

The Frequency Domain



Somewhere in Cinque Terre, May 2005

CS194: Image Manipulation & Computational Photography

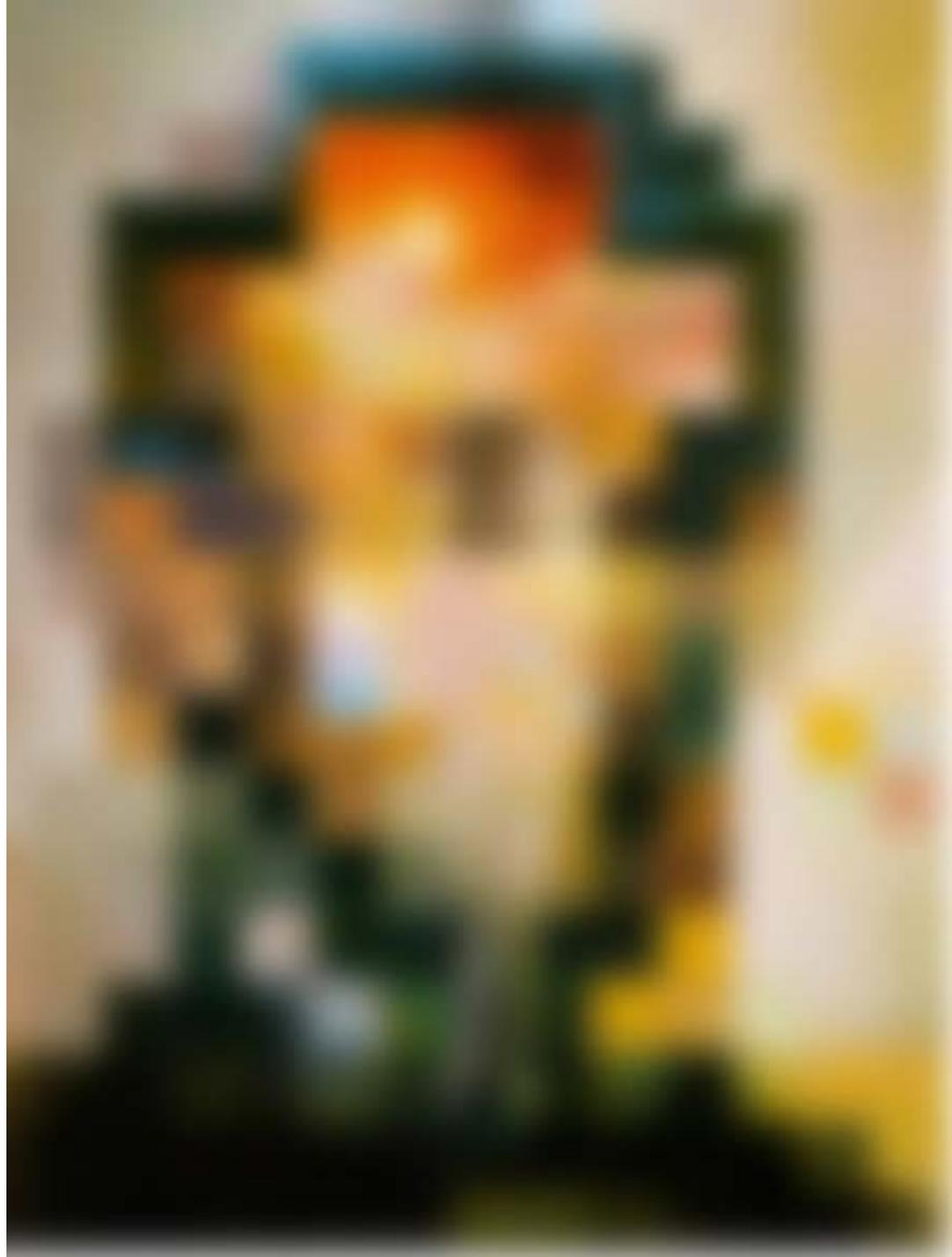
Many slides borrowed
from Steve Seitz

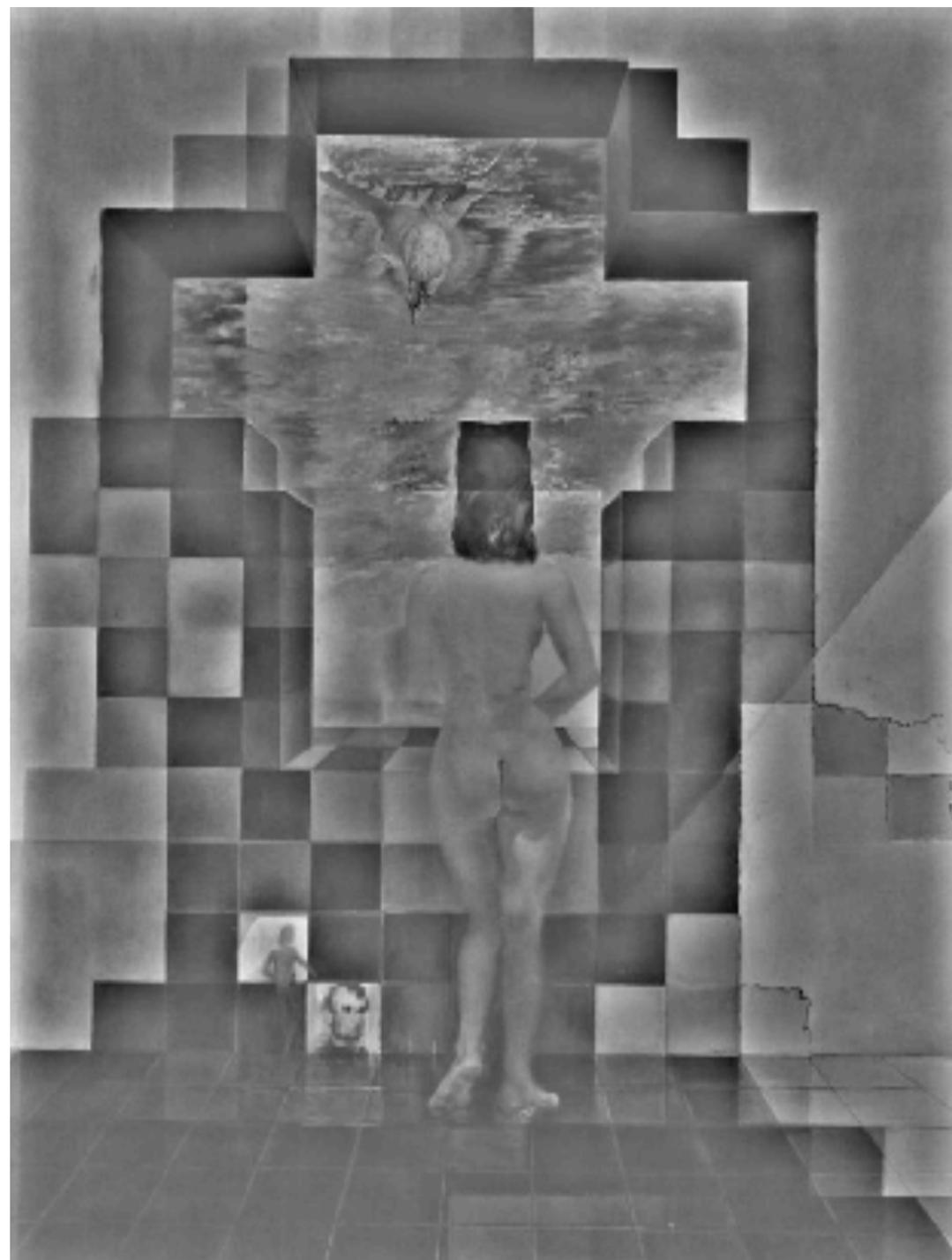
Alexei Efros, UC Berkeley, Fall 2015



Salvador Dalí

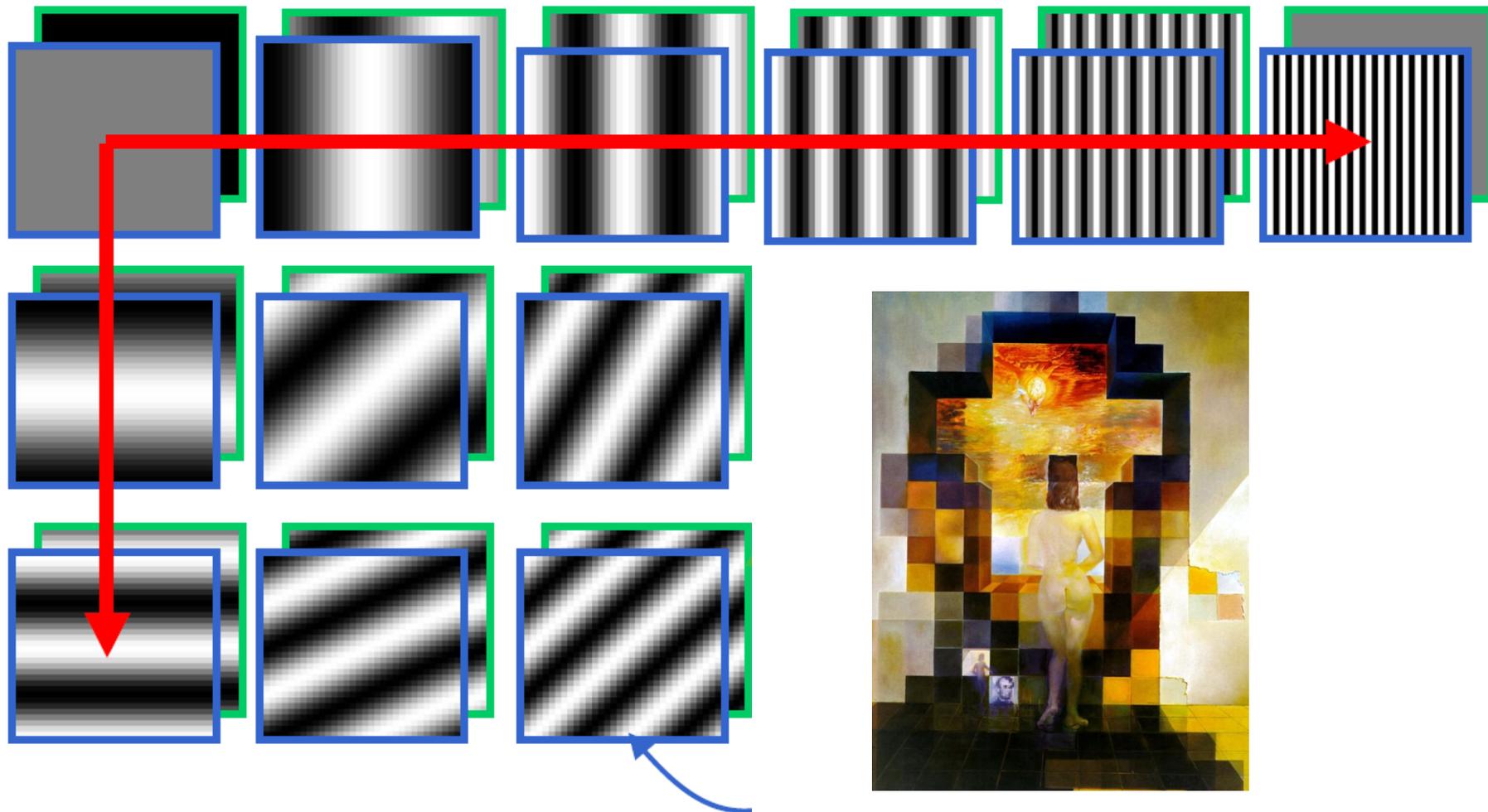
*"Gala Contemplating the Mediterranean Sea,
which at 30 meters becomes the portrait
of Abraham Lincoln", 1976*





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 can be rewritten as a weighted sum of sines and cosines of different frequencies.

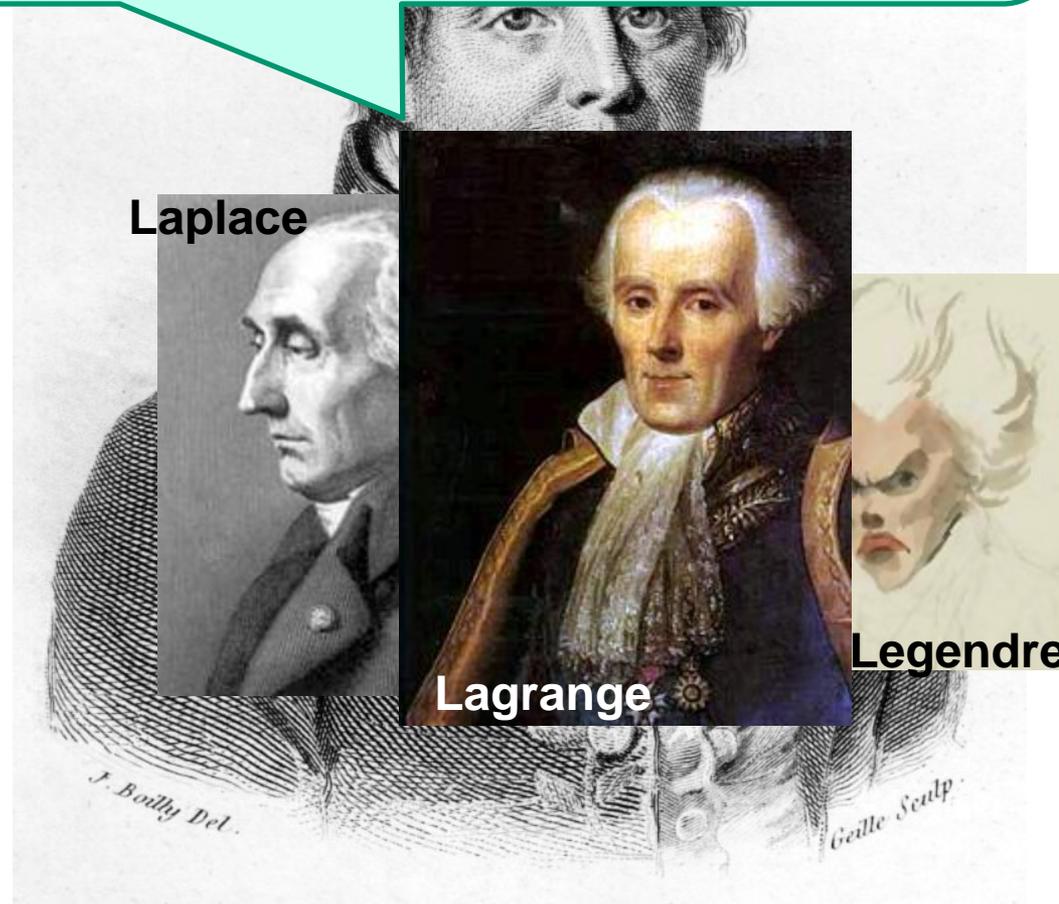
...the manner in which the author arrives at these equations is not exempt of difficulties and...his analysis to integrate them still leaves something to be desired on the score of generality and even rigour.

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



A sum of sines

Our building block:

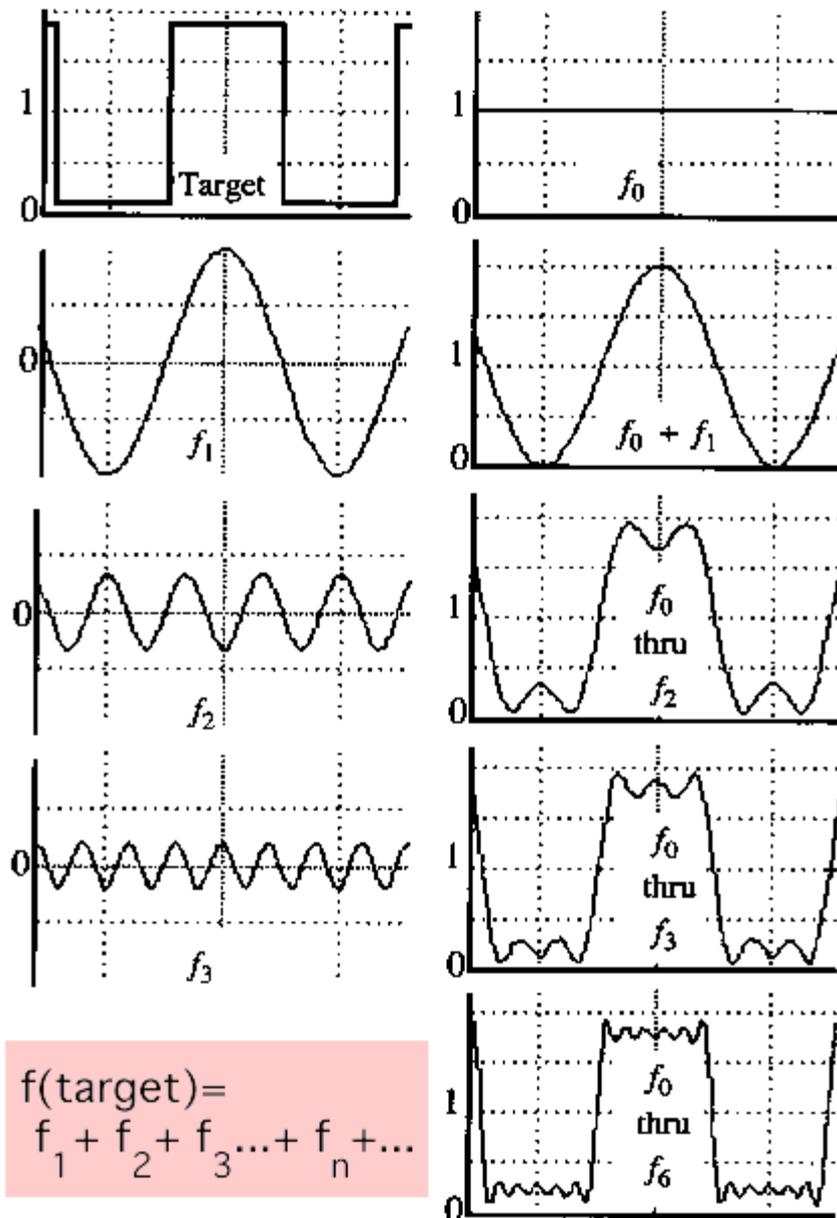
$$A \sin(\omega x + \phi)$$

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?



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 ∞ , $F(\omega)$ holds the amplitude A and phase ϕ of the corresponding sine $A \sin(\omega x + \phi)$

- How can F hold both?

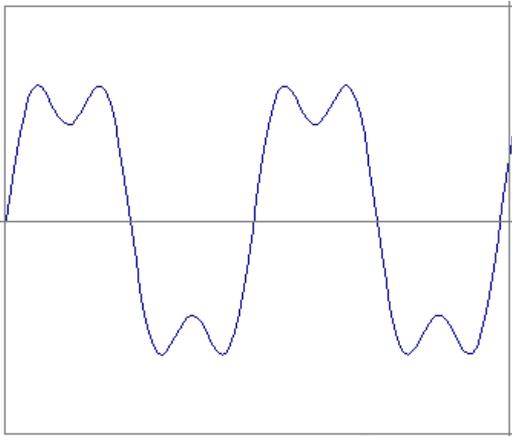
$$F(\omega) = R(\omega) + iI(\omega)$$
$$A = \pm \sqrt{R(\omega)^2 + I(\omega)^2} \quad \phi = \tan^{-1} \frac{I(\omega)}{R(\omega)}$$

We can always go back:



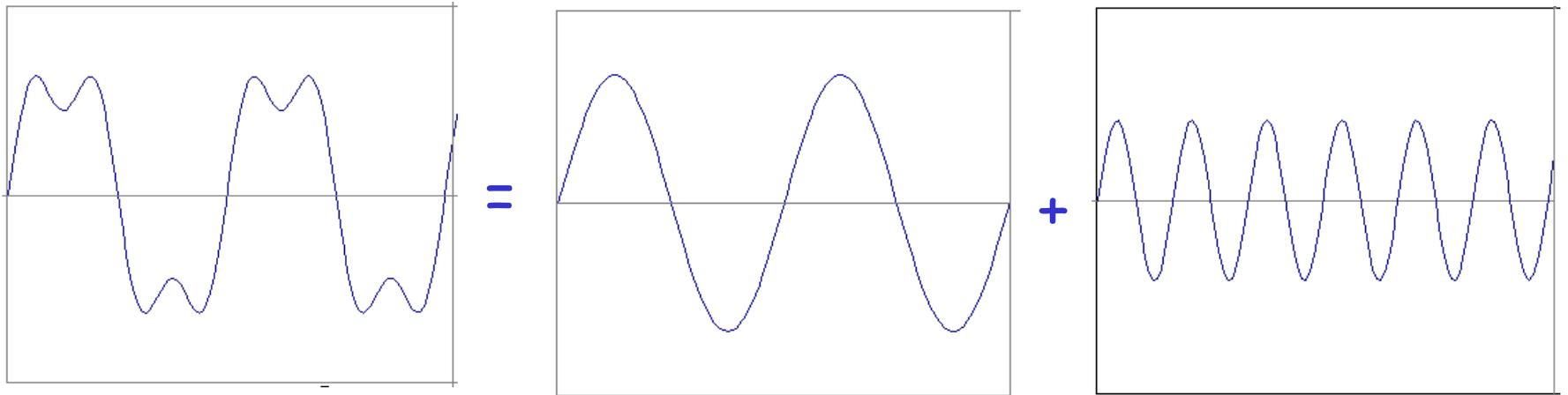
Time and Frequency

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



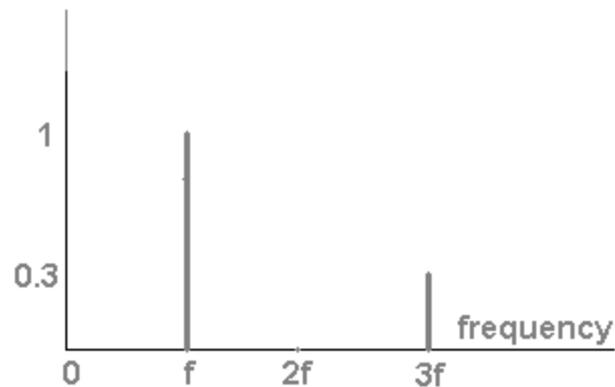
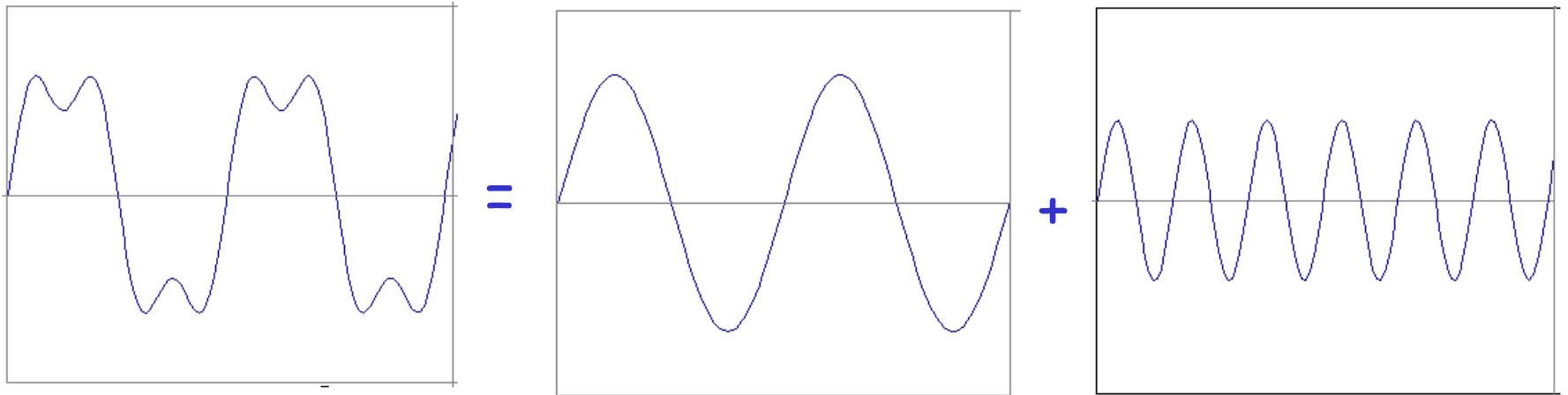
Time and Frequency

