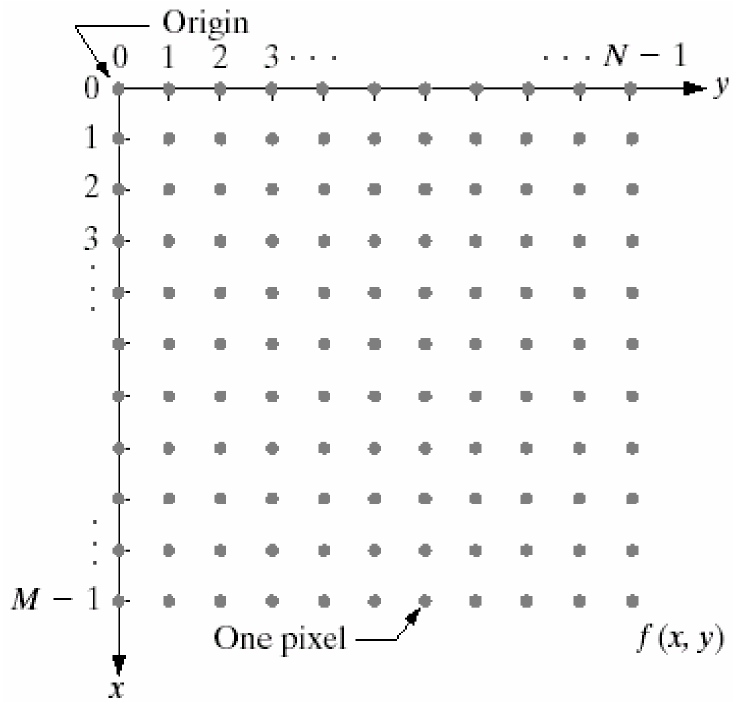
Sampling and Reconstruction

Chapter 4
Intro. to Computer Graphics
Spring 2008, Y.G. Shin

A Simple Image Model

- Image: a 2-D light-intensity function *f(x,y)*
- The value of *f* at (x,y) → the intensity (brightness) of the image at that point
- $0 < f(x,y) < \infty$



Digital Image Acquisition

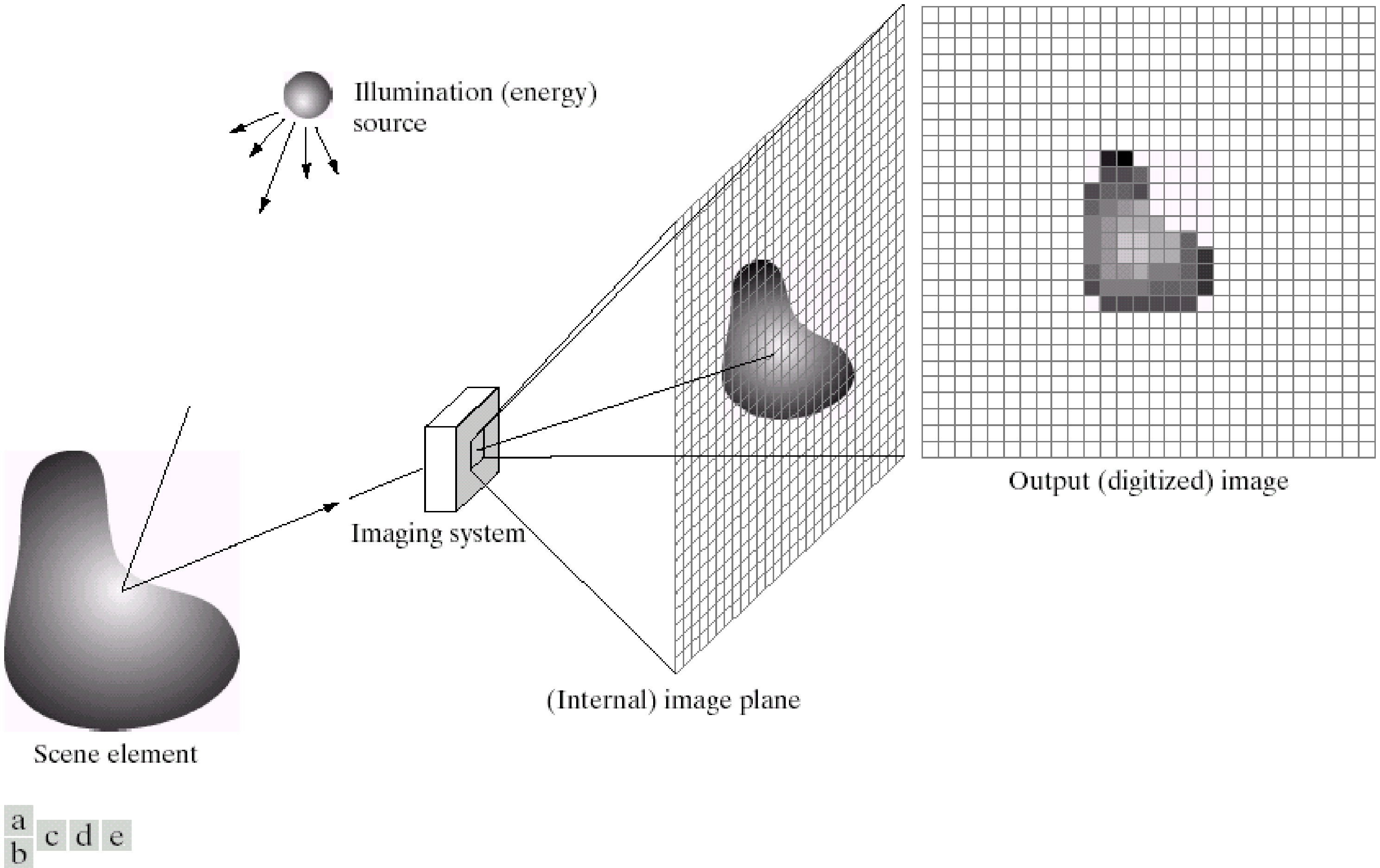


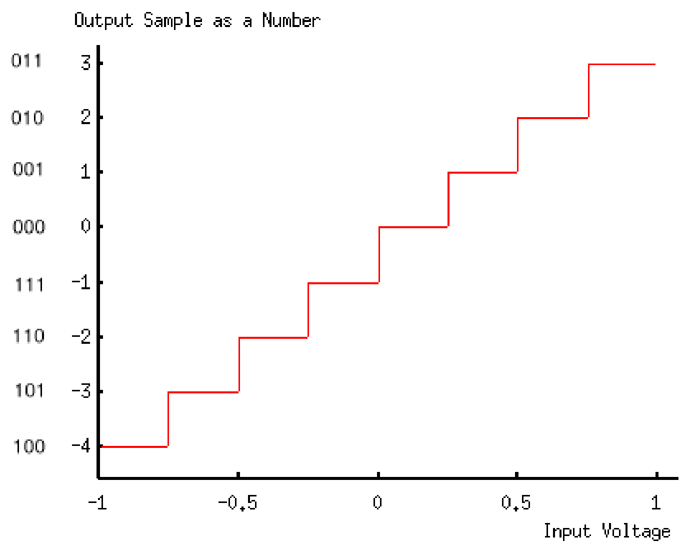
FIGURE 2.15 An example of the digital image acquisition process. (a) Energy ("illumination") source. (b) An element of a scene. (c) Imaging system. (d) Projection of the scene onto the image plane. (e) Digitized image.

Sampling & Quantization

- Sampling: partitioning xy plane into a grid
  - the coordinate of the center of each grid is a pair of elements from the Cartesian product Z x Z (Z^2), Z: set of real integers
- Where Does Sampling Occur?
  - Almost all data we are dealing with is discrete
    - Evaluation of sampled functions at arbitrary sites
    - Volume rendering
    - Isosurface extraction
    - Ray tracing

Sampling & Quantization

- Quantization: once the signal has been sampled, it needs to be quantized to turn the samples into numbers which we can process.



Quantization means that we break the full positive and negative range of the sample value into N sections and then code it in log2 (N) bits.

Digital Image

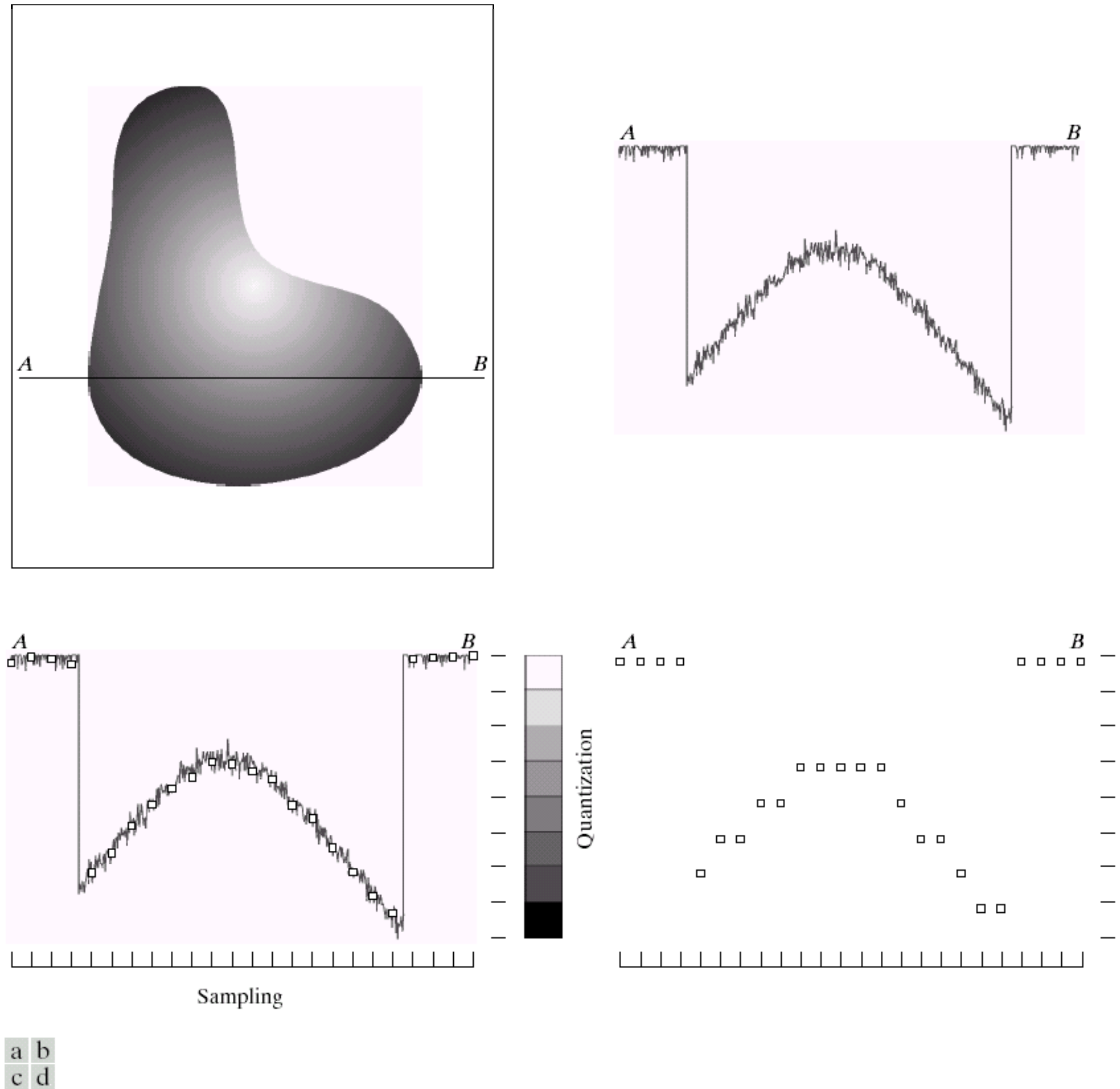


FIGURE 2.16 Generating a digital image. (a) Continuous image. (b) A scan line from A to B in the continuous image, used to illustrate the concepts of sampling and quantization. (c) Sampling and quantization. (d) Digital scan line.

Sampling & Quantization

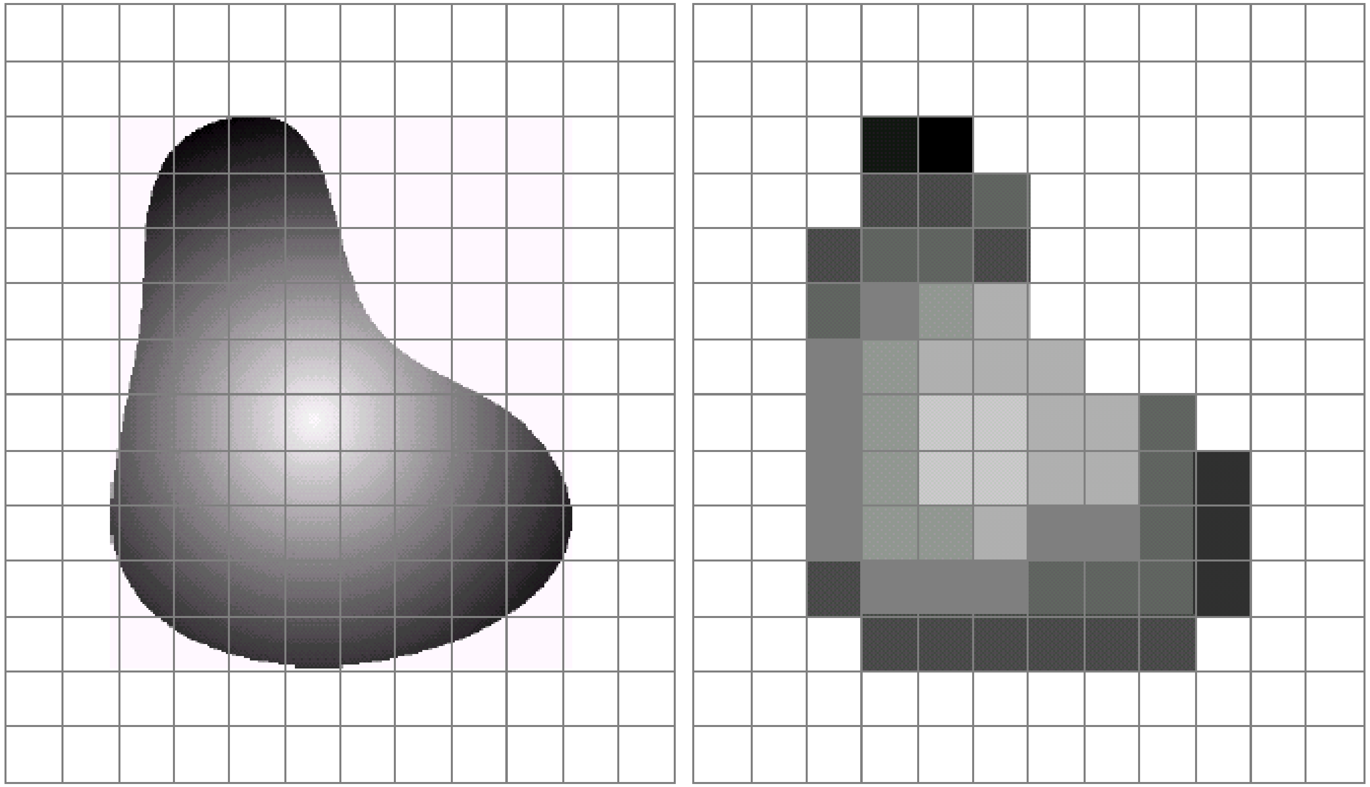


FIGURE 2.17 (a) Continuos image projected onto a sensor array. (b) Result of image sampling and quantization.

Sampling & Quantization

- The digitization process requires decisions about:
  - Values for N,M (where N x M: the image array)
  - AND, the number of discrete gray levels, G, allowed for each pixel.
- Usually, these quantities are integer powers of two: $N=2^n$, $M=2^m$ and $G=2^k$
- Another assumption is that the discrete levels are equally spaced between 0 and L-1 in the gray scale.

Examples

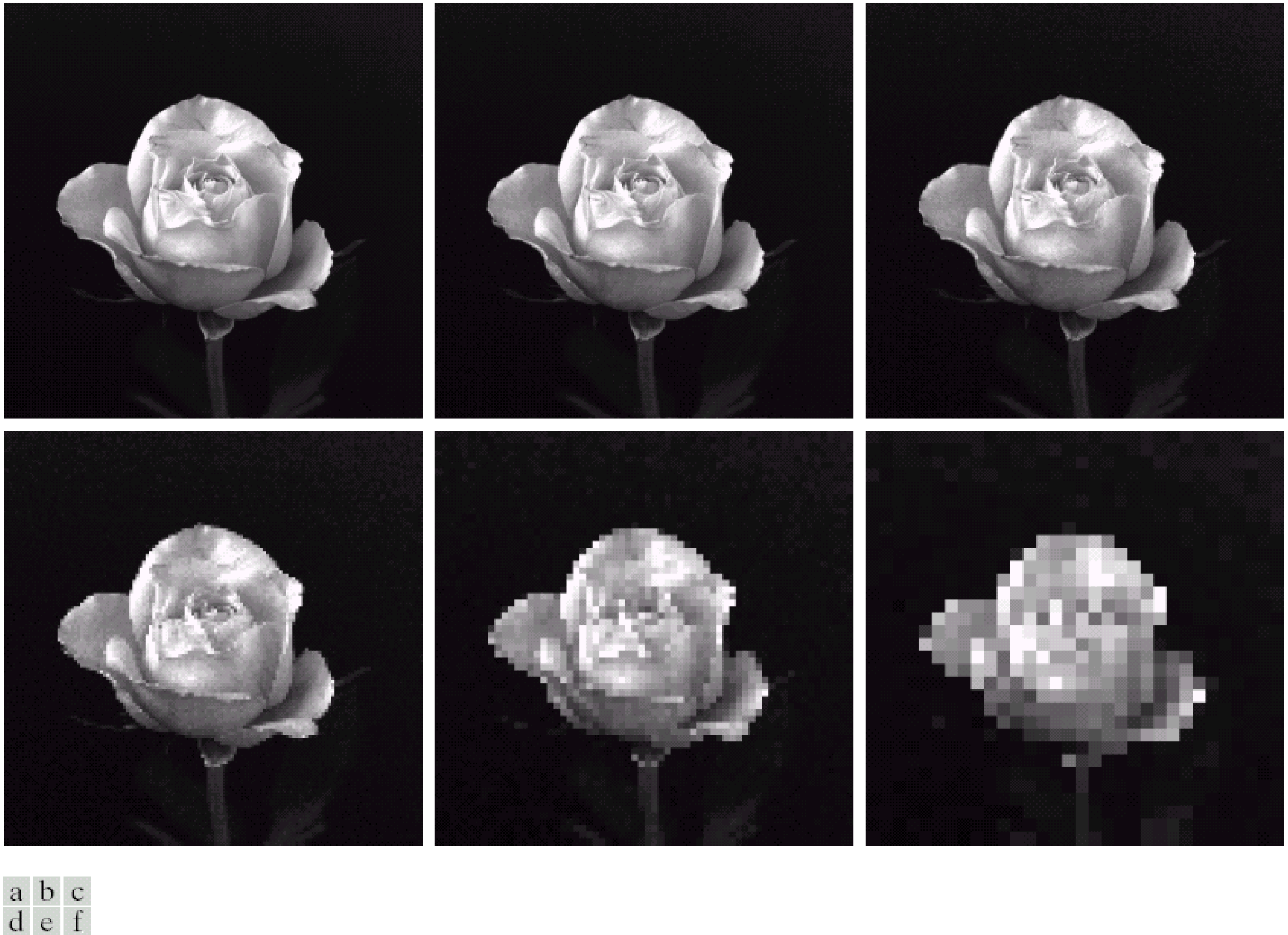
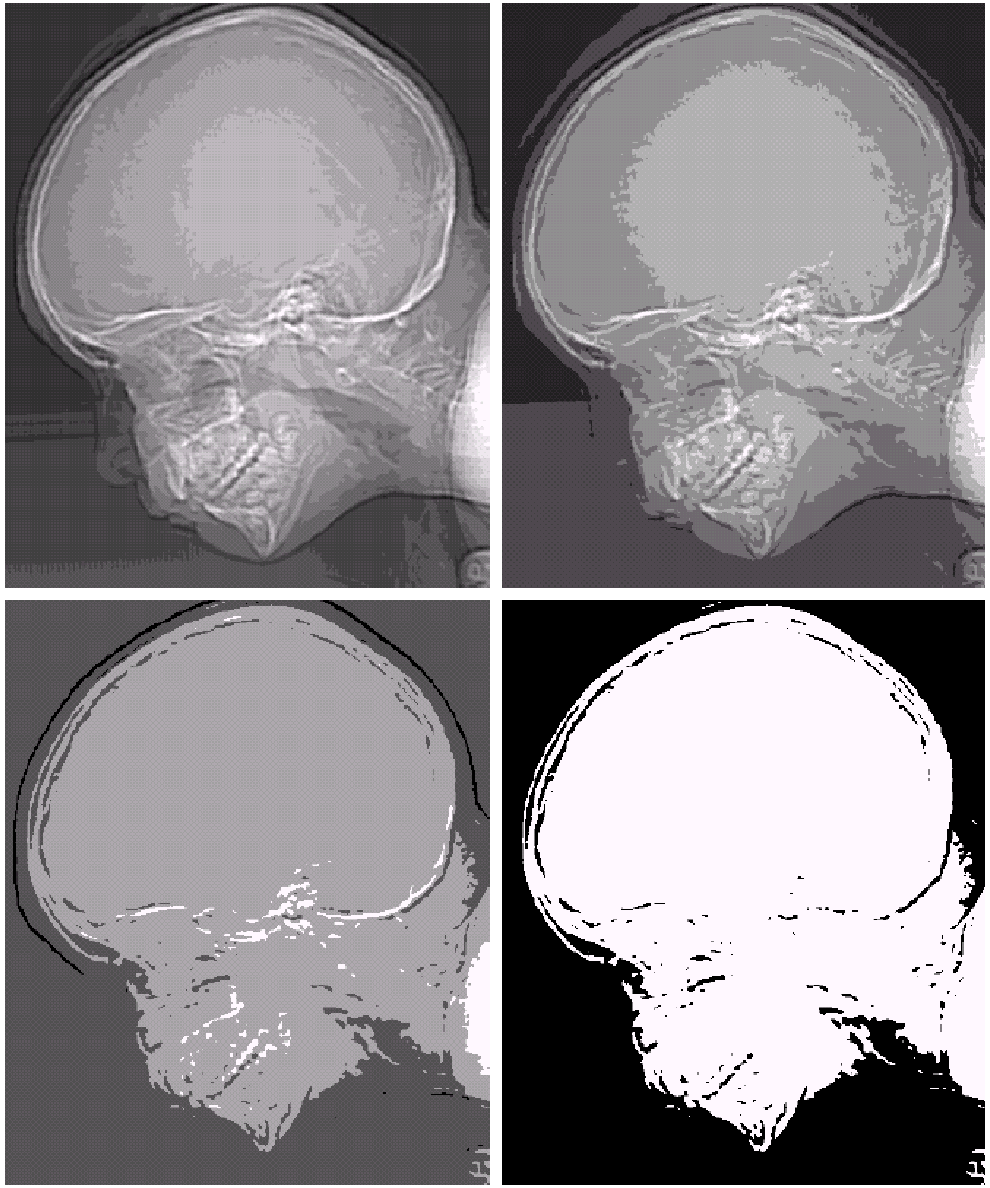


FIGURE 2.20 (a) 1024 × 1024, 8-bit image. (b) 512 × 512 image resampled into 1024 × 1024 pixels by row and column duplication. (c) through (f) 256 × 256, 128 × 128, 64 × 64, and 32 × 32 images resampled into 1024 × 1024 pixels.

Examples



e f
g h

FIGURE 2.21 *(Continued)* (e)–(h) Image displayed in 16, 8, 4, and 2 gray levels. (Original courtesy of Dr. David R. Pickens, Department of Radiology & Radiological Sciences, Vanderbilt University Medical Center.)

Sampling & Quantization

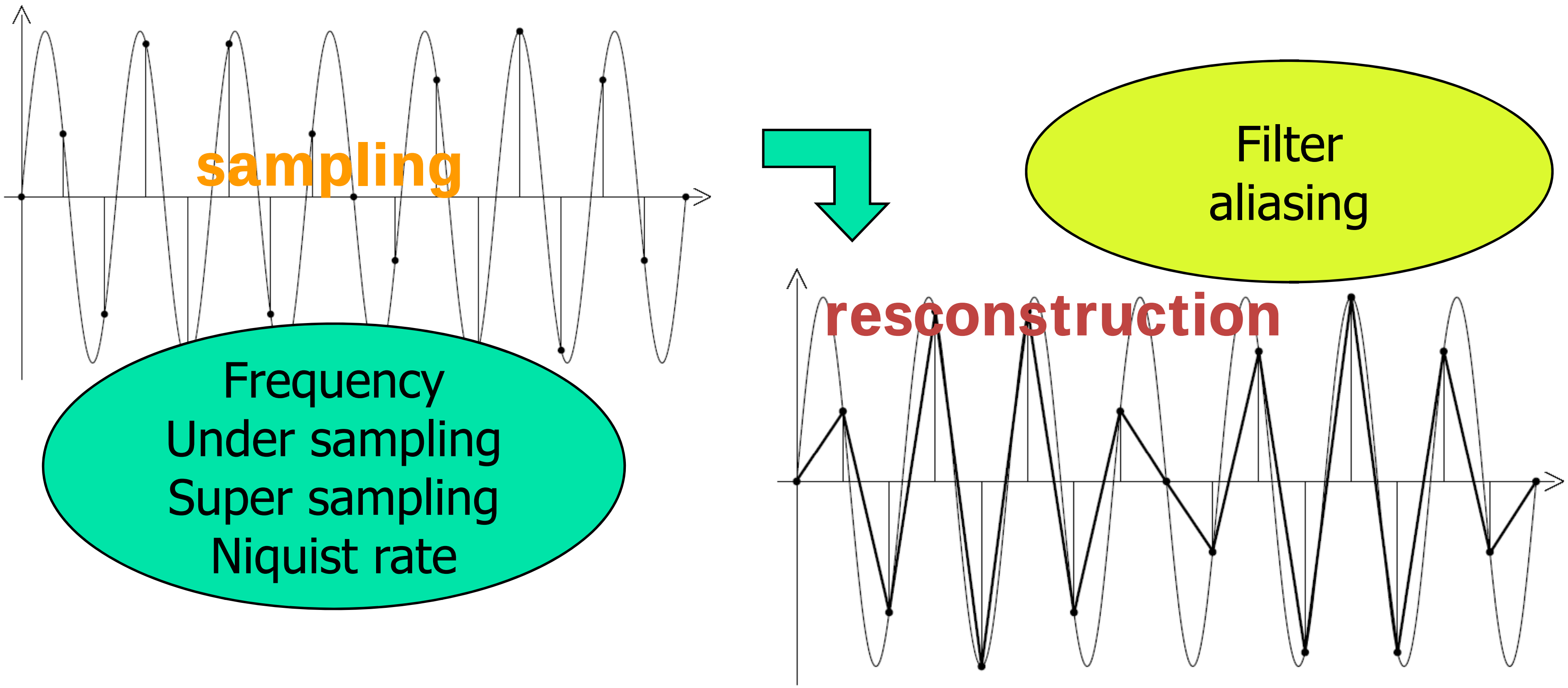
- If b is the number of bits required to store a digitized image then:
  - $b = N \times M \times k$ (if $M=N$, then $b=N^2k$)
- Storage for various values of N and k

TABLE 2.1
Number of storage bits for various values of N and k.

