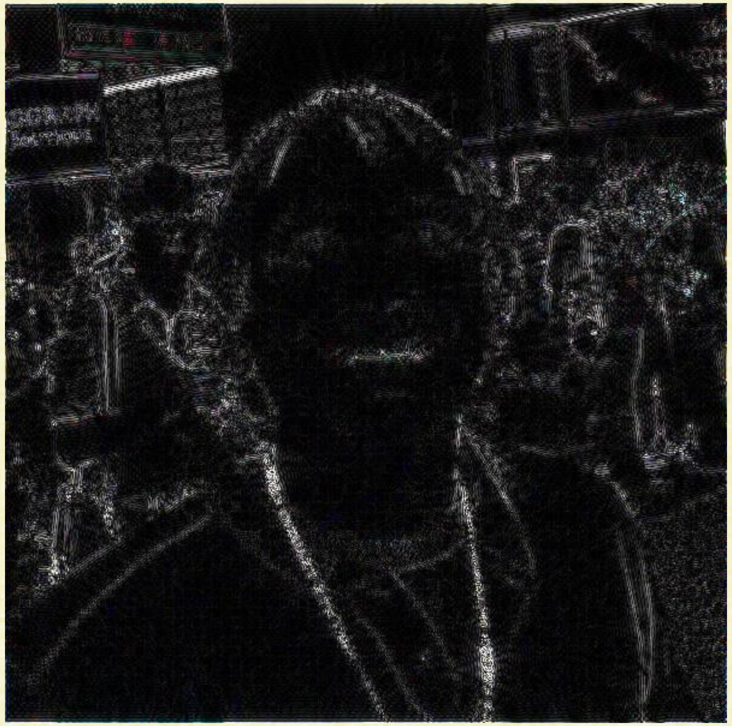
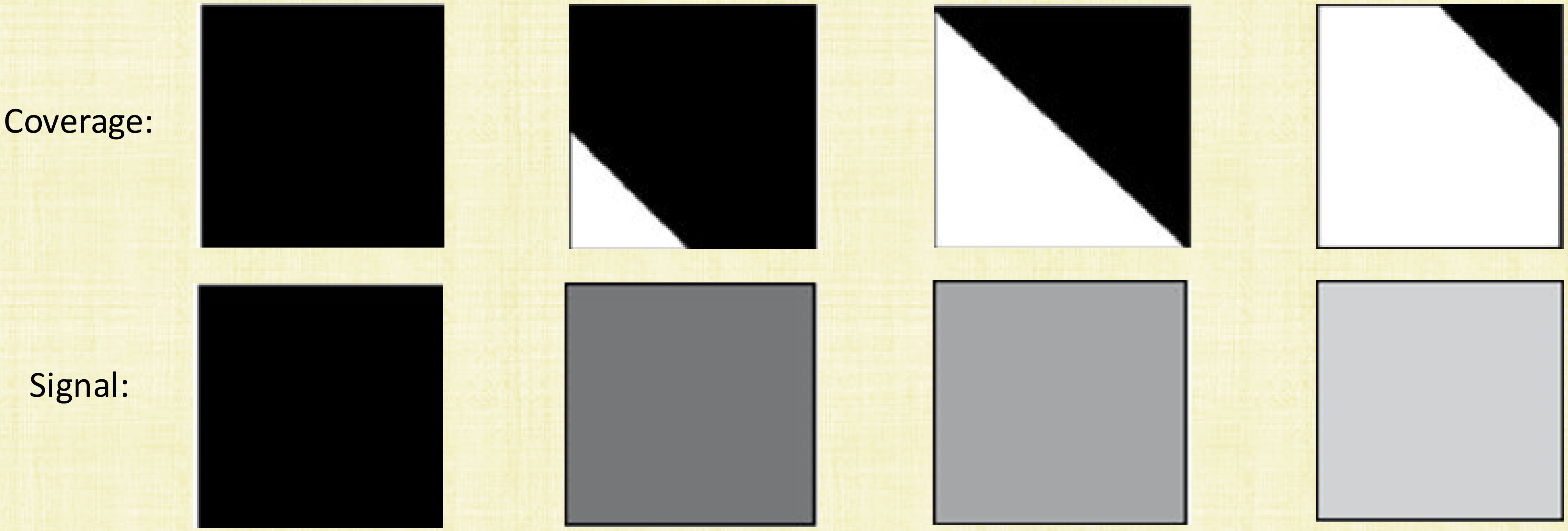
Sampling



Area-Coverage

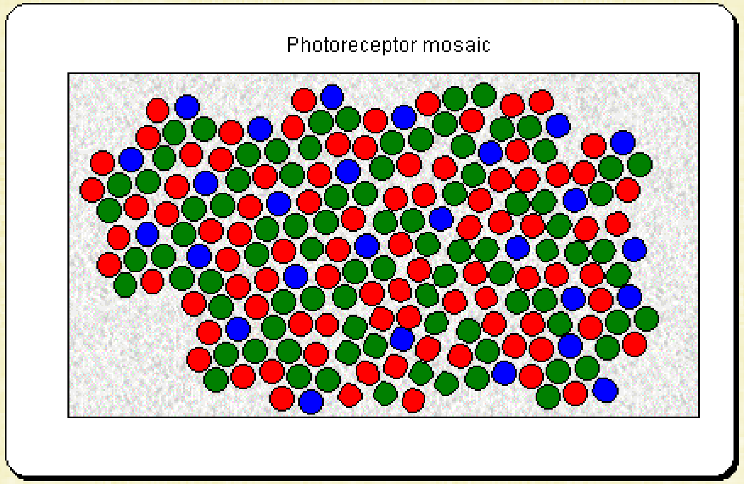
- Real-world sensors obtain a signal based on the fraction their area "covered" by objects

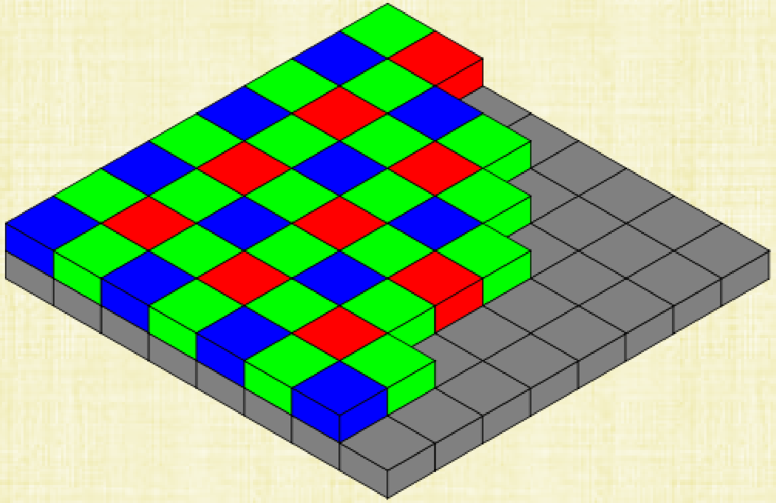


- A ray tracer only gets a sample of the geometry, using a ray-geometry intersection point
- A scanline renderer projects the entire triangle onto the image plane
  - Testing pixel centers against triangles only samples information from the geometry
  - Computing area overlap between triangles and (square) pixels would better mimic real-world sensors

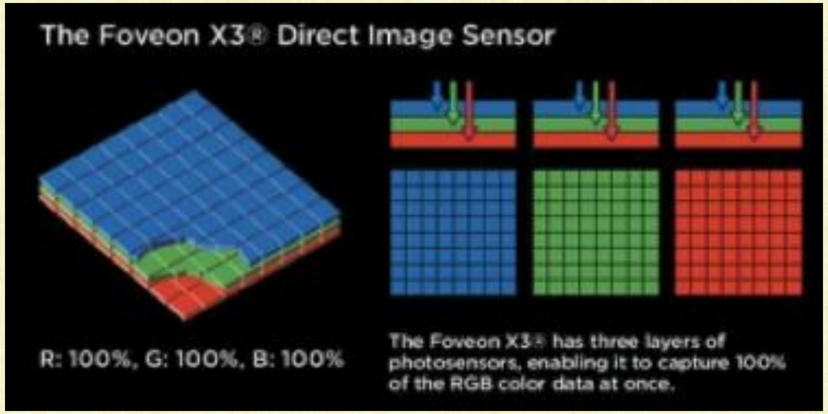
Missing Information

- Eyes/cameras don’t collect all the information either
- The staggered spatial layout of real-world sensors means that there are gaps in information



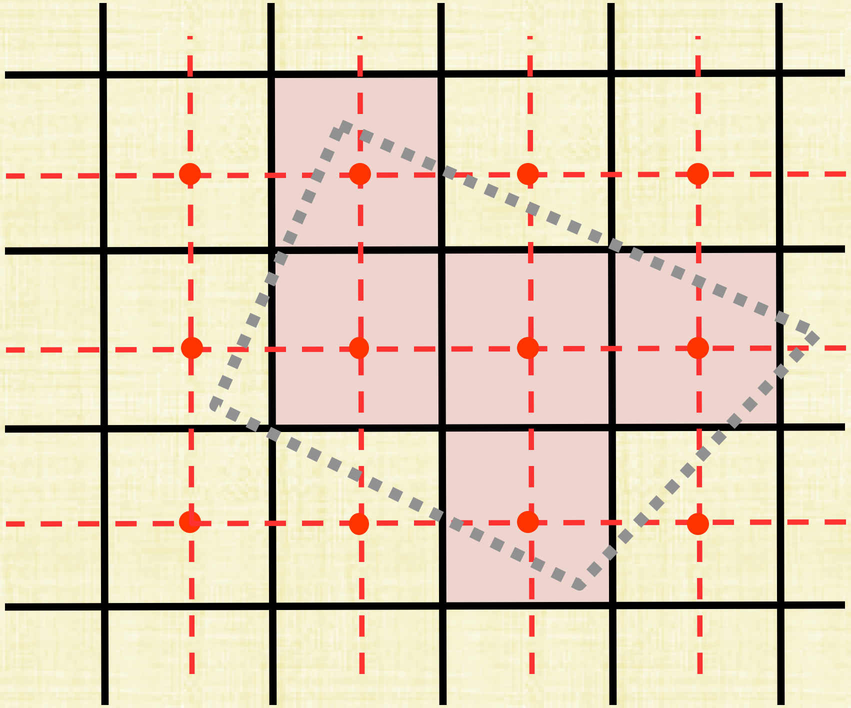


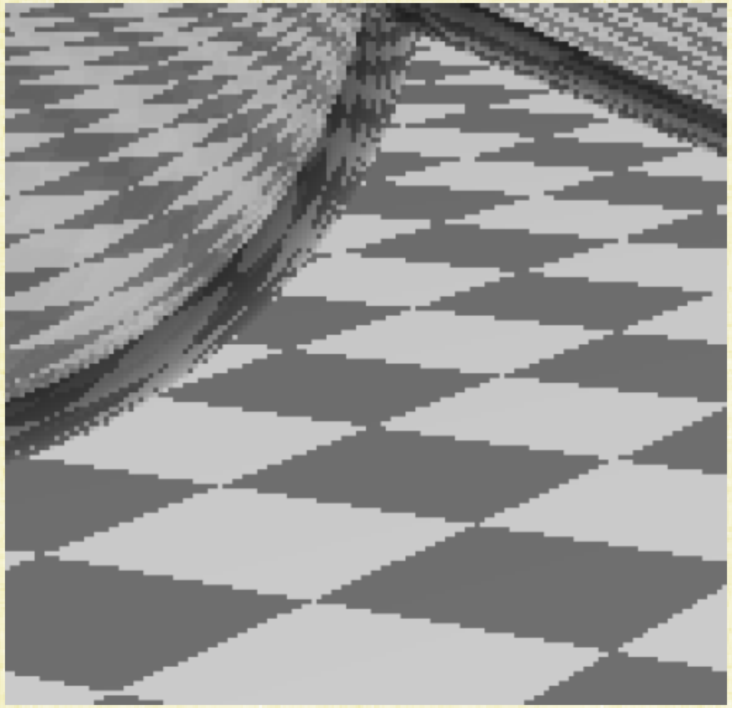
layered approaches could help to circumvent this:



Aliasing

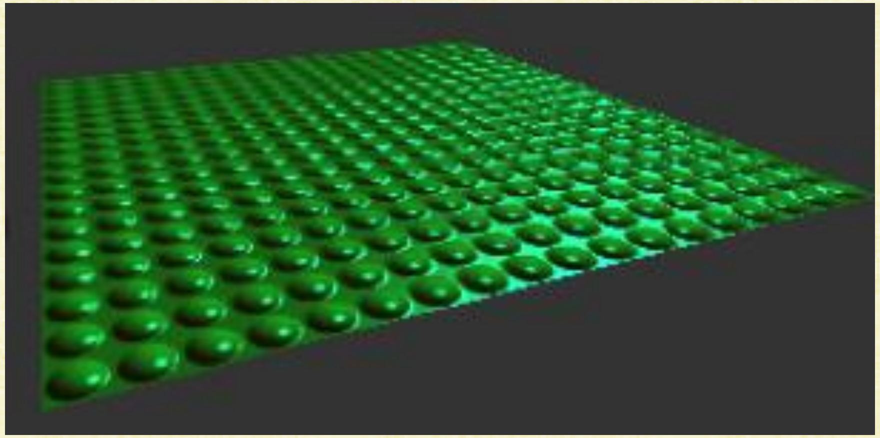
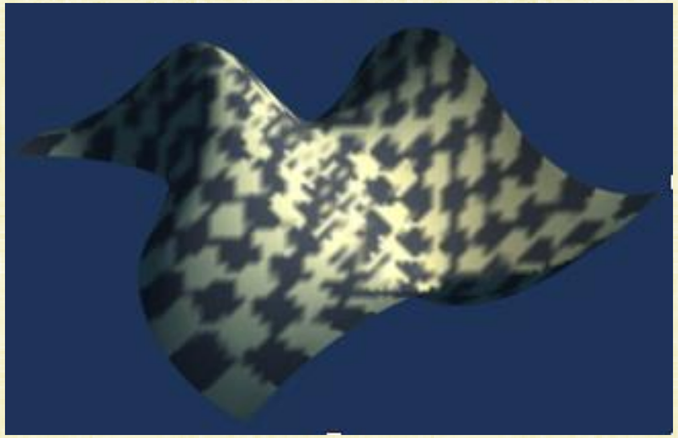
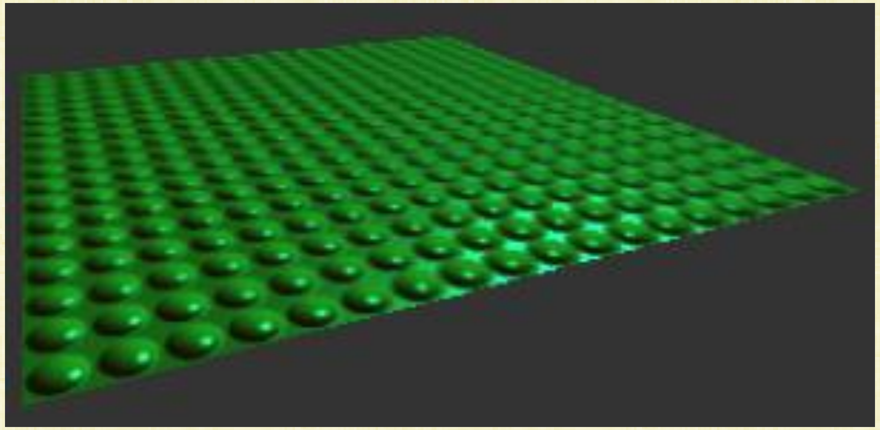
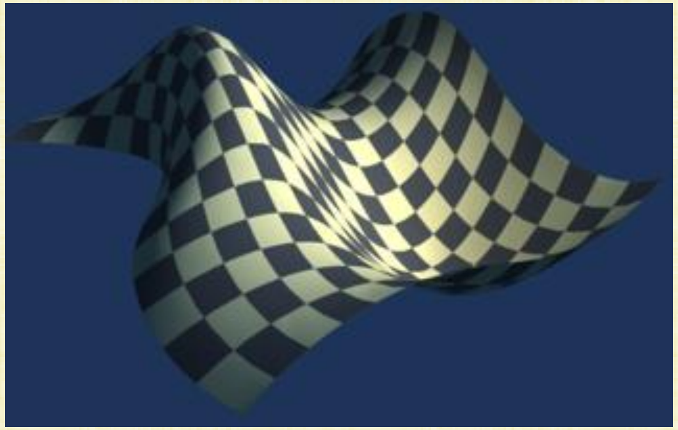
- Testing only the pixel center (with ray-tracing or scanline rasterization) leads to jagged edges
- This sampling causes aliasing artifacts
  - An alias/imposter takes the place of the correct feature
  - A jagged line appears as an imposter, instead of the correct straight line
- Anti-aliasing strategies aim to reduce aliasing artifacts, caused by sampling information





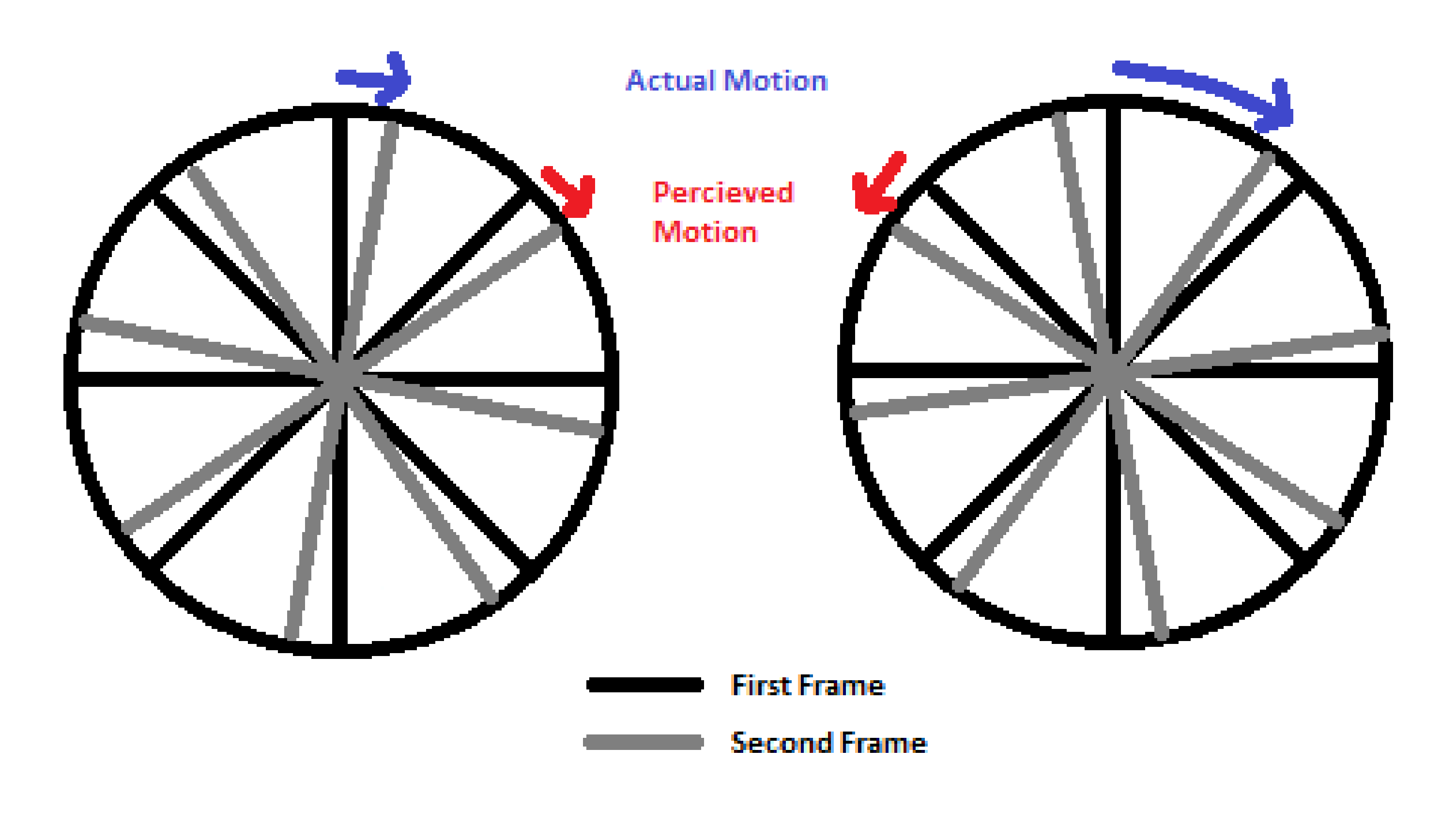
Aliasing: Shaders & Textures

- Aliased normal vectors can cause erroneous sparkling highlights (top left)
- Aliasing can occur when texture mapping objects (top right)

Temporal Aliasing

- A spinning wheel can appear to spin backwards, when the motion is insufficiently sampled in time ("wagon wheel" effect)



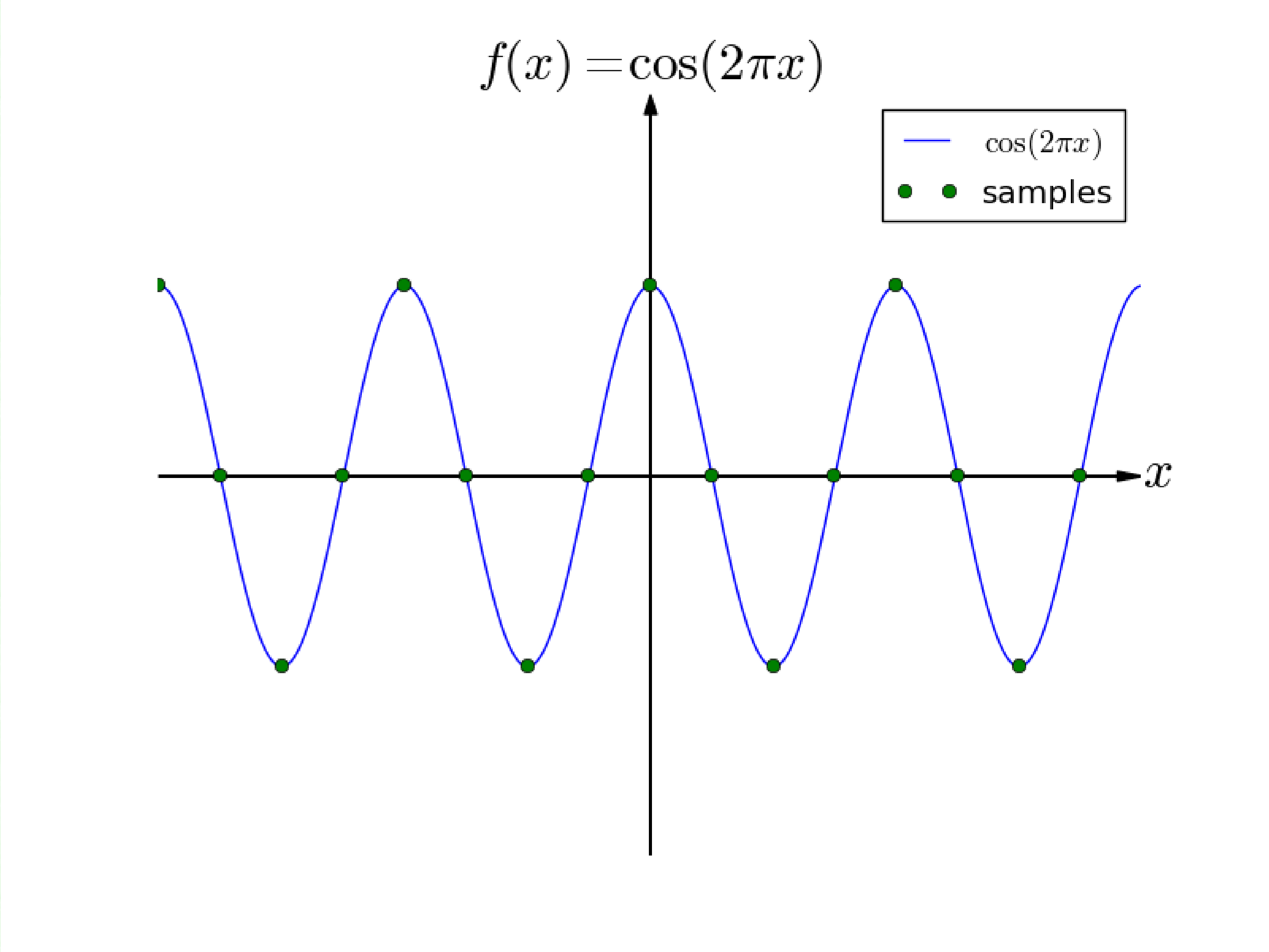
Sampling Rate

- Artifacts can be reduced by increasing the number of samples
- This can be accomplished by increasing the number of pixels in the image; but:
  - It takes longer to render the scene (because there are more pixel colors to determine)
  - Displaying higher-resolution images requires additional storage and computation

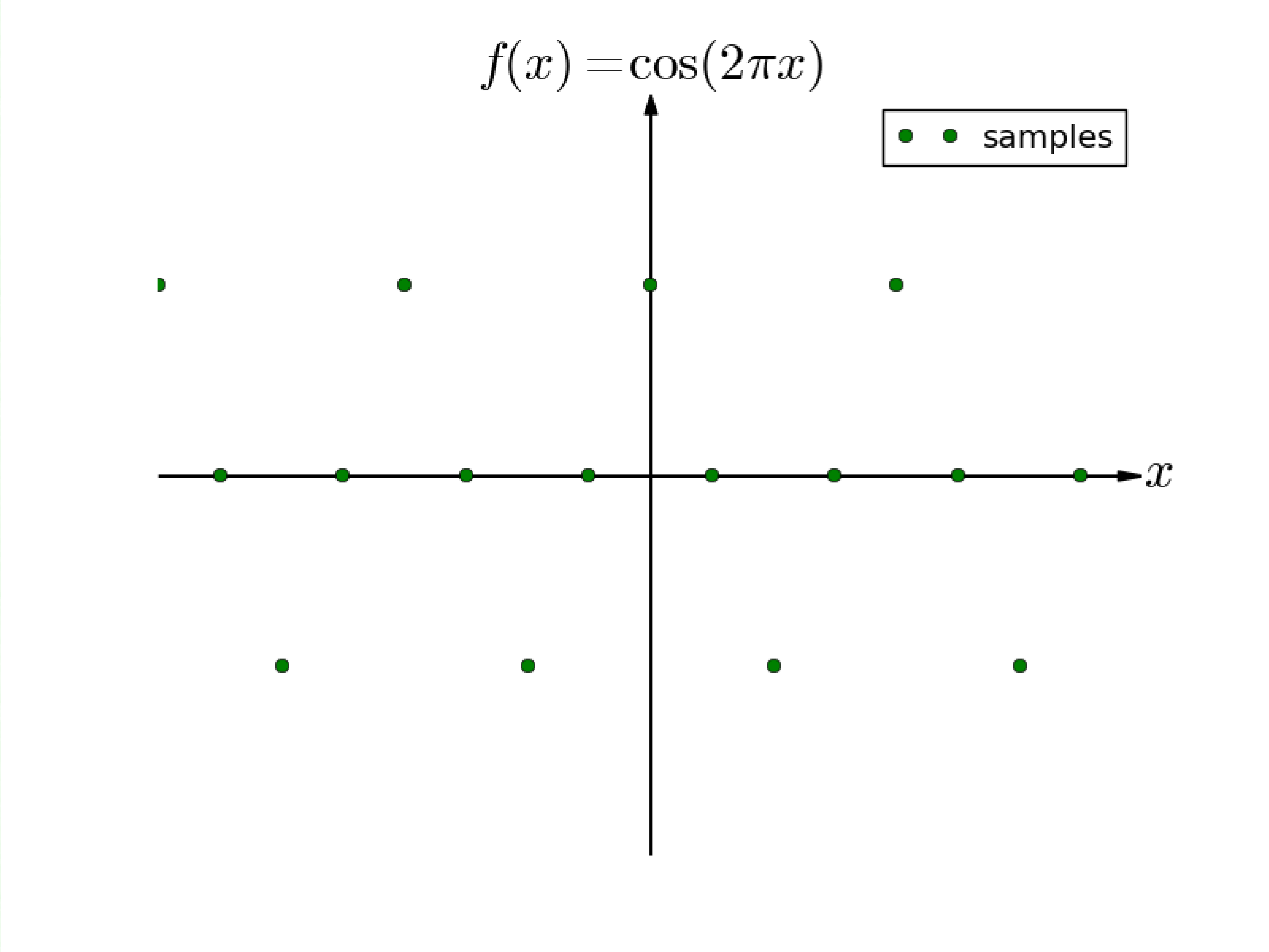
- Thus: Choose the lowest possible sampling rate that doesn't create "noticeable" artifacts

- What is the optimal sampling rate?

