Computer Graphics

Sampling

Konstantin Tretyakov
kt@ut.ee

Nov 06, 2013

Quiz

- Name a popular normal mapping technique.
- Name two environment mapping techniques.
- Name three techniques for implementing shadows in the standard graphics pipeline.
- Name an algorithm that “got an Oscar”.

Quiz

- What is a *picture*?

What is a picture?

- What is a *picture*?
  - A picture is a **function of two variables**

$$p(x, y)$$

  - In general, $x, y \in \mathbb{R}$

What is a picture?

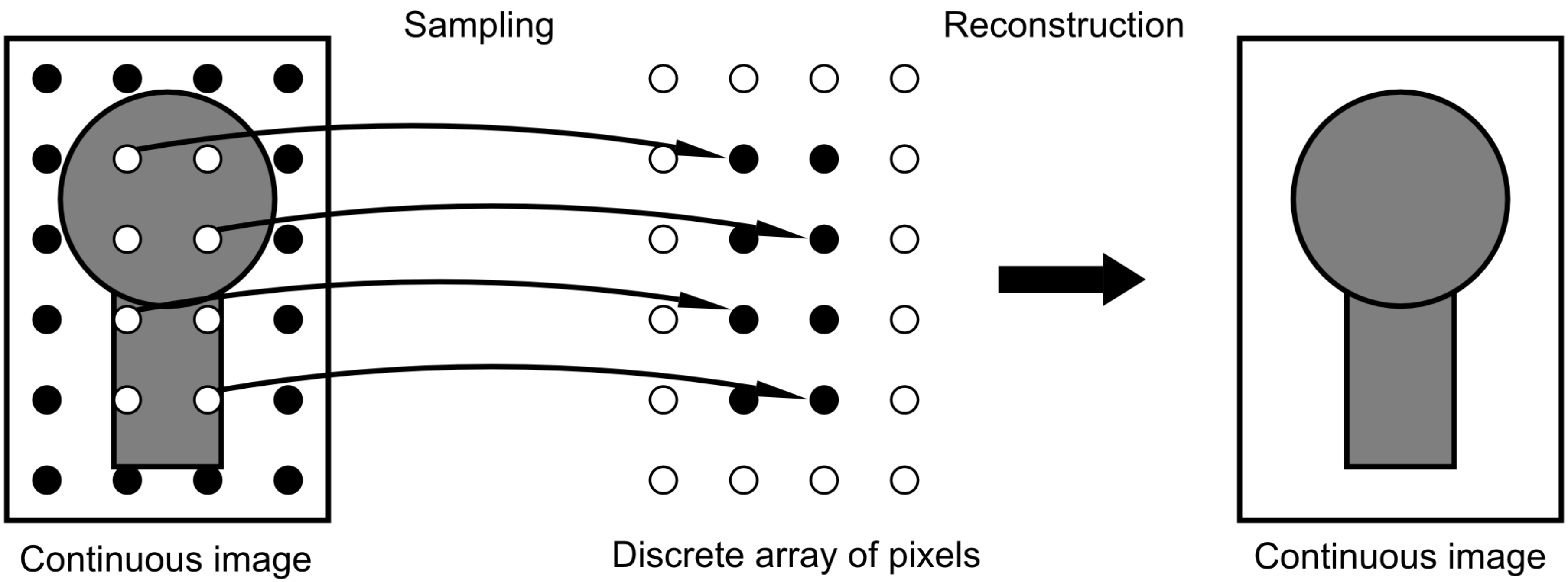
- How can you **store** a function of two variables?

What is a picture?

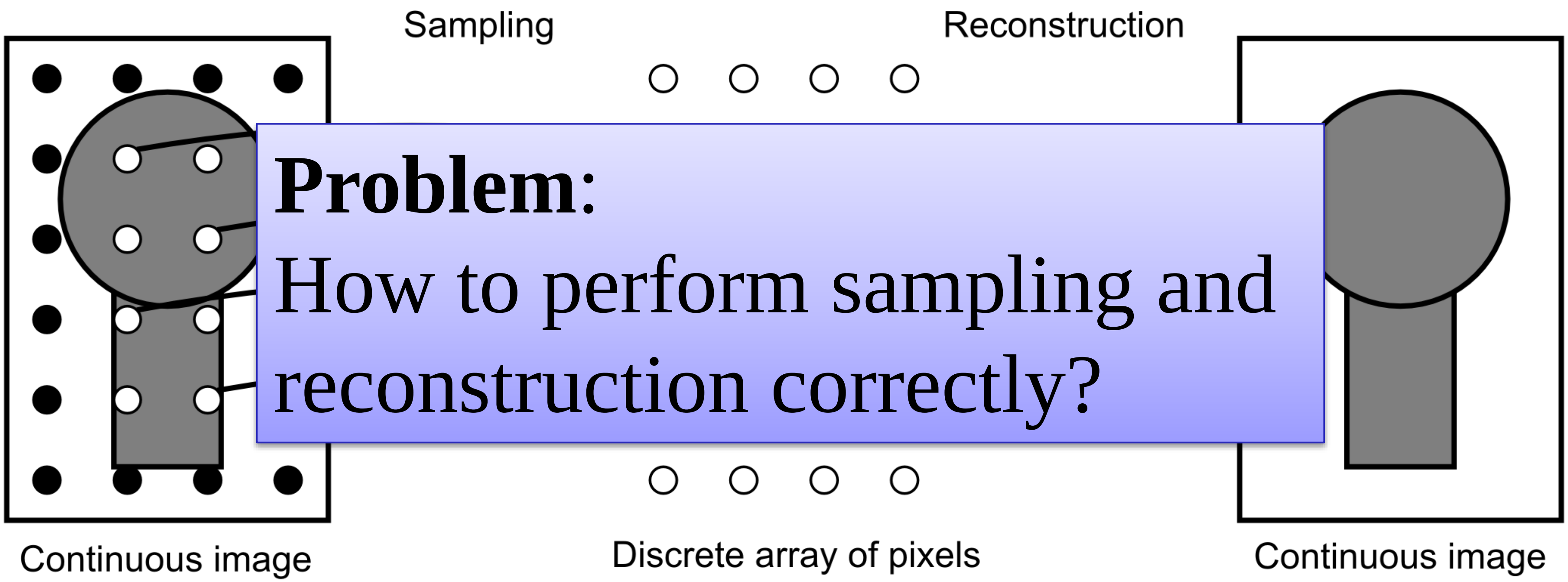
- How can you **store** a function of two variables?
  - Analytically, e.g: $f(x, y) = x^2 + y$
  - By storing a *sample* measured at a finite number of discrete points – *pixels*:

$$\{ \ p(x_1, y_1), p(x_2, y_2), \dots, p(x_n, y_n) \ \}$$

Sampling & Reconstruction



Sampling & Reconstruction



Problem:
How to perform sampling and reconstruction correctly?

Examples

- **Sampling**:
  - Pixels stored in an image file
  - Rays in a raytracing algorithm
  - Z-buffer values
  - Movie frames (*temporal* sampling)
- **Reconstruction**:
  - Image rendering to display (CRT, LCD, …)
  - Showing a video as a sequence of frames
  - Texturing

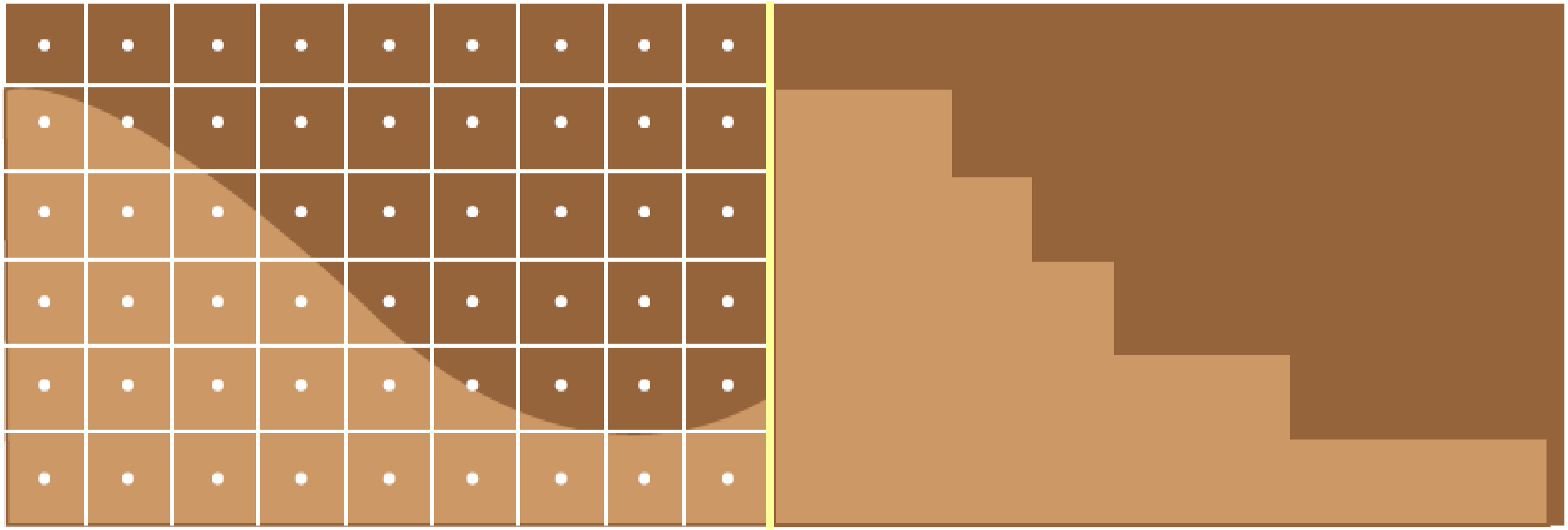
Sampling & reconstruction

- Ideally, we would like the discretization-reconstruction process to be perfect.
- In reality, it is often impossible.
  In this case we would like to at least **avoid introducing things that were not present in the original image**.

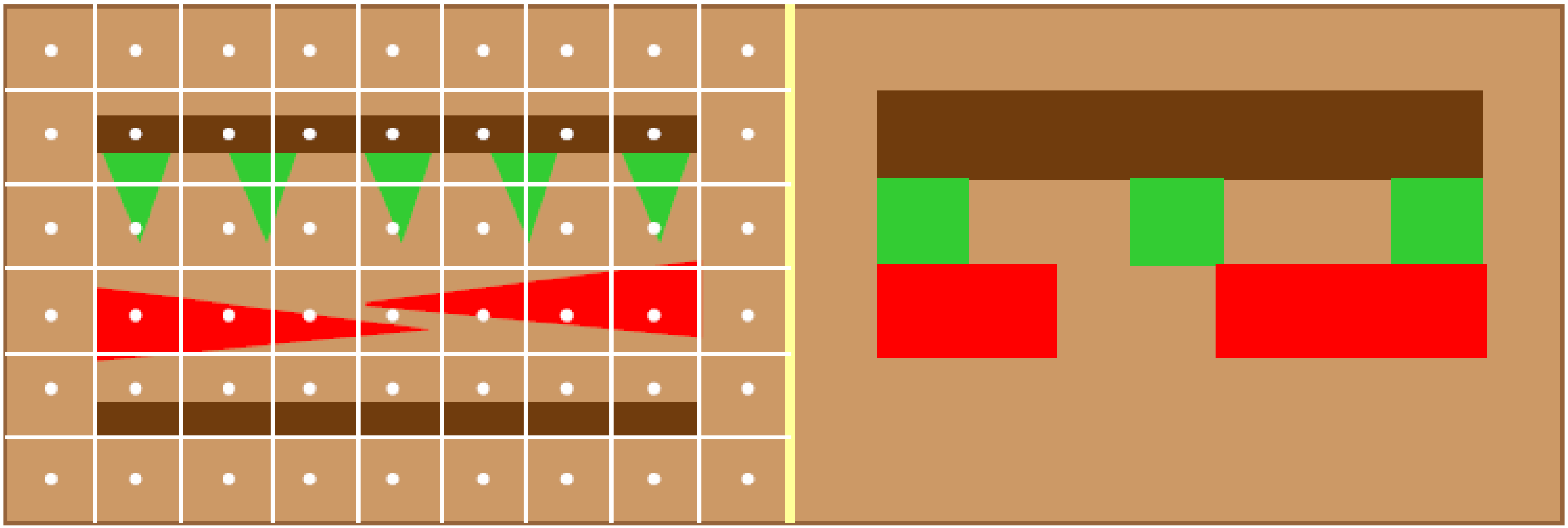
Sampling & reconstruction

- Incorrect sampling introduces *aliasing artifacts:*
  - Jagged edges. Incorrect tiny details. Moiré effects.
- Incorrect reconstruction usually results in less important errors:
  - “Visible pixels”, flashing frames in a movie.

