Computer Graphics

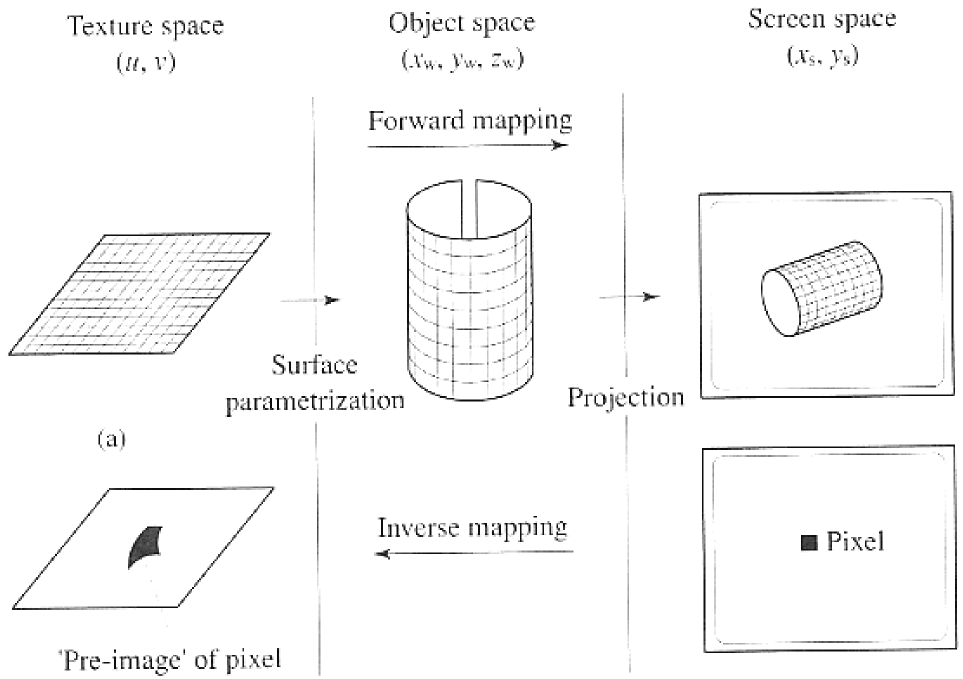
Texture Filtering & Sampling Theory

Hendrik Lensch

Overview

- **Last time**
  – Texture Parameterization
  – Procedural Shading
- **Today**
  – Texturing Filtering

2D Texture Mapping



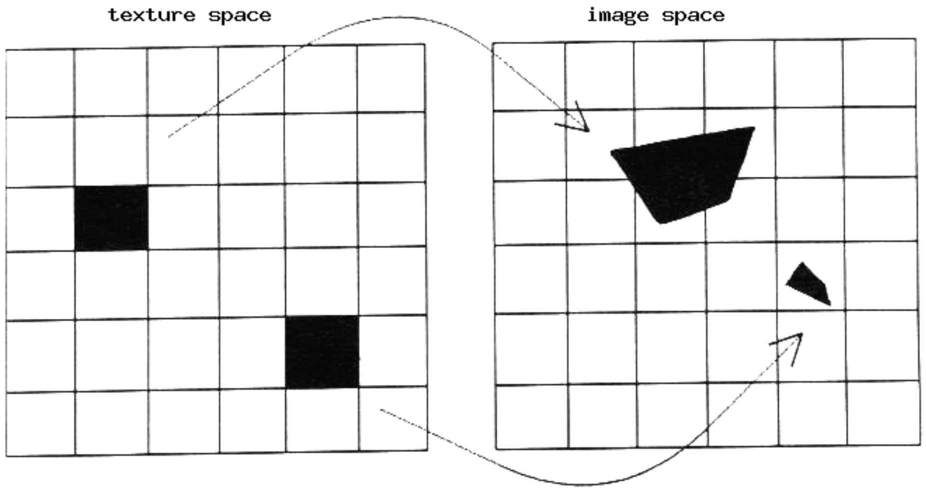
- **Forward mapping**
  - Object surface parameterization
  - Projective transformation
- **Inverse mapping**
  - Find corresponding pre-image/footprint of each pixel in texture
  - Integrate over pre-image

Forward Mapping

- **Maps each texel to its position in the image**
- **Uniform sampling of texture space does not guarantee uniform sampling in screen space**
- **Possibly used if**
  - The texture-to-screen mapping is difficult to invert
  - The texture image does not fit into memory

Texture scanning:

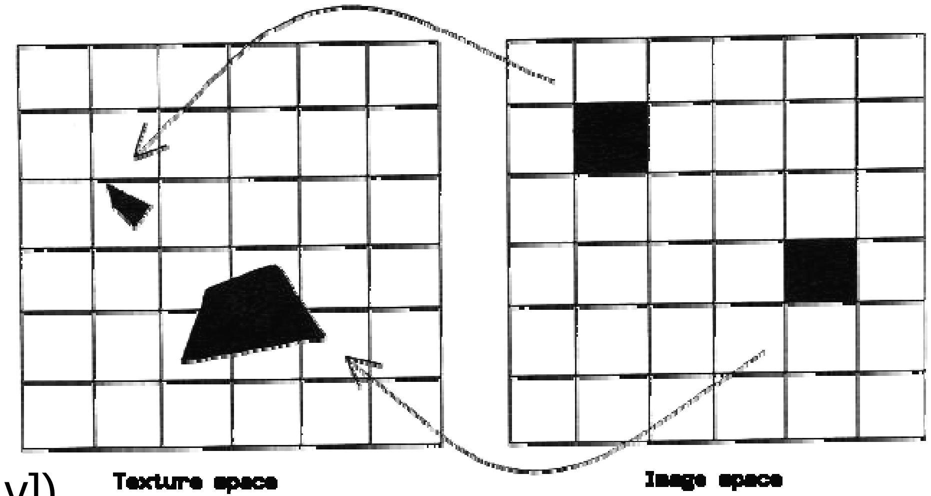
```
for v
  for u
    compute x(u,v) and y(u,v)
    copy TEX[u,v] to SCR[x,y]
(or in general
     rasterize image of TEX[u,v])
```



Inverse Mapping

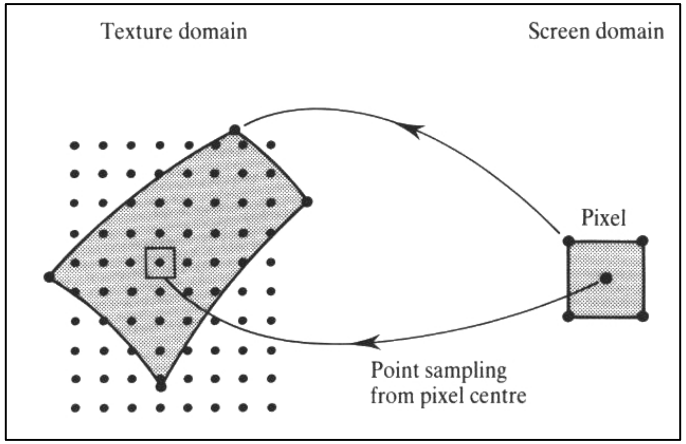
- **Requires inverting the mapping transformation**
- **Preferable when the mapping is readily invertible and the texture image fits into memory**
- **The most common mapping method**
  – for each pixel in screen space, the pre-image of the pixel in texture space is found and its area is integrated over

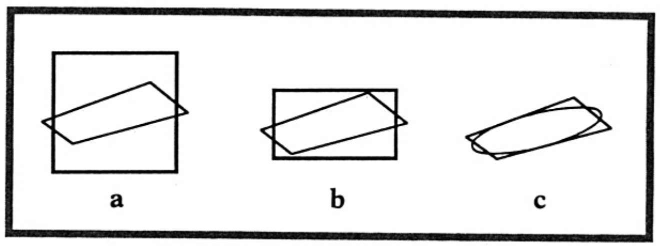
Screen scanning:
for y
  for x
    compute u(x,y) and v(x,y)
    copy TEX[u,v] to SCR[x,y]
(or in general
    integrate over image of SCR[u,v])



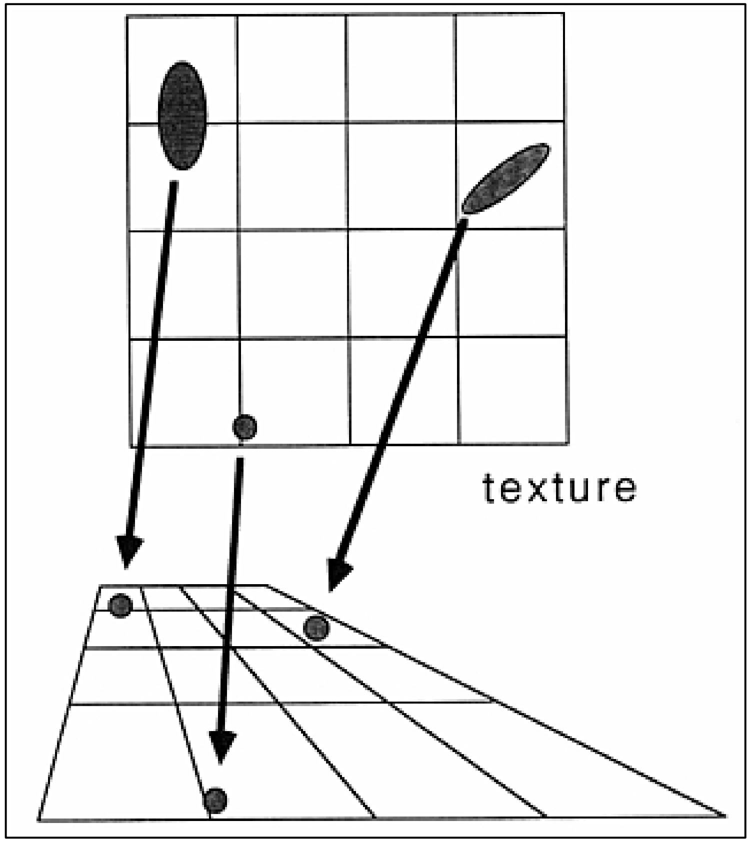
Pixel Pre-Image in Texture Space

A square screen pixel that intersects a curved surface has a curvilinear quadrilateral pre-image in texture space. Most methods approximate the true mapping by a quadrilateral or parallelogram. Or they take multiple samples within a pixel. If pixels are instead regarded as circles, their pre-images are ellipses.



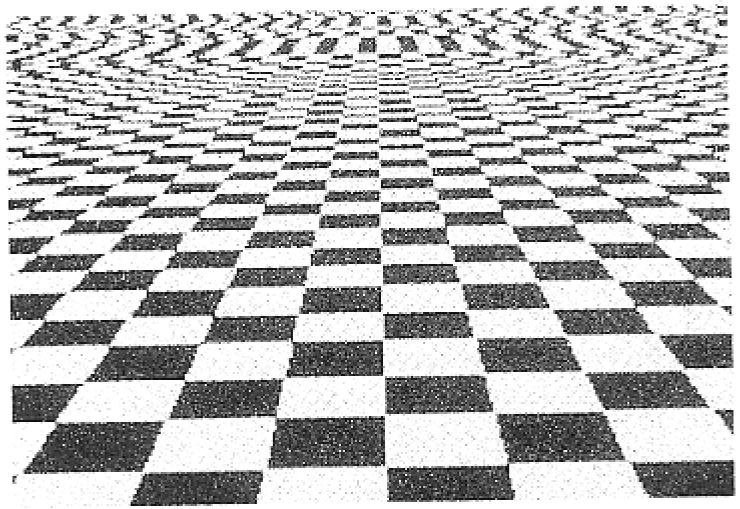


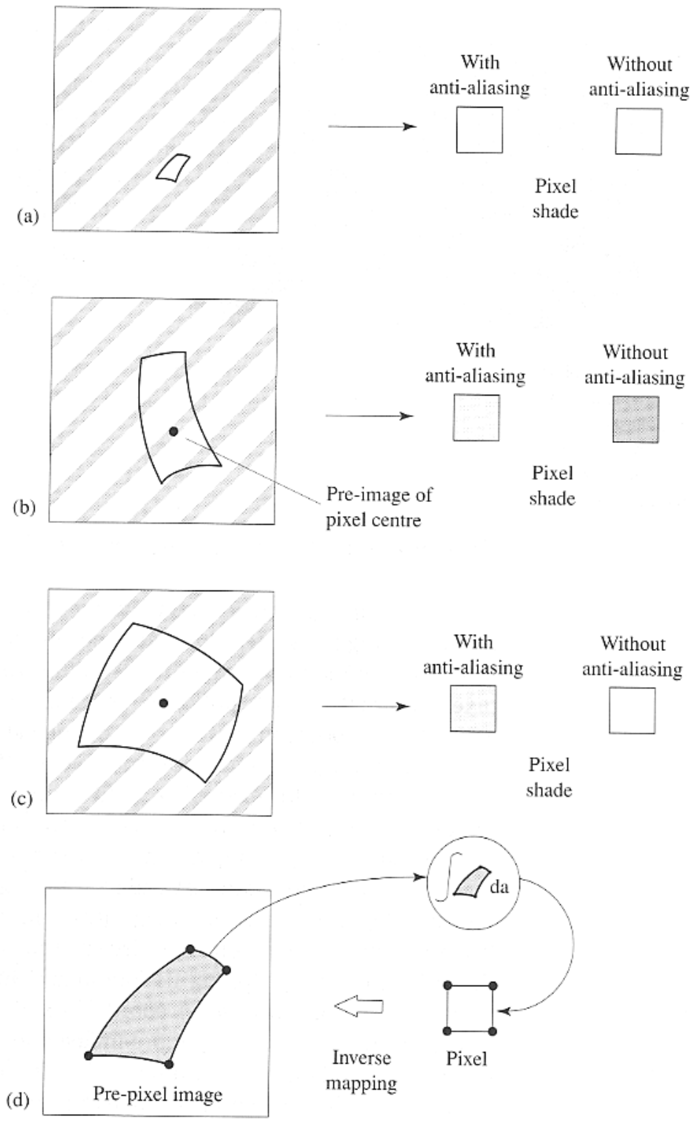
Approximating a quadrilateral texture area with (a) a square, (b) a rectangle, and (c) an ellipse. Too small an area causes aliasing; too large an area causes blurring.



