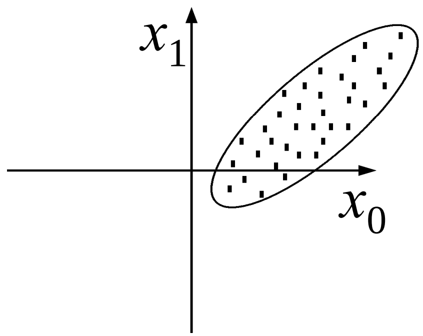
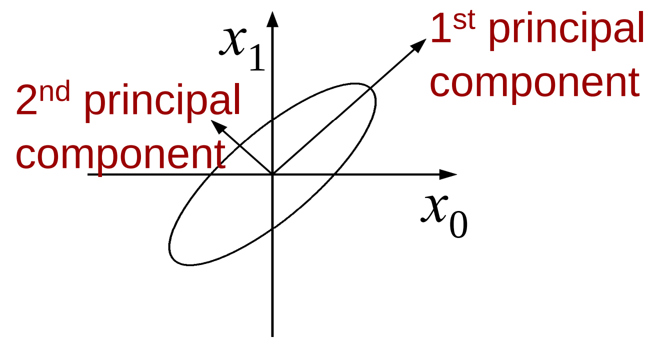
Fourier Analysis *Without Tears*

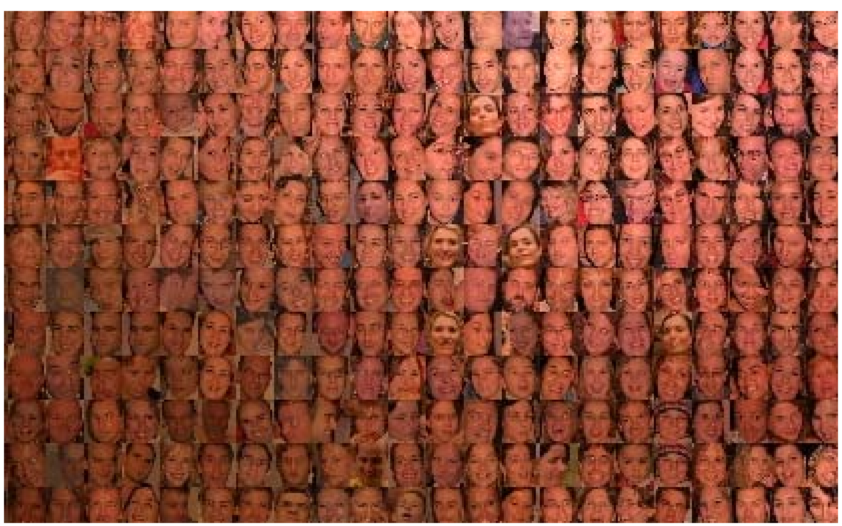
Somewhere in Cinque Terre, May 2005

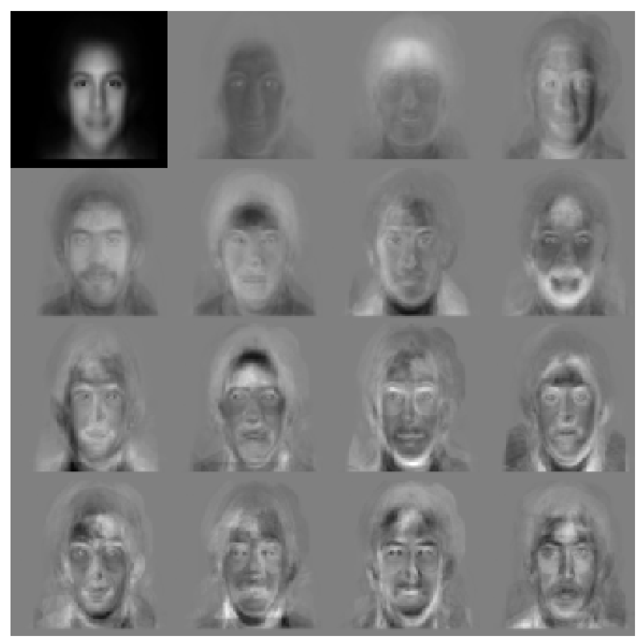
15-463: Computational Photography
Alexei Efros, CMU, Fall 2005