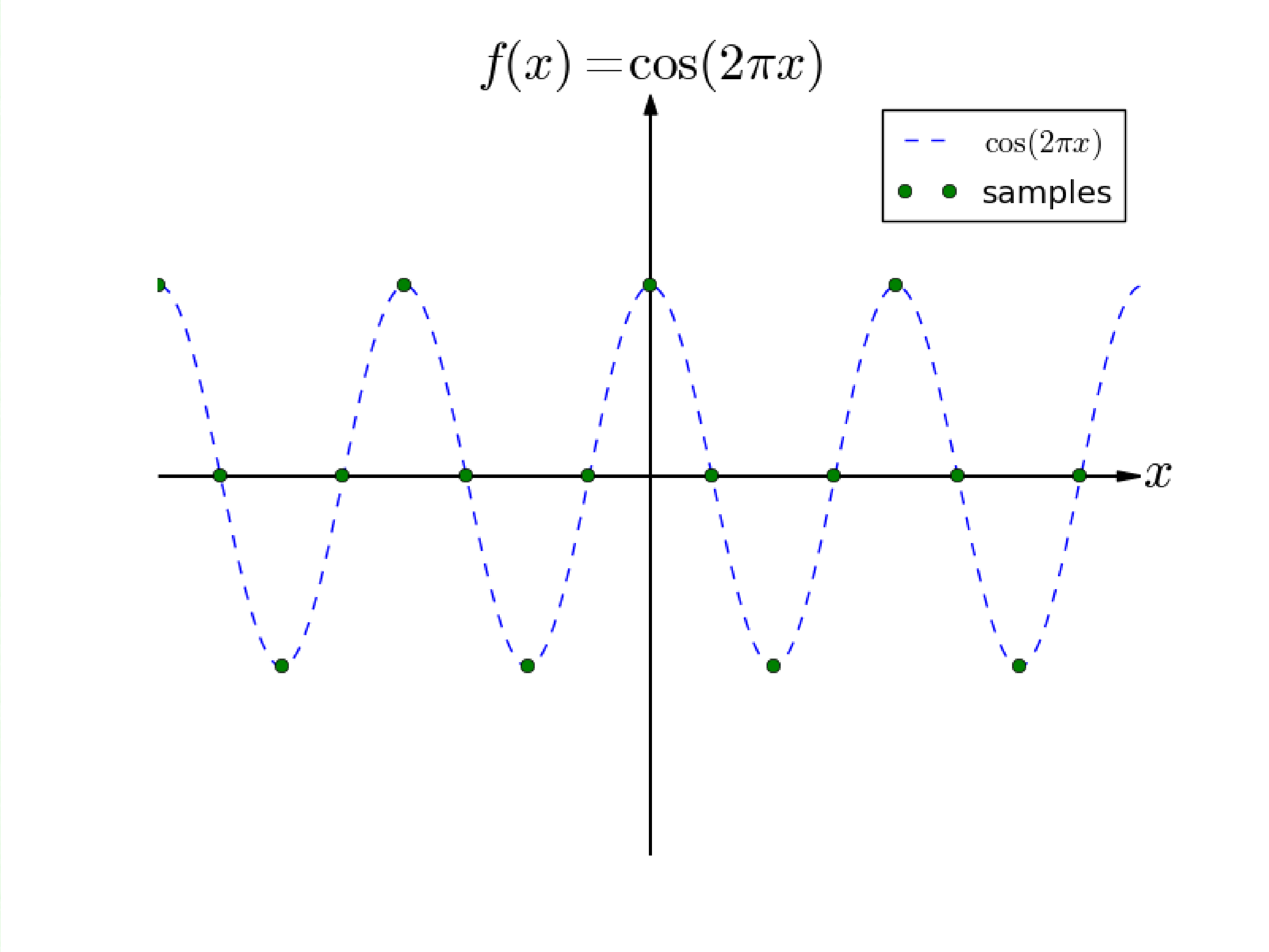
4 samples per period



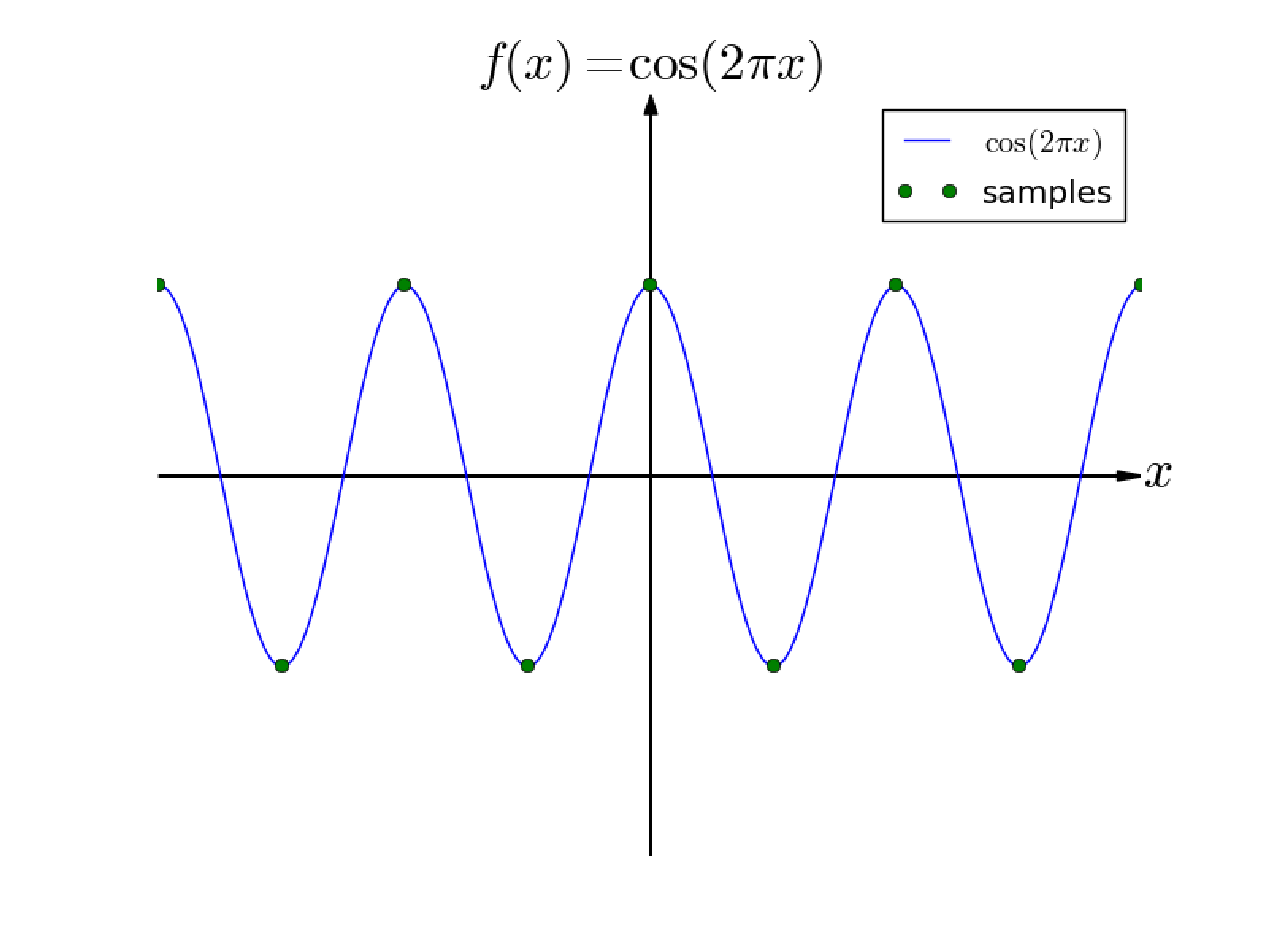
samples



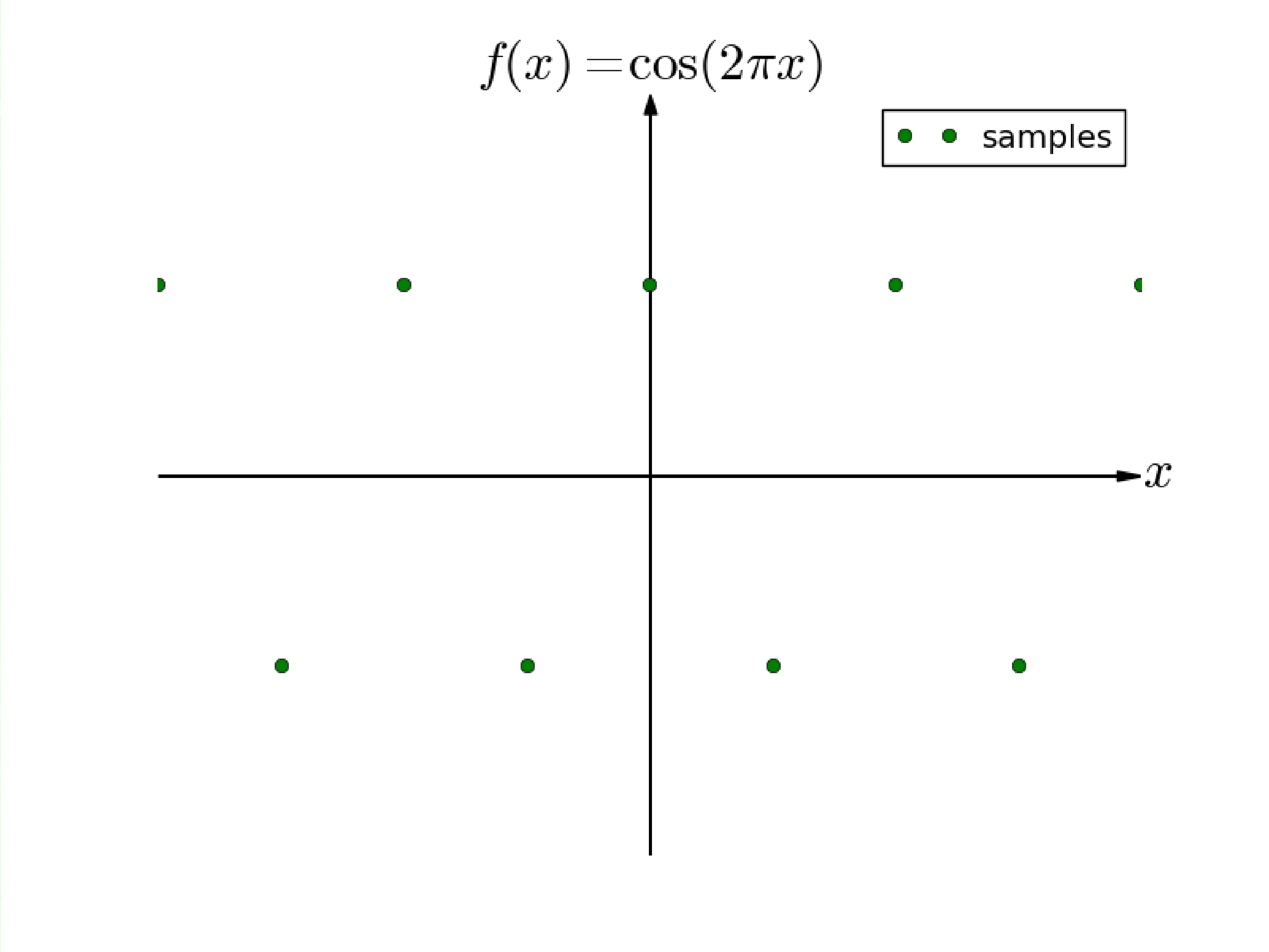
reconstruction



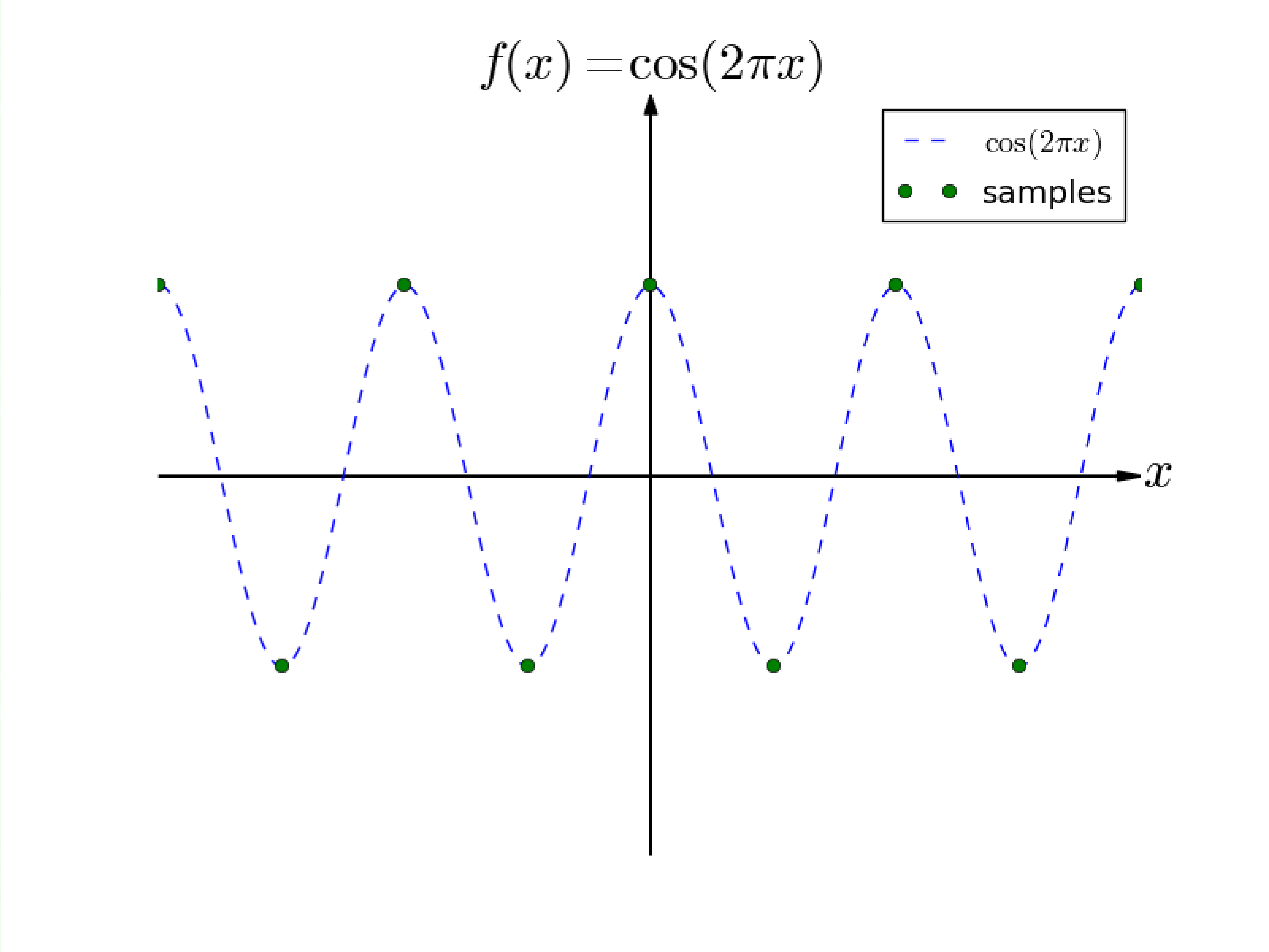
2 samples per period



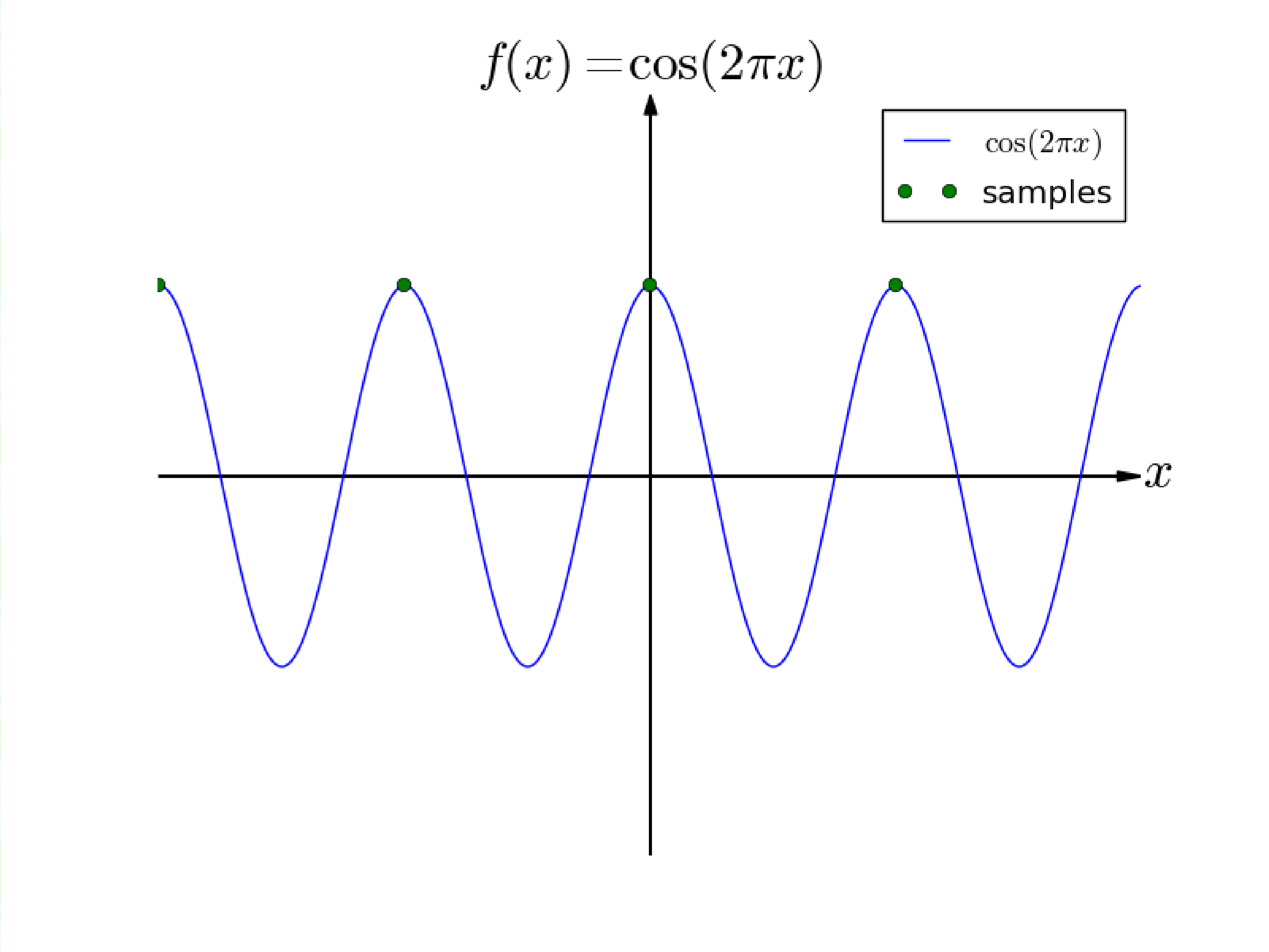
samples



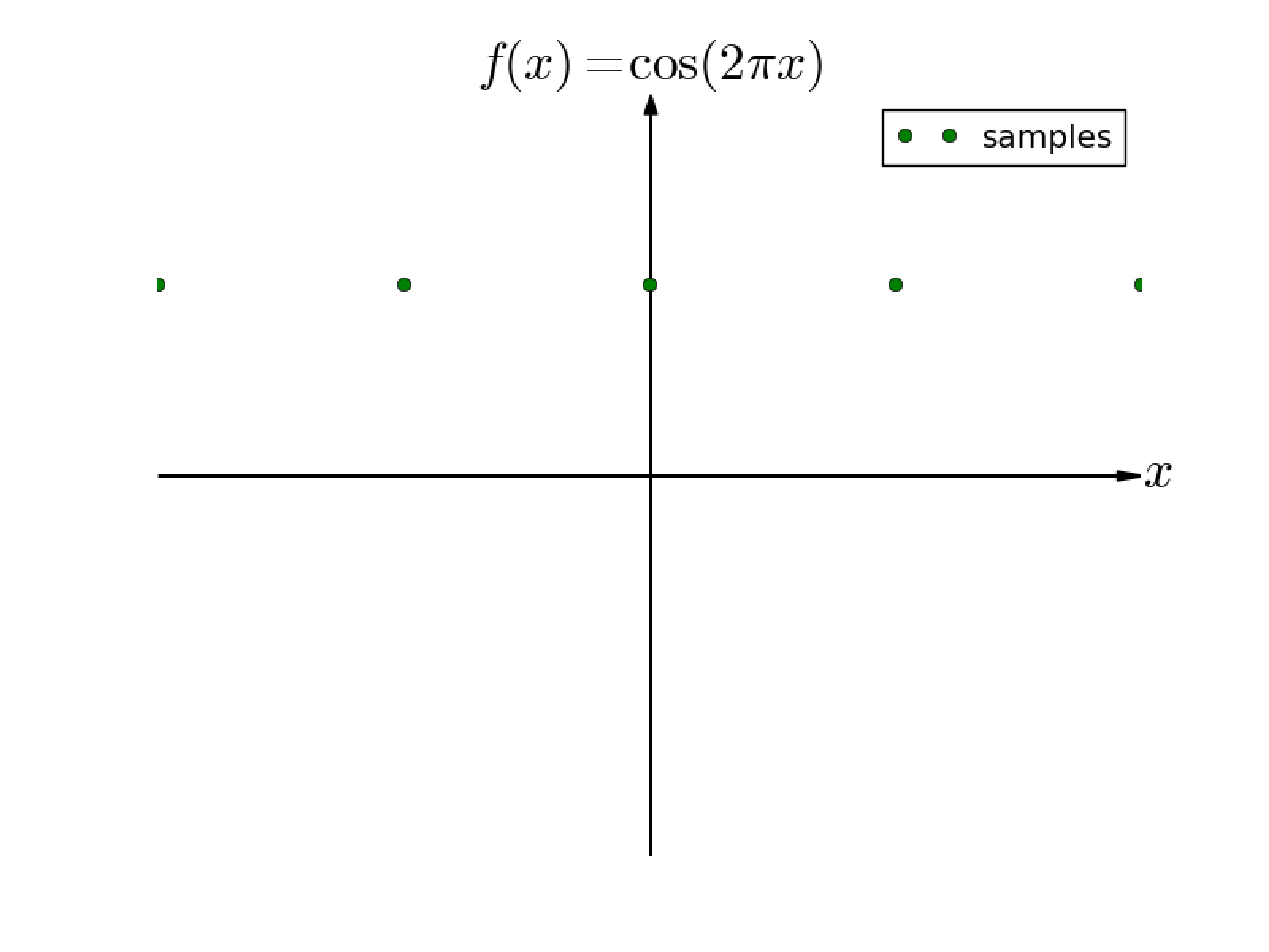
reconstruction



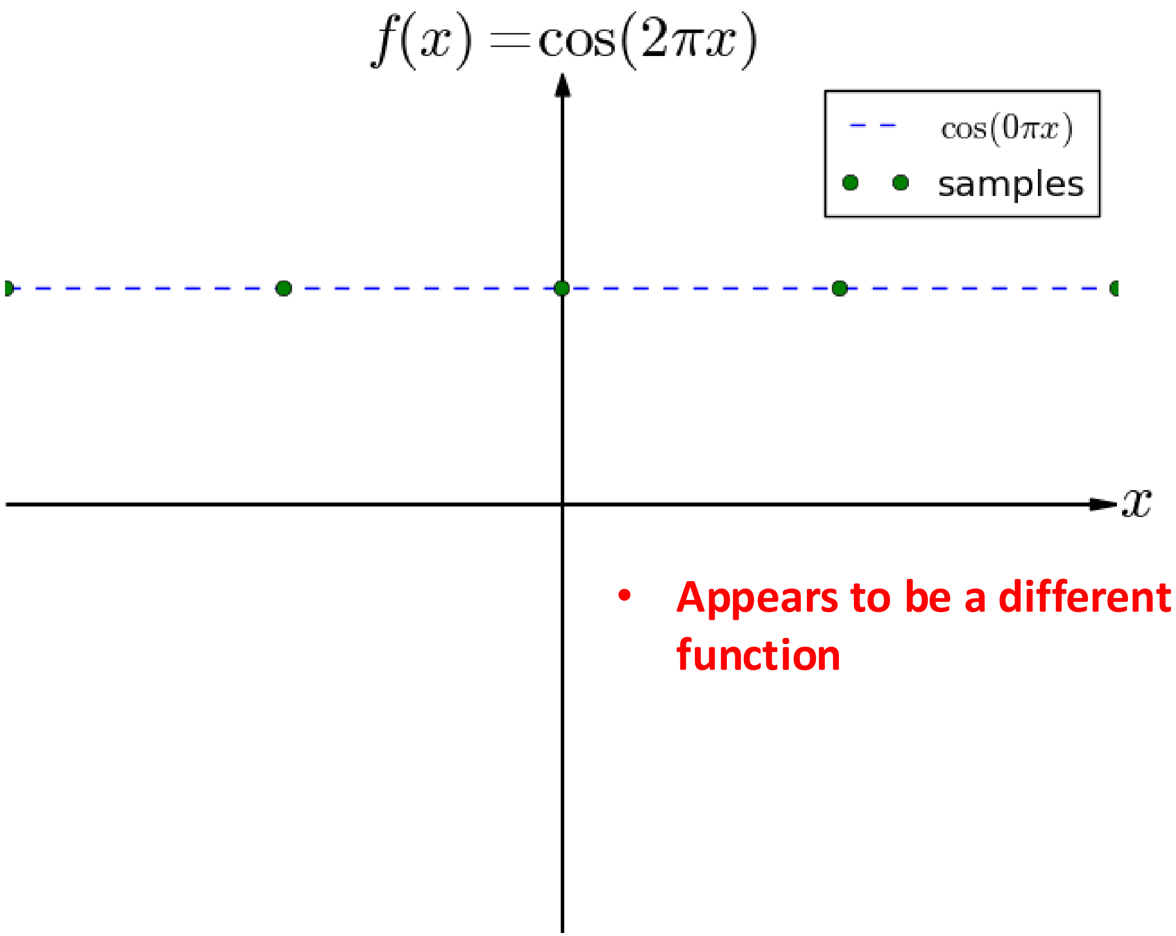
1 sample per period



samples

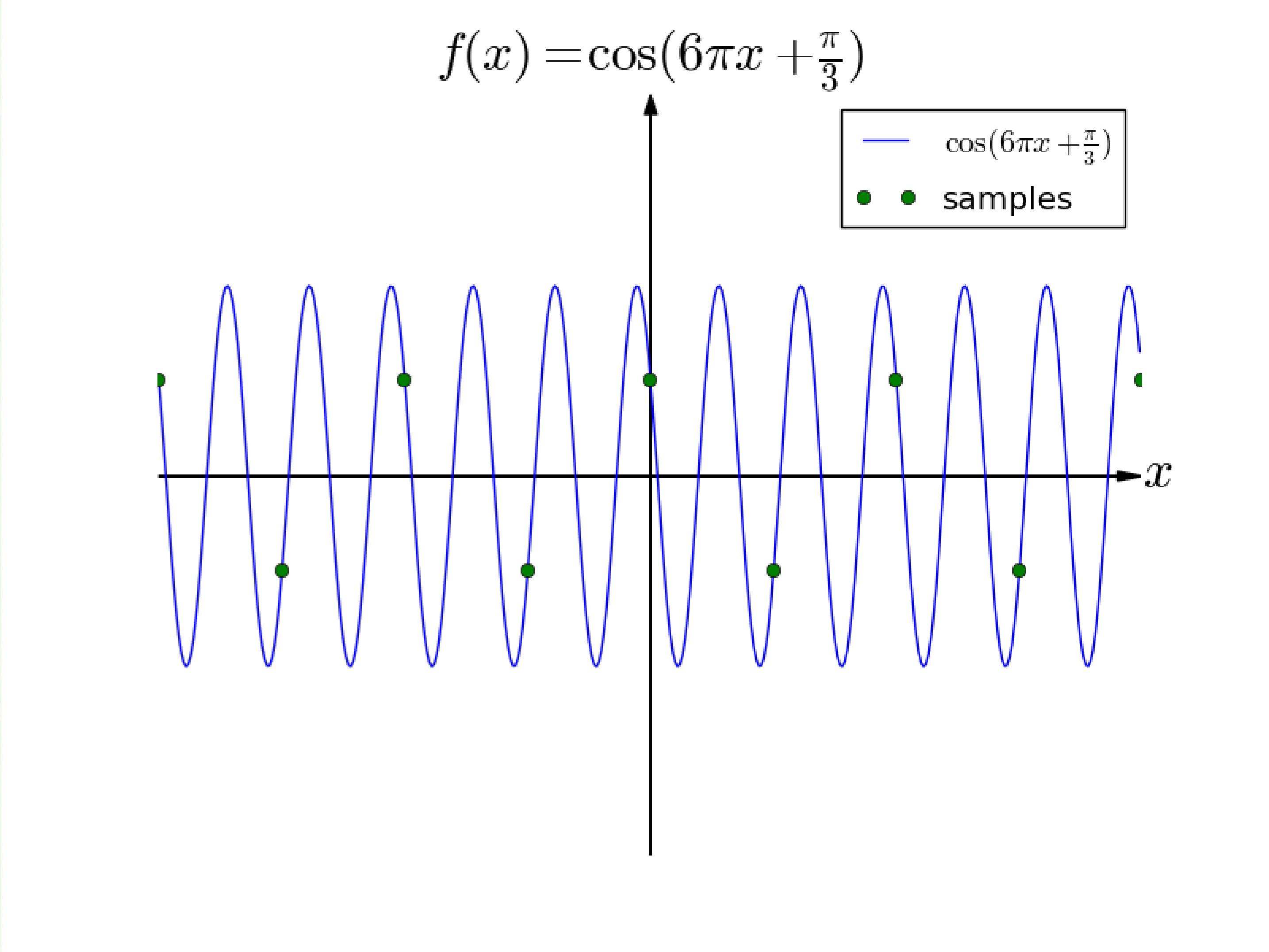


reconstruction

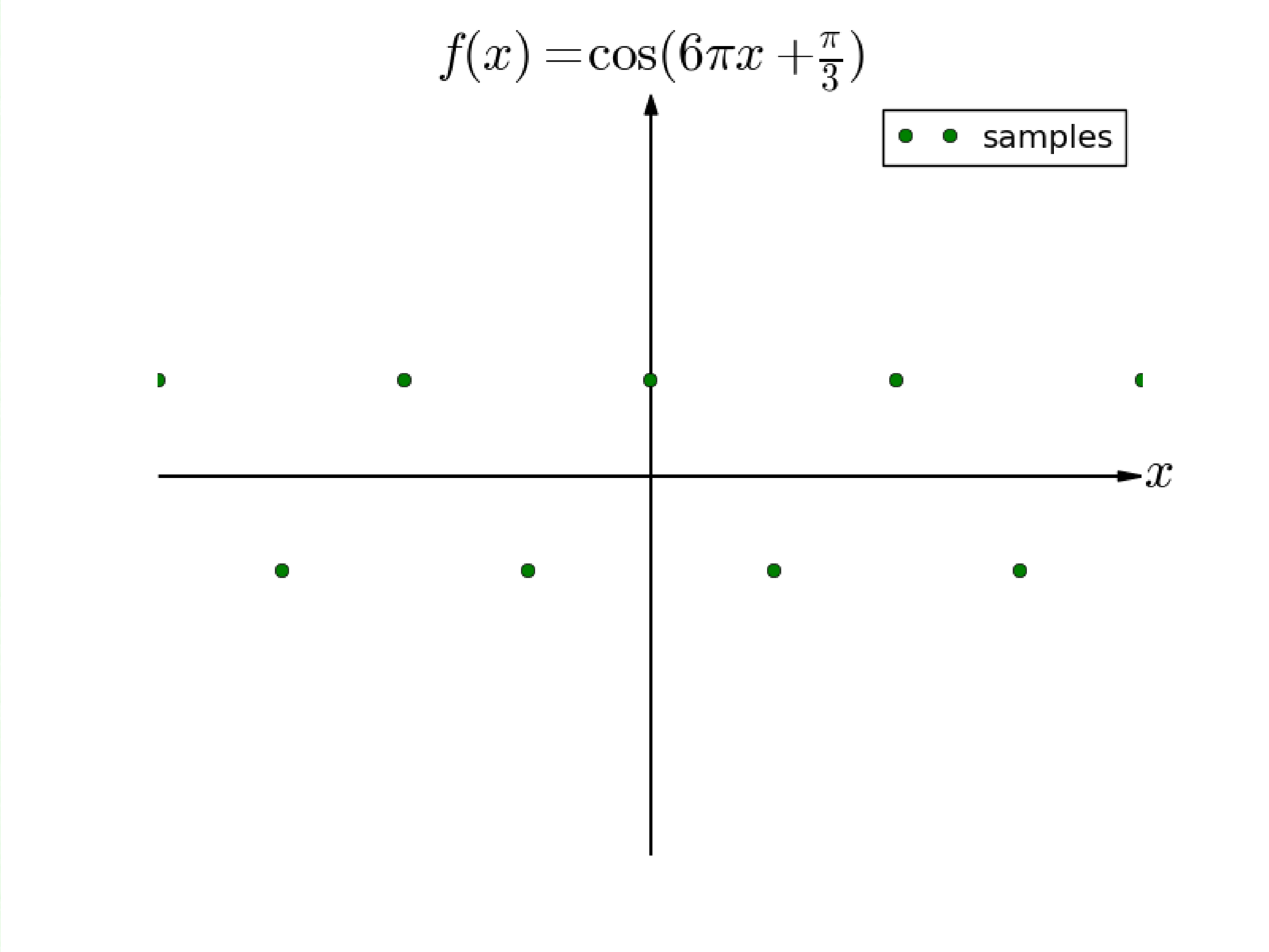


- **Appears to be a different function**

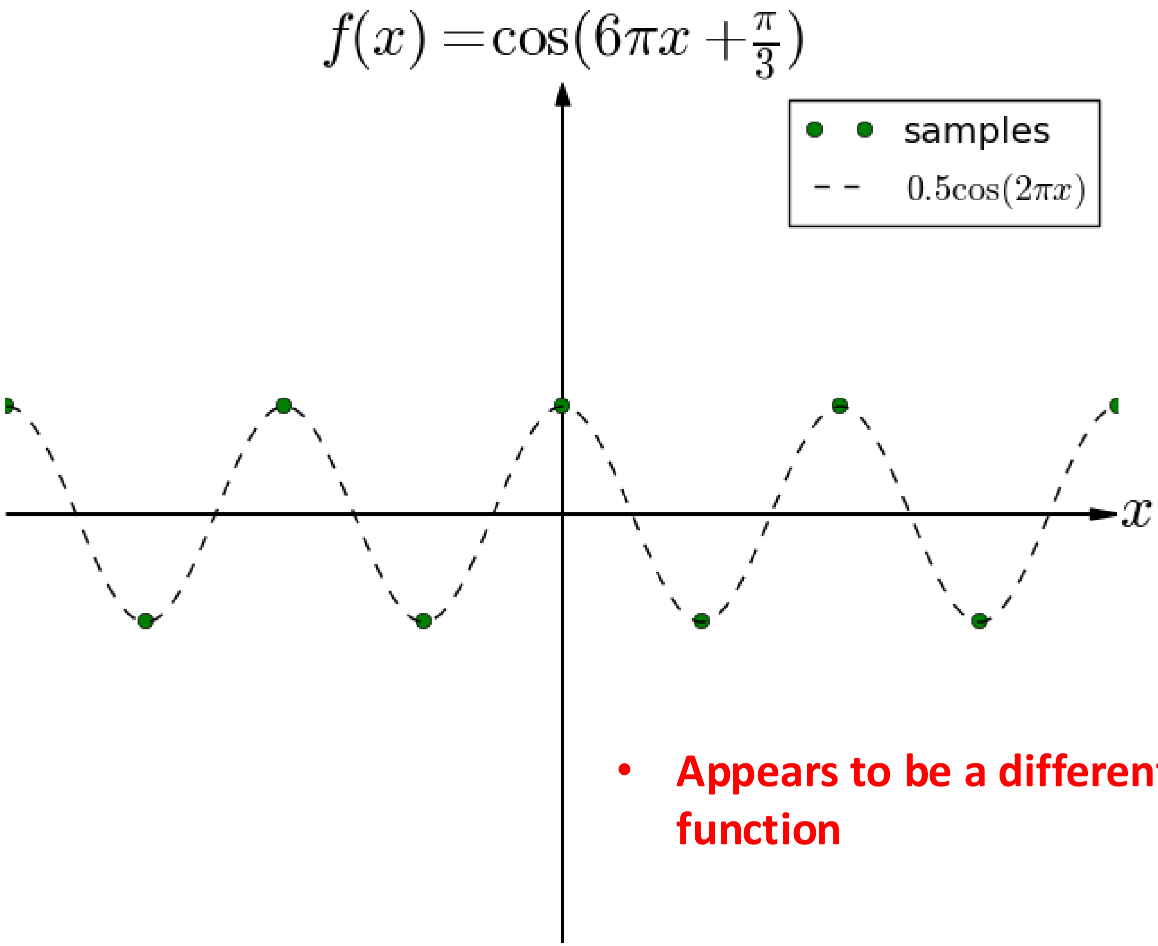
2/3 sample per period



samples

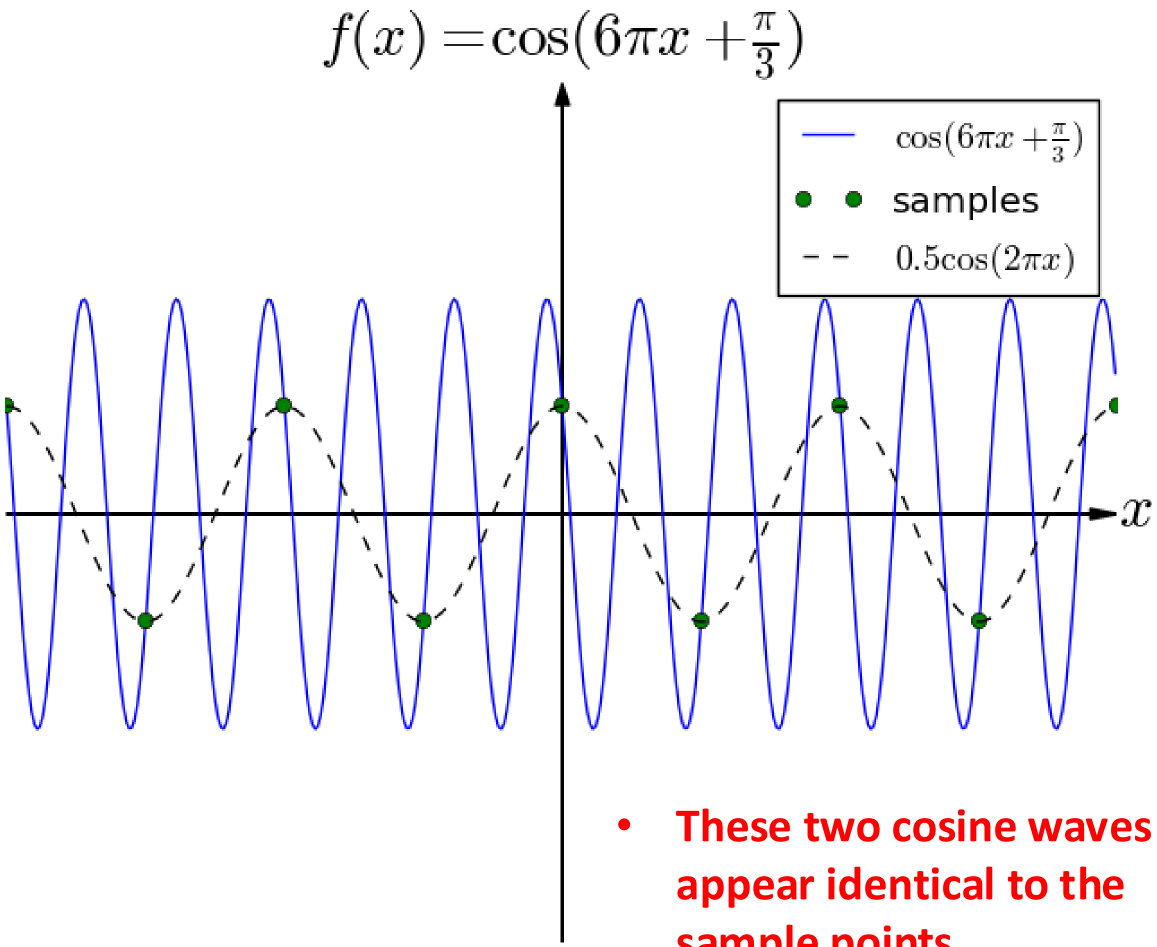


reconstruction



- **Appears to be a different function**

Aliasing



- **These two cosine waves appear identical to the sample points**

Sampling Rate

- Sampling at too low a rate results in aliasing, where two different signals become indistinguishable (or aliased)

- Nyquist-Shannon Sampling Theorem
  - If $f(t)$ contains no frequencies higher than W hertz, it can be completely determined by samples spaced $1/(2W)$ seconds apart
