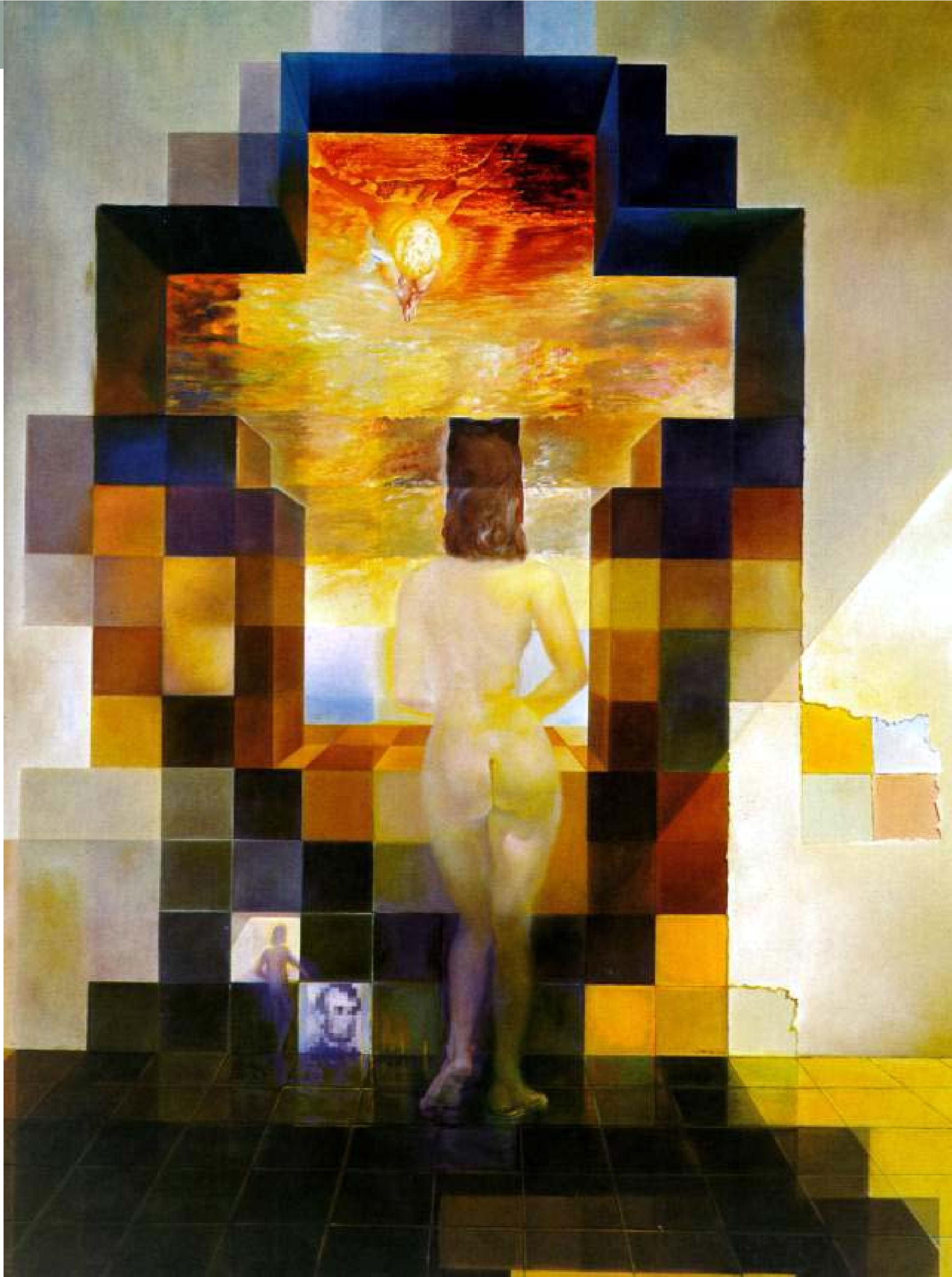
CS 4495 Computer Vision

Frequency and Fourier Transforms

Aaron Bobick
School of Interactive Computing

Administrivia

- Project 1 is (still) on line – get started now!
- Readings for this week: FP Chapter 4 (which includes reviewing 4.1 and 4.2)



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

Decomposing an image

- A basis set is (edit from to Wikipedia):
  - A **basis** *B* of a vector space *V* is a linearly independent subset of *V* that spans *V*.
  - In more detail:suppose that $B = \{ v_1, \ldots, v_n \}$ is a finite subset of a vector space *V* over a field **F** (such as the real or complex numbers **R** or **C**). Then *B* is a basis if it satisfies the following conditions:
    - the *linear independence* property:
      - for all $a_1, \ldots, a_n \in$ **F**, if $a_1v_1 + \ldots + a_nv_n = 0$, then necessarily $a_1 = \ldots = a_n = 0$;
    - and the *spanning* property,
      - for every *x* in *V* it is possible to choose $a_1, \ldots, a_n \in$ **F** such that $x = a_1v_1 + \ldots + a_nv_n$.
  - *Not necessarily orthogonal….*

- If we have a basis set for images, could perhaps be useful for analysis – especially for linear systems because we could consider each basis component independently. *(Why?)*

Images as points in a vector space

- Consider an image as a point in a NxN size space – can rasterize into a single vector $[x_{00} x_{10} x_{20} \ldots x_{(n-1)0} x_{10} \ldots x_{(n-1)(n-1)}]^T$
- The “normal” basis is just the vectors:

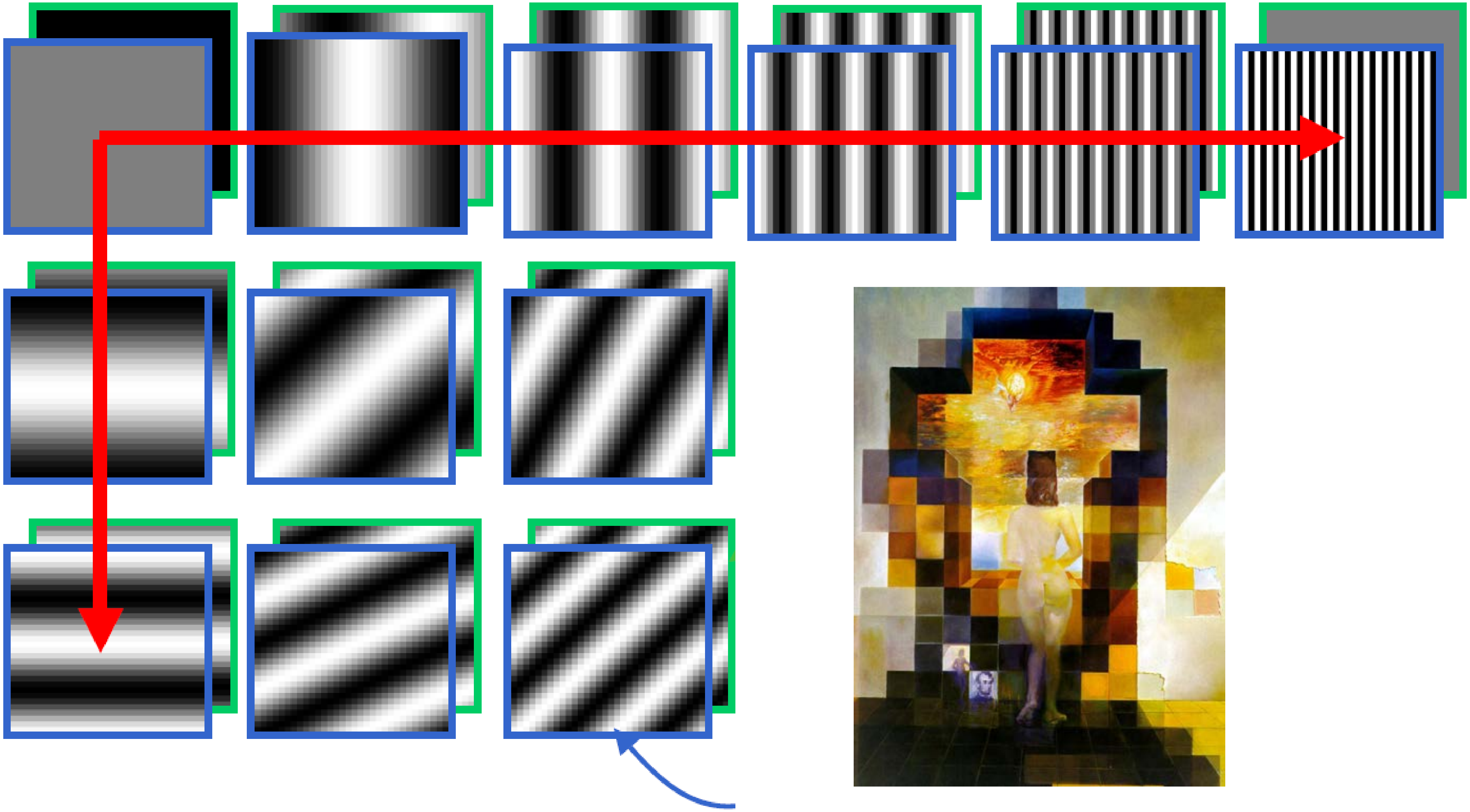
$$[0\;0\;0\;0 \ldots 0 1\;0\;0\;0 \ldots\;0]^T$$

  - Independent
  - Can create any image

- But not very helpful to consider how each pixel contributes to computations.

A nice set of basis

Teases away fast vs. slow changes in the image.



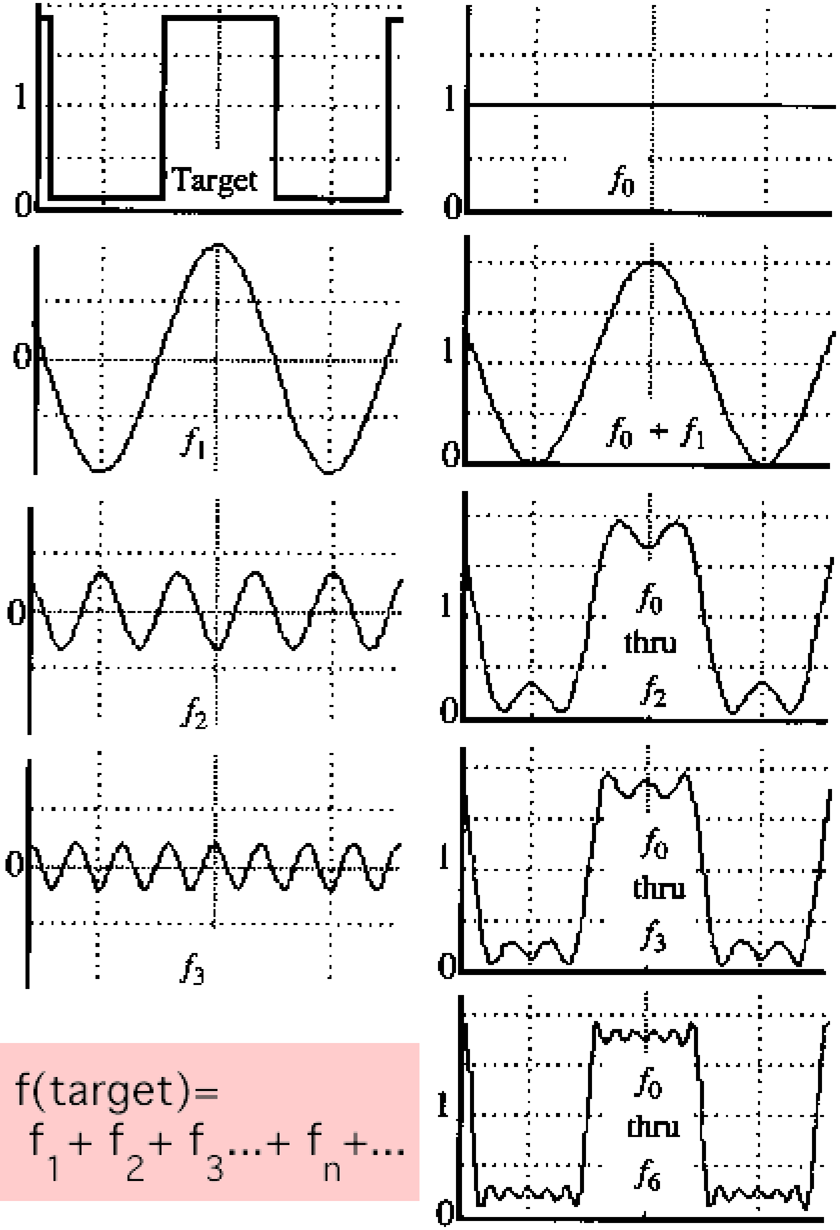
This change of basis has a special name…

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

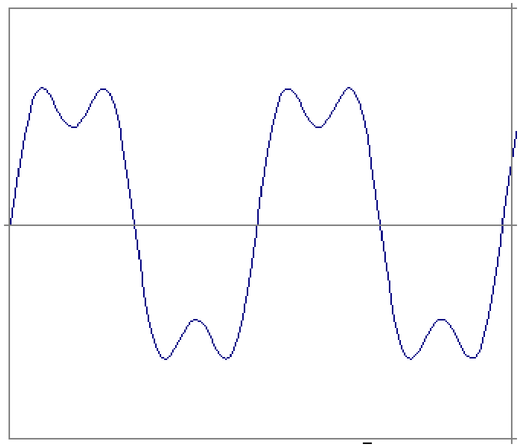
- Our building block:
- $$A\sin(\omega x+\phi)$$
- Add enough of them to get any signal *f(x)* you want!
- How many degrees of freedom?
- What does each control?
- Which one encodes the coarse vs. fine structure of the signal?



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

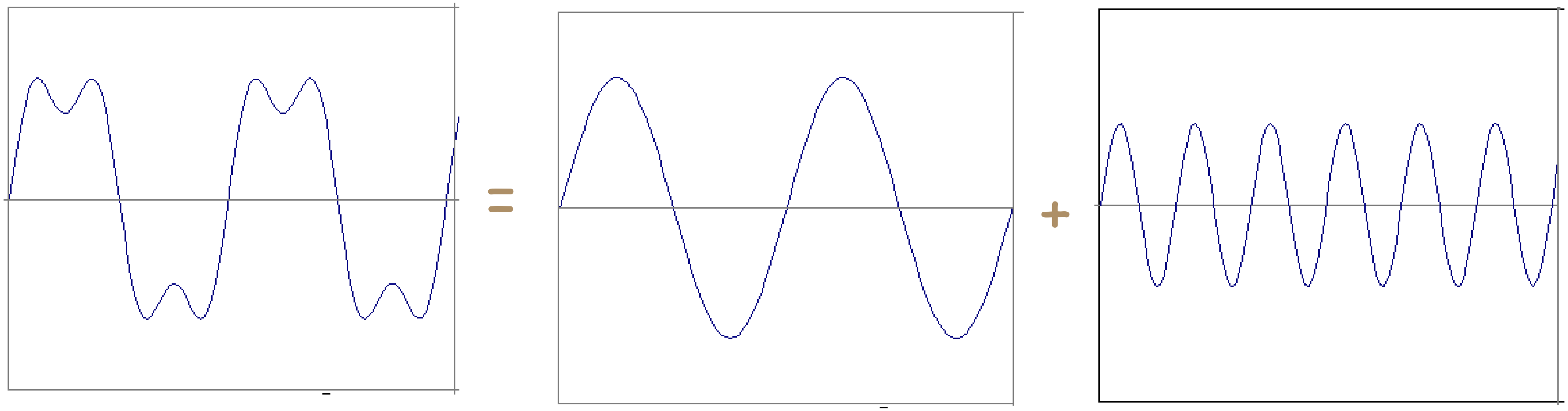
Time and Frequency

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



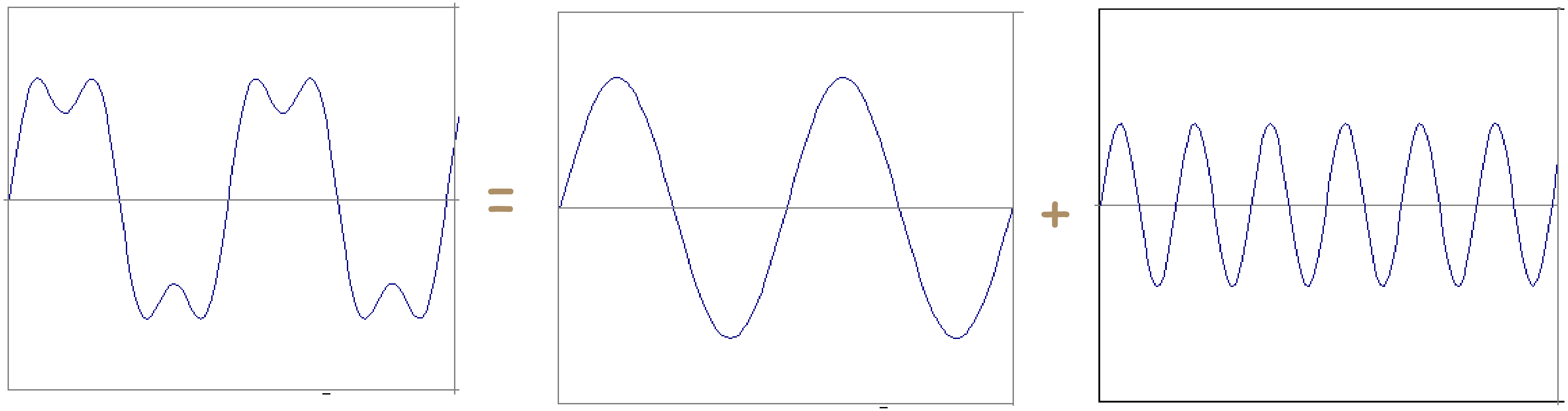
Time and Frequency

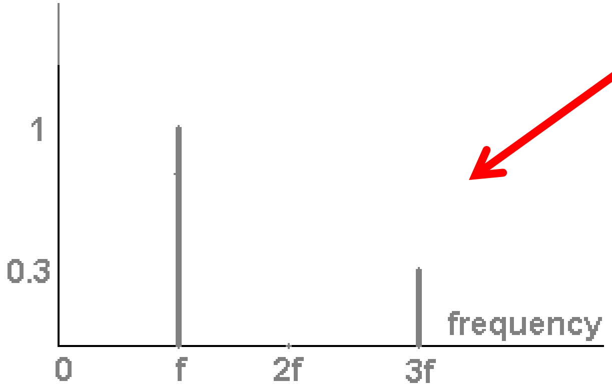
- example : $g(t) = \sin(2pf\ t) + (1/3)\sin(2p(3f)\ t)$



Frequency Spectra - Series

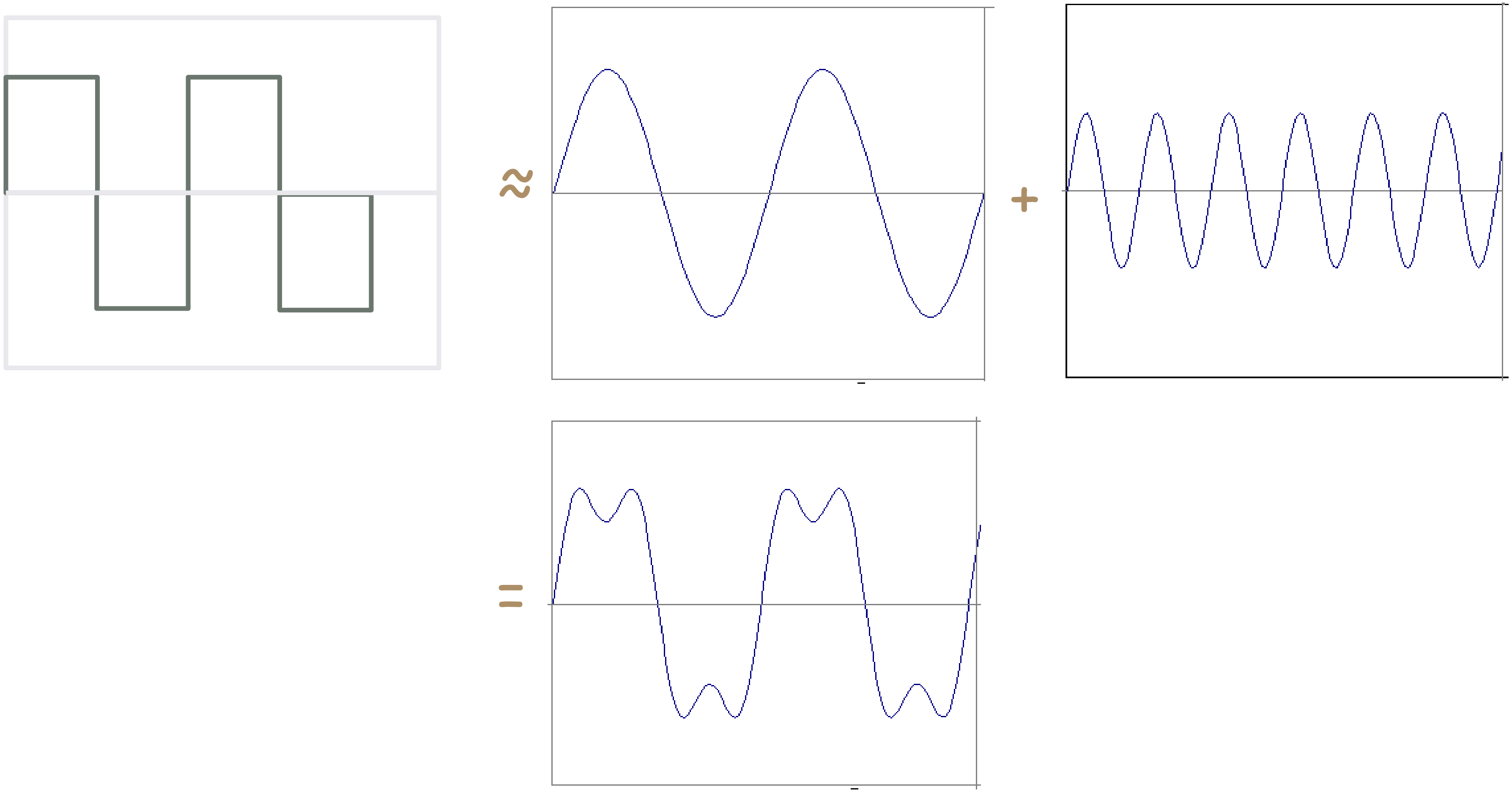
- example : $g(t) = \sin(2pf\ t) + (1/3)\sin(2p(3f)\ t)$



