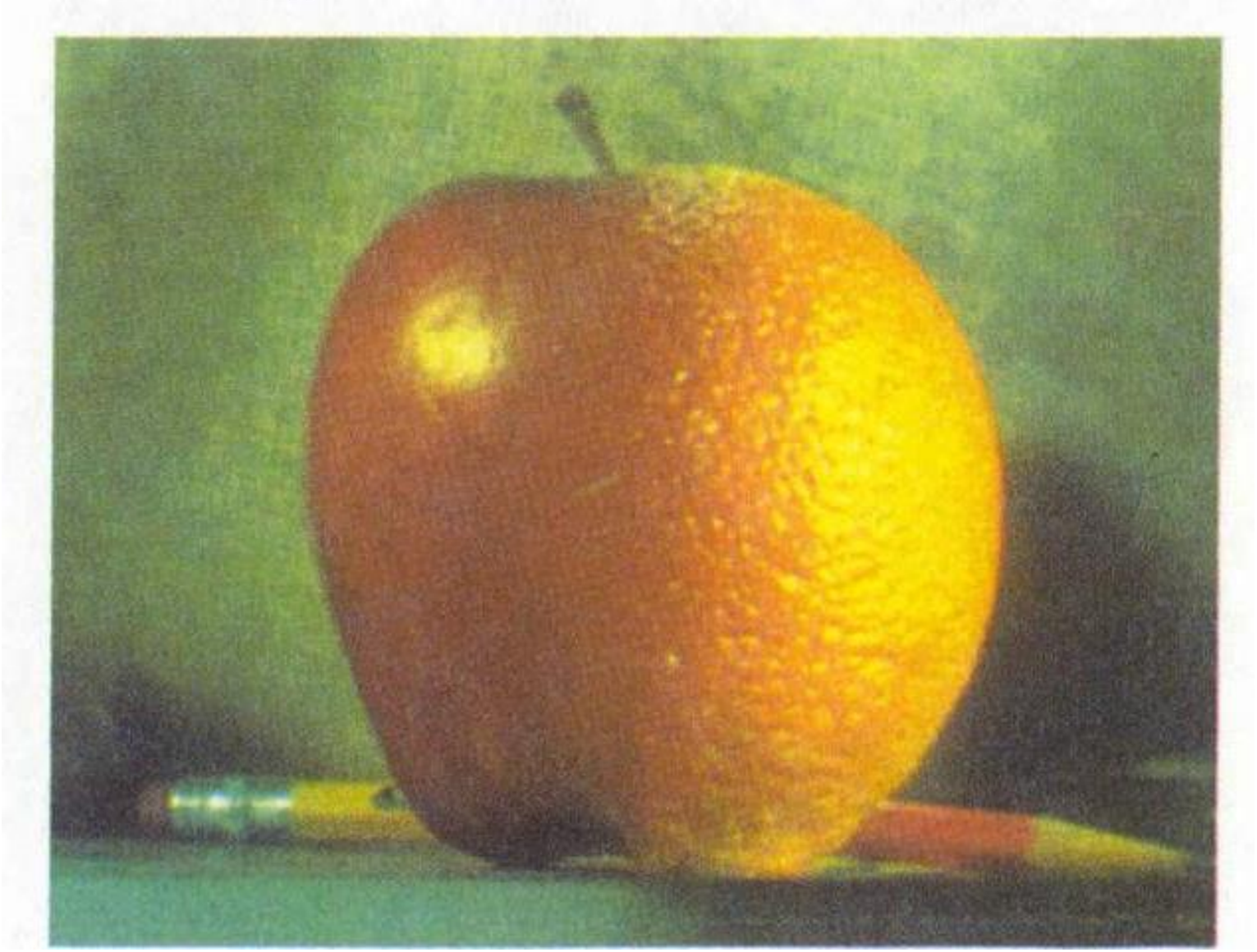
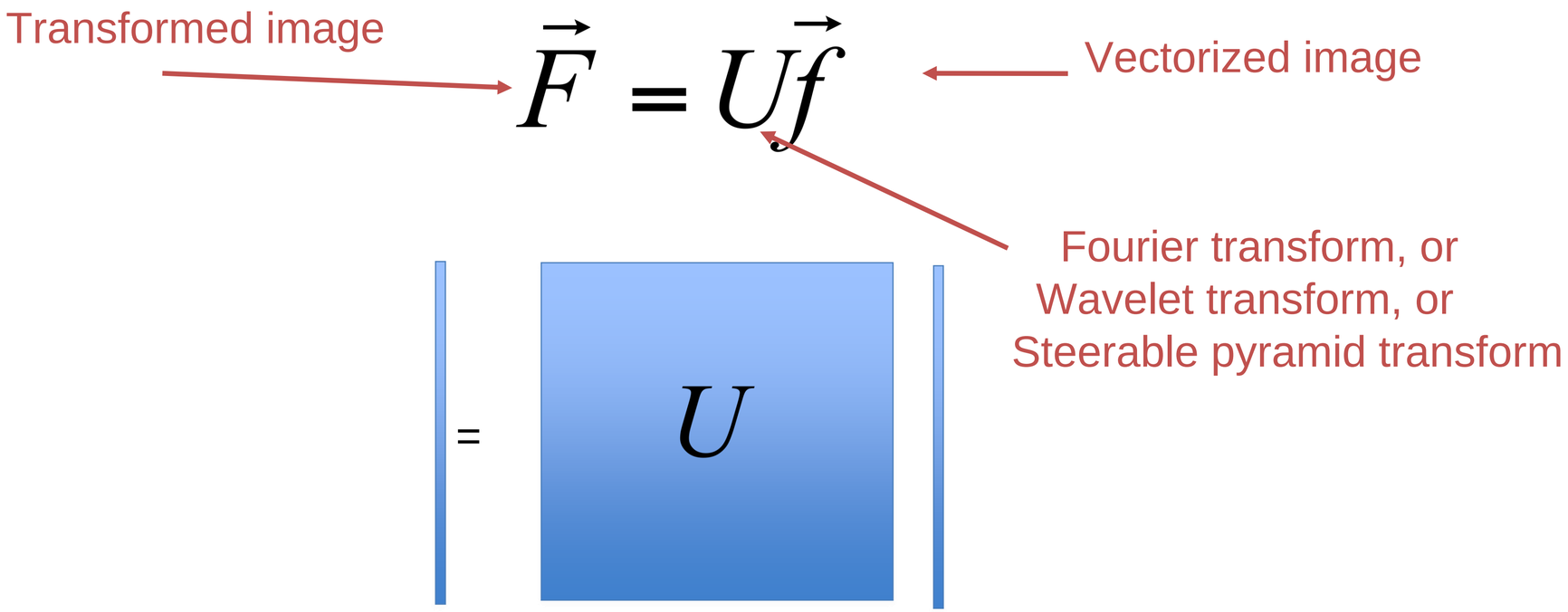
Sampling, Reconstruction, Pyramids



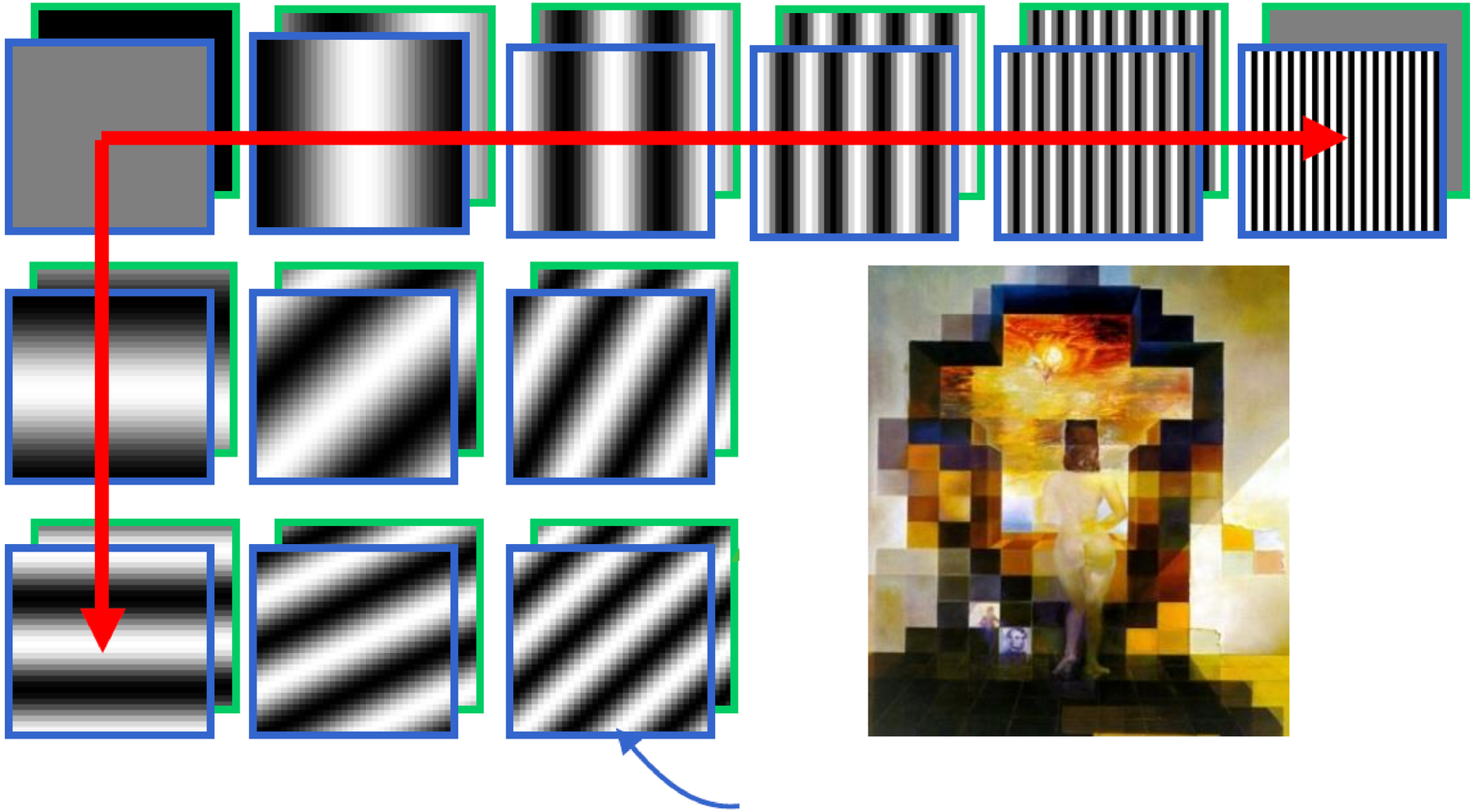
Linear image transformations

In analyzing images, it's often useful to make a change of basis.



A nice set of basis

Teases away fast vs. slow changes in the image.



This change of basis has a special name…

Jean Baptiste Joseph Fourier (1768-1830)

had crazy idea (1807

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

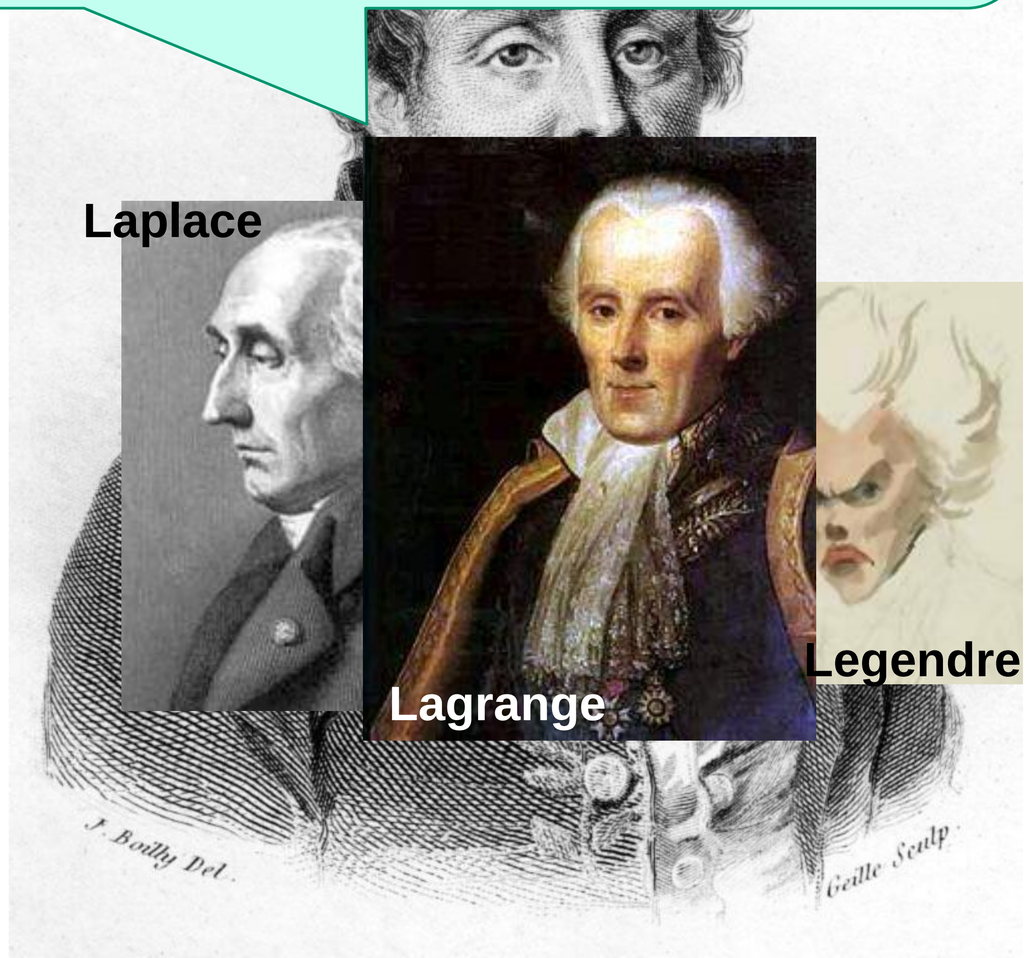
Don't believe it?

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

But it's (mostly) true!

- called Fourier Series

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



A sum of sines

Our building block:

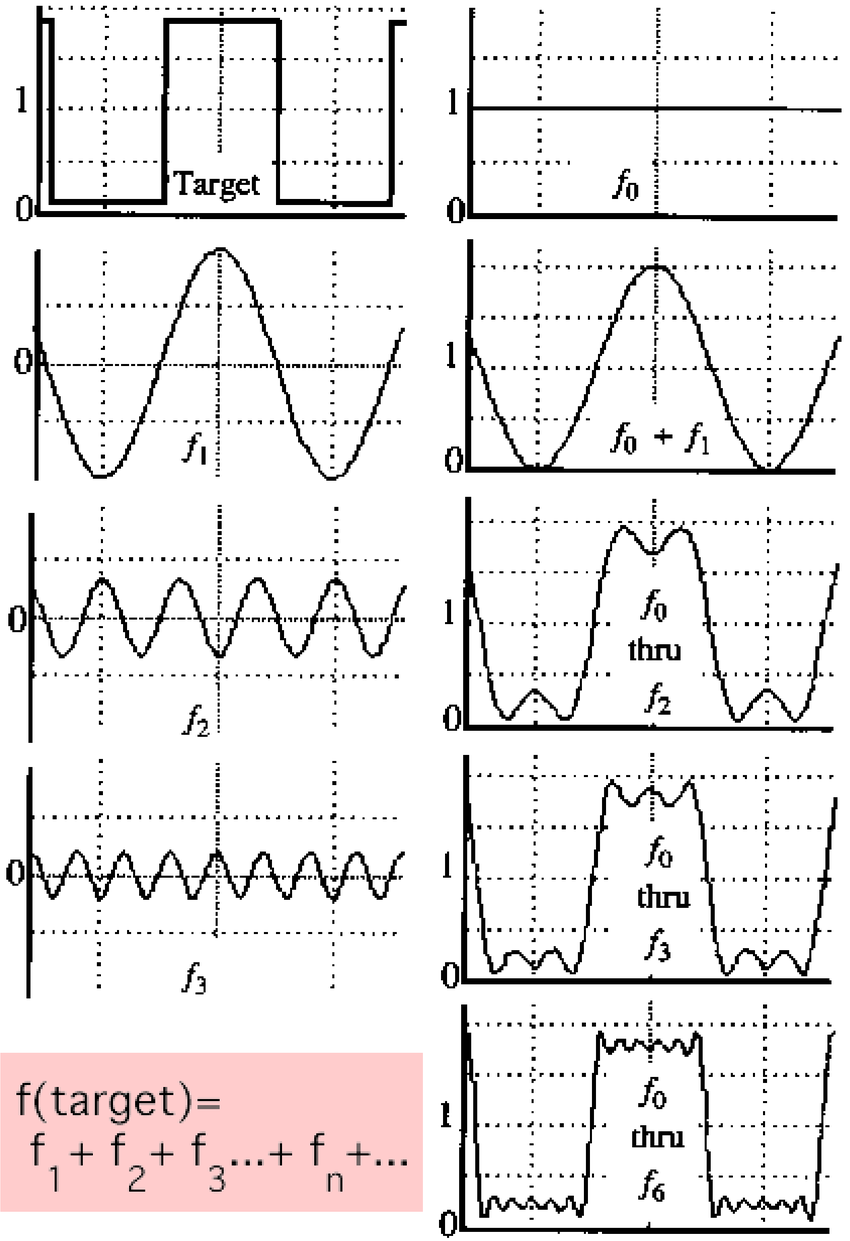
$$A\sin(\omega x + \phi)$$

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

How many degrees of freedom?

What does each control?

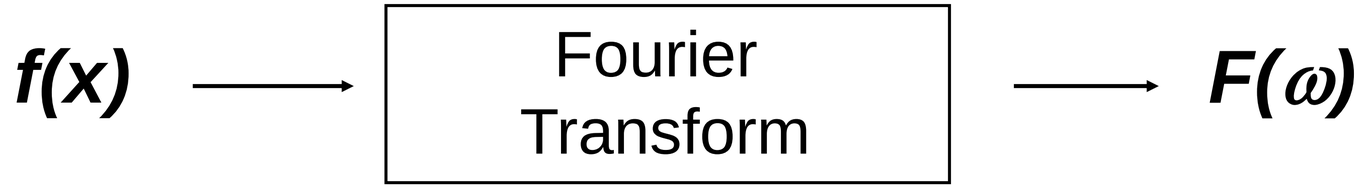
Which one encodes the coarse vs. fine structure of the signal?



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

Fourier Transform

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



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

- How does *F* hold both?

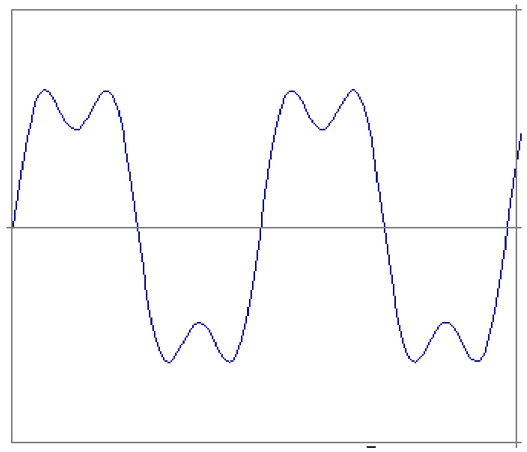
$$F(\omega) = R(\omega) + iI(\omega)$$

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

We can always go back:

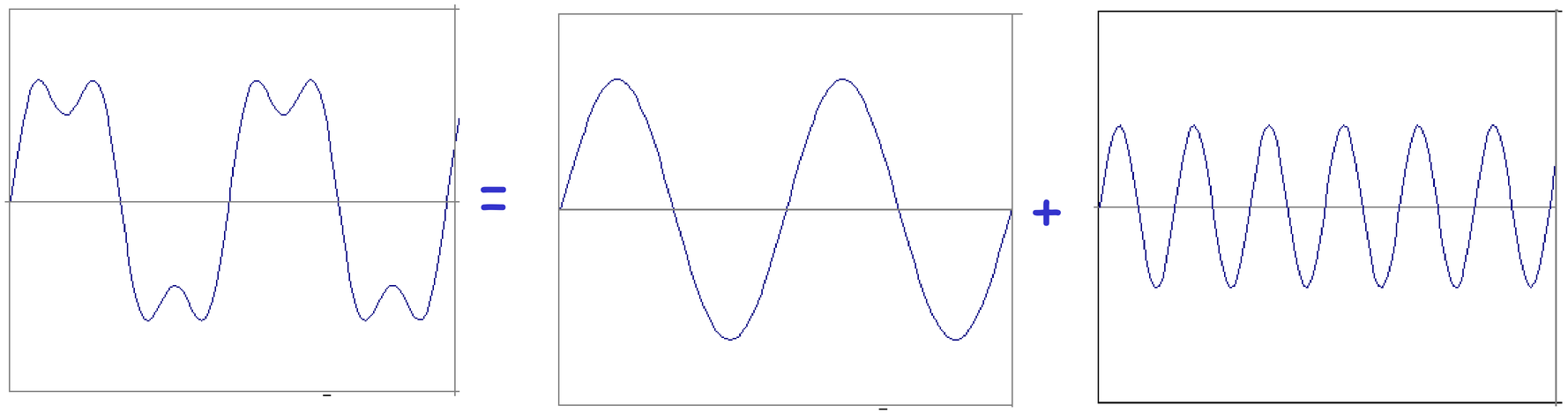
Time and Frequency

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



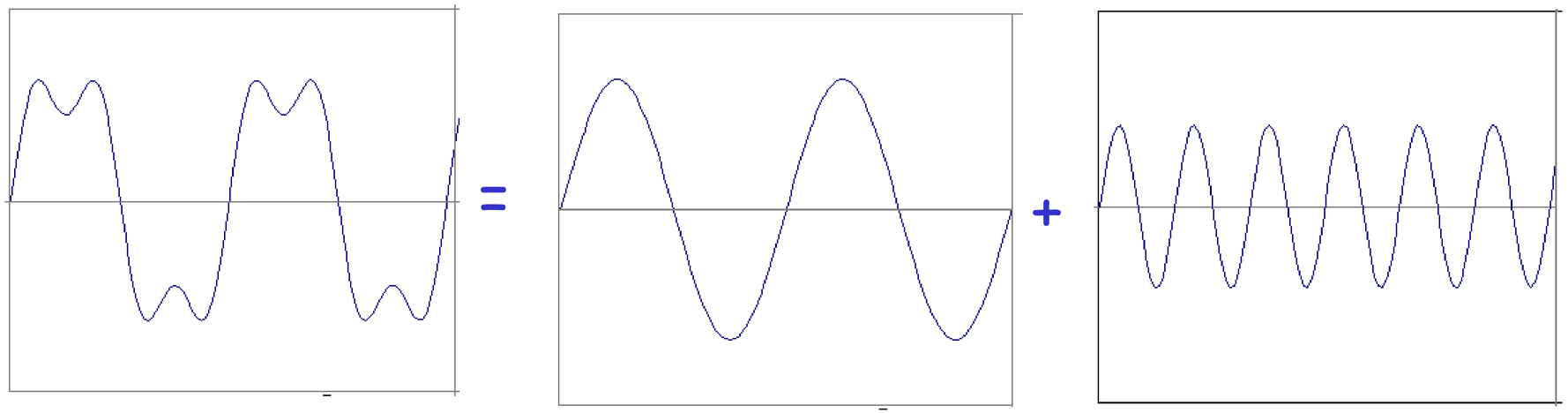
Time and Frequency

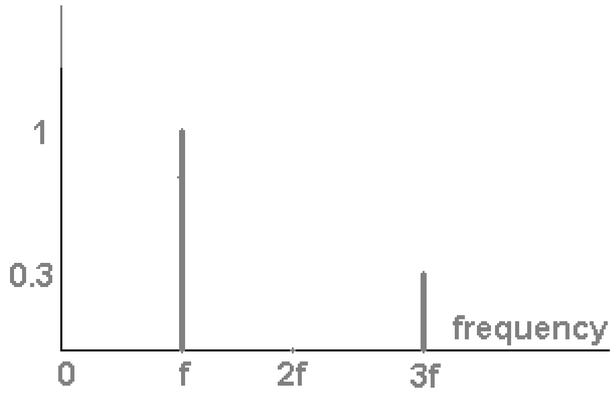
example : $g(t) = \sin(2\pi f t) + (1/3)\sin(2\pi(3f) t)$