Inverse Mapping: Filtering

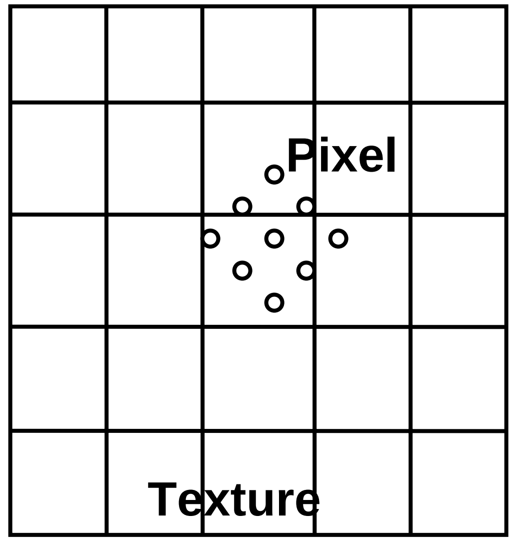
- **Integration of Pre-image**
  - Integration over pixel footprint in texture space
- **Aliasing**
  - Texture insufficiently sampled
  - Incorrect pixel values
  - “Randomly” changing pixels when moving

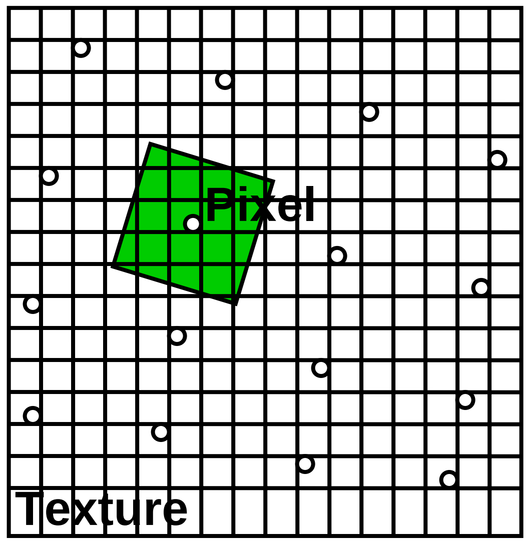




Filtering

- **Magnification**
  - Map few texels onto many pixels
  - Nearest:
    - Take the nearest texel
  - Bilinear interpolation:
    - Interpolation between 4 nearest texels
    - Need fractional accuracy of coordinates
- **Minification**
  - Map many texels to one pixel
    - Aliasing:
      - Reconstructing high-frequency signals with low level frequency sampling
  - Filtering
    - Averaging over (many) associated texels
    - Computationally expensive



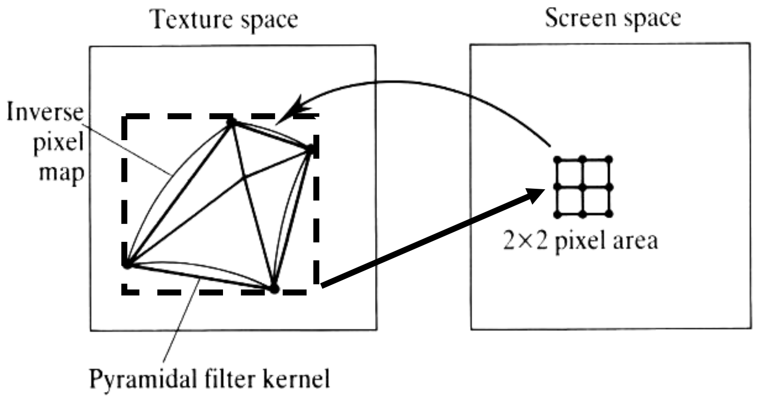


Filtering – Texture Minification

- **Space-variant filtering**
  - Mapping from texture space *(u,v)* to screen space *(x,y)* not affine
  - Filtering changes with position
- **Space variant filtering methods**
  - ***Direct convolution***
    - Numerically compute the Integral
  - ***Pre-filtering***
    - Precompute the integral for certain regions -- more efficient
    - Approximate footprint with regions

Direct Convolution

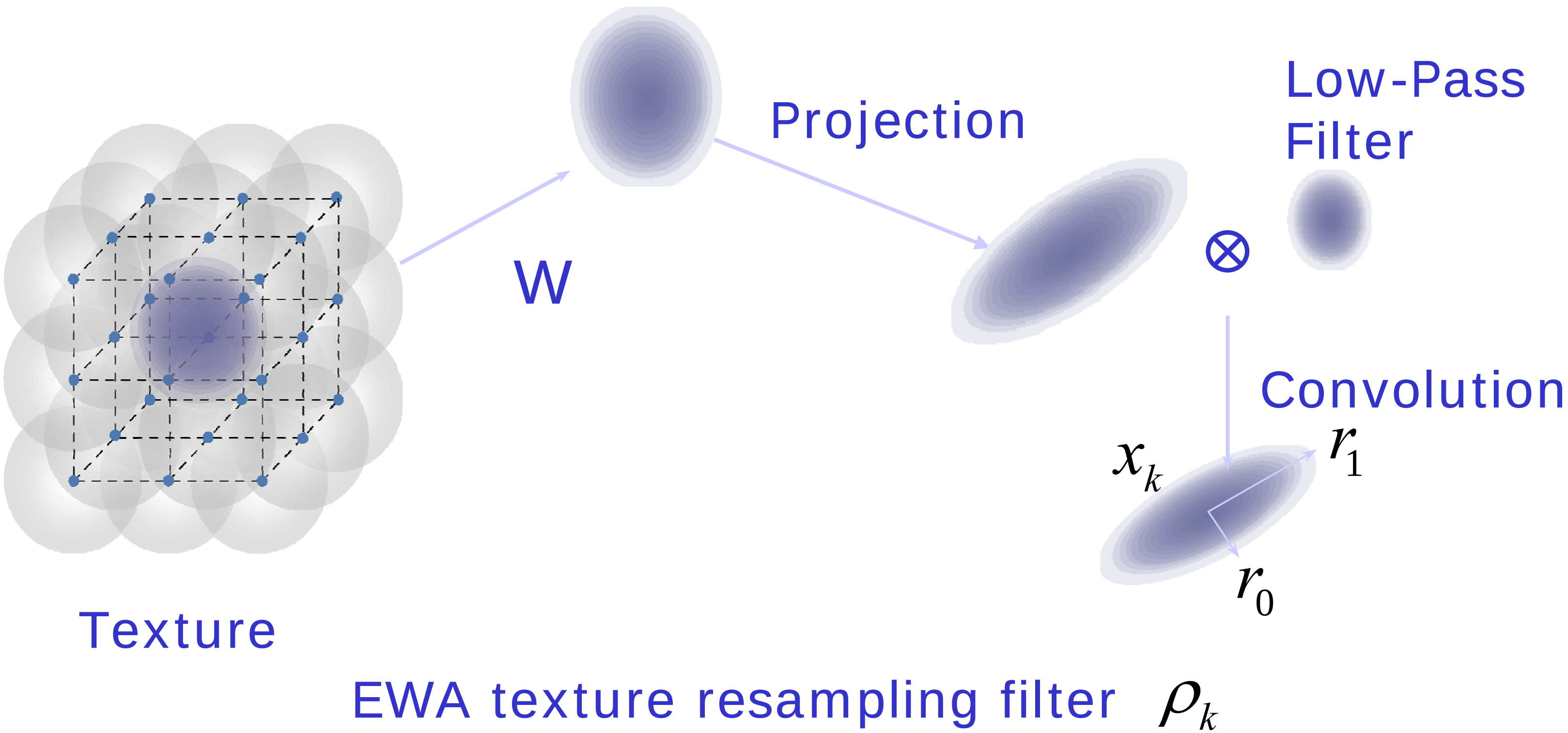
- **Convolution in texture space**
  - Texels weighted according to distance from pixel center (e.g. pyramidal filter kernel)



- **Convolution in image space**
  1. **Center the filter function on the pixel (in image space) and find its bounding rectangle.**
  2. **Transform the rectangle to the texture space, where it is a quadrilateral. The sides of this rectangle are assumed to be straight. Find a bounding rectangle for this quadrilateral.**
  3. **Map all pixels inside the texture space rectangle to screen space.**
  4. **Form a weighted average of the mapped texture pixels using a two-dimensional lookup table indexed by each sample's location within the pixel.**

EWA Filtering

- **Compensate aliasing artifacts due to perspective projection**
- **EWA Filter = low-pass filter $\otimes$ warped reconstruction filter**

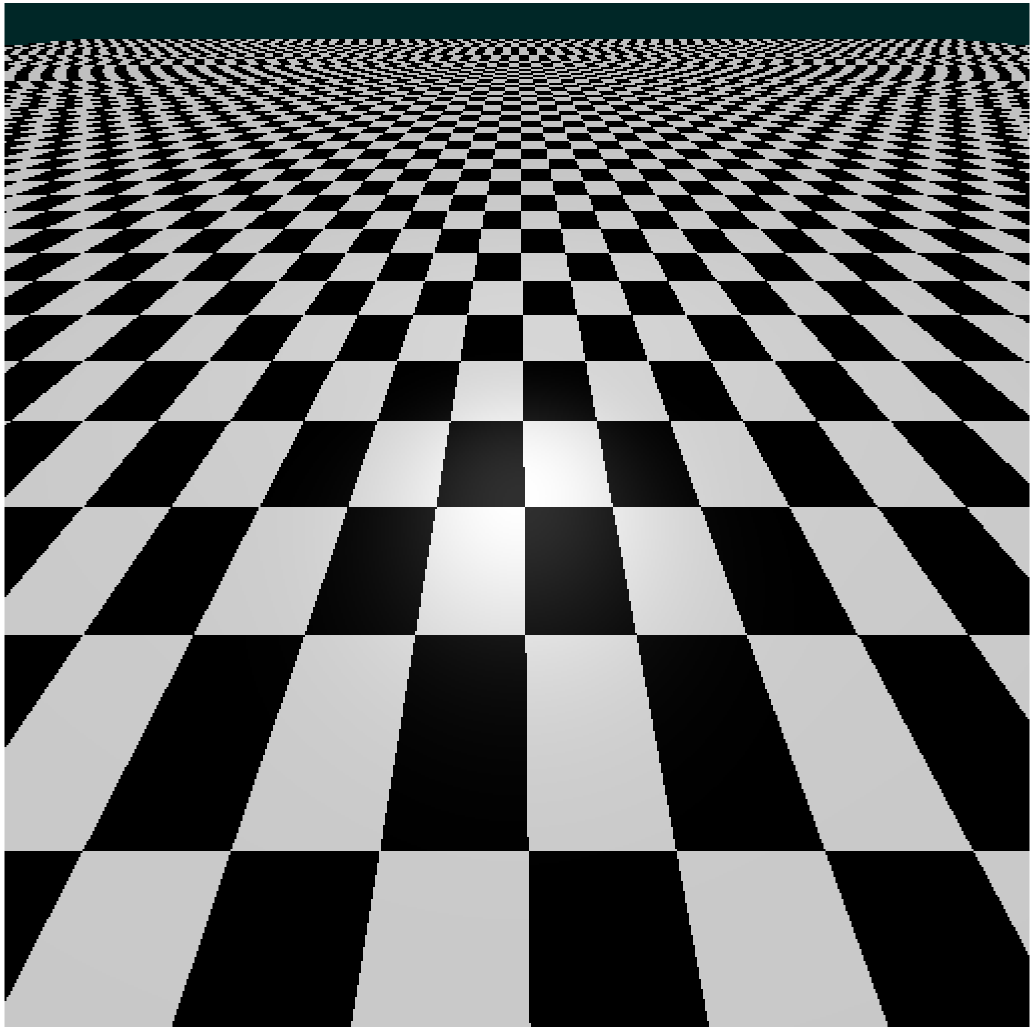


EWA Filtering

- **Four step algorithm**
  1) calculate the ellipse
  2) choose filter
  3) scan conversion in the ellipse
  4) determine the color for the pixel.