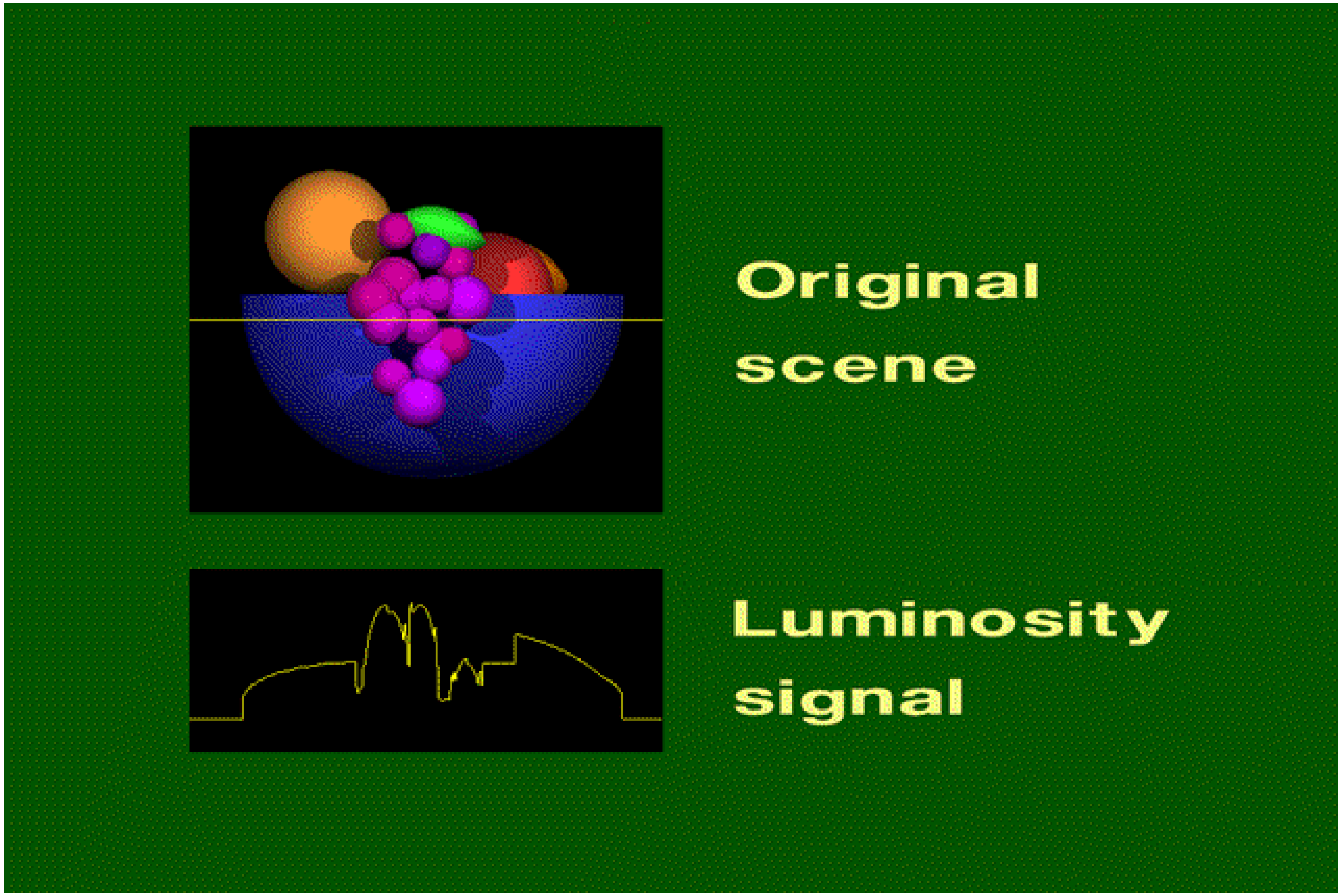
N/k	1 (L = 2)	2 (L = 4)	3 (L = 8)	4 (L = 16)	5 (L = 32)	6 (L = 64)	7 (L = 128)	8 (L = 256)
32	1,024	2,048	3,072	4,096	5,120	6,144	7,168	8,192
64	4,096	8,192	12,288	16,384	20,480	24,576	28,672	32,768
128	16,384	32,768	49,152	65,536	81,920	98,304	114,688	131,072
256	65,536	131,072	196,608	262,144	327,680	393,216	458,752	524,288
512	262,144	524,288	786,432	1,048,576	1,310,720	1,572,864	1,835,008	2,097,152
1024	1,048,576	2,097,152	3,145,728	4,194,304	5,242,880	6,291,456	7,340,032	8,388,608
2048	4,194,304	8,388,608	12,582,912	16,777,216	20,971,520	25,165,824	29,369,128	33,554,432
4096	16,777,216	33,554,432	50,331,648	67,108,864	83,886,080	100,663,296	117,440,512	134,217,728
8192	67,108,864	134,217,728	201,326,592	268,435,456	335,544,320	402,653,184	469,762,048	536,870,912

Sampling & Reconstruction

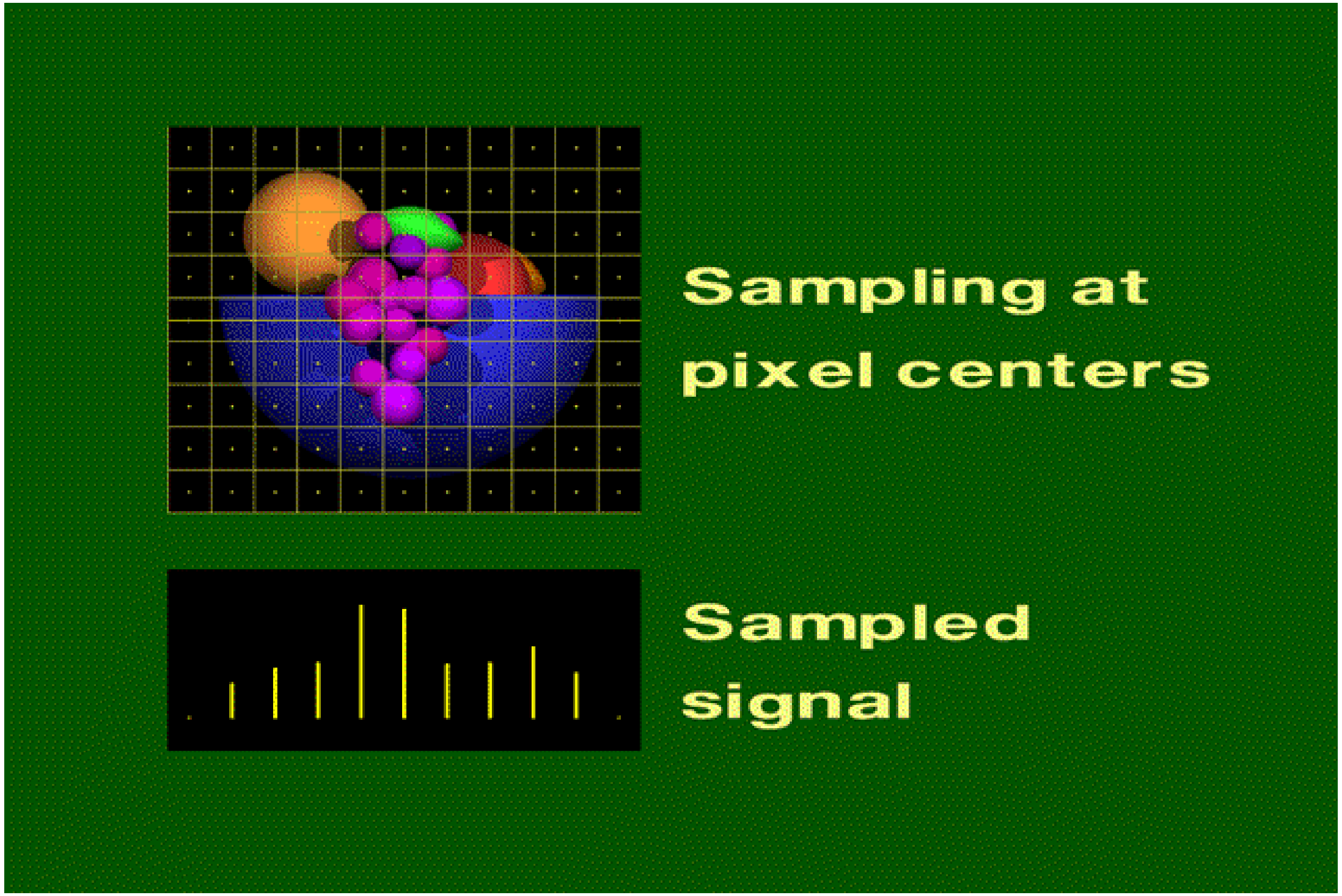
- Reconstruction
  - Given a set of digitized samples, how to approximate the original signal?



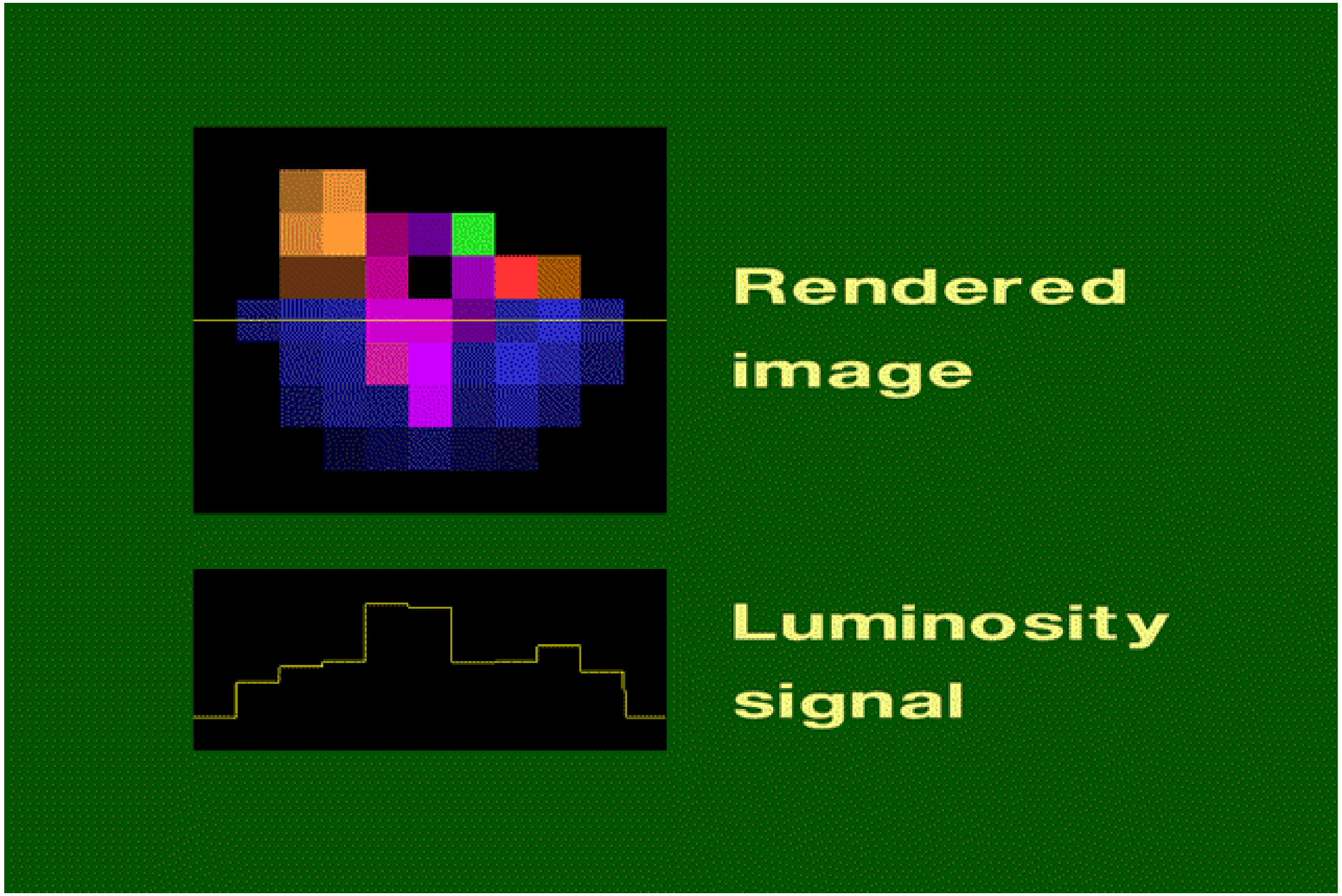
Continuous Luminosity signal



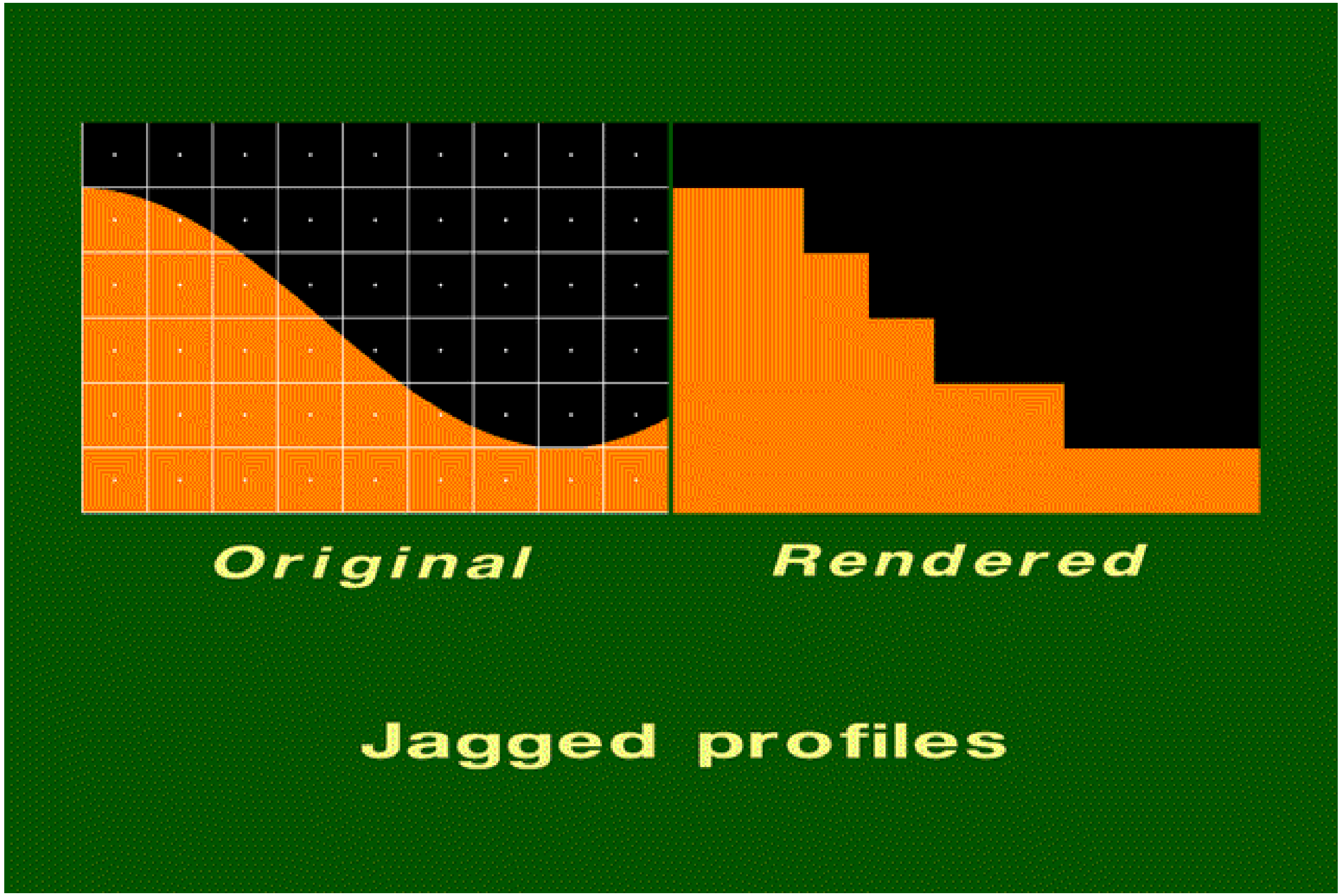
Sampled Luminosity



Reconstructed luminosity



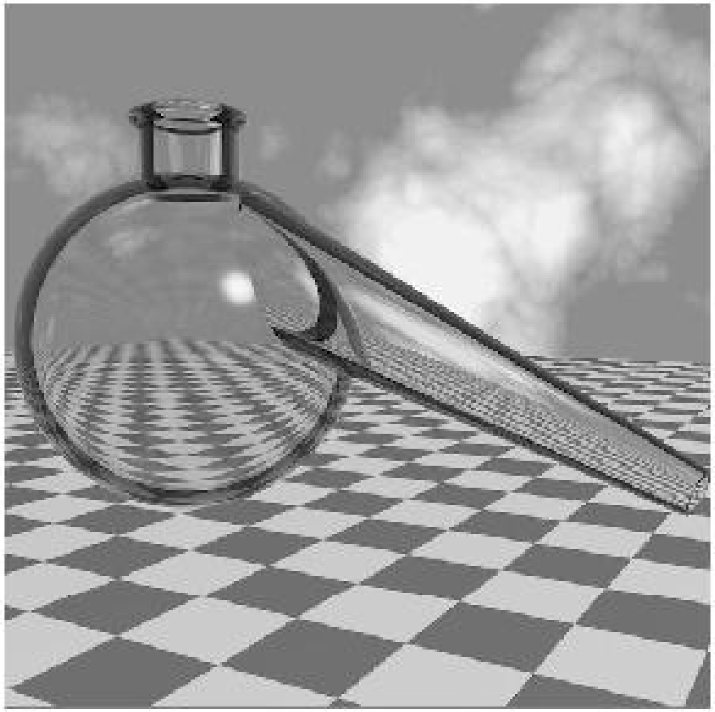
Reconstruction artefact

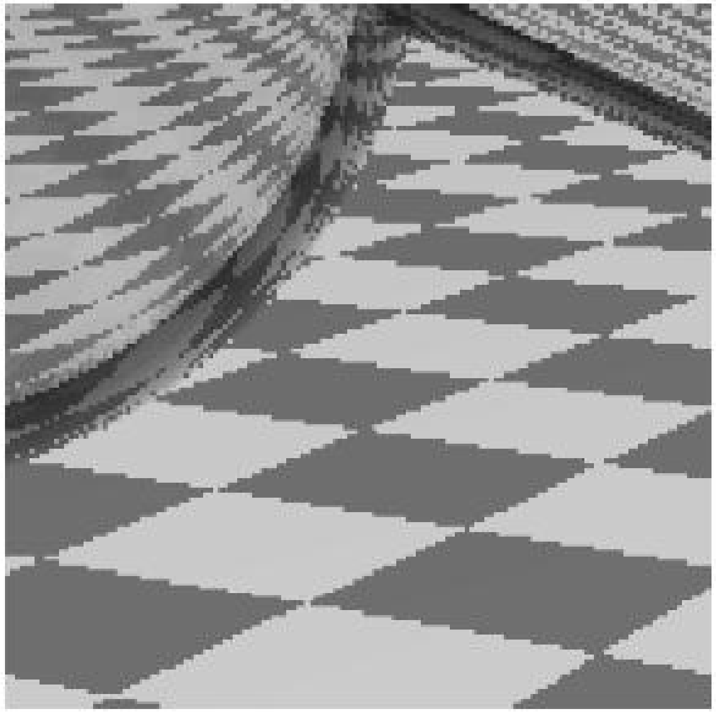


Slide © Rosalee Nerheim-Wolfe

Staircasing or *Jaggies*

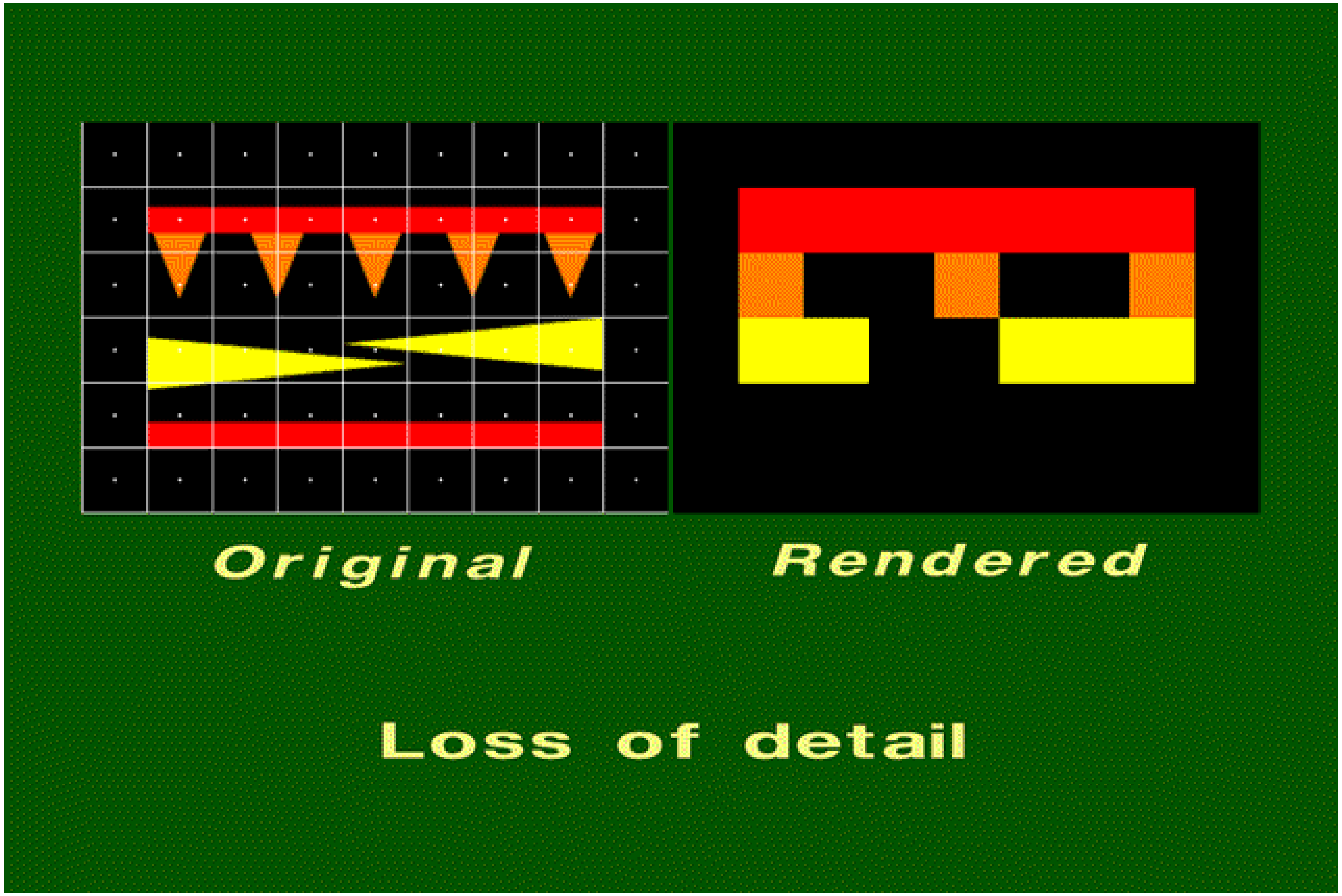
The raster *aliasing* effect – removal is called *antialiasing*





