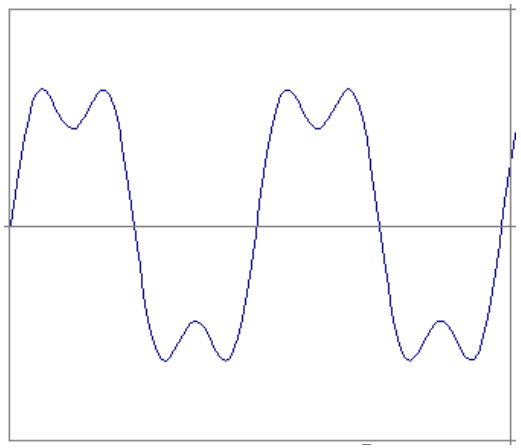
$$F(\omega) = R(\omega) + iI(\omega)$$

$$A = \pm\sqrt{R(\omega)^2 + I(\omega)^2} \qquad \phi = \tan^{-1}\frac{I(\omega)}{R(\omega)}$$

We can always go back:

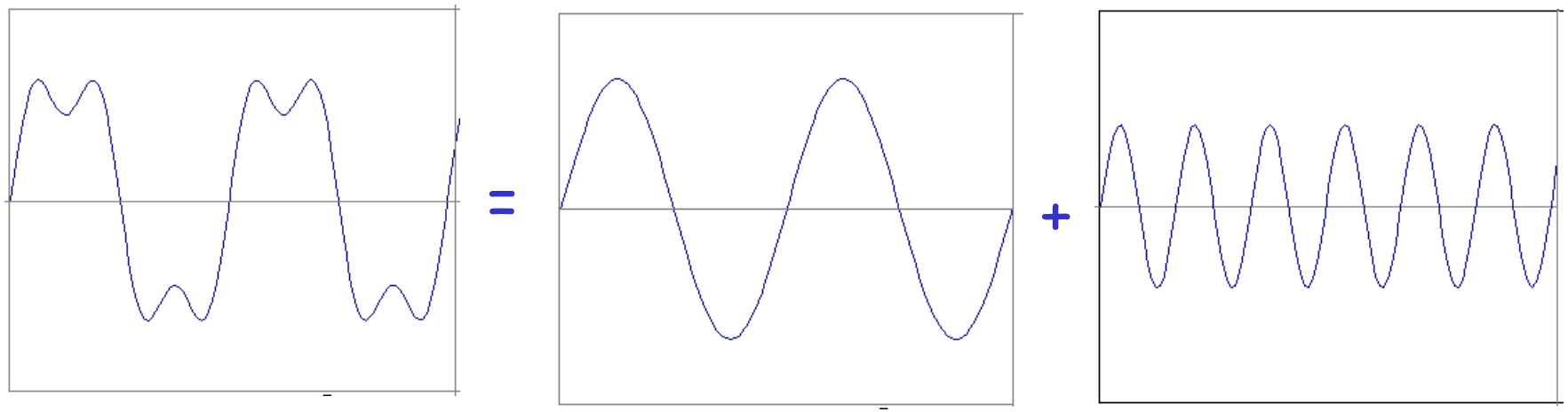
Time and Frequency

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



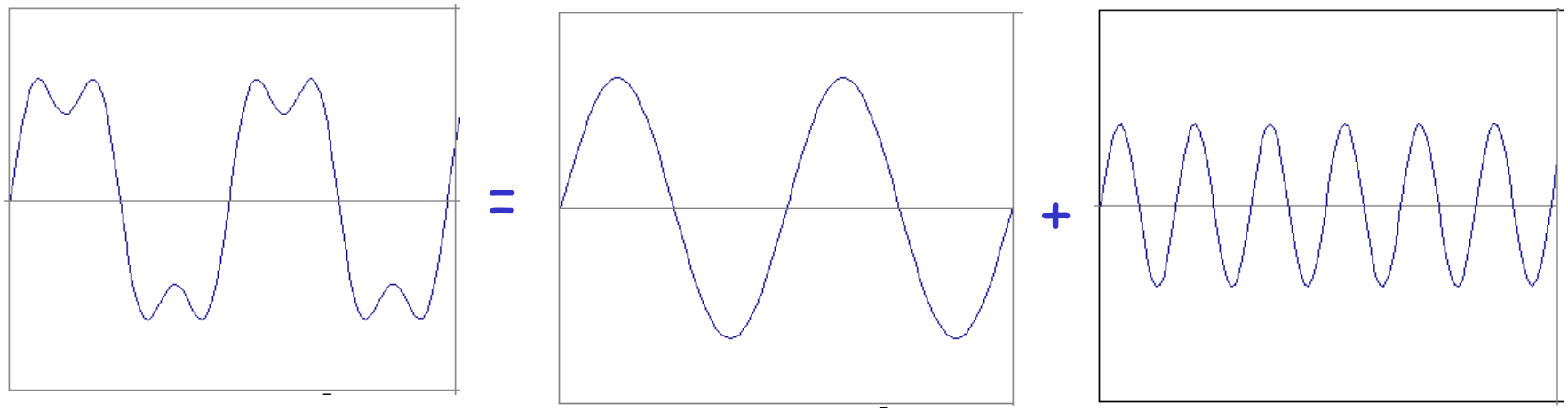
Time and Frequency

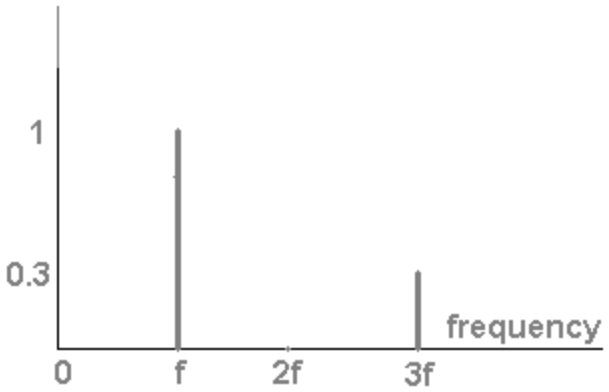
example : $g(t) = \sin(2\pi f t) + (1/3)\sin(2\pi(3f) t)$



Frequency Spectra

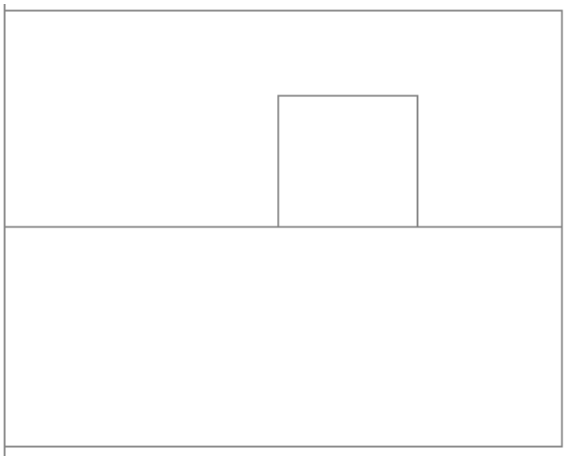
example : $g(t) = \sin(2\pi f\, t) + (1/3)\sin(2\pi(3f)\, t)$



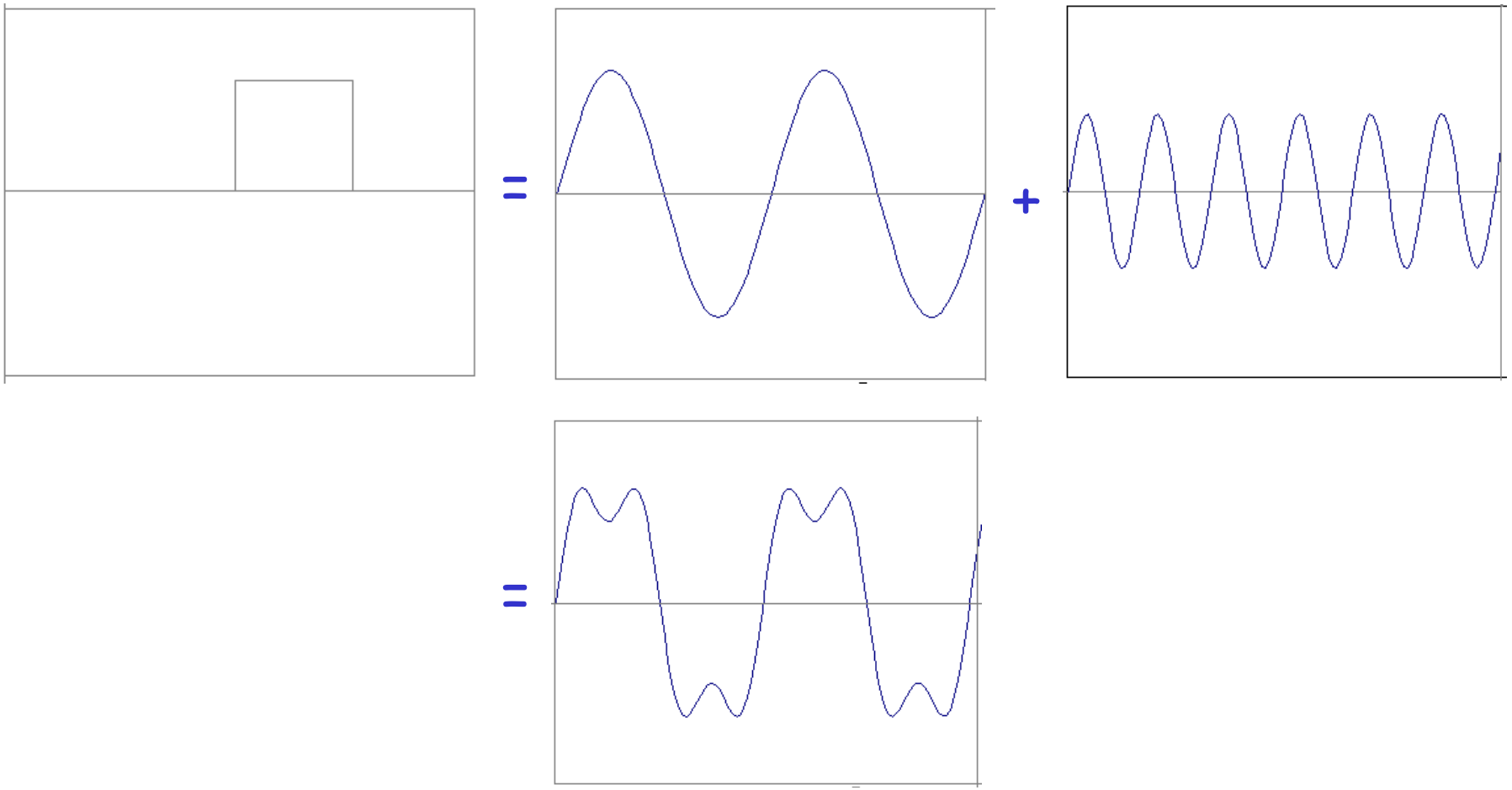


Frequency Spectra

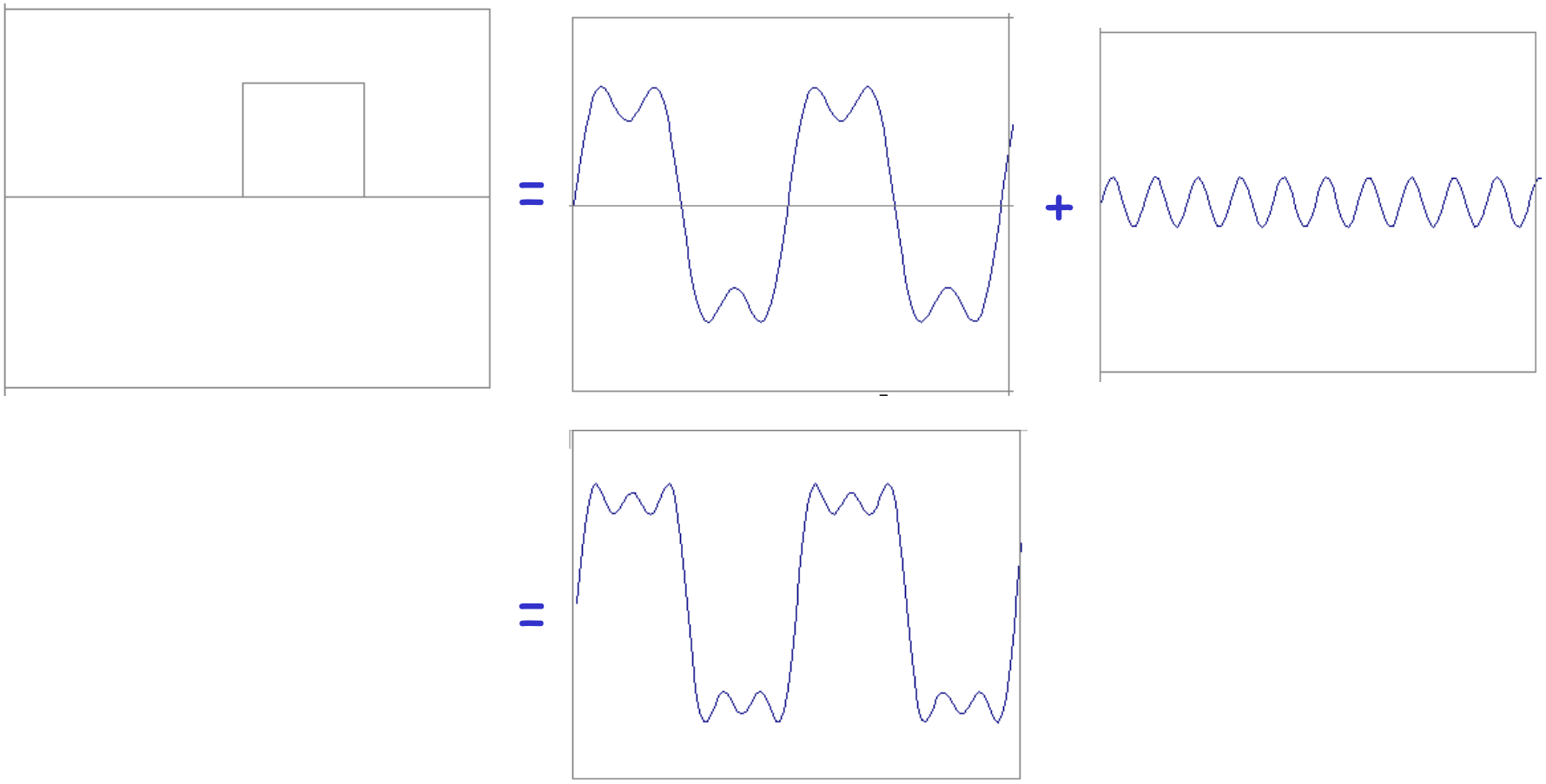
Usually, frequency is more interesting than the phase