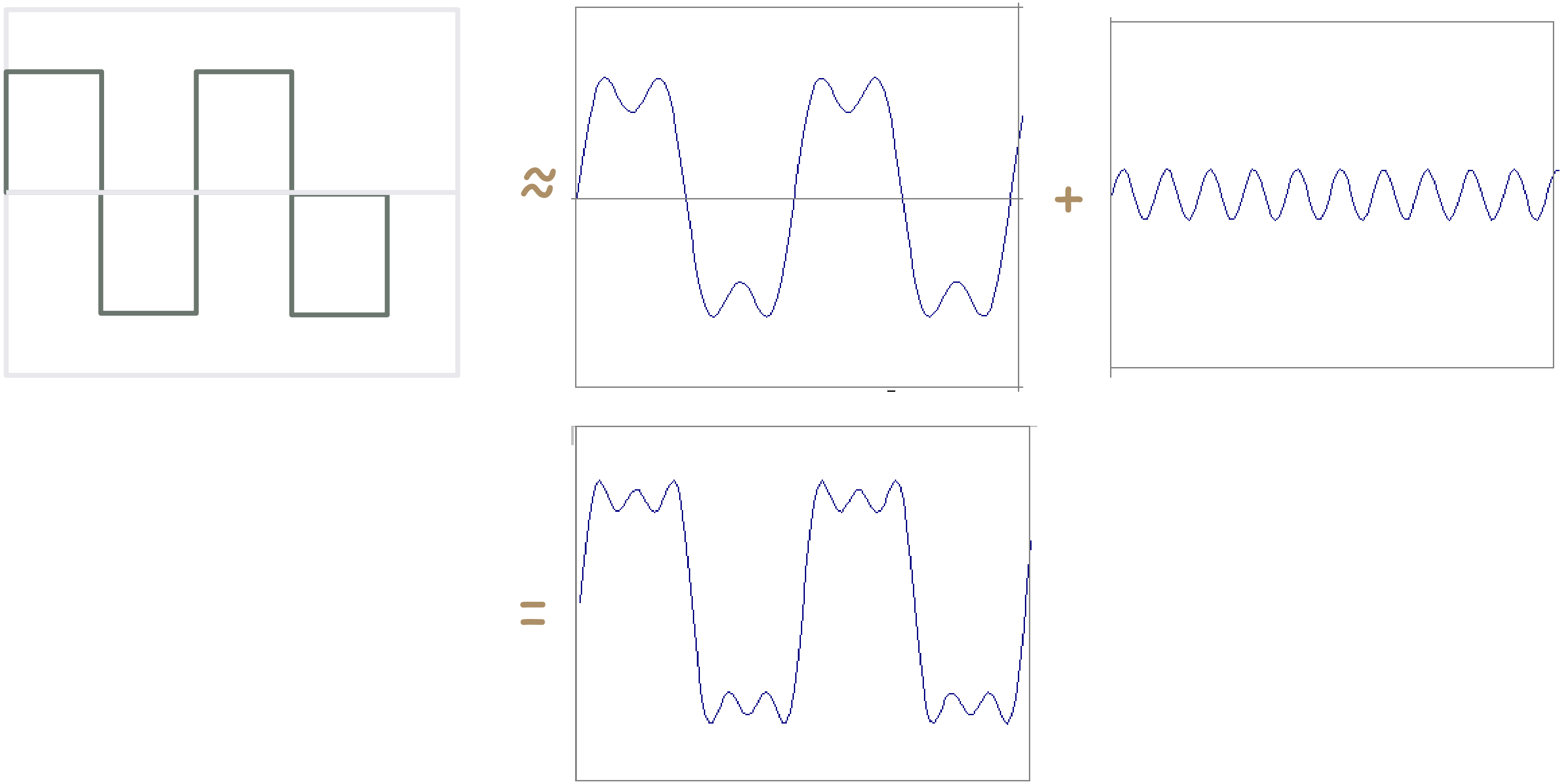


One form of *spectrum* – more in a bit

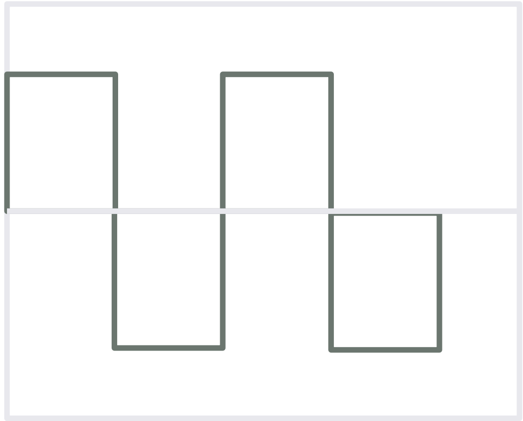
Frequency Spectra - Series



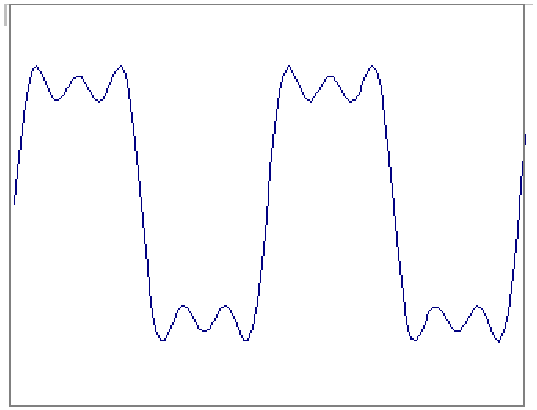
Frequency Spectra - Series



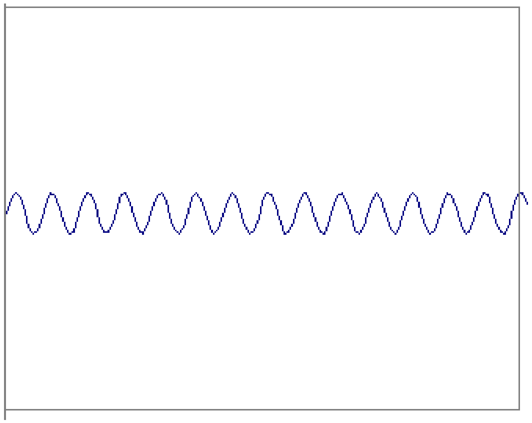
Frequency Spectra - Series



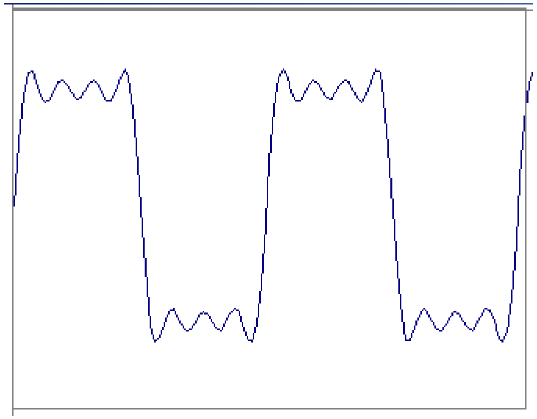
≈



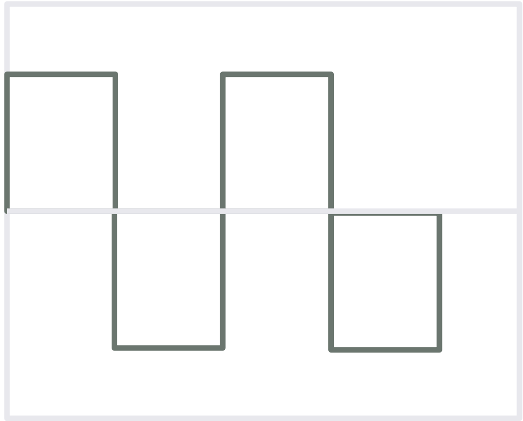
+



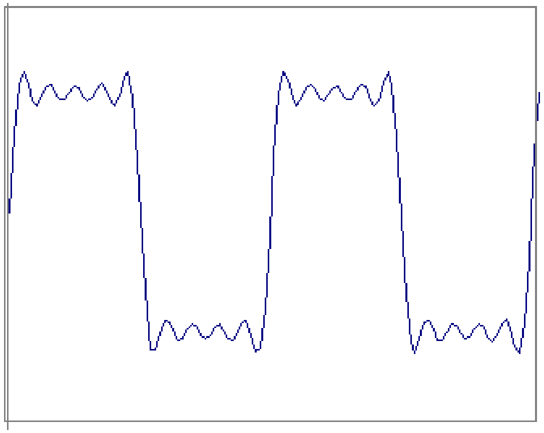
=



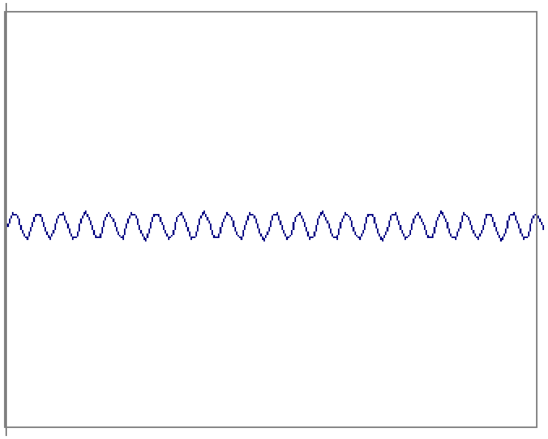
Frequency Spectra - Series



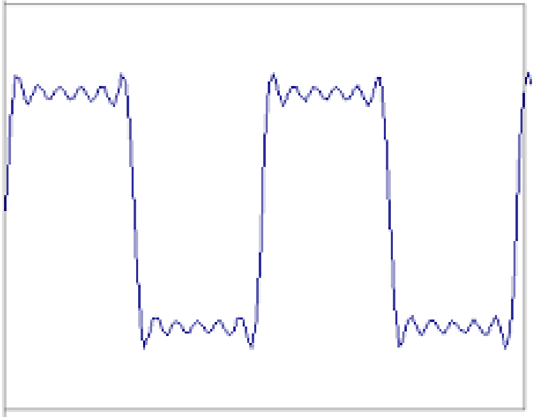
≈



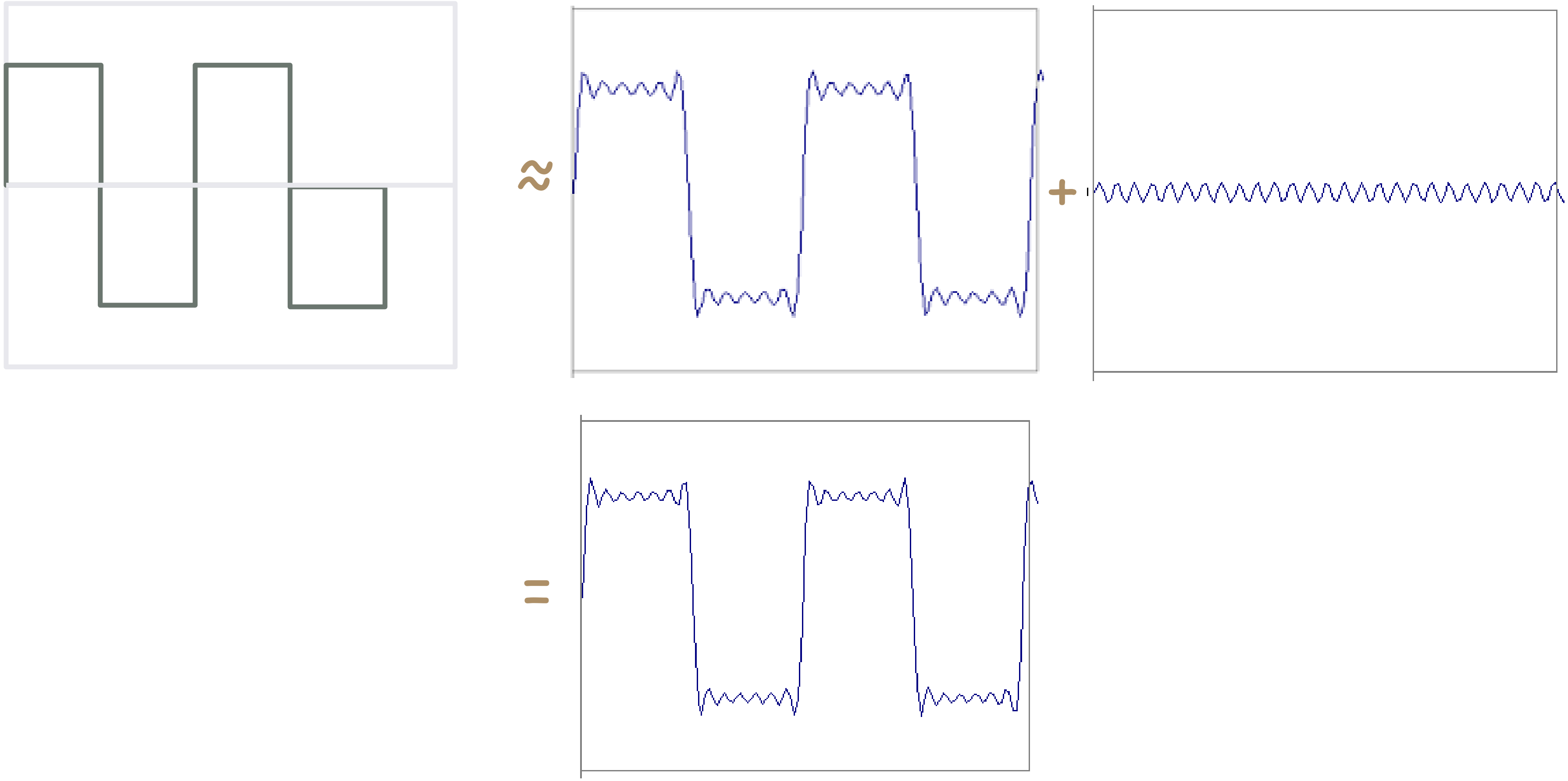
+



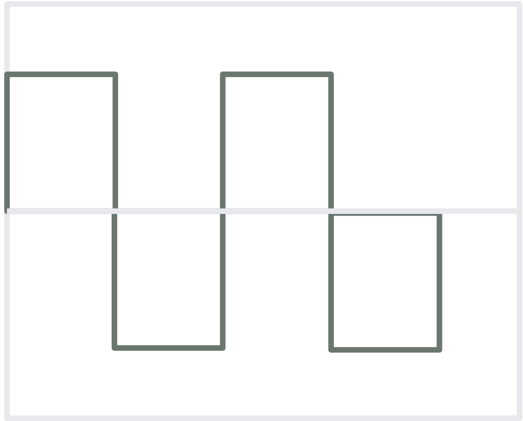
=



Frequency Spectra - Series

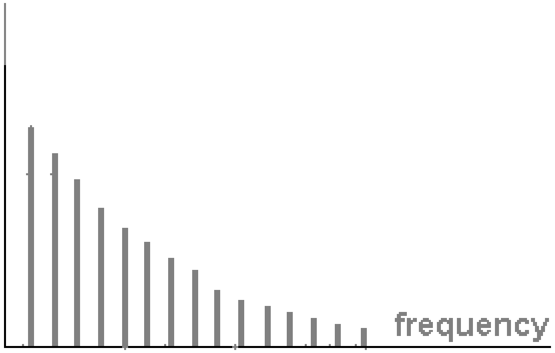


Frequency Spectra - Series



=

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



Usually, frequency is more interesting than the phase for CV because we’re not reconstructing the image

Fourier *Transform*

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

$$\boldsymbol{f(x)} \longrightarrow \boxed{\text{Fourier Transform}} \xrightarrow{A\sin(\omega x+\phi)} \boldsymbol{F(\omega)}$$

For every ω from 0 to inf (actually –inf to inf), ***F(ω)*** holds the amplitude *A* and phase ϕ of the corresponding sine

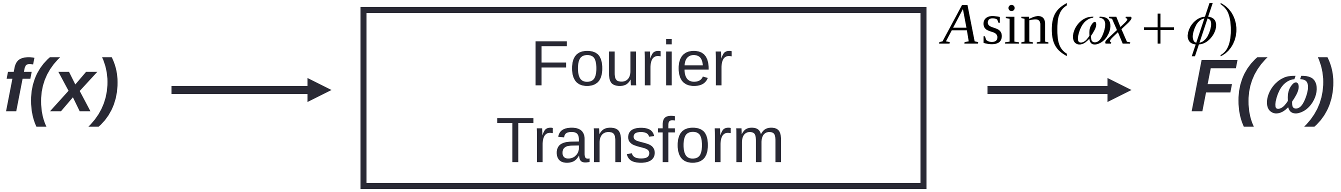
- How can *F* hold both? Complex number trick!

Recall : $e^{ik} = \underset{\textit{Even}}{\cos k} + \underset{\textit{Odd}}{i\sin k} \qquad i = \sqrt{-1}\ \boxed{(\text{or } j)}$

Matlab sinusoid demo…

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 → $A\sin(\omega x+\phi)$ **F(ω)**

For every ω from 0 to inf, (actually –inf to inf), ***F(ω)*** holds the amplitude *A* and phase ϕ of the corresponding sine

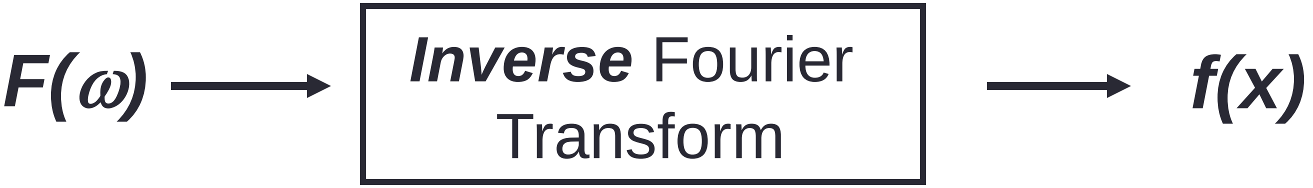
- How can *F* hold both? Complex number trick!

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

$$F(\omega)=\underset{\textit{Even}}{R(\omega)}+i\underset{\textit{Odd}}{I(\omega)}$$

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

And we can go back:



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

Computing FT: Just a basis

- The infinite integral of the product of two sinusoids of *different* frequency is zero. (Why?)

$$\int_{-\infty}^{\infty} \sin(ax+\phi)\sin(bx+\varphi)dx = 0, \text{ if } a \neq b$$

- And the integral is infinite if equal (unless exactly out of phase):

$$\int_{-\infty}^{\infty} \sin(ax+\phi)\sin(ax+\varphi)dx = \pm\infty$$

If ϕ and φ not exactly pi/2 out of phase (sin and cos).

Computing FT: Just a basis

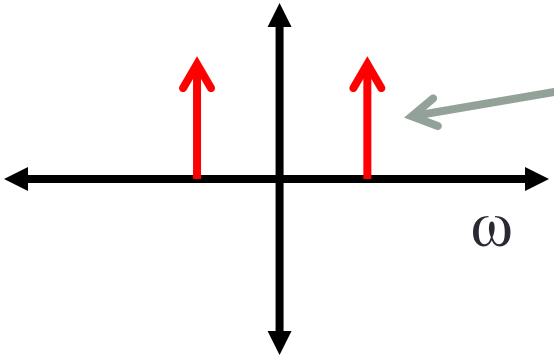
- So, suppose f(x) is a cosine wave of freq ω:

$$f(x) = \cos(2\pi\omega x)$$

- Then:

$$C(u) = \int_{-\infty}^{\infty} f(x)\cos(2\pi u\,x)dx$$

Is infinite if *u* is equal to ω (or - ω) and zero otherwise:



Impulse

Computing FT: Just a basis

- We can do that for all frequencies *u.*
- But we’d have to do that for all *phases*, don’t we???
- No! Any phase can be created by a weighted sum of cosine and sine. Only need each piece:

$$C(u) = \int_{-\infty}^{\infty} f(x)\cos(2\pi u x)dx$$

$$S(u) = \int_{-\infty}^{\infty} f(x)\sin(2\pi u x)dx$$

- Or…

Fourier Transform – more formally

Represent the signal as an infinite weighted sum of an infinite number of sinusoids

$$F(u)=\int_{-\infty}^{\infty} f(x)e^{-i2\pi ux}dx$$

Again: $e^{ik}=\cos k+i\sin k \qquad i=\sqrt{-1}$

Spatial Domain (*x*) → Frequency Domain (*u or s*)

(Frequency Spectrum *F(u)*)

Inverse Fourier Transform (IFT) – add up all the sinusoids at x:

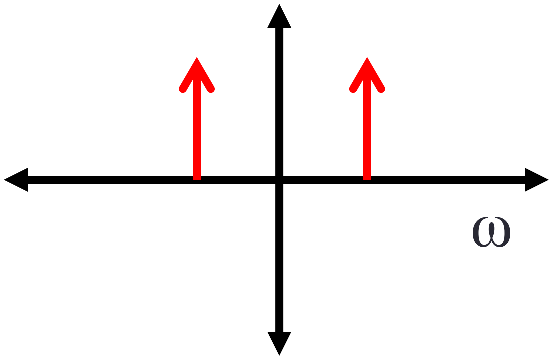
$$f(x)=\int_{-\infty}^{\infty} F(u)e^{i2\pi ux}du$$

Frequency Spectra – Even/Odd

Frequency actually goes from –inf to inf.

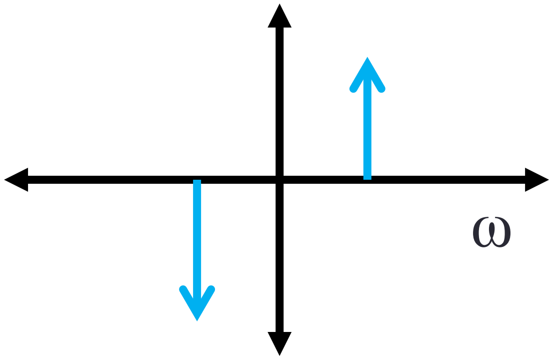
Sinusoid example:

Even (cos)



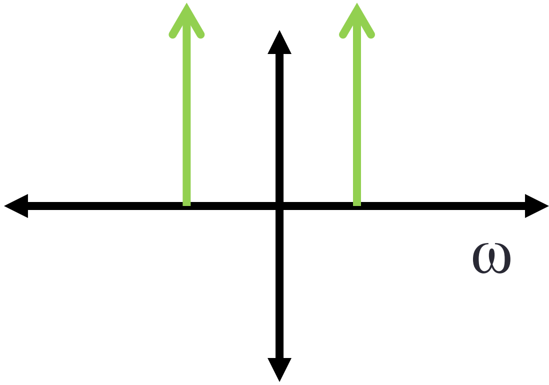
Real

Odd (sin)