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



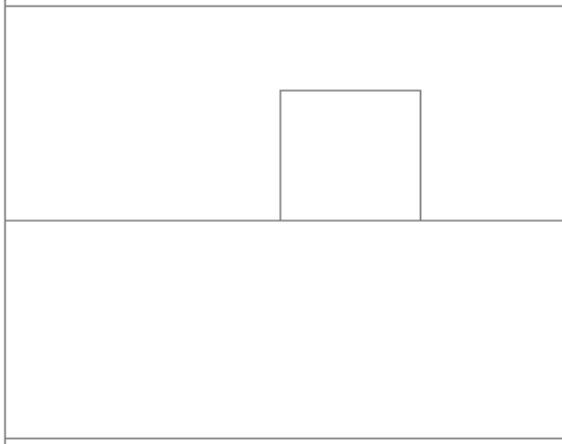
Frequency Spectra

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

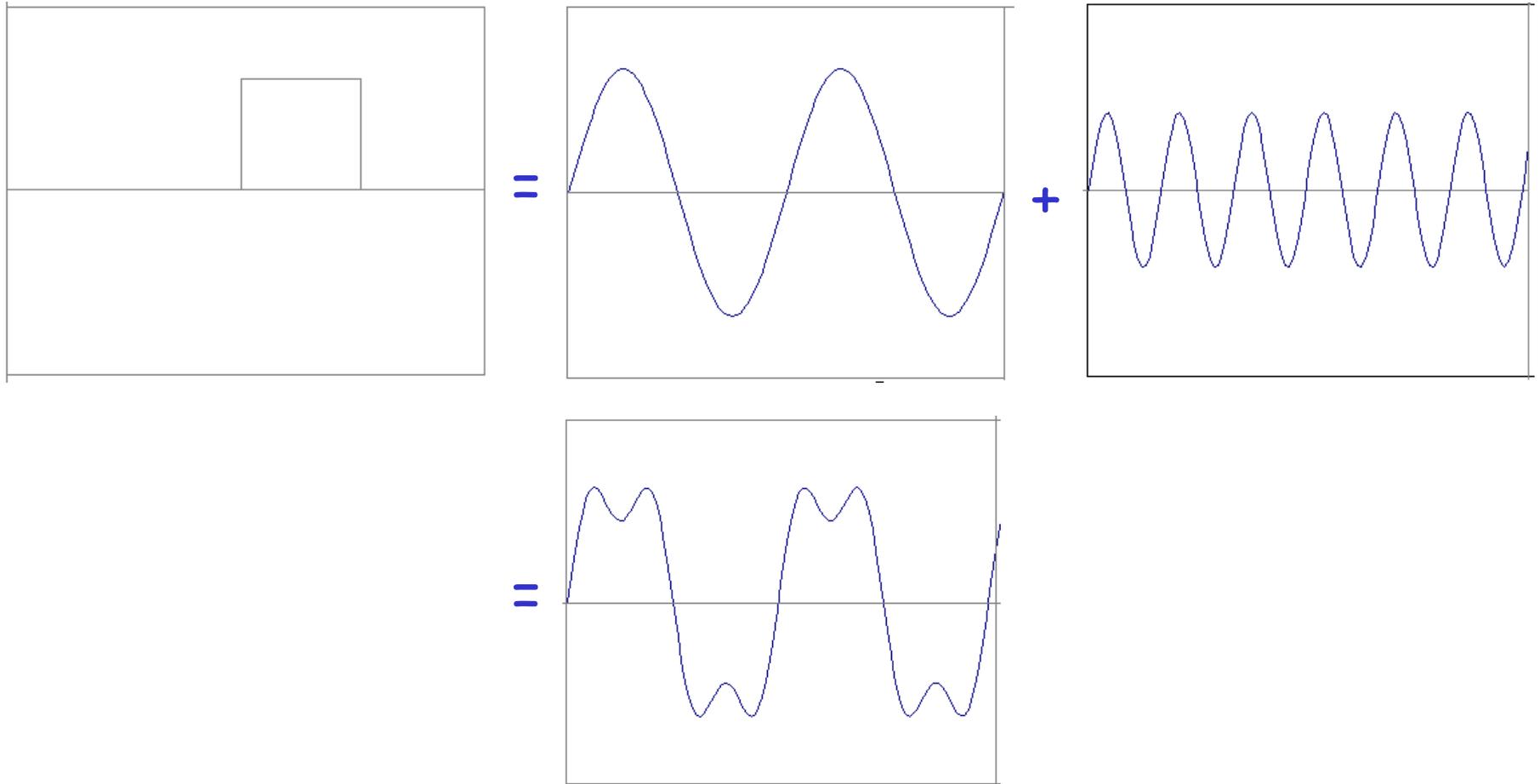


Frequency Spectra

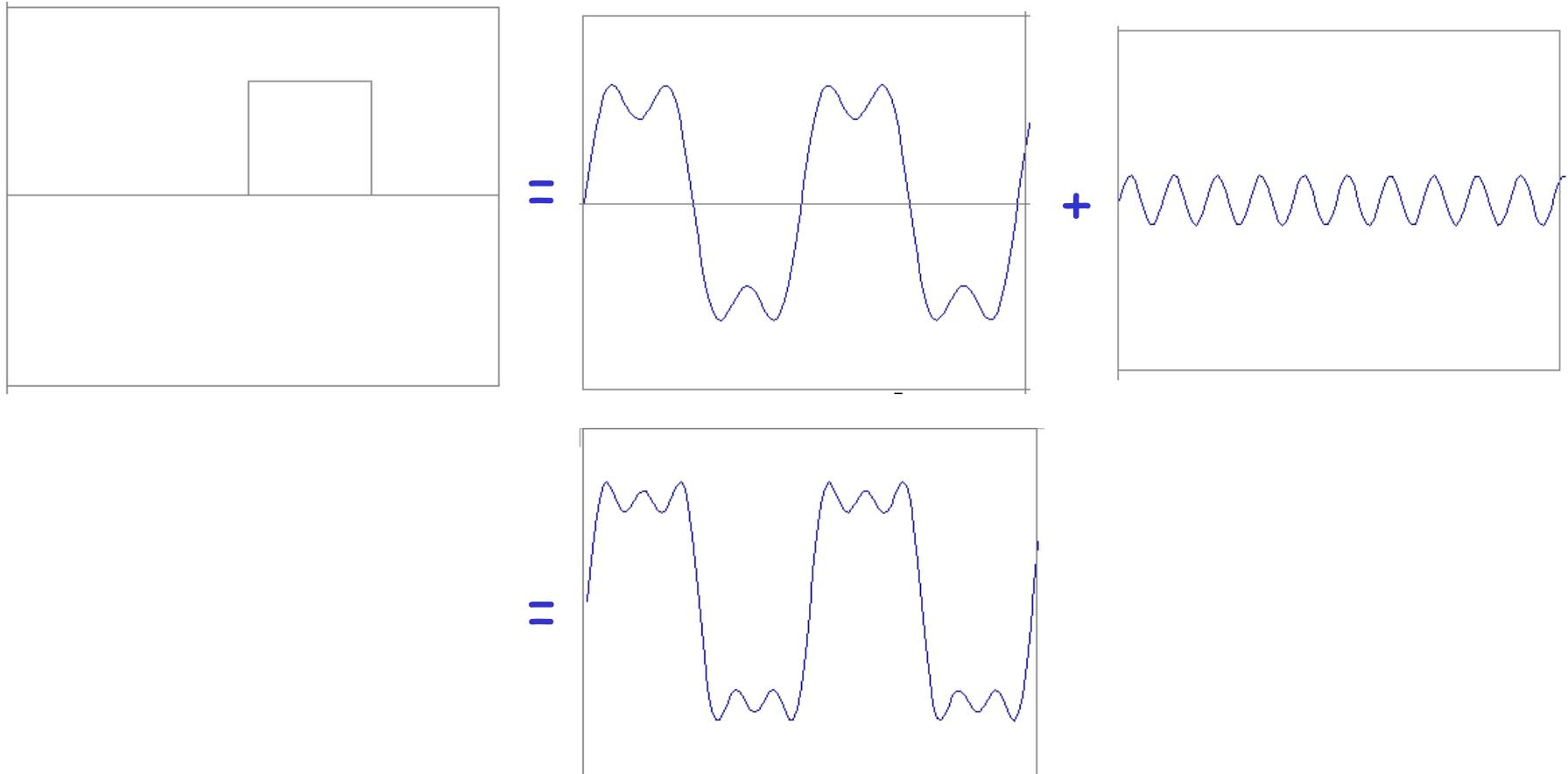
Usually, frequency is more interesting than the phase



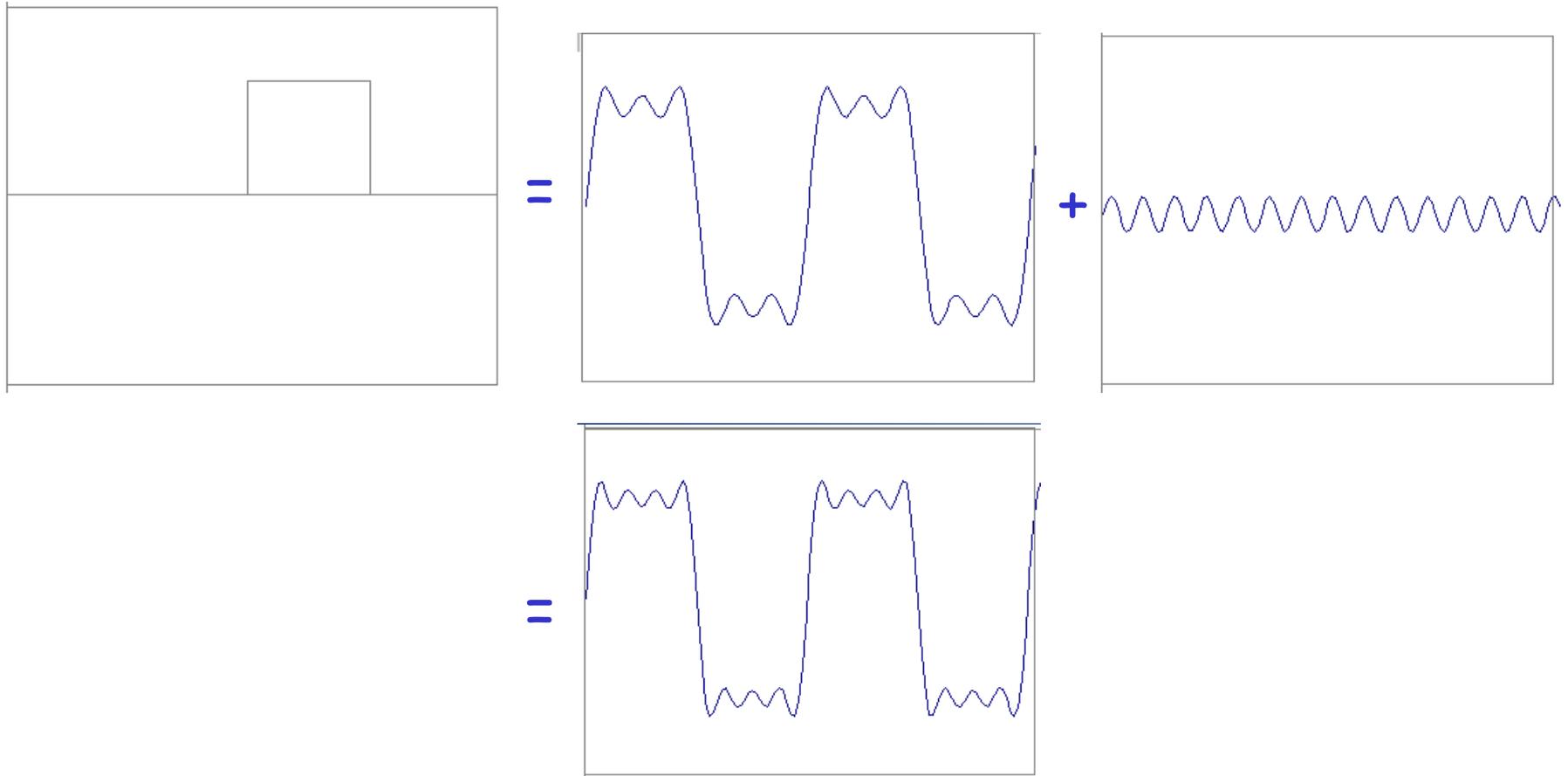
Frequency Spectra



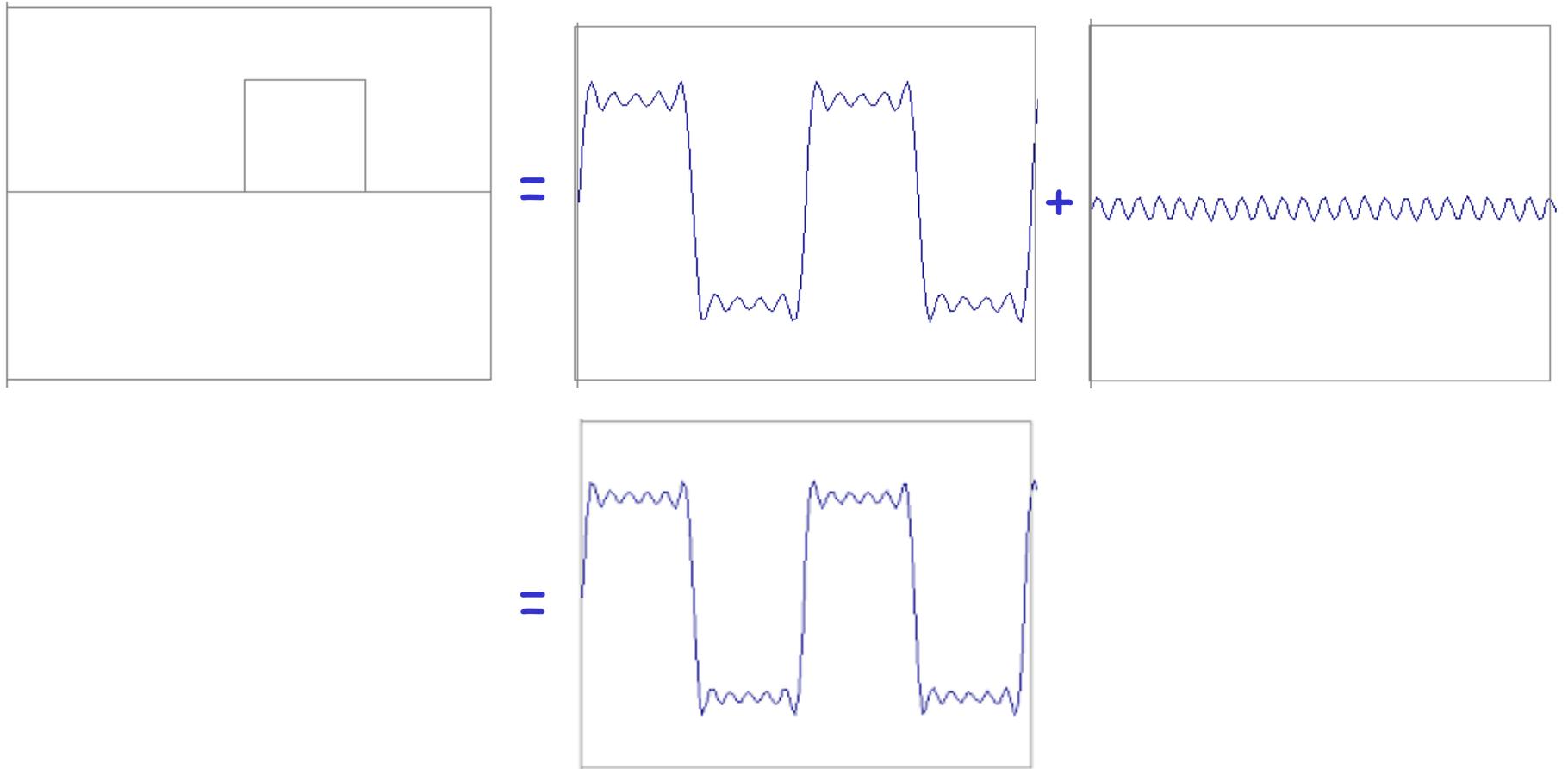
Frequency Spectra



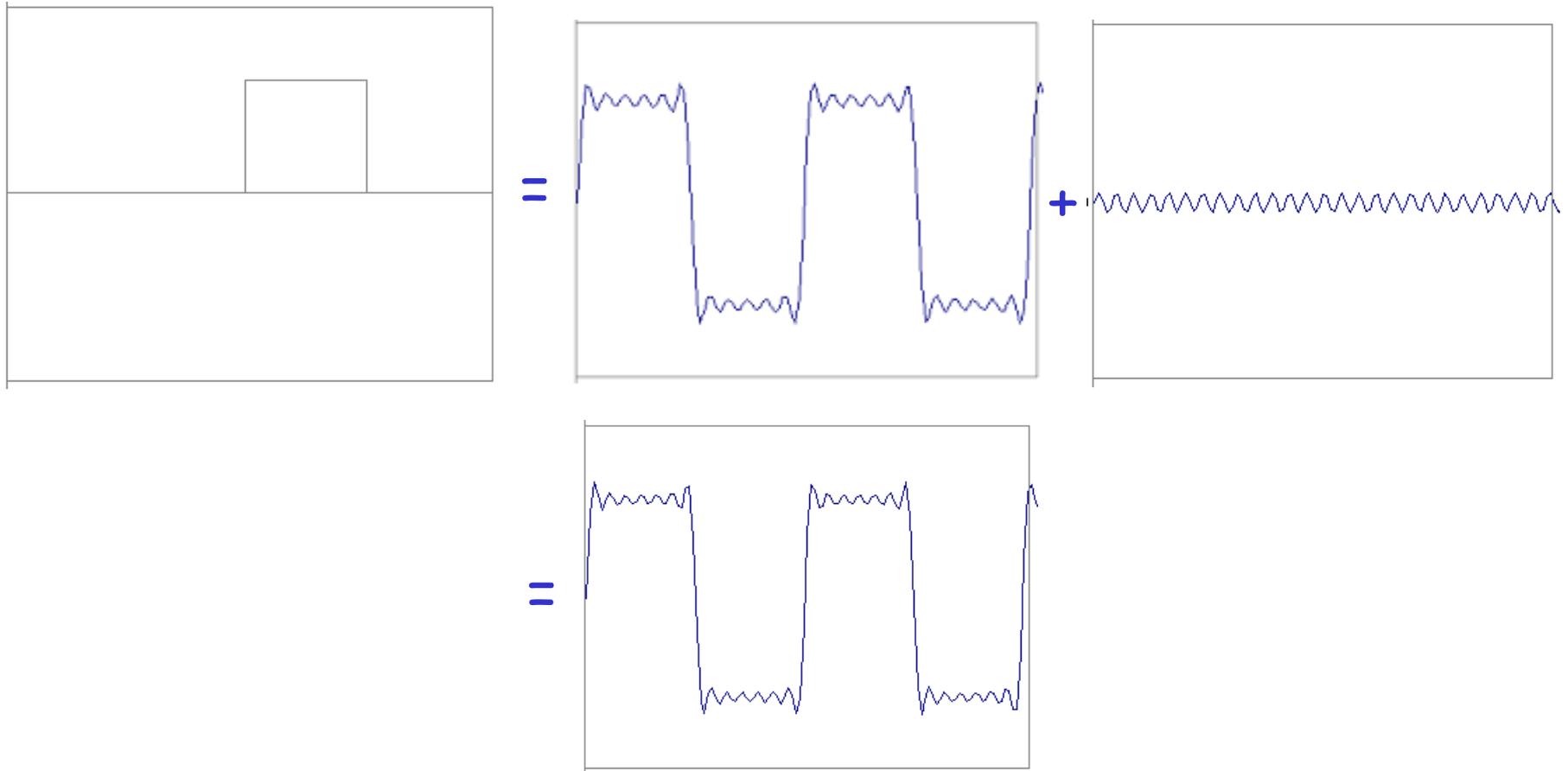
Frequency Spectra



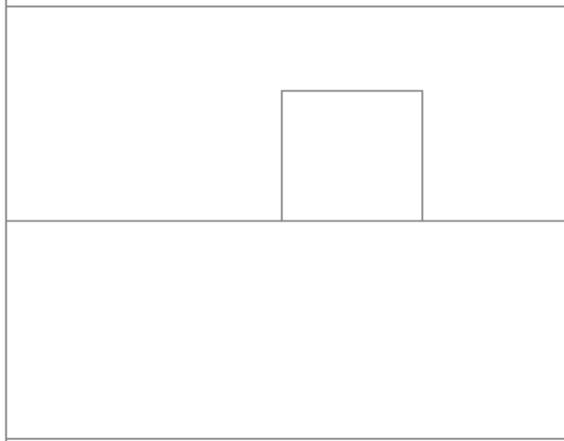
Frequency Spectra



Frequency Spectra

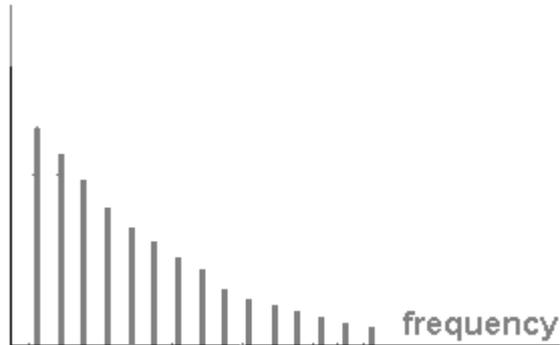


Frequency Spectra

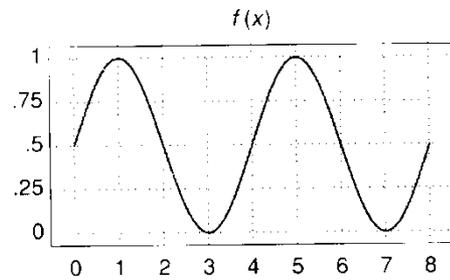


=

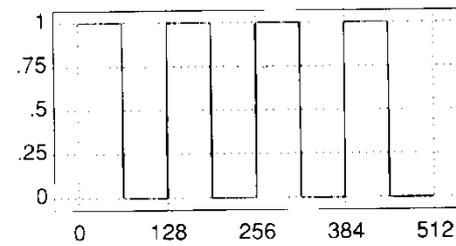
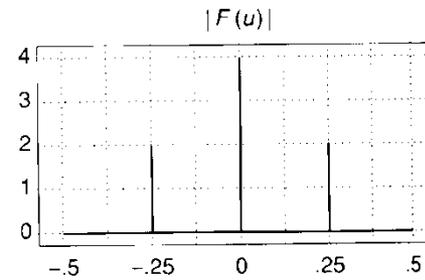
$$A \sum_{k=1}^{\infty} \frac{1}{k} \sin(2\pi kt)$$



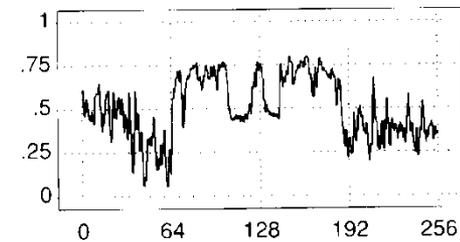
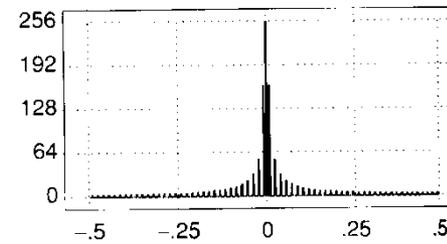
Frequency Spectra



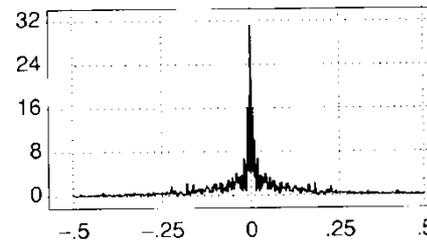
(a)



(b)

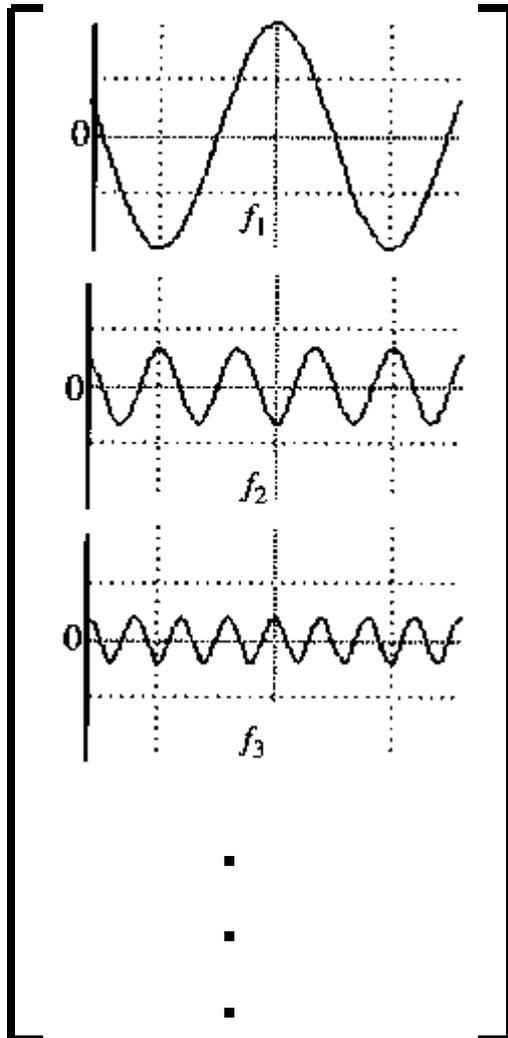


(c)

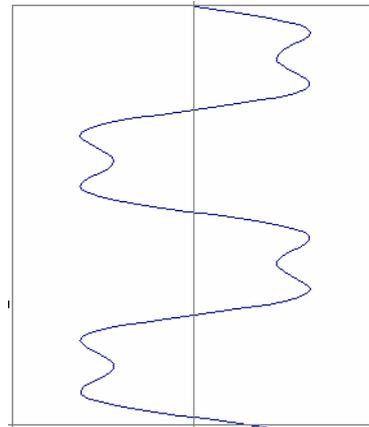


FT: Just a change of basis

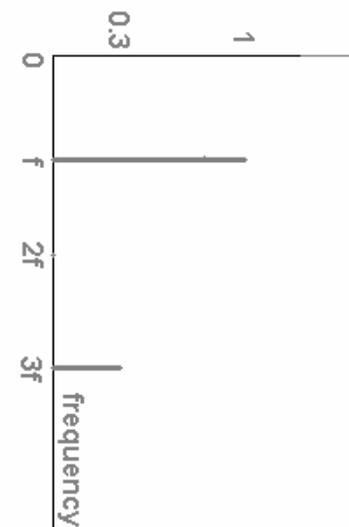
$$\mathcal{M} * f(x) = F(\omega)$$



*

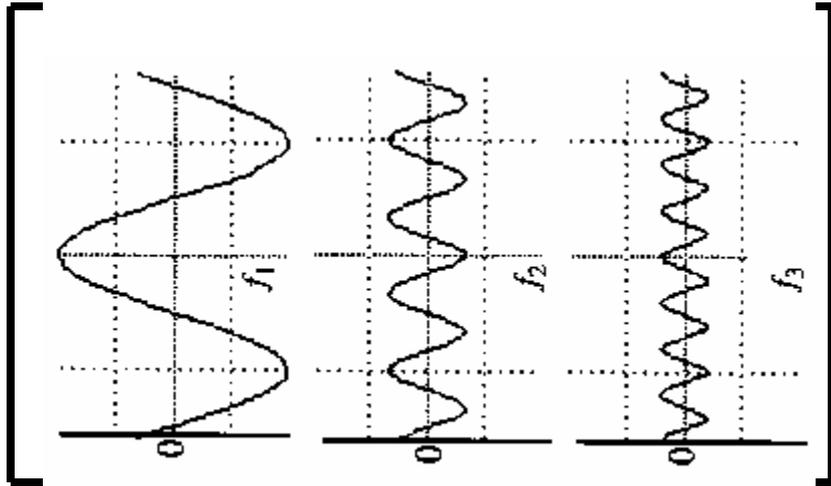


||

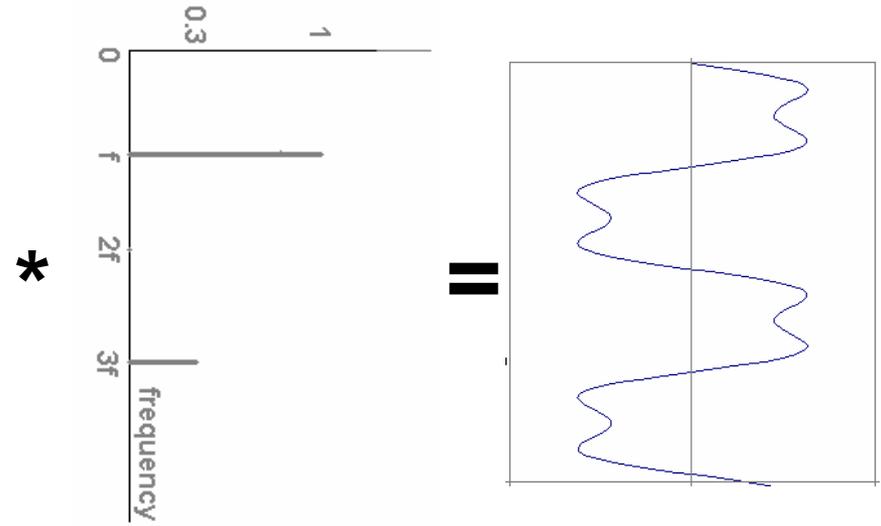


IFT: Just a change of basis

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



•
•
•



Finally: Scary Math

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

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

Finally: Scary Math

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

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

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

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

$$\begin{array}{l} \text{phase can be encoded} \\ \text{by sin/cos pair} \end{array} \rightarrow \begin{array}{l} P \cos(x) + Q \sin(x) = A \sin(x + \phi) \\ A = \pm \sqrt{P^2 + Q^2} \quad \phi = \tan^{-1}\left(\frac{P}{Q}\right) \end{array}$$

