Filtering: Use of the Accumulation Buffer

The **accumulation buffer** has the same resolution as the frame buffer, but has greater depth resolution. We can use the additional resolution to render successive images into one location while retaining numerical accuracy.

In OpenGL, we can clear the accumulation buffer as we do any other buffer, and then can use the function `glAccum` either to add or to multiply values from the frame buffer into the accumulation buffer, or to copy the contents of the accumulation buffer back to the frame. For example, the code

```
glClear(GL_ACCUM_BUFFER_BIT);
for (i=0; i < num_mages; i++)
{
    glClear(GL_COLOR_BUFFER_BIT, GL_DEPTH_BUFFER_BIT)
    display_image(i);
    glAccum(GL_ACCUM, 1.0/(float) num_images);
}
glAccum(GL_RETURN, 1.0);
```

uses the user's function `display_images` to generate a sequence of images into the frame buffer. Each is added into or accumulated into the accumulation buffer, with a scale factor 1 over the number of images. At the end, the accumulated image is copied back to the frame buffer.

We can combine use of the accumulation buffer with pixel mapping to perform various digital-filtering operations. Suppose that we start with a distance image. We can represent the image with an $N \times N$ matrix

$$A = [a_{ij}],$$

of scalar levels. If we process each color component of a color image independently, we can regard the entries in **A** as either individual color components or gray (luminance) levels. A **linear filter** produces a filtered matrix **B** whose elements are

$$b_{ij} = \sum_{k=-m}^{m} \sum_{l=-n}^{n} a_{kl} h_{i-k,j-l}.$$

We say that **B** is the result of **convolving A** with a filter matrix $\mathbf{H}$. In general, the values of m and n are small, and we can represent $\mathbf{H}$ by a small **convolution matrix**.

For each pixel in **A**, we place the convolution matrix over a_{ij}, and take a weighted average of the surrounding points. The values in the matrix are weights. For example, for $n = m = 1$, we can average each pixel with its four surrounding neighbors using the 3×3 matrix

$$\mathbf{H} = \frac{1}{5}\begin{bmatrix} 0 & 1 & 0 \\ 1 & 1 & 1 \\ 0 & 1 & 0 \end{bmatrix}.$$

This filter can be used for antialiasing. We can use more points and can weight the center more heavily with

$$\mathbf{H} = \frac{1}{16}\begin{bmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{bmatrix}.$$

Note that we must define a border around **A** if we want **B** to have the same dimensions. Other operations are possible with small matrices. For example, we can use the matrix

$$\mathbf{H} = \begin{bmatrix} 0 & -1 & 0 \\ -1 & -4 & -1 \\ 0 & -1 & 0 \end{bmatrix},$$

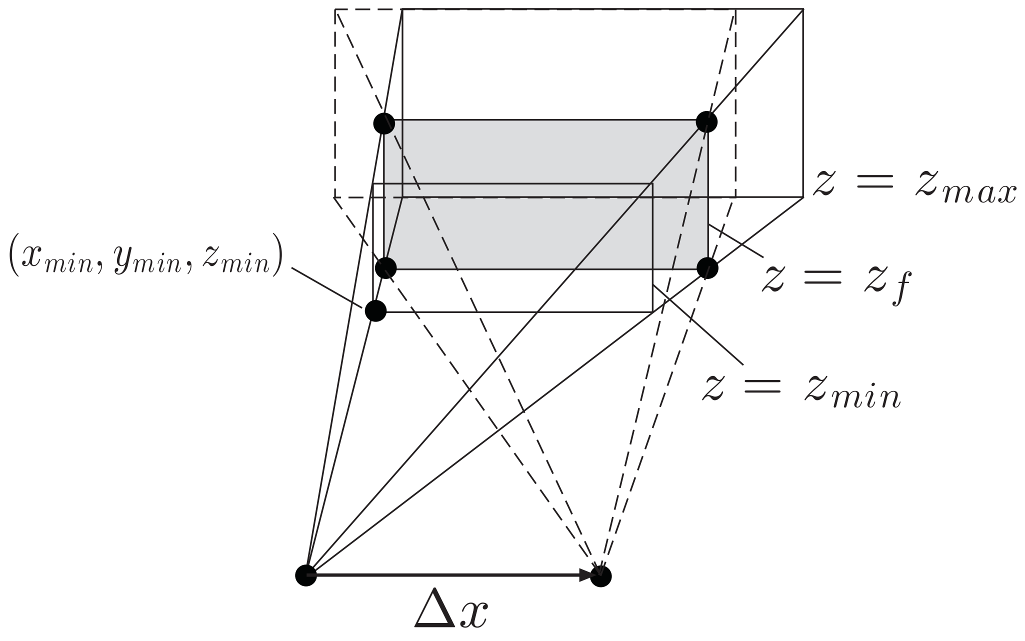
to detect changes in value or edges in the image. If a matrix is $k \times k$, we can implement a filter by accumulating k^2 images in the accumulation buffer, each time adding in a shifted version of **A** using a different filter coefficient in `glAccum`.