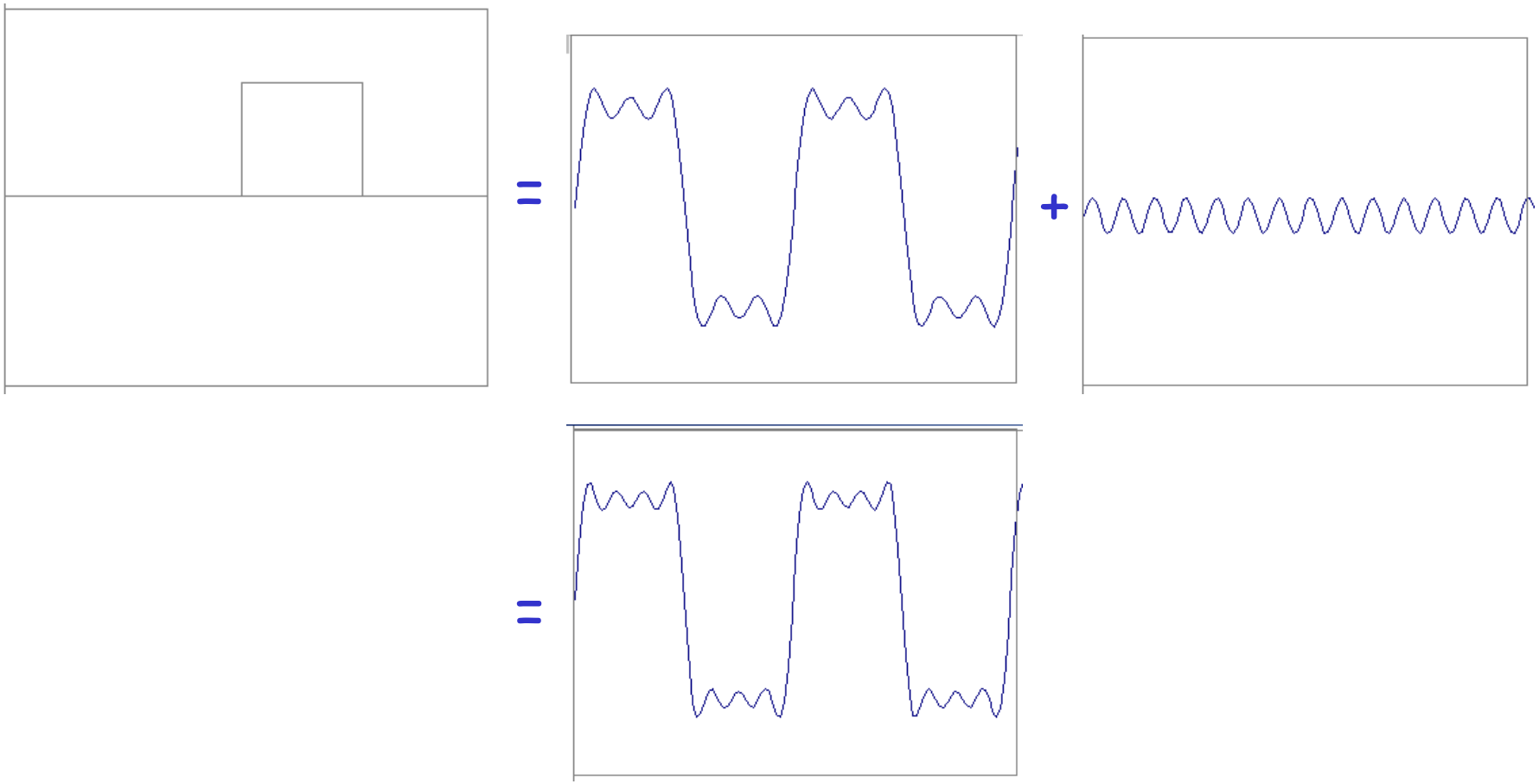
Frequency Spectra



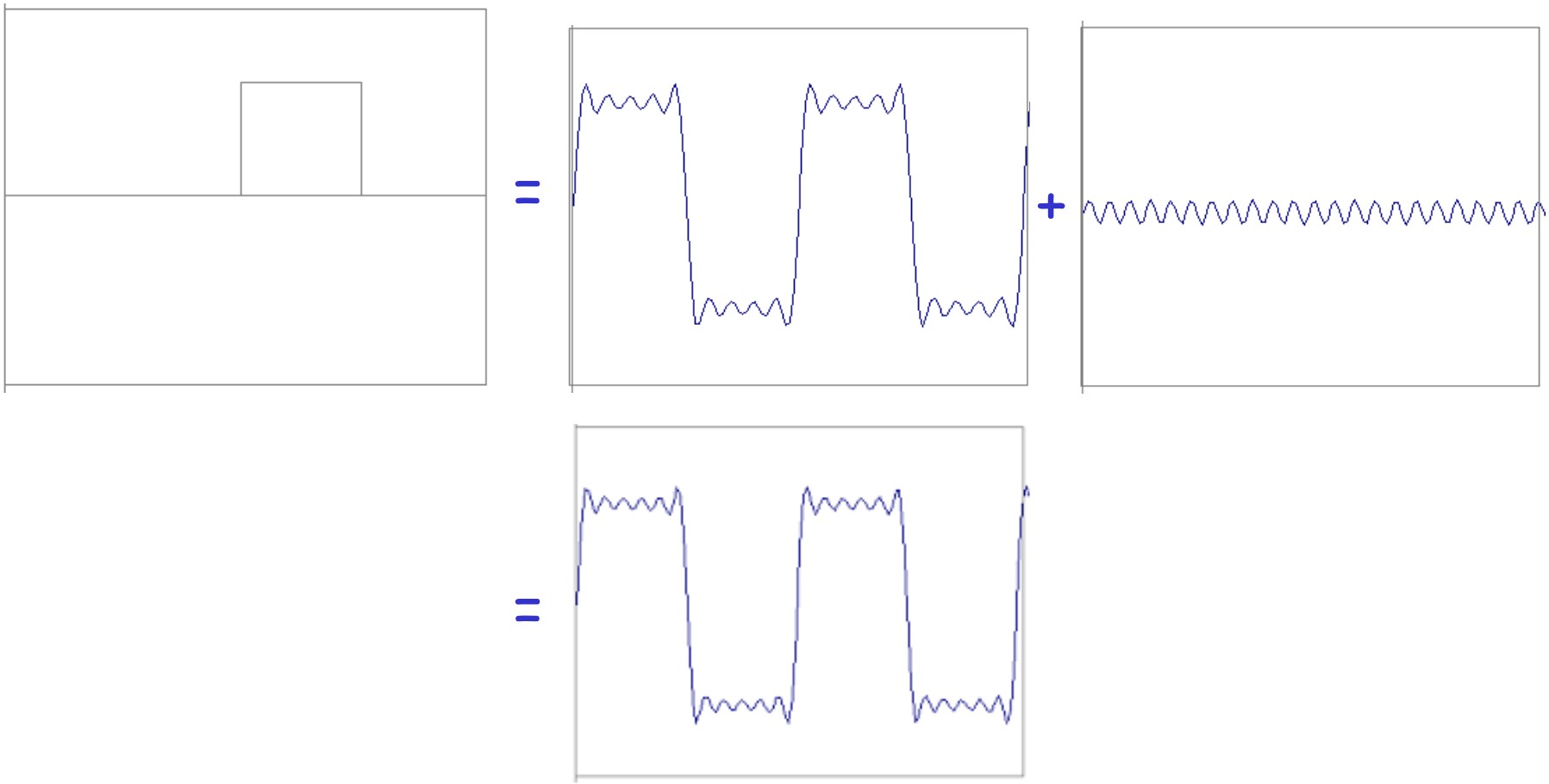
Frequency Spectra



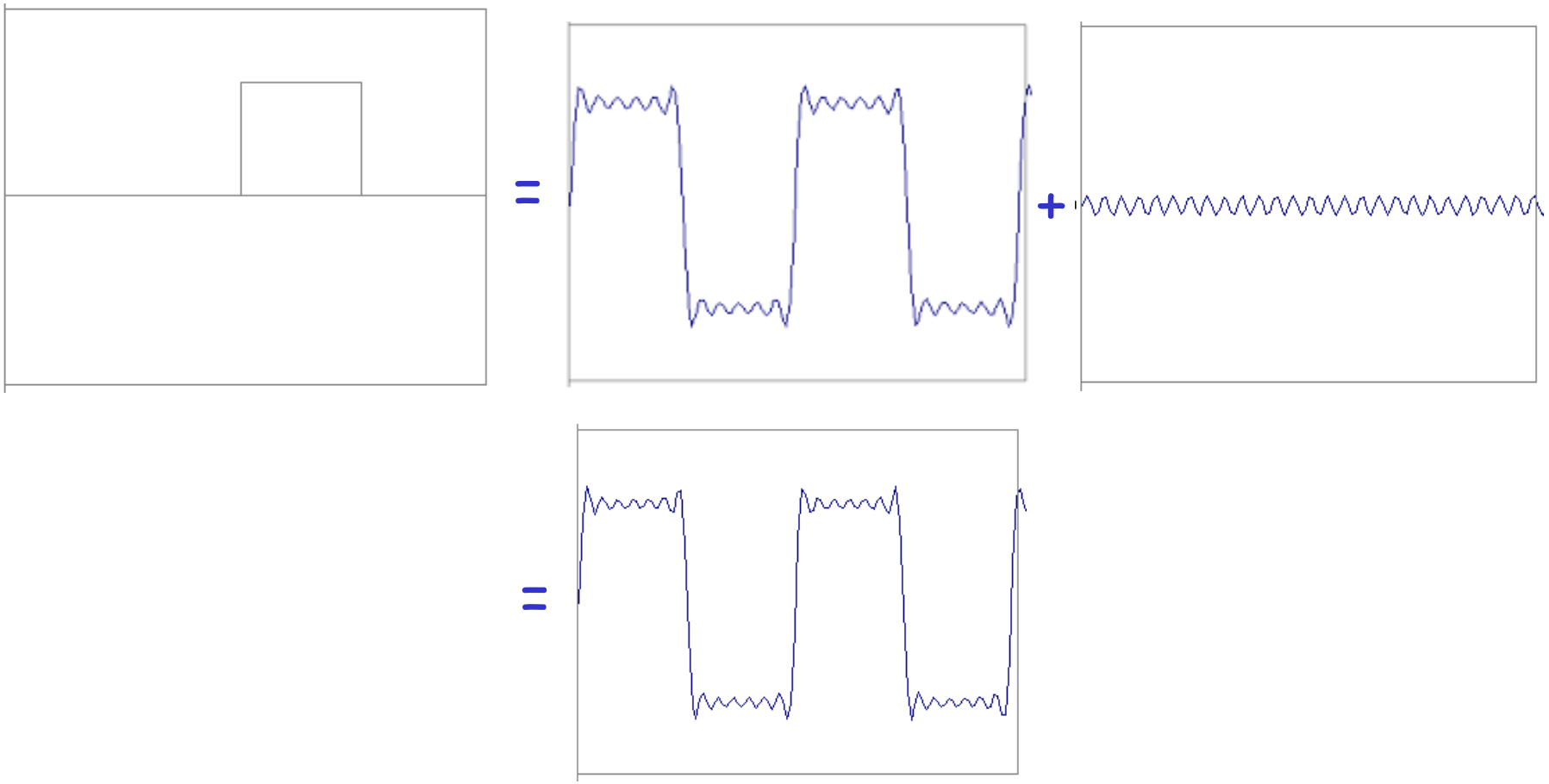
Frequency Spectra



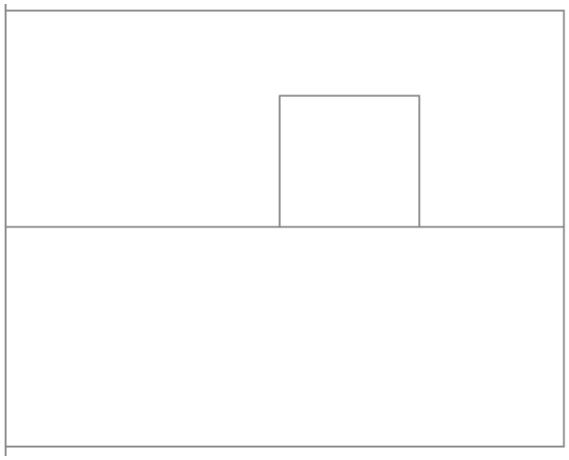
Frequency Spectra



Frequency Spectra

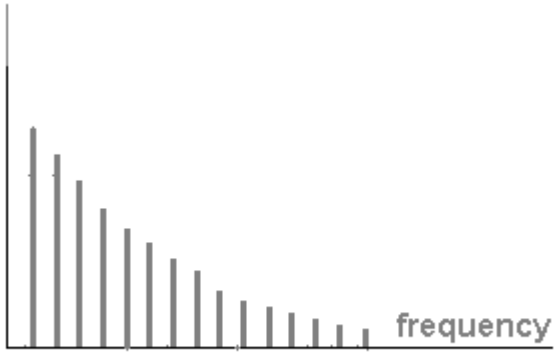


Frequency Spectra

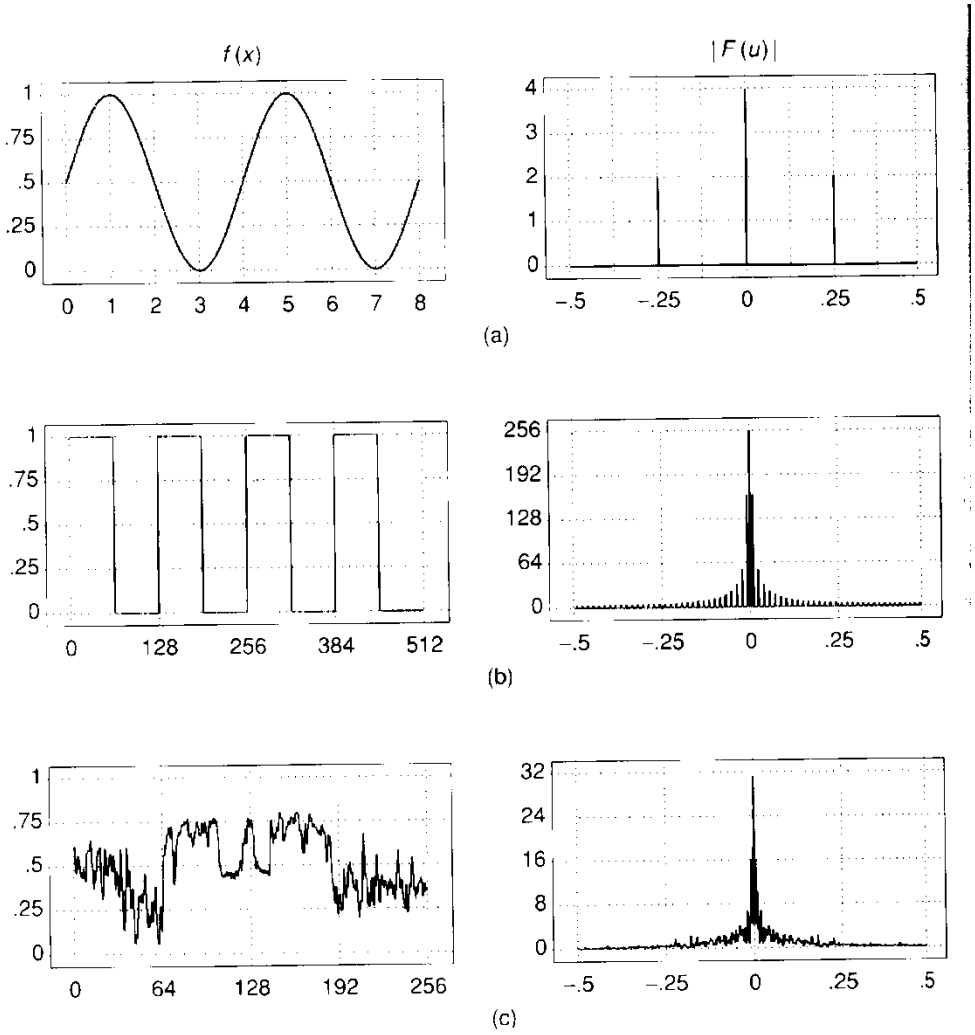


=

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

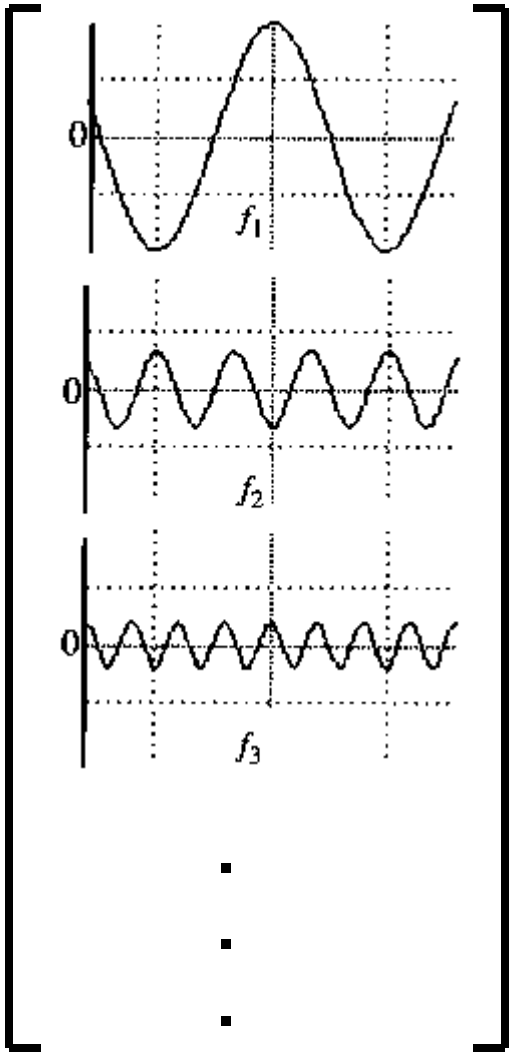


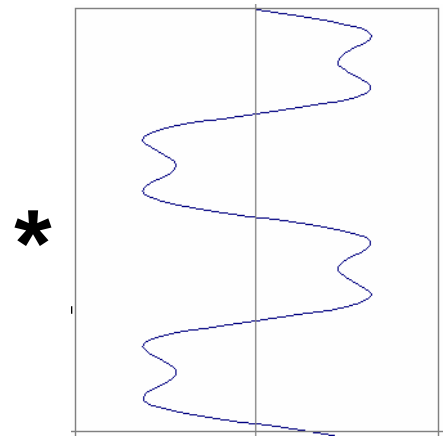
Frequency Spectra

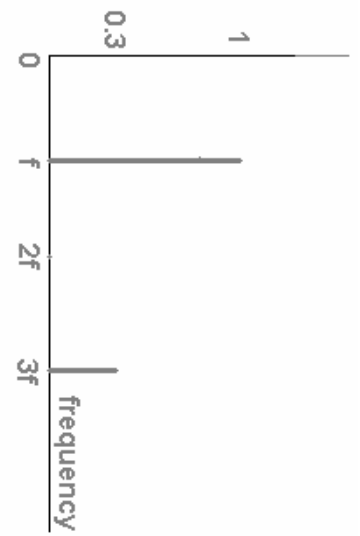


FT: Just a change of basis

$$M * f(x) = F(\omega)$$

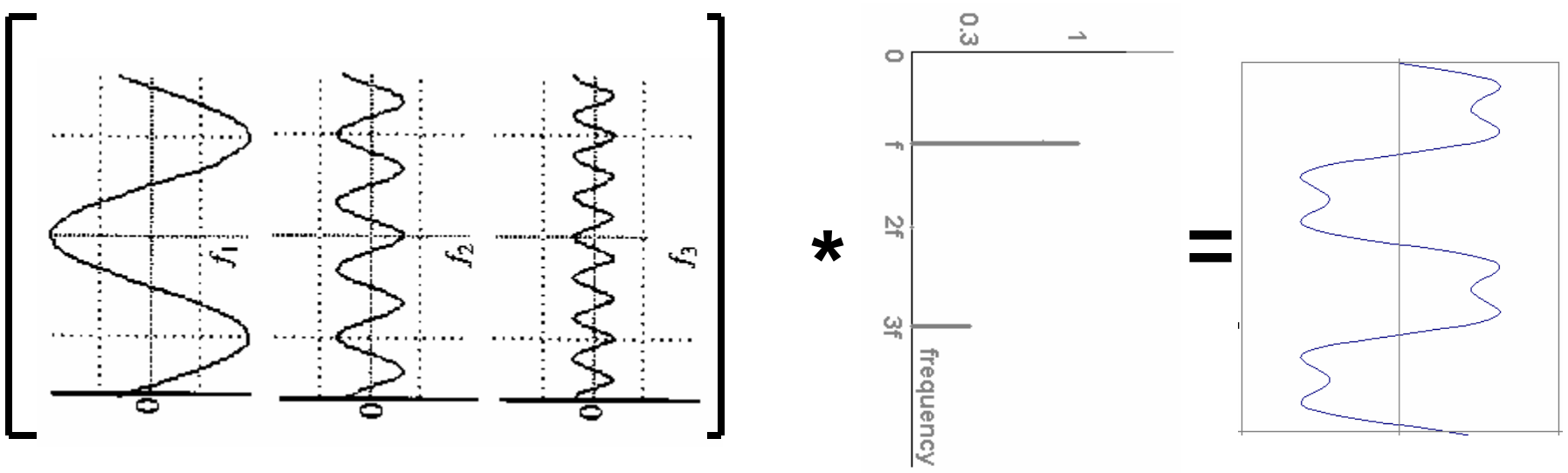






IFT: Just a change of basis

$$M^{-1} * F(\omega) = f(x)$$



.
.
.