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

Extension to 2D

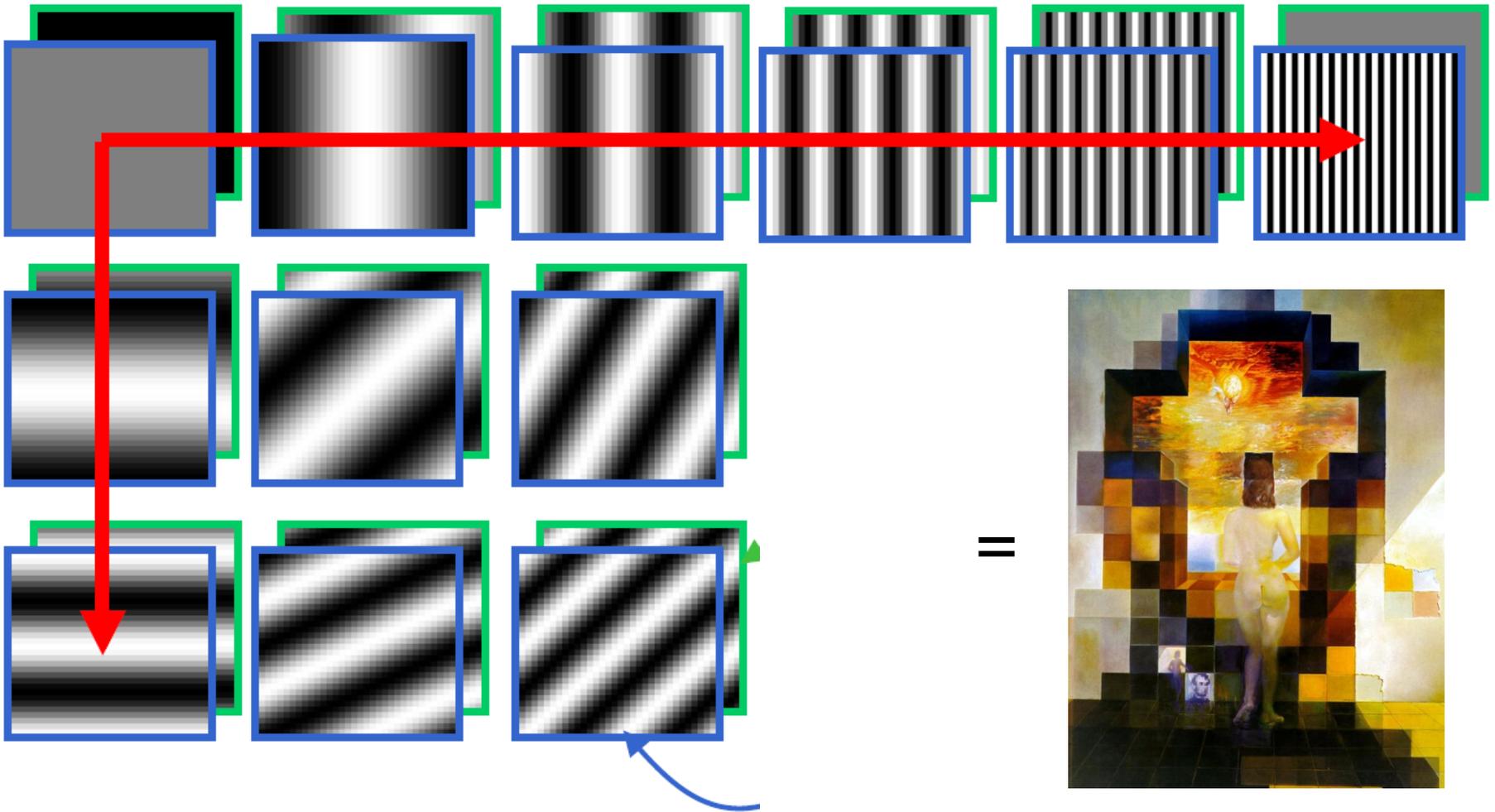
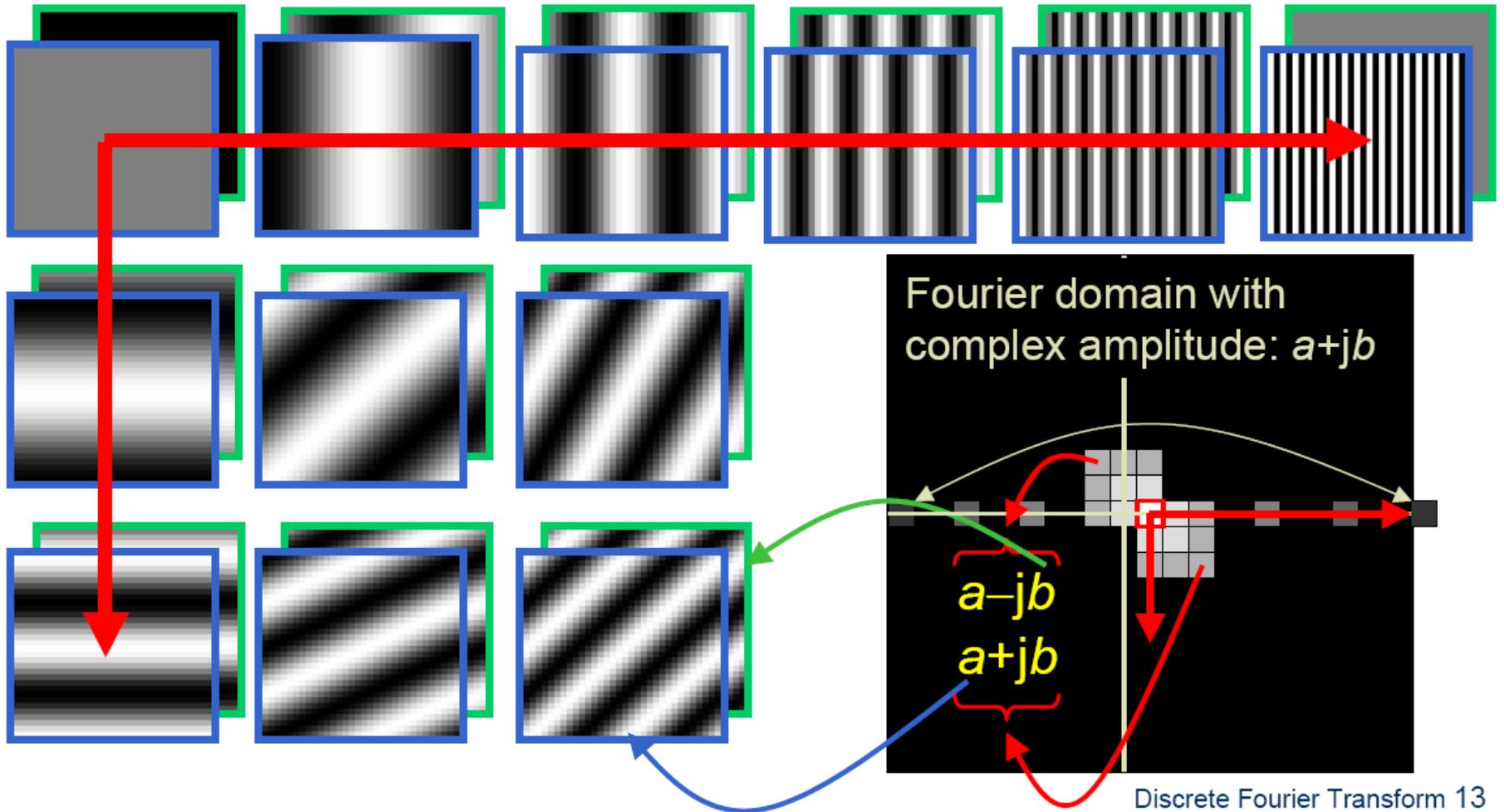


Image as a sum of basis images

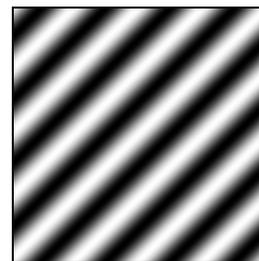
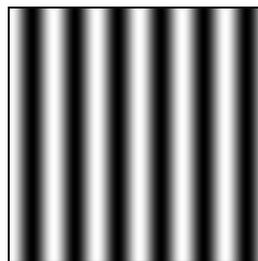
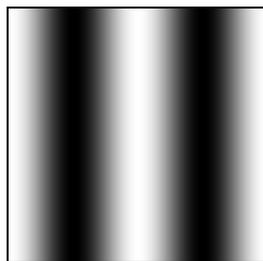
Extension to 2D



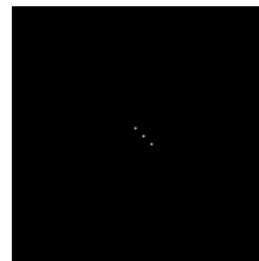
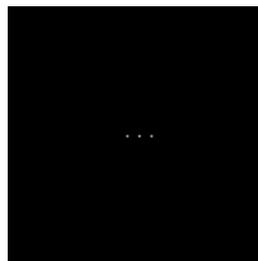
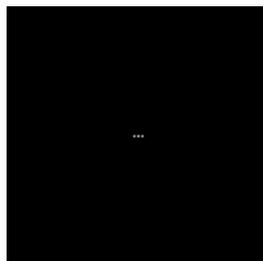
in Matlab, check out: `imagesc(log(abs(fftshift(fft2(im)))));`