We can also use the accumulation buffer for filtering in time and depth. For example, if we jitter an object and render it multiple times, leaving the positions of the other objects unchanged, we get dimmer copies of the jittered object in the final image. If the object is moved along a path, rather than randomly jittered, we see the trail of the object. This **motion-blur** effect is similar to the result of taking a photograph of a moving object using a long exposure time. We can adjust the constant in `glAccum` so as to render the final position of the object with greater opacity, or to create the impression of speed differences.

We can also use filtering in depth to create focusing effects. A red camera cannot produce an image with all objects in focus. Objects within a certain distance from the camera, the camera's **depth of field**, are in focus; objects outside it are out of focus and appear blurred. Computer graphics produces images with an infinite depth of field because we do not have to worry about the limitations of real lenses. Occasionally, however, we want to create an image that looks as thought it were produced by a real camera, or to defocus part of a scene so as to emphasize the objects within a desired depth of field. Once more, we can use the accumulation buffer. This time, the trick is to move the viewer in a manner that leaves a plane fixed, as shown in the figure below. Suppose that we wish to keep the plane at $z = z_f$ in focus, and to leave the near $(z = z_{-\min})$ and far $(z = z_{\max})$ clipping distances unchanged. If we use `glFrustum`, we specify the near clipping rectangle $(x_{\min}, x_{\max}, y_{\min}, y_{\max})$. If we move the viewer from the origin in the x direction by Δx, we must change $x_{\min}$ to

$$x'_{\min} = x_{\min} + \frac{\Delta x}{z_f}(z_f - z_{\textsf{near}}).$$

Similar equations hold for $x_{\max}$, $y_{\min}$, and $y_{\max}$. As we increase Δx and Δy, we create a narrower depth of field.



Depth-of-field jitter

Aliasing and Antialiasing

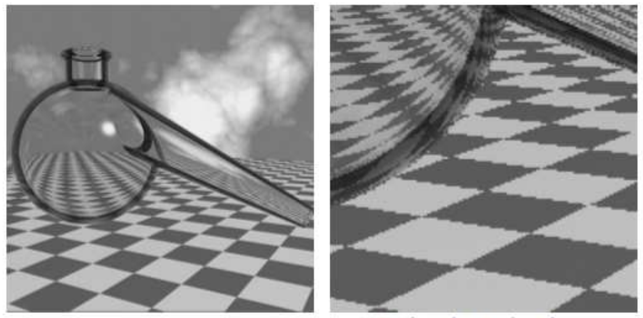
"If the scene contains frequencies greater than the Nyquist Sampling Frequency, then we have an aliasing problem"

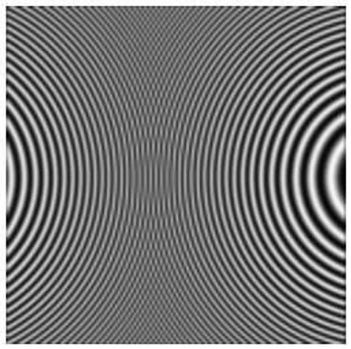
Results of aliasing:

- Jaggies
- Moire (See http://www.mathematik.com/Moire/)
- Flickering small objects
- Sparkling highlights
- Temporal strobing

Preventing aliasing or antialiasing:

1. Analytically prefilter the signal
2. Uniform supersampling and resample
3. Nonuniform or stochastic sampling


Staircasing or jaggies



Spectral Analysis / Fourier Transforms

Spectral representation treats the function as a weighted sum of sines and cosines

Each function has two representations

- Spatial (time) domain – normal representation
- Frequency domain – spectral representation

The *Fourier transform* converts between the spatial and frequency domain

$$\begin{tikzcd} \boxed{\begin{array}{c}\text{Spatial}\\ \text{Domain}\end{array}} \arrow[r, shift left=2, Rightarrow, "F(\omega) = \int_{-\infty}^{\infty} f(x)e^{-i\omega x}dx"] & \boxed{\begin{array}{c}\text{Frequency}\\ \text{Domain}\end{array}} \arrow[l, shift left=2, Rightarrow, "f(x) = \frac{1}{2\pi}\int_{-\infty}^{\infty} F(\omega)e^{i\omega x}d\omega"] \end{tikzcd}$$

