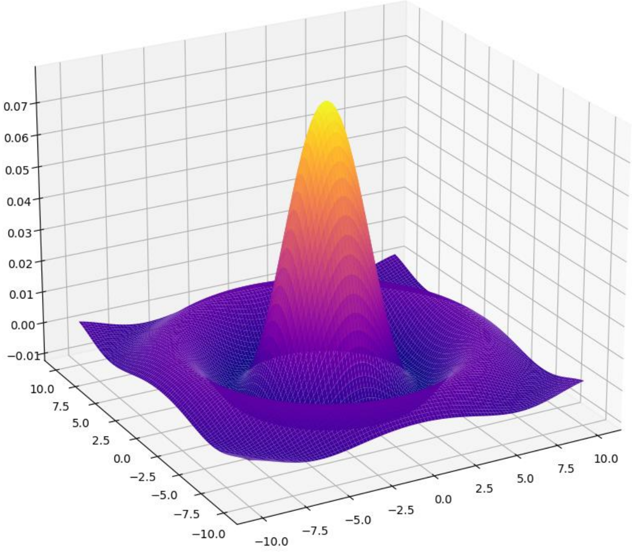
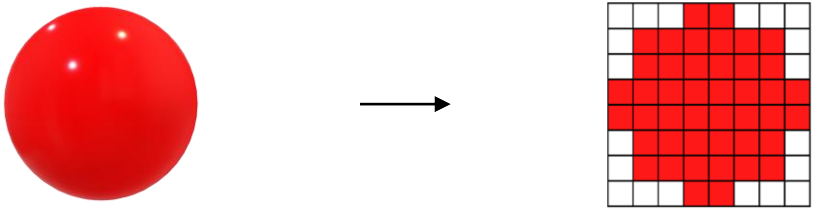
Processing Signals

Prof. Dr. Markus Gross



ETH zürich

Digital Image Formation in Computer Graphics



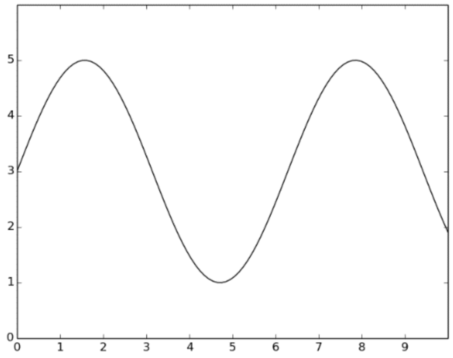
- Main goal of Computer Graphics is to generate 2D images
- Images: continuous 2D functions (signals) that can be
  - Monochrome
  - Color

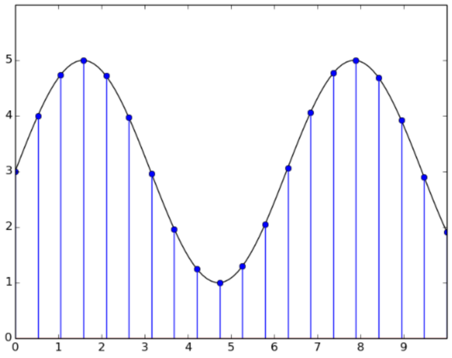
Digital Image Formation in Computer Graphics

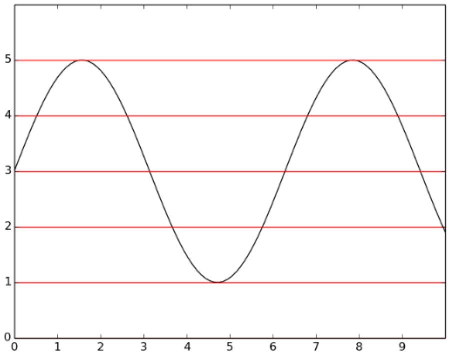
- These functions are represented by a 2D set of discrete samples, i.e., $x, y \in \mathbb{N}$. (x, y) is called a pixel
- The sampling rate determines the resolution of the digital image

Sampling vs. Quantization

- Digitizing an analog signal involves two components
  - Sampling: discretizing the domain of f
  - Quantization: discretizing the image (math.) of f

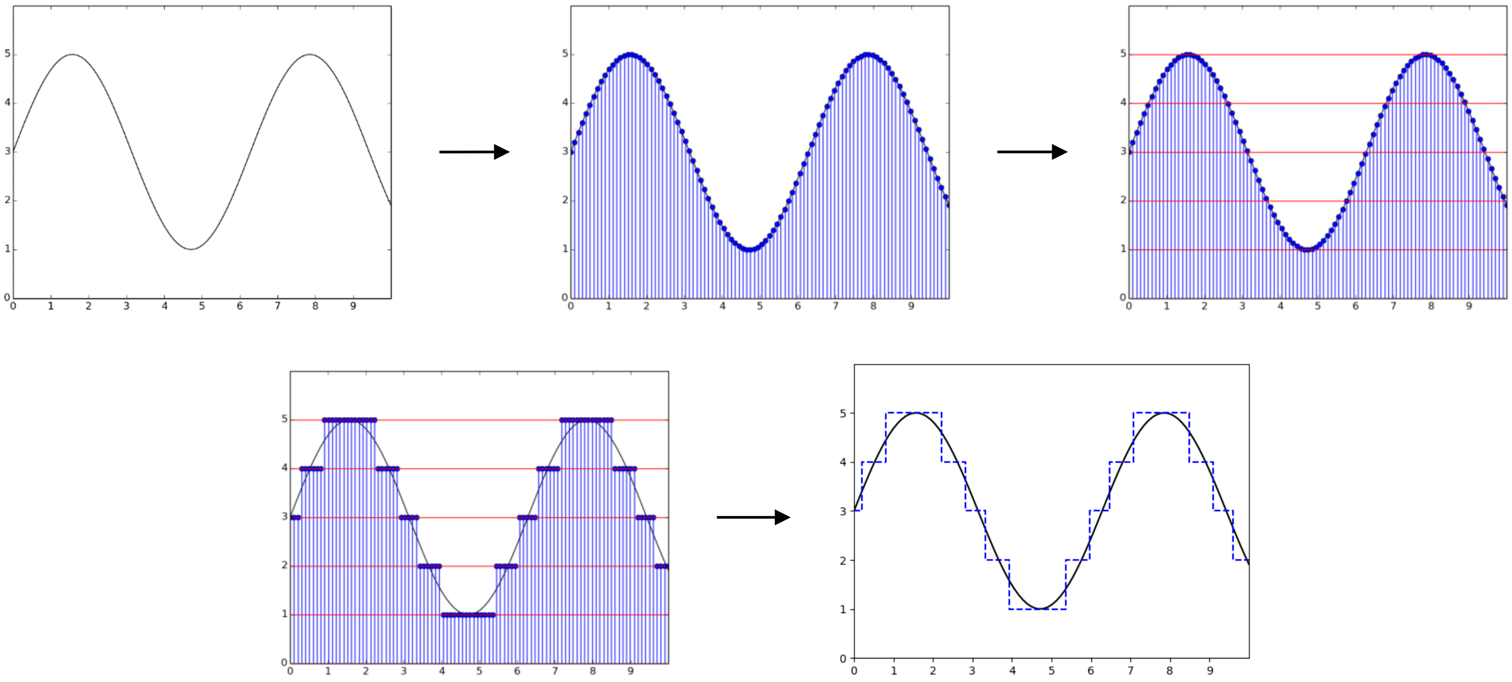






ETH zürich

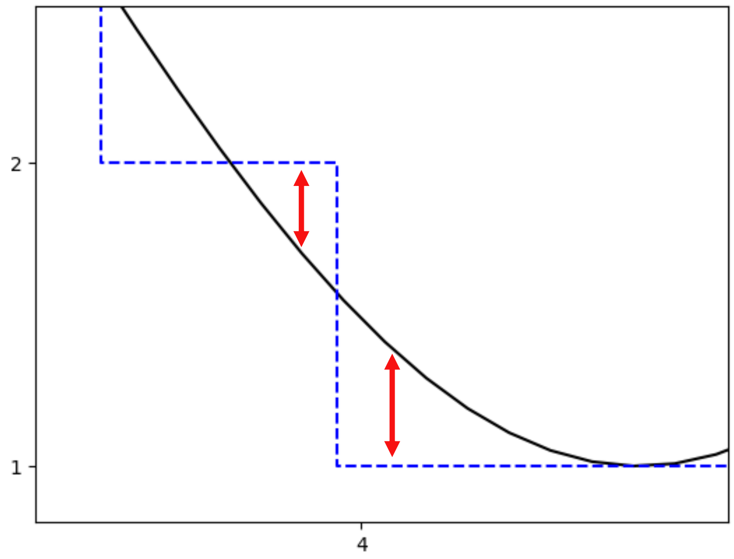
Reconstruction



Reconstruction Artifacts - Quantization

- Quantization levels do not match the statistics of the original signal

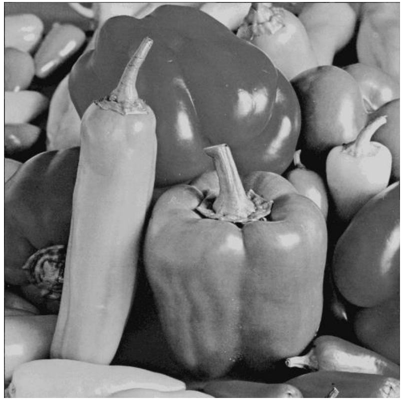
- Example: uniform quantization



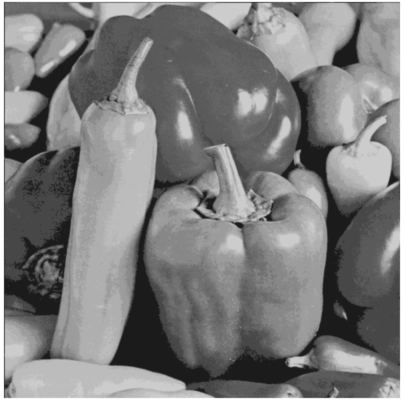
False contours!