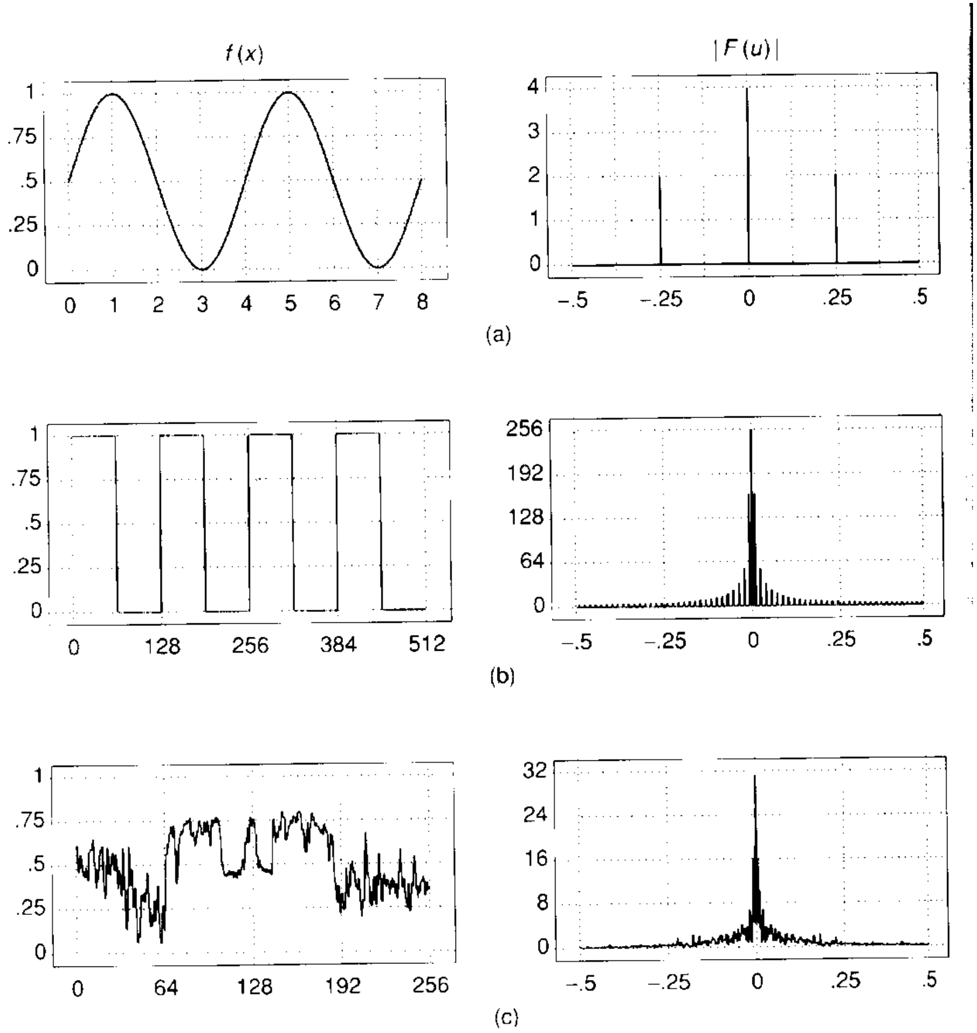
Imaginary

Magnitude

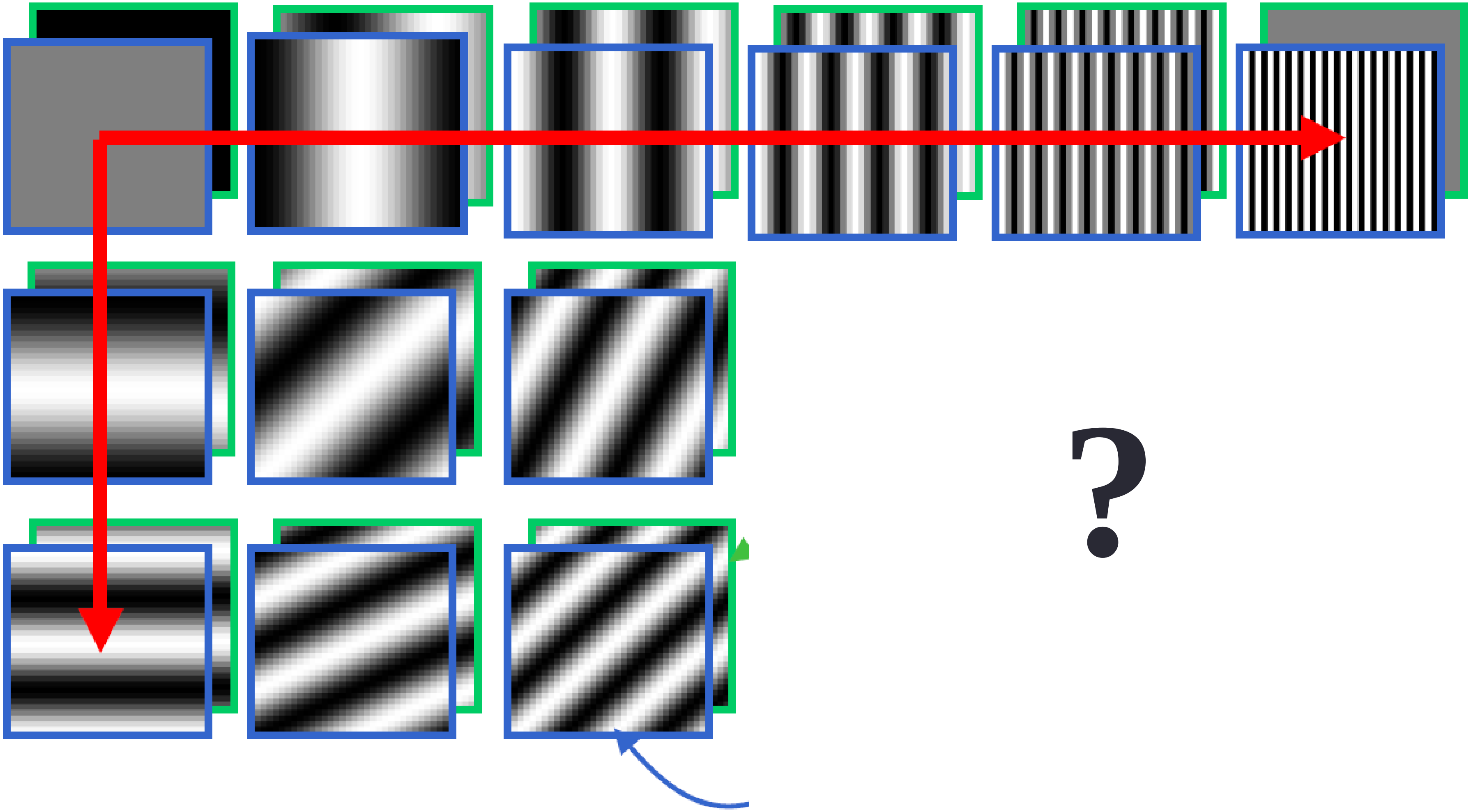


Power

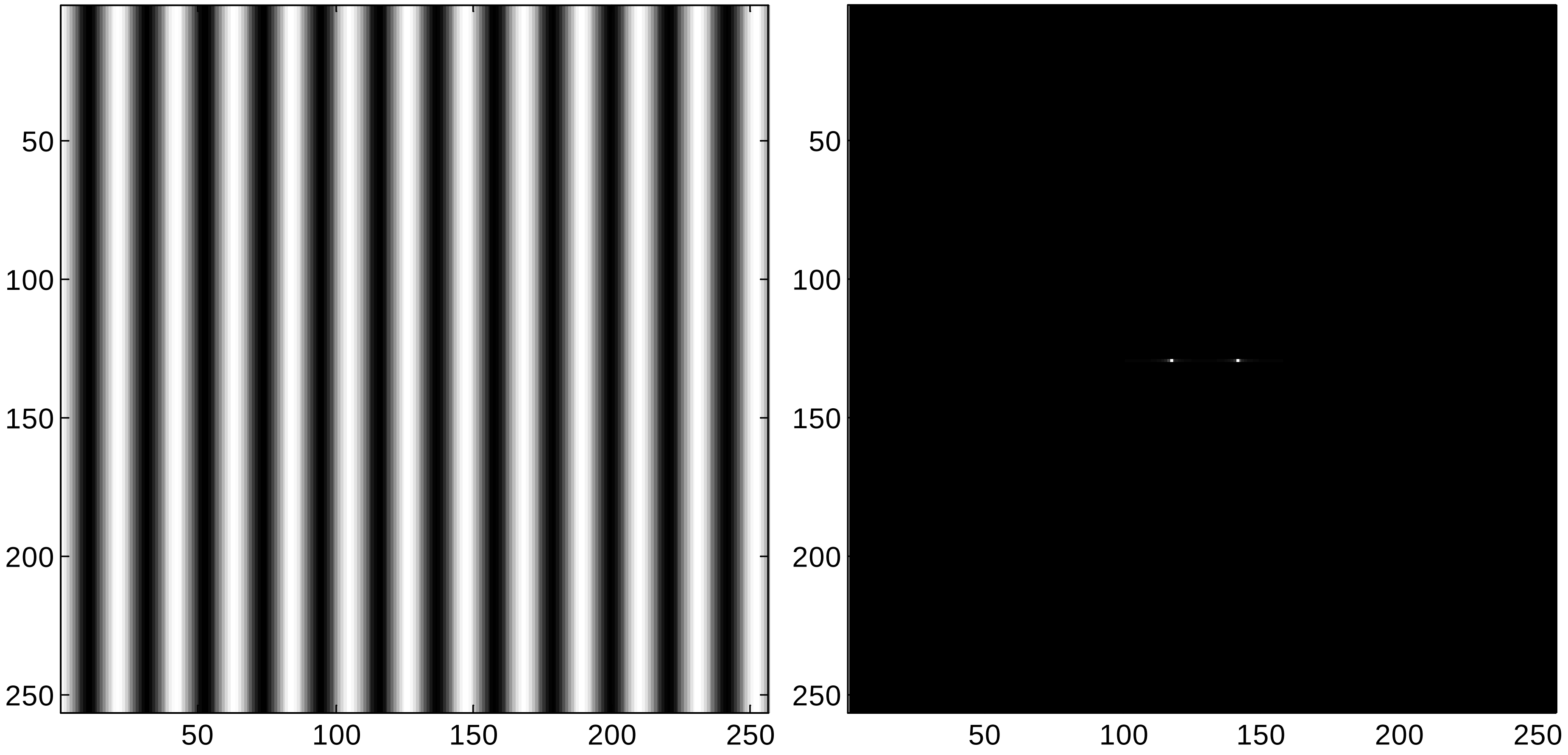
Frequency Spectra



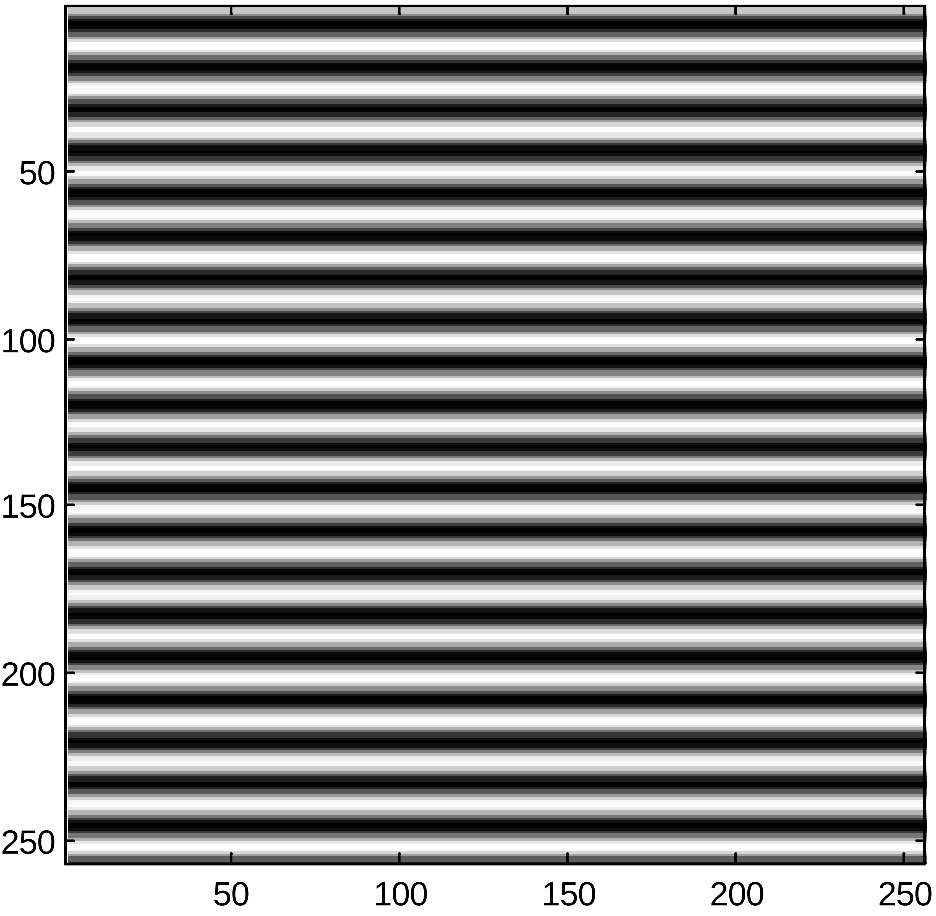
Extension to 2D

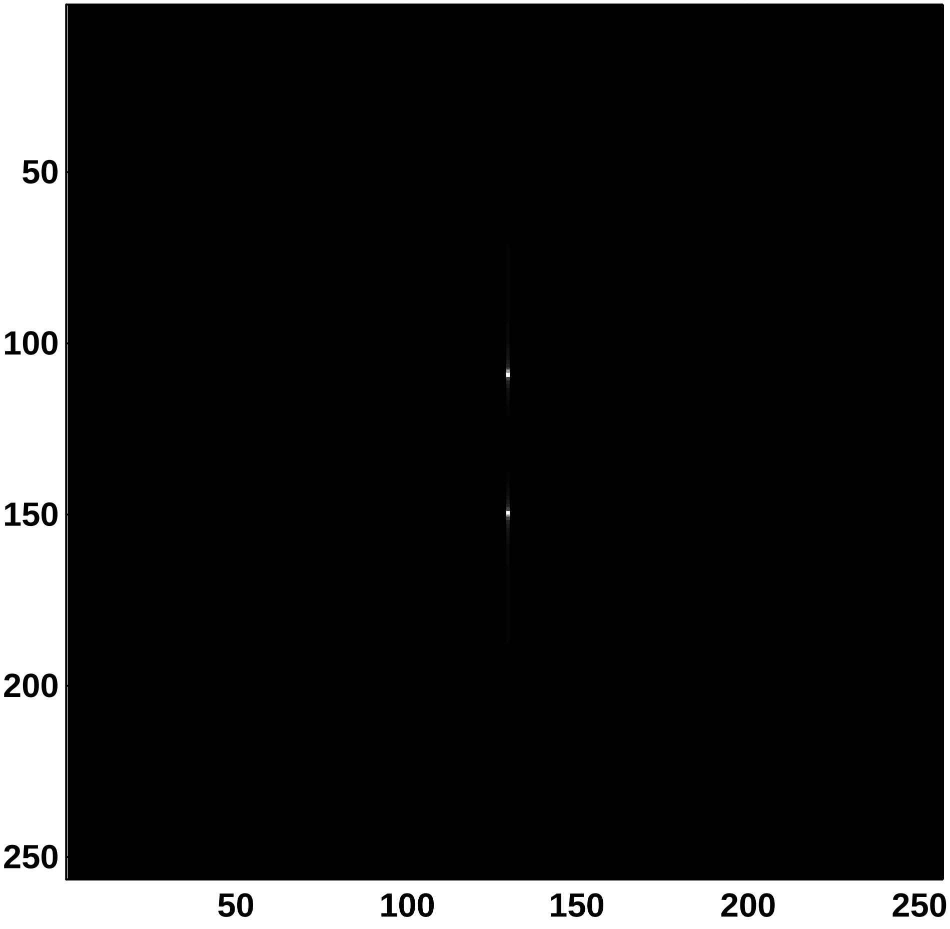


2D Examples – sinusoid magnitudes

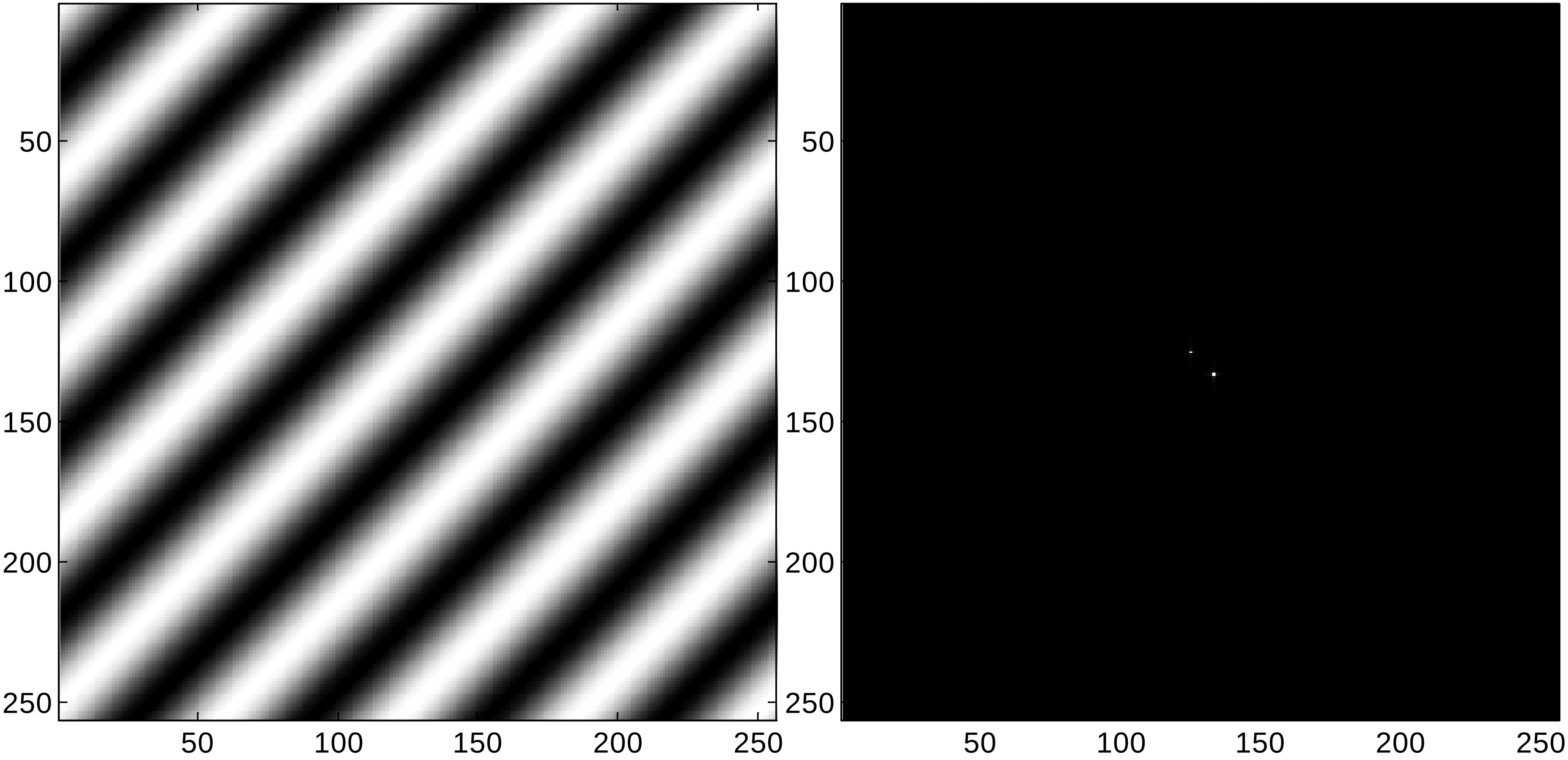


2D Examples – sinusoid magnitudes

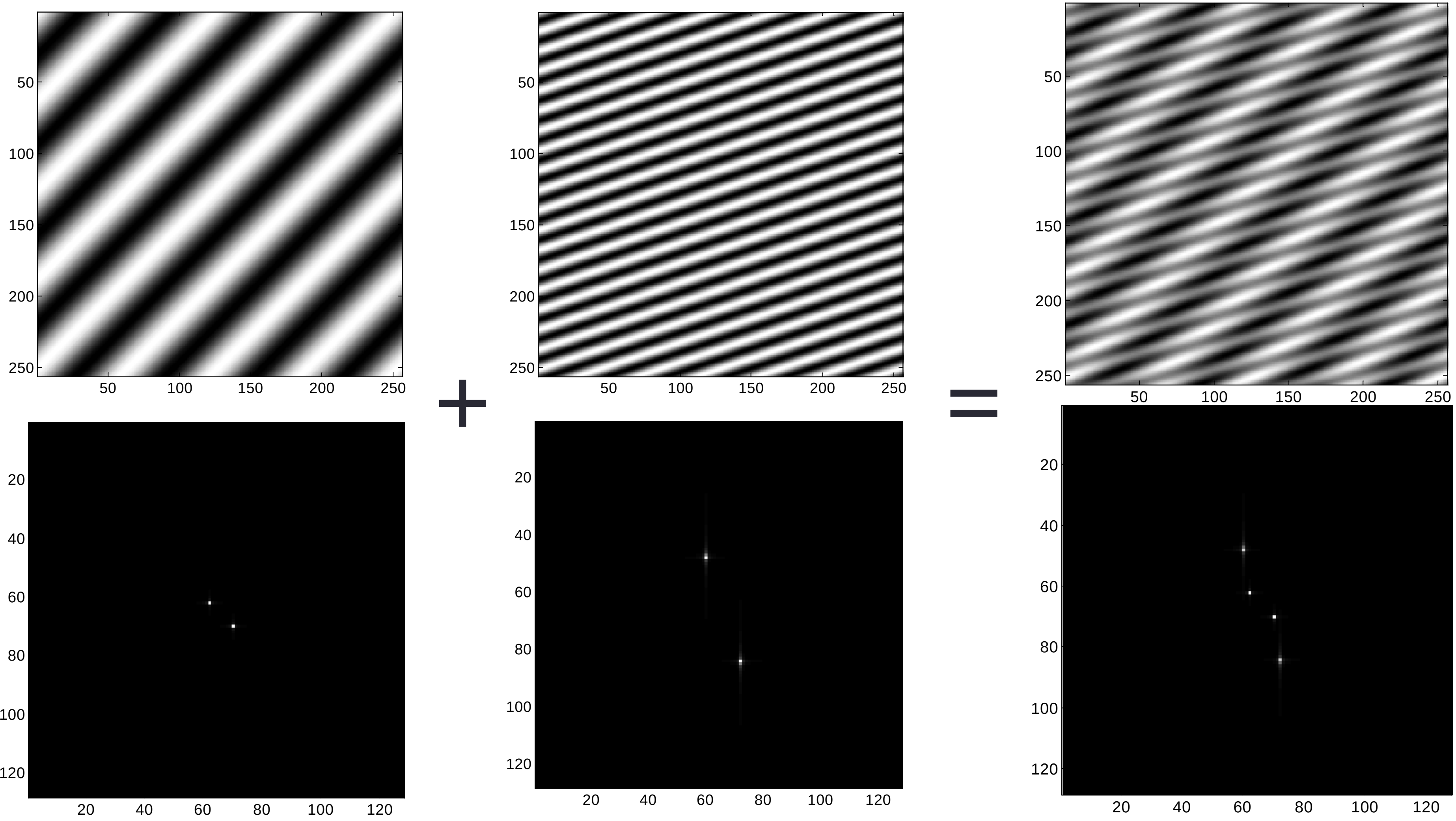




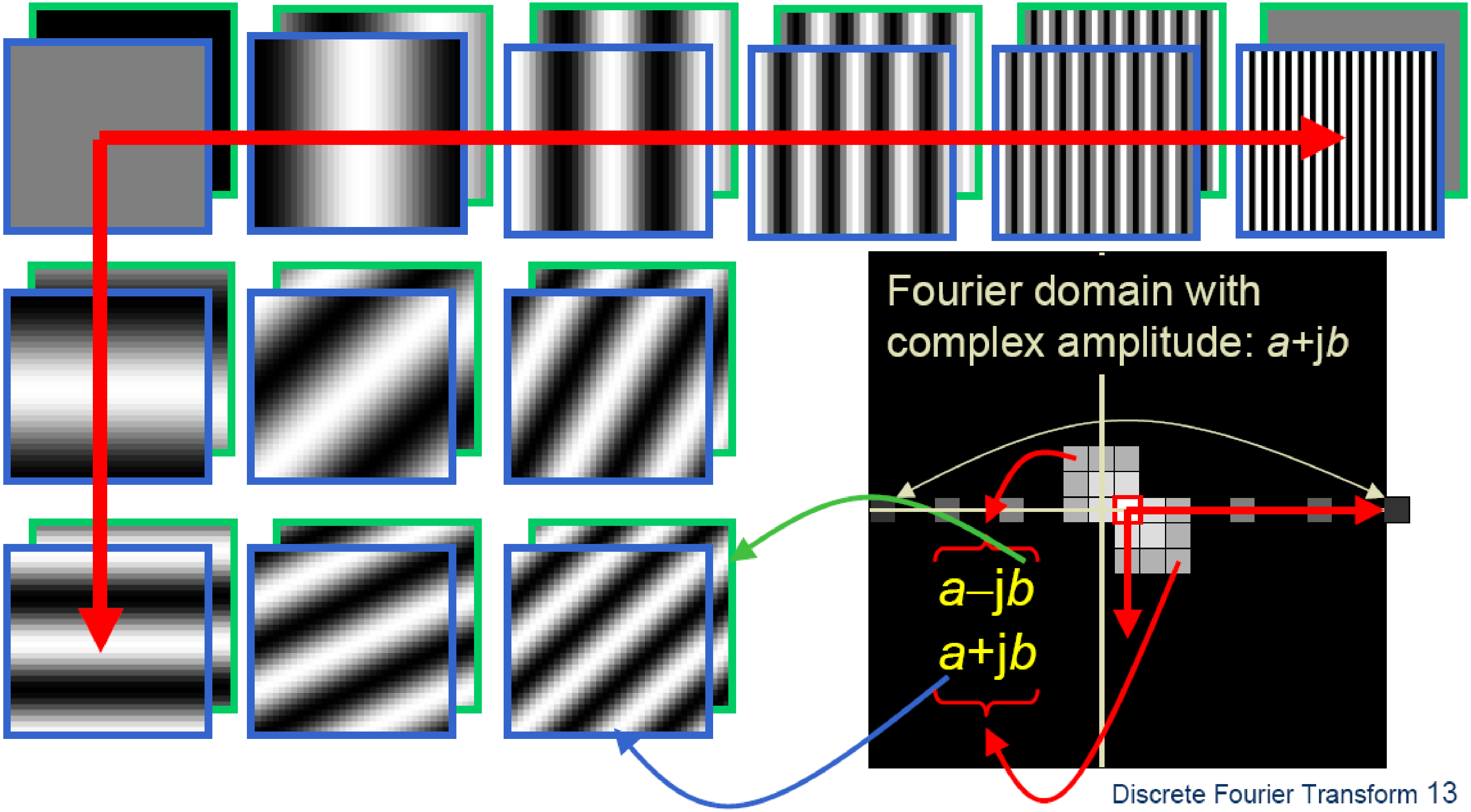
2D Examples – sinusoid magnitudes



Linearity of Sum

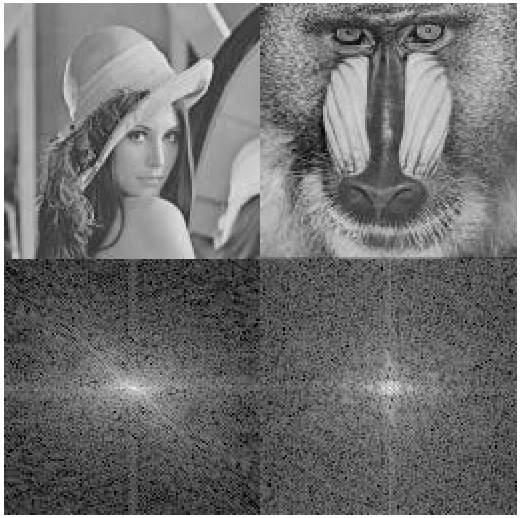


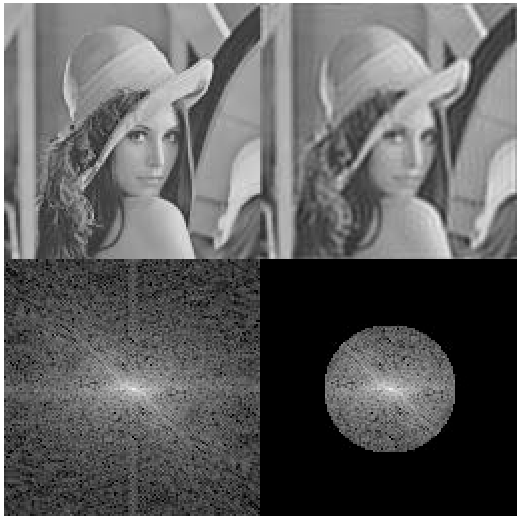
Extension to 2D – Complex plane

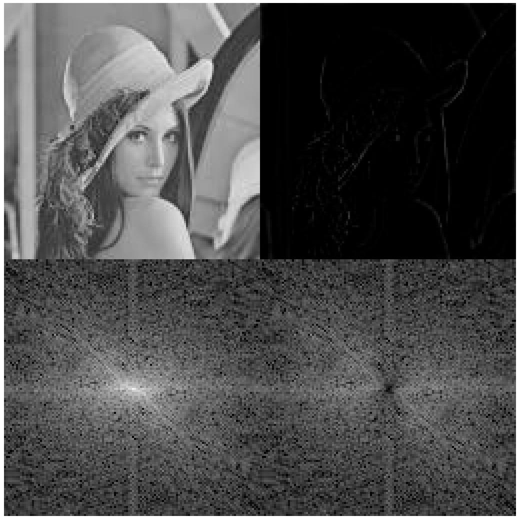


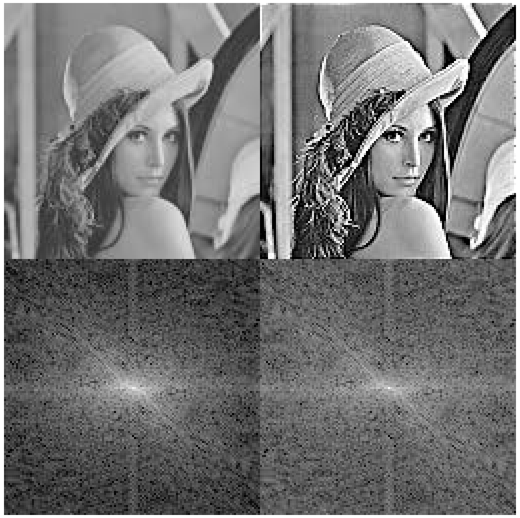
Both a Real and Im version

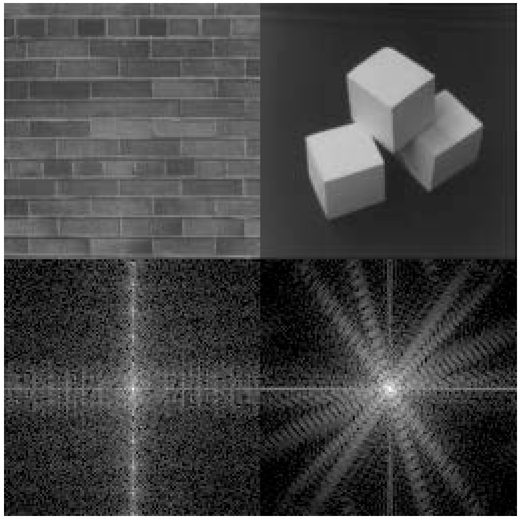
Examples





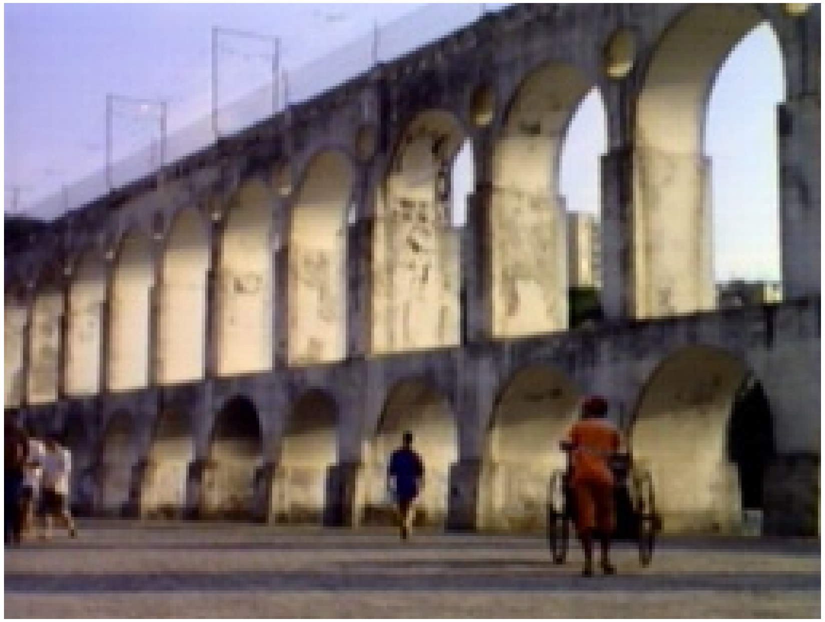


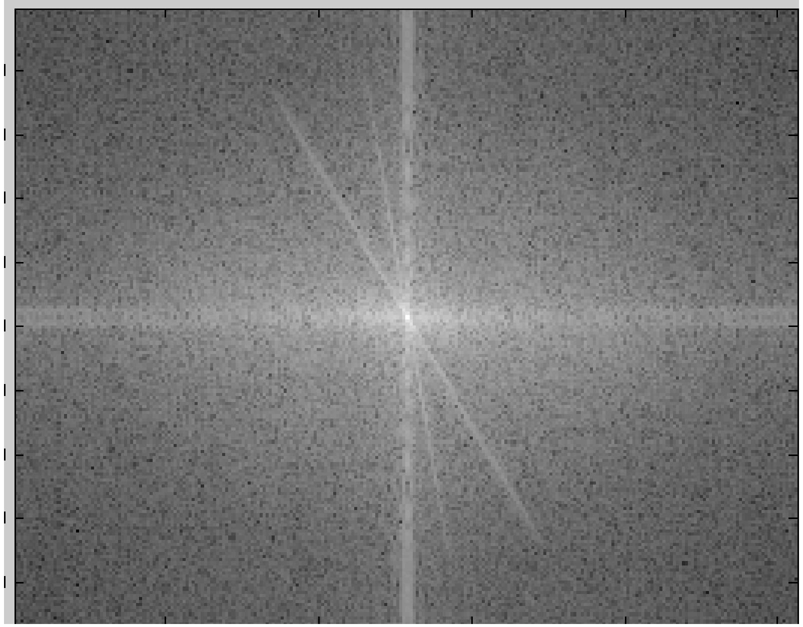




B.K. Gunturk

Man-made Scene





Where is this strong horizontal suggested by vertical center line?