Convolution

Definition

$$h(x) = f \otimes g = \int f(x')g(x - x')dx'$$

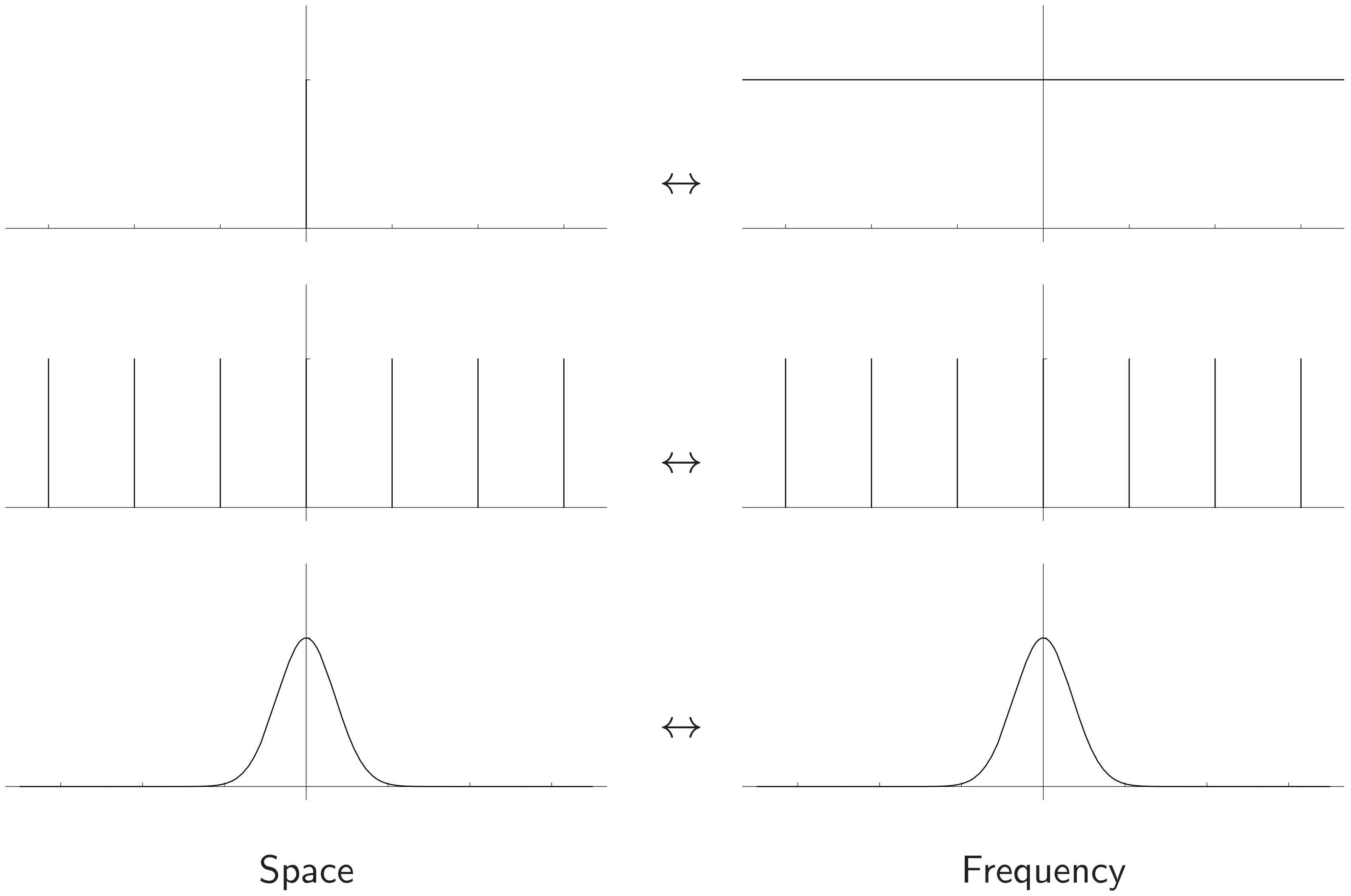
Convolution Theorem: Multiplication in the frequency domain is equivalent to convolution in the space domain

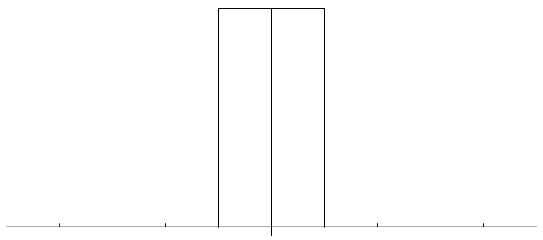
$$f \otimes g \longleftrightarrow F \times G$$

Symmetric Theorem: Multiplication in the space domain is equivalent to convolution in the frequency domain

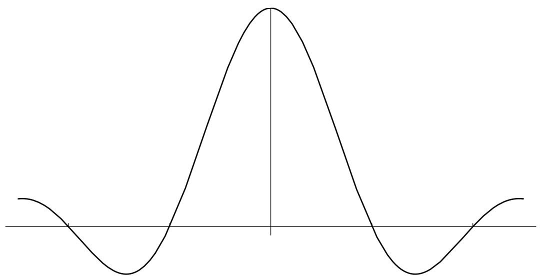
$$f \times g \longleftrightarrow F \otimes G$$