  - That is, a minimum of 2 samples per period are required to prevent aliasing

Anti-Aliasing

- The Nyquist frequency is defined as half the sampling frequency
- If the function being sampled has no frequencies above the Nyquist frequency, then no aliasing occurs

- ***Real world frequencies above the Nyquist frequency appear will appear as aliases to the sampler***
- **Before sampling, remove frequencies higher than the Nyquist frequency**

Fourier Transform

- Transform between the spatial domain $f(x)$ and the frequency domain $F(k)$

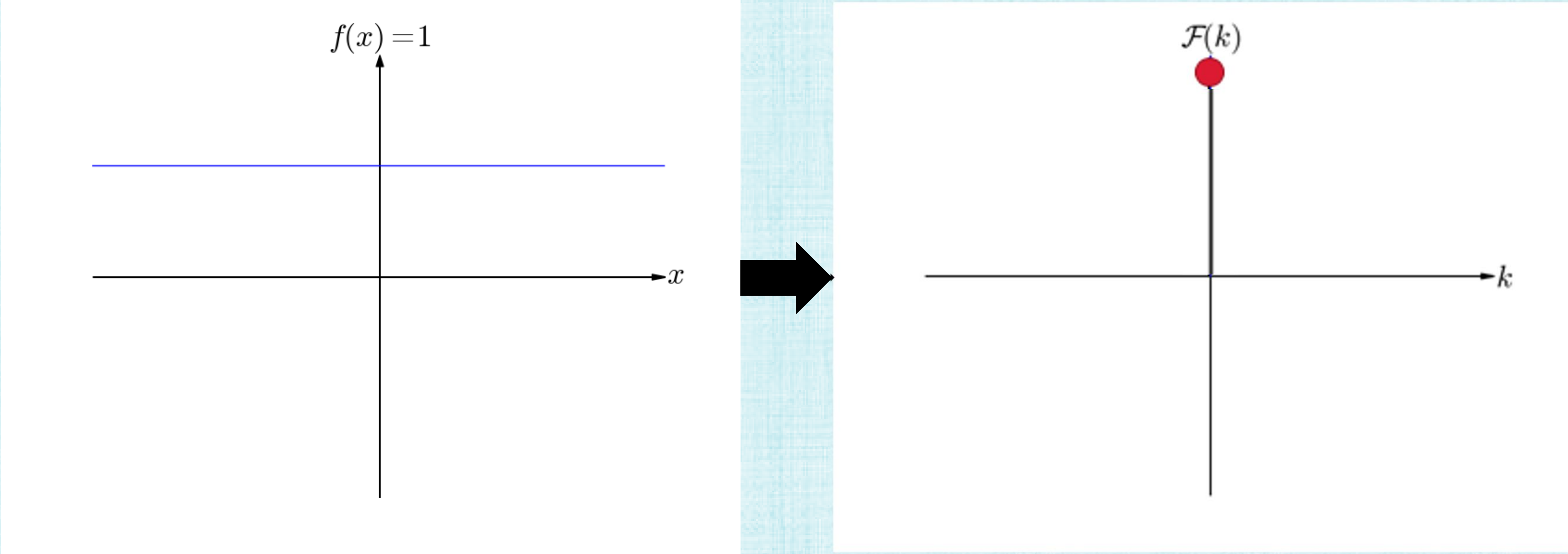
Spatial to Frequency Domain: $F(k) = \int_{-\infty}^{\infty} f(x) e^{-2\pi ikx} dx$

Frequency to Spatial Domain: $f(x) = \int_{-\infty}^{\infty} F(k) e^{2\pi ikx} dk$

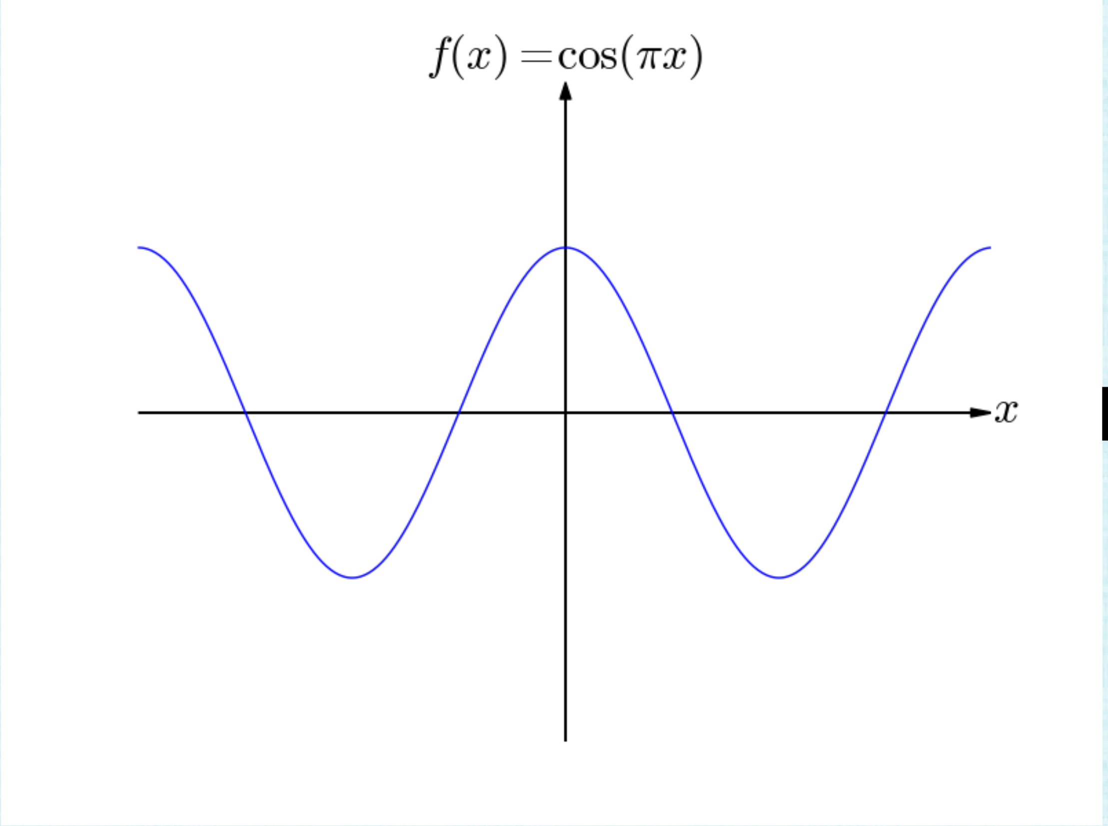
$$e^{i\theta} = \cos\theta + i\sin\theta$$

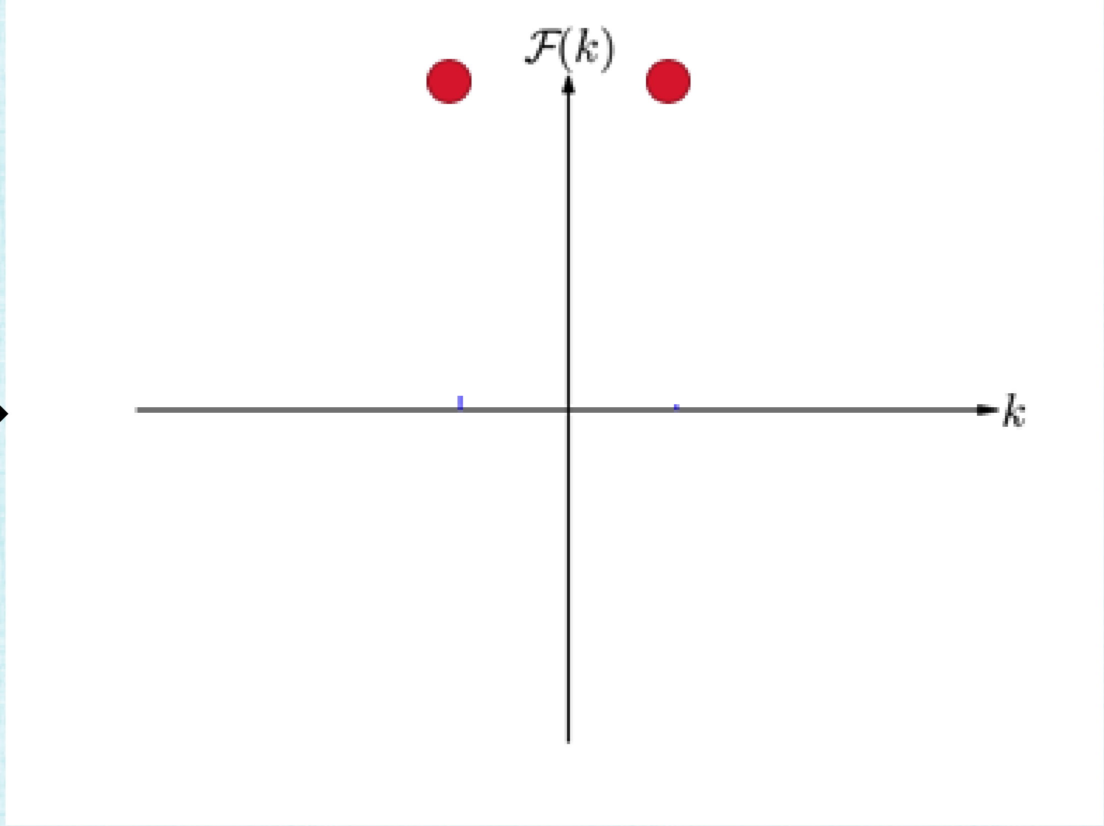
$$\cos\theta = \frac{e^{i\theta} + e^{-i\theta}}{2} \qquad \sin\theta = \frac{e^{i\theta} - e^{-i\theta}}{2i}$$