Fourier Transform and Convolution

Let $g = f * h$

Then $G(u) = \int_{-\infty}^{\infty} g(x) e^{-i2\pi ux} dx$

$$= \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} f(\tau) h(x-\tau) e^{-i2\pi ux} d\tau dx$$

$$= \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} \left[f(\tau) e^{-i2\pi u\tau} d\tau \right] \left[h(x-\tau) e^{-i2\pi u(x-\tau)} dx \right]$$

$$= \int_{-\infty}^{\infty} \left[f(\tau) e^{-i2\pi u\tau} d\tau \right] \int_{-\infty}^{\infty} \left[h(x') e^{-i2\pi ux'} dx' \right]$$

$$= F(u) H(u)$$

Convolution in spatial domain

$\Leftrightarrow$ *Multiplication in frequency domain*

Fourier Transform and Convolution

Spatial Domain *(x)* | Frequency Domain *(u)*

$$g = f * h \quad \longleftrightarrow \quad G = FH$$

$$g = fh \quad \longleftrightarrow \quad G = F * H$$

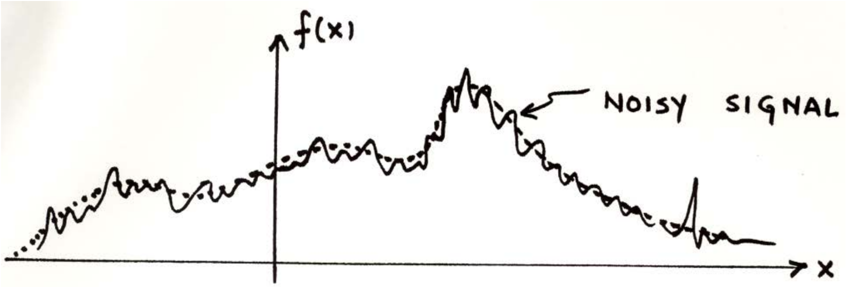
So, we can find *g(x)* by Fourier transform

$$\begin{tikzcd}
g & = & f \arrow[d, "\text{FT}"] & * & h \arrow[d, "\text{FT}"] \\
G \arrow[u, "\text{IFT}"] & = & F & \times & H
\end{tikzcd}$$

Example use: Smoothing/Blurring

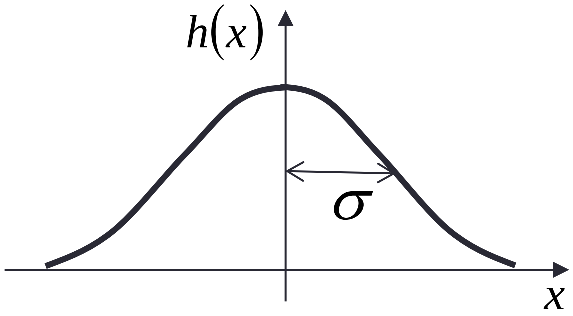
- We want a smoothed function of *f(x)*

$$g(x) = f(x) * h(x)$$



- Let us use a Gaussian kernel

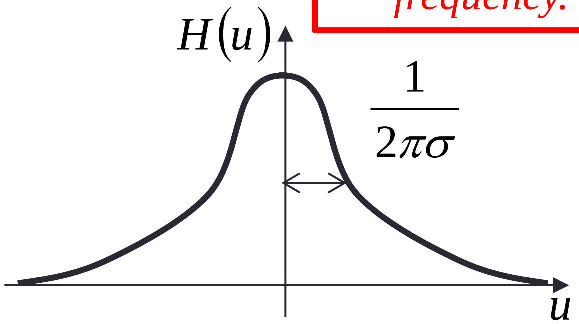
$$h(x) = \frac{1}{\sqrt{2\pi}\sigma} \exp\left[-\frac{1}{2}\frac{x^2}{\sigma^2}\right]$$



Fat Gaussian in space is skinny Gaussian in frequency. Why?

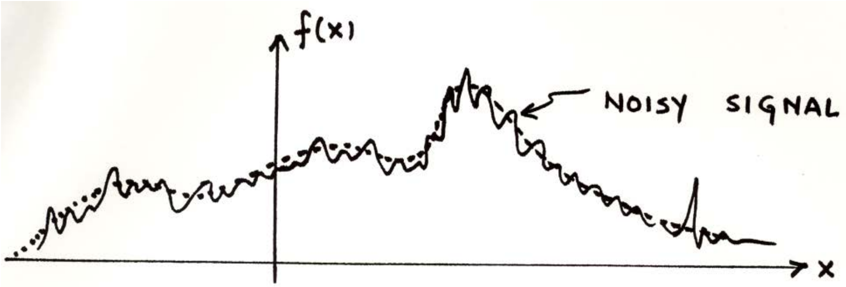
- The Fourier transform of a Gaussian is a Gaussian

$$H(u) = \exp\left[-\frac{1}{2}(2\pi u)^2 \sigma^2\right]$$



Example use: Smoothing/Blurring

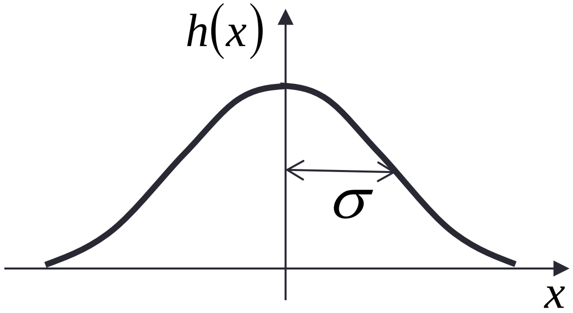
- We want a smoothed function of *f(x)*



$$g(x) = f(x) * h(x)$$

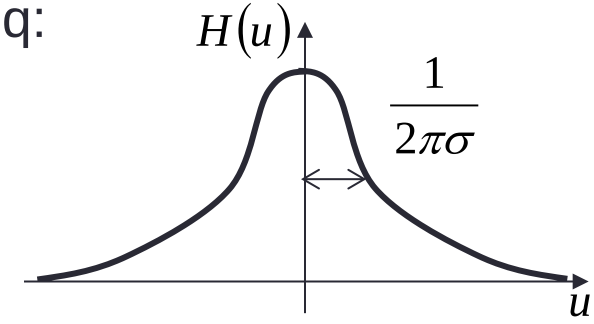
- Let us use a Gaussian kernel

$$h(x) = \frac{1}{\sqrt{2\pi}\sigma} \exp\left[-\frac{1}{2}\frac{x^2}{\sigma^2}\right]$$



- Convolution in space is multiplication in freq:

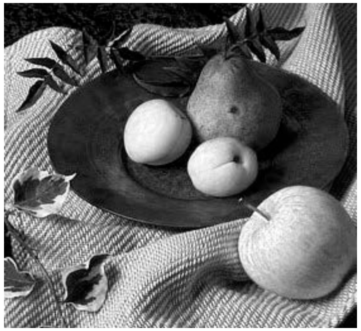
$$G(u) = F(u)H(u)$$

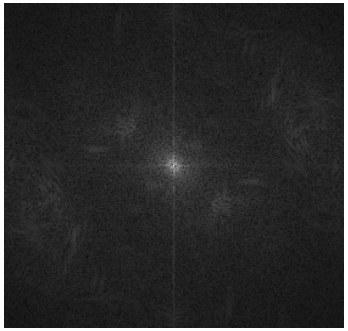


H(u) ***attenuates*** high frequencies in *F(u)* (Low-pass Filter)!

2D convolution theorem example

$f(x,y)$



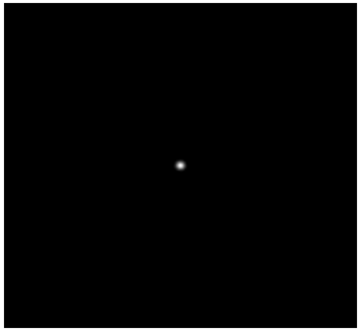


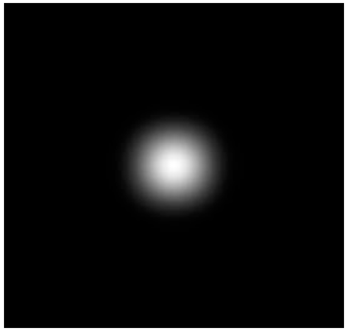
$|F(s_x,s_y)|$

(or $|F(u,v)|$)

$*$ $\times$

$h(x,y)$

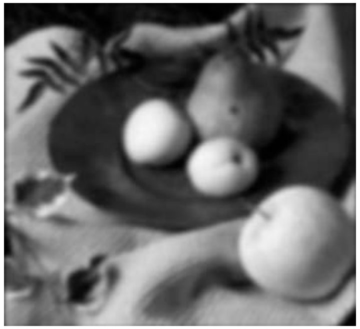


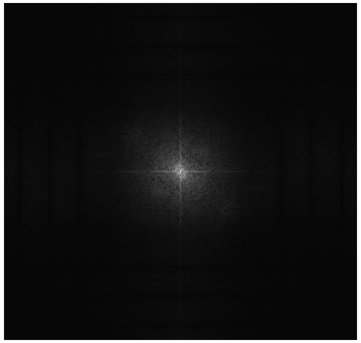


$|H(s_x,s_y)|$

$\Downarrow$ $\Downarrow$

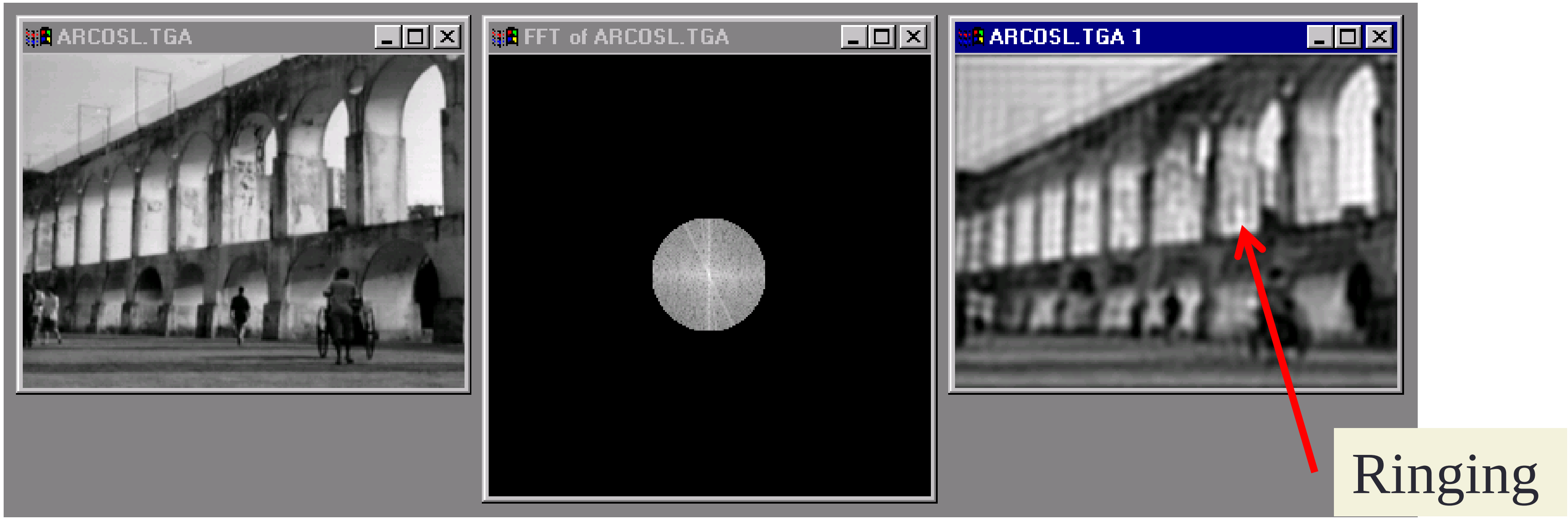
$g(x,y)$



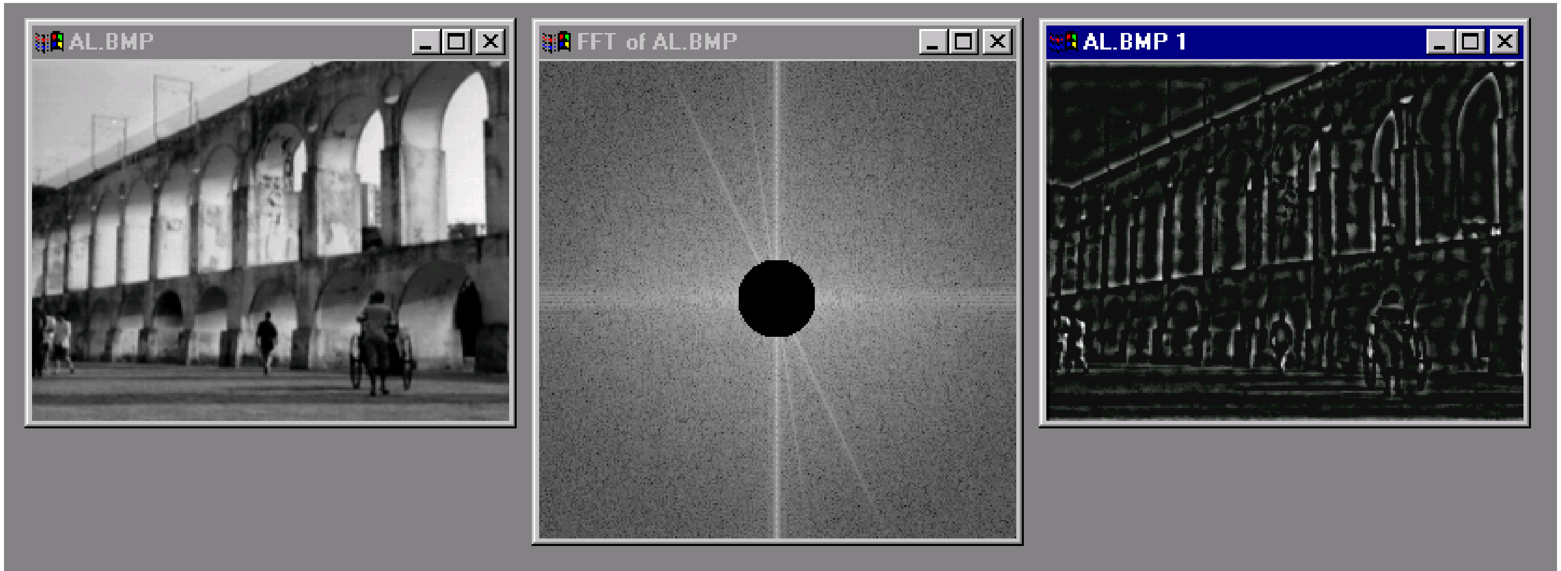


$|G(s_x,s_y)|$

Low and High Pass filtering



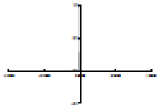
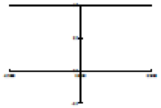
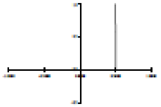
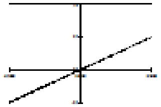
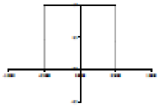
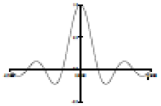
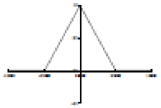
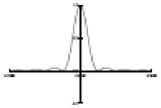
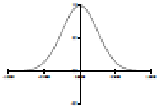
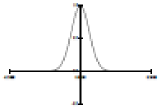
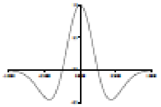
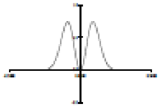
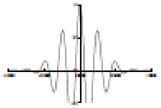
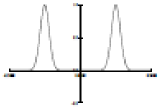
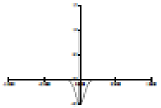
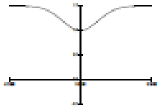
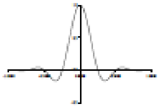
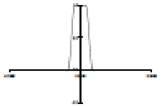
Ringing



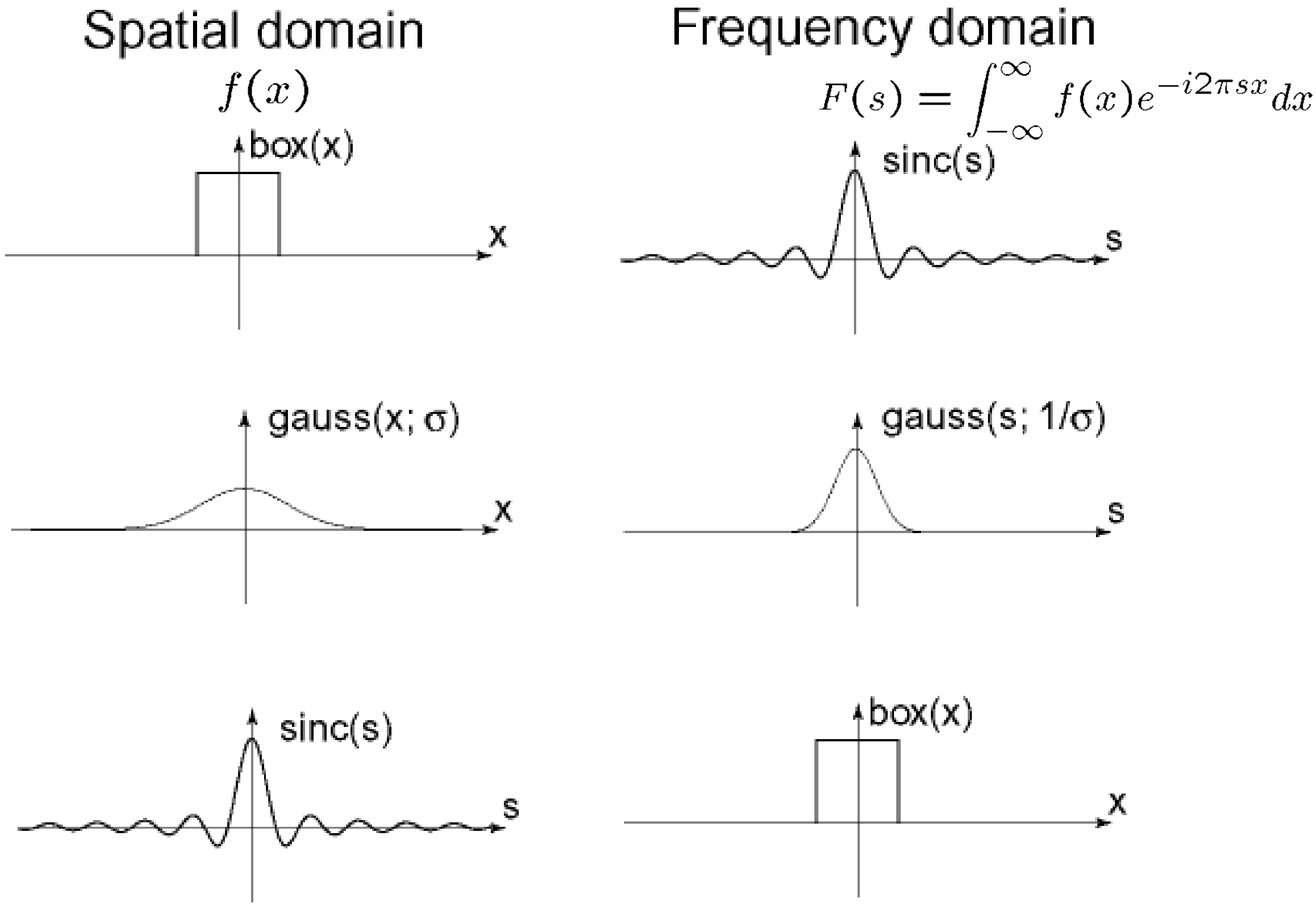
Properties of Fourier Transform

	Spatial Domain *(x)*	Frequency Domain *(u)*
Linearity	$c_1 f(x) + c_2 g(x)$	$c_1 F(u) + c_2 G(u)$
Scaling	$f(ax)$	$\frac{1}{\lvert a \rvert} F\left(\frac{u}{a}\right)$
Shifting	$f(x - x_0)$	$e^{-i2\pi u x_0} F(u)$
Symmetry	$F(x)$	$f(-u)$
Conjugation	$f^*(x)$	$F^*(-u)$
Convolution	$f(x) * g(x)$	$F(u)G(u)$
Differentiation	$\frac{d^n f(x)}{dx^n}$	$(i2\pi u)^n F(u)$

Fourier Pairs (from Szeliski)

Name	Signal			Transform	
impulse		$\delta(x)$	⇔	1	
shifted impulse		$\delta(x-u)$	⇔	$e^{-j\omega u}$	
box filter		$\text{box}(x/a)$	⇔	$a\text{sinc}(a\omega)$	
tent		$\text{tent}(x/a)$	⇔	$a\text{sinc}^2(a\omega)$	
Gaussian		$G(x;\sigma)$	⇔	$\frac{\sqrt{2\pi}}{\sigma}G(\omega;\sigma^{-1})$	
Laplacian of Gaussian		$(\frac{x^2}{\sigma^4}-\frac{1}{\sigma^2})G(x;\sigma)$	⇔	$-\frac{\sqrt{2\pi}}{\sigma}\omega^2G(\omega;\sigma^{-1})$	
Gabor		$\cos(\omega_0x)G(x;\sigma)$	⇔	$\frac{\sqrt{2\pi}}{\sigma}G(\omega\pm\omega_0;\sigma^{-1})$	
unsharp mask		$(1+\gamma)\delta(x)$ $-\gamma G(x;\sigma)$	⇔	$(1+\gamma)-$ $\frac{\sqrt{2\pi}\gamma}{\sigma}G(\omega;\sigma^{-1})$	
windowed sinc		$\text{rcos}(x/(aW))$ $\text{sinc}(x/a)$	⇔	(see Figure 3.29)	

