Computer Graphics I

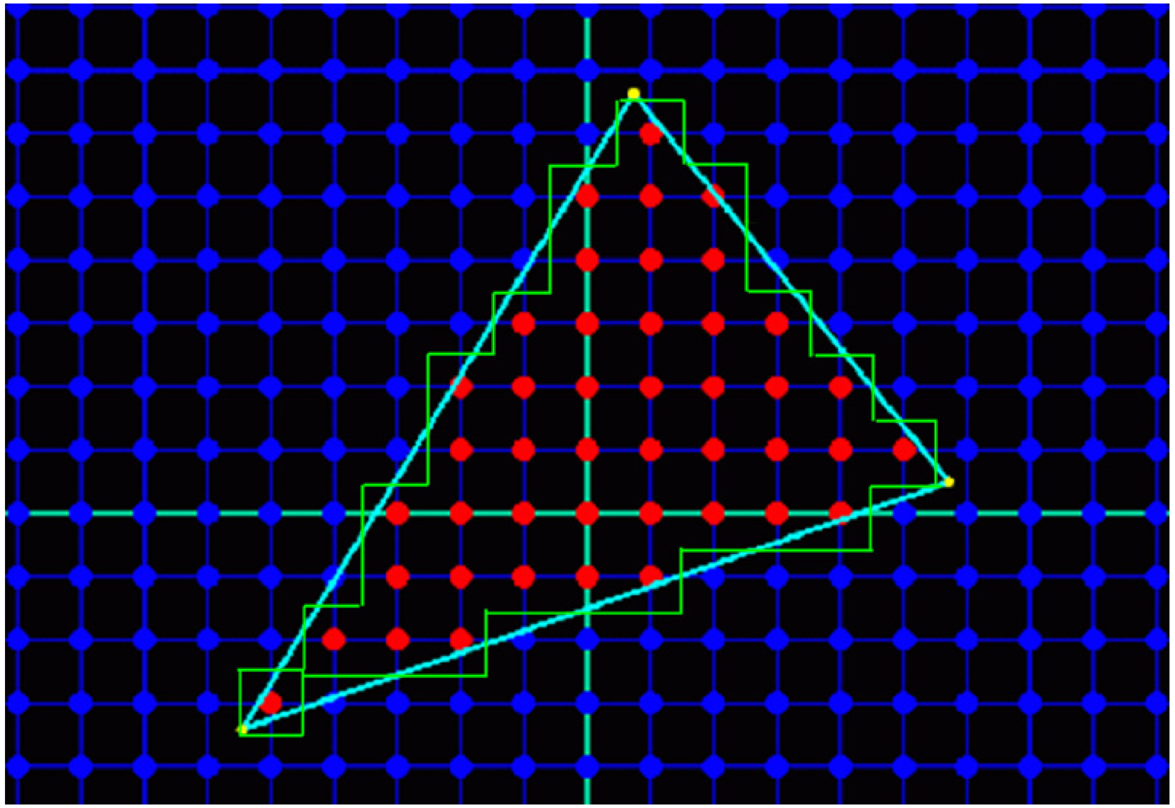
Lecture 11: Sampling and reconstruction

Xiaopei LIU

School of Information Science and Technology
ShanghaiTech University

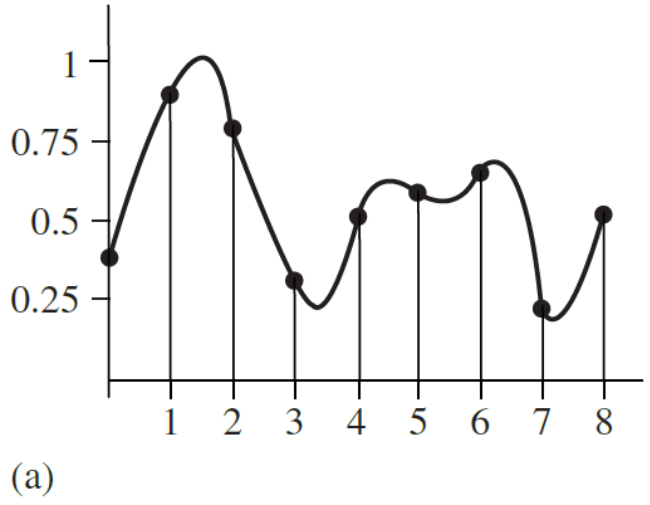
Aliasing problem

- Aliasing is caused by sampling (discretization) and reconstruction

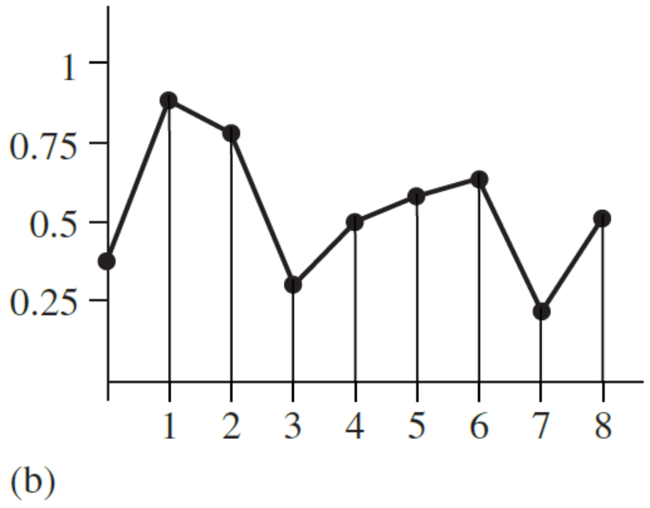


Aliasing problem

- **As an example, consider a 1D function**
  - We sample a continuous function $f(x)$ at discrete locations x'
  - x' is called the *sample position* and $f(x')$ is called the *sample value*



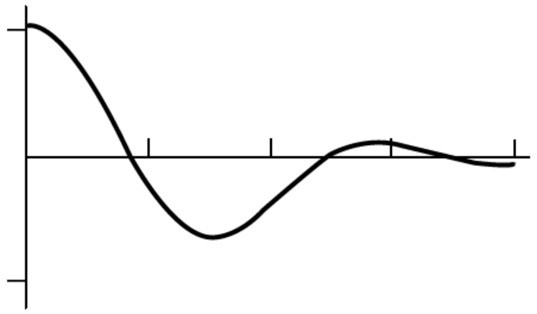
Samples from f(x)



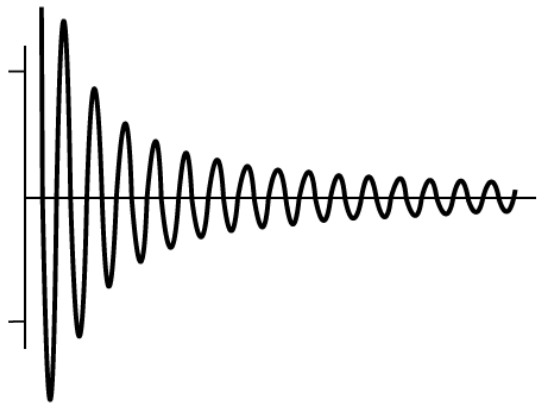
Reconstruction from samples

Frequency domain analysis

- **Reconstruction**
  - The reconstructed function $\tilde{f}$ may not match the original function perfectly, causing errors known as *aliasing*
  - *Fourier analysis* can be used to evaluate the quality of the match between the reconstructed and the original functions



(a) Low frequency function



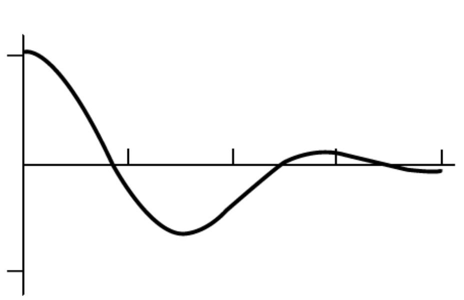
(b) High frequency function

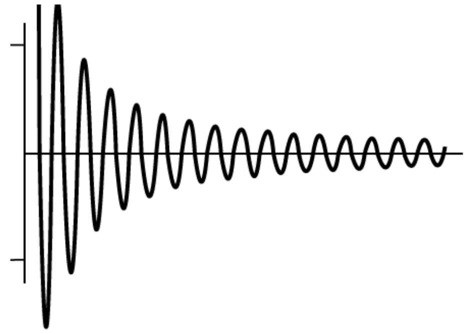
1. Frequency domain analysis

Frequency domain analysis

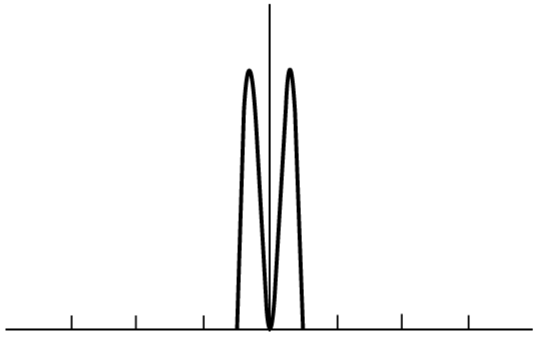
- **One of the foundations for frequency domain analysis is Fourier transform**
  - It represents a function in the *frequency domain*

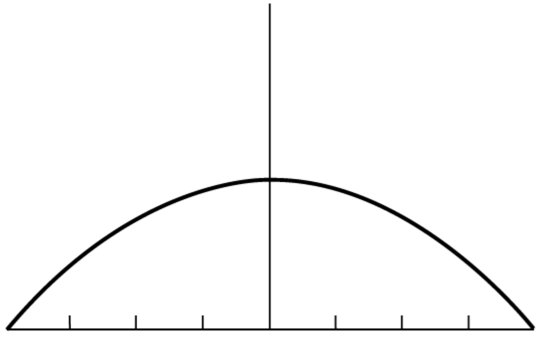
Spatial domain function





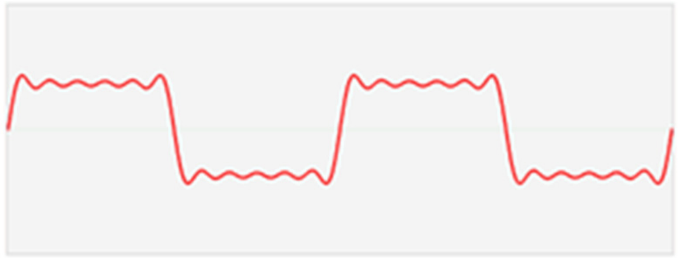
Frequency domain function





Frequency domain analysis

- **Fourier transform**
  - Most functions can be decomposed into a weighted sum of waves, first described by Joseph Fourier
  - Fourier transform converts a function into this representation
  - Gain insight into the error that is introduced by the sampling and reconstruction process
  - How to reduce the perceptual impact of this error



Frequency domain analysis

- **Fourier transform**
  - The Fourier transform of a one-dimensional function *f*(*x*) is:

$$F(\omega) = \int_{-\infty}^{\infty} f(x) e^{-i2\pi\omega x} \mathrm{d}x \quad \text{Recall that } e^{ix} = \cos x + i \sin x \text{, where } i = \sqrt{-1}$$