Capturing what's important

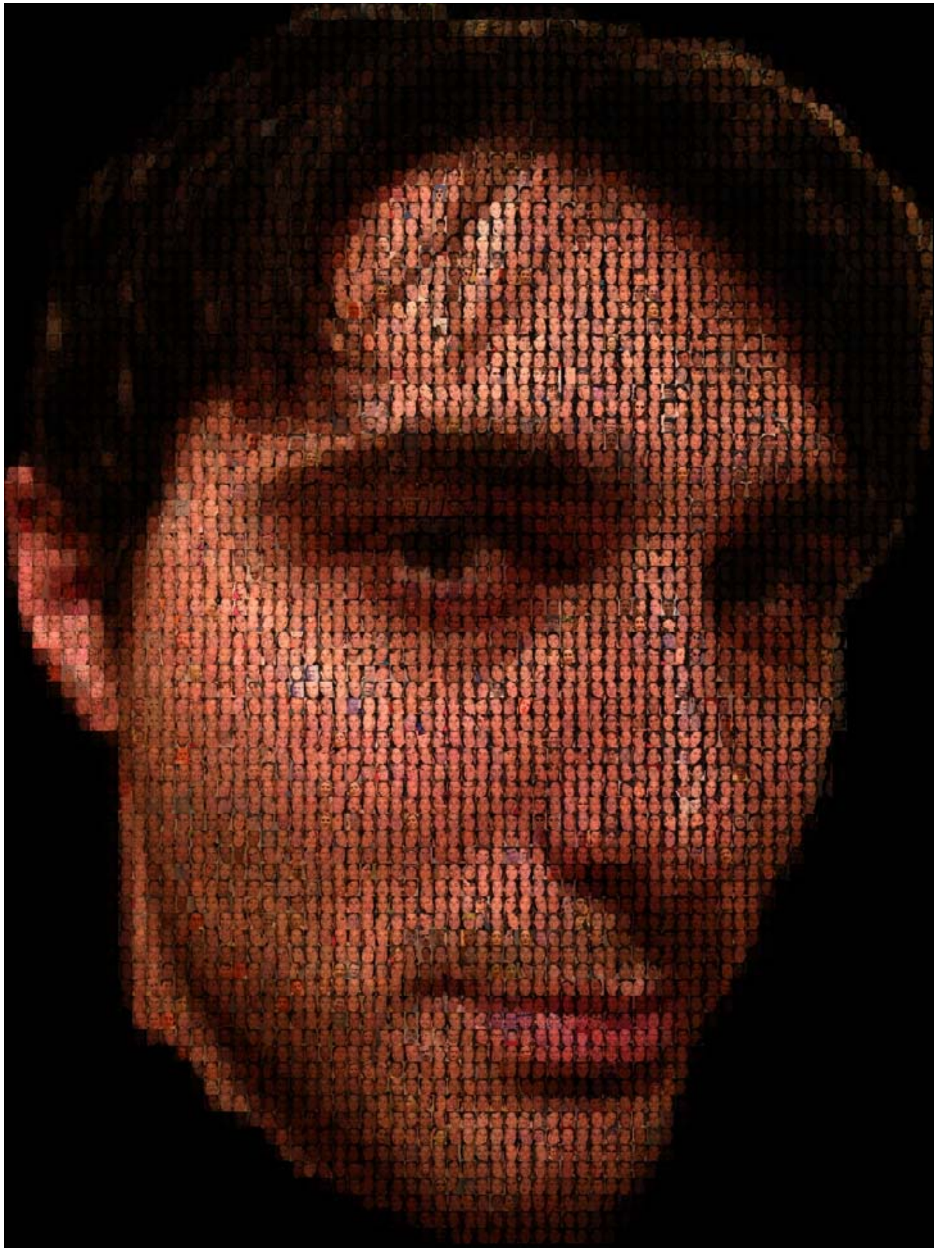


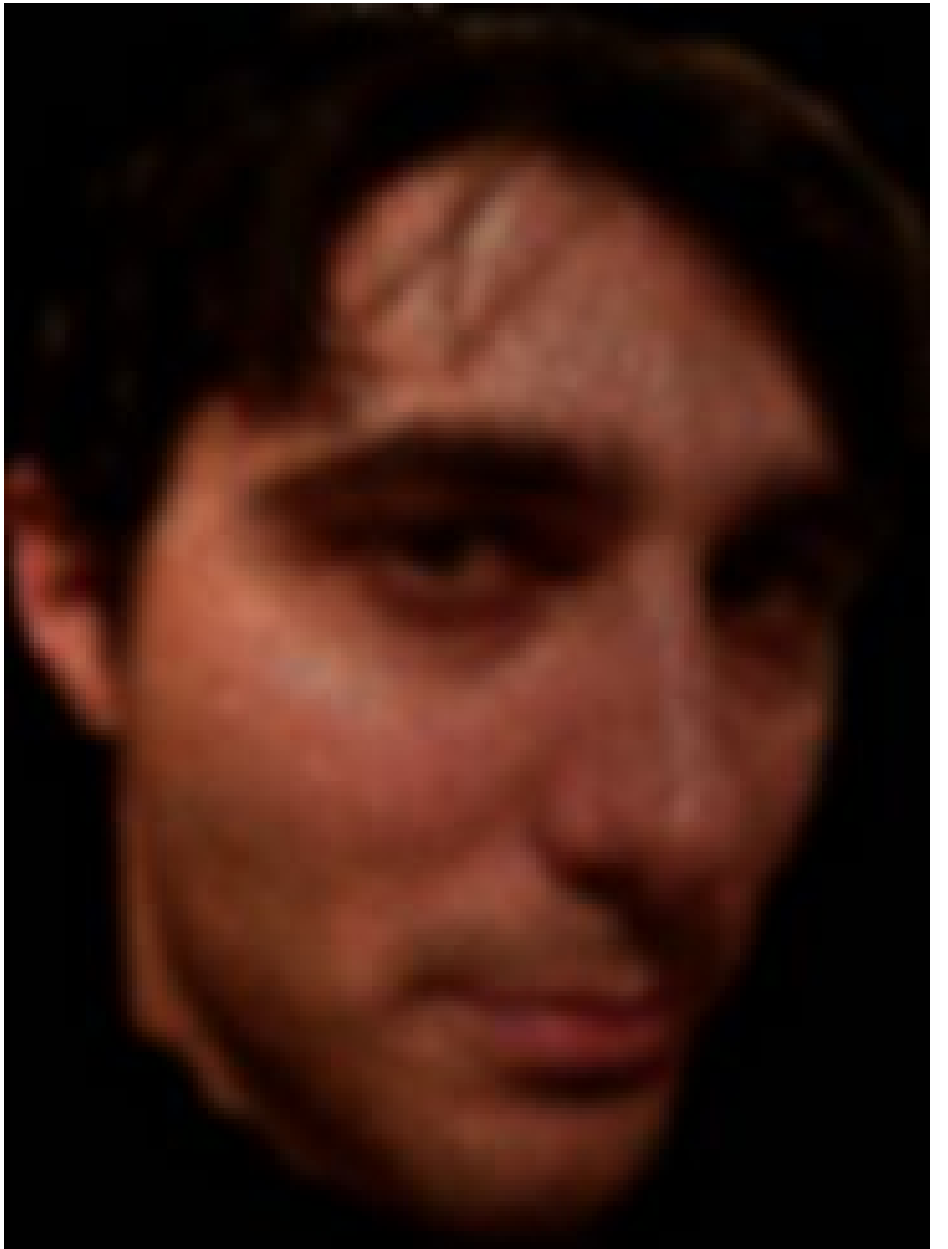






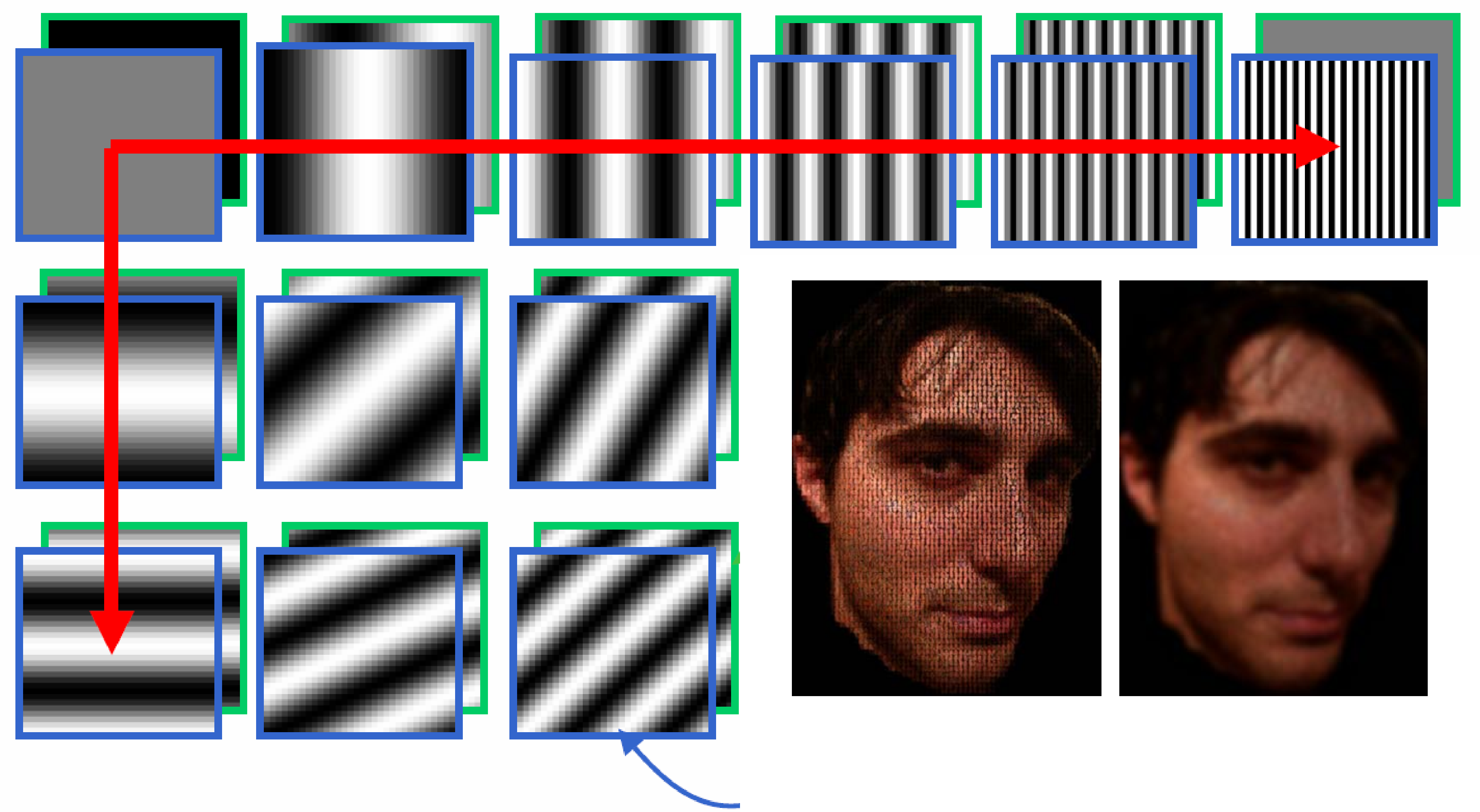
Fast vs. slow changes





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

Don't believe it?

- Neither did Lagrange, Laplace, Poisson and other big wigs
- Not translated into English until 1878!

But it's true!

- called Fourier Series

A sum of sines

Our building block:

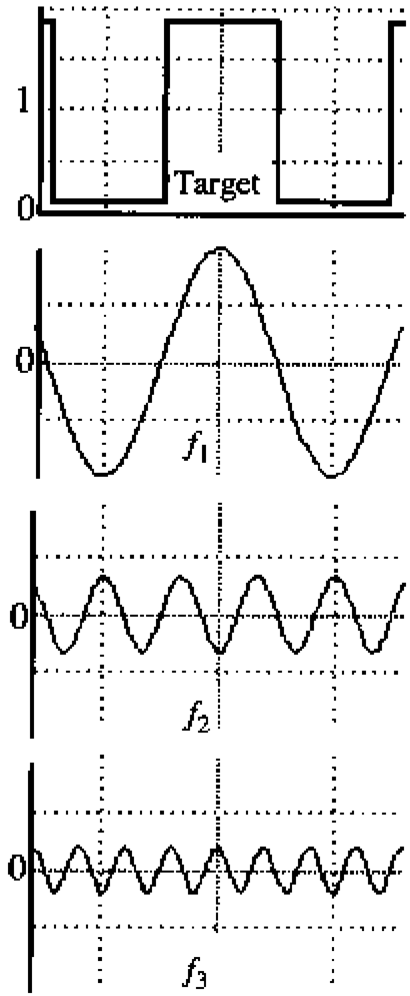
$$A\sin(\omega x + \phi)$$

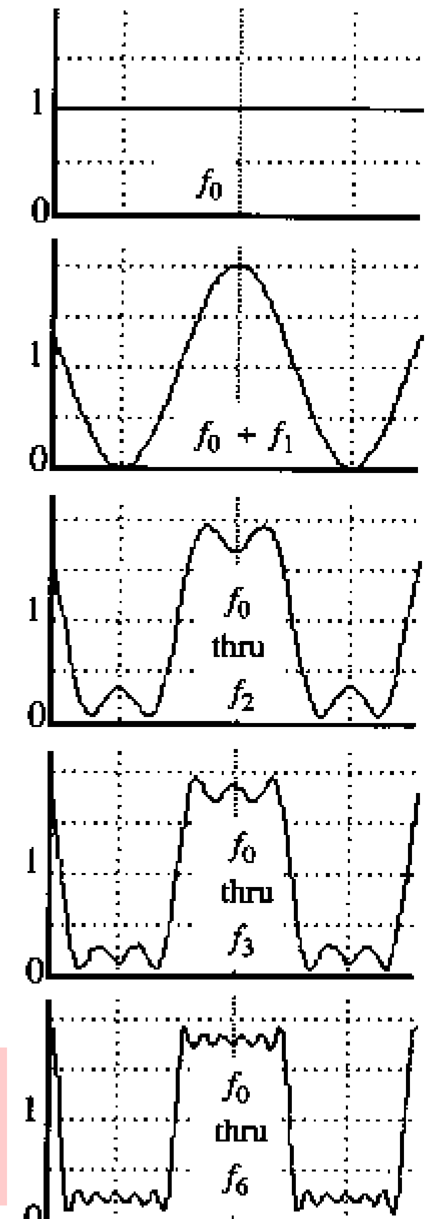
Add enough of them to get any signal *f(x)* you want!

How many degrees of freedom?

What does each control?

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





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*:

f(x) ⟶ Fourier Transform ⟶ ***F(ω)***

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? Complex number trick!