Finally: Scary Math

$$\text{Fourier Transform}: \quad F(\omega) = \int_{-\infty}^{+\infty} f(x) e^{-i\omega x} dx$$

$$\text{Inverse Fourier Transform}: \quad f(x) = \frac{1}{2\pi} \int_{-\infty}^{+\infty} F(\omega) e^{i\omega x} d\omega$$

Finally: Scary Math

$$\text{Fourier Transform}: \quad F(\omega) = \int_{-\infty}^{+\infty} f(x) e^{-i\omega x} dx$$

$$\text{Inverse Fourier Transform}: \quad f(x) = \frac{1}{2\pi} \int_{-\infty}^{+\infty} F(\omega) e^{i\omega x} d\omega$$

…not really scary: $e^{1\omega x} = \cos(\omega x) + i \sin(\omega x)$

is hiding our old friend: $A\sin(\omega x + \phi)$

phase can be encoded by sin/cos pair →

$$P\cos(x) + Q\sin(x) = A\sin(x + \phi)$$

$$A = \pm\sqrt{P^2 + Q^2} \qquad \phi = \tan^{-1}\left(\frac{P}{Q}\right)$$

So it's just our signal *f(x)* times sine at frequency ω

Extension to 2D

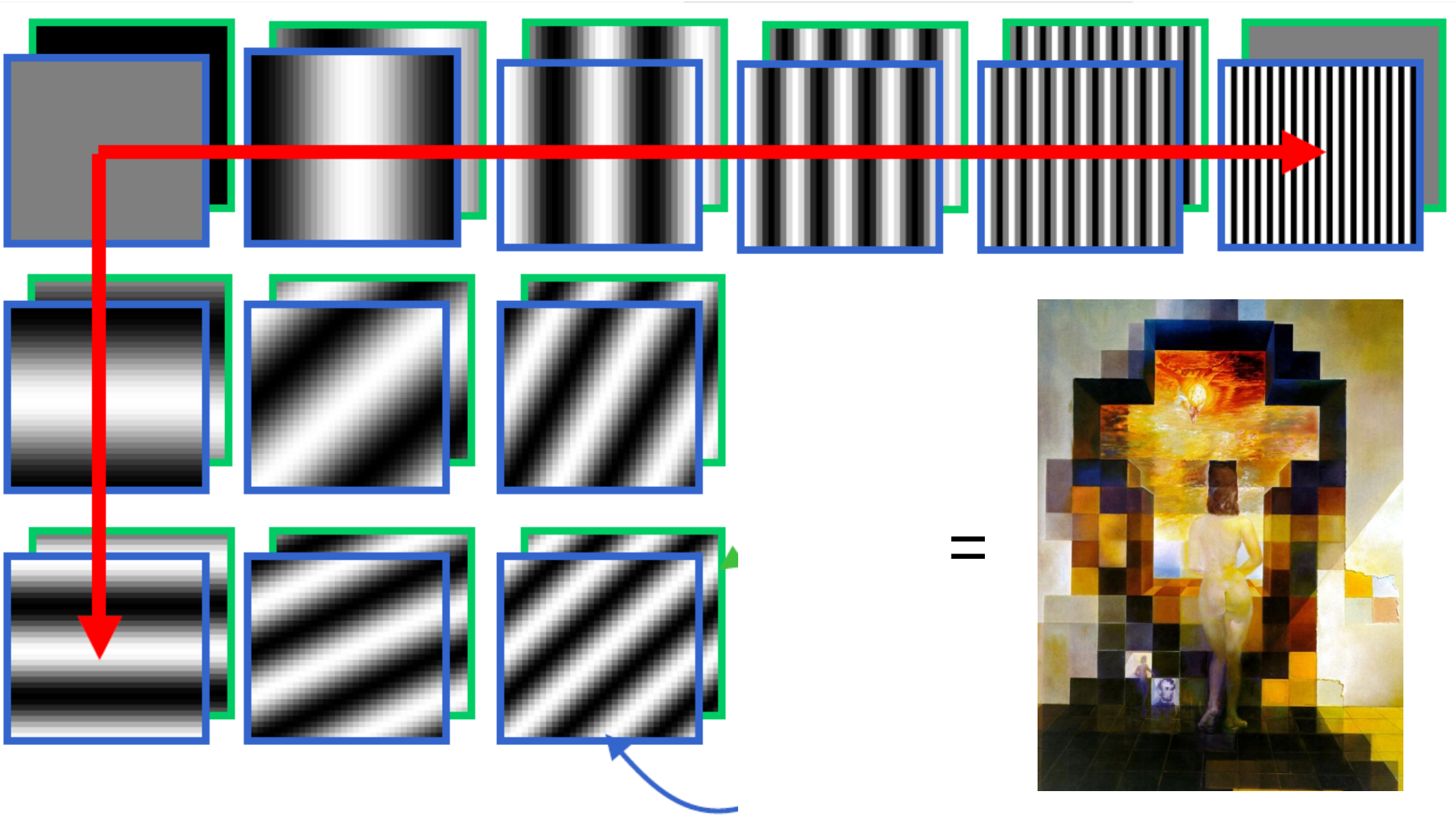
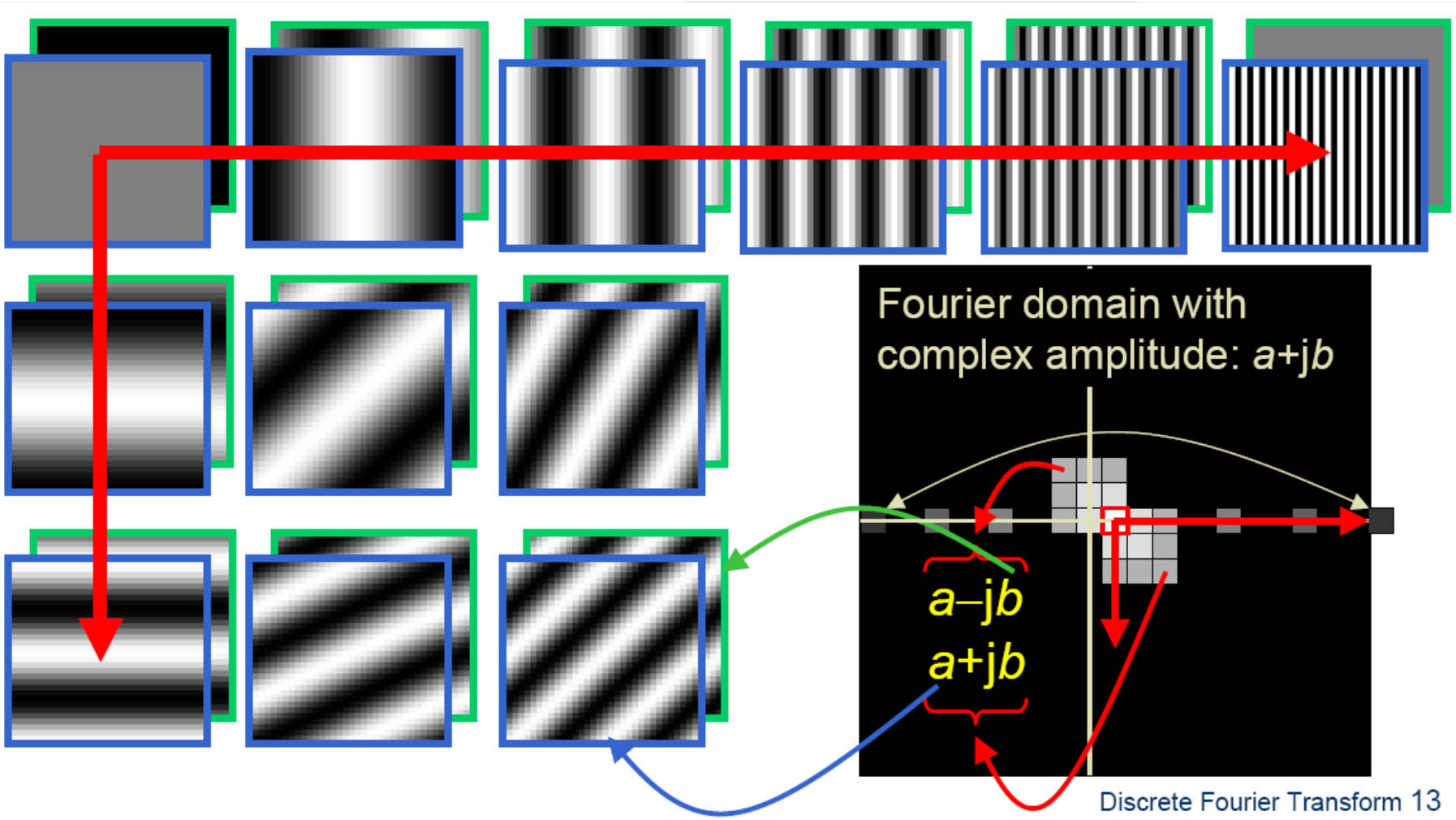


Image as a sum of basis images

Extension to 2D

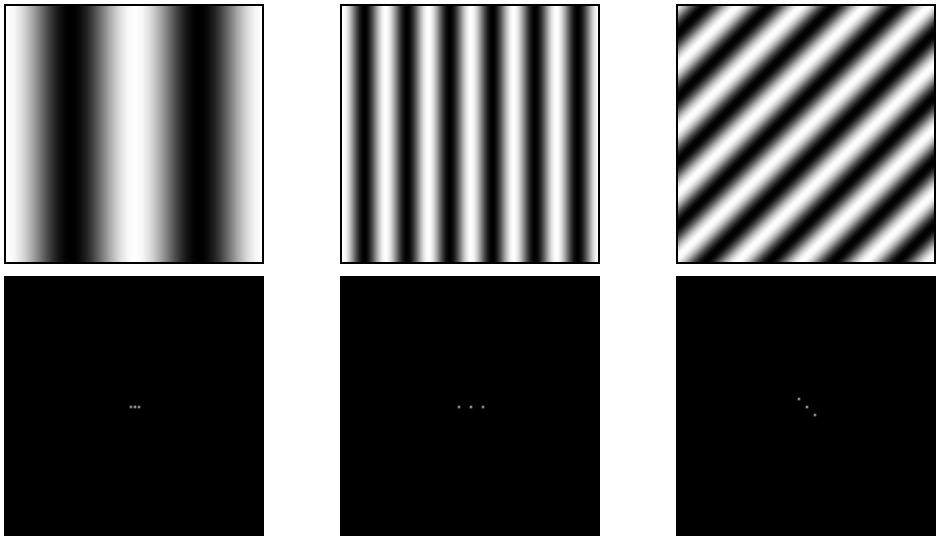


in Matlab, check out: imagesc(log(abs(fftshift(fft2(im)))));

Fourier analysis in images

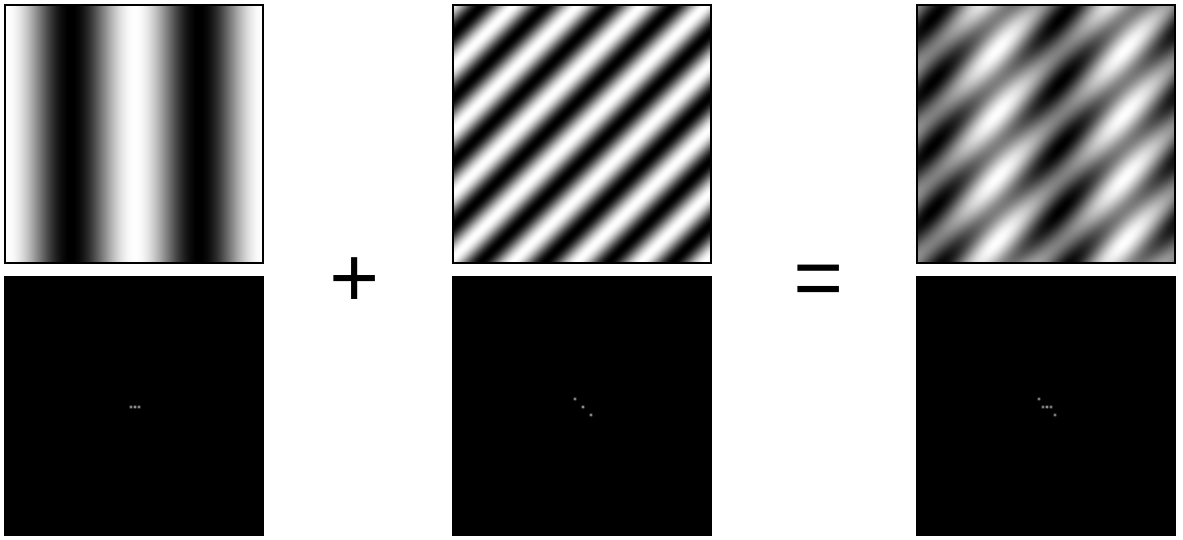
Intensity Image

Fourier Image



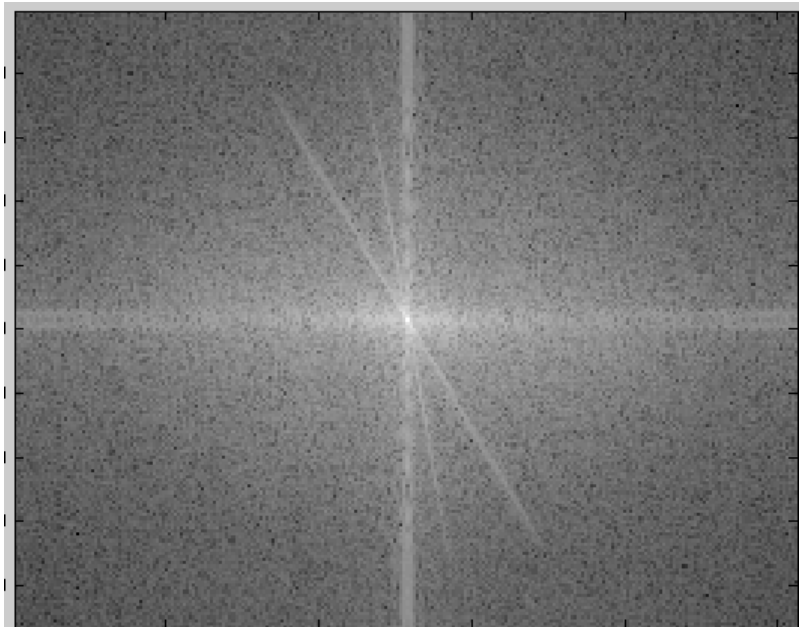
http://sharp.bu.edu/~slehar/fourier/fourier.html#filtering

Signals can be composed



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

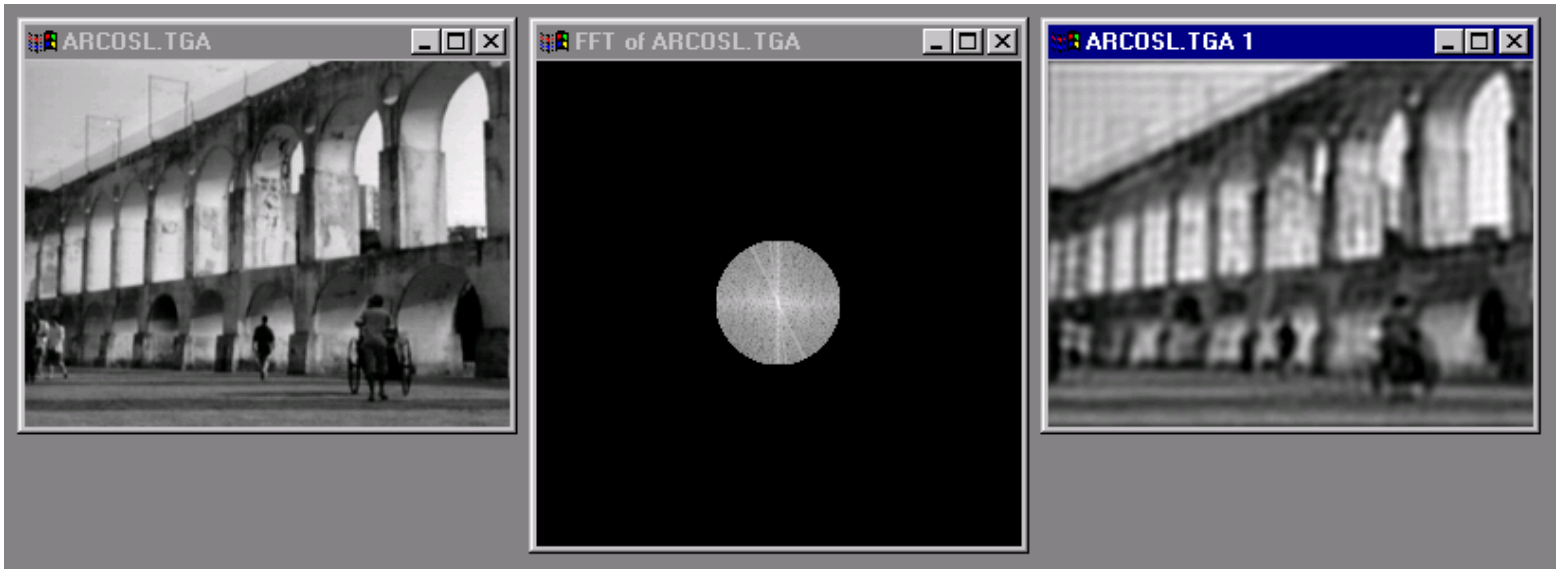
Man-made Scene

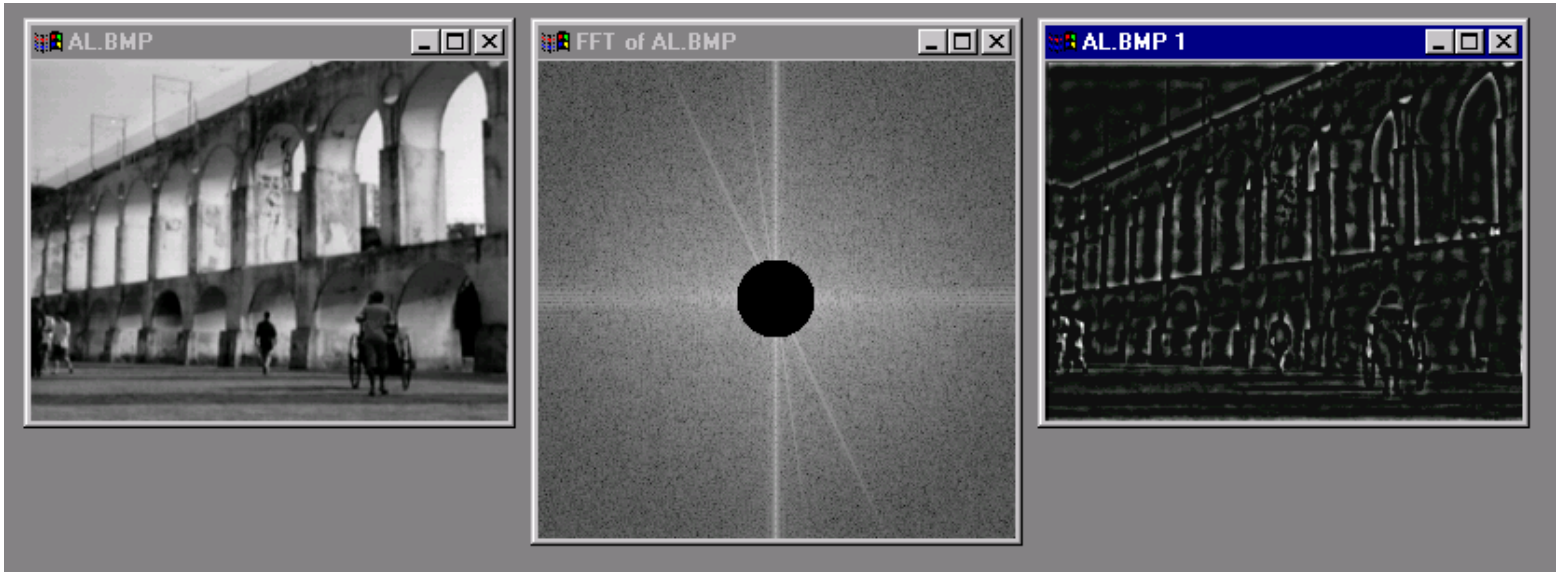


Can change spectrum, then reconstruct

Local change in one domain, courses global change in the other

Low and High Pass filtering





The Convolution Theorem

The greatest thing since sliced (banana) bread!

- 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

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

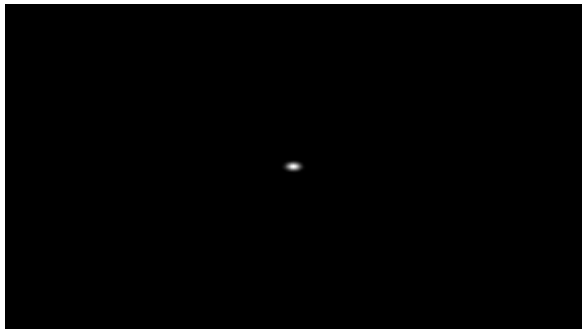
- **Convolution** in spatial domain is equivalent to **multiplication** in frequency domain!