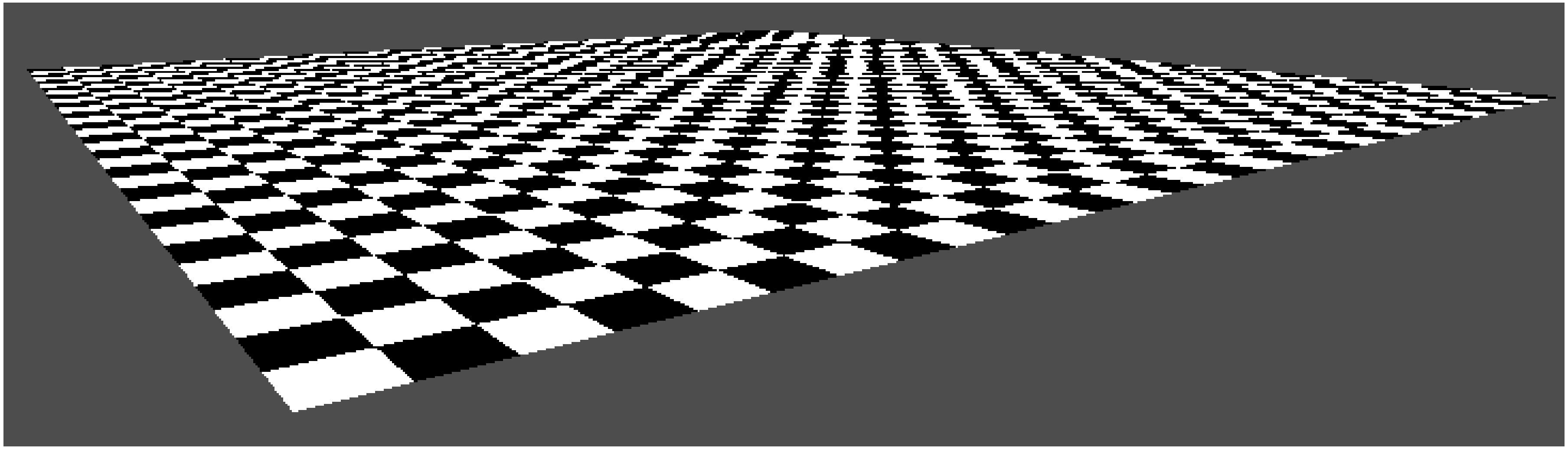
Aliasing: Jagged edges



Aliasing: Improper detail



Aliasing: Texture artifacts

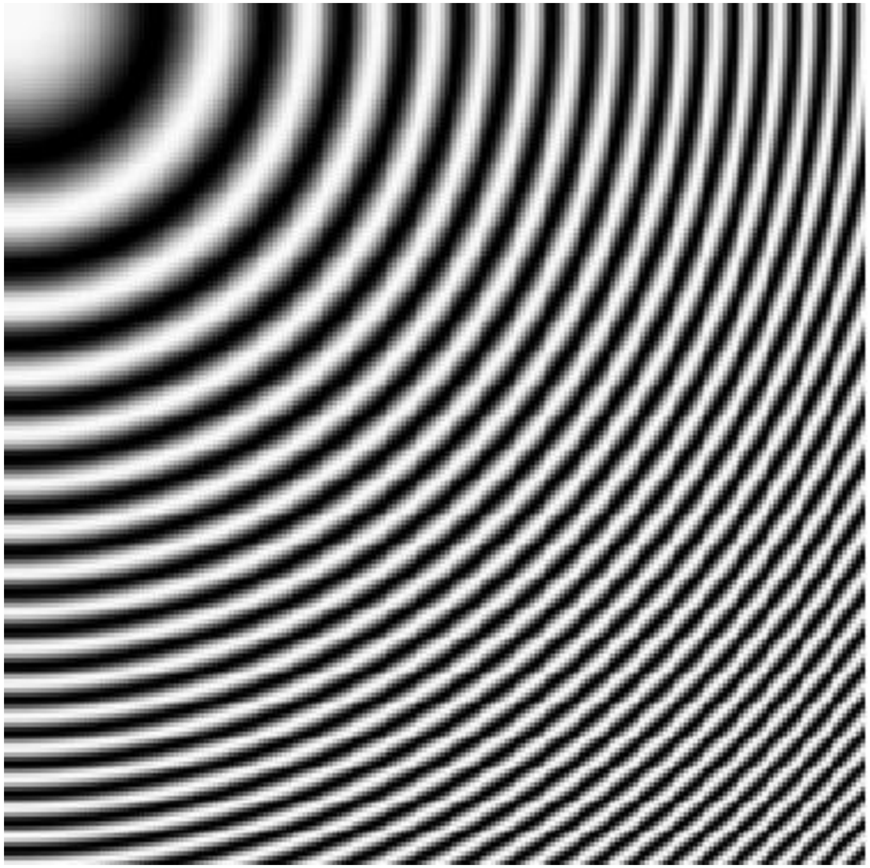


Aliasing: Moiré effects

- Consider a picture $p(x, y) = \cos(x^2 + y^2)$

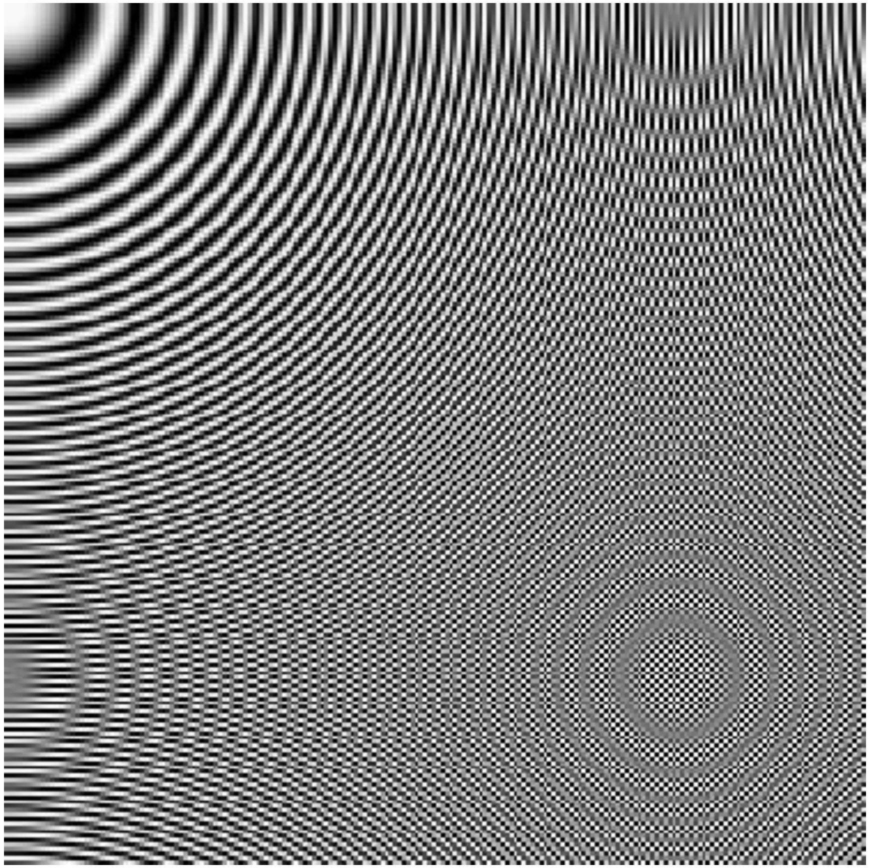
- Discretize it into a 200x200 array:
  - with step 0.05
  - with step 0.10
  - with step 0.20