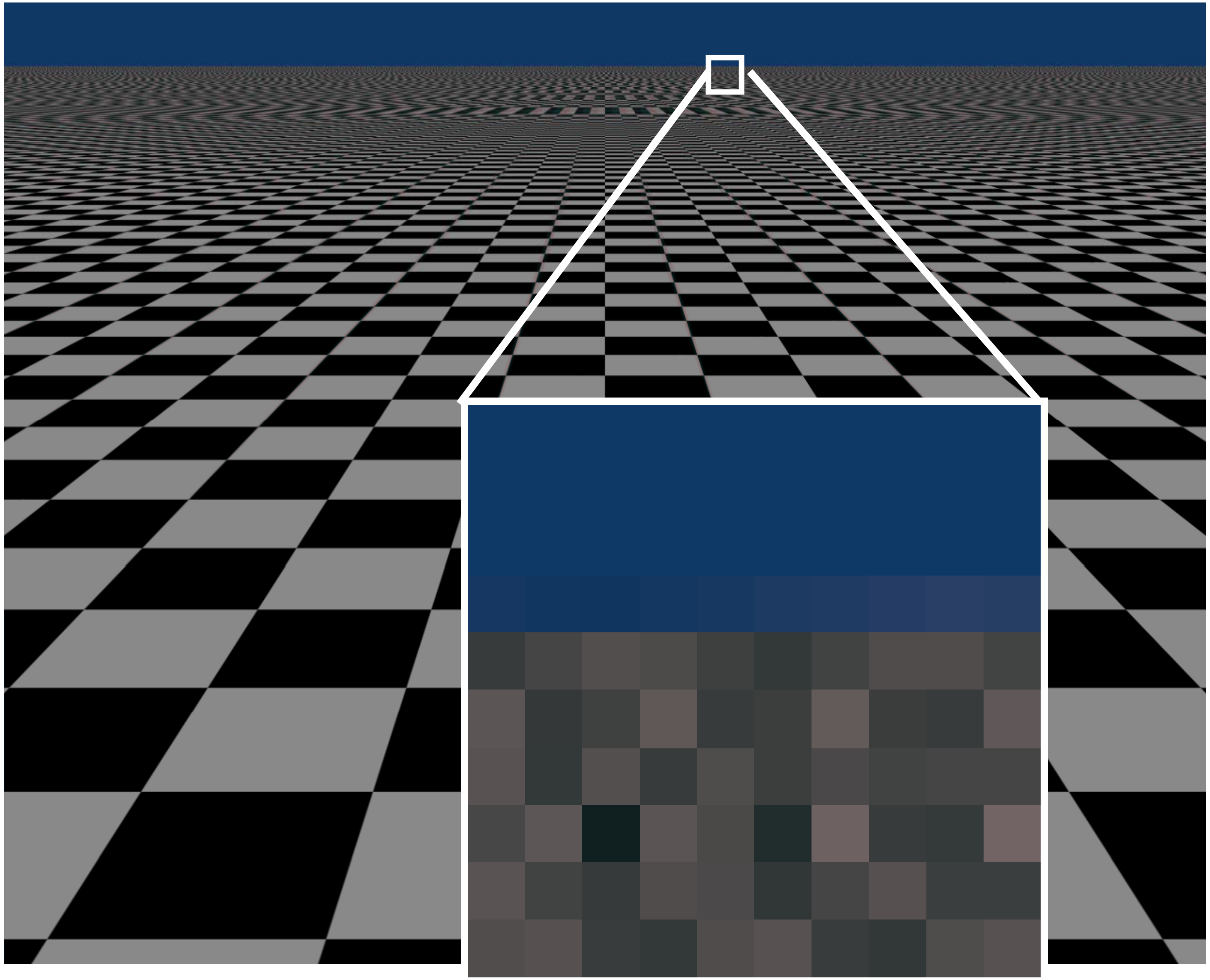
Images by Don Mitchell

Can be a serious problem…

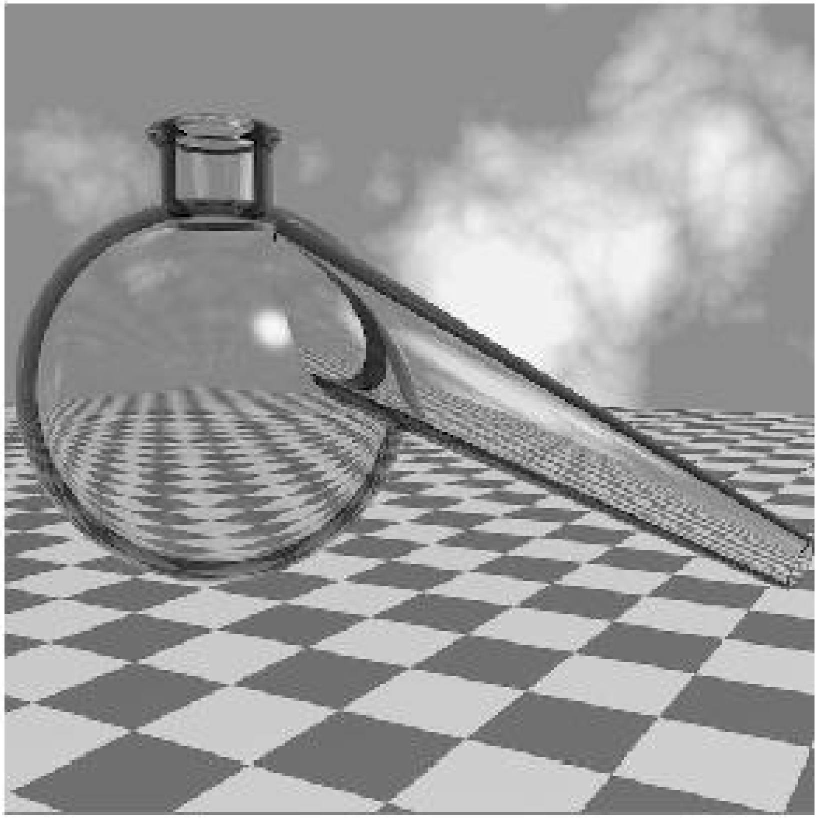
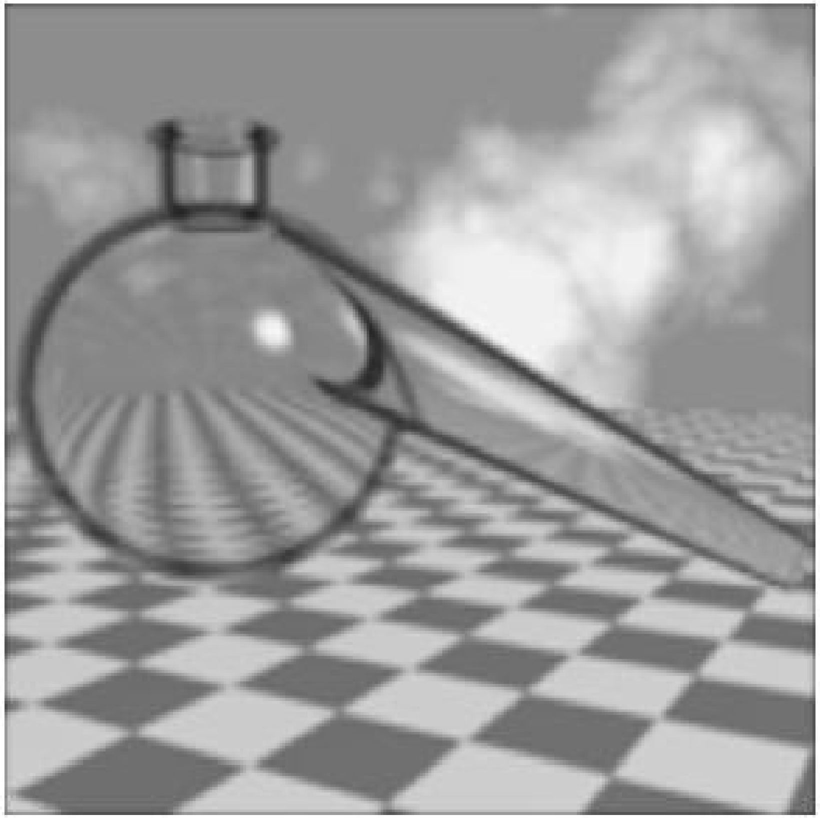


Slide © Rosalee Nerheim-Wolfe

Artifacts
- *Disintegrating textures*

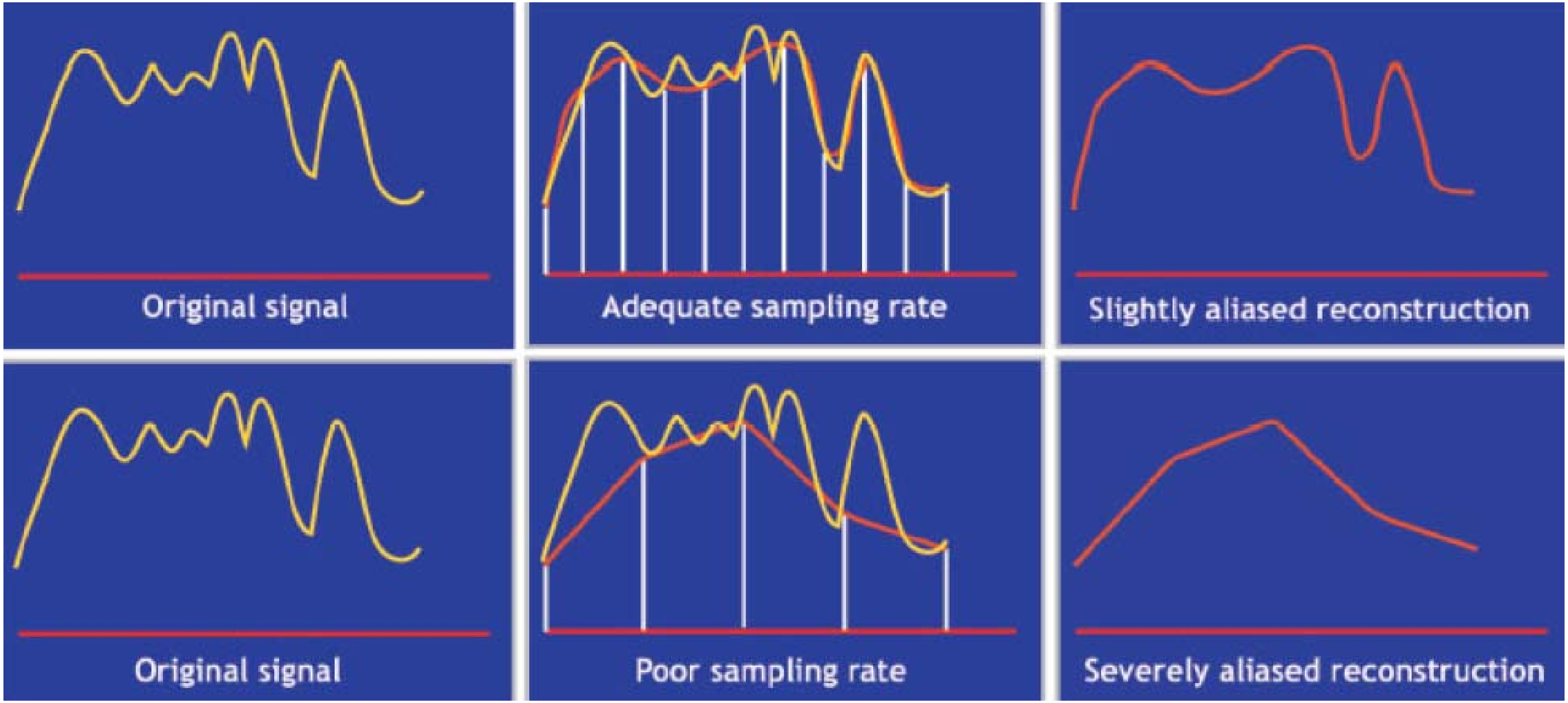


Blurring does not work well.

Removed the *jaggies*, but also all the detail !
→ Reduction in resolution

Aliasing PROBLEM

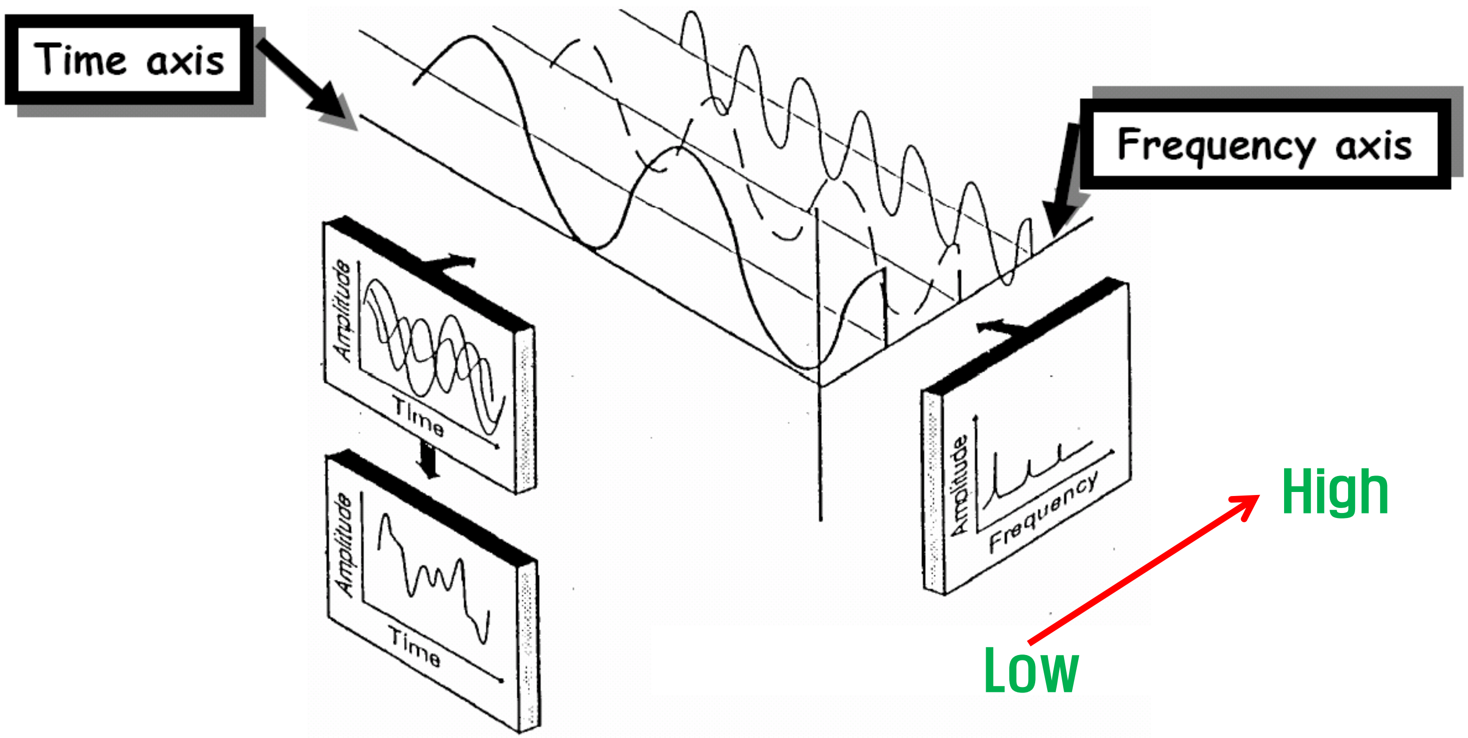


How is antialiasing done?

- We need some mathematical tools to
  - Analyse the sampling and reconstruction
  - Find an optimum solution
- Process of sampling and reconstruction is best understood in frequency domain
  - Use *Fourier transform* to switch between time and frequency domains
  - Function in time domain: ***signal***
  - Function in frequency domain: ***spectrum***

Time and Frequency

- Two independent windows to see one signal



- **Frequency** is measured in hertz (Hz) (the number of cycles of change per second).
- A given **bandwidth** is the difference in hertz between the highest and the lowest frequency.

Time and Frequency

- Any analog signal consists of components at various frequencies.
  - The simplest case is the sine wave, in which all the signal energy is concentrated at one frequency.
  - Analog signals usually have complex waveforms, with components at many frequencies.
  - All non-periodic signals can be represented as a summation of sin's and cos's of all freuencies.

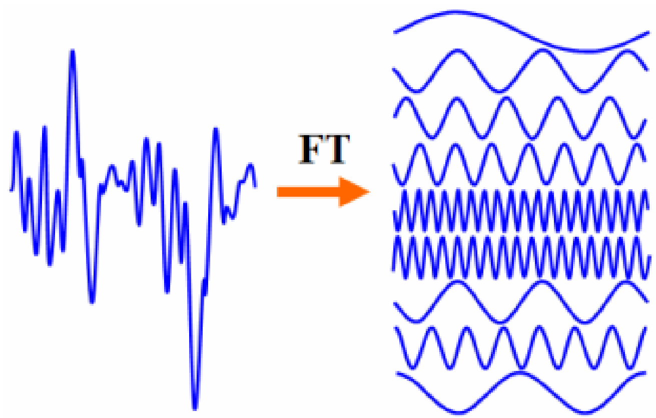
$$f(x) = \frac{1}{2\pi}\int_{-\infty}^{\infty} F(\omega)e^{i\omega x}d\omega$$

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

$$F(\omega) = \int_{-\infty}^{\infty} f(x)e^{-i\omega x}dx$$

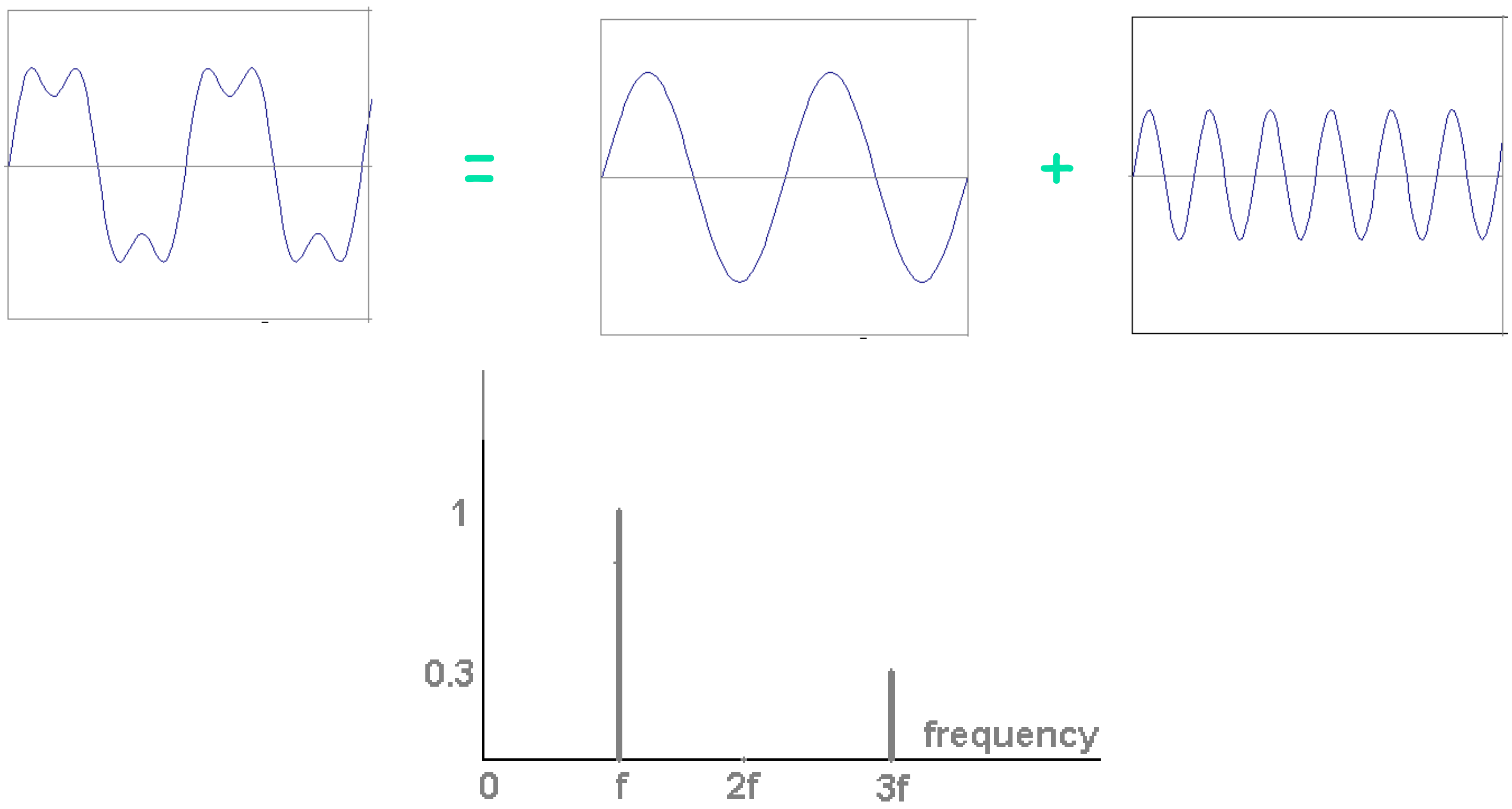
Time and Frequency

- Transform: rule that tells how to obtain a function F(*f*) from another function *f*(*t*)
  - Reveal important properties of *f*
  - More compact representation of *f*
  - *Fourier transform*, *DCT*, *wavelet*



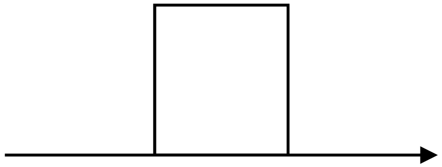
Time and Frequency

- example : $f(t) = \sin(2\pi f t) + \frac{1}{3}\sin(2\pi(3f)t)$

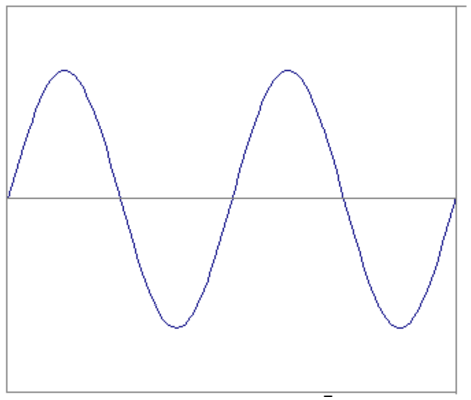


Time and Frequency

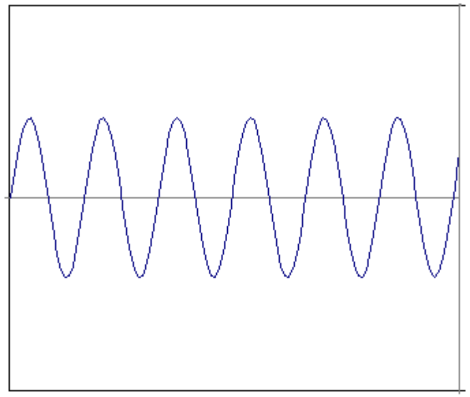
- example : $f(t) = \begin{cases} 1, & -a/2 < t < a/2 \\ 0, & \text{elsewhere} \end{cases}$



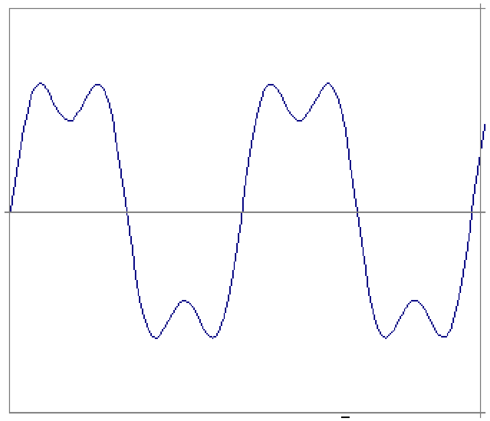
=



+

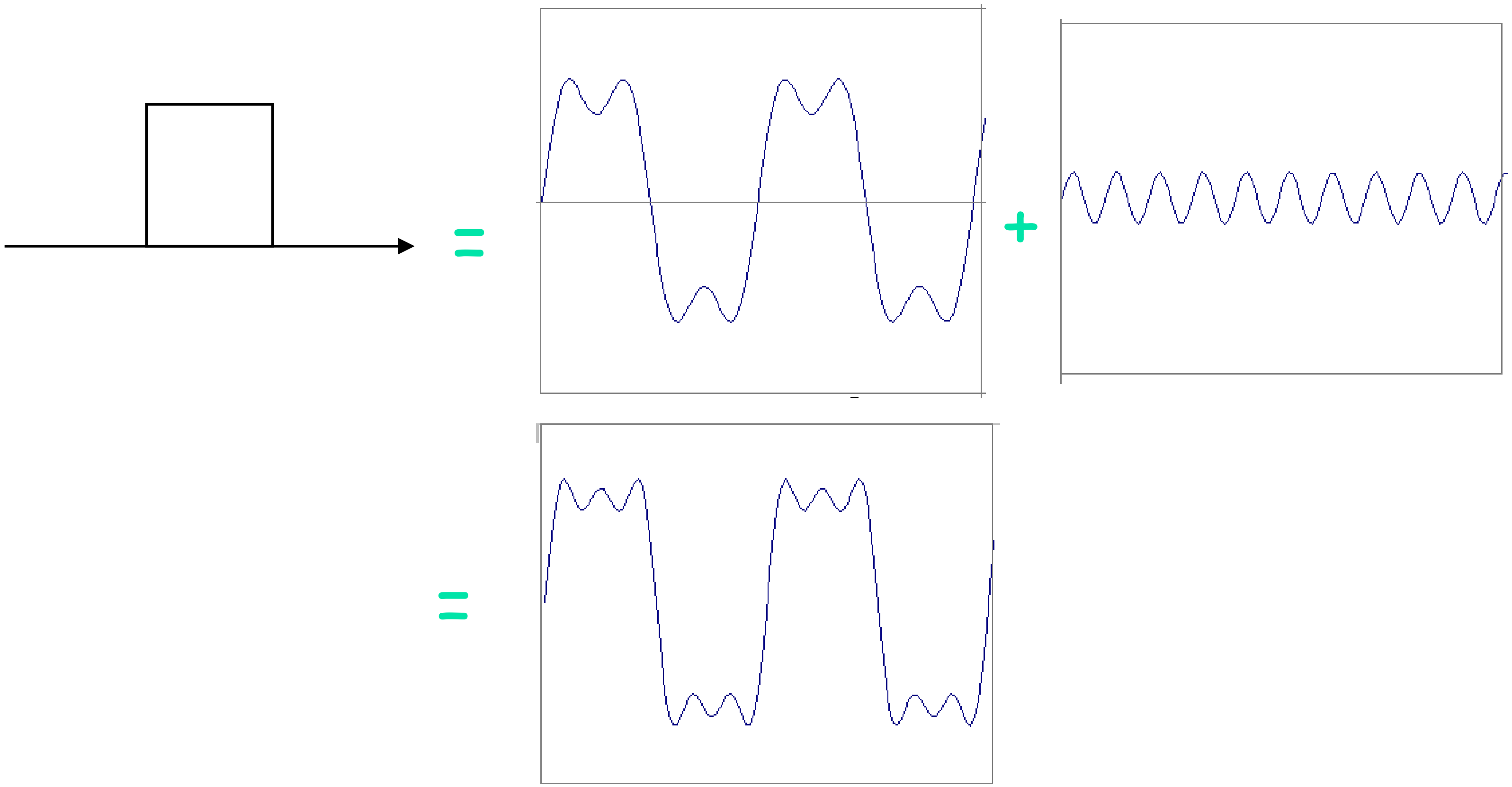


=



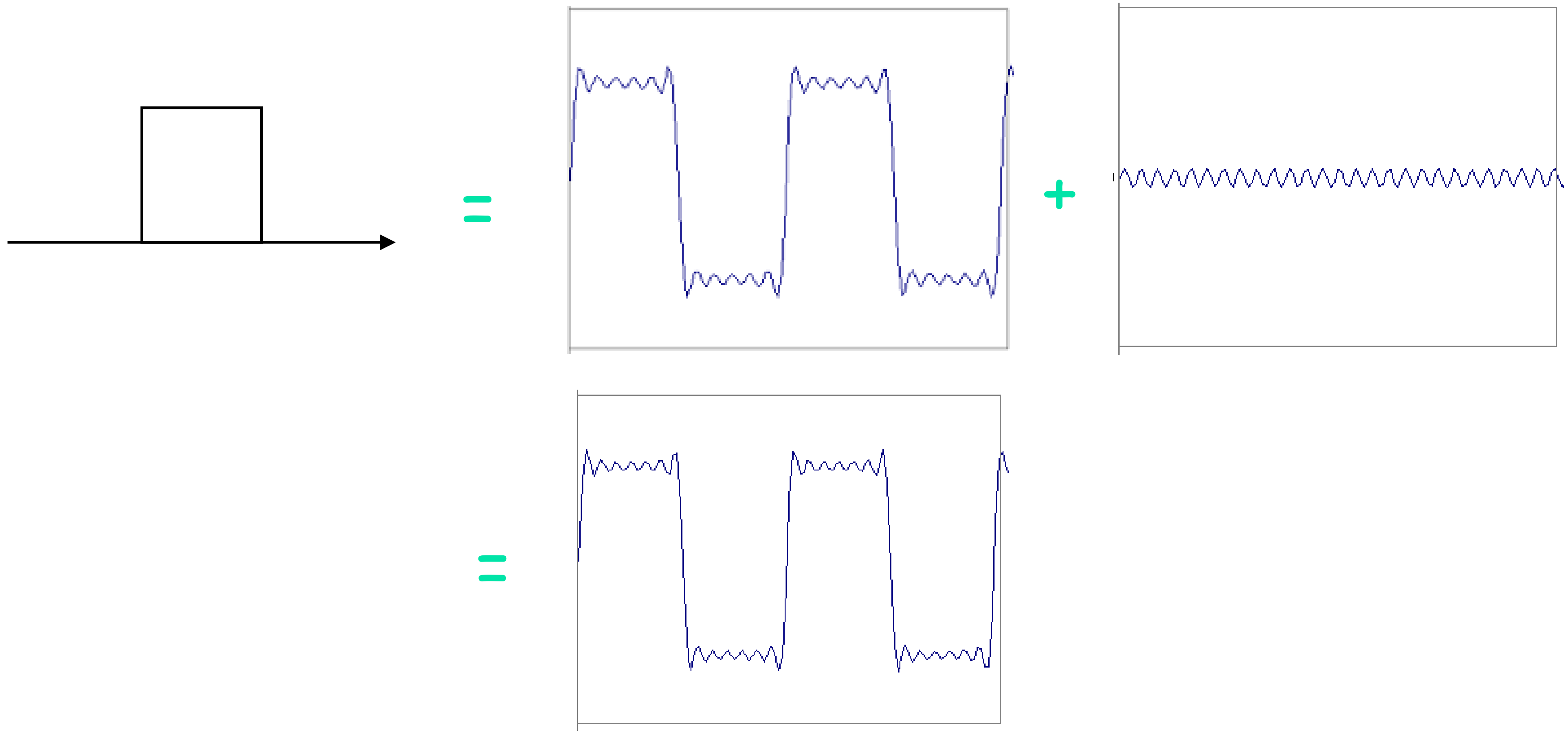
Time and Frequency

- example : $f(t) = \begin{cases} 1, & -a/2 < t < a/2 \\ 0, & \text{elsewhere} \end{cases}$