Fourier Transform smoothing pairs



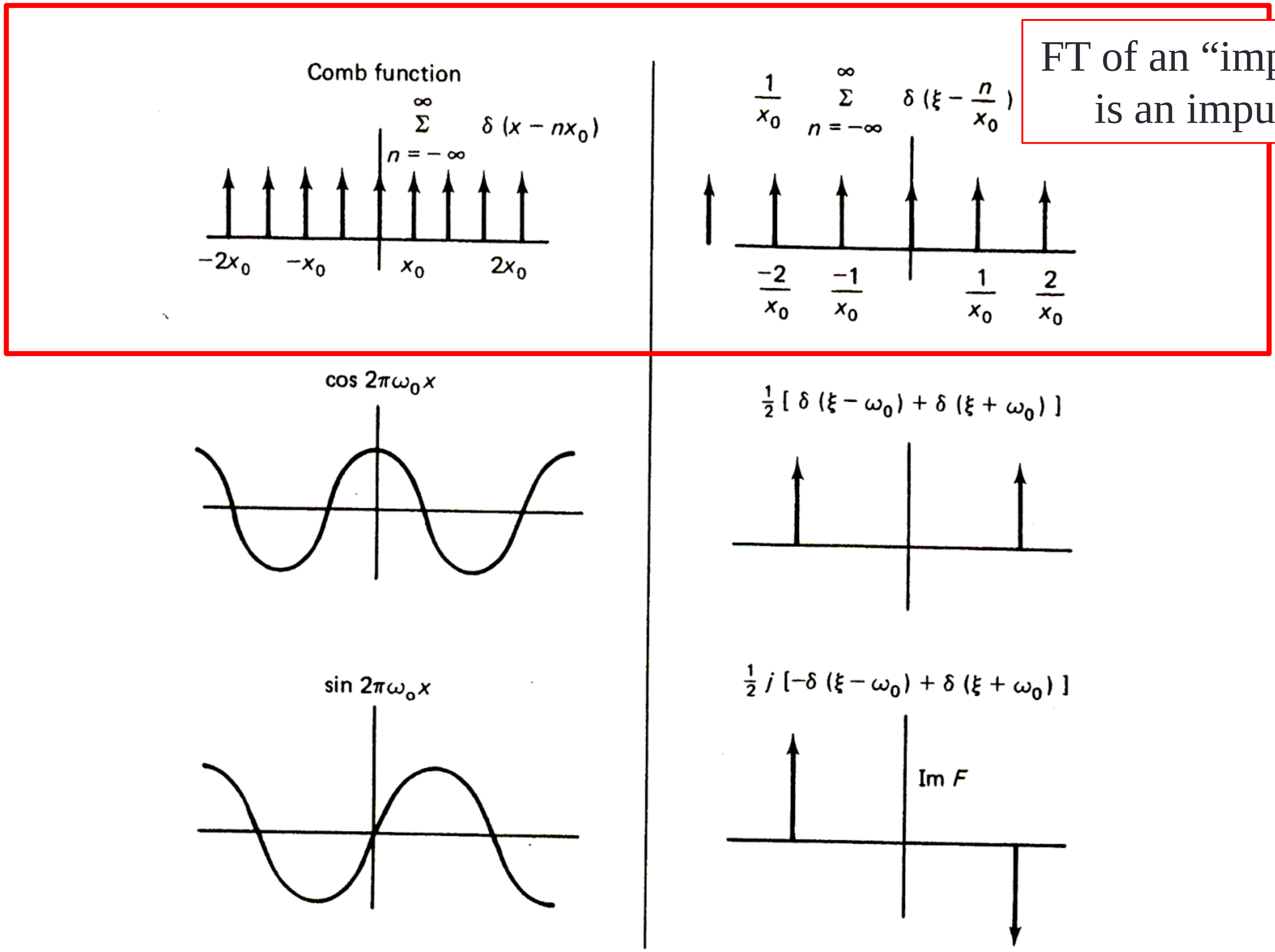
Spatial domain

$f(x)$

Frequency domain

$$F(s) = \int_{-\infty}^{\infty} f(x) e^{-i2\pi s x} dx$$

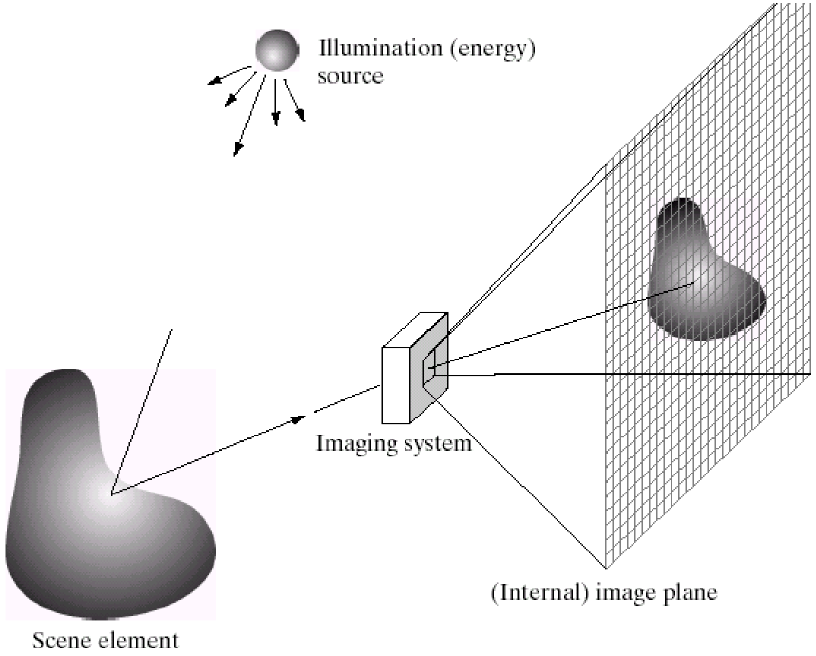
Fourier Transform Sampling Pairs

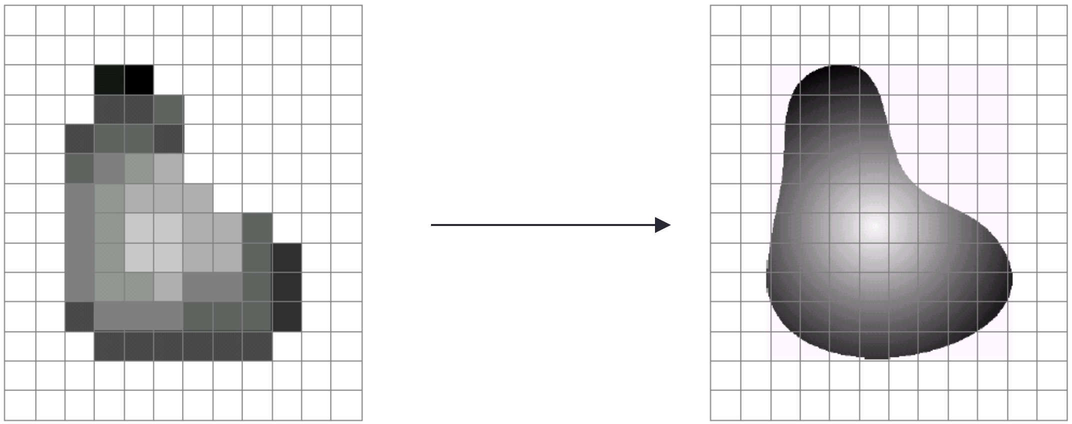

Comb function
$\sum_{n=-\infty}^{\infty} \delta(x - nx_0)$
$-2x_0$
$-x_0$
x_0
$2x_0$
$\frac{1}{x_0} \sum_{n=-\infty}^{\infty} \delta(\xi - \frac{n}{x_0})$
$\frac{-2}{x_0}$
$\frac{-1}{x_0}$
$\frac{1}{x_0}$
$\frac{2}{x_0}$
$\cos 2\pi\omega_0 x$
$\frac{1}{2}[\delta(\xi - \omega_0) + \delta(\xi + \omega_0)]$
$\sin 2\pi\omega_o x$
$\frac{1}{2} i [-\delta(\xi - \omega_0) + \delta(\xi + \omega_0)]$
Im F

FT of an “impulse train” is an impulse train

Sampling and Aliasing

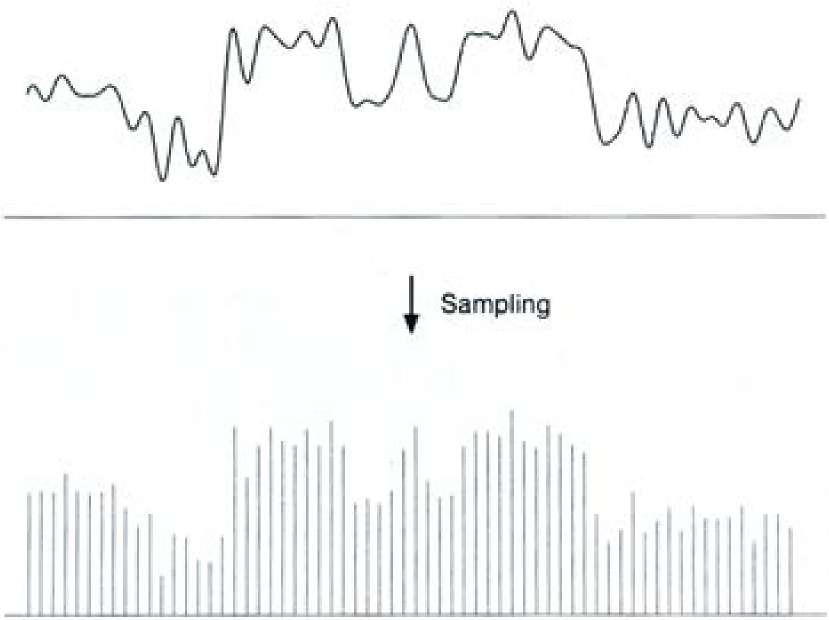
Sampling and Reconstruction





Sampled representations

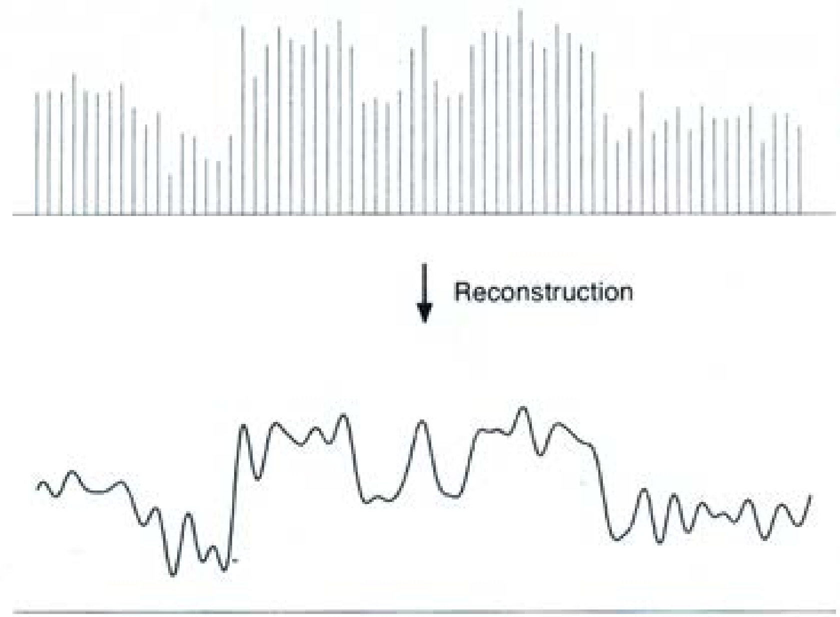
- How to store and compute with continuous functions?
- Common scheme for representation: samples
  - write down the function’s values at many points



S. Marschner

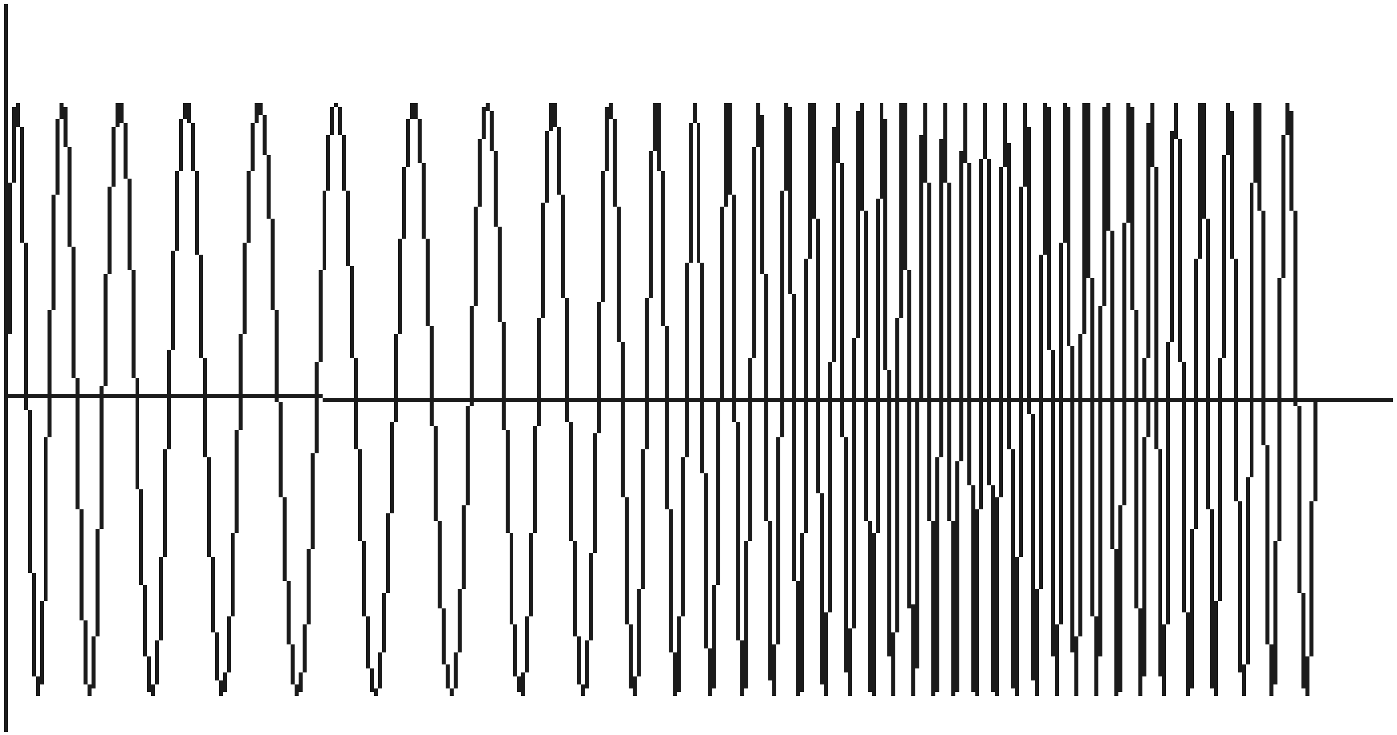
Reconstruction

- Making samples back into a continuous function
  - for output (need realizable method)
  - for analysis or processing (need mathematical method)
  - amounts to “guessing” what the function did in between



S. Marschner

1D Example: Audio

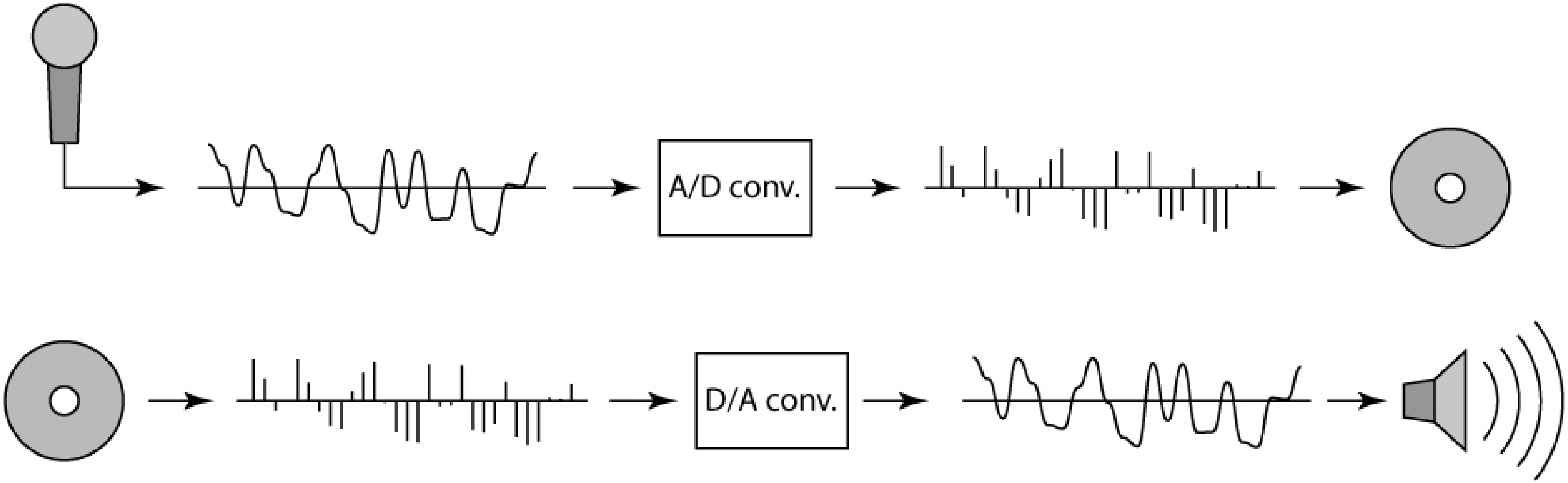


low high

frequencies

Sampling in digital audio

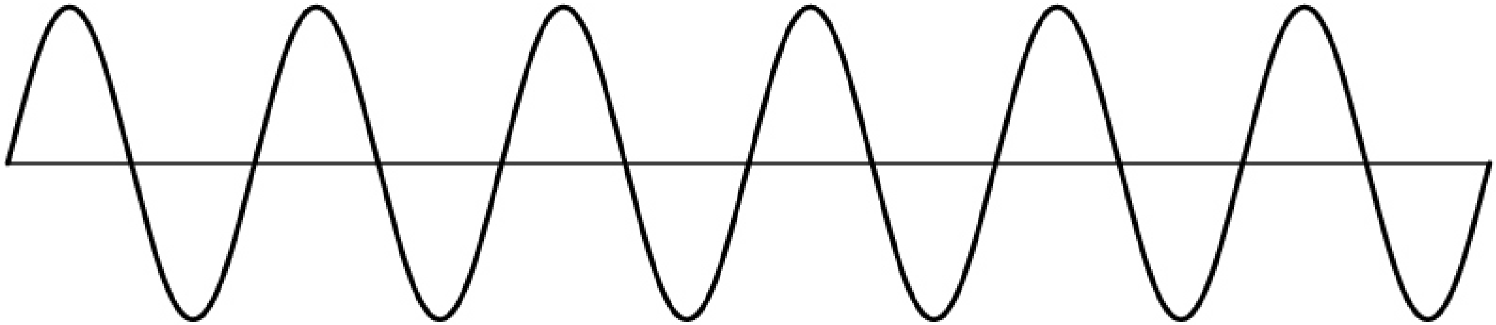
- Recording: sound to analog to samples to disc
- Playback: disc to samples to analog to sound again
  - how can we be sure we are filling in the gaps correctly?



S. Marschner

Sampling and Reconstruction

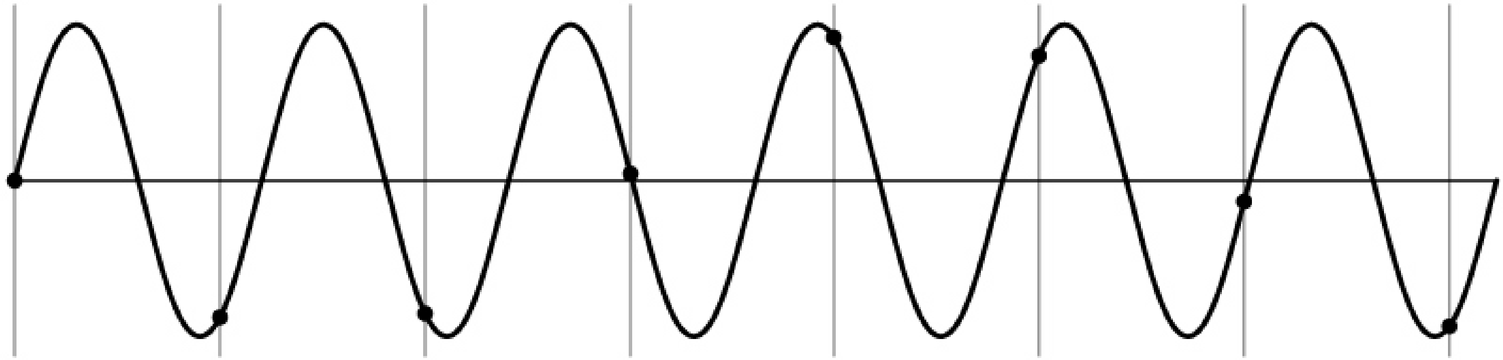
- Simple example: a sign wave



S. Marschner

Undersampling

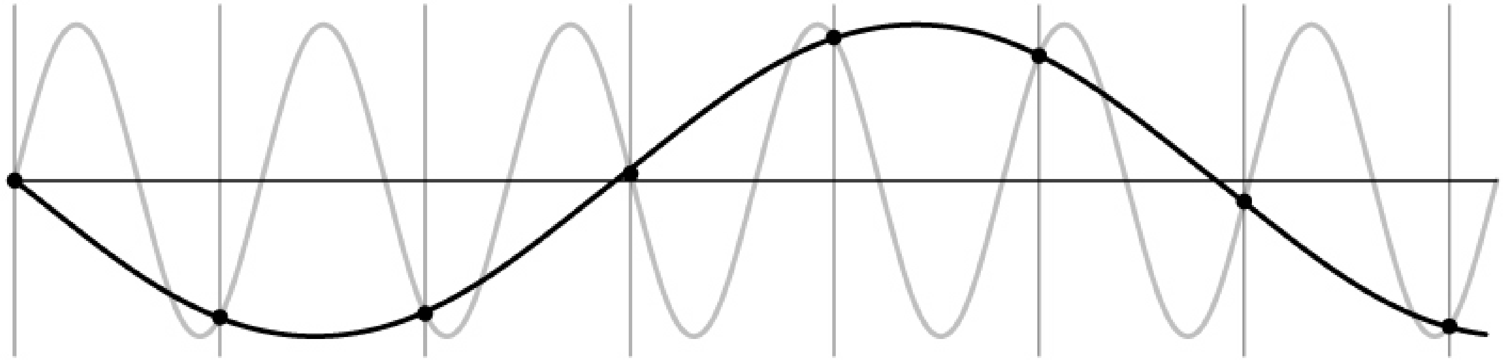
- What if we “missed” things between the samples?
- Simple example: undersampling a sine wave
  - unsurprising result: information is lost



S. Marschner

Undersampling

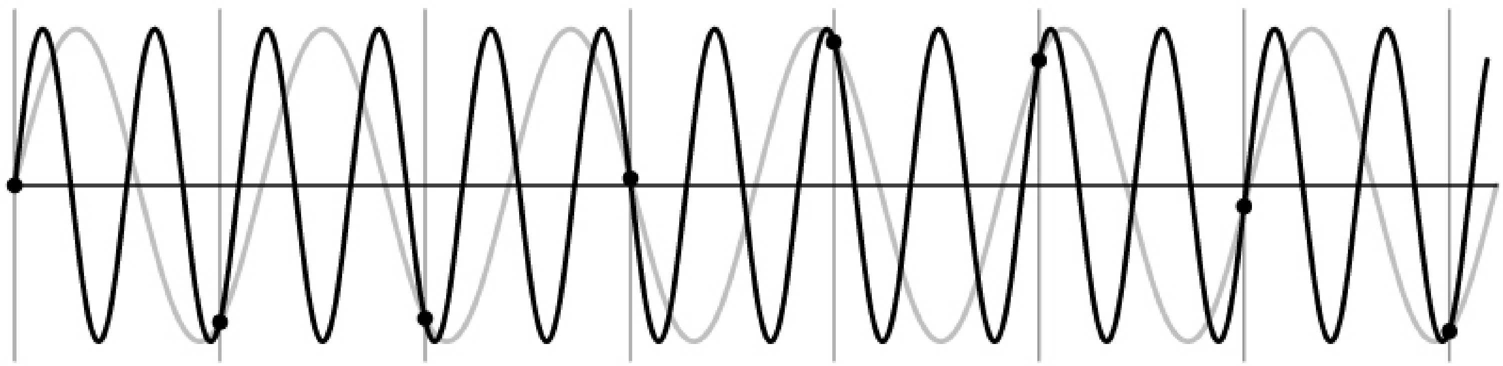
- What if we “missed” things between the samples?
- Simple example: undersampling a sine wave
  - unsurprising result: information is lost
  - surprising result: indistinguishable from lower frequency



S. Marschner

Undersampling

- What if we “missed” things between the samples?
- Simple example: undersampling a sine wave
  - unsurprising result: information is lost
  - surprising result: indistinguishable from lower frequency
  - also was always indistinguishable from higher frequencies
  - *aliasing*: signals “traveling in disguise” as other frequencies

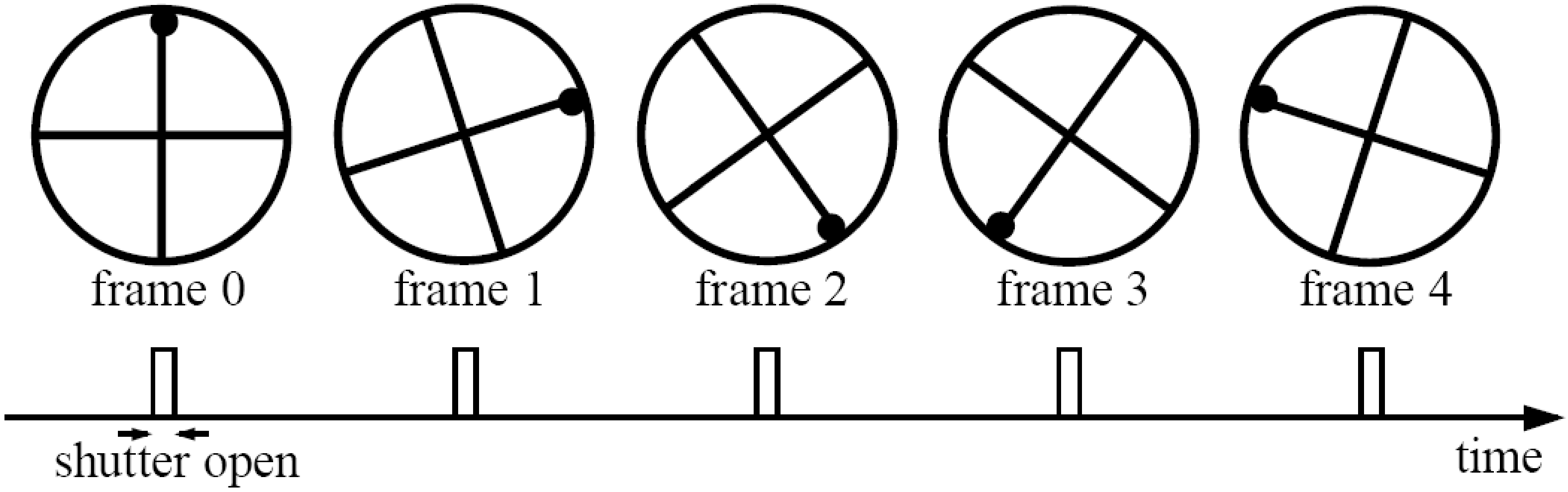


S. Marschner

Aliasing in video

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

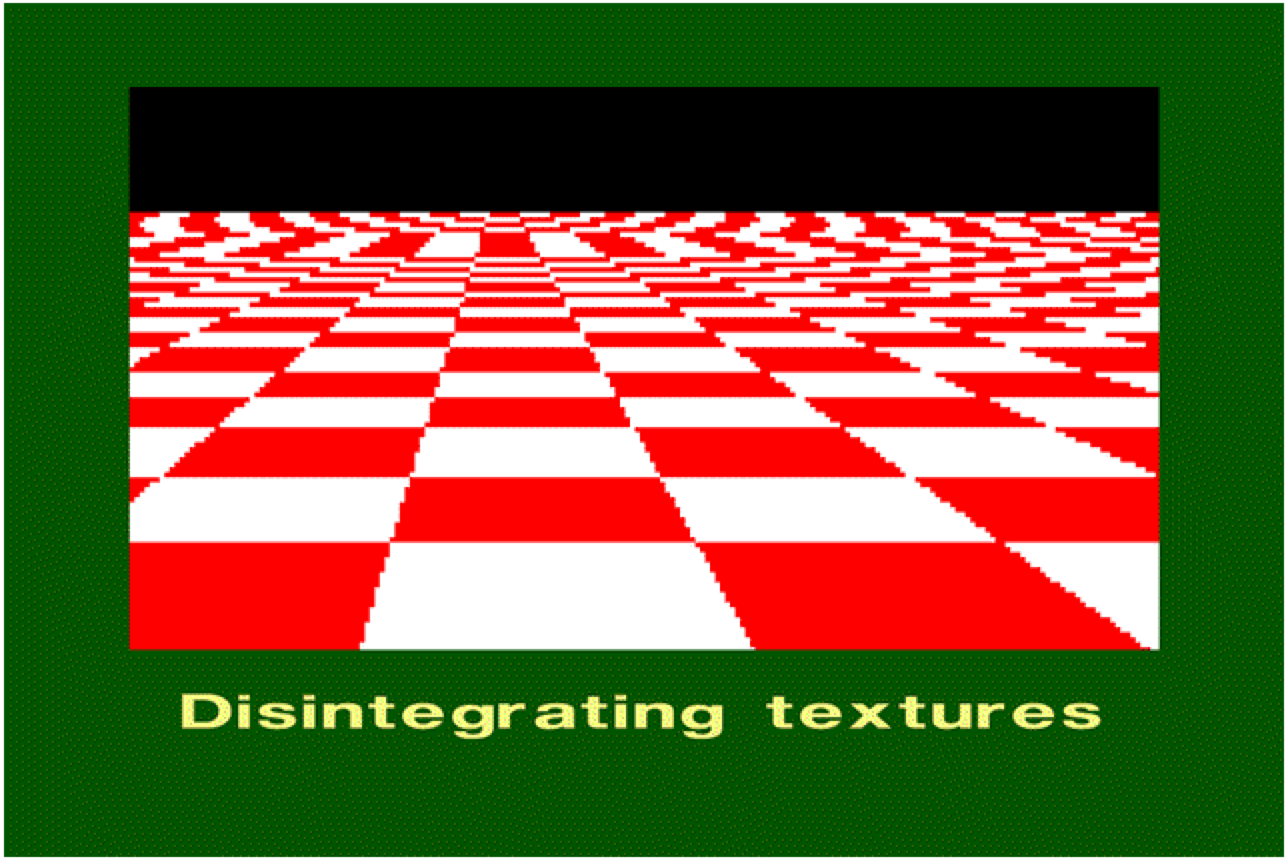
If camera shutter is only open for a fraction of a frame time (frame time = 1/30 sec. for video, 1/24 sec. for film):



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

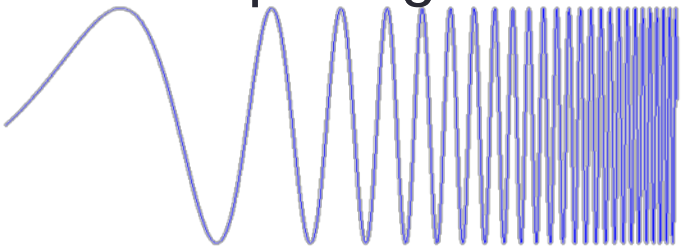
S. Seitz

Aliasing in images

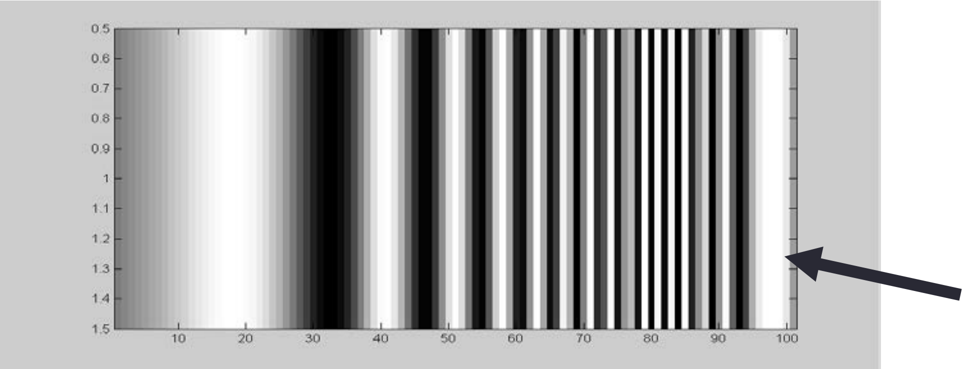


What’s happening?