2D convolution theorem example

$f(x,y)$

$*$

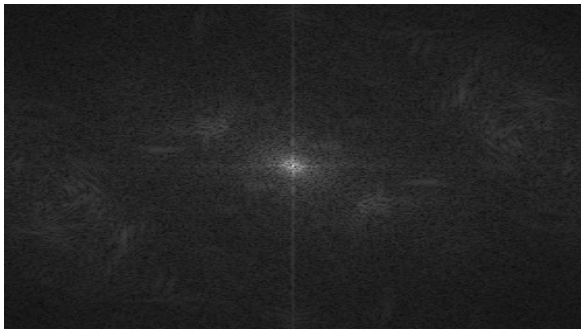
$h(x,y)$



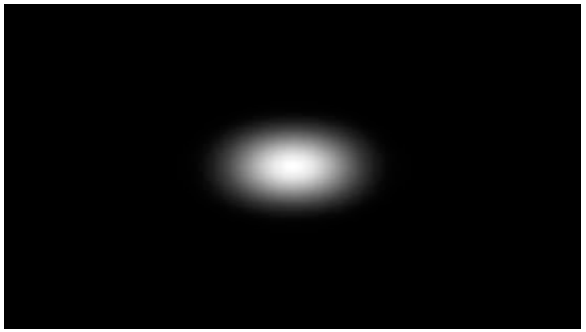
$\Downarrow$

$g(x,y)$

$\asymp$

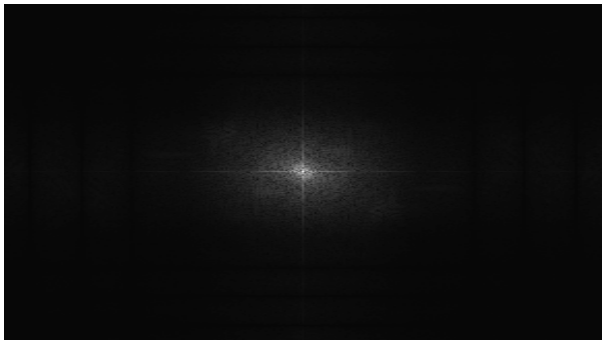


$|F(s_x,s_y)|$



$|H(s_x,s_y)|$

$\Downarrow$

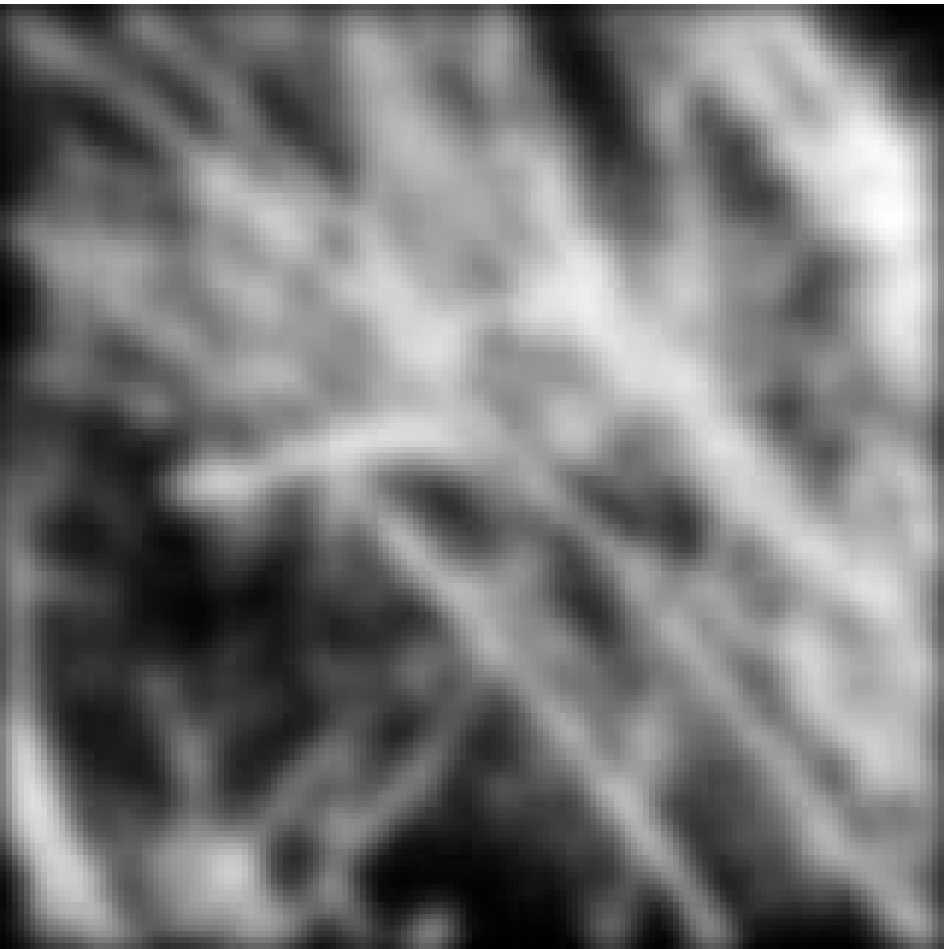


$|G(s_x,s_y)|$

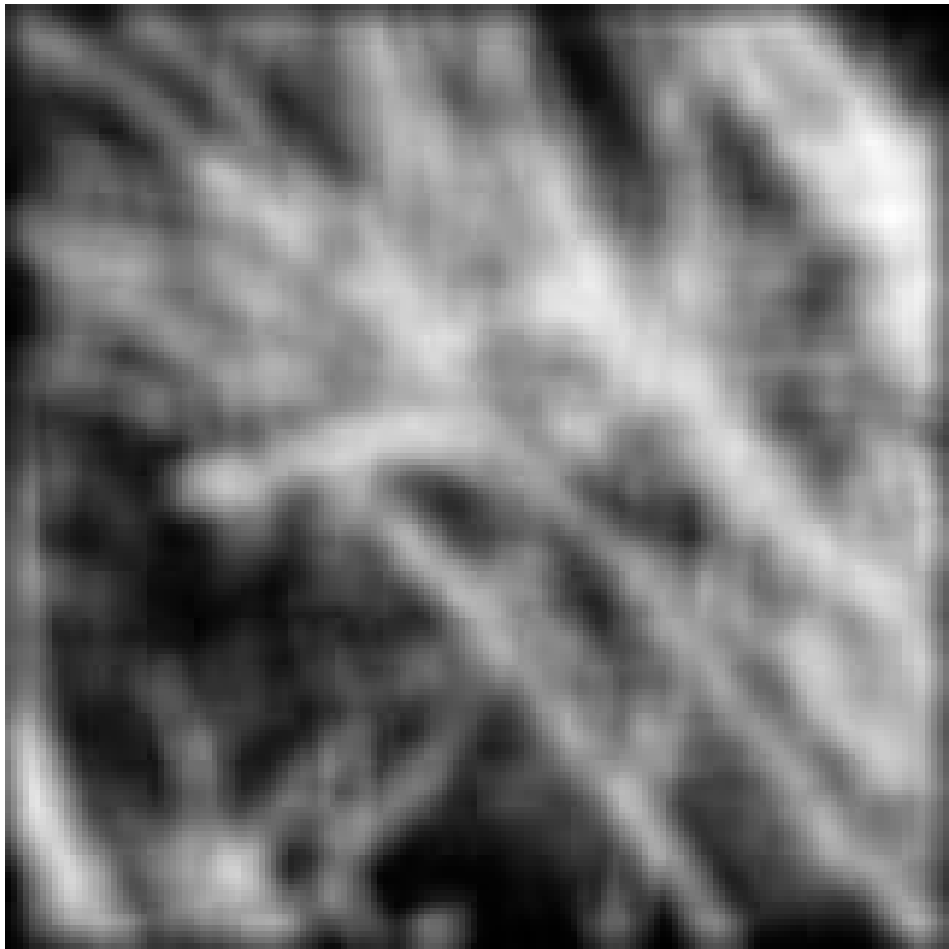
Filtering

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

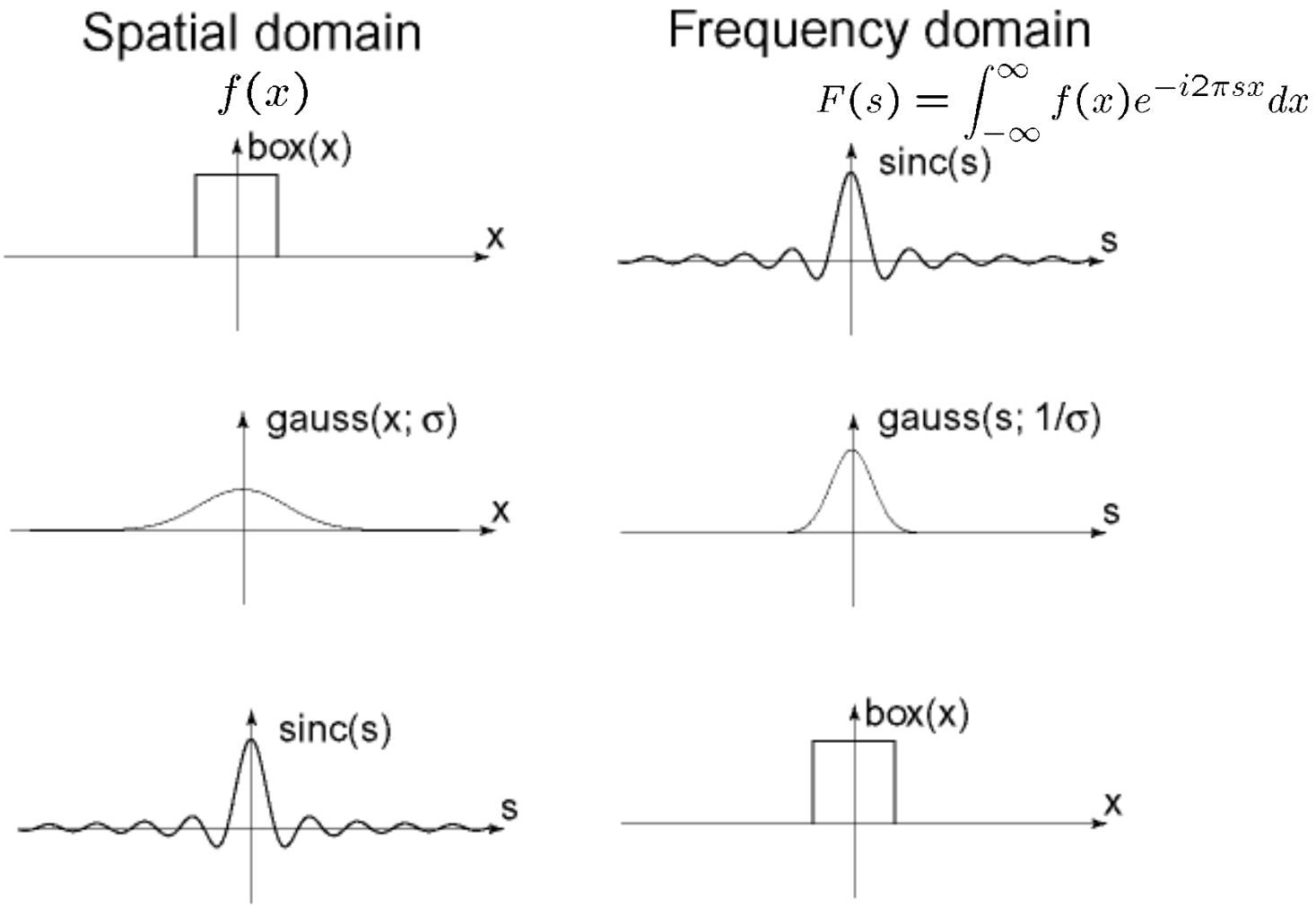
Gaussian 



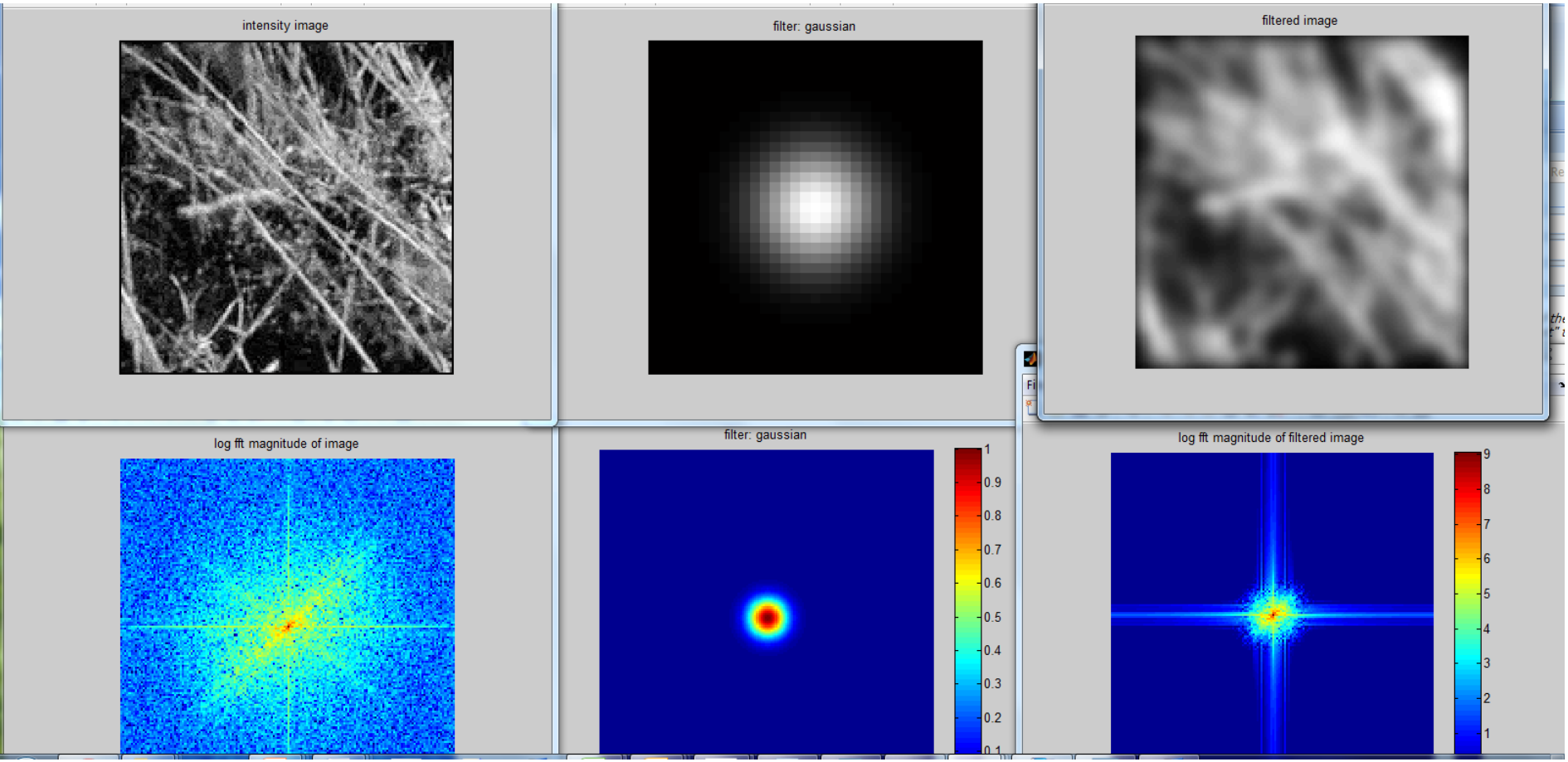
Box filter 



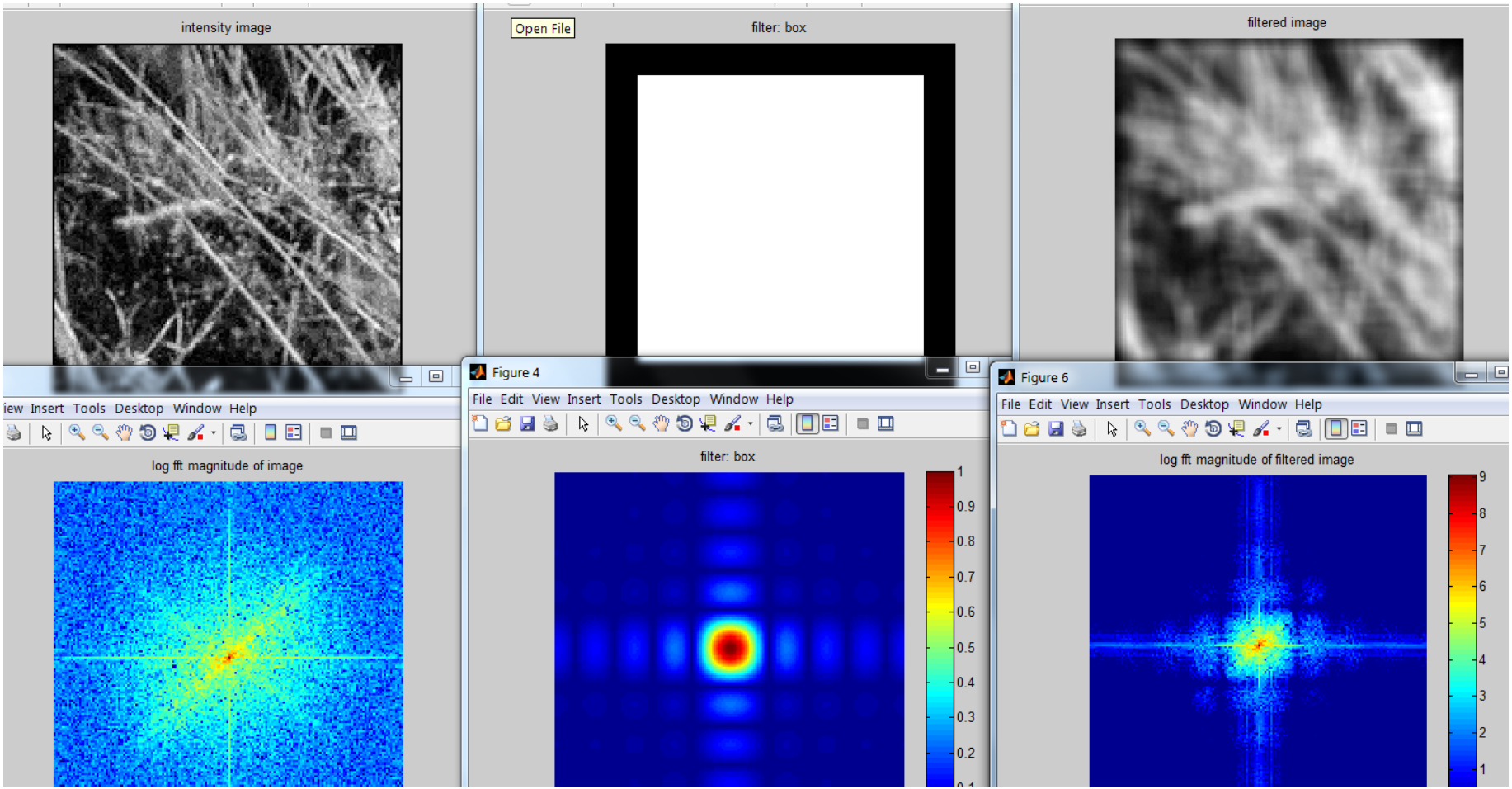
Fourier Transform pairs



Gaussian

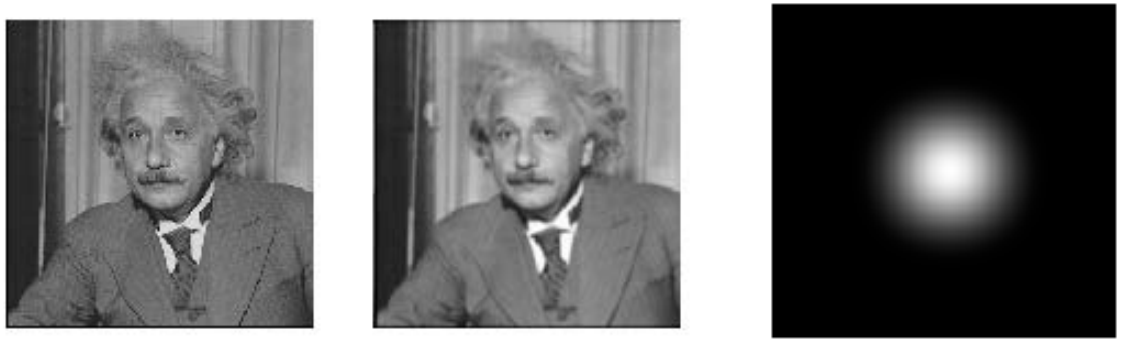

intensity image
filter: gaussian
filtered image
log fft magnitude of image
filter: gaussian
log fft magnitude of filtered image
1
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
9
8
7
6
5
4
3
2
1