Fourier transform pairs



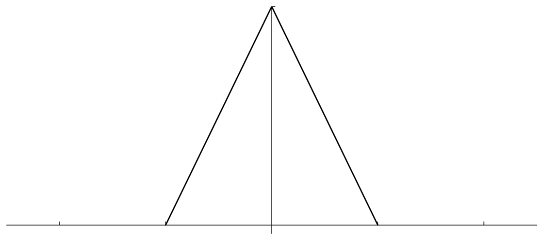


$\leftrightarrow$

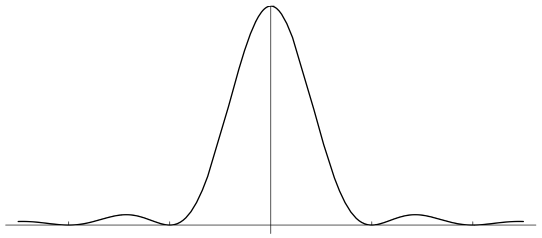


Square

Sinc

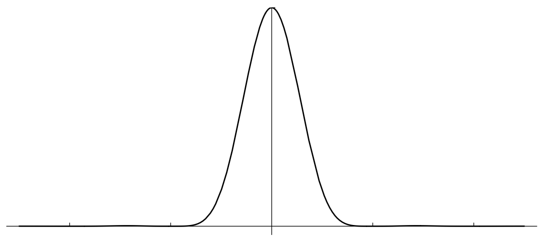


$\leftrightarrow$

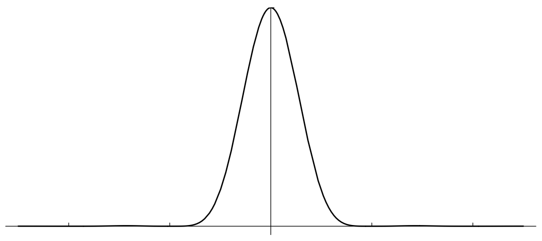


Square $\otimes$ Square

Sinc^2



$\leftrightarrow$



Cubic

Sinc^4

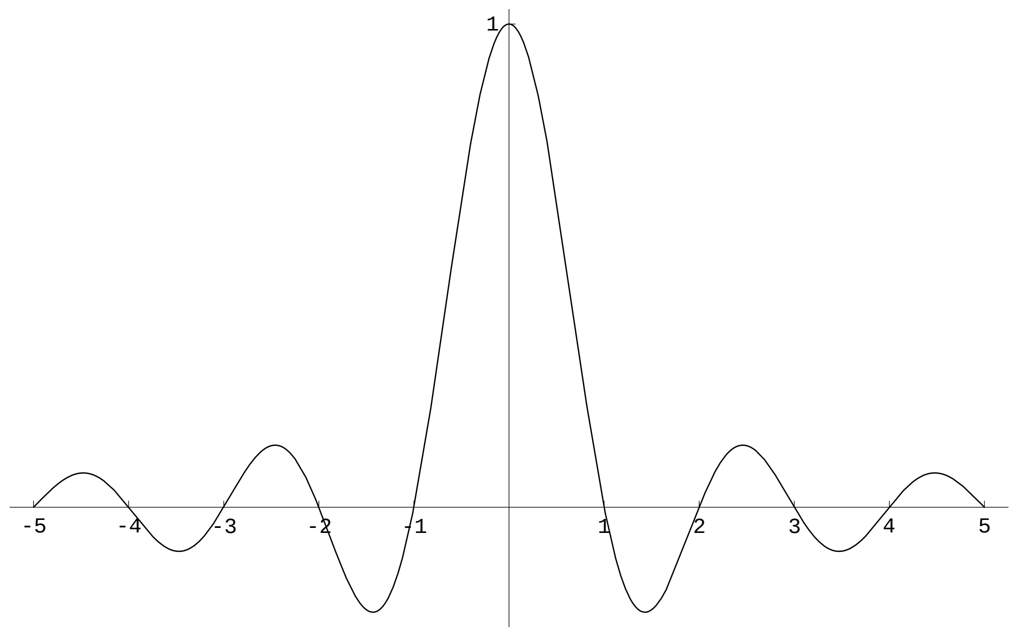
Space

Frequency

1-D Sinc Function

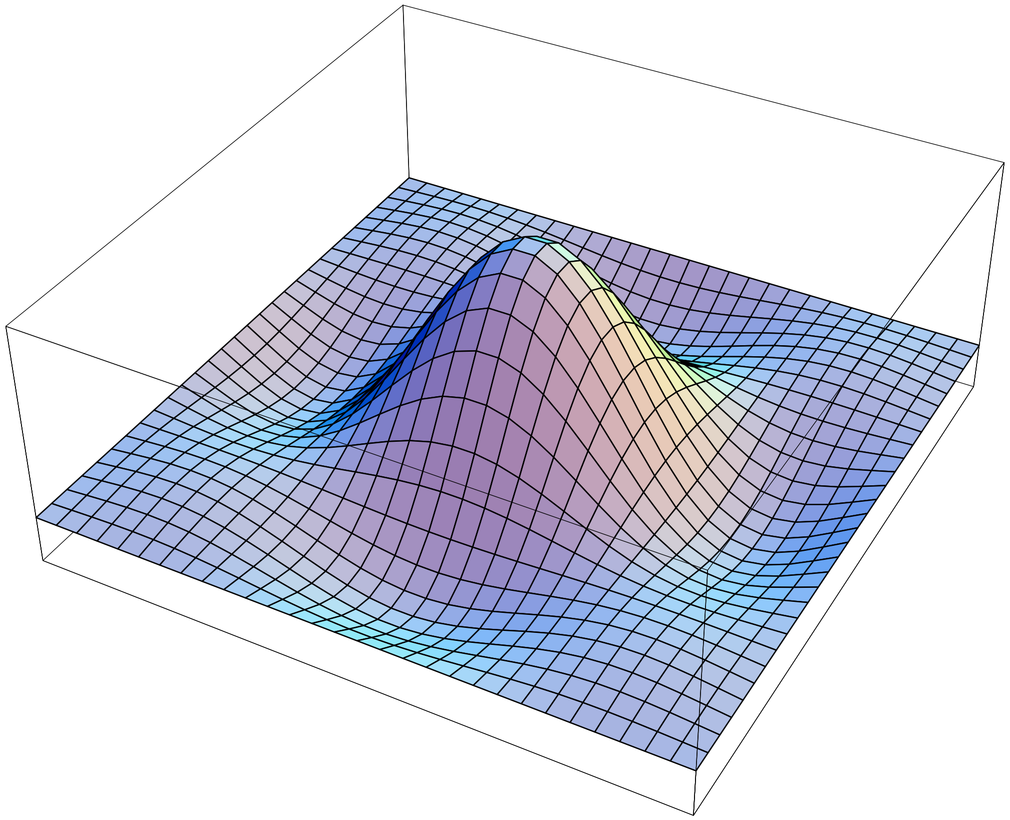
$$\mathsf{sinc}\, x = \frac{\sin \pi x}{\pi x}$$

$$\mathsf{sinc}\, 0 = 1$$