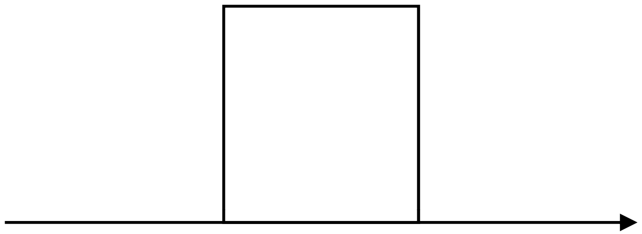
Time and Frequency

- example : $f(t) = \begin{cases} 1, & -a/2 < t < a/2 \\ 0, & \text{elsewhere} \end{cases}$



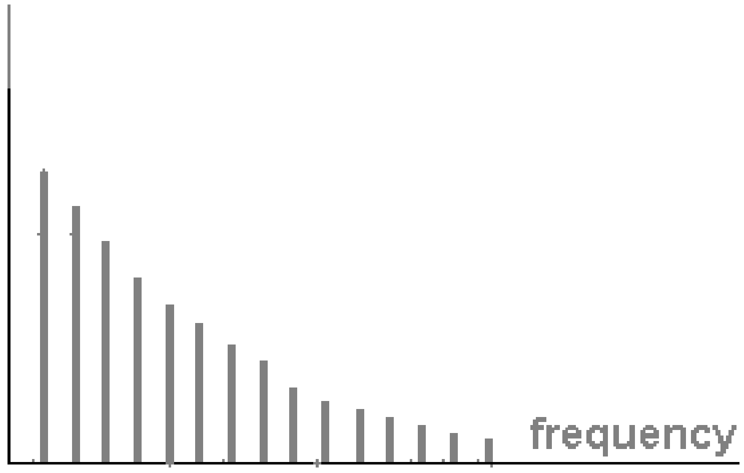
Time and Frequency

- example : $f(t) = \begin{cases} 1, & -a/2 < t < a/2 \\ 0, & \text{elsewhere} \end{cases}$



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

=



Fourier Transform

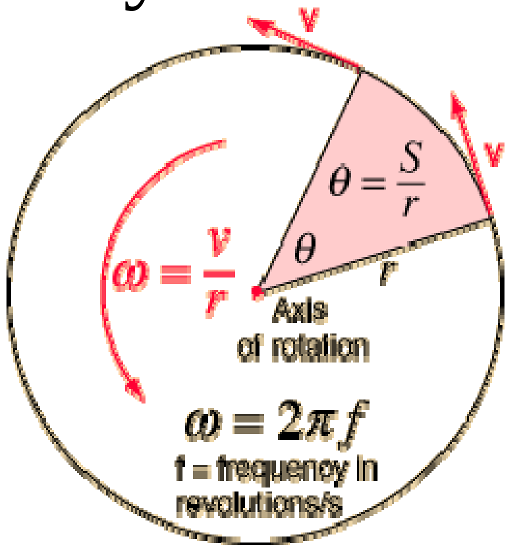
- Many functions *f:R → R* can be written as sums of sine (and cosine) waves that are integer multiple of fundamental (basis) frequencies

$$f(x) = \sum_{\omega} a_{\omega} \sin(\omega x + \theta_{\omega})$$

$\omega = 2\pi \cdot frequency$ is angular velocity

a_{ω} is amplitude

θ_{ω} is phase shift

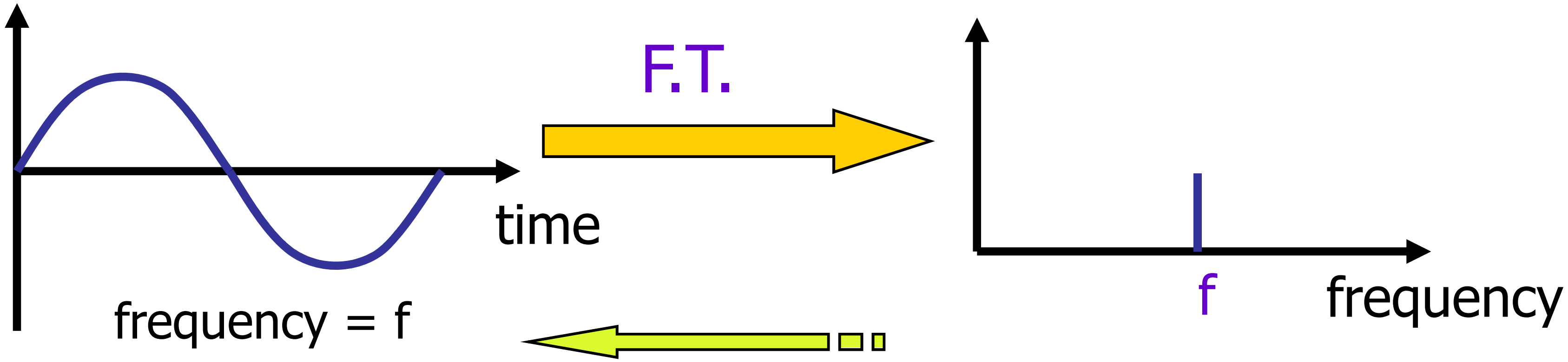


Fourier Transform

- Moving to complex numbers simplifies notation:

$$e^{-i\omega x} = \cos \omega x - i \sin \omega x$$

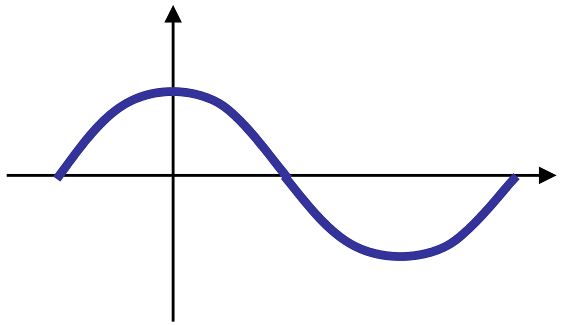
$$\boxed{F(\omega) = \int_{-\infty}^{\infty} f(x) e^{-i\omega x} dx}$$

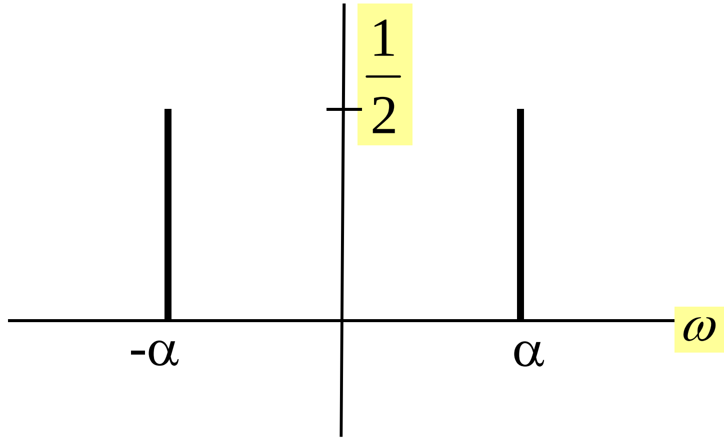


$$\boxed{f(x) = \frac{1}{2\pi} \int_{-\infty}^{\infty} F(\omega) e^{i\omega x} d\omega}$$

inverse Fourier transform

The Fourier Transform of a Cosine





Example: $$\cos(y) = \frac{1}{2}[\exp(iy) + \exp(-iy)]$$

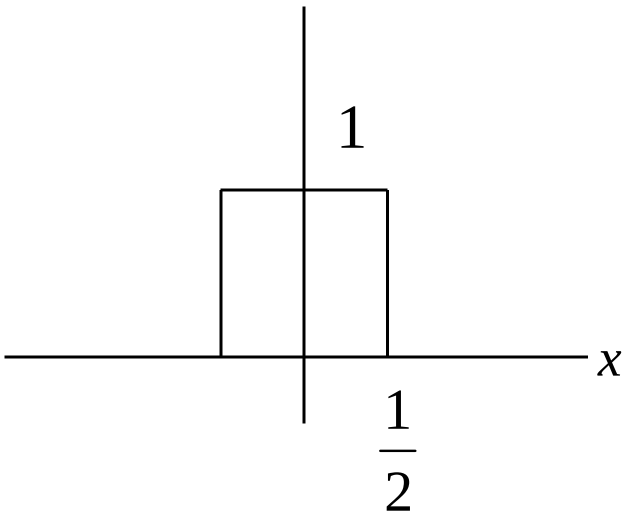
$$F[\cos(2\pi\alpha x)](\omega) = \tfrac{1}{2}\int_0^1 \exp(-2\pi i\omega x)[\exp(2\pi i\alpha x) + \exp(-2\pi i\alpha x)]dx$$

$$= \tfrac{1}{2}\int_0^1 [\exp(2\pi i(\alpha-\omega)x) + \exp(-2\pi i x(\alpha+\omega)x)]dx \qquad = \begin{cases} \frac{1}{2} & \omega = \pm\alpha \\ 0 & otherwise \end{cases}$$

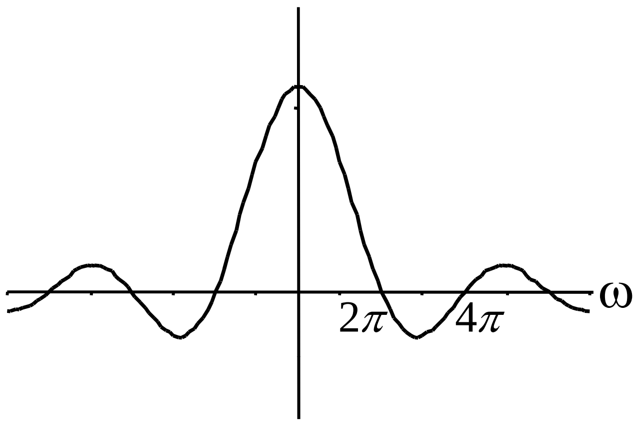
The Fourier Transform of a Box Function

$$\int_{-\infty}^{\infty} square(x) e^{-i\omega x} dx = \int_{-1/2}^{1/2} e^{-i\omega x} dx = \left.\frac{e^{-i\omega x}}{-i\omega x}\right|_{-\frac{1}{2}}^{\frac{1}{2}}$$

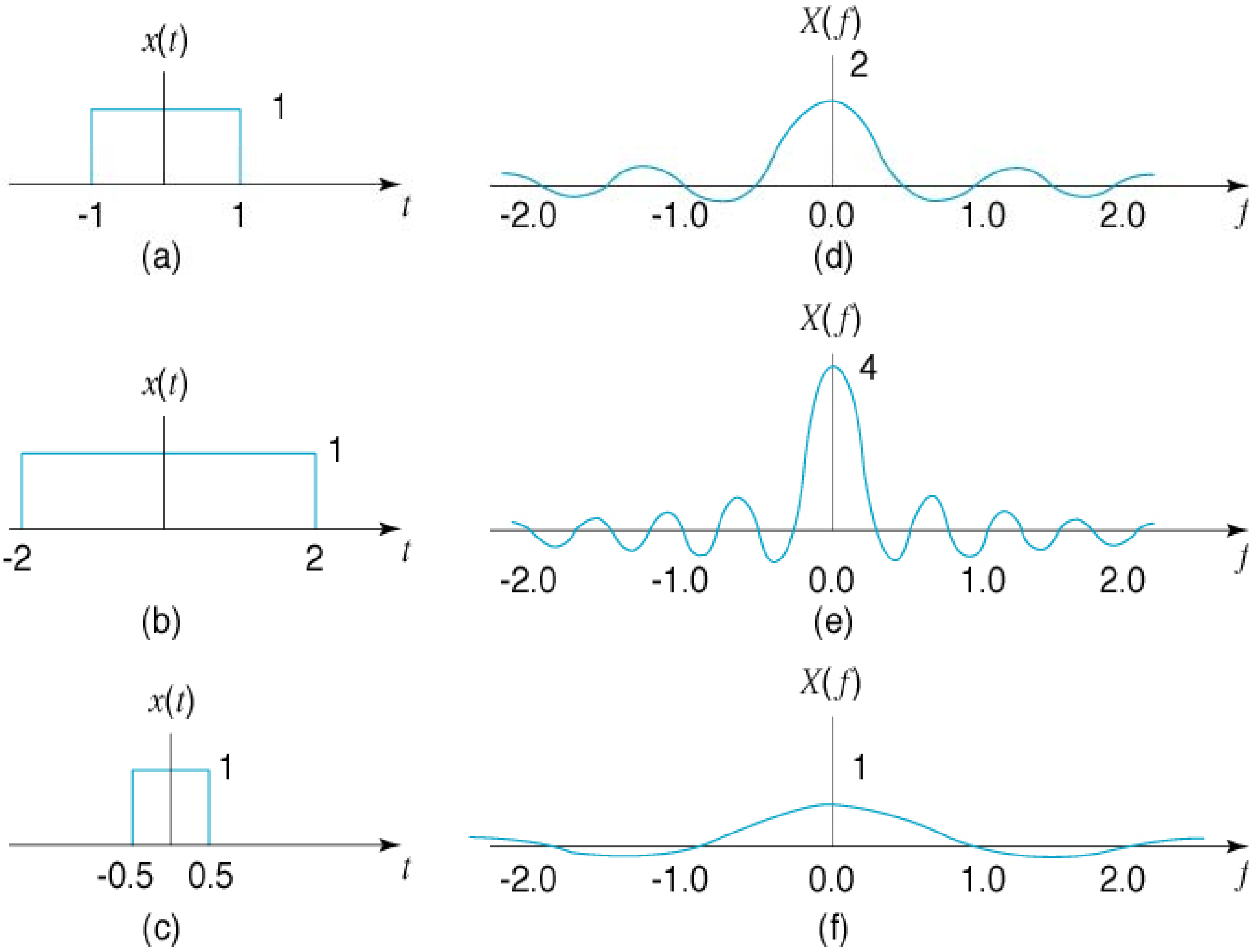
$$= \frac{e^{-\frac{1}{2}i\omega} - e^{\frac{1}{2}i\omega}}{\frac{1}{2}i\omega + \frac{1}{2}i\omega} = \frac{\sin\frac{1}{2}\omega}{\frac{1}{2}\omega} = \text{sinc f}$$



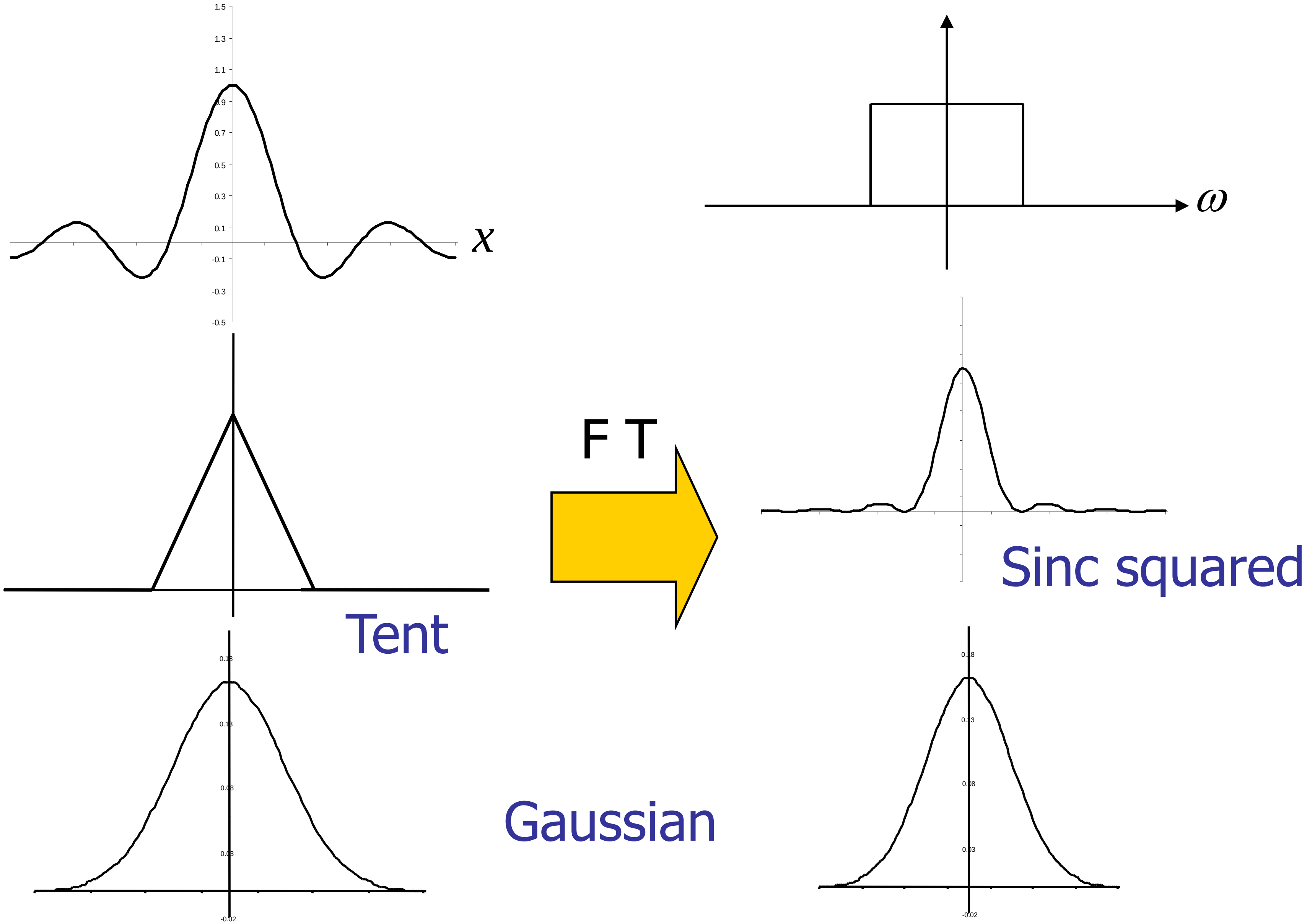
$\Leftrightarrow$



The Fourier Transform of a Box Function

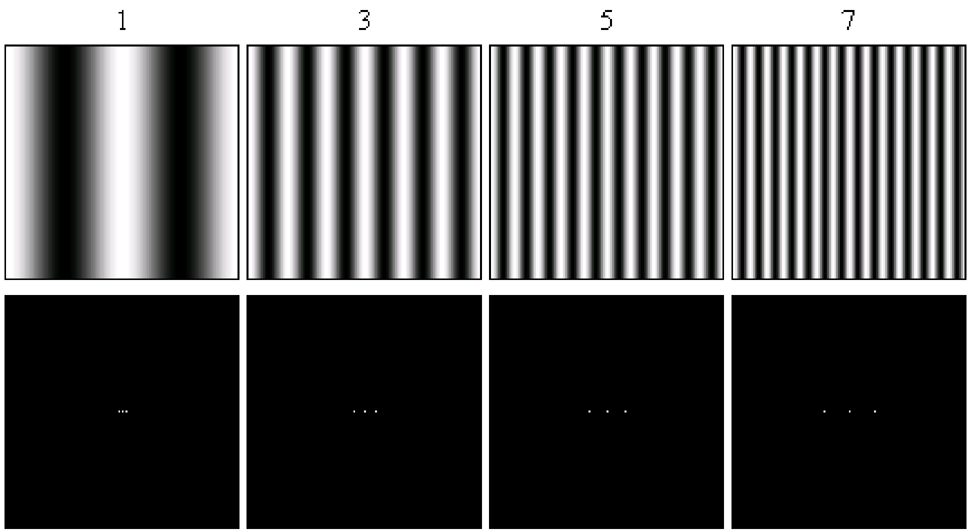


The Fourier Transform



2D Fourier Transform

- Images are 2D, discrete functions and FT will only contain discrete frequencies in quantized amounts
- Numerical algorithm: Fast Fourier Transform (FFT) computes discrete Fourier transforms

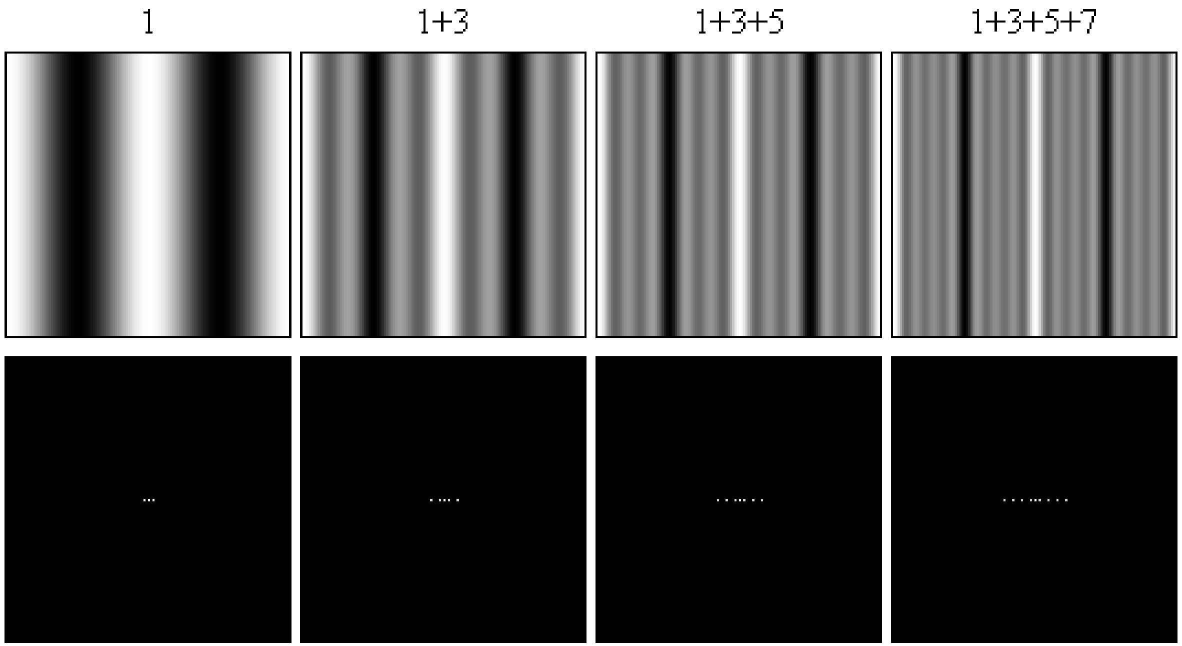


spatial domain

frequency domain

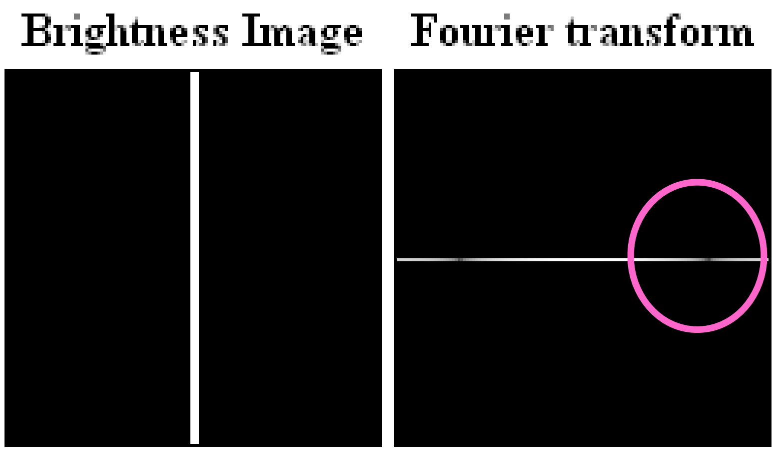
- Every pixel of the Fourier image is a spatial frequency value, the magnitude of that value is encoded by the brightness of the pixel.
- There is also a "DC term" corresponding to zero frequency, that represents the average brightness across the whole image

Fourier Transform



spatial domain

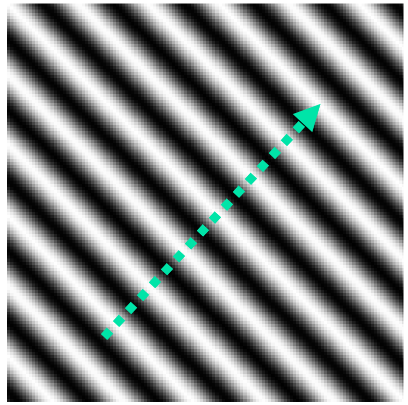
frequency domain



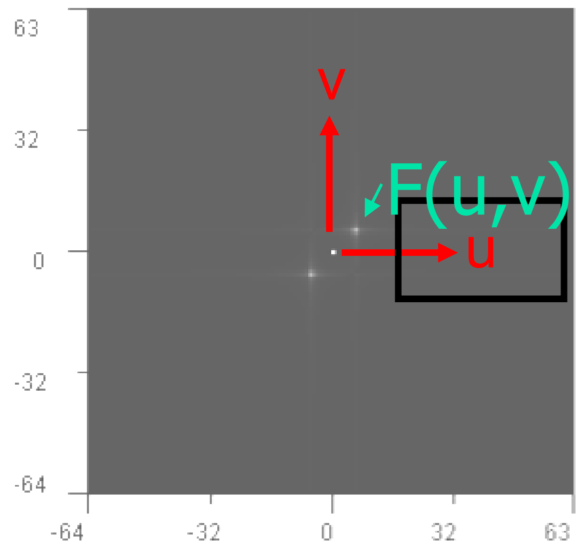
High frequency!!