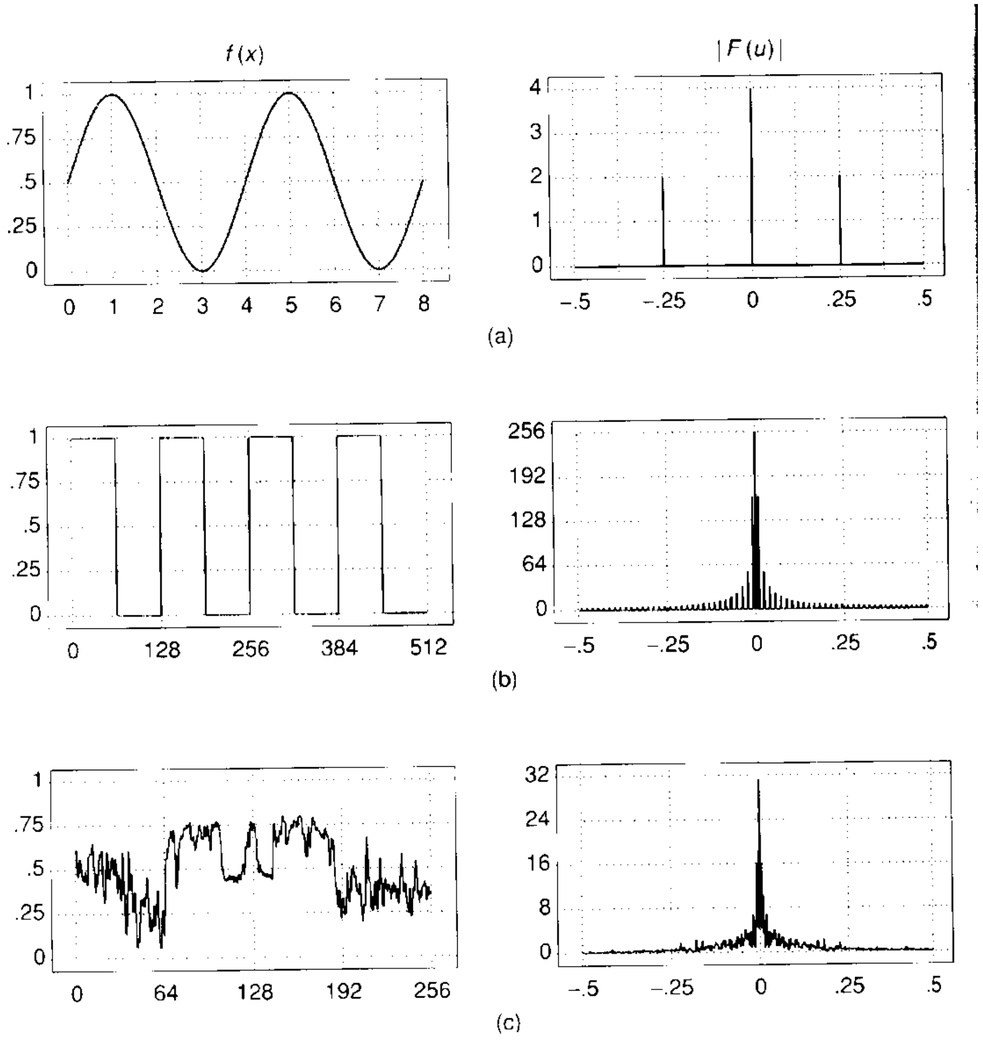
Frequency Spectra

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

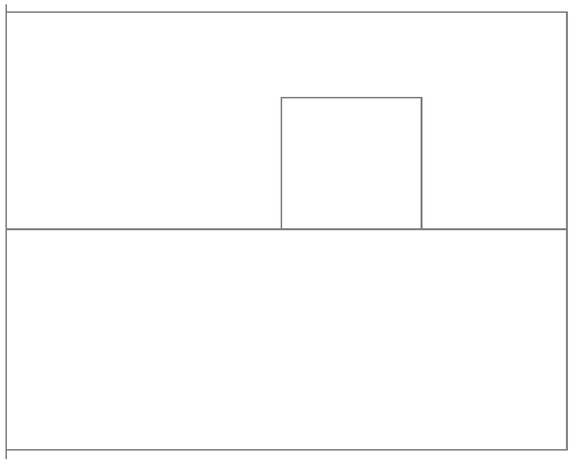




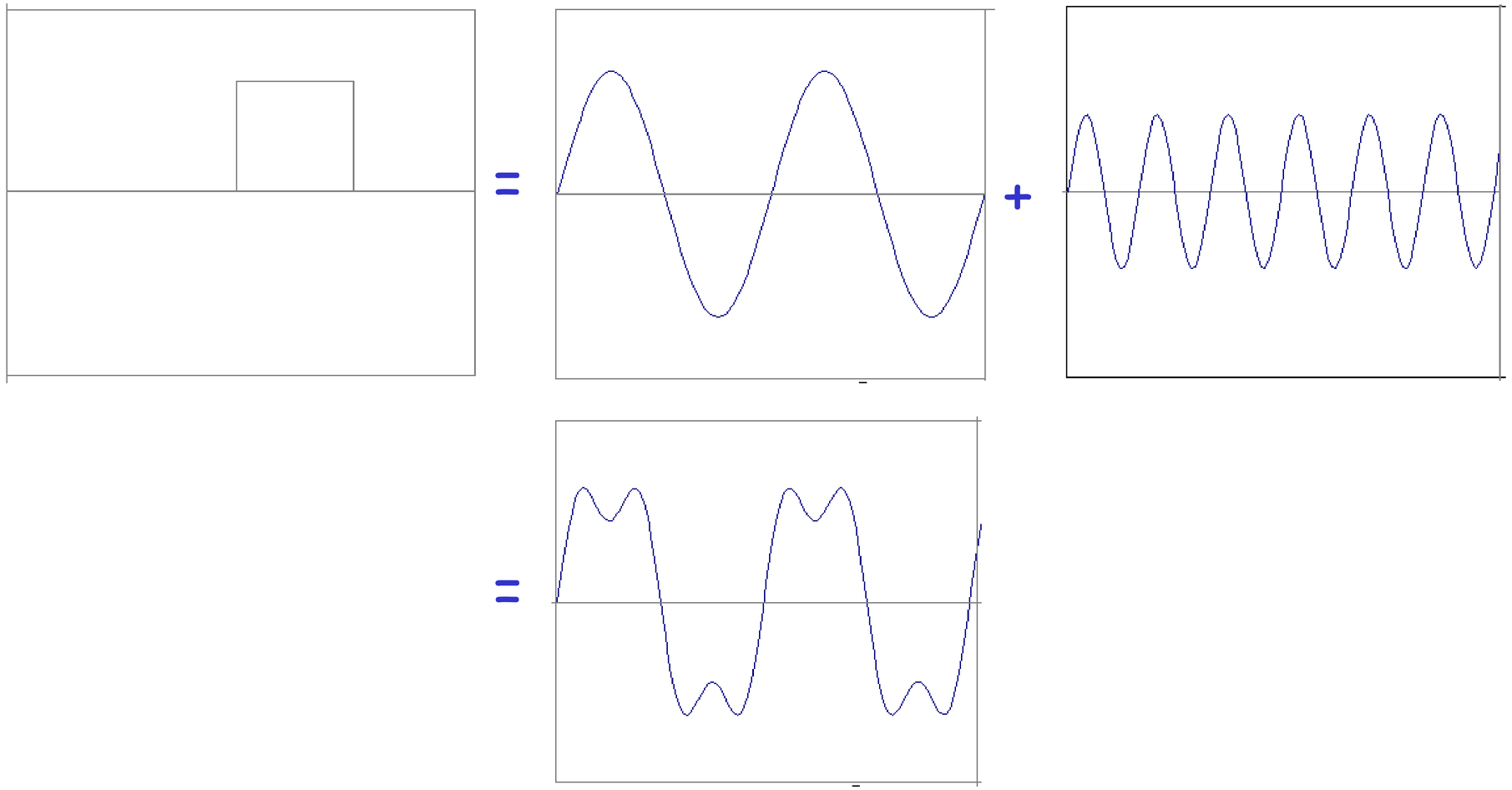
Frequency Spectra



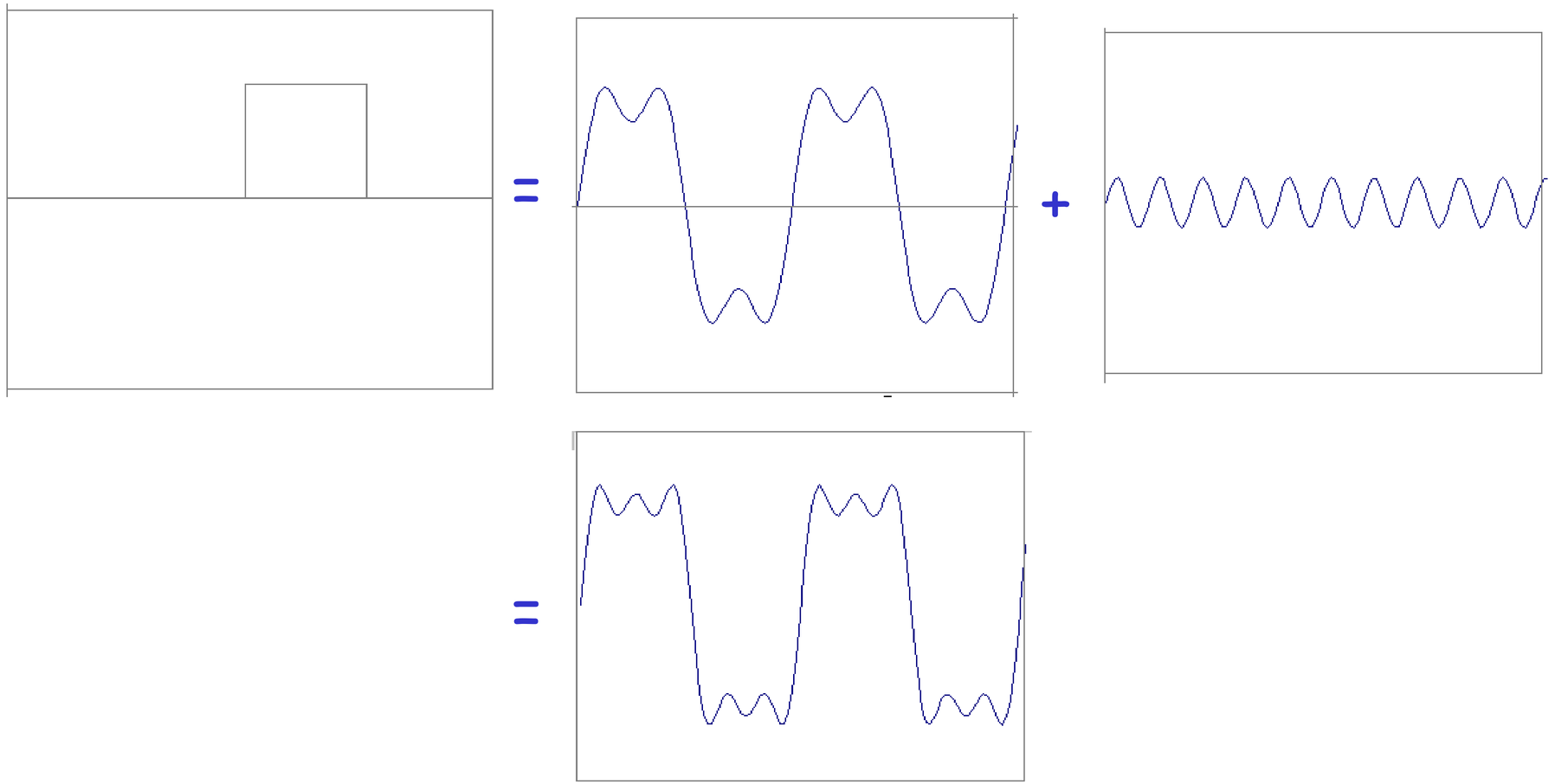
Frequency Spectra



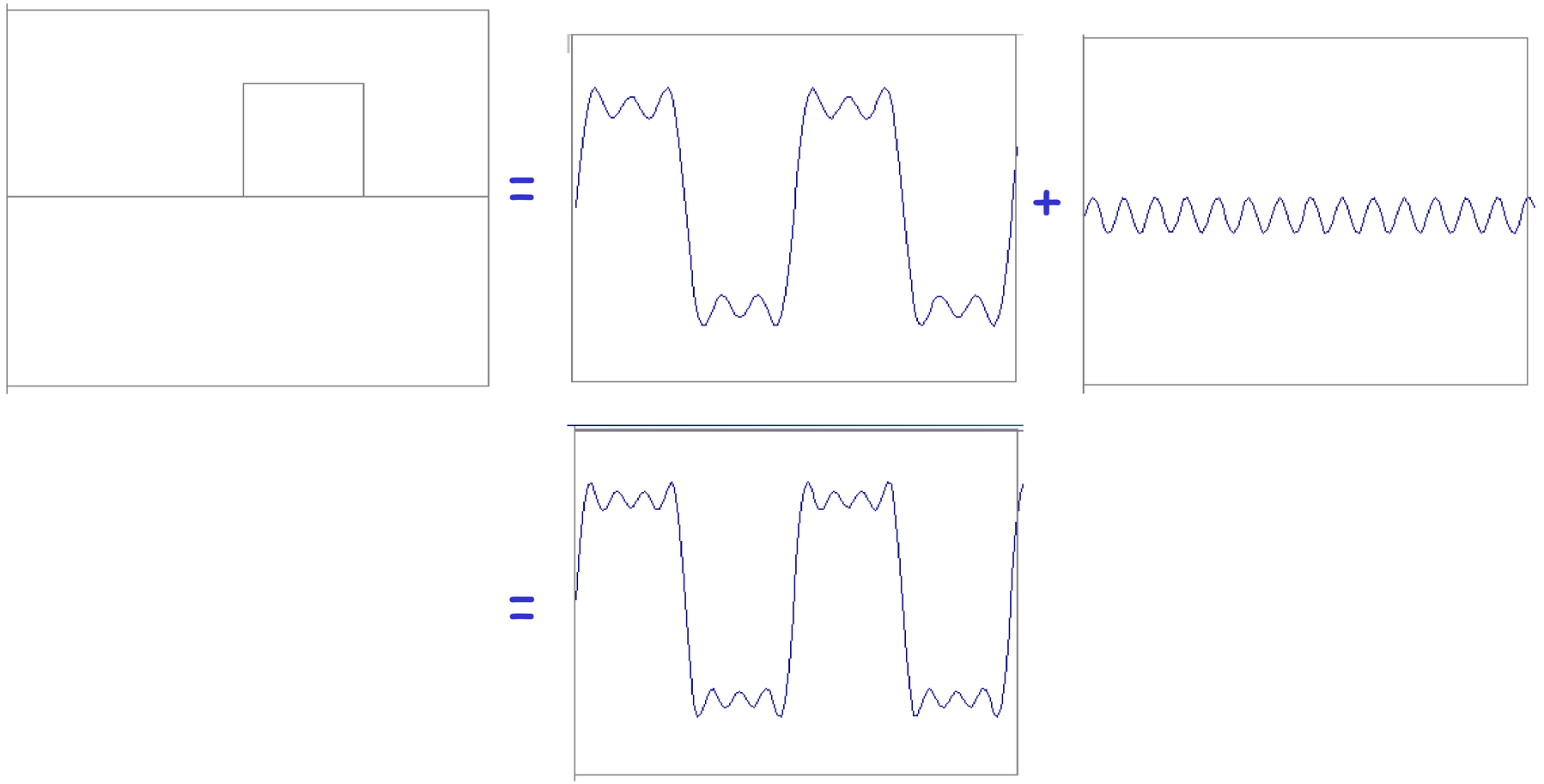
Frequency Spectra



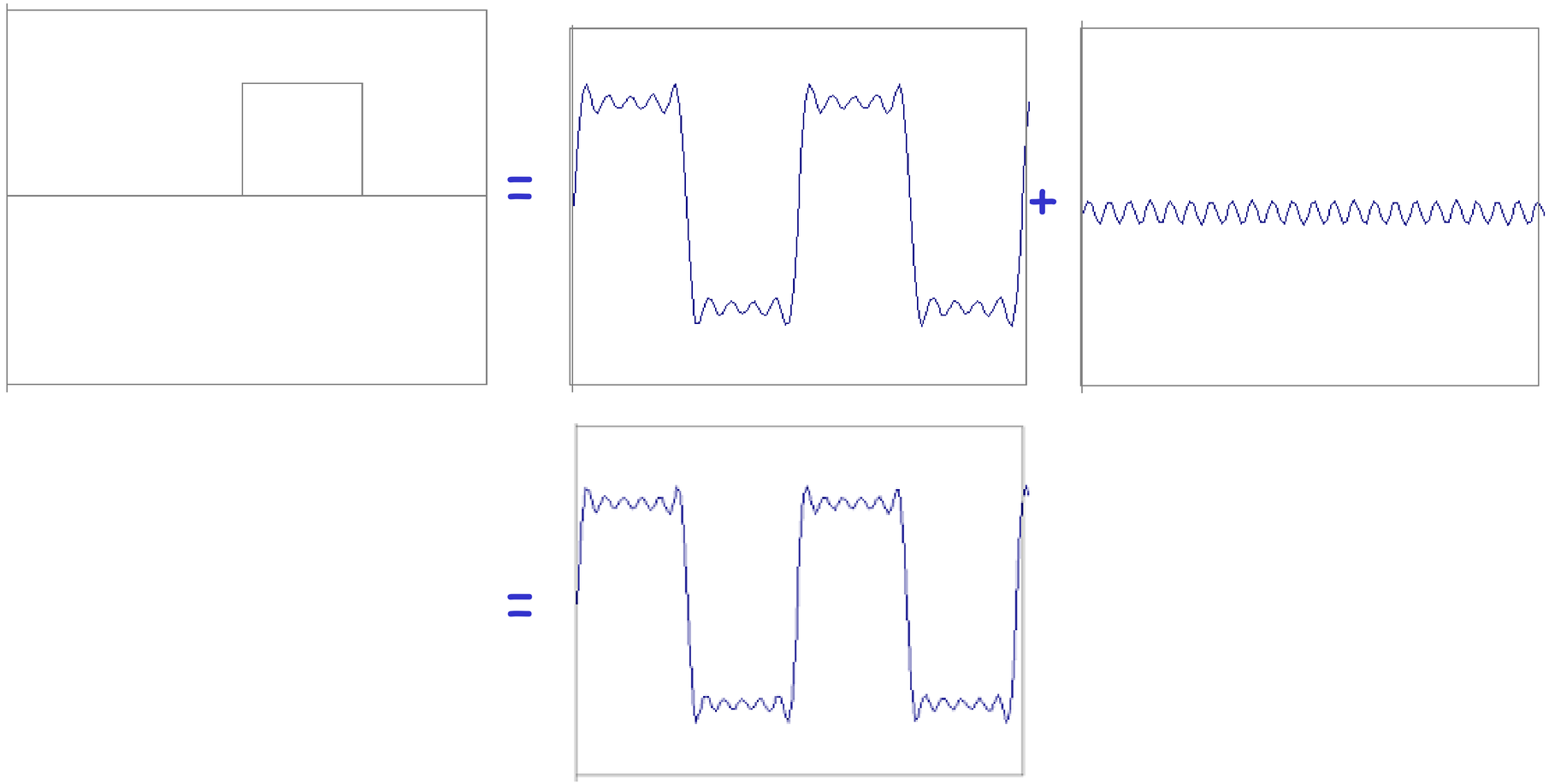
Frequency Spectra



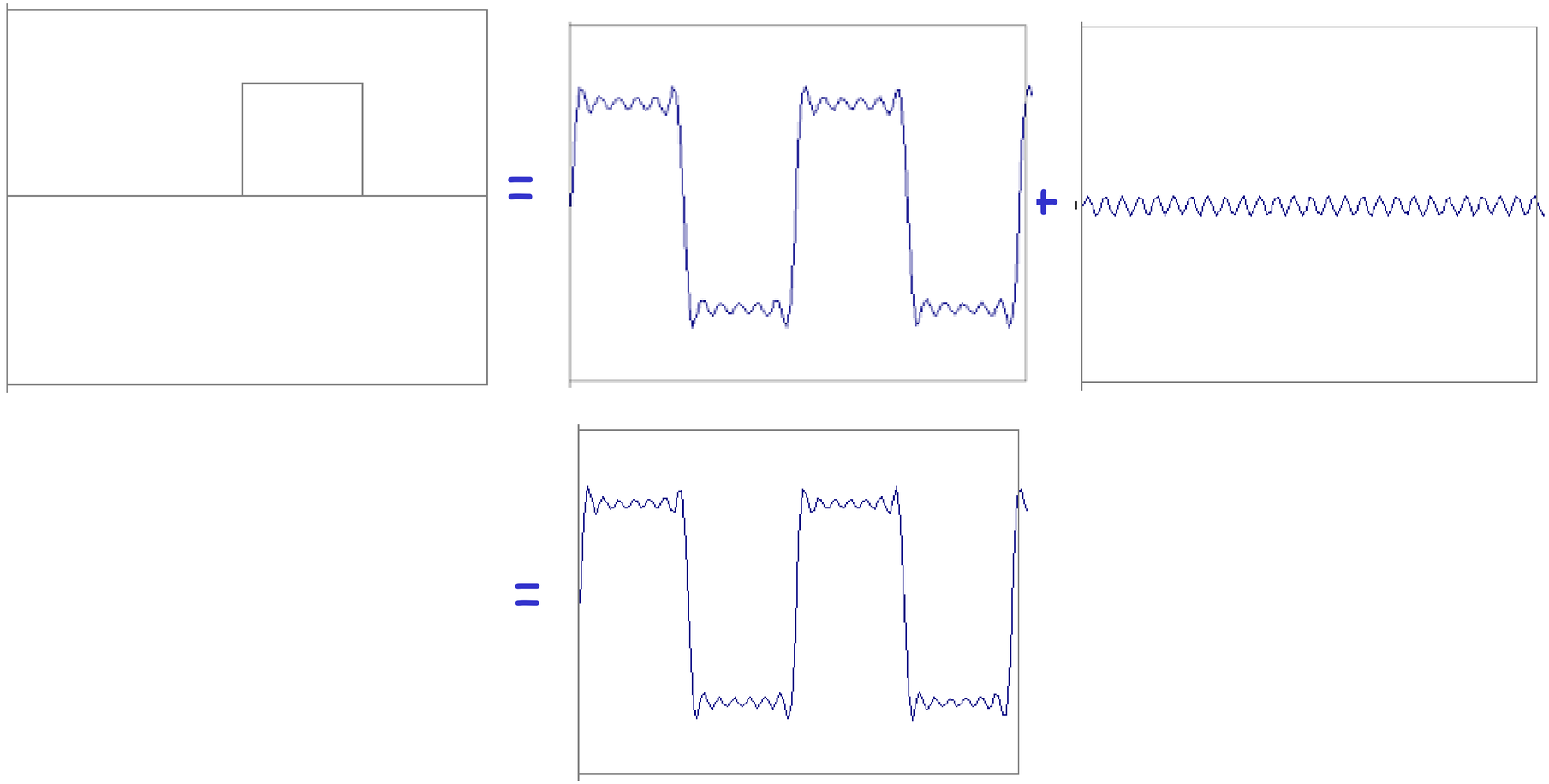
Frequency Spectra



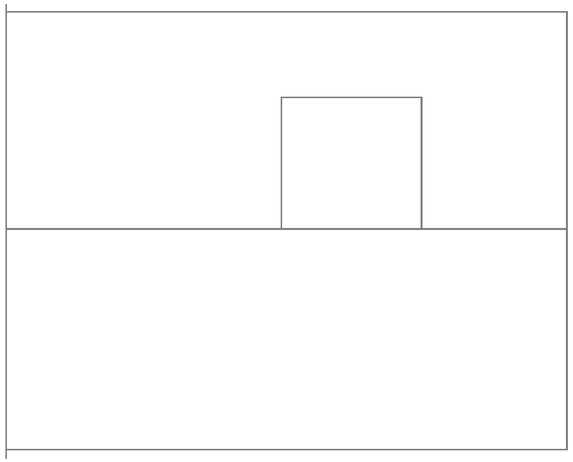
Frequency Spectra



Frequency Spectra

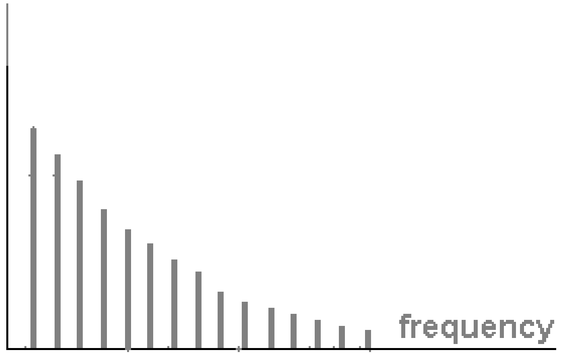


Frequency Spectra

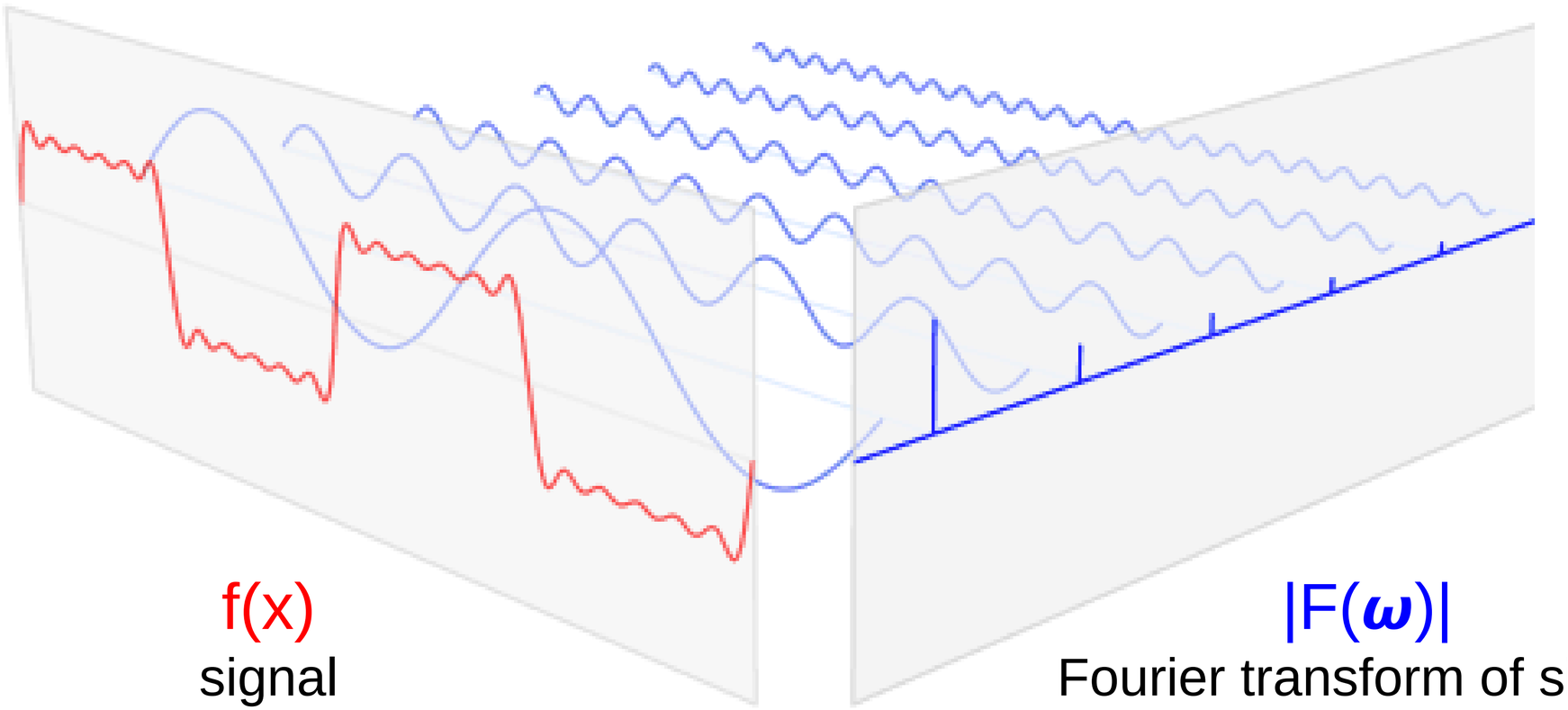


=

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



Furrier Transform

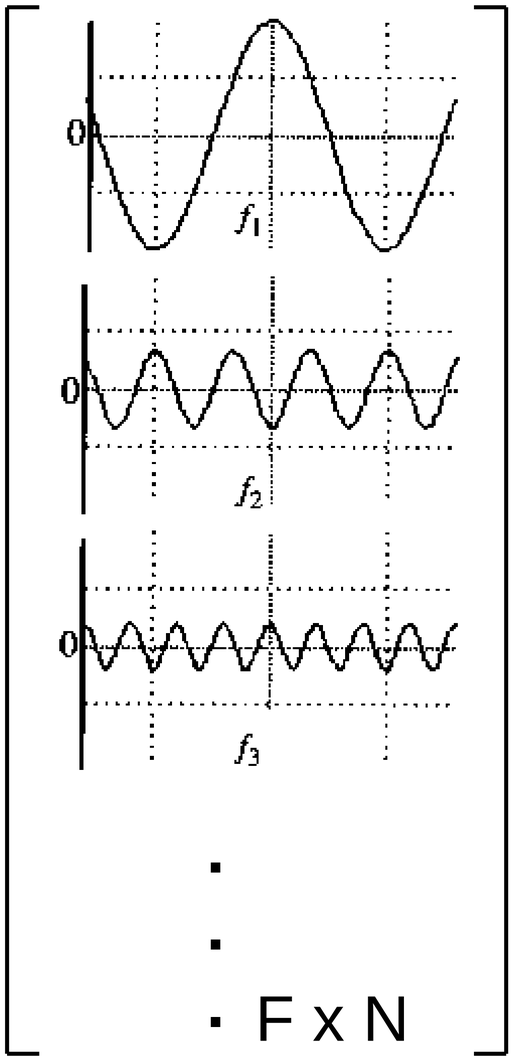


f(x)
signal

|F(***ω***)|
Fourier transform of signal

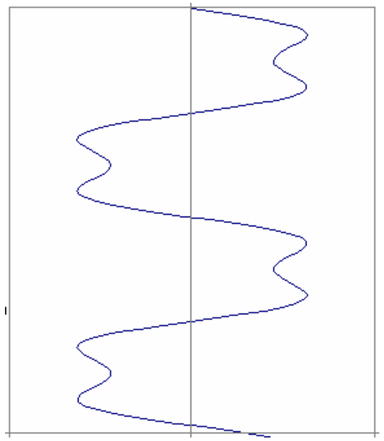
FT: Just a change of basis

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



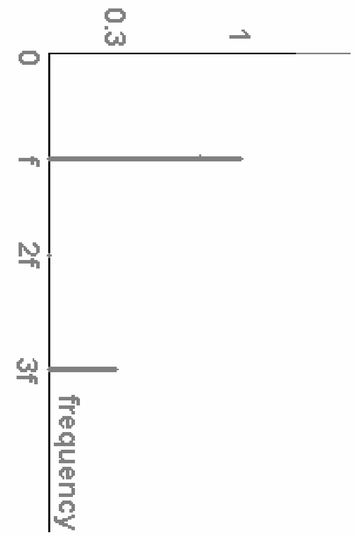
F x N

*



N x 1

=



F x 1

Self-inverting transforms

$$\vec{F} = U\vec{f} \quad \Longleftrightarrow \quad \vec{f} = U^{-1}\vec{F}$$

Same basis functions are used for the inverse transform

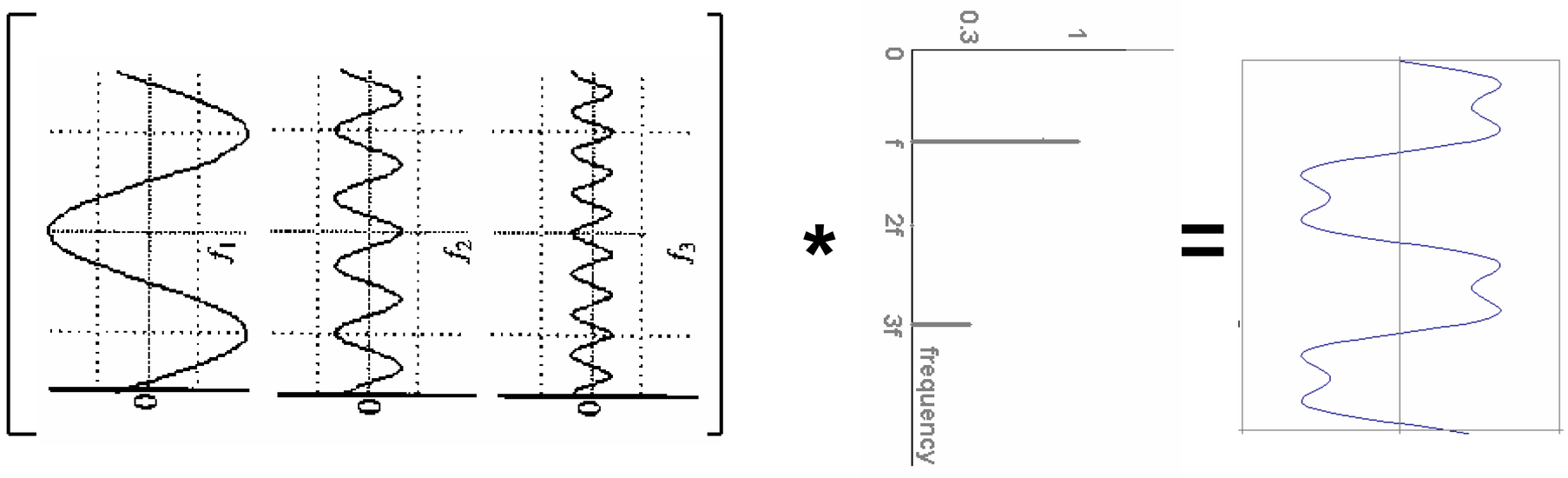
$$\vec{f} = U^{-1}\vec{F}$$

$$= U^{+}\vec{F}$$

U transpose and complex conjugate

IFT: Just a change of basis

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



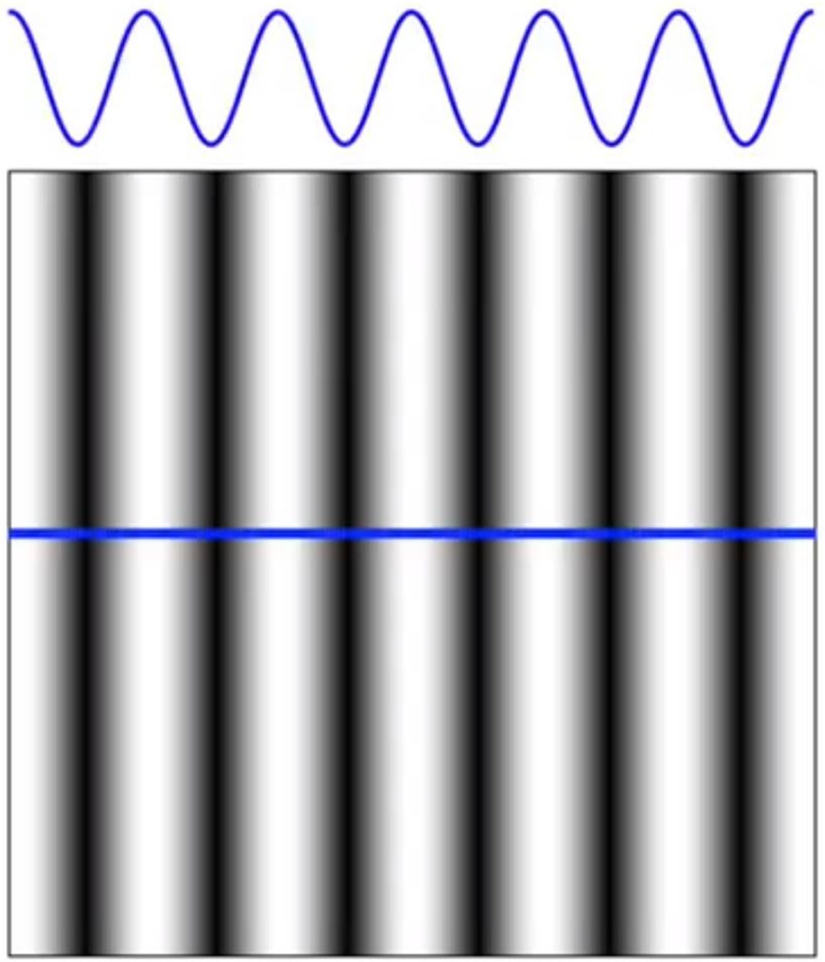
.
.
. N x F F x 1 N x 1

Finally: Scary Math

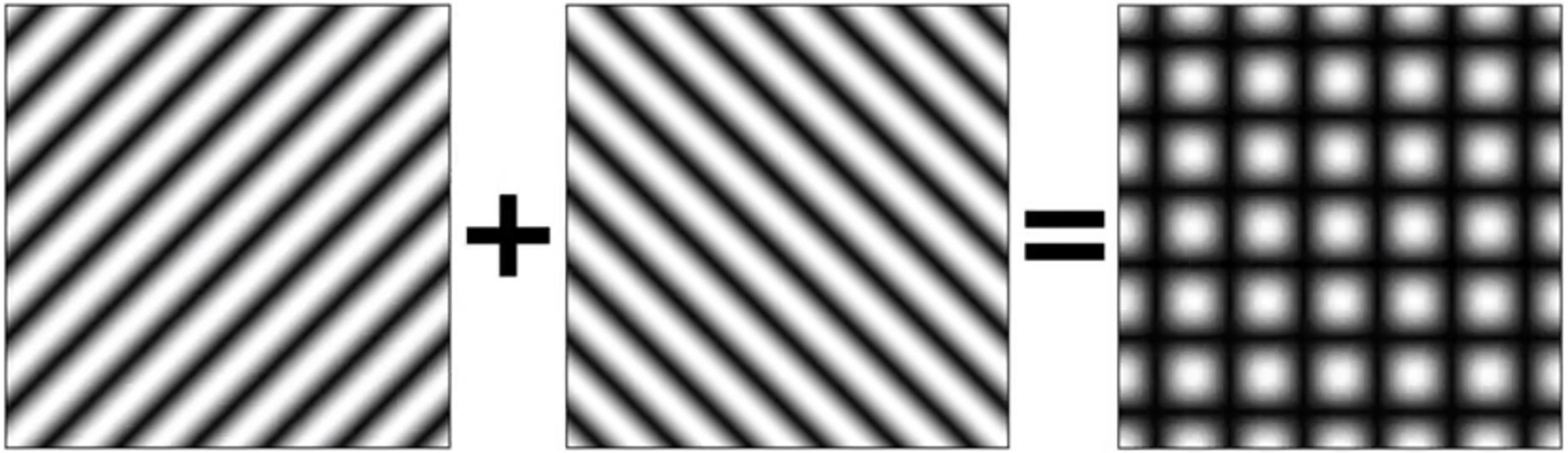
$$\text{Fourier Transform}: \quad F(\omega) = \int_{-\infty}^{+\infty} f(x) e^{-i\omega x} dx$$