False Contours – Example

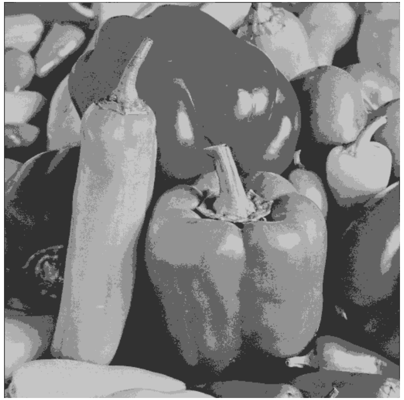
255 levels



25 levels



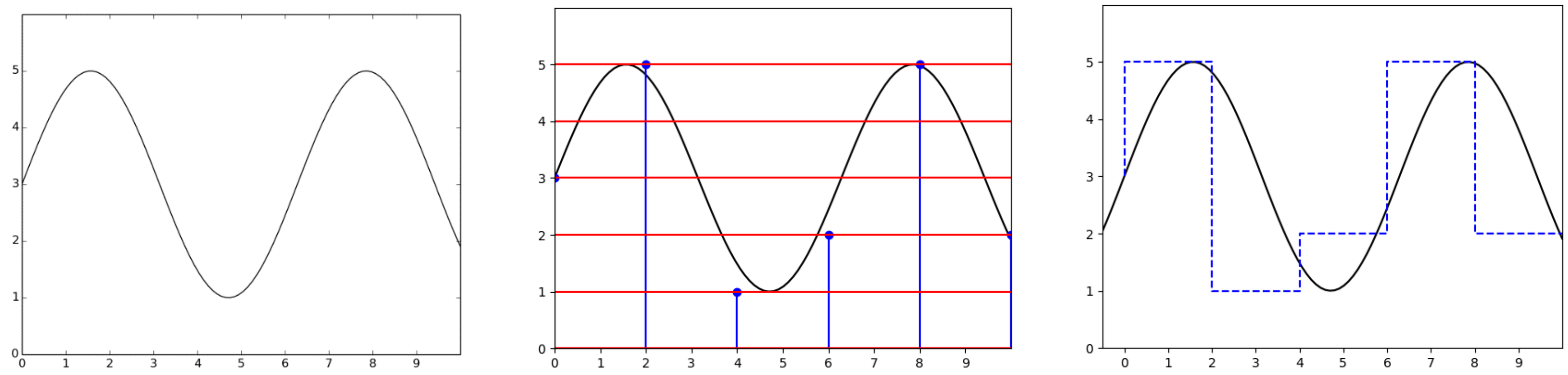
10 levels



5 levels

Reconstruction Artifacts - Sampling

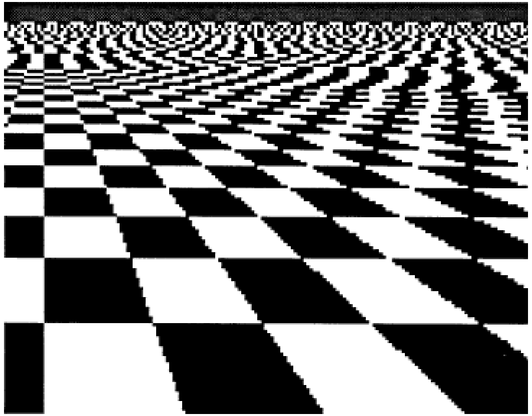
- Sampling rate too low:



Aliasing artefacts!

Aliasing in Computer Graphics

- Loss of detail
- Additional frequency components in the reconstructed signal
- We will see more examples later in the lecture



Signal Processing

- Aliasing is well understood in signal processing
- Interpret images as 2D signals
- Aliasing = sampling of L^2 functions below the Nyquist frequency

Nyquist-Shannon Sampling Theorem

- A continuous signal can be perfectly reconstructed if the sampling rate is at least twice the maximum frequency

- 2D case

$$\Delta x \geq 2u_{max} \quad \Delta y \geq 2v_{max}$$

1D Fourier Transform

- What are u_{max}, v_{max} for a 2D image?

- Fourier analysis!

- Represent f through harmonic waves

1D Fourier Transform

- Continuous:

$$f(x) = \int_{-\infty}^{\infty} F(u) e^{j2\pi ux}\, du$$

- The amplitudes $F(u)$ of waves with frequency u (spectrum) are computed as

$$F(u) = \int_{-\infty}^{\infty} f(x) e^{-j2\pi ux}\, dx$$

2D Fourier Transform - Continuous

- Continuous

$$F(u,v) = \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} f(x,y)e^{-j2\pi(ux+vy)}\,dx\,dy$$

- Continuous Inverse

$$f(x,y) = \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} F(u,v)e^{j2\pi(ux+vy)}\,du\,dv$$

2D Fourier Transform - Discrete

- Discrete

$$F(u,v) = \frac{1}{MN} \sum_{x=0}^{M-1} \sum_{y=0}^{N-1} f(x,y) e^{-j2\pi\left(\frac{ux}{M}+\frac{vy}{N}\right)}$$

- Discrete inverse

$$f(x,y) = \sum_{u=0}^{M-1} \sum_{v=0}^{N-1} F(u,v) e^{j2\pi\left(\frac{ux}{M}+\frac{vy}{N}\right)}$$

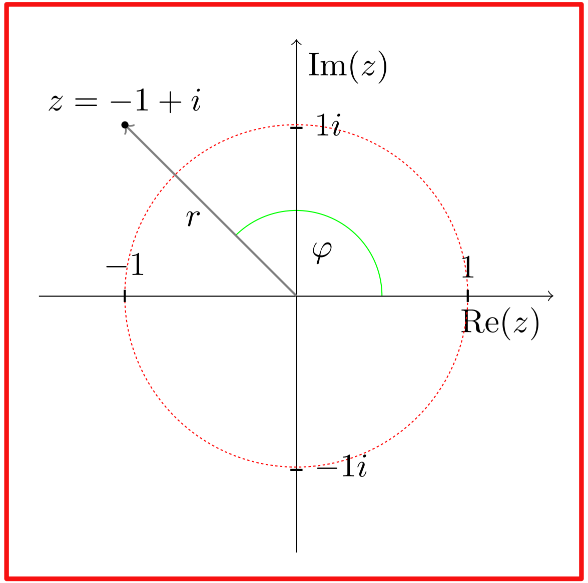
Fourier Transform - Properties

- $F(u, v) = a + ib \in \mathbb{C}$
- Magnitude

$$r = |F| = \sqrt{a^2 + b^2}, \qquad \forall u, v$$

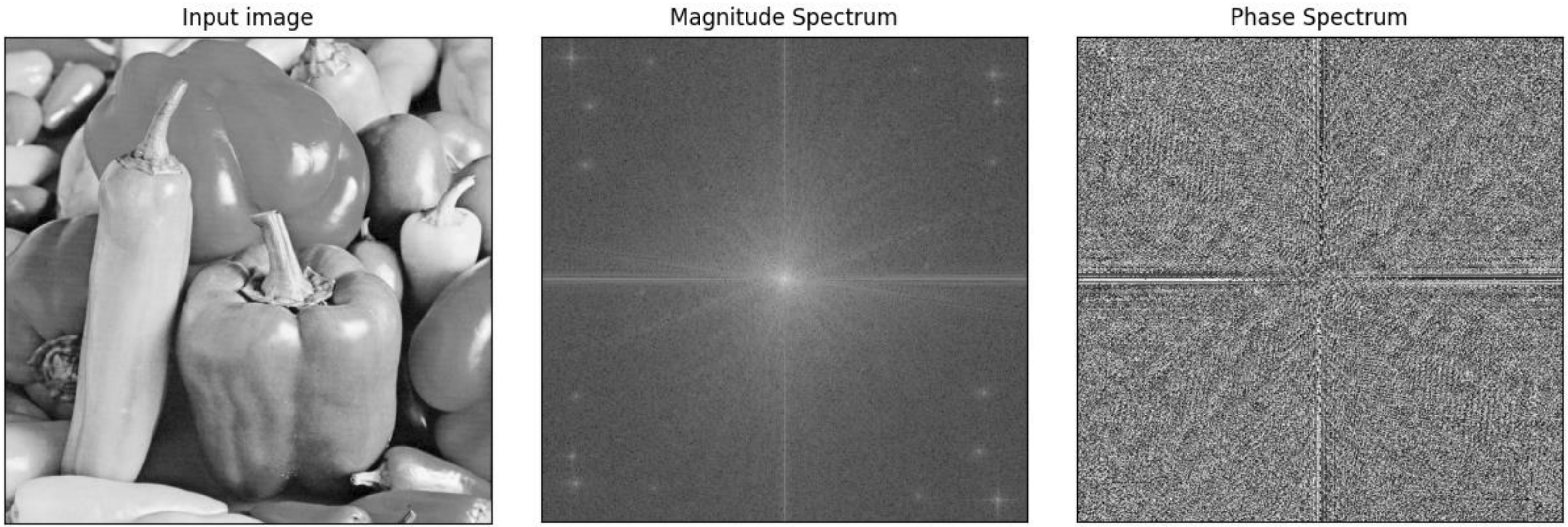
- Phase

$$\phi = \arg(F) = \begin{cases} \arctan\left(\frac{b}{a}\right) & a > 0 \\ \arctan\left(\frac{b}{a}\right) + \pi & a < 0, b \geq 0 \\ \arctan\left(\frac{b}{a}\right) - \pi & a < 0, b < 0 \\ +\frac{\pi}{2} & a = 0, b > 0 \\ -\frac{\pi}{2} & a = 0, b < 0 \\ \text{undefined} & a = 0, b = 0 \end{cases}, \qquad \forall u, v$$