Fourier analysis in images

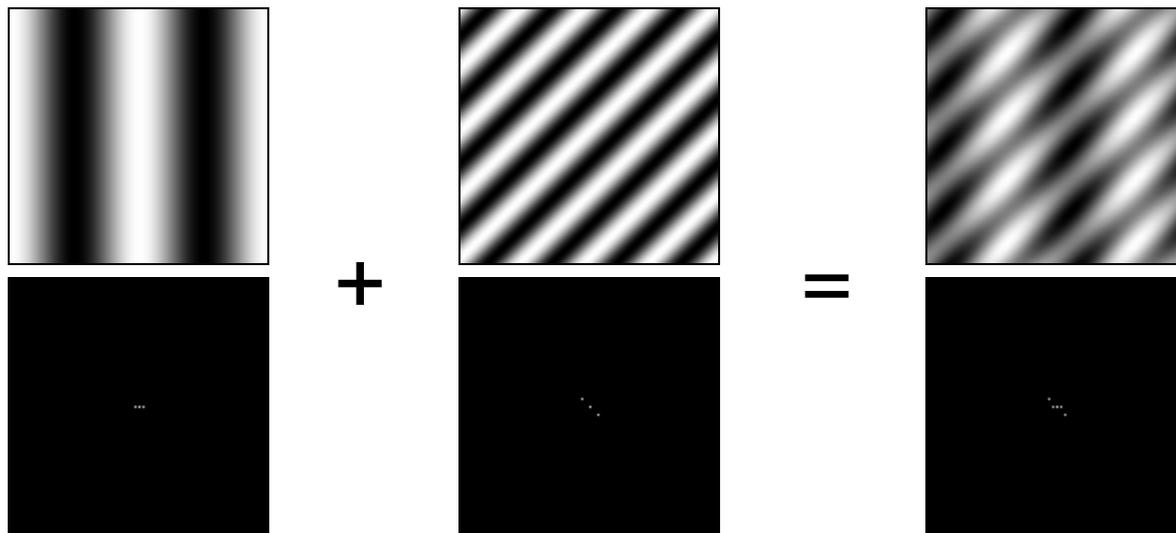
Intensity Image



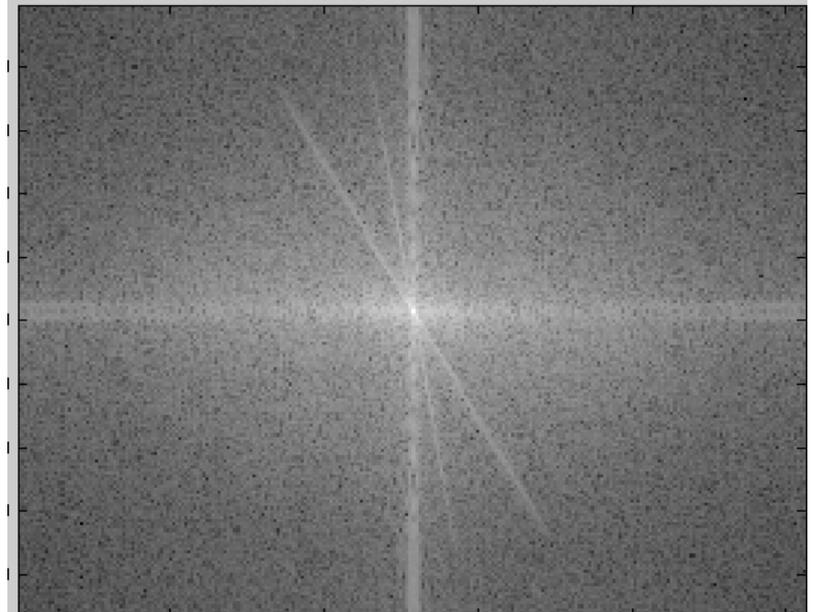
Fourier Image



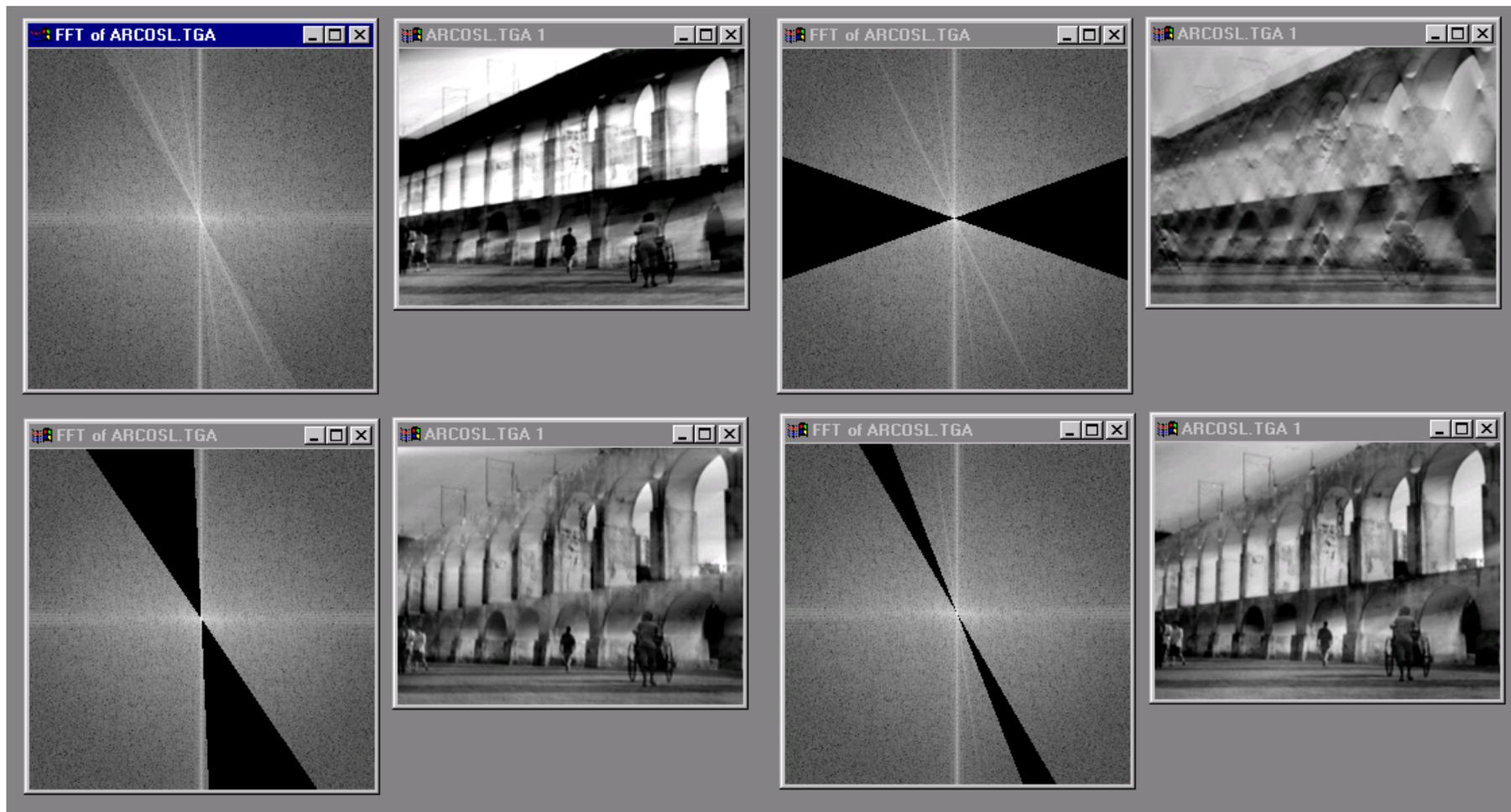
Signals can be composed



Man-made Scene

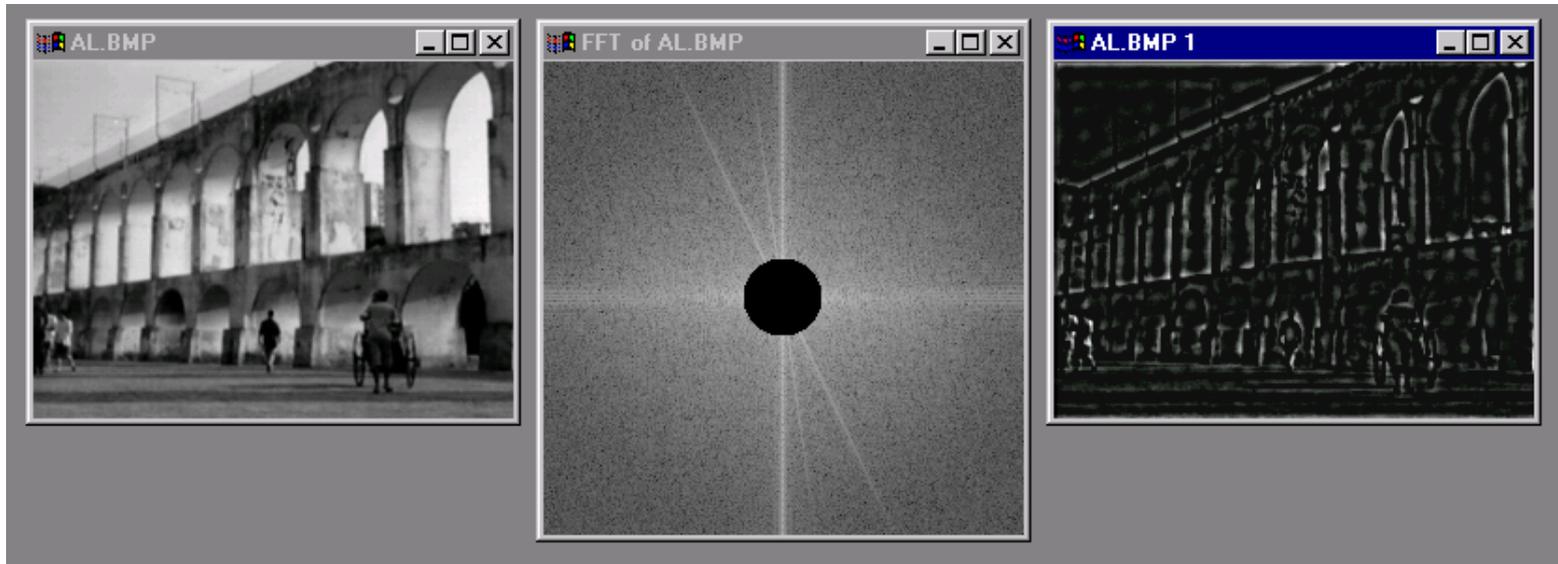


Can change spectrum, then reconstruct



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

$$F[g * h] = F[g]F[h]$$

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

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

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

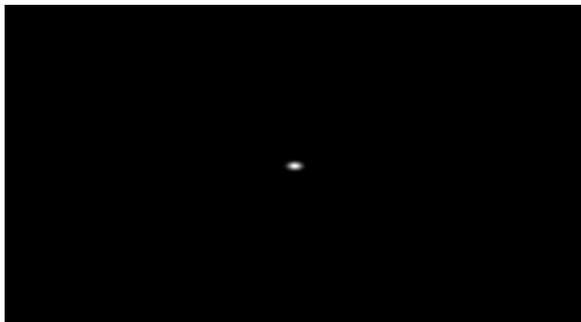
2D convolution theorem example

$f(x,y)$



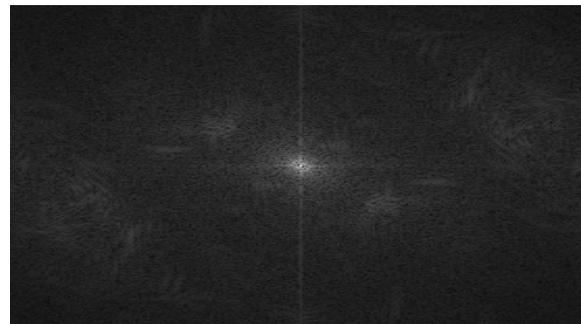
*

$h(x,y)$



⇓

$g(x,y)$



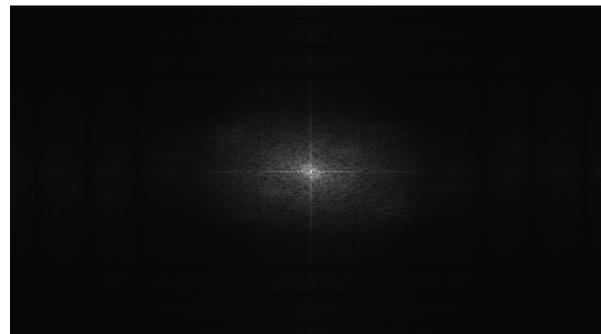
×

$|F(s_x, s_y)|$



⇓

$|H(s_x, s_y)|$

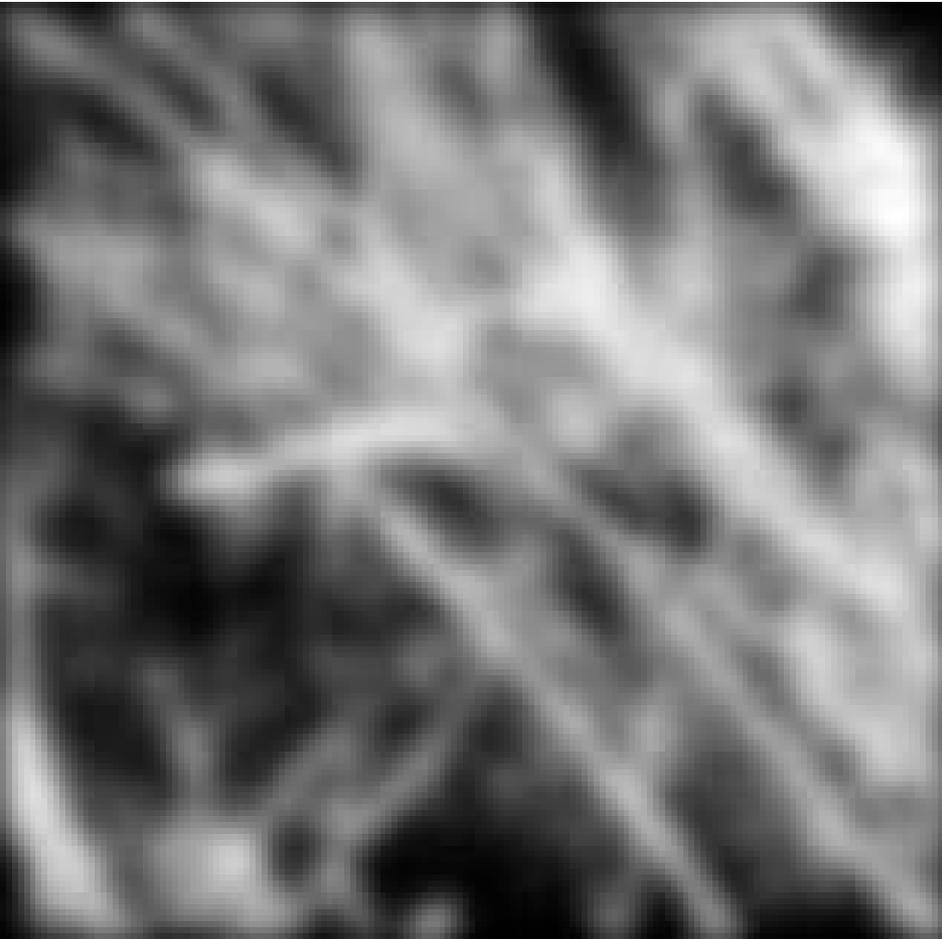


$|G(s_x, s_y)|$

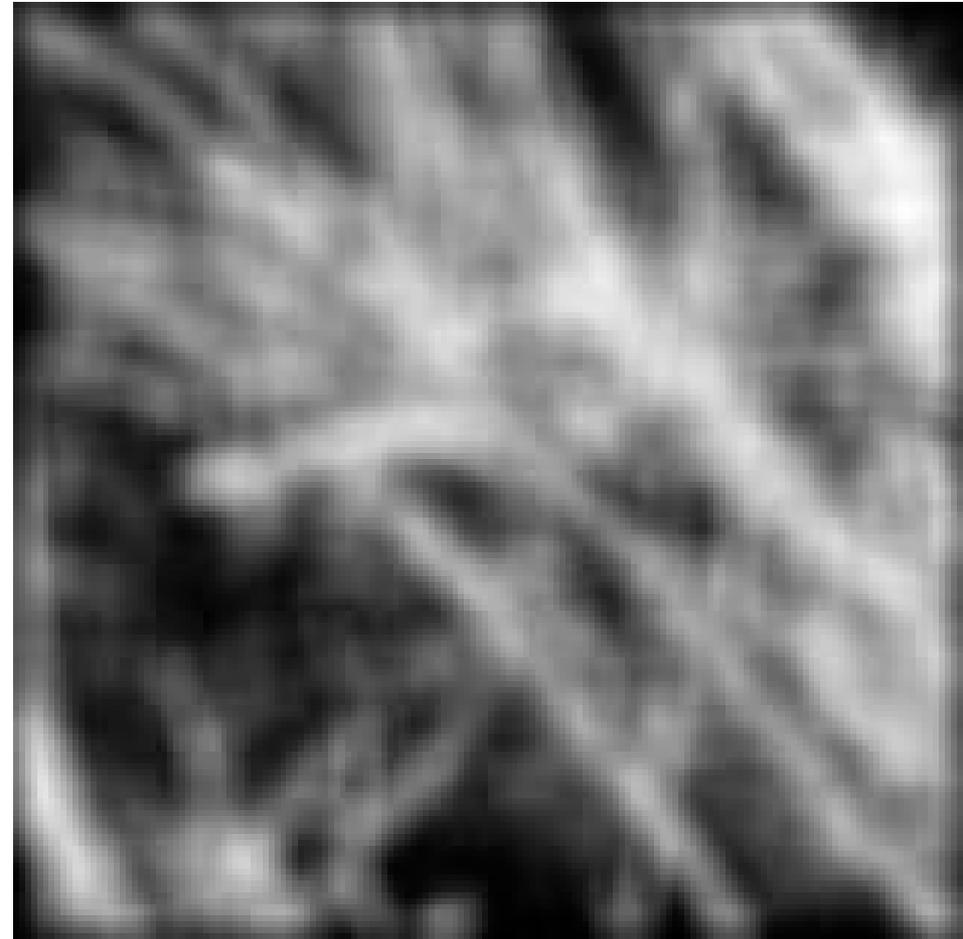
Filtering

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

Gaussian

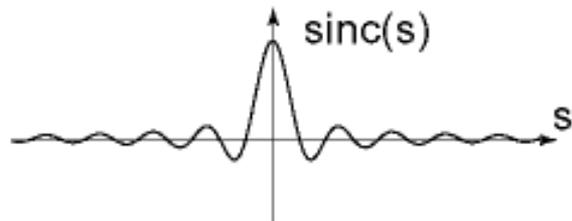
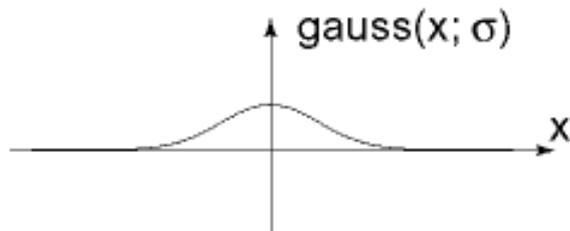
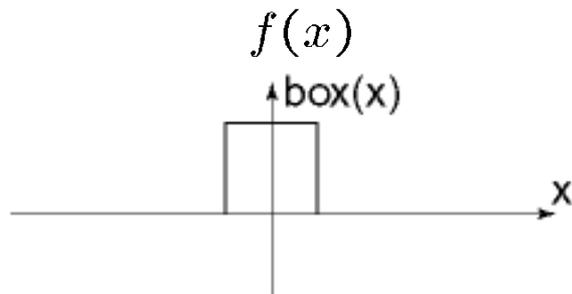


Box filter

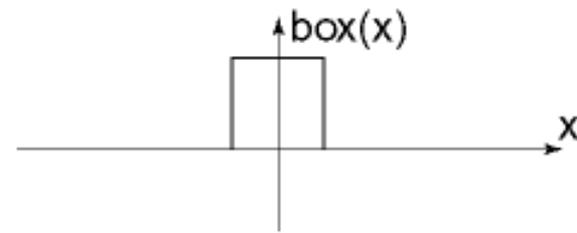
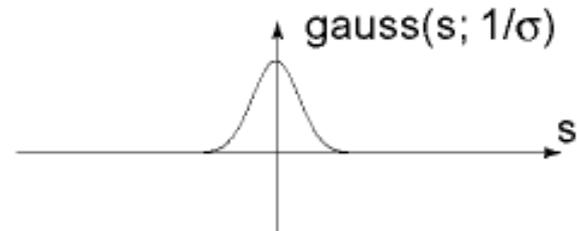
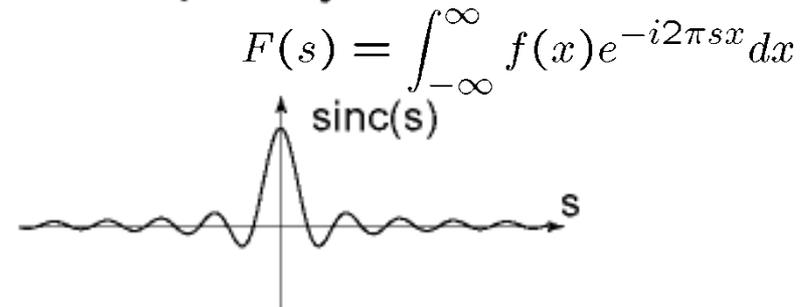


Fourier Transform pairs

Spatial domain



Frequency domain

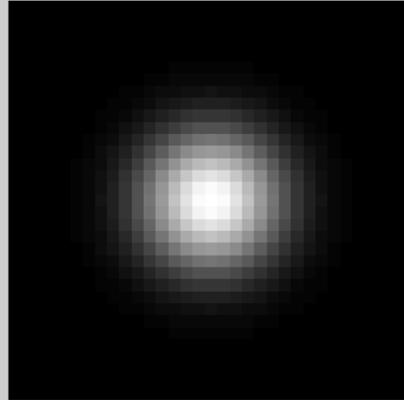


Gaussian

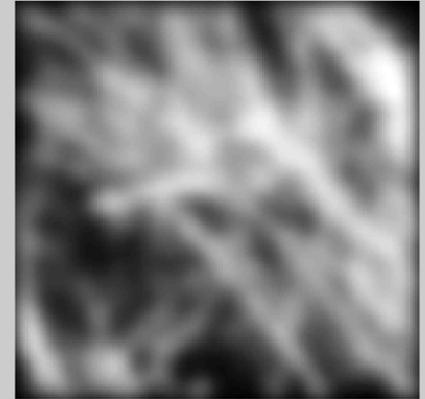
intensity image



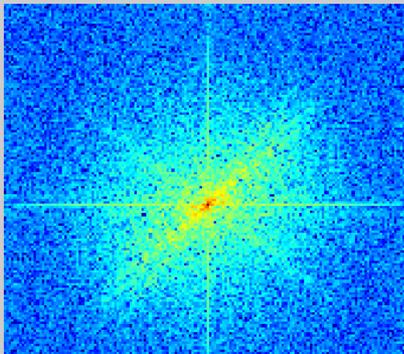
filter: gaussian



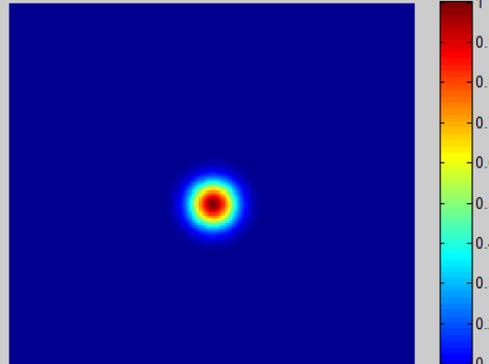
filtered image



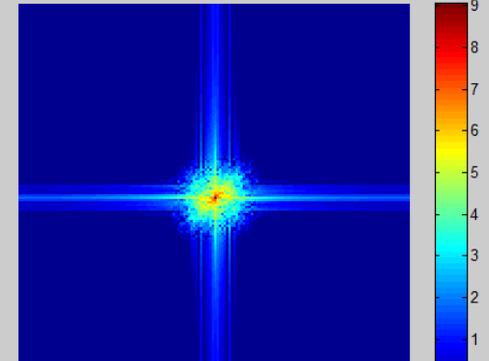
log fit magnitude of image



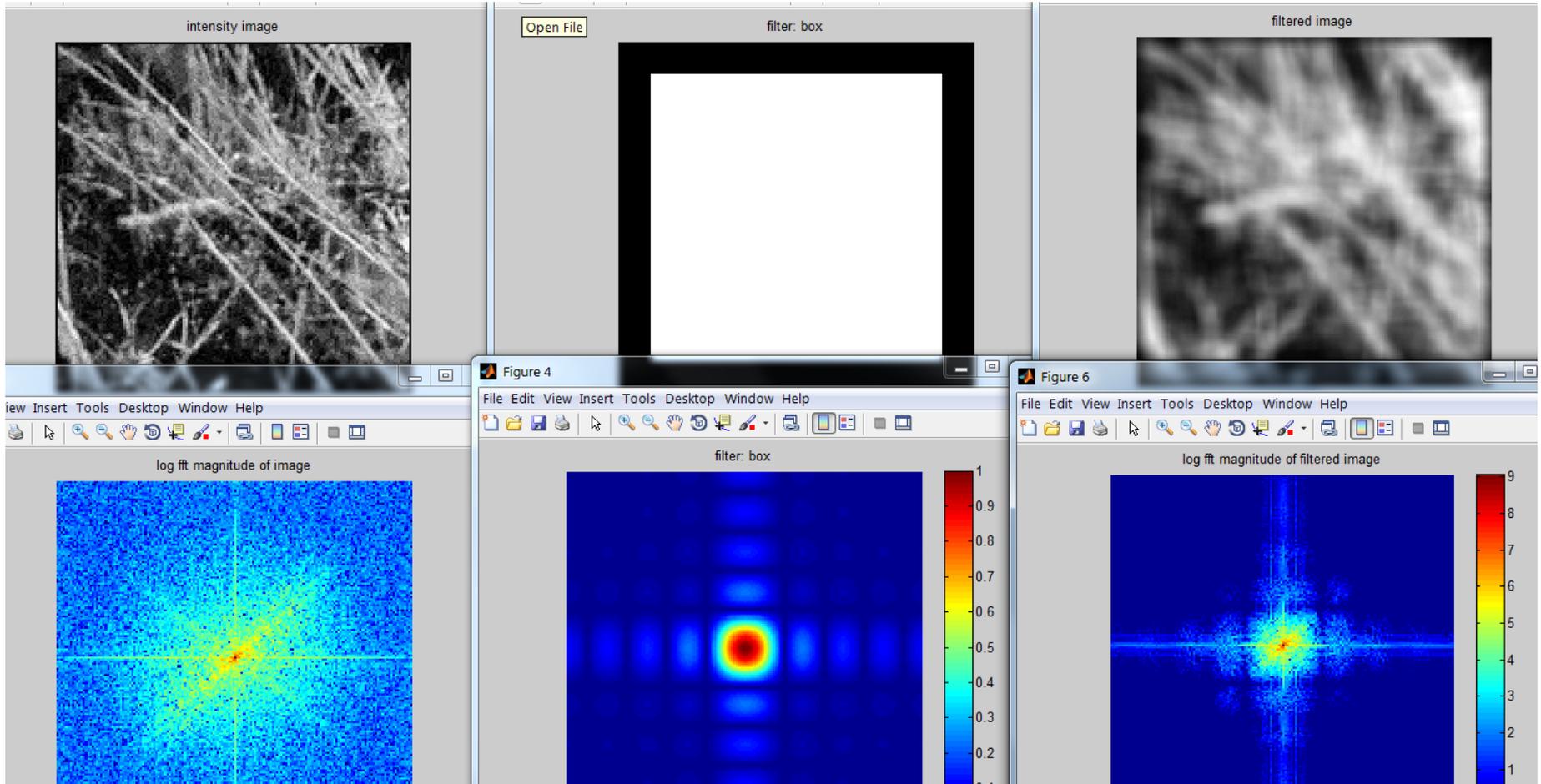
filter: gaussian



log fit magnitude of filtered image

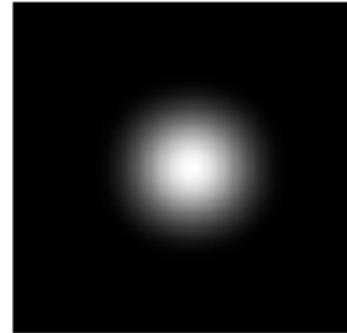
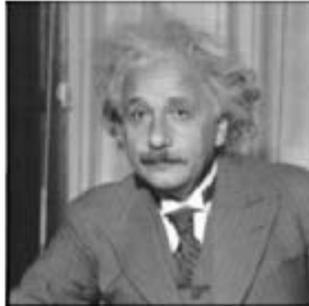
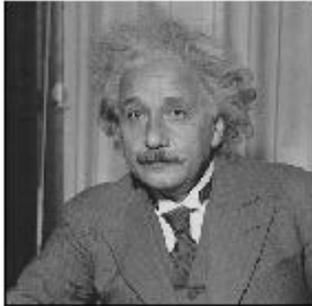


Box Filter

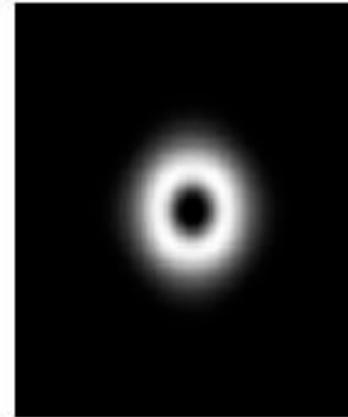
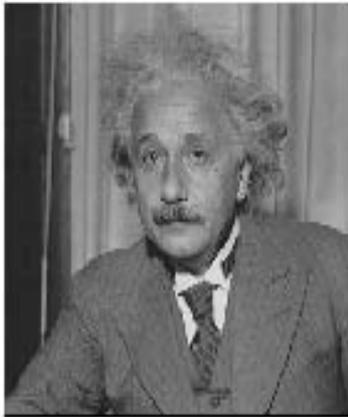


Low-pass, Band-pass, High-pass filters

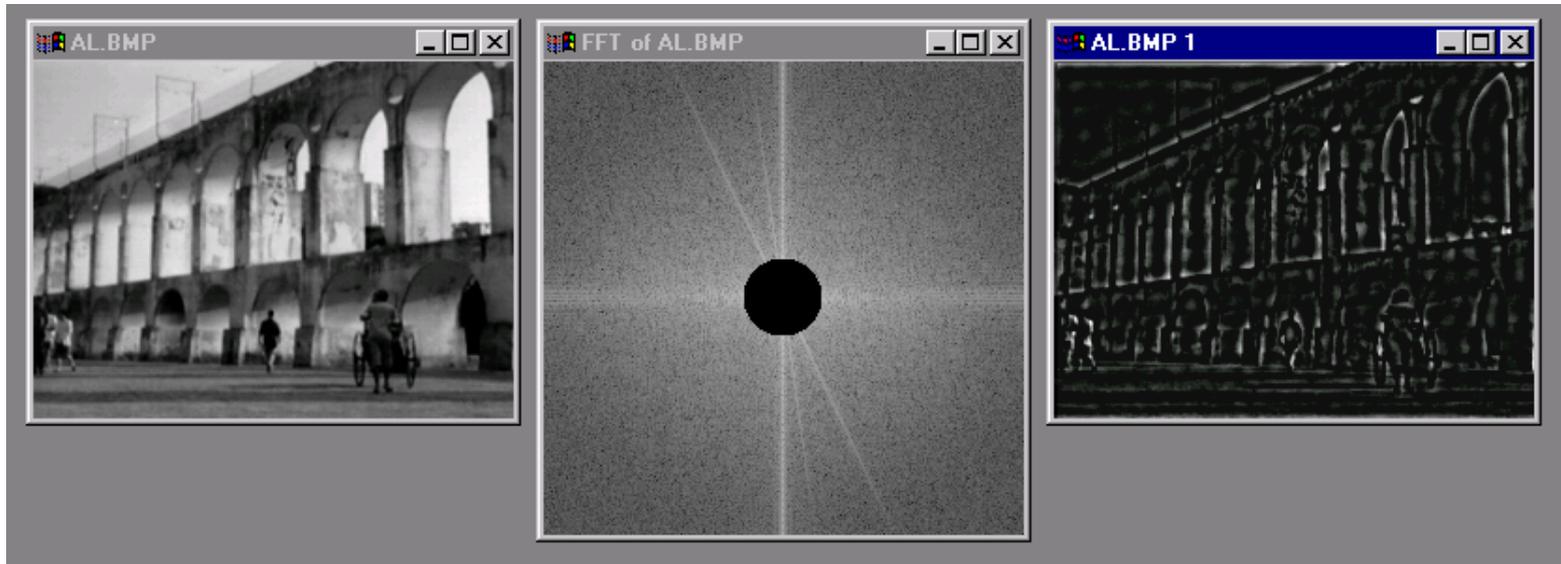
low-pass:



High-pass / band-pass:



Edges in images



What does blurring take away?



original

What does blurring take away?



smoothed (5x5 Gaussian)

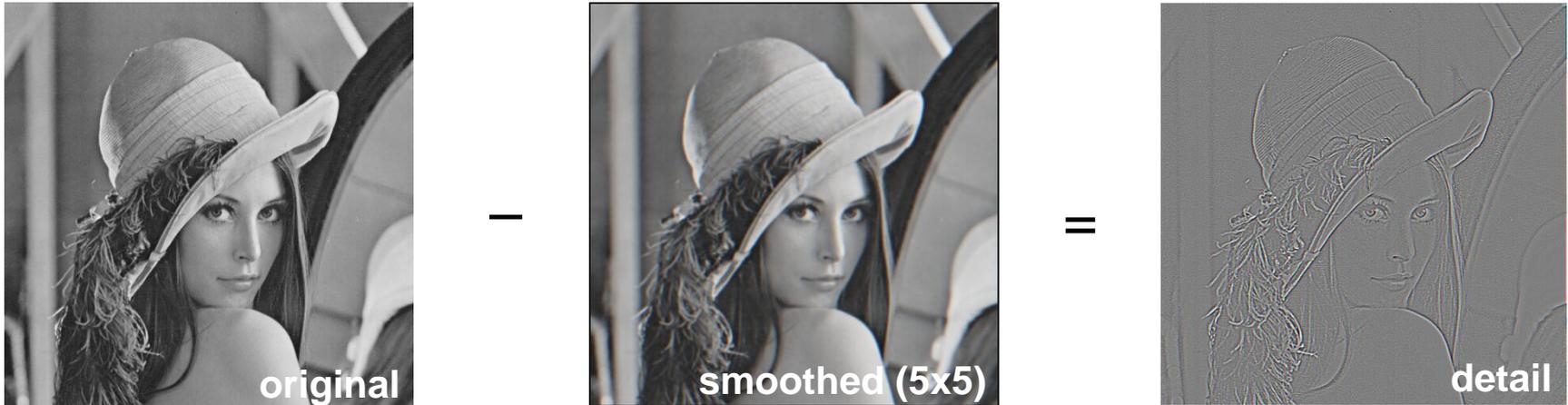
High-Pass filter



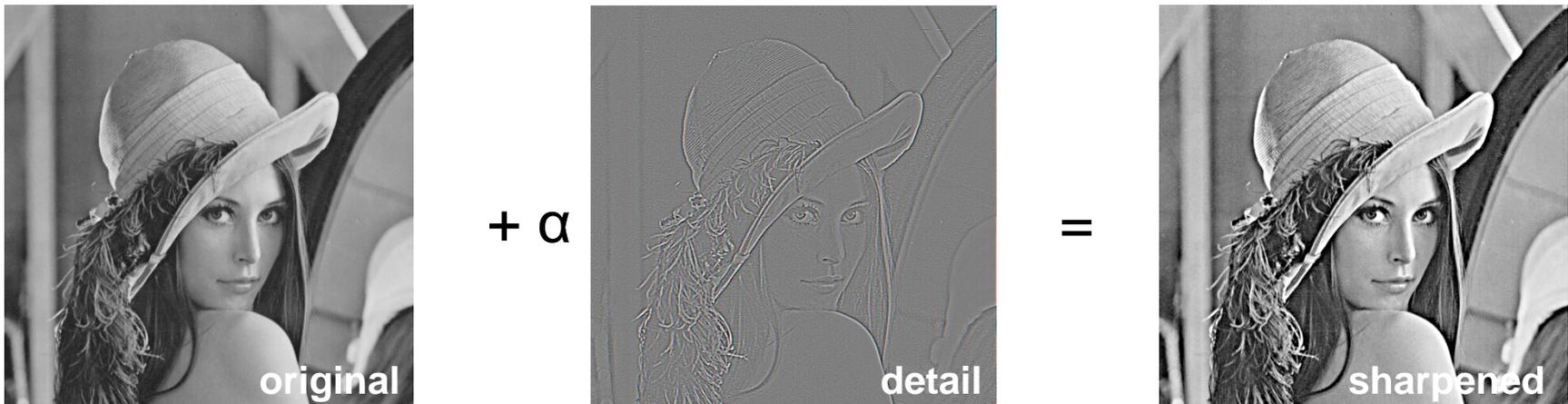
smoothed – original

Image “Sharpening”

What does blurring take away?



Let's add it back:



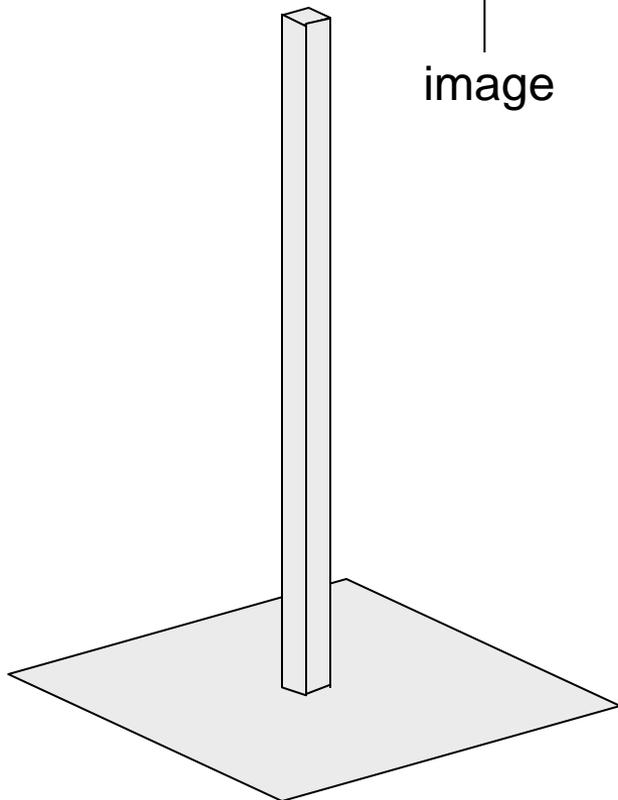
Unsharp mask filter

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

↑
image

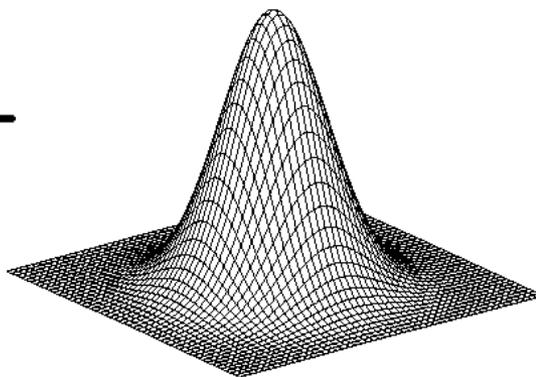
↑
blurred
image

↑
unit impulse
(identity)



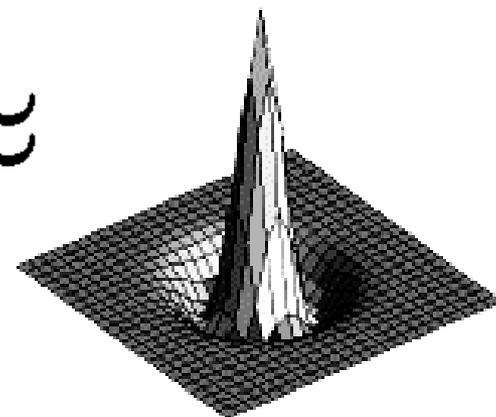
unit impulse

—



Gaussian

≈

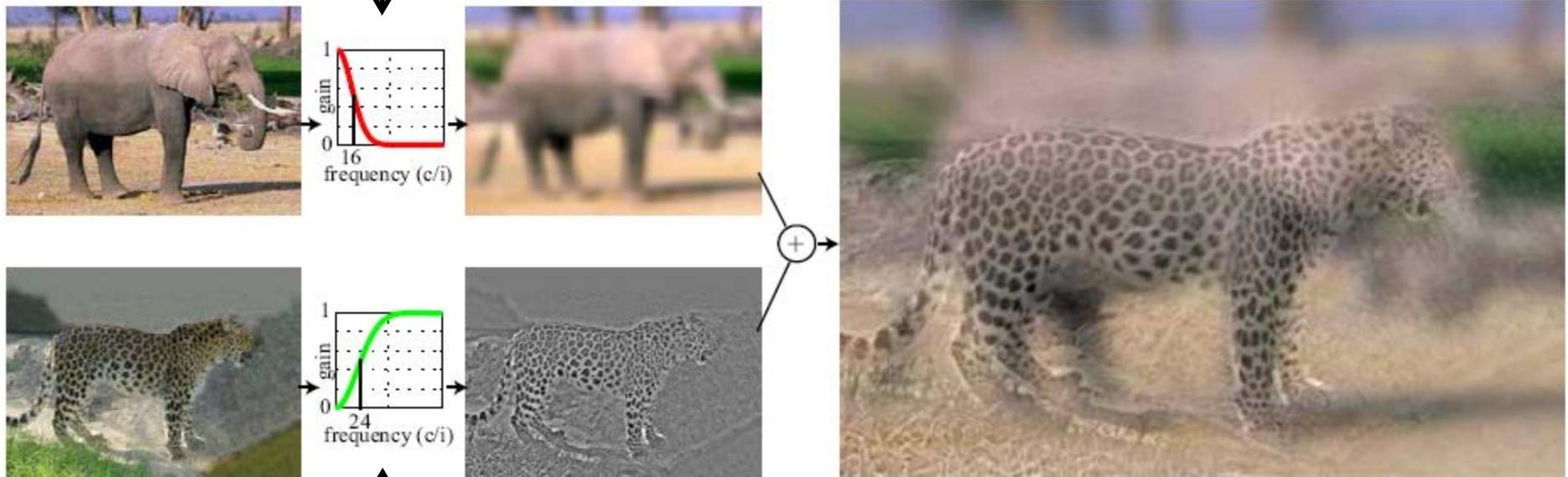


Laplacian of Gaussian

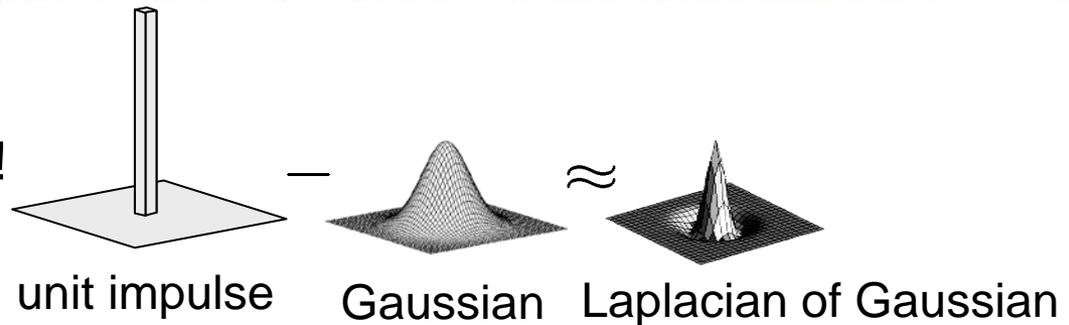
Hybrid Images

A. Oliva, A. Torralba, P.G. Schyns,
["Hybrid Images,"](#) SIGGRAPH 2006

Gaussian Filter!