Box Filter


intensity image
Open File
filter: box
filtered image
iew Insert Tools Desktop Window Help
log fft magnitude of image
Figure 4
File Edit View Insert Tools Desktop Window Help
filter: box
1
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
Figure 6
File Edit View Insert Tools Desktop Window Help
log fft magnitude of filtered image
9
8
7
6
5
4
3
2
1

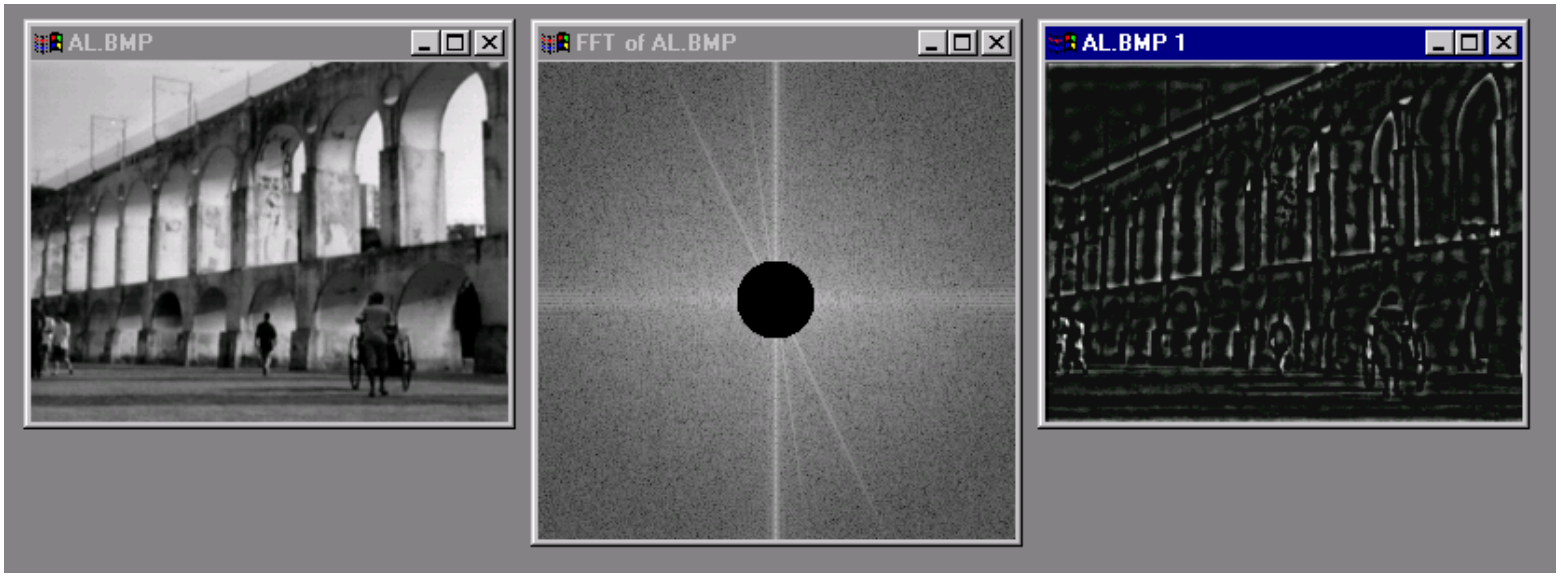
Low-pass, Band-pass, High-pass filters

low-pass:



High-pass / band-pass:

Edges in images



What does blurring take away?

original

What does blurring take away?

smoothed (5x5 Gaussian)

High-Pass filter

smoothed – original

Image “Sharpening”

What does blurring take away?

–

=

Let’s add it back:

+ α

=

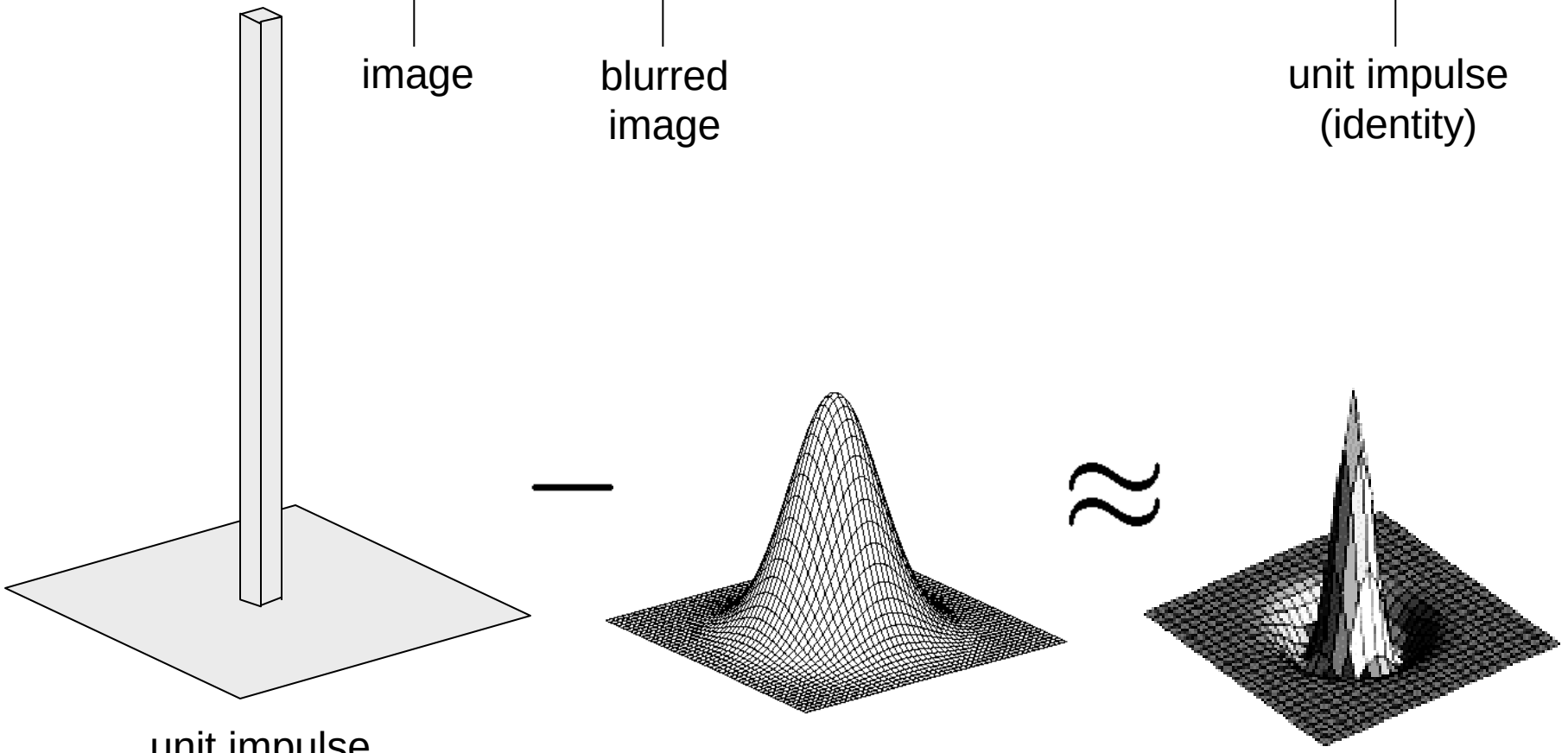
Unsharp mask filter

$$f + \alpha(f - f * g) = (1+\alpha) f - \alpha\, f * g = f * ((1+\alpha)e - g)$$

image

blurred image

unit impulse (identity)



unit impulse

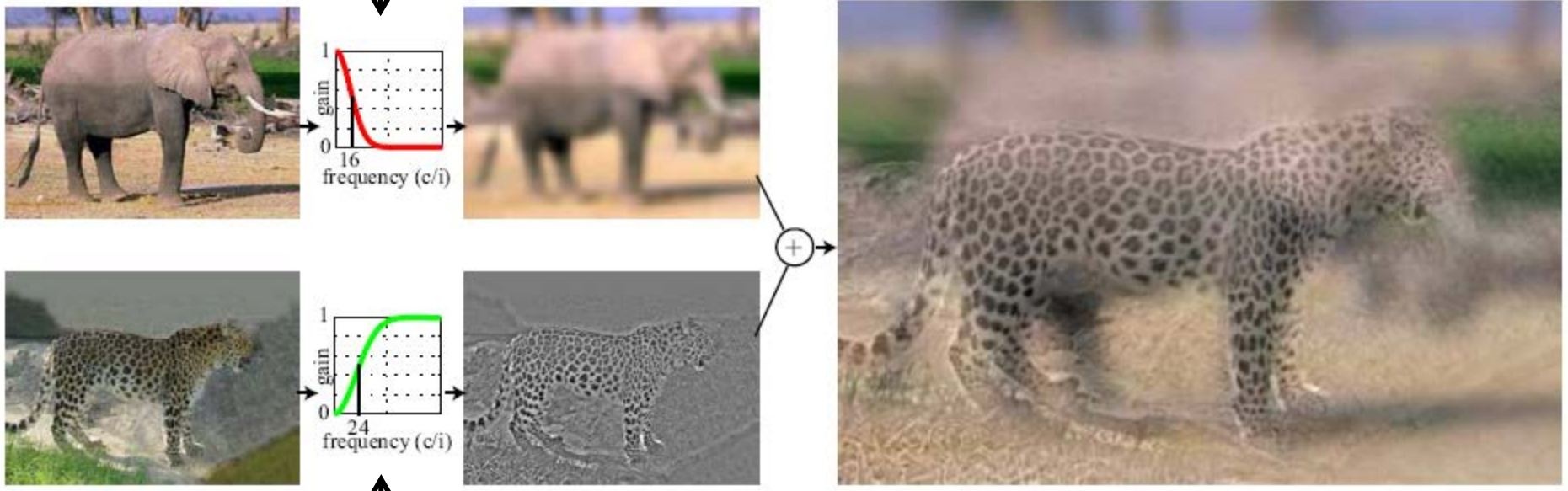
Gaussian

Laplacian of Gaussian

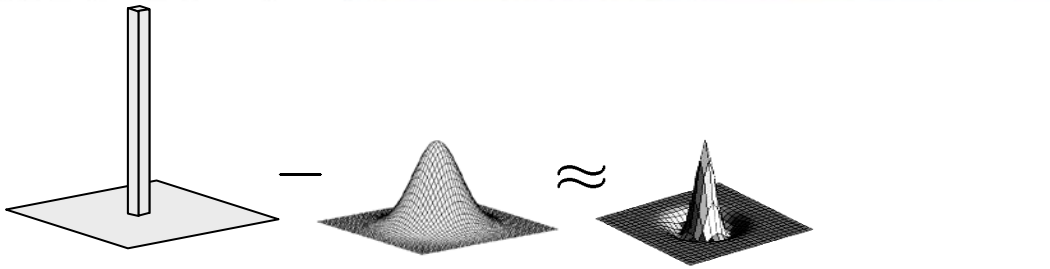
Hybrid Images

A. Oliva, A. Torralba, P.G. Schyns, "Hybrid Images," SIGGRAPH 2006

Gaussian Filter!



Laplacian Filter!



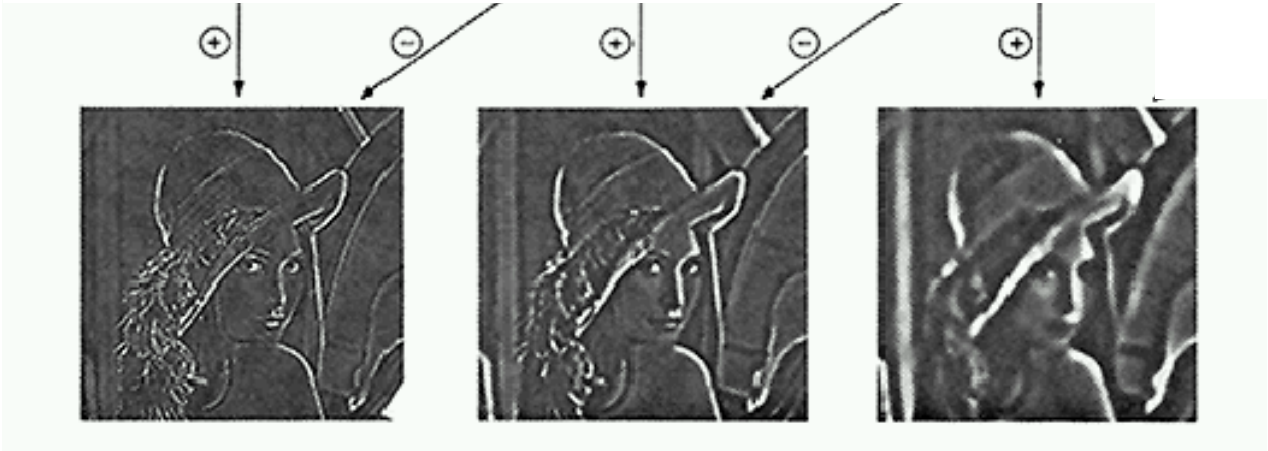
Salvador Dali
“Gala Contemplating the Mediterranean Sea, which at 30 meters becomes the portrait of Abraham Lincoln”, 1976

Band-pass filtering

Gaussian Pyramid (low-pass images)

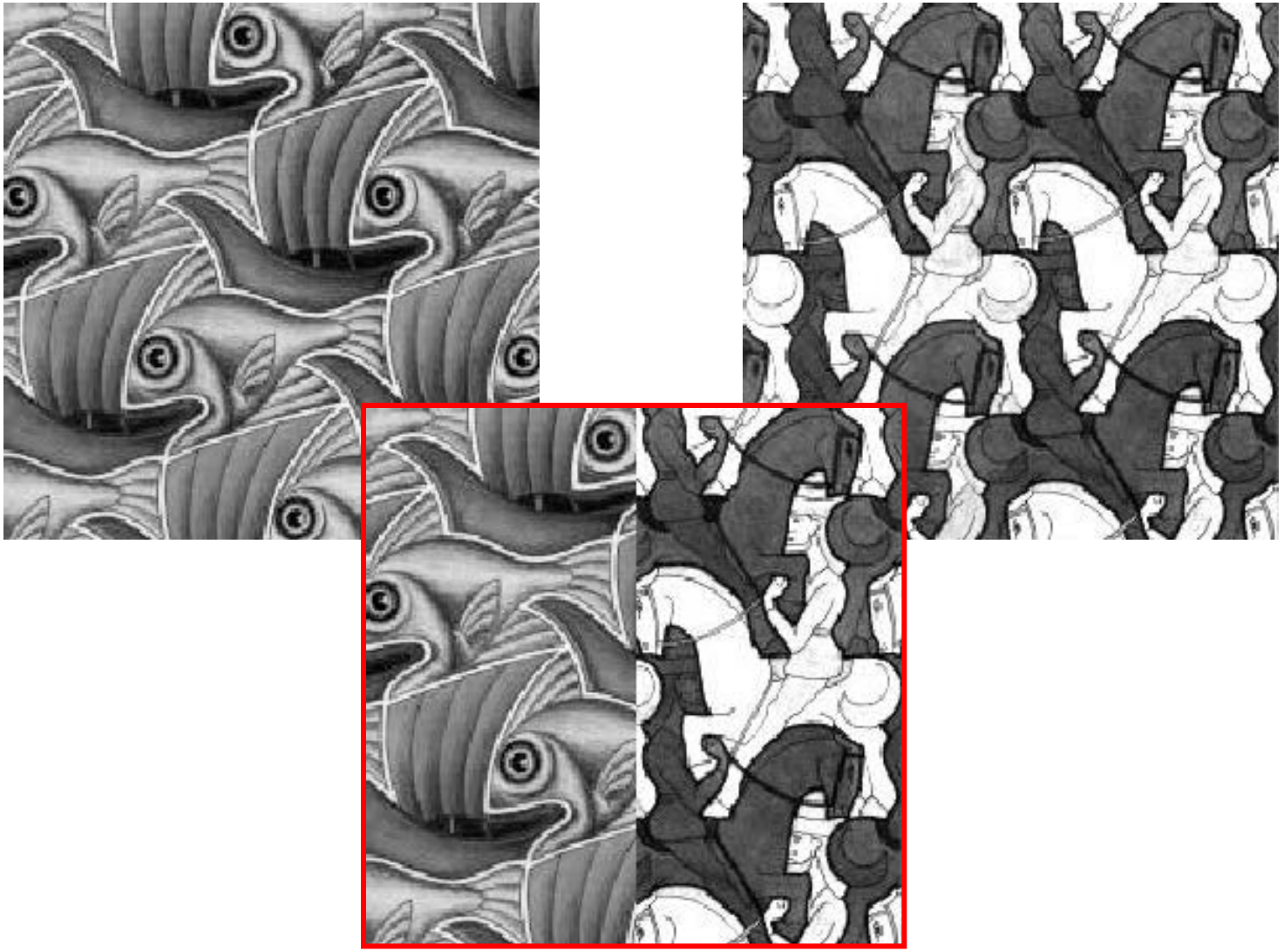
Laplacian Pyramid

Original image

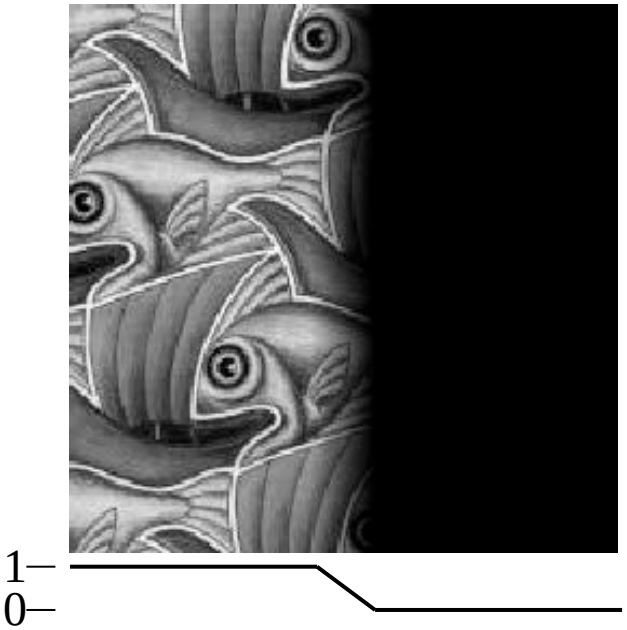


How can we reconstruct (collapse) this pyramid into the original image?