2D Fourier Transform

- What F(u, v) means in spatial domain?
  - There is a signal S with $\sqrt{u^2+v^2}$ frequency
  - The orientation of S is $\tan^{-1}(v/u)$
  - The weight of S in the whole image is the value of F(u, v)

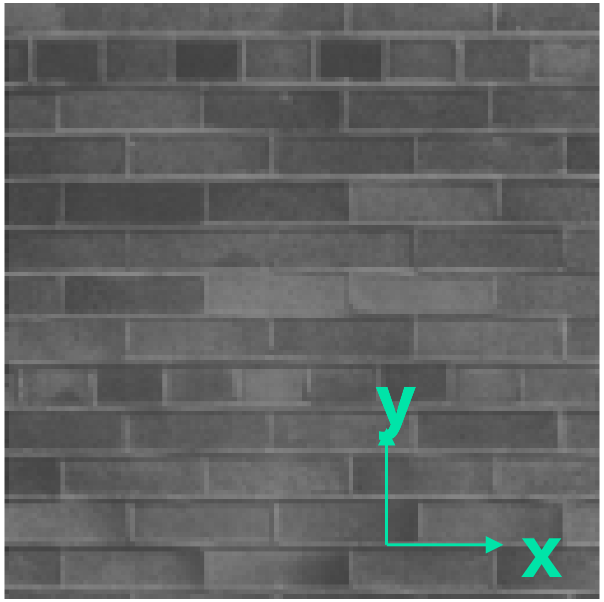


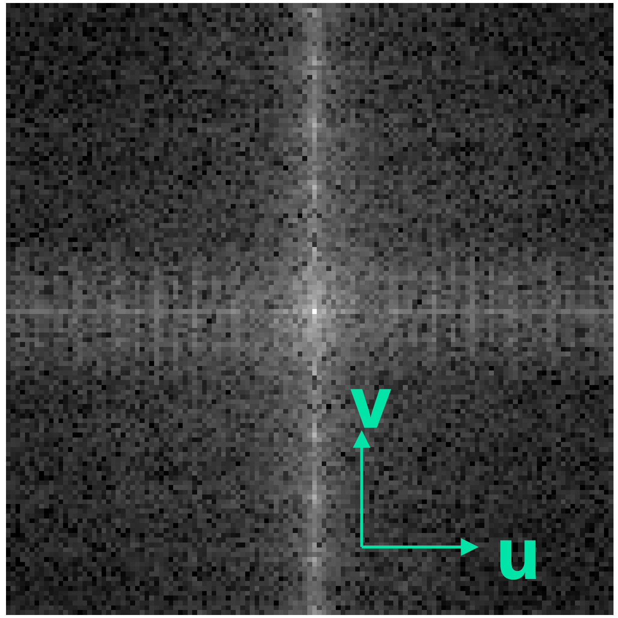
spatial domain

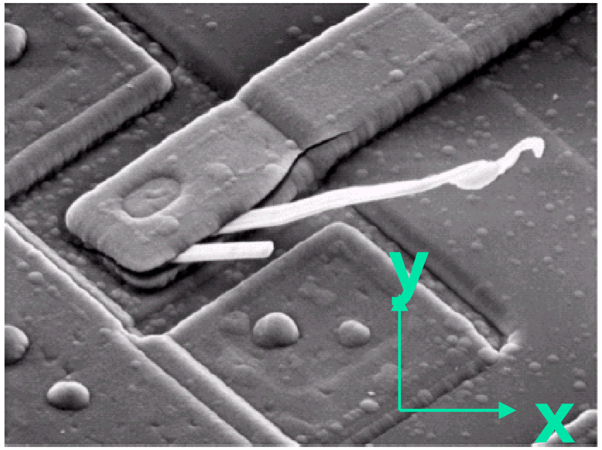


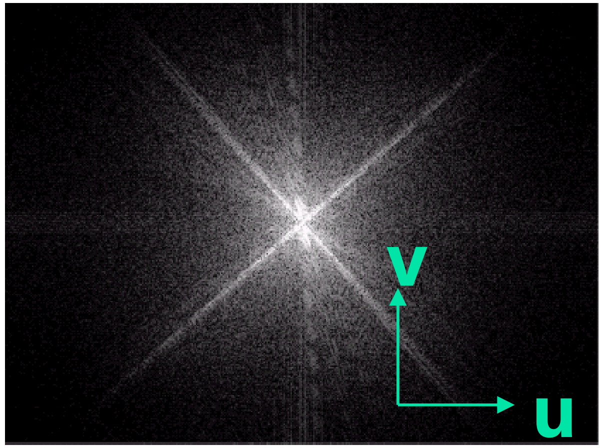
frequency domain

2D Fourier Transform
- Examples





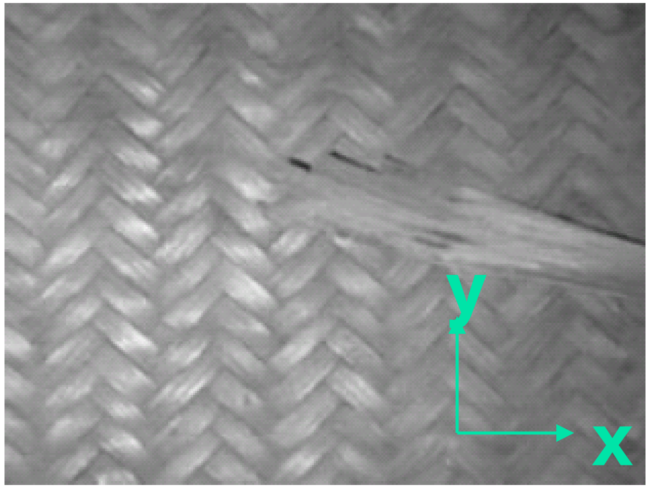


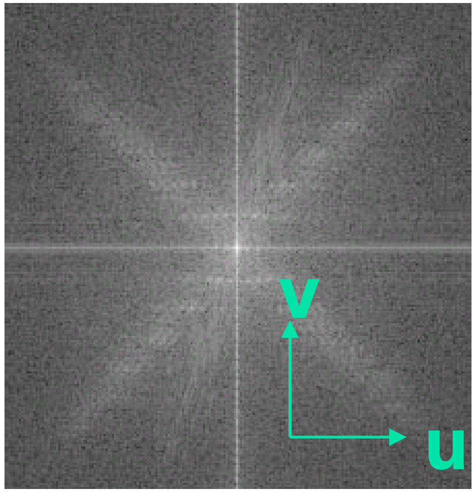


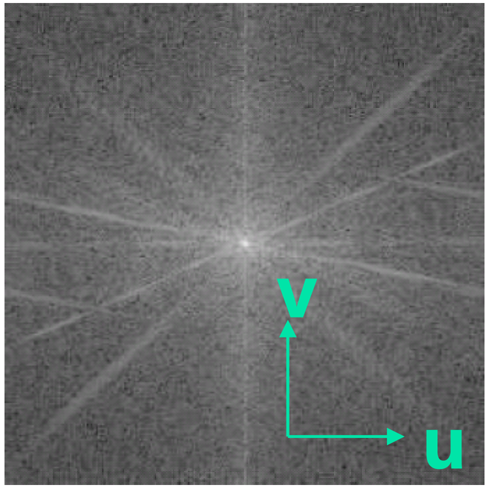
spatial domain

frequency domain

2D Fourier Transform
- Examples



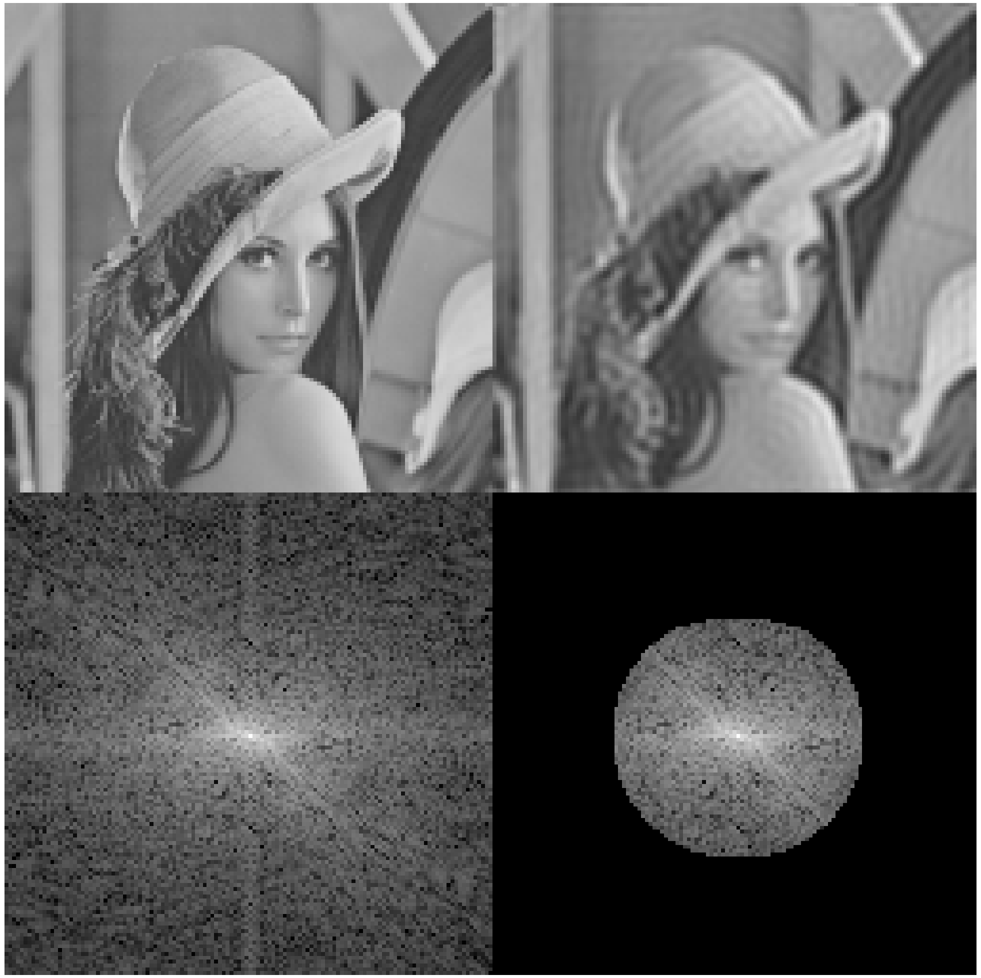




spatial domain

frequency domain

2D Fourier Filtering

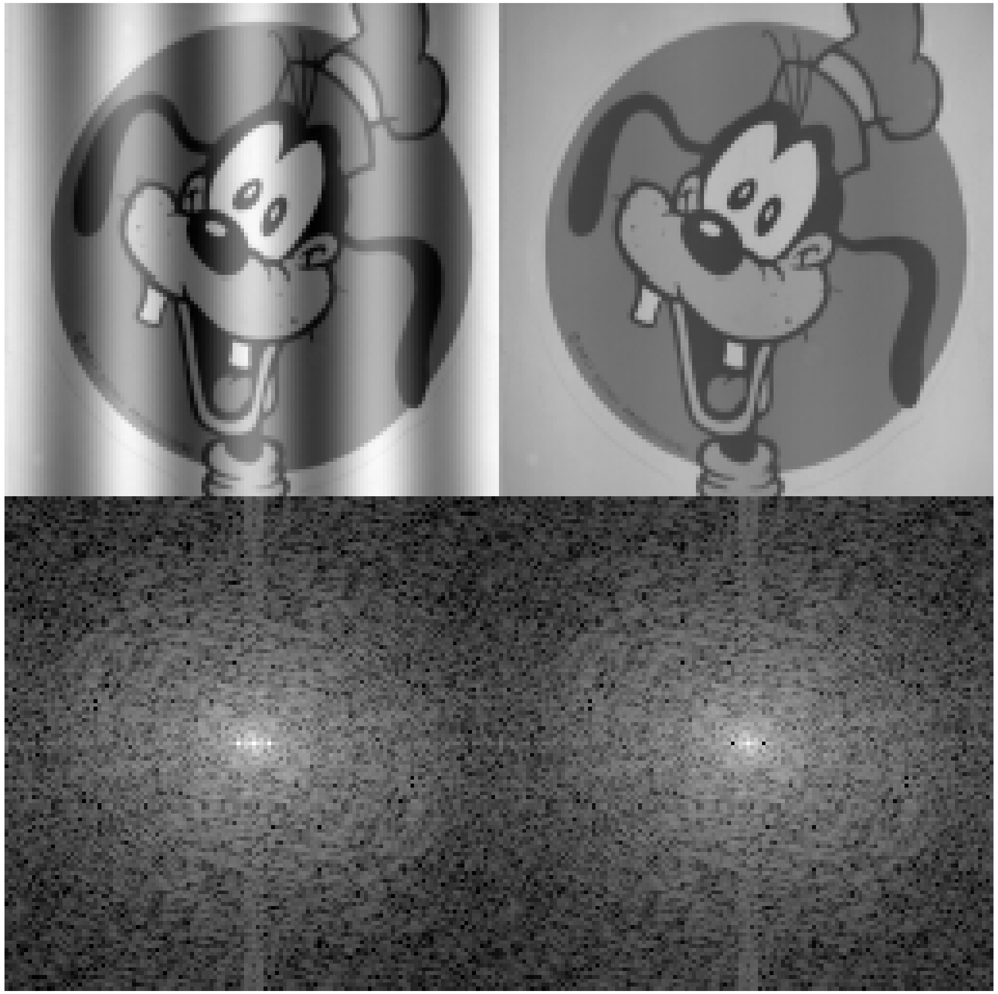


Low pass filter

High pass filter

2D Fourier Filtering

Image enhancement



Noise removal

Convolution

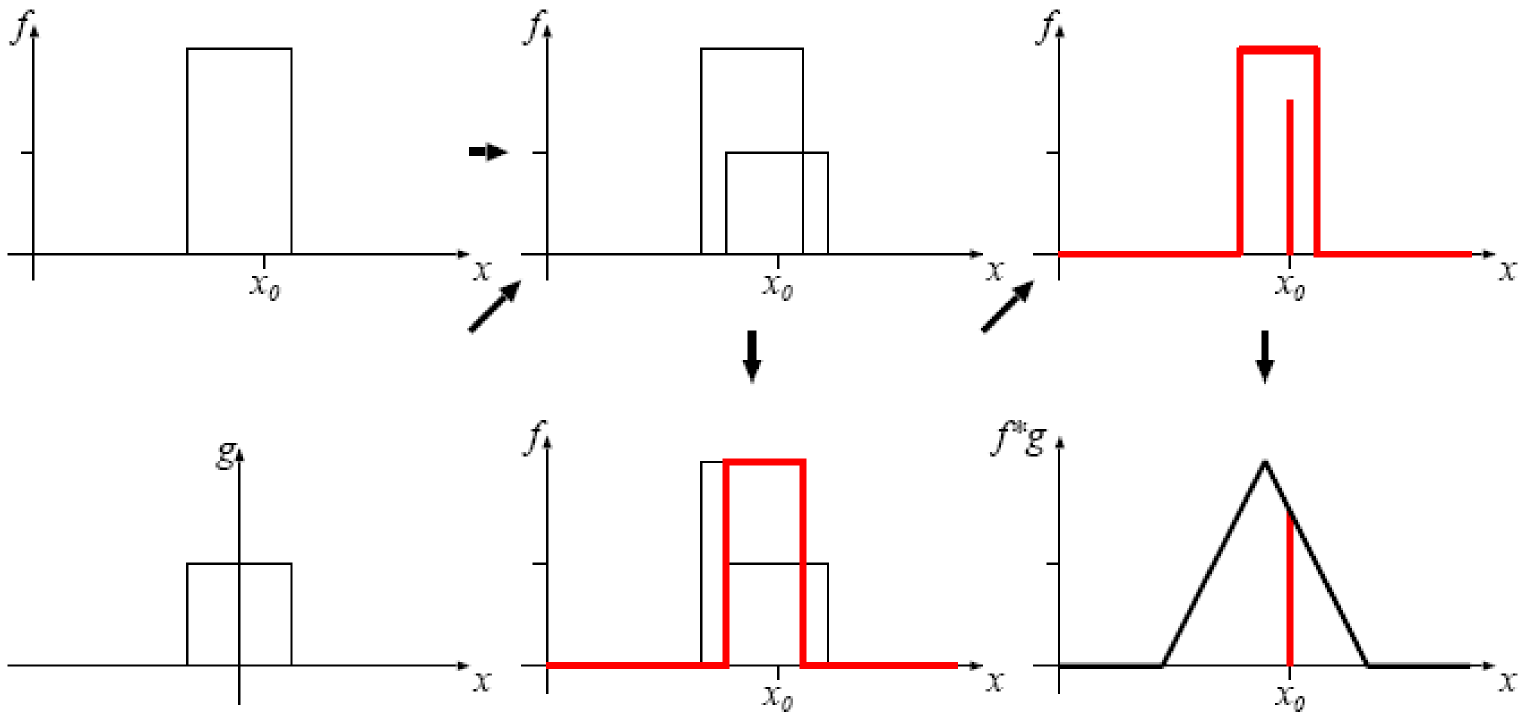
- One of the most common methods for filtering a function is called **convolution**.
- In 1D, convolution is defined as:

$$f(x) \otimes g(x) = \int_{-\infty}^{\infty} f(t) g(x-t) dt$$

- The convolution operator is a generalized formula to express weighted averaging of an input signal *f* and a weight function or filter kernel *g*
  - Qualitatively: Slide the filter to each position, *x*, then sum up the function multiplied by the filter at that position
- One important application of convolution is reconstructing sampled signals

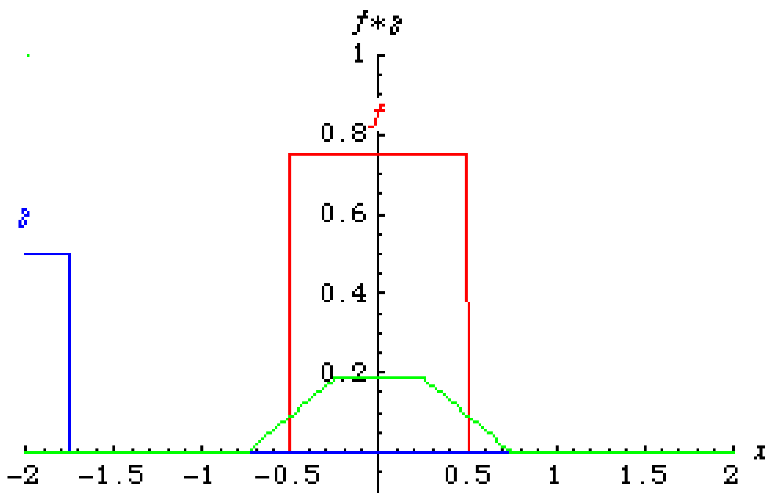
Convolution

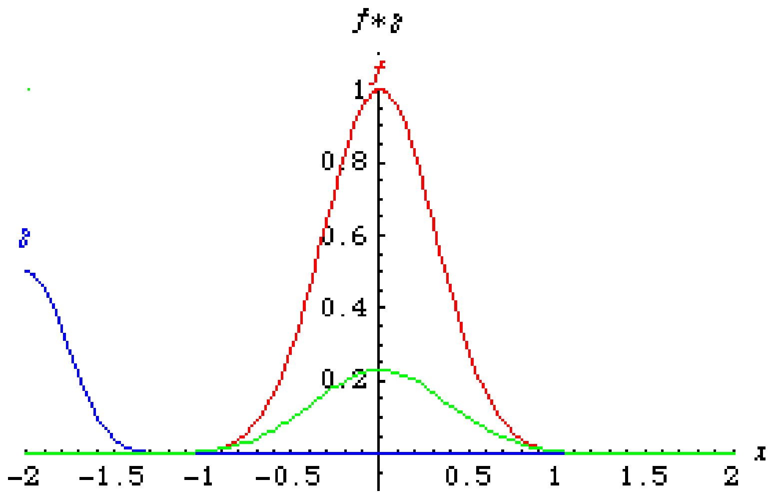
$$f(x_0) \otimes g(x_0) = \int_{-\infty}^{\infty} f(t)g(x_0 - t)dt$$



Convolution

- Green curve is the convolution of the Red curve, f(x), and the Blue curve, g(x).
- The grey region indicates the product f(t)g(t – x)





Convolution Theorem

- Convolution theorem: *Convolution* in the *spatial* domain is equivalent to *multiplication* in the *frequency* domain.

$$f \otimes g = F \cdot G$$

- Symmetric theorem: *Convolution* in the *frequency* domain is equivalent to *multiplication* in the *spatial* domain.

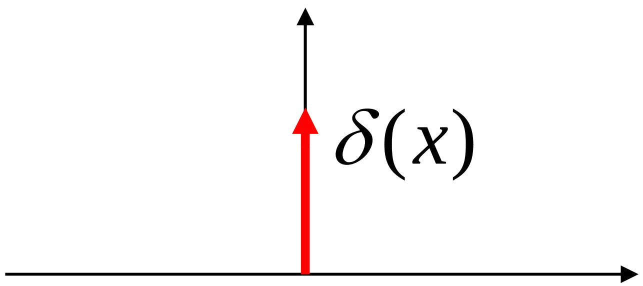
$$f \cdot g = F \otimes H$$

Delta Function

- The impulse (Dirac delta function), δ(*x*), is a handy tool for sampling theory.
- It has zero width, infinite height, and unit area.

$$\delta(x) = \begin{cases} 0 & x \neq 0 \\ a & x = 0 \end{cases}$$

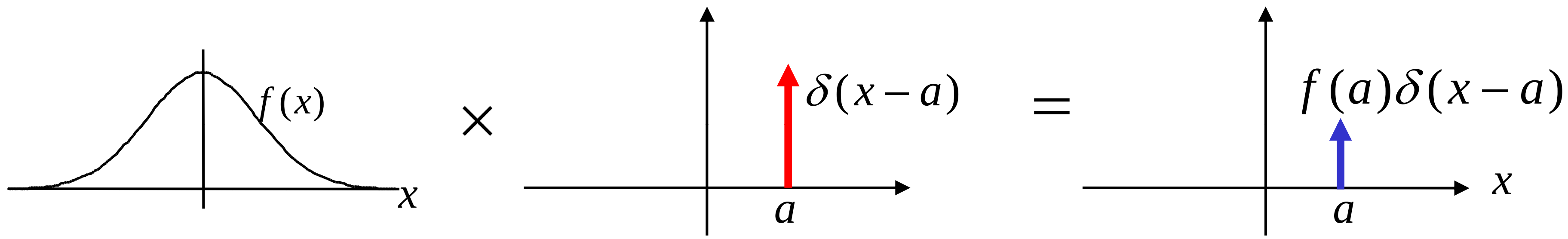
$$\int_{-\infty}^{\infty} \delta(x)dx = 1$$



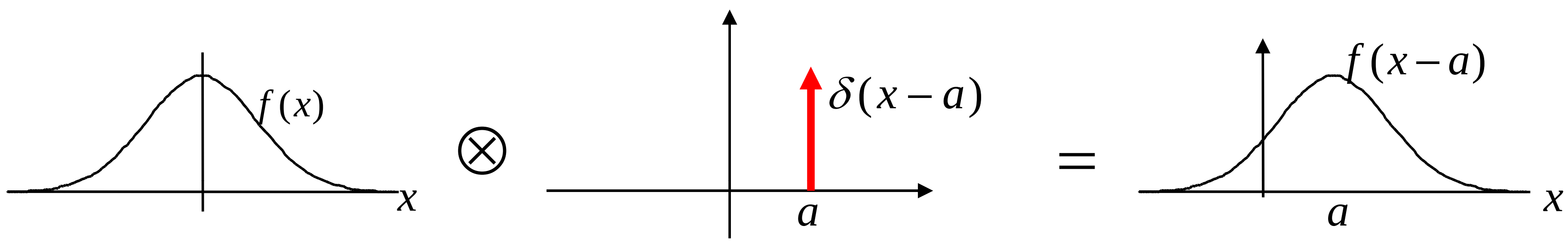
- Fourier transform of the delta function is a constant

Delta Function

- For sampling, the delta function has two important properties.
- **Sifting:** $f(x)\delta(x-a) = f(a)\delta(x-a)$



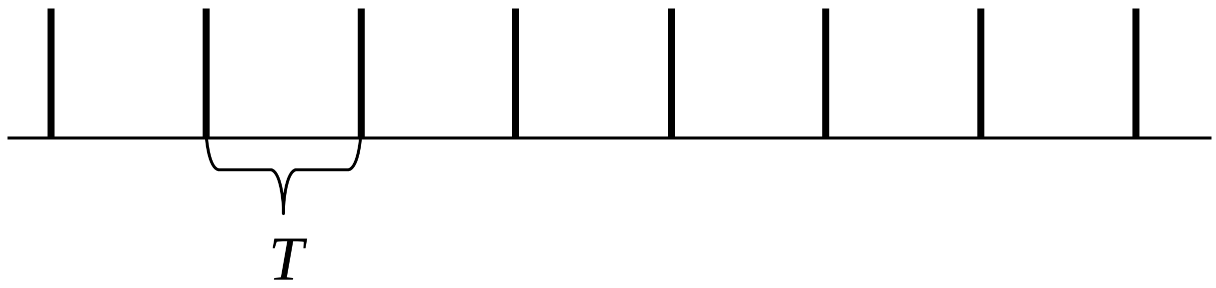
- **Shifting:** $f(x)\otimes\delta(x-a) = f(x-a)$



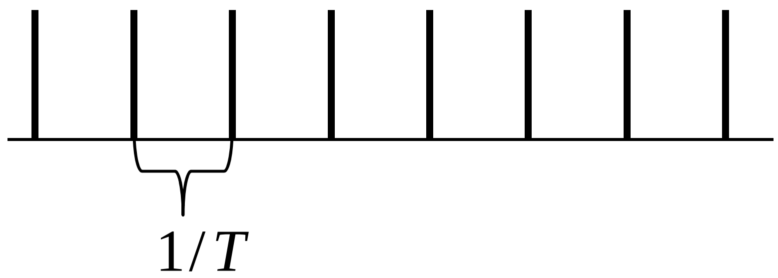
Comb Function

- Comb function is an infinite series of equidistant Dirac impulses

$$c_T(x) = \sum_{k=-\infty}^{\infty} \delta(x - kT)$$



- Fourier transform of the comb function takes the same form:



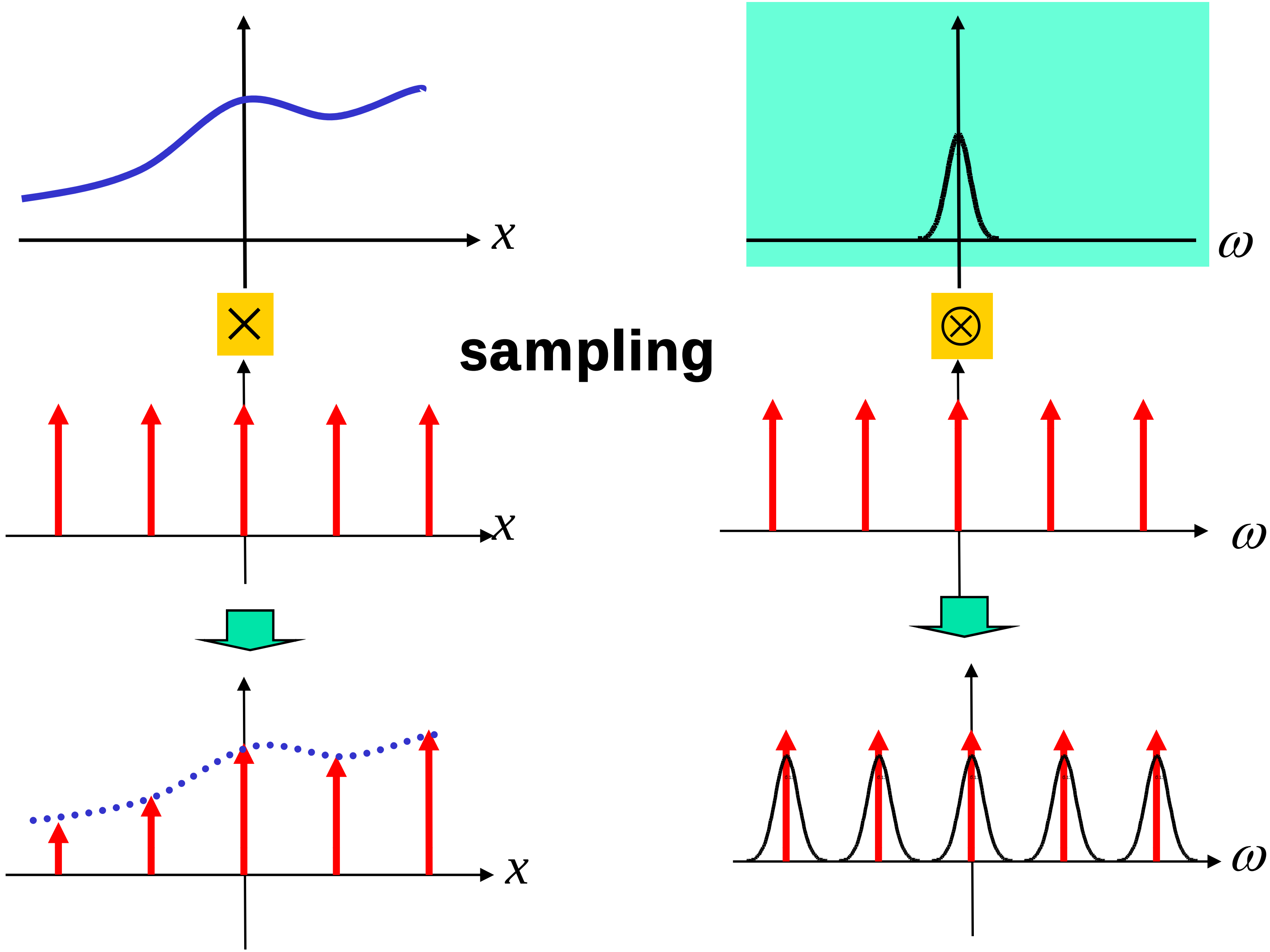
Sampling Function

- The Fourier transform of the sampling function(e.g., comb function) is itself a sampling function.
  - The sample spacing is the inverse.