Laplacian Filter!





Salvador Dalí

*"Gala Contemplating the Mediterranean Sea,
which at 30 meters becomes the portrait
of Abraham Lincoln", 1976*

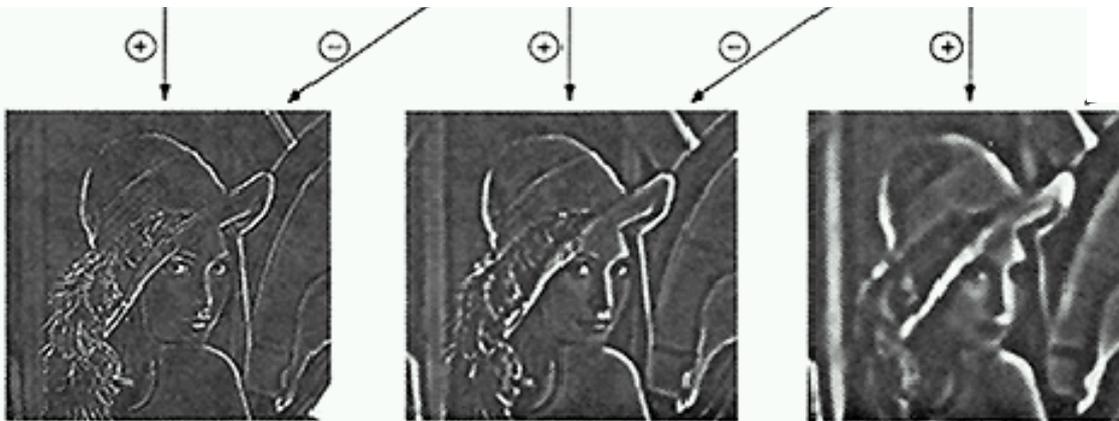
Band-pass filtering

Gaussian Pyramid (low-pass images)



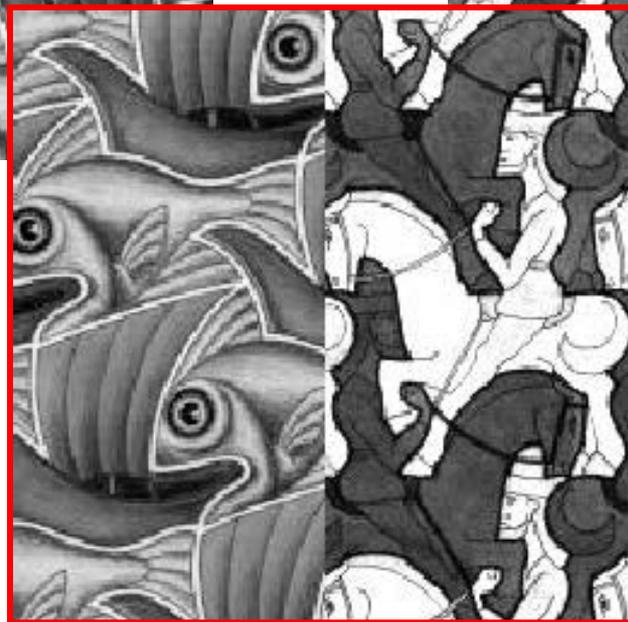
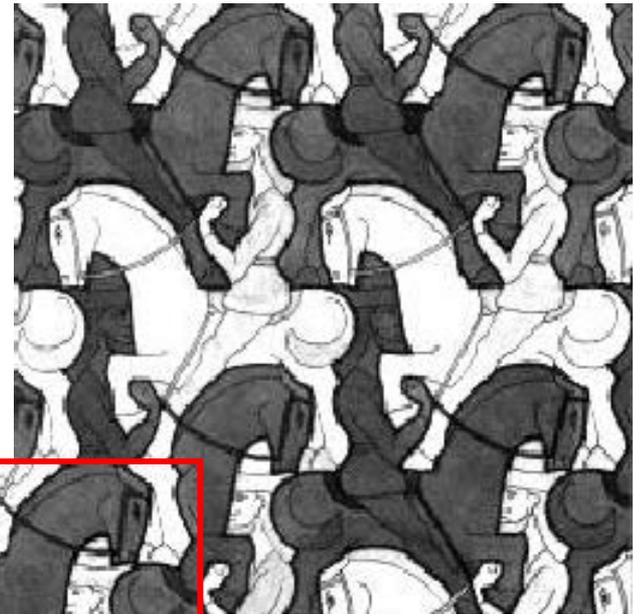
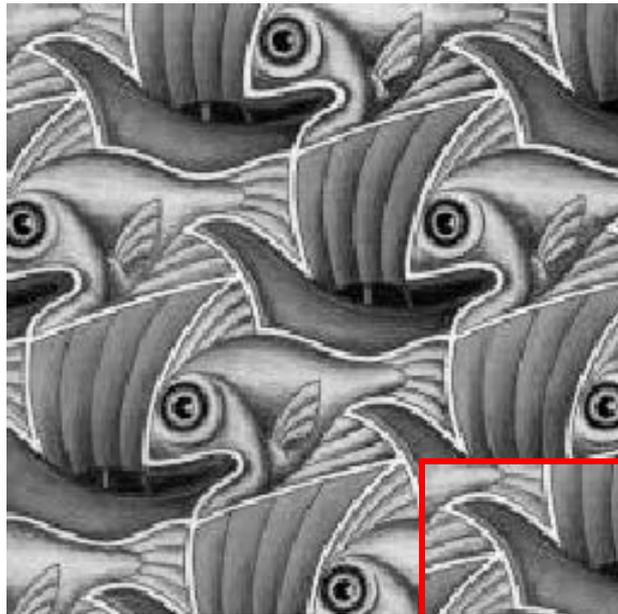
Laplacian Pyramid

Original
image

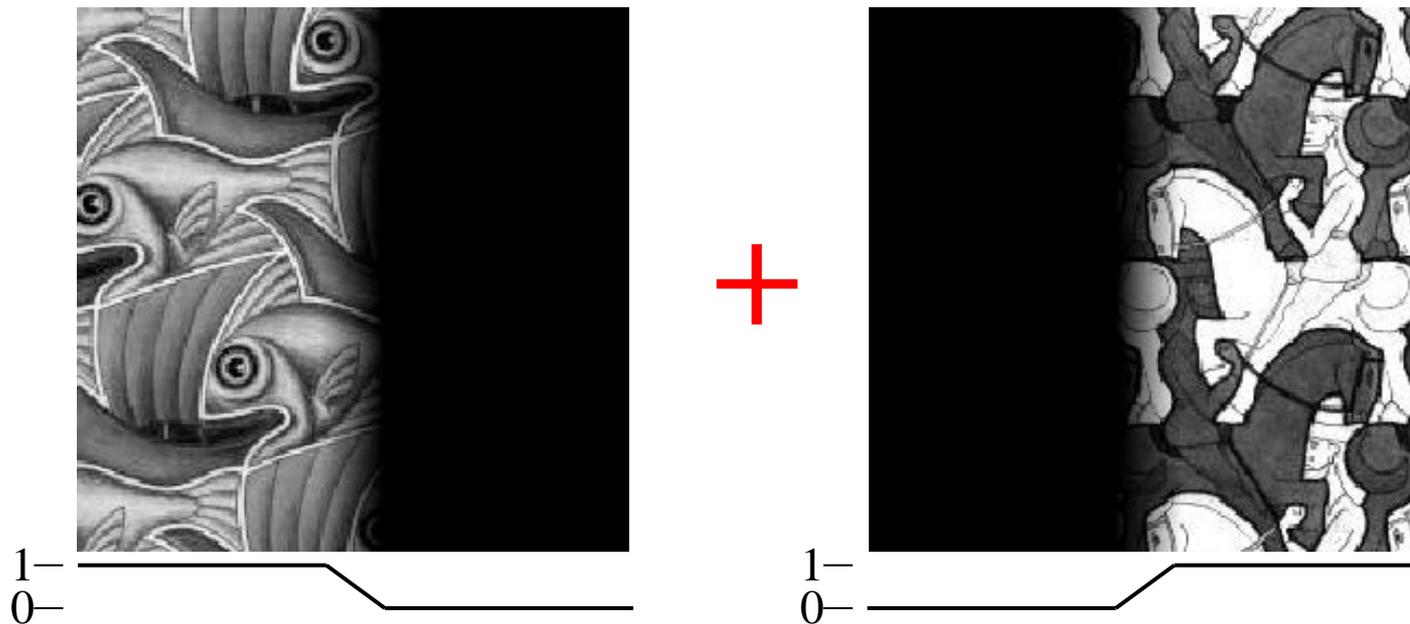


How can we reconstruct (collapse) this pyramid into the original image?

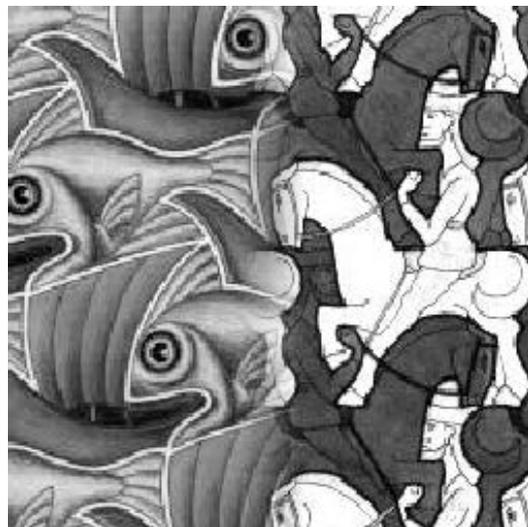
Blending



Alpha Blending / Feathering

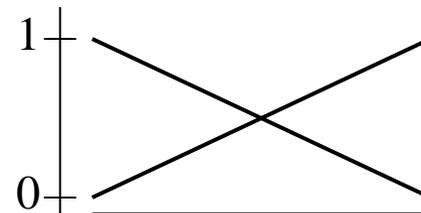
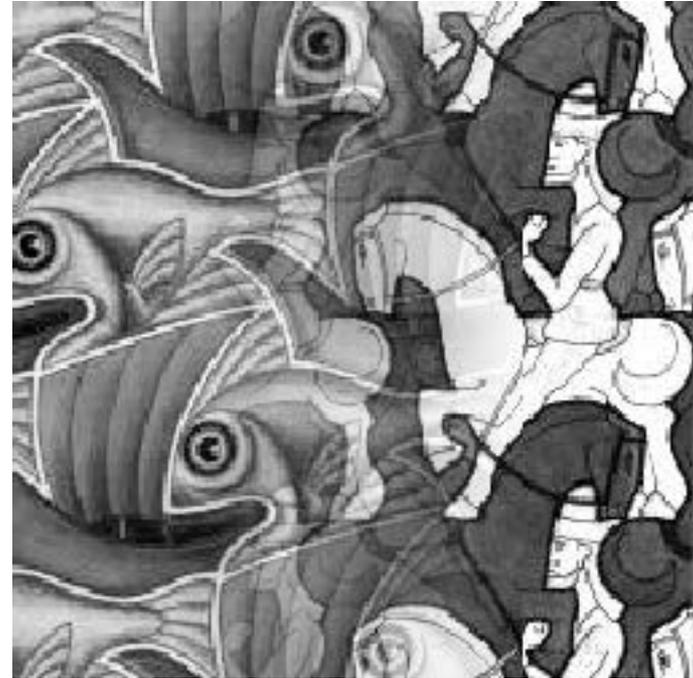
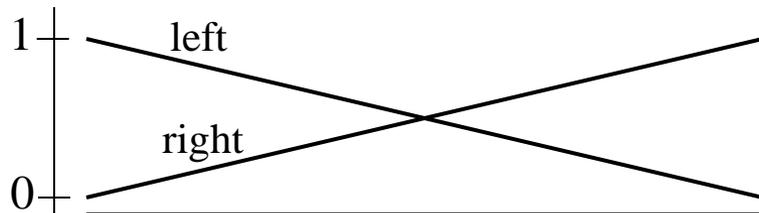
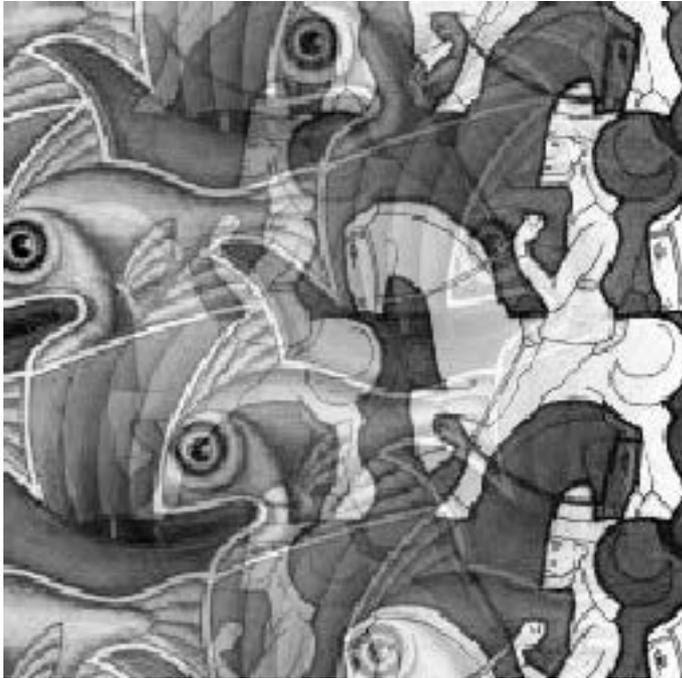


=

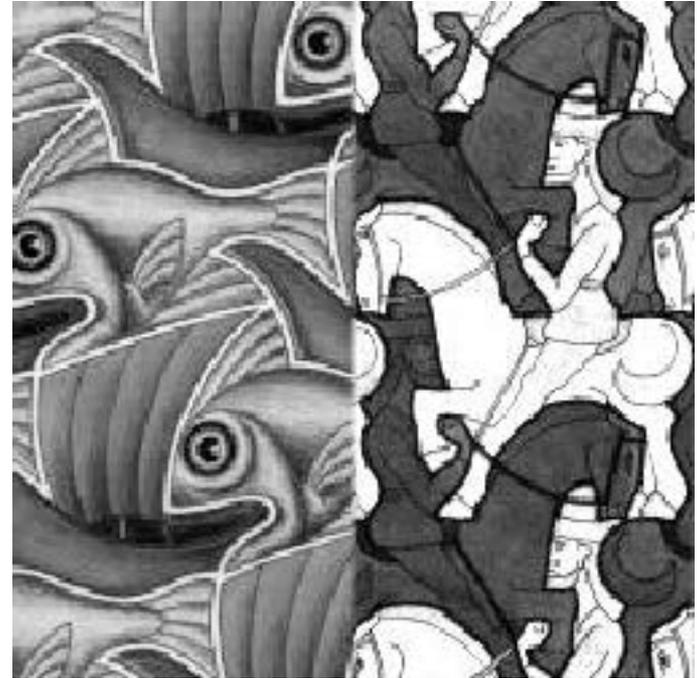
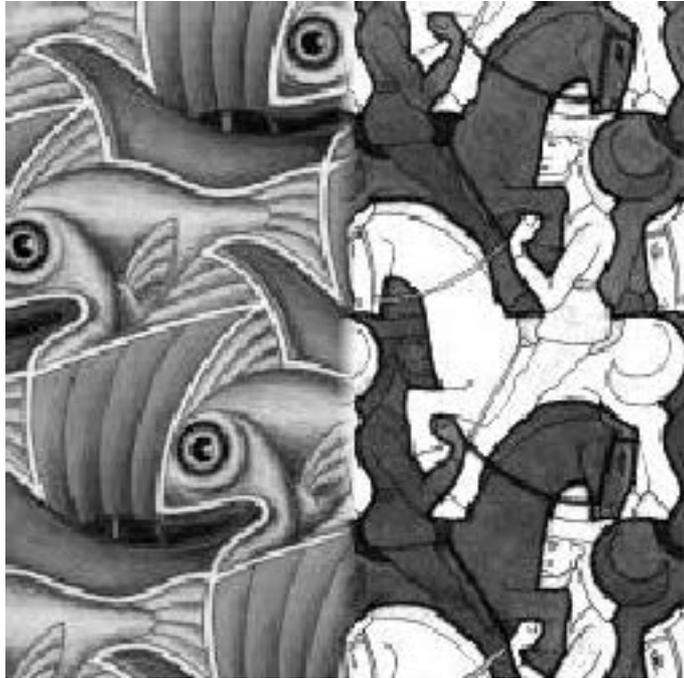


$$I_{\text{blend}} = \alpha I_{\text{left}} + (1-\alpha) I_{\text{right}}$$

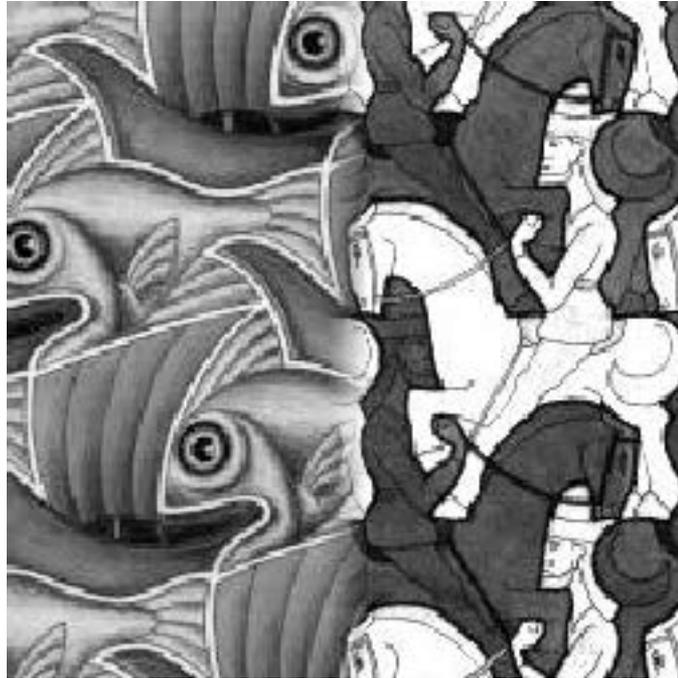
Affect of Window Size



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

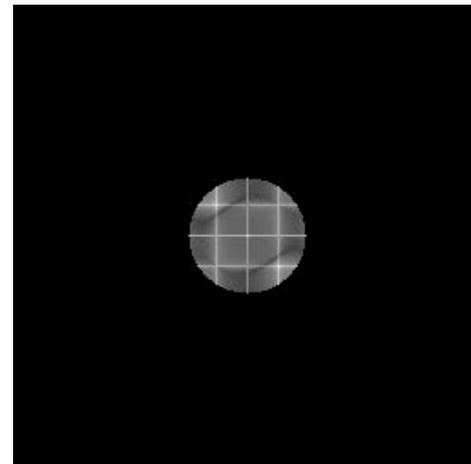
- window $\leq 2 \times$ size of smallest prominent feature

Natural to cast this in the *Fourier domain*

- largest frequency $\leq 2 \times$ size of smallest frequency
- image frequency content should occupy one “octave” (power of two)



FFT
→



What if the Frequency Spread is Wide



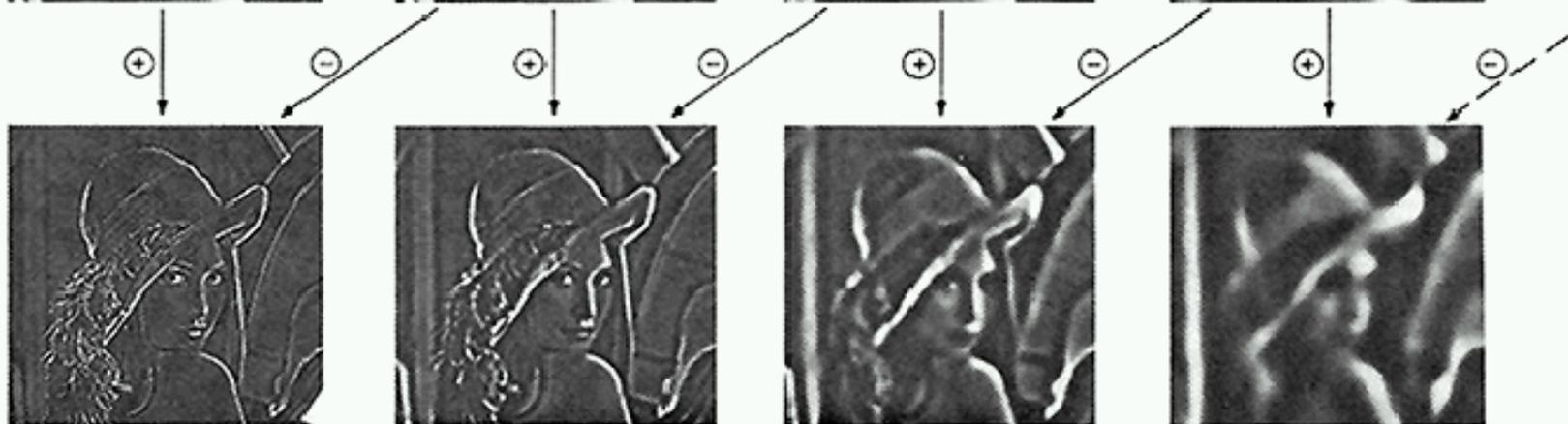
Idea (Burt and Adelson)

- Compute $F_{\text{left}} = \text{FFT}(I_{\text{left}})$, $F_{\text{right}} = \text{FFT}(I_{\text{right}})$
- Decompose Fourier image into octaves (bands)
 - $F_{\text{left}} = F_{\text{left}}^1 + F_{\text{left}}^2 + \dots$
- 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*