Input signal:

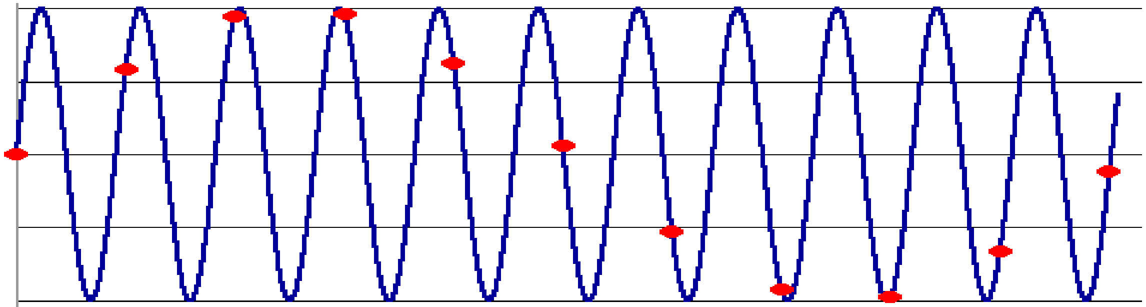


Plot as image:



Alias!
Not enough samples

x = 0:.05:5; imagesc(sin((2.^x).*x))



Antialiasing

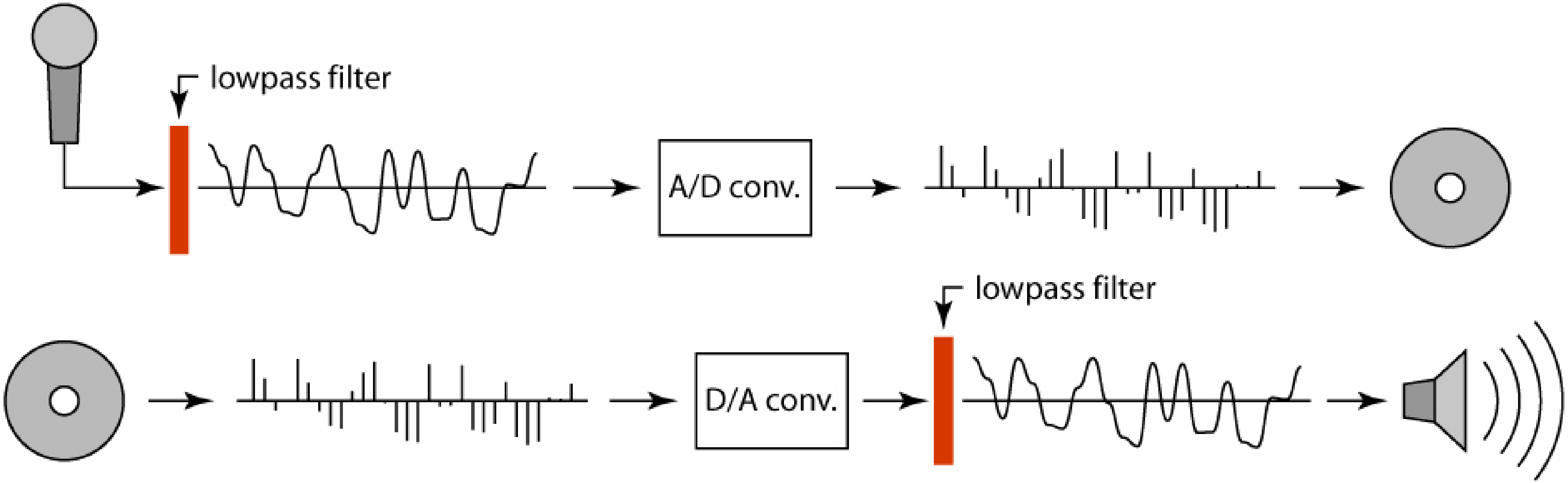
- What can we do about aliasing?

- Sample more often
  - Join the Mega-Pixel craze of the photo industry
  - But this can’t go on forever

- Make the signal less “wiggly”
  - Get rid of some high frequencies
  - Will loose information
  - But it’s better than aliasing

Preventing aliasing

- Introduce lowpass filters:
  - remove high frequencies leaving only safe, low frequencies
  - choose lowest frequency in reconstruction (disambiguate)



S. Marschner

(Anti)Aliasing in the Frequency Domain

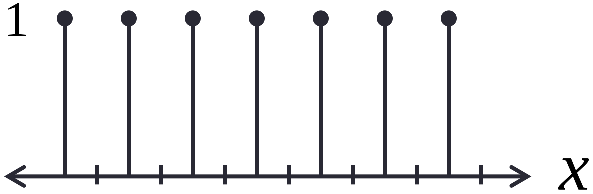
Impulse Train

- Define a *comb* function (impulse train) in 1D as follows

$$comb_M[x] = \sum_{k=-\infty}^{\infty} \delta[x - kM]$$

where *M* is an integer

$comb_2[x]$



B.K. Gunturk

Impulse Train in 2D *(bed of nails)*

$$comb_{M,N}(x,y) \triangleq \sum_{k=-\infty}^{\infty}\sum_{l=-\infty}^{\infty}\delta\left(x-kM, y-lN\right)$$

- Fourier Transform of an impulse train is also an impulse train:

$$\underbrace{\sum_{k=-\infty}^{\infty}\sum_{l=-\infty}^{\infty}\delta\left(x-kM, y-lN\right)}_{comb_{M,N}(x,y)} \Leftrightarrow \underbrace{\frac{1}{MN}\sum_{k=-\infty}^{\infty}\sum_{l=-\infty}^{\infty}\delta\left(u-\frac{k}{M}, v-\frac{l}{N}\right)}_{comb_{\frac{1}{M},\frac{1}{N}}(u,v)}$$

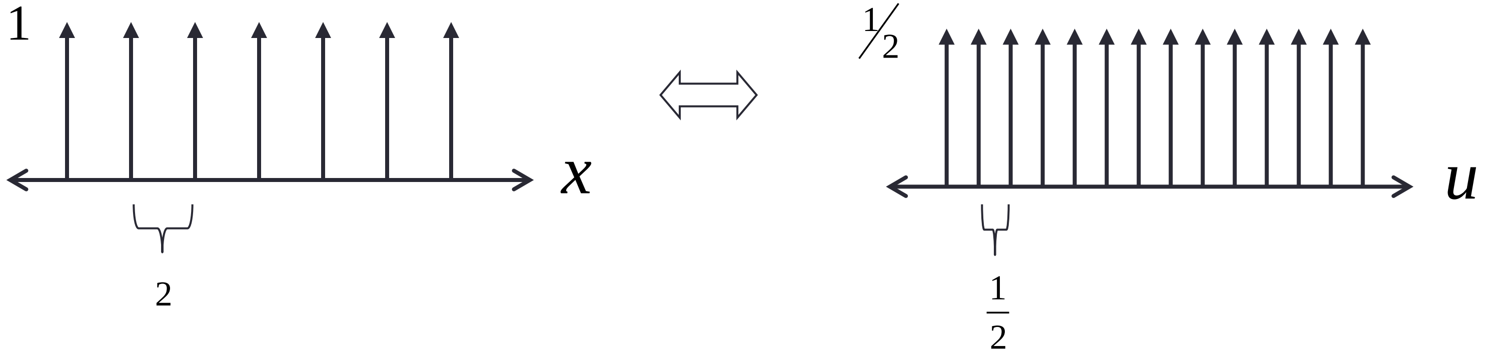
As the comb samples get further apart, the spectrum samples get closer together!

B.K. Gunturk

Impulse Train in 1D

$comb_2(x)$

$\frac{1}{2} comb_{\frac{1}{2}}(u)$

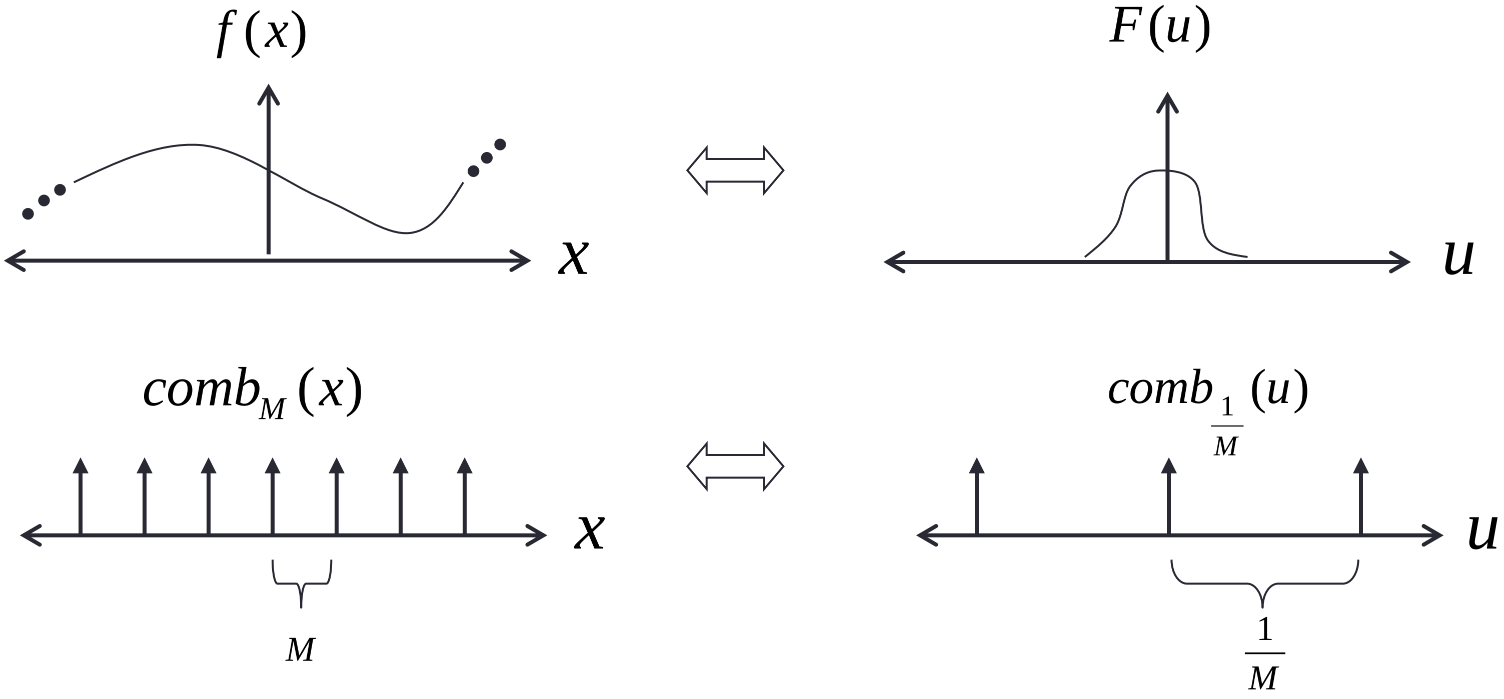


Remember:

Scaling $f(ax)$ $\frac{1}{|a|} F\left(\frac{u}{a}\right)$

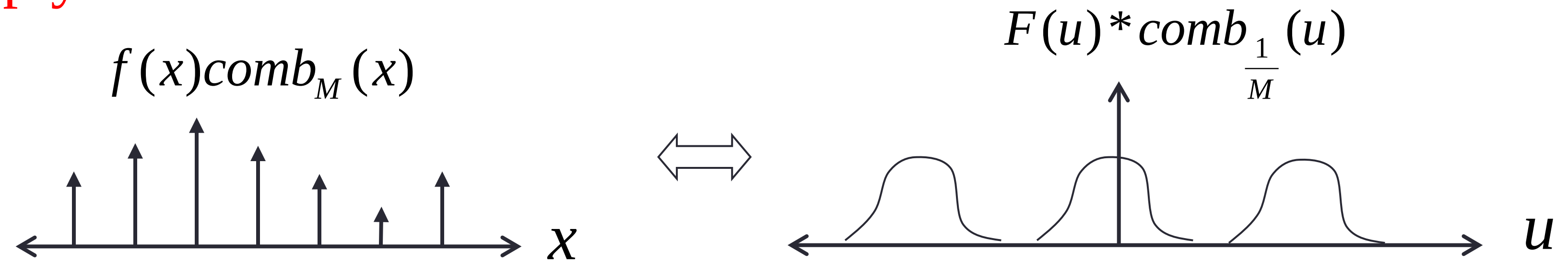
B.K. Gunturk

Sampling low frequency signal



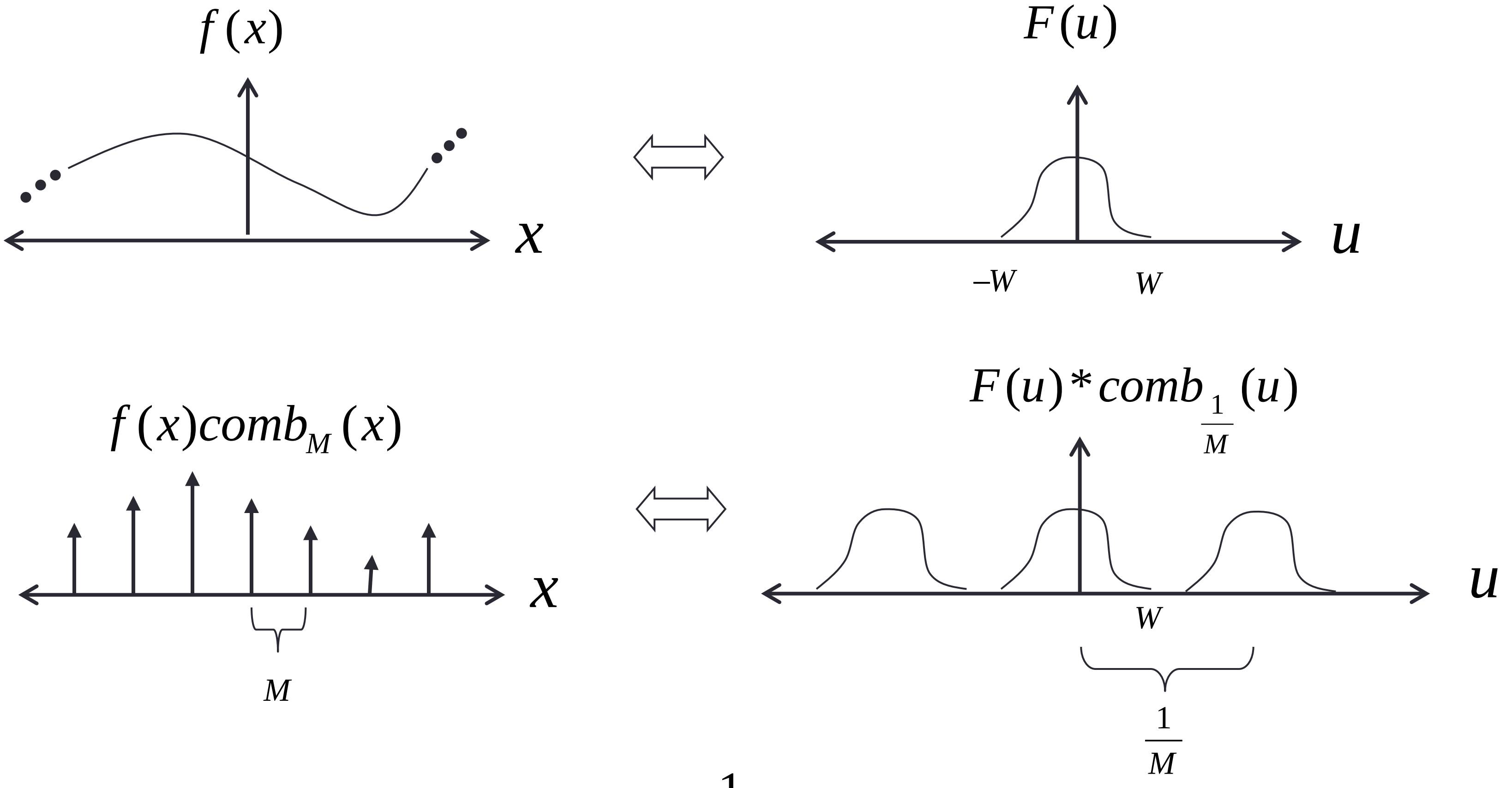
Multiply:

Convolve:



B.K. Gunturk

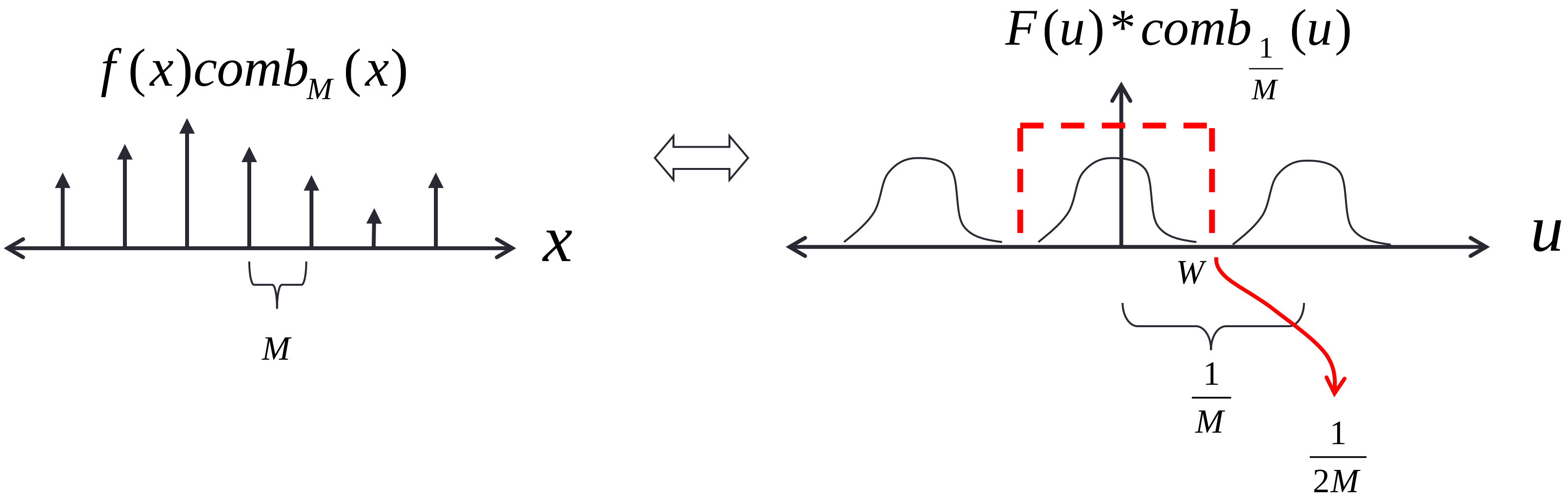
Sampling low frequency signal



No “problem” if $\frac{1}{M} > 2W$

B.K. Gunturk

Sampling low frequency signal

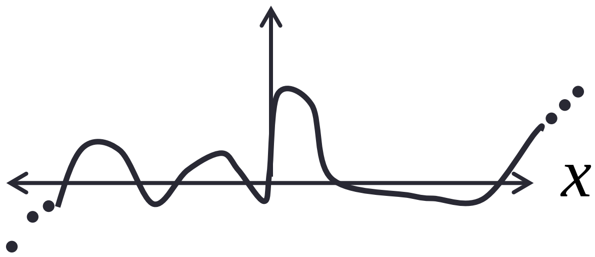


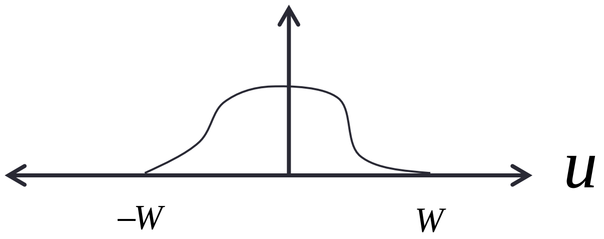
If there is no overlap, the original signal can be recovered from its samples by low-pass filtering.

B.K. Gunturk

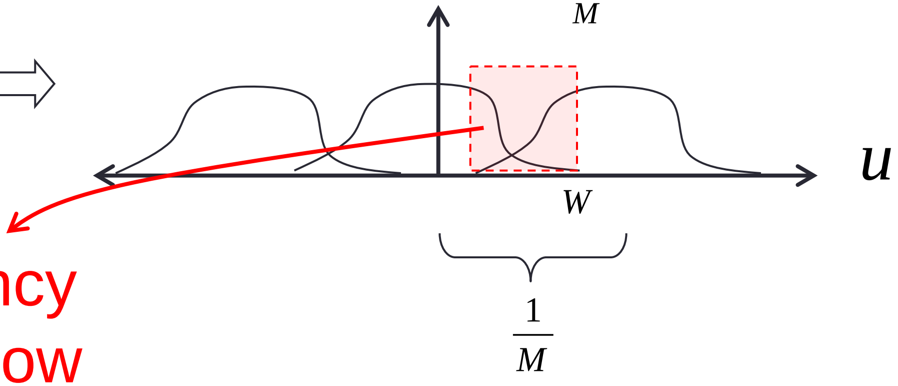
Sampling high frequency signal

$f(x)$ $\Leftrightarrow$ $F(u)$





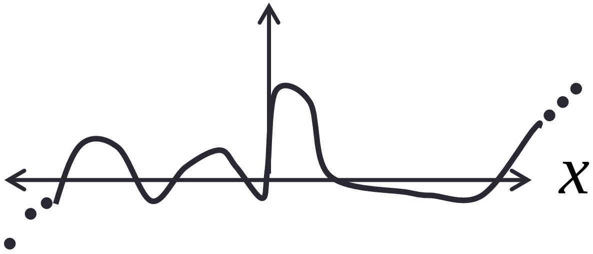
$f(x)comb_M(x)$ $\Leftrightarrow$ $F(u)*comb_{\frac{1}{M}}(u)$

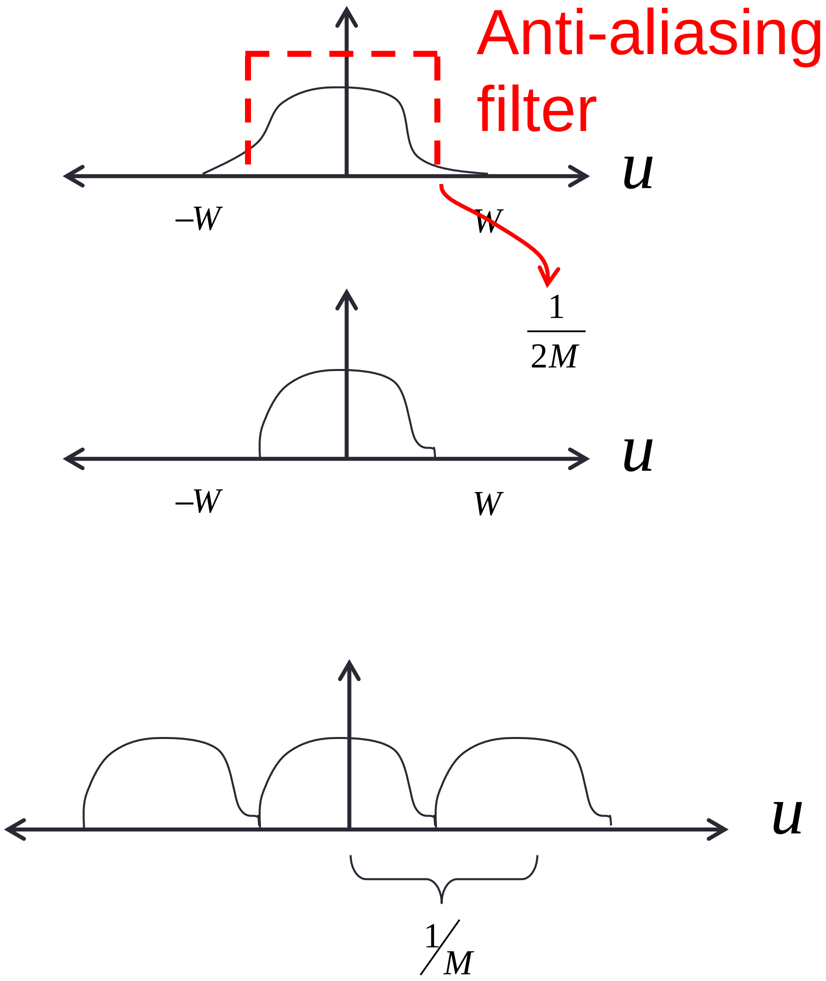


Overlap: The high frequency energy is folded over into low frequency. It is "aliasing" as lower frequency energy. And you cannot fix it once it has happened.