Aliasing: Moiré effects



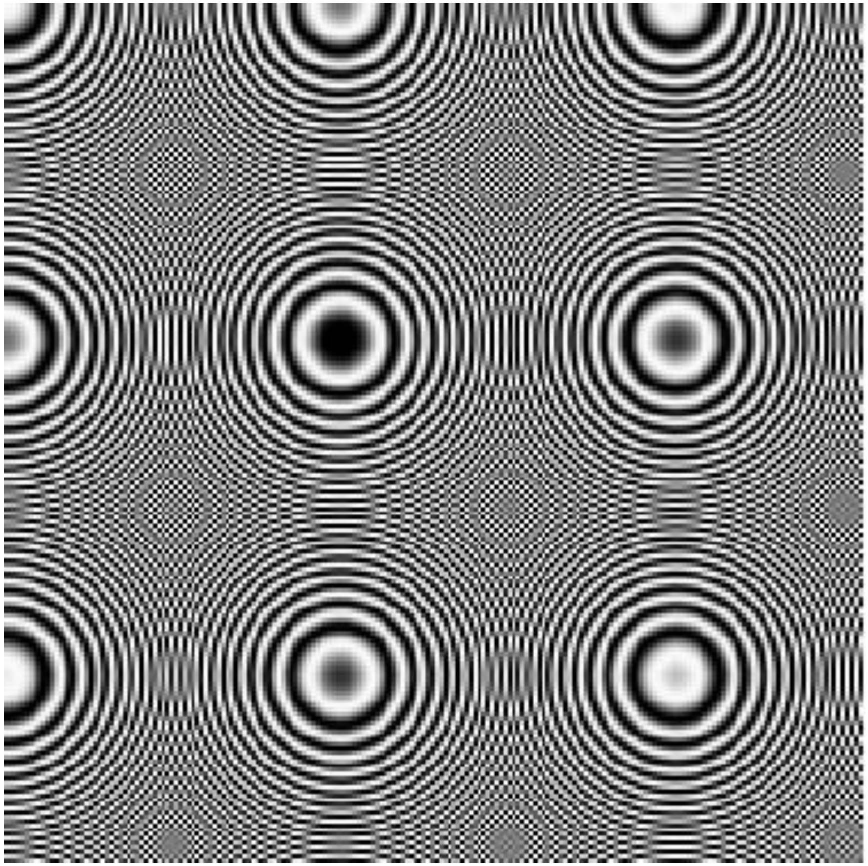
$$p_{ij} = p(0.05i, 0.05j)$$

Aliasing: Moiré effects



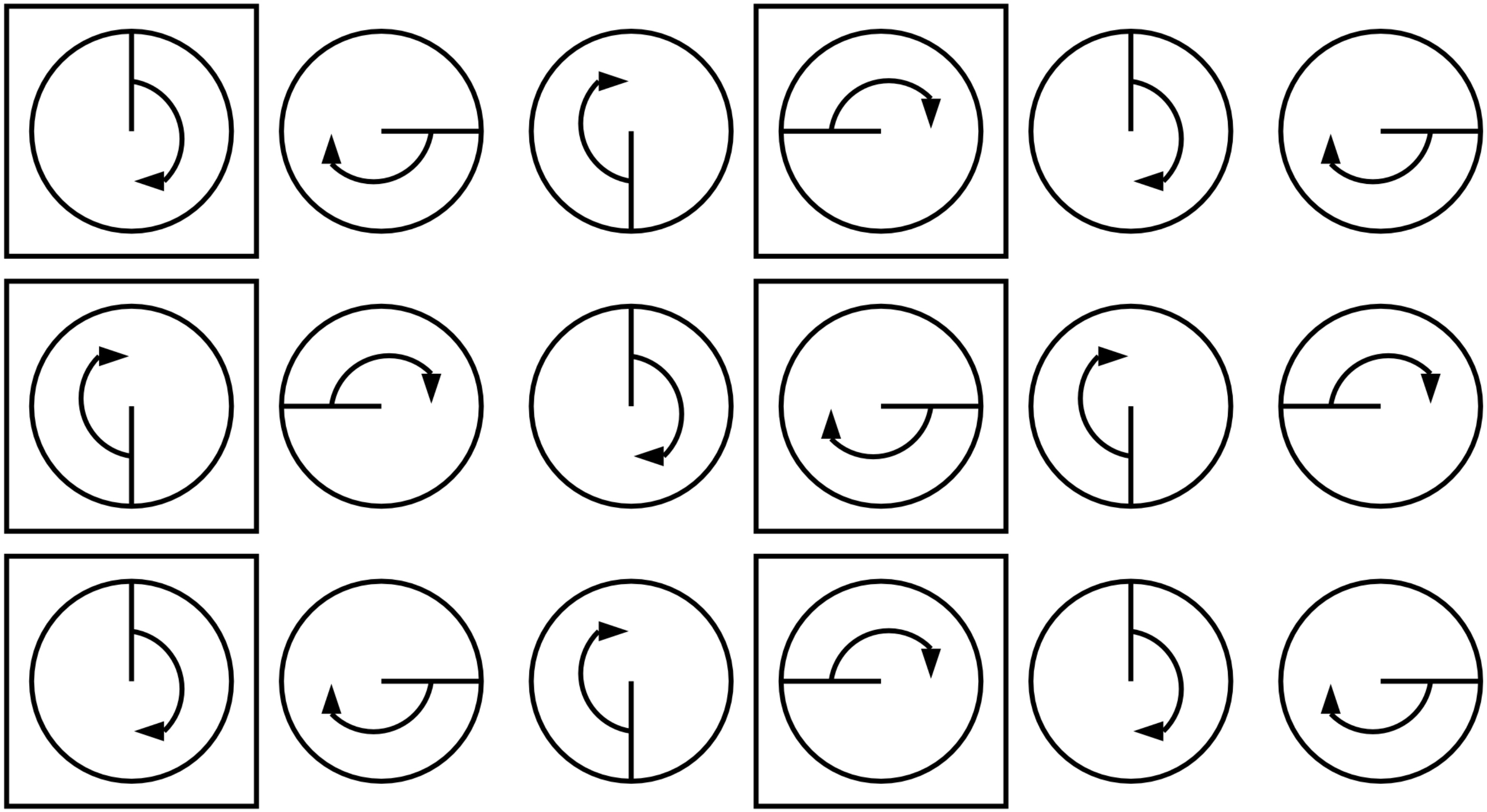
$$p_{ij} = p(0.10i, 0.10j)$$

Aliasing: Moiré effects

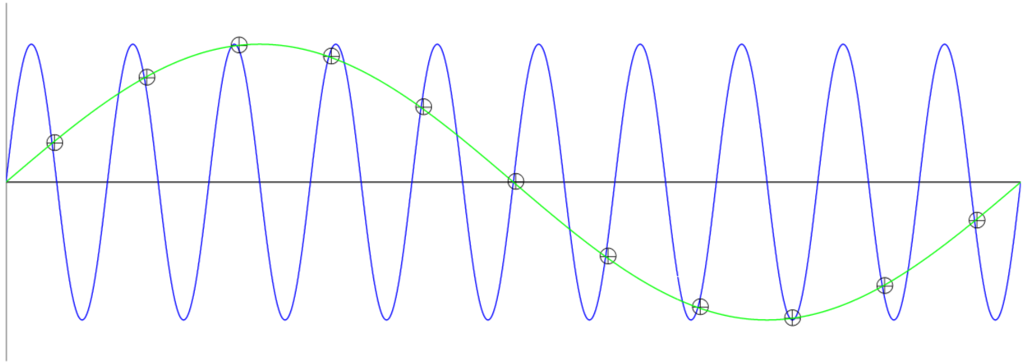


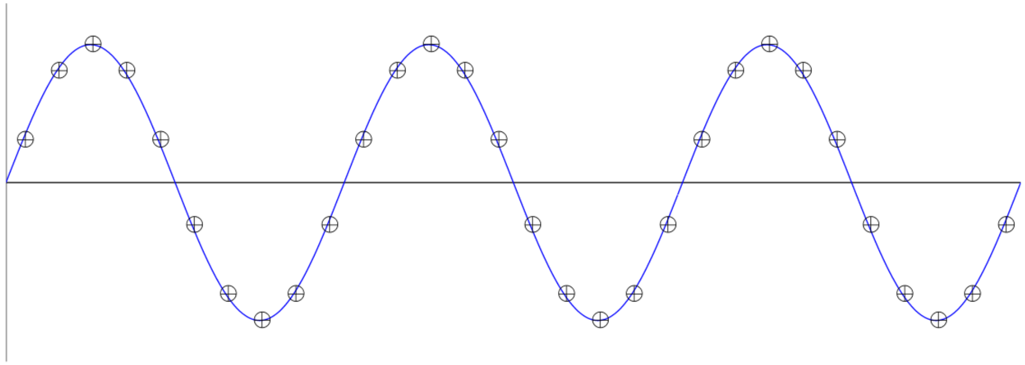
$$p_{ij} = p(0.20i, 0.20j)$$

Temporal aliasing

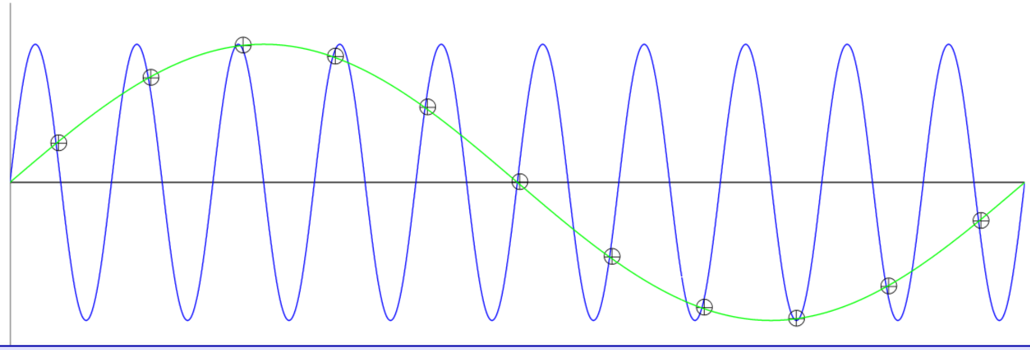


What's the problem?

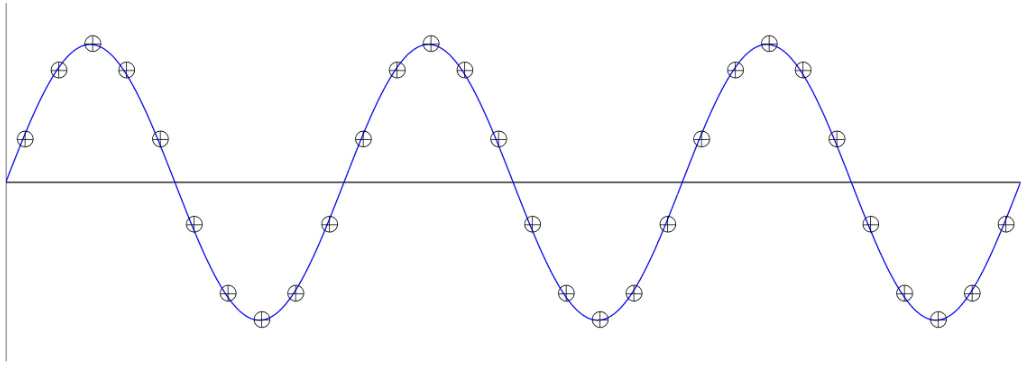




What's the problem?

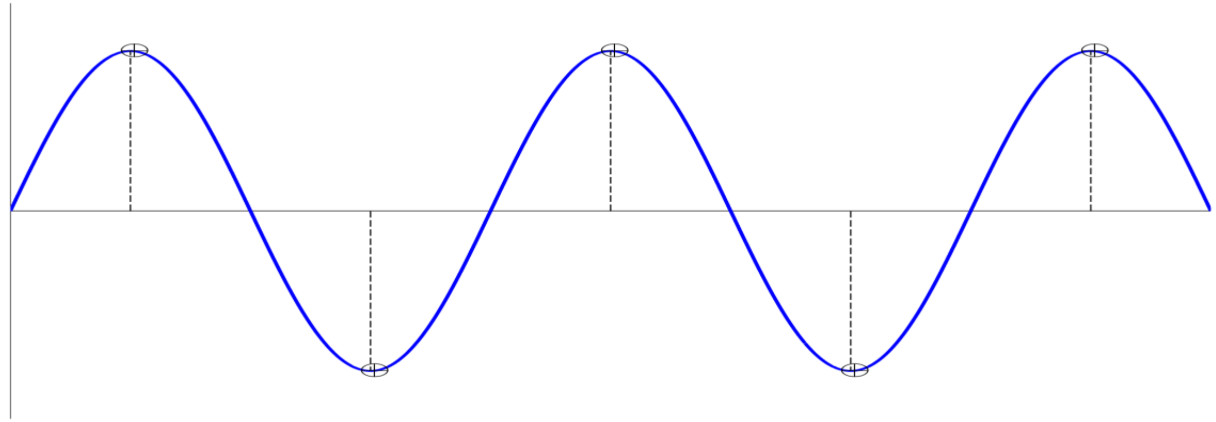


You cannot sample a fast-changing signal too sparsely!



The Nyquist theorem

- It turns out that in order to ensure correct discretization, *the discretization frequency must be at least twice the highest signal frequency*.



Correct discretization

- Simple rule: pick discretization frequency at least twice as high as the highest frequency in the image.
- Sometimes it is impossible.
  - We do not want to store huge pixel arrays
  - We do not know the actual frequency spectrum
- In this case we need to eliminate high frequencies before discretization

Frequency domain

- For proper understanding we must introduce the notion of a *frequency domain*.

$$f(t) = \int_{-\infty}^{\infty} F(w) e^{i2\pi wt} df$$

$$F(w) = \int_{-\infty}^{\infty} f(t) e^{-i2\pi wt} df$$

Frequency domain

As we know we can represent an n-dimensional vector in an arbitrary *basis*.

$$\boldsymbol{v} = \sum_i v_i^B \boldsymbol{b}_i$$

where (assuming B is orthonormal) v_i^B can be found as the *projection* of $\boldsymbol{v}$ on the corresponding basis vector:

$$v_i^B = \langle \boldsymbol{v}, \boldsymbol{b}_i \rangle = \boldsymbol{v}^T \boldsymbol{b}_i$$

Frequency domain

The vector's components,

$$\boldsymbol{v} = (v_1, v_2, \dots, v_n)$$

are simply its coordinates with respect to the *canonical basis*:

$$(1, 0, 0, \dots, 0),$$
$$(0, 1, 0, \dots, 0),$$
$$(0, 0, 1, \dots, 0),$$
$$\dots$$

Frequency domain

Representation in other bases can be informative.