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

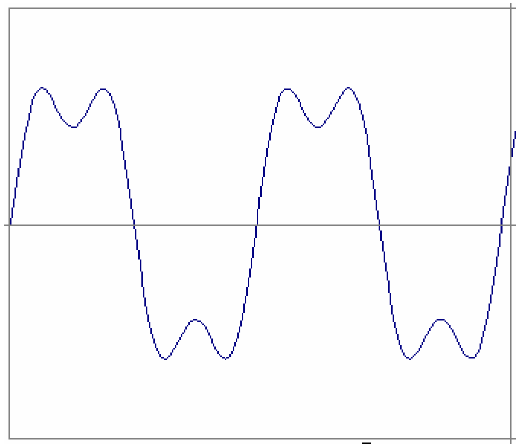
$$A=\pm\sqrt{R(\omega)^2+I(\omega)^2} \qquad \phi=\tan^{-1}\frac{I(\omega)}{R(\omega)}$$

We can always go back:

F(ω) ⟶ Inverse Fourier Transform ⟶ ***f(x)***

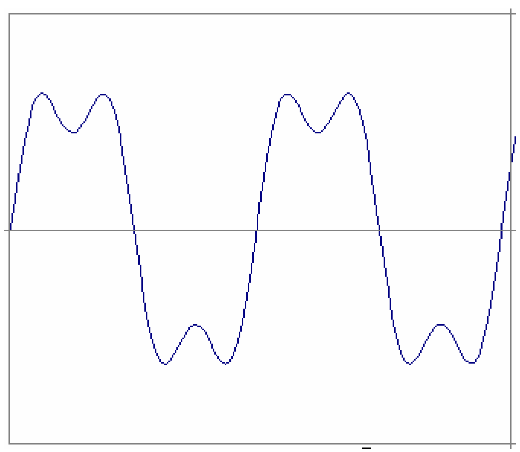
Time and Frequency

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

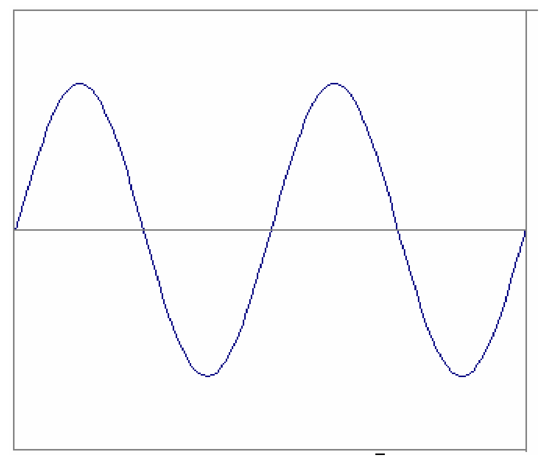


Time and Frequency

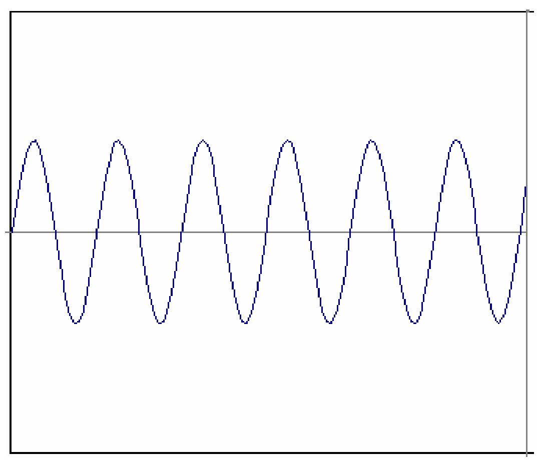
example : $g(t) = \sin(2pf\ t) + (1/3)\sin(2p(3f)\ t)$



=

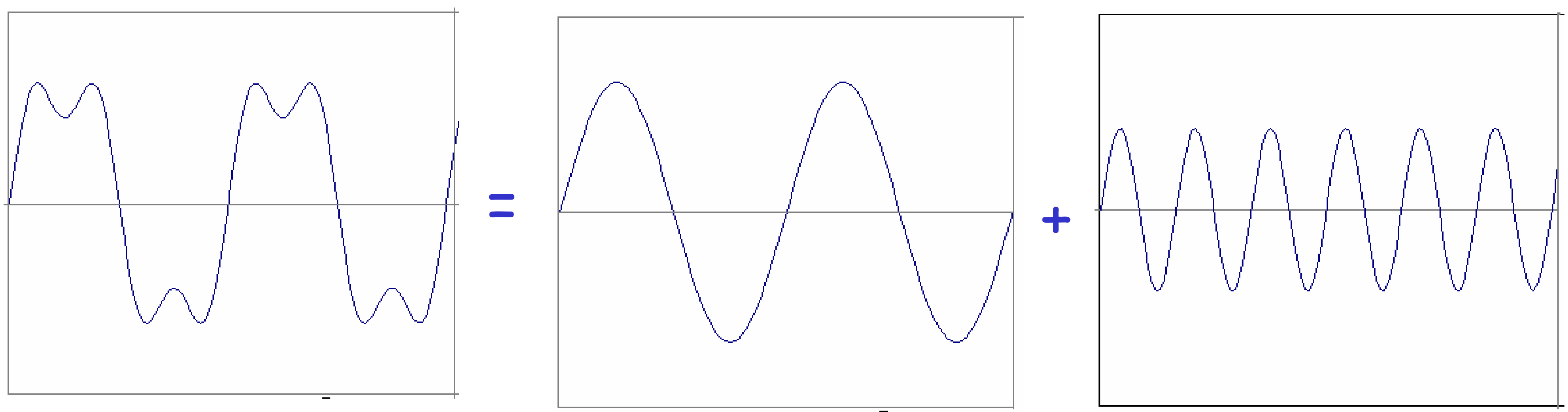


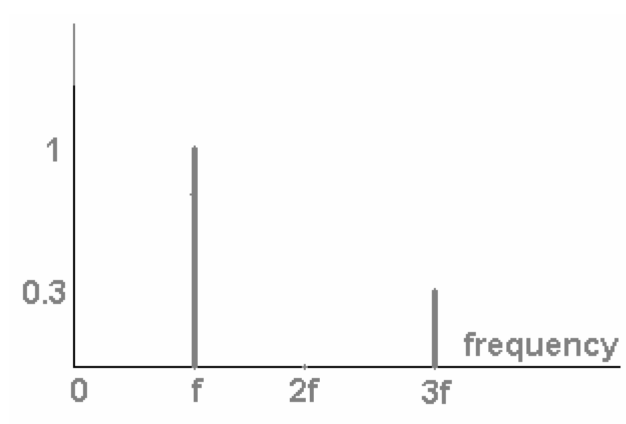
+



Frequency Spectra

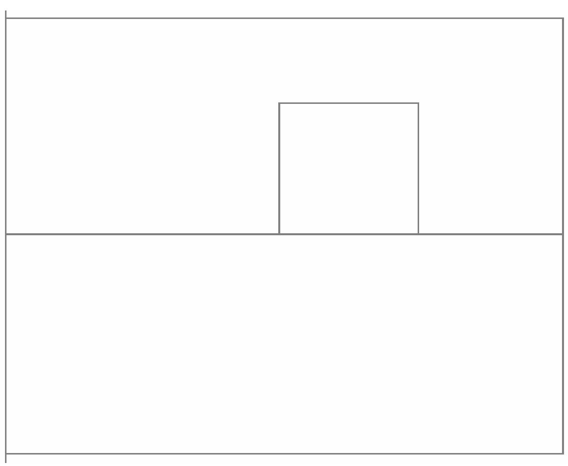
example : $g(t) = \sin(2pf\,t) + (1/3)\sin(2p(3f)\,t)$