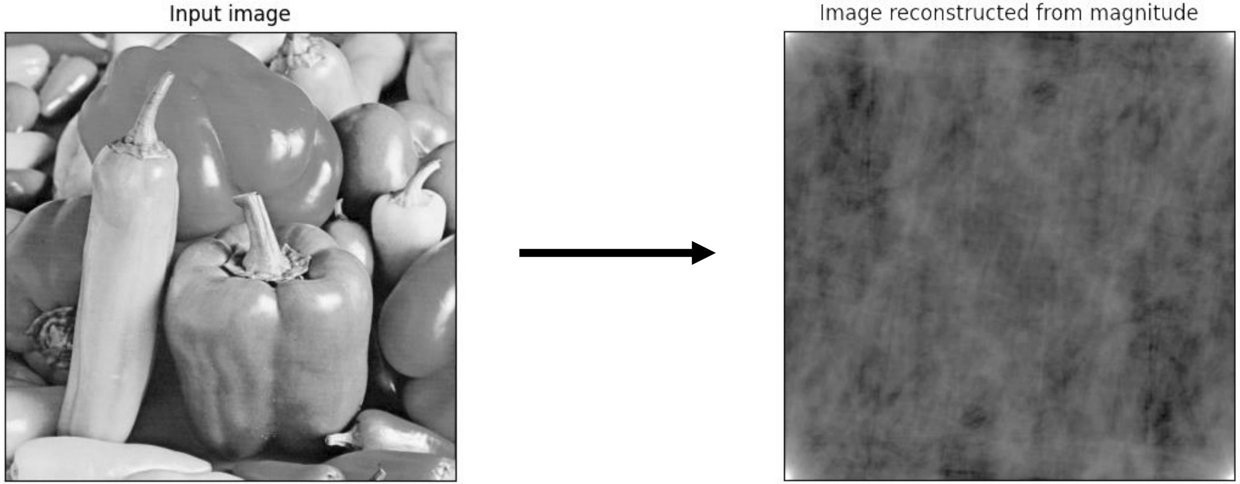
https://blogs.ethz.ch/MatheII-2020/2020/03/19/polardarstellung-und-einheitskreis/

2D Fourier Transform – Phase and Magnitude



- DC component centered
- Log scaling
- Symmetry properties

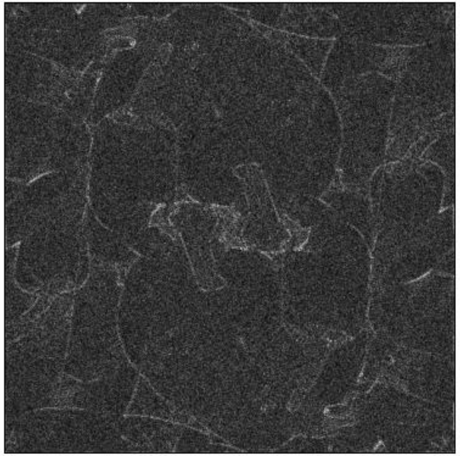
2D Fourier Transform – Reconstruction from Magnitude



2D Fourier Transform – Reconstruction from Phase

Input image

Image reconstructed from phase



Phase contains information on structure of image (e.g., edges)!

Avoiding Aliasing

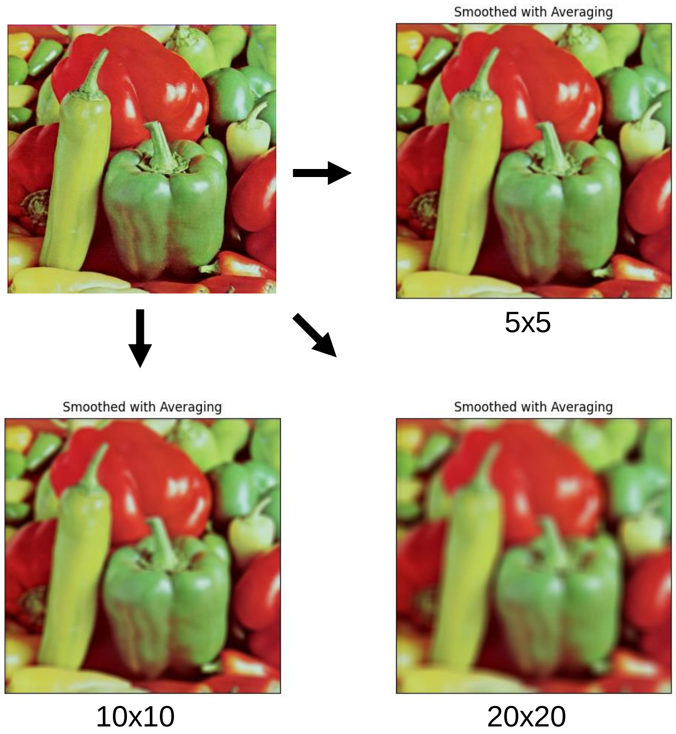
- Let u_{max} be the maximum u for which $|F(u)| > 0$

- If we choose $\Delta x \geq 2u_{max}$, we can avoid aliasing artefacts

Avoiding Aliasing

- Alternative:
  - $F(u) = 0$, if $u > \frac{1}{2} u_{max}$
  - Corresponds to applying a low-pass filter to the signal before sampling it

- For 2D images, this corresponds to smoothing the image



Avoiding Aliasing

- We will now explore why low-pass filtering the signal works

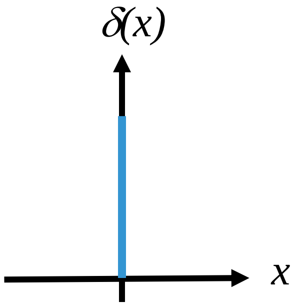
- For this, we consider an analysis in the Fourier domain

  …

Mathematical Representation of Sampling

- Recap: Dirac delta

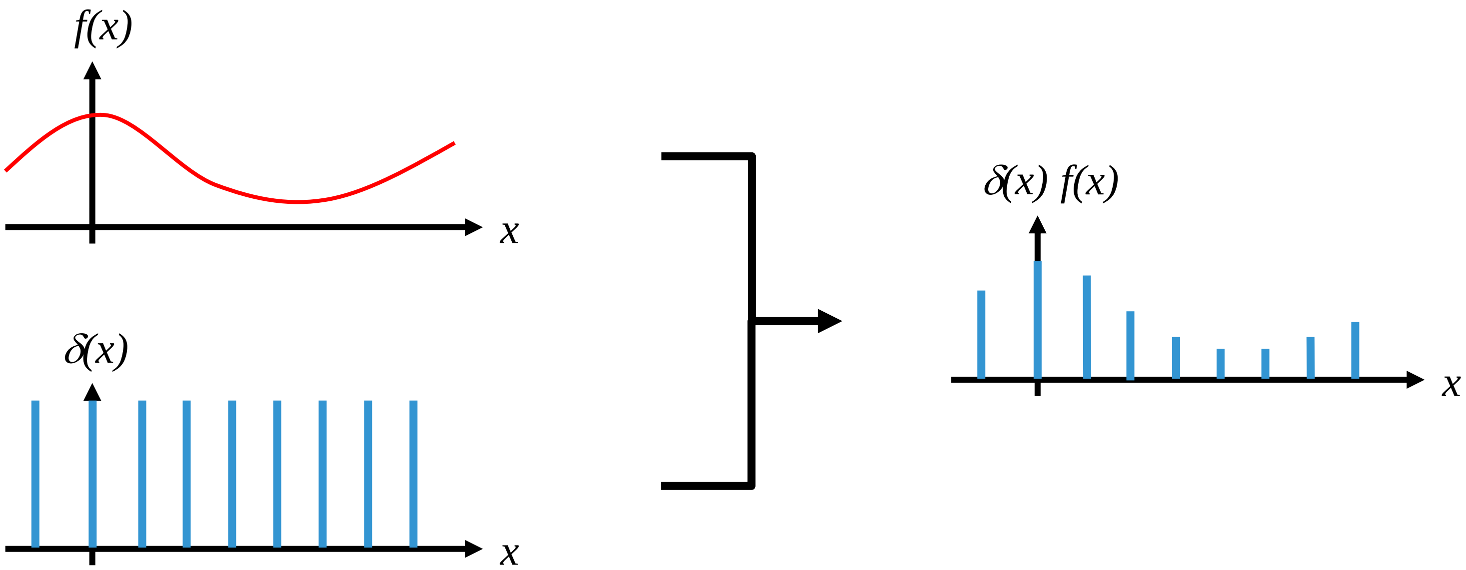
$$\delta(x) = \begin{cases} 0, & x \neq 0 \\ \text{undefined}, & x = 0 \end{cases} \quad \text{with} \quad \int_{-\infty}^{\infty} \delta(x)\, dx = 1$$