2-D Sinc Function

$$\mathsf{sinc}\,(x,y) = \mathsf{sinc}\,(x)\ \mathsf{sinc}\,(y)$$



Nyquist Sampling Theorem

> A signal can be reconstructed from its samples without loss of information, if the original signal has no frequencies above 1/2 the sampling frequency

For a given bandlimited function, the rate at which it must be sampled is called the Nyquist Frequency

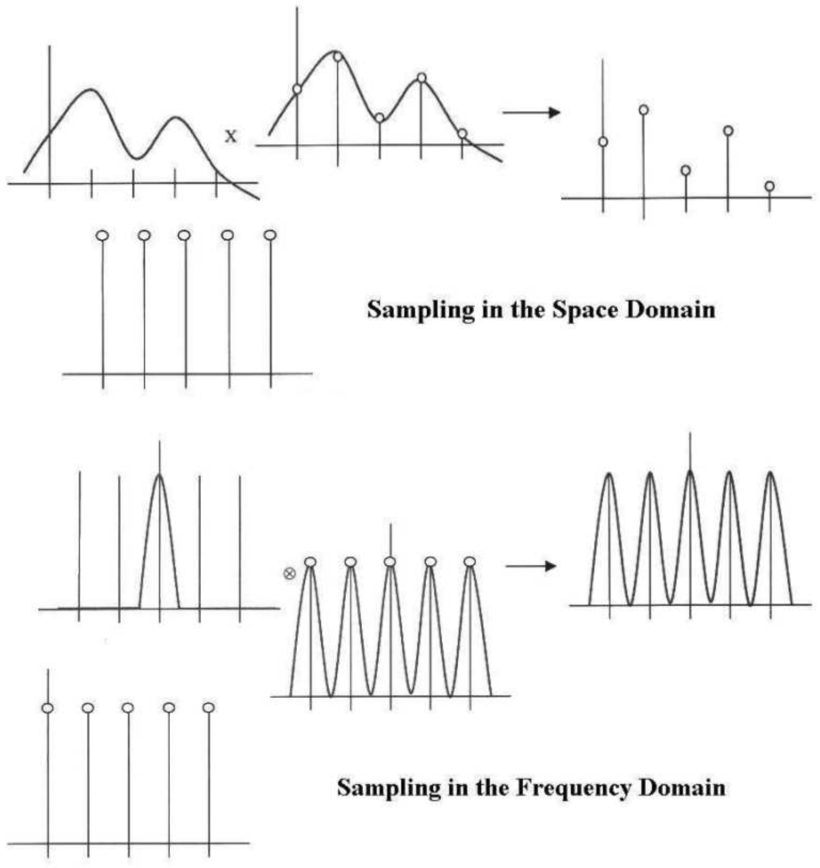
We can reconstruct a continuous function $f(x)$ from its samples $\{f_i\}$ by the formula