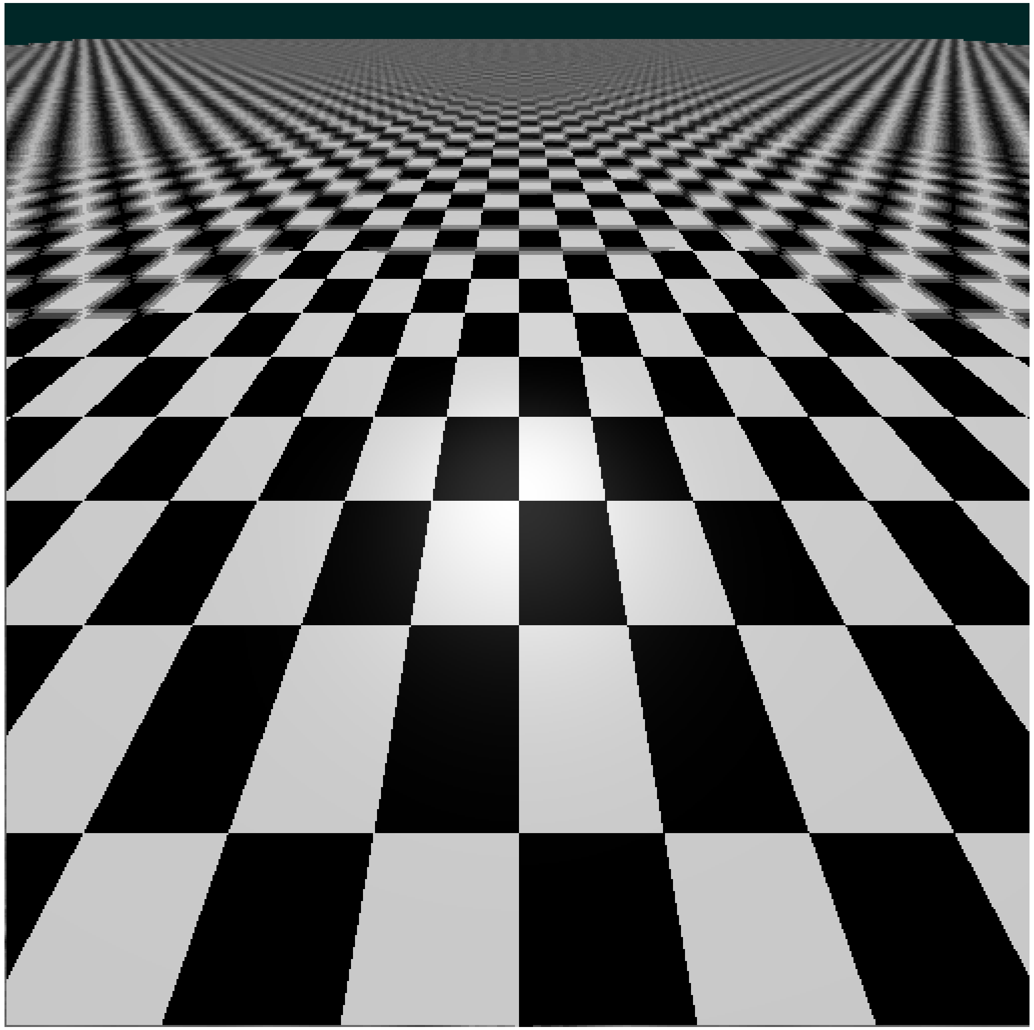
Without Anti-aliasing

- **checker board gets distorted**



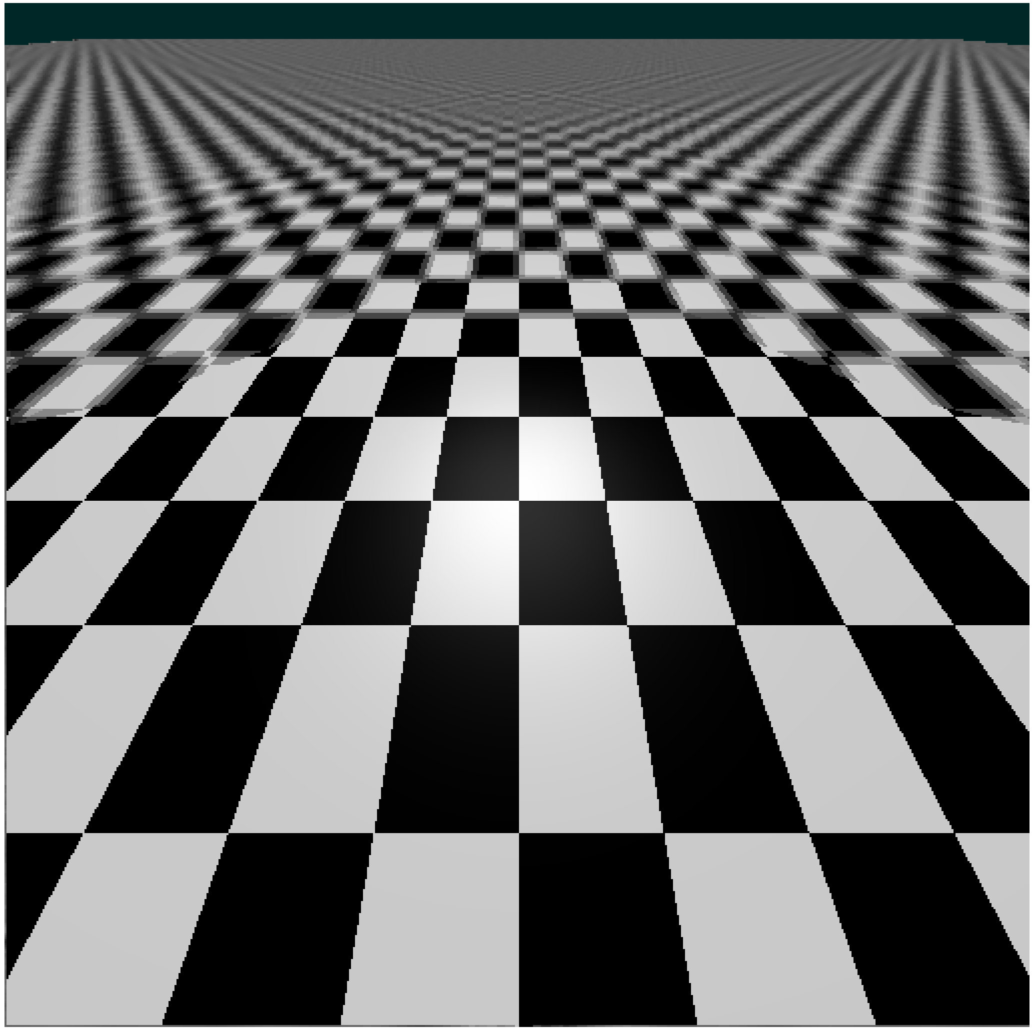
EWA Filtering

- **elliptical filtering plus Gaussian**

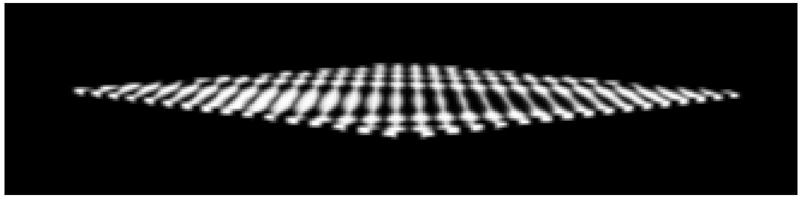


EWA Filtering

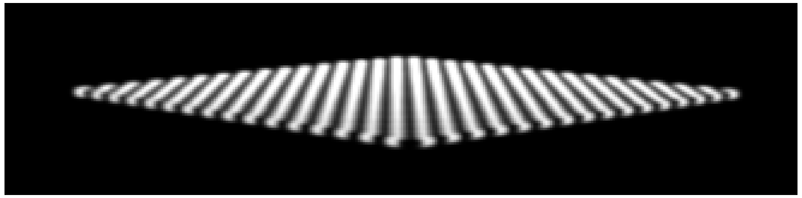
- **Gaussian blur selected too large -> blurry image**



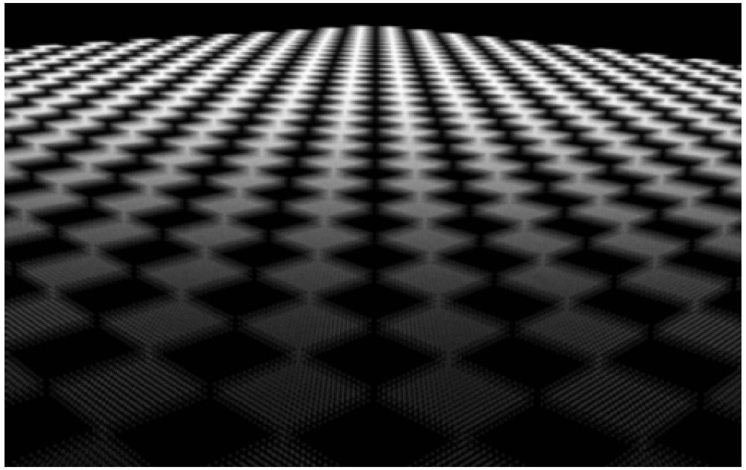
EWA Splatting



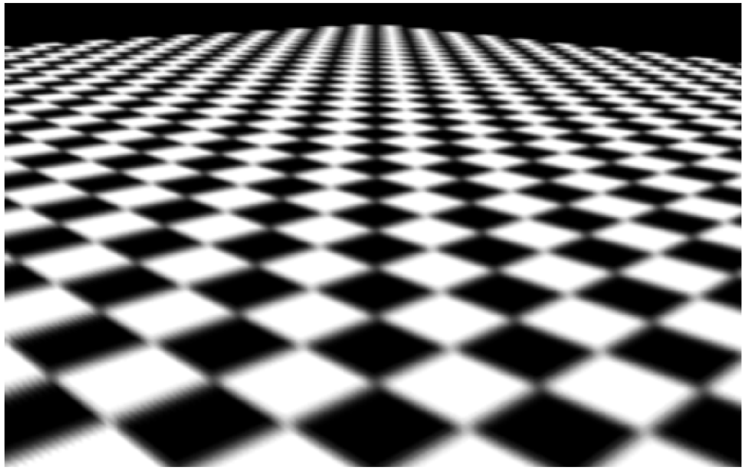
Reconstruction filter only: 6.25 fps



EWA filter: 4.97 fps

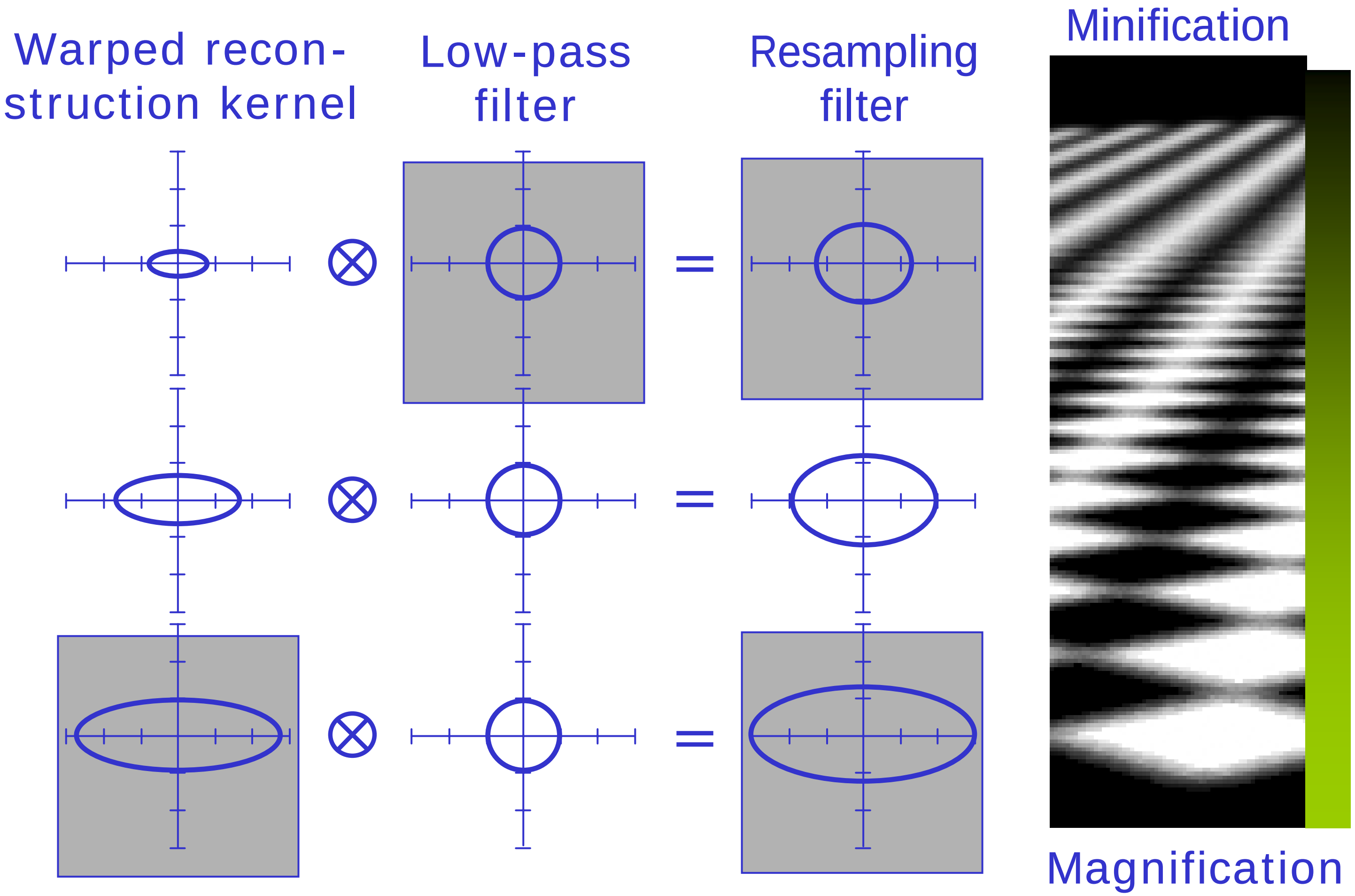


Low-pass filter only: 6.14 fps



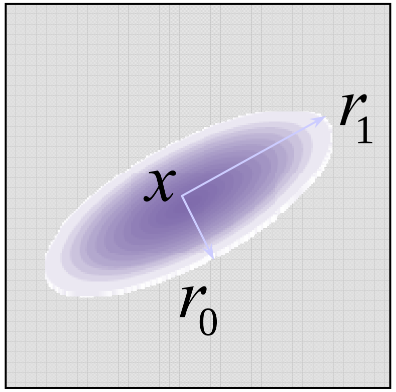
EWA filter: 3.79 fps

Analysis of EWA Filter



Analysis of EWA Filter

- **Shape of EWA Splat is dependent on distance from the view plane**



EWA splat

$$r_0 = \sqrt{\frac{r_k}{\boxed{u_2^2}} + r_h^2} \qquad r_1 = \sqrt{\frac{r_k^2\left(1 + x_0^2 + x_1^2\right)}{\boxed{u_2^2}} + r_h^2}$$