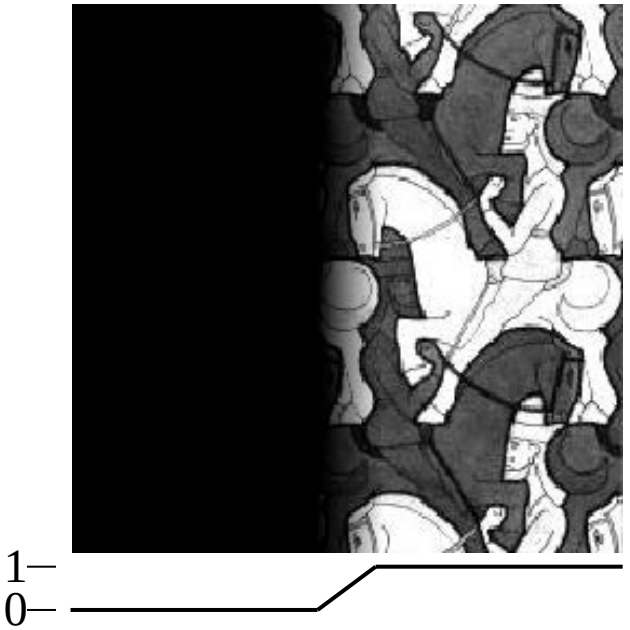
Blending



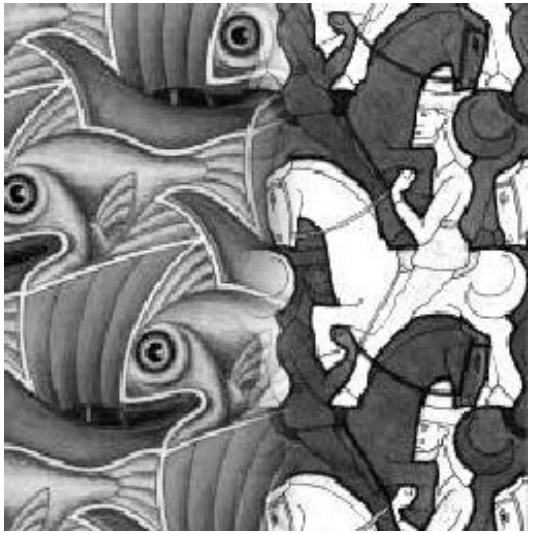
Alpha Blending / Feathering



+

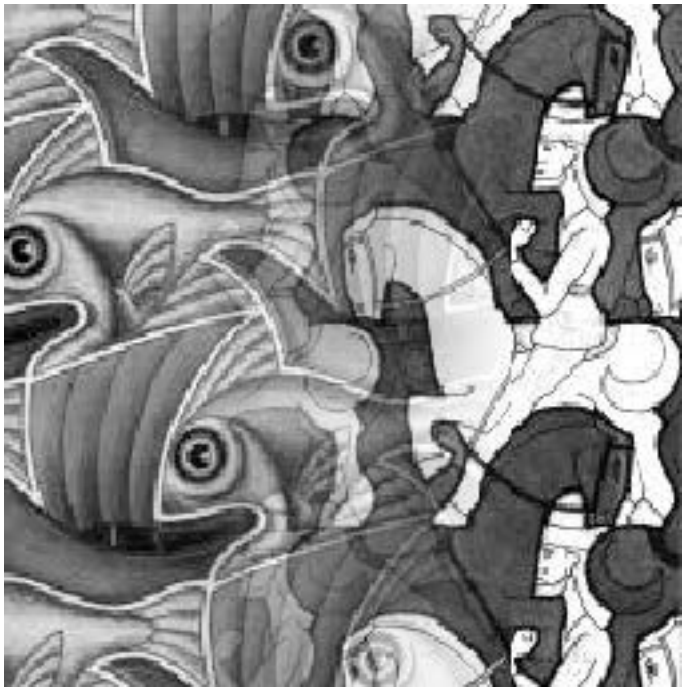


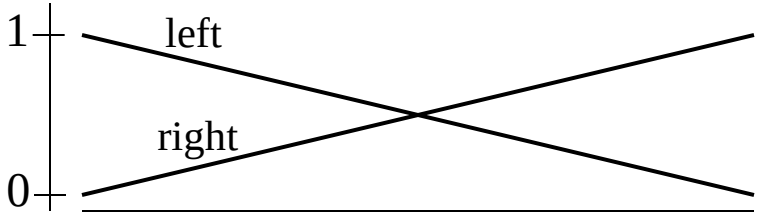
=

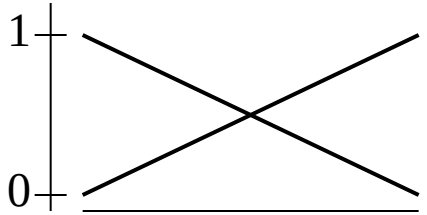


$$I_{blend} = \alpha I_{left} + (1-\alpha) I_{right}$$

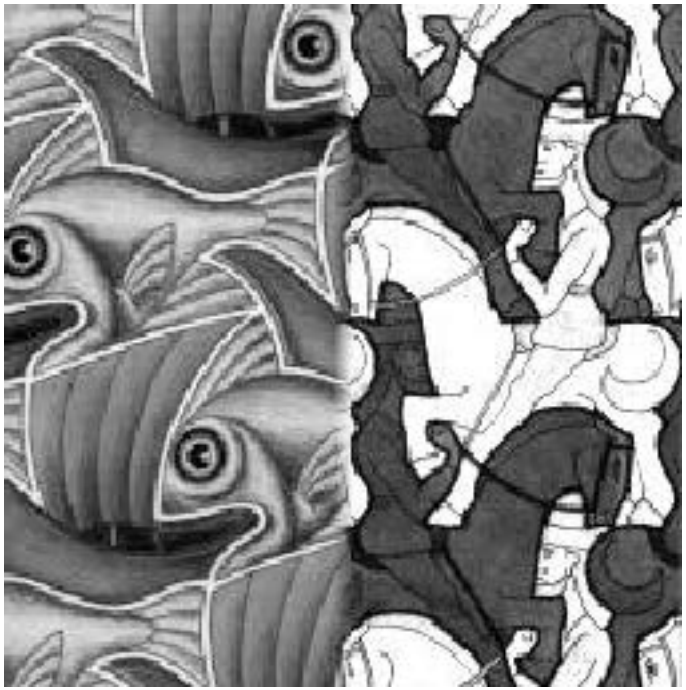
Affect of Window Size

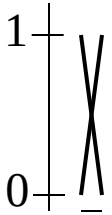


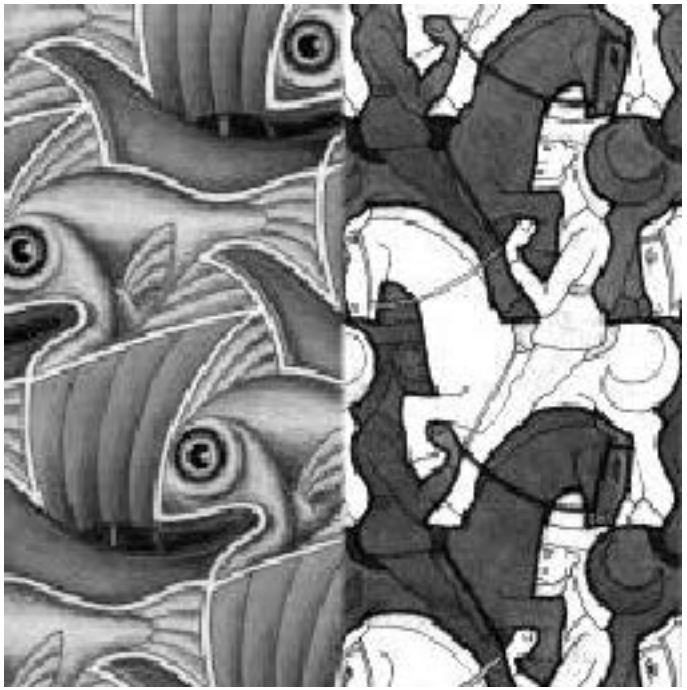


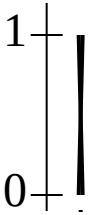


Affect of Window Size

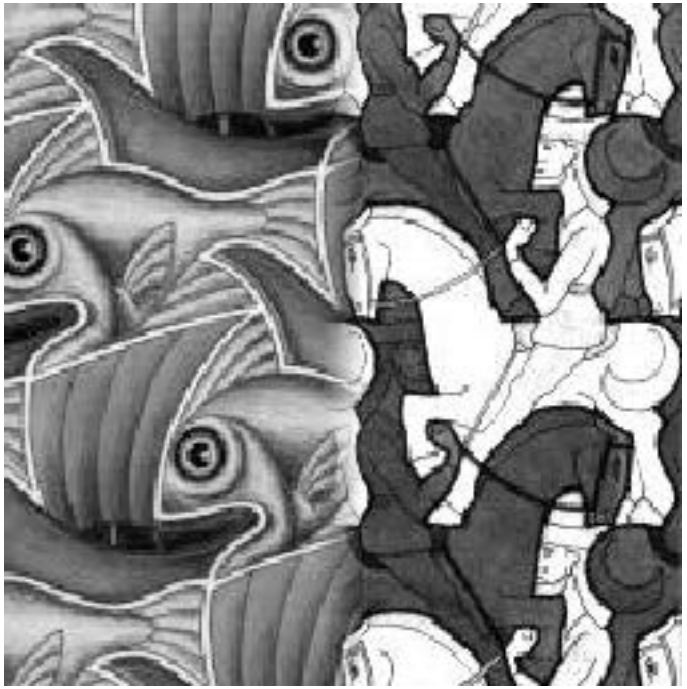


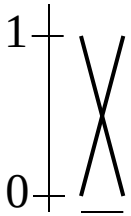






Good Window Size





“Optimal” Window: smooth but not ghosted

What is the Optimal Window?

To avoid seams

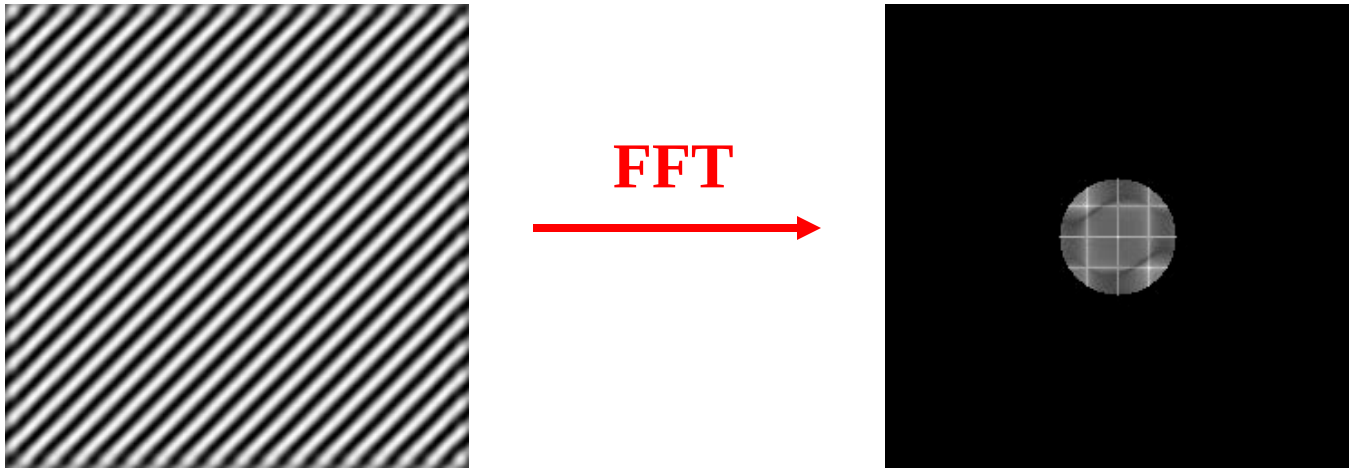
- window = size of largest prominent feature

To avoid ghosting

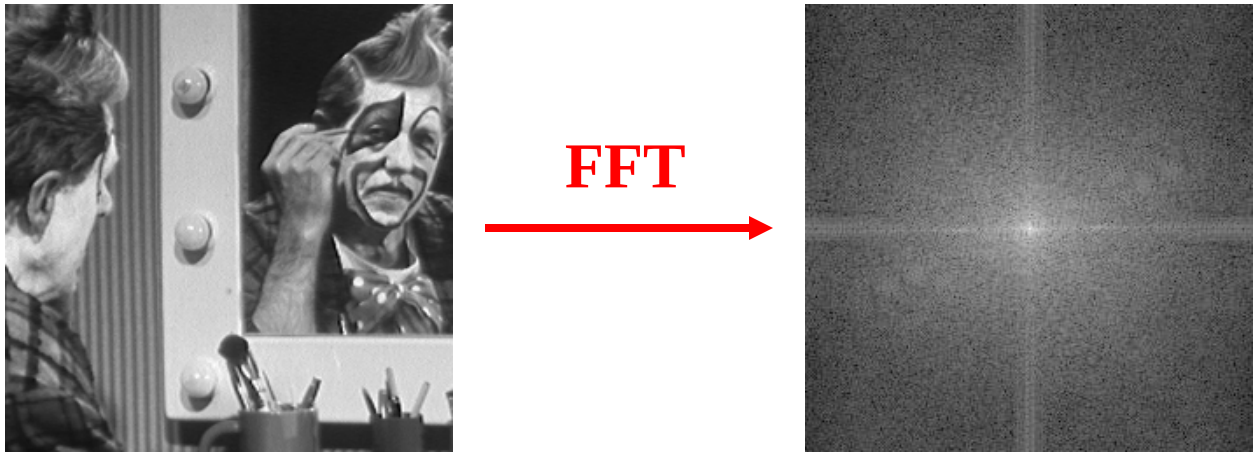
- window <= 2*size of smallest prominent feature

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



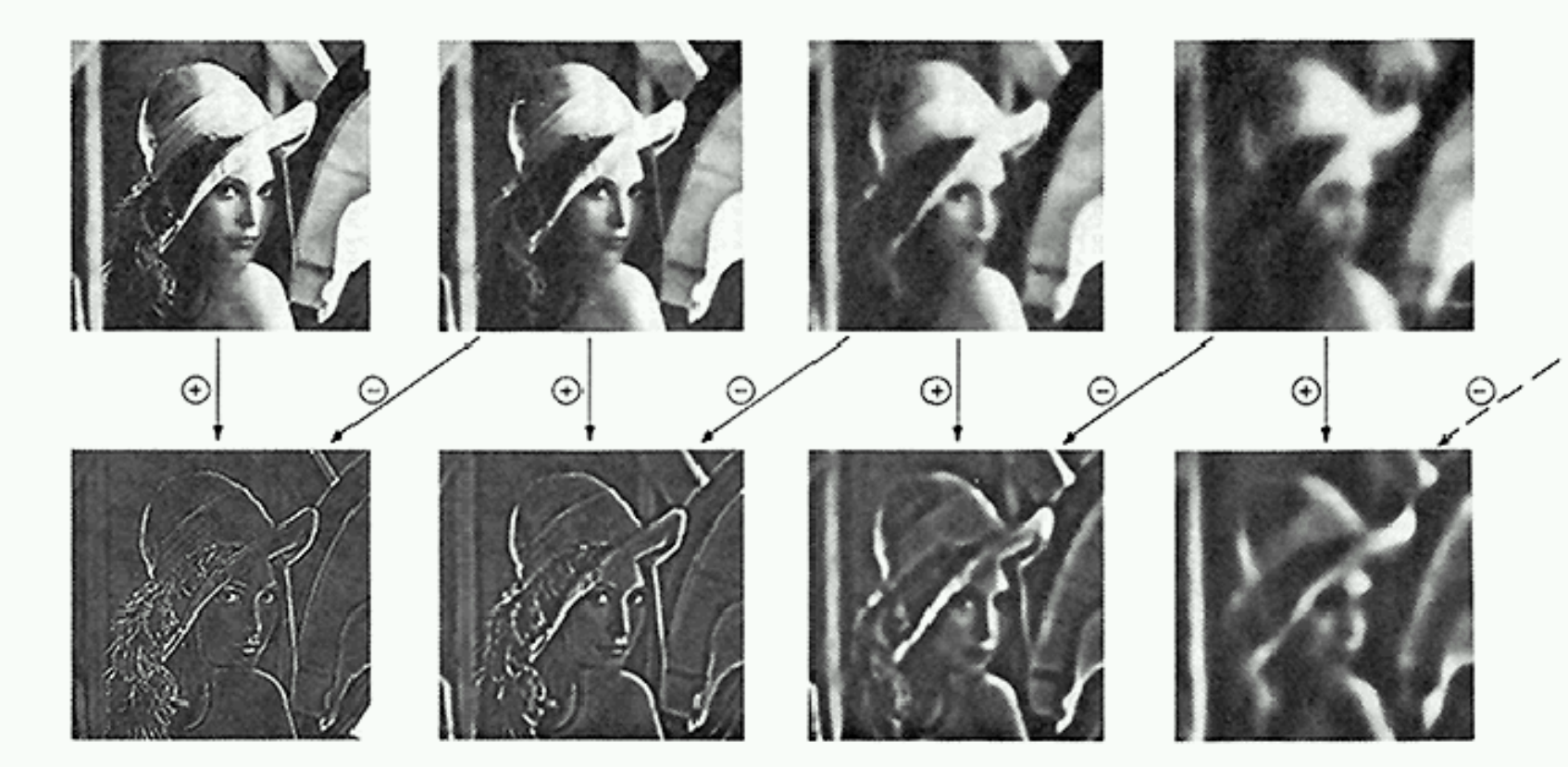
Idea (Burt and Adelson)