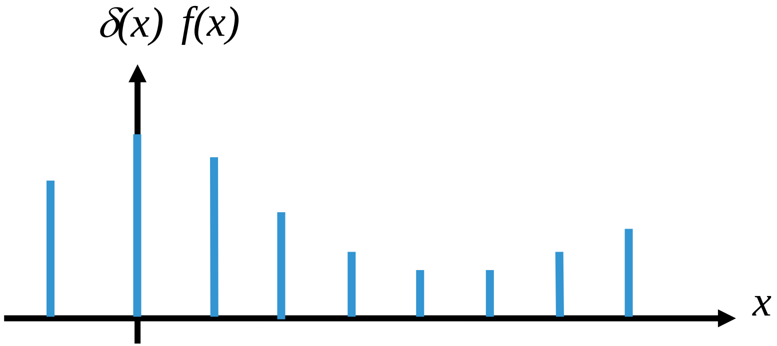
Used to model an impulse

Mathematical Representation of Sampling

- Sampling: multiplication with sequence of delta functions



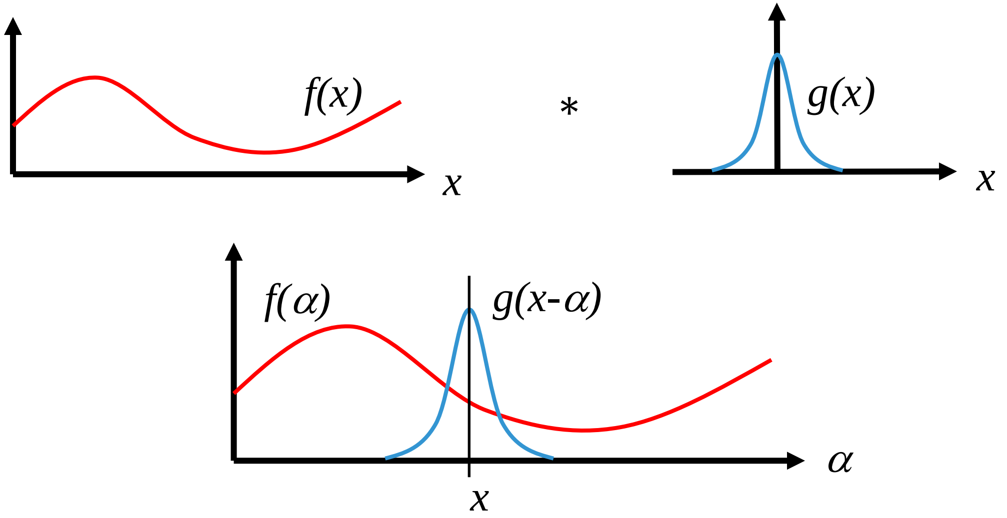
Mathematical Representation of Sampling



$$\underbrace{\int_{-\infty}^{\infty} f(\alpha)\delta(x-\alpha)\,d\alpha}_{\text{Convolution!}} = (f * \delta)(x)$$

Convolution

$$(f * g)(x) = \int_{-\infty}^{\infty} f(\alpha)g(x-\alpha)d\alpha$$



1D Convolution

- Continuous

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

- Discrete

$$f(x) * g(x) = \sum_{m=0}^{M-1} f(m) g(x - m)$$

2D Convolution

- Continuous

$$f(x,y) * g(x,y) = \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} f(\alpha,\beta)g(x-\alpha,y-\beta)\,d\alpha\,d\beta$$

- Discrete

$$f(x,y) * g(x,y) = \sum_{m=0}^{M-1}\sum_{n=0}^{N-1} f(m,n)g(x-m,y-n)$$

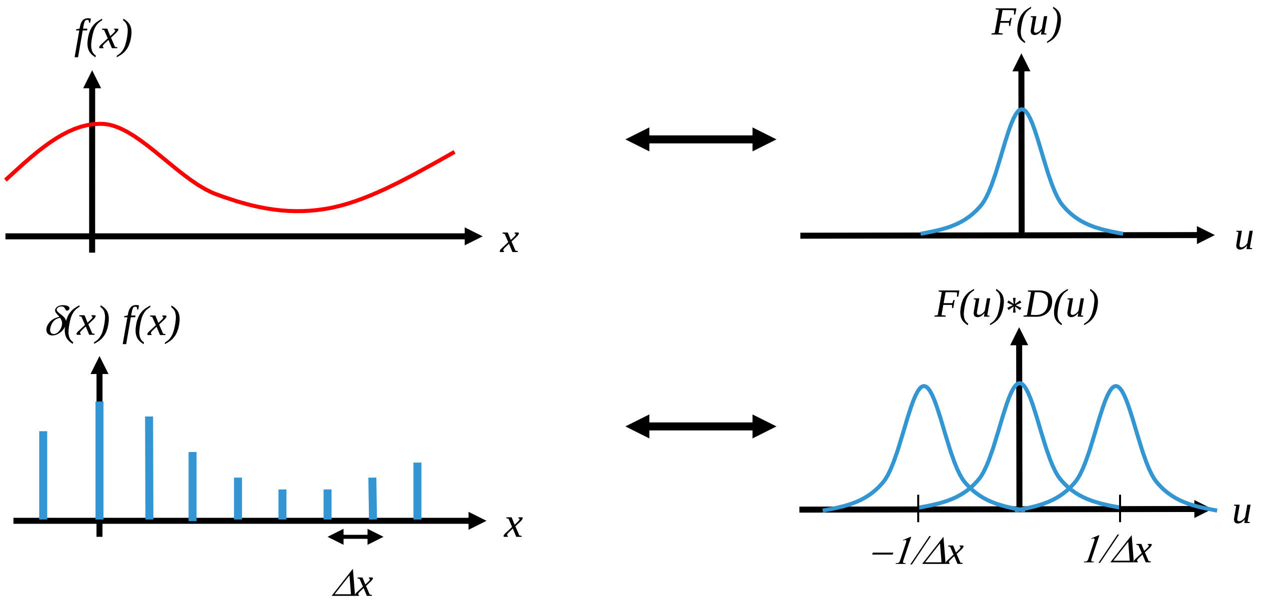
Convolution Theorem

$$(f * g)(x) \equiv F(u) \cdot G(u)$$

$$(F * G)(u) \equiv f(x) \cdot g(x)$$

- Frequencies of f are modulated with frequencies of g
  - Convolution as the application of a frequency filter on f
  - g is the characteristic function (or transfer function), which attenuates or amplifies the frequencies of f

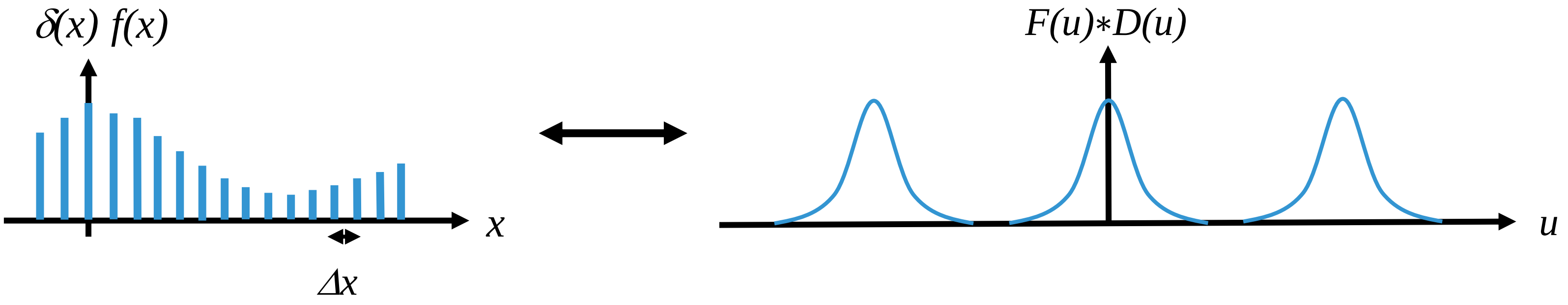
Source for Aliasing in the Frequency Domain



Overlap of Fourier transforms
leads to aliasing!

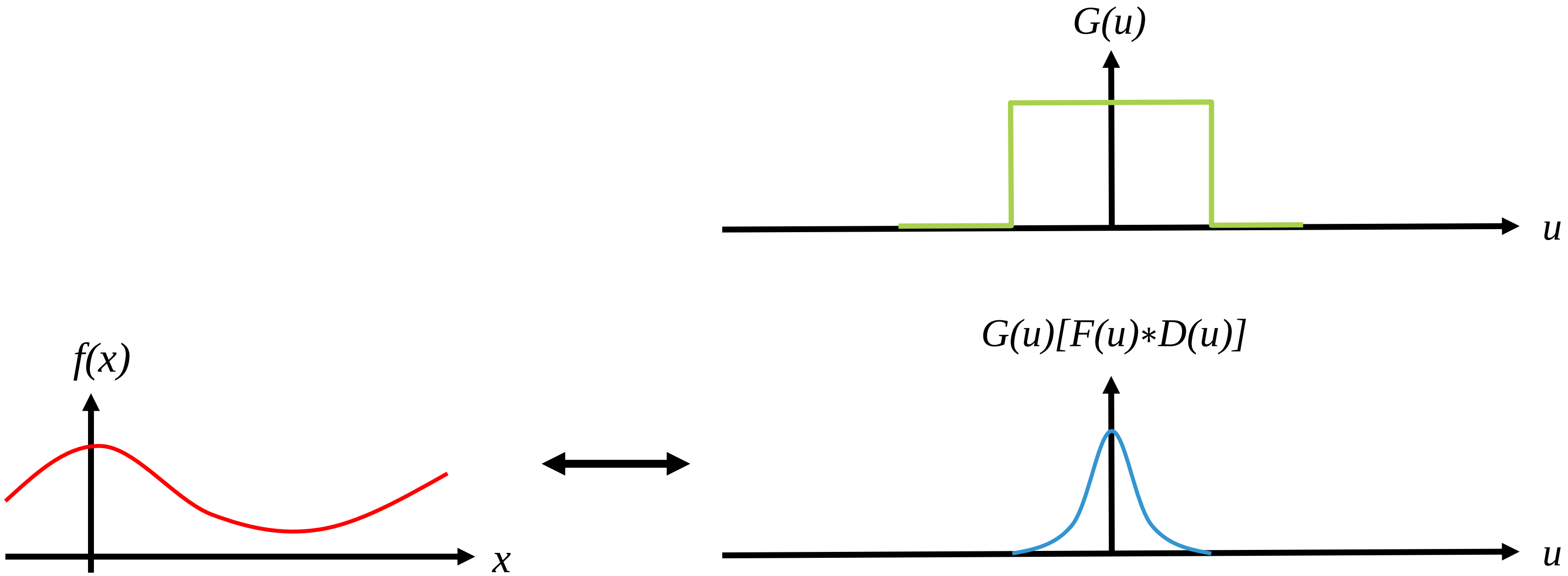
Avoiding Aliasing

- Avoid aliasing by using a higher sampling rate …



Avoiding Aliasing

- … and applying a low-pass filter



Bandlimiting

- Out-of-band energy in real-world signals
- Bandlimiting restricts amplitude of spectrum to zero for frequencies beyond the cut-off frequency
- Idea: restrict the bandwith to satisfy the Nyquist-Shannon Theorem