$$f(x, y) = \sum_{i=-\infty}^{\infty} f_i \,\mathsf{sinc}\,(x - x_i).$$

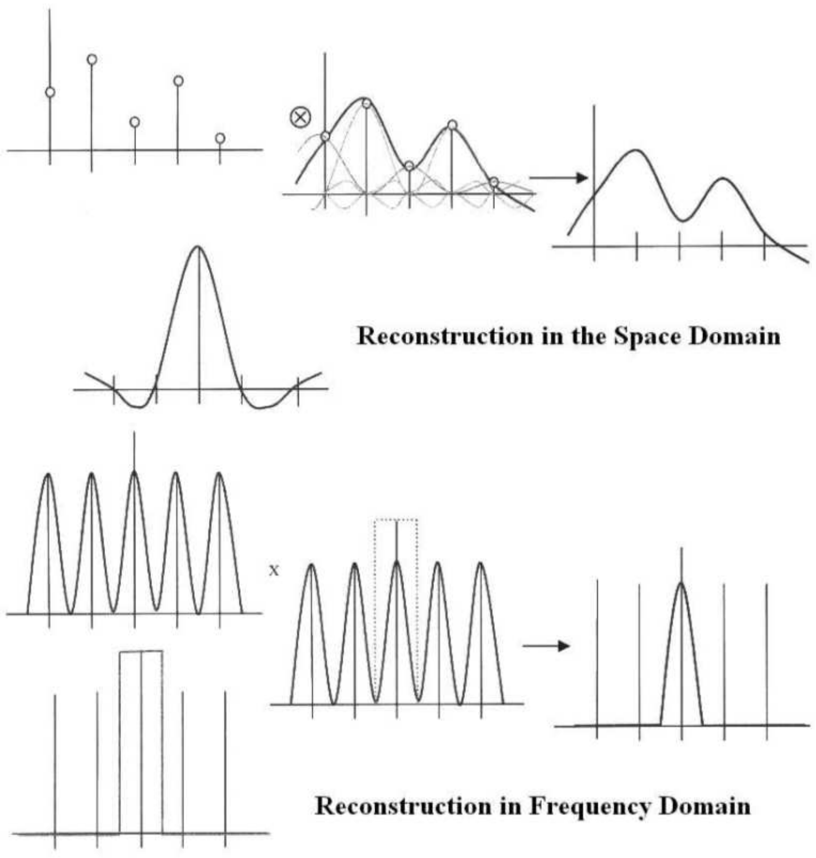
The two-dimensional version of the reconstruction formula for a function $f(x, y)$ with ideal samples $\{f_{ij}\}$ is

$$f(x, y) = \sum_{i=-\infty}^{\infty} \sum_{j=-\infty}^{\infty} f_{ij} \,\mathsf{sinc}\,(x - x_i)\,\mathsf{sinc}\,(y - y_i).$$

Sampling (Spatial, Frequency Domains)


x
Sampling in the Space Domain
⊗
Sampling in the Frequency Domain

Reconstruction (Spatial, Frequency Domains)