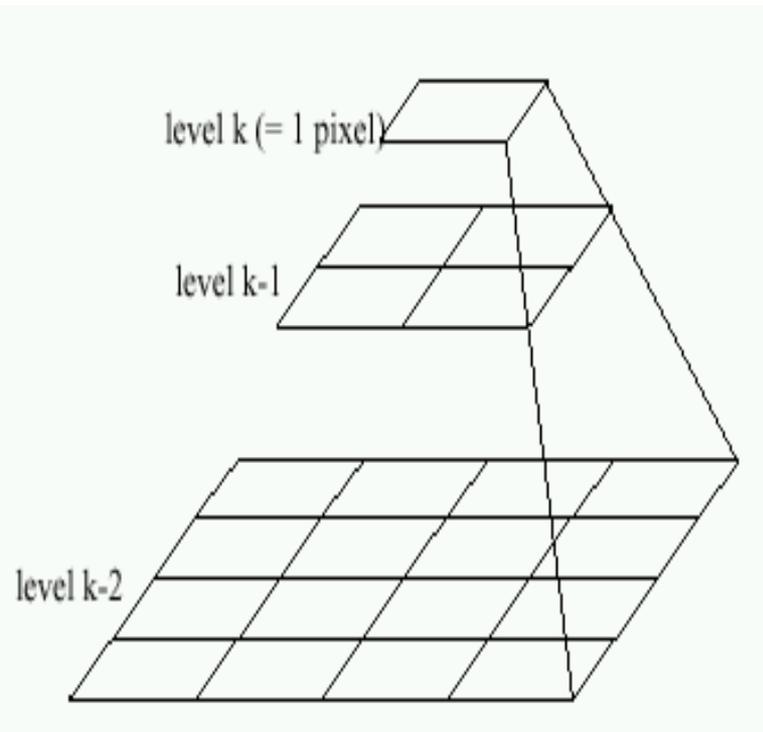
Octaves in the Spatial Domain

Lowpass Images

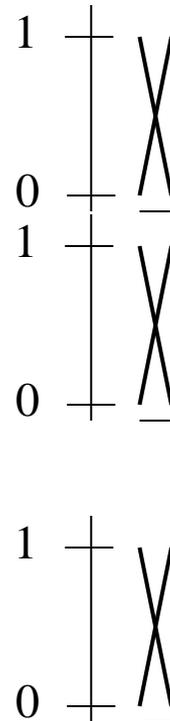


Bandpass Images

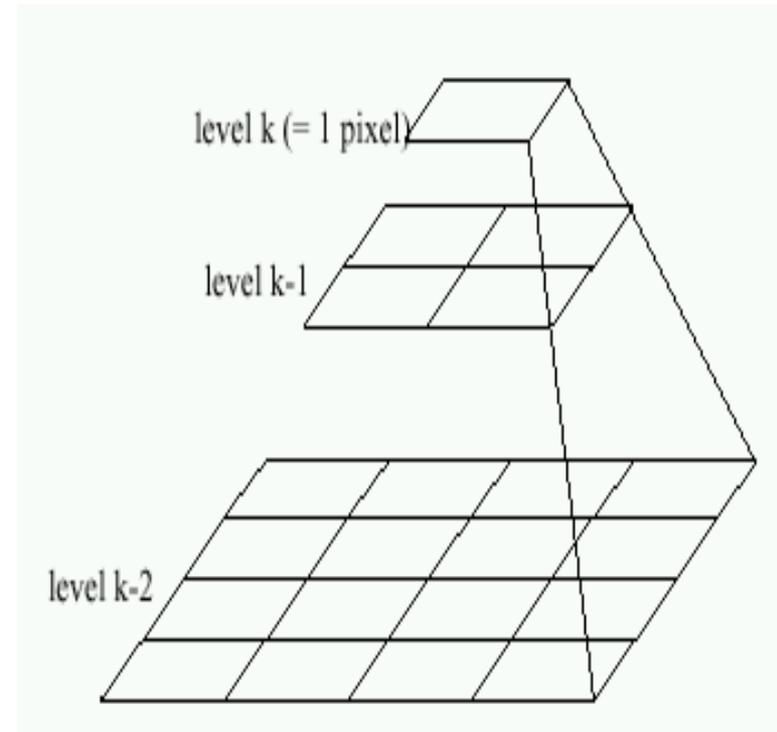
Pyramid Blending



Left pyramid

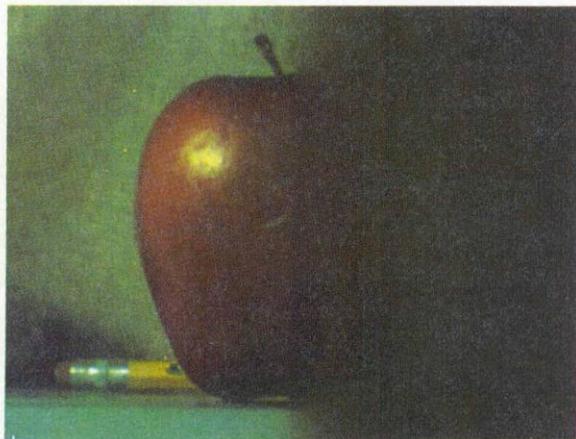
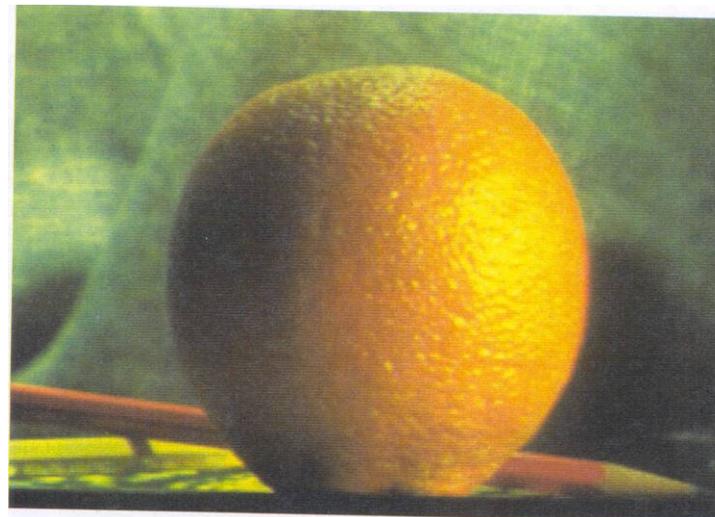


blend

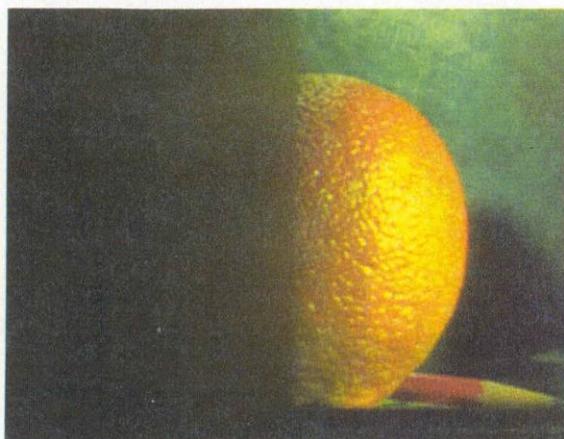


Right pyramid

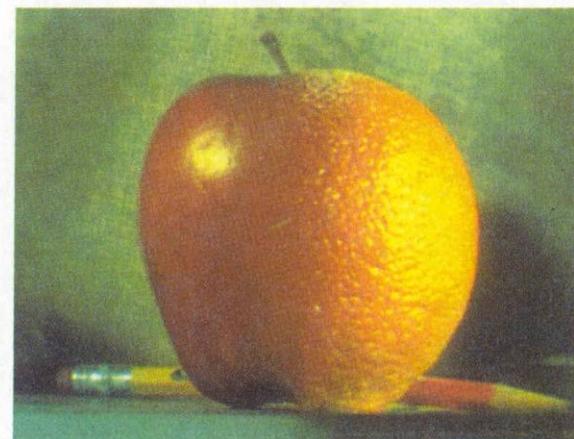
Pyramid Blending



(d)

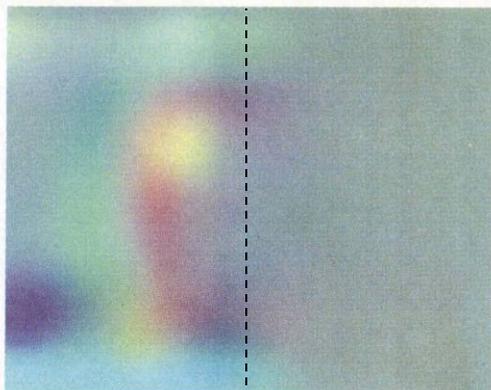


(h)

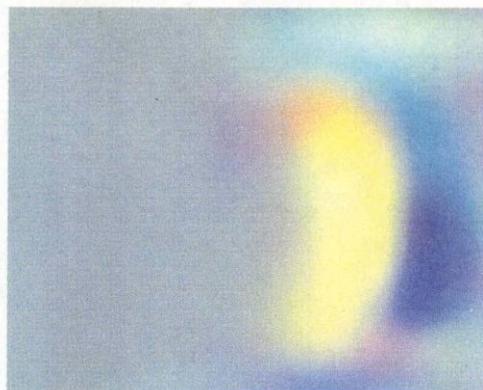


(l)

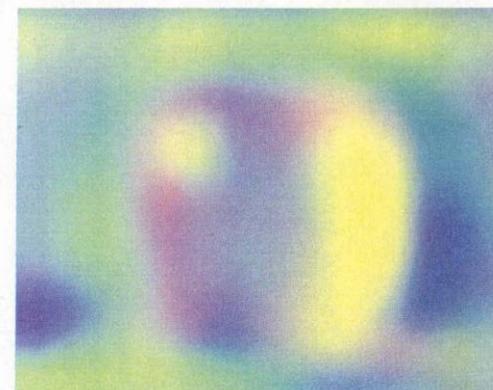
laplacian
level
4



(c)

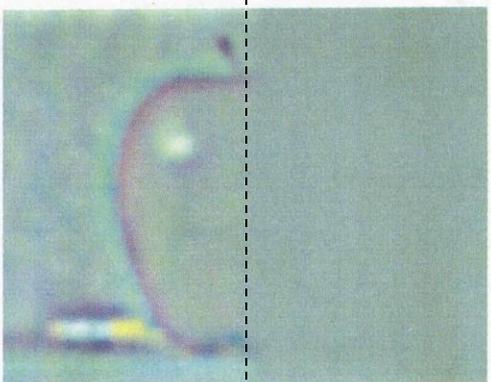


(g)

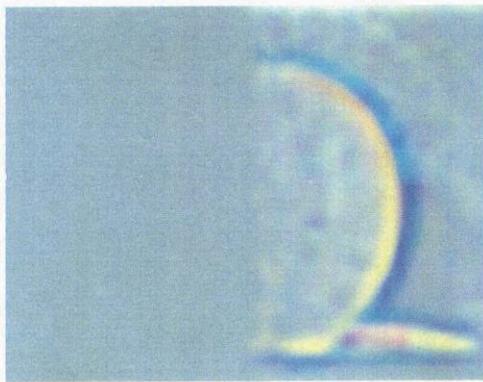


(k)

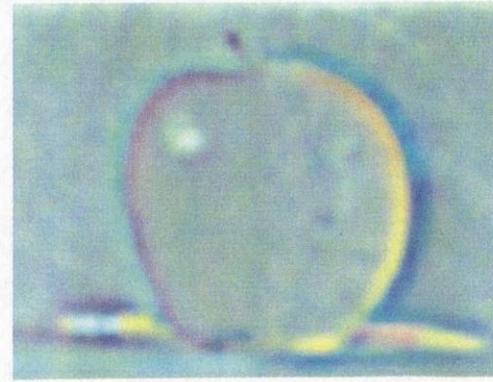
laplacian
level
2



(b)

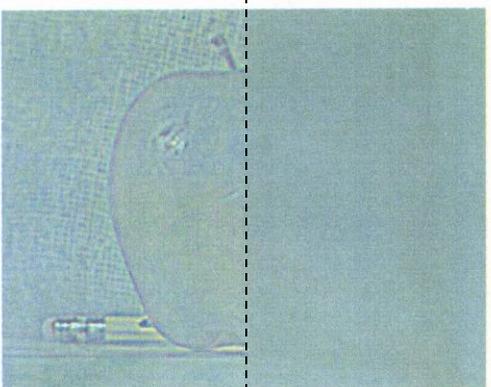


(f)

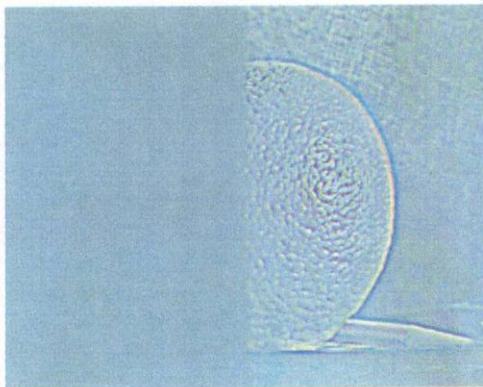


(j)

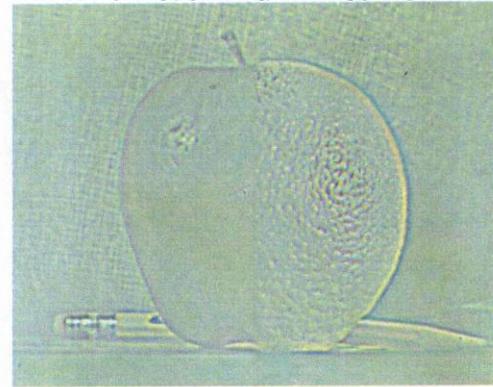
laplacian
level
0



(a)



(e)



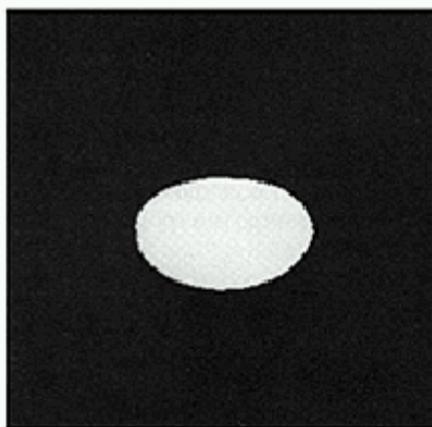
(i)

left pyramid

right pyramid

blended pyramid

Blending Regions

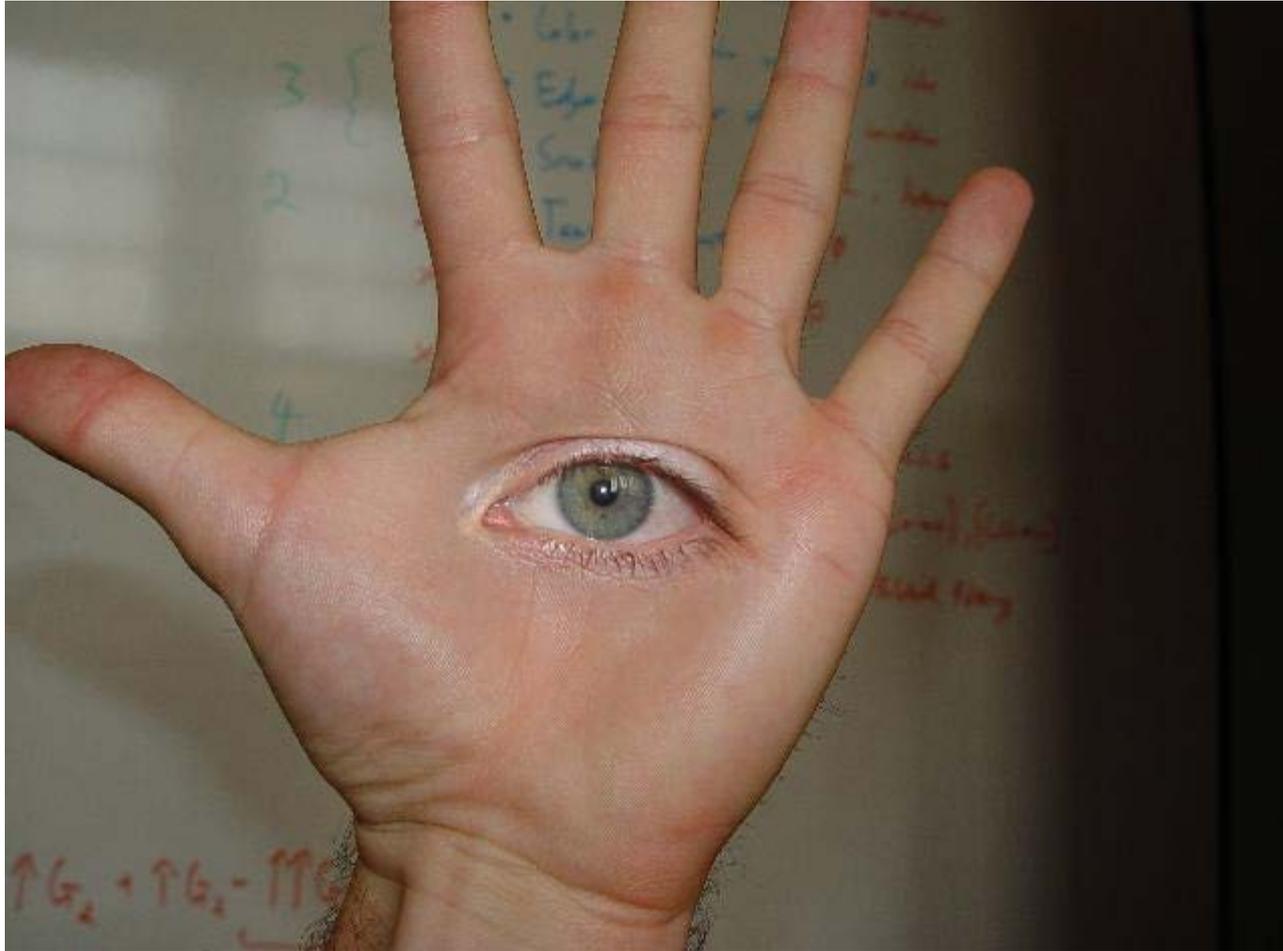


Laplacian Pyramid: Blending

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:
 - $LS(i,j) = GR(l,j) * LA(i,j) + (1 - GR(l,j)) * LB(i,j)$
4. Collapse the LS pyramid to get the final blended image

Horror Photo



© david dmartin (Boston College)

Results from this class (fall 2005)



© Chris Cameron

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 “Laplacian Stack” Blending



Low frequency ($\lambda > 2$ pixels)



High frequency ($\lambda < 2$ pixels)

Linear Blending

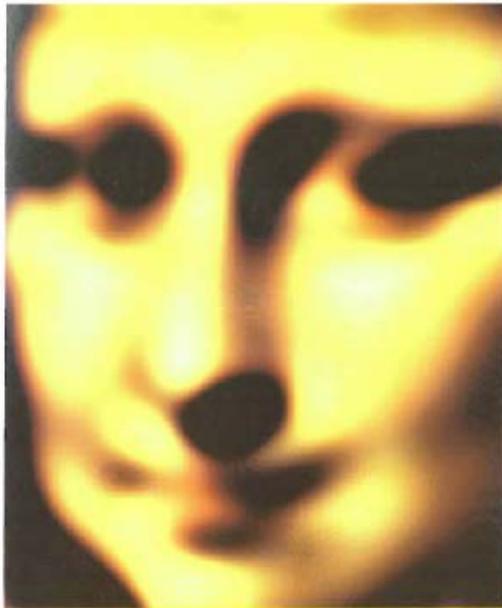


2-band Blending

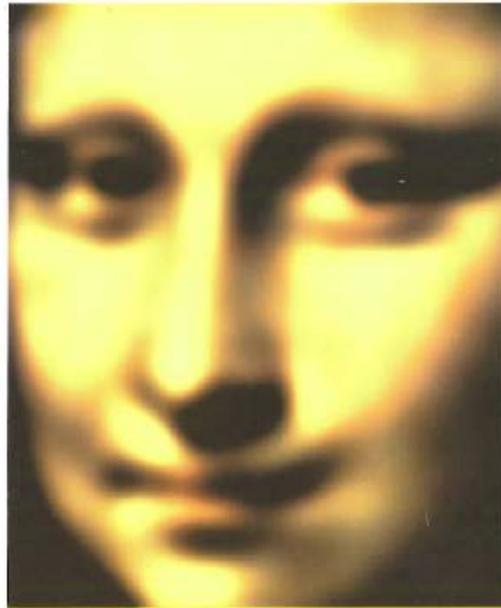


Da Vinci and Peripheral Vision





coarse components
(peripheral vision)



medium components
(near peripheral vision)



fine details
(central vision)

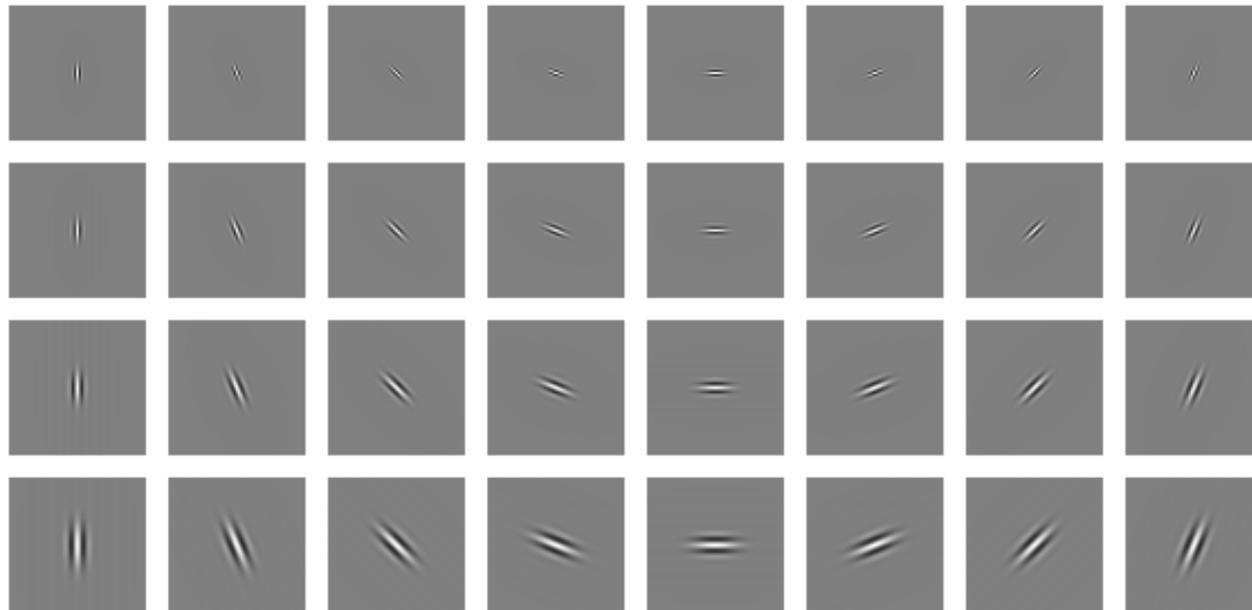
Leonardo playing with peripheral vision

Clues from Human Perception

Early processing in humans filters for various orientations and scales of frequency

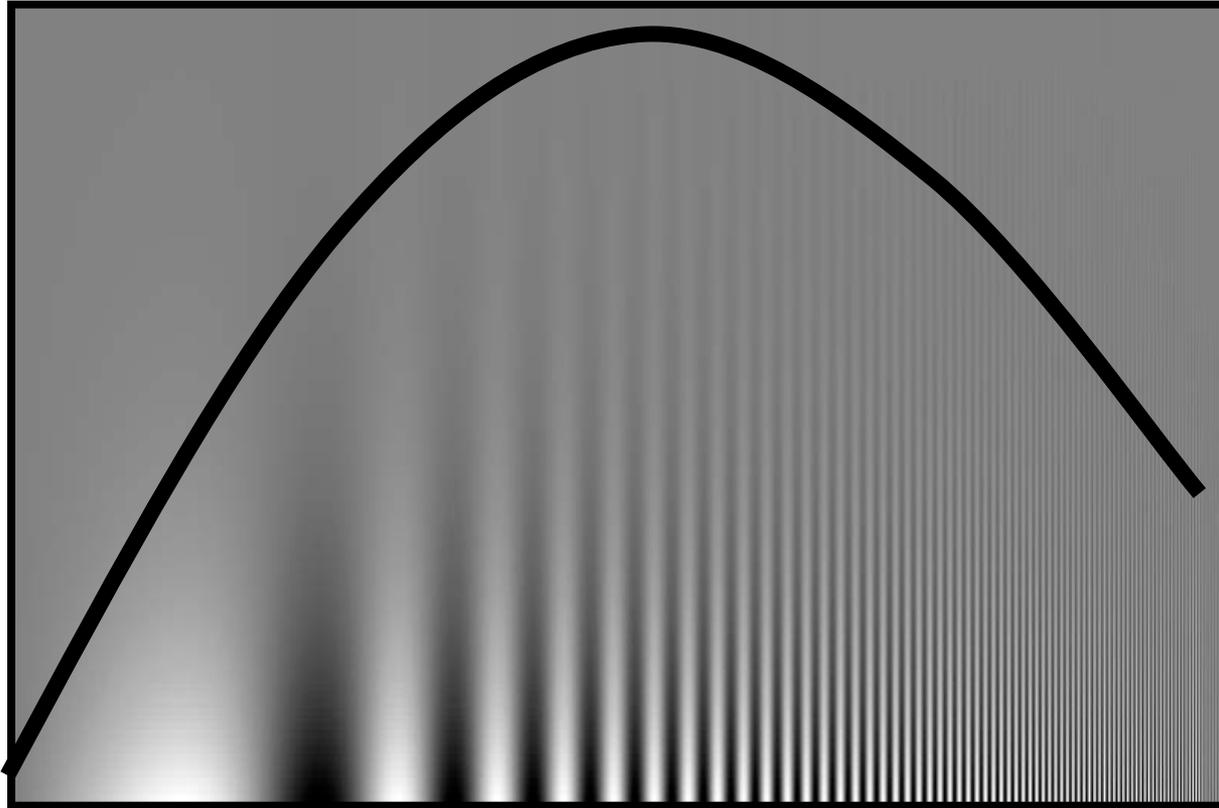
Perceptual cues in the mid frequencies dominate perception