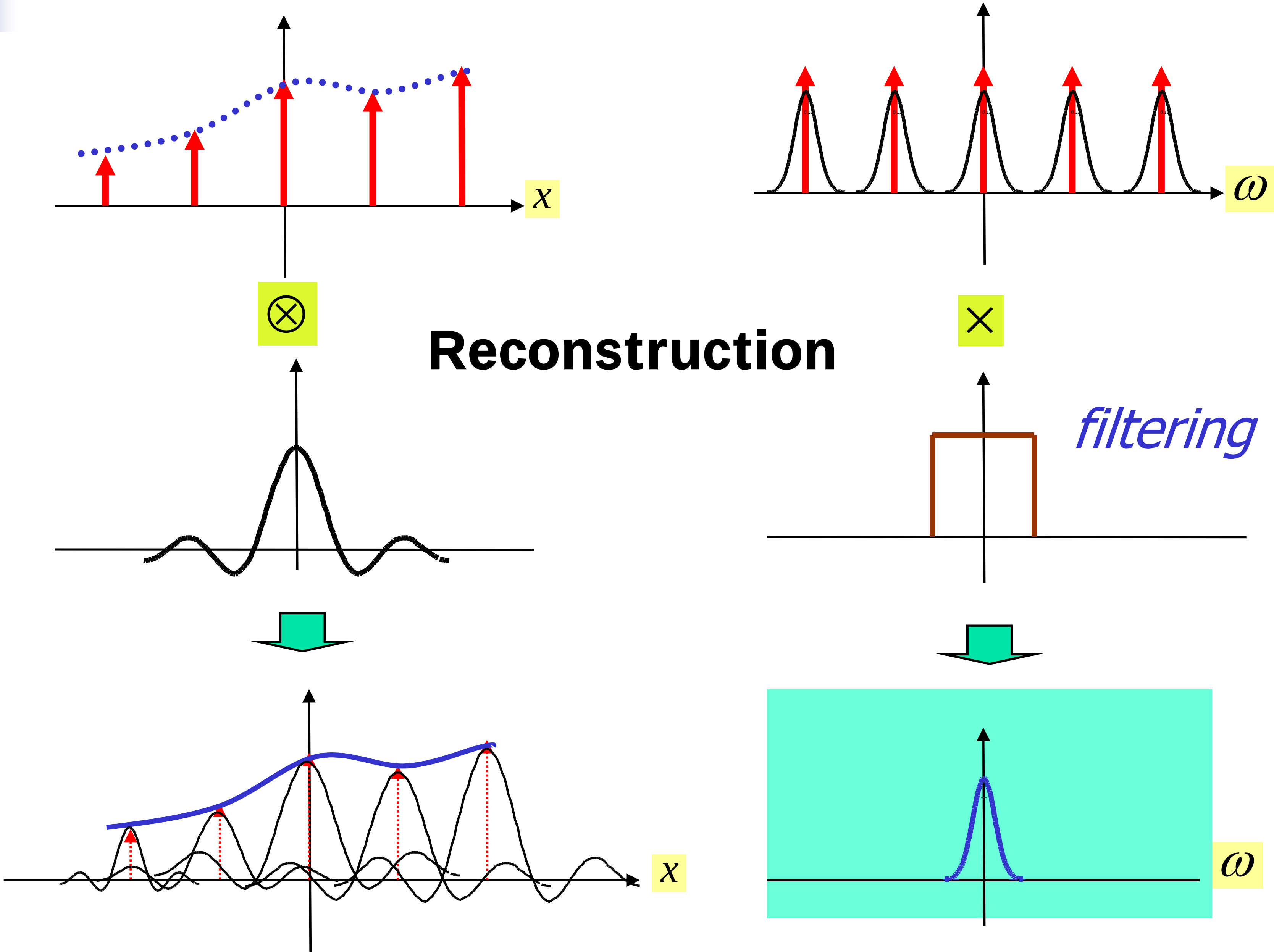
$$S_T(x) \Leftrightarrow S_{\frac{1}{T}}(\omega)$$

- Remember convolution in the spatial domain is the same as multiplication in the frequency domain

Sampling and Reconstruction

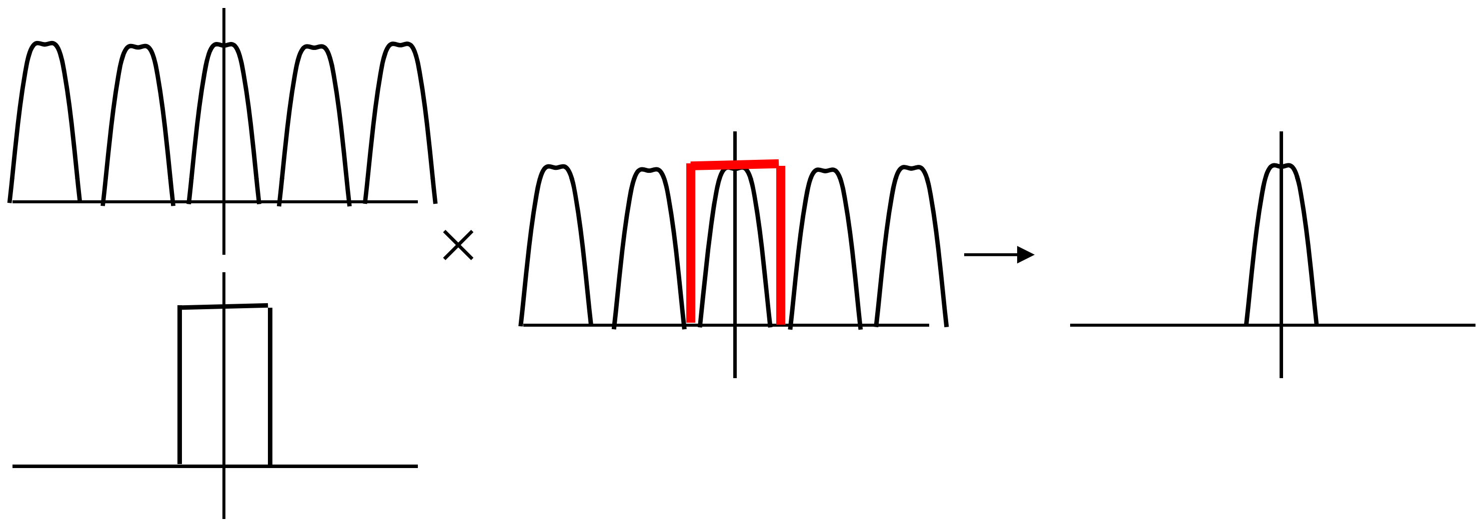


Sampling and Reconstruction



Reconstruction

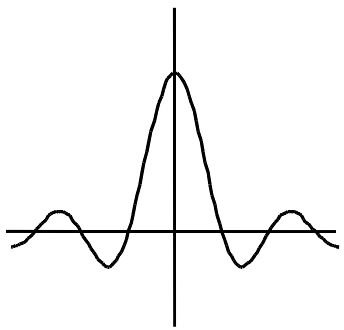
- To reconstruct, we must restore the original spectrum
- That can be done by multiplying by a filter like square pulse



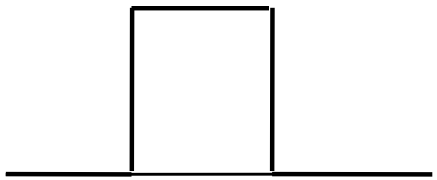
- Multiplying by a square pulse in the frequency domain is the same as convolving with a *sinc* function in the spatial domain

Sinc Filter

$$\operatorname{sinc}(x) = \frac{\sin(\pi x)}{\pi x}$$



Spatial: sinc

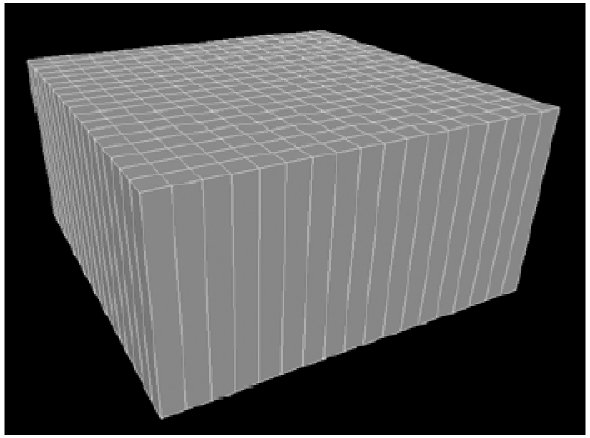


Frequency: box

- Perfect low-pass filter
- Cuts off all frequencies above a threshold
- Oscillates to infinity: need too many samples
- We use other functions similar to a sinc to filter

Box Filter

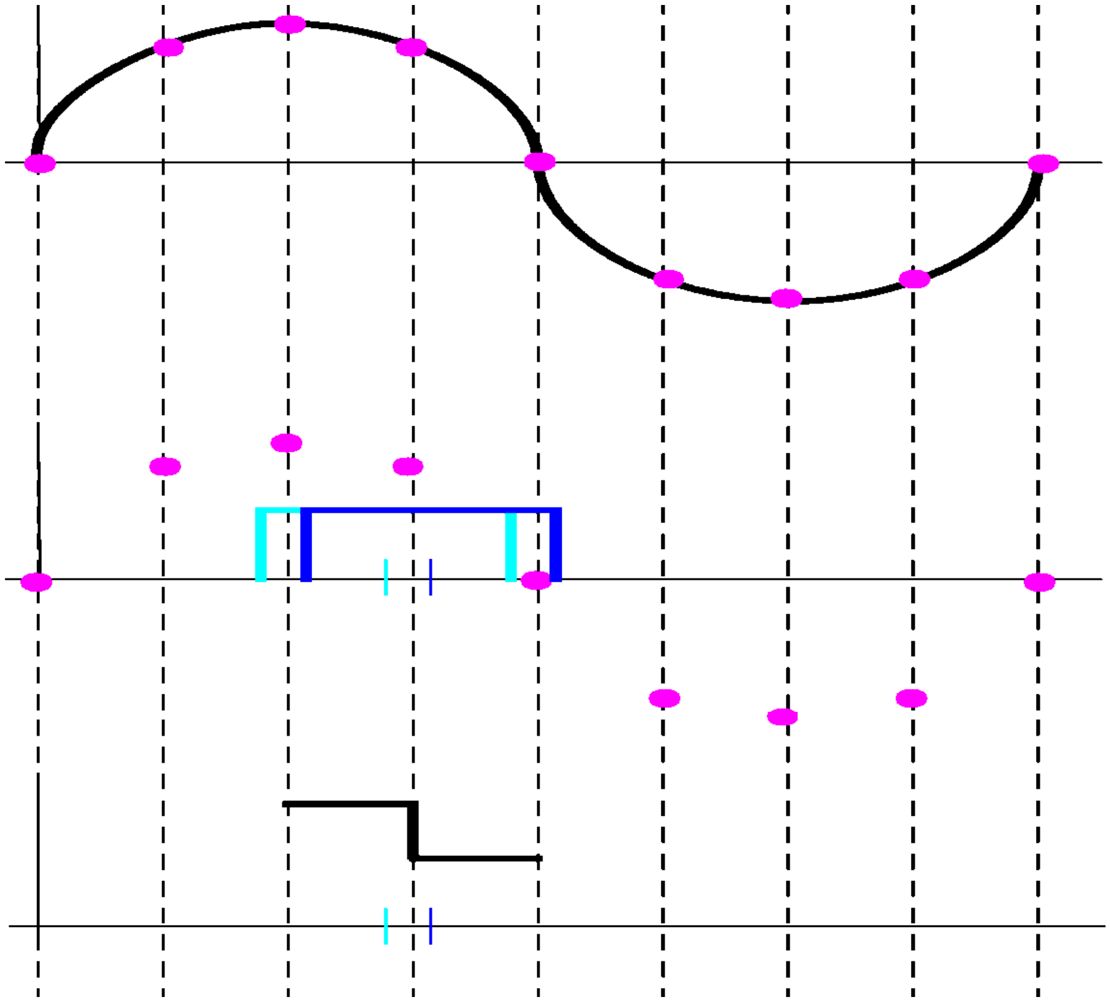
- Smooths out function by averaging neighbors
- Keeps low frequencies and reduces high frequencies (low-pass filter)
- Equally weights all samples
- In frequency domain, contains sidelobes to infinity



$$\frac{1}{9}\begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}$$

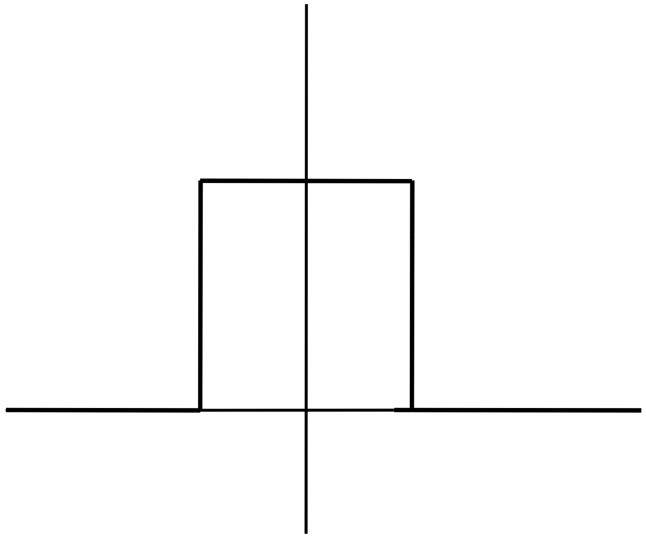
Box Filter

- Lousy for steadily varying signals, for instance, *sin(x)*

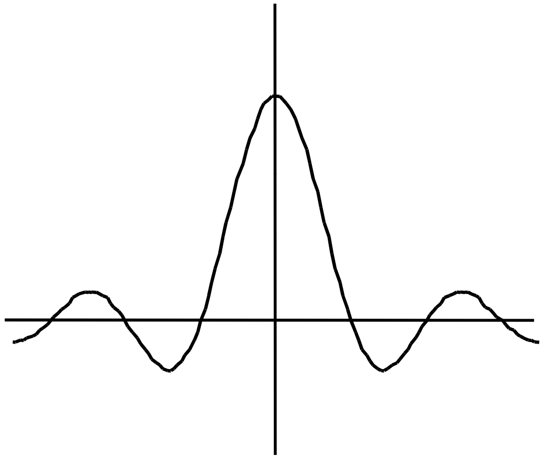


Box Filter

$$box(x) = \begin{cases} 1, & \text{if } |x| < T,\ T \in R^{+} \\ 0, & \text{elsewhere} \end{cases}$$



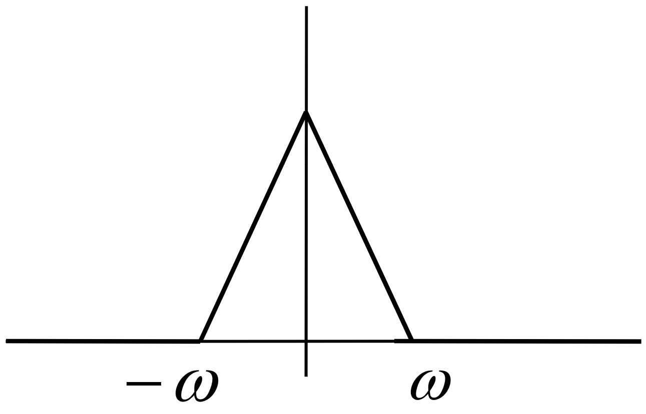
Spatial: Box



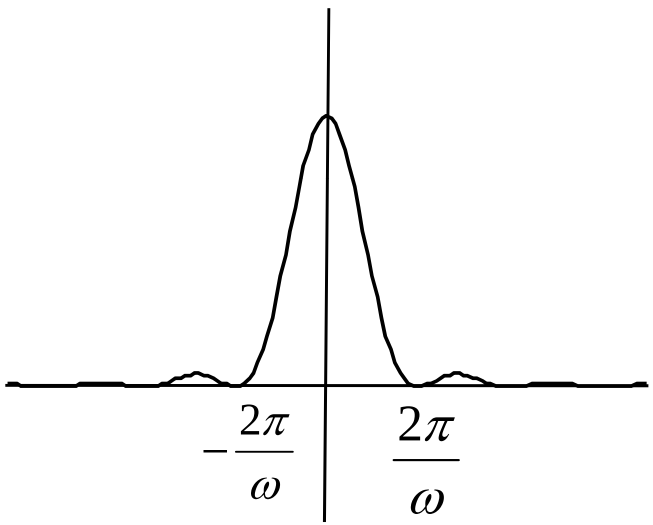
Frequency: sinc

Tent Filter

$$\mathrm{tent}_{\omega}(x) = 1 - \frac{|x|}{\omega} \quad \text{if } |x| < \omega$$
$$= 0 \qquad \text{otherwise}$$



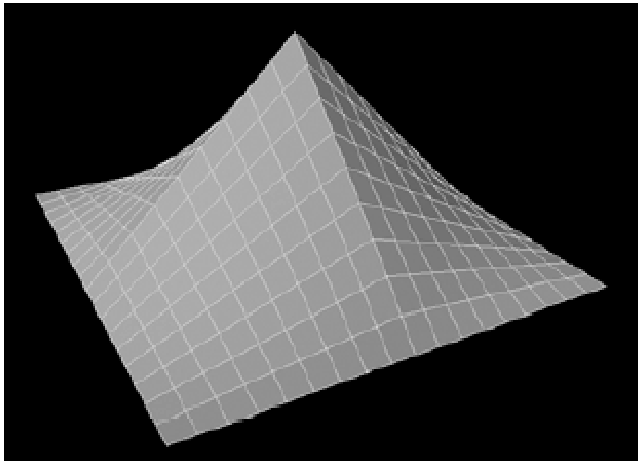
Spatial: Tent



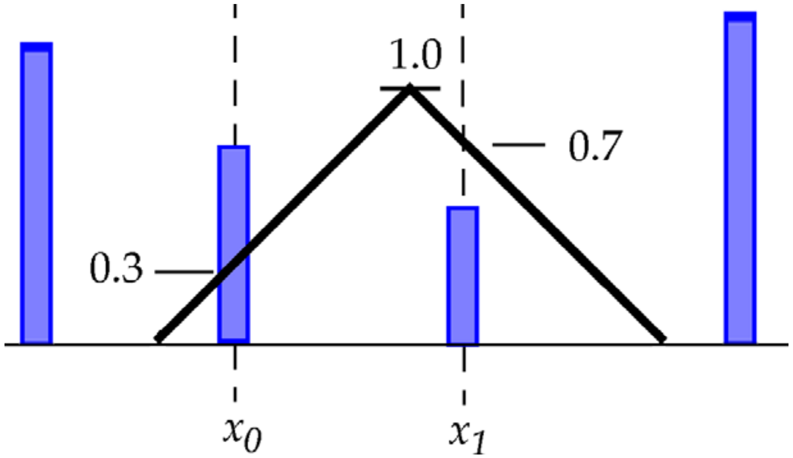
Frequency: sinc squared

Tent Filter

- Acts as linear interpolation filter
- Reduces high frequencies more
- Weights center sample more
- Other samples weighted linearly

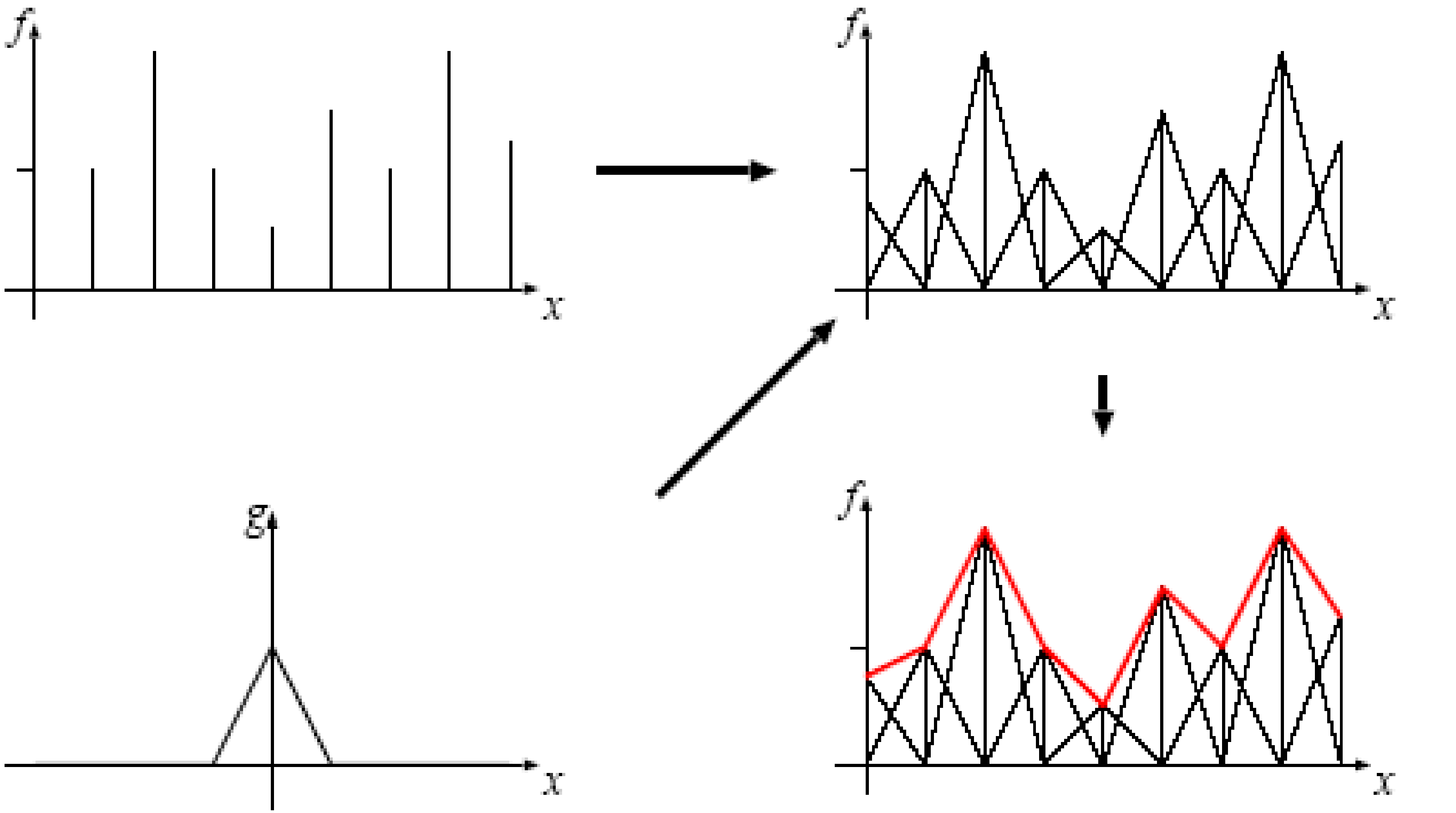


$$\frac{1}{16}\begin{bmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{bmatrix}$$



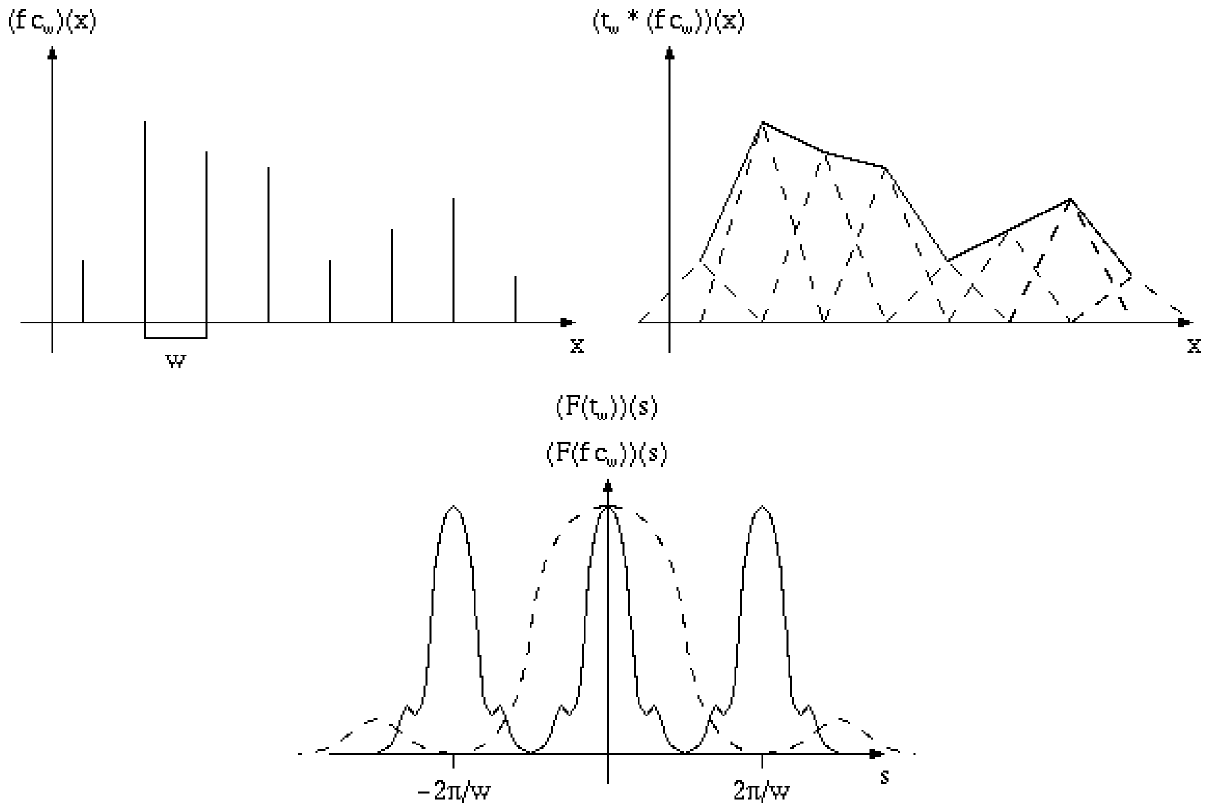
$$\text{weight} = 0.3 \cdot x_0 + 0.7 \cdot x_1$$

Tent Filter

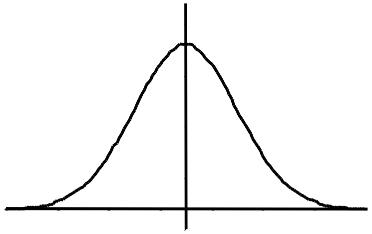


Tent Filter

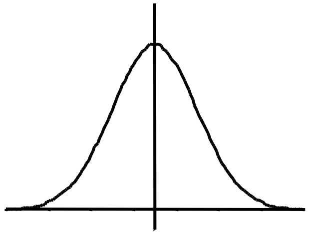
- Reconstructing a function using linear interpolation.
  - the Bartlett filter not only does not separate the original spectrum from the replications, it also aliases high-frequency components into the reconstruction due to its infinite support.