Sampling high frequency signal

$$f(x) \Longleftrightarrow F(u)$$




Anti-aliasing filter

$$f(x) * h(x) \Longleftrightarrow$$

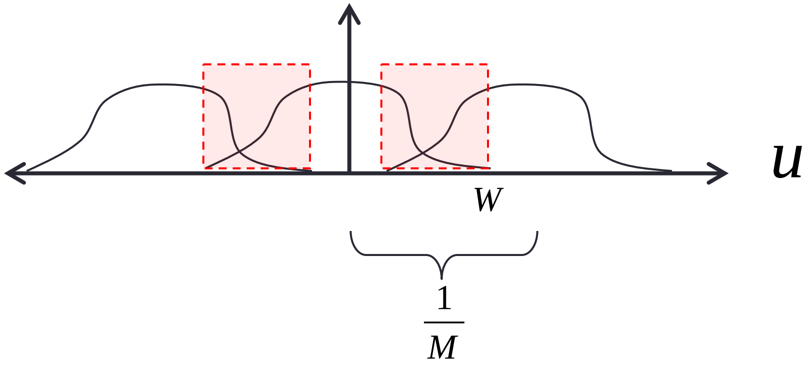
$$\left[f(x) * h(x) \right] comb_M(x) \Longleftrightarrow$$

B.K. Gunturk

Sampling high frequency signal

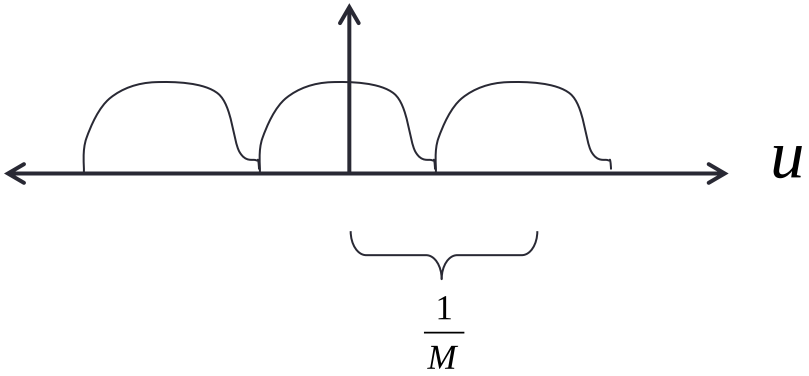
- Without anti-aliasing filter:

$$f(x)comb_M(x) \Longleftrightarrow$$



- With anti-aliasing filter:

$$[f(x)*h(x)]comb_M(x) \Longleftrightarrow$$



B.K. Gunturk

Aliasing in Images

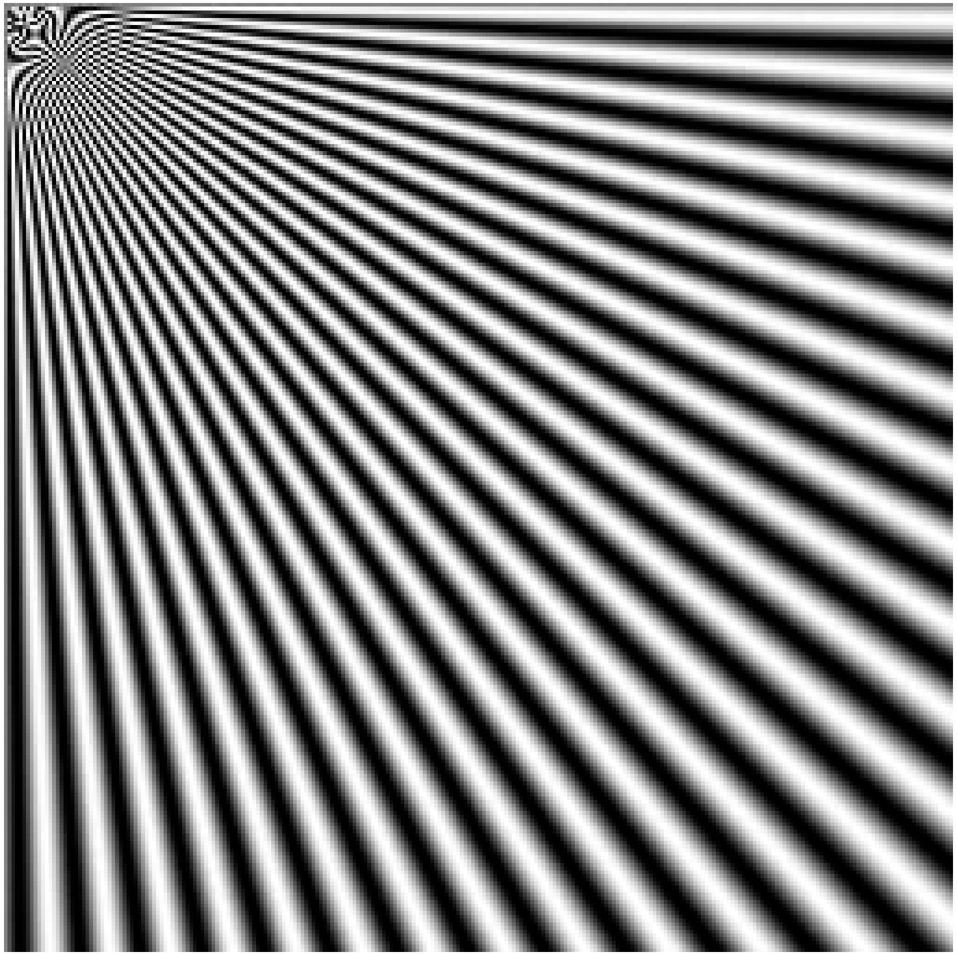
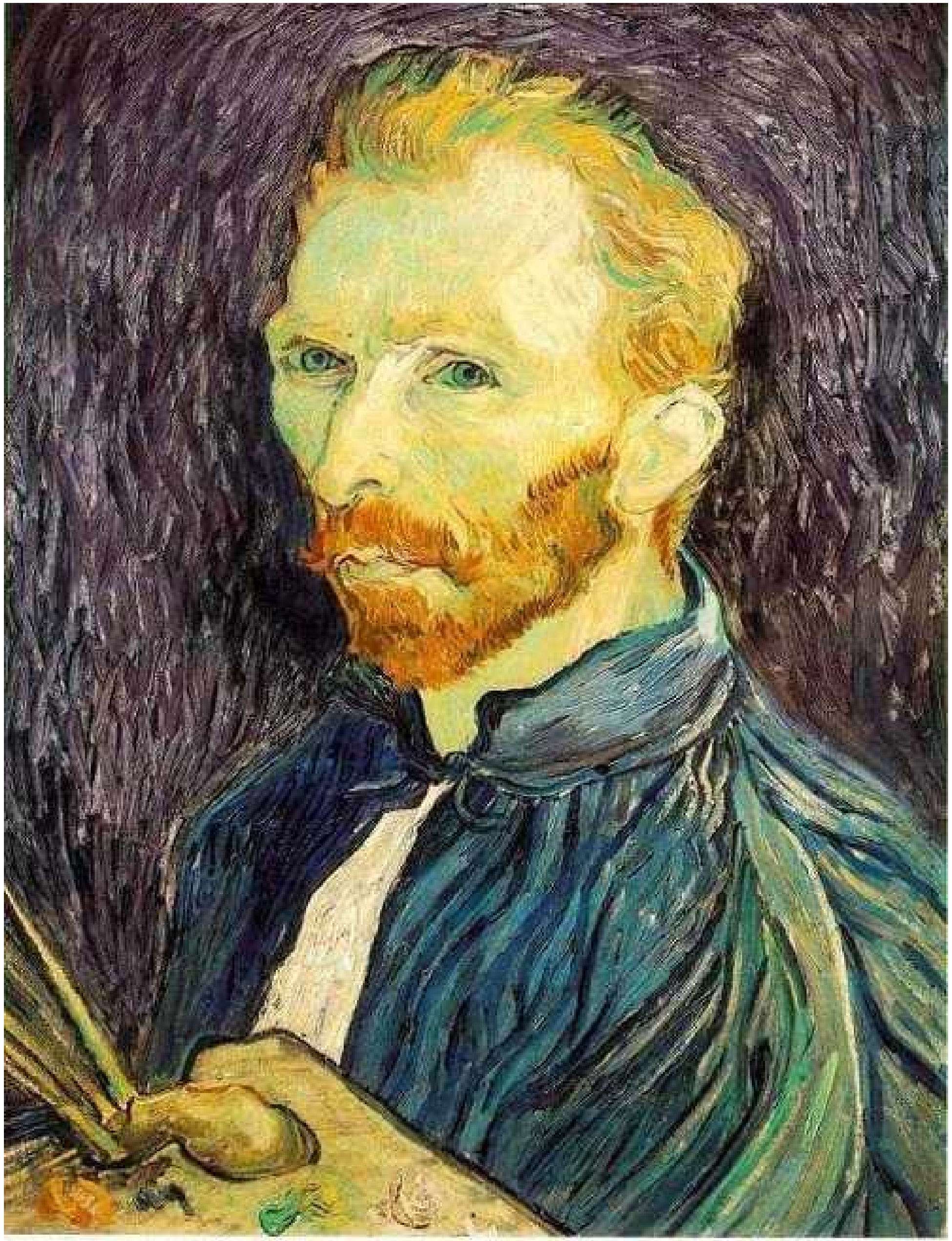


Image half-sizing

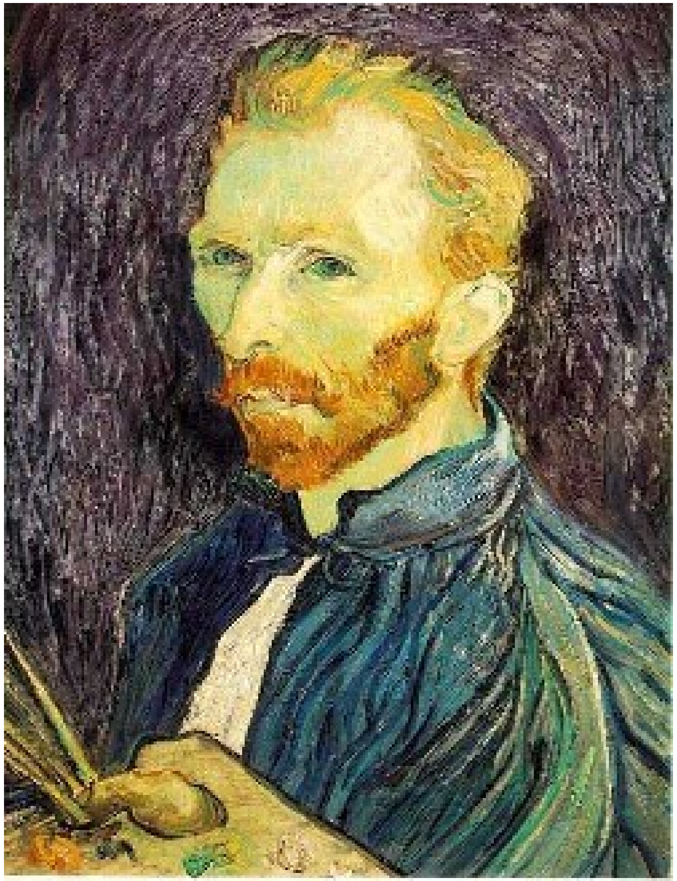
This image is too big to fit on the screen. How can we reduce it?

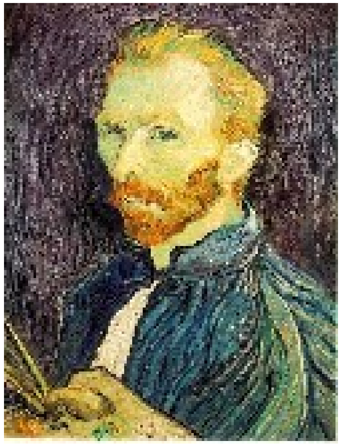
How to generate a half-sized version?



S. Seitz

Image sub-sampling





1/4

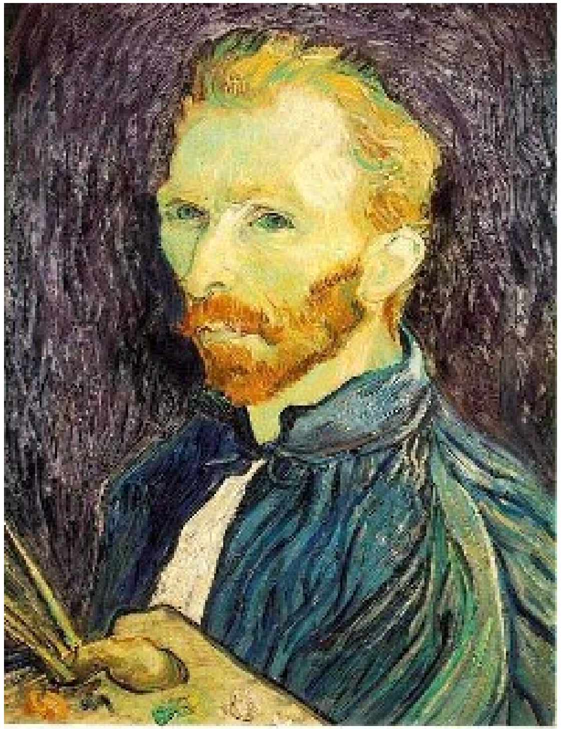
1/8

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

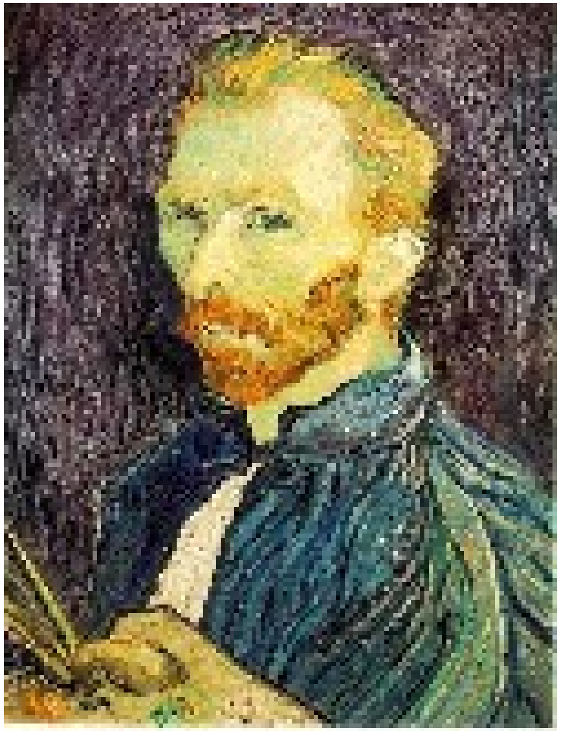
- called *image sub-sampling*

S. Seitz

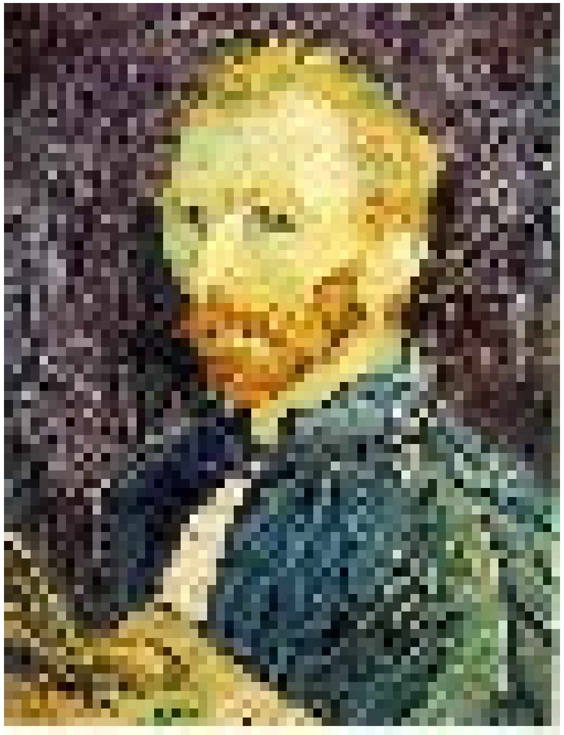
Image sub-sampling



1/2



1/4 (2x zoom)

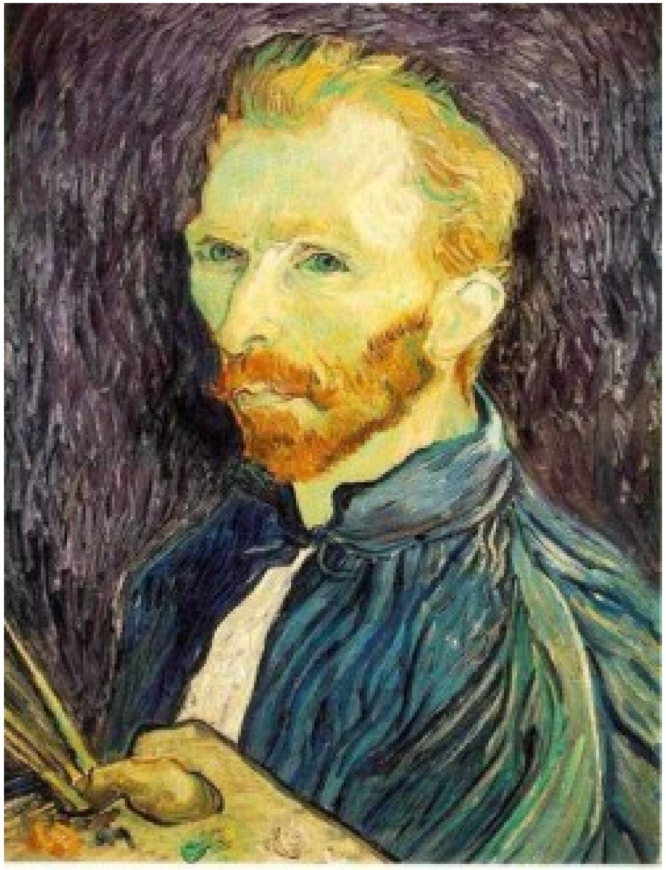


1/8 (4x zoom)

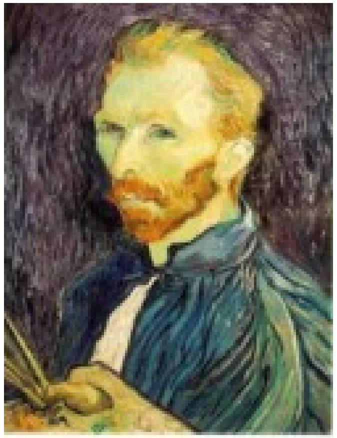
Aliasing! What do we do?

S. Seitz

Gaussian (lowpass) pre-filtering



Gaussian 1/2



G 1/4

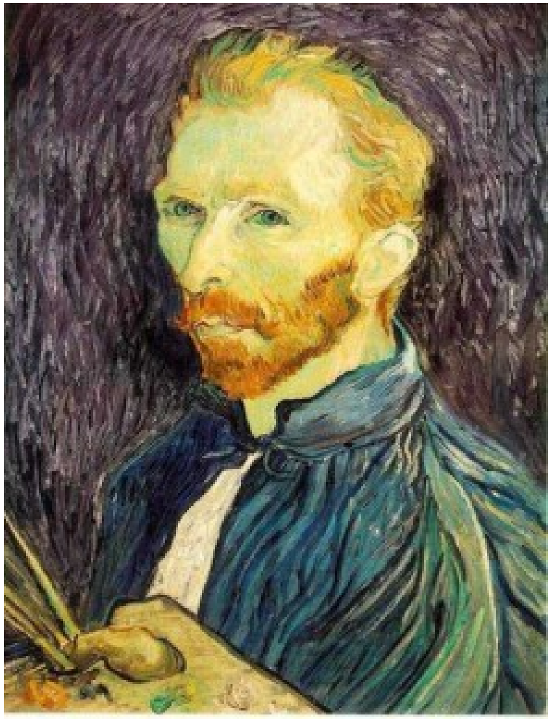
G 1/8

Solution: filter the image, *then* subsample

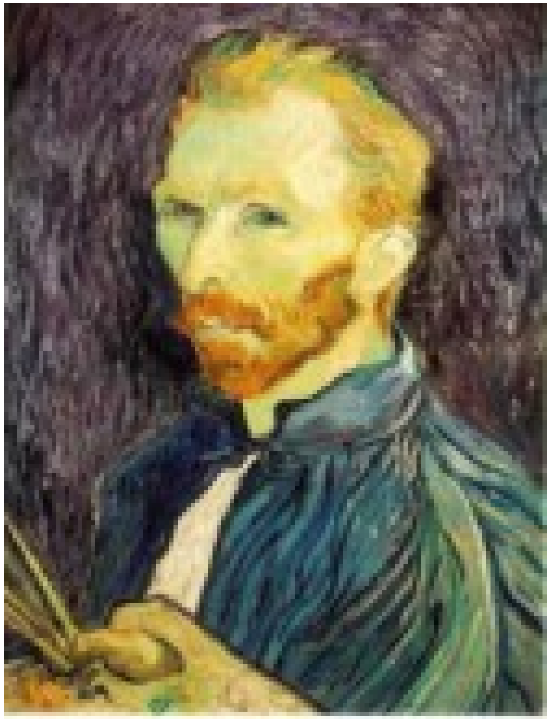
- Filter size should double for each ½ size reduction. Why?

S. Seitz

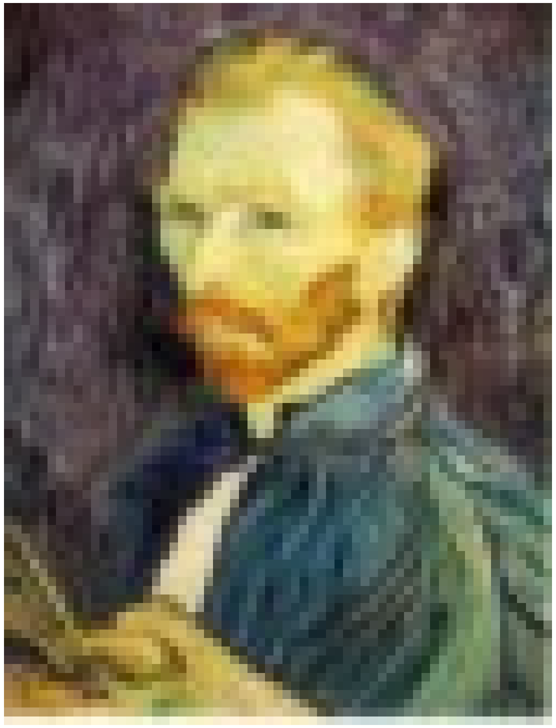
Subsampling with Gaussian pre-filtering



Gaussian 1/2



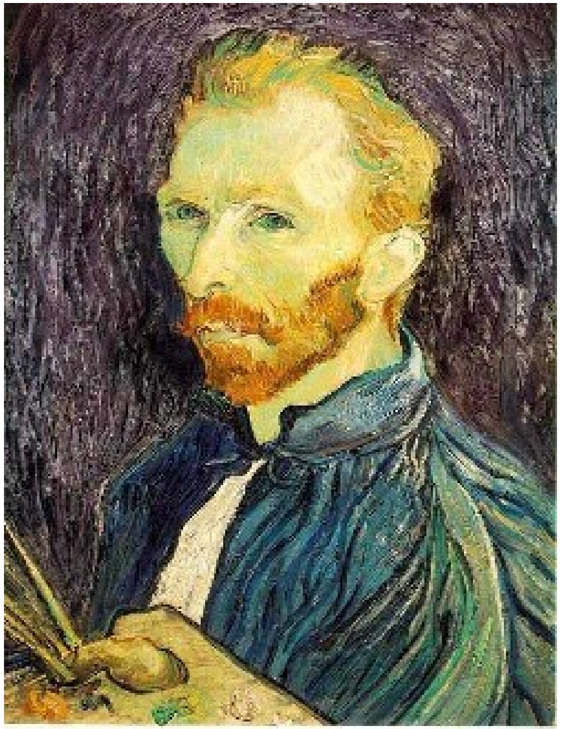
G 1/4



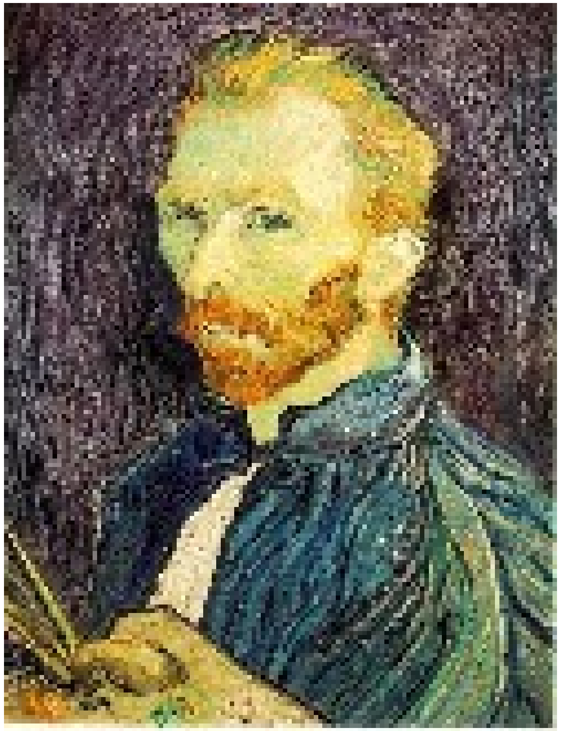
G 1/8

S. Seitz

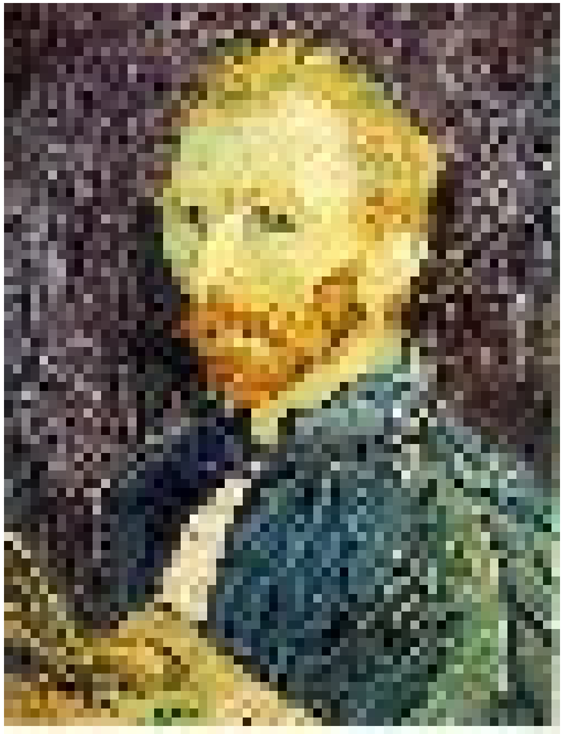
Compare with...