When we see an image from far away, we are effectively subsampling it



Early Visual Processing: Multi-scale edge and blob filters

Frequency Domain and Perception



Campbell-Robson contrast sensitivity curve

Freq. Perception Depends on Color



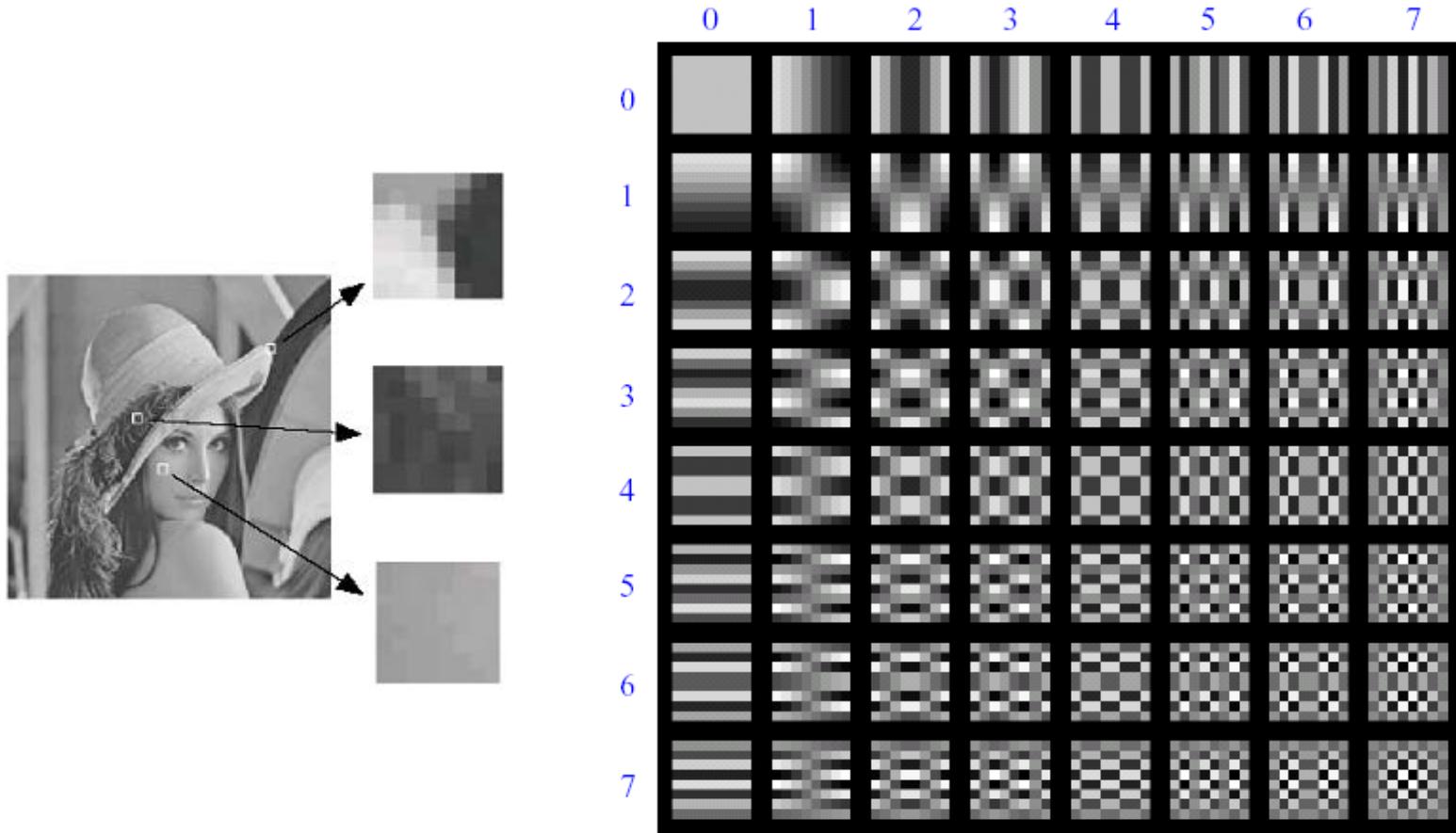
R

G

B



Lossy Image Compression (JPEG)



Block-based Discrete Cosine Transform (DCT)

Using DCT in JPEG

The first coefficient $B(0,0)$ is the DC component, the average intensity

The top-left coeffs represent low frequencies, the bottom right – high frequencies

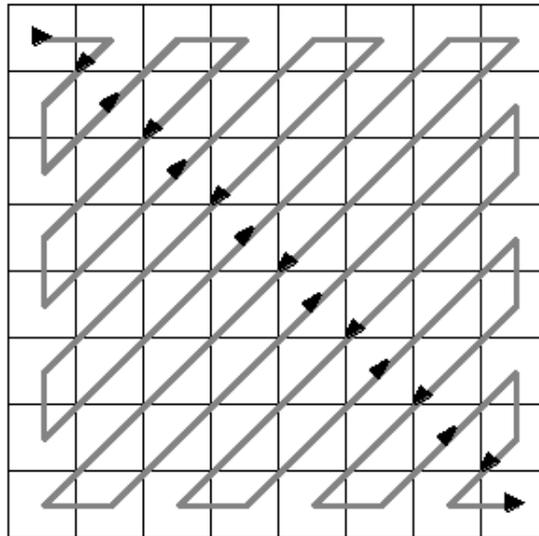


Image compression using DCT

Quantize

- More coarsely for high frequencies (which also tend to have smaller values)
- Many quantized high frequency values will be zero

Encode

- Can decode with inverse dct

Filter responses

$$G = \begin{matrix} & & & \xrightarrow{u} & & & & & \\ \begin{matrix} \downarrow v \\ \end{matrix} & \begin{bmatrix} -415.38 & -30.19 & -61.20 & 27.24 & 56.13 & -20.10 & -2.39 & 0.46 \\ 4.47 & -21.86 & -60.76 & 10.25 & 13.15 & -7.09 & -8.54 & 4.88 \\ -46.83 & 7.37 & 77.13 & -24.56 & -28.91 & 9.93 & 5.42 & -5.65 \\ -48.53 & 12.07 & 34.10 & -14.76 & -10.24 & 6.30 & 1.83 & 1.95 \\ 12.12 & -6.55 & -13.20 & -3.95 & -1.88 & 1.75 & -2.79 & 3.14 \\ -7.73 & 2.91 & 2.38 & -5.94 & -2.38 & 0.94 & 4.30 & 1.85 \\ -1.03 & 0.18 & 0.42 & -2.42 & -0.88 & -3.02 & 4.12 & -0.66 \\ -0.17 & 0.14 & -1.07 & -4.19 & -1.17 & -0.10 & 0.50 & 1.68 \end{bmatrix} \end{matrix}$$

Quantization table

$$Q = \begin{bmatrix} 16 & 11 & 10 & 16 & 24 & 40 & 51 & 61 \\ 12 & 12 & 14 & 19 & 26 & 58 & 60 & 55 \\ 14 & 13 & 16 & 24 & 40 & 57 & 69 & 56 \\ 14 & 17 & 22 & 29 & 51 & 87 & 80 & 62 \\ 18 & 22 & 37 & 56 & 68 & 109 & 103 & 77 \\ 24 & 35 & 55 & 64 & 81 & 104 & 113 & 92 \\ 49 & 64 & 78 & 87 & 103 & 121 & 120 & 101 \\ 72 & 92 & 95 & 98 & 112 & 100 & 103 & 99 \end{bmatrix}$$

Quantized values

$$B = \begin{bmatrix} -26 & -3 & -6 & 2 & 2 & -1 & 0 & 0 \\ 0 & -2 & -4 & 1 & 1 & 0 & 0 & 0 \\ -3 & 1 & 5 & -1 & -1 & 0 & 0 & 0 \\ -3 & 1 & 2 & -1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

JPEG Compression Summary

Subsample color by factor of 2

- People have bad resolution for color

Split into blocks (8x8, typically), subtract 128

For each block

- a. Compute DCT coefficients
- b. Coarsely quantize
 - Many high frequency components will become zero
- c. Encode (e.g., with Huffman coding)

Block size in JPEG

Block size

- small block
 - faster
 - correlation exists between neighboring pixels
- large block
 - better compression in smooth regions
- It's 8x8 in standard JPEG

JPEG compression comparison

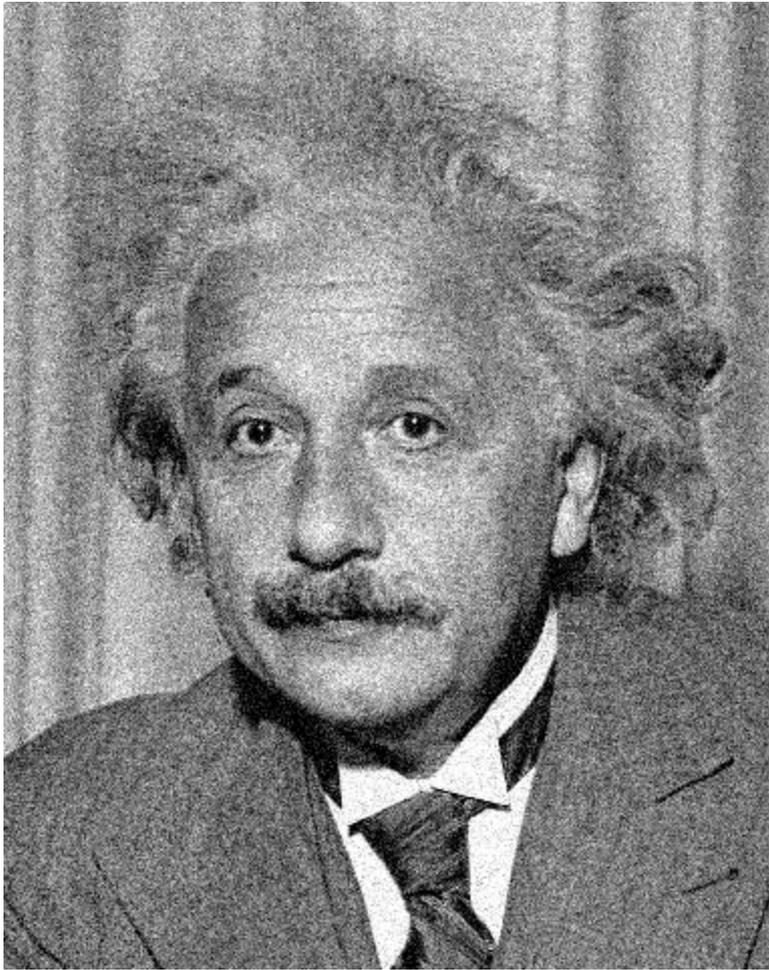


89k

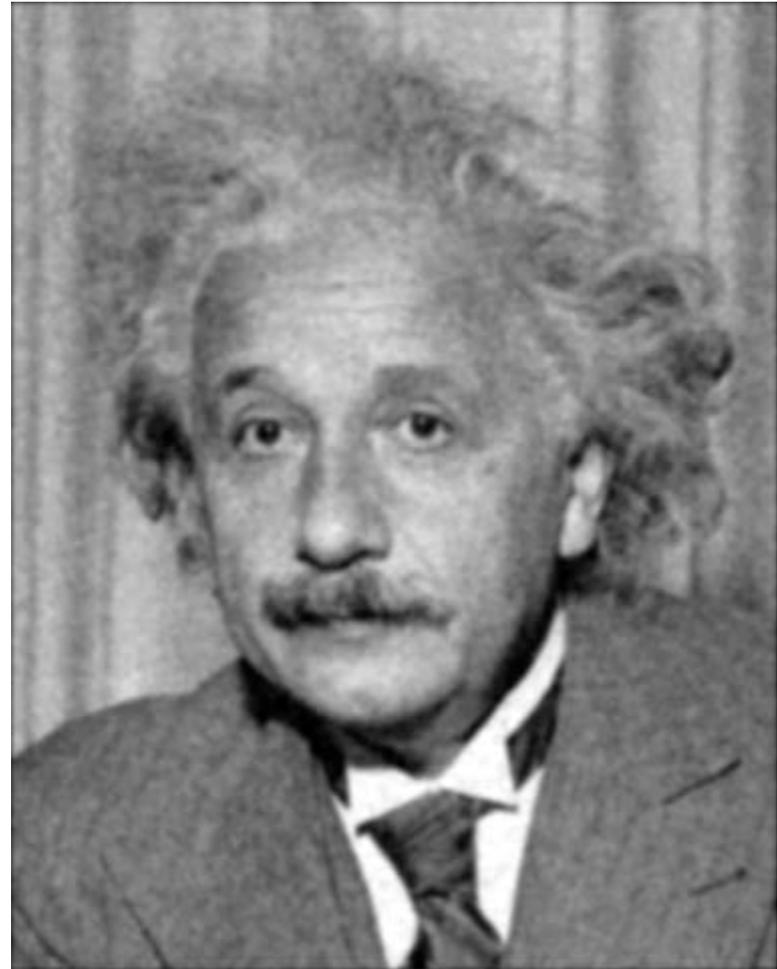


12k

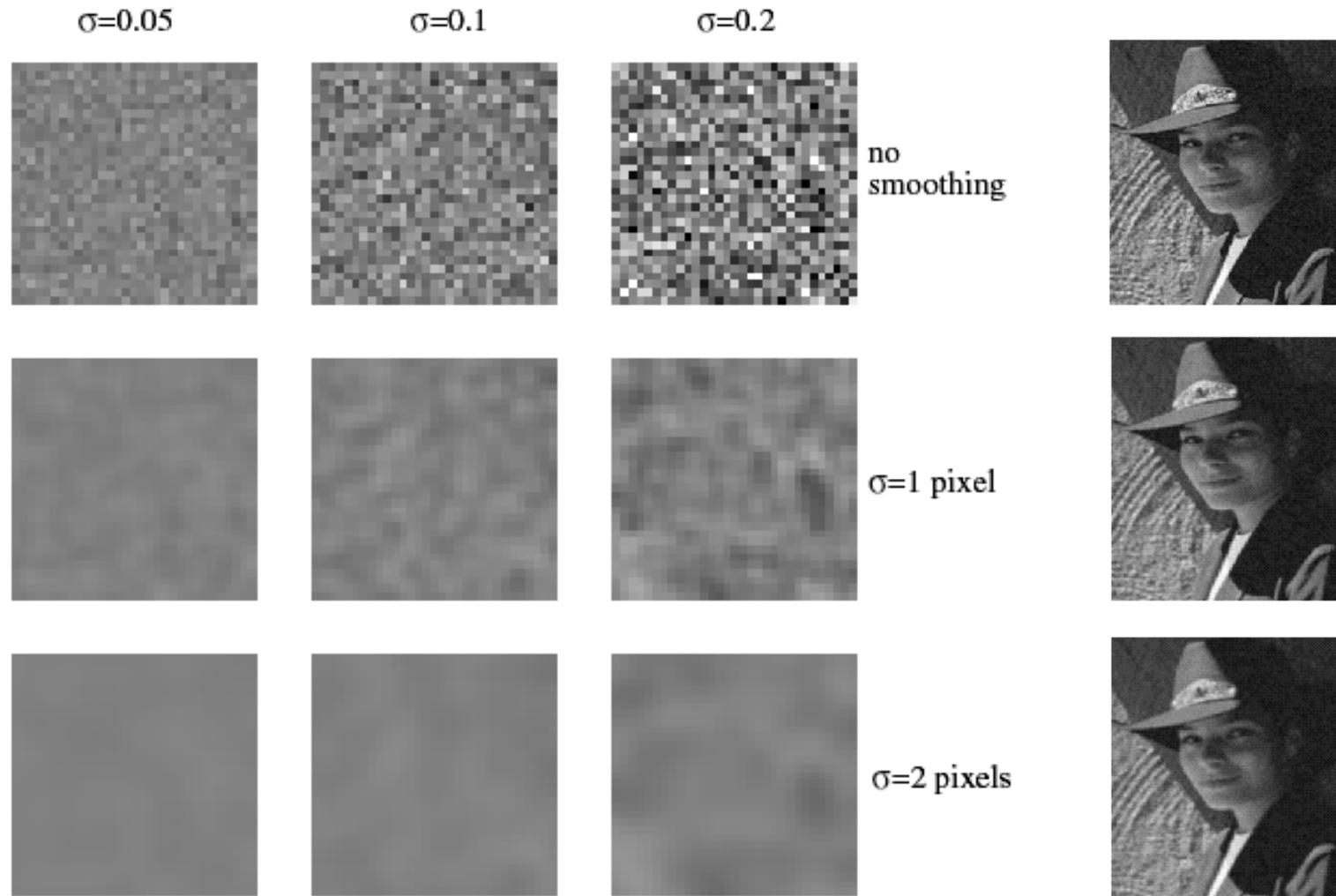
Denoising



Additive Gaussian Noise



Reducing Gaussian noise



Smoothing with larger standard deviations suppresses noise, but also blurs the image

Reducing salt-and-pepper noise by Gaussian smoothing

3x3



5x5

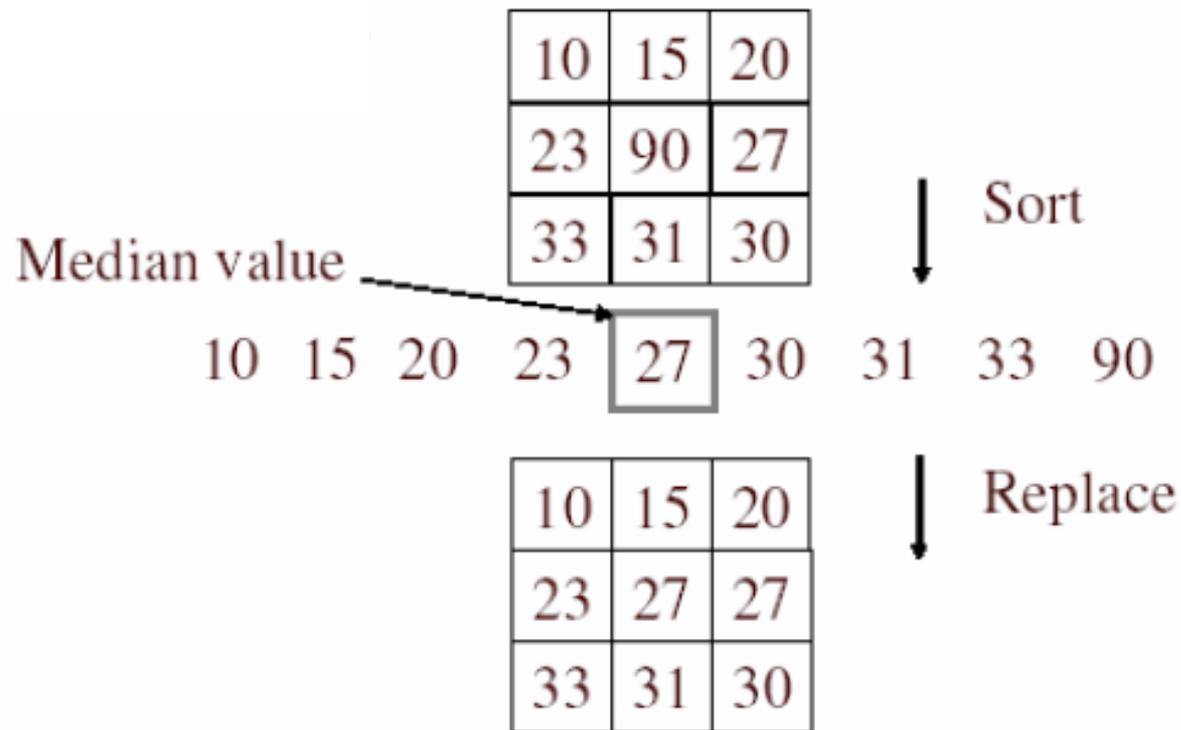


7x7



Alternative idea: Median filtering

A **median filter** operates over a window by selecting the median intensity in the window



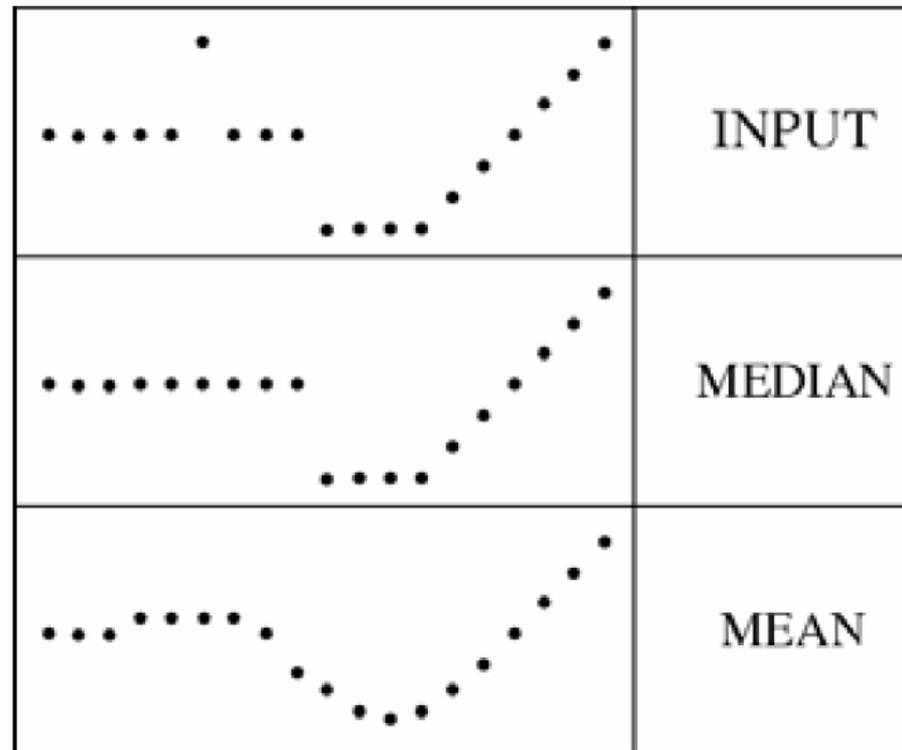
- Is median filtering linear?

Median filter

What advantage does median filtering have over Gaussian filtering?

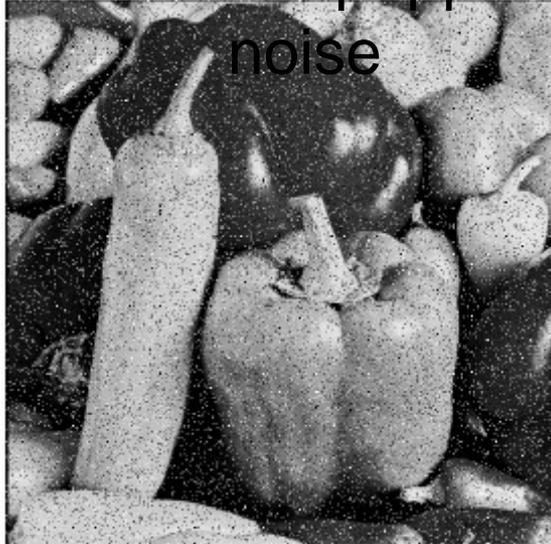
- Robustness to outliers

filters have width 5 :

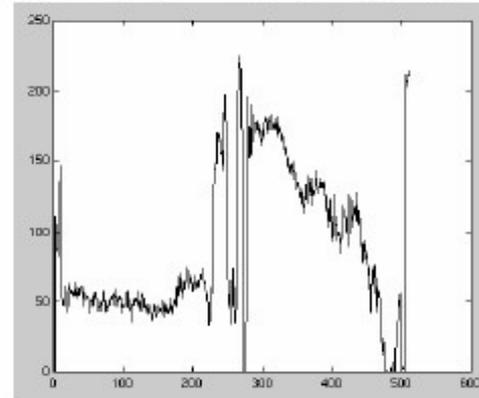
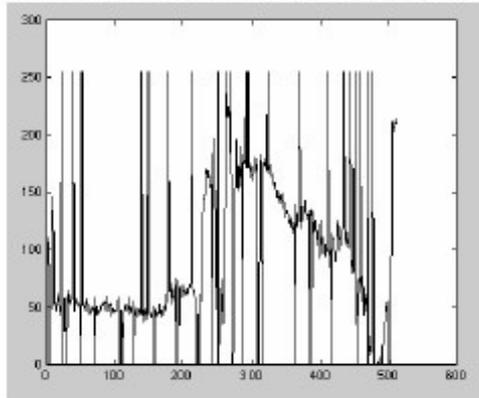


Median filter

Salt-and-pepper
noise



Median filtered



MATLAB: `medfilt2(image, [h w])`

Median vs. Gaussian filtering

3x3

5x5

7x7

Gaussian



Median



**A Gentle Introduction
to Bilateral Filtering
and its Applications**

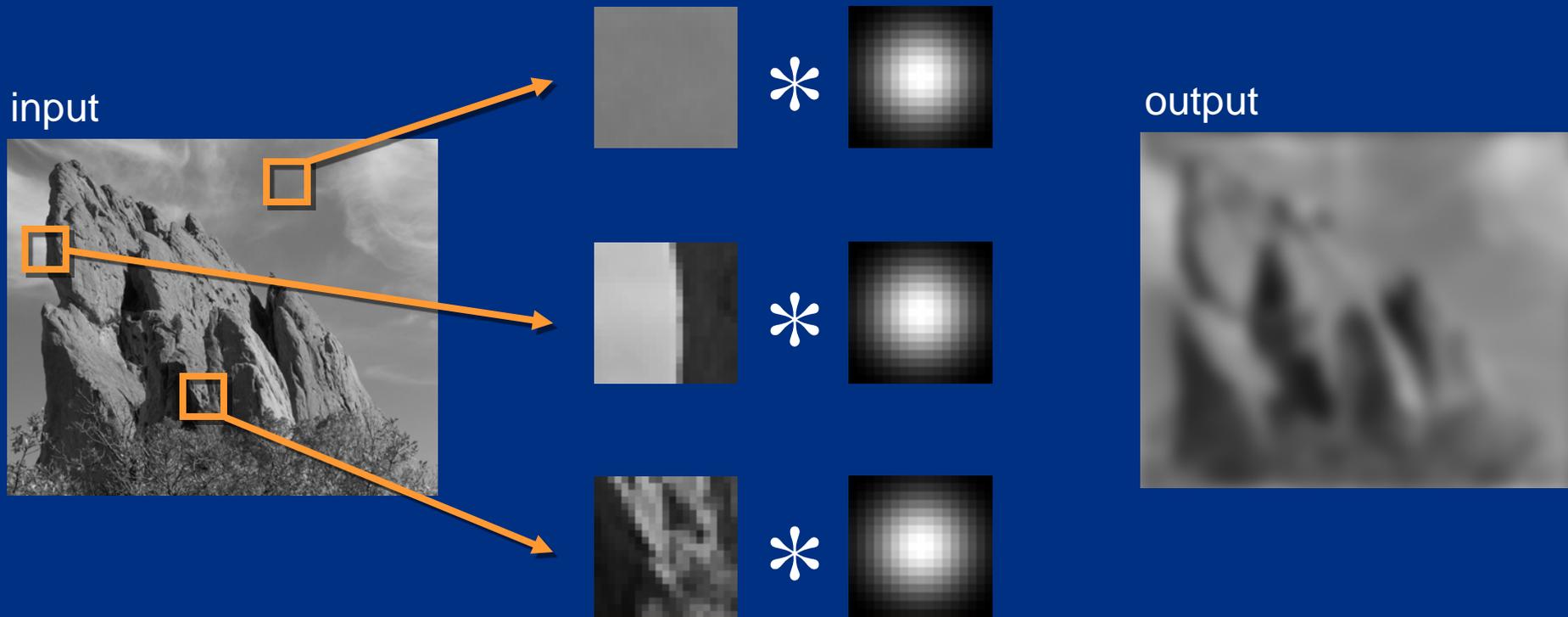


SIGGRAPH2007

**“Fixing the Gaussian Blur”:
the Bilateral Filter**

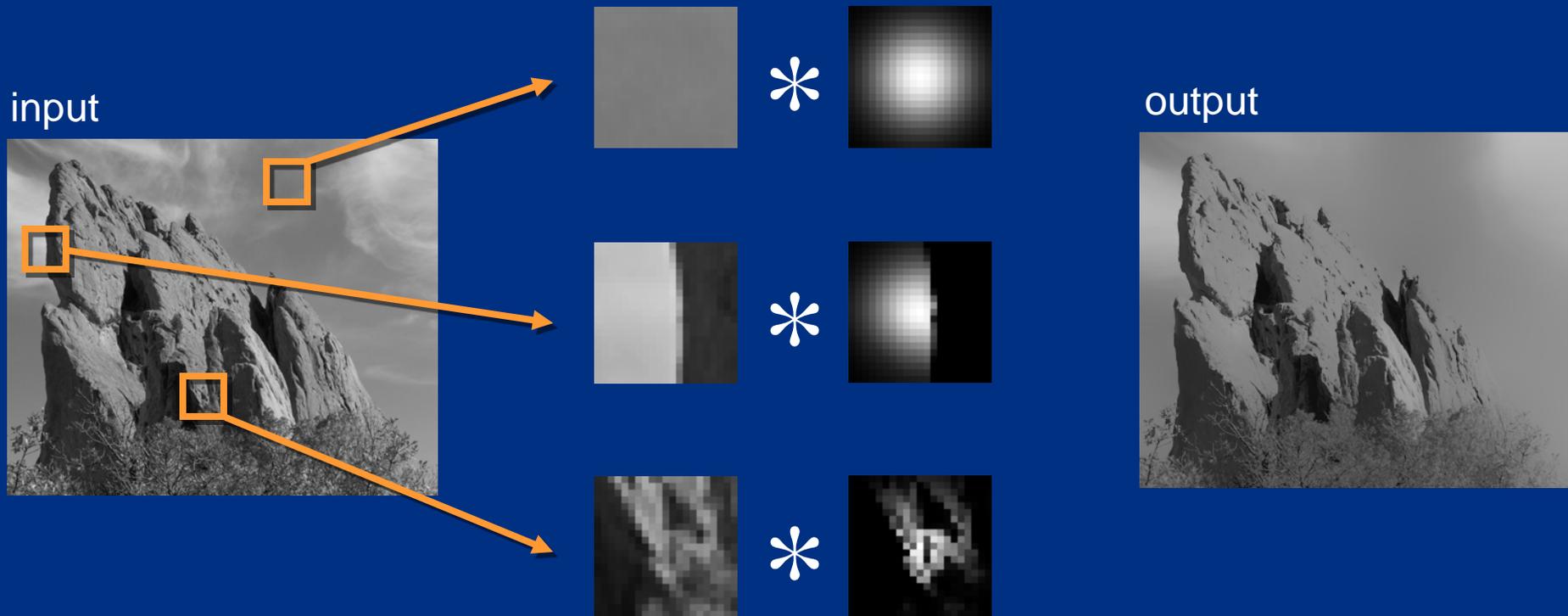
Sylvain Paris – MIT CSAIL

Blur Comes from Averaging across Edges



Bilateral Filter [Aurich 95, Smith 97, Tomasi 98]

No Averaging across Edges



The kernel shape depends on the image content.

