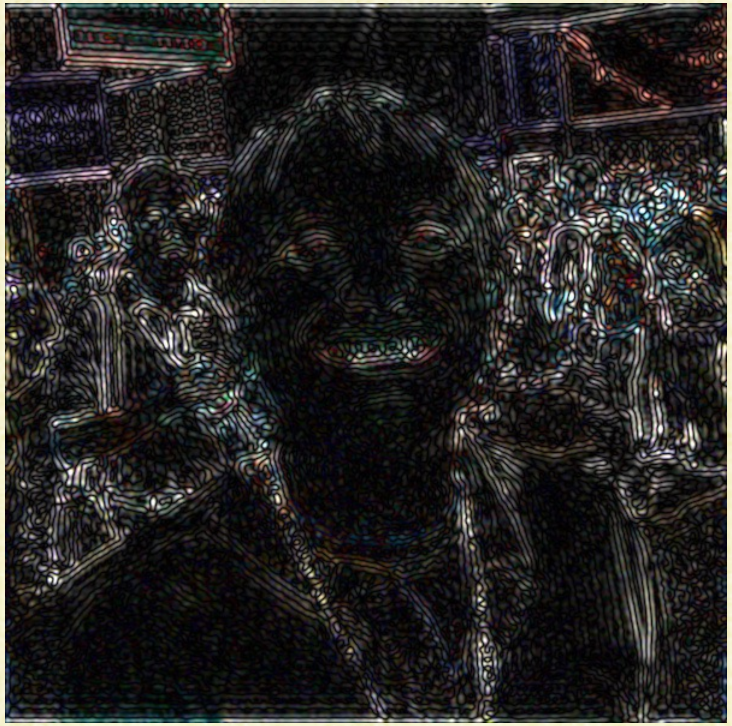
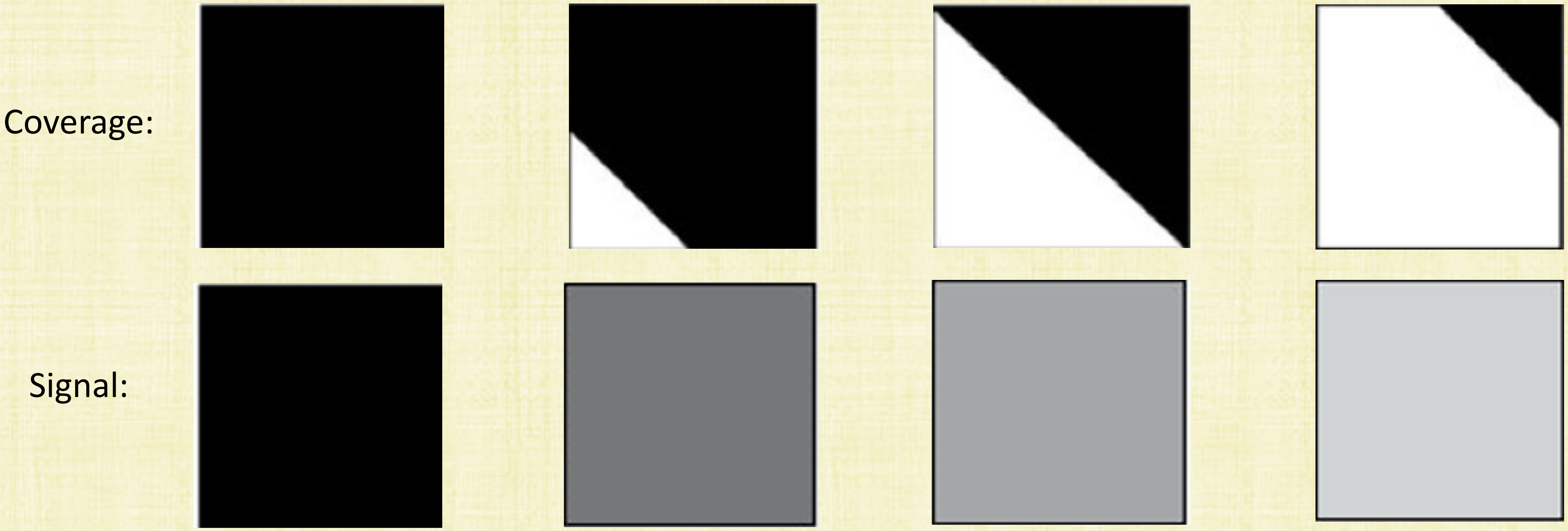
Sampling