Frequency domain

Consider, for example, the basis, consisting of discrete cosine functions with different frequencies:

$$\boldsymbol{b}_0 = (1, 1, 1, 1, \dots, 1),$$

$$\boldsymbol{b}_1 = \left(\cos\left(\frac{0.5}{n}\pi\right), \cos\left(\frac{1.5}{n}\pi\right), \dots, \cos\left(\frac{n-0.5}{n}\pi\right)\right),$$

$$\boldsymbol{b}_2 = \left(\cos\left(2\frac{0.5}{n}\pi\right), \cos\left(2\frac{1.5}{n}\pi\right), \dots, \cos\left(2\frac{n-0.5}{n}\pi\right)\right),$$

$$\dots$$

$$\boldsymbol{b}_k = \left(\cos\left(k\frac{0.5}{n}\pi\right), \cos\left(k\frac{1.5}{n}\pi\right), \dots, \cos\left(k\frac{n-0.5}{n}\pi\right)\right),$$

Frequency domain

- Representation of a vector in this basis tells us for every frequency “how much” of it is present in the vector.

- E.g. many real-life pictures, when represented in this basis, will have small coefficients for high-frequency components.

Frequency domain

- Representation of a vector in this basis tells us for every frequency “how much” of it is present in the vector.

- E.g. many real-life pictures, when represented in this basis, will have small coefficients for high-frequency components.
  - This is the core idea behind JPEG compression.

Space vs Frequency domain

$$\boldsymbol{v} = (v_1, \ldots, v_n)$$

$$\hat{\boldsymbol{v}} = (a_1, \ldots, a_n)$$

$$\boldsymbol{v} = \sum_k a_k \mathbf{cos}(k\,\cdot)$$

$$a_k = \langle \boldsymbol{v}, \mathbf{cos}(k\,\cdot) \rangle$$

Space vs Frequency domain

The same idea applies to functions.

$$f(x) \qquad \hat{f}(w)$$

$$\boldsymbol{f} = \int \hat{f}(w)\boldsymbol{b} \qquad \hat{f}(w) = \langle f, \boldsymbol{b} \rangle$$

Space vs Frequency domain

The same idea applies to functions.

$$f(x) \qquad \hat{f}(w)$$

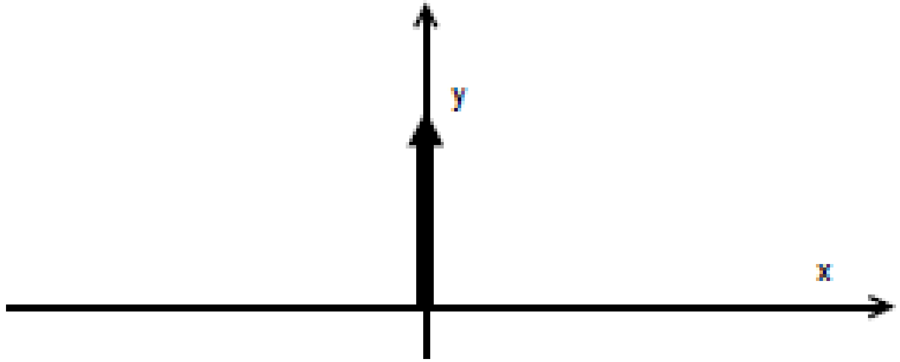
$$f(x) = \int \hat{f}(w) b_w(x) dw$$

$$\hat{f}(w) = \int f(x) b_w(x) dx$$

Dirac's delta function

- The "Dirac's delta function" corresponds to an infinitely short unit impulse:

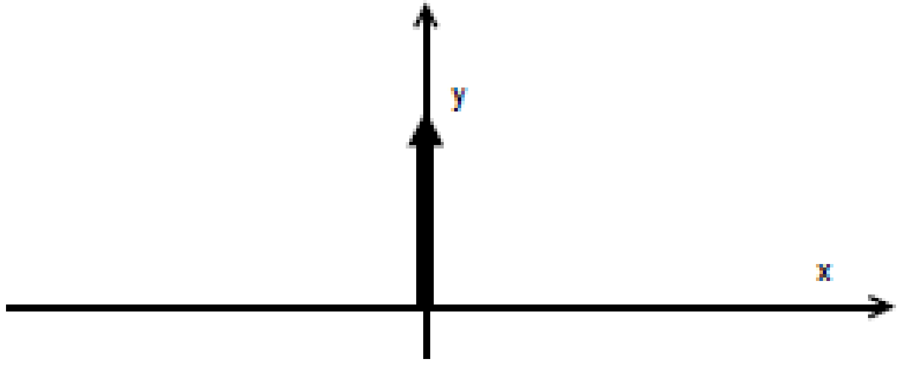
$$\delta(x) = \begin{cases} \infty, & \text{if } x = 0 \\ 0, & \text{otherwise} \end{cases}$$



Dirac’s delta function

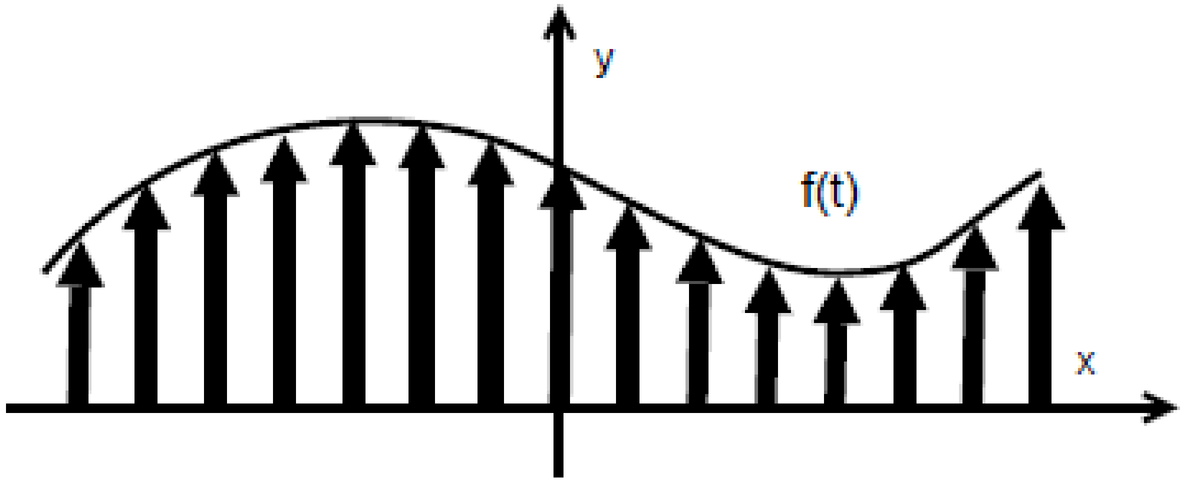
- The “Dirac’s delta function” corresponds to an infinitely short *unit* impulse :

$$\int_{-\infty}^{\infty} \delta(x)\,\mathrm{d}x = 1$$



Canonical basis for functions

- Every function is its own representation in the basis of Dirac delta functions:



Space vs Frequency domain

The most important frequency-domain basis for functions is the **complex Fourier basis**:

$$b_w(x) = e^{i2\pi \cdot wx}$$
$$= \cos(2\pi wx) + i \sin(2\pi wx)$$

Transformation to and from this basis is called the *Fourier* and *inverse Fourier* transform.

Example: $\cos(2\pi A x)$

Space domain

$$\cos(2\pi A x)$$

Frequency domain

$$\frac{1}{2} e^{i2\pi \cdot Ax} + \frac{1}{2} e^{i2\pi \cdot (-A)x}$$

$$\hat{f}(w) = \frac{1}{2}\delta(w - A) + \frac{1}{2}\delta(w + A)$$

Example: $\mathbf{box}_{a}(x)$

Space domain

$$\mathrm{box}_{\mathrm{a}}(x) = 1,$$
$$\text{when } x \in [-a, a],$$
$$0 \text{ otherwise}$$

Frequency domain

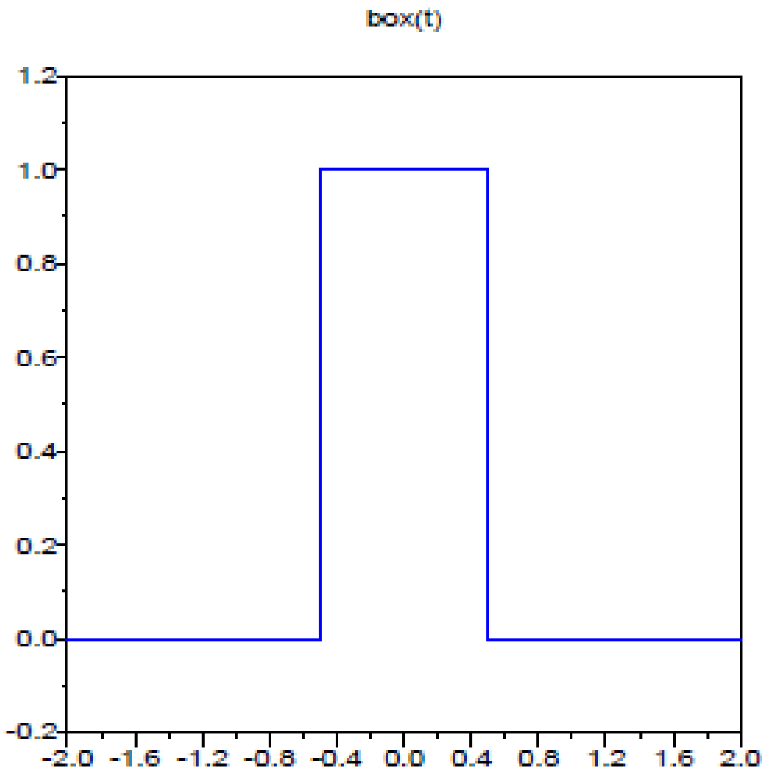
$$\widehat{\mathrm{box}}_{a}(w) = \frac{\sin(2\pi a w)}{\pi w}$$