- 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*

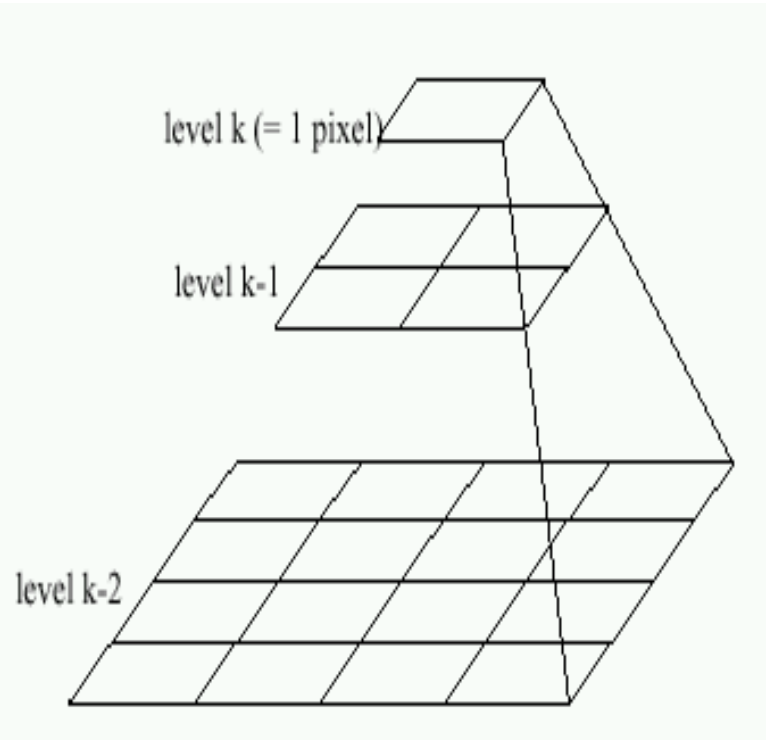
Octaves in the Spatial Domain

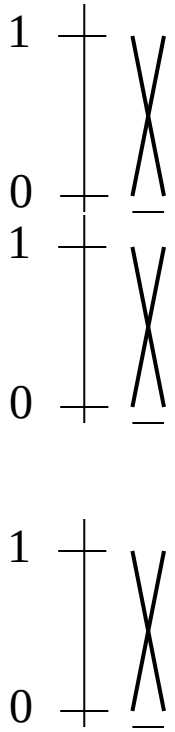
Lowpass Images

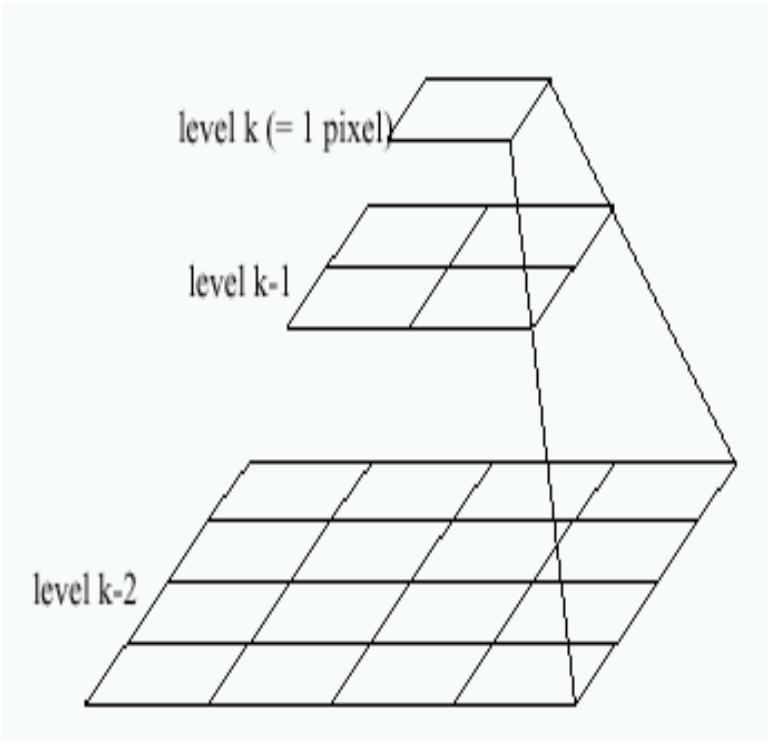


Bandpass Images

Pyramid Blending





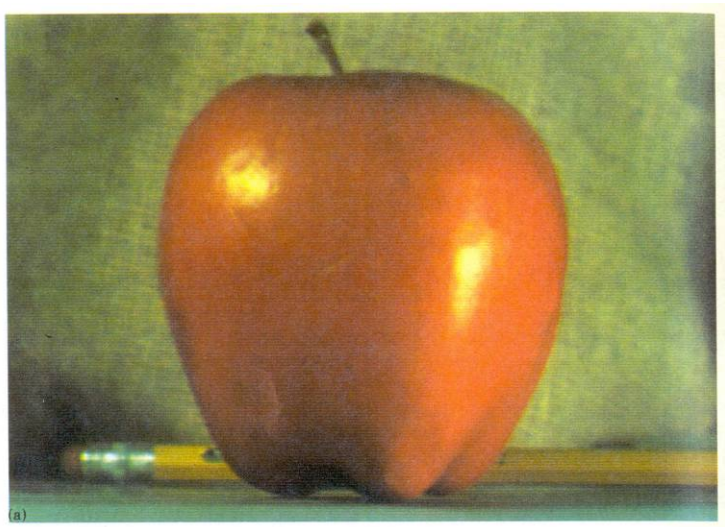


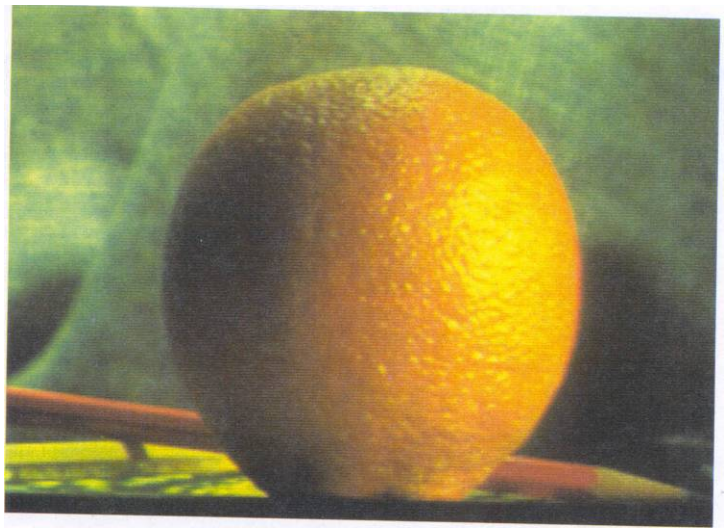
Left pyramid

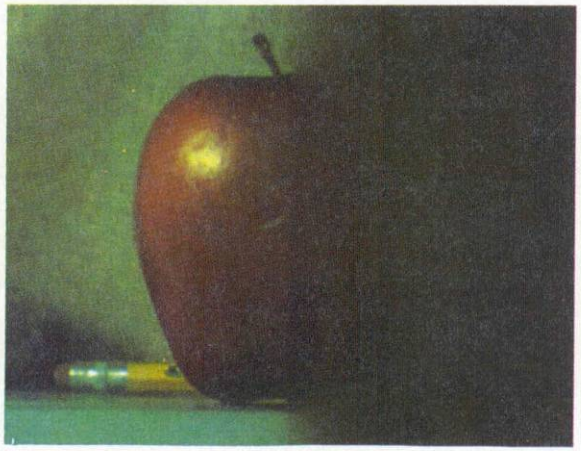
blend

Right pyramid

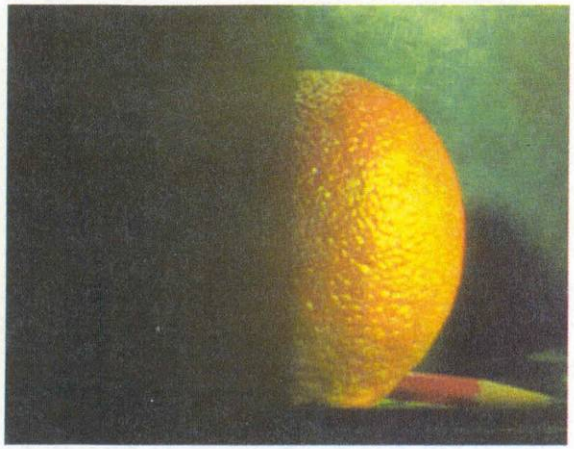
Pyramid Blending



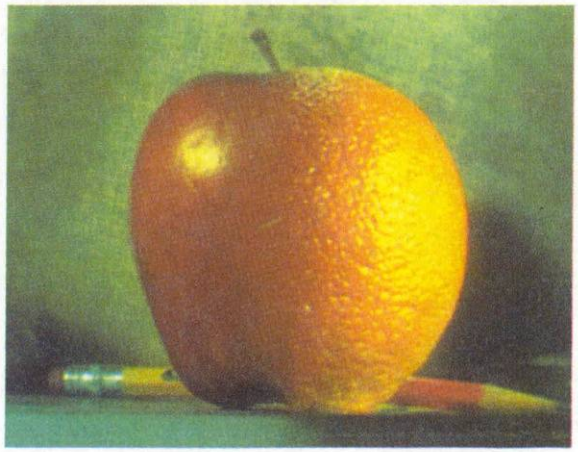




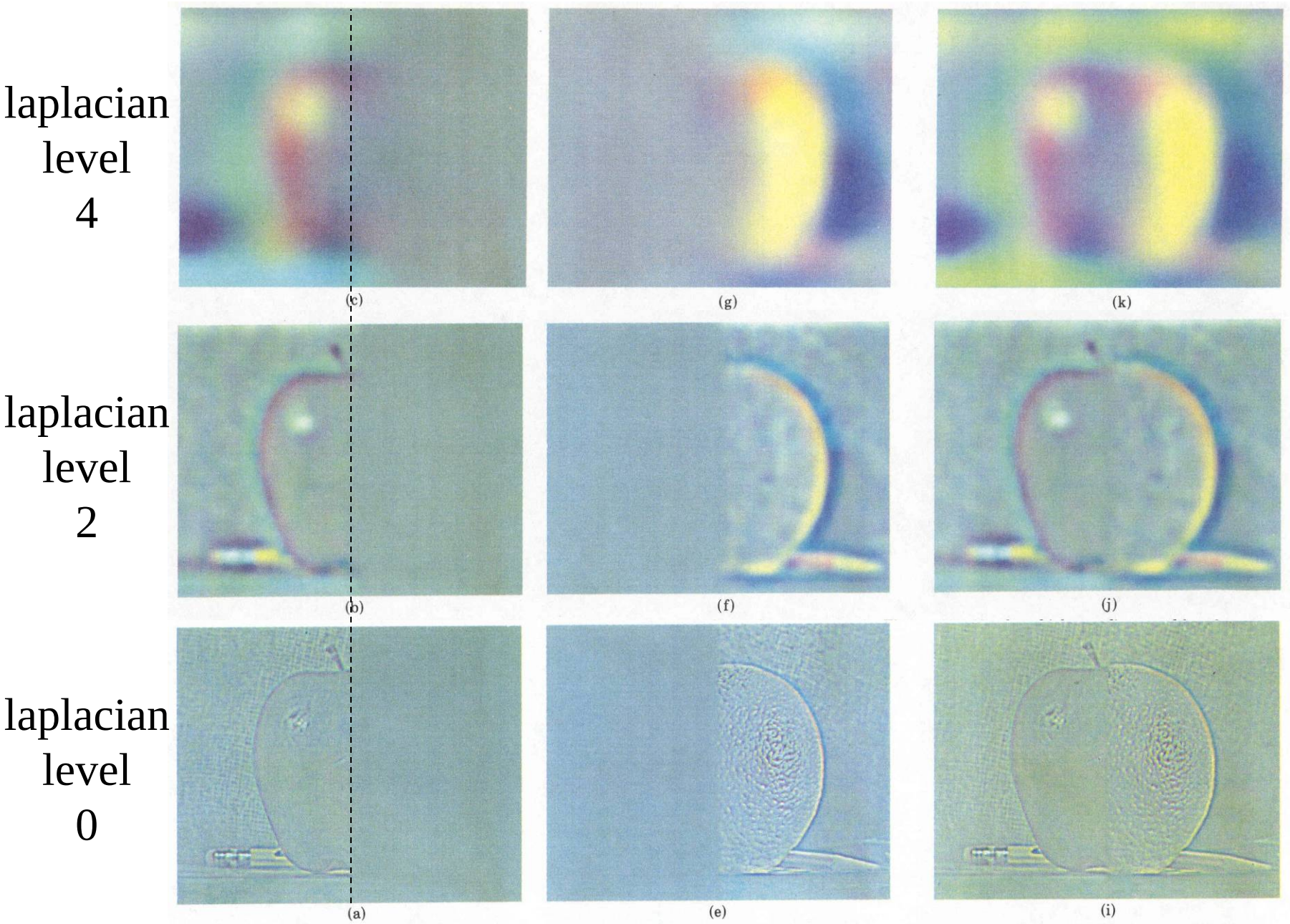
(d)



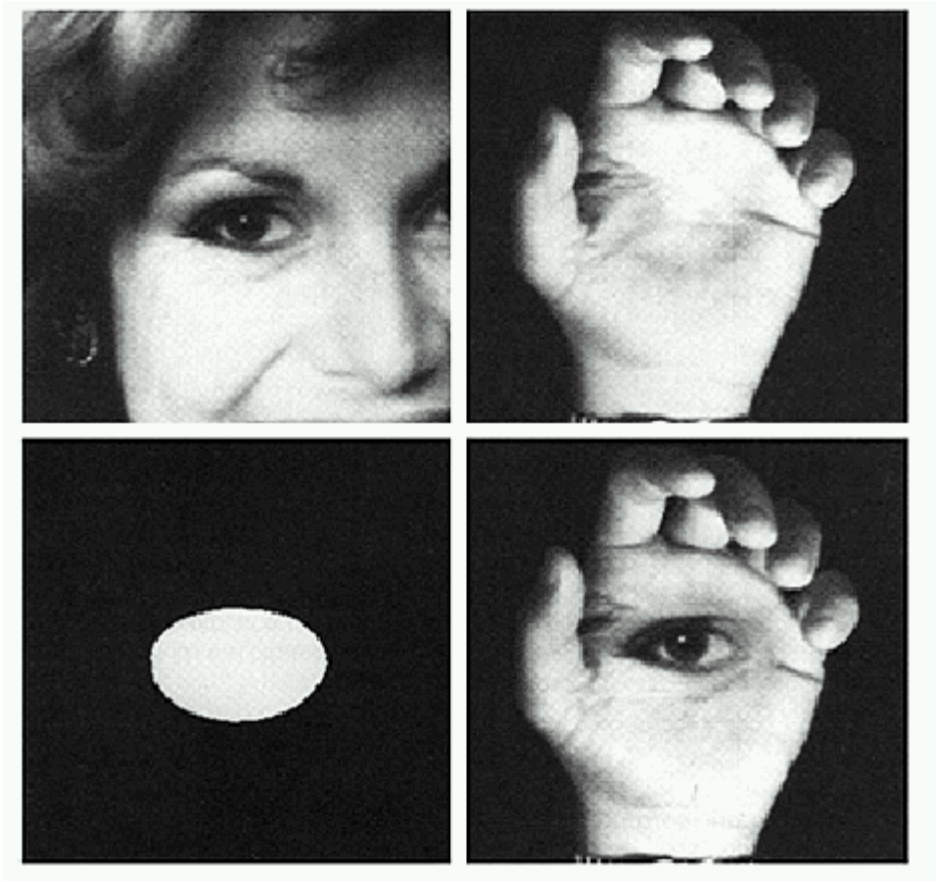
(h)



(l)


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

Blending Regions

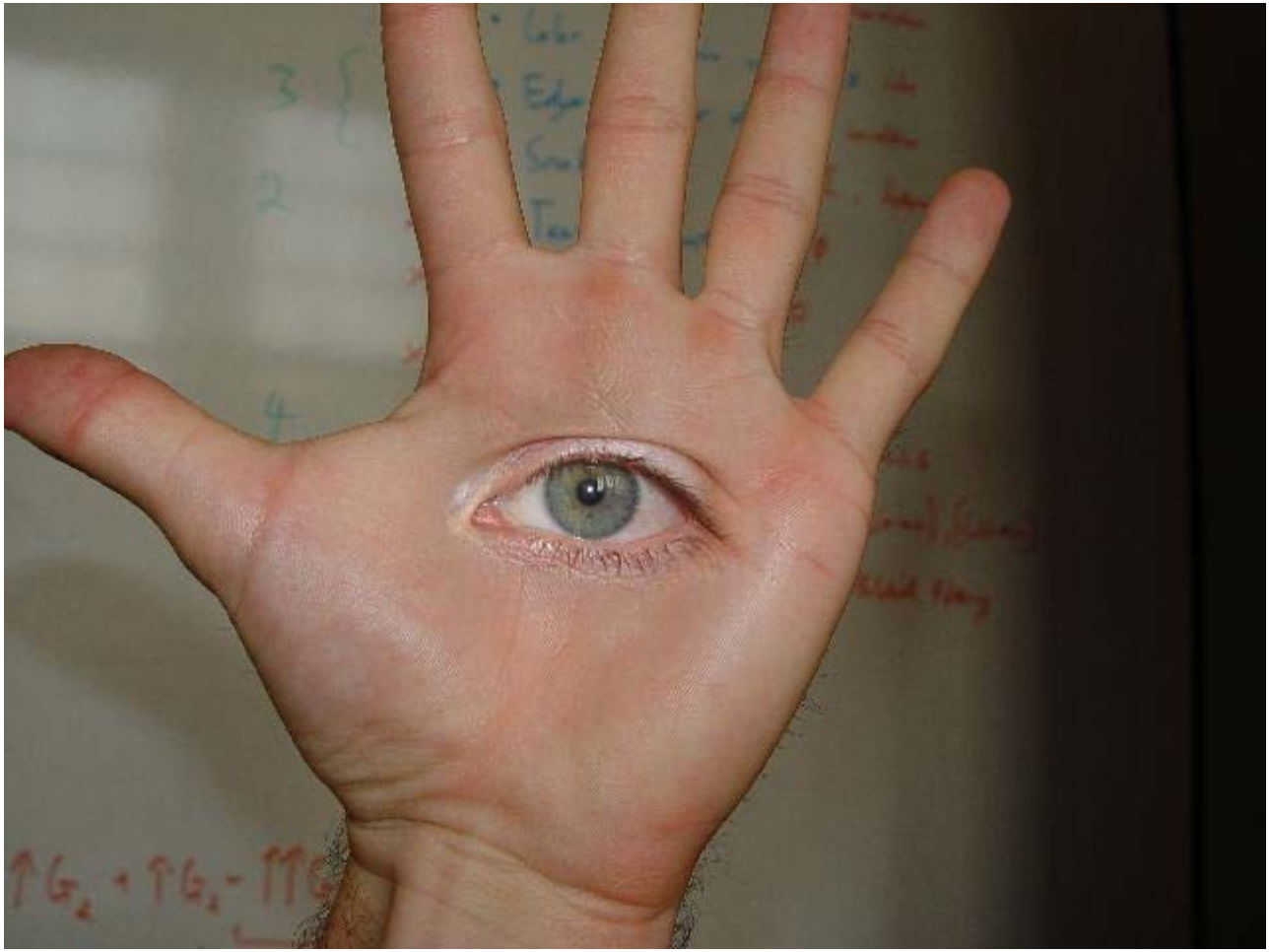


Laplacian Pyramid: Blending

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

Horror Photo



Results from this class (fall 2005)

© Chris Cameron

Season Blending (St. Petersburg)

Season Blending (St. Petersburg)

Simplification: Two-band Blending

Brown & Lowe, 2003

- Only use two bands: high freq. and low freq.
- Blends low freq. smoothly
- Blend high freq. with no smoothing: use binary alpha

2-band Blending

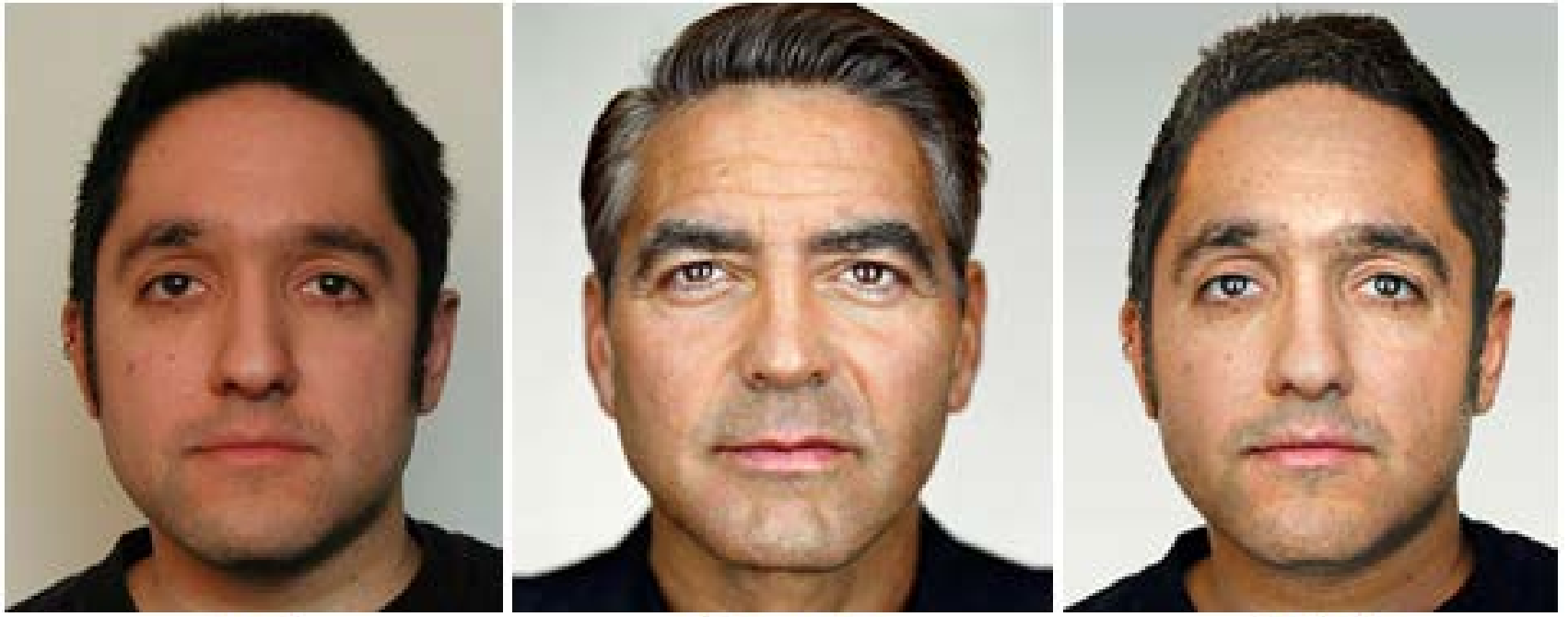
Low frequency (λ > 2 pixels)