    - Frequency ω
  - Fourier transform operator $\mathcal{F}\{f(x)\} = F(\omega)$
  - Linear property:
    - $\mathcal{F}\{af(x)\} = a\mathcal{F}\{f(x)\}$
    - $\mathcal{F}\{f(x) + g(x)\} = \mathcal{F}\{f(x)\} + \mathcal{F}\{g(x)\}$

Frequency domain analysis

- **Inverse Fourier transform**
  - Spatial domain function can be uniquely determined by its frequency domain function with inverse Fourier transform

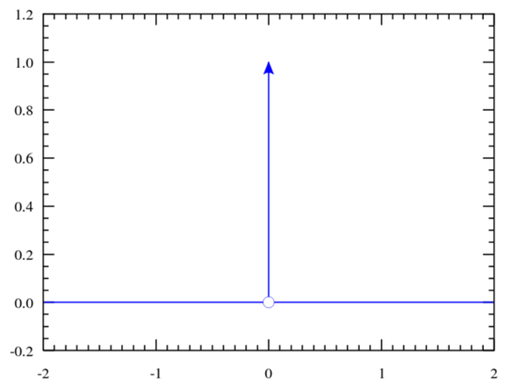
$$f(x) = \int_{-\infty}^{\infty} F(\omega) \mathrm{e}^{i 2\pi \omega x} \, \mathrm{d}\omega$$

  - Fourier transforms are pairwise
    - Dual to each other

Frequency domain analysis

- **Some Fourier transform pairs**

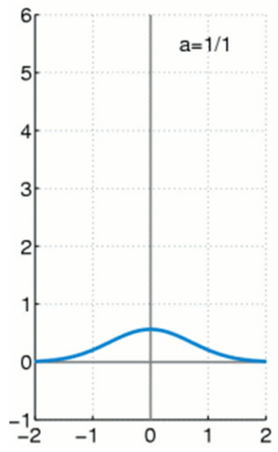
Spatial Domain	**Frequency Space Representation**
Box: $f(x) = 1$ if $\lvert x\rvert < 1/2$, 0 otherwise	Sinc: $f(\omega) = \mathrm{sinc}(\omega) = \sin(\pi\omega)/(\pi\omega)$
Gaussian: $f(x) = \mathrm{e}^{-\pi x^2}$	Gaussian: $f(\omega) = \mathrm{e}^{-\pi\omega^2}$
Constant: $f(x) = 1$	Delta: $f(\omega) = \delta(\omega)$
Sinusoid: $f(x) = \cos x$	Translated delta: $f(\omega) = \pi(\delta(1/2 - \omega) + \delta(1/2 + \omega))$
Shah: $f(x) = \mathrm{III}_T(x) = \sum_i \delta(x - Ti)$	Shah: $f(\omega) = \mathrm{III}_{1/T}(\omega) = (1/T)\sum_i \delta(\omega - i/T)$



$$\int \delta(x)\mathrm{d}x = 1, \text{and for all } x \neq 0, \delta(x) = 0$$

$$\delta(x) = \begin{cases} +\infty, & x = 0 \\ 0, & x \neq 0 \end{cases}$$

$$\int f(x)\delta(x)\,\mathrm{d}x = f(0)$$

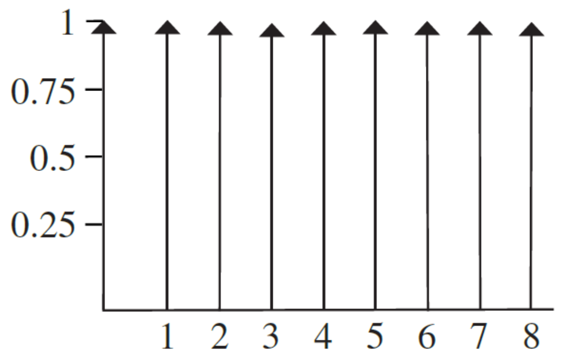


The limit (in the sense of distributions) of the sequence of zero-centered normal distributions

Frequency domain analysis

- **Sampling function**
  - The shah function

$$\text{III}_T(x) = \sum_{i=-\infty}^{\infty} \delta(x - iT)$$

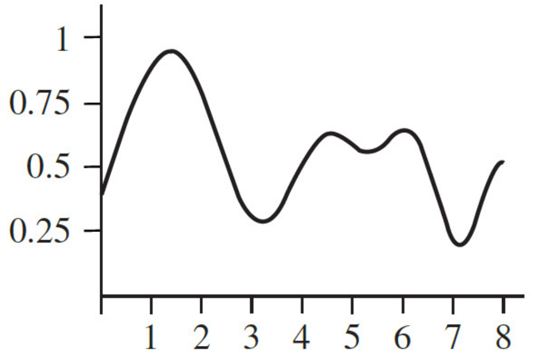


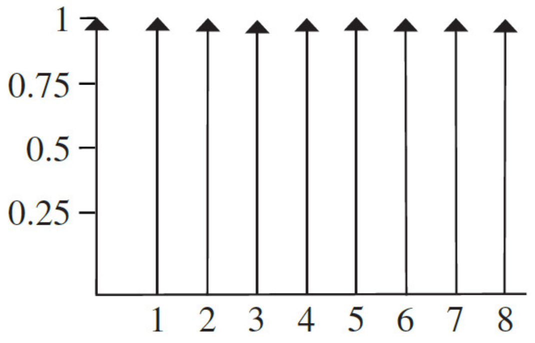
  - T defines the period, or sampling rate

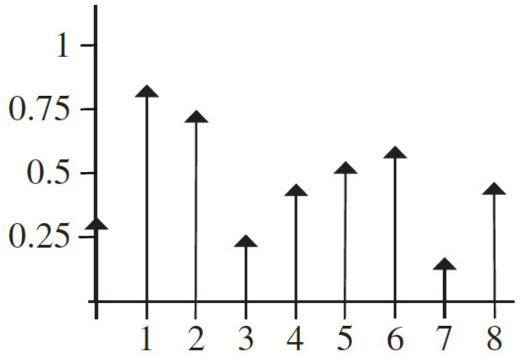
Frequency domain analysis

- **Sampling function**
  - Using shah function to sample
    - The multiplication yields an infinite sequence of values of the function at equally spaced points

$$\text{III}_T(x)f(x) = \sum_i \delta(x - iT)f(iT)$$







Frequency domain analysis

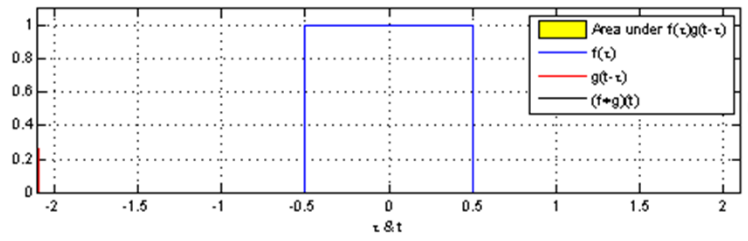
- **Reconstruction**
  - These sample values can be used to define a ***reconstructed function*** $\tilde{f}$
  - By choosing a reconstruction filter function *r*(*x*) and computing the convolution

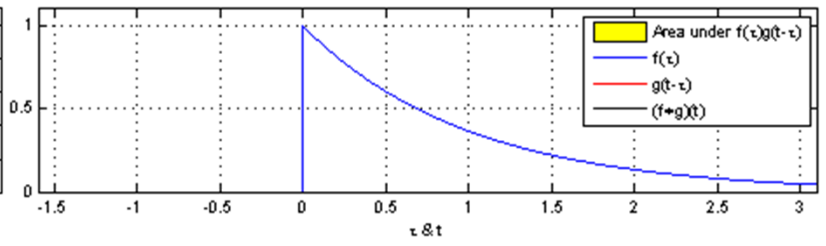
$$\int_{-\infty}^{\infty} \delta(\tau) g(t-\tau)\, d\tau = g(t)$$

$$\left(\mathrm{III}_T(x) f(x)\right) \otimes r(x)$$

  - The convolution operation is defined as

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

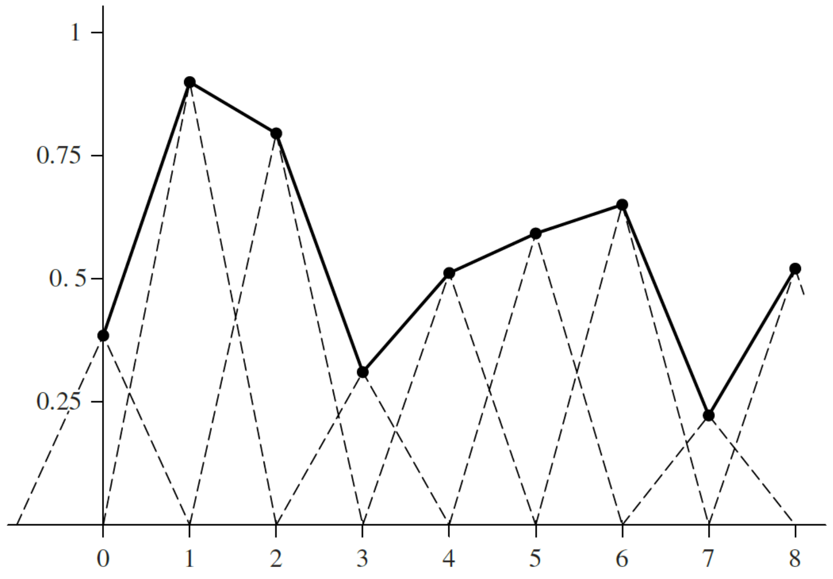


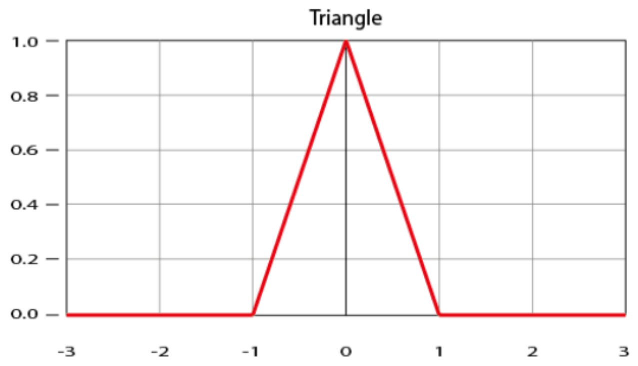


Frequency domain analysis

- **Reconstruction**
  - Convolution gives a weighted sum of scaled instances of the reconstruction filter centered at the sample points
  - For example, using triangle reconstruction filter

$$r(x) = \max(0, 1 - |x|)$$





Frequency domain analysis

- **Fourier analysis to the sampling and reconstruction process**
  - For now, assume function *f*(*x*) is <u>band-limited</u>
  - There exists some frequency ω_0 beyond which *f*(*x*) contains no frequencies
  - Band-limited functions have frequency space representations with compact support

$$F(\omega) = 0 \text{ for all } |\omega| > \omega_0$$

Frequency domain analysis

- **An important idea in Fourier analysis**
  - Fourier transform of the product of two functions can be shown to be the convolution of their individual Fourier transforms, and vice versa

$$\mathcal{F}\{f(x)g(x)\} = F(\omega) \otimes G(\omega)$$

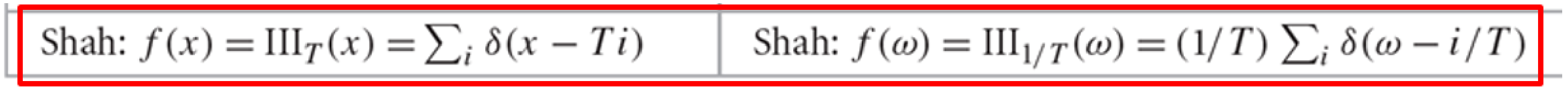
$$\mathcal{F}\{f(x) \otimes g(x)\} = F(\omega)G(\omega)$$

$$\Downarrow$$

$$\mathcal{F}\{\mathrm{III}_T(x)f(x)\} = F(\omega) \otimes \mathrm{III}_{1/T}(\omega)$$