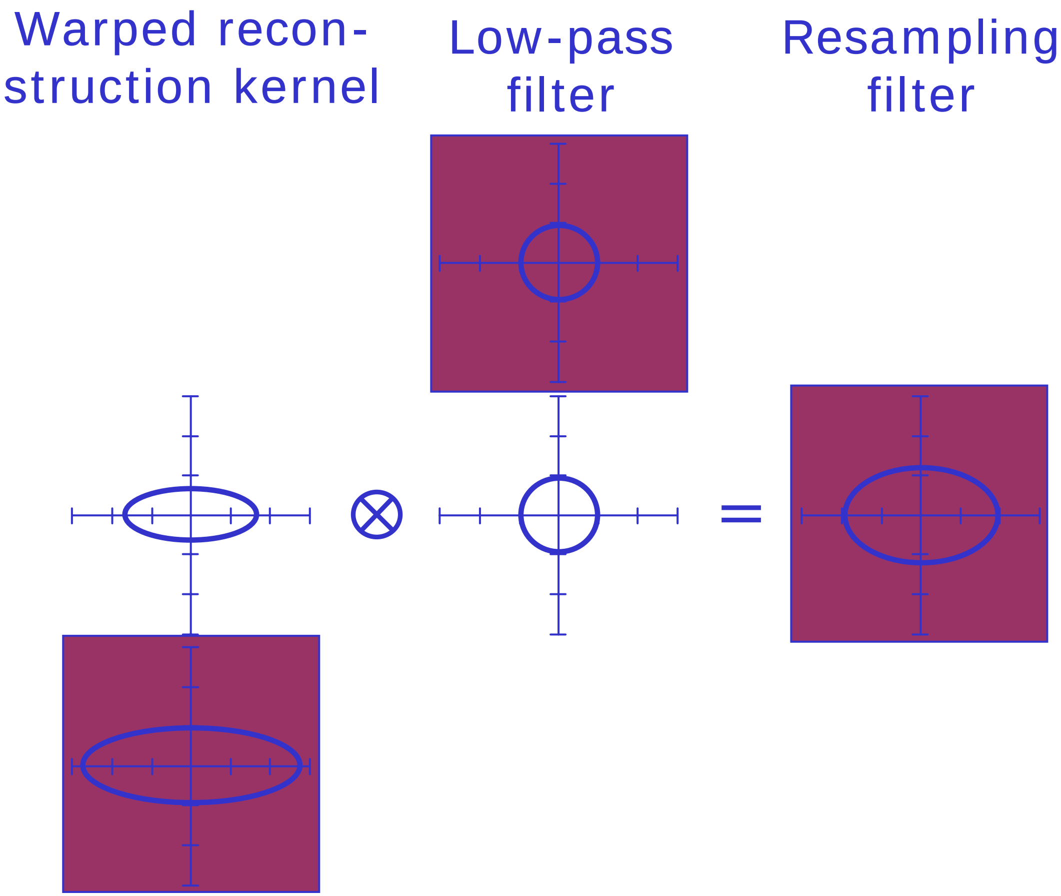
r_h Low-pass filter radius

r_k Reconstruction filter radius

u_2 Distance to the view plane

Note that $1 + x_0^2 + x_1^2 \in \left(1.0,\ 1.0 + Constant\right)$

Adaptive EWA Filtering



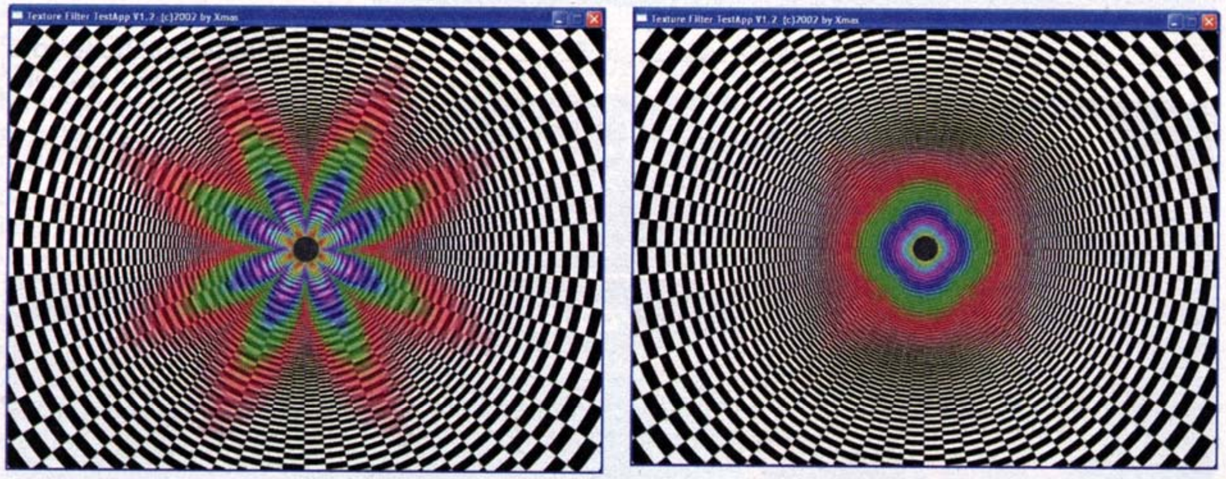
if $u_2 > A$
use low-pass filter

if $A < u_2 < B$
use EWA filter

if $u_2 < B$
use reconstruction filter

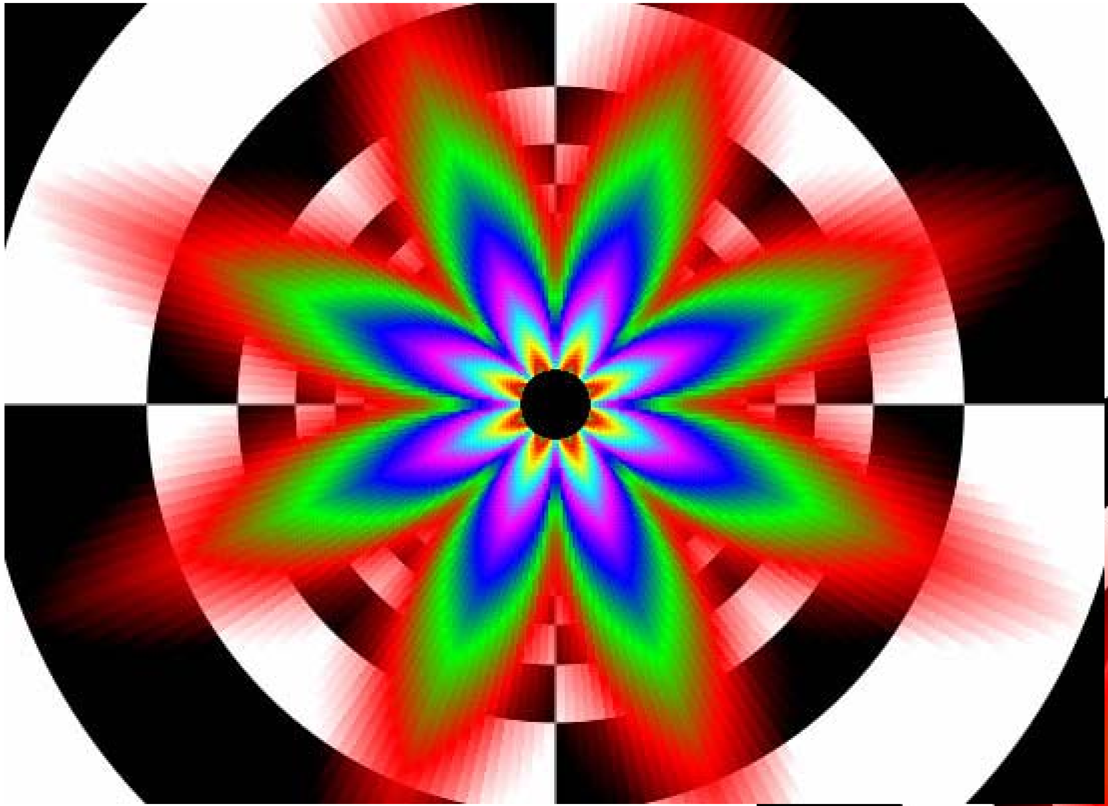
Anisotropic Filtering

- **Footprint Assembly on GPUs**
  - Integration Across Footprint of Pixel
  - HW: Choose samples that best approximate footprint
  - Weighted average of samples

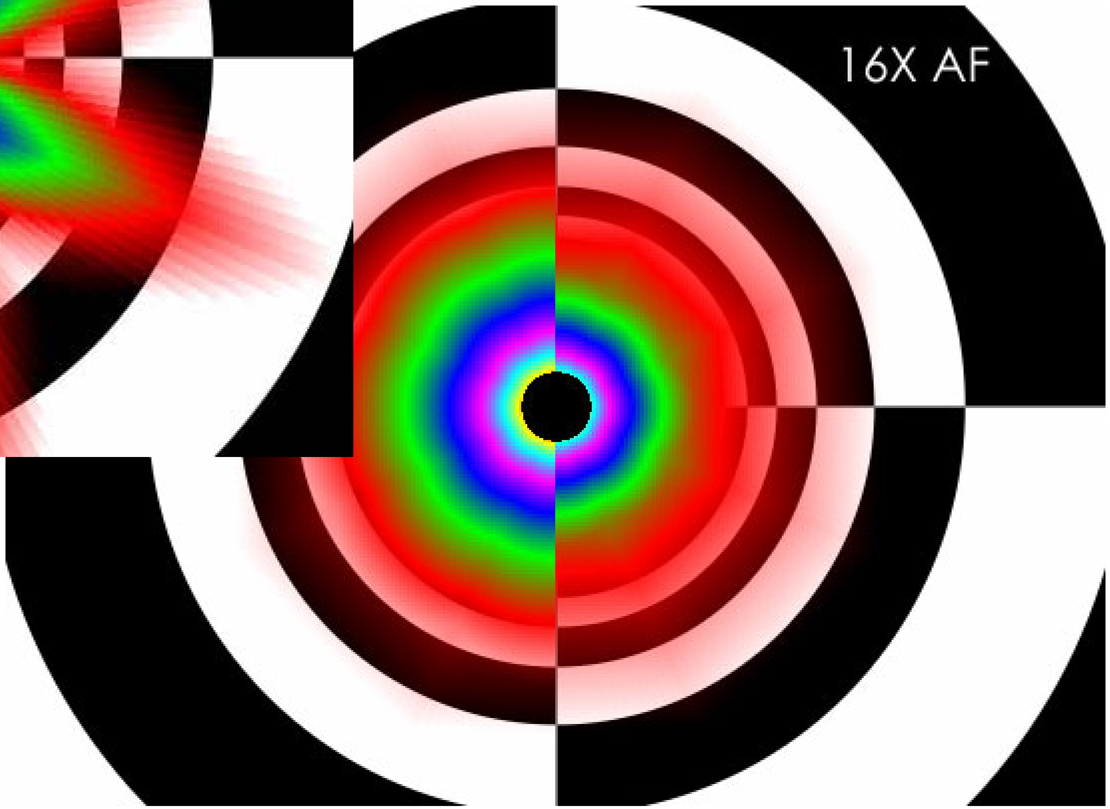


In die Röhre geblickt: ATI verwendet bei anisotroper Filterung häufig schon dicht beim Betrachter die detailverminderte, rot dargestellte Texturstufe (links). Nvidia schaltet erst später auf die erste Verkleinerungsstufe um (rechts).

Texture Filtering in Hardware



G70



G80

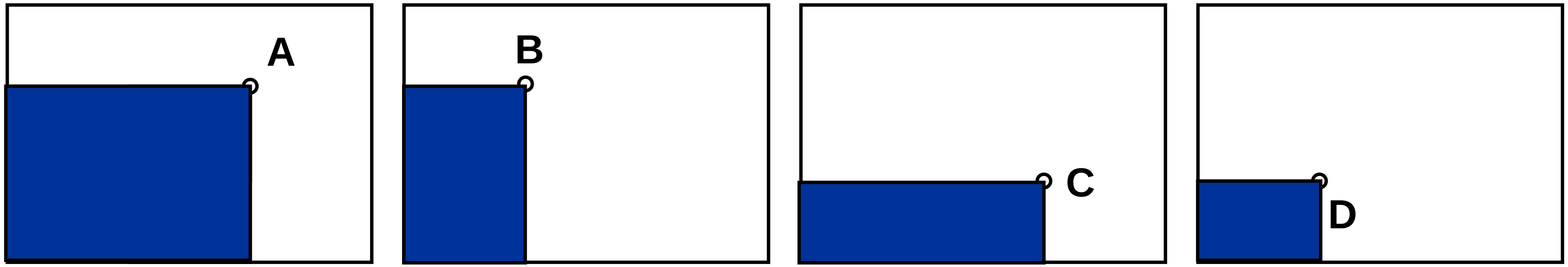
Source: Anandtech, © 2006

Filtering – Texture Minification

- **Direct convolution methods are slow**
  - A pixel pre-image can be arbitrarily large along silhouettes or at the horizon of a textured plane
  - Horizon pixels can require averaging over thousands of texture pixels
  - Texture filtering cost grows in proportion to projected texture area
- Speed up
  - **The texture can be prefiltered so that during rendering only a few samples will be accessed for each screen pixel**
- Two data structures are commonly used for prefiltering:
  - Integrated arrays (***summed area tables***)
  - Image pyramids (***mipmap***s) Space-variant filtering

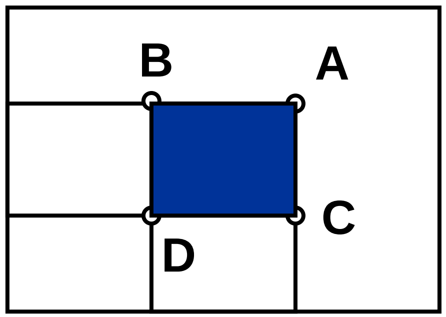
Summed Area Tables

- **Per texel, store sum from (0, 0) to (u, v)**



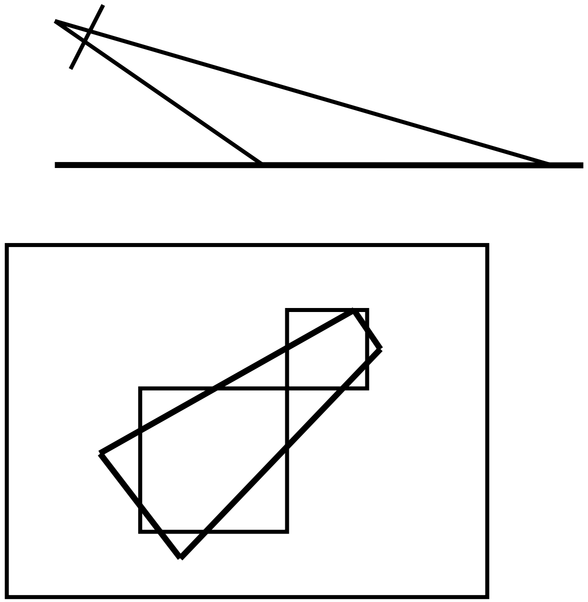
- **Many bits per texel (sum !)**
- **Evaluation of 2D integrals in constant time!**

$$\int_{Bx}^{Ax}\int_{Cy}^{Ay} I(x,y)\,dx\,dy = A - B - C + D$$



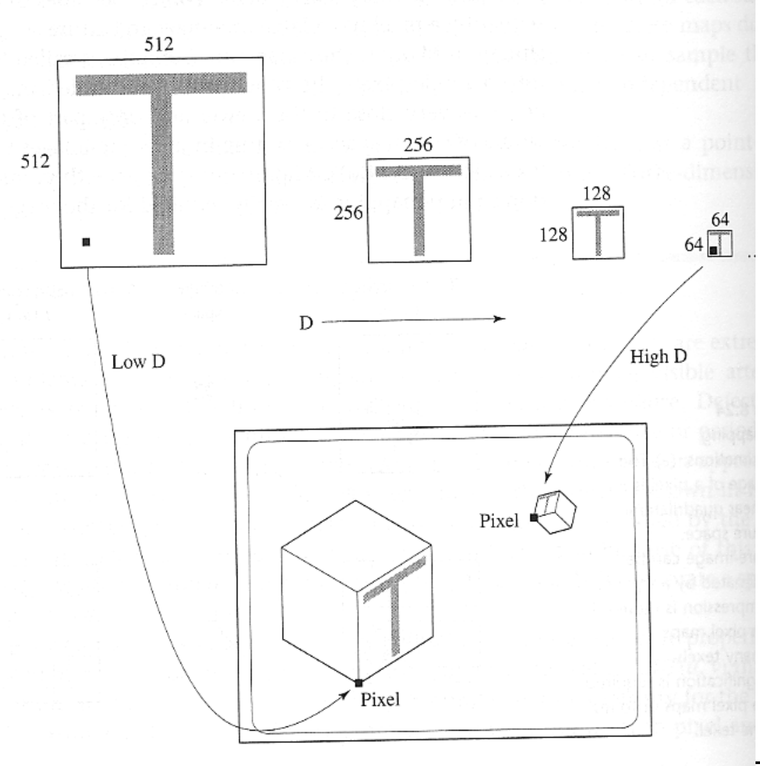
Integrated Arrays

- **Footprint assembly**
  - Good for space variant filtering
    - e.g. inclined view of terrain
  - Approximation of the pixel area by rectangular texel-regions
  - The more footprints the better accuracy
  - Often fixed number of texels because of economical reasons



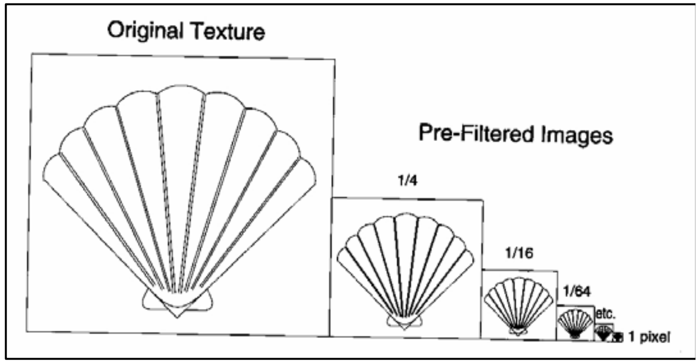
MipMapping

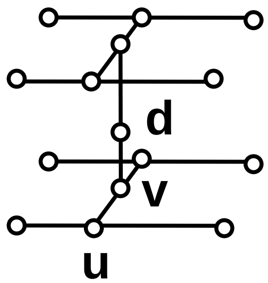
- **Texture available in multiple resolutions**
  - Pre-processing step
- **Rendering: select appropriate texture resolution**
  - Selection is usually per pixel !!
  - Texel size(n) < extent of pixel footprint < texel size(n+1)

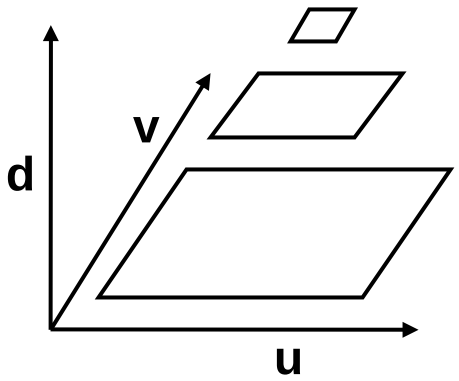


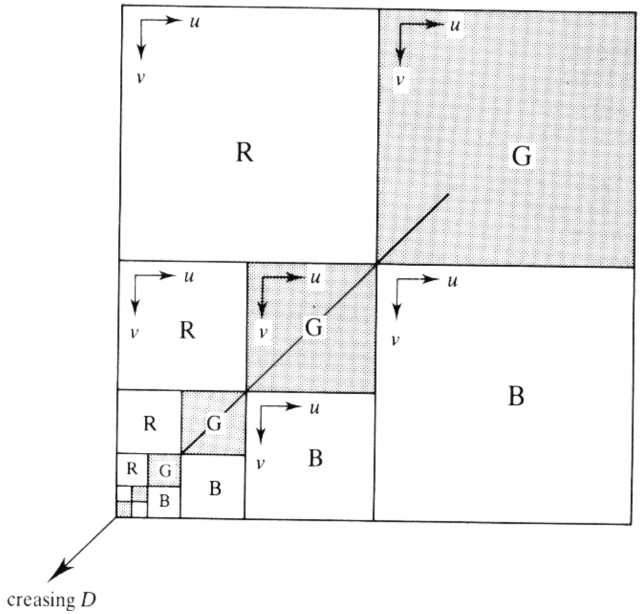
MipMapping II

- **Multum In Parvo (MIP): much in little**
- **Hierarchical resolution pyramid**
  - Repeated averaging over 2x2 texels
- **Rectangular arrangement (RGB)**
- **Reconstruction**
  - Tri-linear interpolation of 8 nearest texels