Area-Coverage

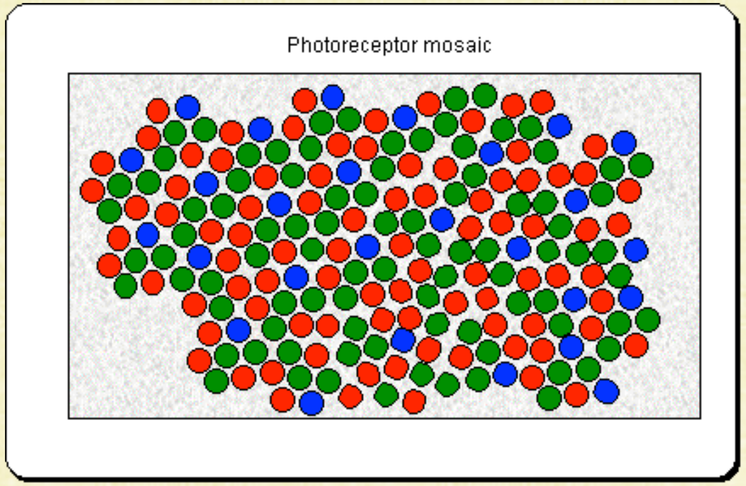
- Real-world sensors get a signal based on the area fraction of the sensor "covered" by objects

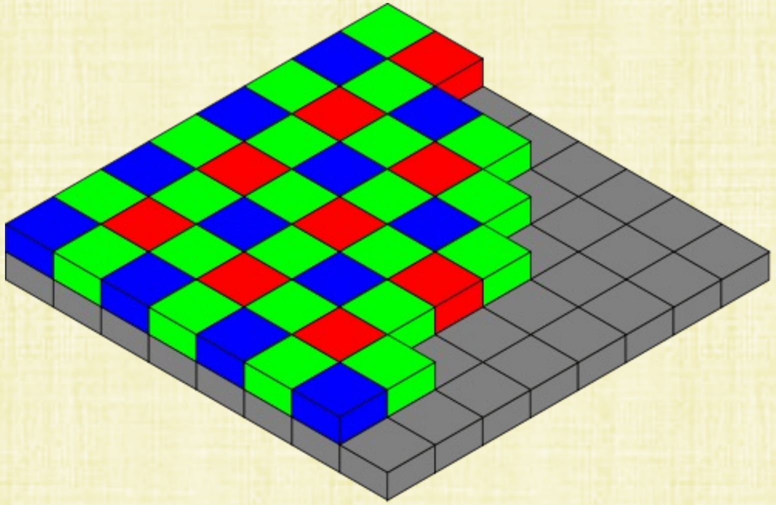


- A ray tracer only gets a sample of the geometry (using a ray-geometry intersection point)
- A scanline renderer projects the entire triangle onto the image plane
  - Testing pixel centers against triangles only uses sample information from the geometry
  - Computing area overlap between triangles and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

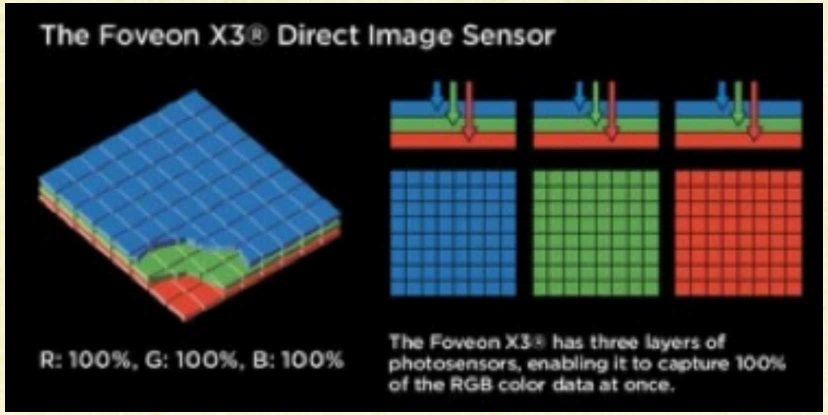
Missing Information

- Eyes/cameras don't collect all of the information either
- The staggered spatial layout of real-world sensors means that large regions lack information for certain wavelengths (layered approaches can help to circumvent this)