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

Extending to 2D



Addition still works in 2D



Extension to 2D

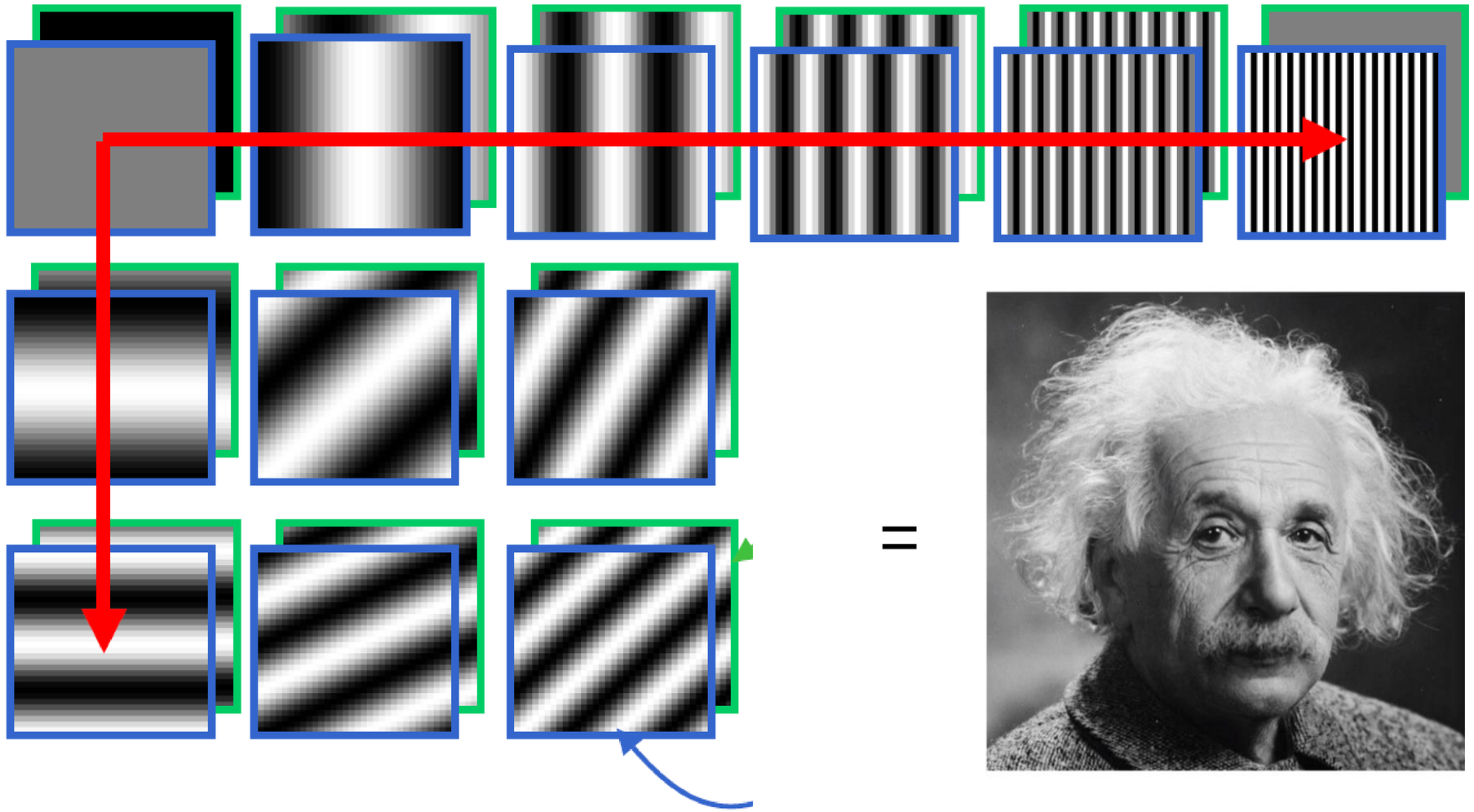


Image as a sum of basis images

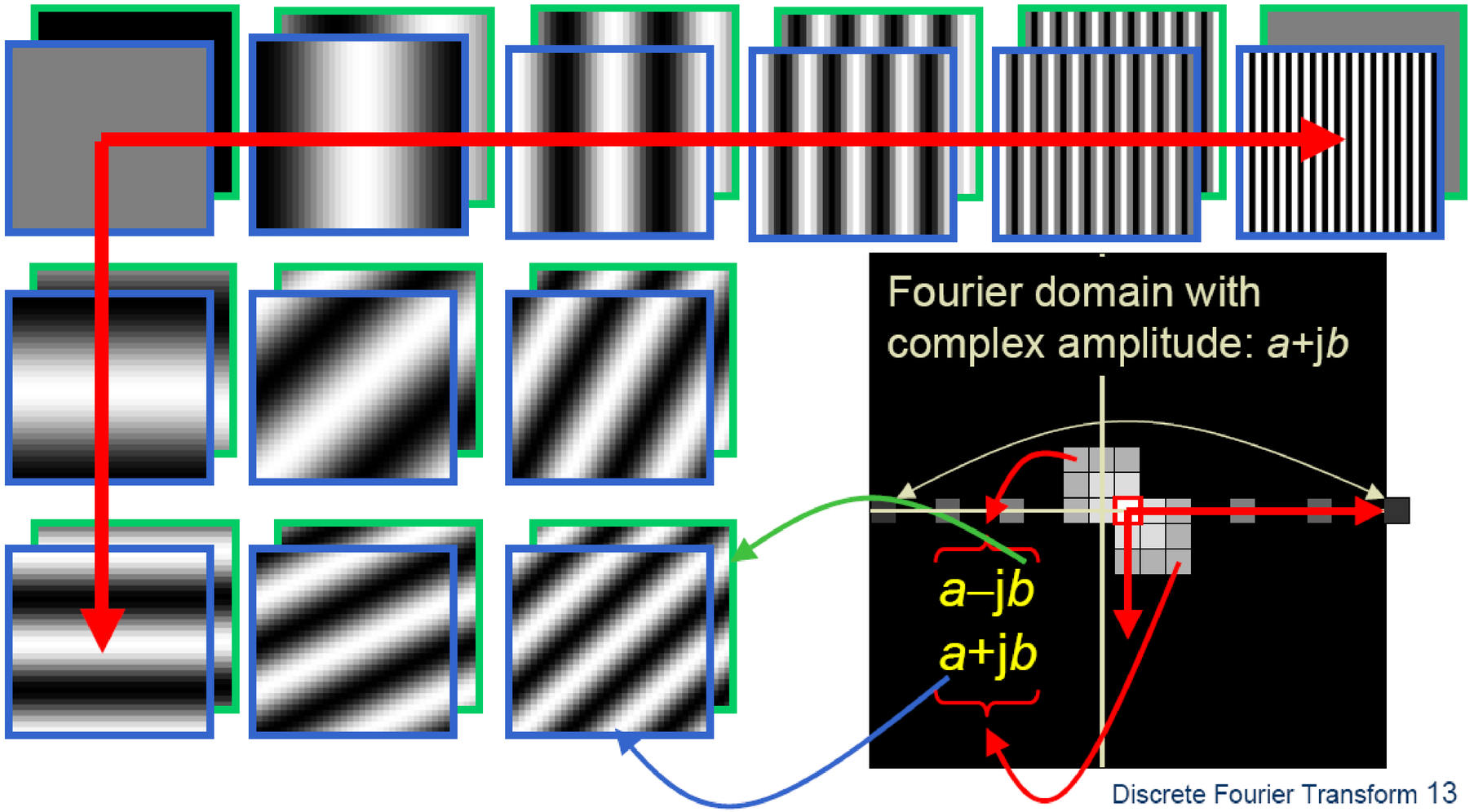
Contrast x3

2

4


100

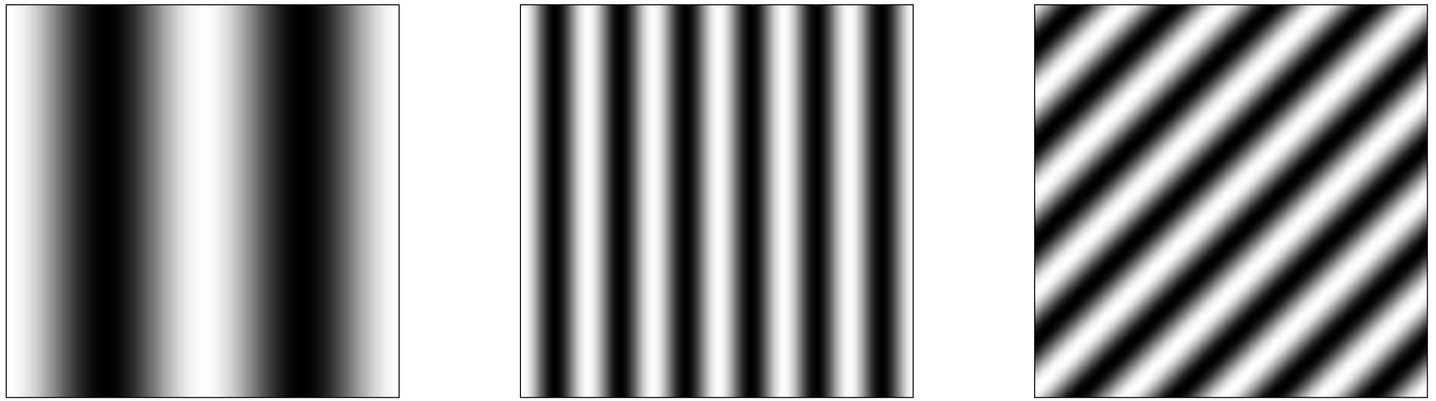
Extension to 2D



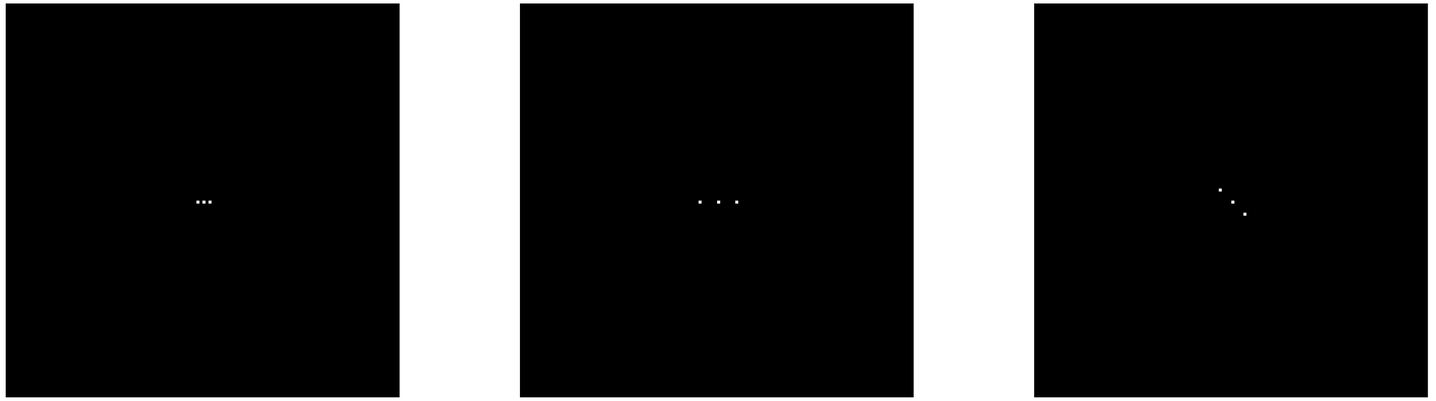
in Matlab, check out: imagesc(log(abs(fftshift(fft2(im)))));

Fourier analysis in images

Intensity Image

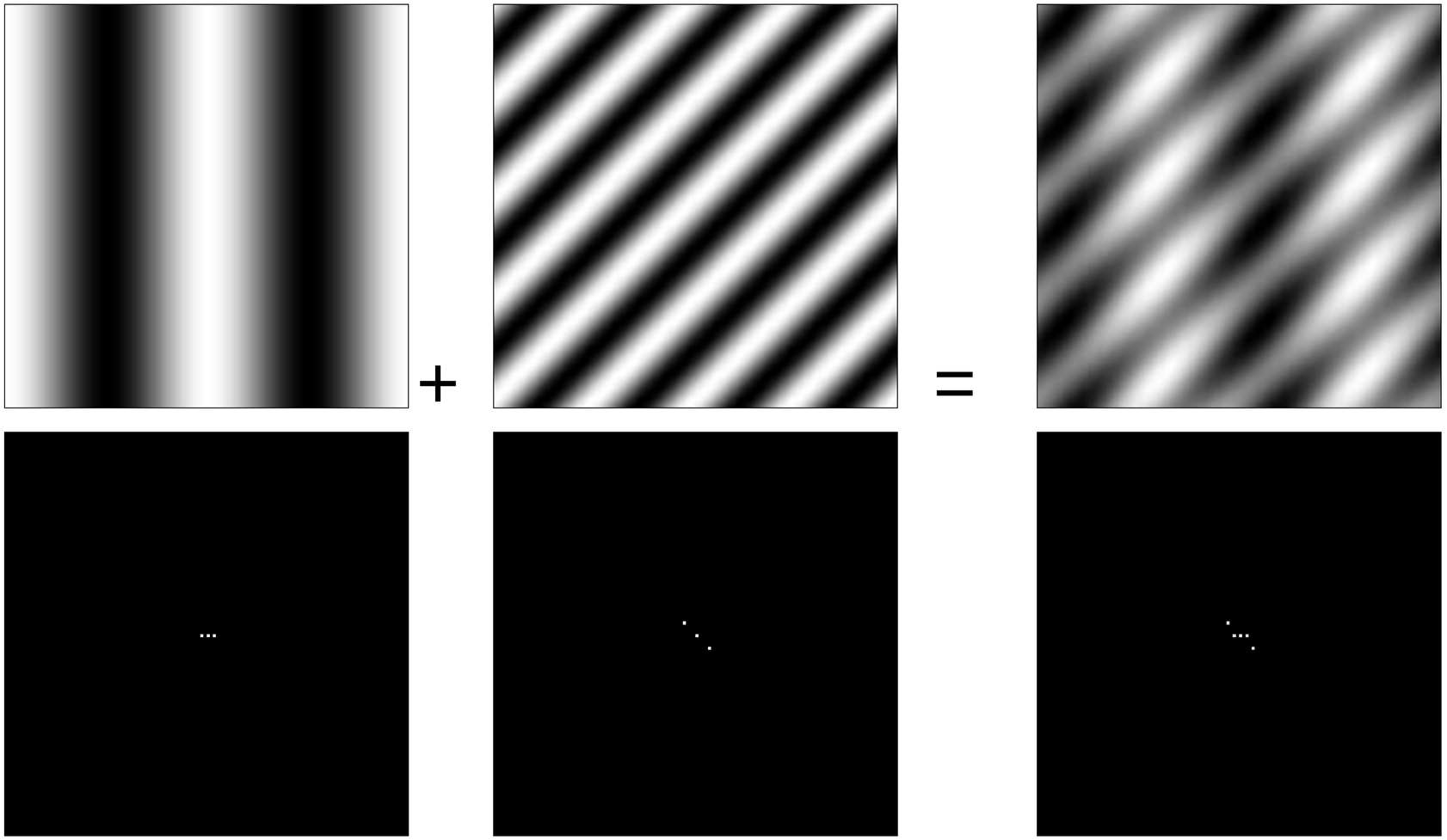


Fourier Image



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

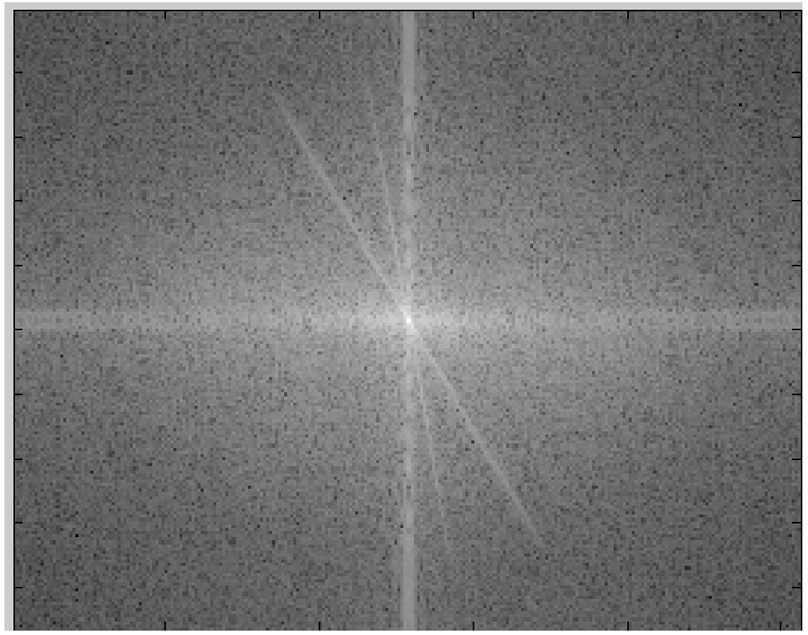
Signals can be composed



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

Man-made Scene

Amplitude Spectrum

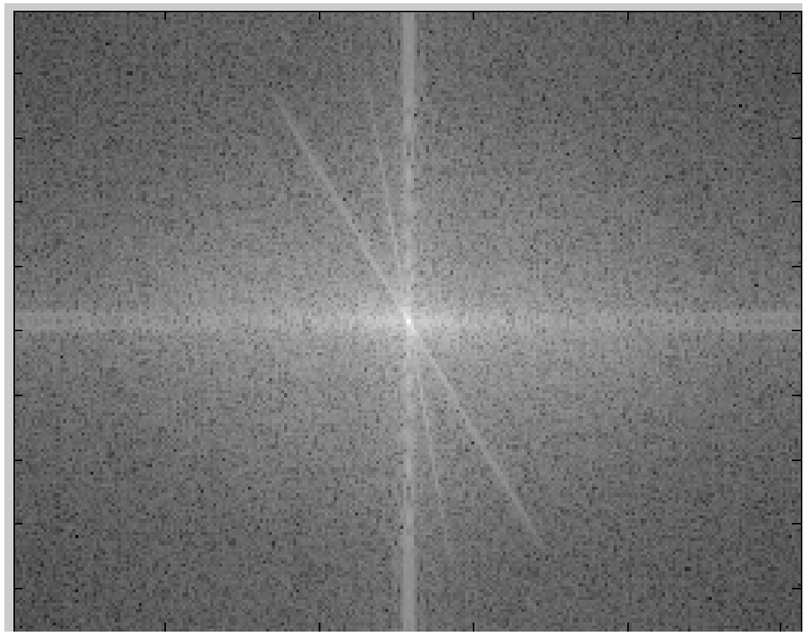


Can change spectrum, then reconstruct

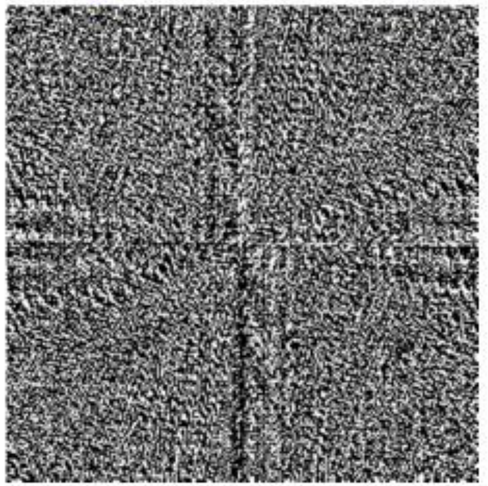
Local change in one domain, courses global change in the other

What about phase?

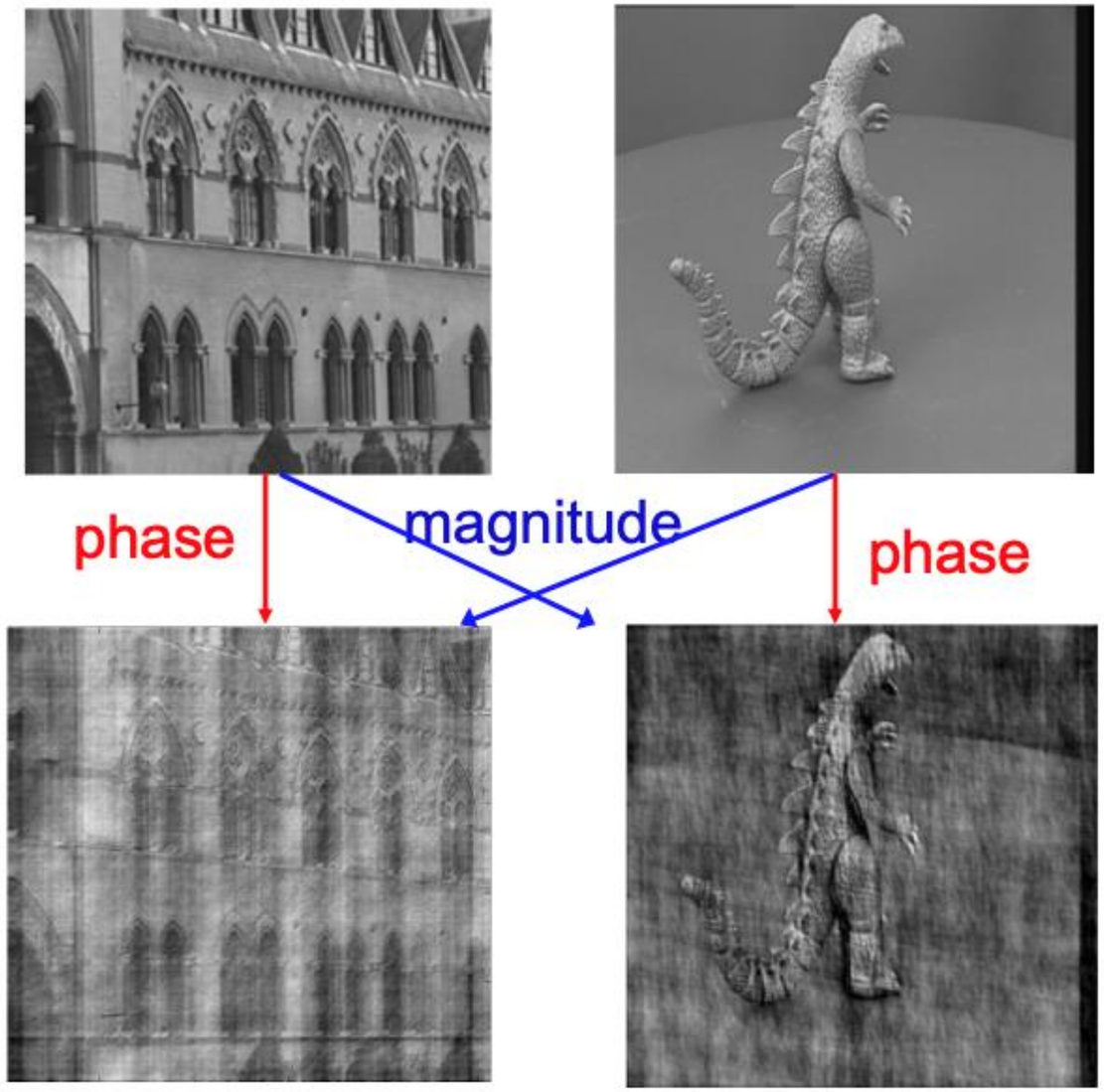
Amplitude Spectrum



what does phase look like, you ask?
(less visually informative)

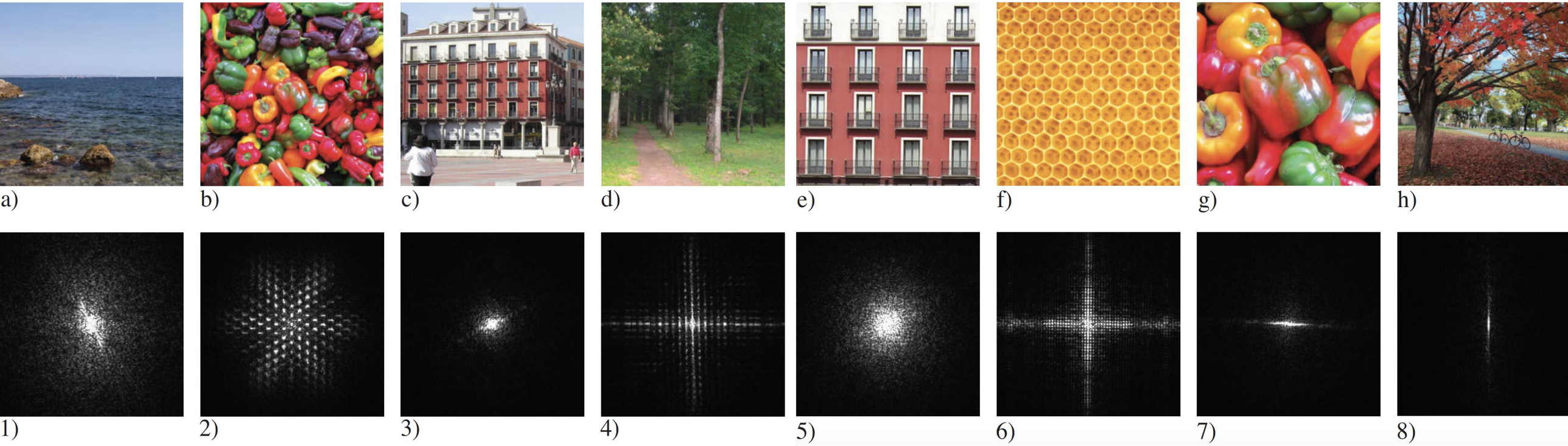


The importance of Phase

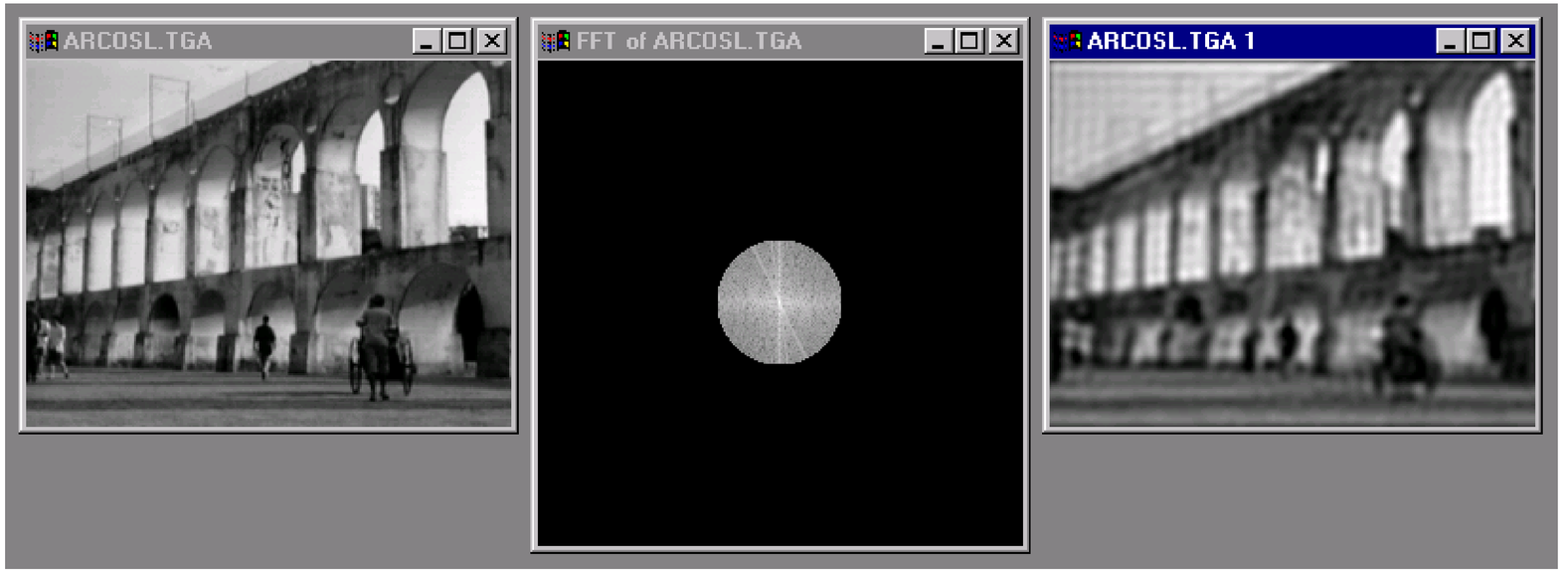


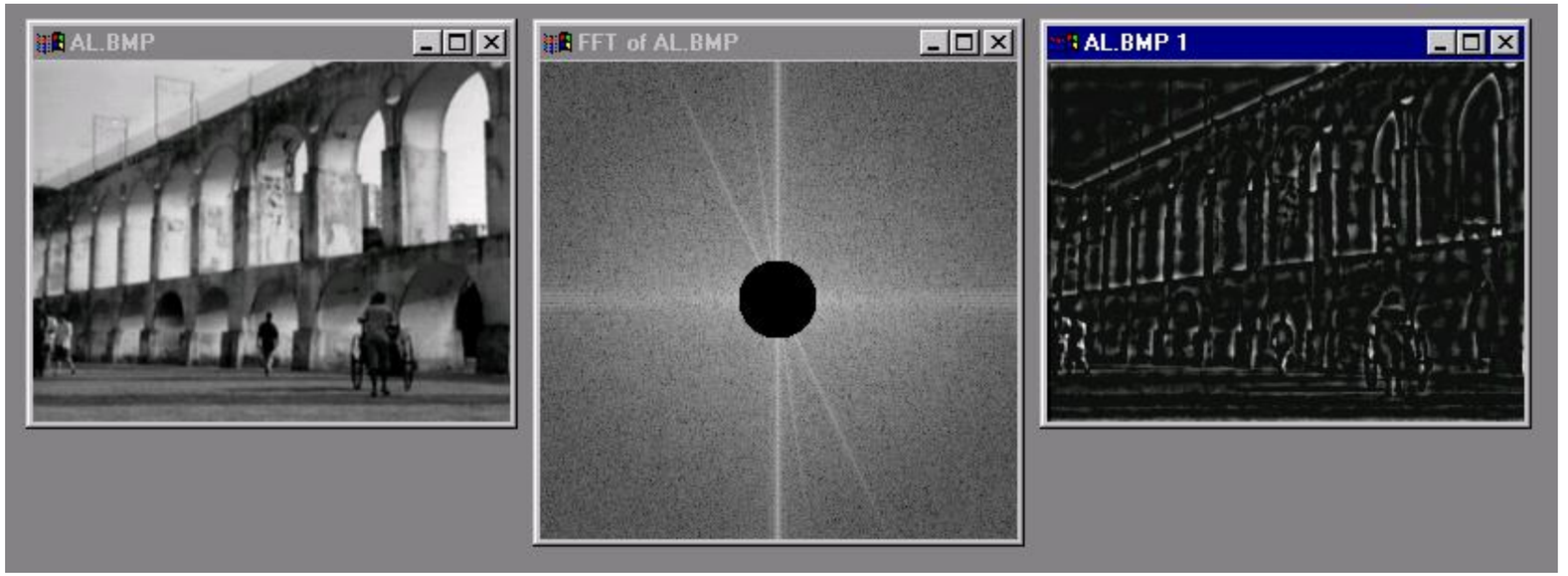
Slide: Andrew Zisserman

The DFT Game: find the right pairs



Low and High Pass filtering





The Convolution Theorem

The greatest thing since sliced (banana) bread!

- The Fourier transform of the convolution of two functions is the product of their Fourier transforms

$$\mathrm{F}[g * h] = \mathrm{F}[g]\mathrm{F}[h]$$

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

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

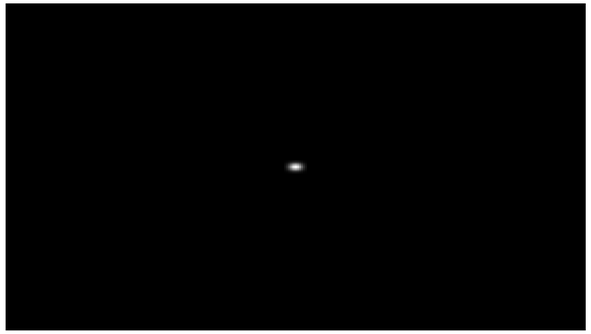
- **Convolution** in spatial domain is equivalent to **multiplication** in frequency domain!