Frequency domain analysis

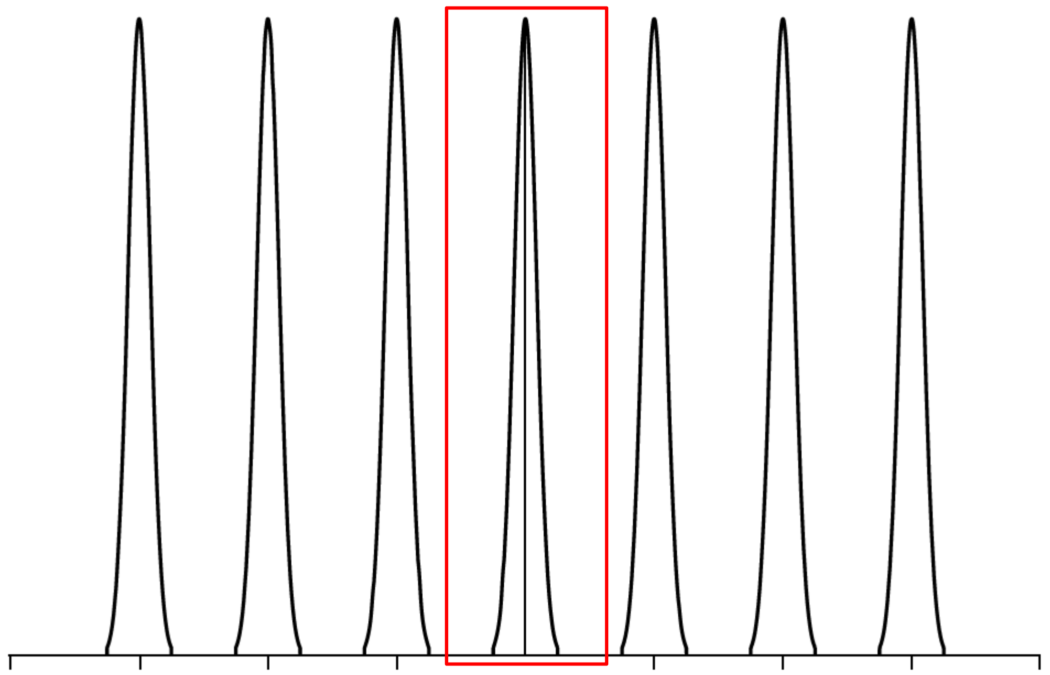
- **Apply to sampling formulation**
  - The Fourier transform of a shah function with period T is another shah function with period 1/T
  - It means that if the samples are farther apart in the spatial domain, they are closer together in the frequency domain
  - The frequency domain representation of the sampled signal is given by the convolution of *F*(*ω*) and this new shah function
  - Convolving with a shah function yields an infinite sequence of copies of *F*(*ω*)



Shah: $f(x) = \mathrm{III}_T(x) = \sum_i \delta(x - Ti)$	Shah: $f(\omega) = \mathrm{III}_{1/T}(\omega) = (1/T)\sum_i \delta(\omega - i/T)$

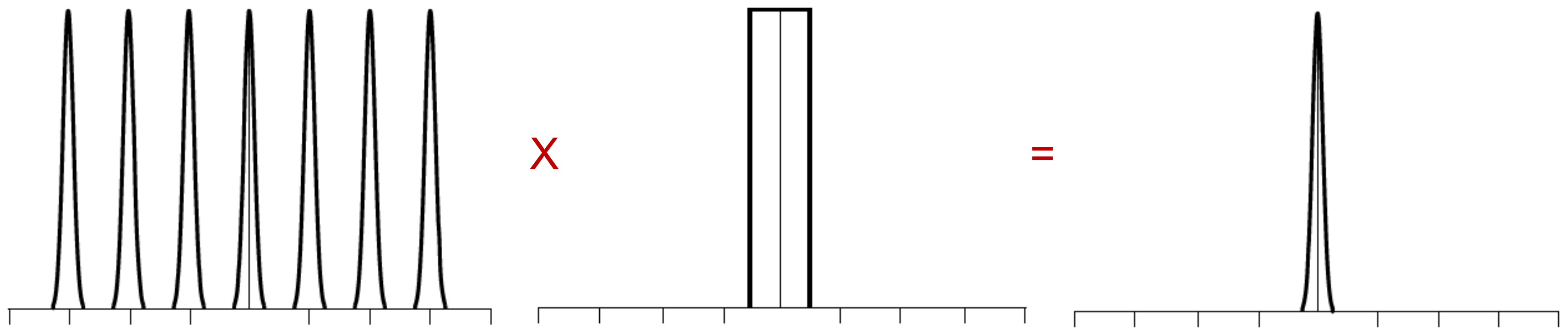
Frequency domain analysis

- **Frequency space representation of the series of samples**



Frequency domain analysis

- **How do we reconstruct the original function?**
  - Just discard all of the spectrum copies except the one centered at the origin
  - Multiply by a box function of the appropriate width



$$\Pi_T(x) = \begin{cases} 1/(2T) & |x| < T \\ 0 & \text{otherwise.} \end{cases}$$

Frequency domain analysis

- **How do we reconstruct the original function?**
  - This multiplication step corresponds to convolution with the reconstruction filter in the frequency domain

$$\tilde{F} = \big(F(\omega) \otimes \mathrm{III}_{1/T}(\omega)\big)\, \Pi_T(x)$$

  - Important result
    - We have been able to determine the ***exact*** frequency space representation of *f* (*x*), purely by sampling it at a set of regularly spaced points
    - No additional information except requiring band-limit property

Frequency domain analysis

- **Reconstruction in spatial domain**
  - Applying the equivalent process in the spatial domain will likewise recover *f*(*x*) exactly
  - Inverse Fourier transform of the box function is the sinc function, ideal reconstruction in the spatial domain is found by

$$\tilde{f} = \left(f(x)\mathrm{III}_T(x)\right) \otimes \mathrm{sinc}(x)$$

$$\Downarrow$$

$$\tilde{f}(x) = \sum_{i=-\infty}^{\infty} \mathrm{sinc}(x-i)\,f(i)$$

Frequency domain analysis

- **Unfortunately**
  - sinc function has infinite extent, it is necessary to <u>use all of the sample values *to compute any particular value f*(*x*) *in the spatial domain*</u>
  - Filters with finite spatial extent are preferable for practical implementations even though they don't reconstruct the original function perfectly

  - <u>*A key question*</u>
    - A filter that preserve frequency representation as much as possible while being localized in spatial domain

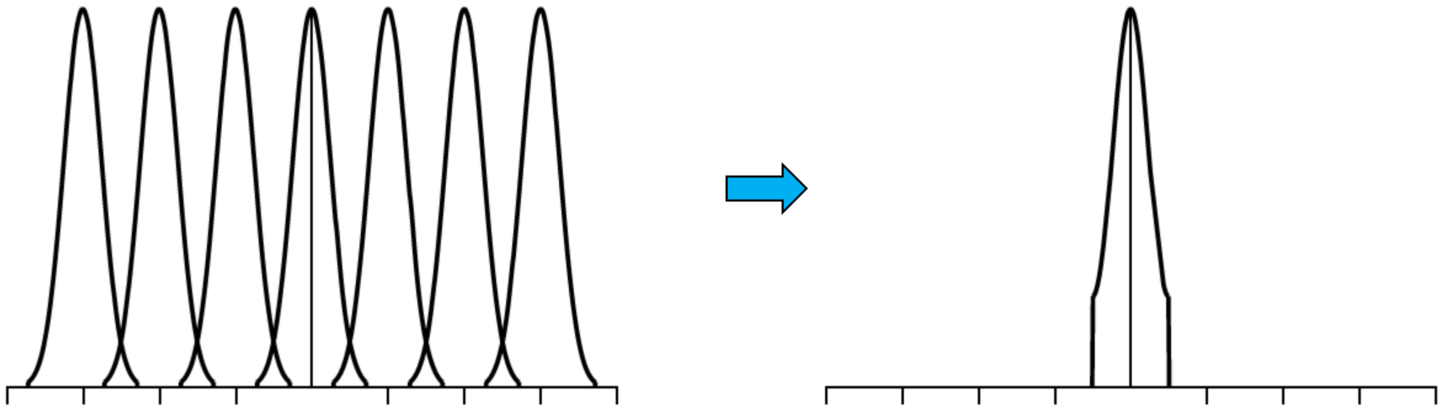
2. Aliasing

Aliasing

- **The most serious practical problems with ideal sampling and reconstruction**
  - The assumption that the signal is *band limited*
  - For signals that are not band limited, the signals aren't sampled at a sufficiently high sampling rate
  - The key to successful reconstruction is the ability to *exactly recover the original spectrum* $F(\omega)$

Aliasing

- **If the original function was sampled with a lower sampling rate**
  - Pushing the copies of the spectrum F(ω) closer together
  - If the copies get too close together, they start to overlap



Aliasing

- **A possible solution**
  - Simply increase the sampling rate until the copies of the spectrum do not overlap
- **Nyquist frequency**
  - Sample the signal with at least twice the maximum frequency
  - This minimum sampling frequency is called the *Nyquist frequency*

Aliasing

- **Non-band-limited signals**
  - Spectra with infinite support and will always overlap
  - Few of the interesting functions in computer graphics are band limited
  - In particular, any function containing a discontinuity (edge) cannot be band limited
  - It is necessary to apply <u>*different methods*</u> to counteract the aliasing error in a rendering process

3. Anti-aliasing techniques

Antialiasing techniques

- **Non-uniform sampling**
  - Turn the regular aliasing artifacts into noise
  - Less distracting to the human visual system

- **Adaptive sampling**
  - Take additional samples in regions with frequencies higher than the Nyquist limit
  - Finding all of the places where super-sampling is needed is difficult
  - In general, adjacent sample values cannot tell us with certainty what is really happening between them