Constant Function

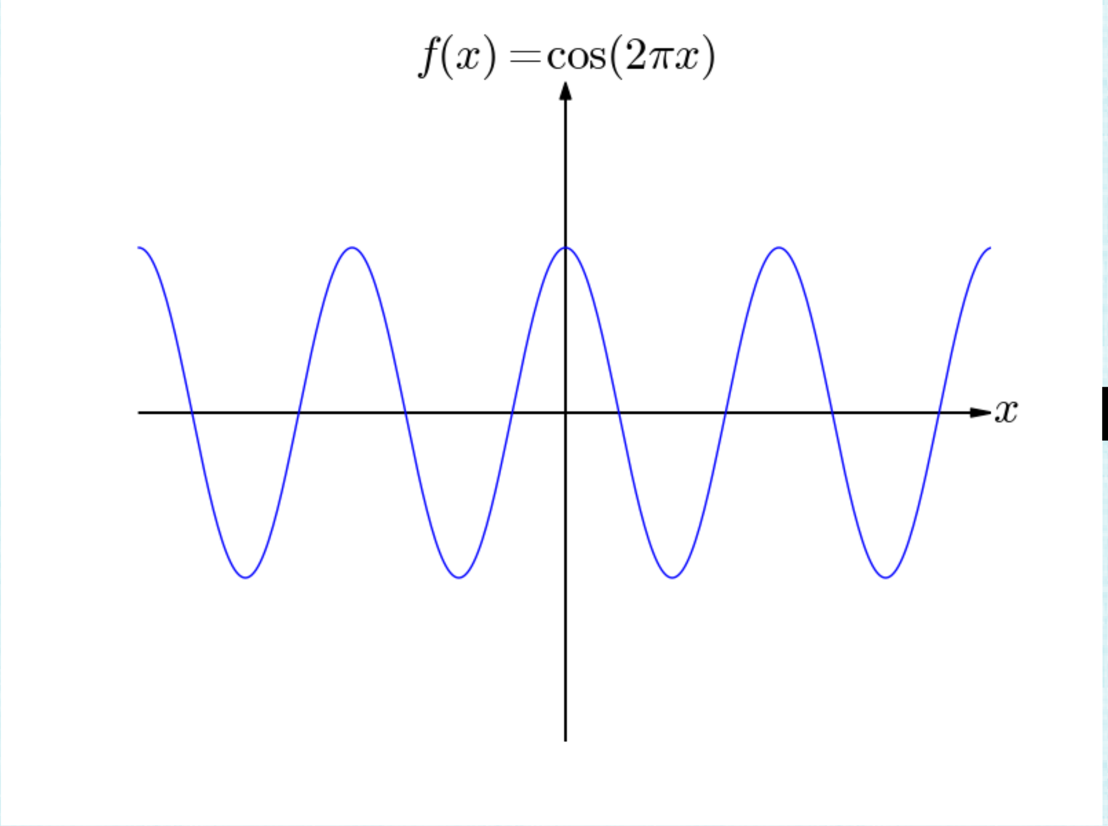


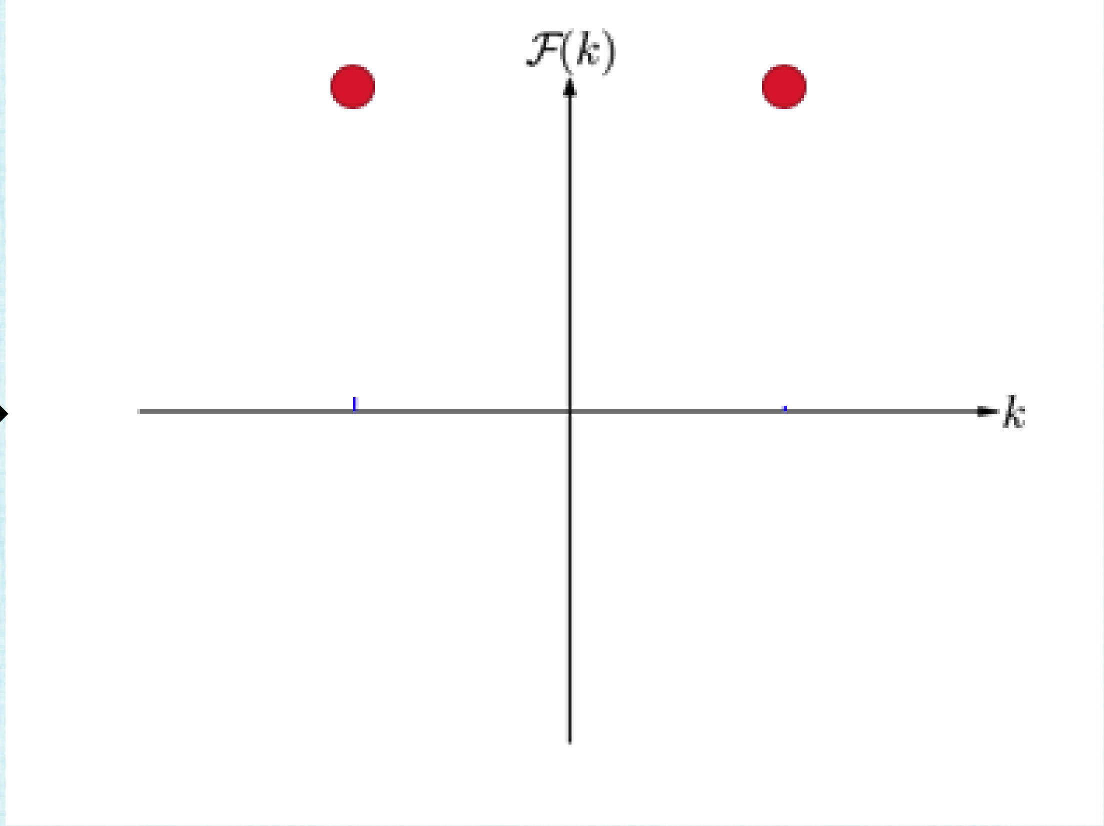
Low Frequency Cosine



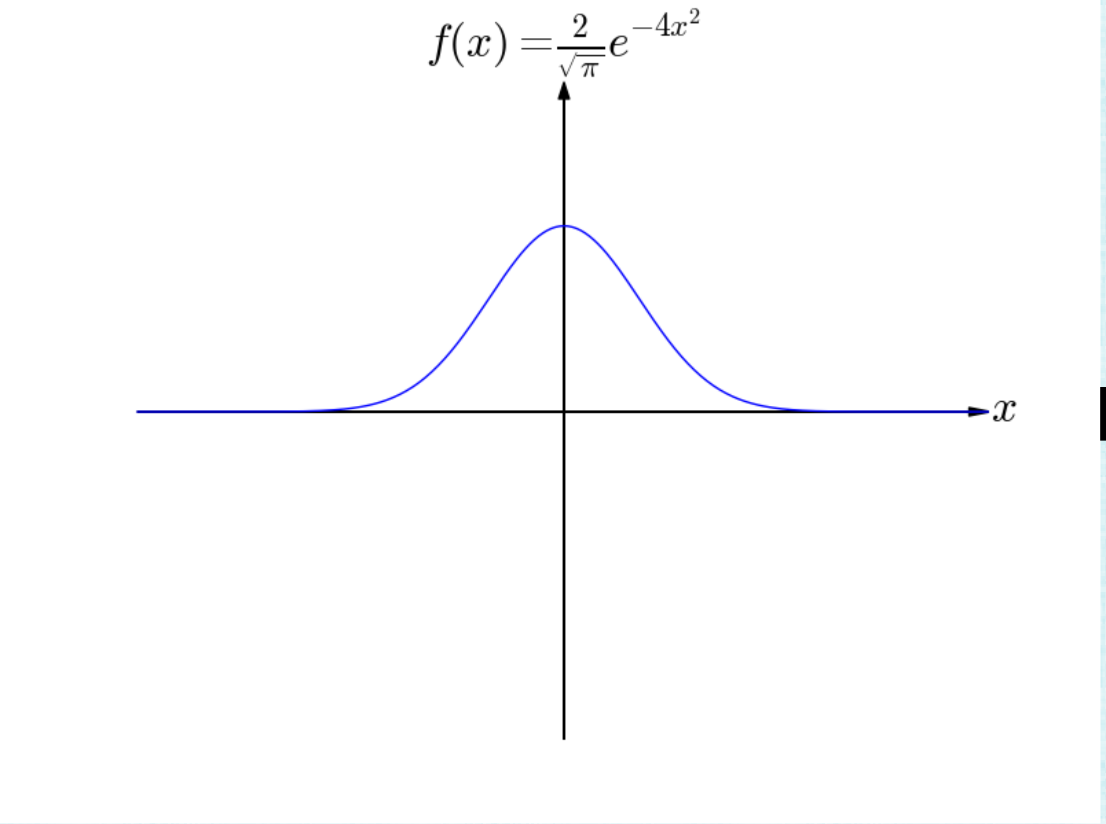


High Frequency Cosine

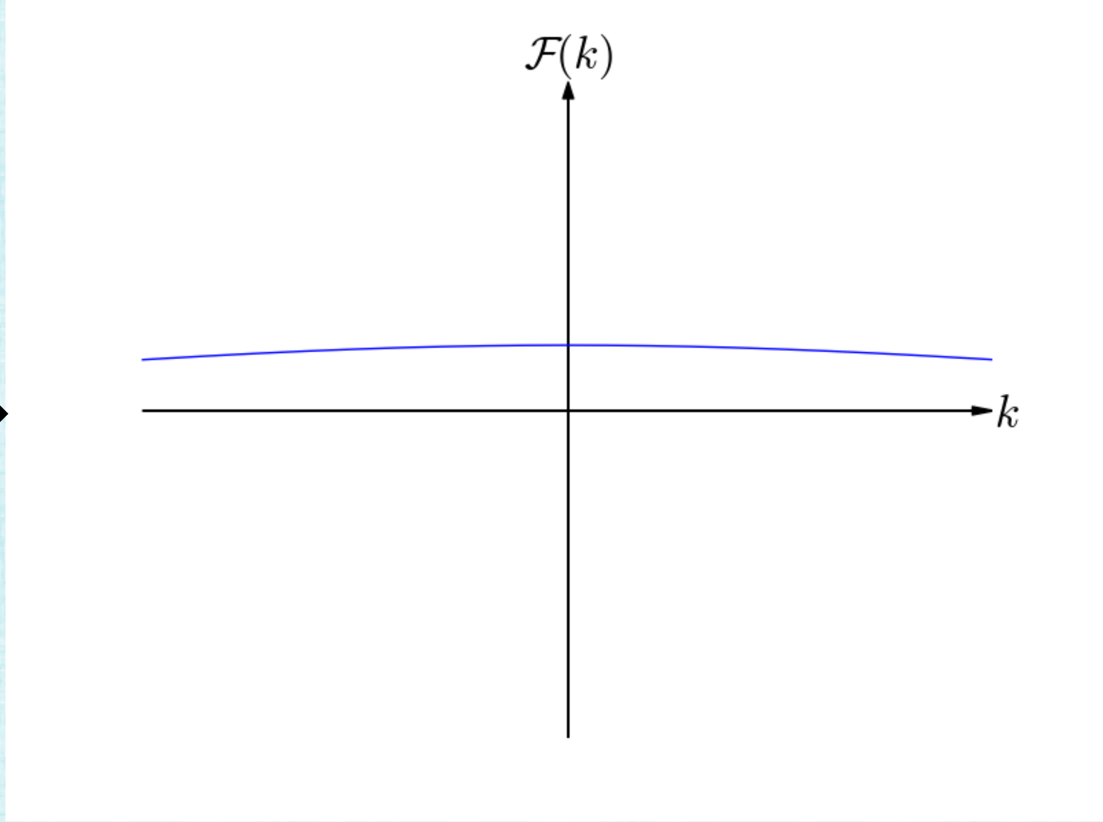




Narrow Gaussian

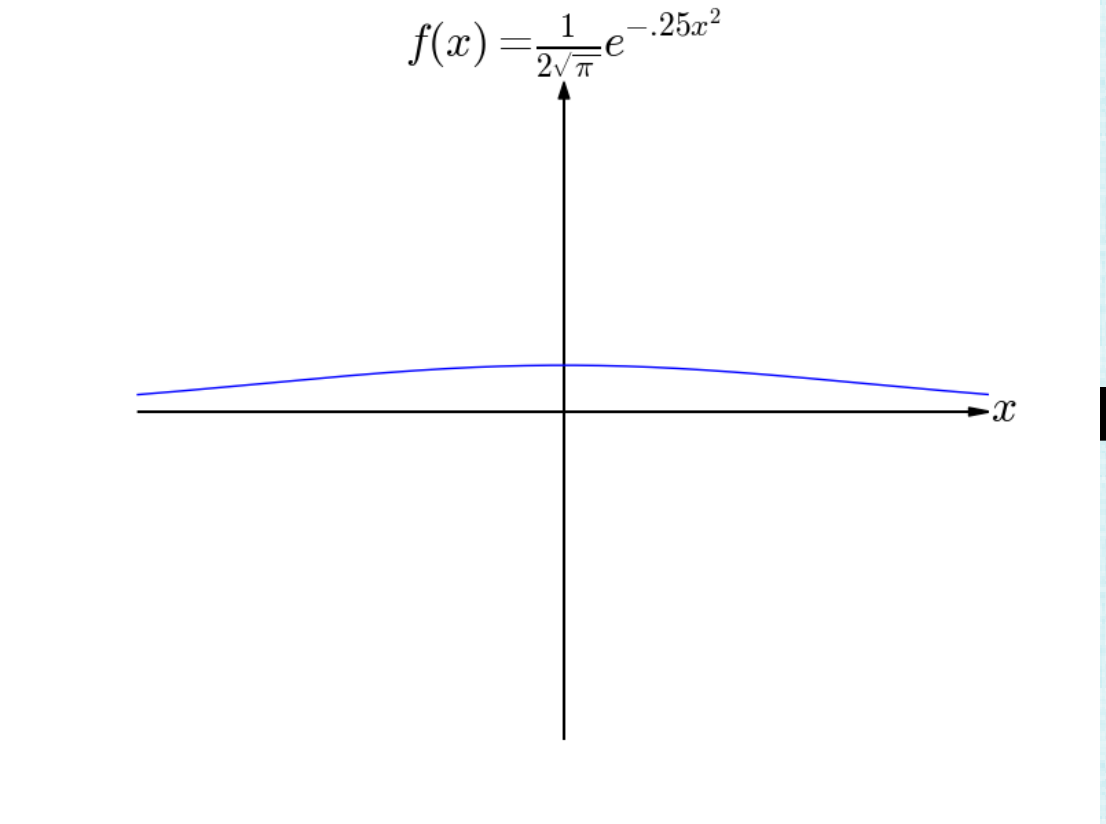


Narrow

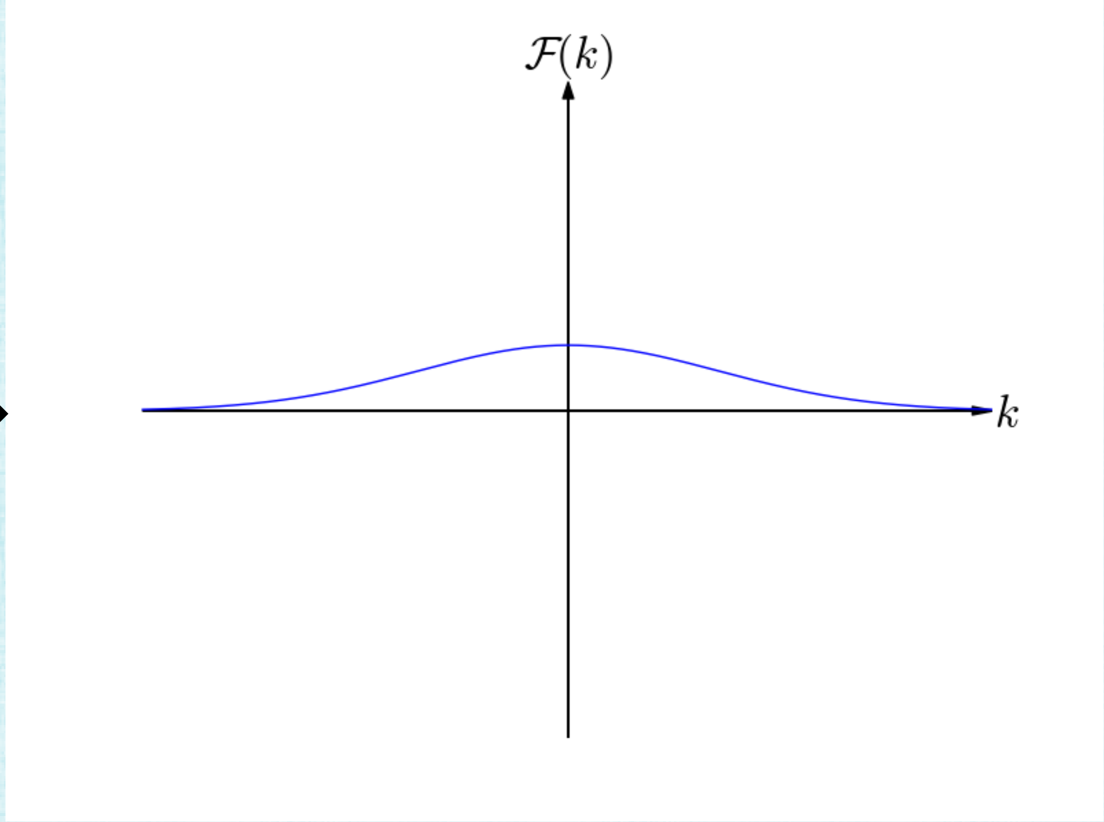


Wide

Wider Gaussian

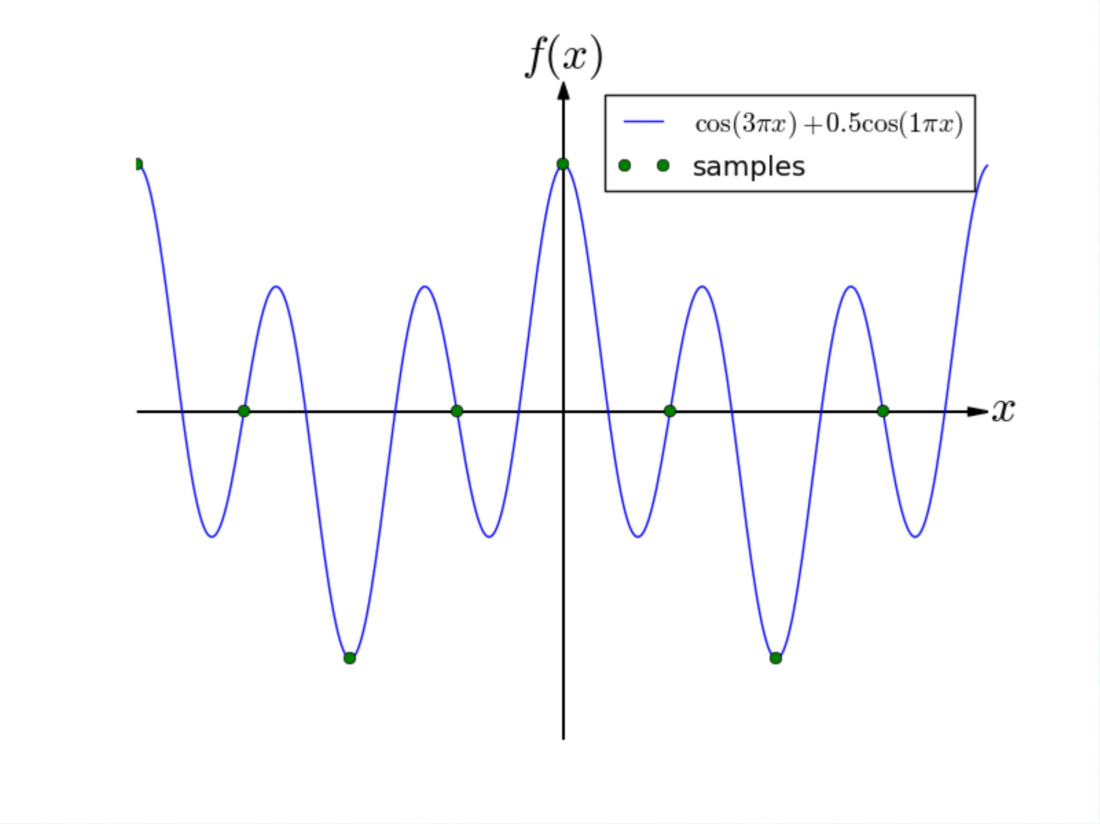


Wider

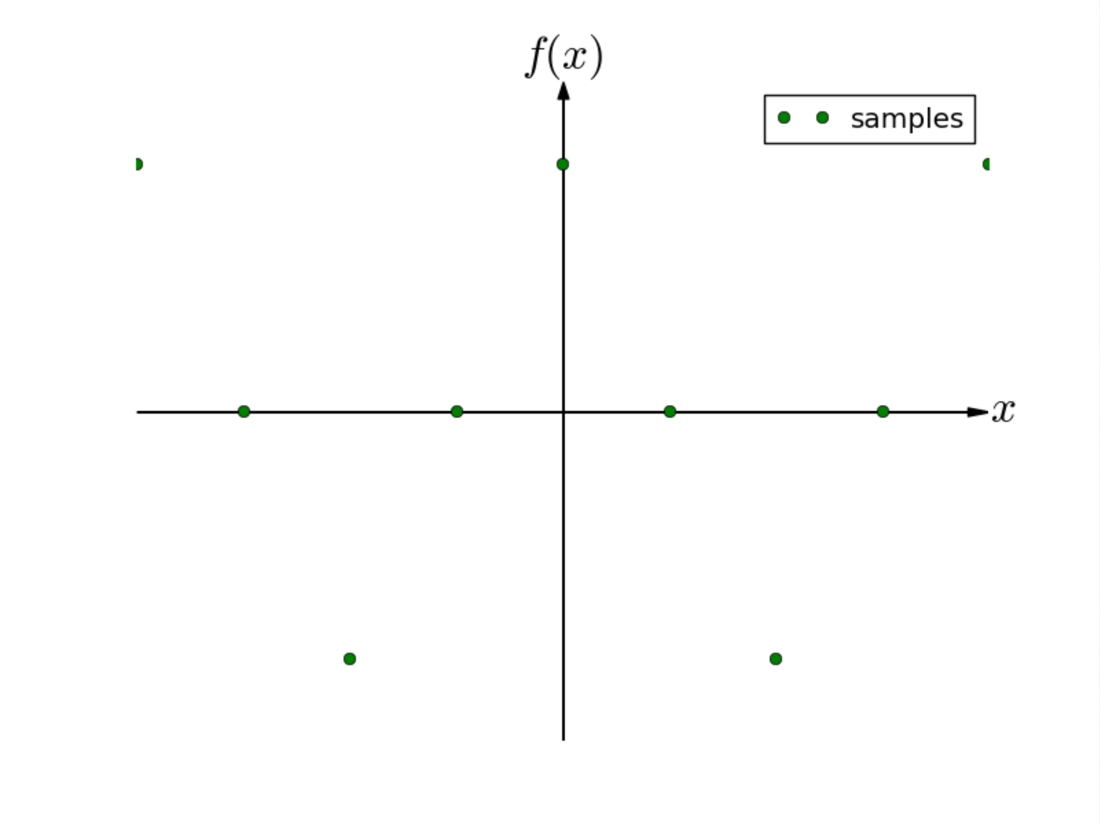


Narrower

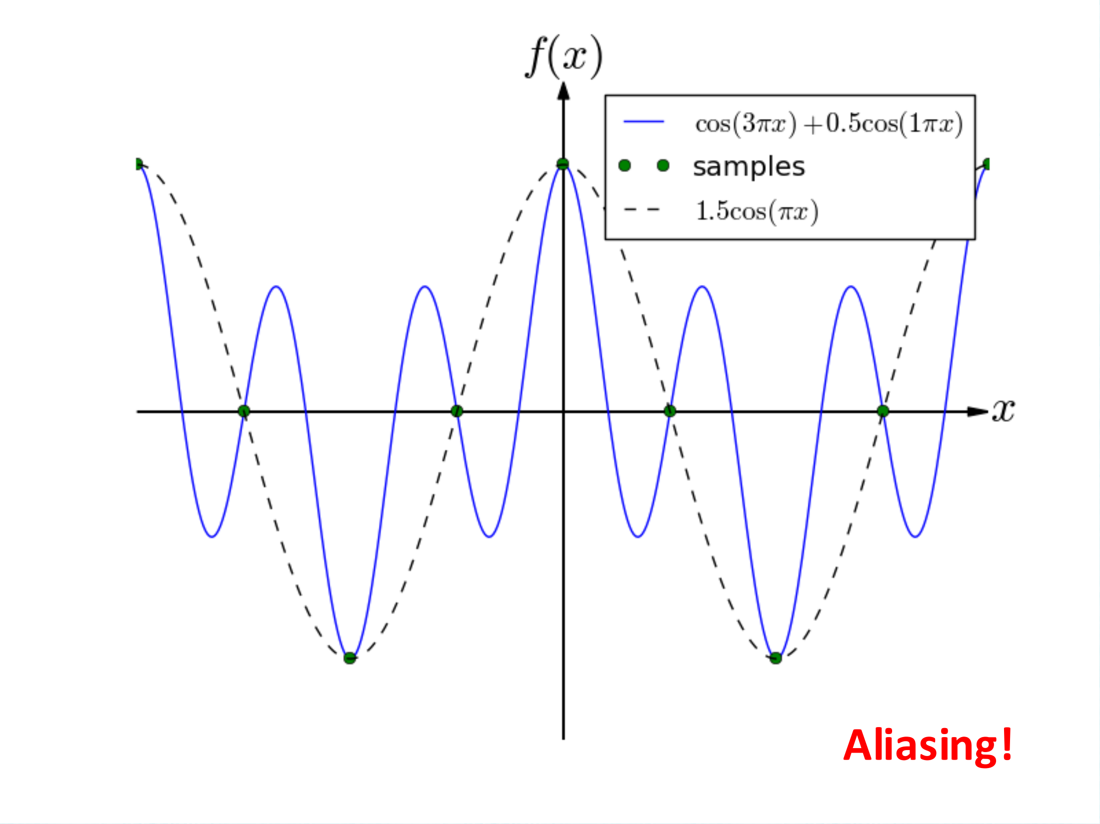
sum of two different cosine functions



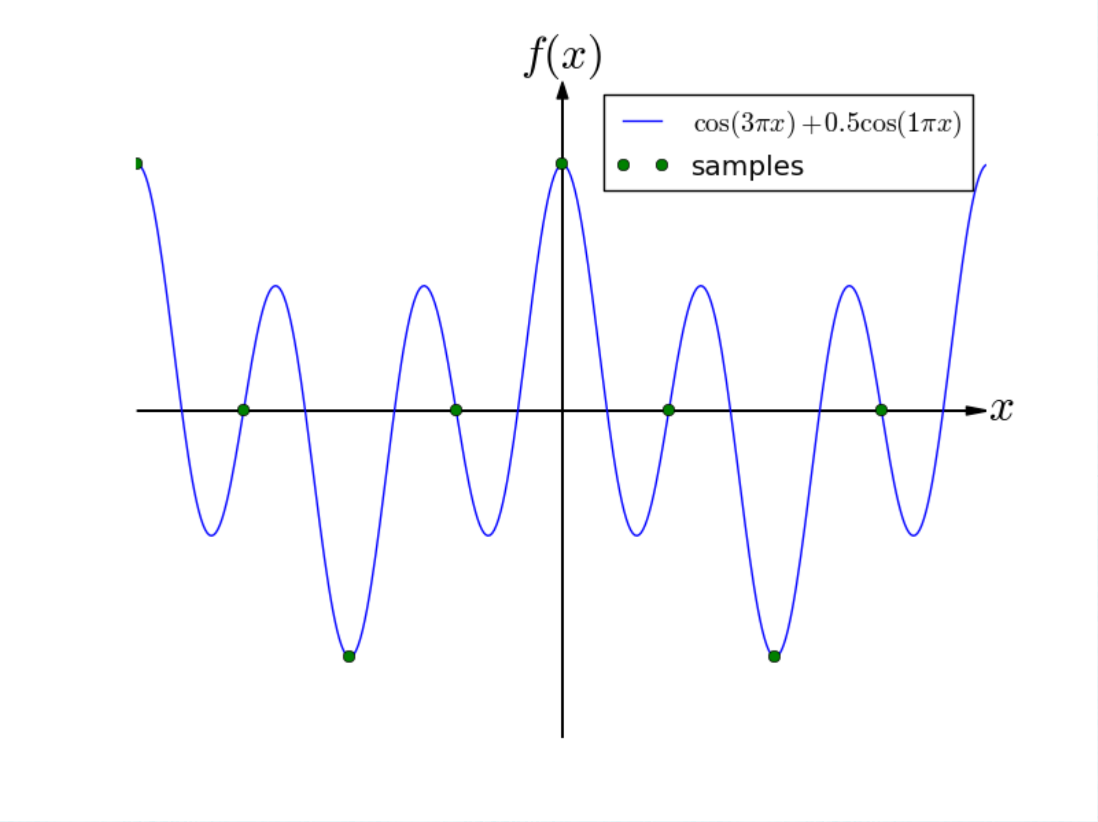
samples

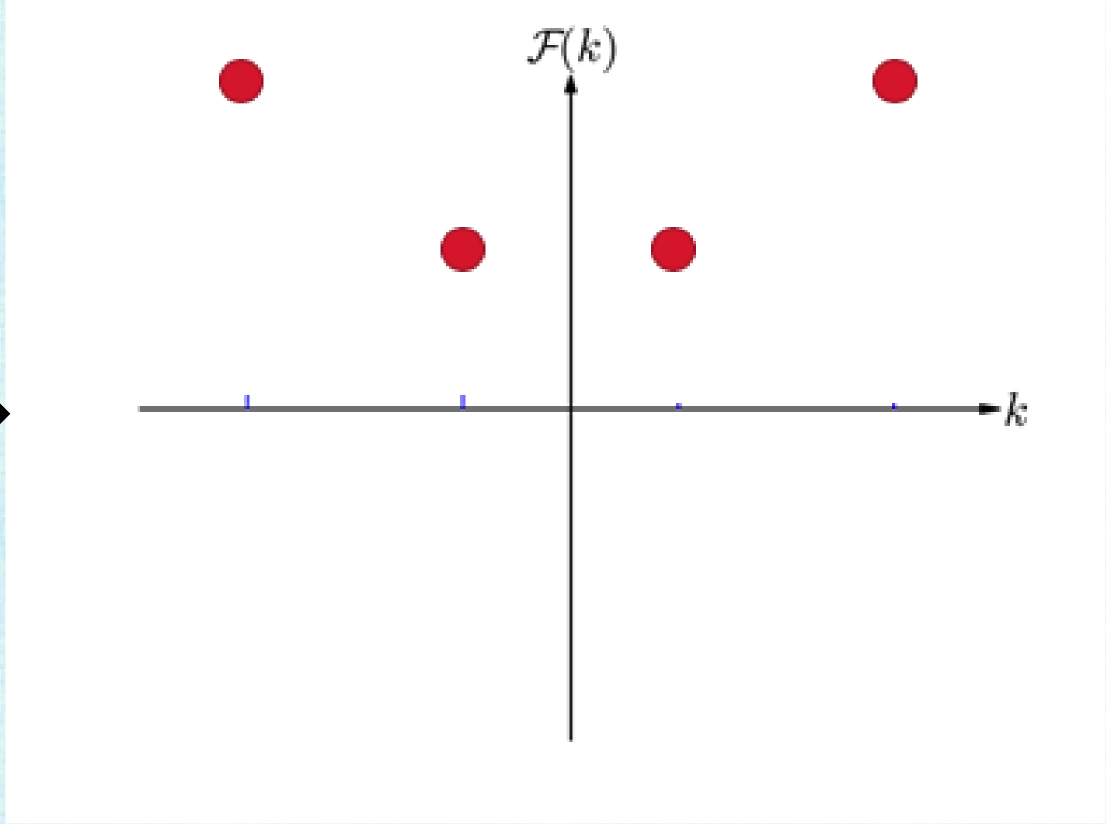


reconstruction

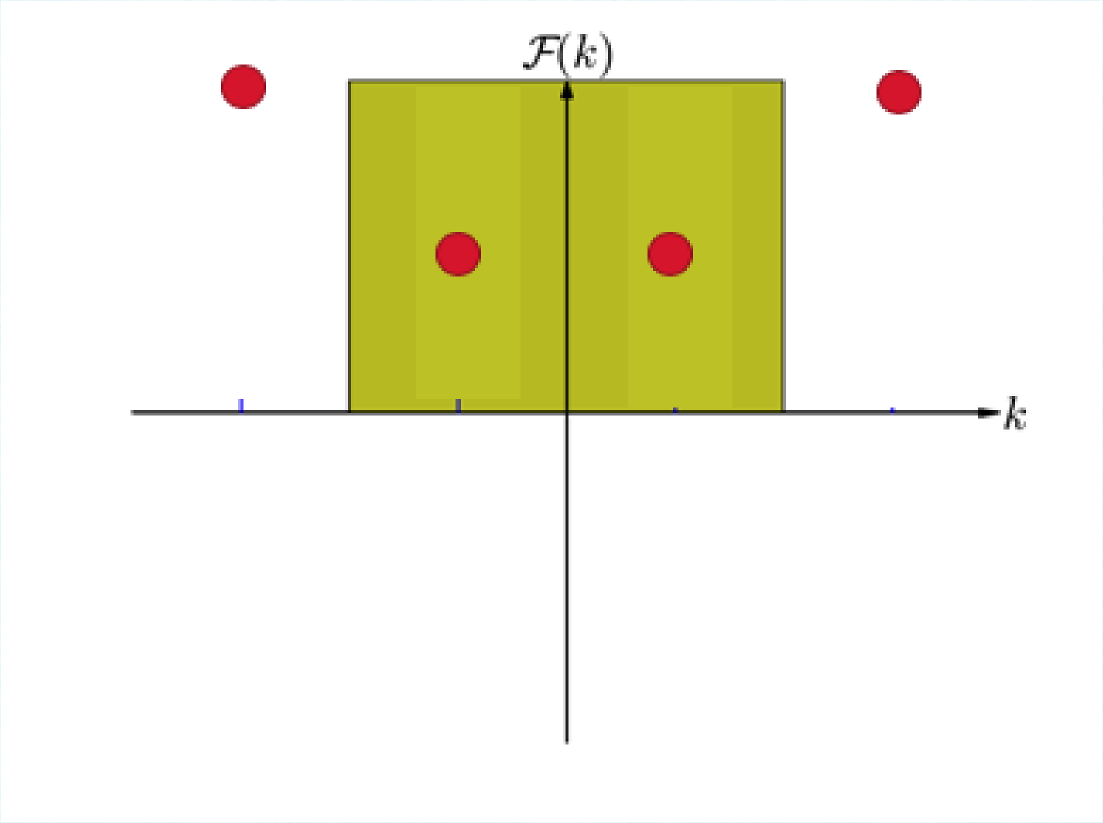


Fourier transform