2D convolution theorem example

$f(x,y)$

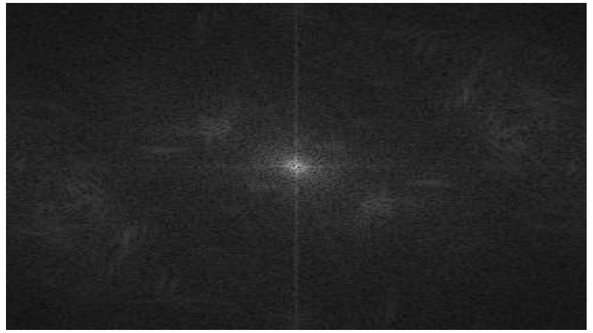
$*$

$h(x,y)$

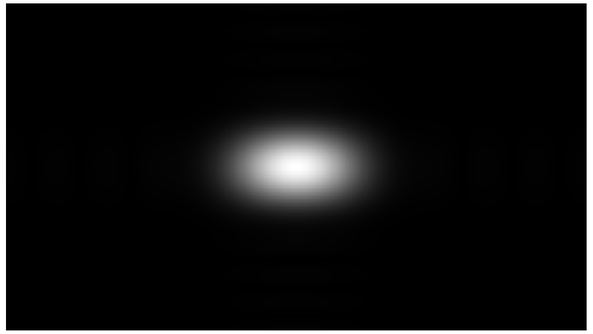


$\Downarrow$

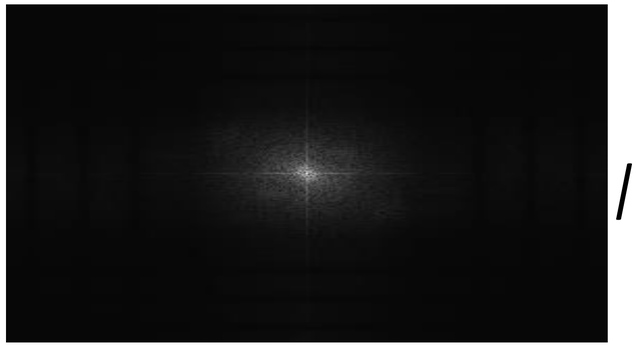
$g(x,y)$



$\times$



$\Downarrow$



$|F(s_x,s_y)|$

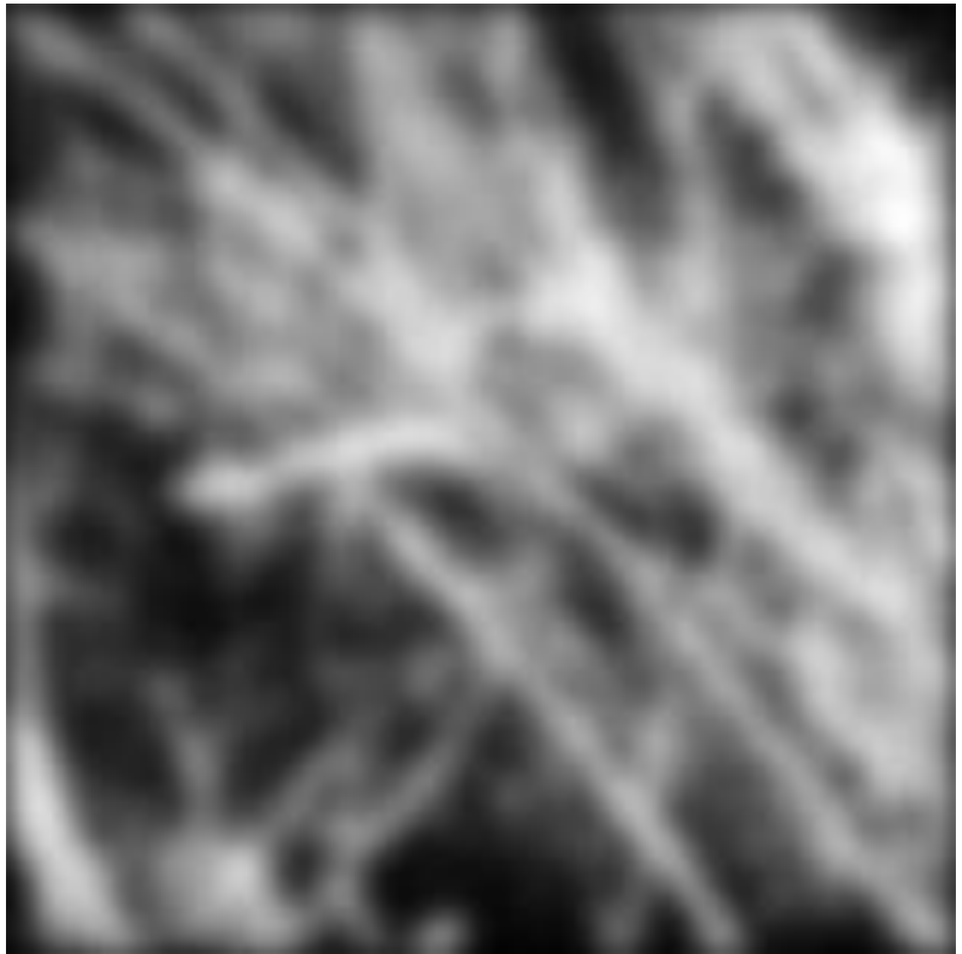
$|H(s_x,s_y)|$

$|G(s_x,s_y)|$

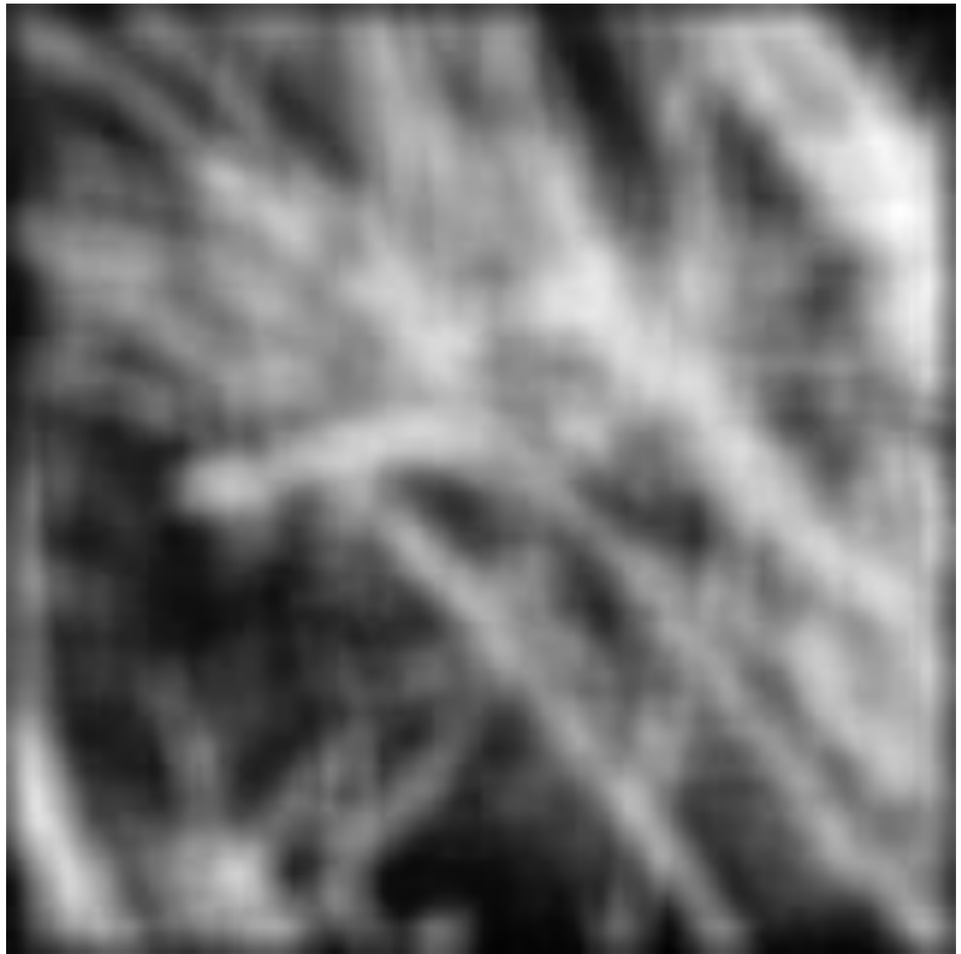
Back to Filtering

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

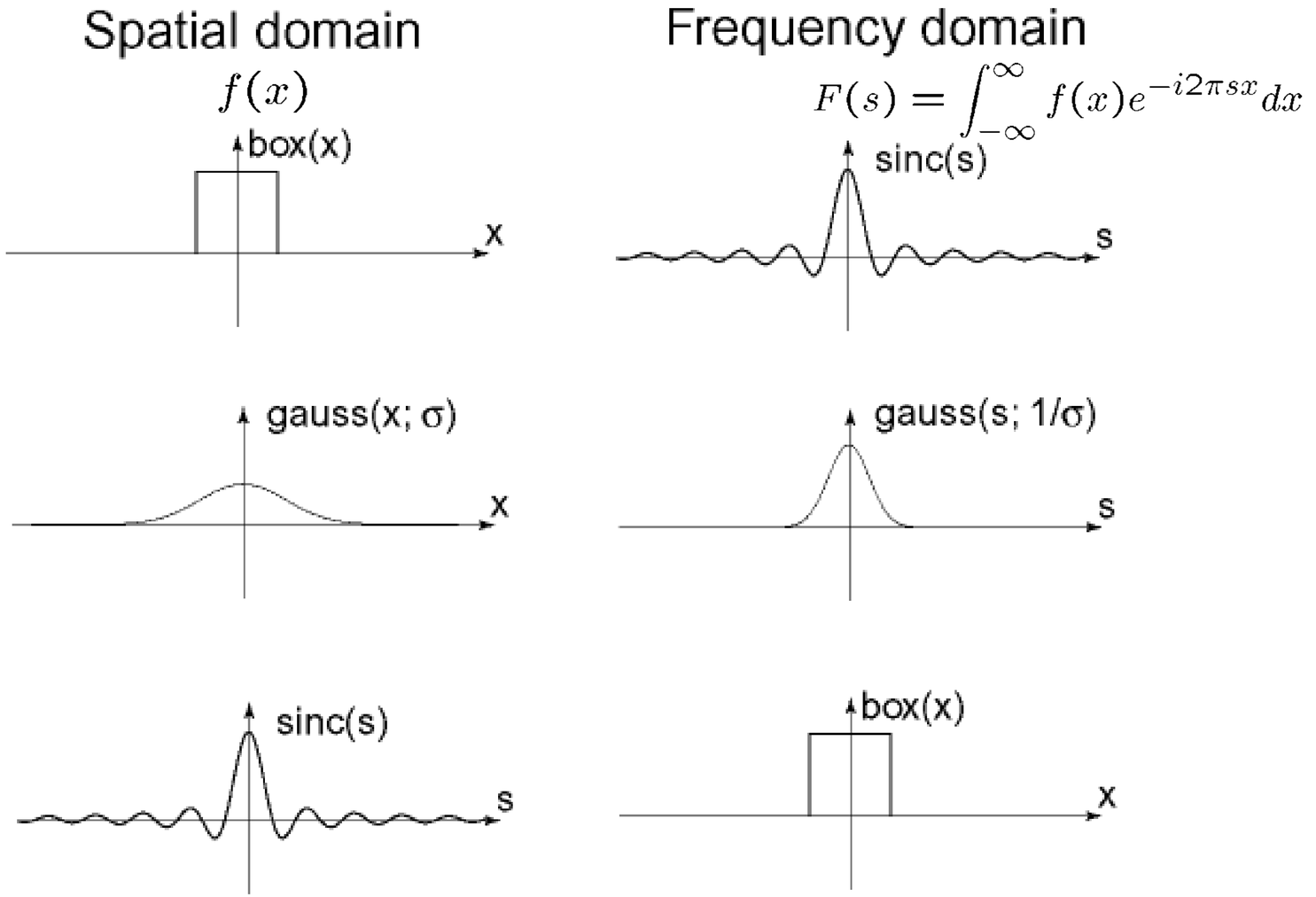
Gaussian 



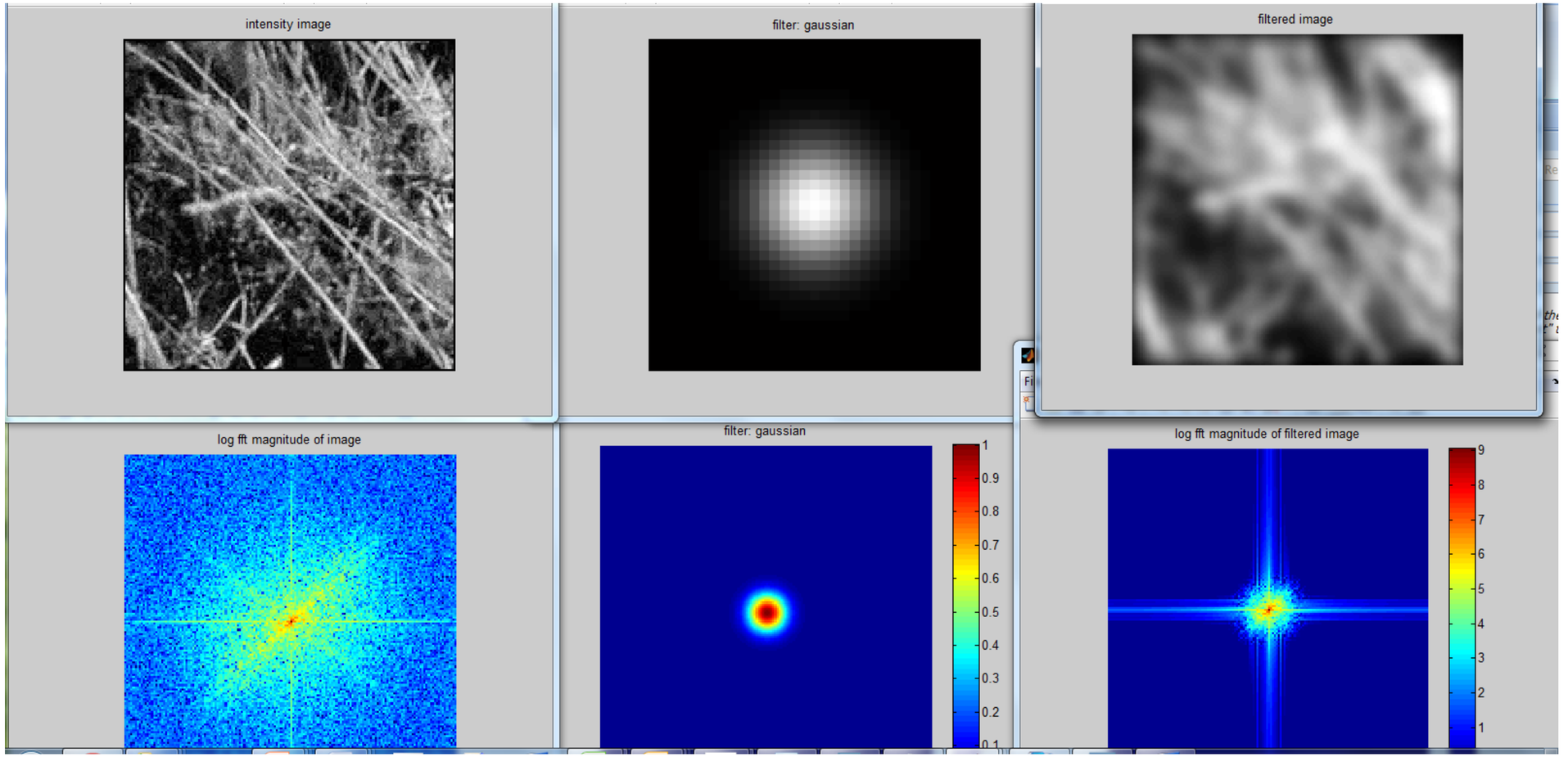
Box filter 



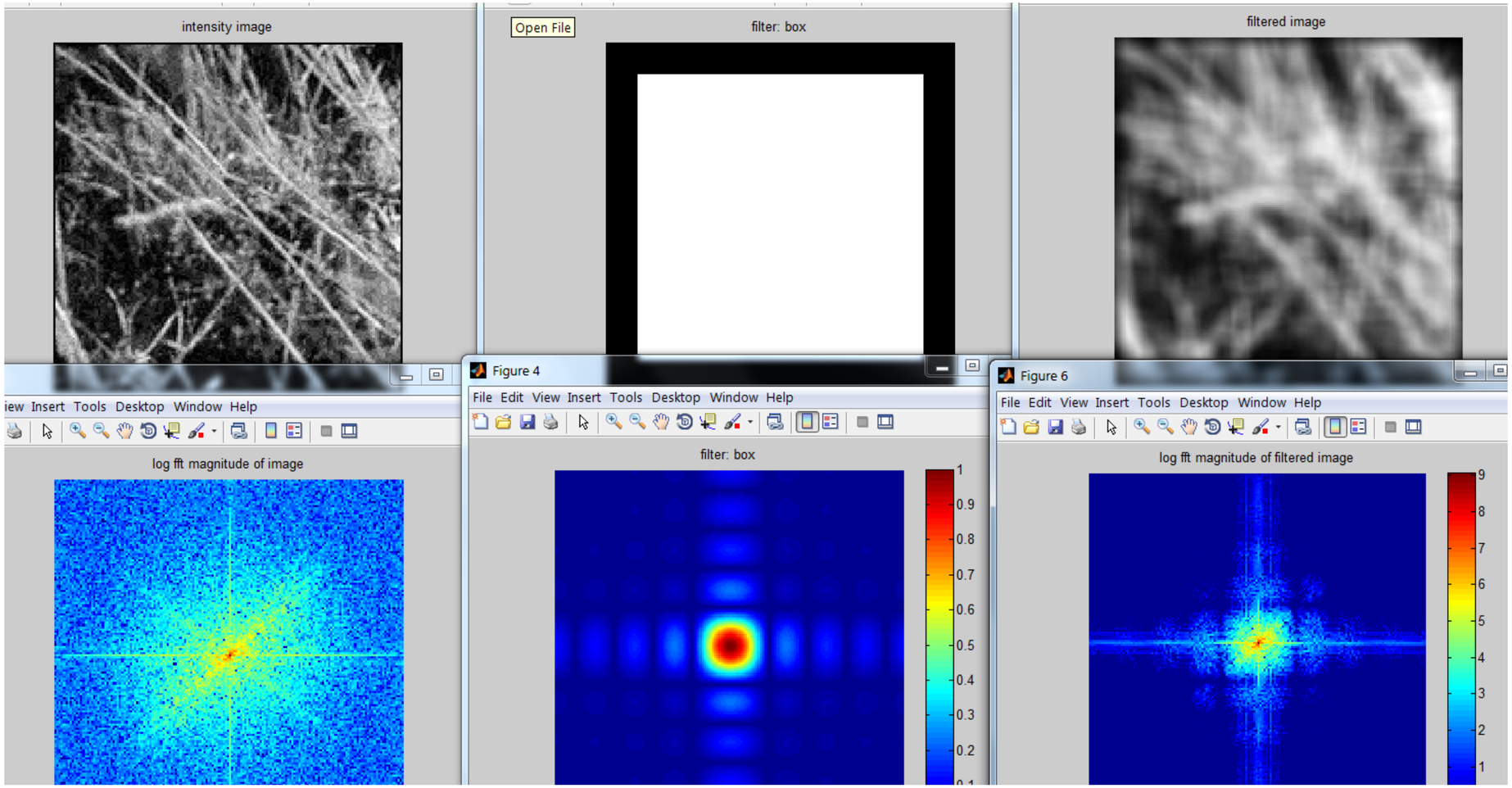
Fourier Transform pairs



Gaussian

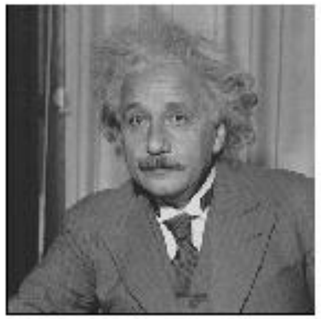
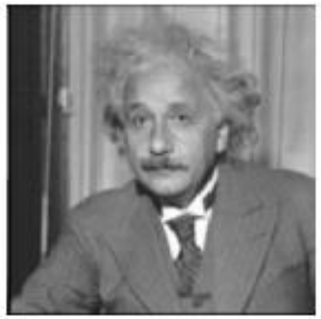
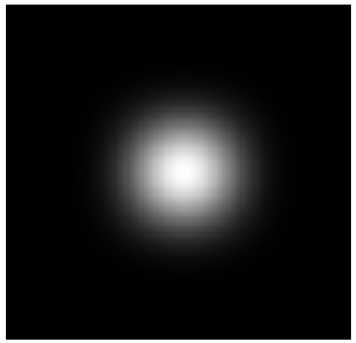

intensity image
filter: gaussian
filtered image
log fft magnitude of image
filter: gaussian
log fft magnitude of filtered image
1
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
9
8
7
6
5
4
3
2
1

Box Filter

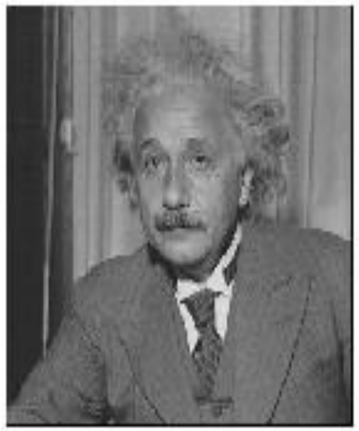

intensity image
Open File
filter: box
filtered image
iew Insert Tools Desktop Window Help
log fft magnitude of image
Figure 4
File Edit View Insert Tools Desktop Window Help
filter: box
1
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
Figure 6
File Edit View Insert Tools Desktop Window Help
log fft magnitude of filtered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

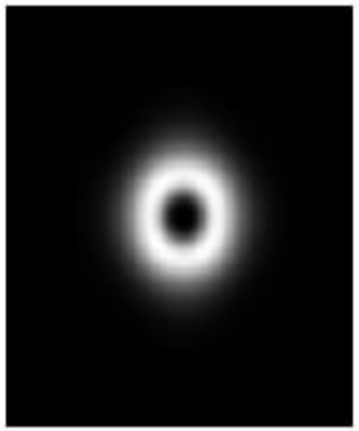
Low-pass, Band-pass, High-pass filters

low-pass:

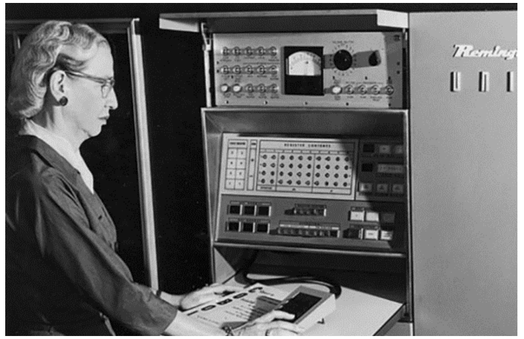
High-pass / band-pass:





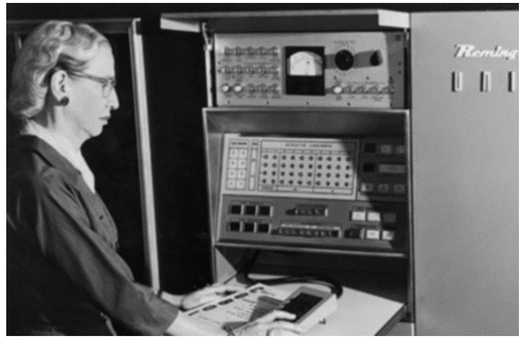
Low Pass vs. High Pass filtering

Image



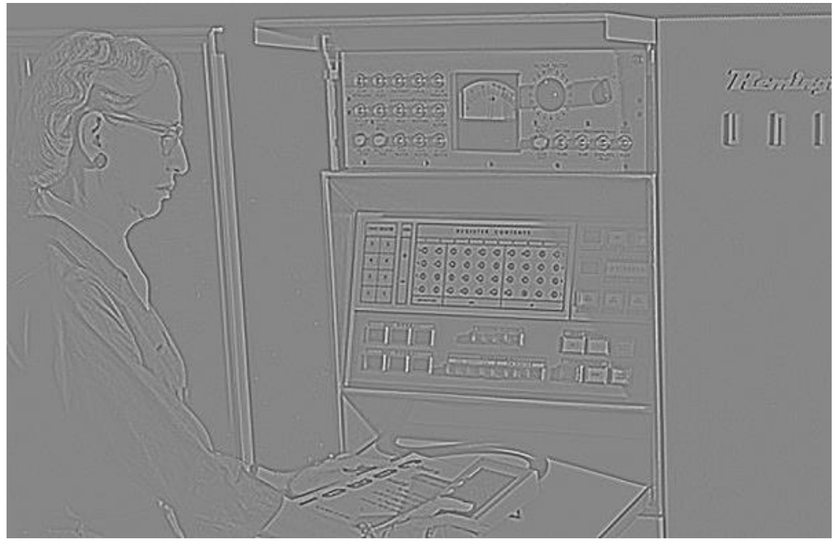
-

Smoothed



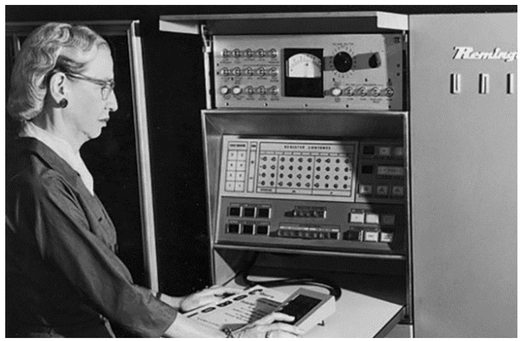
Details

=



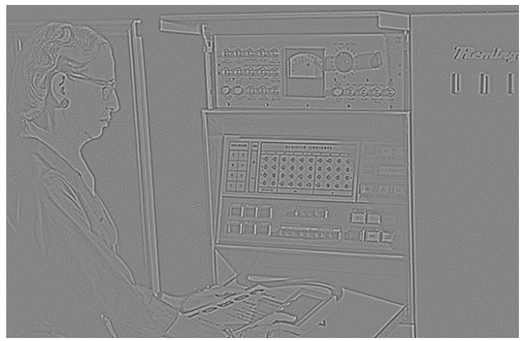
Filtering – Sharpening

Image



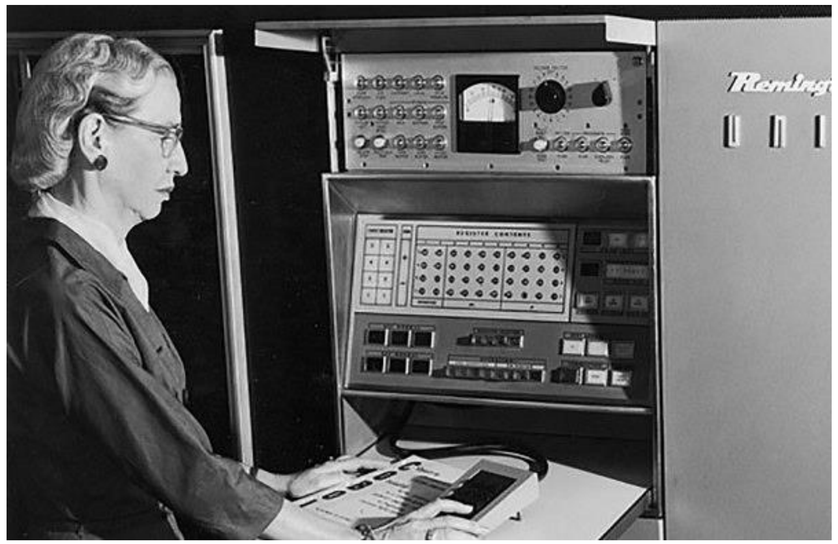
+α

Details



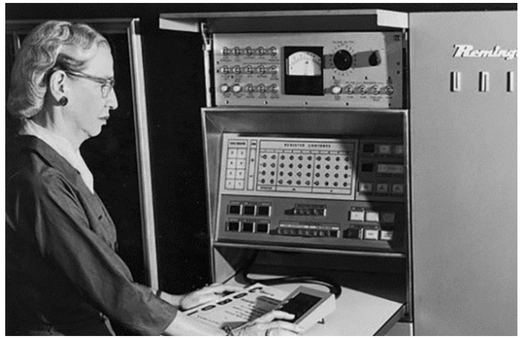
“Sharpened” α=1

=



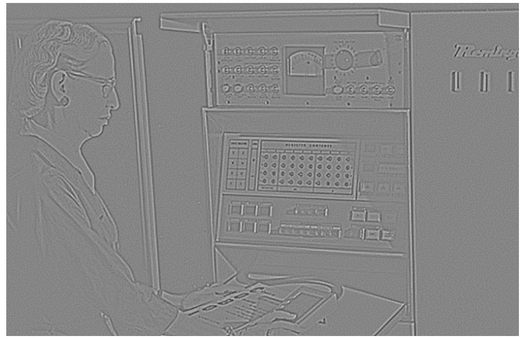
Filtering – Sharpening

Image



+α

Details

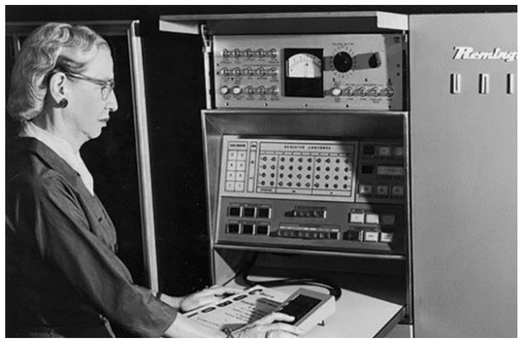


“Sharpened” α=0

=

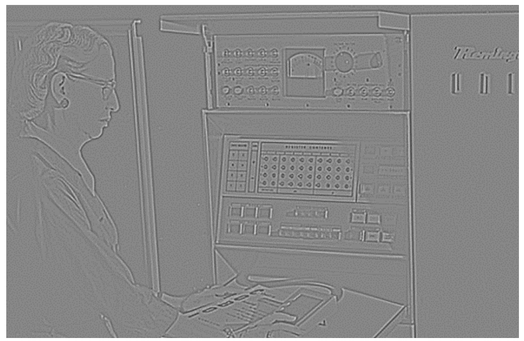
Filtering – Sharpening

Image



+α

Details

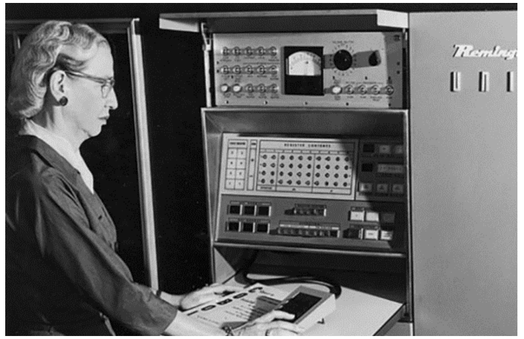


“Sharpened” α=2

=

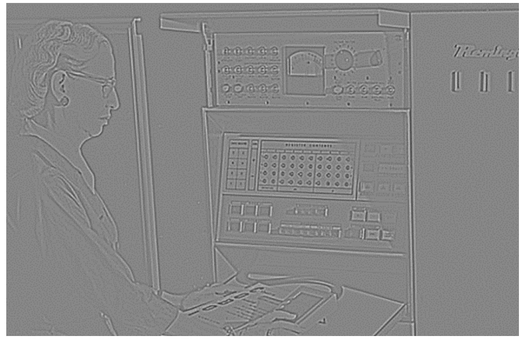
Filtering – Sharpening

Image



+α

Details

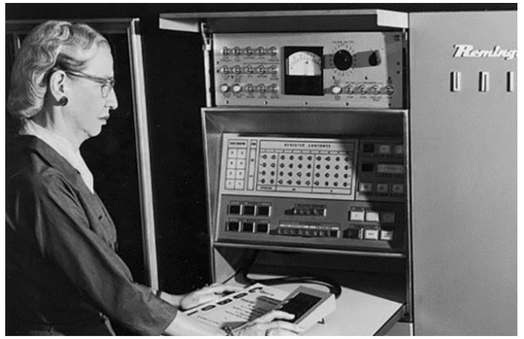


“Sharpened” α=0

=

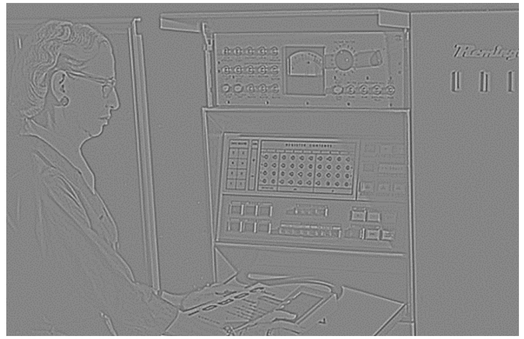
Filtering – Extreme Sharpening

Image



+α

Details



“Sharpened” α=10

=

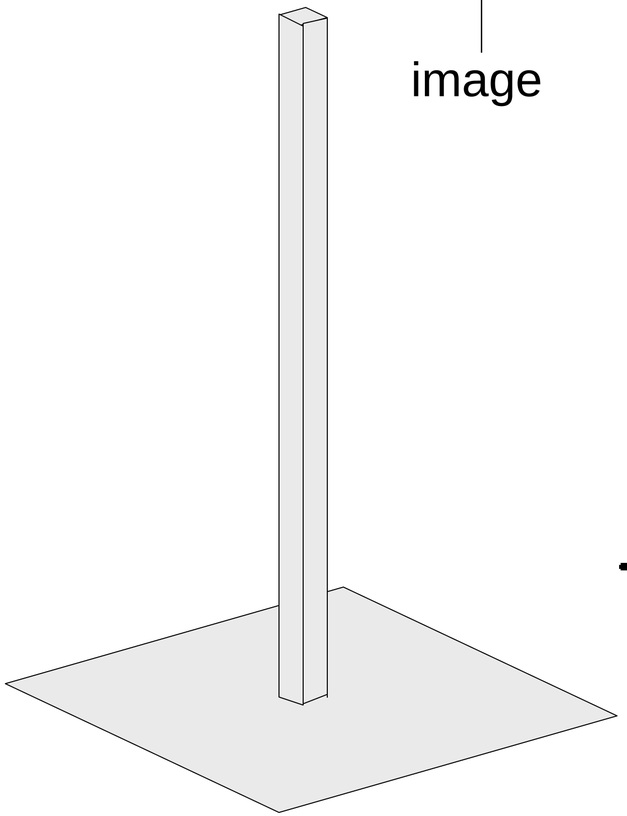
Unsharp mask filter

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

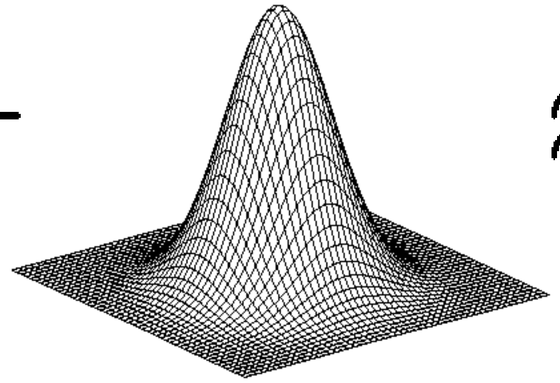
image

blurred image

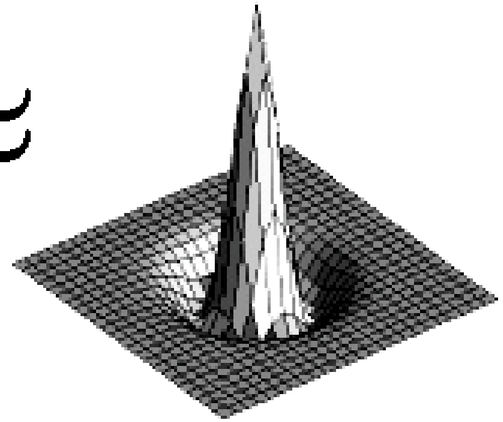
unit impulse (identity)