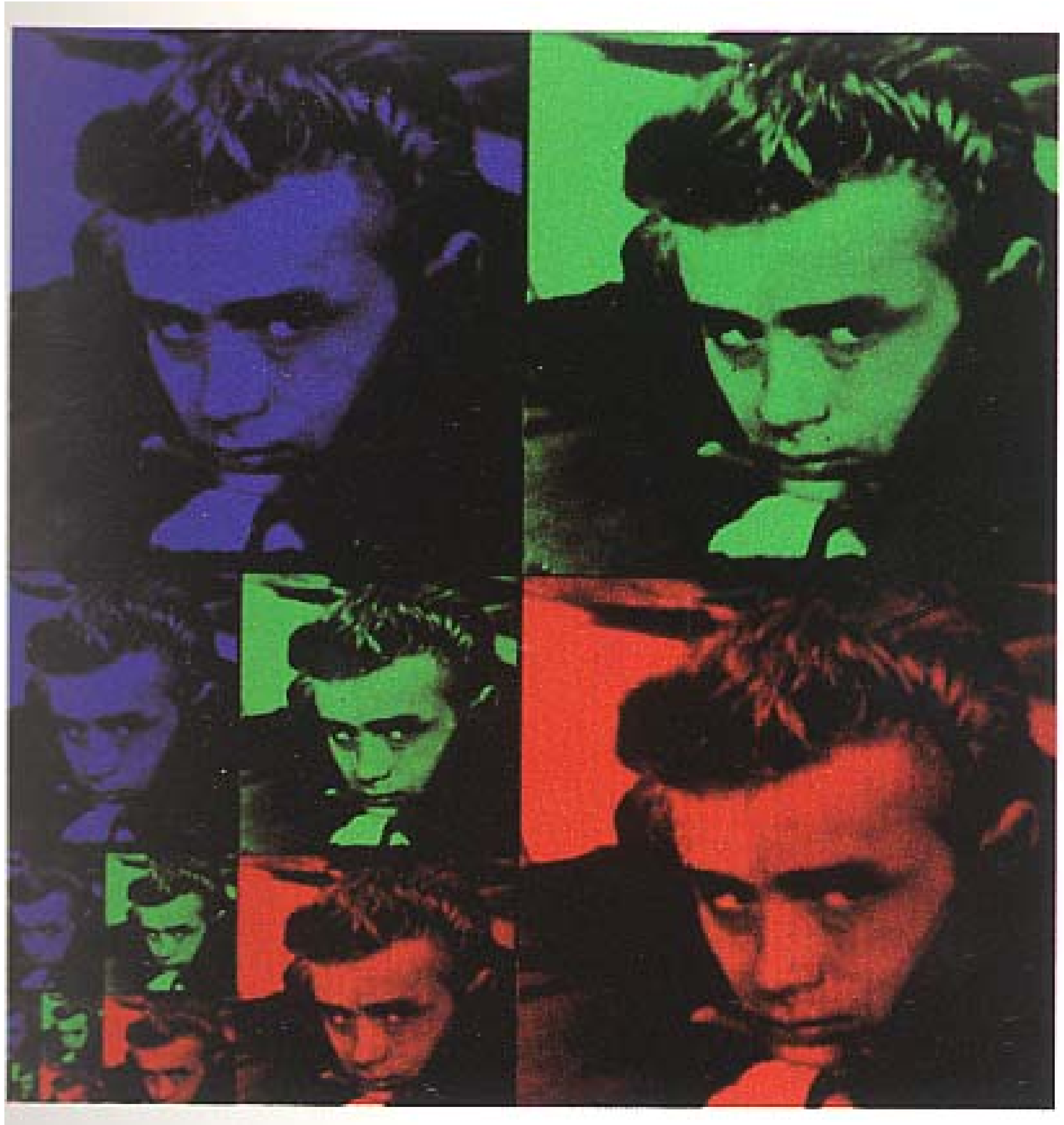








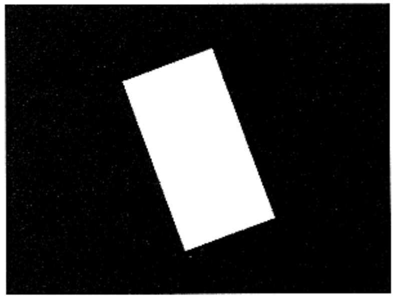
MipMap Example



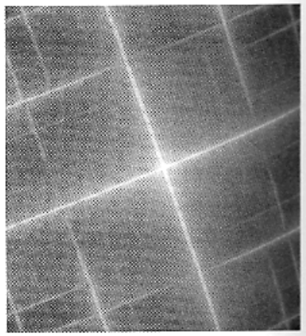
MipMaps

- **Why is MipMapping sometimes faster?**
  - Bottleneck is memory bandwidth
  - Using of texture caches
  - Texture minification required for far geometry
- **No MipMap**
  - „Random“ access to texture
  - Always 4 new texels
- **MipMap**
  - Next pixel at about one texel distance (1:1 mapping)
  - Access to 8 texels at a time, but
    - Most texels are still in the cache

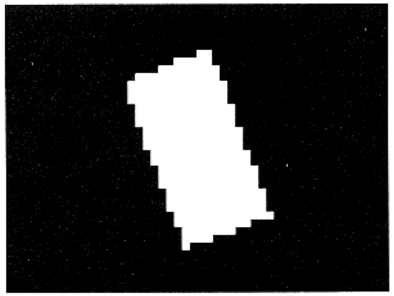
Aliasing



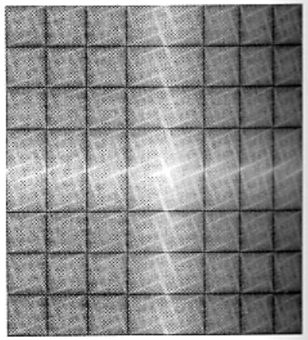
(a) Simulation of a perfect line



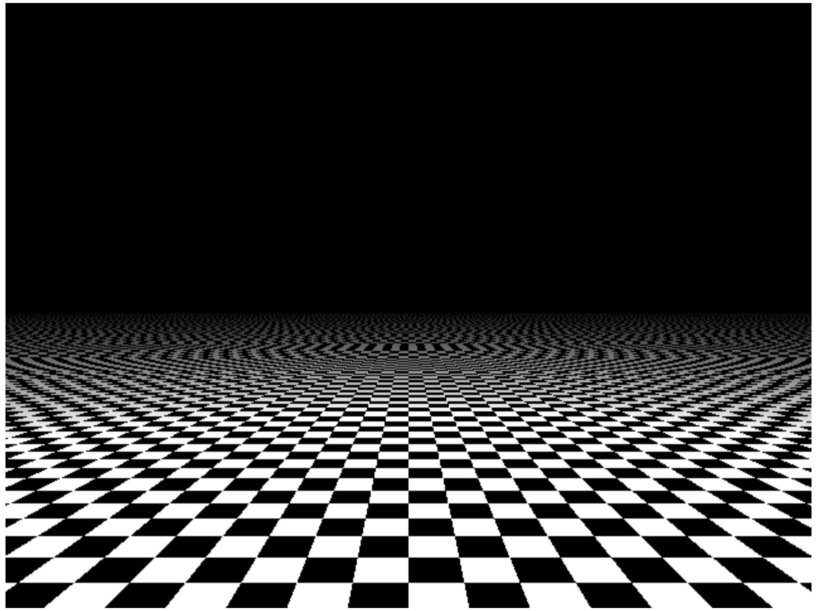
(b) Fourier transform of (a)

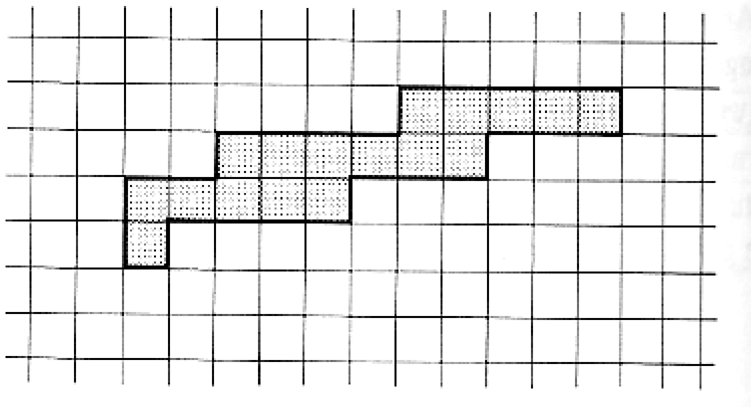


(c) Simulation of a jagged line



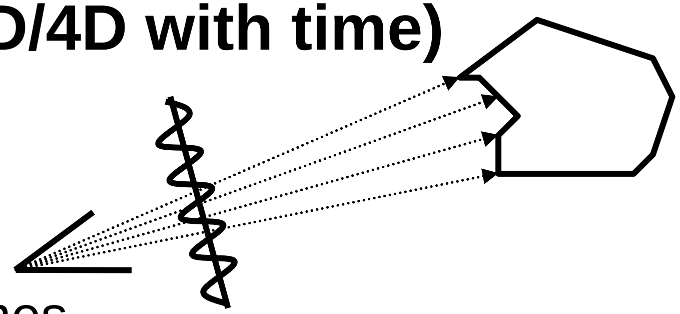
(d) Fourier transform of (c)



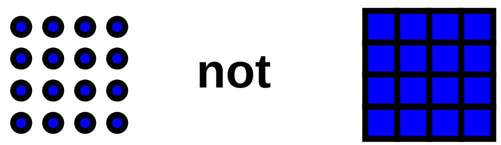


The Digital Dilemma

- **Nature: continuous signal (2D/3D/4D with time)**
  - Defined at every point



- **Acquisition: sampling**
  - Rays, pixel/texel, spectral values, frames, ...
- **Representation: discrete data**
  - Discrete points, discretized values



- **Reconstruction: filtering**
  - Mimic continuous signal
- **Display and perception: faithful**
  - Hopefully similar to the original signal, no artifacts

Sensors

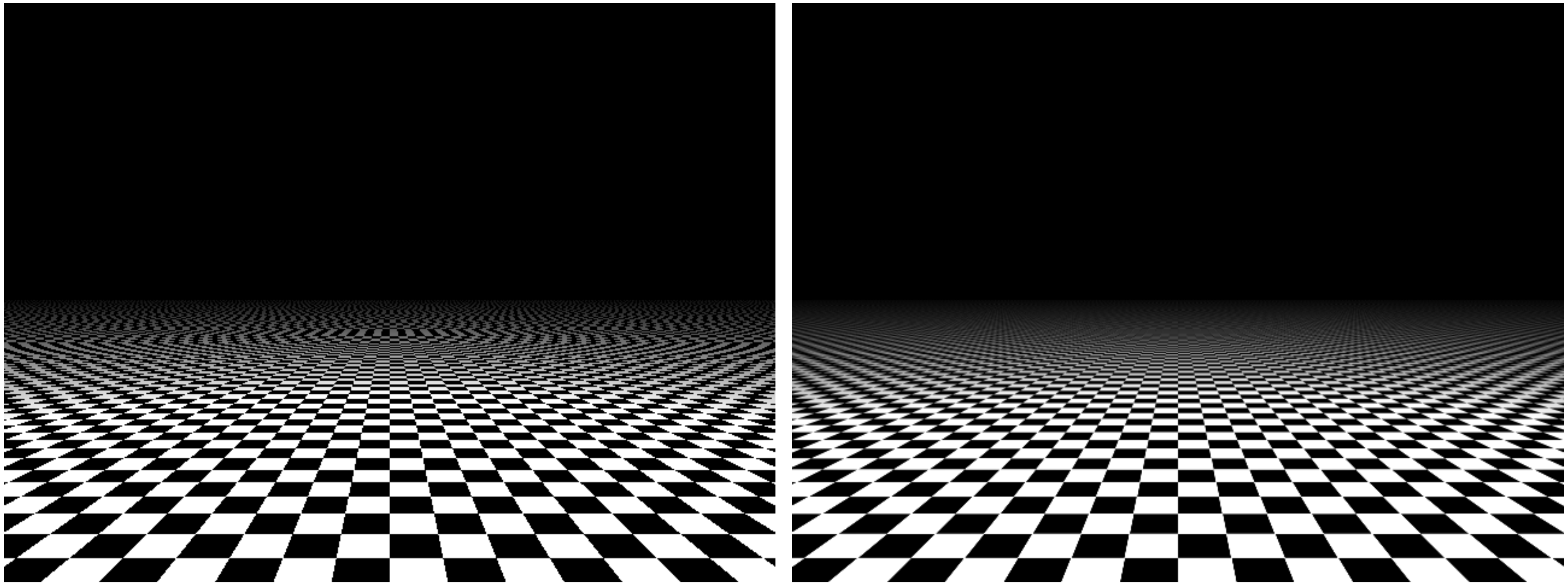
- **Sampling of signals**
  - Conversion of a continuous signal to discrete samples by integrating over the sensor field
  - Required by physical processes

$$R(i,j) = \int_{A_{ij}} E(x,y) P_{ij}(x,y)\, dx\, dy$$

- **Examples**
  - Photo receptors in the retina
  - CCD or CMOS cells in a digital camera
- **Virtual cameras in computer graphics**
  - Integration is too expensive and usually avoided
  - Ray tracing: mathematically ideal point samples
    - Origin of aliasing artifacts !

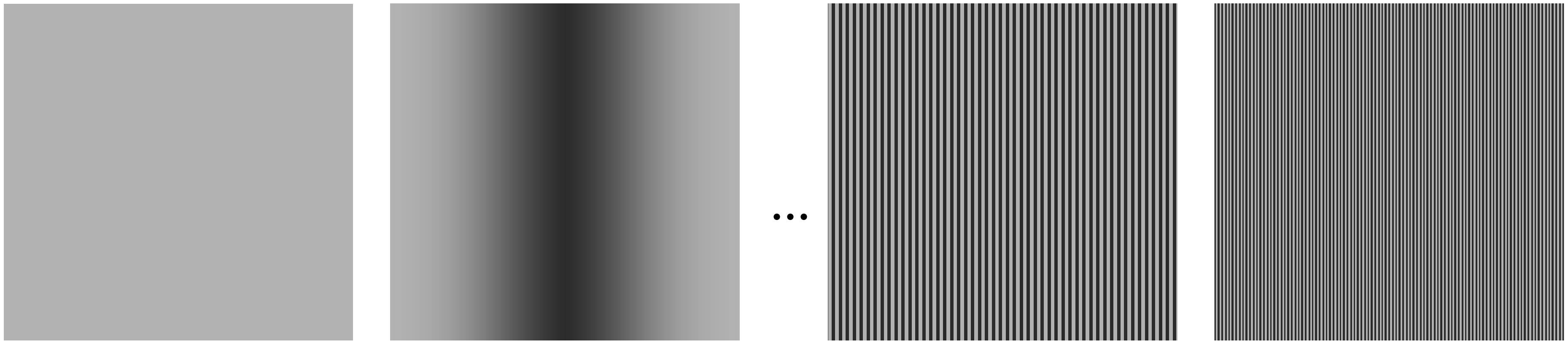
Aliasing

- **Ray tracing**
  - Textured plane with one ray for each pixel (say, at pixel center)
    - No texture filtering: equivalent to modeling with b/w tiles
  - Checkerboard period becomes smaller than two pixels
    - At the Nyquist limit
  - Hits textured plane at only one point, black or white by “chance



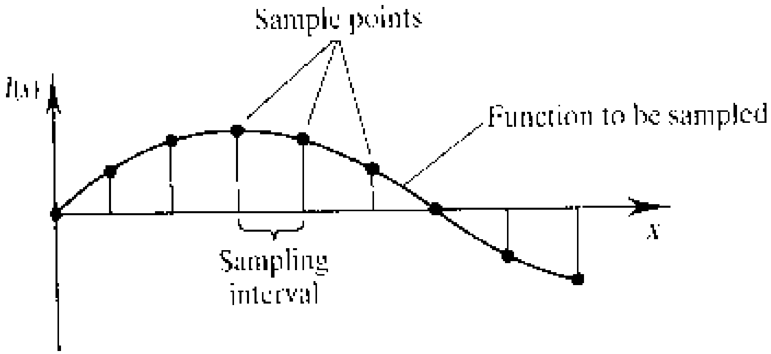
Spatial Frequency

- **Frequency: period length of some structure in an image**
  - Unit [1/pixel]
  - Range: -0.5…0.5 ($-\pi \ldots \pi$)
- **Lowest frequency**
  - Image average
- **Highest frequency: Nyquist limit**
  - In nature: defined by wavelength of light
  - In graphics: defined by image resolution