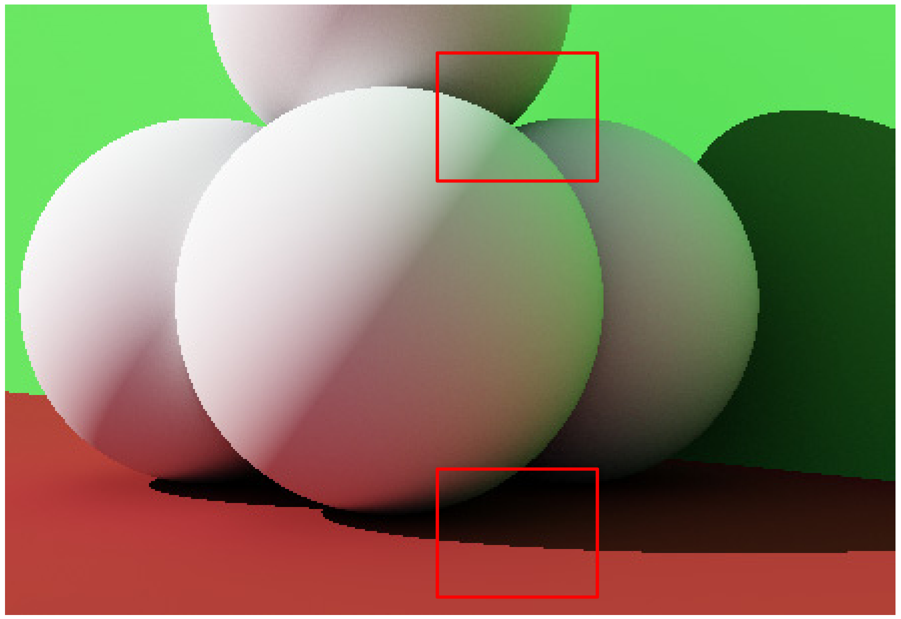
Antialiasing techniques

- **Pre-filtering**
  - Filter (i.e., blur) the original function
  - No high frequencies remain that can't be captured accurately at the sampling rate being used

- **Application to image synthesis**
  - <u>Good news</u>: with ray tracer, we can evaluate this function at any (x, y)
  - <u>Bad news</u>: it's not generally possible to remove (by pre-filtering) the high frequencies before sampling
  - *<u>Strategies</u>*: increasing the sampling rate + turn aliasing into noise

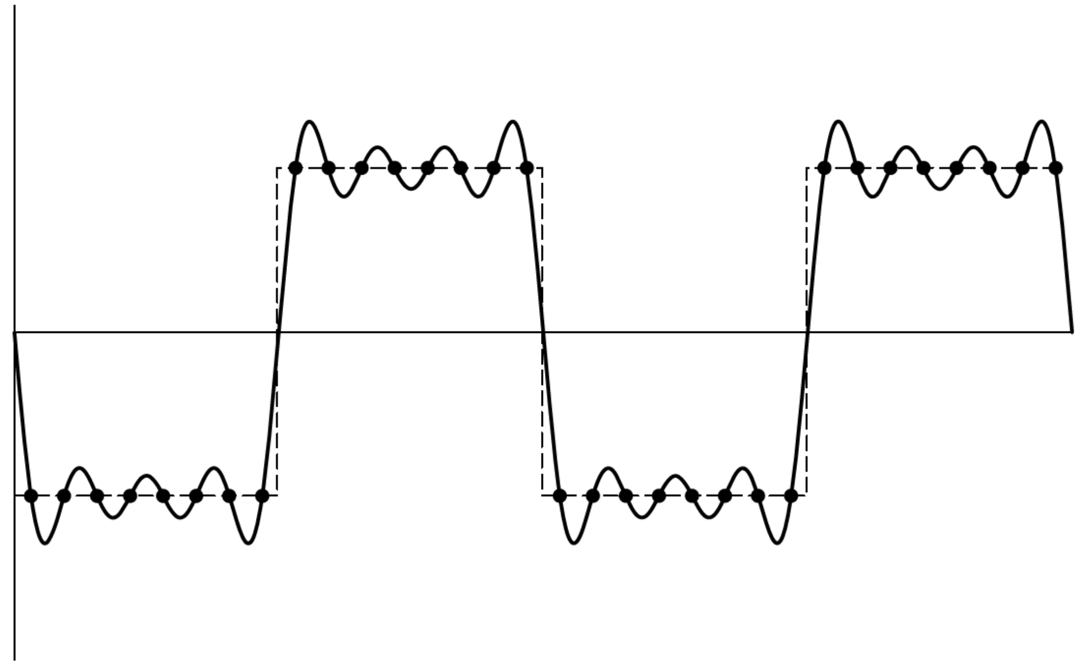
Sources of aliasing in rendering

- **Geometry**
  - One of the most common causes of aliasing
  - Object's boundary introduces a step function



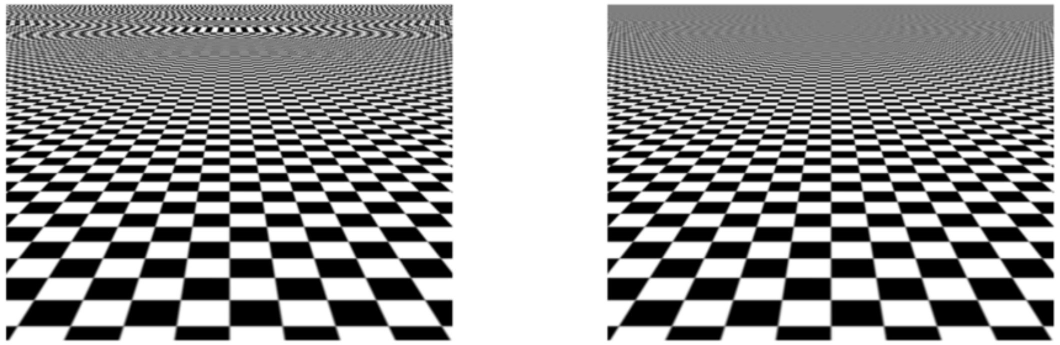
Sources of aliasing in rendering

- **The perfect (sinc) reconstruction filter**
  - Cause artifacts when applied to aliased samples
  - Ringing artifacts appear in the reconstructed function
  - Such an effect is known as the *Gibbs phenomenon*



Sources of aliasing in rendering

- **Texture and materials on an object**
  - Texture maps that haven't been filtered correctly
  - Small highlights on shiny surfaces



- **The key insight about aliasing in rendered images**
  - We can never remove all of its sources
  - Develop techniques to mitigate its impact on the quality of the final image

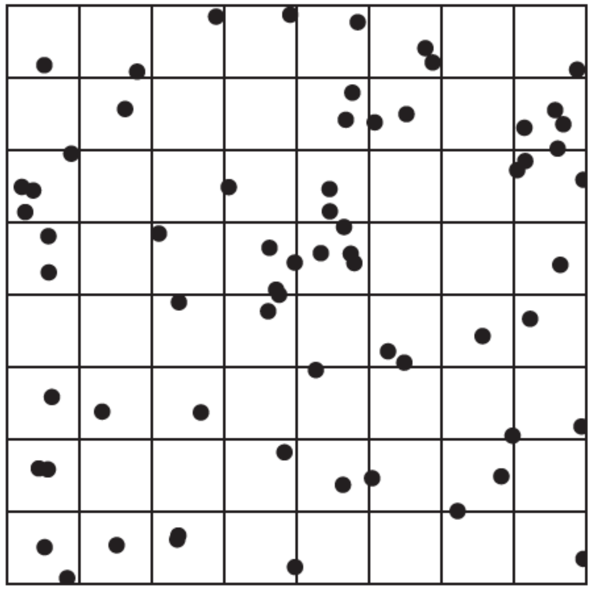
4. Sampling

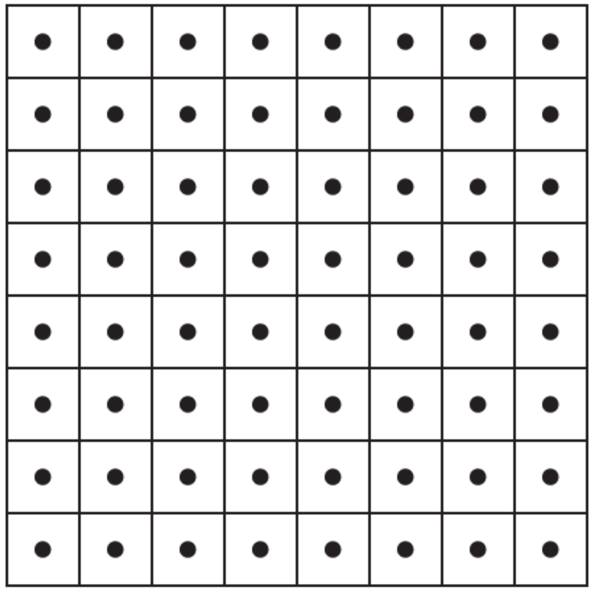
Stratified sampling

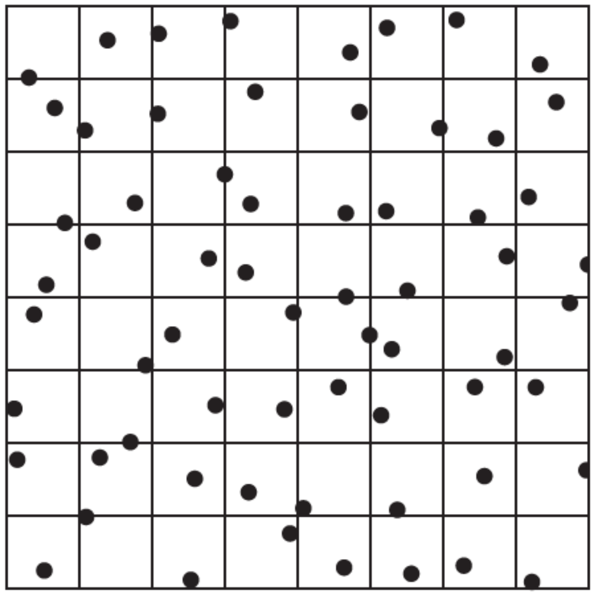
- **Divide the image plane into rectangular regions**
  - Generate a single sample inside each region
  - These regions are commonly called *strata*
  - Place each sample at a random point inside each stratum by *jittering*
  - The non-uniformity from this jittering helps turn aliasing into noise