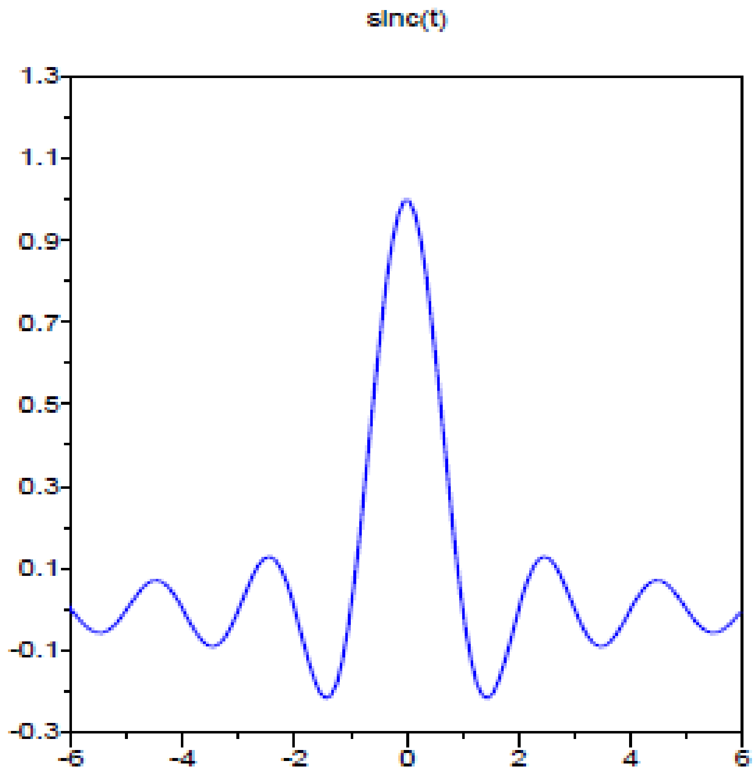
$$= 2a\,\mathrm{sinc}(2aw)$$

Example: $\mathbf{box}_a(x)$

Space domain

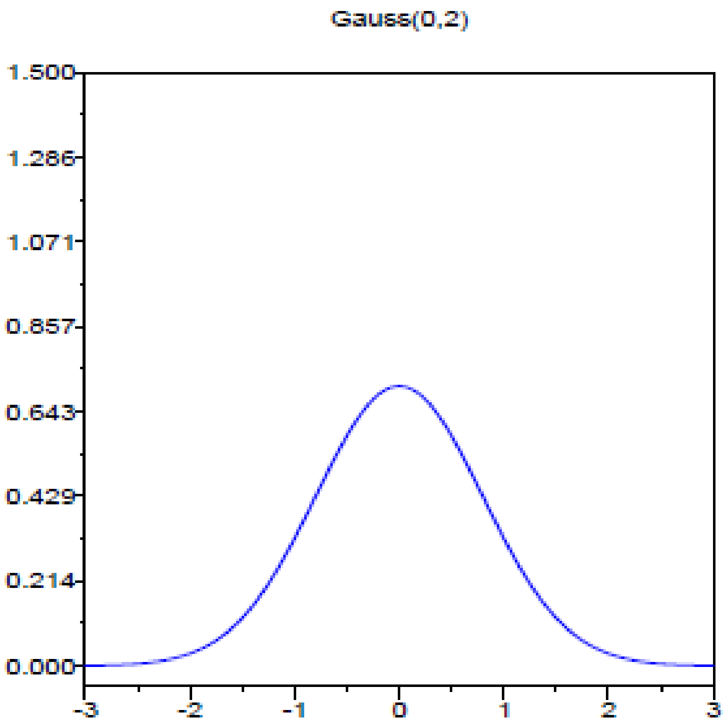
Frequency domain



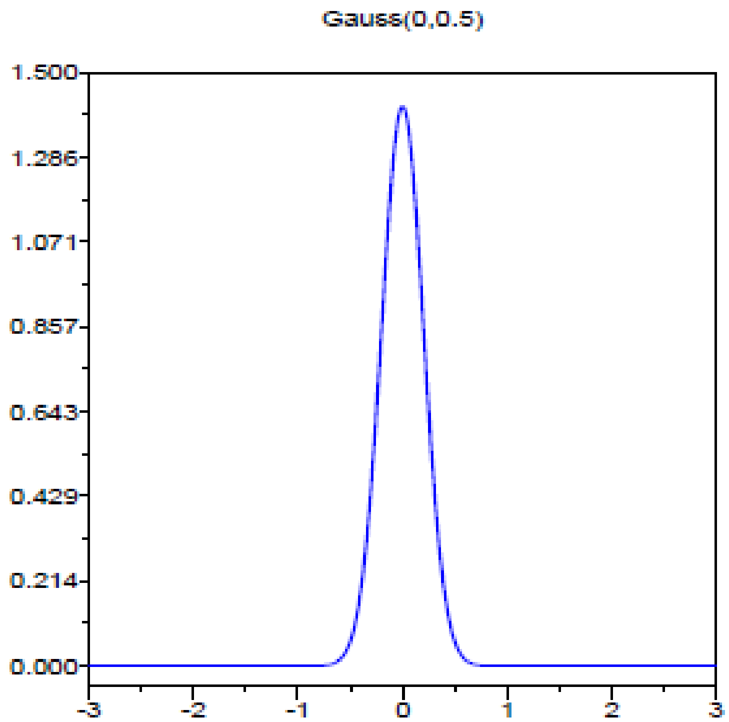


Example: Gaussian

Space domain



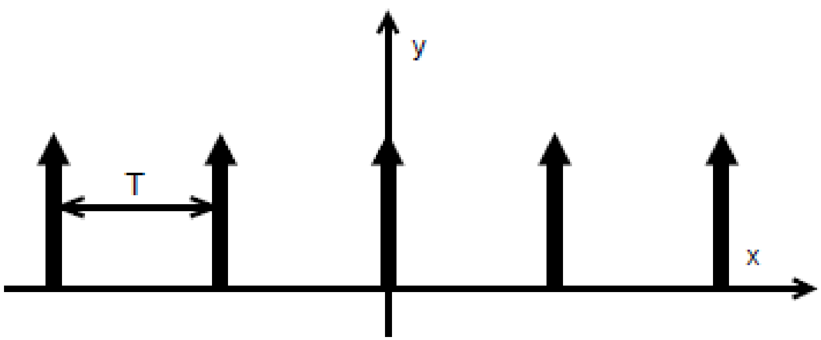
Frequency domain

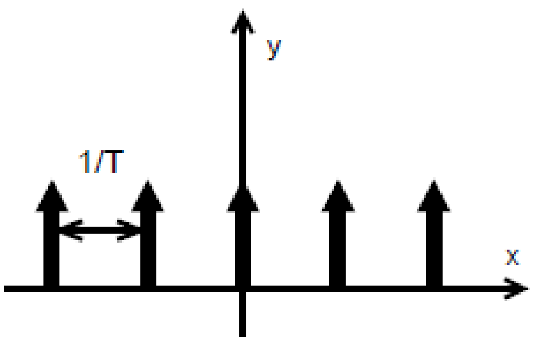


Example: Delta-comb

Space domain | Frequency domain

$$\delta_T^*(t) \leftrightarrow \delta_{1/T}^*(w)$$





Convolution

An important property of the Fourier transform:

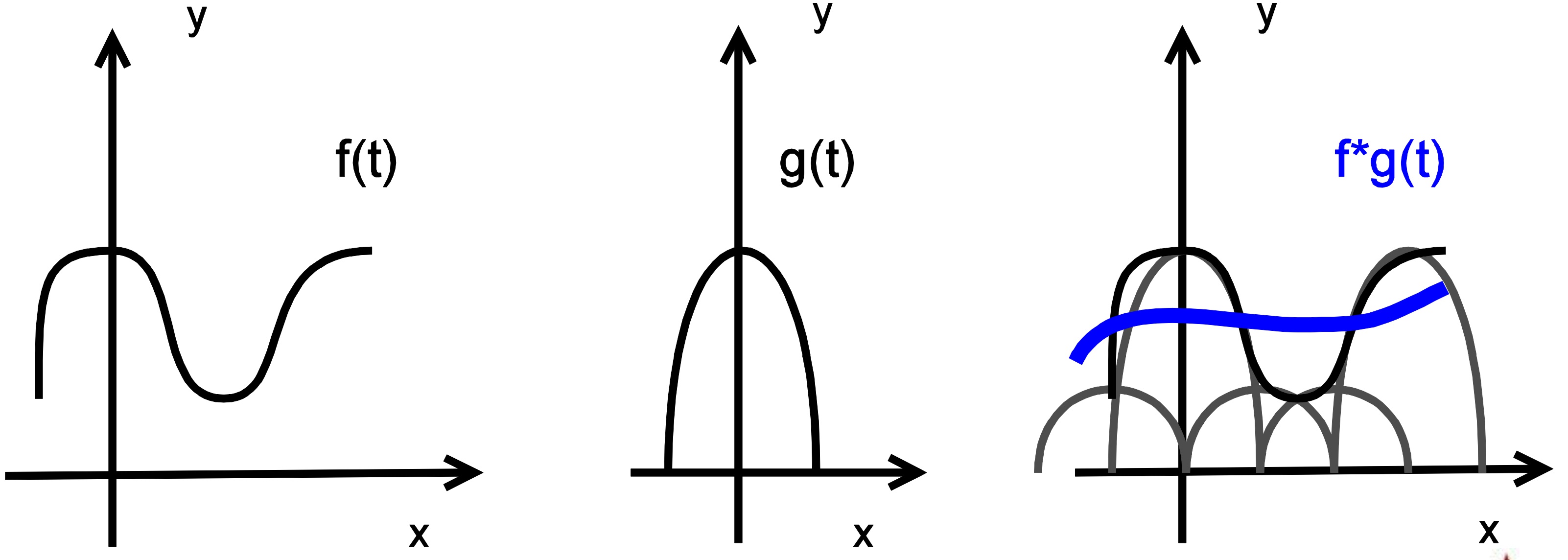
$$f(t)g(t) \leftrightarrow \hat{f}(w) * \hat{g}(w)$$

$$f(t) * g(t) \leftrightarrow \hat{f}(w)\hat{g}(w)$$

where * denotes *convolution*.

Convolution

$$(f * g)(t) = \int_{-\infty}^{\infty} f(x)g(t-x)dx = \int_{-\infty}^{\infty} g(x)f(t-x)dx$$