unit impulse

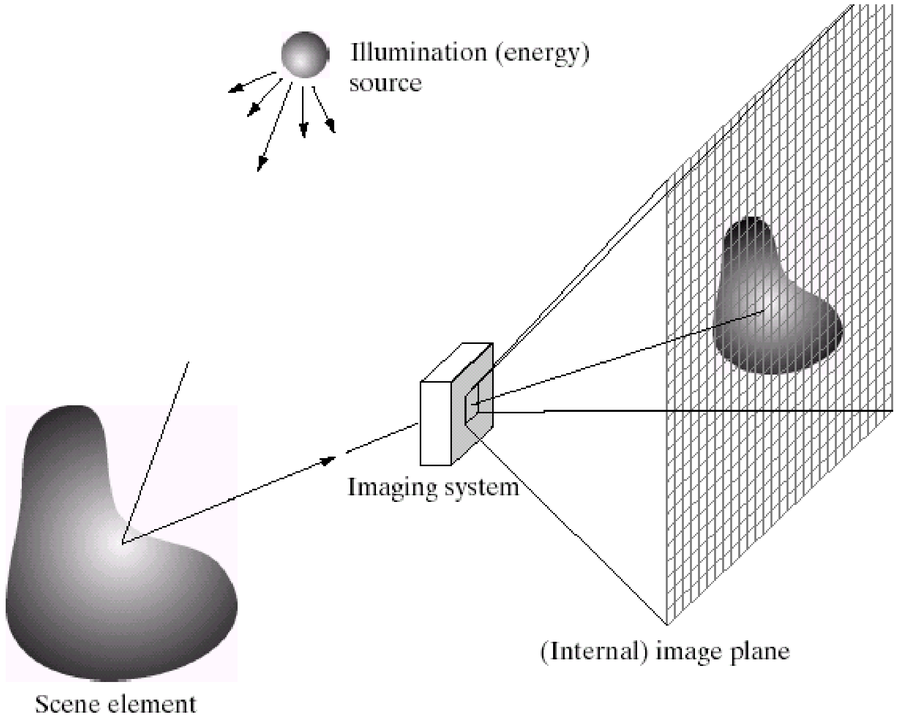


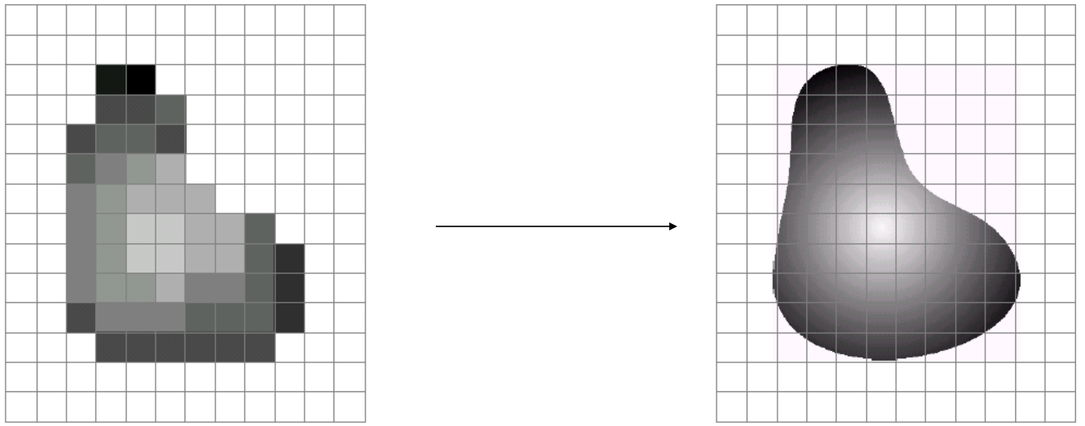
Gaussian



Laplacian of Gaussian

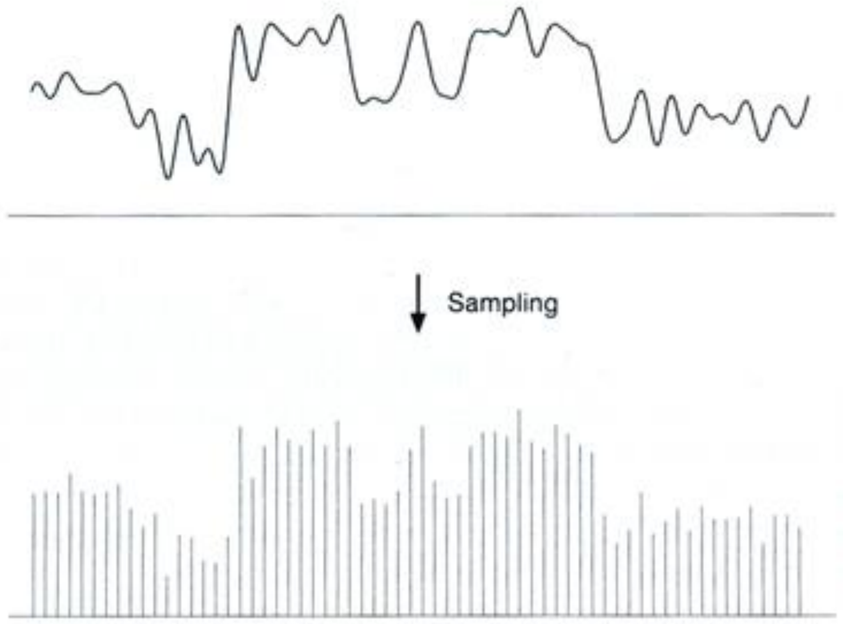
Sampling and Reconstruction





Sampled representations

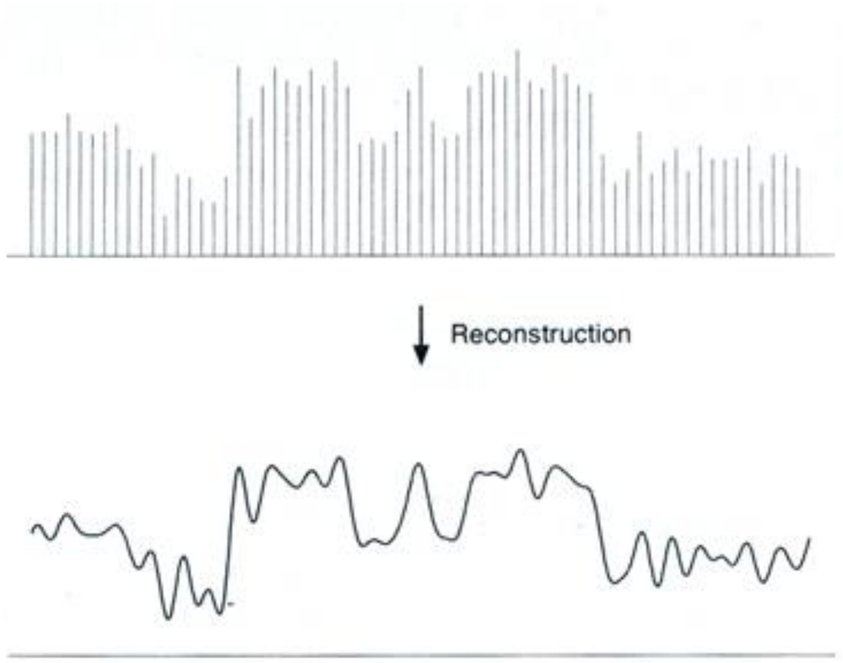
- How to store and compute with continuous functions?
- Common scheme for representation: samples
  - write down the function's values at many points



[FvDFH fig.14.14b / Wolberg]

Reconstruction

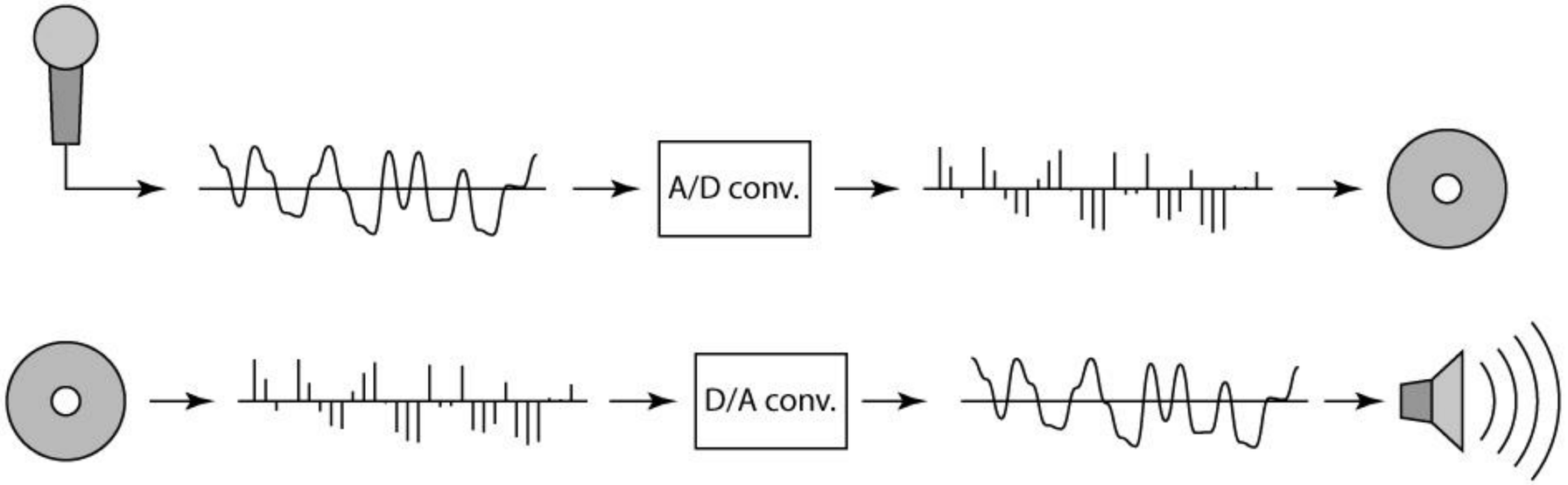
- Making samples back into a continuous function
  - for output (need realizable method)
  - for analysis or processing (need mathematical method)
  - amounts to “guessing” what the function did in between

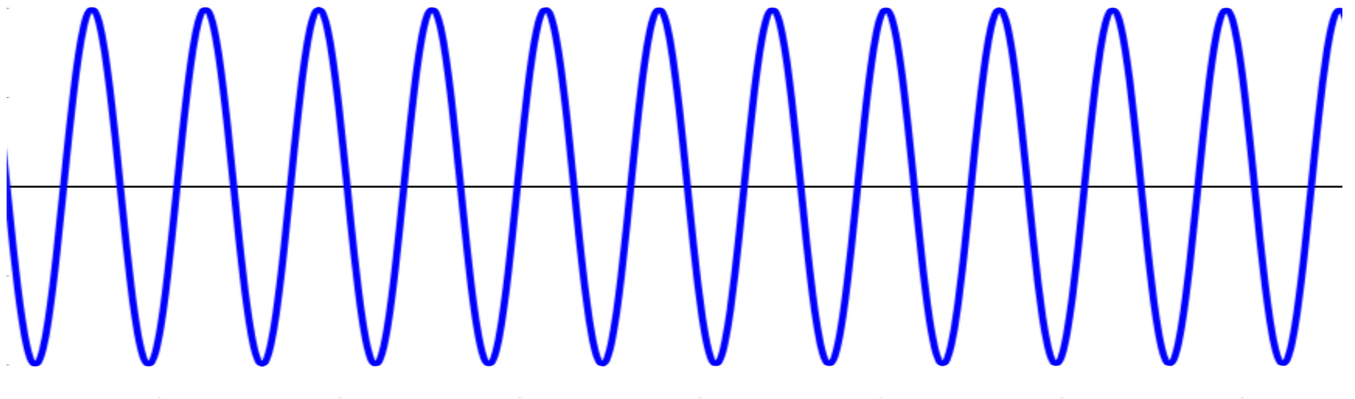


[FvDFH fig.14.14b / Wolberg]

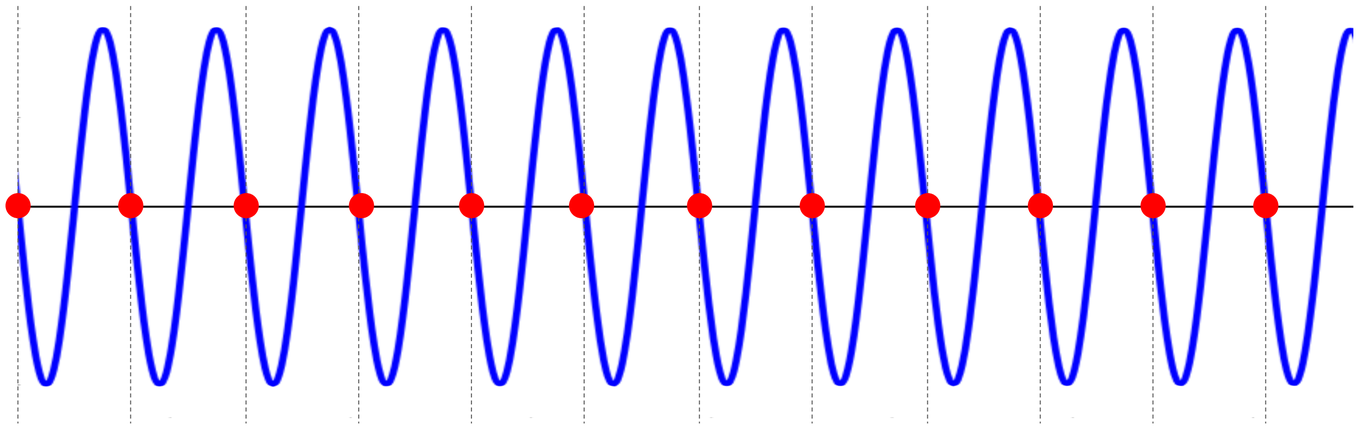
Sampling in digital audio

- 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?



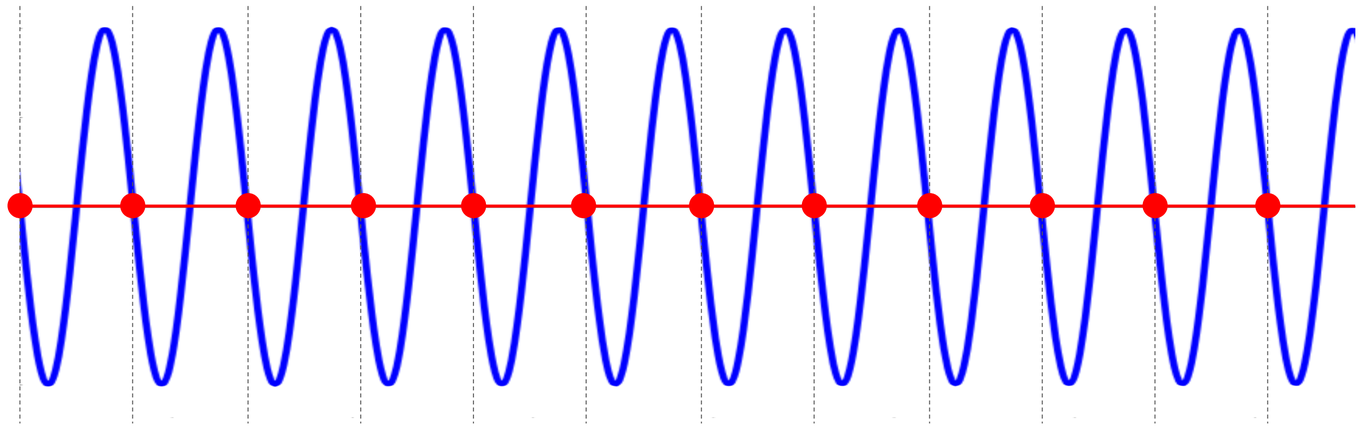


f(x)

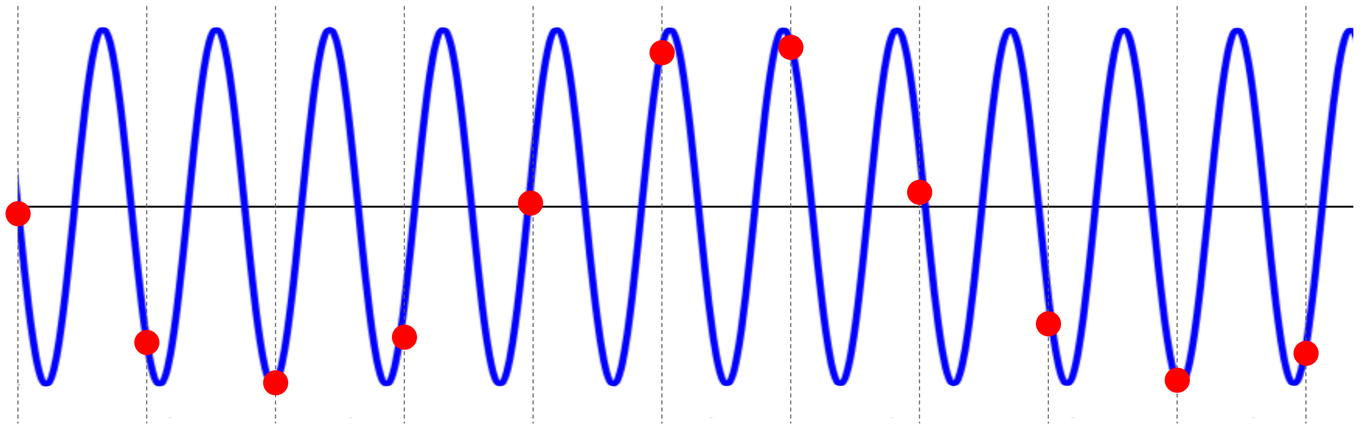


f(x)

g(x)



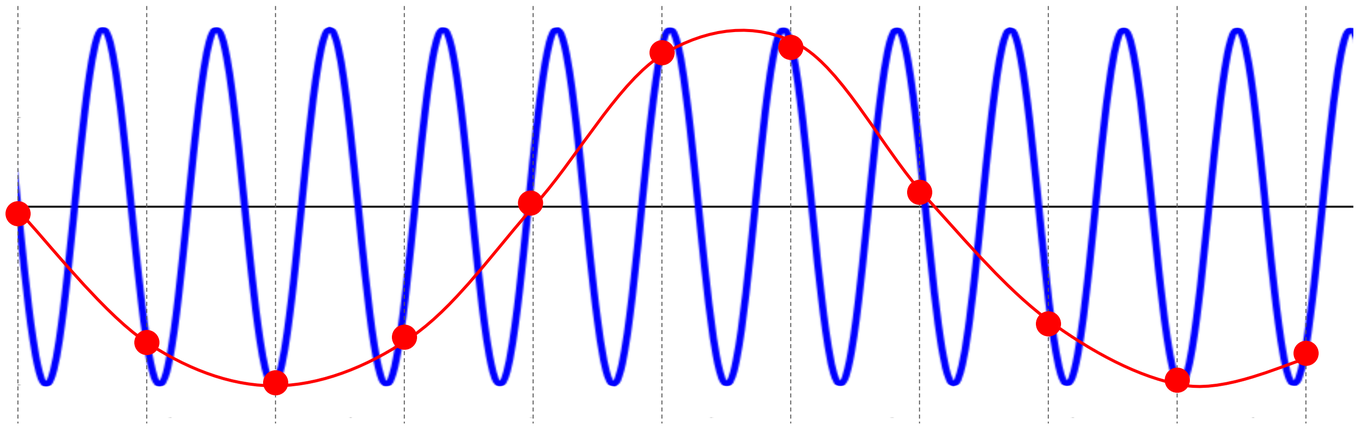
f(x)



f(x)

Aliasing

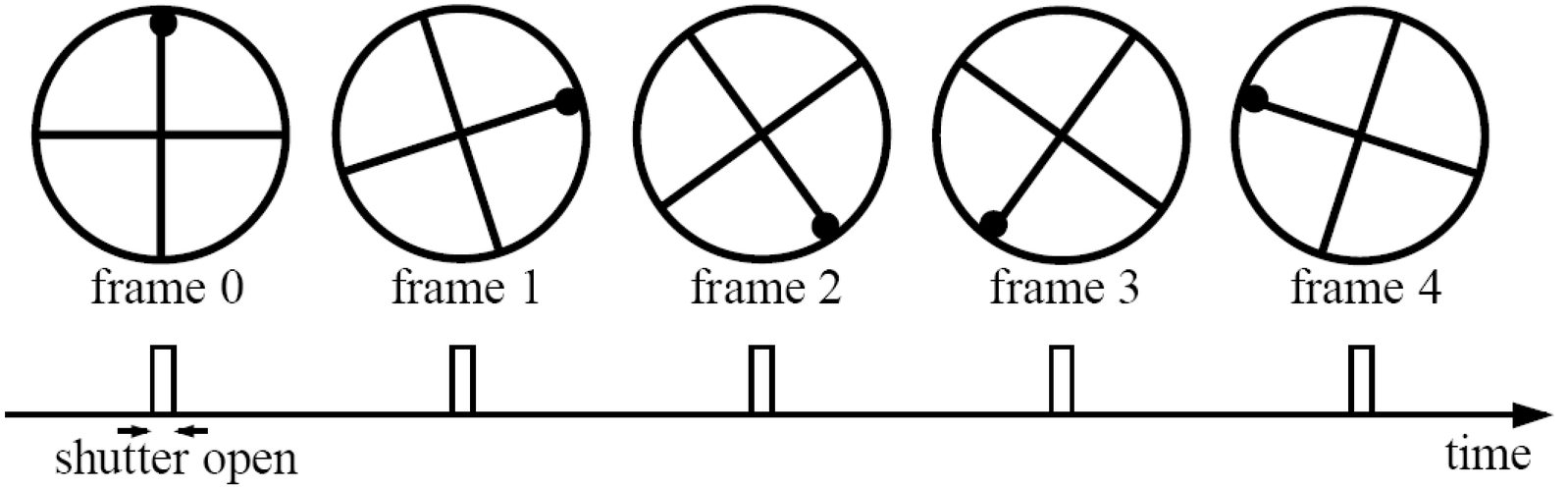
g(x) is an “alias” of f(x)



f(x)

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

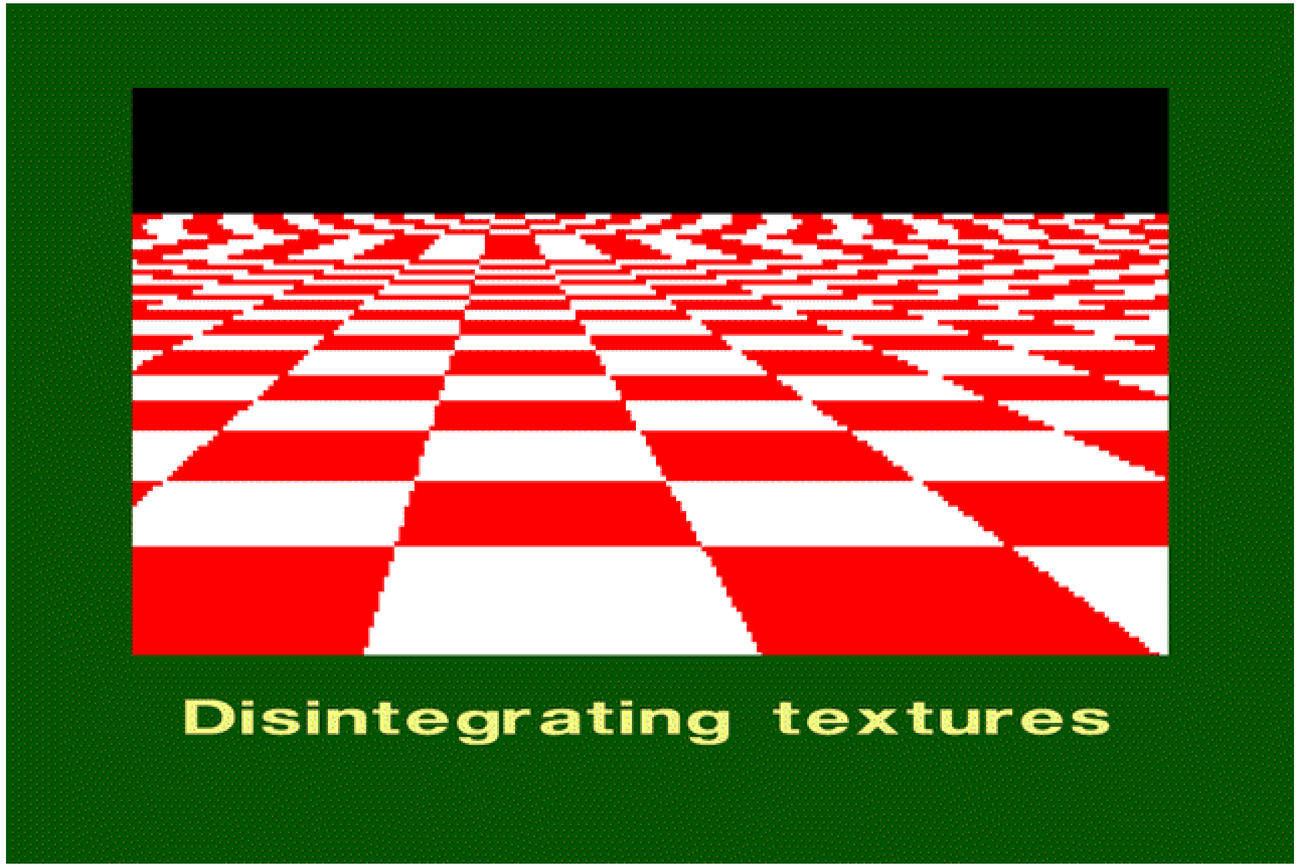
If camera shutter is only open for a fraction of a frame time (frame time = 1/30 sec. for video, 1/24 sec. for film):



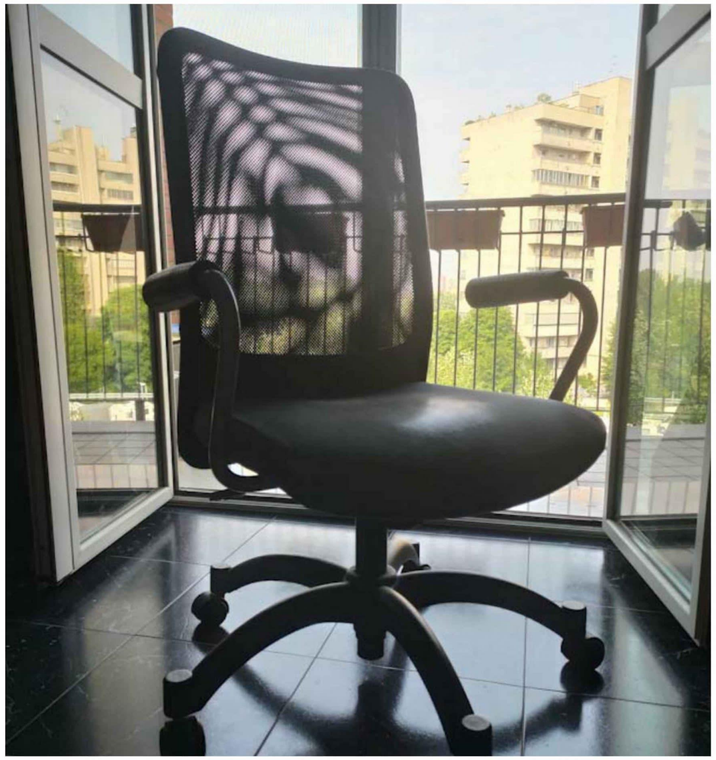
Without dot, wheel appears to be rotating slowly backwards! (counterclockwise)

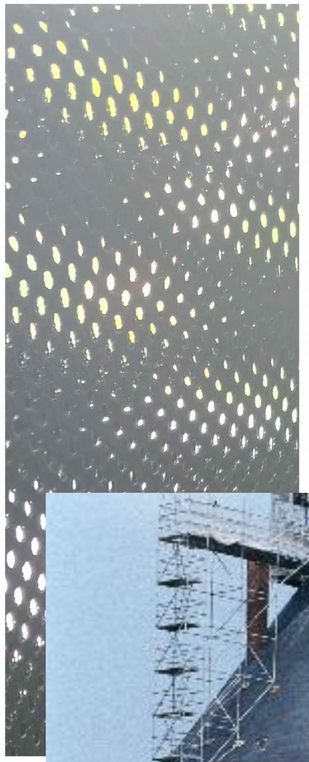

TEACH ME

Aliasing in images



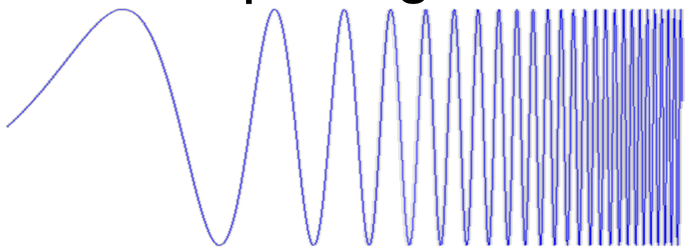
Aliasing in real images



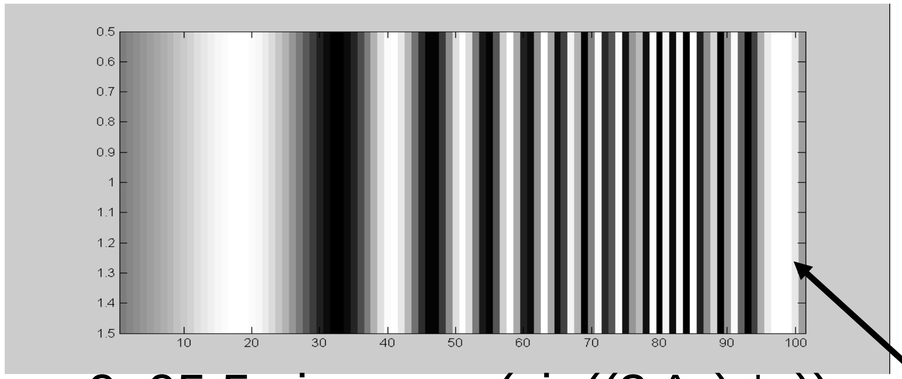


What’s happening?