Filtering

- Filter = transfer function
- Goal: enhance or attenuate components of a signal
- Recap Convolution Theorem:
  - Apply the filter to the signal through convolution in the spatial domain

FIR vs. IIR Filters

- FIR filter
  - Finite support
  - Stable
  - High complexity
- IIR filter
  - Infinite support
  - Unstable
  - Low complexity

Filter Mask

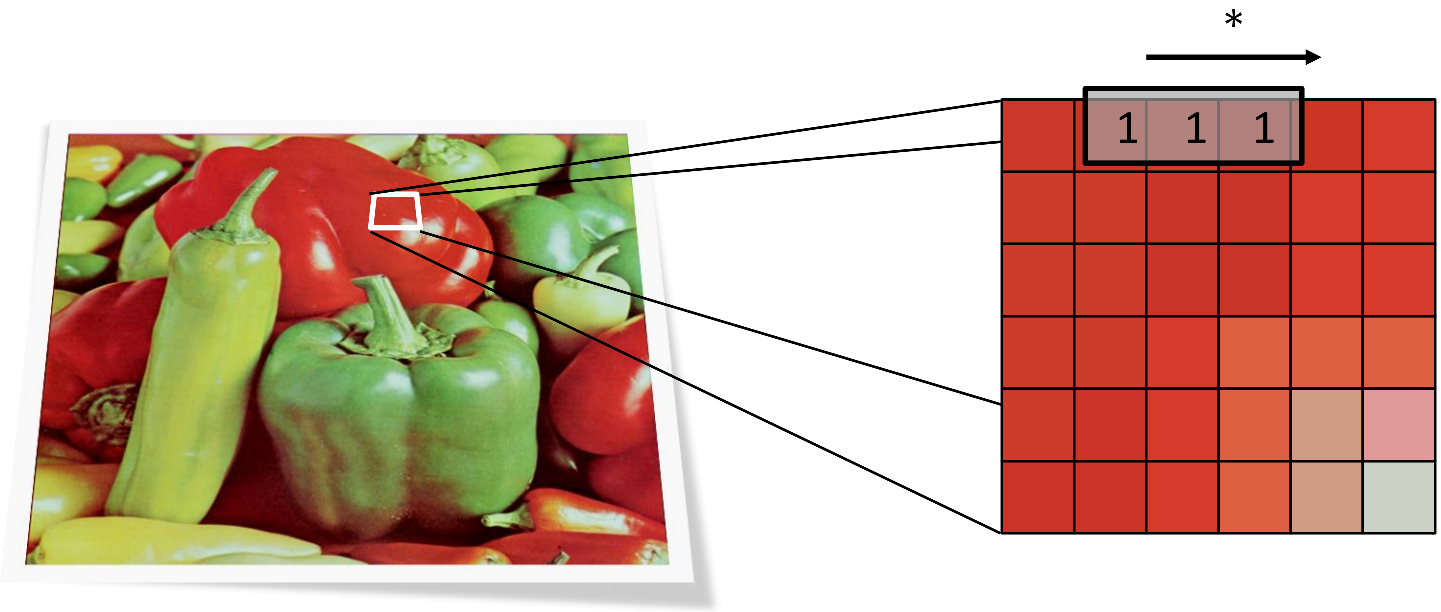
- Example: 1D Averaging Filter

$$f(x_n) \leftarrow \frac{1}{3} \cdot \big(f(x_{n-1}) + f(x_n) + f(x_{n+1})\big)$$

- Represent filter through a filter mask

$$m_{\text{avg}} = \begin{bmatrix} 1 & 1 & 1 \end{bmatrix}$$

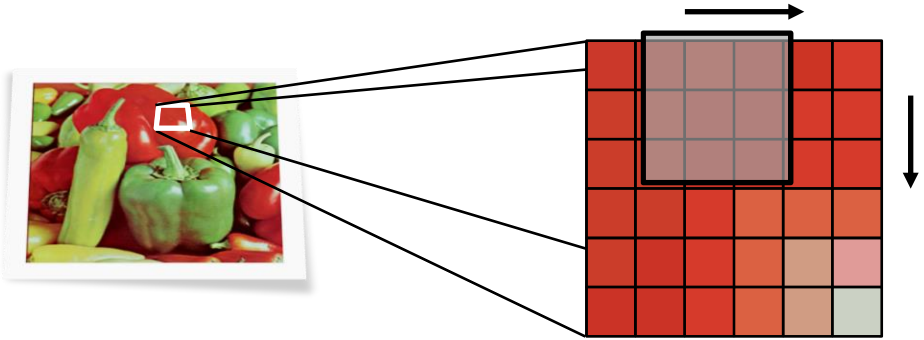
Filter Mask



Linear Separability

- Separate a 2D filter into two 1D filters to reduce computational costs

$$M_{avg} = \begin{bmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{bmatrix} = \begin{bmatrix} 1 \\ 2 \\ 1 \end{bmatrix} \cdot \begin{bmatrix} 1 & 2 & 1 \end{bmatrix}$$

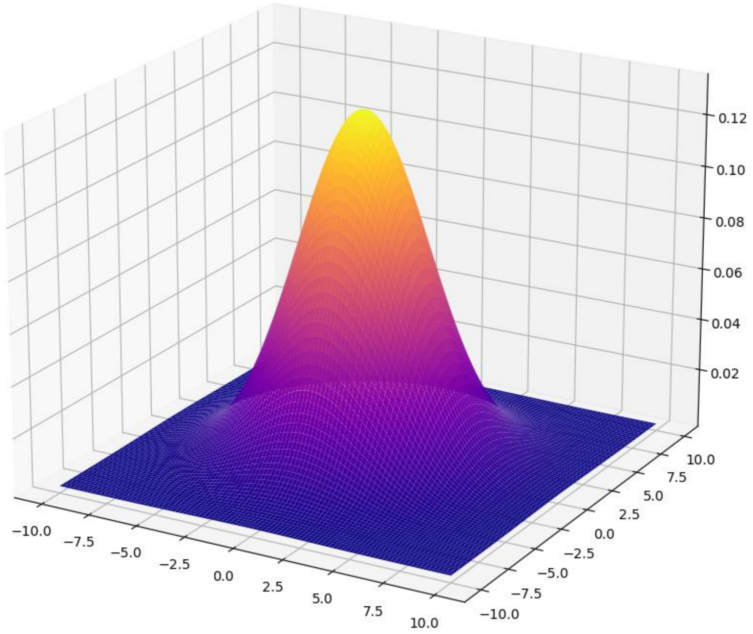


Antialiasing Filters – Gaussian

- Gaussian (2D)

$$G_\sigma(x,y)=\frac{1}{\sqrt{2\pi\sigma^2}}\cdot e^{-\frac{x^2+y^2}{2\sigma^2}}$$

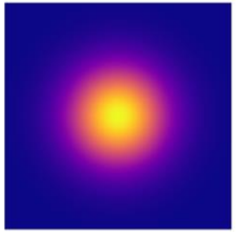
  - Low-pass filter
  - 2D Gaussian filter is separable



$\sigma = 3$

Antialiasing Filters – Gaussian

*

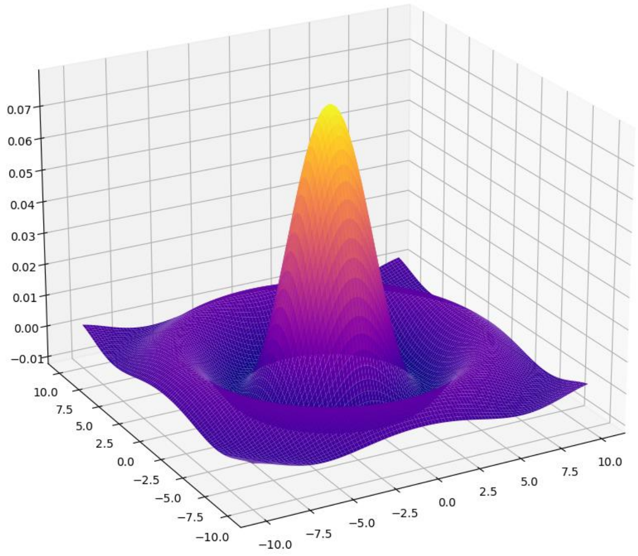


=

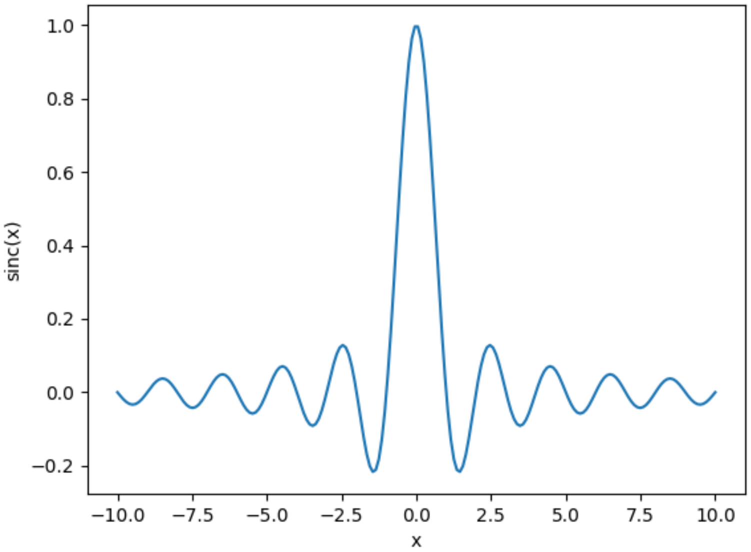
- Closer pixels have a higher weight

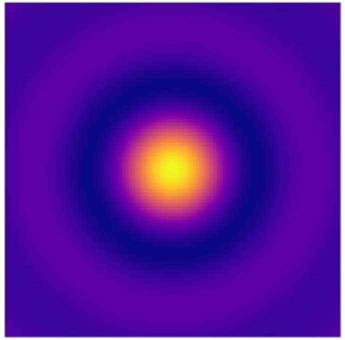
Antialiasing Filters – Sinc

- Sinc (2D)
  - $S_{\omega_c}(x,y) = \frac{\omega_c sinc(\frac{\omega_c x}{\pi})}{\pi} \times \frac{\omega_c sinc(\frac{\omega_c y}{\pi})}{\pi}$
  - Ideal low-pass filter
  - IIR filter