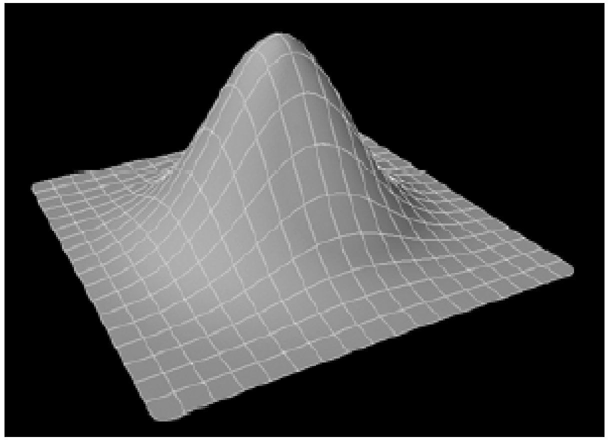
Gaussian Filter



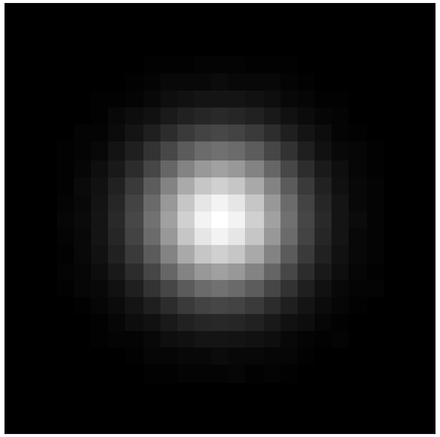
Spatial: Gaussian



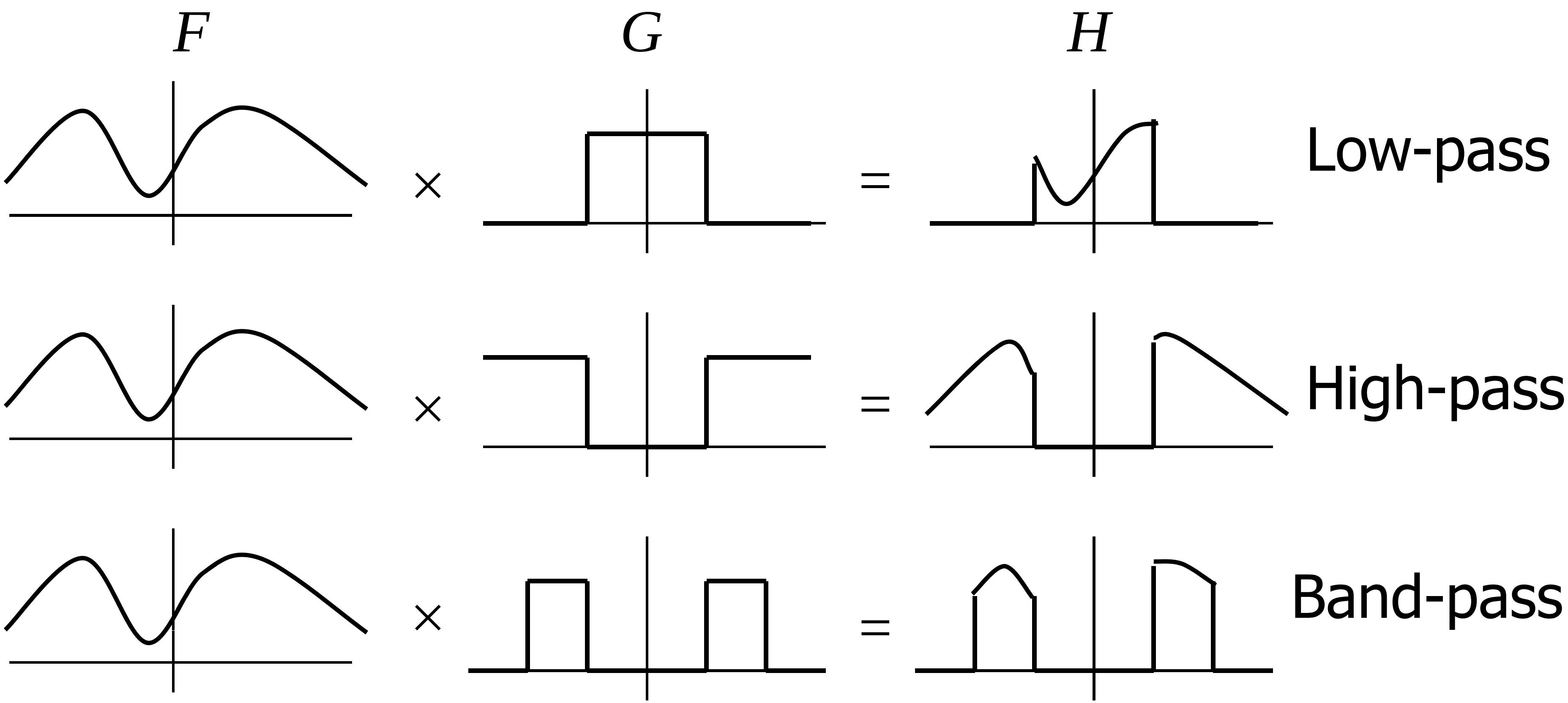
Frequency: Gaussian



- Reduces high frequences even more
- No sharp edges like in box, tent



Qualitative Filters



Filtering in spatial domain

- Work in the discrete spatial domain
- Convert the filter into a matrix, the *filter mask*
- Move the matrix over each point in the image, multiply the entries by the pixels below, then sum
  - Eg. 3x3 box filter
    - averages

$$\frac{1}{9}\begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}$$

Filtering in Spatial domain

I

10	11	10	0	0	1
9	10	11	1	0	1
10	9	10	0	2	1
11	10	9	10	9	11
9	10	11	9	99	11
10	9	9	11	10	10

F

1/9

1	1	1
1	1	1
1	1	1

O

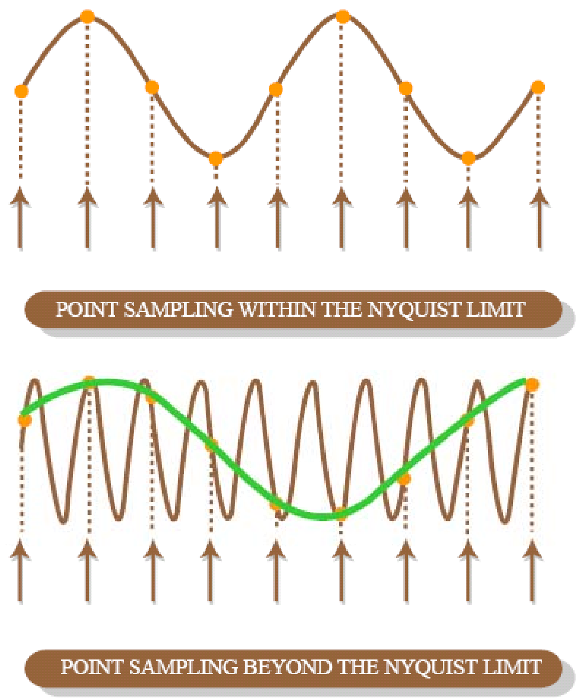
X	X	X	X	X	X
X	10				X
X					X
X					X
X					X
X	X	X	X	X	X

1/9.(10x1 + 11x1 + 10x1 + 9x1 + 10x1 + 11x1 + 10x1 + 9x1 + 10x1) = 1/9.(90) = 10

The Nyquist Theorem

When we can reconstruct the original continuous-time signal from its samples ?

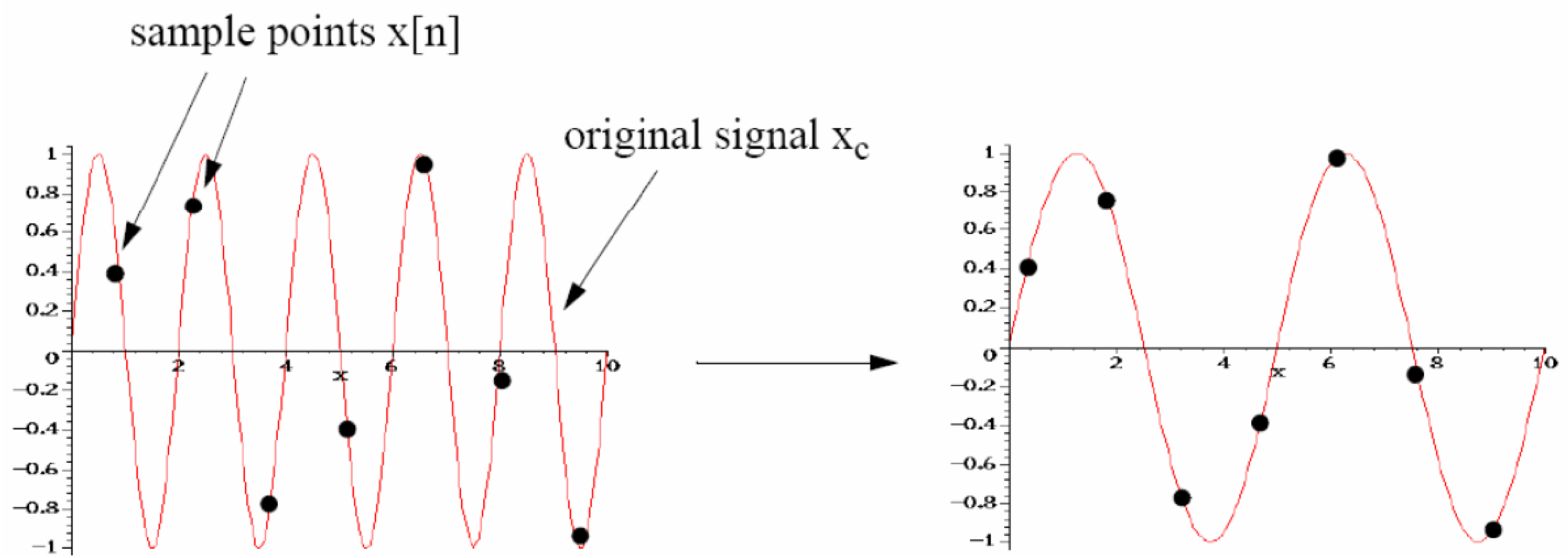
- There is a minimum frequency with which functions must be sampled – the *Nyquist frequency*
  - Twice the maximum frequency present in the signal
  - Example: Human ear hears frequencies up to 20 kHz → CD sample rate is 44.1 kHz.
- Not all sampling schemes allow reconstruction
  - eg: Sampling with a box



The Nyquist Theorem
- Example 1

Frequency of original signal: 0.5 Hz

Sampling frequency: 0.7 Hz



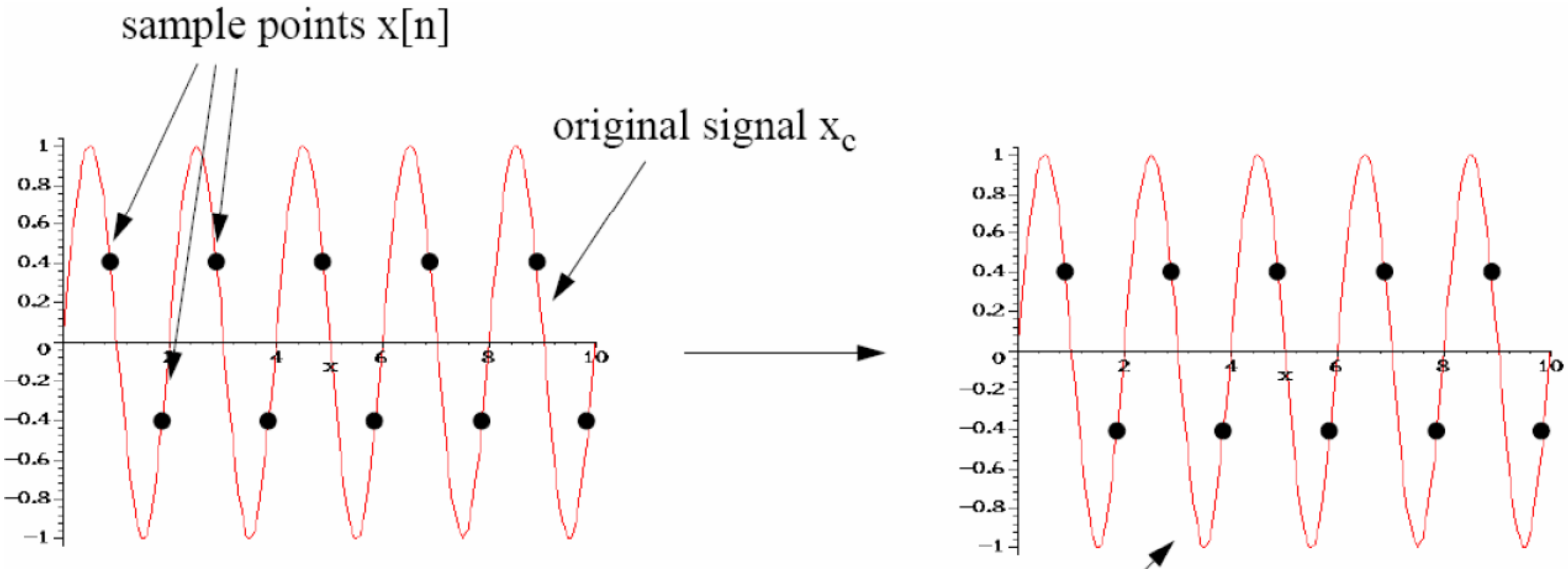
*Sampling freq. < 2*bandwidth*

The original wave cannot be recovered.

The Nyquist Theorem
- Example 2

Frequency of original signal: 0.5 Hz

Sampling frequency: 1.0 Hz

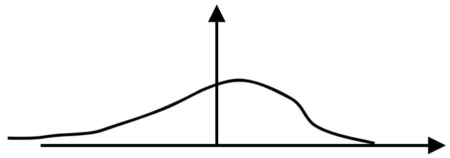


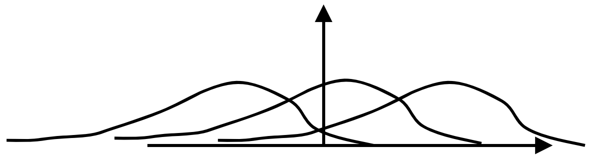
Sampling freq. = 2*bandwidth

The original wave may be recovered.

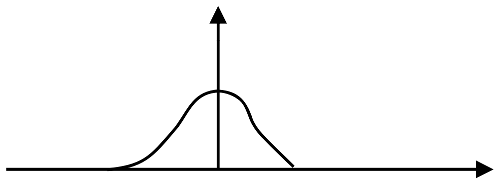
Sources of Aliasing

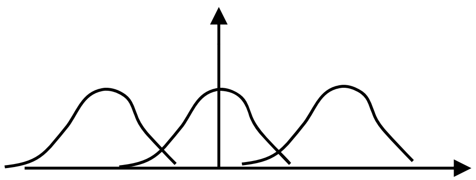
- Non-bandlimited signal



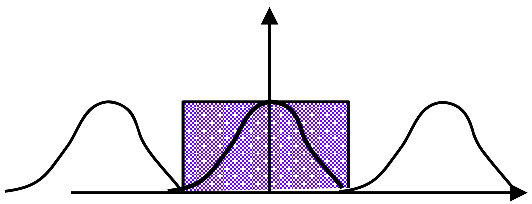


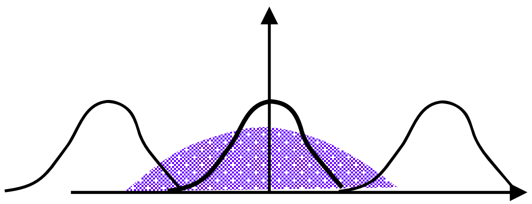
- Low sampling rate (below Nyquist)





- Non perfect reconstruction



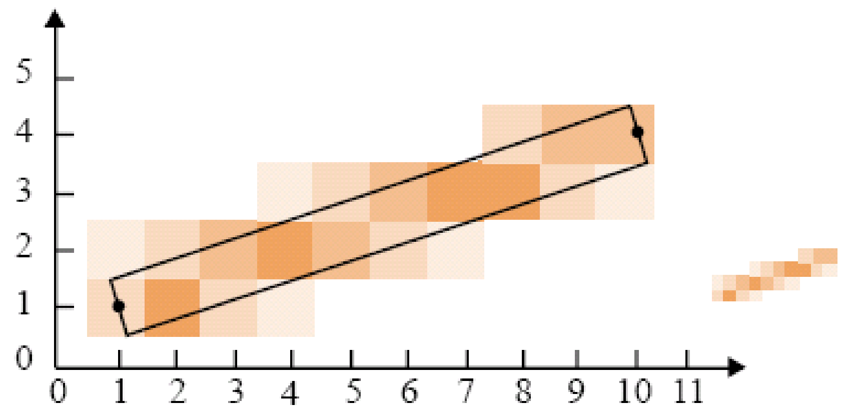


Prefiltering

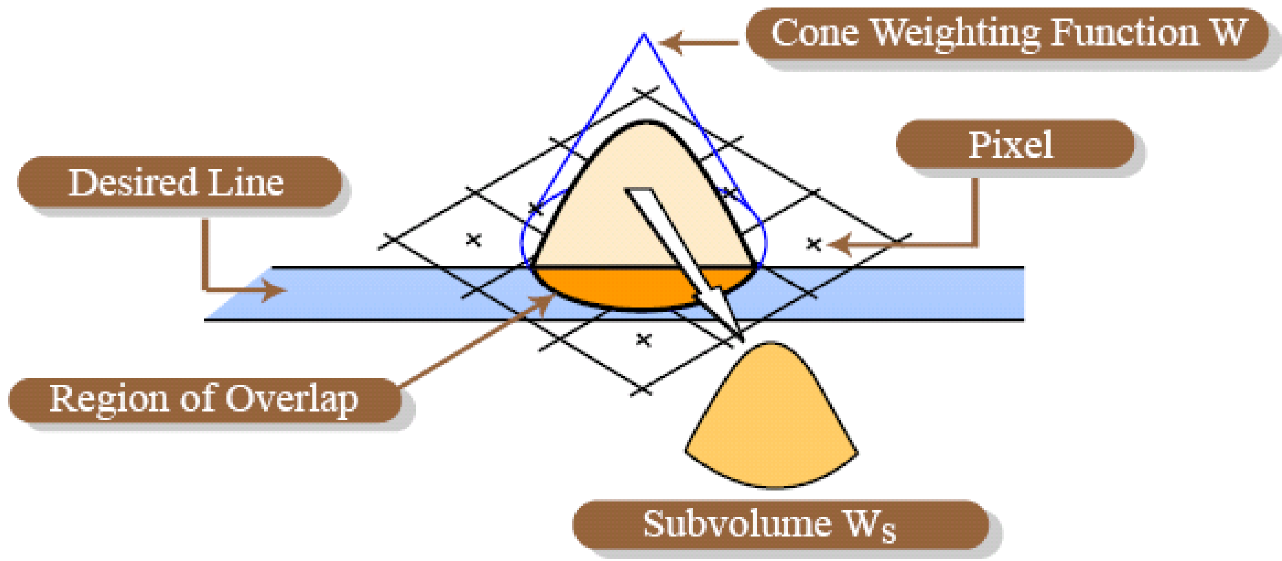
- Before sampling the image, use a low-pass filter to eliminate frequencies above the Nyquist limit
- This blurs the image, but ensures that no high frequencies will be misrepresented as low frequencies
- Determines pixel intensity based on the amount that a particular pixel is covered by an object in the scene. Determining such areas requires extensive calculations and integral approximations

Basis for Prefiltering

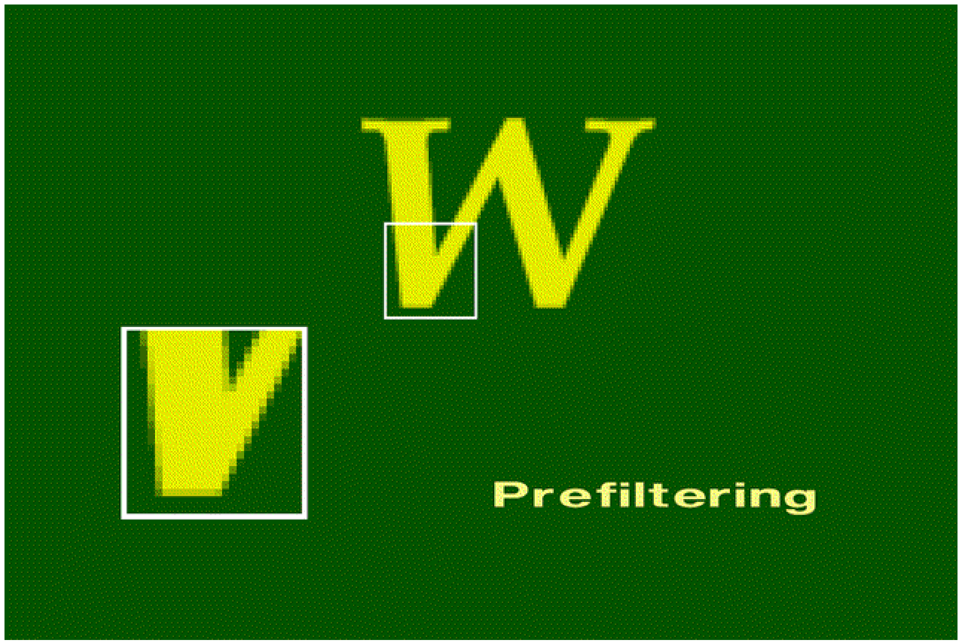
1. Treat a pixel as an area
2. Compute weighted amount of object overlap



What weighting function should we use?
How in volume rendering?



Prefiltering - *example*



Postfiltering

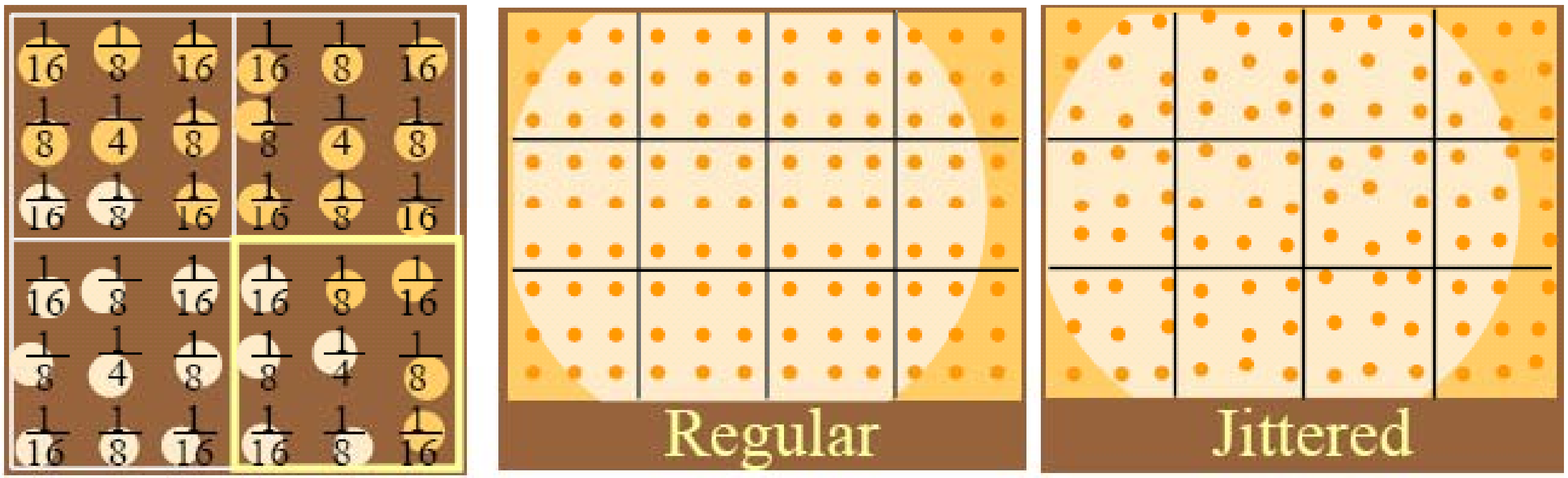
- Postfiltering, also known as supersampling
- Sample image at higher resolution than final image, then "average down"
- "Average down" means multiply by low-pass function in frequency domain
- Doesn't eliminate aliasing, just shifts the Nyquist limit higher
  - Cannot fix some scenes (e.g., checkerboard)
  - Badly inflates storage requirements
- Relatively easy and often works all right in practice
- *Can be added to a standard renderer : A-buffer*

Postfiltering

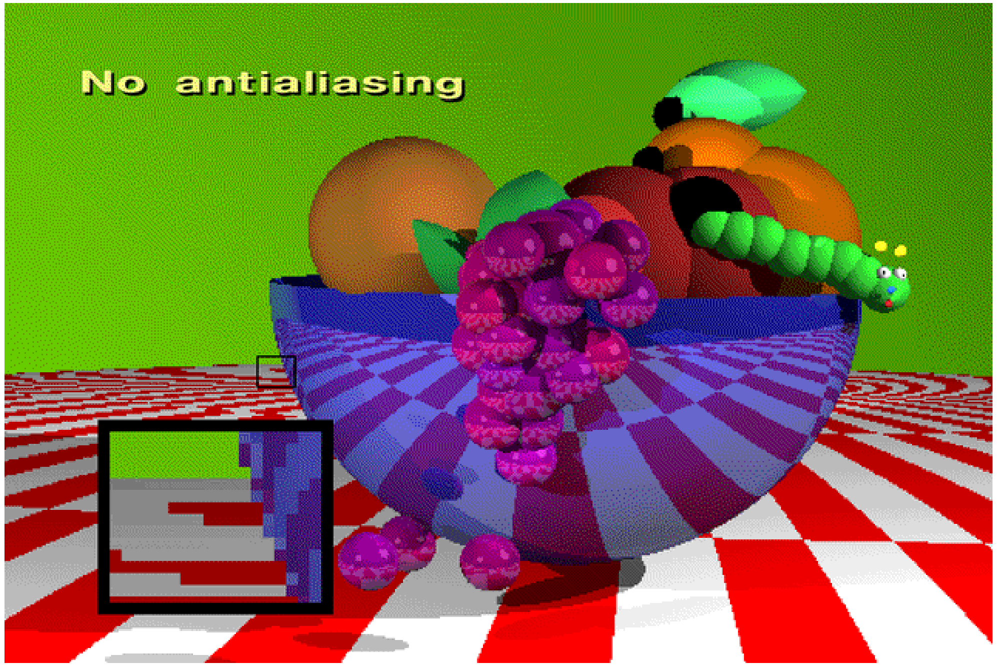
- The two steps in the postfiltering process are:
  1. Sample the scene at *n* times the display resolution.
  2. The color of each pixel in the rendered image will be an average of several samples.
- A filter provides the weights used to compute the average.