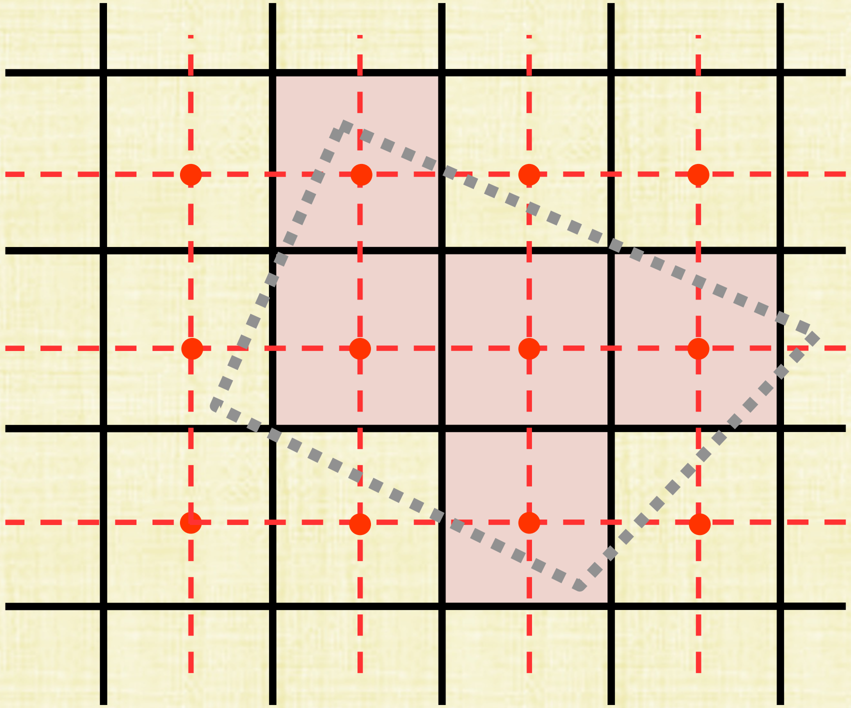


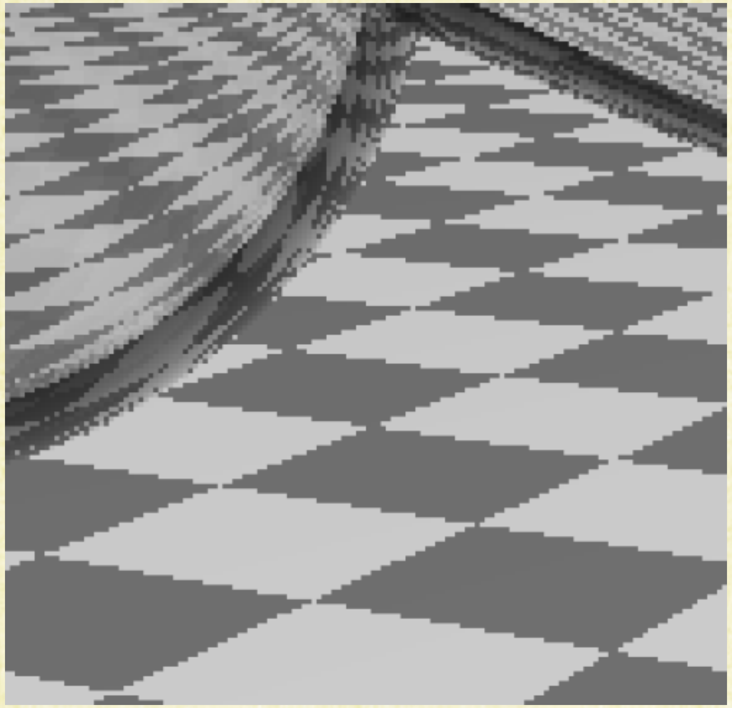
layered approaches can help to circumvent this:



Aliasing

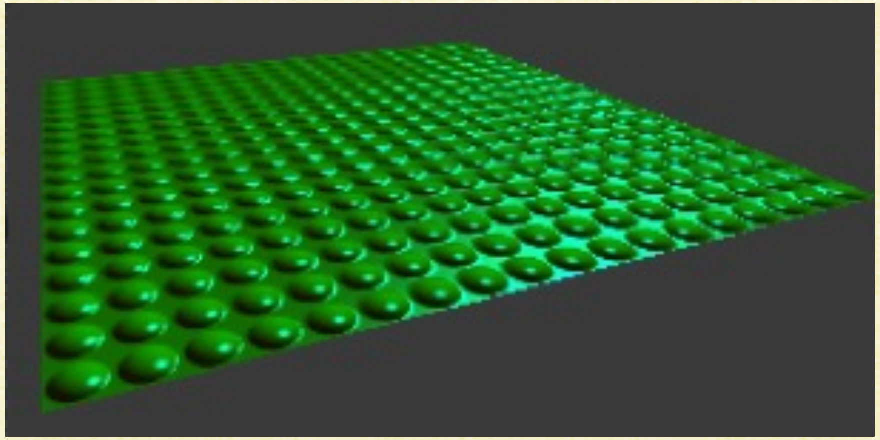
- Testing only the pixel center (with ray-tracing or scanline rasterization) leads to jagged edges
- This causes aliasing artifacts (an alias/imposter takes the place of the correct feature)
- A jagged line appears instead of the correct straight line
- Anti-aliasing strategies aim to reduce aliasing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