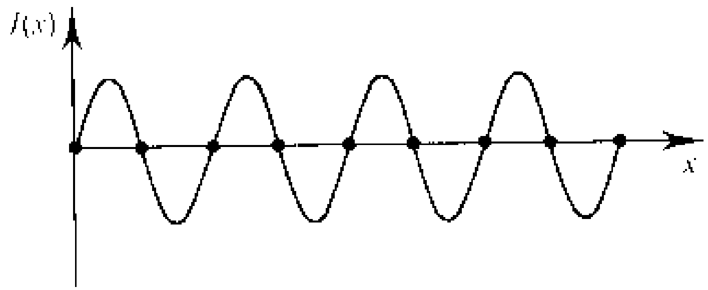


Nyquist Frequency

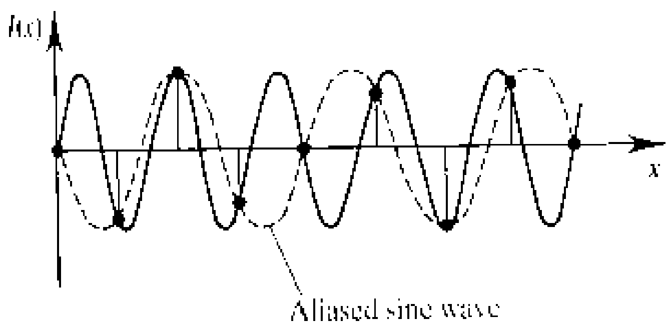
- **Highest (spatial) frequency that can be represented**
- **Determined by image resolution (pixel size)**



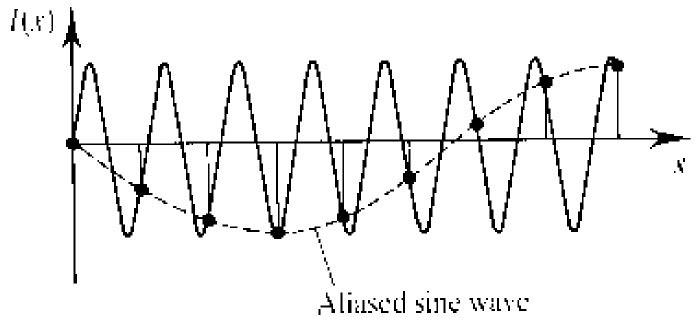
Spatial frequency < Nyquist



Spatial frequency = Nyquist
2 samples / period



Spatial frequency > Nyquist



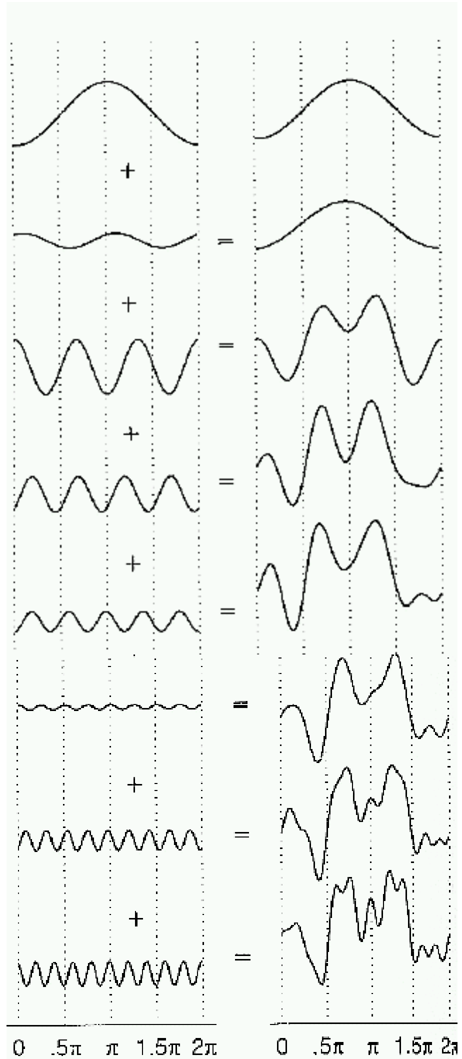
Spatial frequency >> Nyquist

Fourier Transformation

- **Any continuous function f(x) can be expressed as an integral over sine and cosine waves:**

$$F(k) = F_x[f(x)](k) = \int_{-\infty}^{\infty} f(x) e^{-2\pi ikx} dx \quad \text{Analysis}$$

$$f(x) = F_x^{-1}[F(k)](x) = \int_{-\infty}^{\infty} F(k) e^{2\pi ikx} dk \quad \text{Synthesis}$$



- **Division into even and odd parts**

$$f(x) = \frac{1}{2}[f(x) + f(-x)] + \frac{1}{2}[f(x) - f(-x)] = E(x) + O(x)$$

- **Transform of each part**

$$F[f(x)](k) = \int_{-\infty}^{\infty} E(x)\cos(2\pi kx)dx - i\int_{-\infty}^{\infty} O(x)\sin(2\pi kx)dx$$

Fourier Transformation

- **Any periodic, continuous function can be expressed as the sum of an (infinite) number of sine or cosine waves:**

$$\mathbf{f(x)=\Sigma_k\, a_k \sin(2\pi*k*x) + b_k \cos(2\pi*k*x)}$$

  - Decomposition of signal into different frequency bands
    - Spectral analysis
  - k: frequency band
    - k=0 mean value
    - k=1 function period, lowest possible frequency
    - k=1.5? not possible, periodic function f(x) = f(x+1)
    - k_{max}? band limit, no higher frequency present in signal
  - a_k,b_k : (real-valued) Fourier coefficients
    - Even function f(x)= f(-x): $a_k = 0$
    - Odd function f(x)= -f(-x): $b_k = 0$

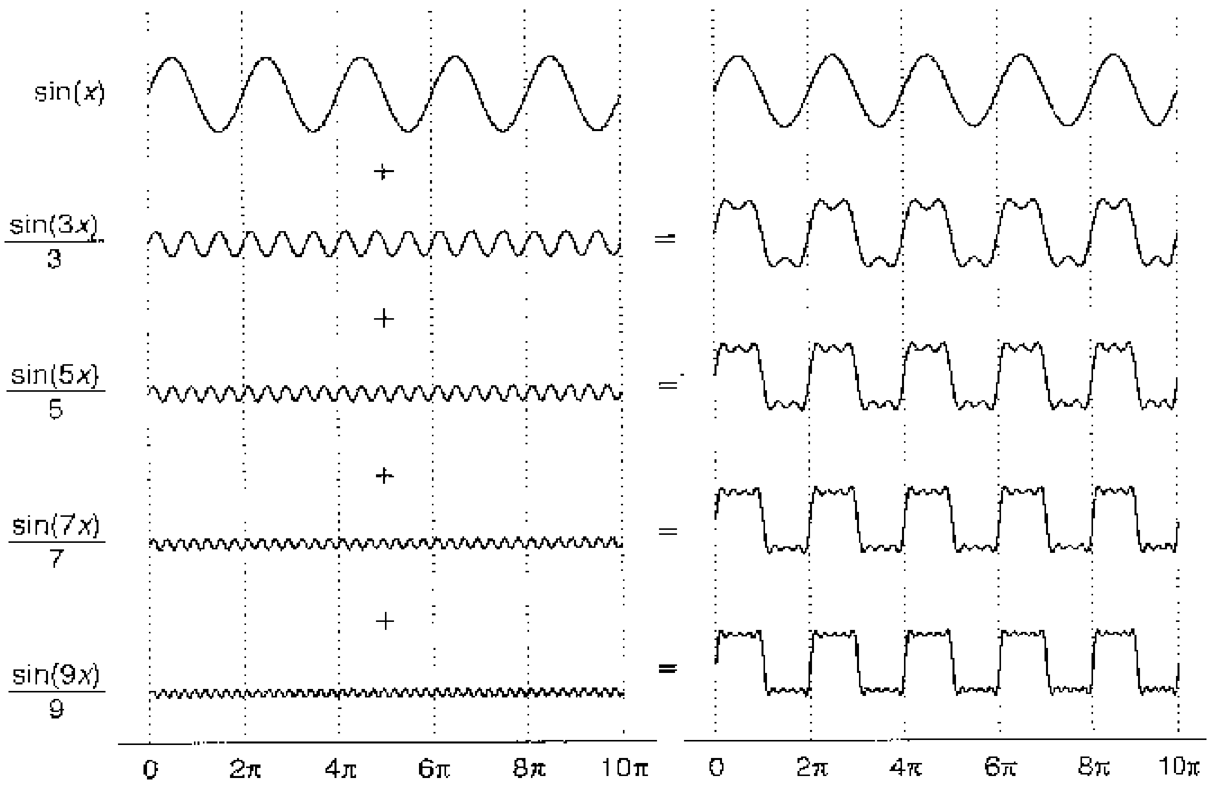
Fourier Synthesis Example

- **Periodic, uneven function: square wave**

$$f(x) = 0.5 \ \forall \ 0 < (x \bmod 2\pi) < \pi$$
$$= -0.5 \ \forall \ \pi < (x \bmod 2\pi) < 2\pi$$

$$a_k = \int \sin(k^*x)^* f(x)\,dx \qquad f(x)=\Sigma_k\, a_k \sin(k^*x)$$

- $a_0 = 0$
- $a_1 = 1$
- $a_2 = 0$
- $a_3 = 1/3$
- $a_4 = 0$
- $a_5 = 1/5$
- $a_6 = 0$
- $a_7 = 1/7$
- $a_8 = 0$
- $a_9 = 1/9$
- …



Discrete Fourier Transform

- **Equally-spaced function samples**
  - Function values known only at discrete points
    - Physical measurements
    - Pixel positions in an image !
- **Fourier Analysis**

$$\mathbf{a_k = 1/\,N\ \sum_i \sin(2\pi\ k\ i\ /\ N)\ f_i\ ,\qquad b_k = 1/\,N\ \sum_i \cos(2\pi\ k\ i\ /\ N)\ f_i}$$

  - Sum over all measurement points N
  - k=0,1,2, …, ? Highest possible frequency ?