Stratified sampling

- **Sampling comparison**

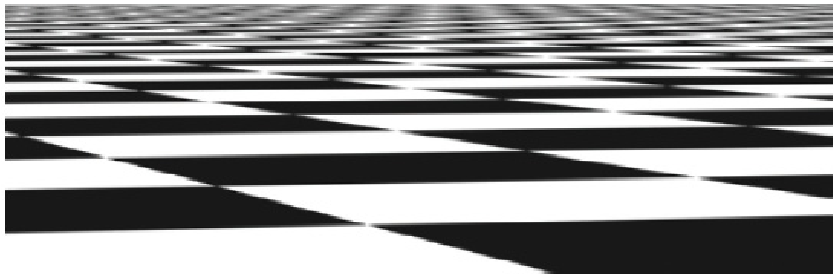

(a) Random sampling over entire region


(b) Uniform sampling


(c) Stratified jittered sampling

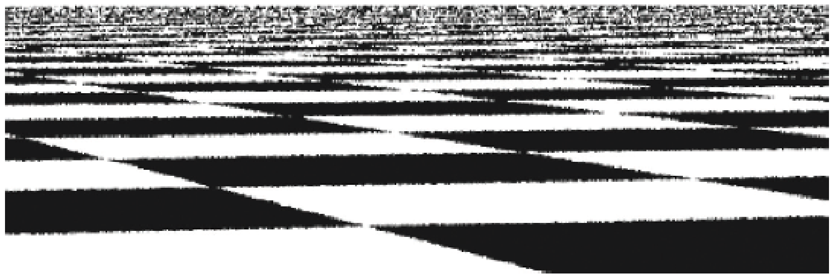
Stratified sampling

- **Rendering comparison**

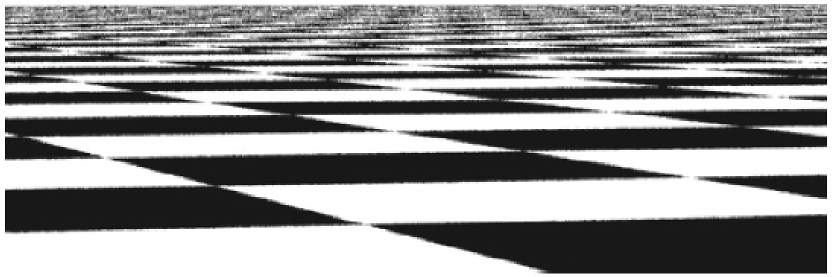


(a) A reference image, rendered with 256 samples per pixel

(b) An image rendered with one sample per pixel, without jittering



(c) The result of jittering the image samples, with only one sample per pixel



(d) The result of four jittered samples per pixel

Low discrepancy sampling

- **Why stratified sampling is not good enough?**
  - Samples may not be uniform
    - Magnify the noise
  - Require random but uniform sampling pattern

- **Discrepancy definition**
  - A volume in n-dimension $B = \{[0, v_1] \times [0, v_2] \times \cdots \times [0, v_s]\}$
  - A sequence of sample points $P = x_1, \ldots, x_N$
  - The discrepancy of P with respect to B is:

$$D_N(B, P) = \sup_{b \in B} \left| \frac{\sharp\{x_i \in b\}}{N} - \lambda(b) \right|$$

$\sharp\{x_i \in b\}$ is the number of points in b

$\lambda(b)$ is the volume of b

Low discrepancy sampling

- **Radical inverse**
  - A positive integer value n can be expressed in a base b with a sequence of digits $d_m \ldots d_2 d_1$($d_i \in [0, b-1]$):

$$n = \sum_{i=1}^{\infty} d_i b^{i-1}$$

  - The radical inverse function Φ_b in base b converts a nonnegative integer n to a floating-point value in [0,1)

$$\Phi_b(n) = 0.d_1 d_2 \ldots d_m$$

Low discrepancy sampling

- **Halton and Hammersley sequences**
  - Halton sequence
    - To generate n-dimensional Halton sequence, we use the radical inverse base *b*, with different base for each dimension of the pattern
    - The basis used must all be relatively prime to each other
    - A natural choice is to use the first n prime numbers ($p_1,\ldots,p_n$)

$$x_i = (\Phi_2(i), \Phi_3(i), \Phi_5(i), \ldots, \Phi_{p_n}(i))$$

    - The discrepancy of Halton sequence in d-dimensional space is

$$D_N(x_i) = O\left(\frac{(\log N)^d}{N}\right)$$

Low discrepancy sampling

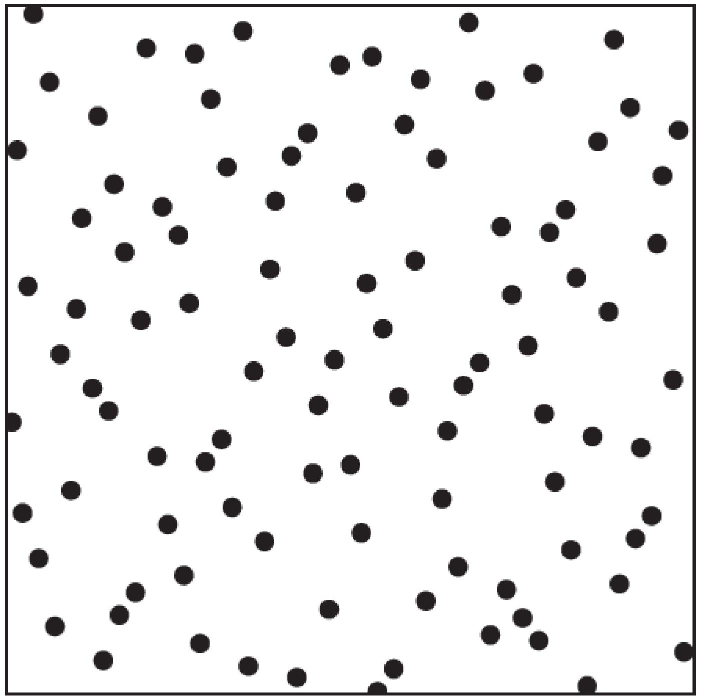
- **Halton and Hammersley sequences**
  - Hammersley sequence
    - If the number of samples *N* is fixed, the Hammersley point set can be used

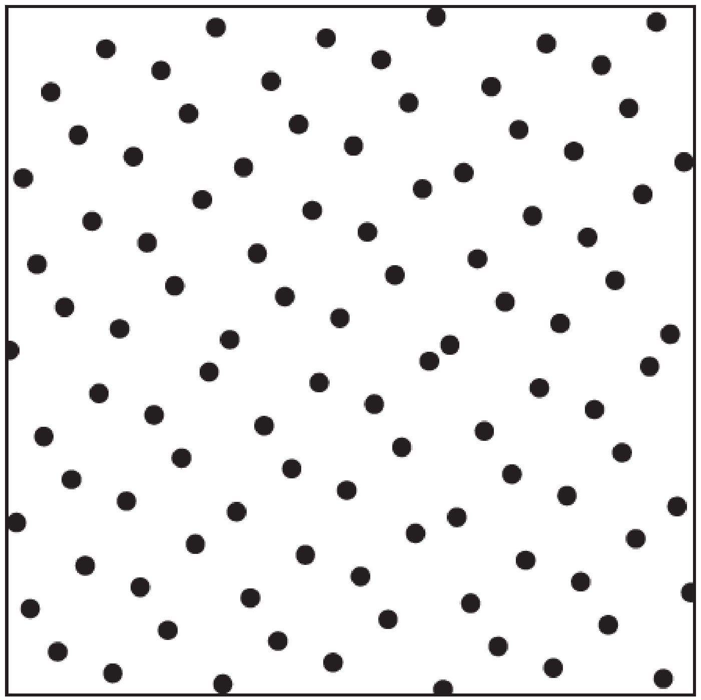
$$x_i = \left(\frac{i}{N}, \Phi_{b_1}(i), \Phi_{b_2}(i), \ldots, \Phi_{b_n}(i) \right)$$

    - The basis b_i are relatively prime

Low discrepancy sampling

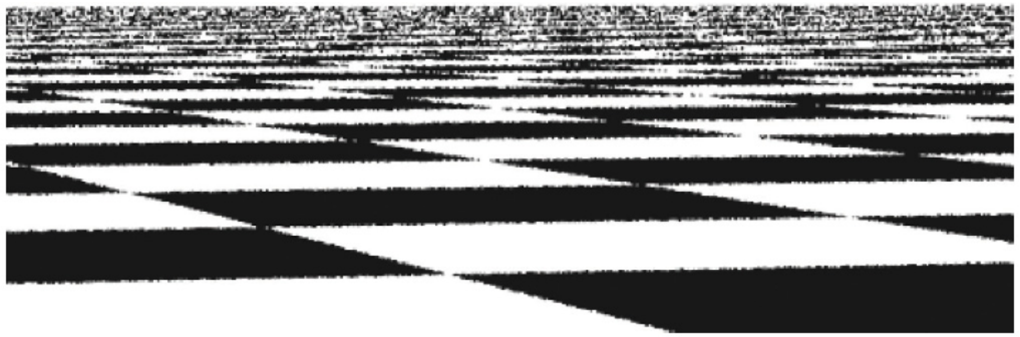
- **Halton and Hammersley sequences**

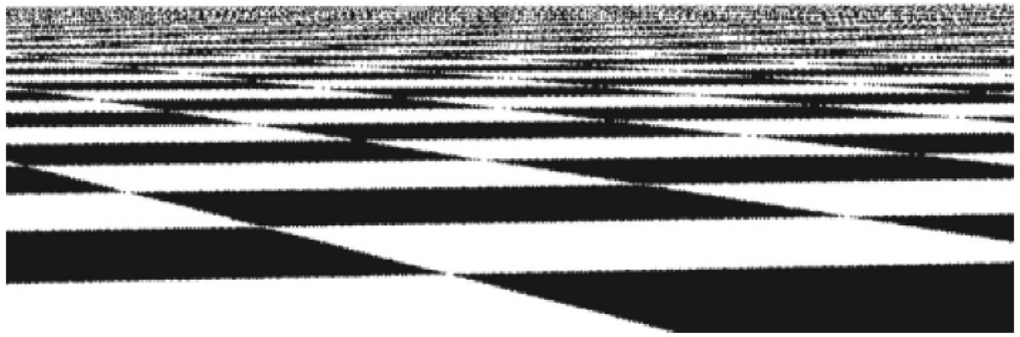

Sampling based on Halton sequence


Sampling based on Hammersley sequence

Low discrepancy sampling

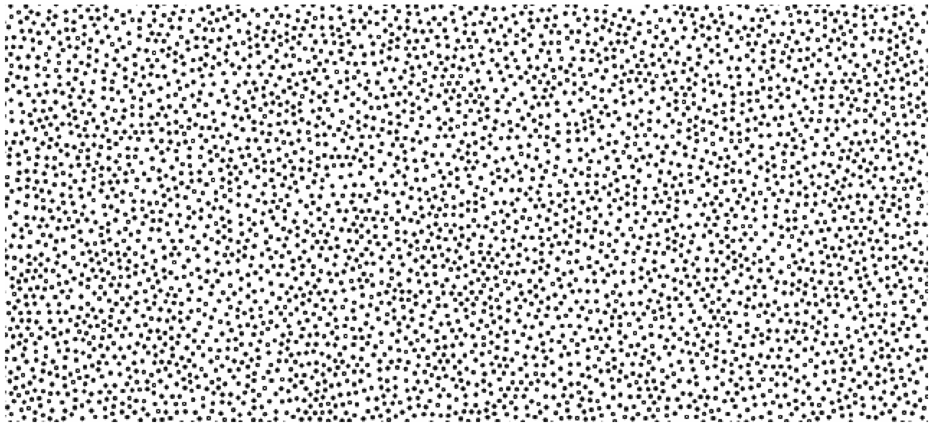
- **Rendering comparison**


(a) The jittered stratified sampler with a single sample per pixel and


(b) the Halton sequence sampler with a single sample per pixel.