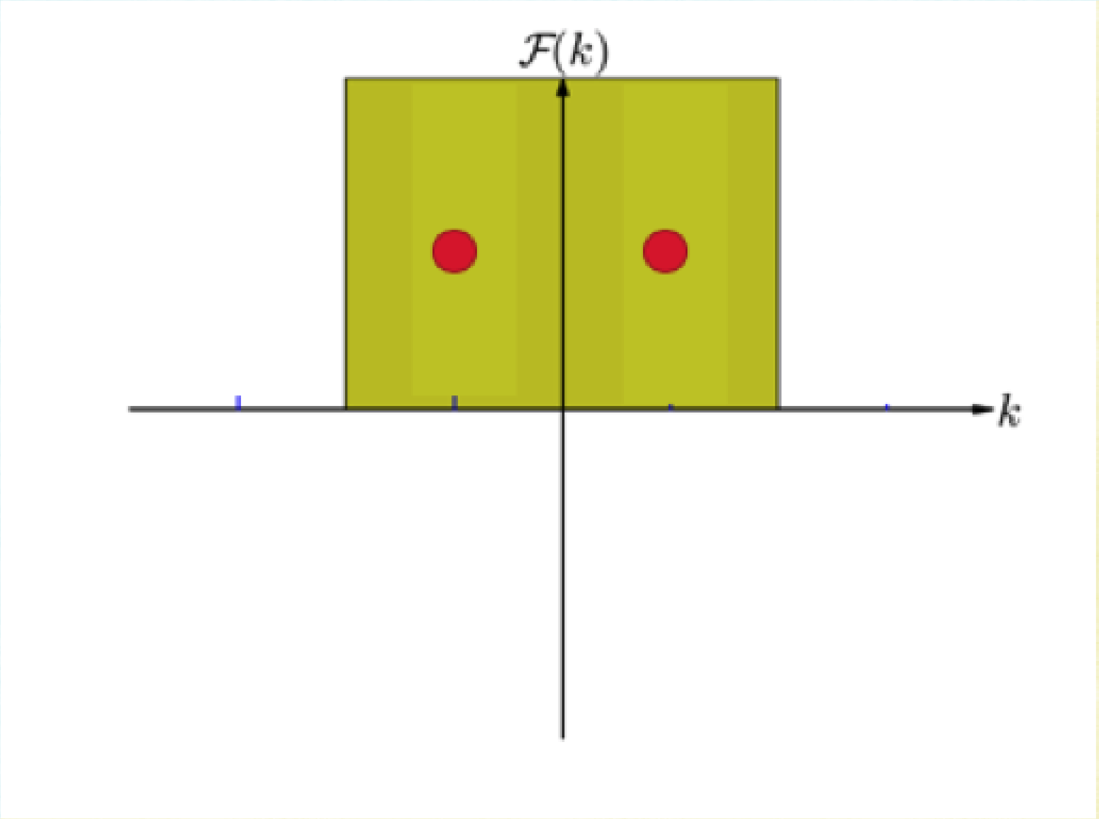




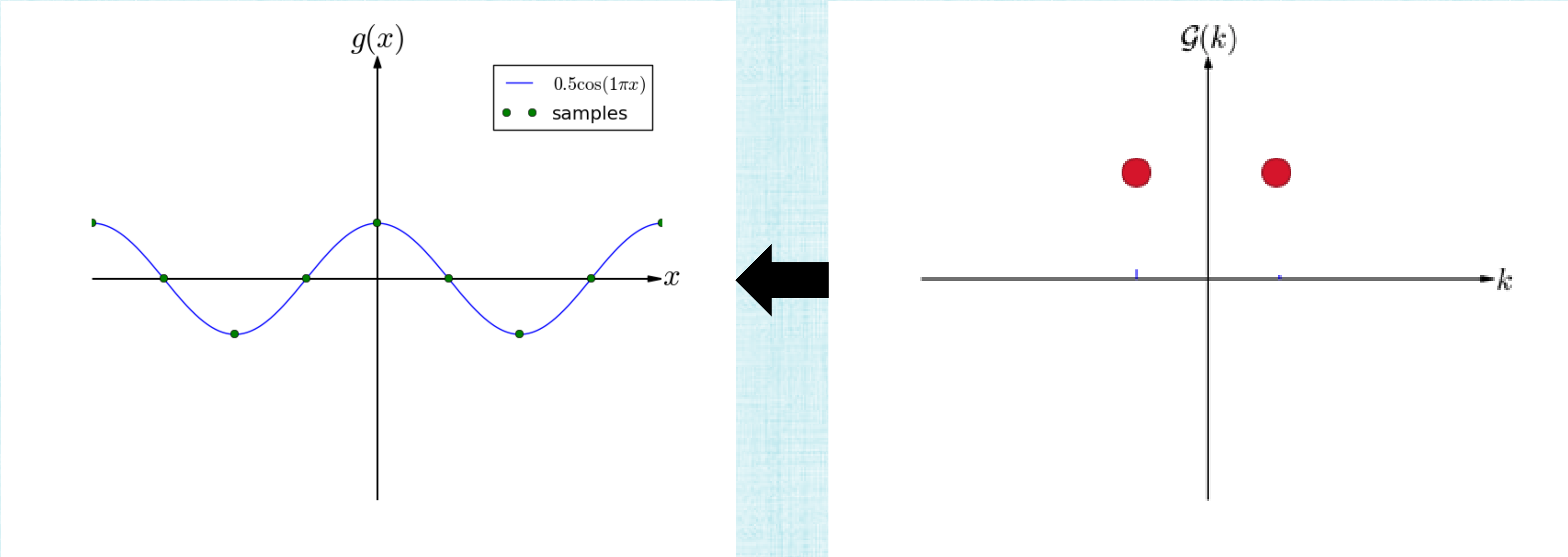
identify Nyquist frequency bounds



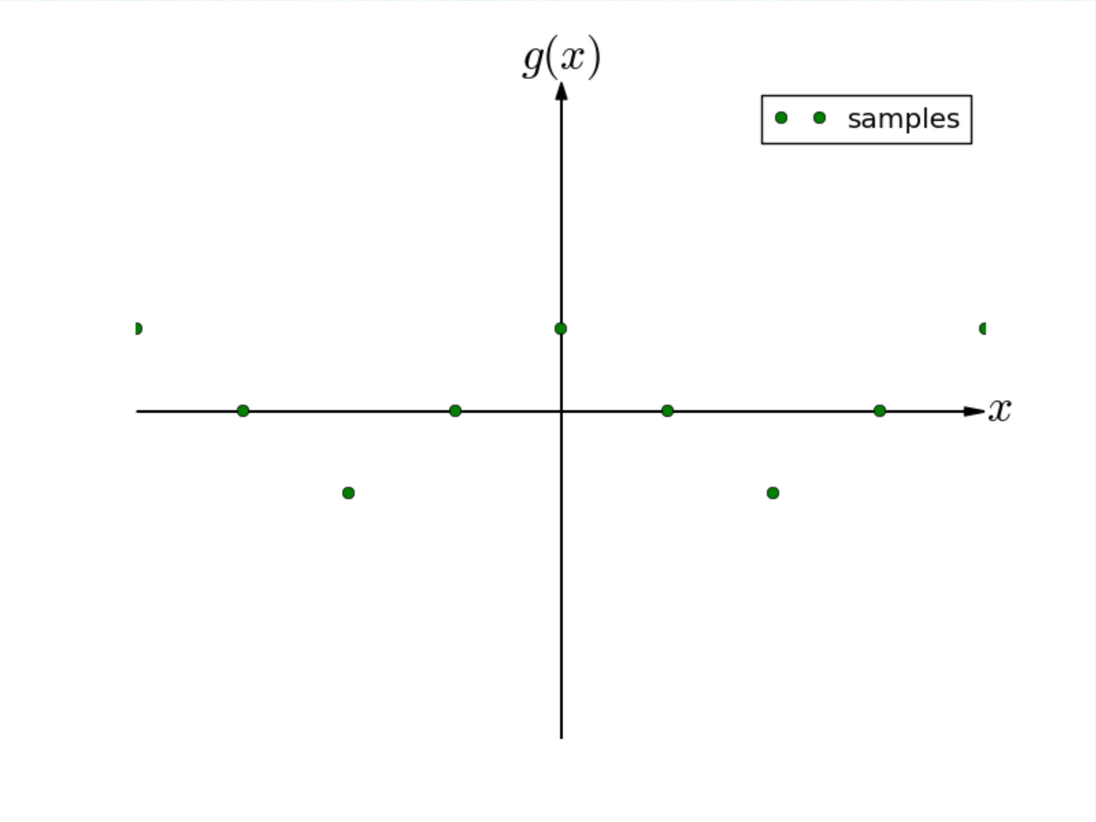
remove the high frequencies



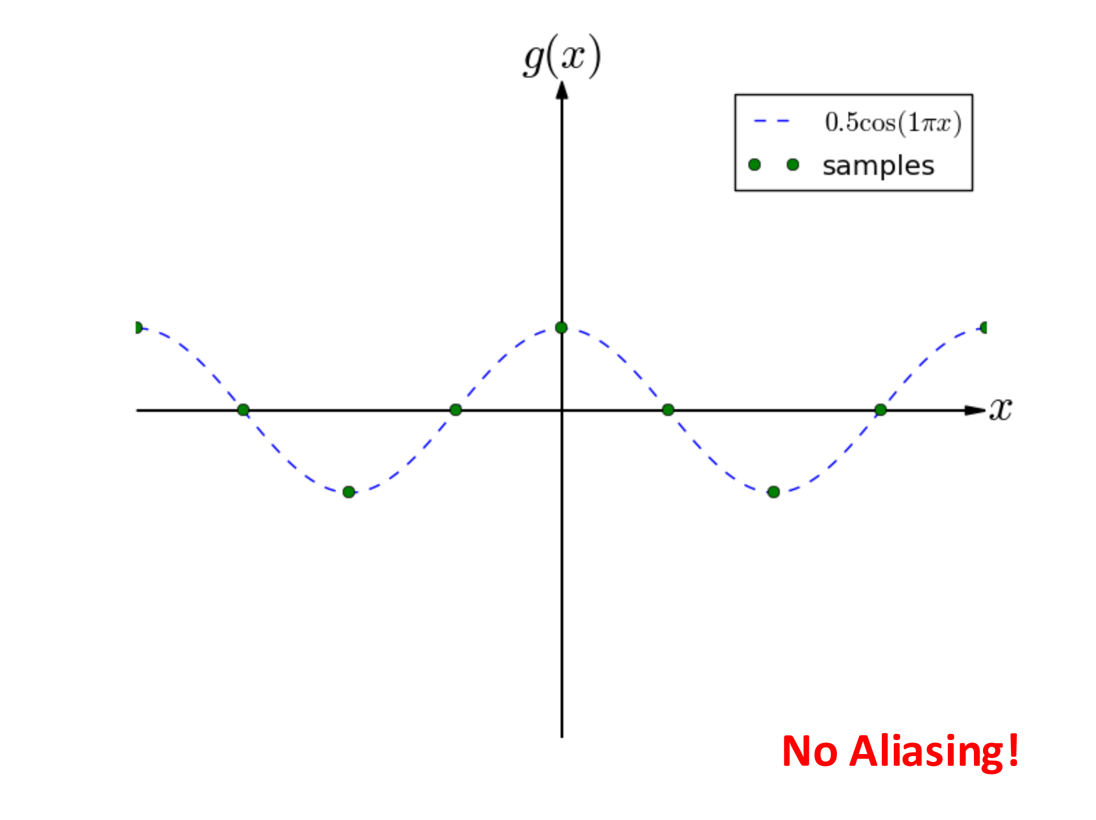
inverse Fourier transform



samples



reconstruction



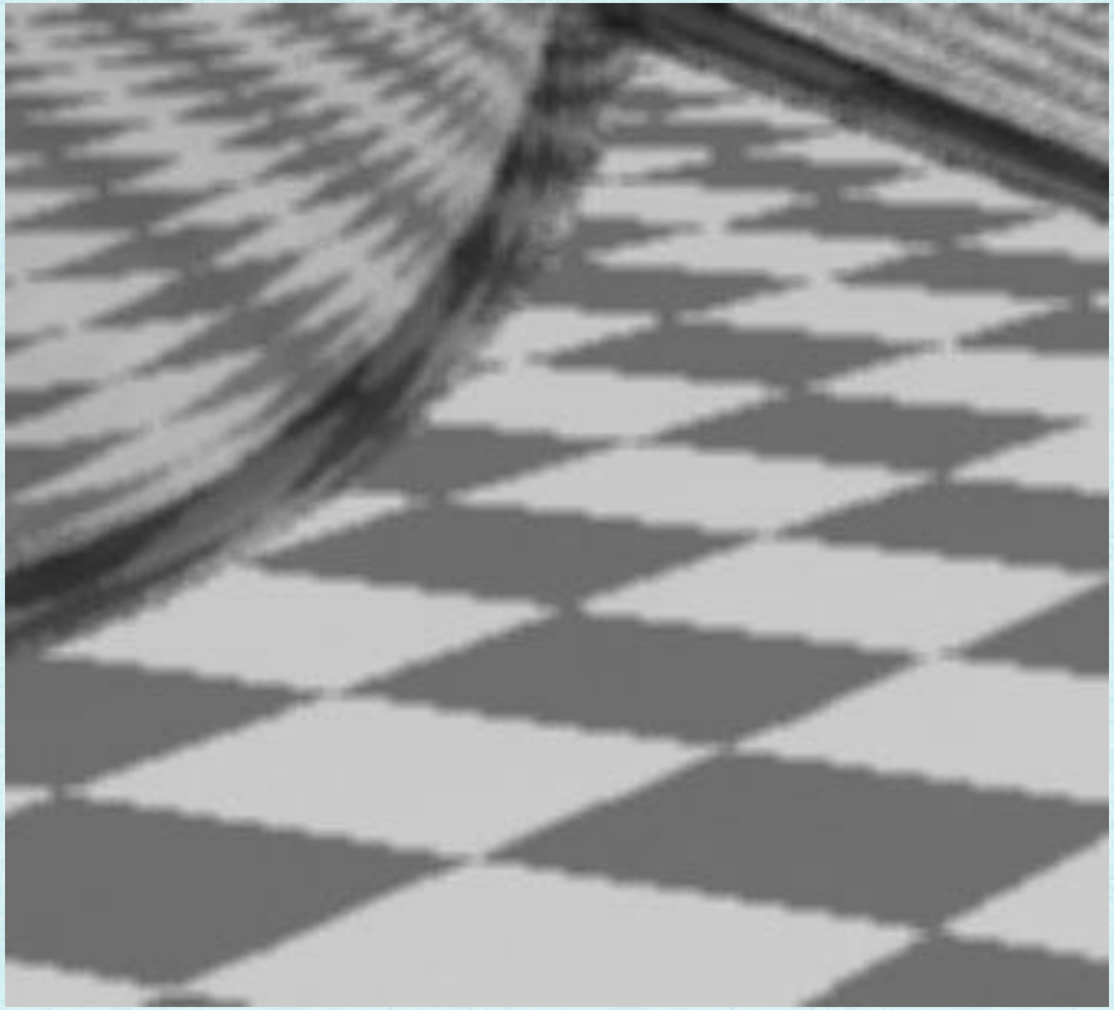
Anti-Aliasing

- Sampling causes higher frequencies to masquerade as lower frequencies
- After sampling, can no longer untangle the mixed high/low frequencies

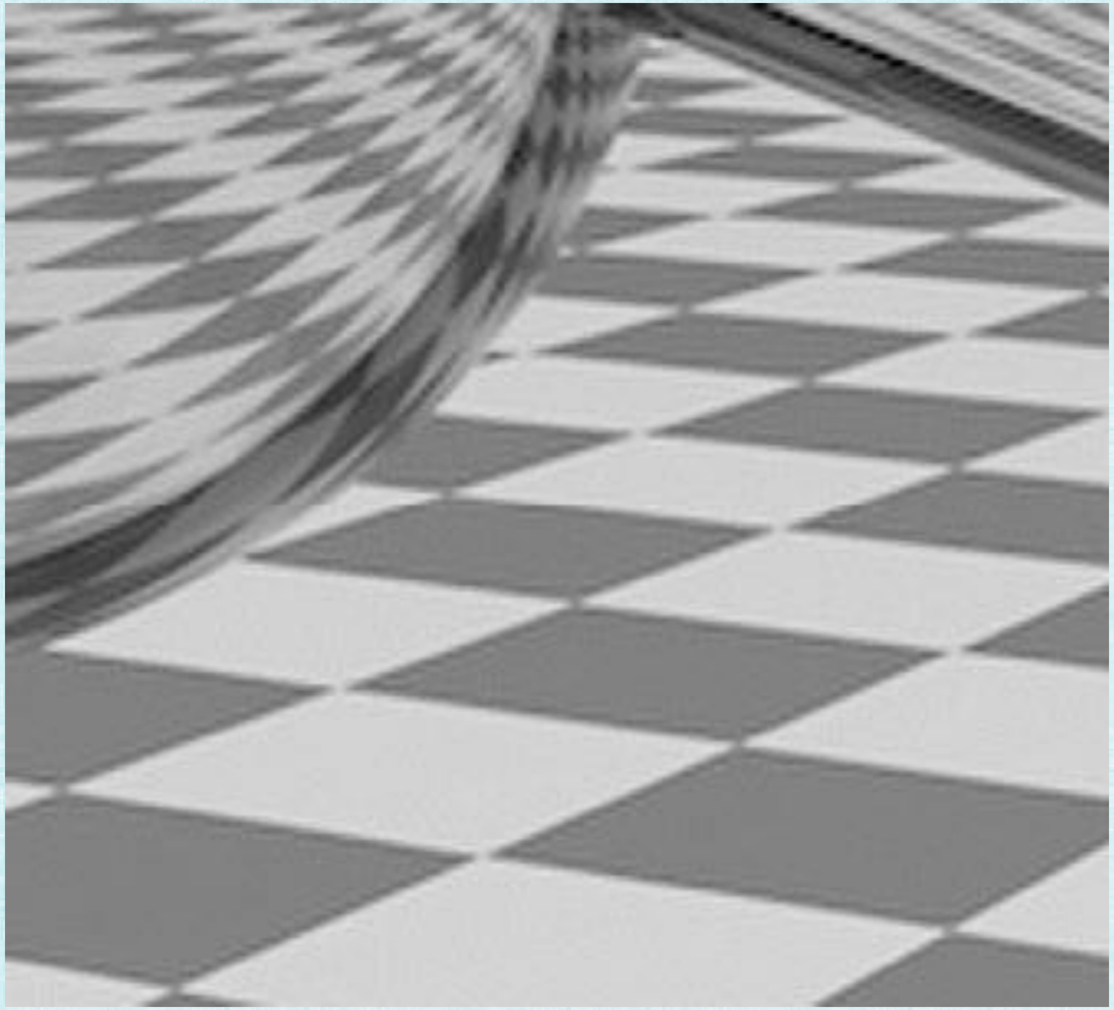
- Remove the high frequencies before sampling (in order to avoid aliasing)

- Part of the signal is lost
- But, that part of the signal was not representable by the sampling rate anyways