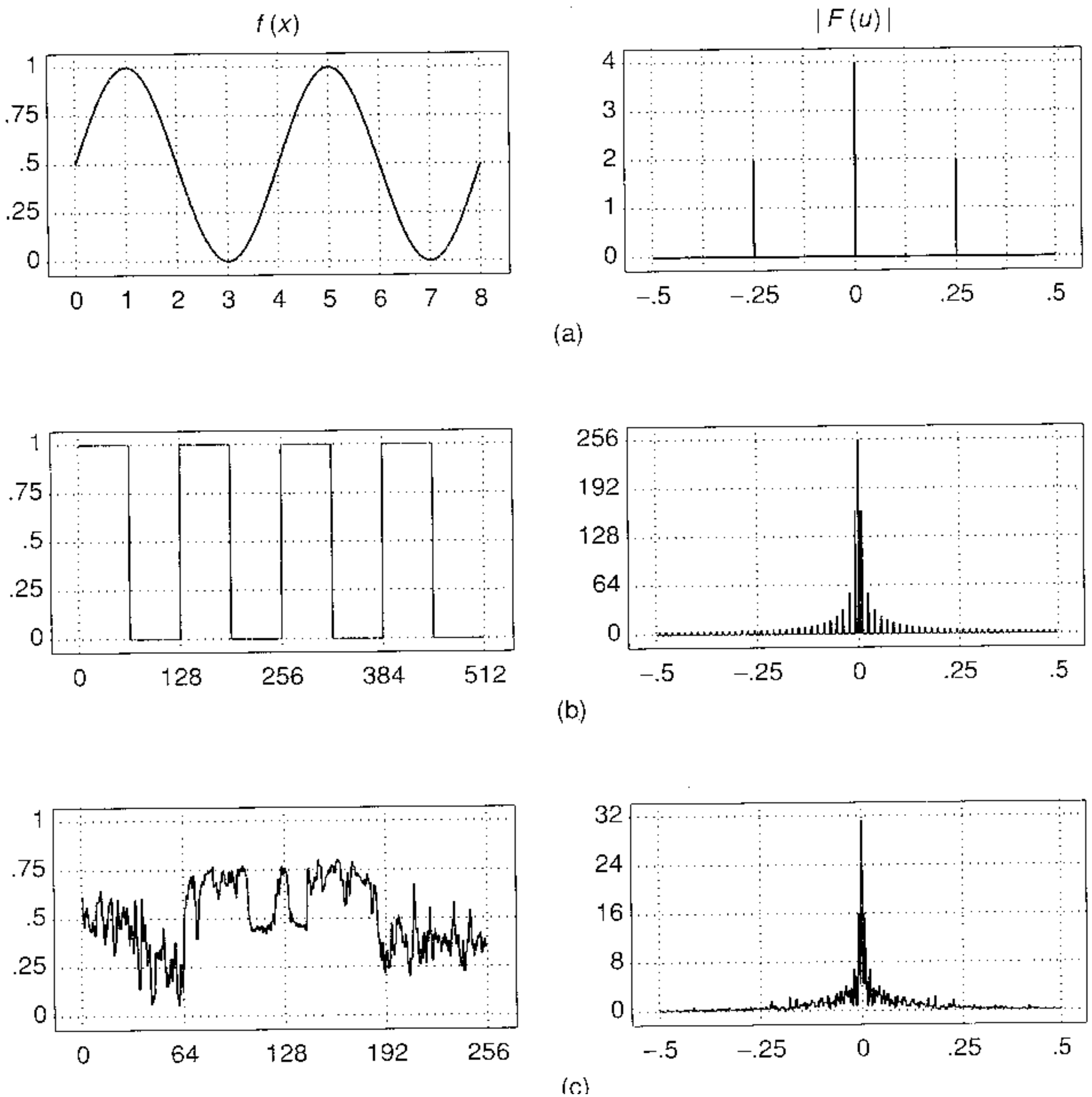
$\Rightarrow$ **Nyquist frequency**

- Sampling rate N_i
- 2 samples / period $\Leftrightarrow$ 0.5 cycles per pixel

$\Rightarrow k \leq N / 2$

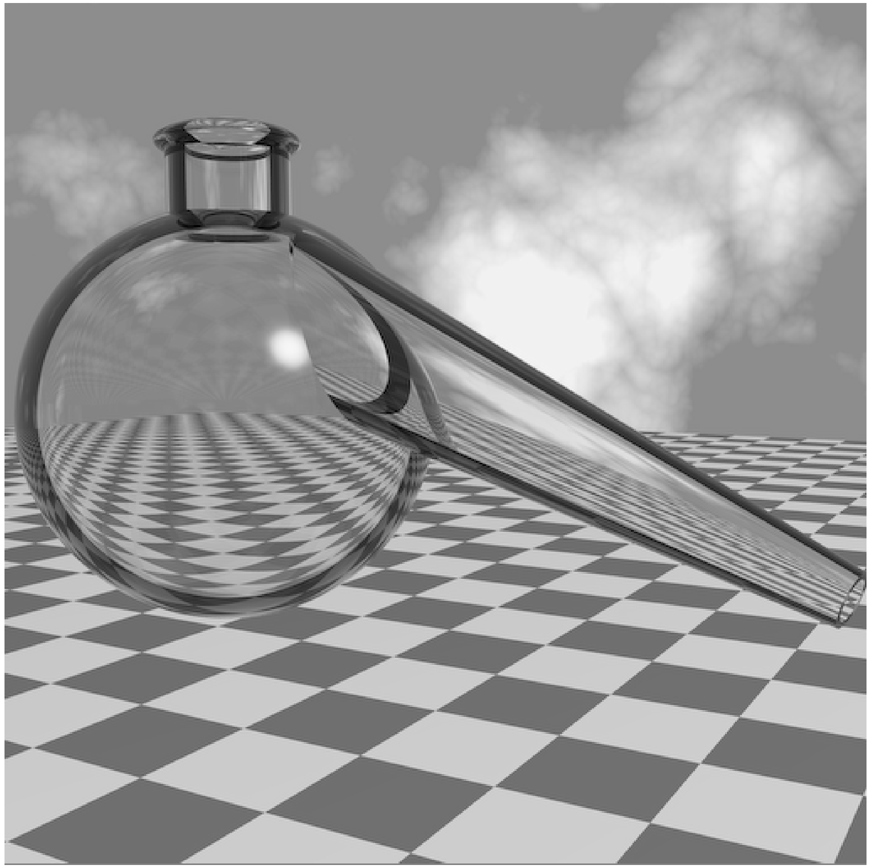
Spatial vs. Frequency Domain

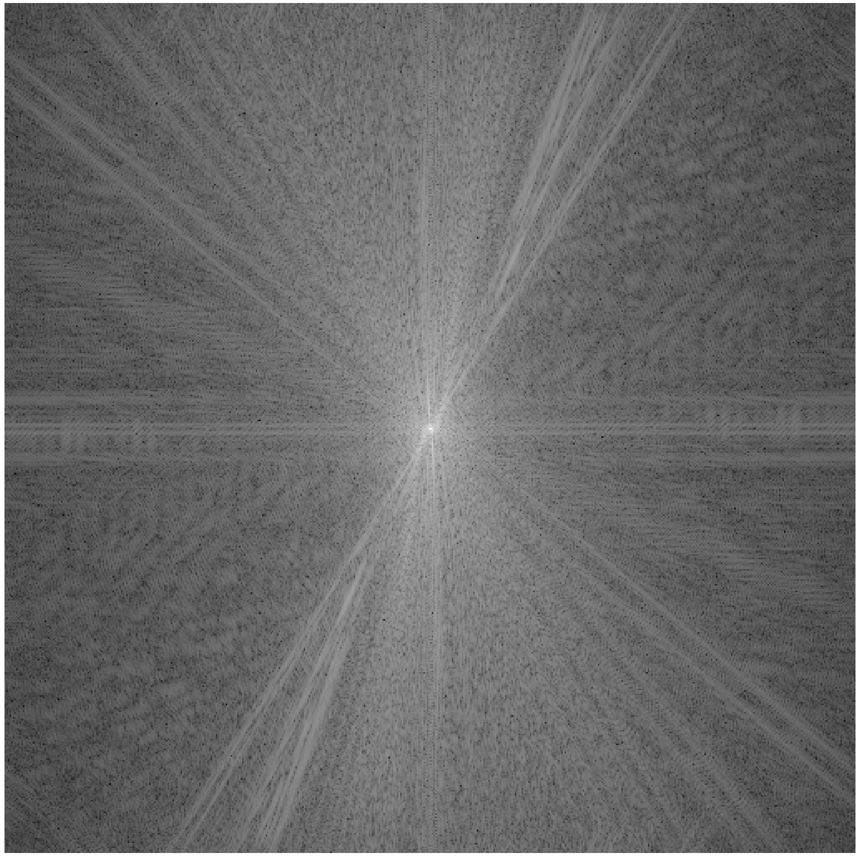
- **Examples** (pixel vs cycles per pixel)
  - Sine wave with positive offset
  - Square wave
  - Scanline of an image



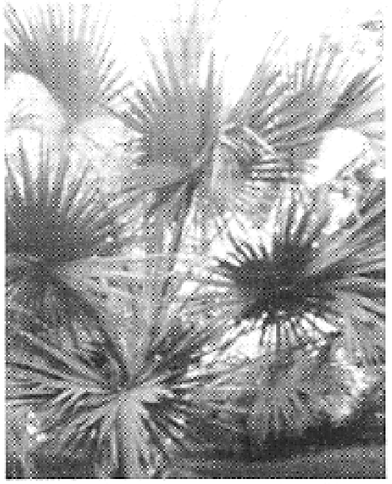
2D Fourier Transform

- 2 separate 1D Fourier transformations along x- and y-direction
- Discontinuities: orthogonal direction in Fourier domain !

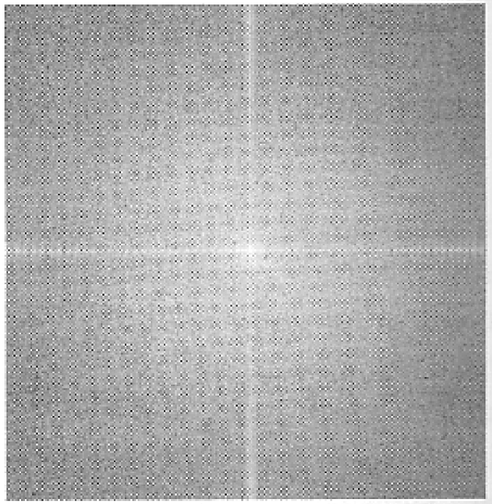




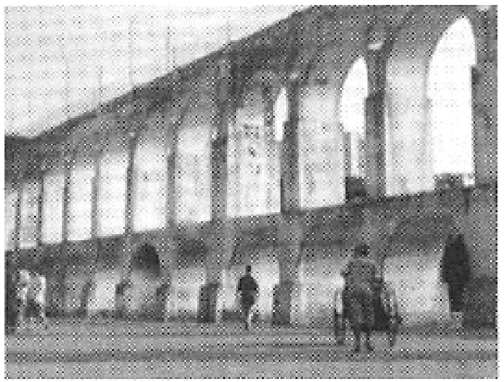
2D Fourier Transforms



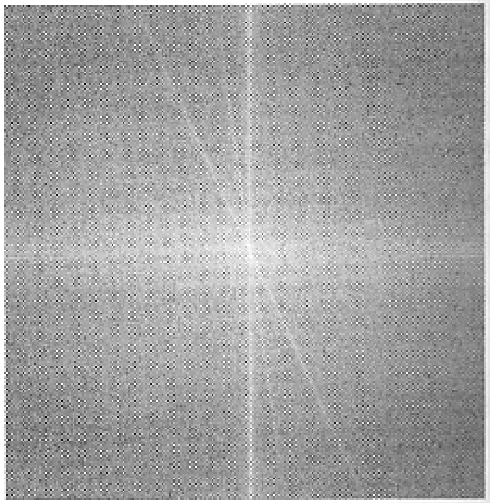
(a) Bush



Fourier transform $|F(u, v)|$



(b) Arcos da Lapa
(Rio de Janeiro)



Fourier transform $|F(u, v)|$

Spatial vs. Frequency Domain

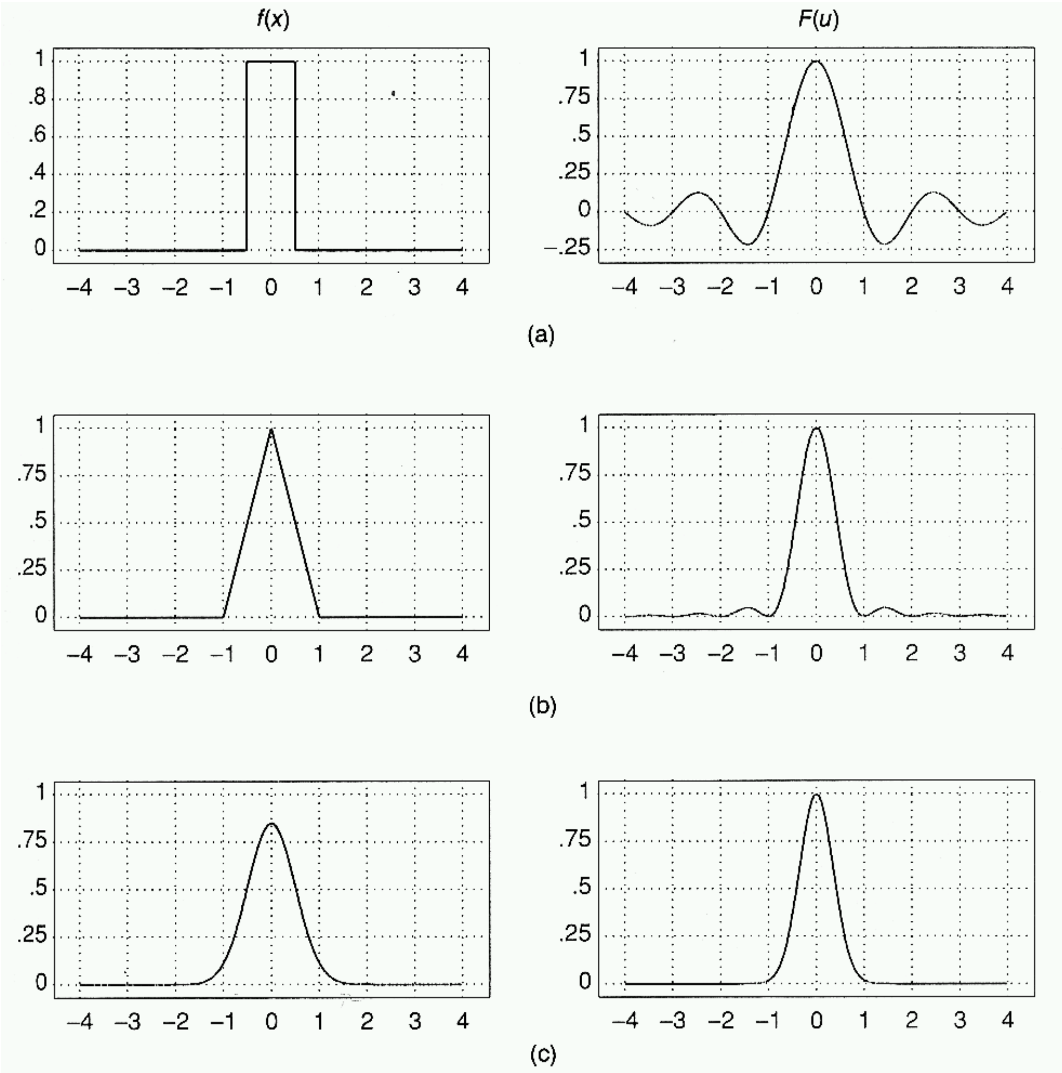
- **Important basis functions**
  - Box ←→ sinc

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

$$\operatorname{sinc}(0) = 1$$

$$\int \operatorname{sinc}(x)dx = 1$$

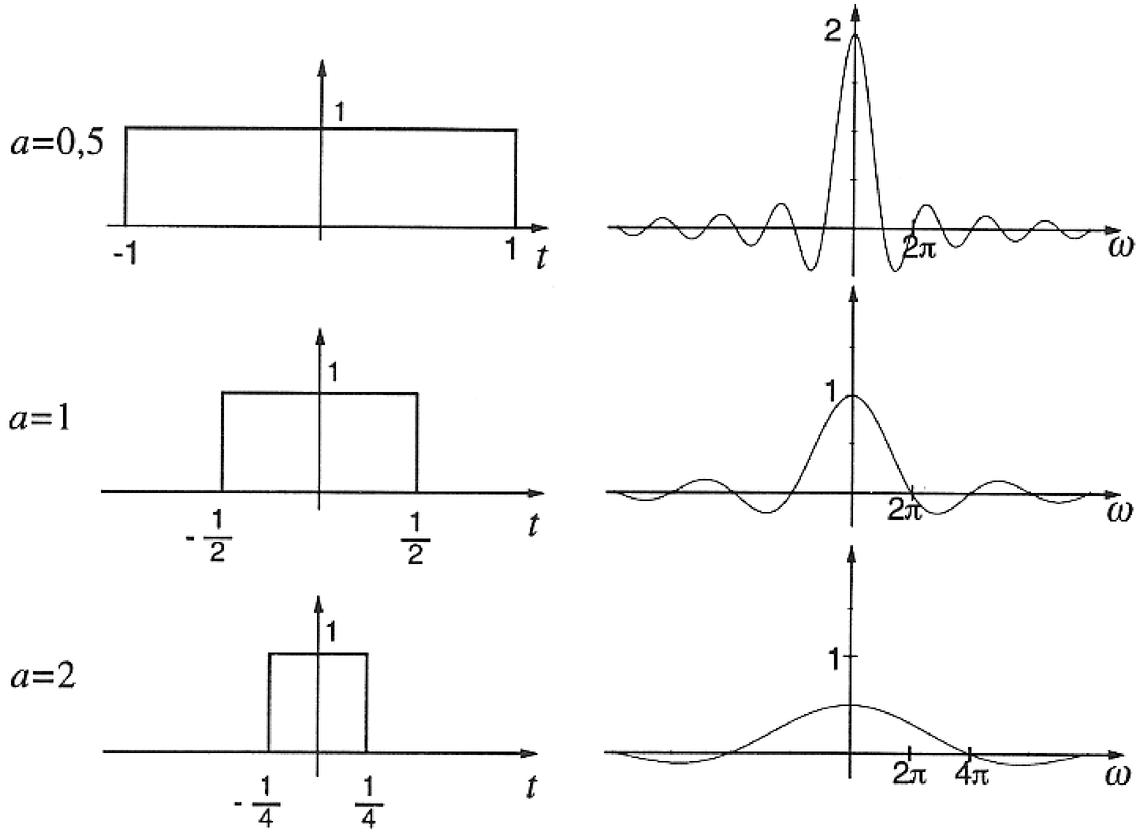
    - Wide box → small sinc
    - Negative values
    - Infinite support
  - Triangle ←→ sinc^2
  - Gauss ←→ Gauss



Spatial vs. Frequency Domain

- **Transform behavior**
- **Example: box function**
  - Fourier transform: sinc
  - Wide box: narrow sinc
  - Narrow box: wide sinc

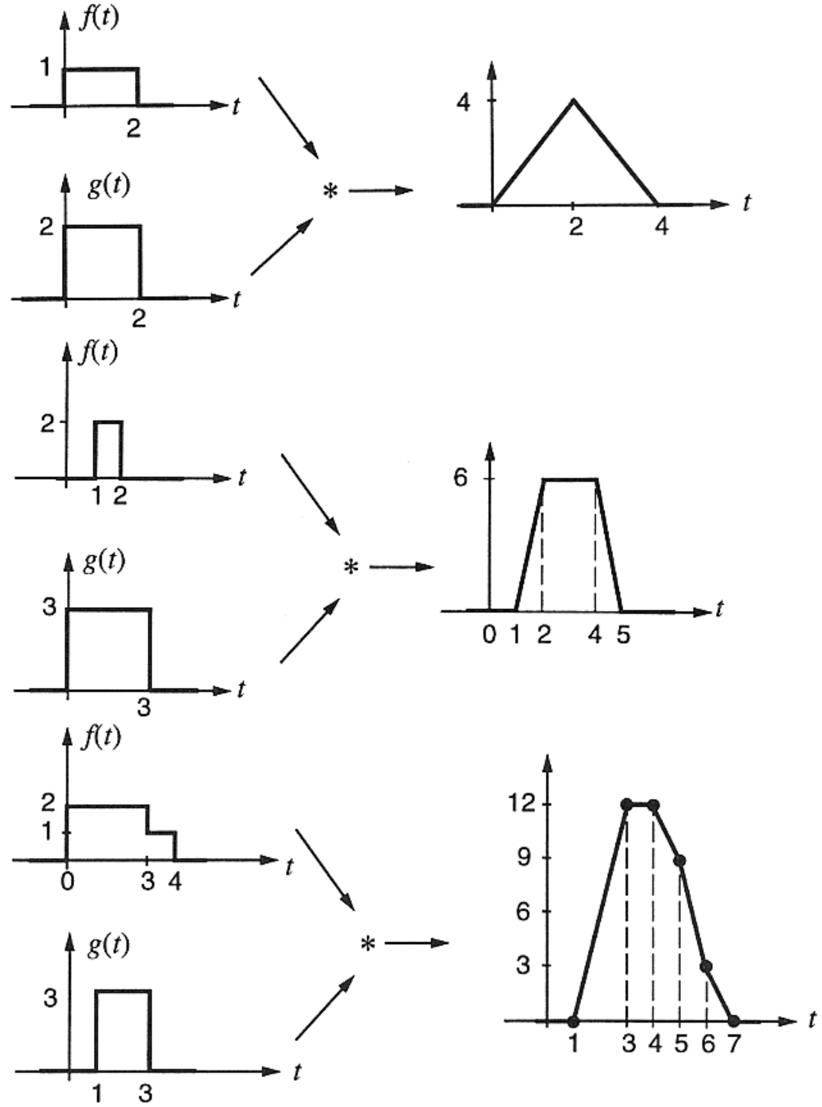
$$\mathrm{rect}(at) \quad \circ\!\!-\!\!\bullet \quad \frac{1}{|a|}\,\mathrm{si}\left(\frac{\omega}{2a}\right).$$