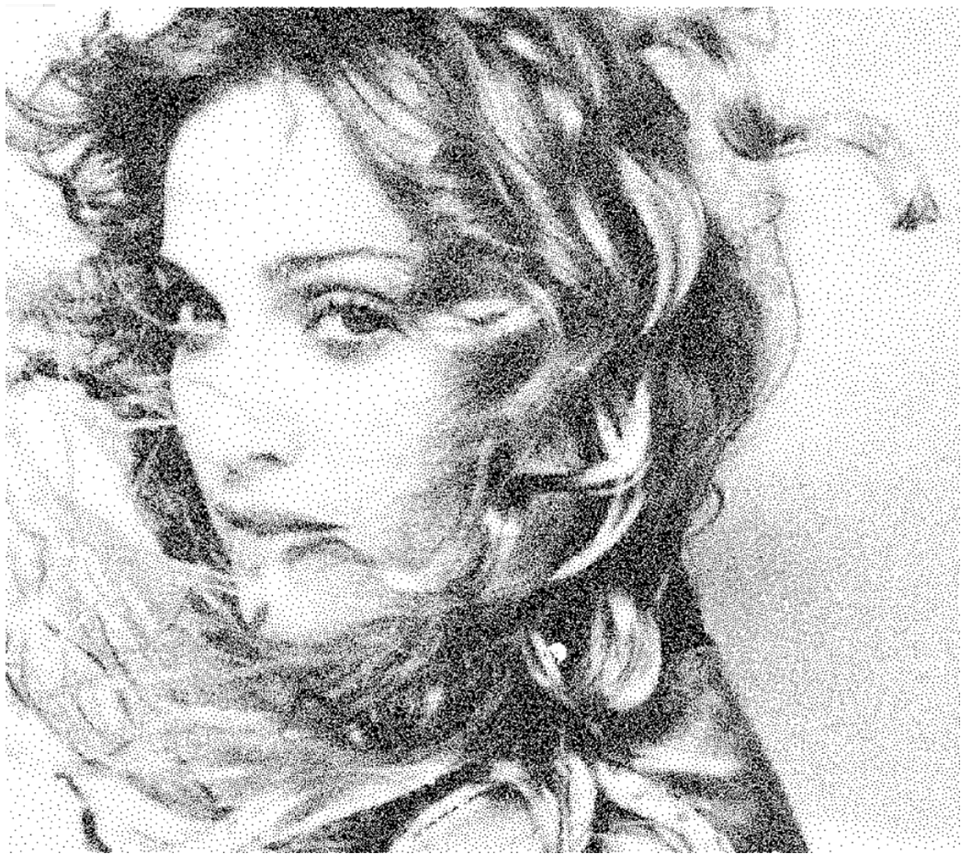
Poisson-disk sampling

- **Poisson disk pattern**
  - Well-separated sample placement which has been shown to be an excellent image sampling pattern
  - An easy way to generate Poisson disk patterns is with *dart throwing*, but there is an O(N) algorithm proposed by Robert Bridson
    - Fast Poisson Disk Sampling in Arbitrary Dimensions, SIGGRAPH 2007



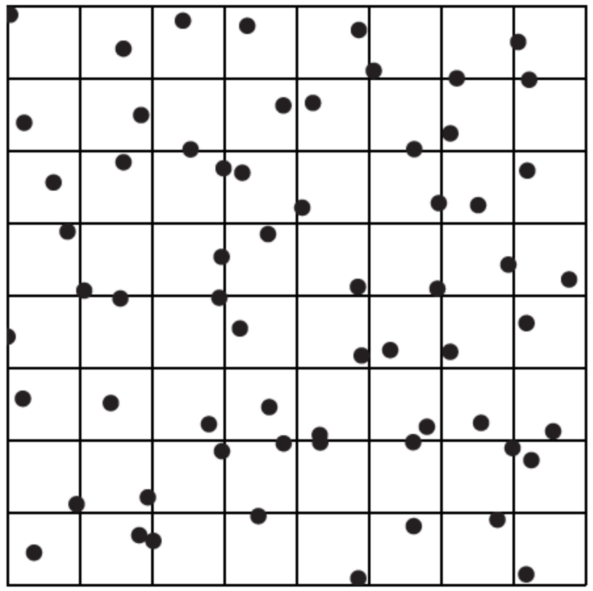
Poisson-disk sampling

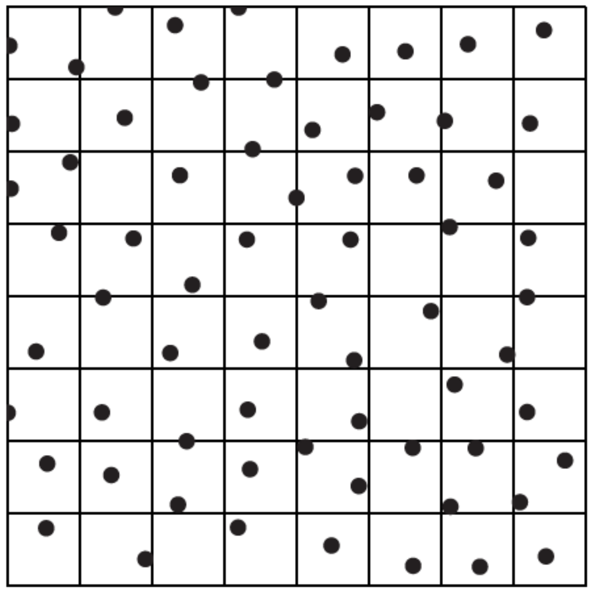
- **Examples to sample an image**



Best-candidate sampling

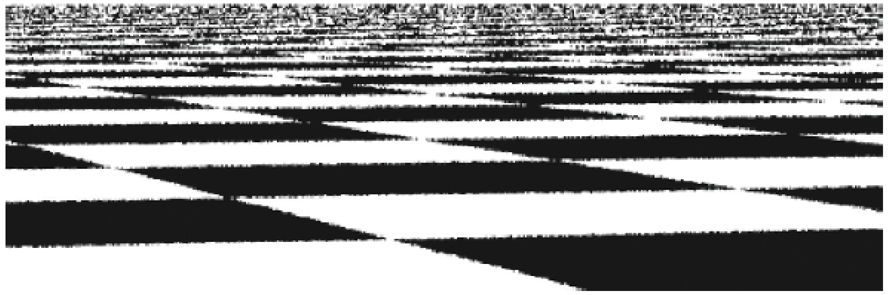
- **Don Mitchell's best-candidate algorithm**
  - Each time a new sample is to be computed
    - A large number of random candidates are generated
    - The candidates are compared to the previous samples
    - The one farthest away is added to the pattern

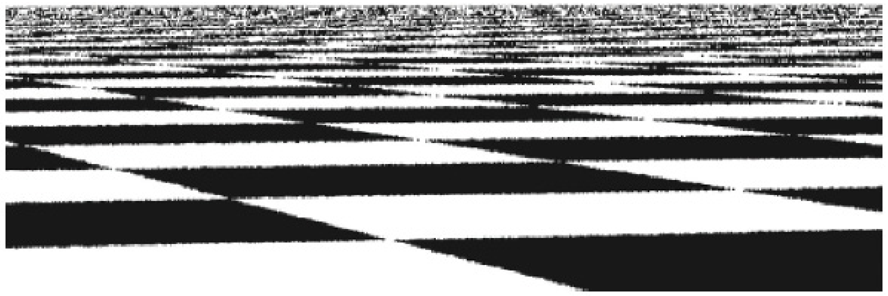

A jittered pattern

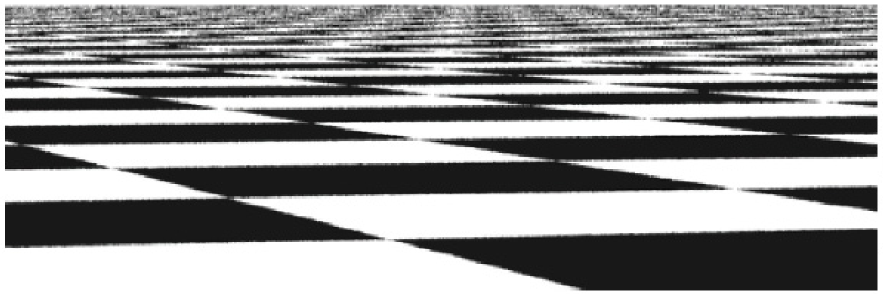

A best-candidate generated pattern

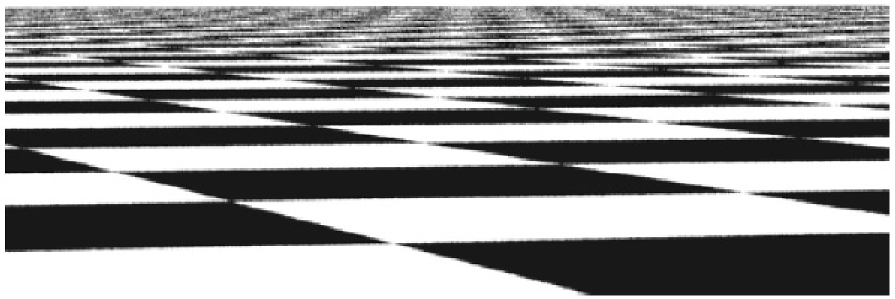
Best-candidate sampling

- **Rendering comparison**


(a) The stratified pattern with a single sample per pixel


(b) The best-candidate pattern with a single sample per pixel


(c) The stratified pattern with four samples per pixel


(d) The four-sample best-candidate pattern

5. Image reconstruction

Image reconstruction

- **Given image samples, we need to do three things to get final pixel values**
  - 1. Reconstruct a continuous image function $\tilde{L}$ from image samples
  - 2. Pre-filter the function $\tilde{L}$ to remove any frequencies past the Nyquist limit
  - 3. Sample $\tilde{L}$ at the pixel locations to obtain final pixel values

- **Note**
  - Since we will resample $\tilde{L}$ at only pixel locations, it's not necessary to construct an explicit representation
  - Combine the first two steps using a single filter function

Image reconstruction

- **Keep in mind that**
  - Ideal reconstruction depends on uniform sampling
  - Perfect reconstruction isn’t generally possible in practice
  - The purpose of reconstruction focuses on minimizing reconstruction error
  - Experience tells that applying perfect reconstruction techniques to samples for image synthesis generally does not result in the highest-quality images

Image reconstruction

- **Interpolate samples near a pixel for reconstruction**
  - Computing a weighted average

$$I(x, y) = \frac{\sum_i f(x - x_i, y - y_i) L(x_i, y_i)}{\sum_i f(x - x_i, y - y_i)}$$