Input signal:



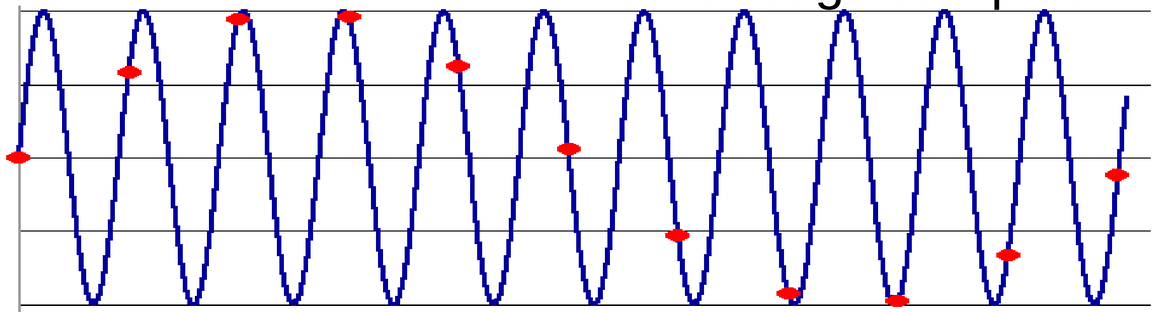
Plot as image:



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

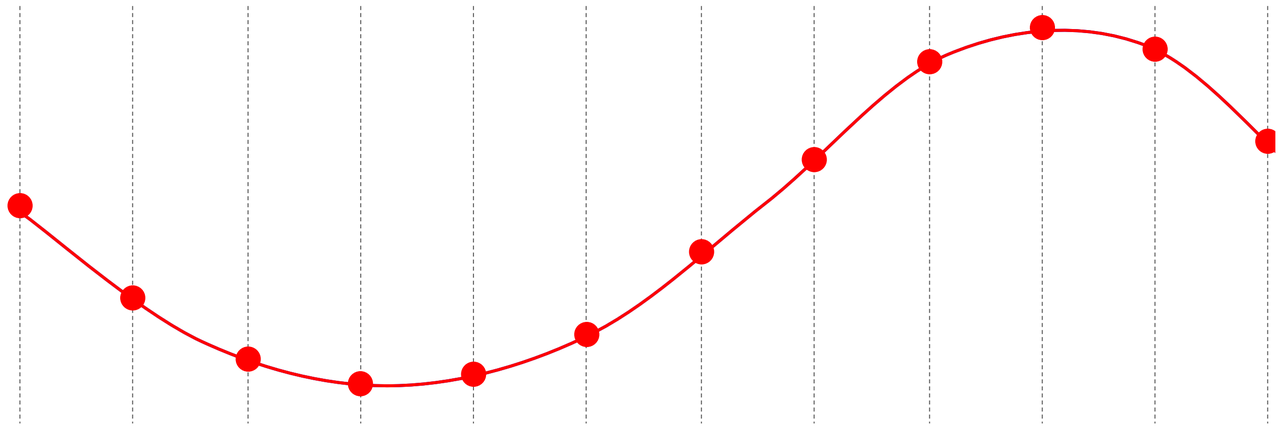
Alias!

Not enough samples



Aliasing

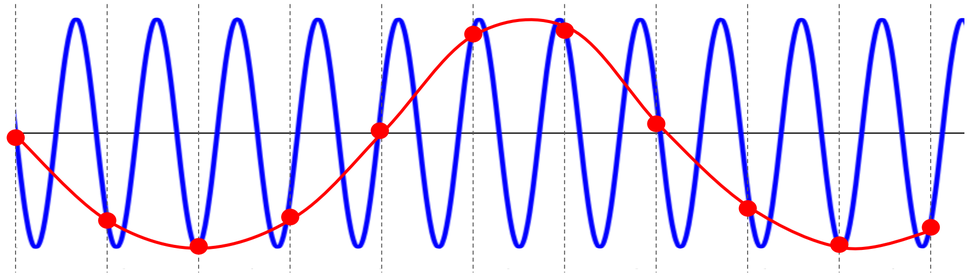
$g(x) \sim= f(x)$



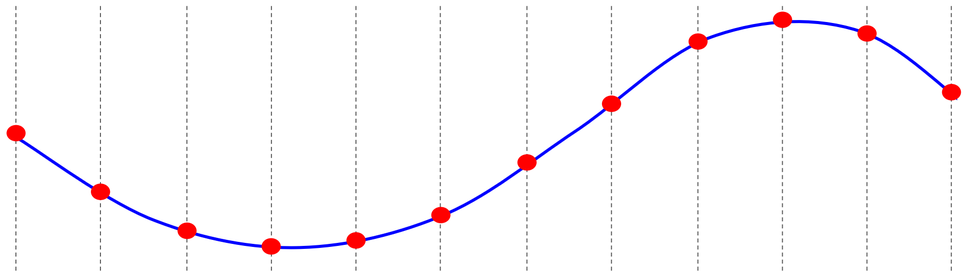
$f(x)$

No aliasing in this case

Sampling and the Nyquist rate



< 1 sample per cycle



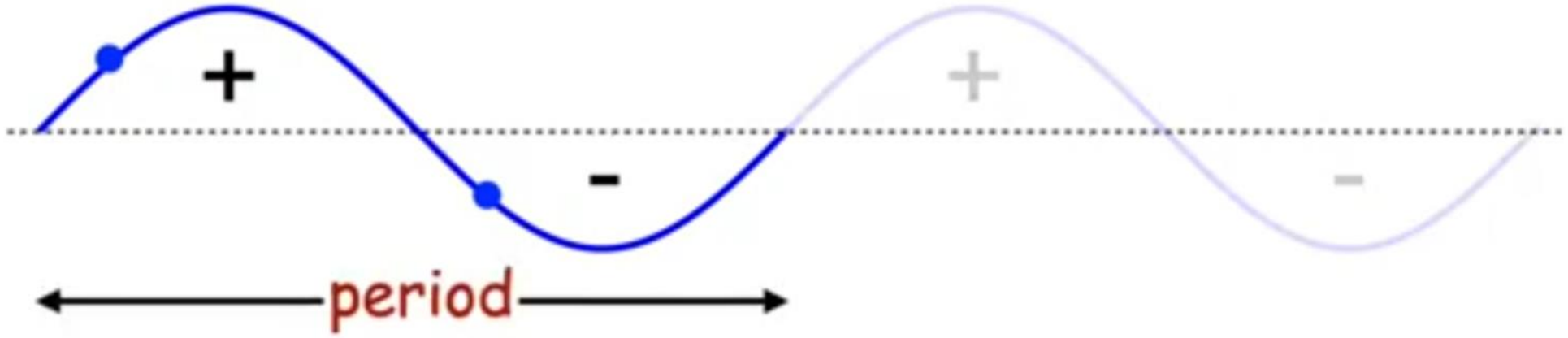
> 10 samples per cycle

To avoid aliasing

- **≥ two samples per cycle**

This minimum sampling rate is called the **Nyquist rate**

To prevent aliasing…



>2 samples / period

Image Scaling

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

How to generate a half-sized version?

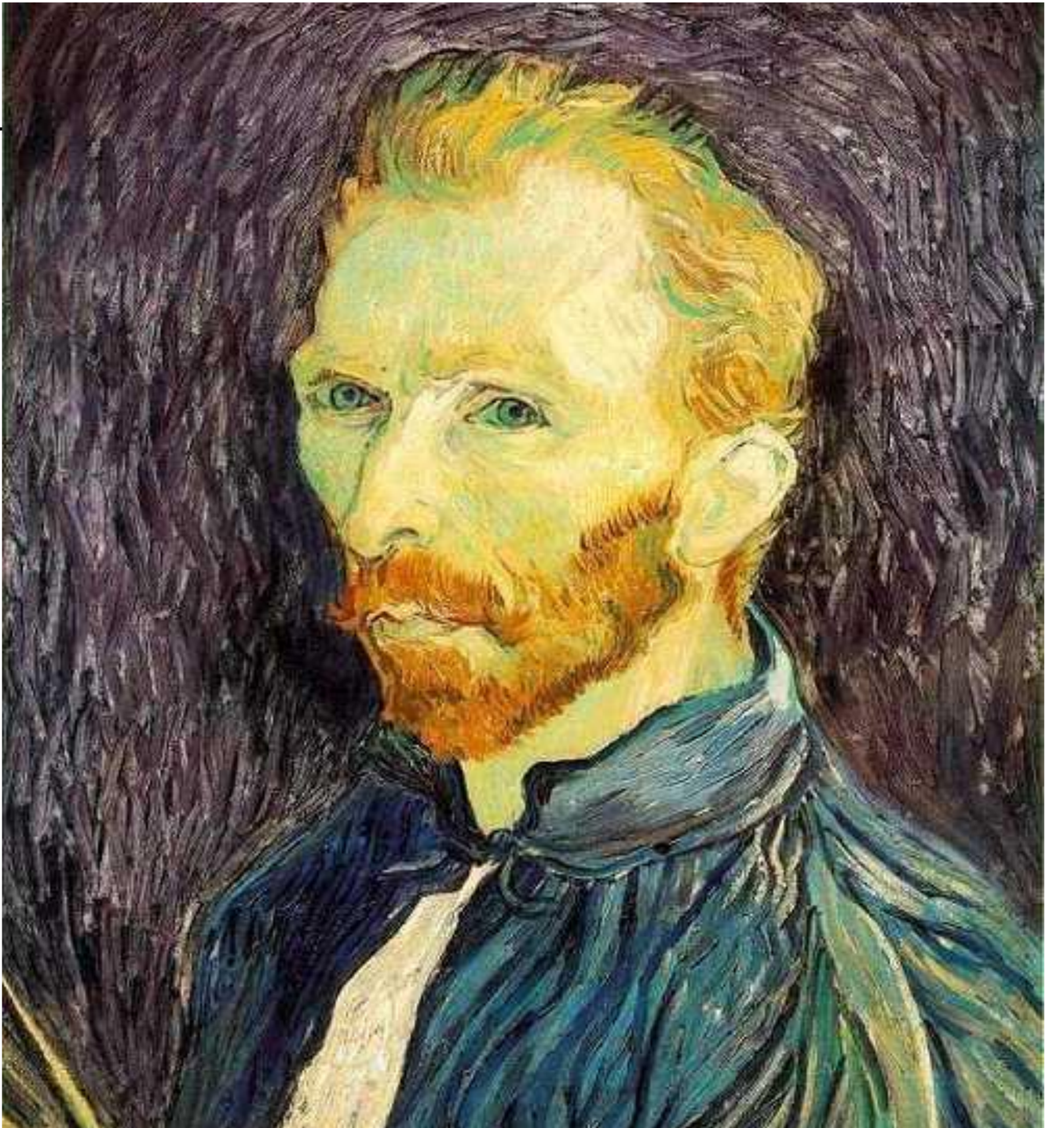
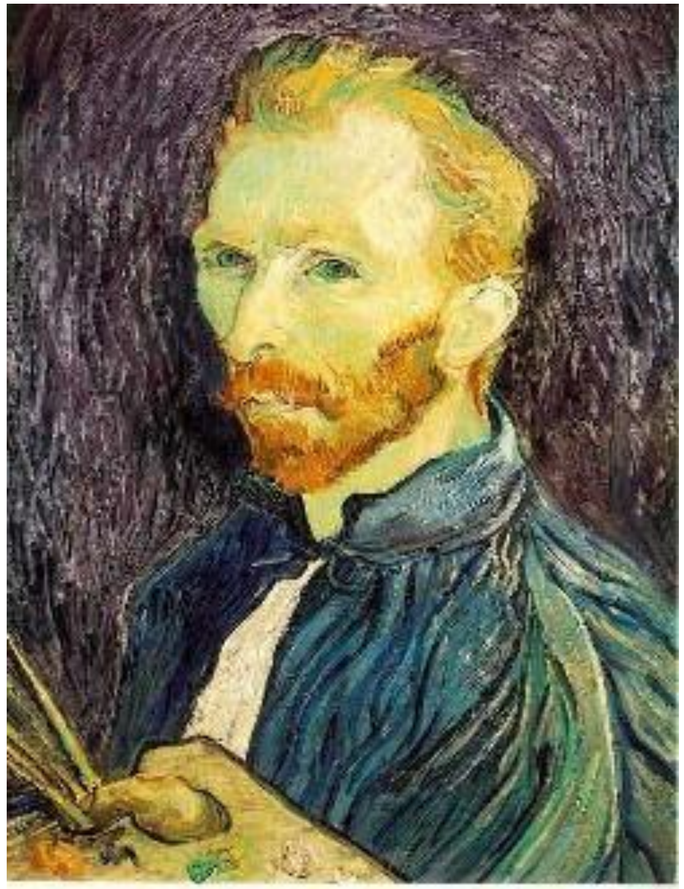


Image sub-sampling

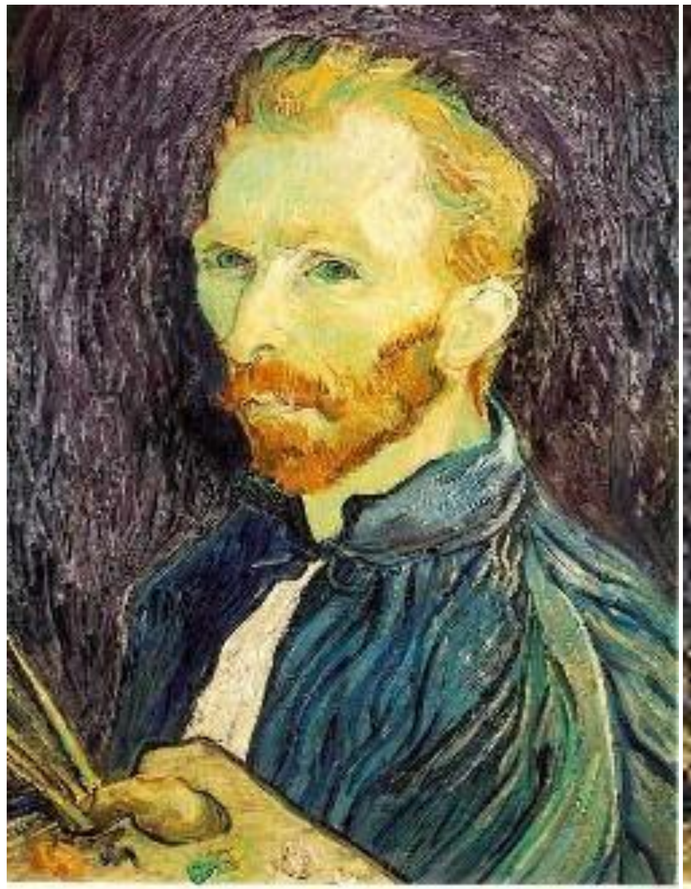


1/4

1/8

Throw away every other row and column to create a 1/2 size image
- called *image sub-sampling*

Image sub-sampling



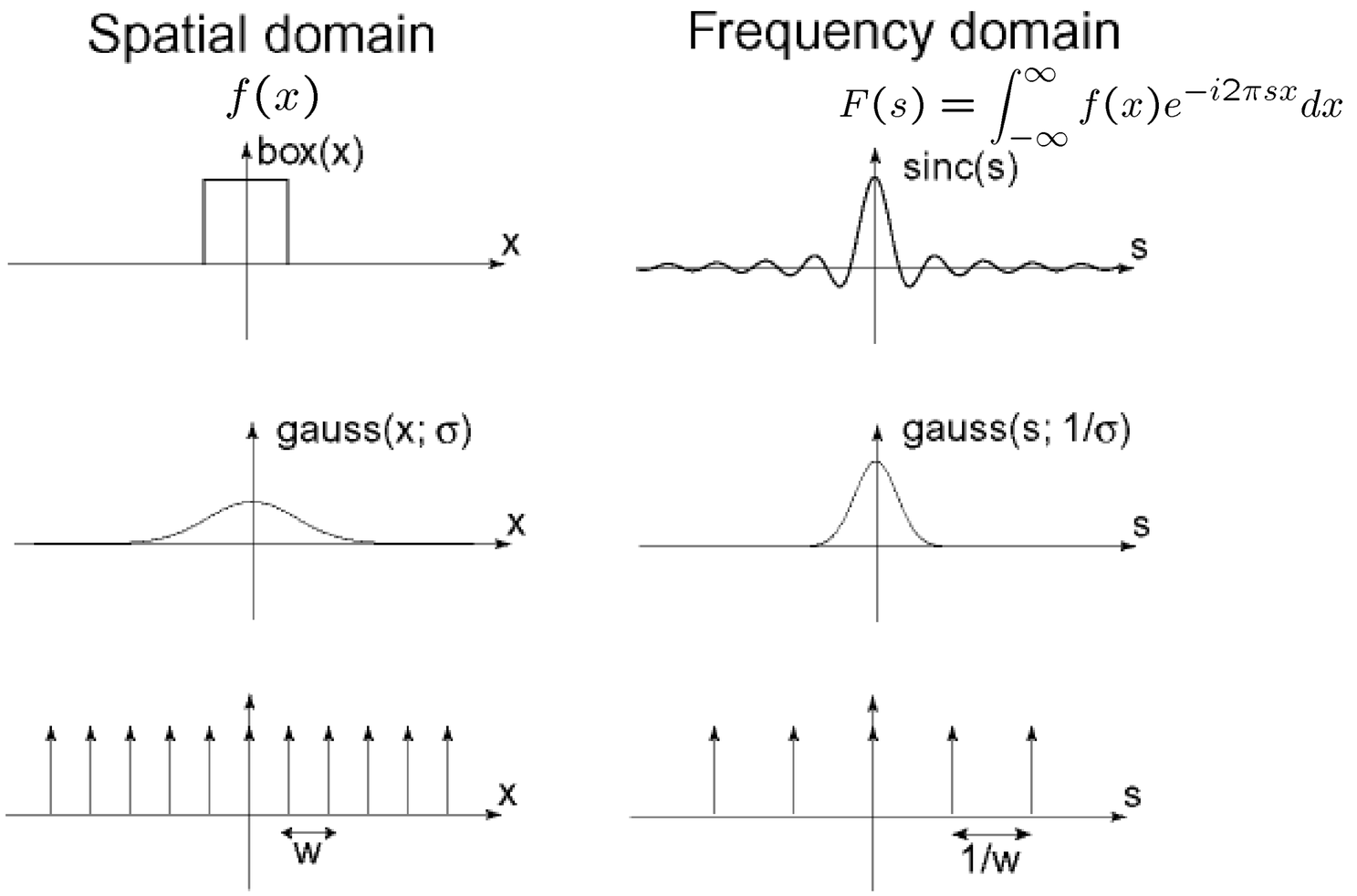
1/2

1/4 (2x zoom)

1/8 (4x zoom)

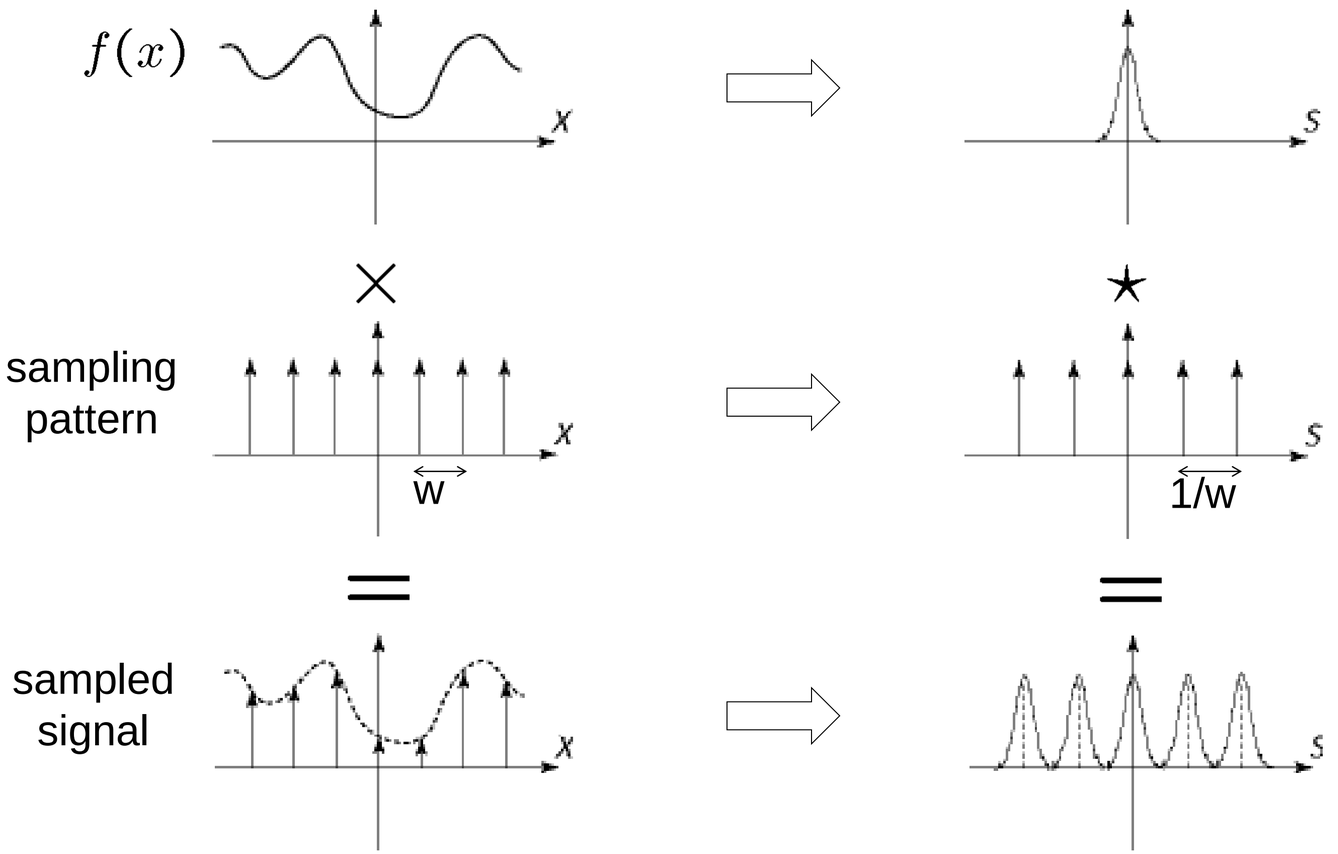
Why does this look so crufty?

Fourier transform



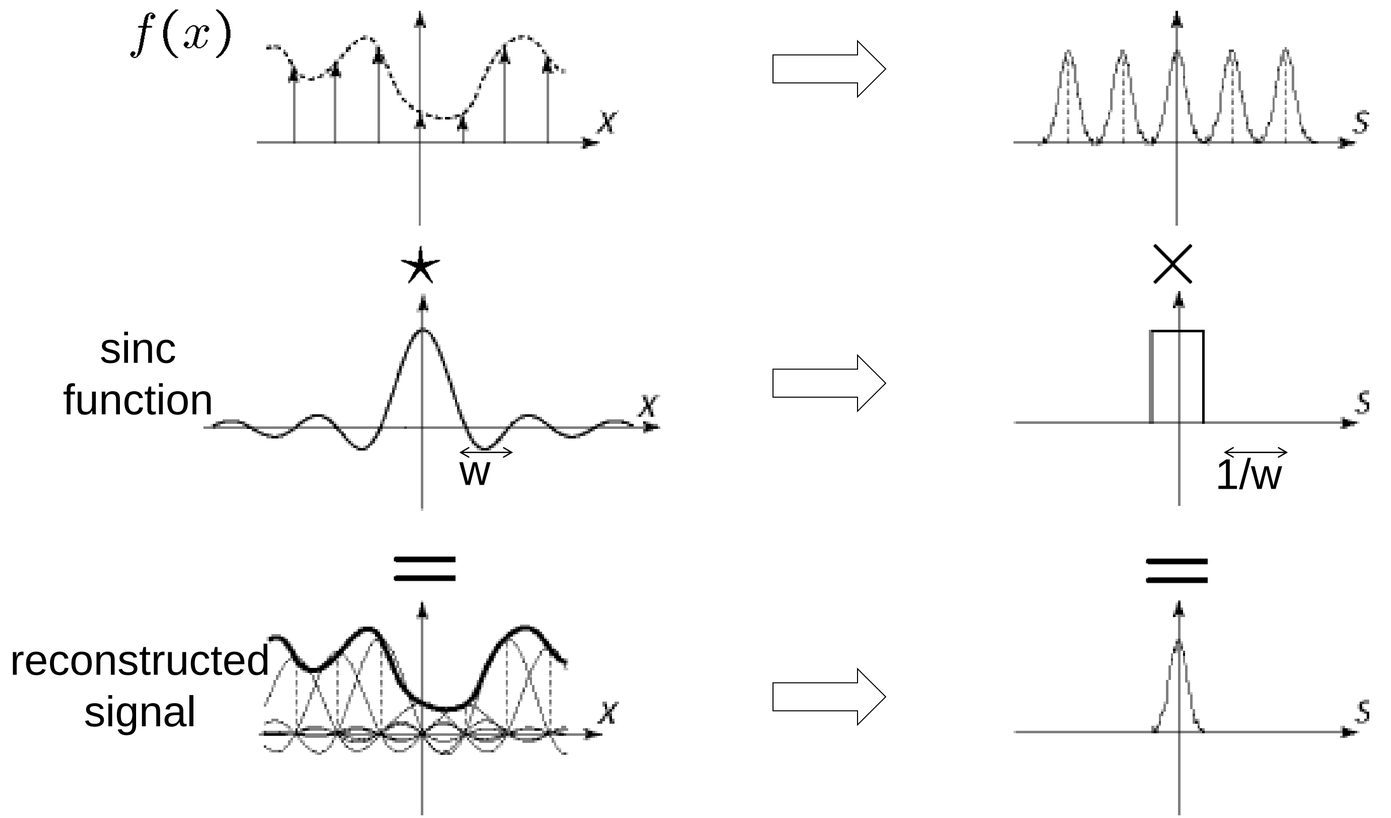
Sampling

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



Reconstruction

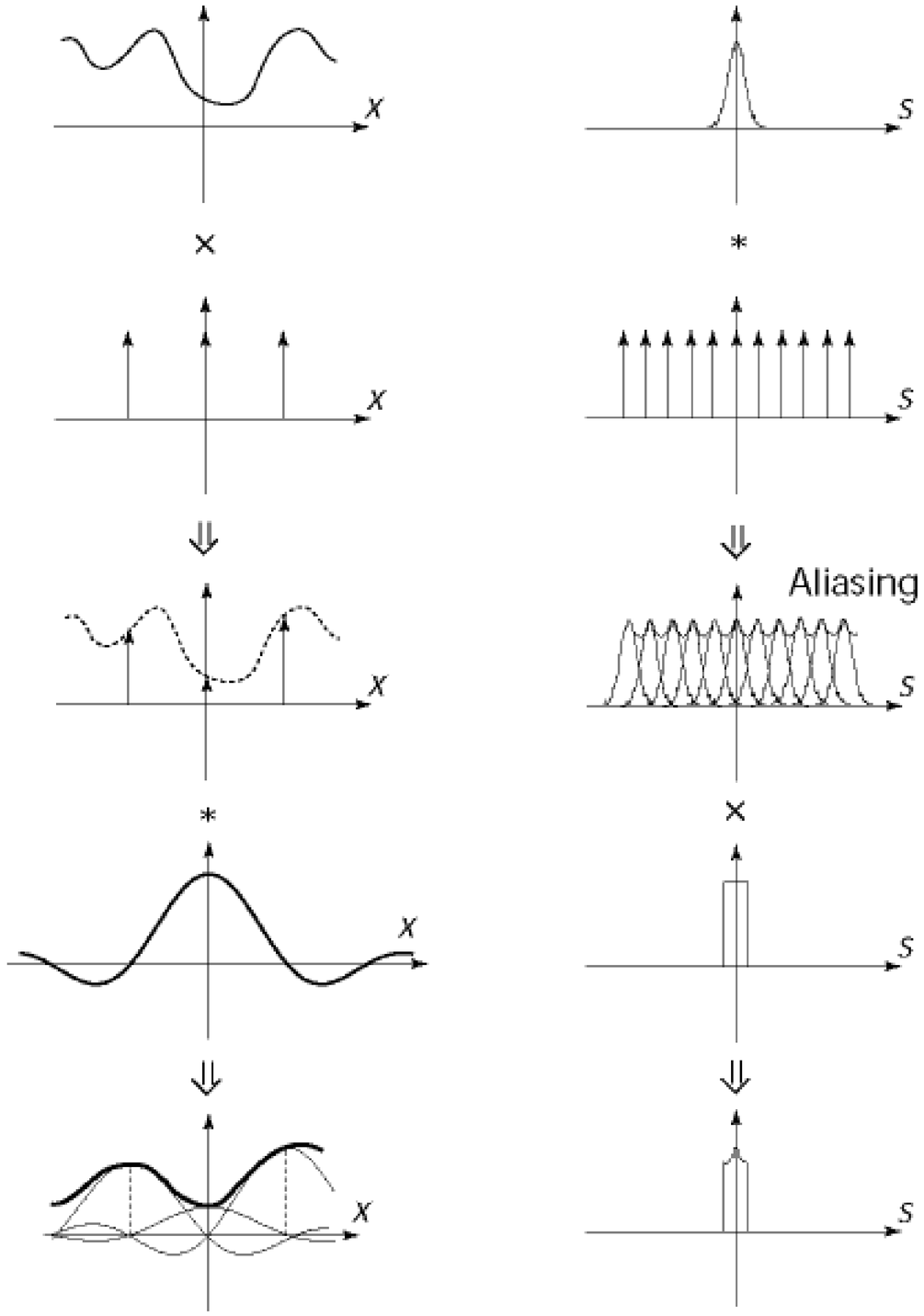
$$F(s) = \int_{-\infty}^{\infty} f(x) e^{-i2\pi sx} dx$$



Spatial domain

Frequency domain

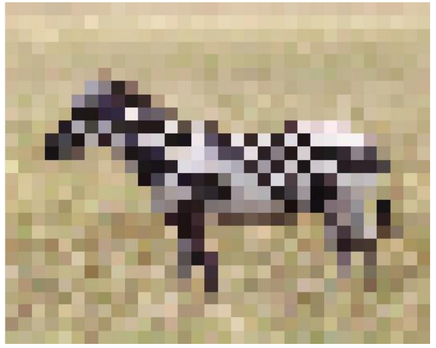
What happens when the sampling rate is too low?