Blurring vs. Anti-Aliasing



blurring jaggies after sampling



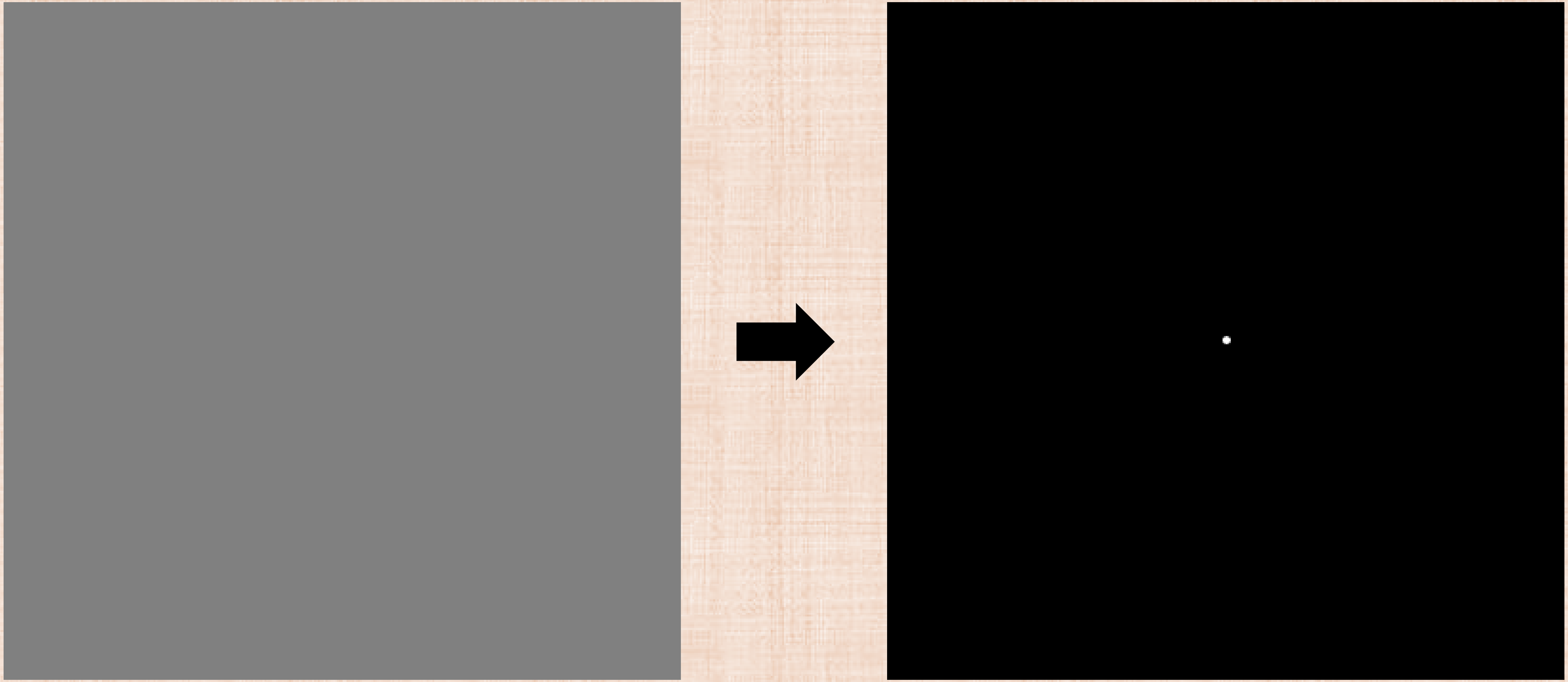
removing high frequencies before sampling

Images

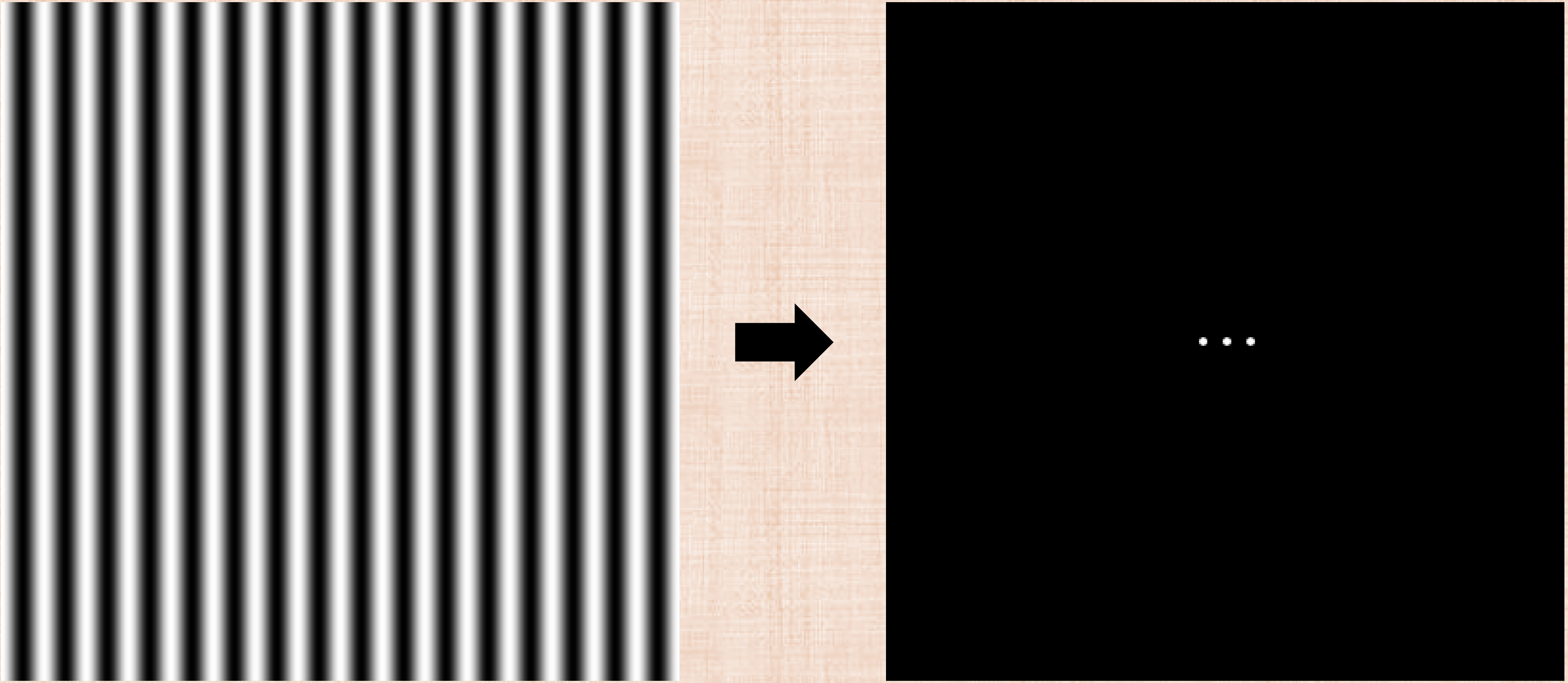
- Images have <u>discrete</u> values (and are not continuous functions)
  - Use a <u>discrete</u> version of the Fourier transform
  - The Fast Fourier Transform (FFT) computes the <u>discrete</u> Fourier transform (and its inverse) in $O(n \log n)$ complexity (where n is the number of samples)
- Images are <u>2D</u> (not 1D)
  - A <u>2D</u> discrete Fourier transform can computed using 1D transforms along each dimension

1. Transform into the frequency domain
   - Discrete image values are transformed into another array of discrete values
2. Remove high frequencies that would alias onto lower frequencies
3. Inverse transform back out of the frequency domain