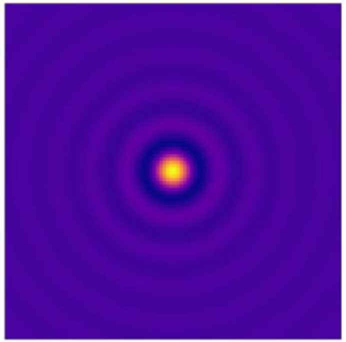


Antialiasing Filters – Sinc





$\omega_c = 1$



$\omega_c = 3$

ω_c is the cut-off freqeuncy

Antialiasing Filters – B-Spline

- B-Spline filter of degree n

$$\beta^n = \underbrace{\beta^0 * \ \ldots * \ \beta^0}_{n + 1 \text{ times}} \qquad \beta^0 = \begin{cases} 1, & x \in [t_i, t_{i+1}] \\ 0, & \text{otherwise} \end{cases}$$

- $t_0, \ldots, t_m$ are the B-spline knots

Antialiasing Filters – B-Spline

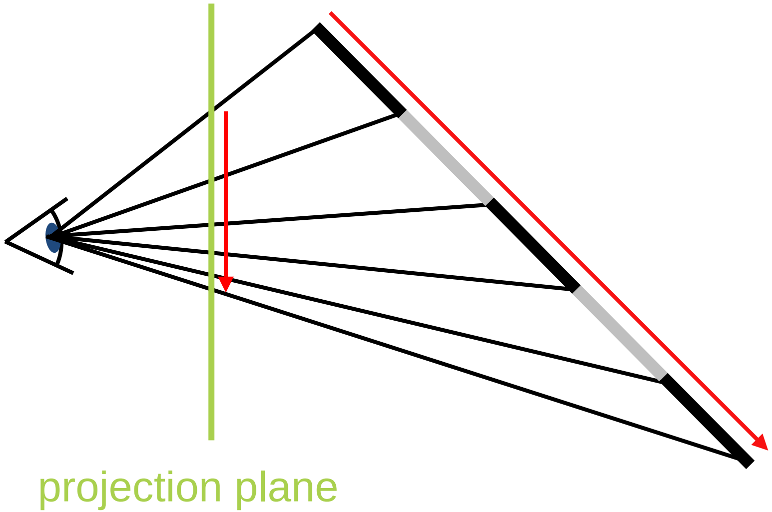
- Allows for locally adaptive smoothing

- Adapt size of convolution kernel to underlying signal characteristics

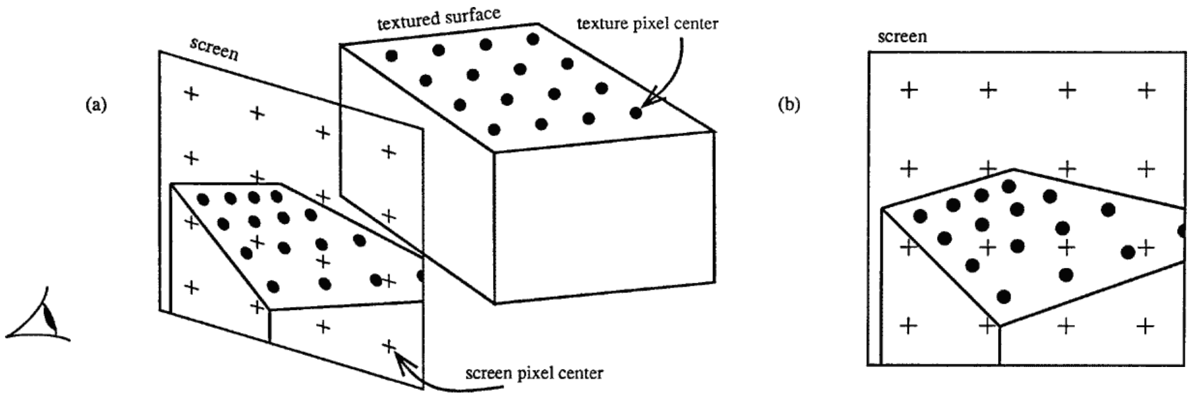
- Example
  - Wide splines for smooth image regions
  - Narrow splines for edge regions

Perspective Projection

- Linear variation in world coordinates yields non-linear variation in screen coordinates



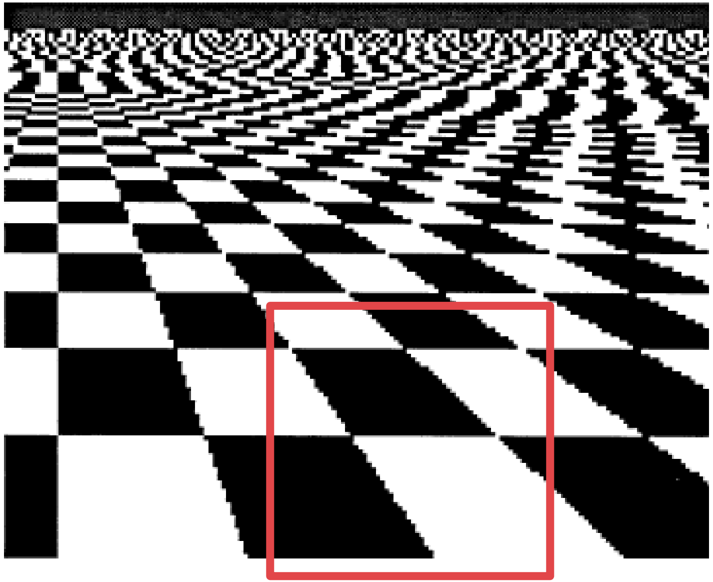
Perspective Projection



- Non-uniform sampling pattern on screen
- Optimal resampling filter is spatially variant

Texture Aliasing

- **Magnification**
  - Pixel in texture image maps to area larger than one pixel
  - Jaggies

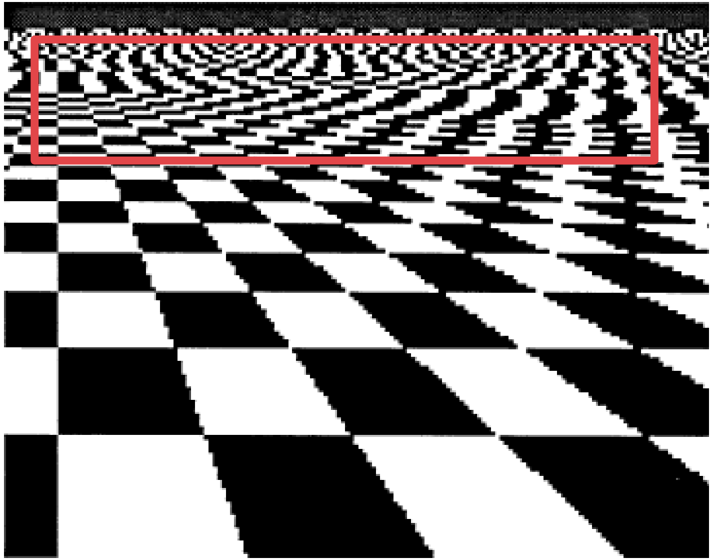


$\Longrightarrow$ Bilinear interpolation

Texture Aliasing

- **Minification**
  - Pixel in texture image maps to area smaller than one pixel
  - Moiré patterns