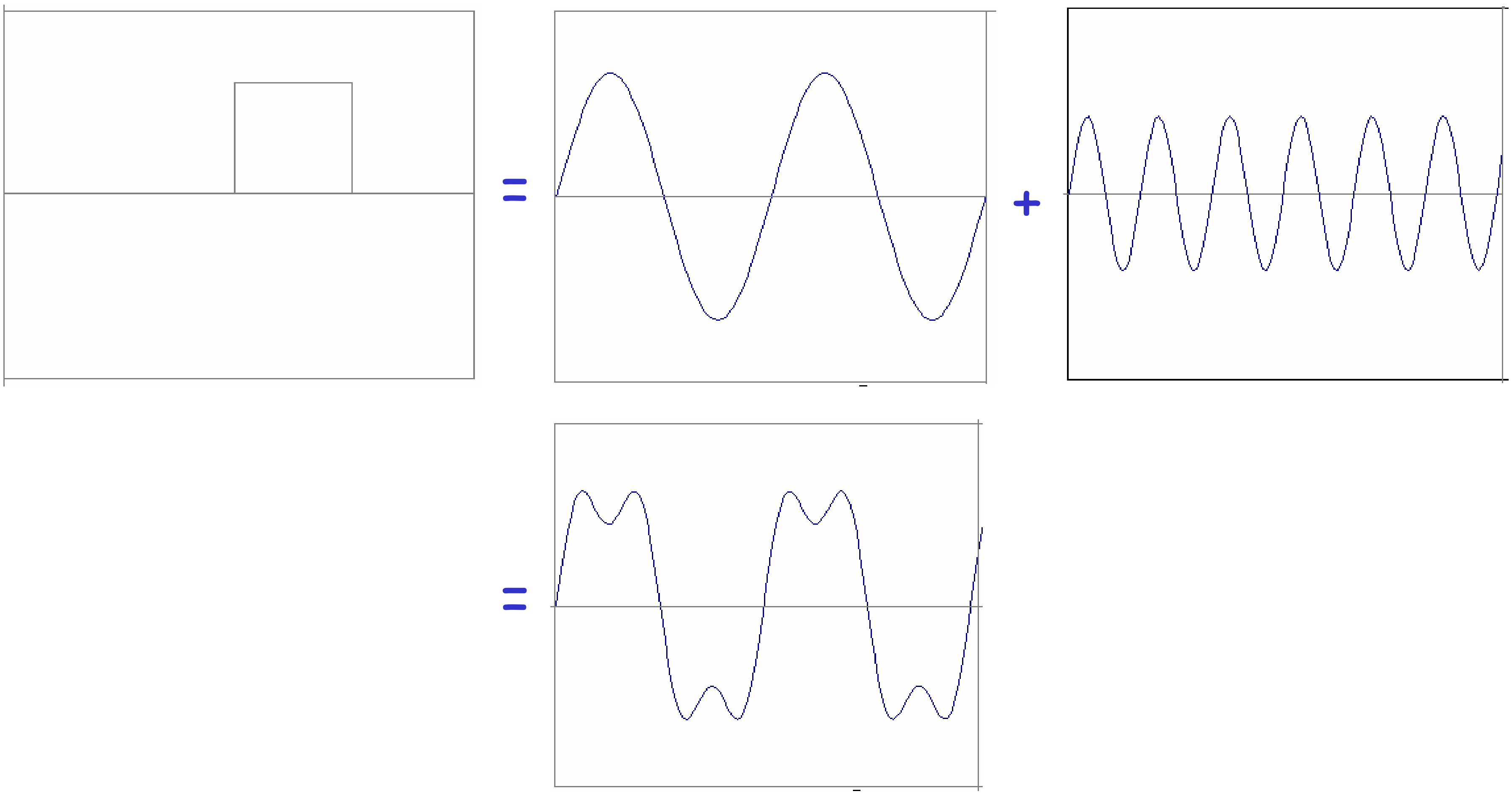


Frequency Spectra

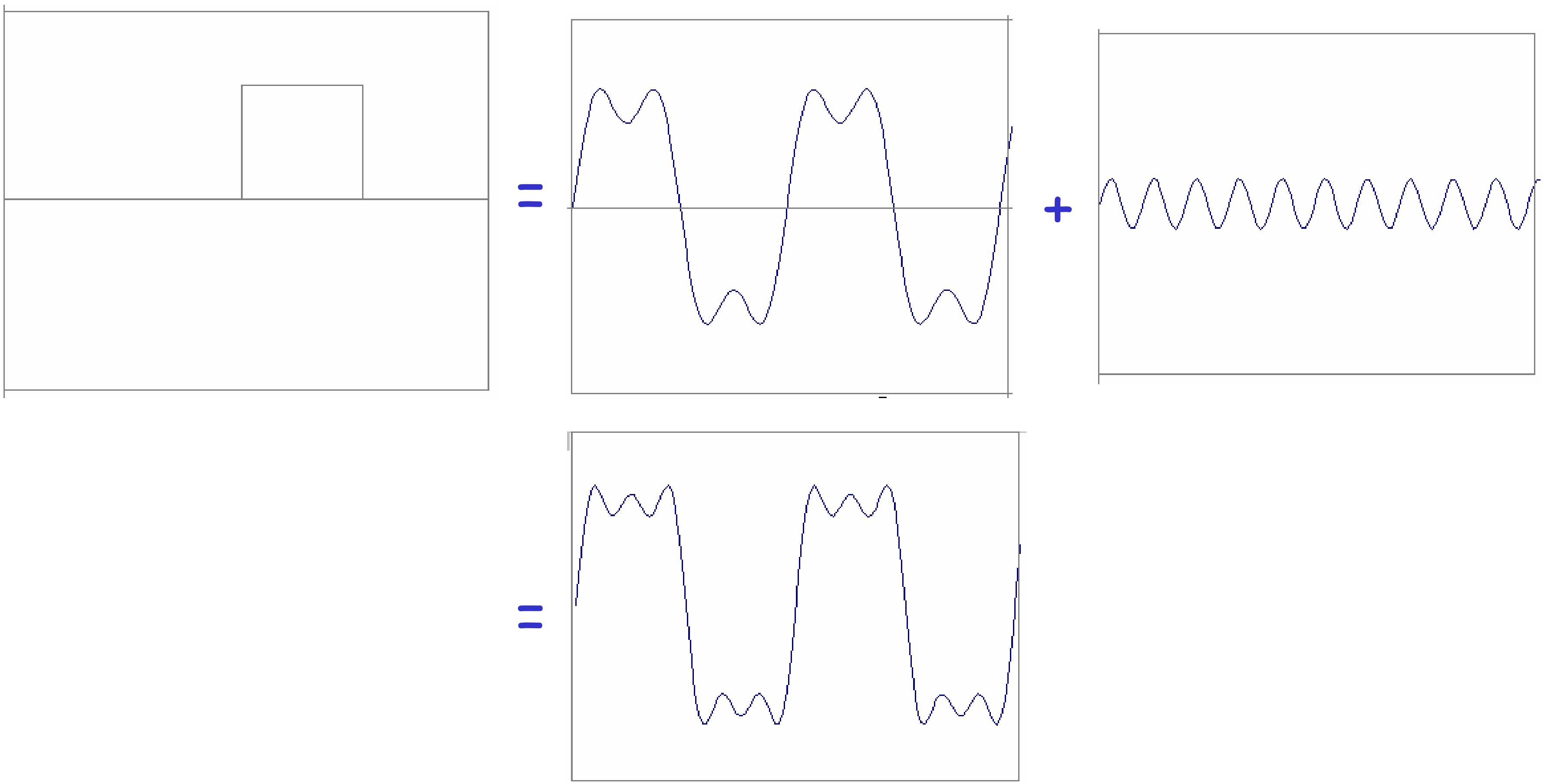
Usually, frequency is more interesting than the phase