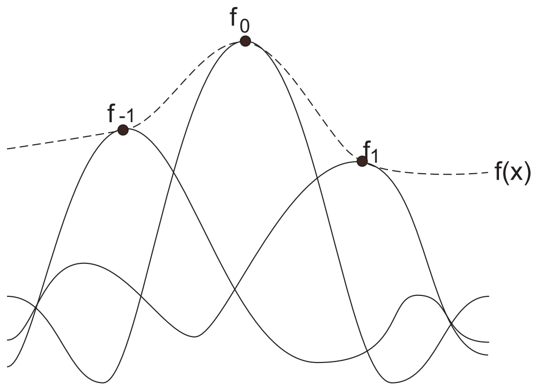

⊗
Reconstruction in the Space Domain
x
Reconstruction in Frequency Domain

Ideal Reconstruction

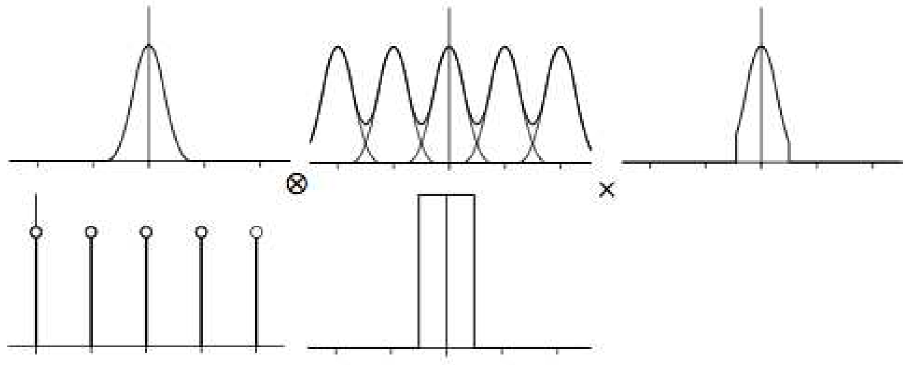
Ideally, use a perfect low-pass filter – the sinc function – to bandlimit the sampled signal and thus remove all copies of the spectra introduced by sampling

Unfortunately,

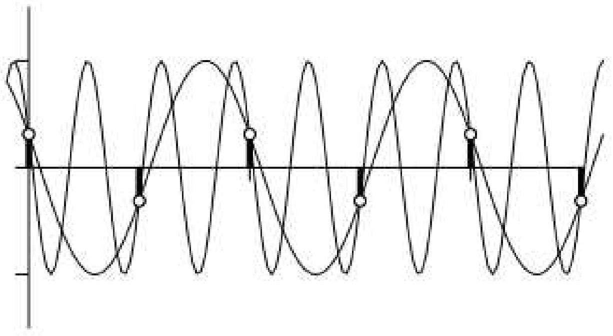
- The sinc has infinite extent and we must use simpler filters with finite extents. Physical processes in particular do not reconstruct with sincs
- The sinc may introduce ringing which are perceptually objectionable



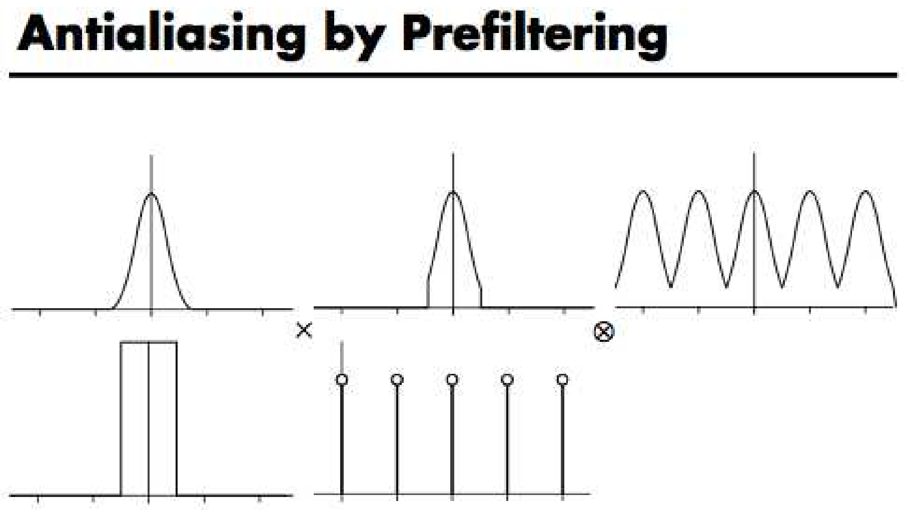
Undersampling: Aliasing

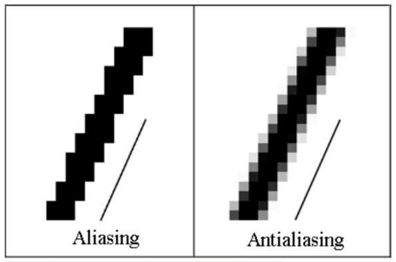


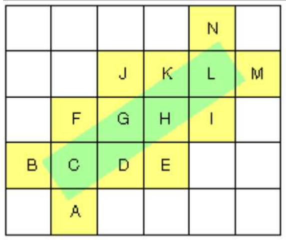
"Aliases"



Ideally, low-pass with a perfect filter (a sinc function) to bandlimit the function to the Nyquist sampling rate. Unfortunately, the sinc has infinite extent and we must use simpler filters (like a box filter, or area average).