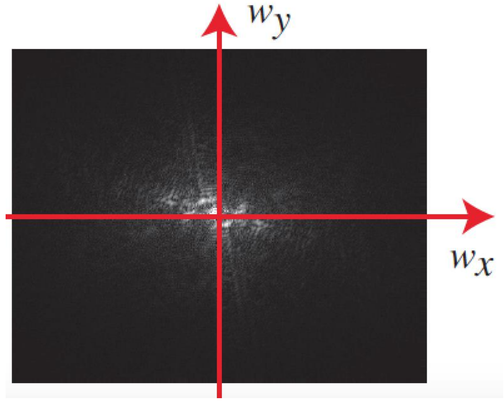


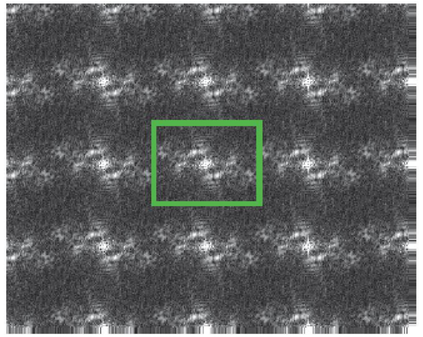
103×128

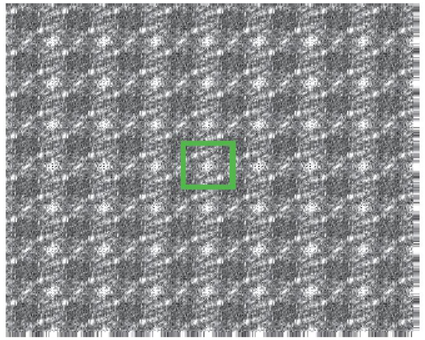
52×64

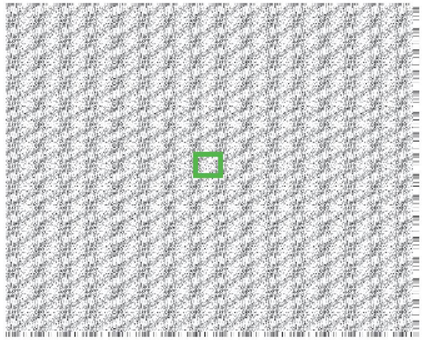
26×32





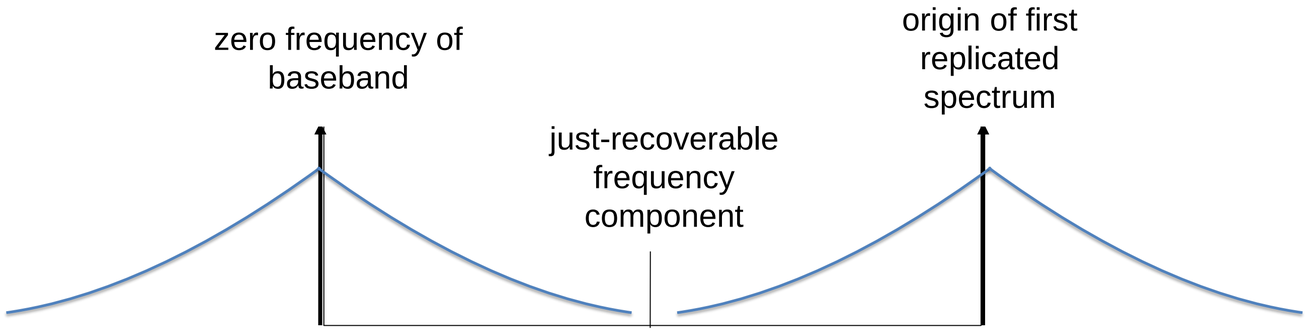




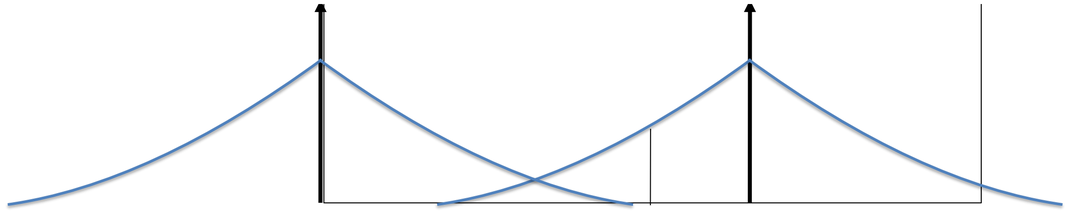


Aliasing

aliasing



no aliasing

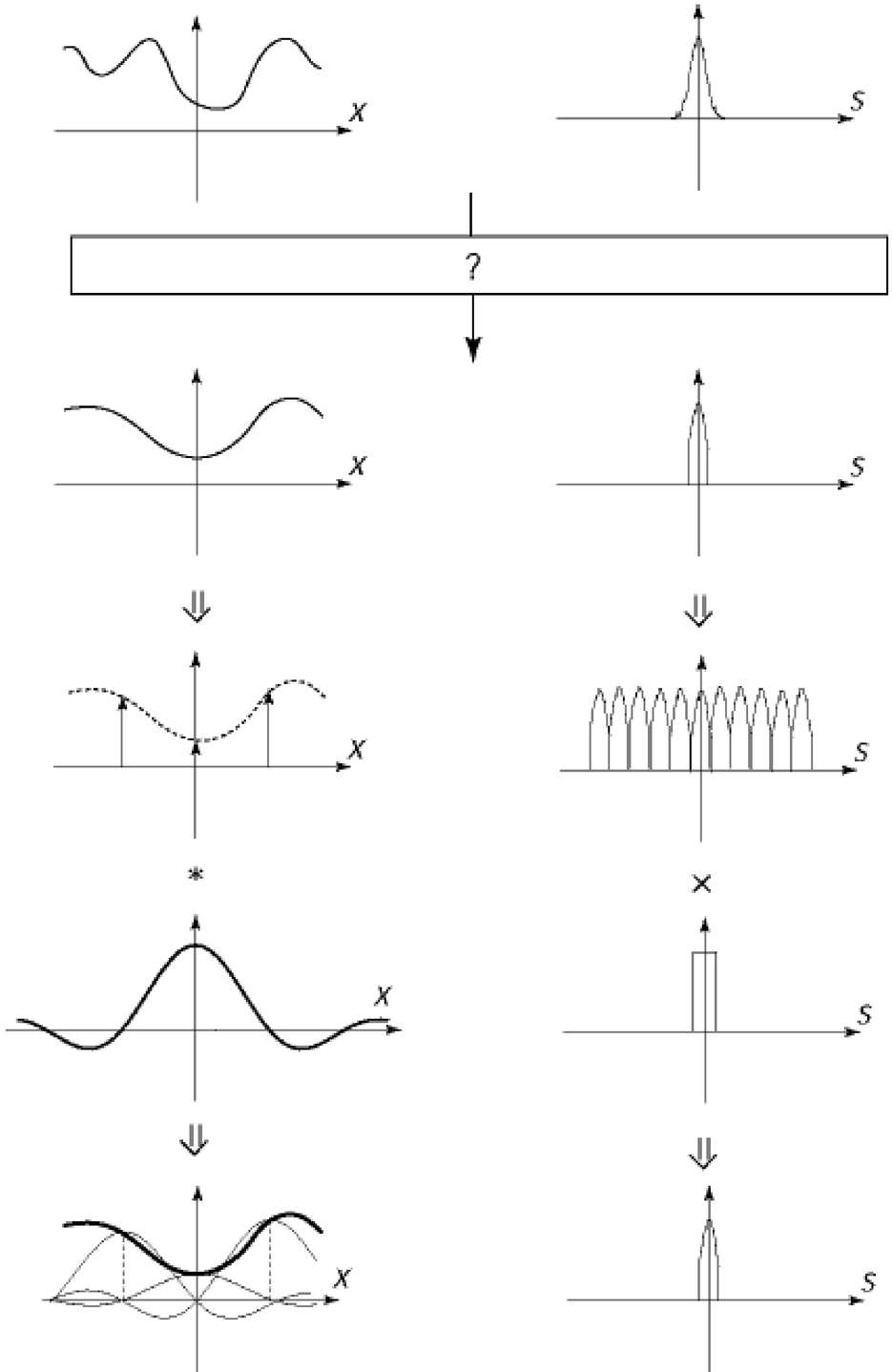


aliasing

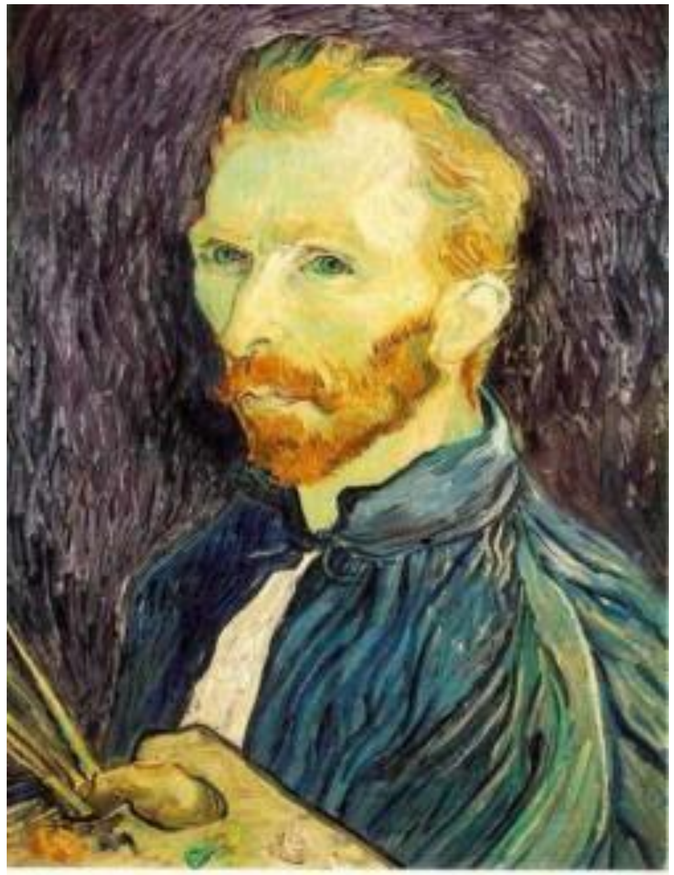
Anti-aliasing by

pre-filtering

- theoretical ideal pre-filter is a sinc function
- Gaussian, cubic filters work better in practice



Subsampling with Gaussian pre-filtering



Gaussian 1/2

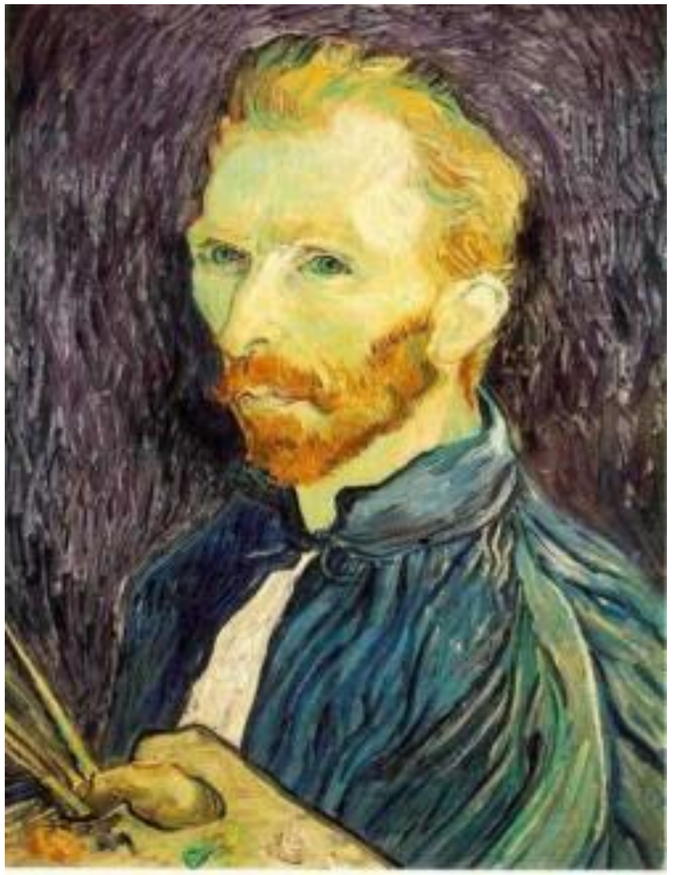
G 1/4

G 1/8

Solution: filter the image, *then* subsample

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

Subsampling with Gaussian pre-filtering



Gaussian 1/2

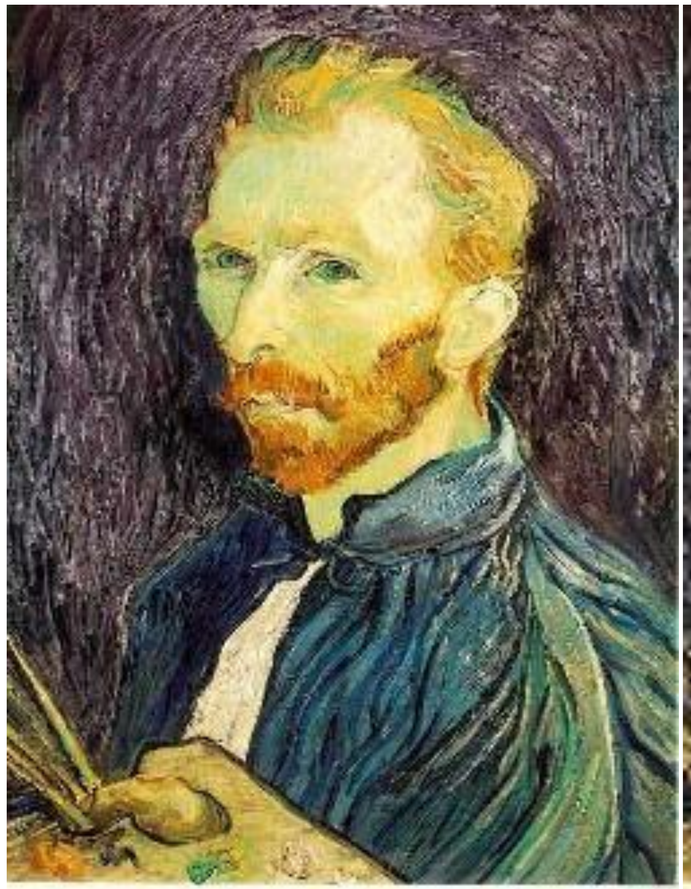
G 1/4

G 1/8

Solution: filter the image, *then* subsample

- Filter size should double for each ½ size reduction. Why?
- How can we speed this up?

Compare with...



1/2

1/4 (2x zoom)

1/8 (4x zoom)

Why does this look so crufty?

Antialising filtering

Before sampling, apply a low pass-filter to remove all the frequencies that will produce aliasing.

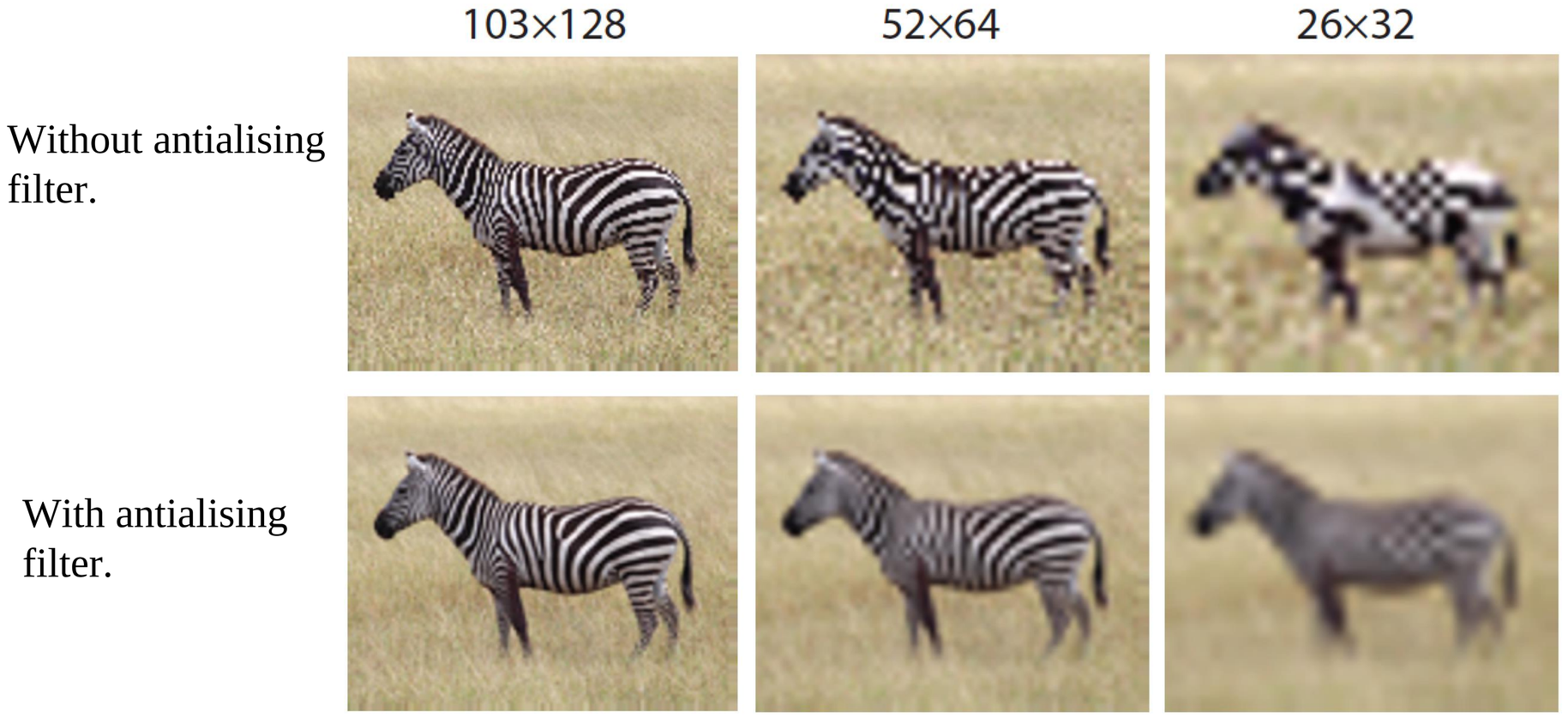
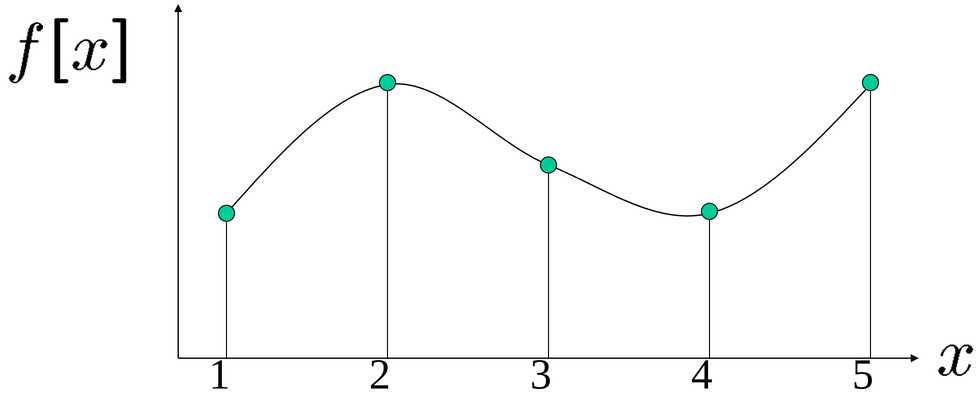


Image resampling

So far, we considered only power-of-two subsampling

- What about arbitrary scale reduction?
- How can we increase the size of the image?

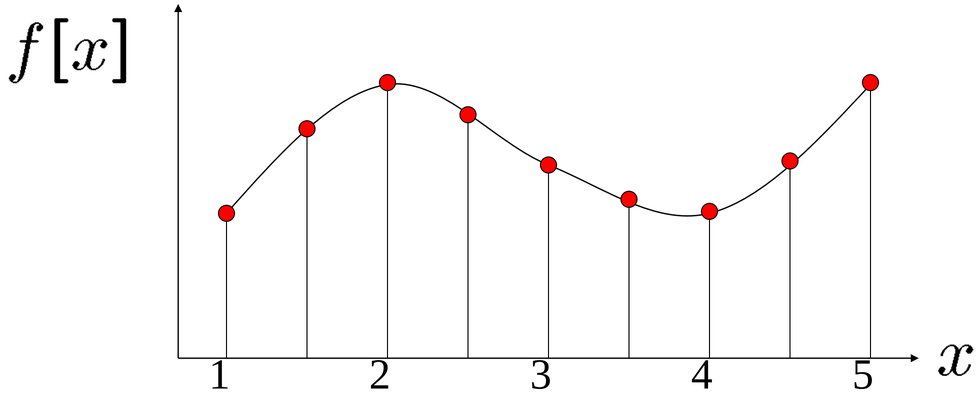


d = 1 in this example

Image resampling

So far, we considered only power-of-two subsampling

- What about arbitrary scale reduction?
- How can we increase the size of the image?



d = 1 in this example

Recall how a digital image is formed

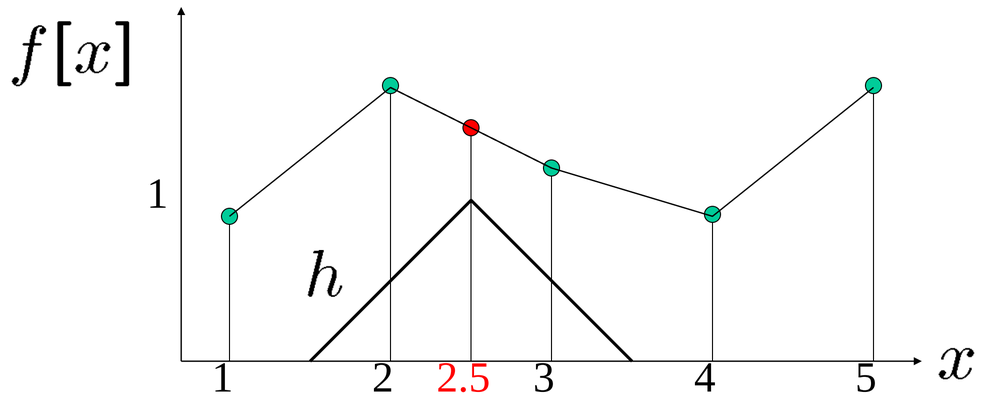
$$f[x, y] = \text{quantize}\{f(xd, yd)\}$$

- It is a discrete point-sampling of a continuous function
- If we could somehow reconstruct the original function, any new image could be generated, at any resolution and scale

Image resampling

So what to do if we don't know f

- Answer: guess an approximation $\tilde{f}$
- Can be done in a principled way: filtering



d = 1 in this example

Image reconstruction

- Convert f to a continuous function

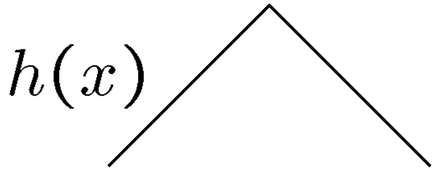
$$f(x) = f[\tfrac{x}{d}] \text{ when } \tfrac{x}{d} \text{ is an integer, 0 otherwise}$$

- Reconstruct by convolution:

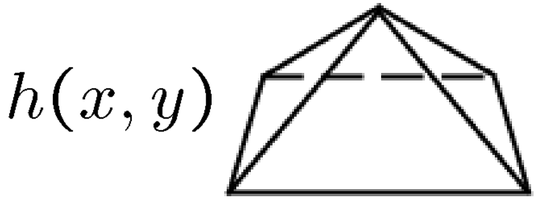
$$\tilde{f} = h \star f$$

Resampling filters

What does the 2D version of this hat function look like?



performs
linear interpolation



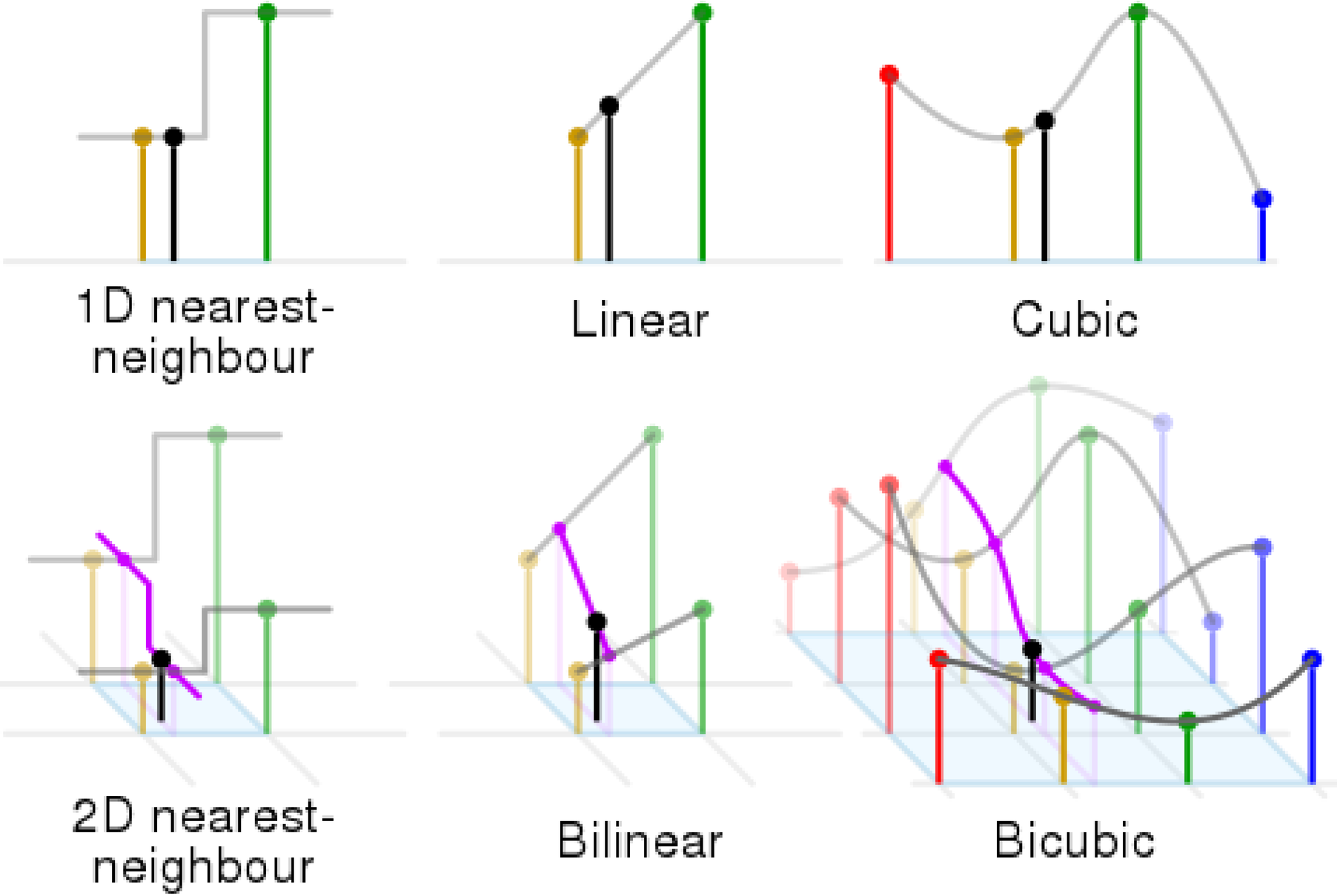
(tent function) performs
bilinear interpolation

Better filters give better resampled images

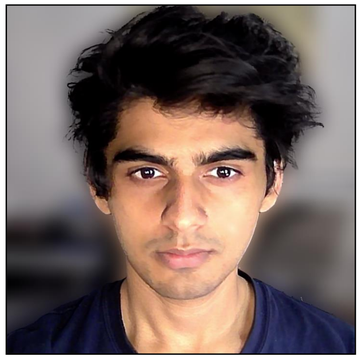
- Bicubic is common choice
  - fit 3rd degree polynomial surface to pixels in neighborhood
  - can also be implemented by a convolution