Bilateral Filter Definition: an Additional Edge Term

Same idea: weighted average of pixels.

$$BF[I]_p = \frac{1}{W_p} \sum_{q \in S} G_{\sigma_s}(\|p - q\|) G_{\sigma_r}(|I_p - I_q|) I_q$$

new
not new
new

normalization factor *space* weight *range* weight

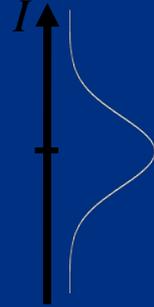
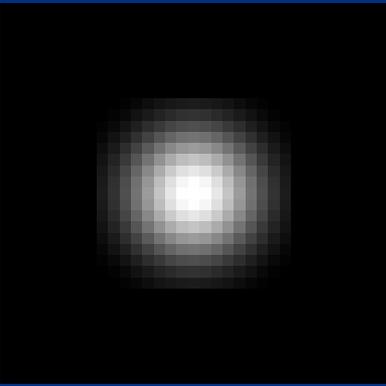
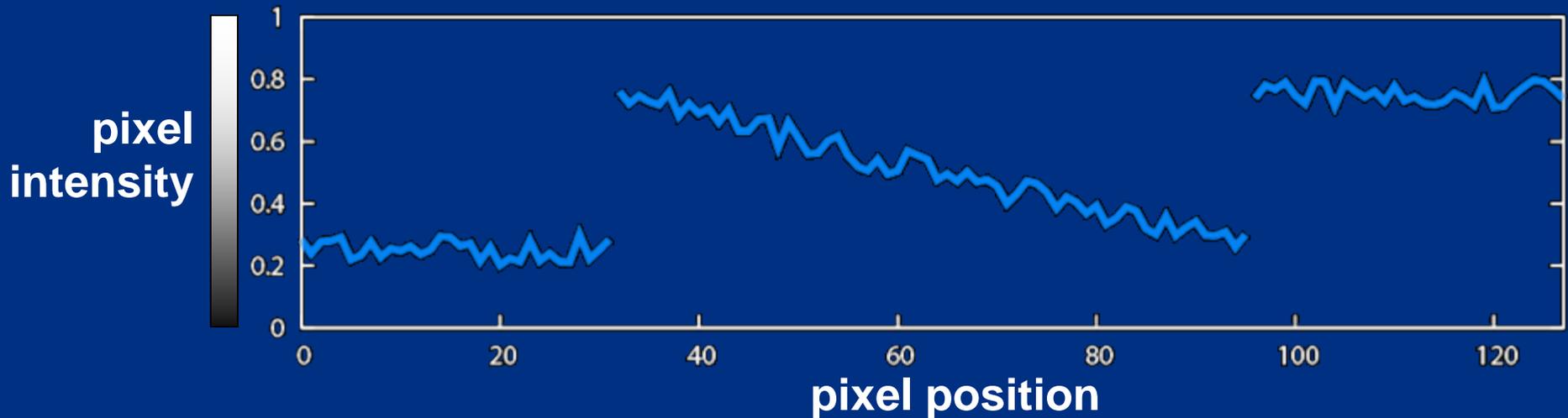


Illustration a 1D Image

- 1D image = line of pixels

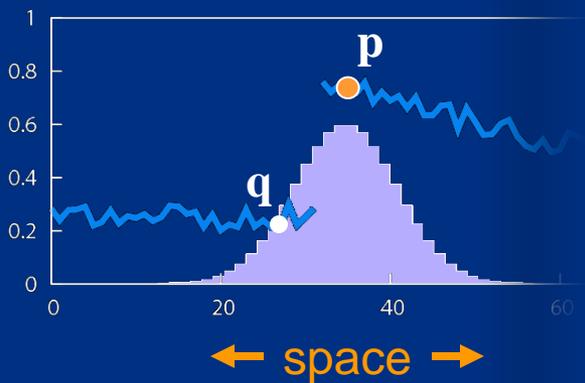


- Better visualized as a plot



Gaussian Blur and Bilateral Filter

Gaussian blur

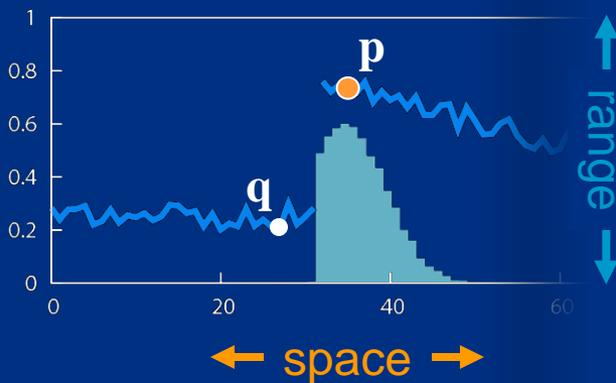


$$GB[I]_p = \sum_{q \in S} G_{\sigma}(\|p - q\|) I_q$$

space

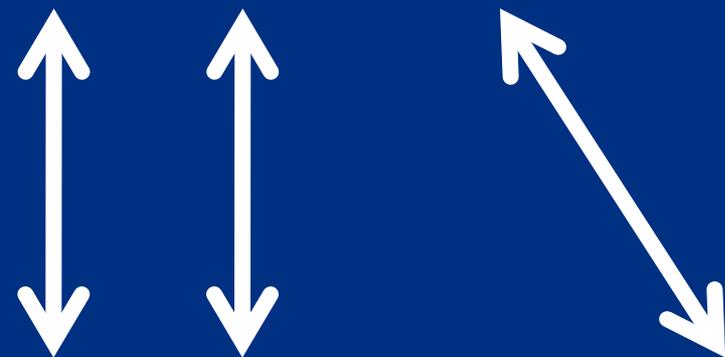
Bilateral filter

[Aurich 95, Smith 97, Tomasi 98]



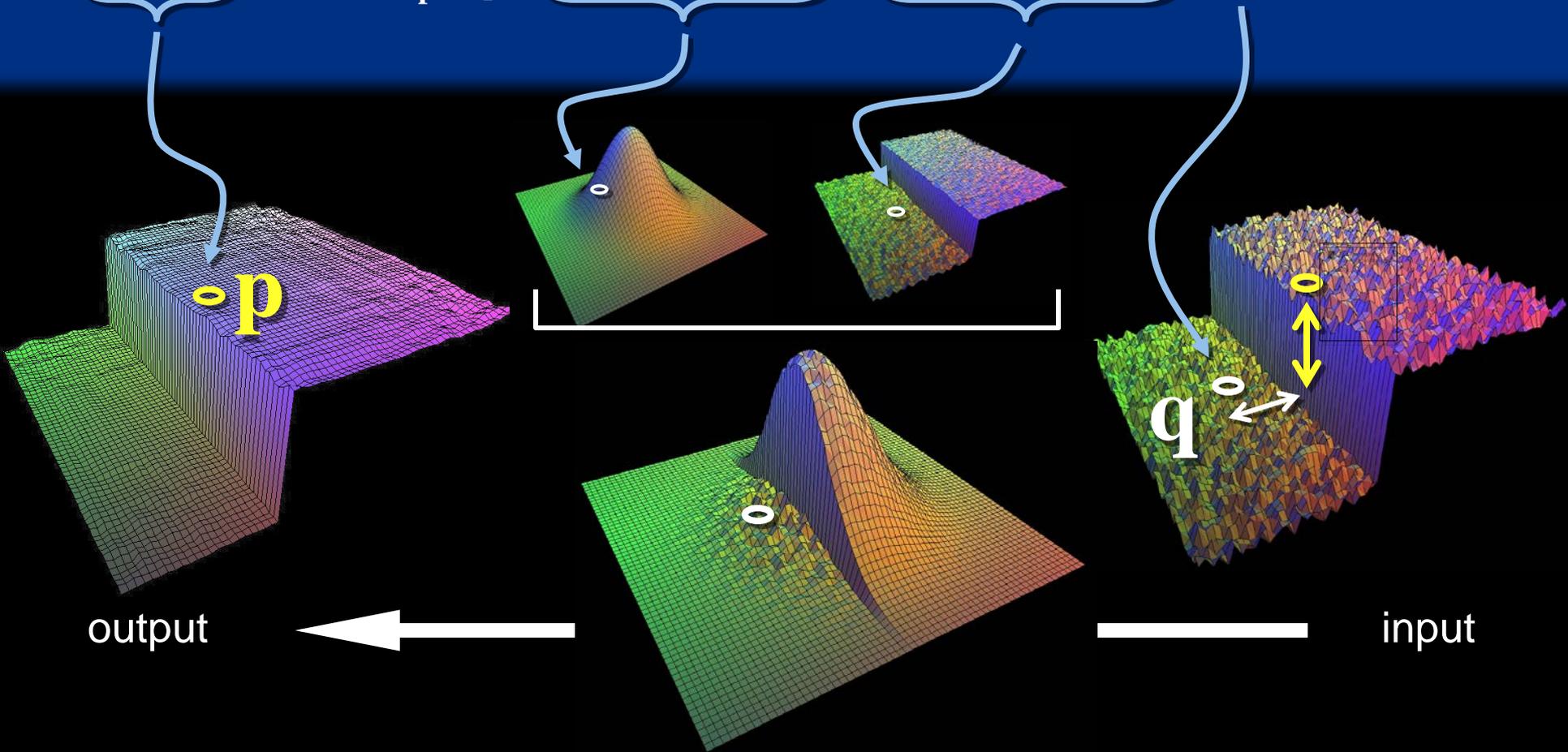
$$BF[I]_p = \frac{1}{W_p} \sum_{q \in S} G_{\sigma_s}(\|p - q\|) G_{\sigma_r}(|I_p - I_q|) I_q$$

normalization space range



Bilateral Filter on a Height Field

$$BF[I]_p = \frac{1}{W_p} \sum_{q \in S} \underbrace{G_{\sigma_s}(\|\mathbf{p} - \mathbf{q}\|)}_{\text{spatial}} \underbrace{G_{\sigma_r}(\|I_p - I_q\|)}_{\text{range}} I_q$$



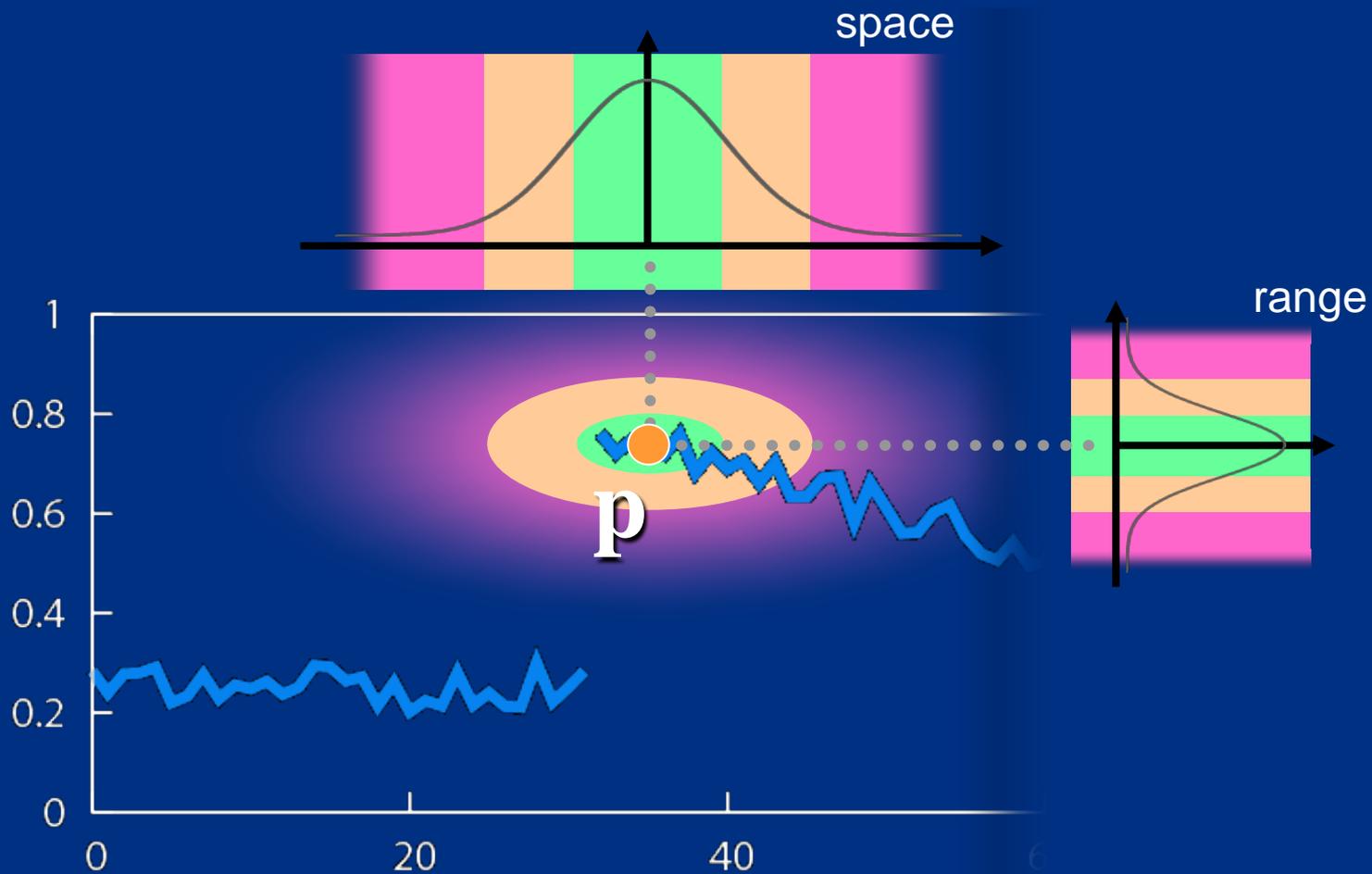
Space and Range Parameters

$$BF[I]_p = \frac{1}{W_p} \sum_{q \in S} G_{\sigma_s}(\|p - q\|) G_{\sigma_r}(|I_p - I_q|) I_q$$


- space σ_s : spatial extent of the kernel, size of the considered neighborhood.
- range σ_r : “minimum” amplitude of an edge

Influence of Pixels

Only pixels close in space and in range are considered.



Exploring the Parameter Space



input

$$\sigma_r = 0.1$$



$$\sigma_r = 0.25$$



$$\sigma_r = \infty$$

(Gaussian blur)



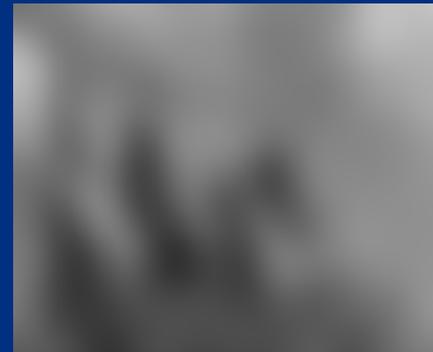
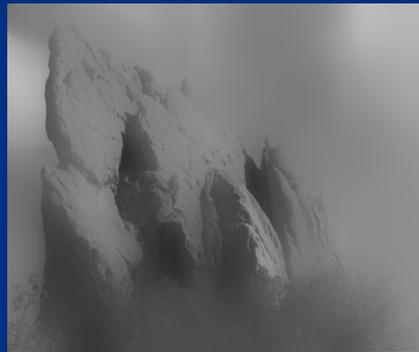
$$\sigma_s = 2$$



$$\sigma_s = 6$$



$$\sigma_s = 18$$



Varying the Range Parameter



input

$\sigma_s = 2$

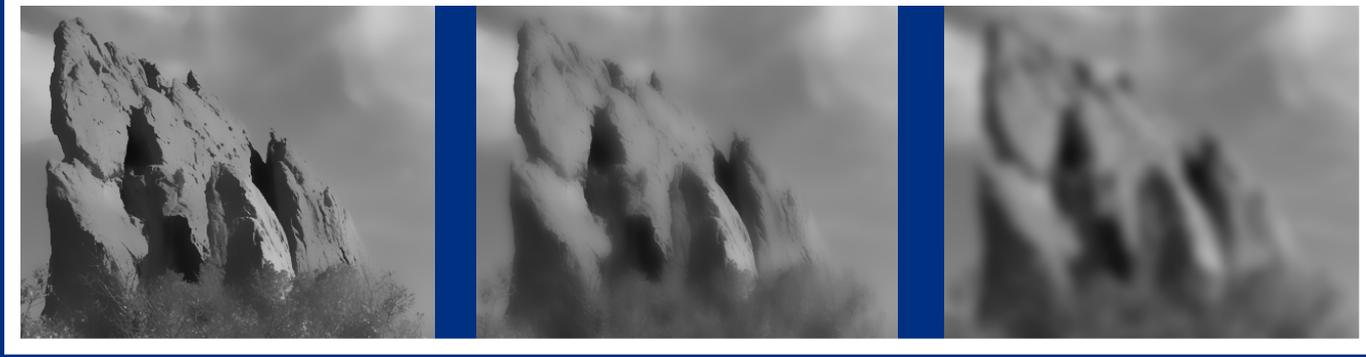
$\sigma_r = 0.1$

$\sigma_r = 0.25$

$\sigma_r = \infty$
(Gaussian blur)



$\sigma_s = 6$



$\sigma_s = 18$



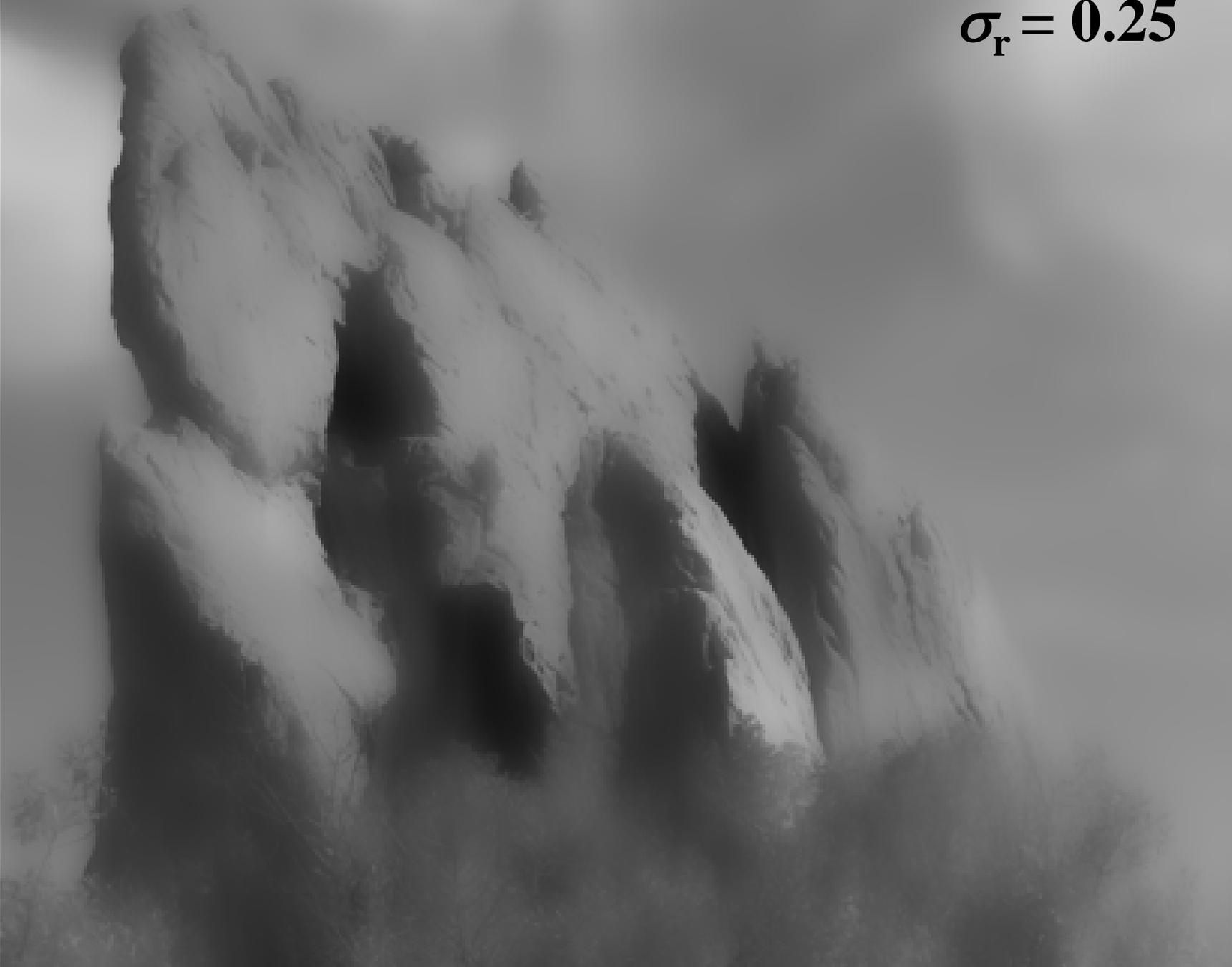
input



$$\sigma_r = 0.1$$



$$\sigma_r = 0.25$$



$$\sigma_r = \infty$$

(Gaussian blur)



Varying the Space Parameter



input

$\sigma_s = 2$



$\sigma_r = 0.1$

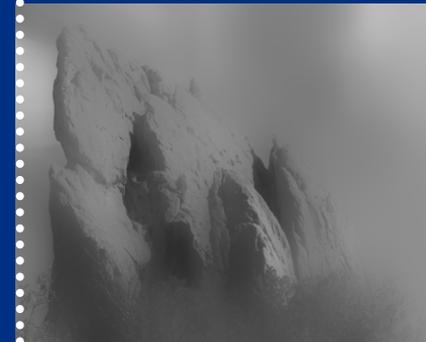
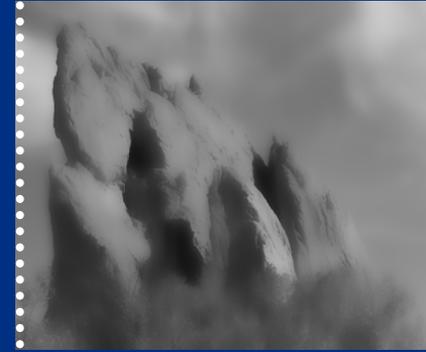
$\sigma_s = 6$



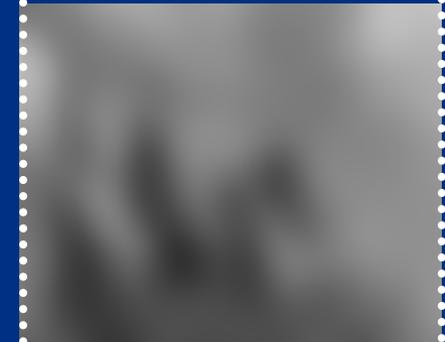
$\sigma_s = 18$



$\sigma_r = 0.25$



$\sigma_r = \infty$
(Gaussian blur)



input



$$\sigma_s = 2$$



$$\sigma_s = 6$$



$$\sigma_s = 18$$