- $L(x_i, y_i)$: the radiance value of the i-th sample at (x_i, y_i)
- f is a filter function

Image reconstruction

- **Pixel at location (x,y)**
  - Sinc filter is not appropriate here
  - May produce Gibbs phenomenon

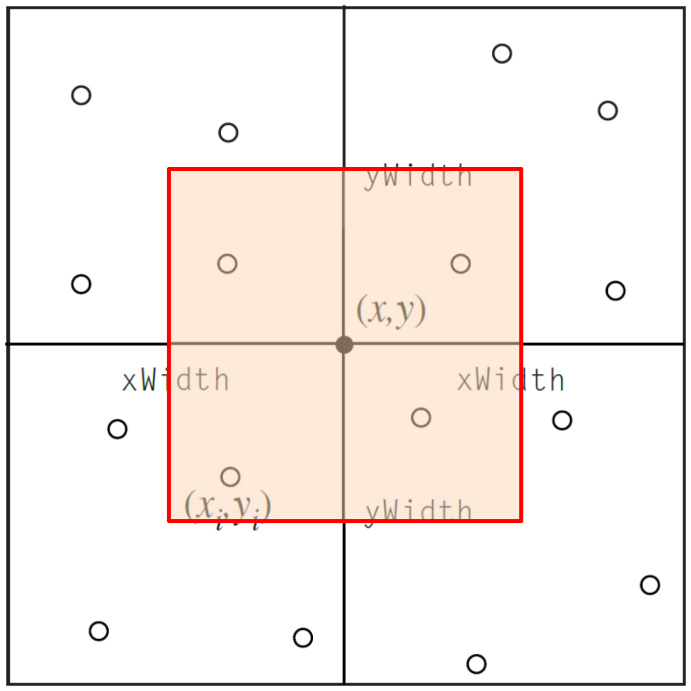


Image reconstruction

- **Filter functions**
  - Box filter
    - Equally weights all samples within a square region of the image
    - Efficient, but about the worst filter possible

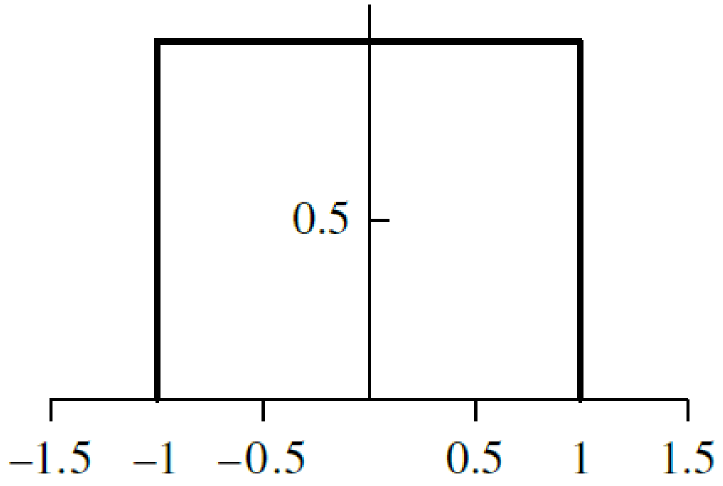
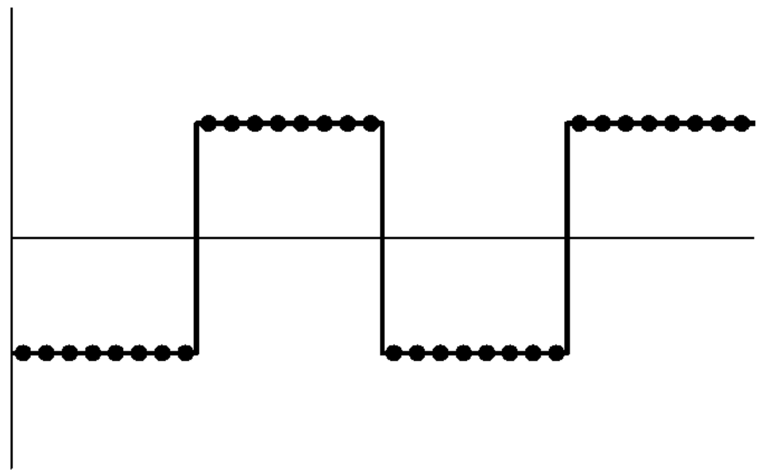


Image reconstruction

- **Filter functions**
  - Box filter
    - Comparison for reconstructing different signals


(a)

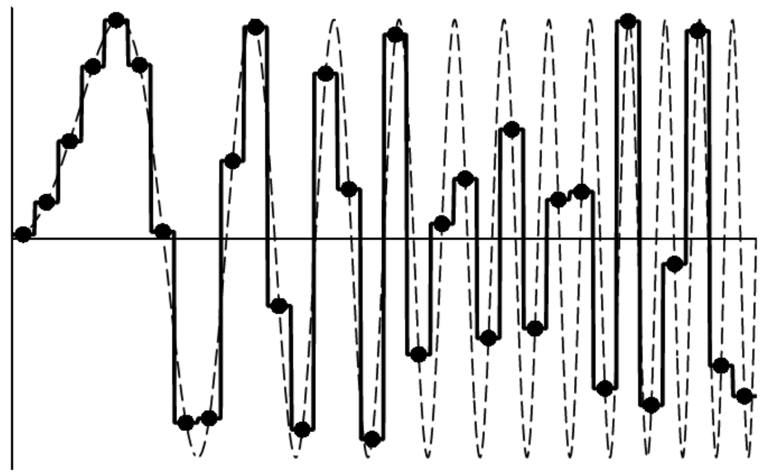

(b)

Image reconstruction

- **Filter functions**
  - Triangle filter
    - Slightly better result than box filter

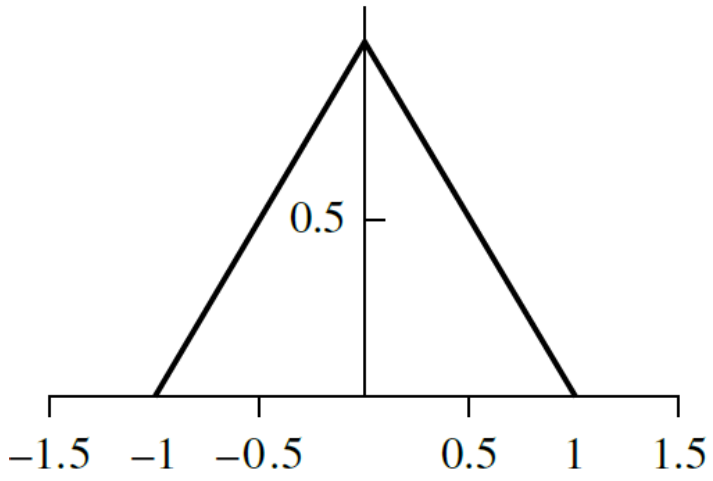


Image Reconstruction

- **Filter functions**
  - Gaussian filter
    - Give a reasonably good result in practice
    - Tend to cause slight blurring of the final image compared to some of the other filters

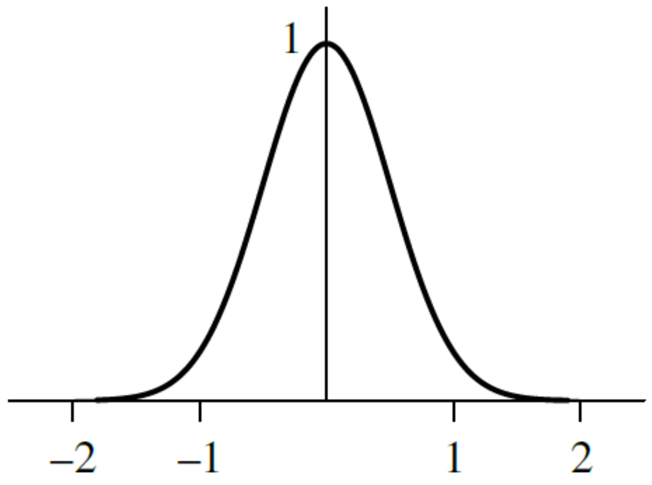


Image reconstruction

- **Filter functions**
  - Mitchell filter
    - Trade-off between *ringing* and *blurring*
    - This filter function takes on negative values, which improves the sharpness of the edges
    - Final pixel values can therefore become negative: clamping

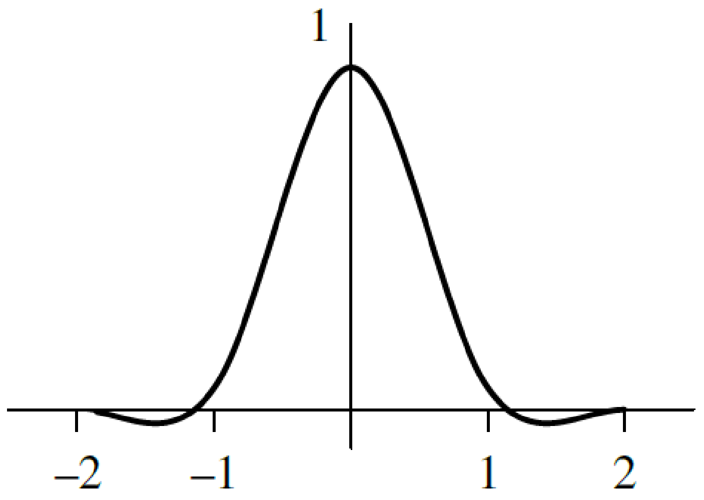


Image reconstruction

- **Filter functions**
  - Mitchell filter
    - Comparison for reconstructing different signals