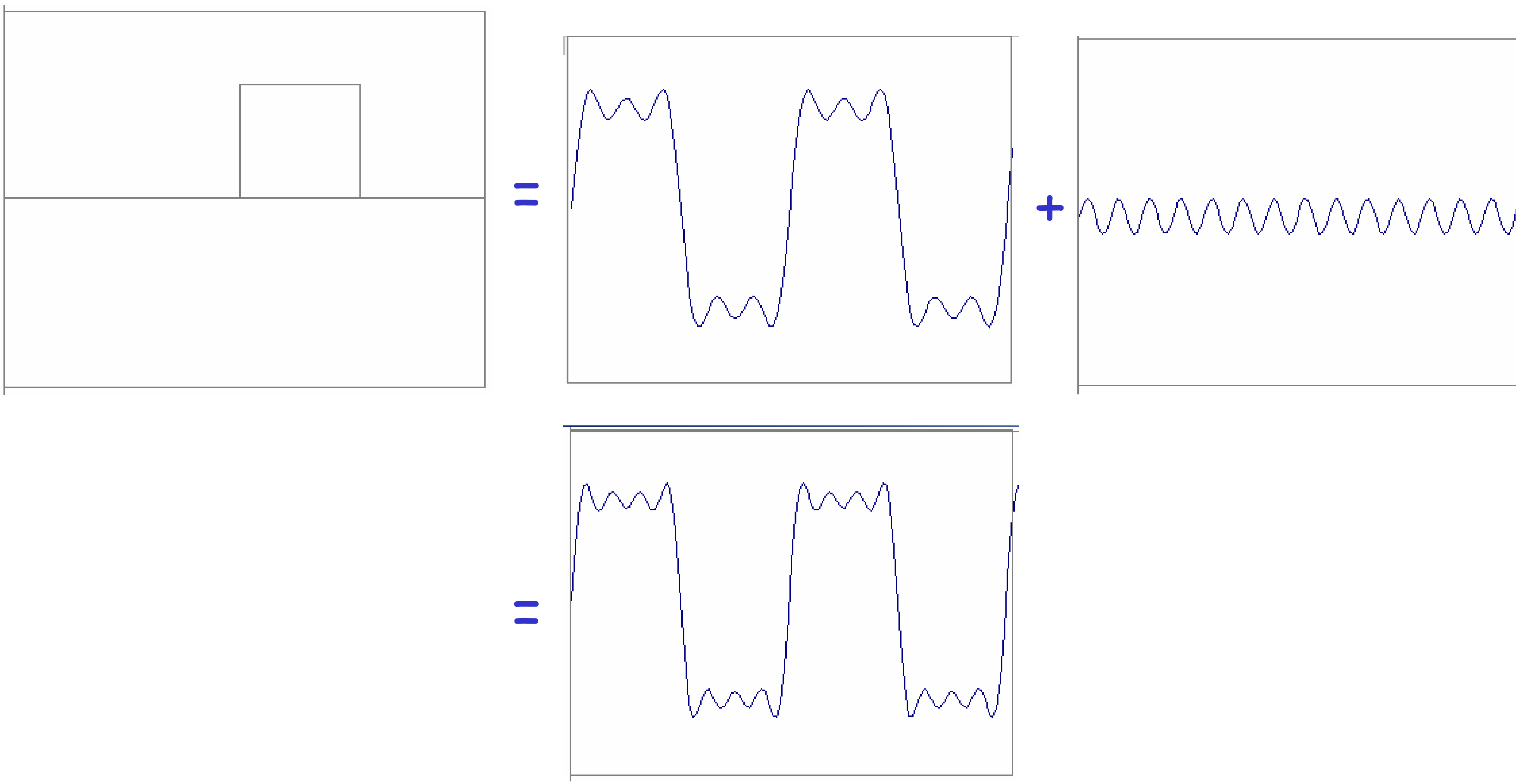
Frequency Spectra



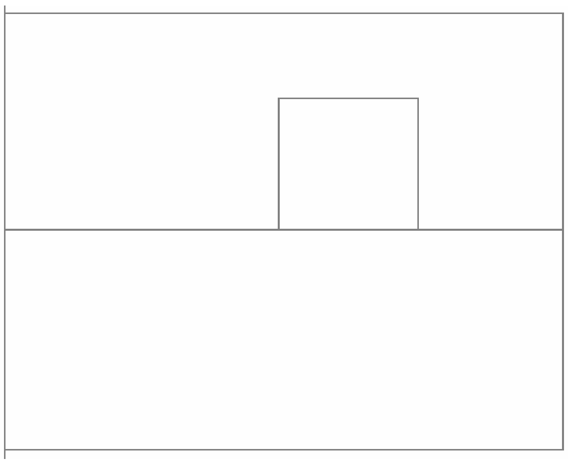
Frequency Spectra



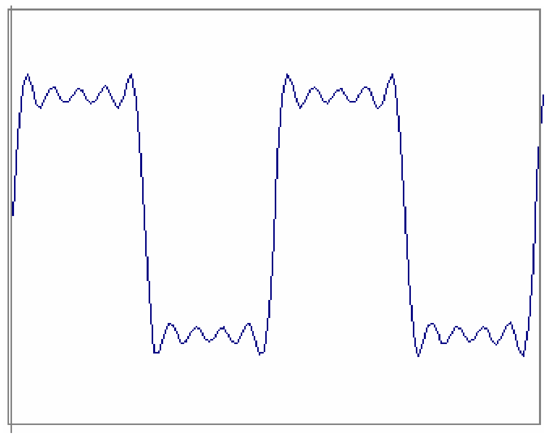
Frequency Spectra



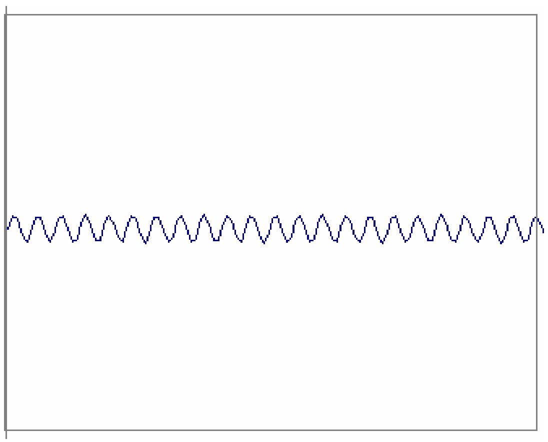
Frequency Spectra



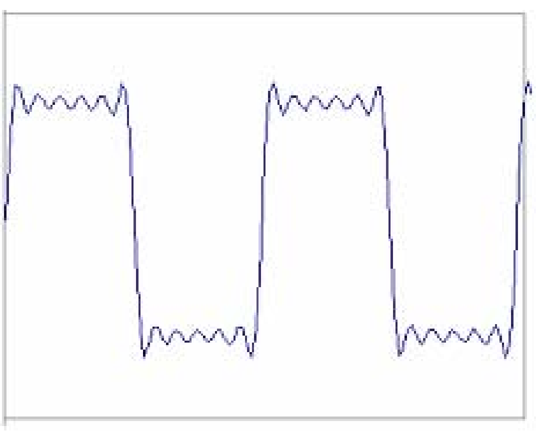
=



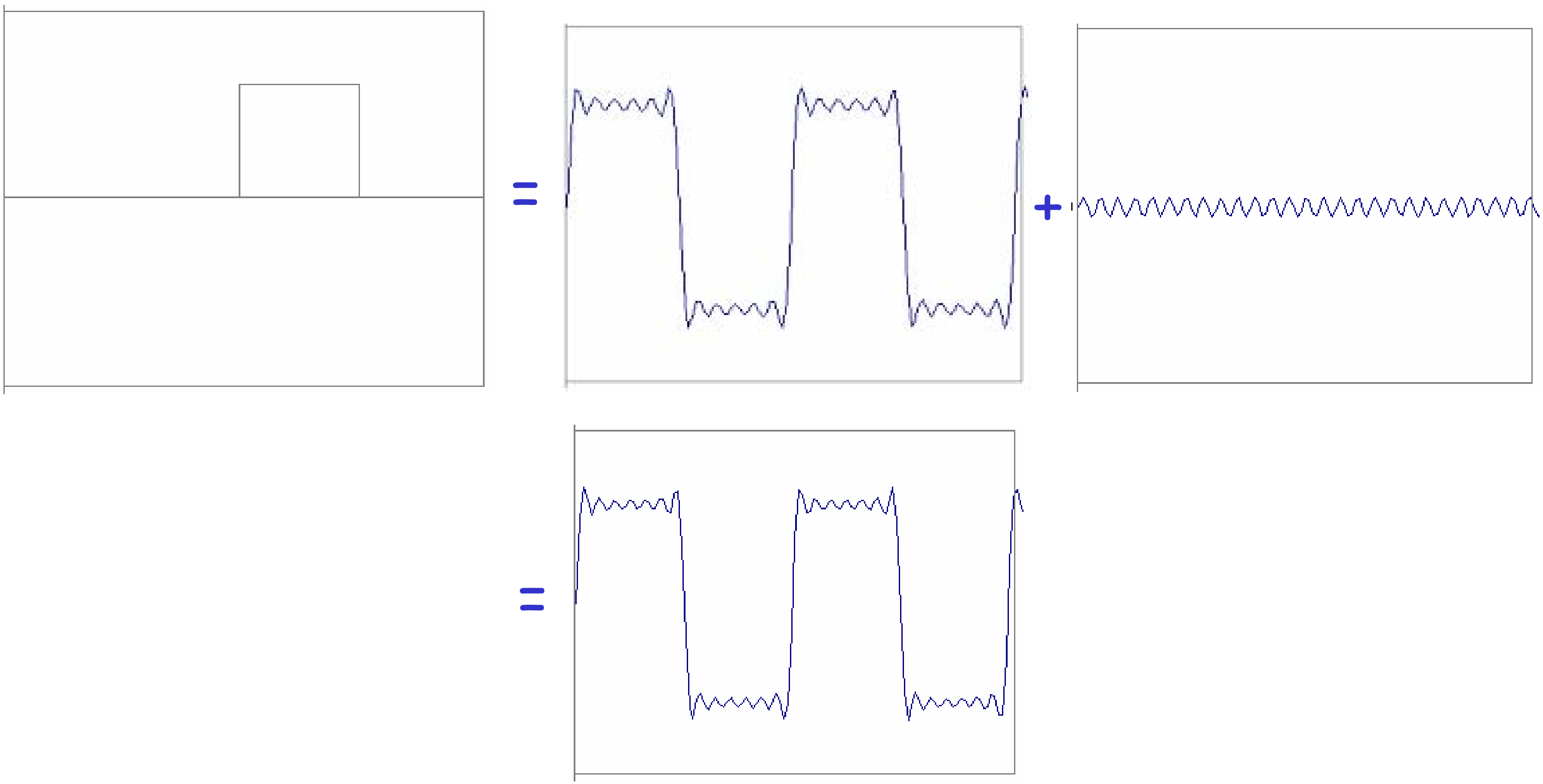
+



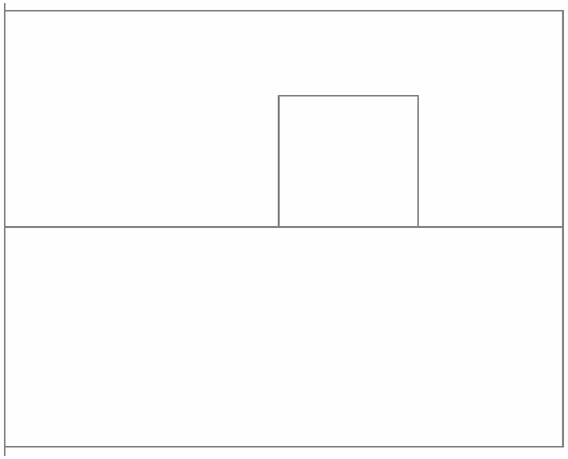
=



Frequency Spectra

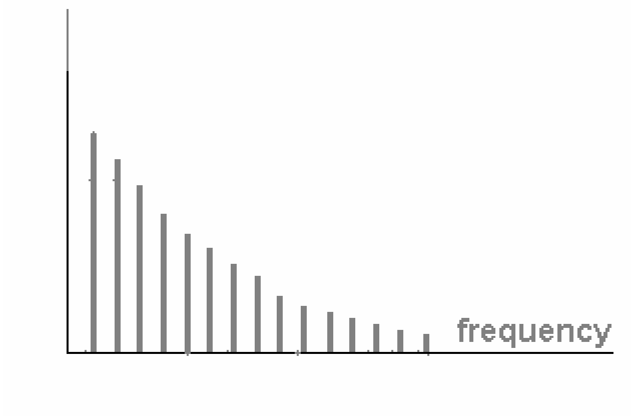


Frequency Spectra

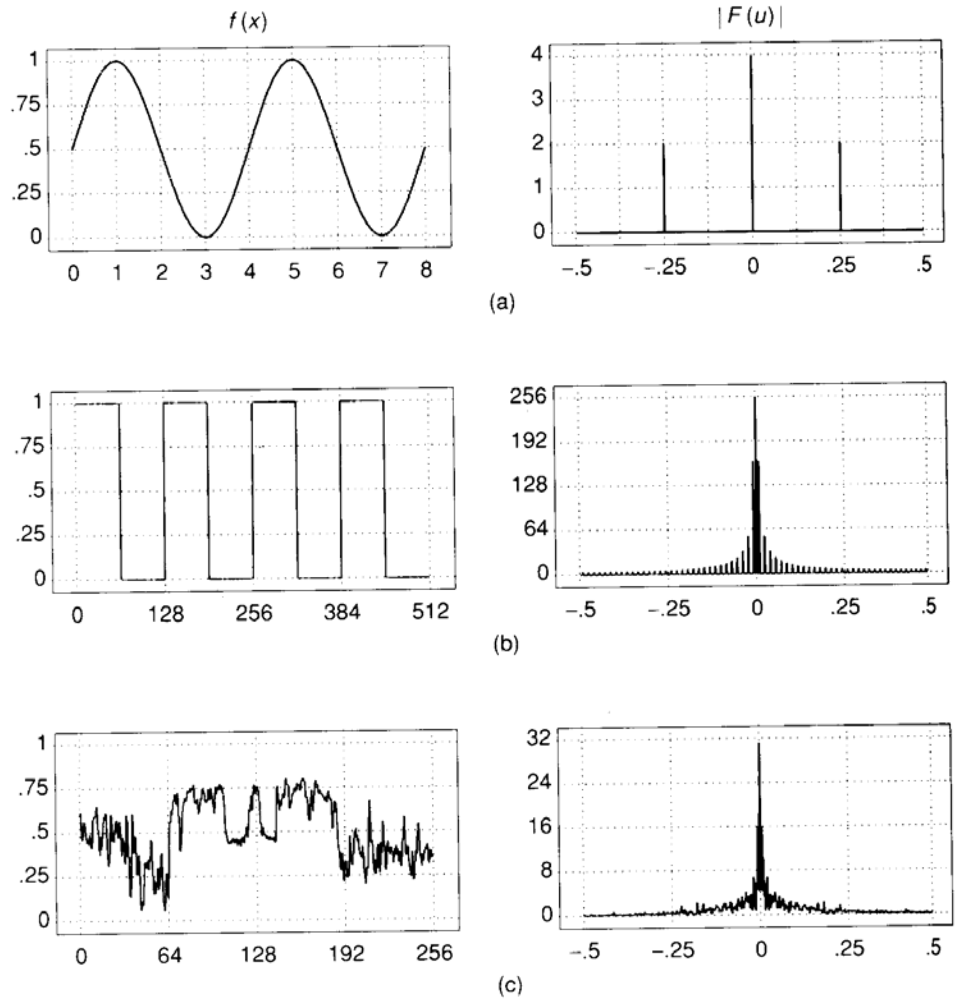


=

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

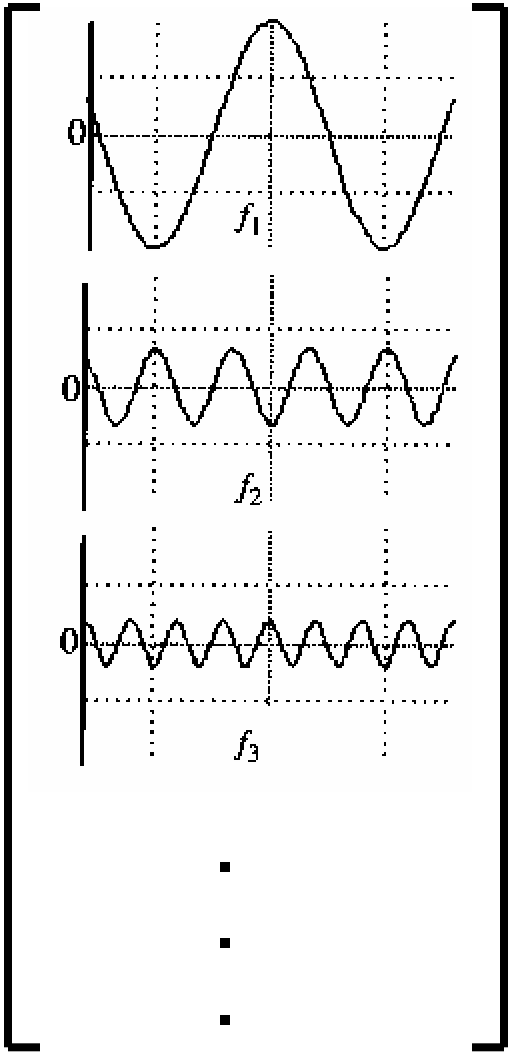


Frequency Spectra