Steve Seitz's 5-min video:
https://youtu.be/Xj129kA3Ci0?si=cAf5isBb0GzdLi-7

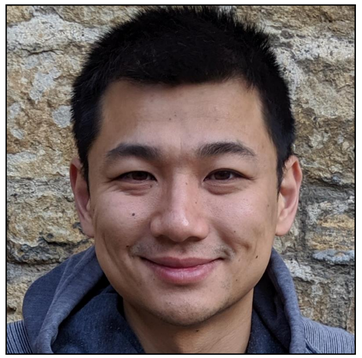
Types of Interpolation



On Aliased Resizing and Surprising Subtleties in GAN Evaluation



Gaurav Parmar



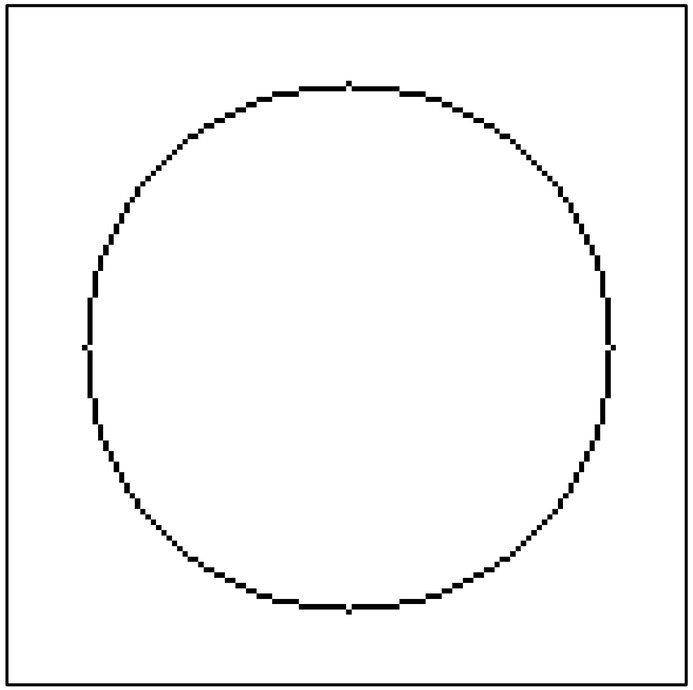
Richard Zhang

Jun-Yan Zhu

Carnegie Mellon University

Downsampling a circle

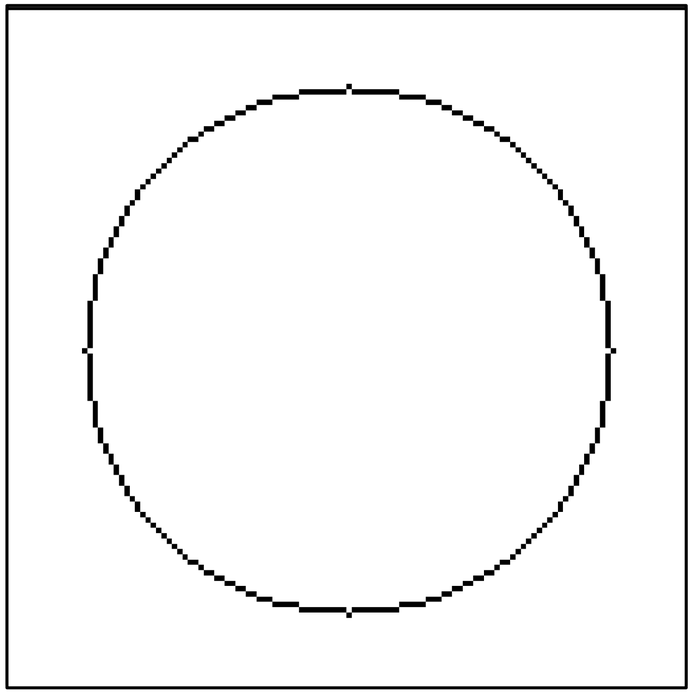
input image



Downsampling a circle

naïve nearest

↓1.0

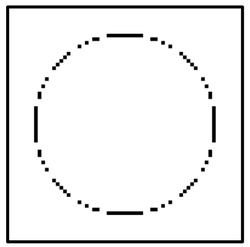


128x128

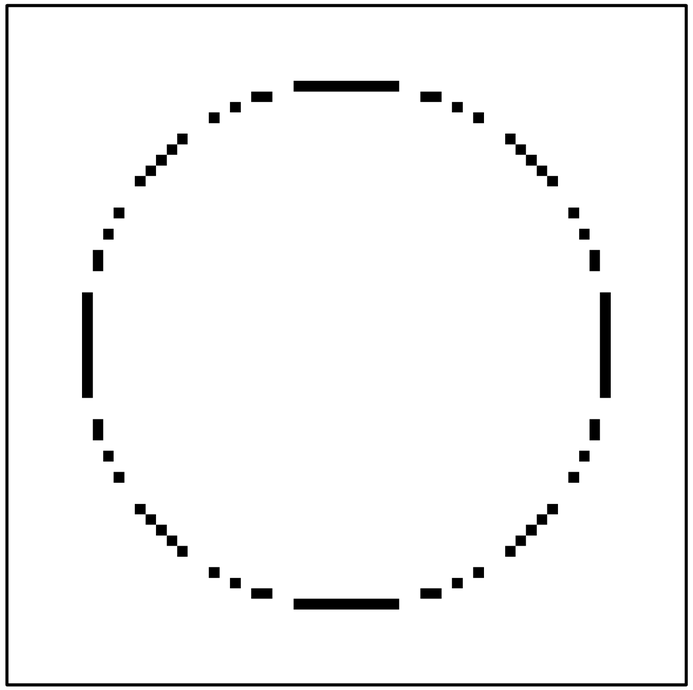
Downsampling a circle

↓2

naïve
nearest

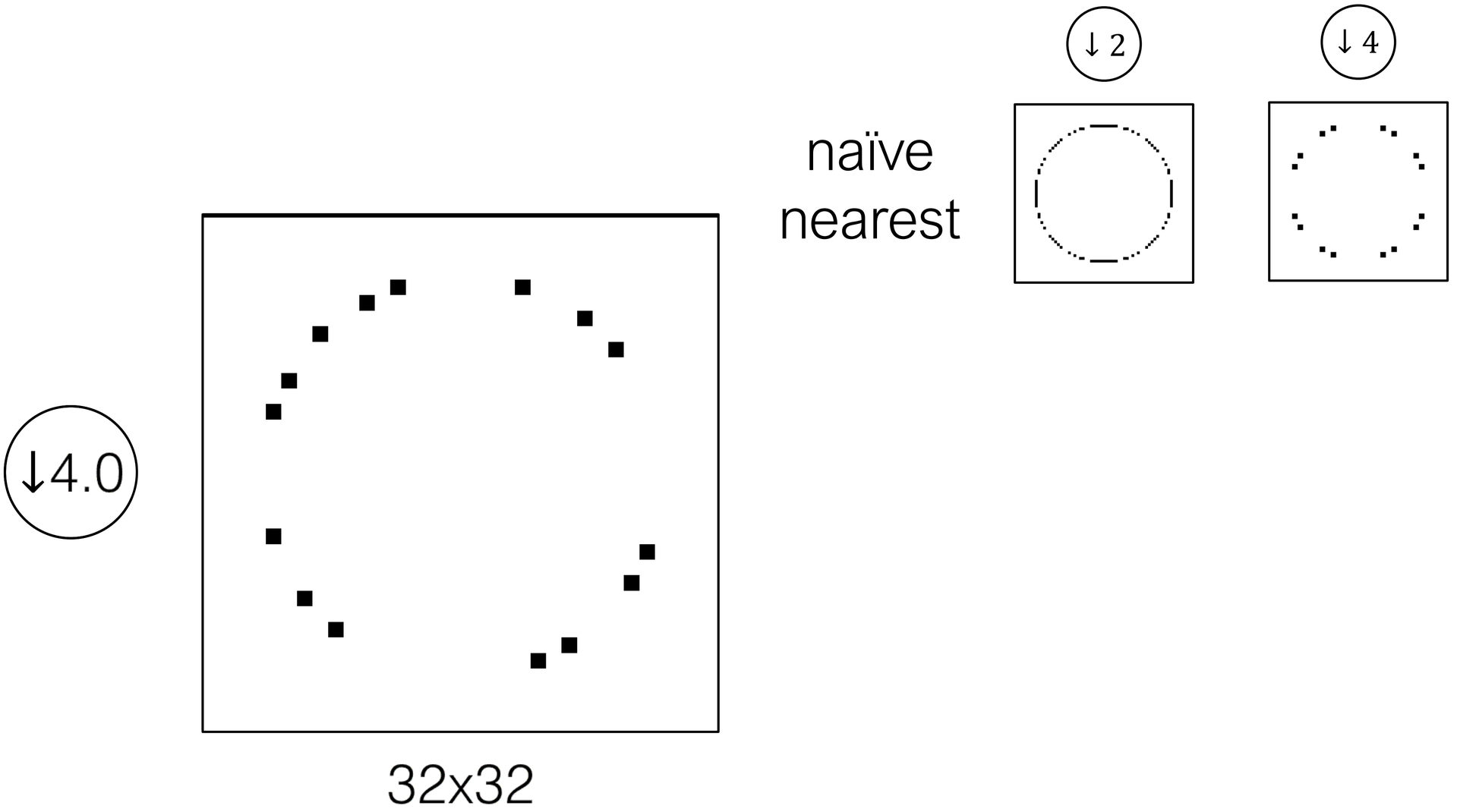


↓2.0

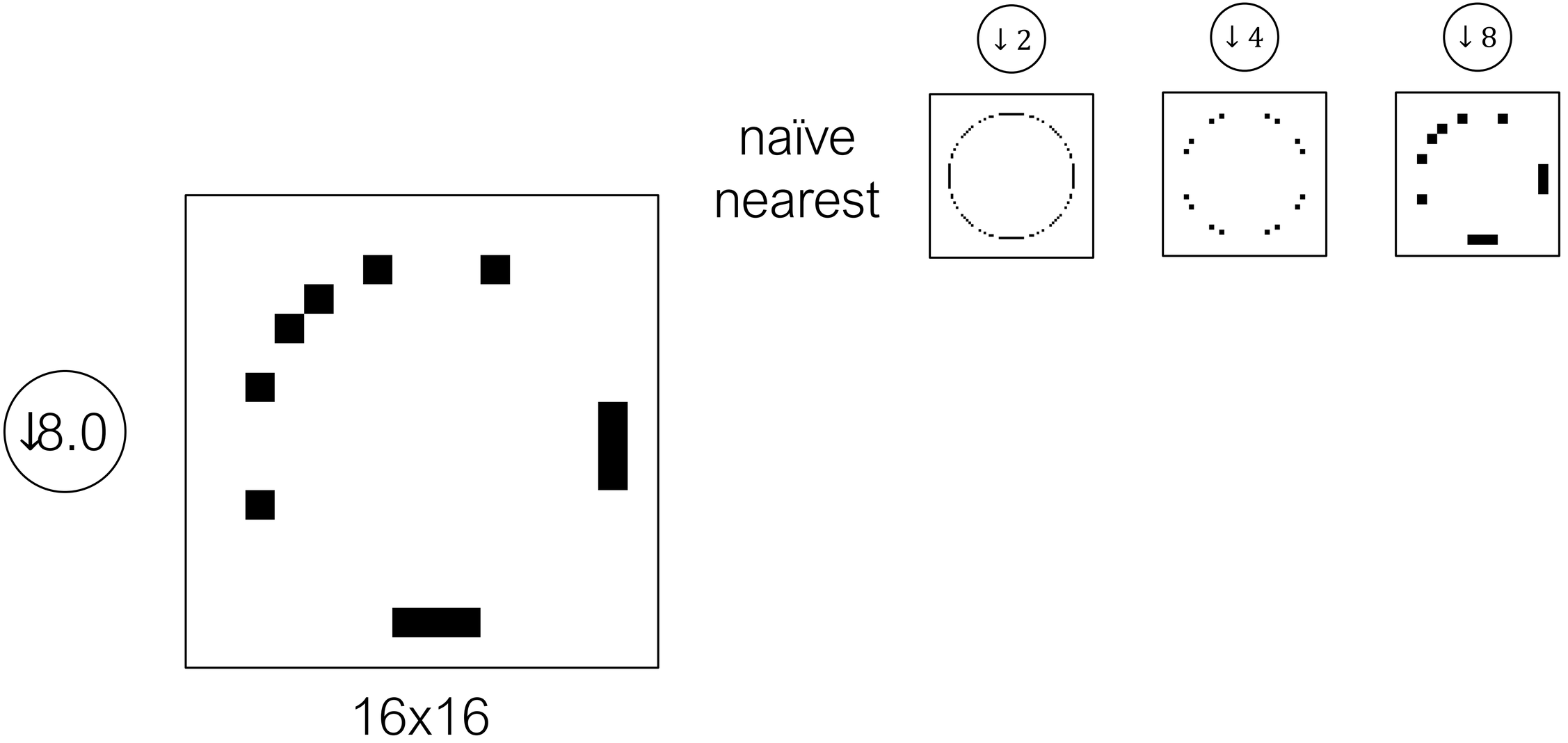


64x64

Downsampling a circle

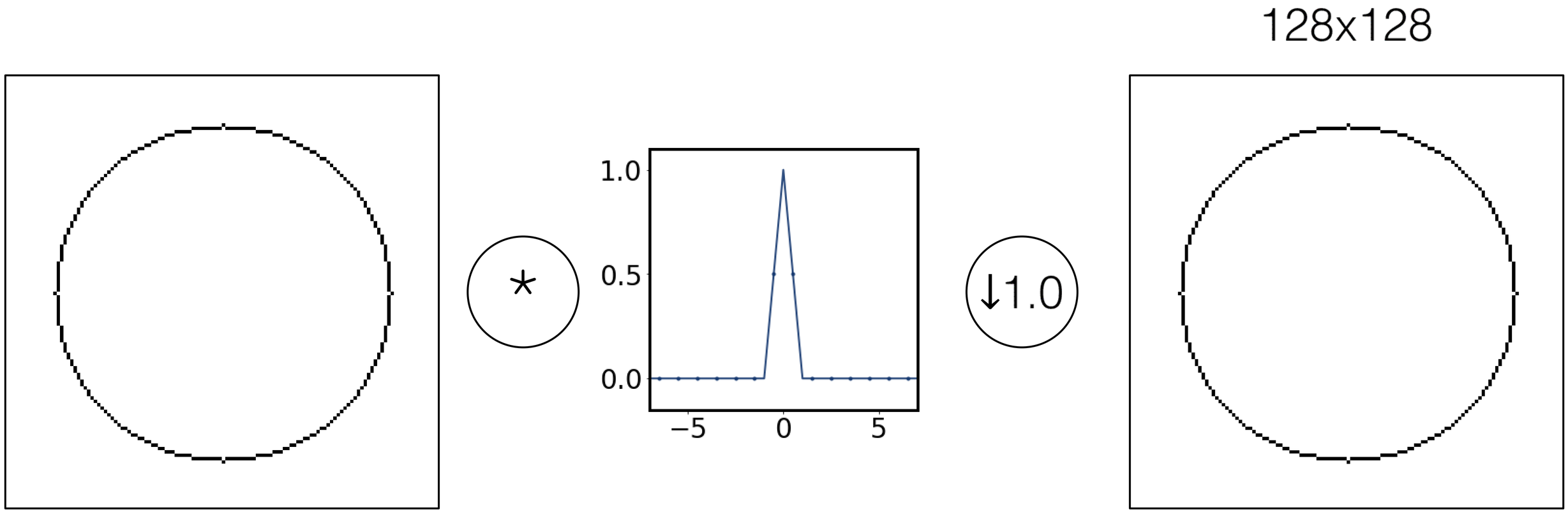


Downsampling a circle



Downsampling a circle

prefiltering the image

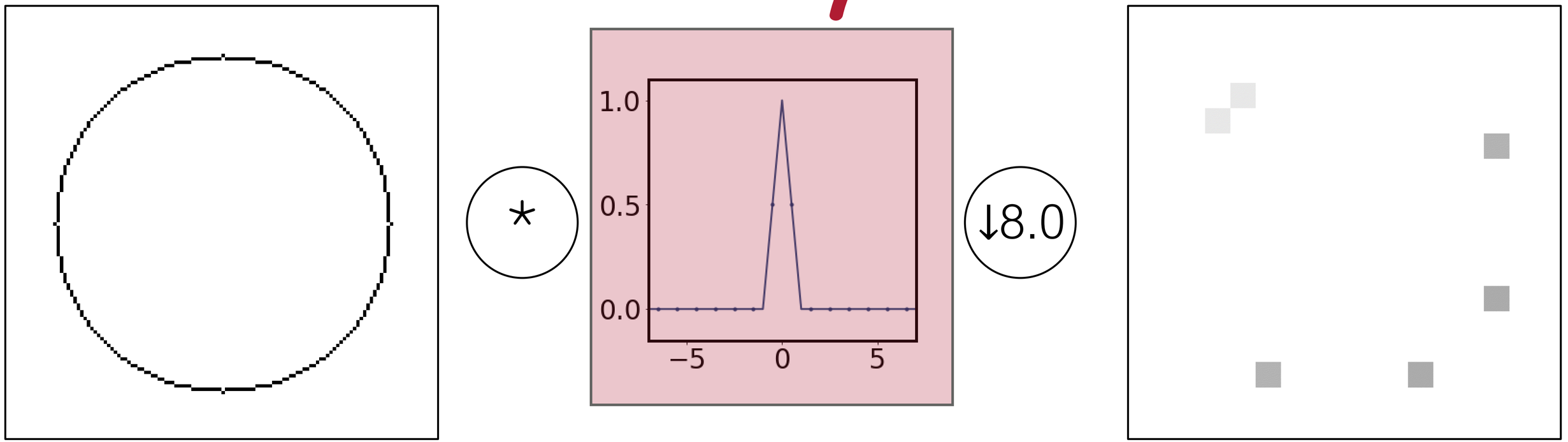


Downsampling a circle

prefiltering the image

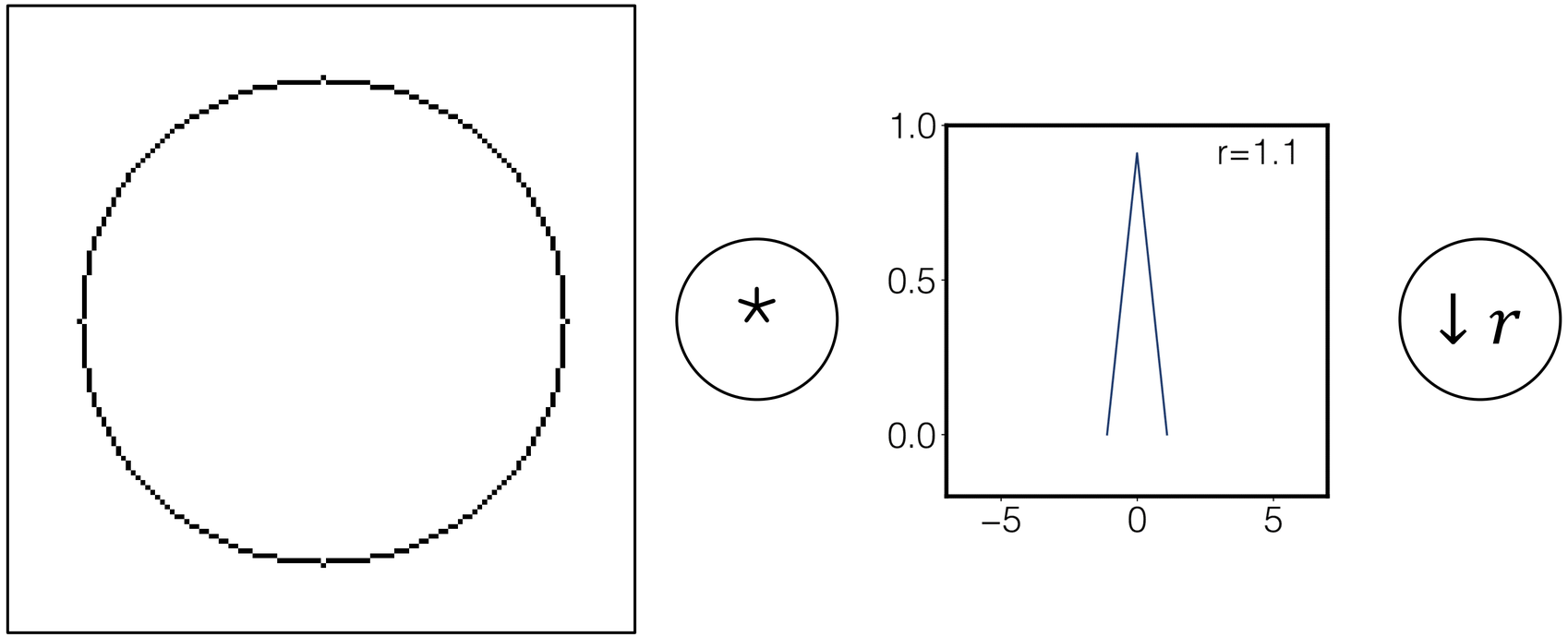
should not be fixed!

128x128



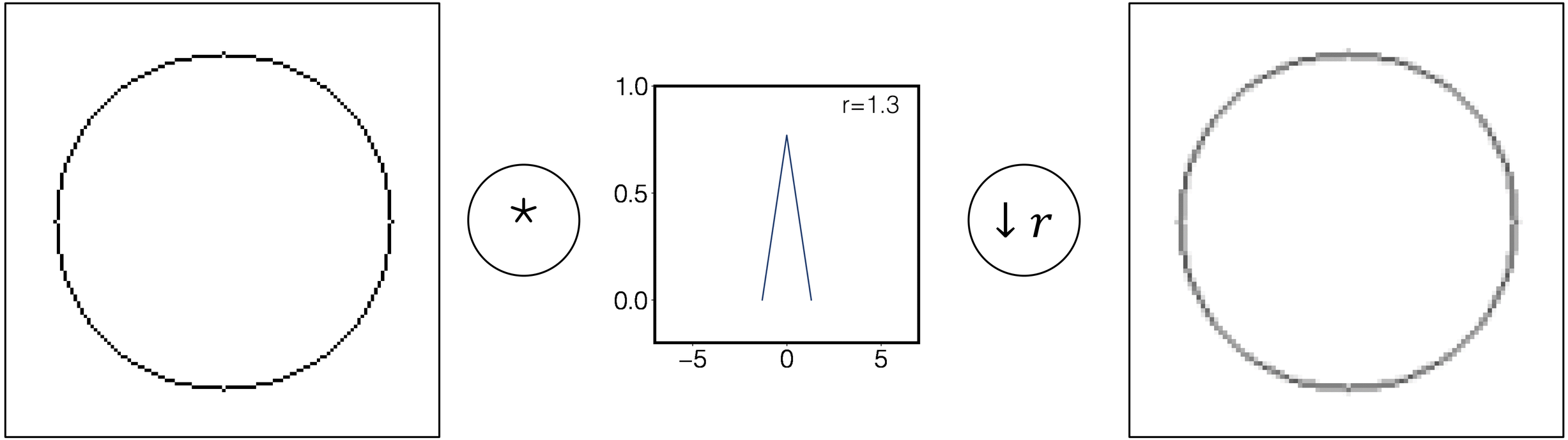
Downsampling a circle

prefiltering the image , adapting the width

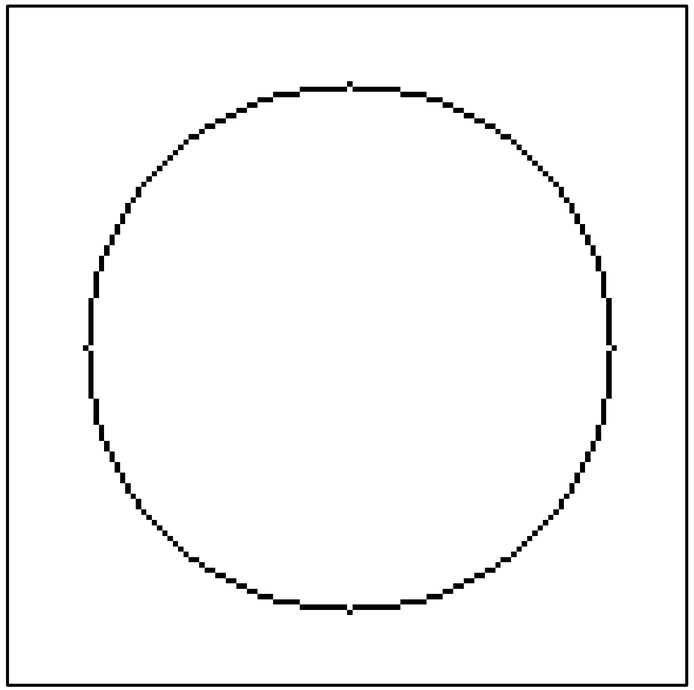


Downsampling a circle

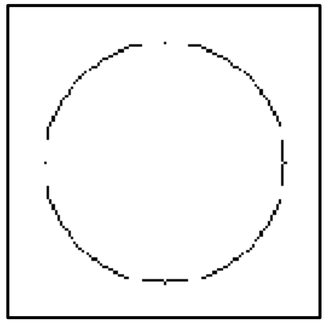
prefiltering the image , adapting the width



Downsampling a circle

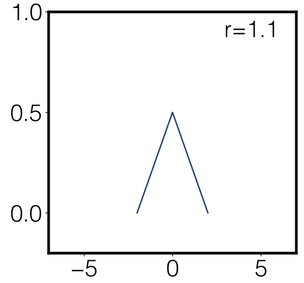


naïve
nearest

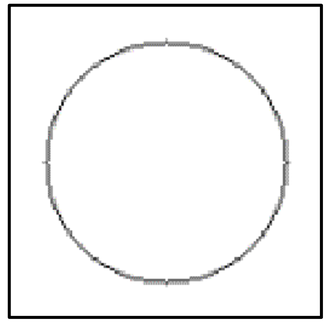


fixed
filter

$*$

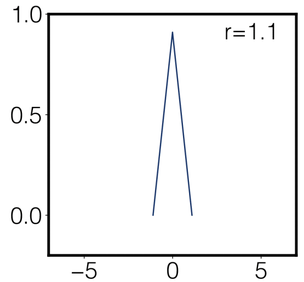


$\downarrow r$

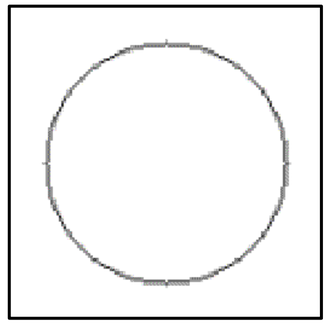


adaptive
filter

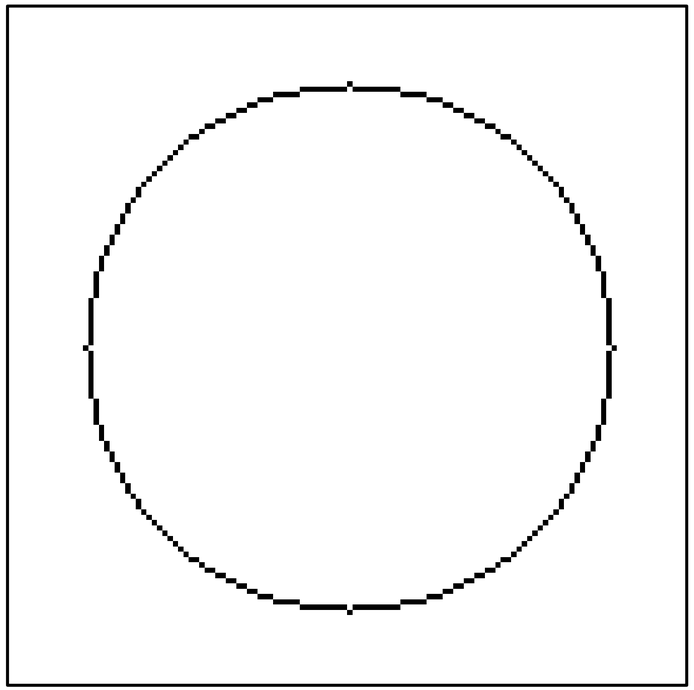
$*$



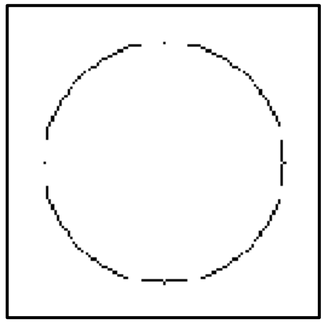
$\downarrow r$



Downsampling a circle

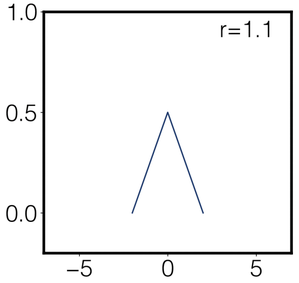


naïve
nearest

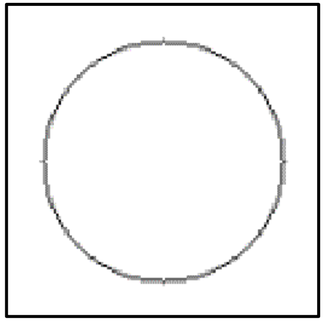


fixed
filter

$*$

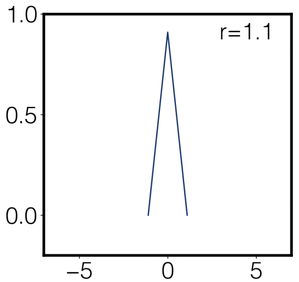


$\downarrow r$

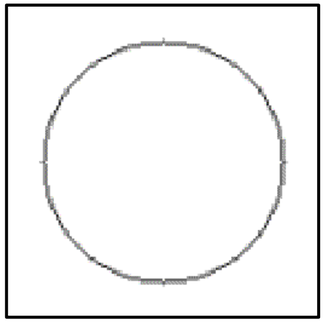


adaptive
filter

$*$



$\downarrow r$



PyTorch
(with default flags)

TensorFlow

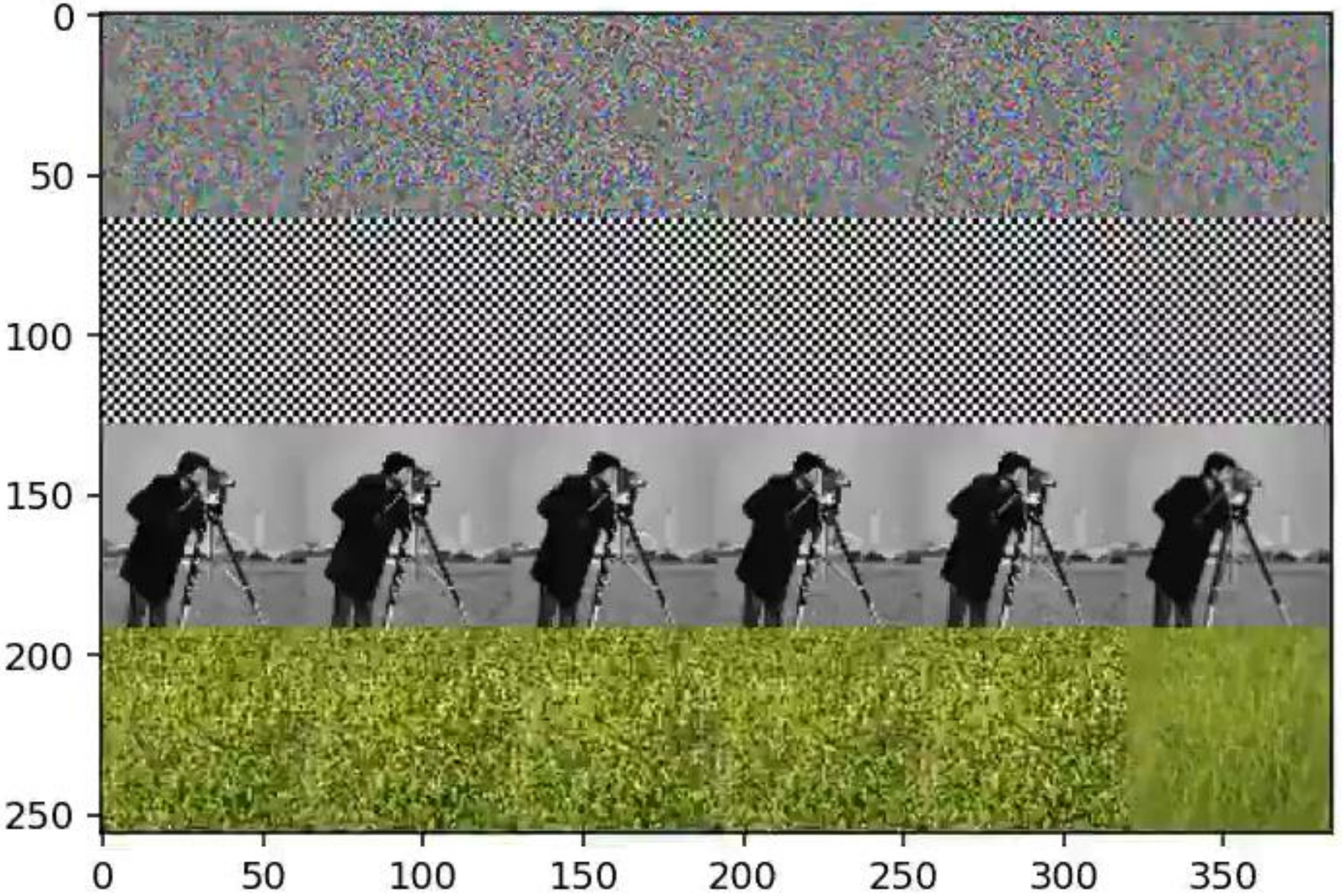
OpenCV

MXNet

PIL

A real problem!

128 x 128 → 64x64



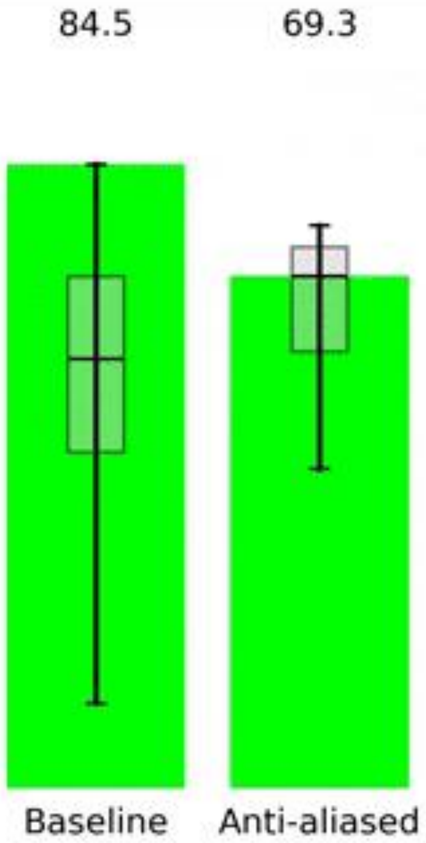
Open-CV:
default, bicubic, Lanczos4

Pytorch:
bilinear, bicubic

PIL: Lanczos

Credit: @jaakkolehtinen

problems in ConvNets too



`pip install antialiased-cnns`

Making Convolutional Networks Shift-Invariant Again,
Richard Zhang ICML 2019