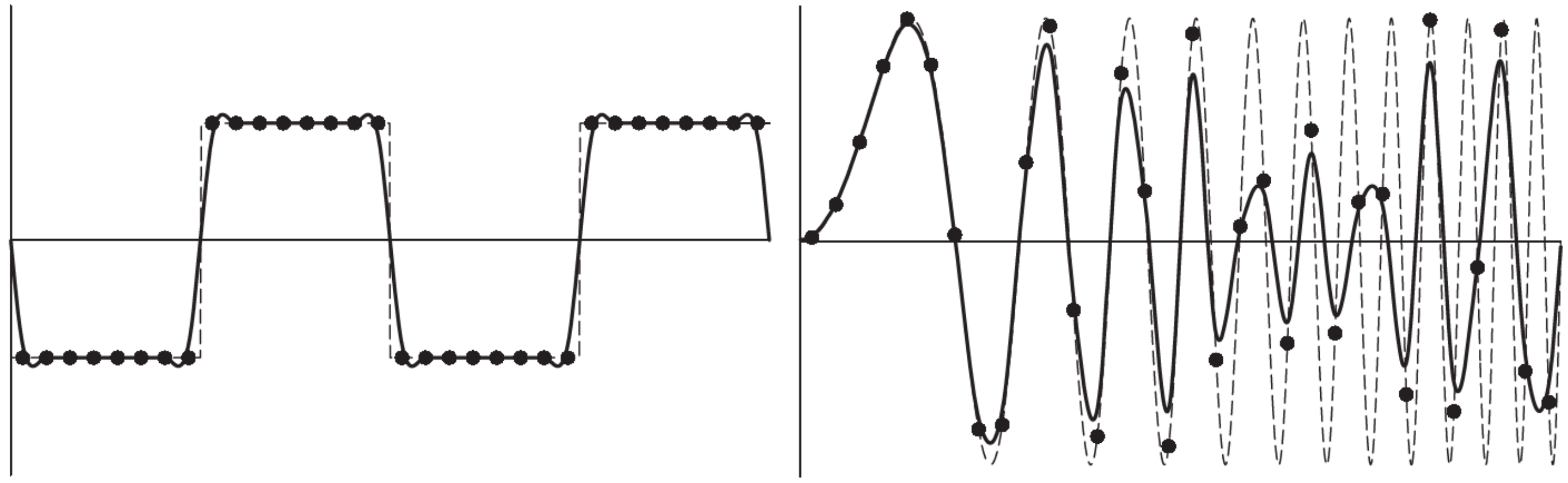
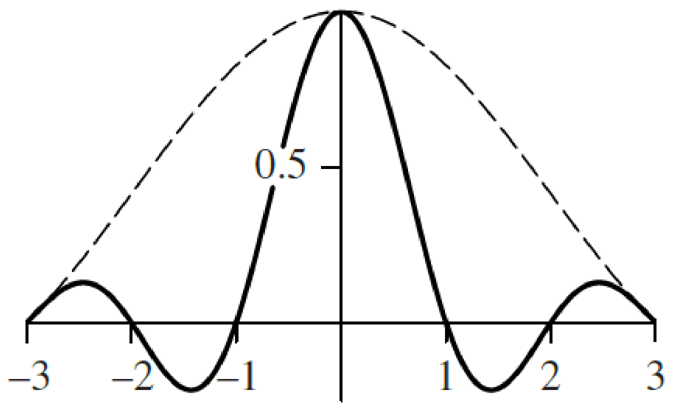


Image reconstruction

- **Filter functions**
  - Windowed-sinc filter
    - In practice, the sinc filter is often multiplied by another function that goes to zero after some distance
    - Multiply sinc function with a Lanczos windowing function



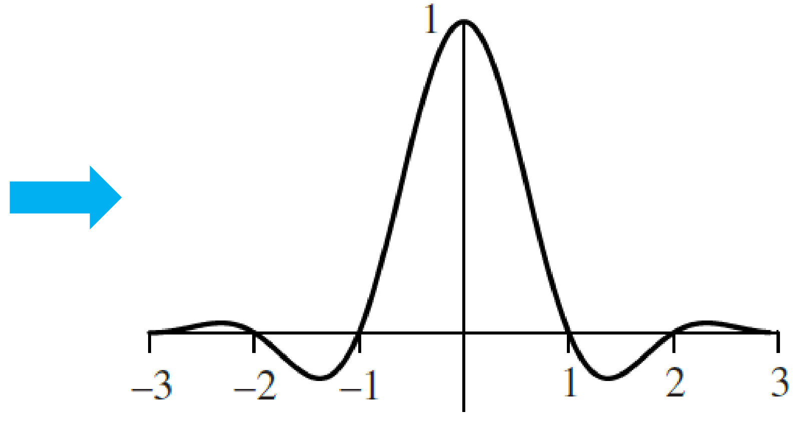


Image reconstruction

- **Filter functions**
  - Windowed-sinc filter
    - Comparison for reconstructing different signals

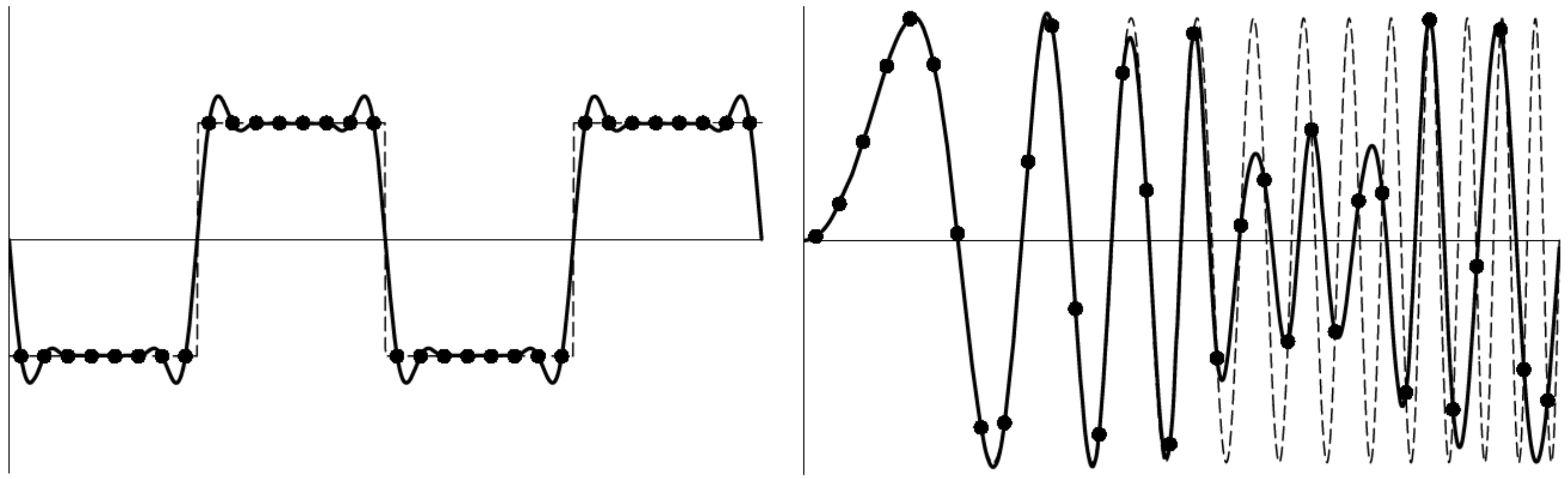
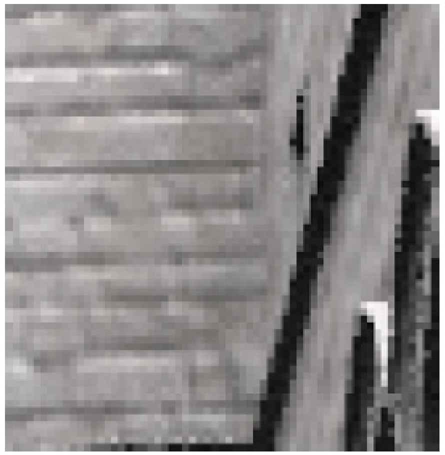
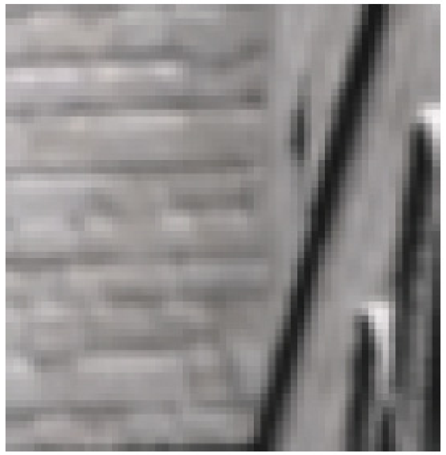


Image reconstruction

- **Different filter effects**


Box filter


Gaussian filter

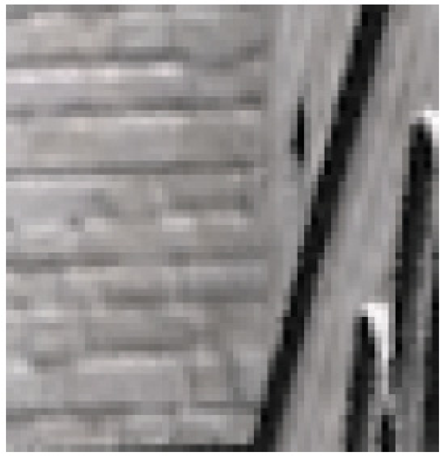
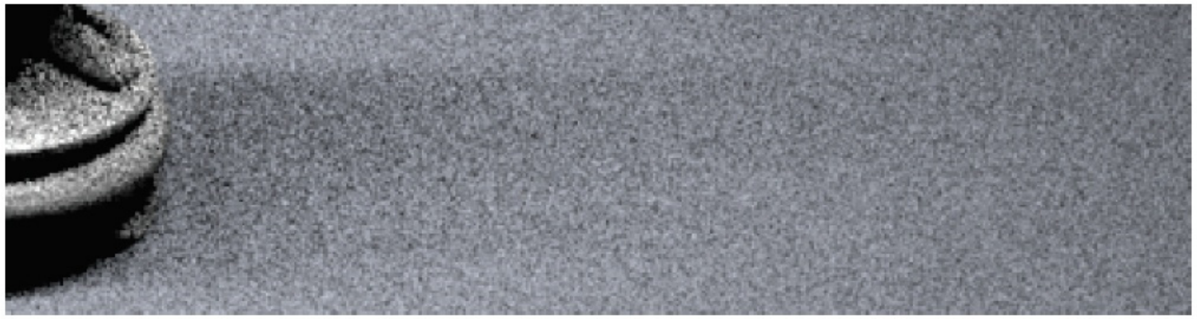

Mitchell filter

Image reconstruction

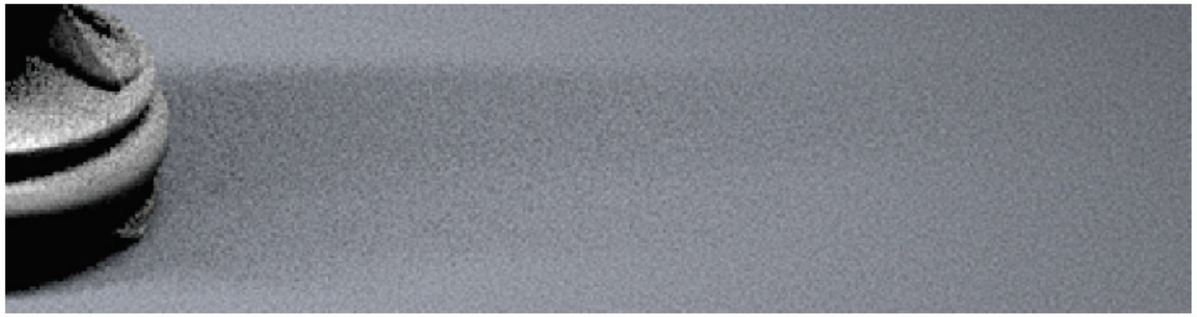
- **Choice of pixel reconstruction filter**
  - Mitchell filter has been used to reconstruct pixels in the soft shadows example

Stratified sampling



(a)

Low-discrepancy sampling



(b)

Next lecture: Numerical integration