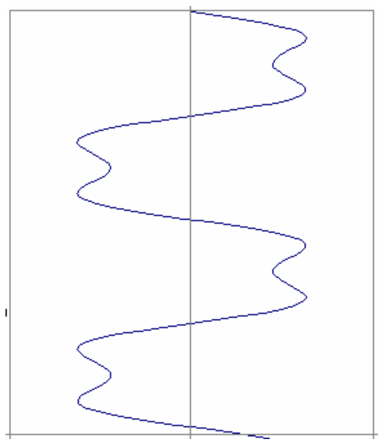


FT: Just a change of basis

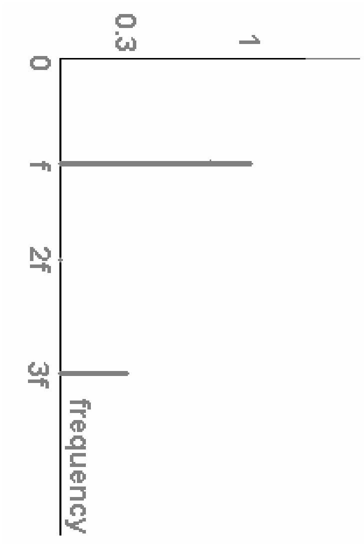
$$M * f(x) = F(\omega)$$



*

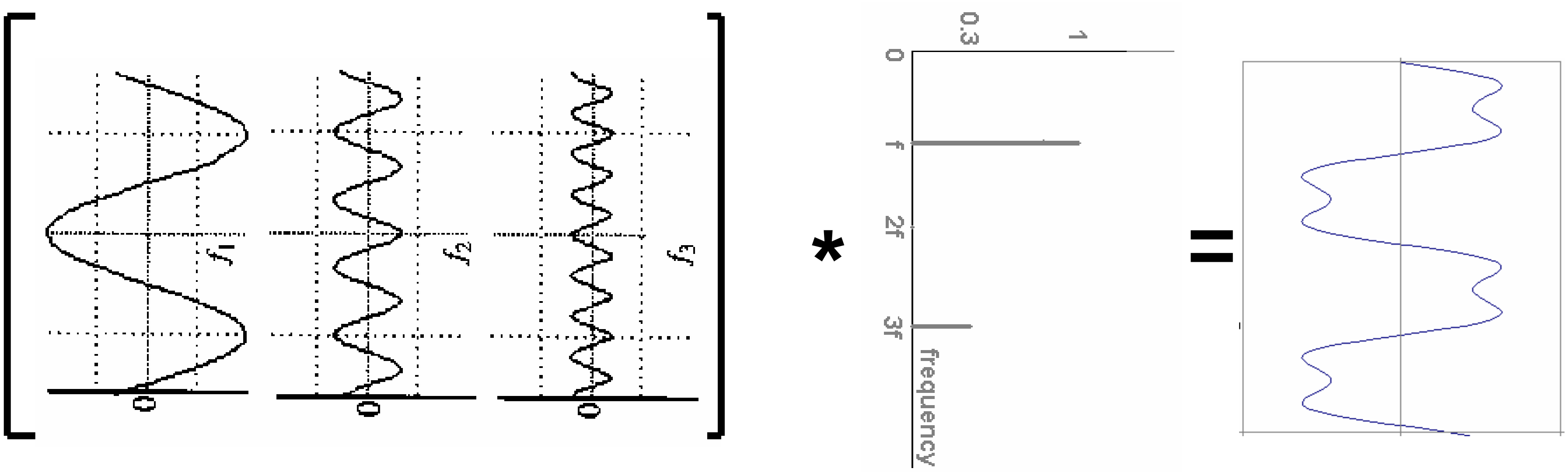


=



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: $e^{i\omega x} = \cos(\omega x) + i \sin(\omega x)$

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

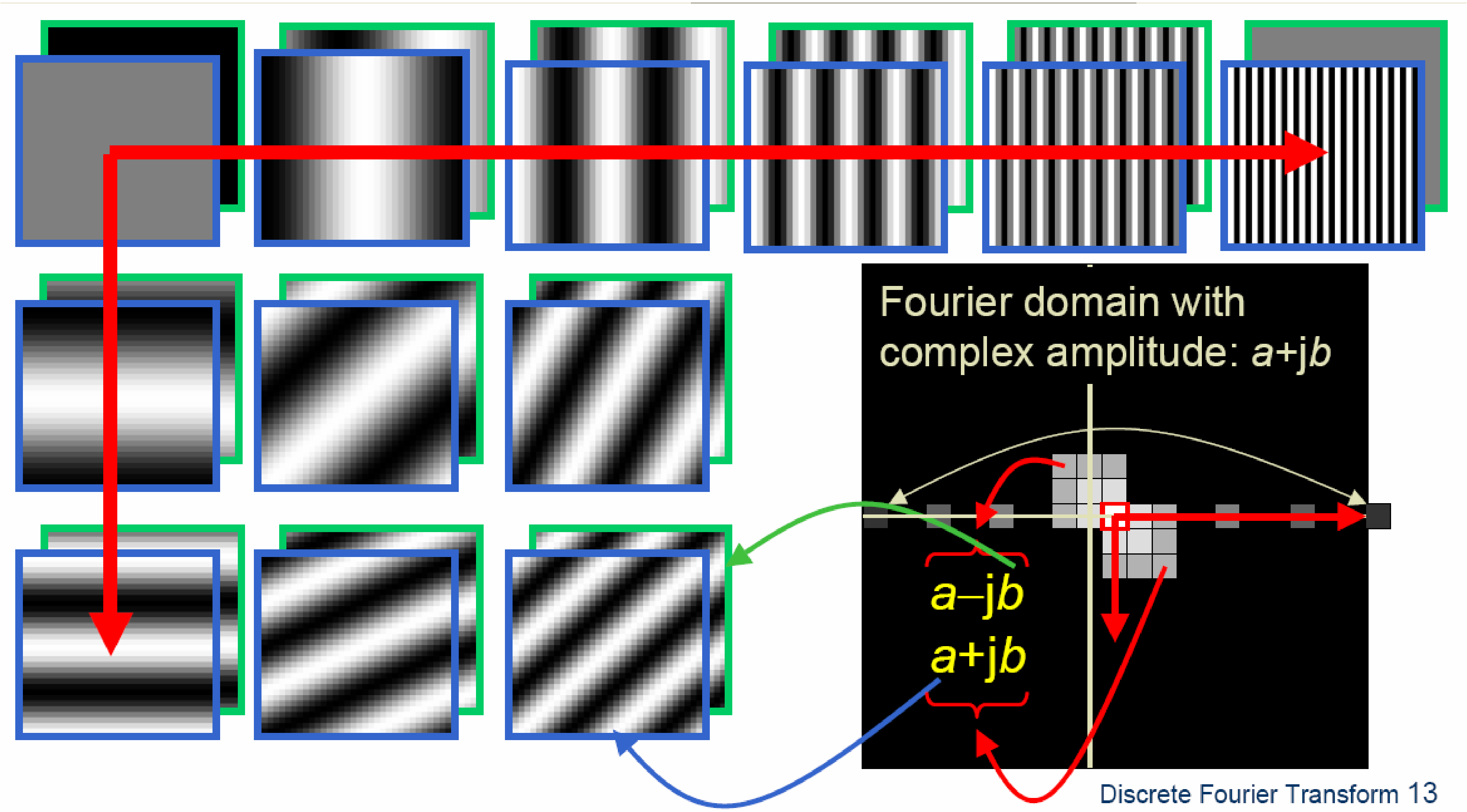
phase can be encoded by sin/cos pair $\rightarrow$

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

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

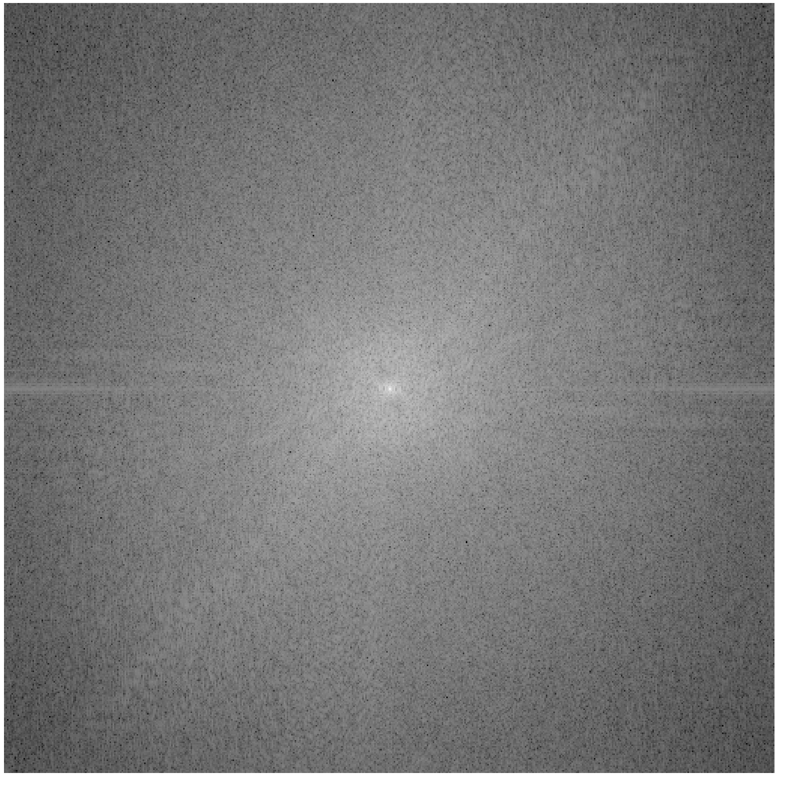
So it's just our signal *f(x)* times sine at frequency ω