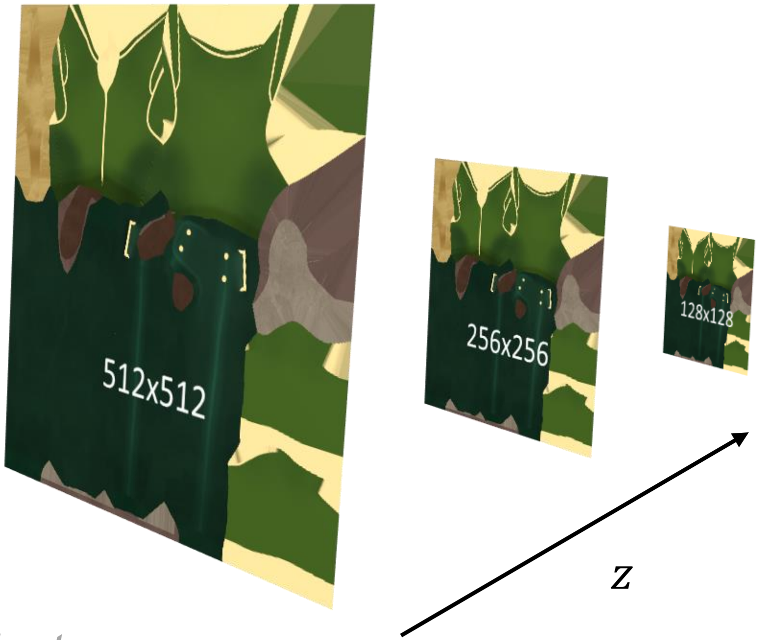
⟹ Mipmapping



Mipmapping

- Store texture at multiple resolutions

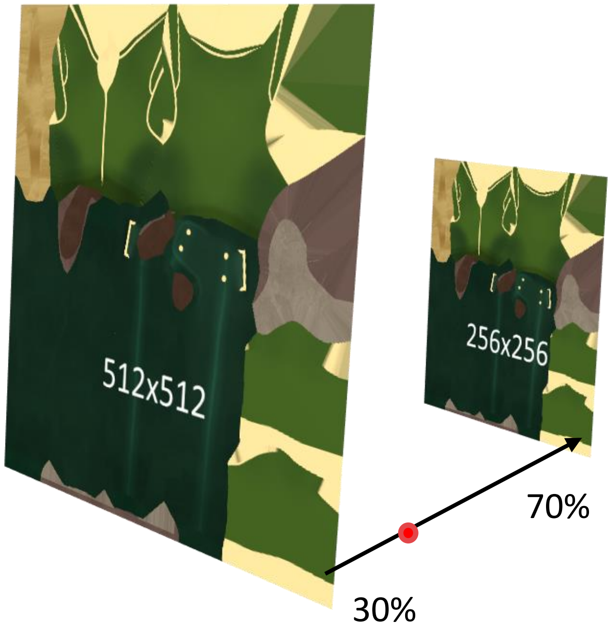
- Choose resolution level depending on projected size of triangle



More details in the Geometry & Textures lecture …

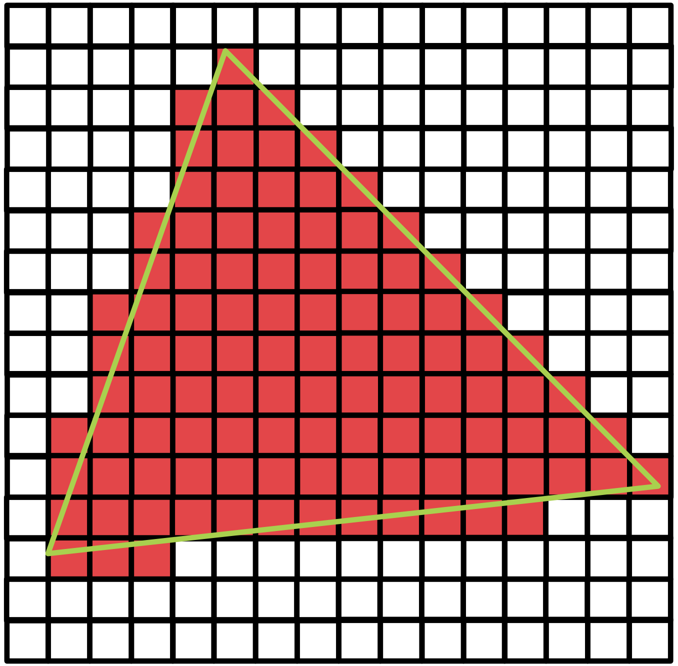
Mipmapping

- Filtering
  - Nearest: center of pixel on texture determines color
  - Bilinear: weighted avg. of overlapping texels
  - Trilinear: linear filtering of two mipmap levels



Geometry Aliasing

- At the edges of polygons



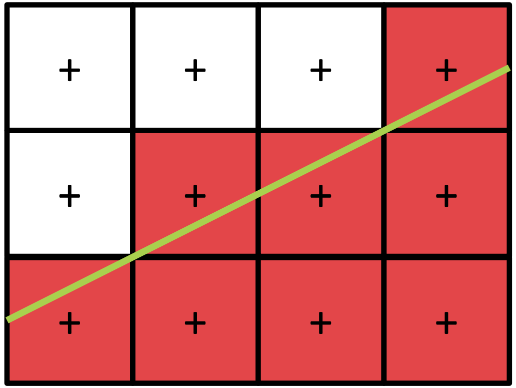
⇒ Supersampling

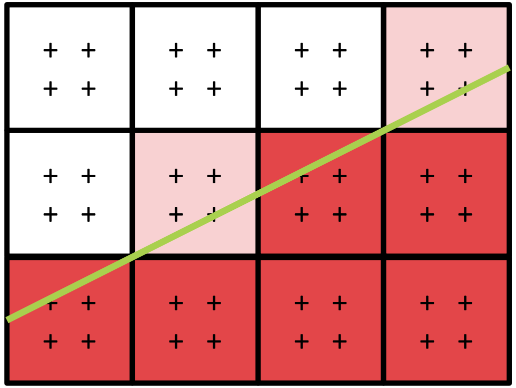
Supersampling

- Introduce multiple color samples per pixels
- Final color of pixel averaged from the samples that fall into this pixel
- Different patterns possible