High frequency (λ < 2 pixels)

Linear Blending

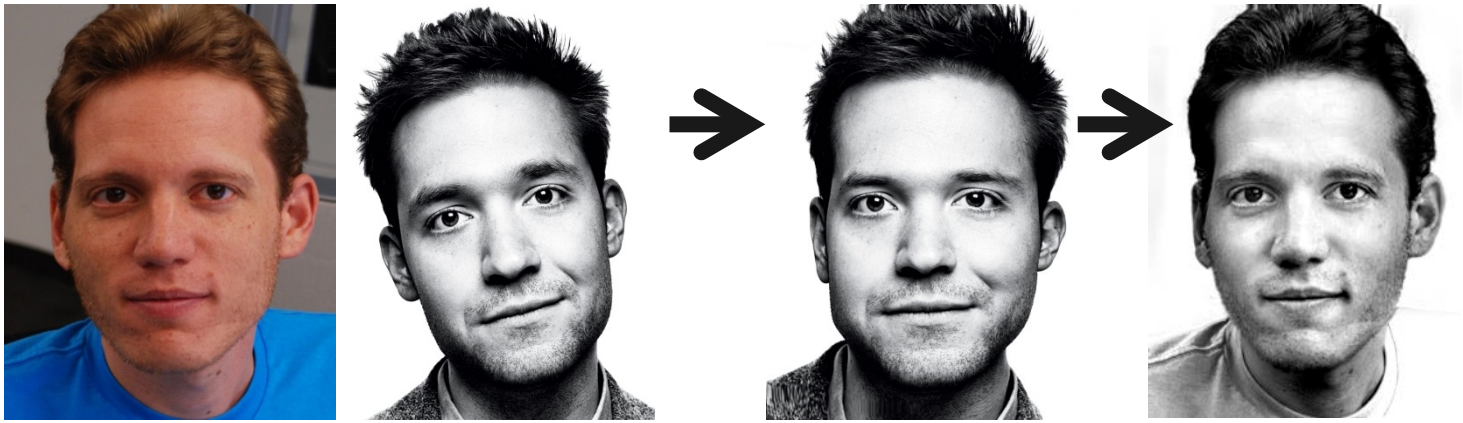
2-band Blending

“Style Transfer for Headshot Portraits” (SIGGRAPH ‘14)

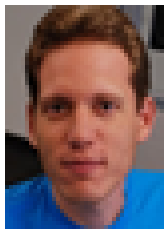


Input Example Output

Pipeline



Step 2: multi-scale local transfer



Input

Example

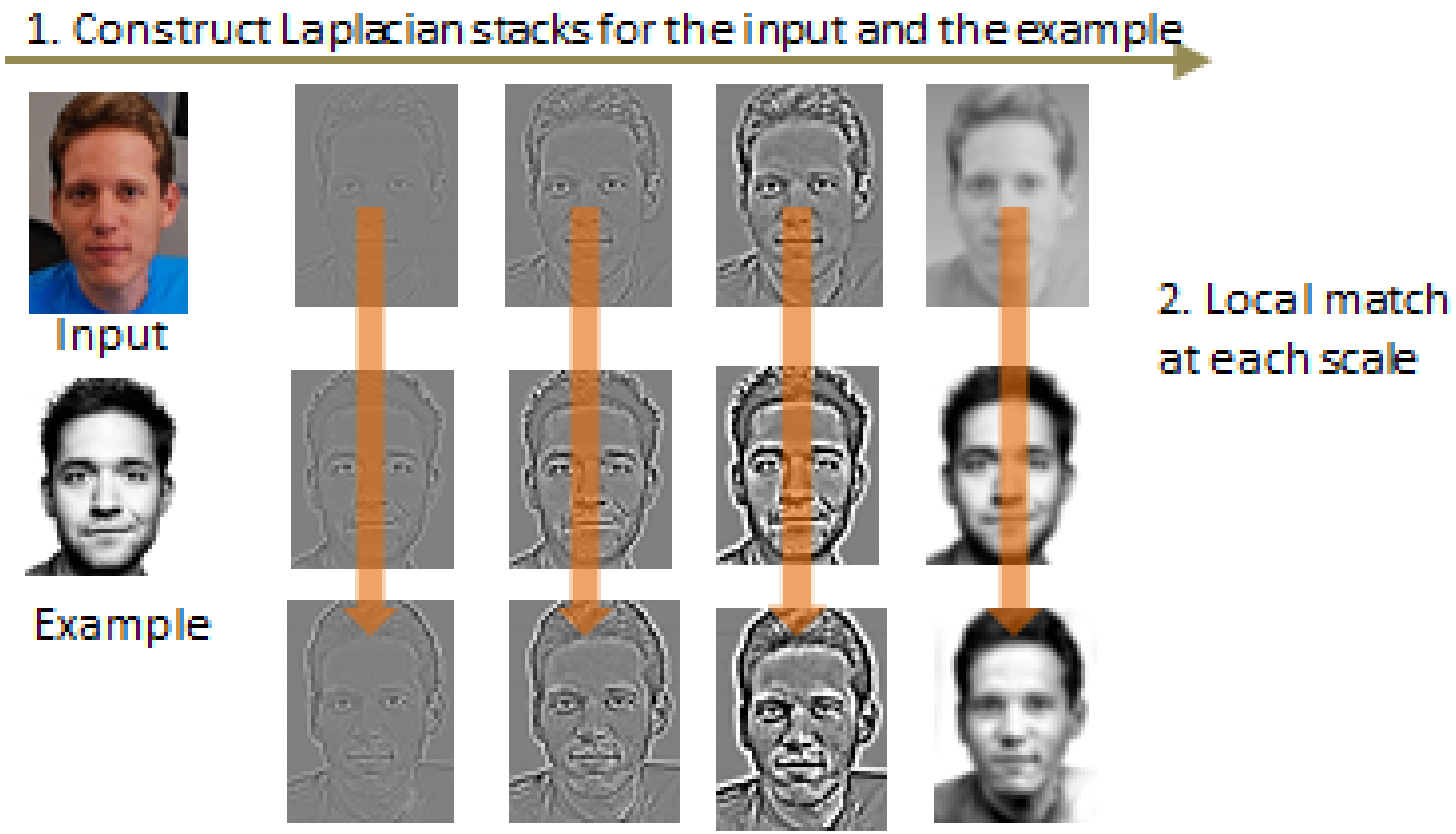
Step 2: multi-scale local transfer

1. Construct Laplacian stacks for the input and the example

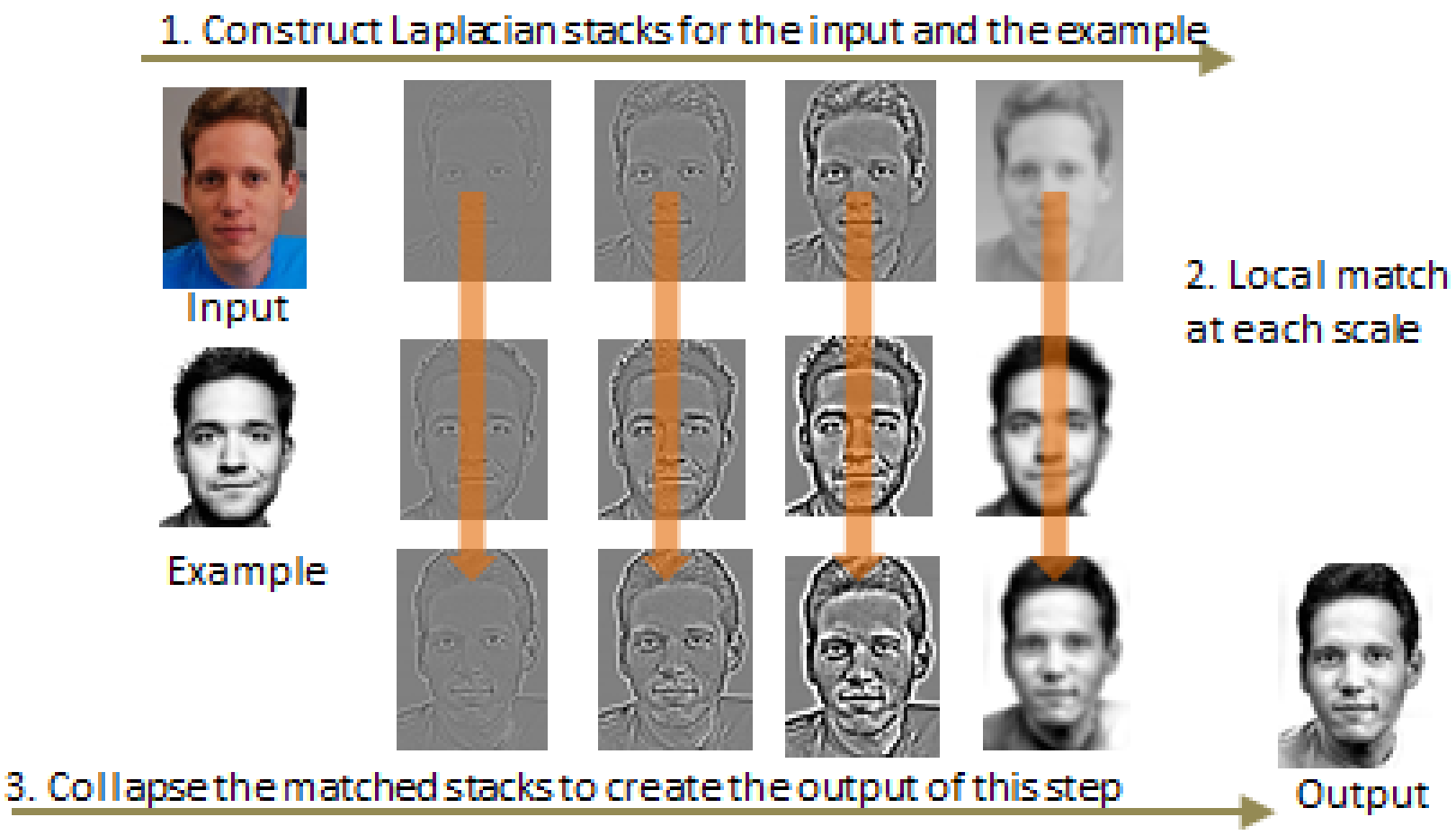
Input

Example

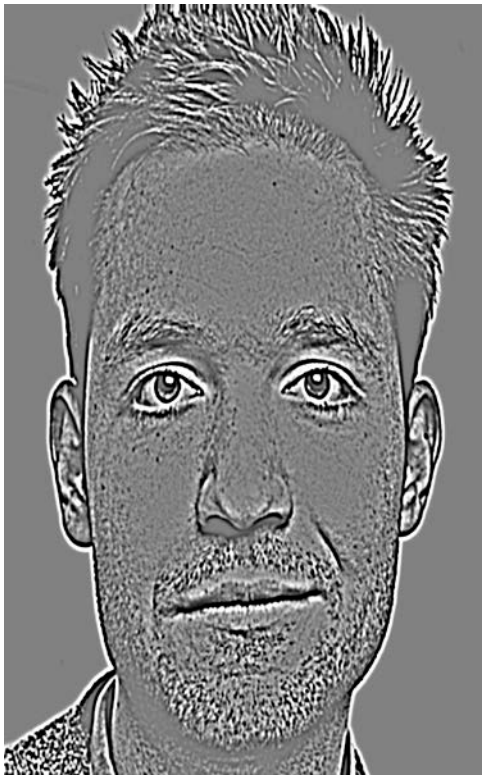
Step 2: multi-scale local transfer



Step 2: multiscale transfer of local statistics

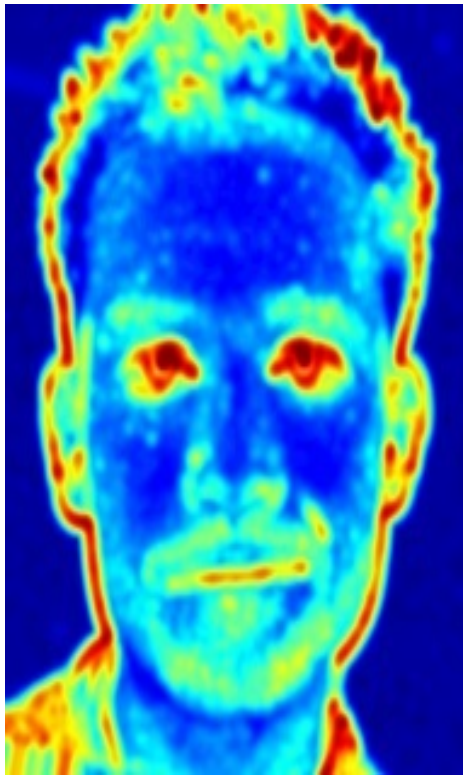


Local energy S



L

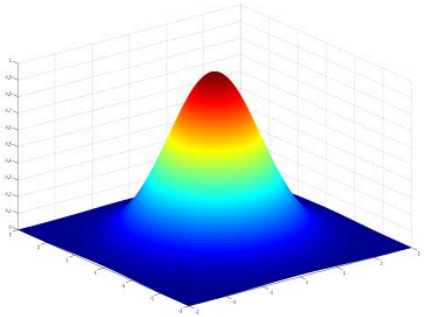
Example Laplacian



S

Local energy

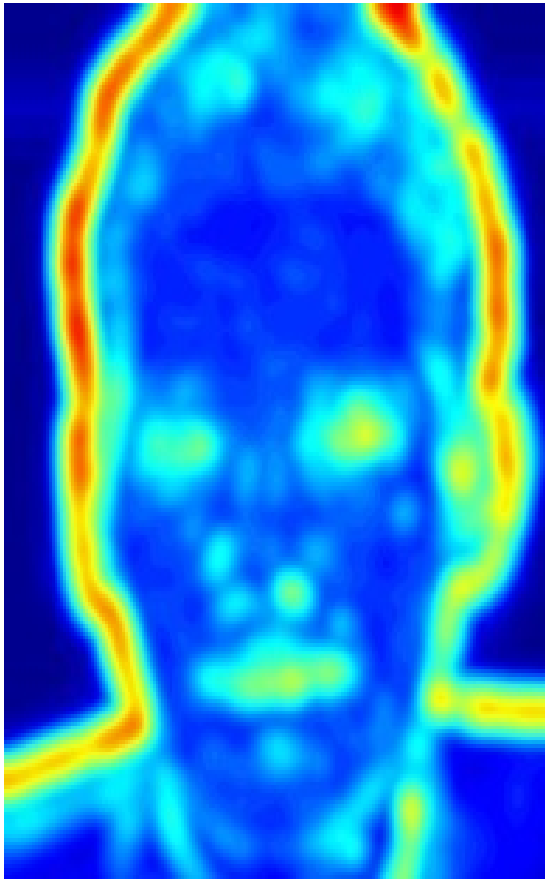
$$S = L^2 \otimes G_\ell$$



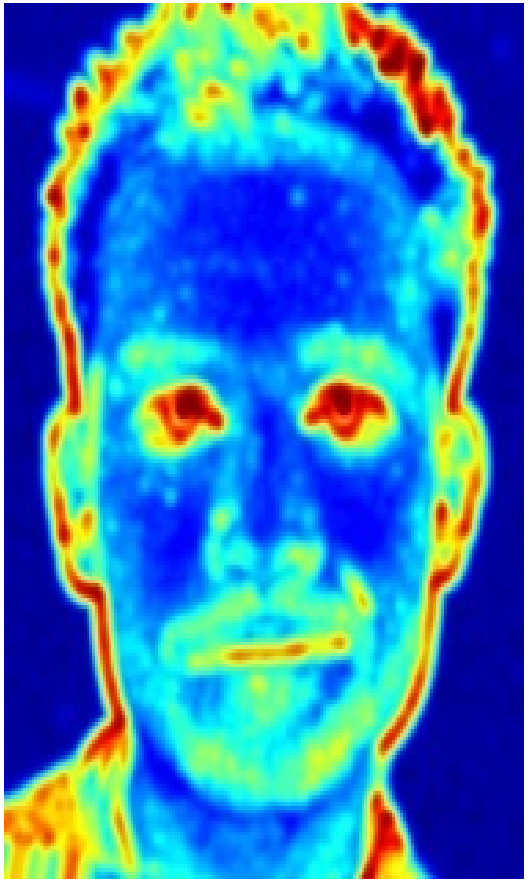
G_ℓ

Gaussian kernel at this scale

At each scale: match local energy



Input energy

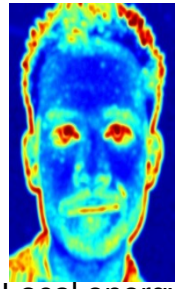


Example energy

At each scale: match local energy

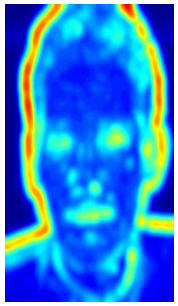
Compute
the gain map

Example Laplacian



Local energy S[E]

Input Laplacian



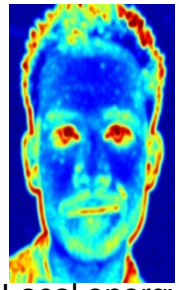
Local energy S[I]

Gain map = $\sqrt{\frac{\text{warp}(S[E])}{S[I]}}$

At each scale: match local energy

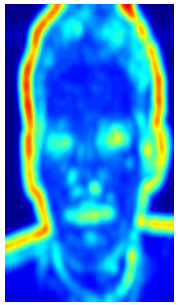
Compute
the gain map

Example Laplacian



Local energy S[E]

Input Laplacian



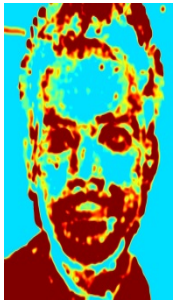
Local energy S[I]

Gain map = $\sqrt{\frac{\text{warp}(S[E])}{S[I]}}$

Modulate
the input Laplacian

Input Laplacian

×



Gain map

=

Output Laplacian