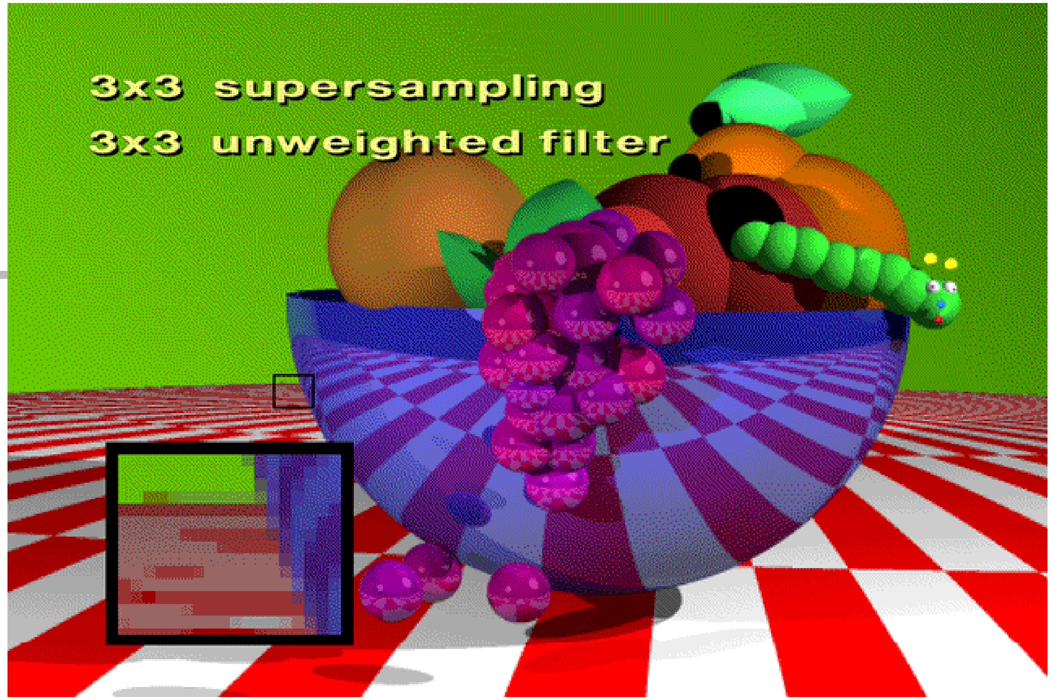
Sampling in the postfiltering method

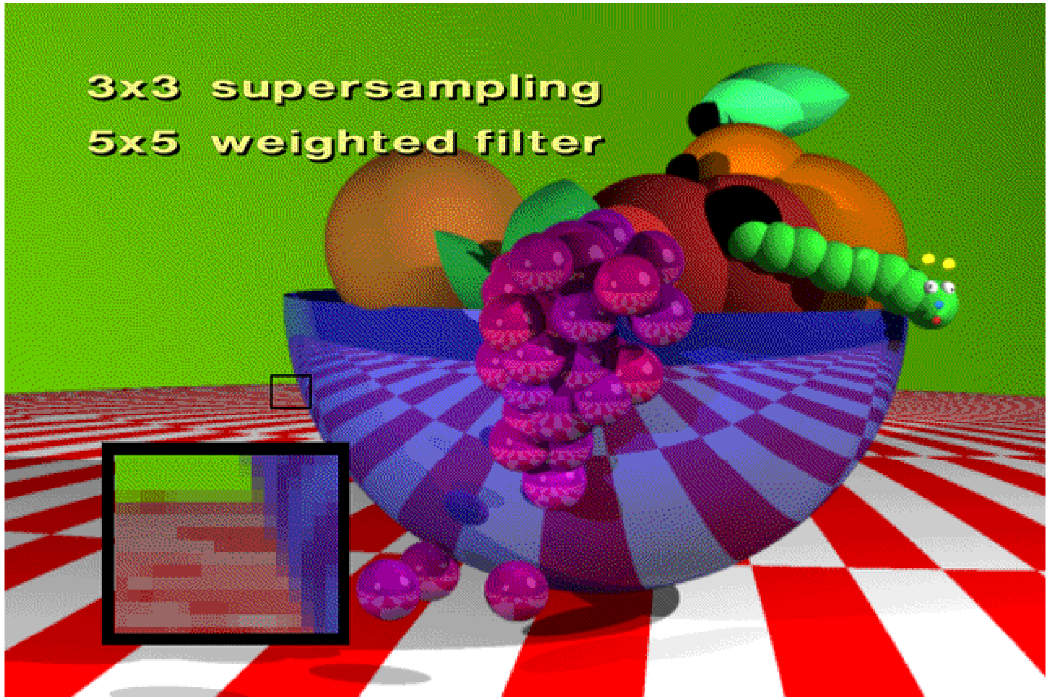
- Supersampling from a 4x3 image.
- Compute the weighted average of many samples (9 samples for each pixel)
- Sampling can be done randomly or regularly. The method of perturbing the sample positions is known as "jittering."



Antialiasing



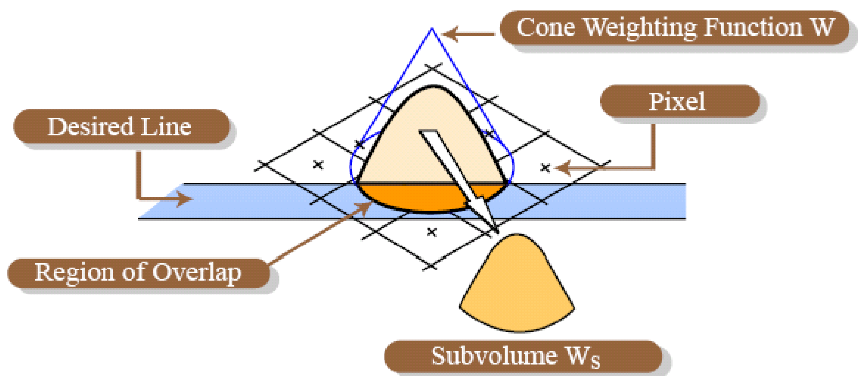


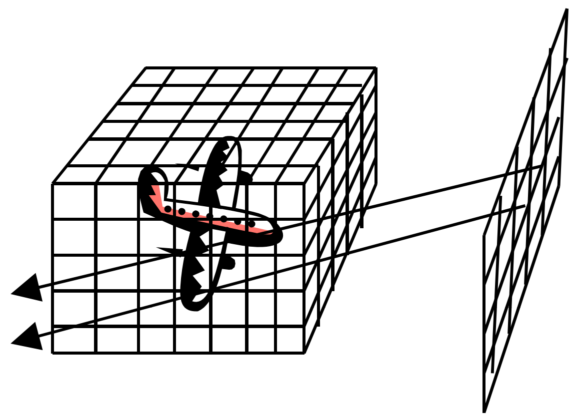


http://www.siggraph.org/education/materials/HyperGraph/aliasing

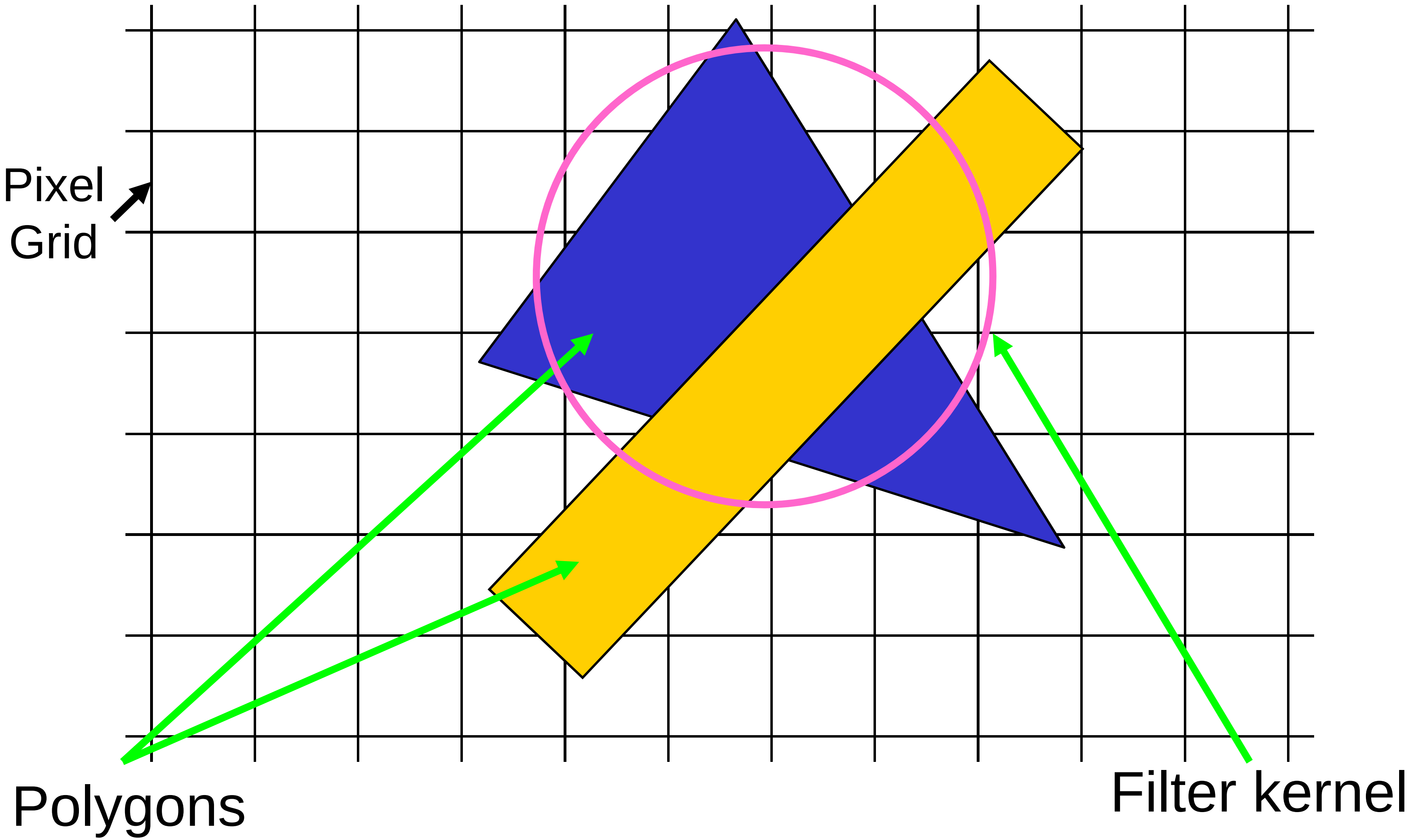
Antialiasing in the continuous domain

- Problem with Prefiltering:
  - Sampling and image generation inextricably linked in most renderers
    - Z-buffer algorithm
    - Ray tracing
  - *Why?*
- Still, some approaches try to approximate the effect of convolution in the continuous domain - splatting



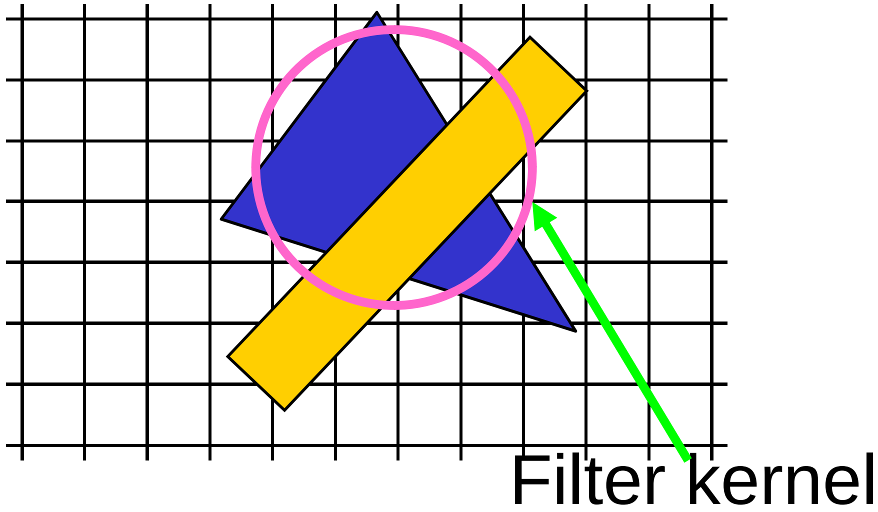


Antialiasing in the continuous domain



Antialiasing in the continuous domain

- The good news
  - Exact polygon coverage of the filter kernel can be evaluated
  - *What does this entail?*
    - Clipping
    - Hidden surface determination



Antialiasing in the continuous domain

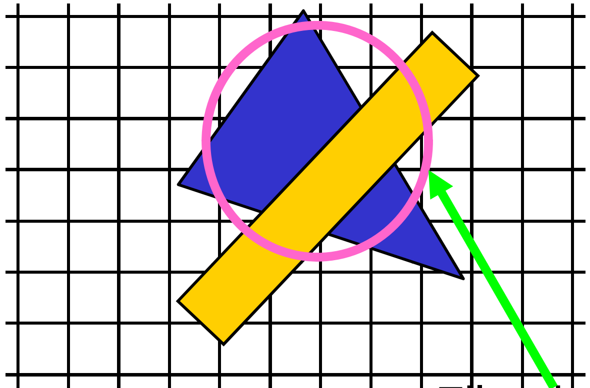
- The bad news
  - Evaluating coverage is very expensive
  - The intensity variation is too complex to integrate over the area of the filter

    Q: *Why does intensity make it harder?*

    A: Because polygons might not be flat- shaded

    Q: *How bad a problem is this?*

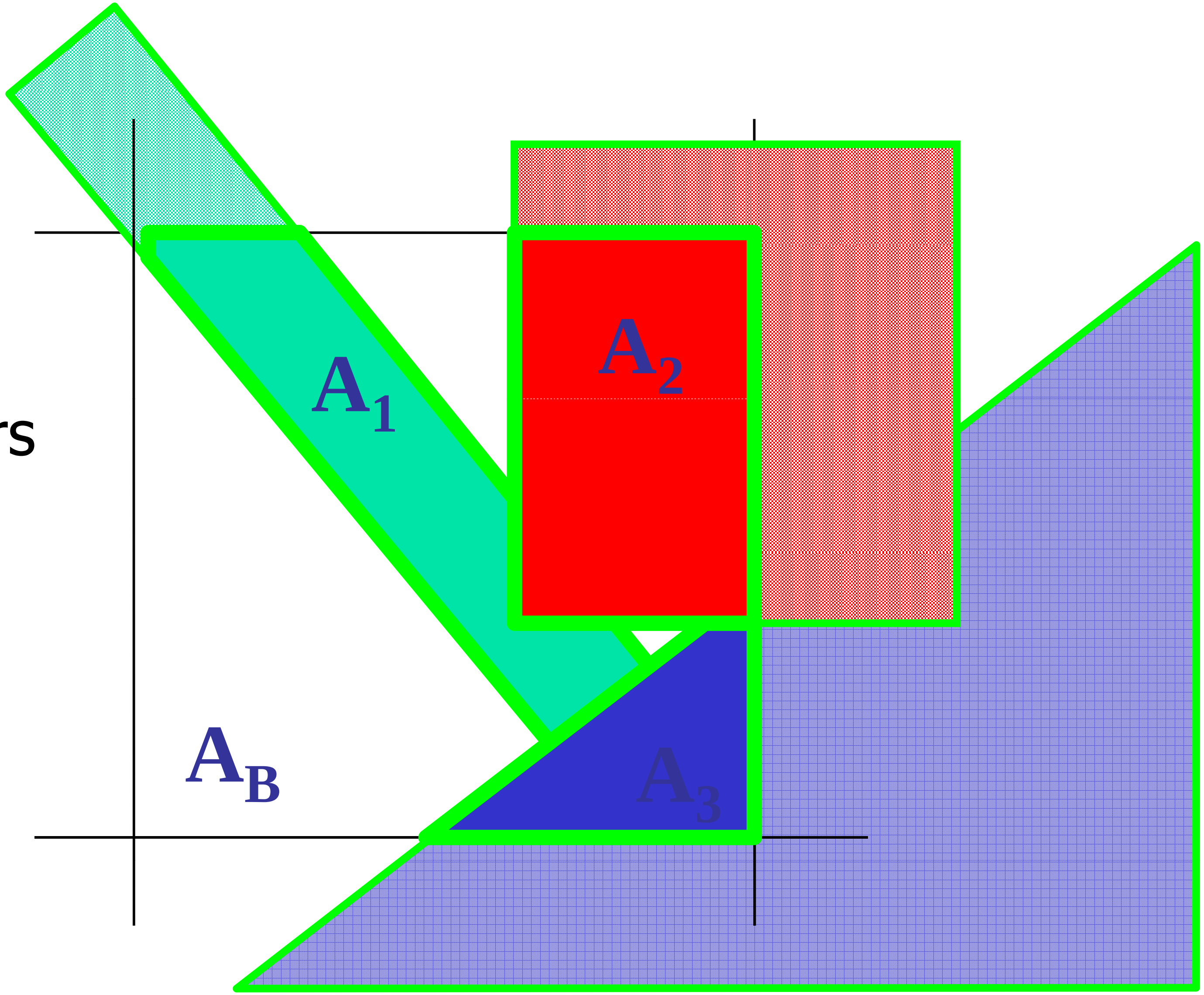
    A: Intensity varies slowly within a pixel, so shape changes are more important



Filter kernel

Catmull's Algorithm

- Find fragment areas
- Multiply by fragment colors
- Sum for final pixel color



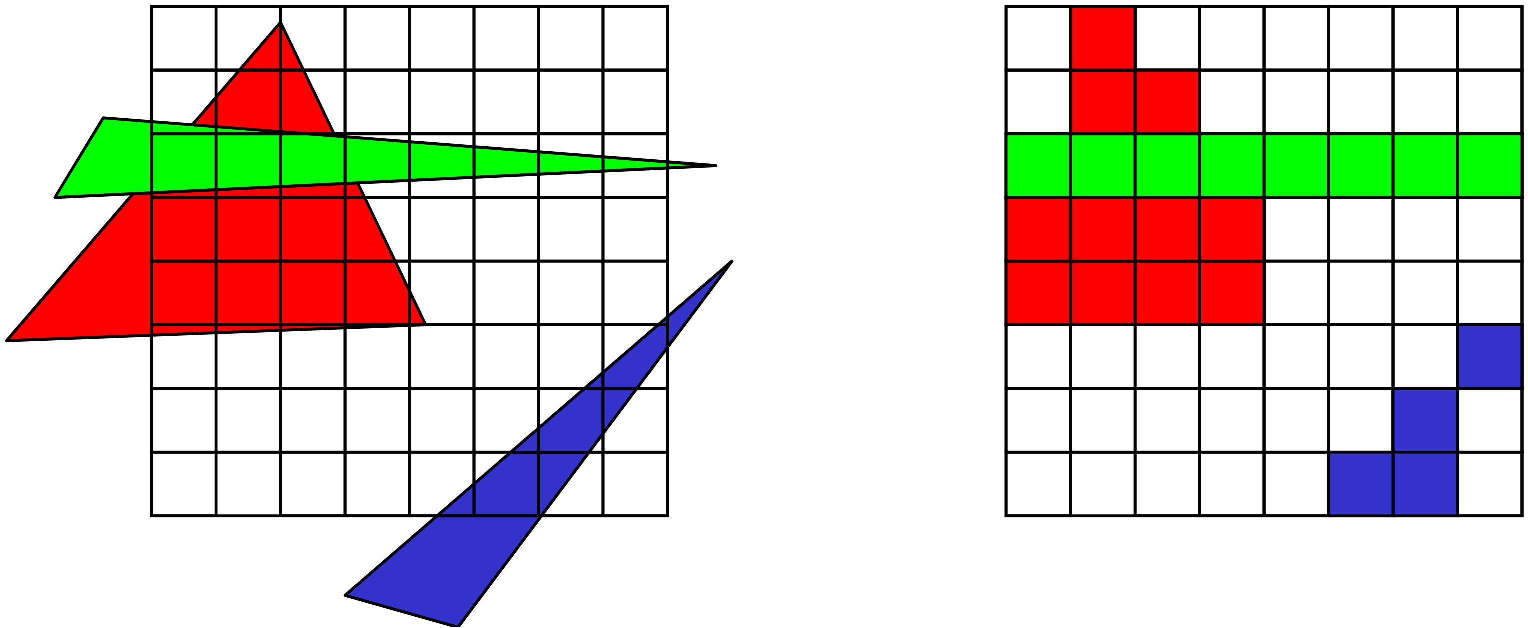
Catmull's Algorithm

- First real attempt to filter in continuous domain
- Very expensive
  - Clipping polygons to fragments
  - Sorting polygon fragments by depth

  (*What's wrong with this as a hidden surface algorithm?*)
- Equates to box filter (*Is that good?*)

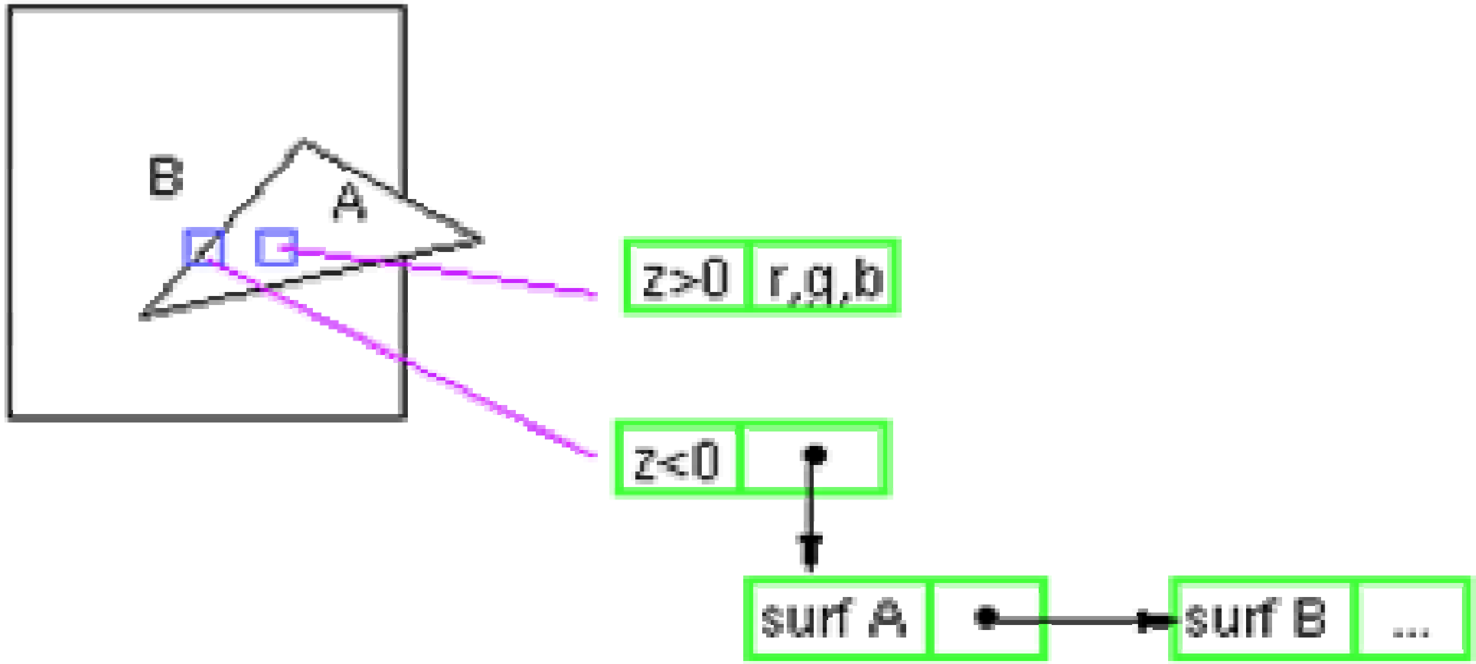
A-Buffer

- Accumulation buffer
- Idea: approximate continuous filtering by subpixel sampling
- Summing areas now becomes simple
- Commonly used in software to generate high quality renderings but not in real-time



A-Buffer

- *z-buffer: one visible surface per pixel*
- *A-buffer: linked list of surfaces*



Antialiasing Strategies

- Supersampling: sample at higher resolution, then filter down
  - Pros:
    - Conceptually simple
    - Easy to retrofit existing renderers
    - Works well most of the time
  - Cons:
    - High storage costs
    - Doesn't eliminate aliasing, just shifts Nyquist limit upwards
- A-Buffer: approximate pre-filtering of continuous signal by sampling
  - Pros:
    - Integrating with scan-line renderer keeps storage costs low
    - Can be efficiently implemented with clever bitwise operations
  - Cons:
    - Still basically a super-sampling approach
    - Doesn't integrate with ray-tracing

What you learn?

- Sampling
- Transform (signal ⇔ frequency)
- Filters
- Aliasing & Anti-alising
- Nyquist Limit
- Why jagged effects in CG image?
- How you solve that?