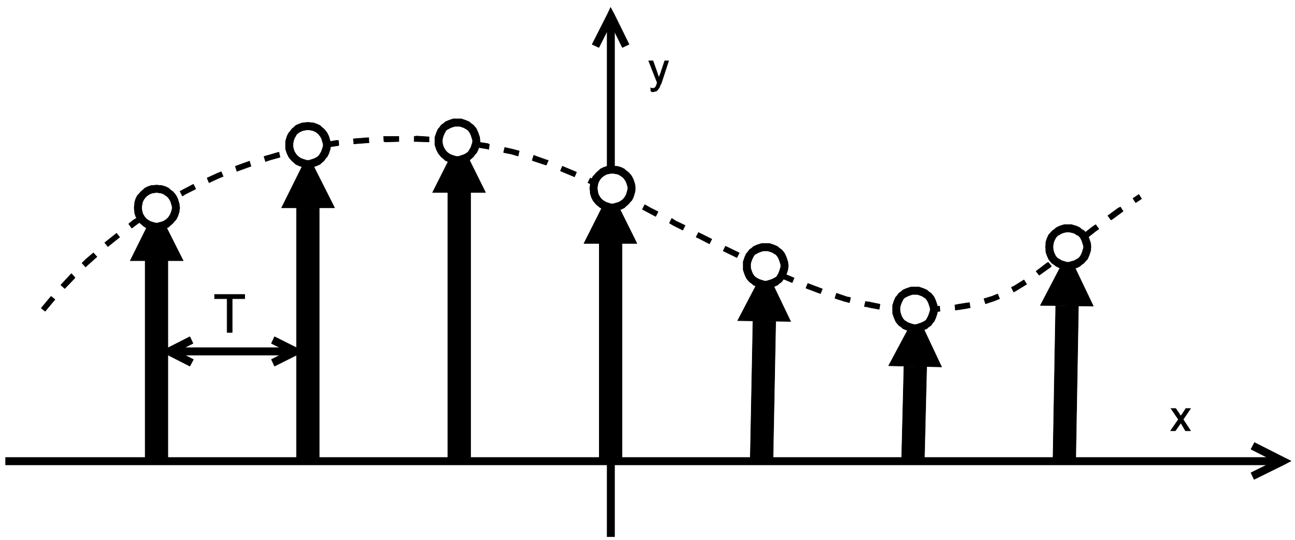


Convolution with a 5x5 box filter

Convolution with a (-1,0,1) filter

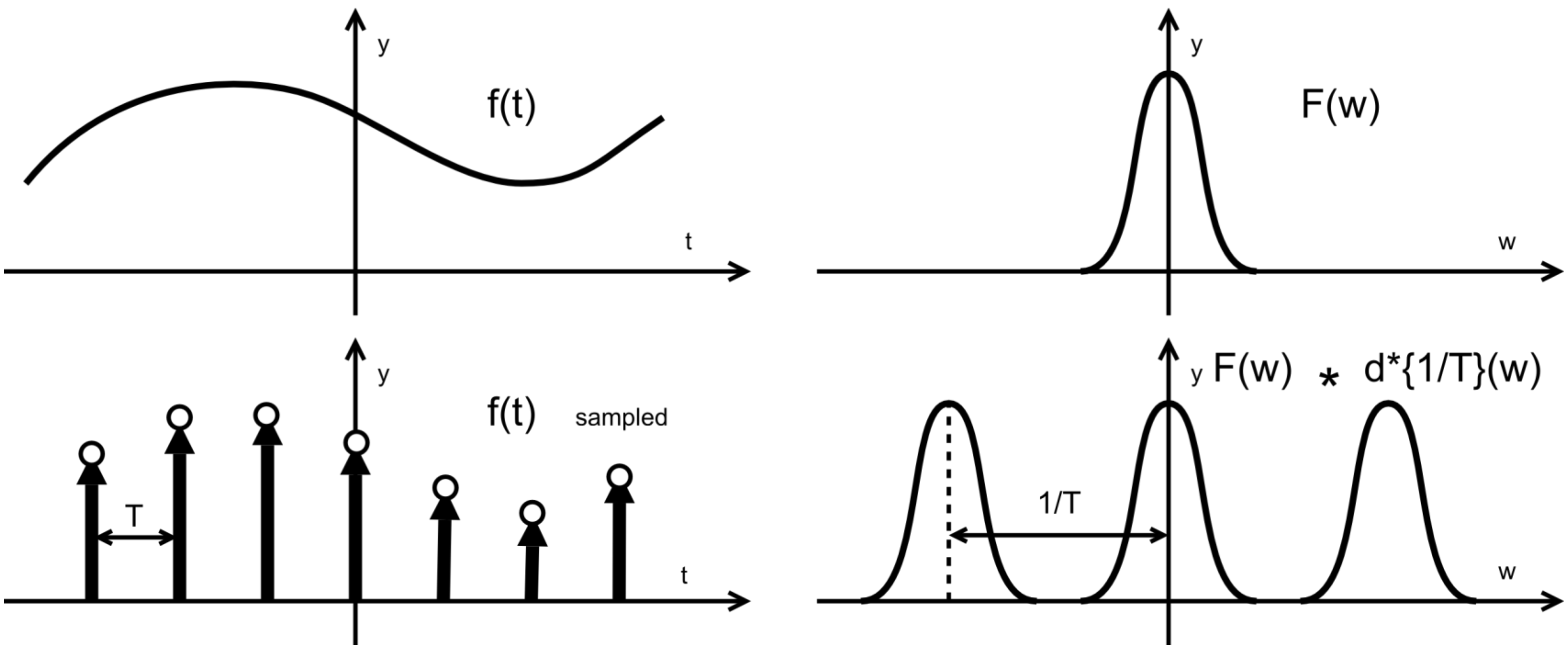
Back to sampling

- We shall represent sampling as a *multiplication with the Dirac's comb.*



$$f_{\text{sampled}}(x) = f(x)\delta_T^*(x)$$

Sampling & Frequency domain

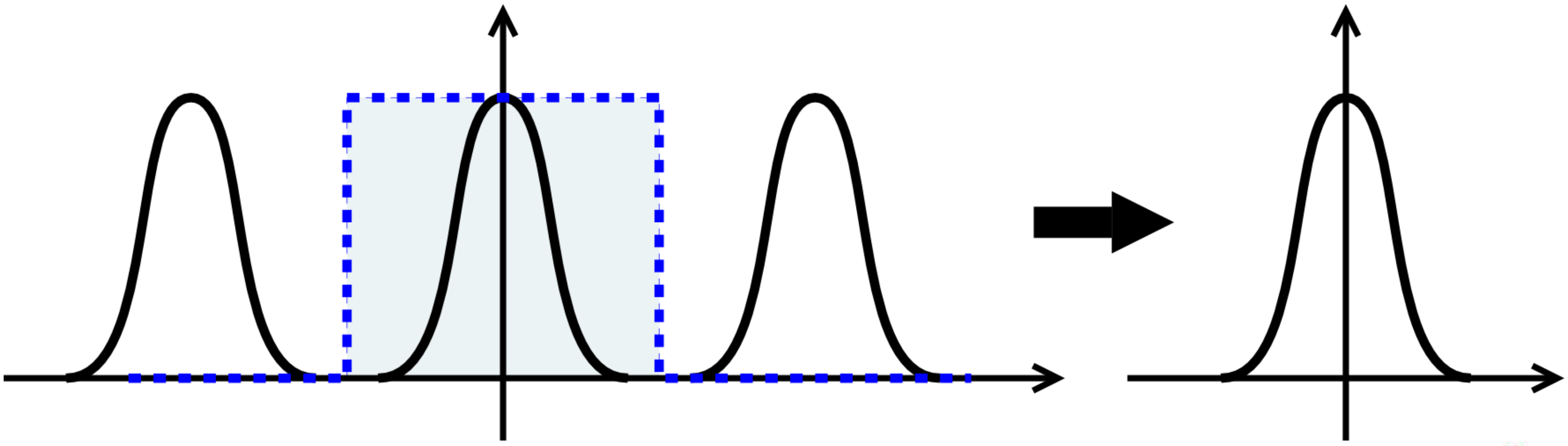


$$f(t) \leftrightarrow F(w)$$

$$f(t)\delta_T^*(t) \leftrightarrow F(w) * \delta_{1/T}^*(w)$$

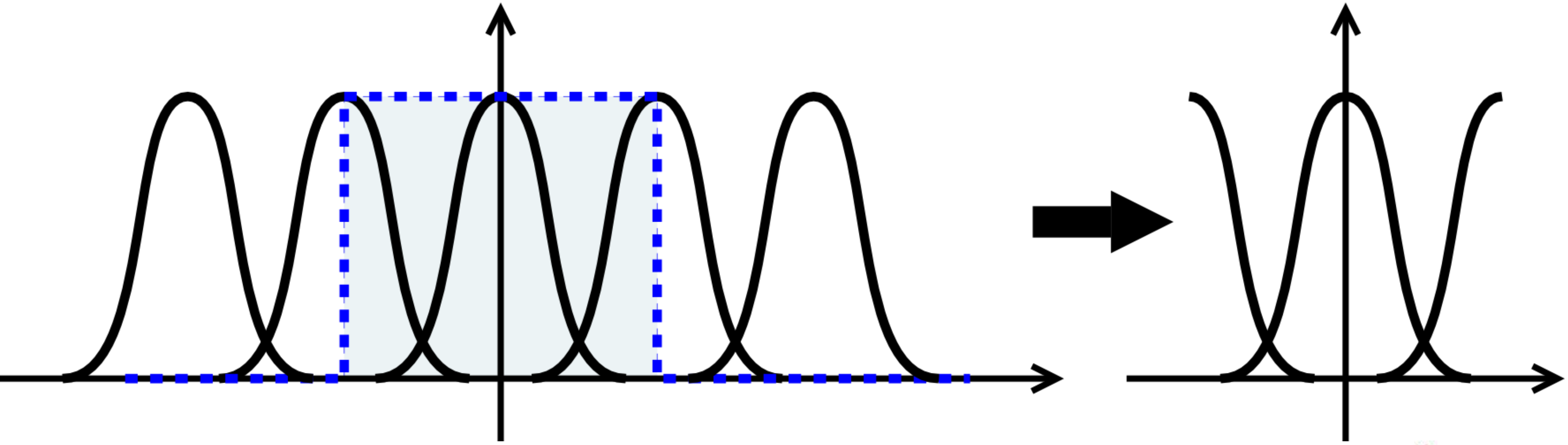
Sampling & Frequency domain

- If the whole spectrum of f fits into the period 1/T, we can restore the spectrum of the original signal by multiplying with the box function.



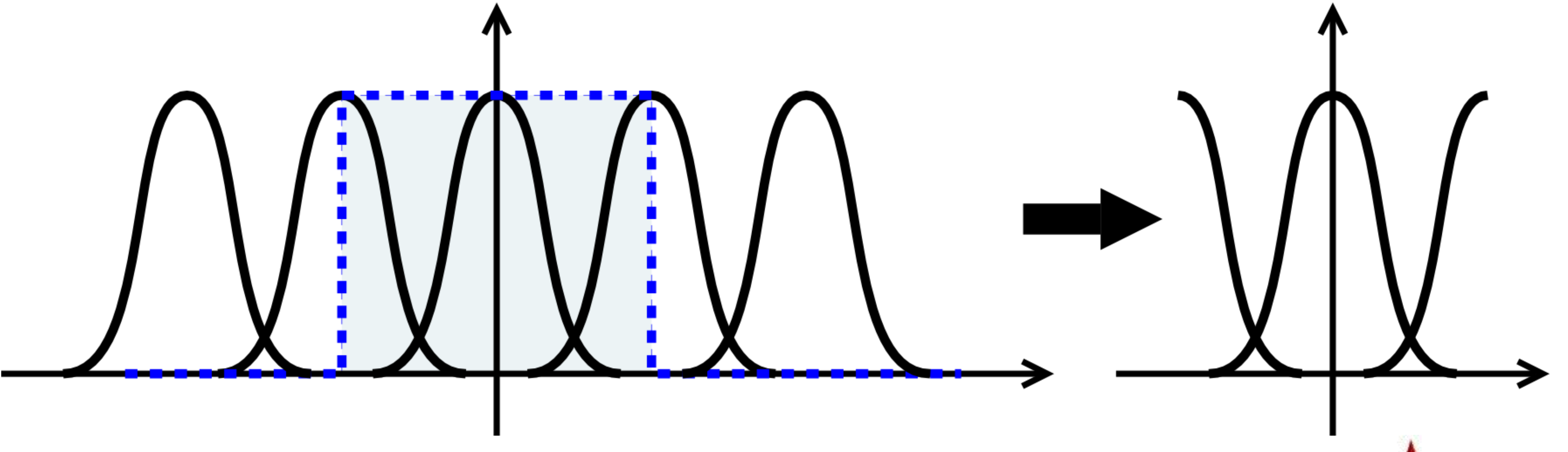
Sampling & Frequency domain

- If 1/T is too small, it is impossible to recover the original spectrum:



Sampling & Frequency domain

- The higher frequencies will get into the space of lower frequencies and vice-versa. Hence the name: *aliasing*.

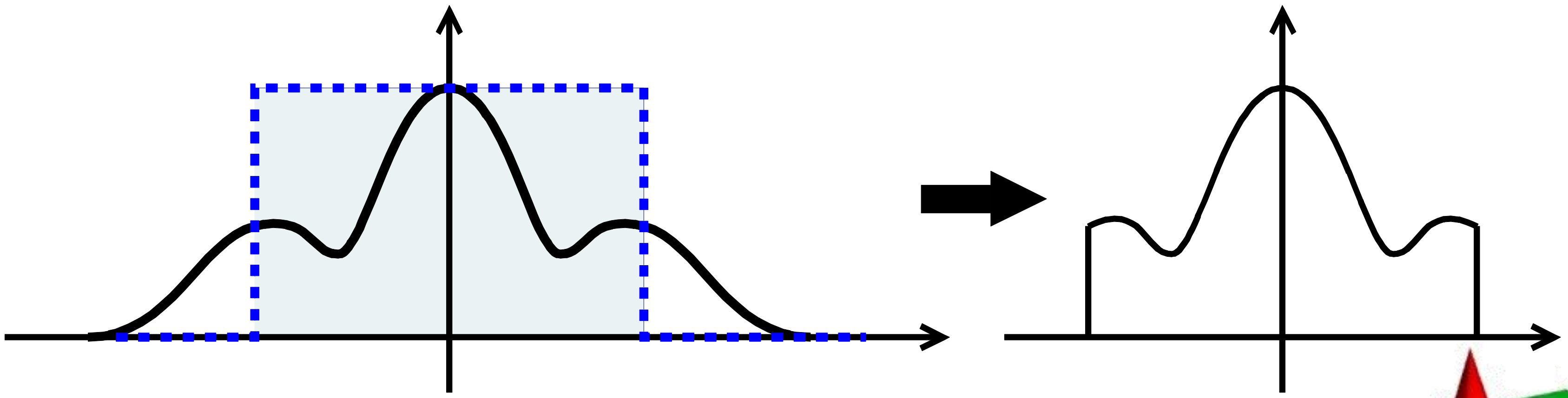


Nyquist theorem

- So in order to be able to perfectly reconstruct $f(x)$ from a sampled version, the spectrum $\hat{f}(w)$ must "fit" into a single period of length 1/T.
- Consequently, the sampling frequency (1/T) must be at least twice the size of the largest frequency in the signal.