Extension to 2D

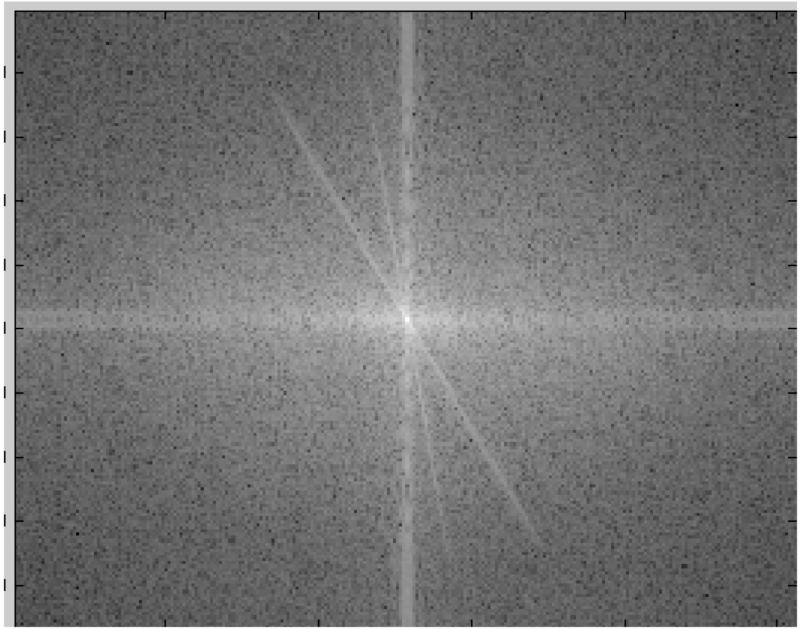


in Matlab, check out: imagesc(log(abs(fftshift(fft2(im)))));