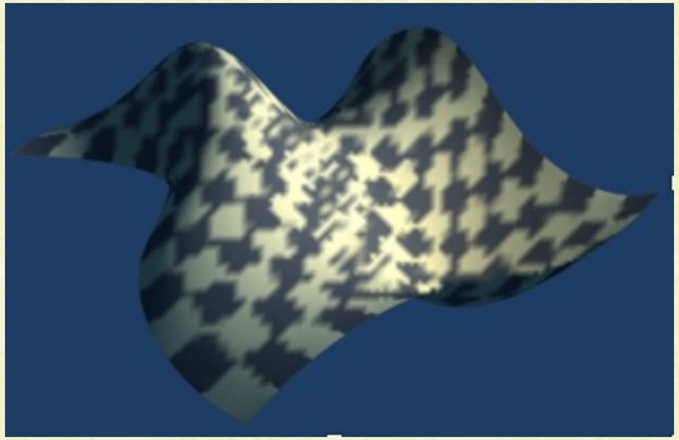


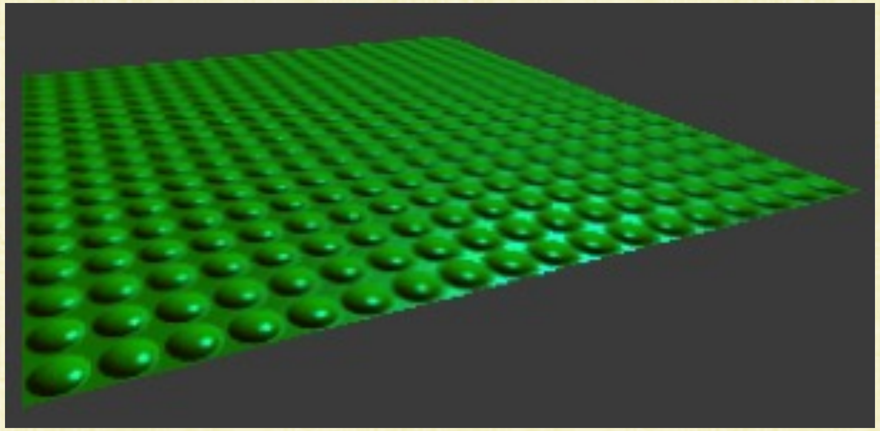


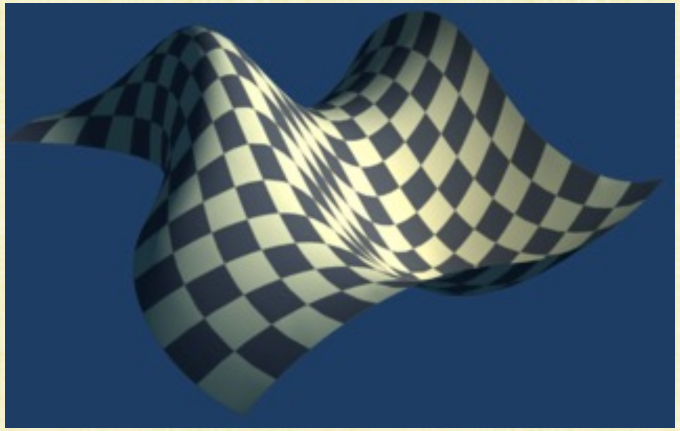
Aliasing: Shaders & Textures

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



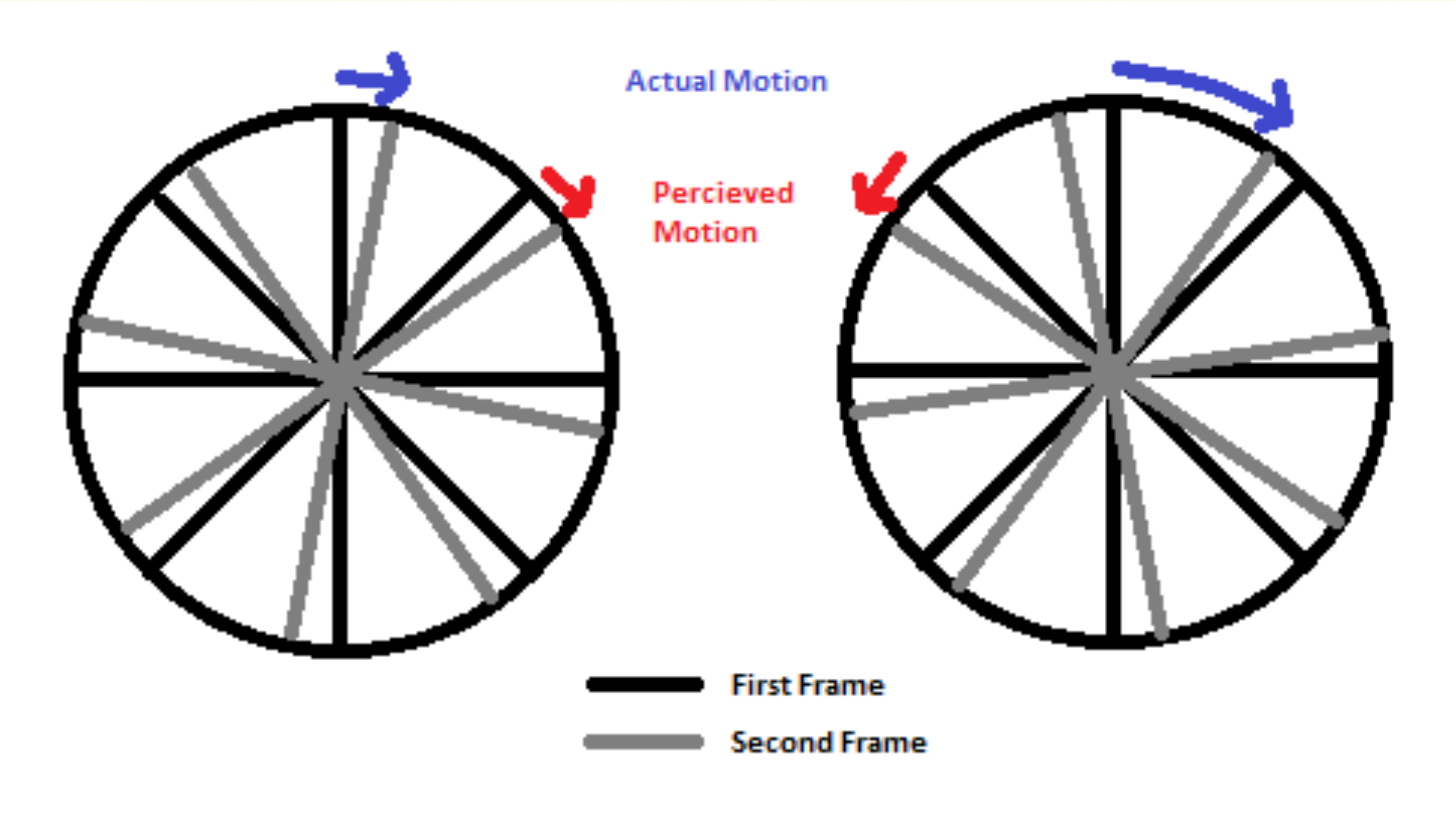






Temporal Aliasing

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