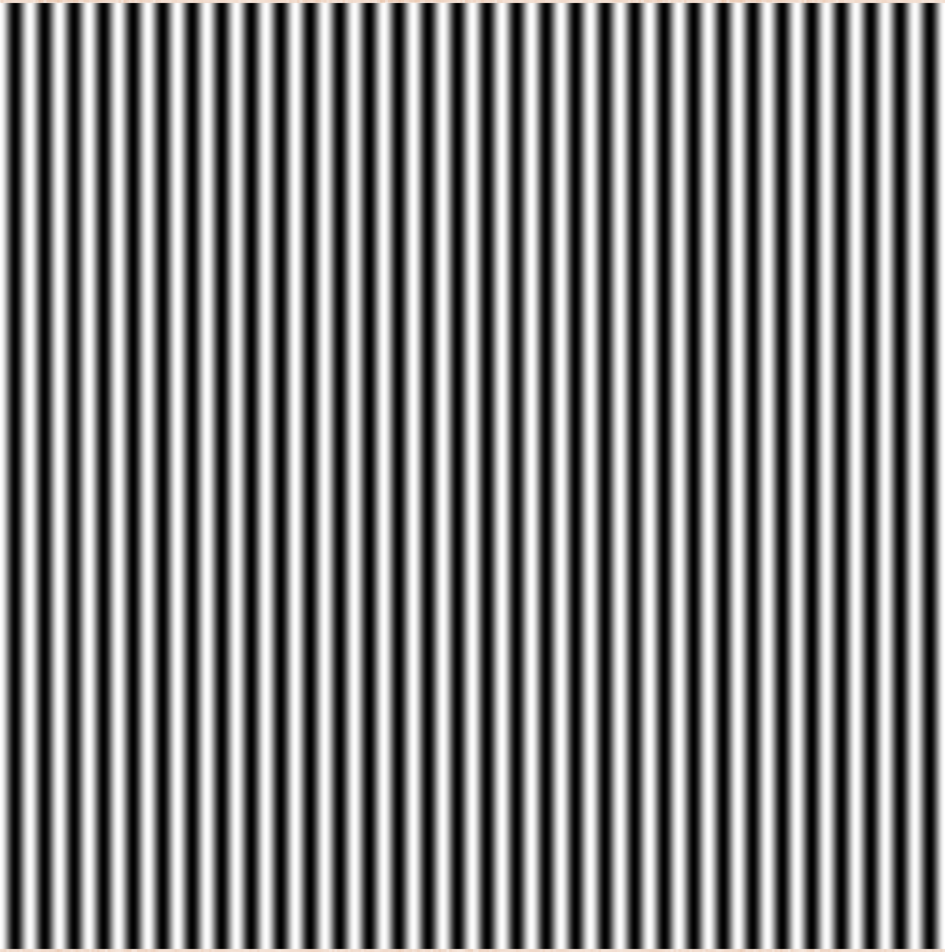
Constant Function

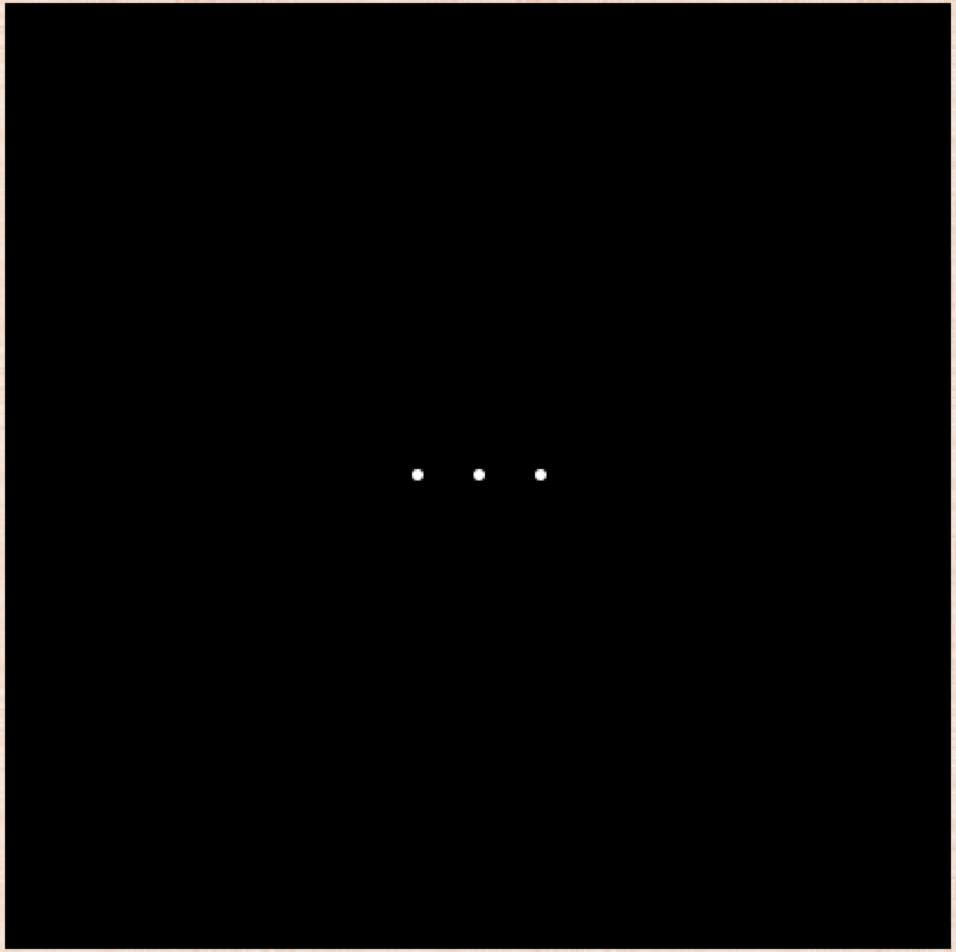


$$\sin(2\pi/32)\,x$$

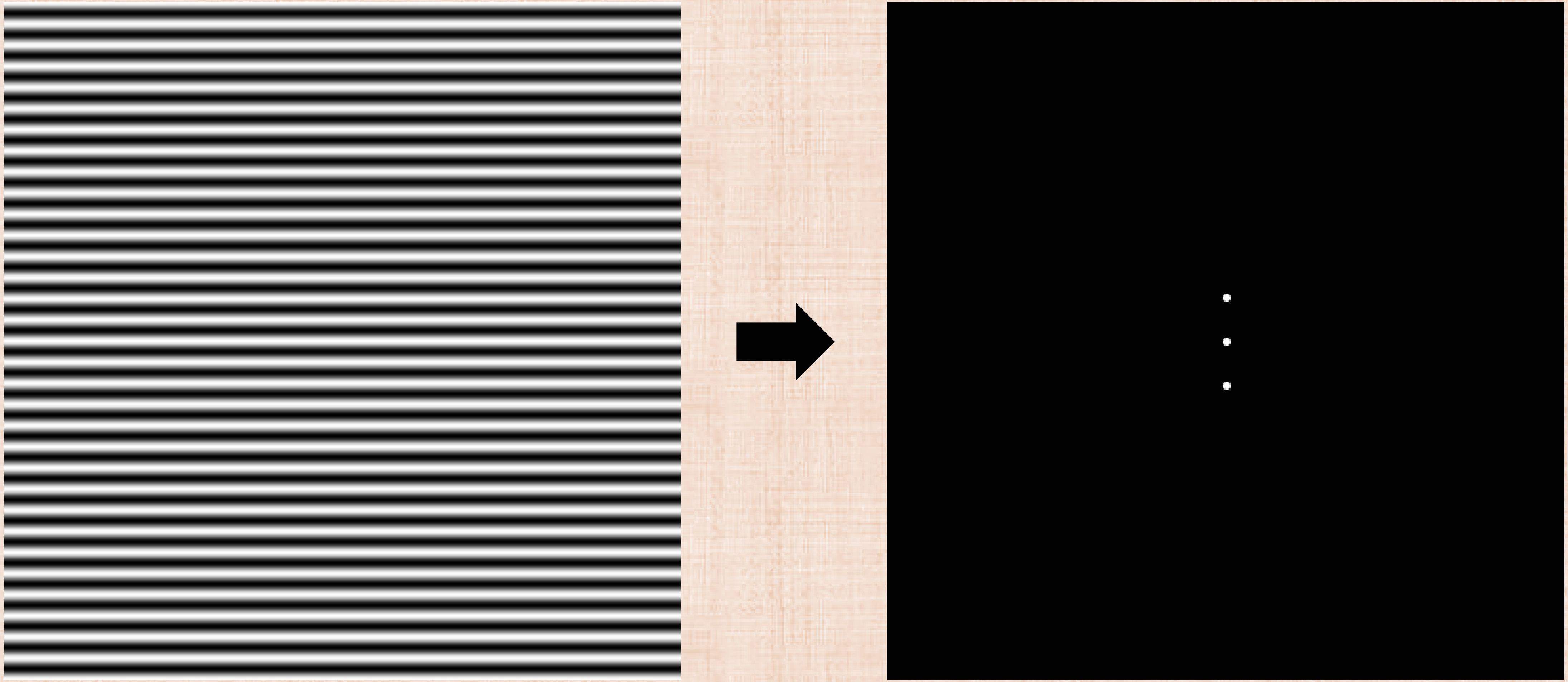


$$\sin(2\pi/16)\,x$$

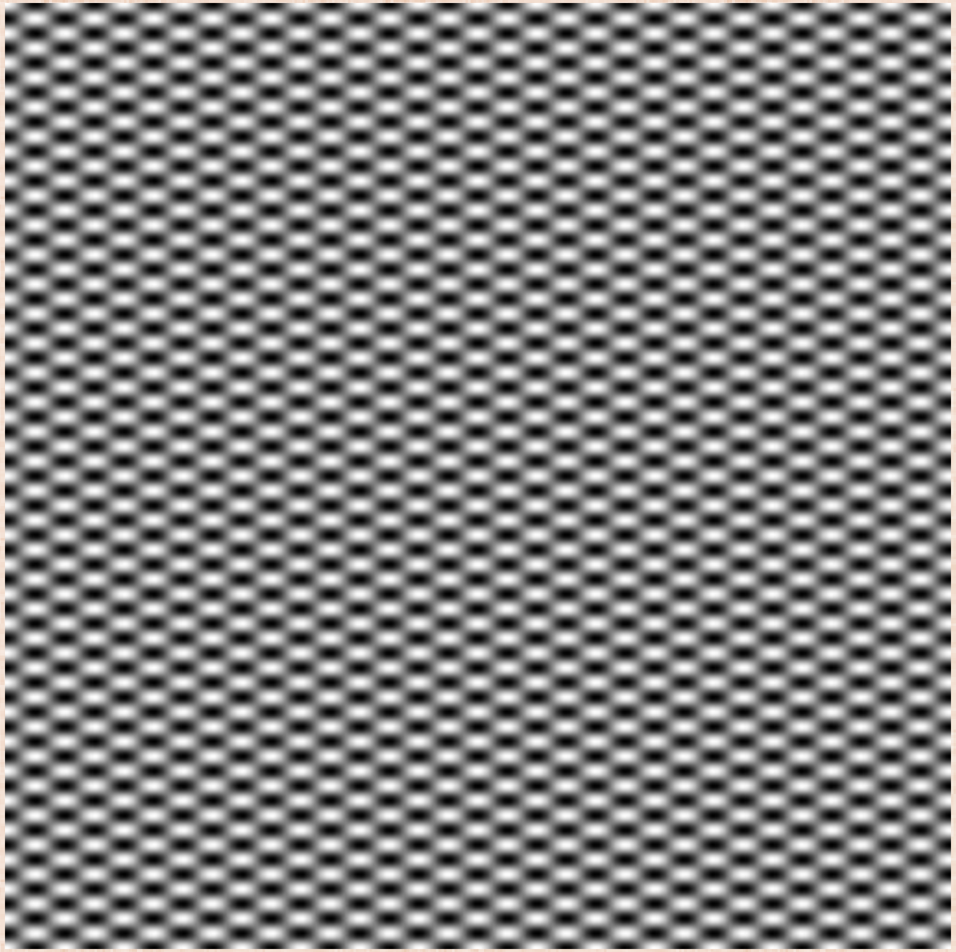


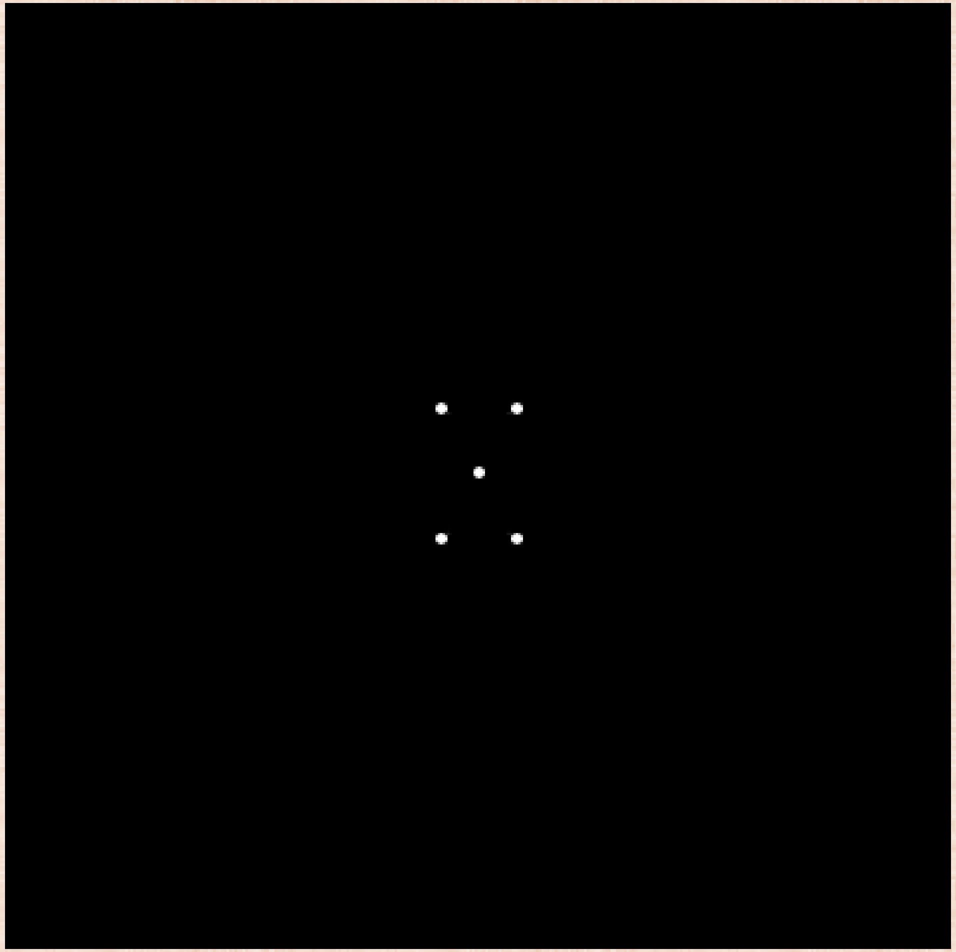


$\sin(2\pi/16)\,y$

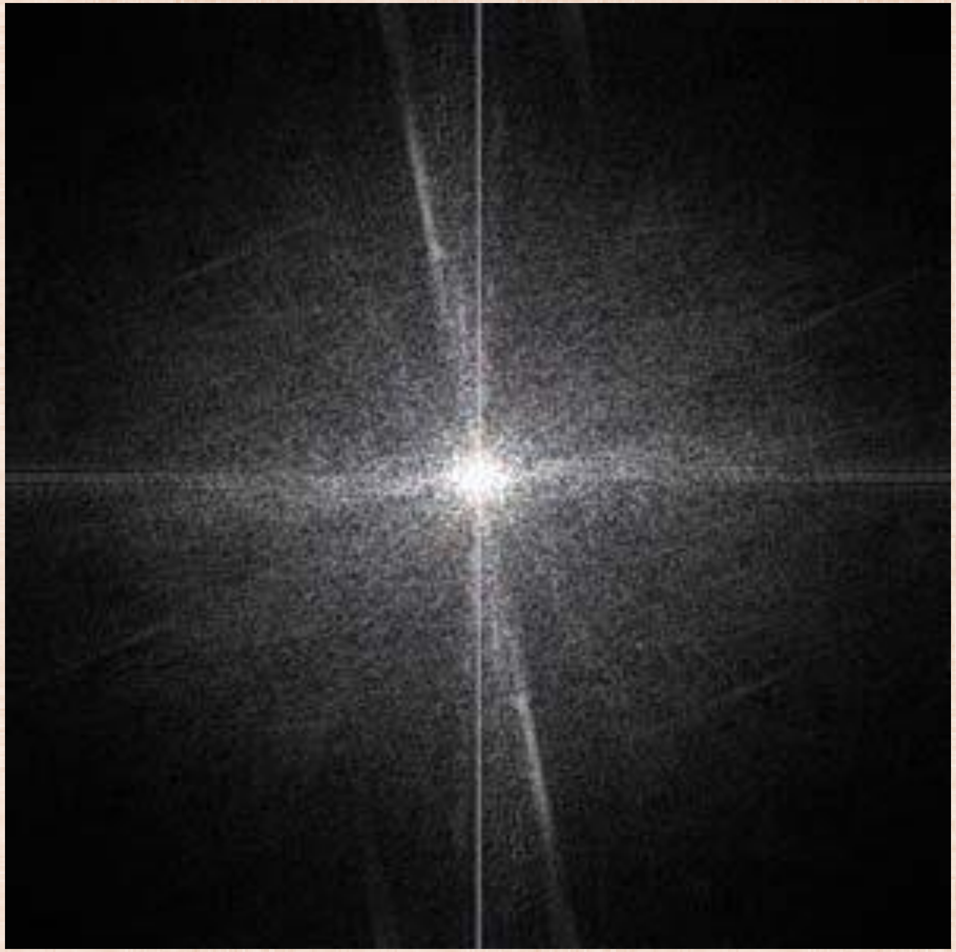


$\sin(2\pi/32)\,x \;*\; \sin(2\pi/16)\,y$

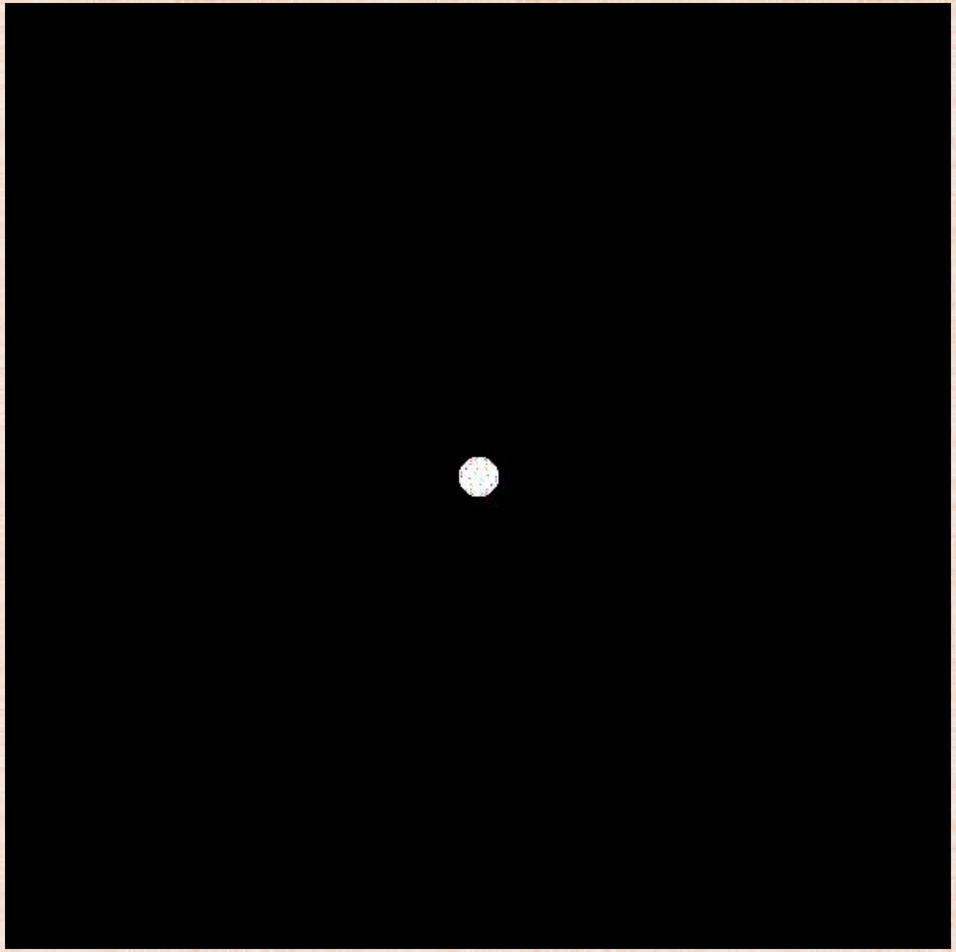




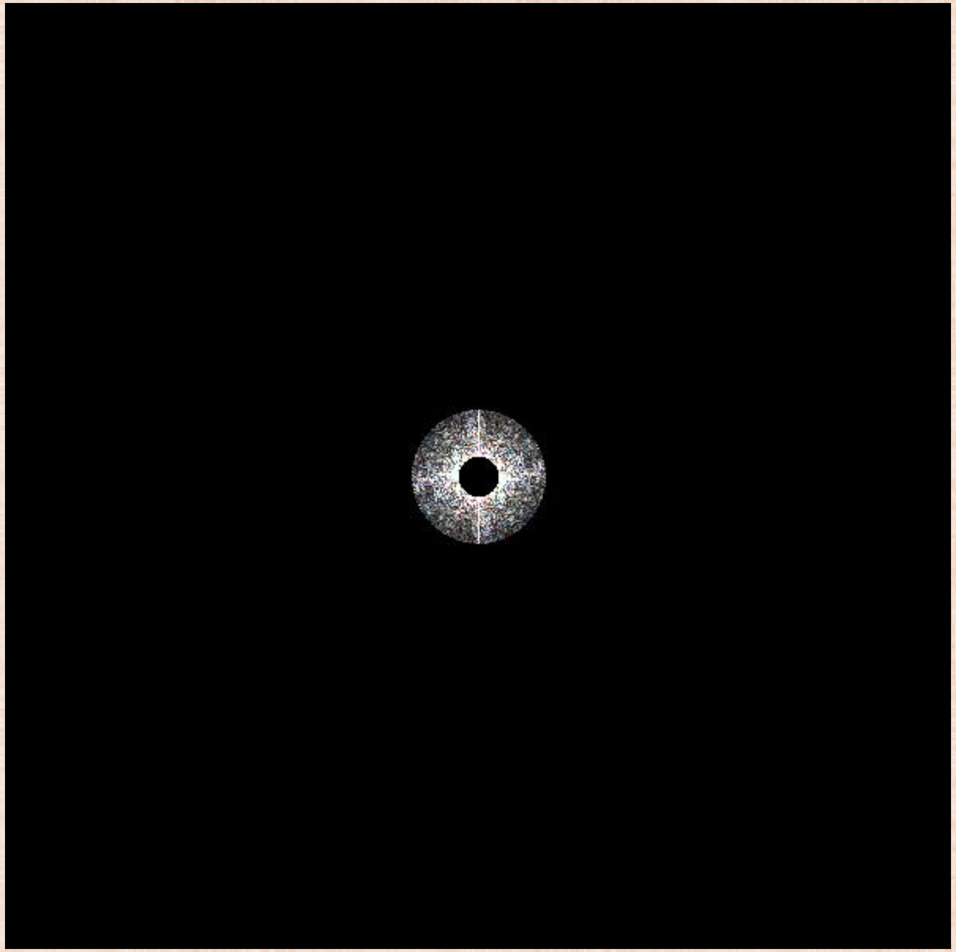
An obvious star!



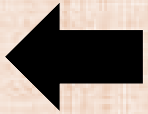
lowest frequencies

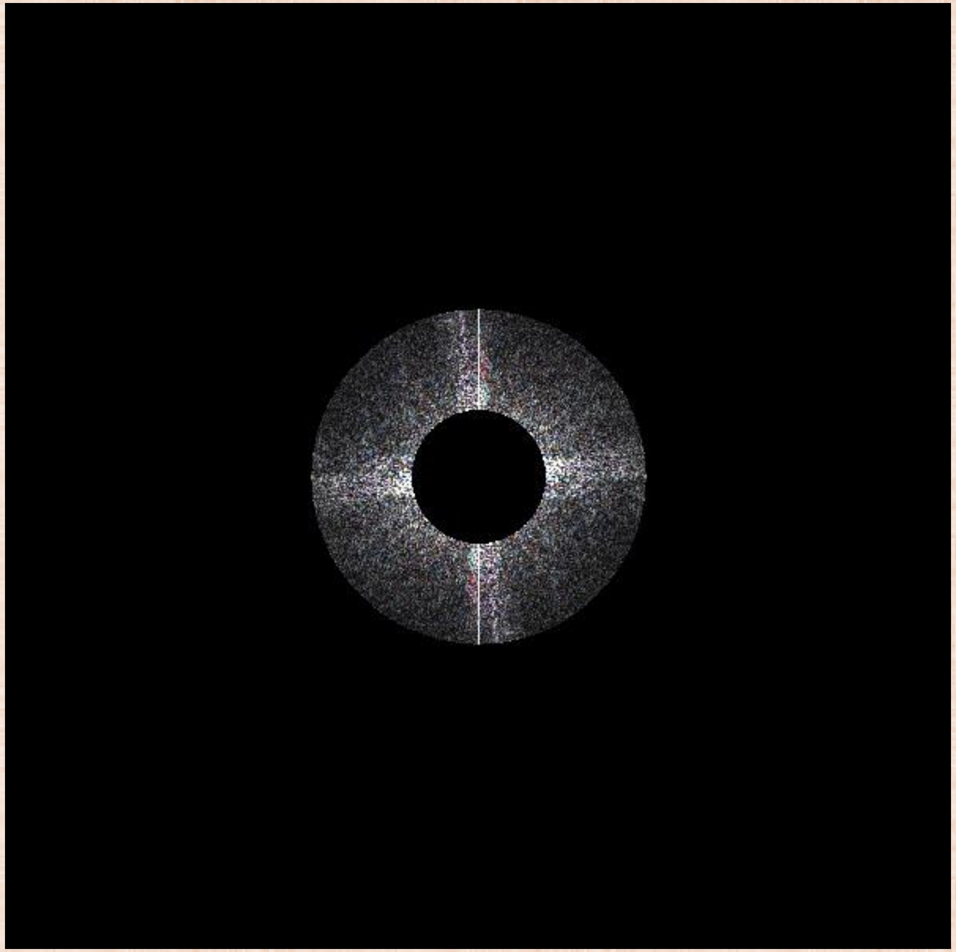


intermediate frequencies

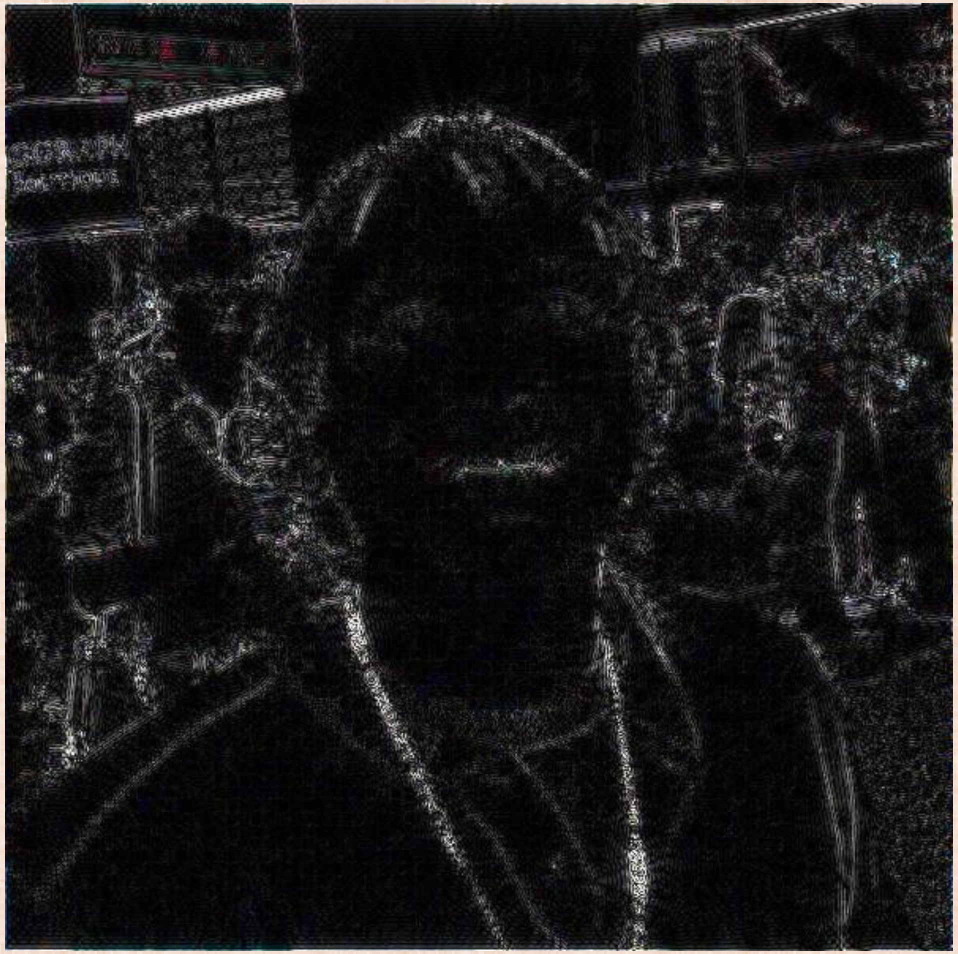


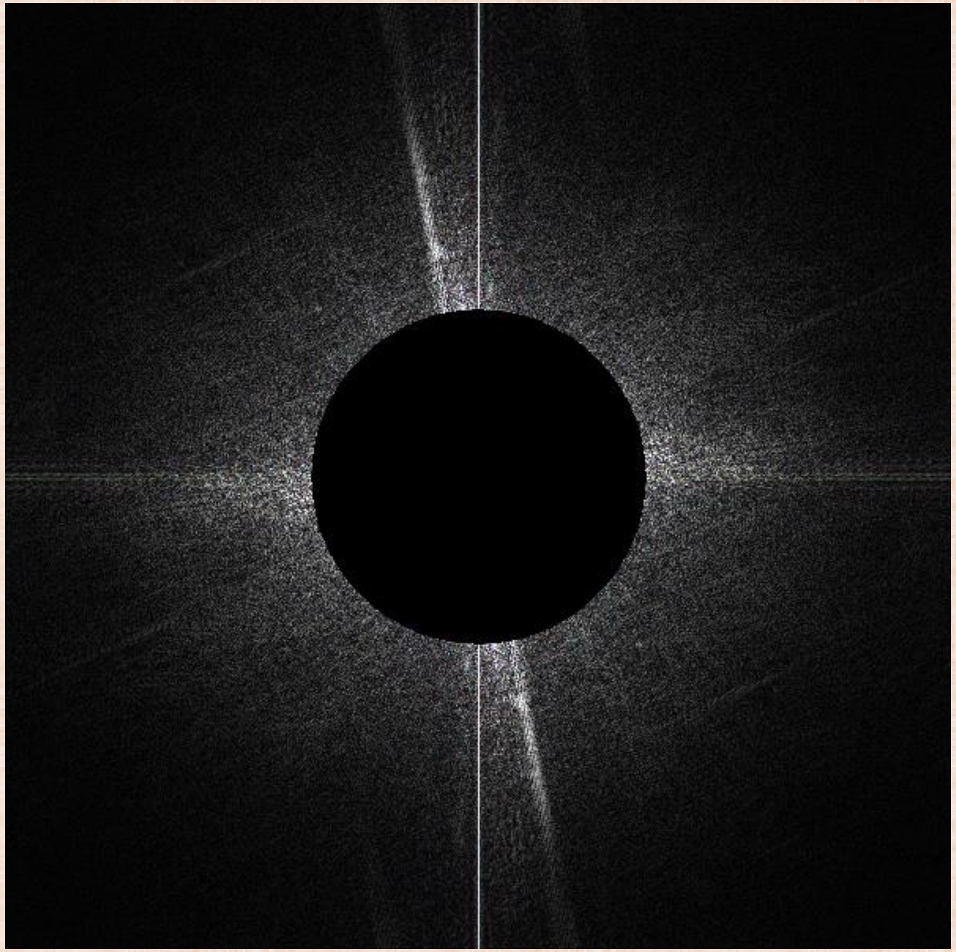
larger intermediate frequencies





highest frequencies (edges)





What about Ray Tracing?

- Unlike 1D functions and 2D images, there is no good way to put the 3D scene (made up of triangles) into the frequency domain
- If we sample the scene before removing the higher frequencies, those higher frequencies will alias onto lower frequencies
- So, we need a way to remove higher frequencies without transforming into the frequency domain
- That's called convolution

Convolution

- Let f and g be functions in the spatial domain
- Let $F(f)$ and $F(g)$ be transformations of f and g into the frequency domain
- In our prior examples: f was on the left and $F(f)$ was on the right

- Removing higher frequencies of $F(f)$ is equivalent to multiplying by a Heaviside function $F(g)$
  - $F(g) = 1$ for smaller frequencies, and $F(g) = 0$ for larger frequencies
- Then, the inverse transform $F^{-1}(F(f)F(g))$ gave the final result

- Thus, the convolution of f and g is defined via:

$$f * g = F^{-1}(F(f)F(g))$$

Convolution Integral

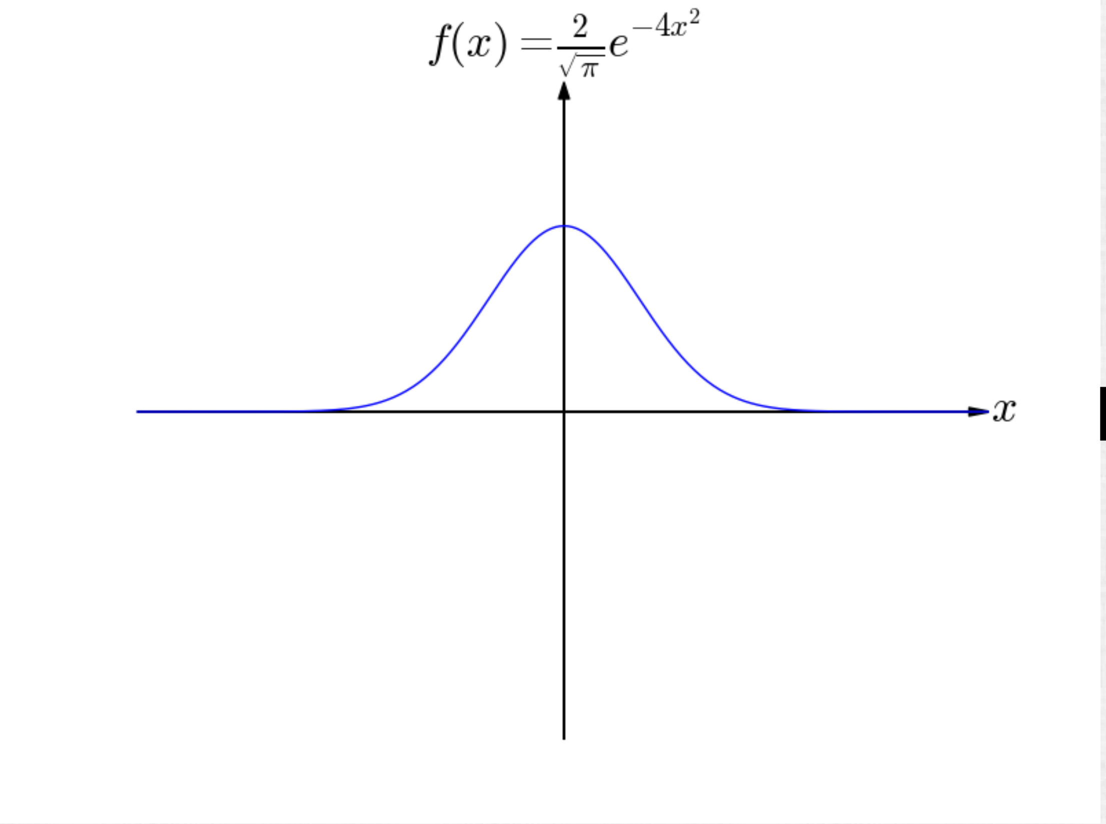
- Convolution can be achieved without the Fourier Transform:

$$(f * g)(t) = \int_{-\infty}^{\infty} f(\tau)g(t-\tau)d\tau = \int_{-\infty}^{\infty} f(t-\tau)g(\tau)d\tau$$

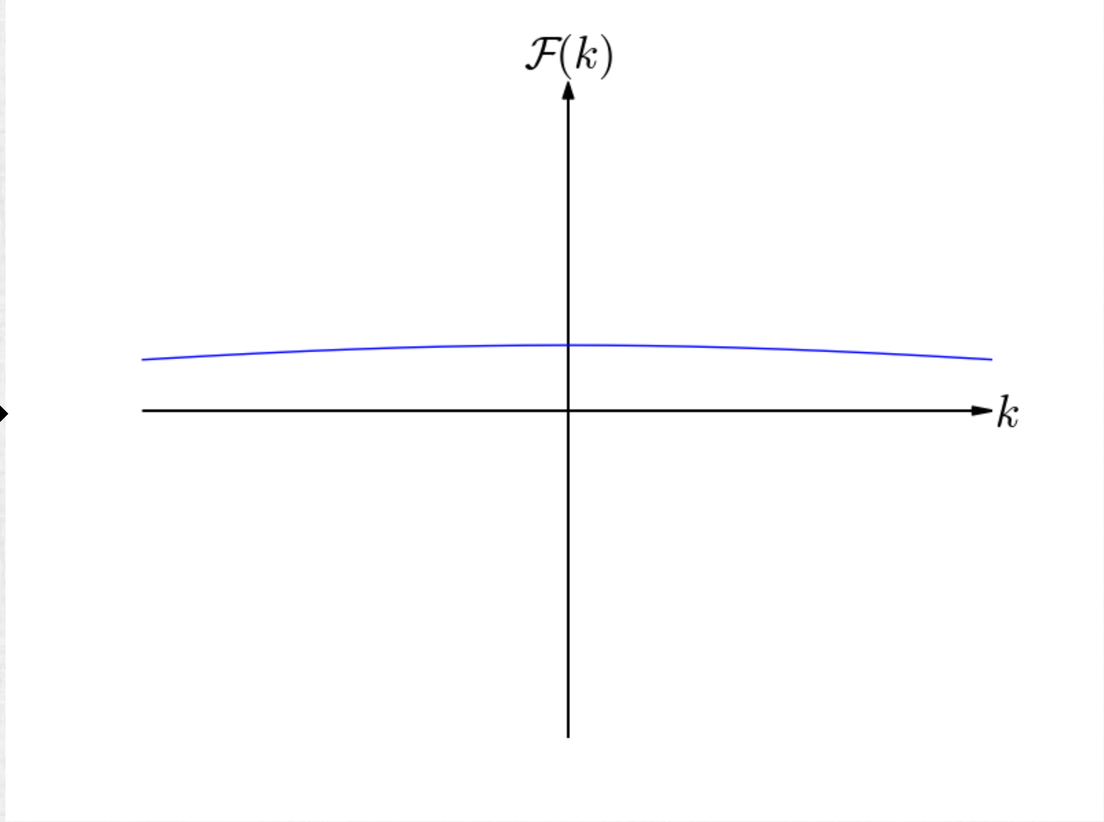
- A narrower g makes the integral more efficient to compute
- A narrower $F(g)$ better removes high frequencies (as we have seen)

- But, they can’t both be narrow
  - Recall: the narrower Gaussian had wider frequencies, and the wider Gaussian had narrower frequencies