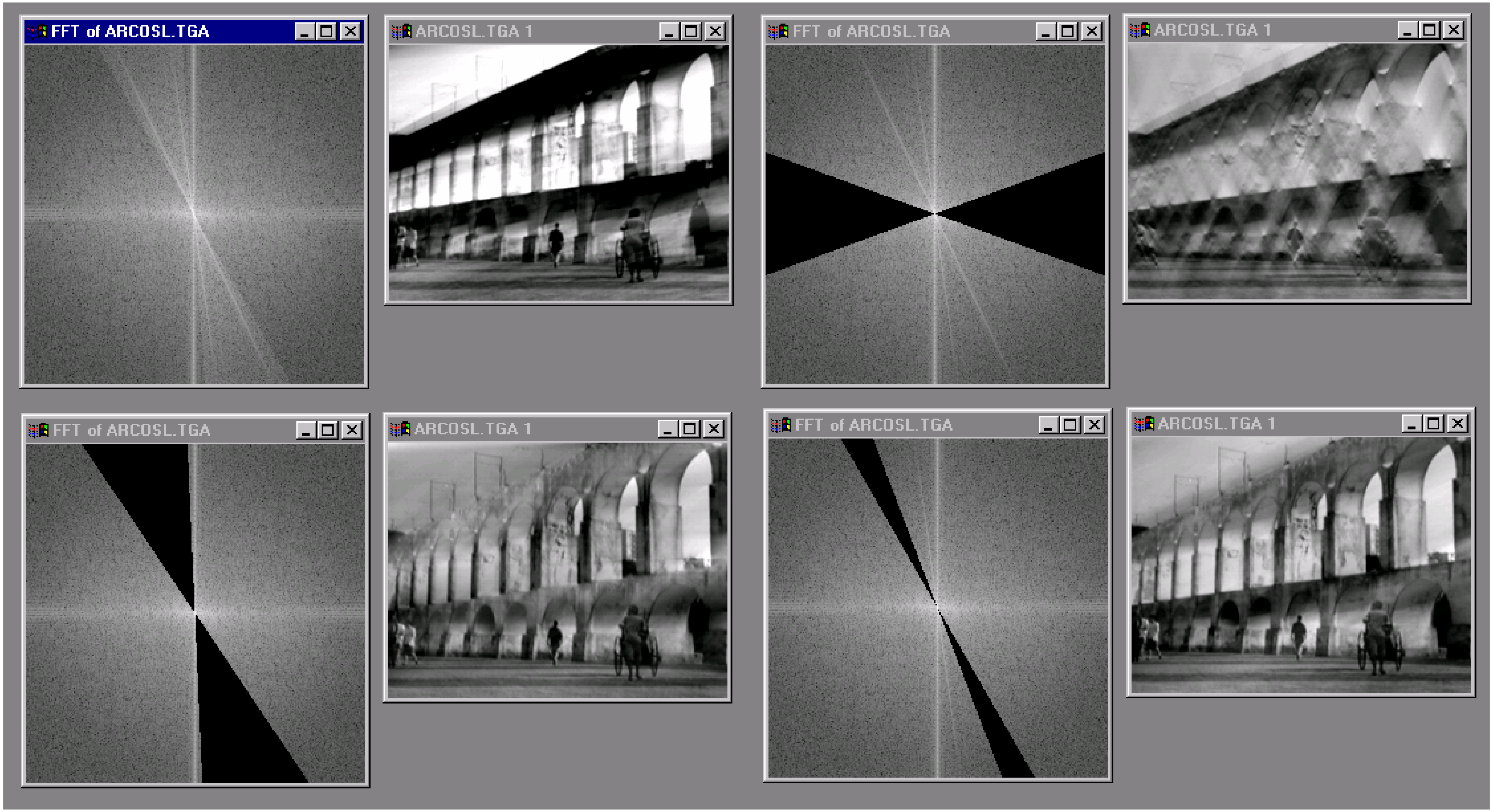
2D FFT transform



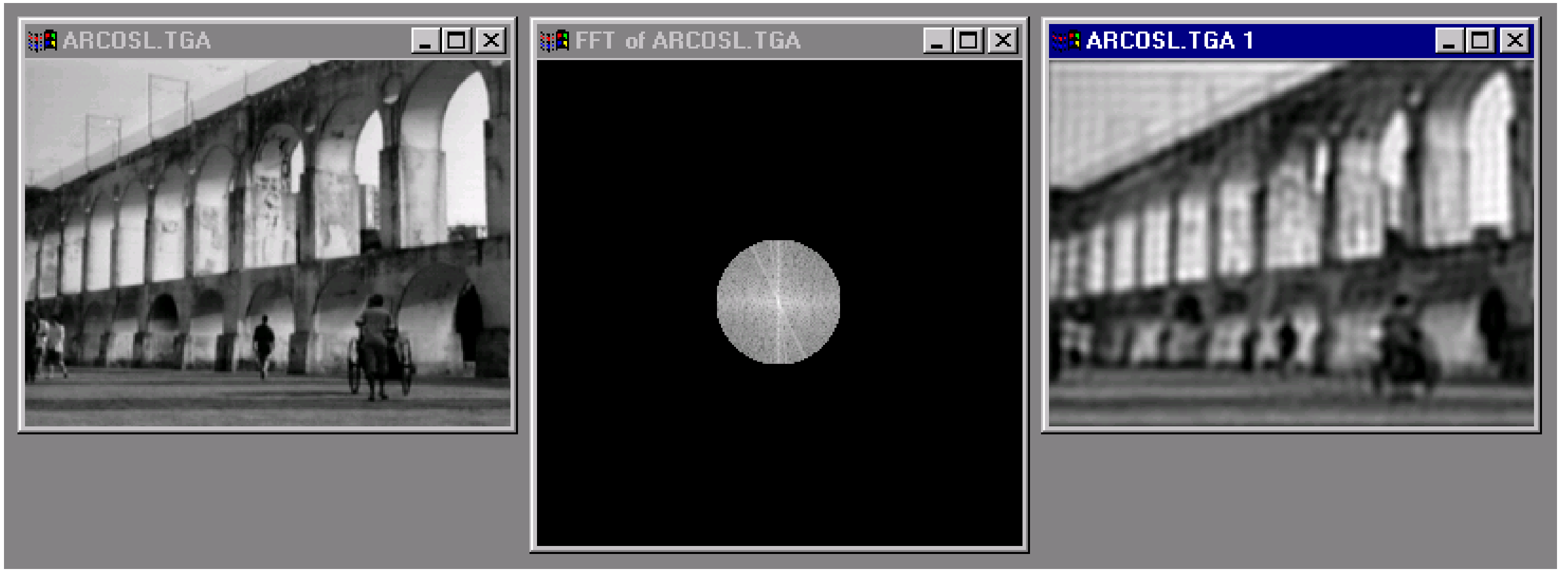
Man-made Scene

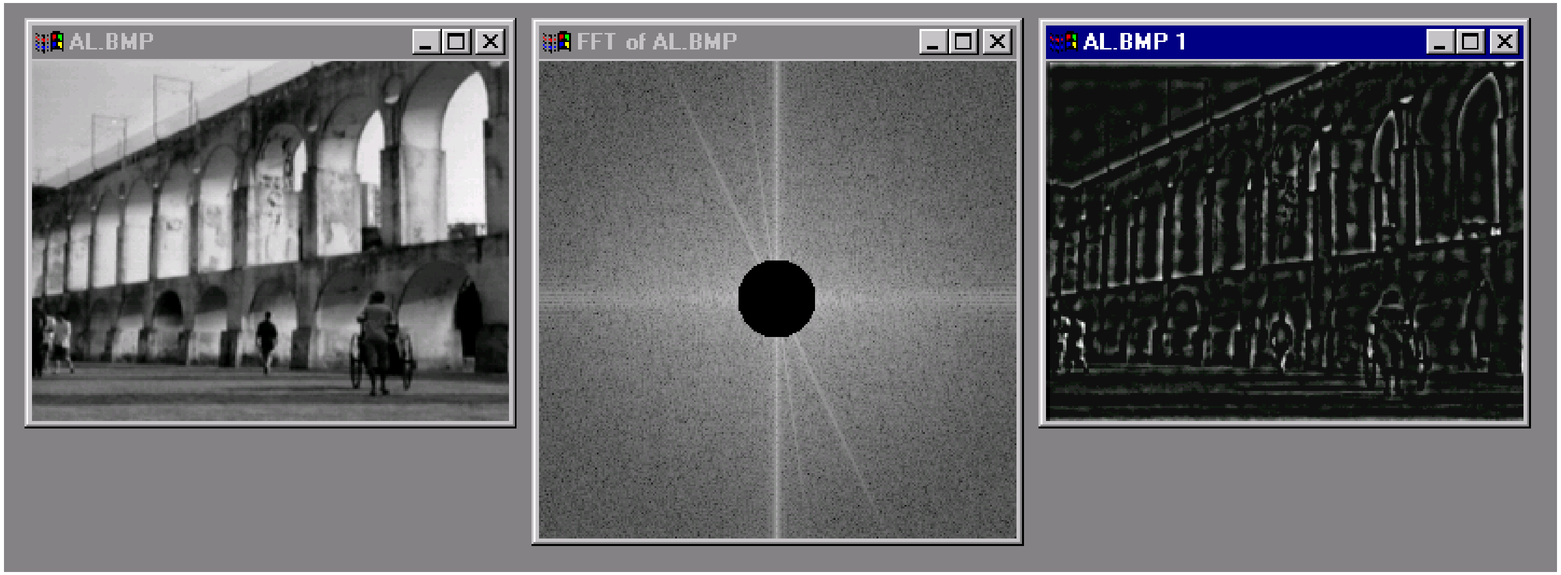


Can change spectrum, then reconstruct

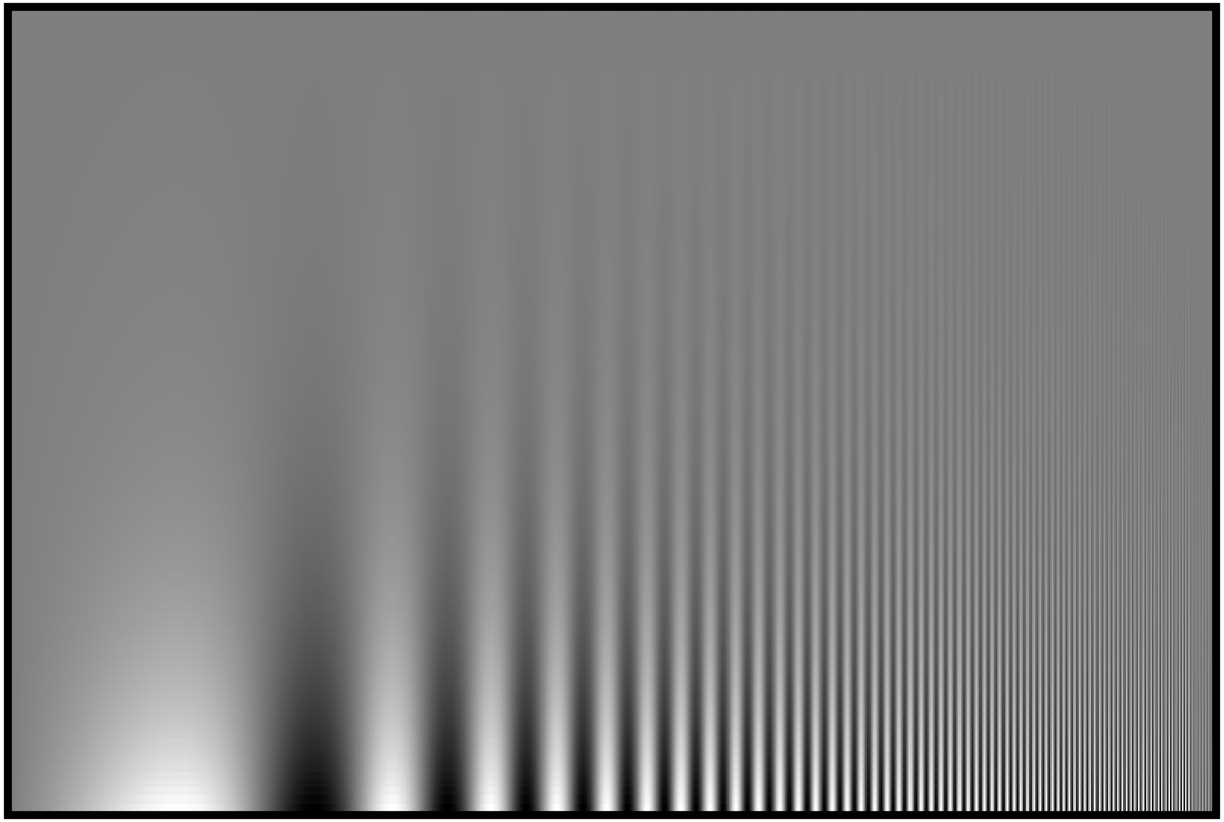


Most information in at low frequencies!



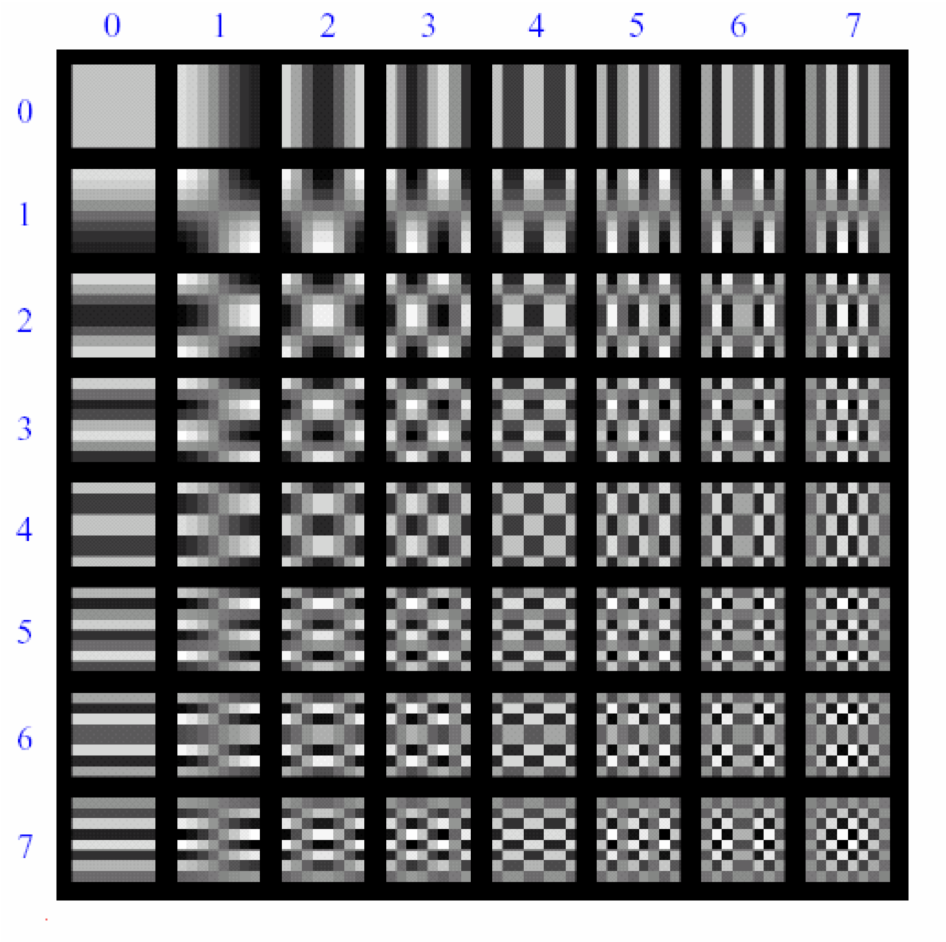


Campbell-Robson contrast sensitivity curve



We don’t resolve high frequencies too well…
… let’s use this to compress images… JPEG!

Lossy Image Compression (JPEG)



Block-based Discrete Cosine Transform (DCT)

Using DCT in JPEG

A variant of discrete Fourier transform

- Real numbers
- Fast implementation

Block size

- small block
  - faster
  - correlation exists between neighboring pixels
- large block
  - better compression in smooth regions

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

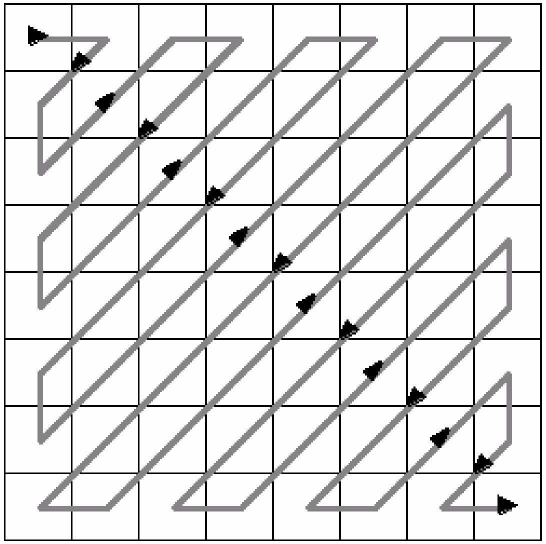


Image compression using DCT

DCT enables image compression by concentrating most image information in the low frequencies

Loose unimportant image info (high frequencies) by cutting $B(u,v)$ at bottom right

The decoder computes the inverse DCT – IDCT

- Quantization Table

3	5	7	9	11	13	15	17
5	7	9	11	13	15	17	19
7	9	11	13	15	17	19	21
9	11	13	15	17	19	21	23
11	13	15	17	19	21	23	25
13	15	17	19	21	23	25	27
15	17	19	21	23	25	27	29
17	19	21	23	25	27	29	31

JPEG compression comparison

89k

12k