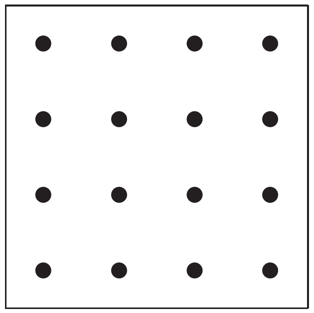


A	.040510
B	.040510
C	.878469
D	.434259
E	.007639
F	.141435
G	.759952
H	.759952
I	.141435
J	.007639
K	.434259
L	.878469
M	.040510
N	.040510

Uniform Supersampling

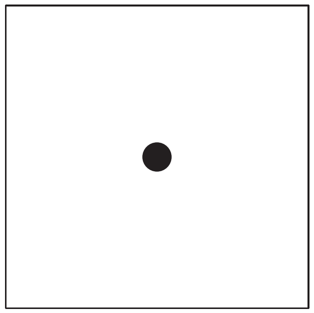
Increasing the sampling rate moves each copy of the spectra further apart, potentially reducing the overlap and thus aliasing

Resulting samples must be resampled (filtered) to image sampling rate



Samples

$$\text{Pixel} = \sum_s w_s \text{Sample}_s$$



Samples

Non-uniform Sampling

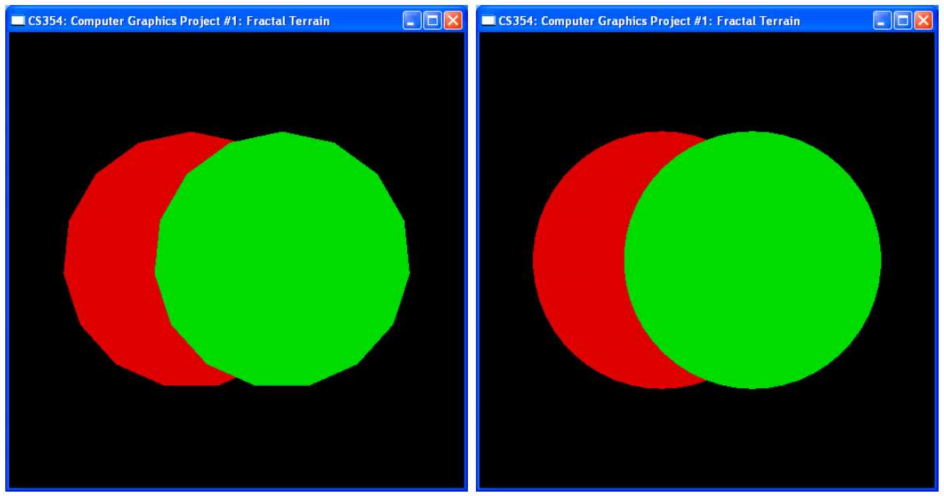
Uniform sampling

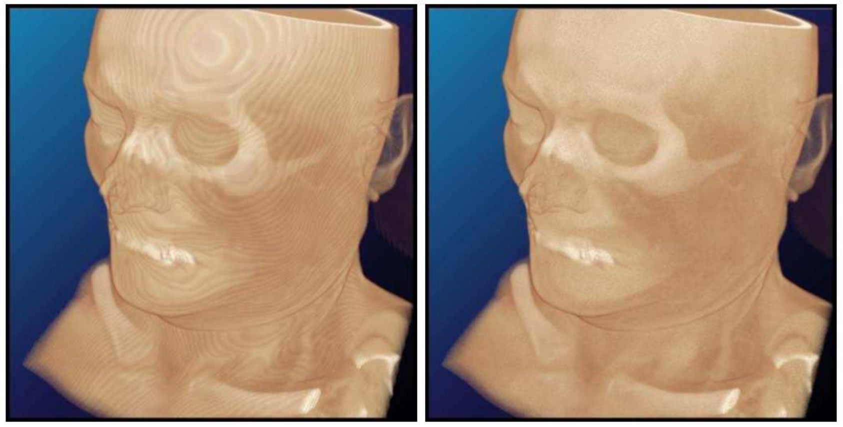
- The spectrum of uniformly spaced samples is also a set of uniformly spaced spikes
- Multiplying the signal by the sampling pattern corresponds to placing a copy of the spectrum at each spike (in frequency space)
- Aliases are coherent, and very noticable

Non-uniform sampling

- Samples at non-uniform locations have a different spectrum; a single spike plus noise
- Sampling a signal in this way converts aliases into broadband noise
- Noise is incoherent, and much less objectionable

Antialiasing via Non-uniform sampling


CS354: Computer Graphics Project #1: Fractal Terrain
CS354: Computer Graphics Project #1: Fractal Terrain



Reading Assignment and News

Please review the appropriate anatialiasing sections related to this lecture in chapter 7, and associated exercises, of the recommended text.

(Recommended Text: Interactive Computer Graphics, by Edward Angel, Dave Shreiner, 6th edition, Addison-Wesley)

Please track Blackboard for the most recent Announcements and Project postings related to this course.

(http://www.cs.utexas.edu/users/bajaj/graphics2012/cs354/)