Supersampling

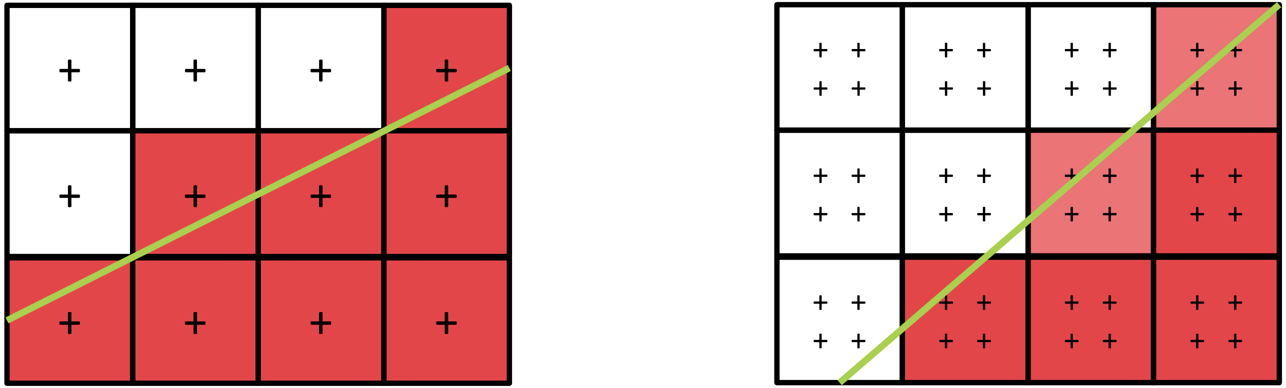
- Uniform





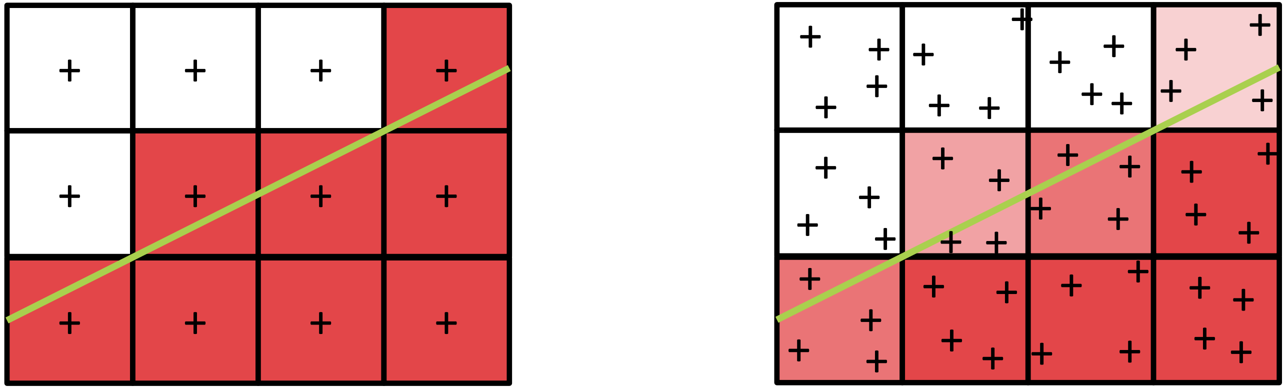
Supersampling

- Uniform



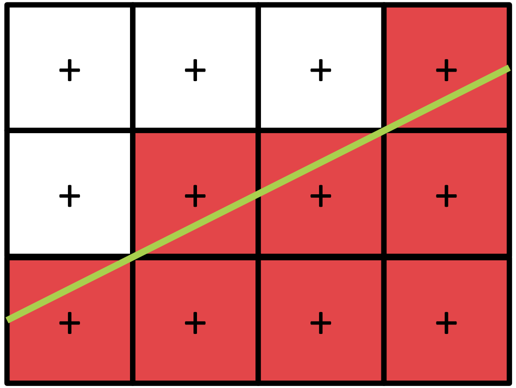
Supersampling

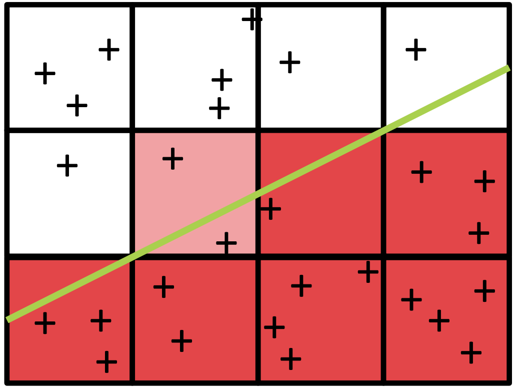
- Jittering



Supersampling

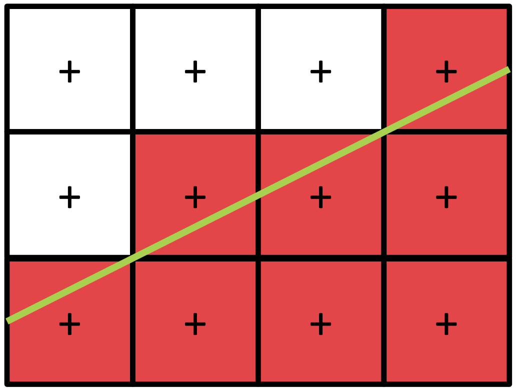
- Stochastic

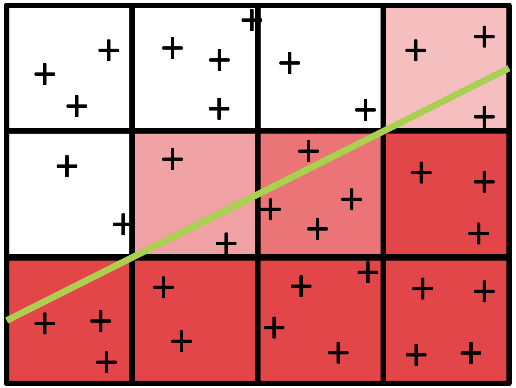




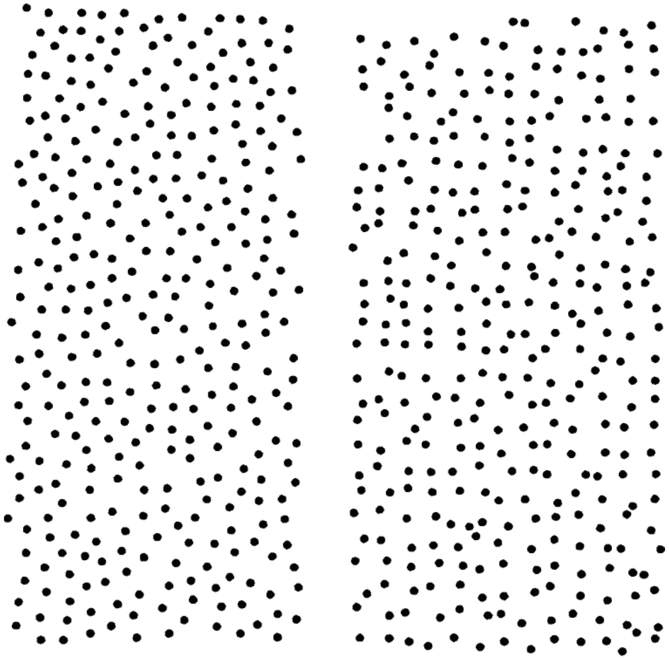
Supersampling

- Poisson



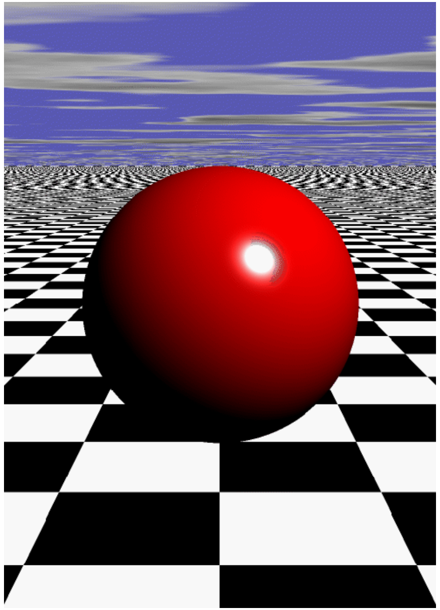


Poisson vs. Jittering

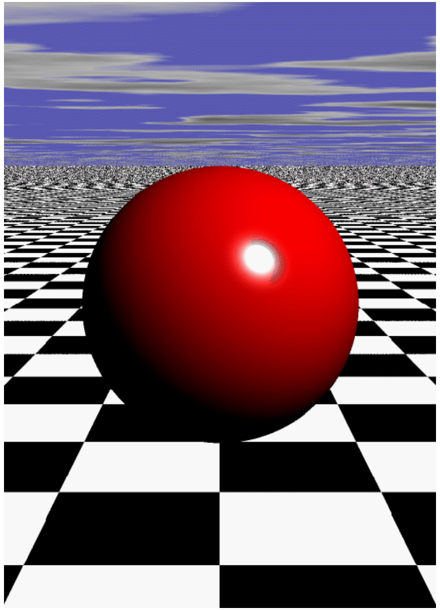


(a) (b)

Comparison

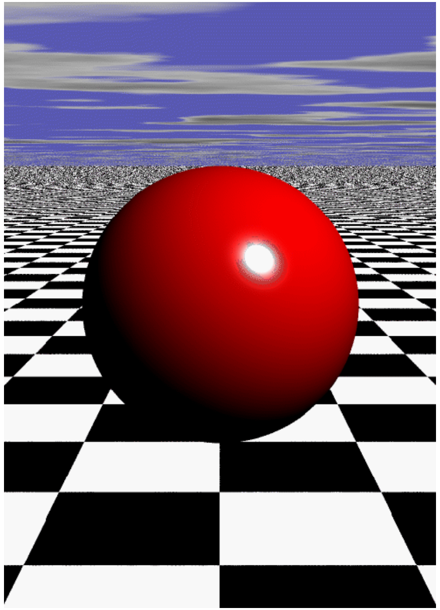


4 Rays/pixel

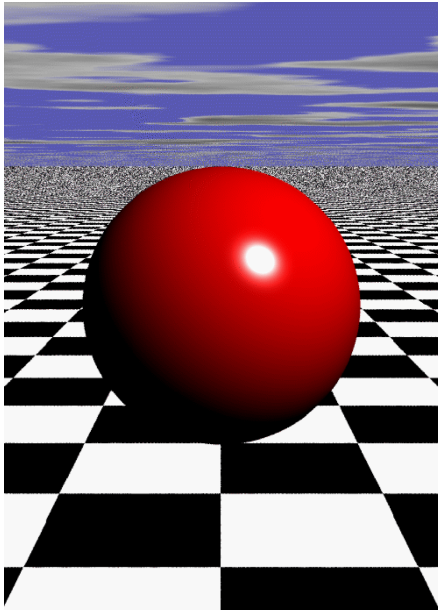


Jitter = 0.3

Comparison

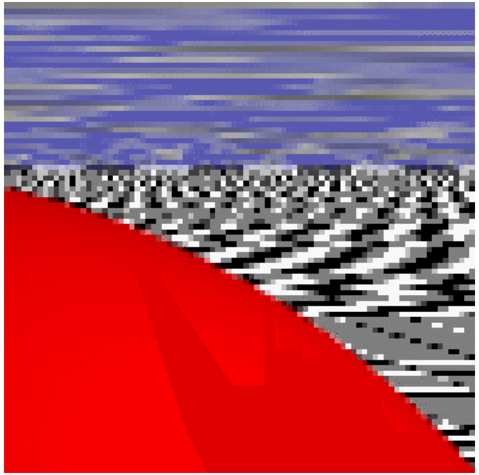


Jitter = 0.5

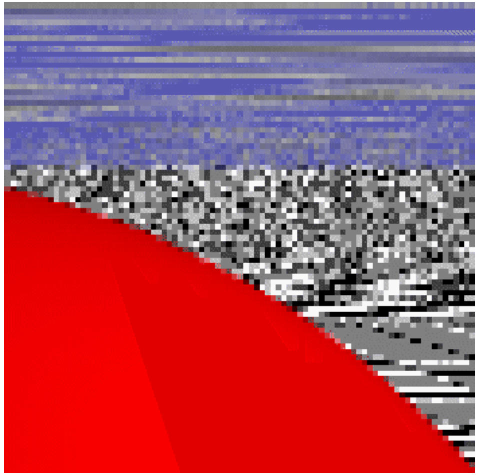


Jitter = 1.0

Comparison

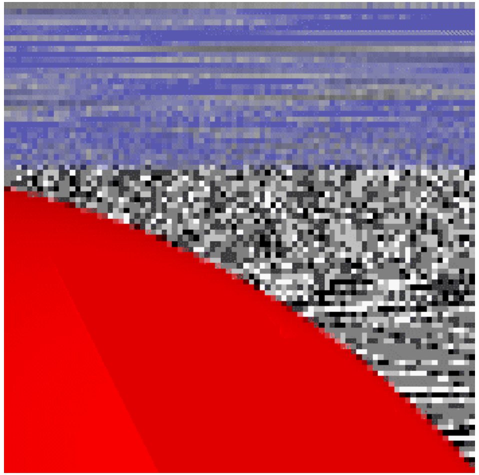


4 Rays/pixel

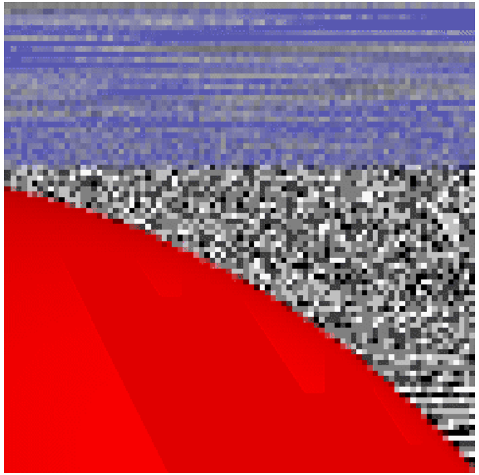


Jitter = 0.3

Comparison



Jitter = 0.5



Jitter = 1.0

Conclusion

- Aliasing is caused by overlaps in the Fourier domain
- Smoothing can reduce the amount of aliasing
- Texture aliasing can be reduced through bilinear interpolation and mipmapping
- Geometry aliasing can be reduced through supersampling