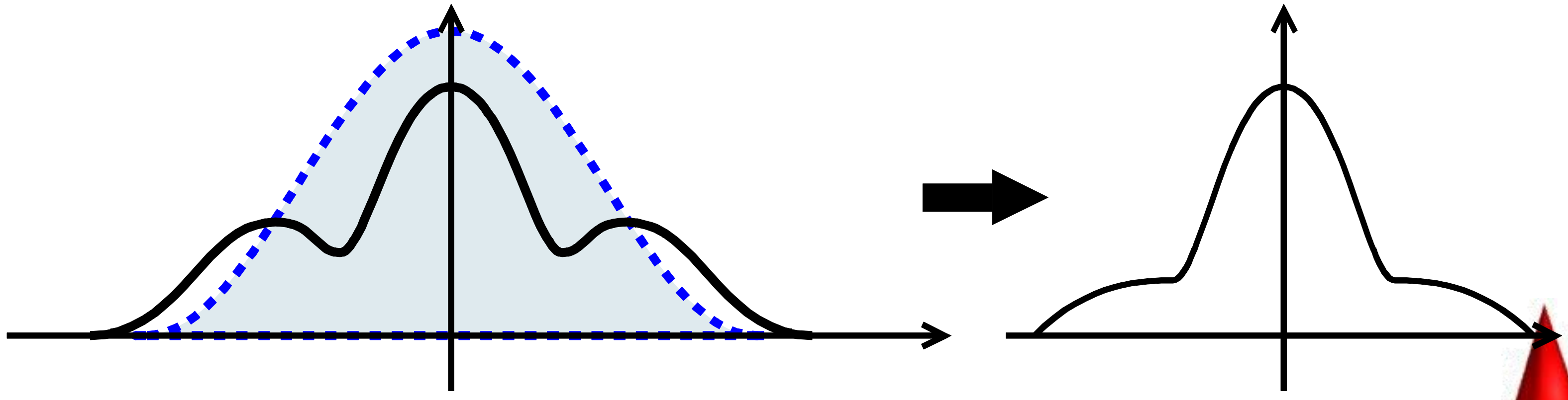
Correct sampling

- If we cannot sample at high enough frequency, we need to *band-limit* the signal, i.e. cut away the higher frequencies.
- Ideally, this means multiplying the spectrum with a box function:



Band limiting

- Spectrum multiplication with a box function means convolution with a sinc function, which is inefficient.
- Instead we can do band limiting by multiplying with a Gaussian.



Band limiting

- Multiplying the spectrum with a Gaussian corresponds to a convolution with a Gaussian mask (i.e. “blur”-filtering).

- An even cruder approximation is to simply average over square regions. This is what mipmapping achieves.

Anti-aliasing

Convolution with a Gaussian-like function is the core idea behind *anti-aliasing rasterization*.

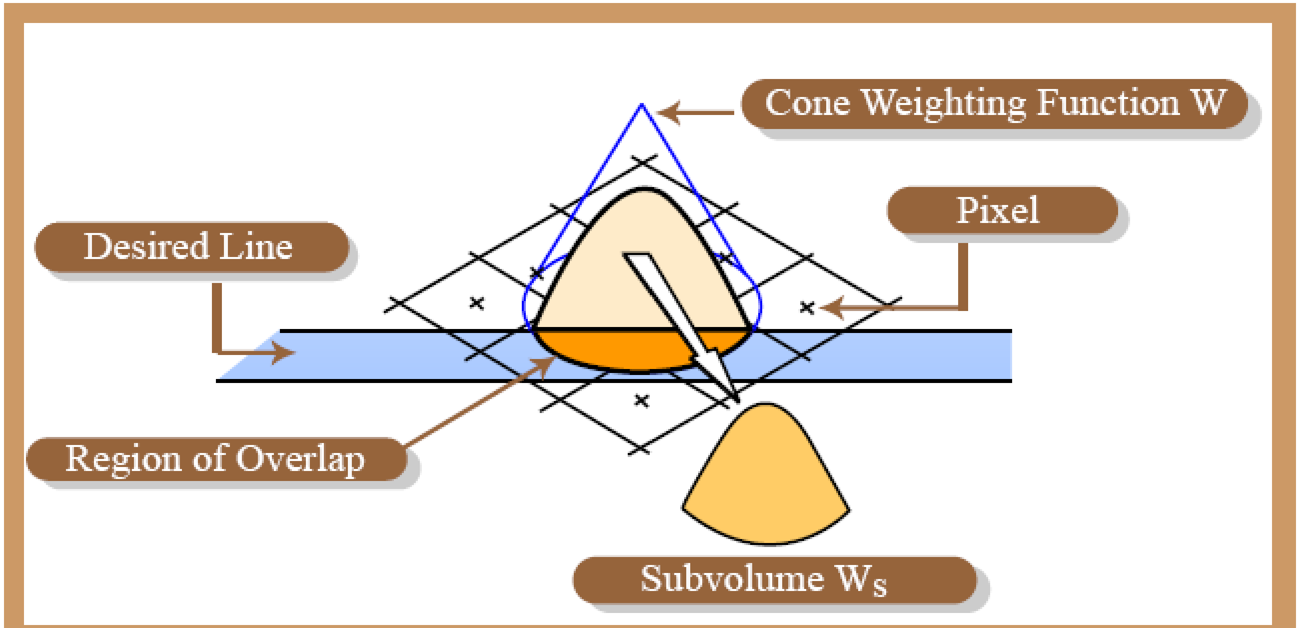


Image adapted from:
Foley et al. *Computer Graphics Principles and Practice*. 2nd Ed. Pearson Addison Wesley, June 1990.
ASIN: 0201121107

Anti-aliasing

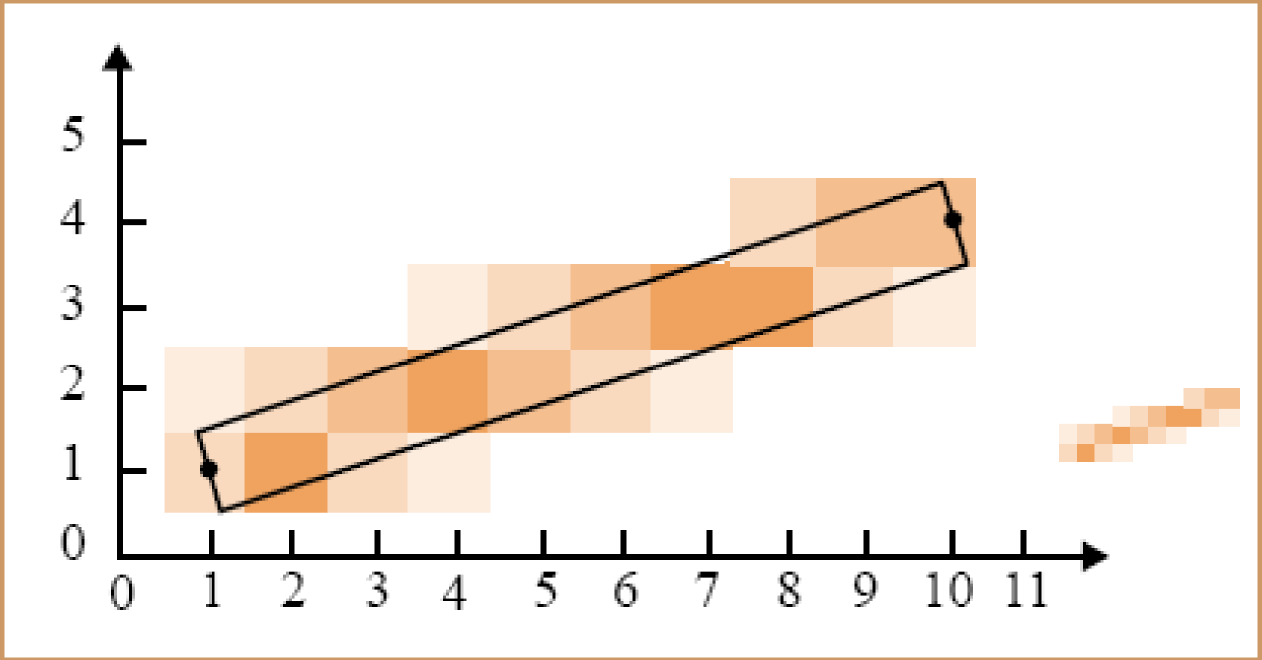


Image adapted from:
Foley et al. *Computer Graphics Principles and Practice*. 2nd Ed. Pearson Addison Wesley, June 1990. ASIN: 0201121107

Anti-aliasing

- One simple and practical way to convolve with a Gaussian while rendering is to add together several frames per pixel, each slightly shifted and weighted with a Gaussian.

- This can be done via the accumulation buffer or using *multisampling*.

Reconstruction

- Suppose we did our best to prepare the pixels and avoid aliasing.

- How do we reconstruct the actual image?

Nearest neighbor reconstruction

- The most “straightforward” reconstruction method is to assume that each pixel is a tiny square. However, this is not the correct thing to do.

Reconstruction

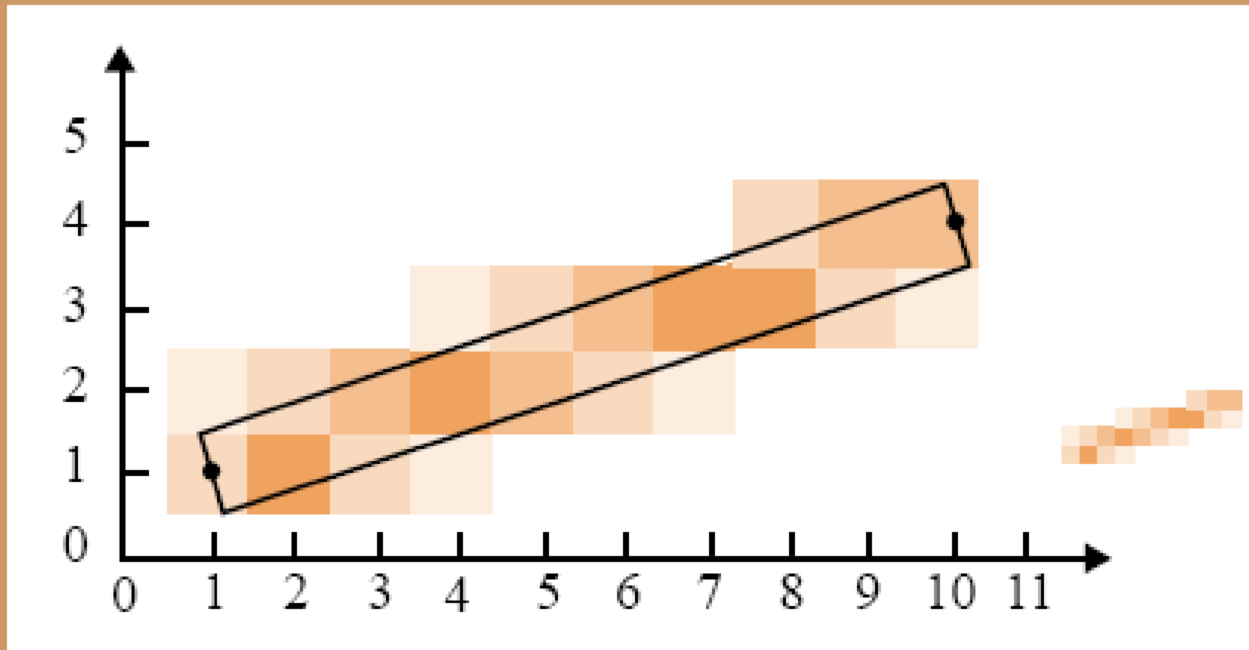


Image adapted from:
Foley et al. *Computer Graphics Principles and Practice*. 2nd Ed. Pearson Addison Wesley, June 1990. ASIN: 0201121107