Convolution

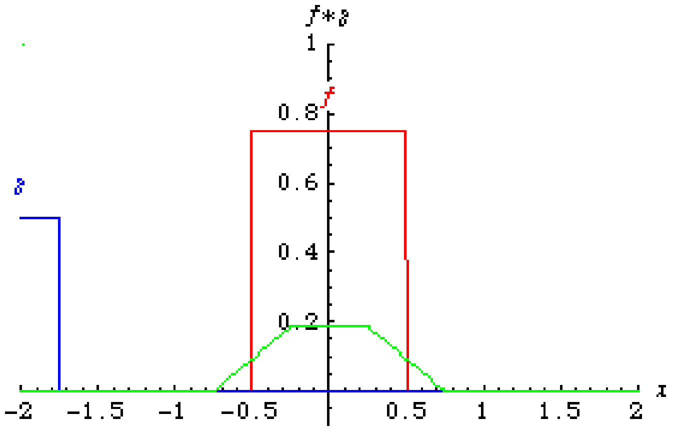
$$f(x)\otimes g(x)=\int_{-\infty}^{\infty} f(\tau)g(x-\tau)\,d\tau$$

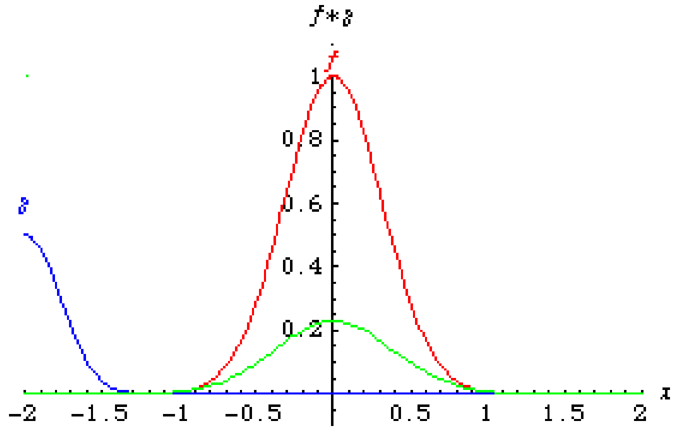
- **Two functions** f, g
- **Shift one function against the other by** x
- **Multiply function values**
- **Integrate overlapping region**
- **Numerical convolution: Expensive operation**
  - For each x: integrate over non-zero domain



Convolution

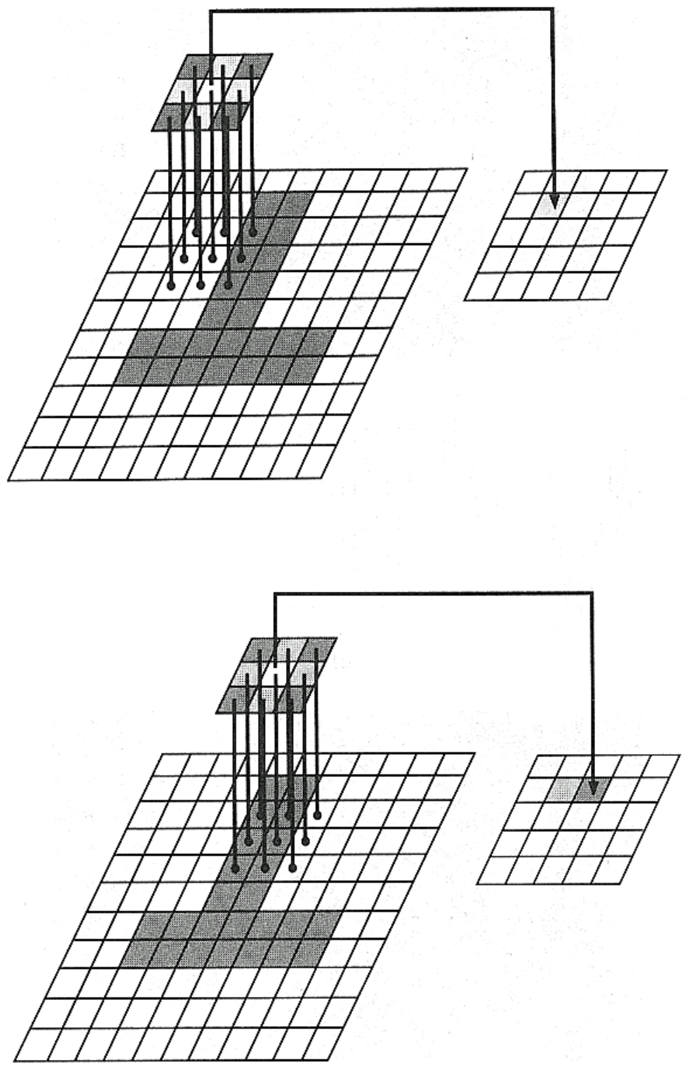
- **Examples**
  - Box functions
  - Gauss functions





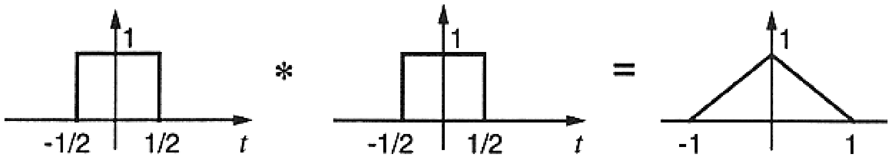
Convolution and Filtering

- **Technical Realization**
  - In image domain
  - Pixel mask with weights
  - OpenGL: Convolution extension
- **Problems (e.g. sinc)**
  - Large filter support
    - Large mask
    - A lot of computation
  - Negative weights
    - Negative light?



Convolution Theorem

- **Convolution in image domain: multiplication in Fourier domain**
- **Convolution in Fourier domain: multiplication in image domain**
  - Multiplication much cheaper than convolution !

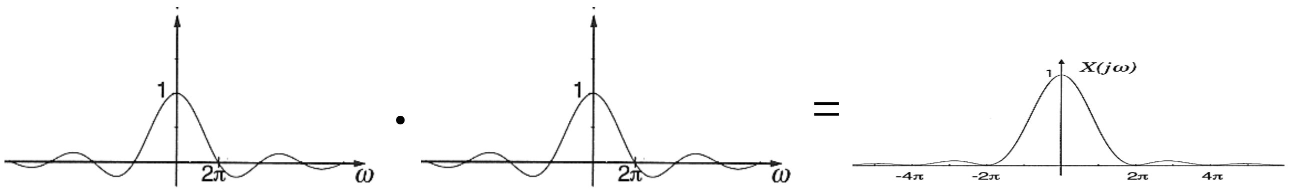


$$x(t) = \begin{cases} t+1 & \text{für } -1 \leq t < 0 \\ -t+1 & \text{für } 0 \leq t < 1 \\ 0 & \text{sonst} \end{cases}$$

$$\mathrm{rect}(t) \ast \mathrm{rect}(t) = x(t)$$

$$\circ\!\!-\!\!\bullet \qquad \circ\!\!-\!\!\bullet \qquad \circ\!\!-\!\!\bullet$$

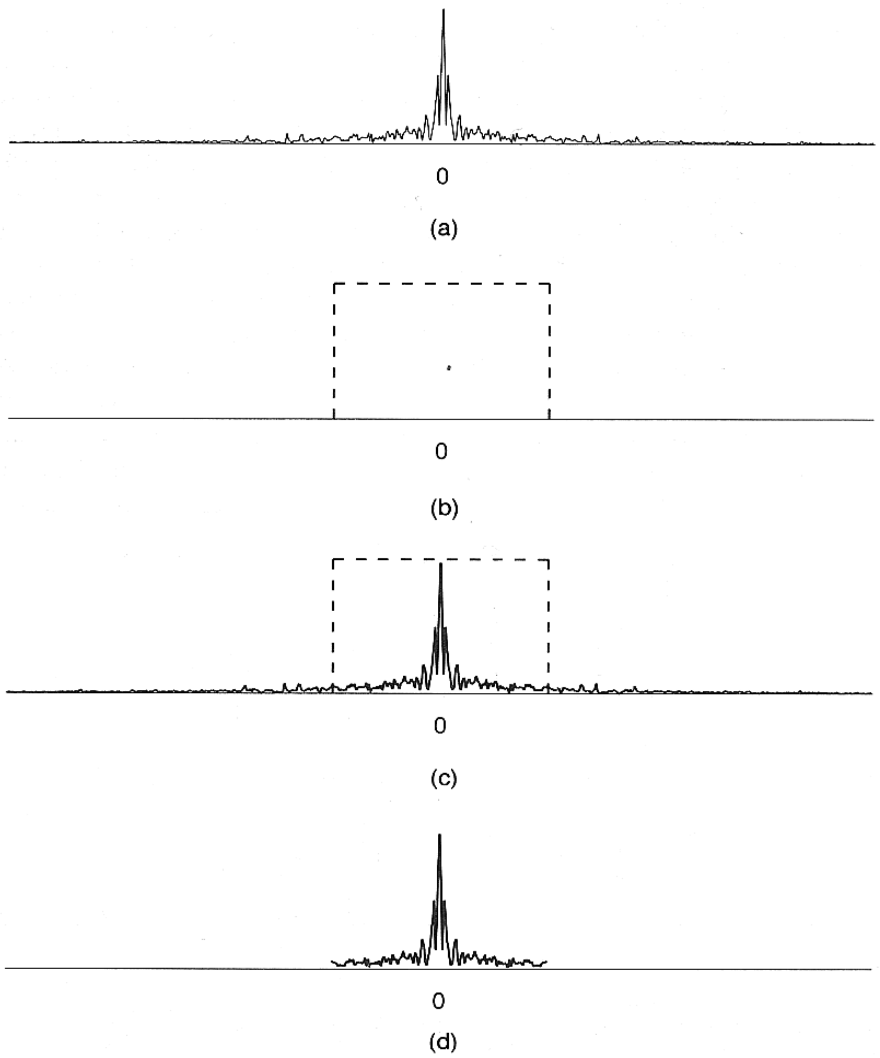
$$\mathrm{si}\left(\frac{\omega}{2}\right) \cdot \mathrm{si}\left(\frac{\omega}{2}\right) = X(j\omega)\,.$$



$$X(j\omega) = \mathrm{si}^2\left(\frac{\omega}{2}\right).$$

Filtering

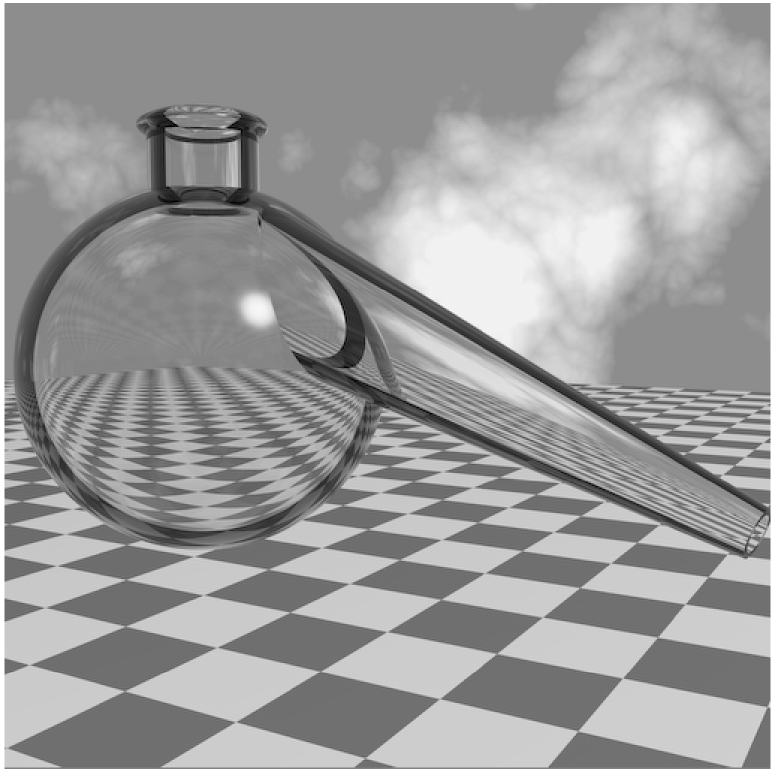
- **Low-pass filtering**
  - Convolution with sinc in spatial domain, or
  - Multiplication with box in frequency domain
- **High-pass filtering**
  - Only high frequencies
- **Band-pass filtering**
  - Only intermediate

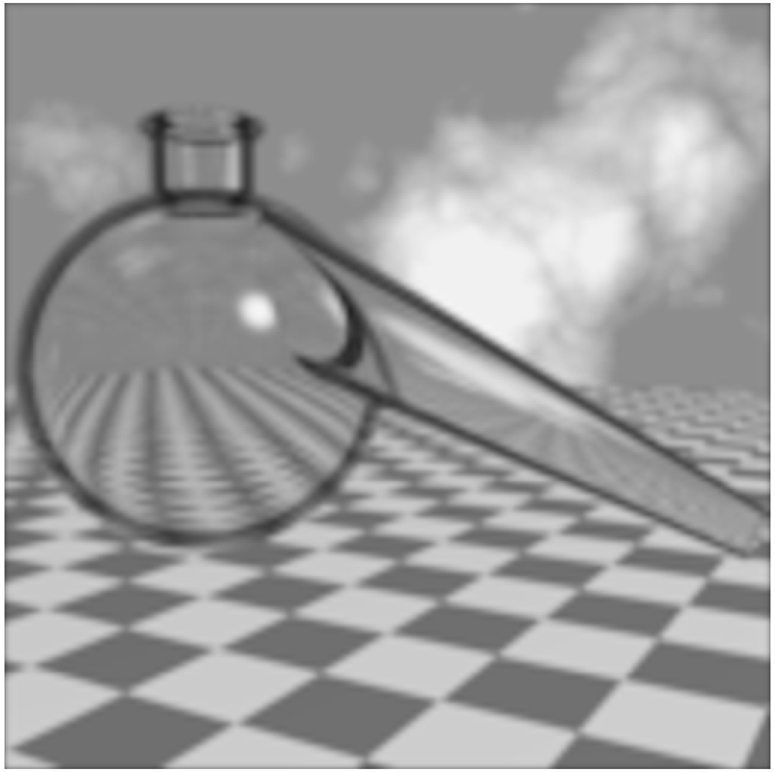


Low-pass filtering in frequency domain: multiplication with box

Low-Pass Filtering

- **„Blurring“**





High-Pass Filtering

- **Enhances discontinuities in image**
  - Useful for edge detection