1/2



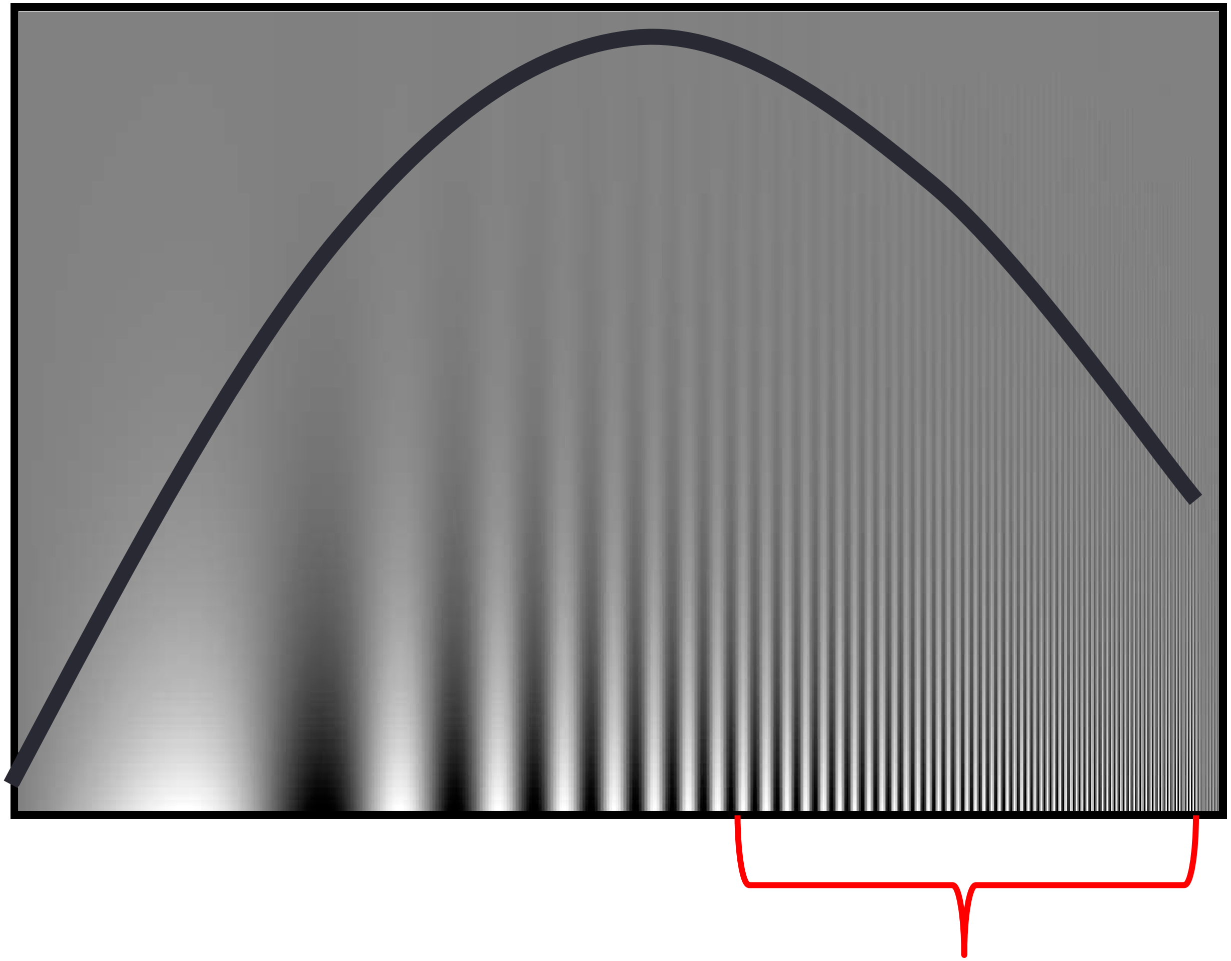
1/4 (2x zoom)



1/8 (4x zoom)

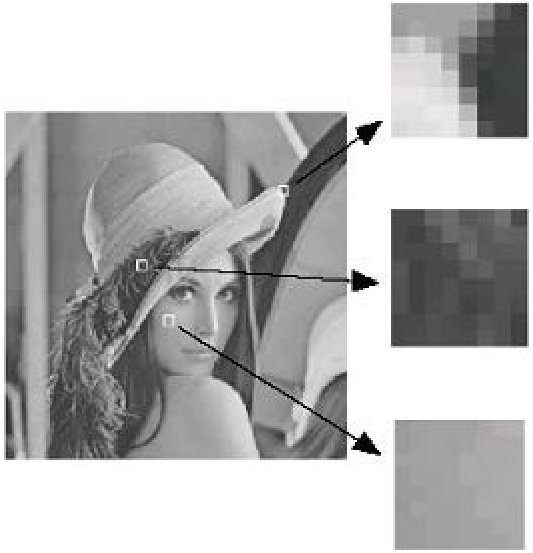
S. Seitz

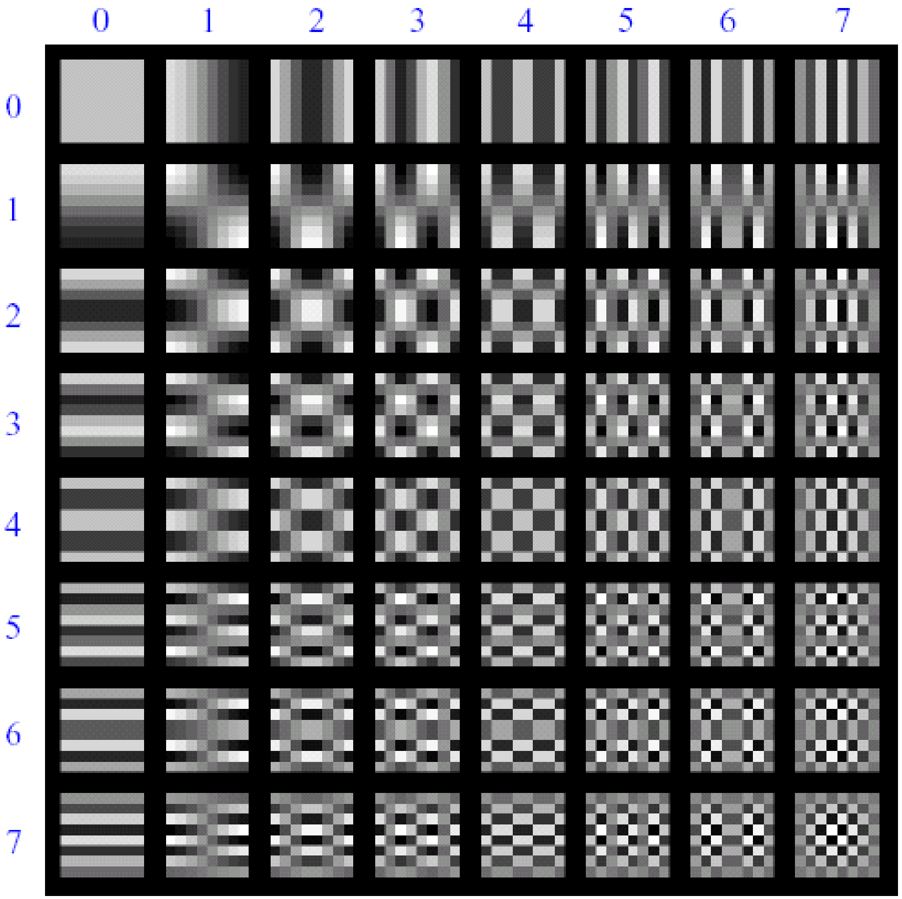
Campbell-Robson contrast sensitivity curve



The higher the frequency the less sensitive human visual system is…

Lossy Image Compression (JPEG)





Block-based Discrete Cosine Transform (DCT) on 8x8

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

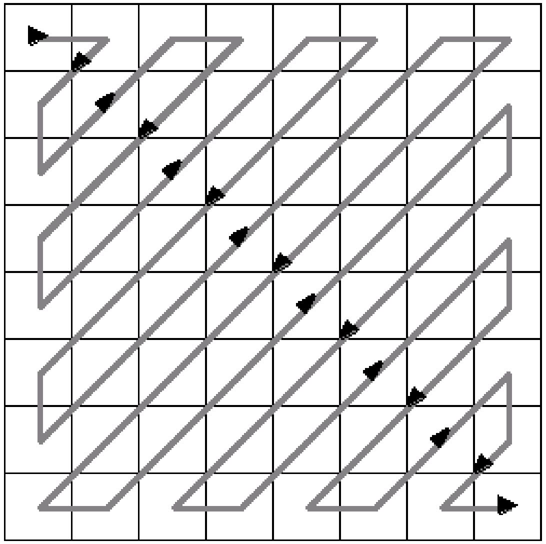


Image compression using DCT

- 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

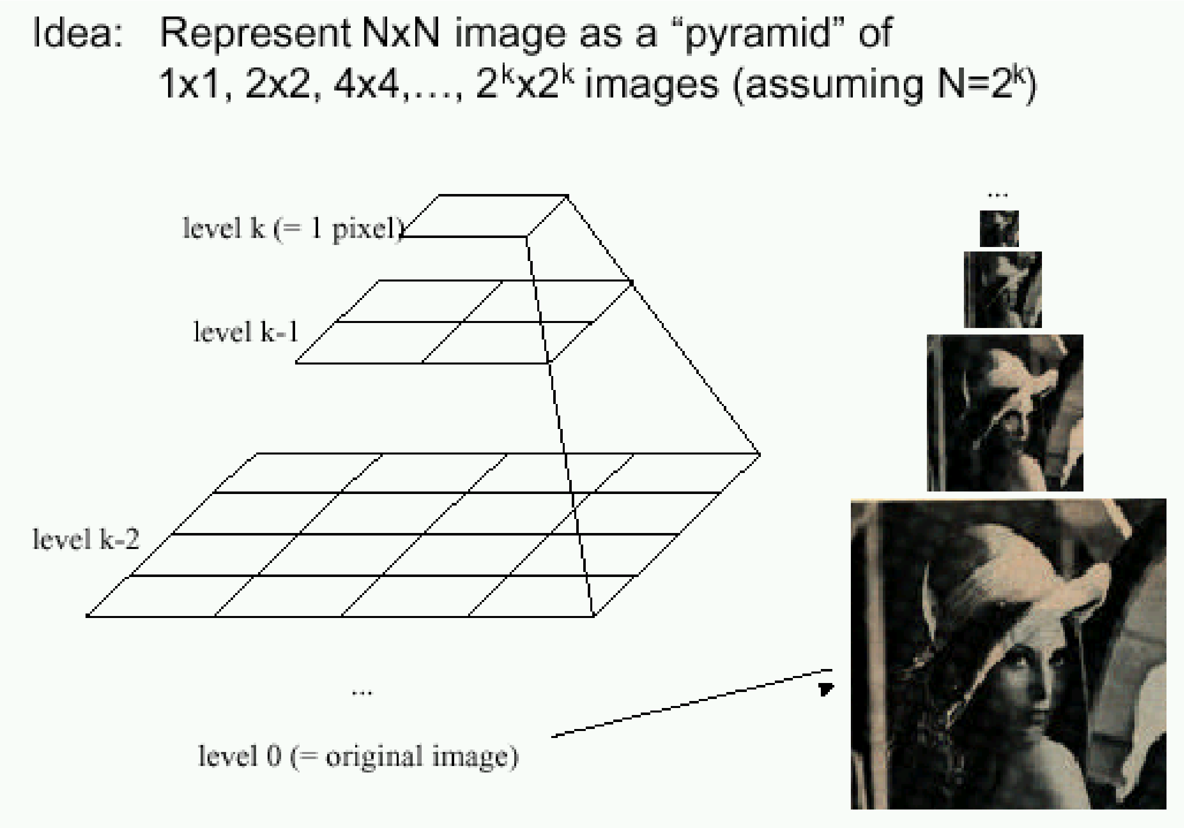
12k

Maybe the end?

Or not!!!

- *A teaser on pyramids…*

Image Pyramids



Idea: Represent NxN image as a "pyramid" of 1x1, 2x2, 4x4,…, 2^kx2^k images (assuming N=2^k)

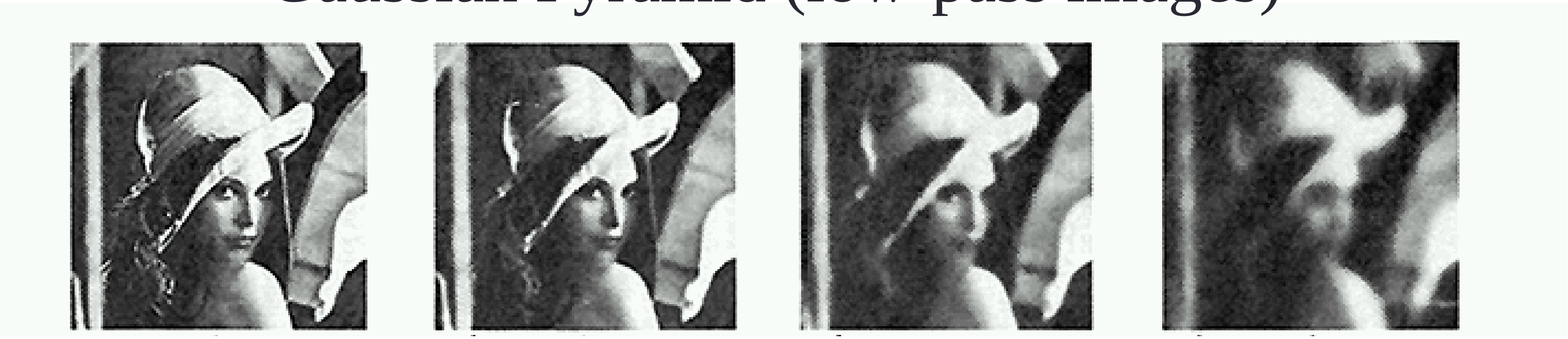
Known as a **Gaussian Pyramid** [Burt and Adelson, 1983]

- In computer graphics, a *mip map* [Williams, 1983]
- A precursor to *wavelet transform*

S. Seitz

Band-pass filtering

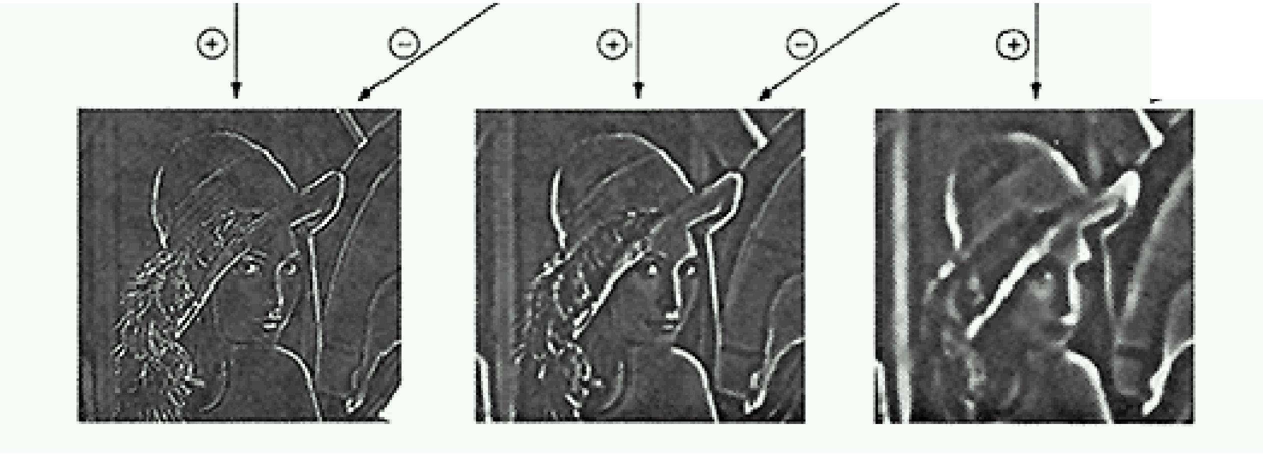
Gaussian Pyramid (low-pass images)



These are "bandpass" images (almost).

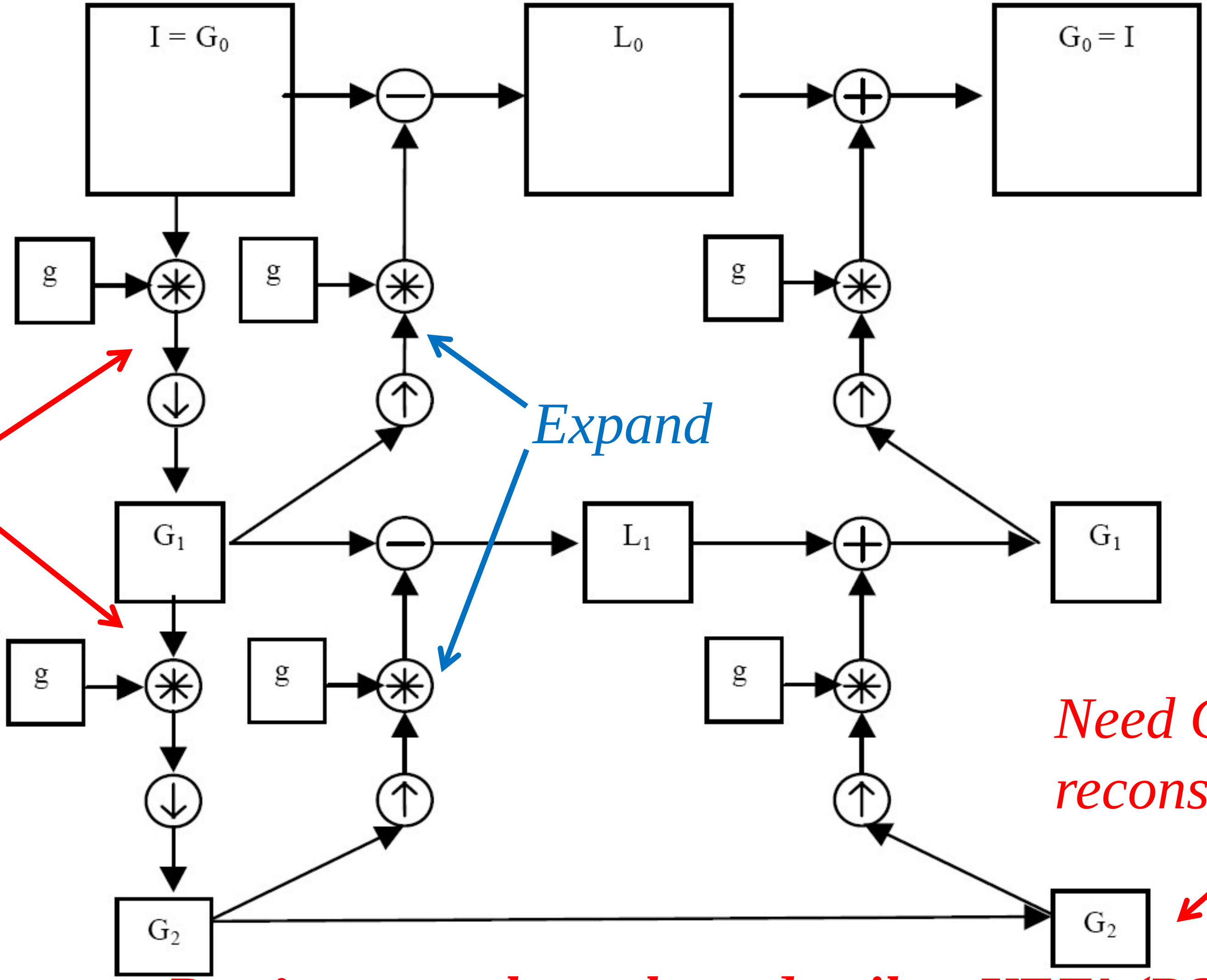
Laplacian Pyramid

Original image



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

Computing the Laplacian Pyramid



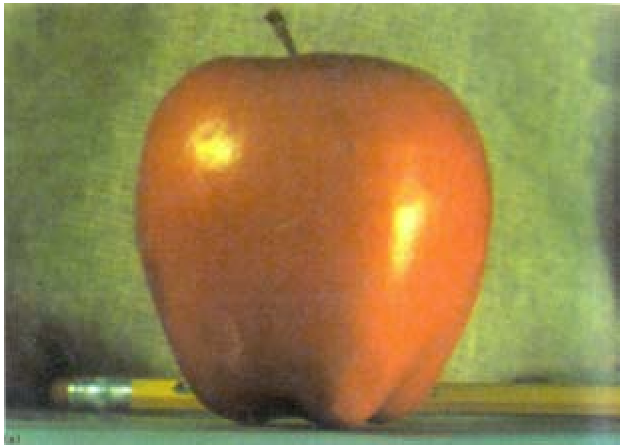
Reduce

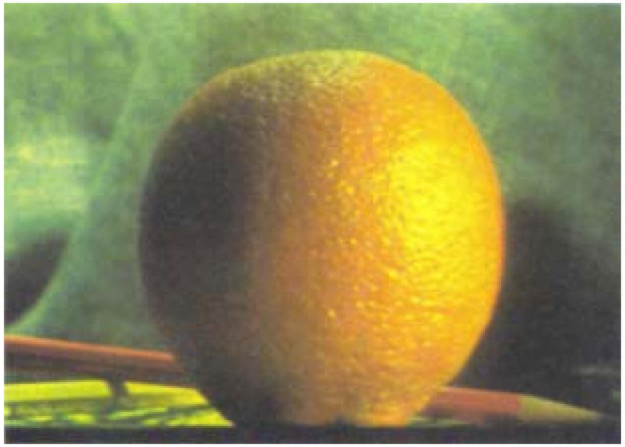
Expand

Need G_k to reconstruct

Don't worry about these details – YET! (PS

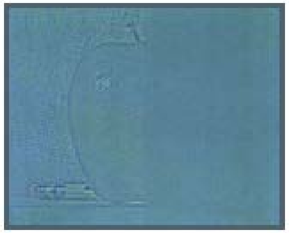
What can you do with band limited imaged?





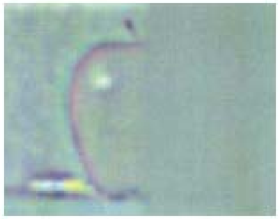
Apples and Oranges in bandpass

Fine L_0



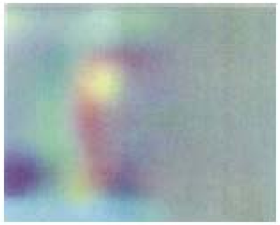
(a)

L_2



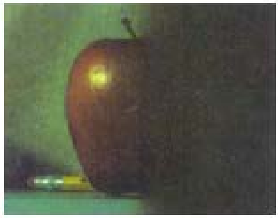
(d)

Coarse L_4



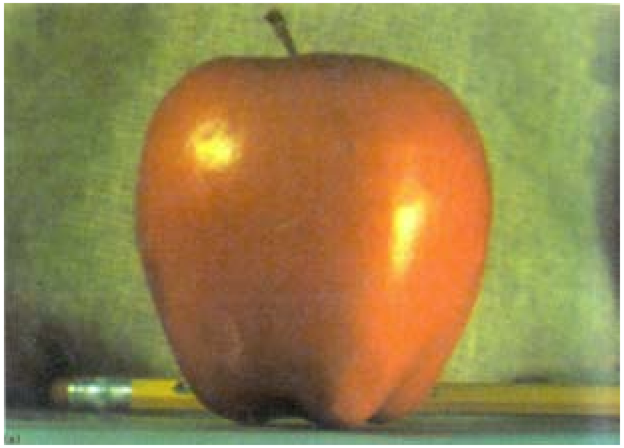
(g)

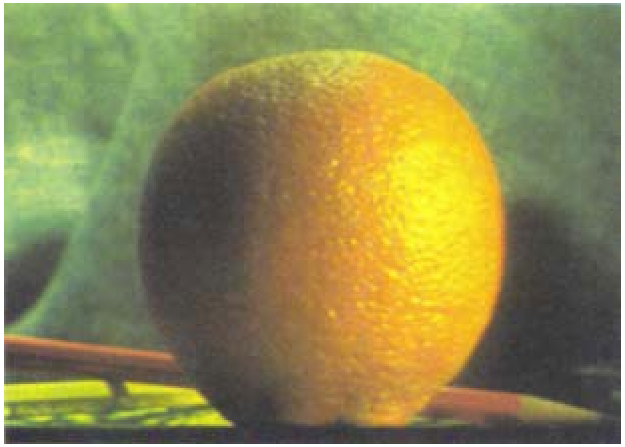
Reconstructed

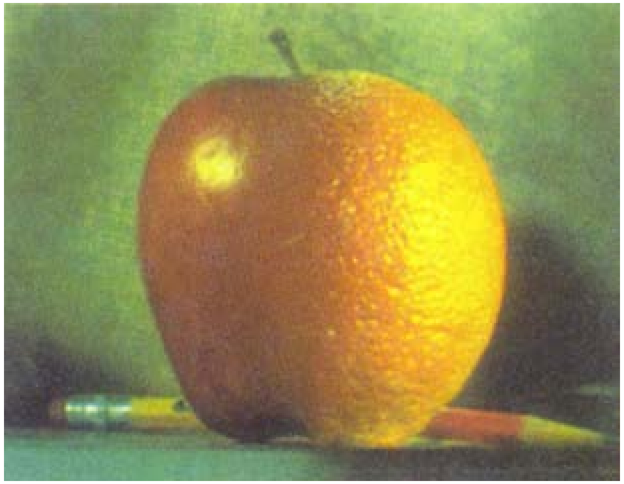


(j)

What can you do with band limited imaged?







Really the end…