Recall: Narrow Gaussian

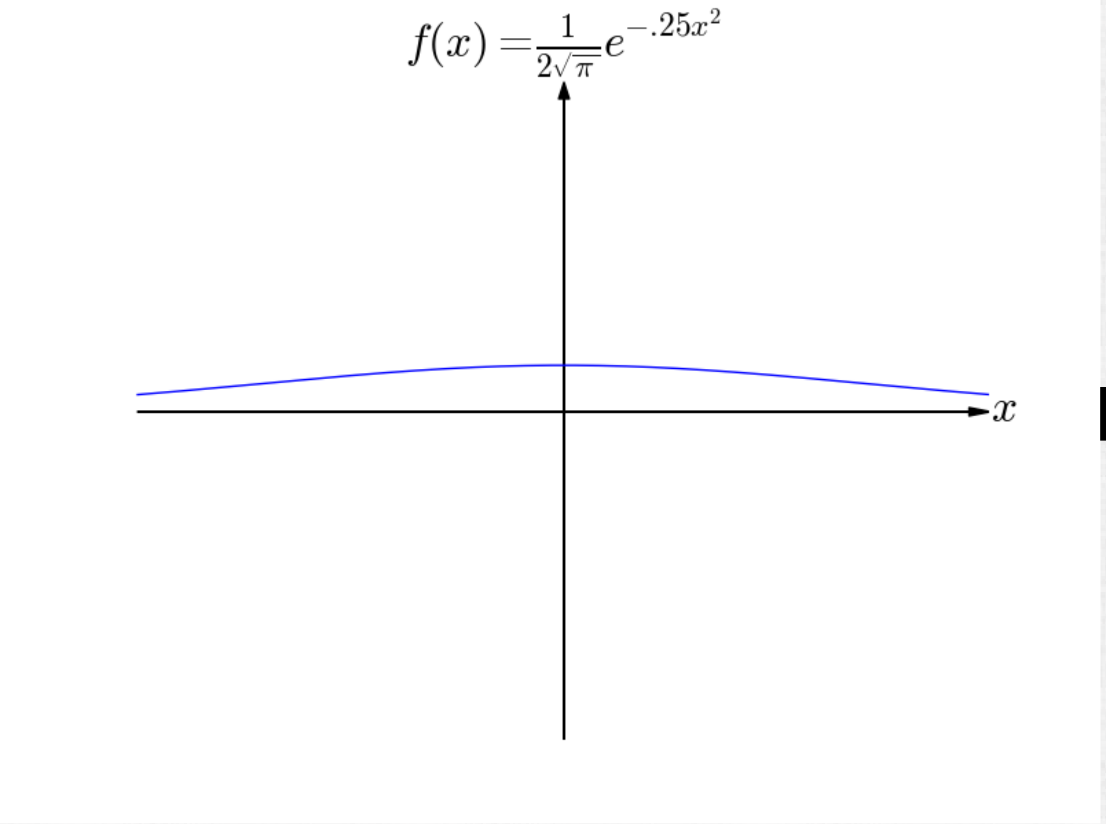


Narrow

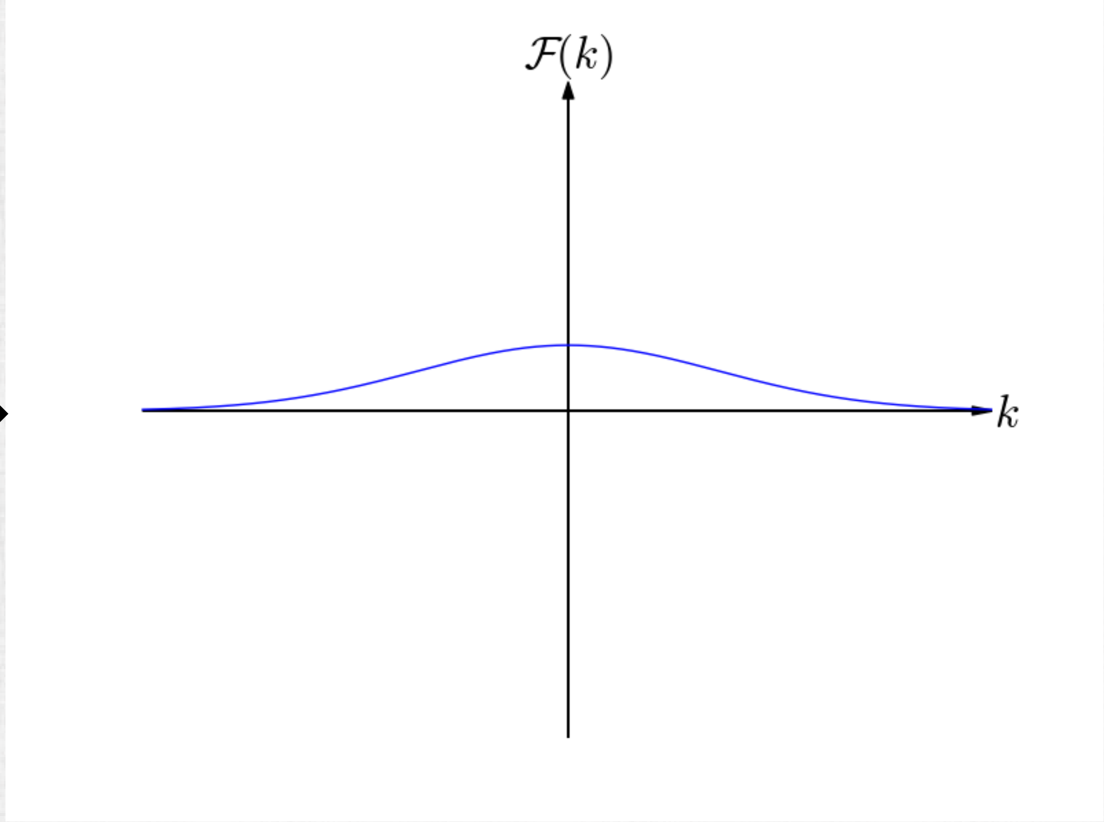


Wide

Recall: Wider Gaussian



Wider



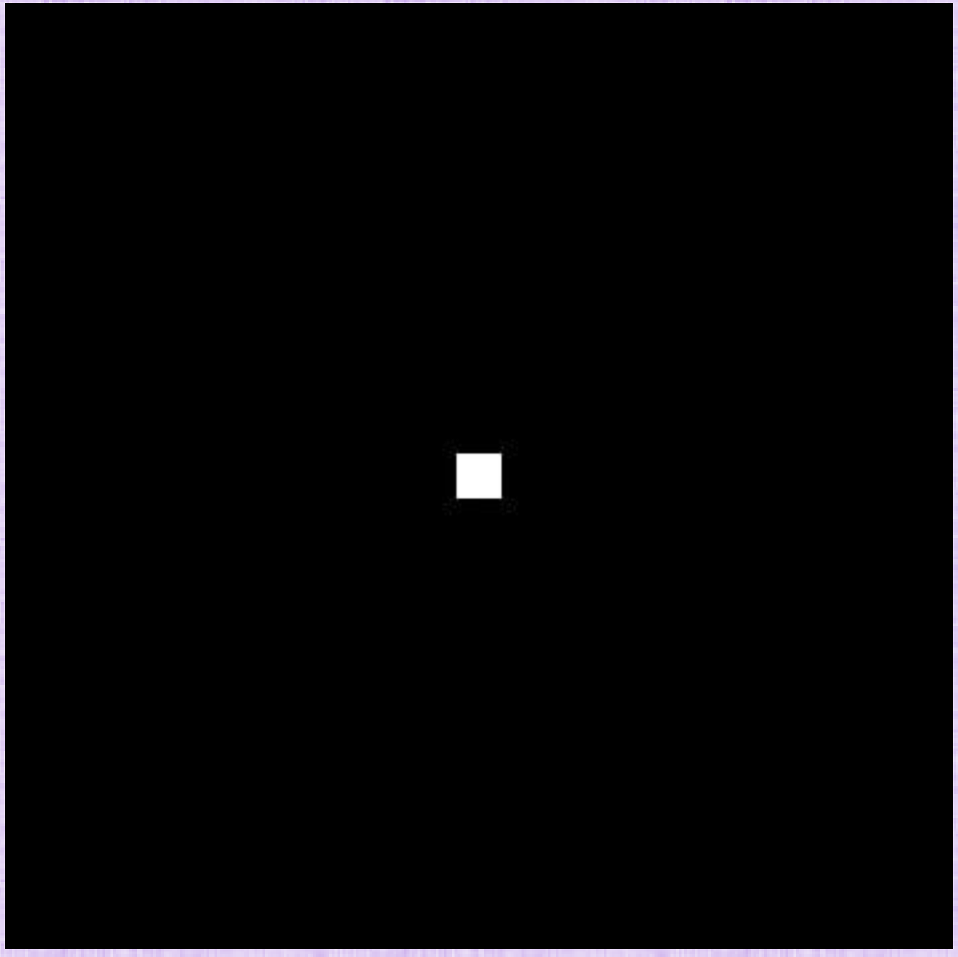
Narrower

Box Filter

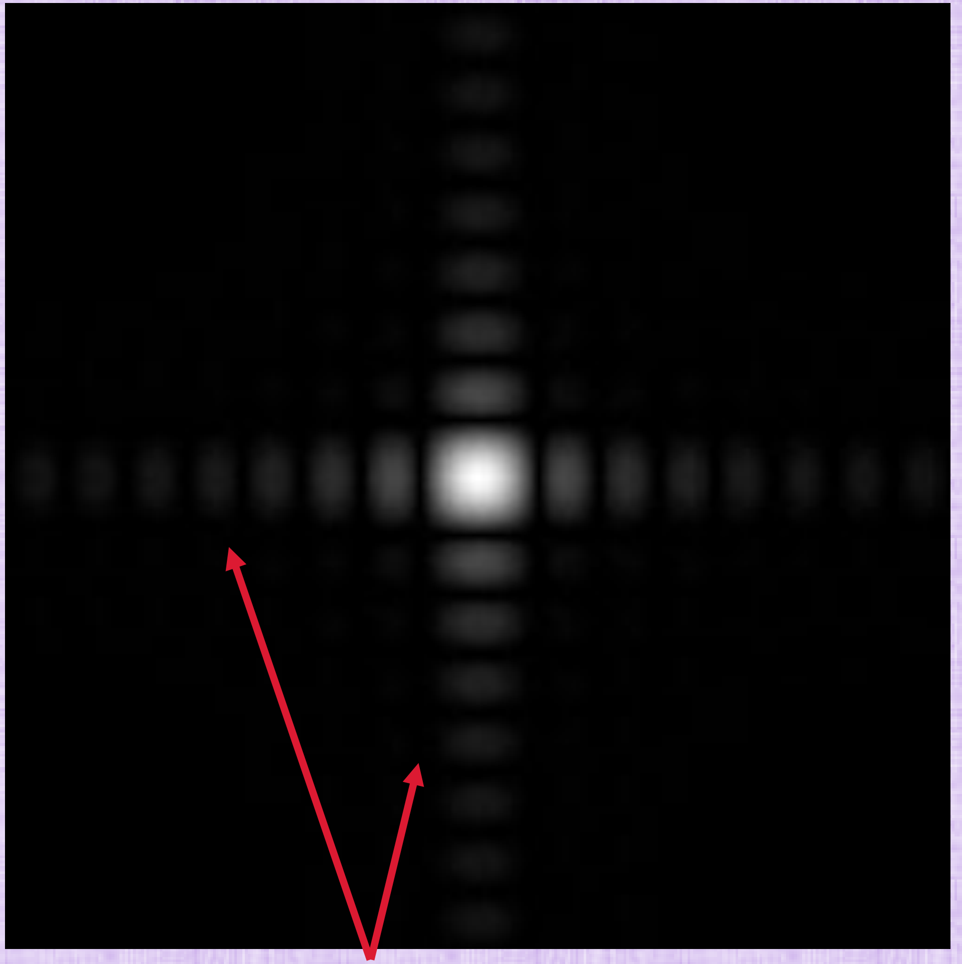
- Let g have nonzero values in an NxN block of pixels (surrounding the origin), and be zero elsewhere
- The discrete convolution integral can be computed via:
  - overlay the filter g on the image; then, multiply the corresponding entries, and sum the results
- The final result is (typically) defined to be at the center of the filter

$^1/_{16}$	$^1/_{16}$	$^1/_{16}$	$^1/_{16}$
$^1/_{16}$	$^1/_{16}$	$^1/_{16}$	$^1/_{16}$
$^1/_{16}$	$^1/_{16}$	$^1/_{16}$	$^1/_{16}$
$^1/_{16}$	$^1/_{16}$	$^1/_{16}$	$^1/_{16}$

Narrow Box Filter



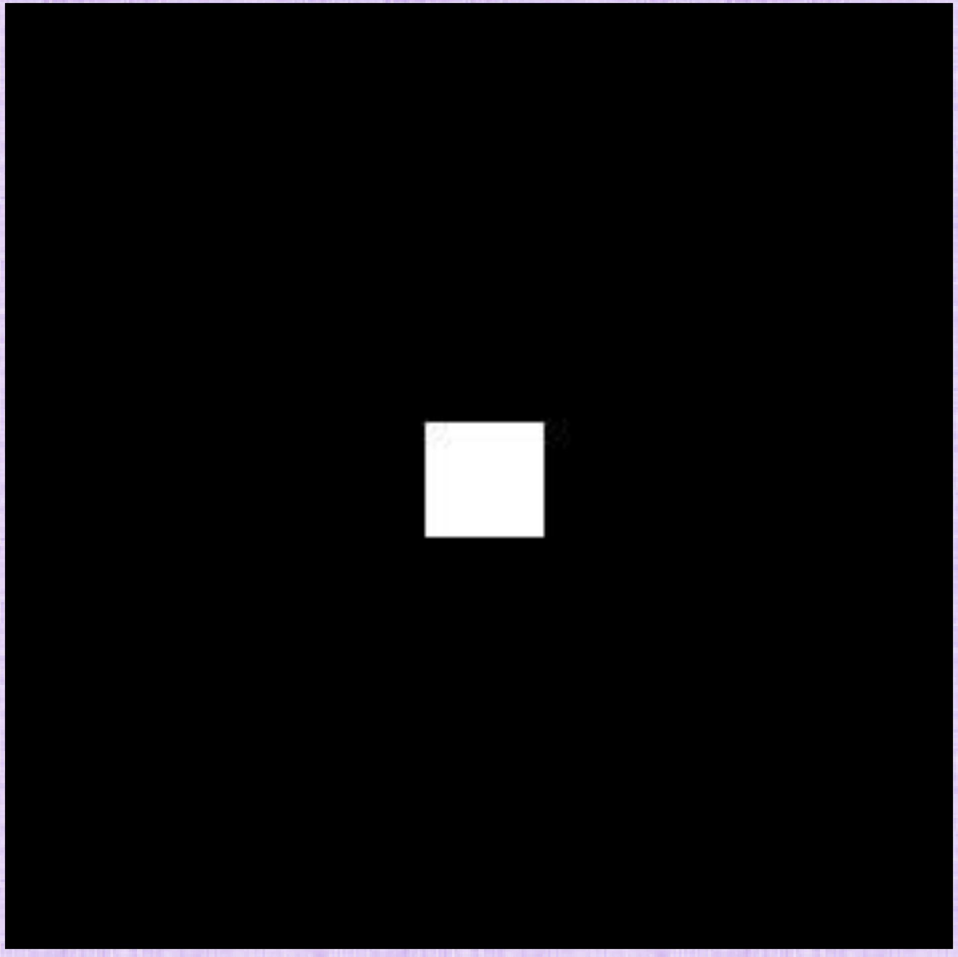
reasonable size for the convolution integral



removes most but not all of the high frequencies

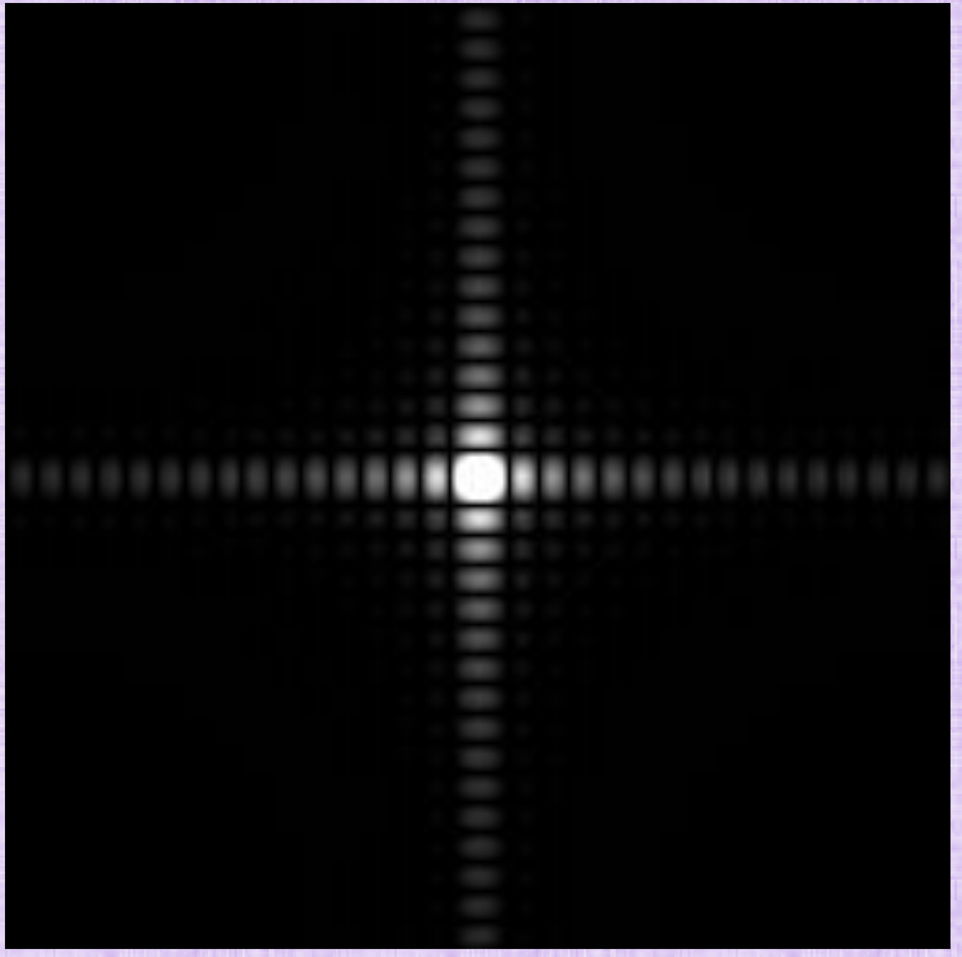
Wider Box Filter

g



more expensive convolution integral

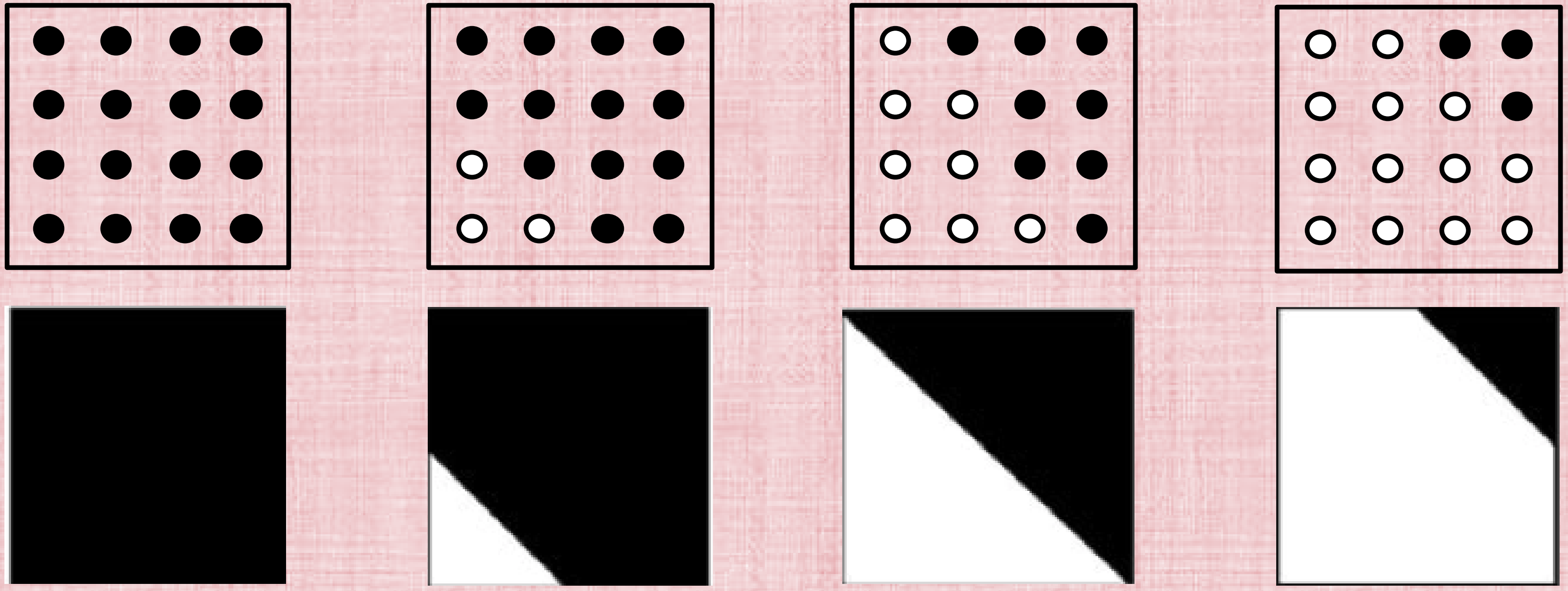
$F(g)$



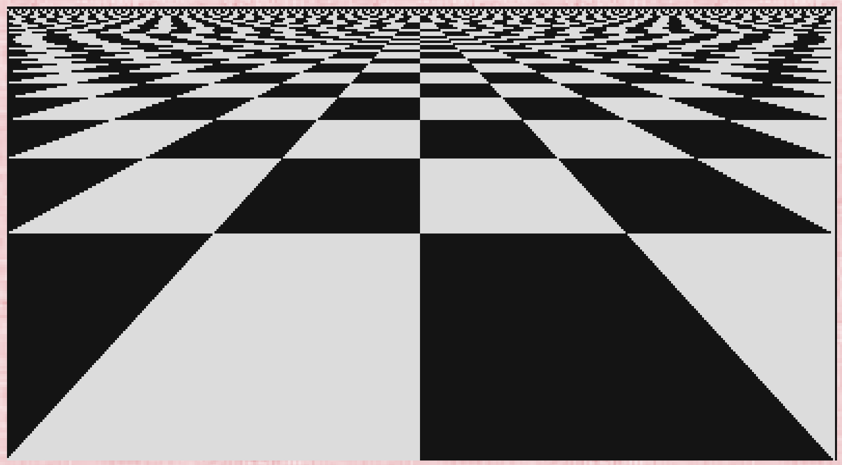
removes more of the high frequencies

Super-Sampling for Ray Tracing

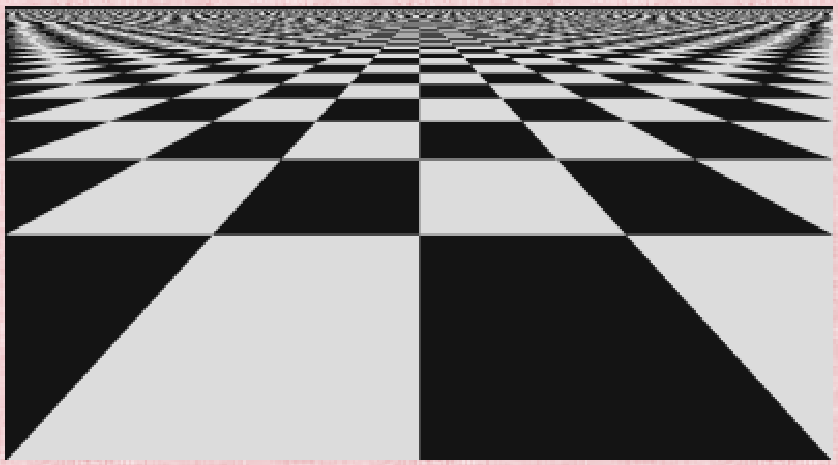
- Collect extra information/samples (in each pixel), and average the result (e.g. with a box filter)
  - Rendering a 100x100 image with 4x4 super-sampling is equivalent to rendering a 400x400 image
  - That properly represents (without aliasing) frequencies up to 4 times higher than the 100x100 image would
  - Then, apply a 4x4 box filter to remove as much of those higher frequencies as possible
- Converges to the area coverage integral, as the number samples per pixel increases
  - Efficiency: only super-sample pixels that have high frequencies (e.g. edges)
  - Better to use pseudo-random Monte-Carlo super-sampling, instead of uniform super-sampling



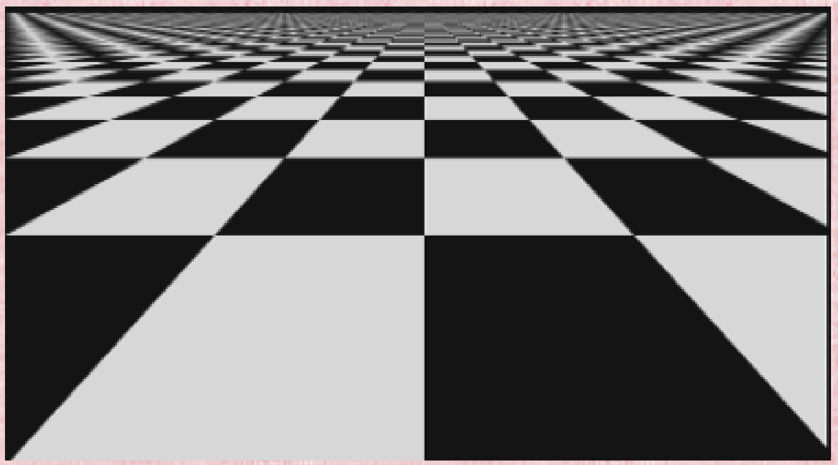
Comparison



Point Sampling

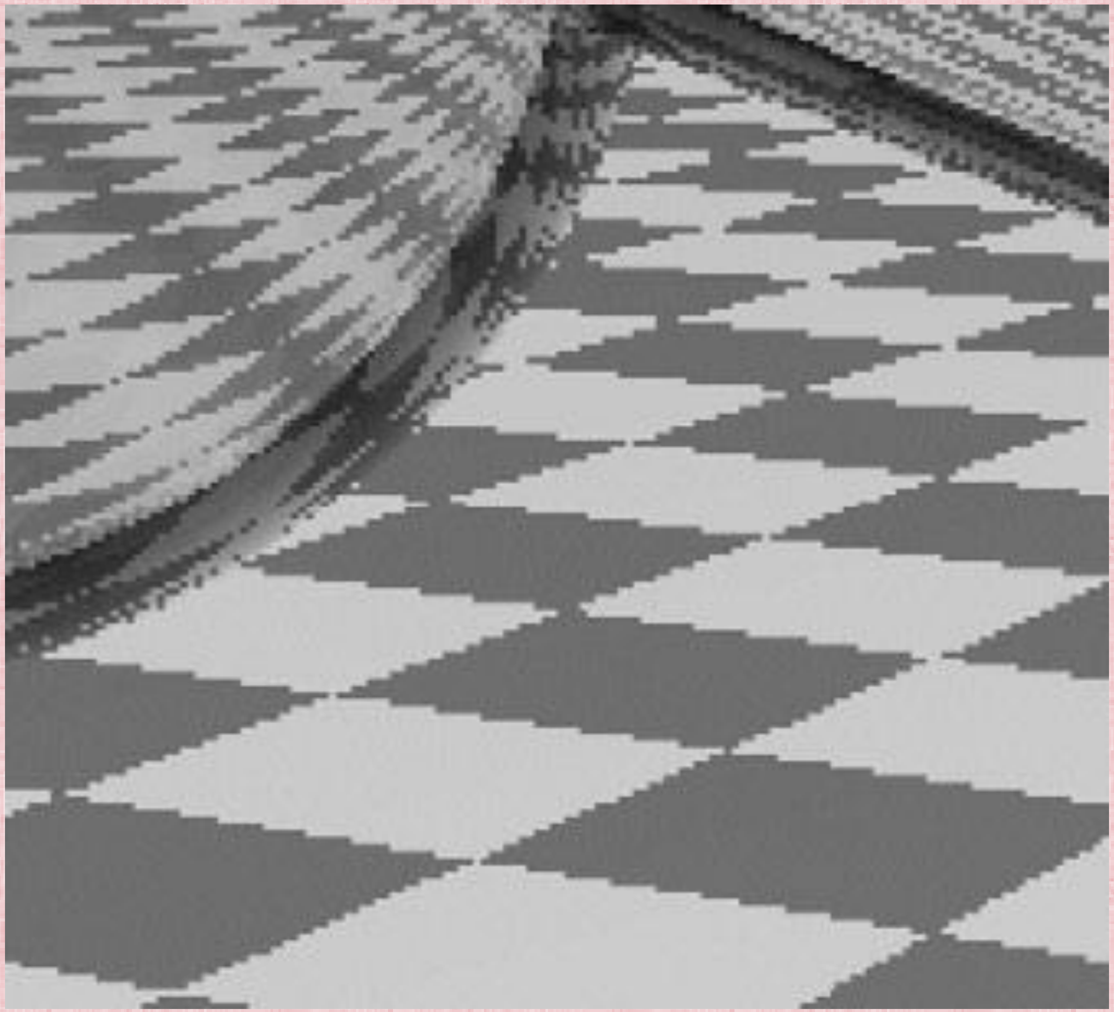


4x4 Super-Sampling

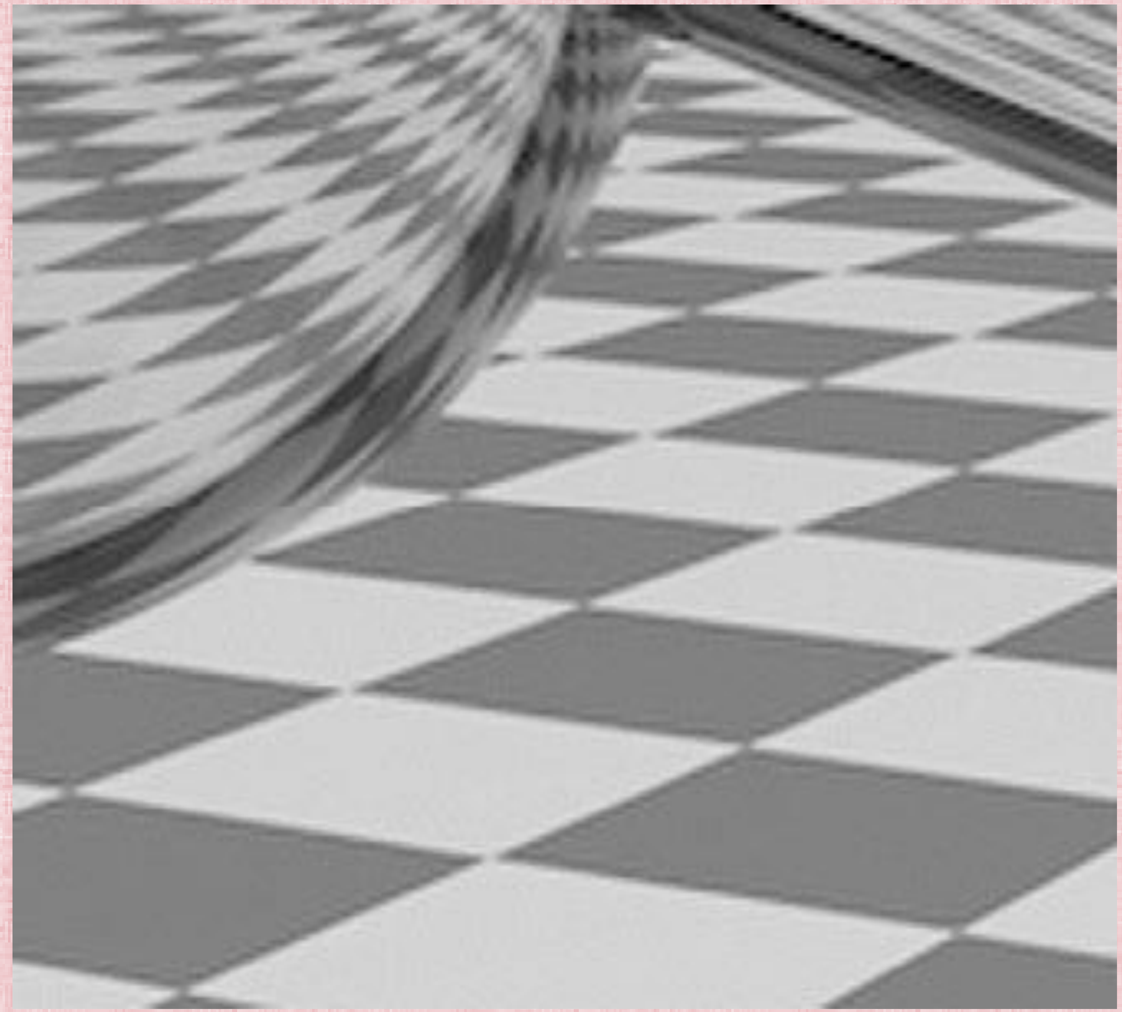


Exact Area Coverage

Super-Sampling for Ray Tracing



Jaggies



Anti-Aliased