Reconstruction

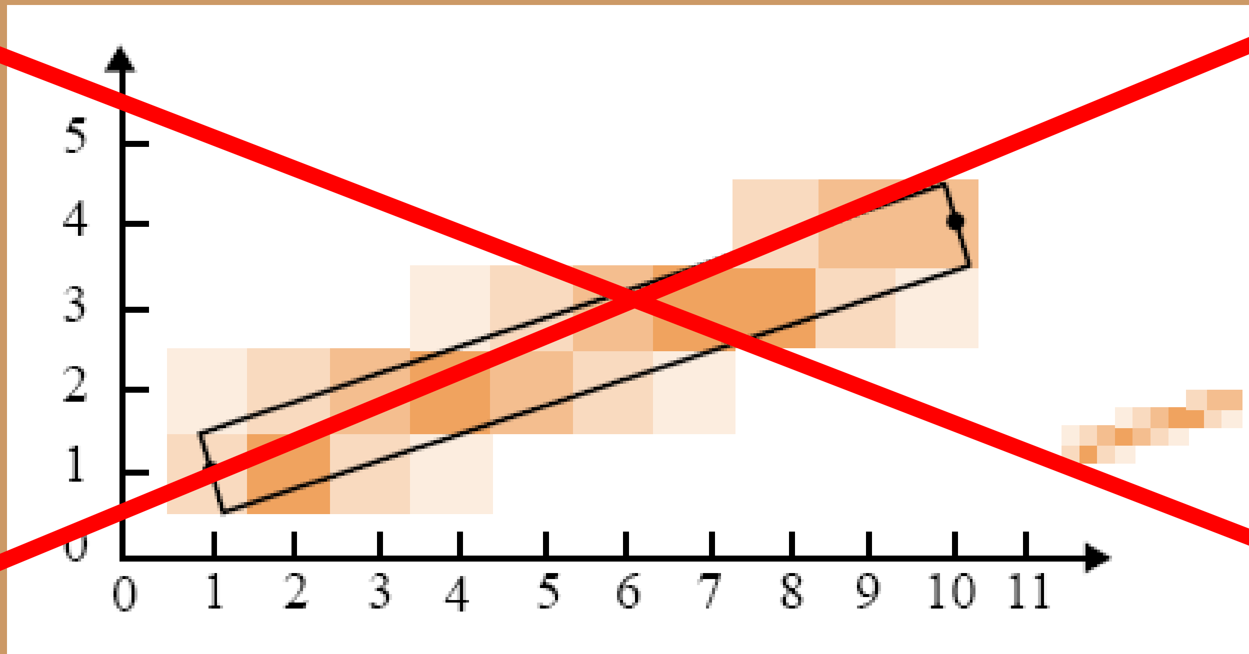
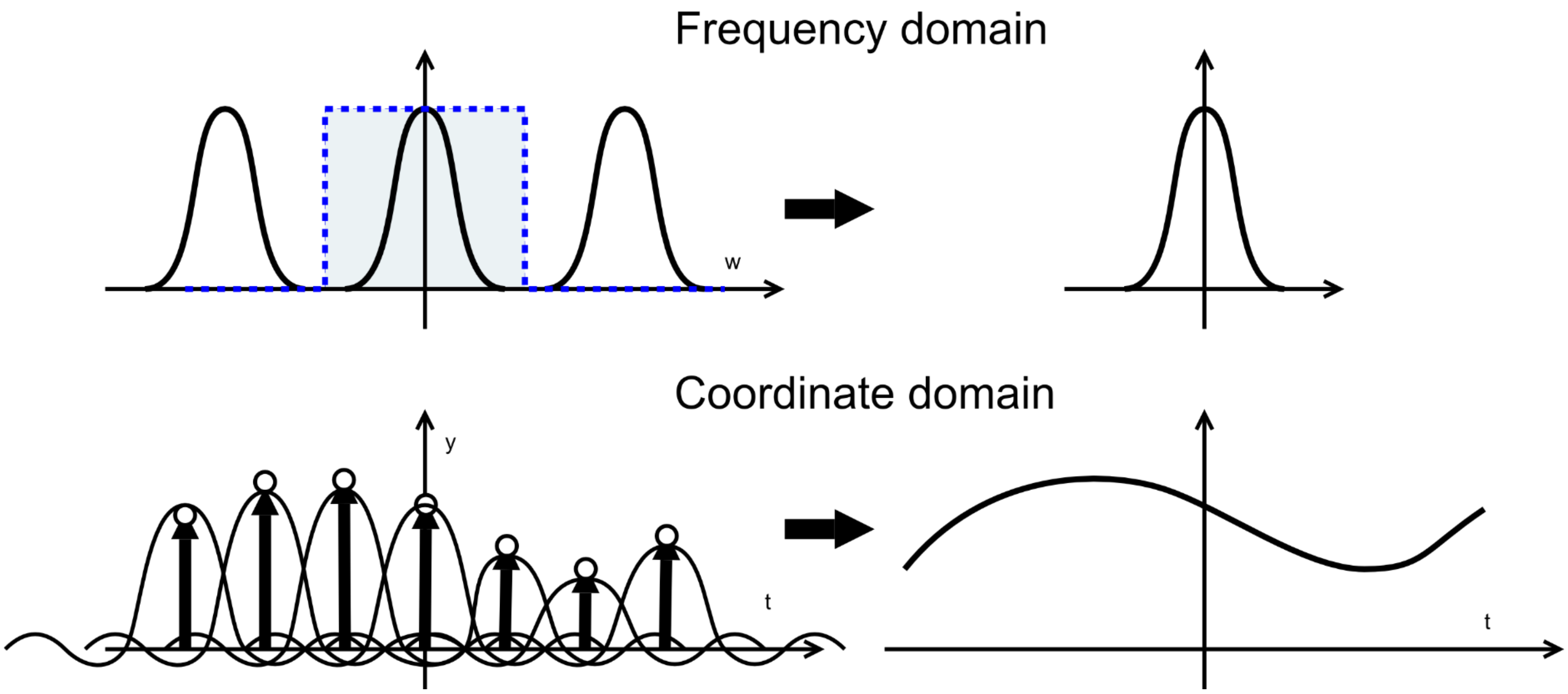


Image adapted from:
Foley et al. *Computer Graphics Principles and Practice*. 2nd Ed. Pearson Addison Wesley, June 1990. ASIN: 0201121107

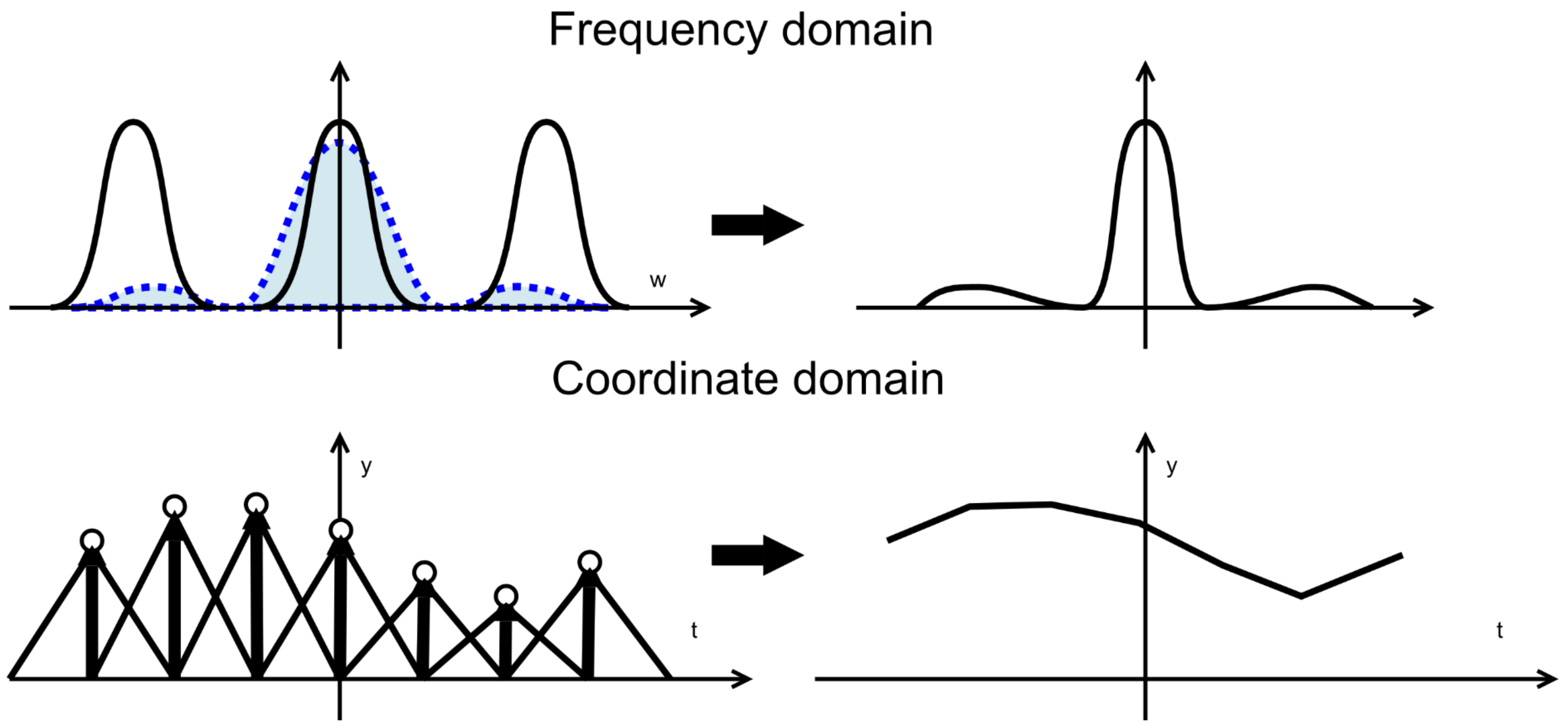
Perfect reconstruction



Perfect reconstruction

- To perfectly reconstruct the signal (or image $p(x, y)$ from its sampled form we need to take a convolution with the sinc function.
- This is often impractical, and we would convolve with a Gaussian or a linear function instead.

(Bi)linear filtering



Nearest vs Linear filter

Gauss filter

Gaussian filter is computationally more expensive but results in better quality than linear filter.

CRT monitors perform Gaussian reconstruction on their pixels:

Conclusion

- **Sampling**
  - Must be done with correct discretization frequency
  - Usually implies low-pass filtering (i.e. averaging)
- **Reconstruction**
  - Requires filtering (i.e. convolution)
  - Ideal filter – sinc function. In practice (bi)linear or Gaussian is often used instead.

Food for thought

- Reconstruction and sampling often come together during *resampling*.
  - Texturing
  - Picture operations
- Suppose you use an image editor to rotate a picture 45 degrees. Think about the operation in terms of a reconstruction + sampling step. What filters should be used to get a perfect result?

Food for thought

- How to address problems of temporal aliasing?