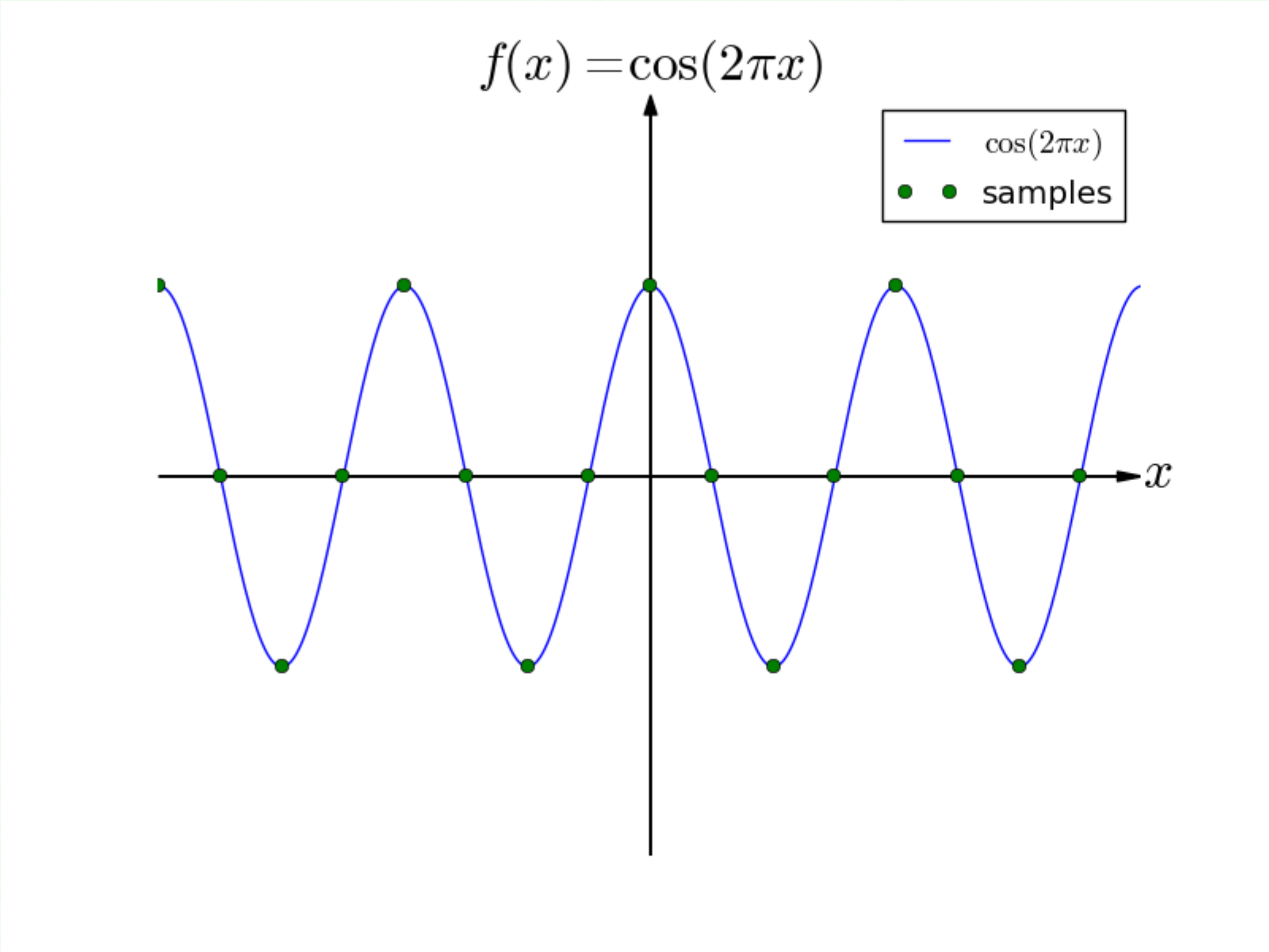


Sampling Rate

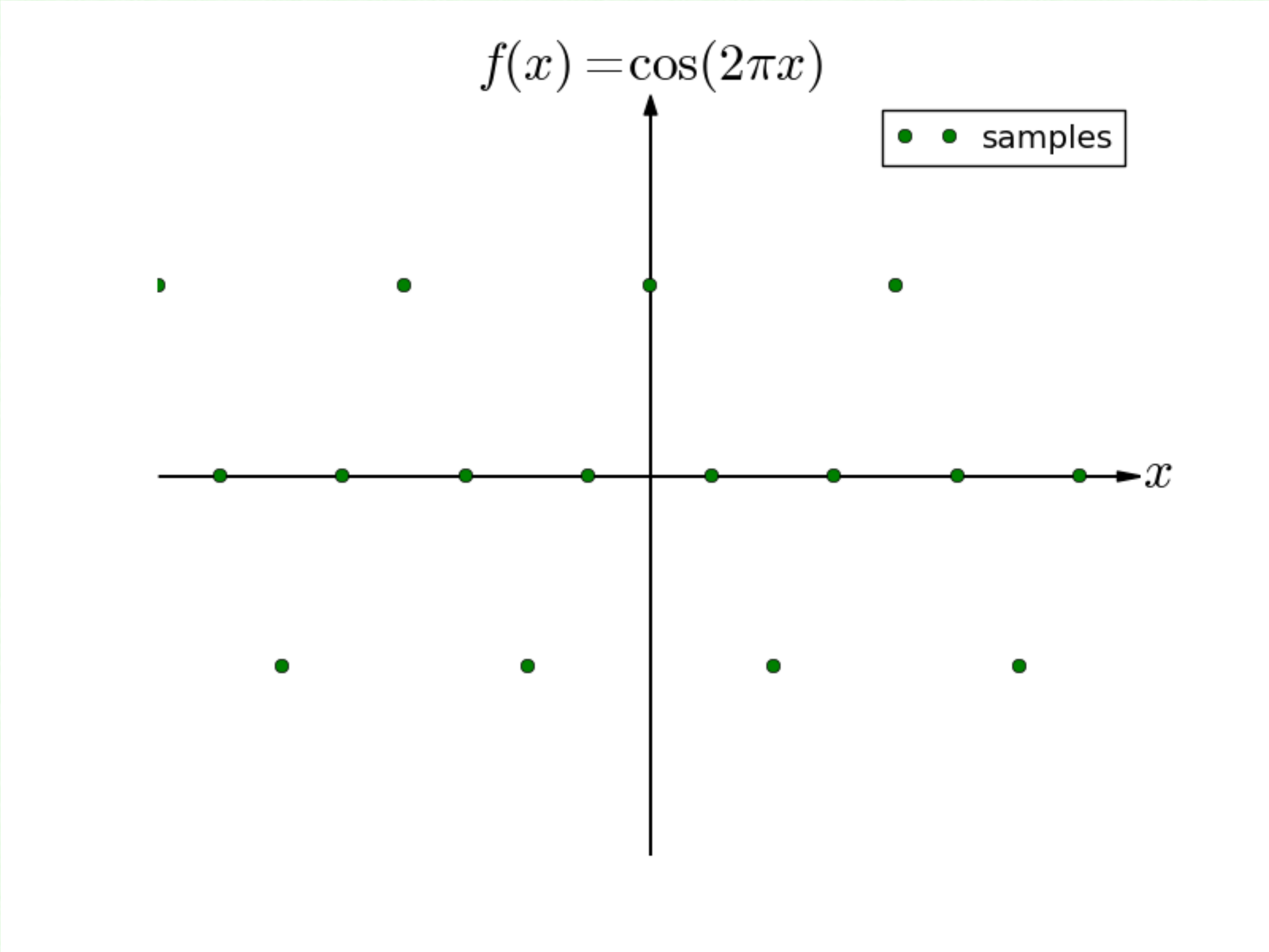
- Artifacts can be reduced by increasing the number of samples (per unit area)
- This can be accomplished by increasing the number of pixels in the image; but:
  - It takes longer to render the scene (because there are more pixels colors to determine)
  - Displaying higher-resolution images requires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

- Instead: Optimize the Sample Rate!
- Use the lowest possible sampling rate that does not result in "noticeable" artifacts
- What is the optimal sampling rate?

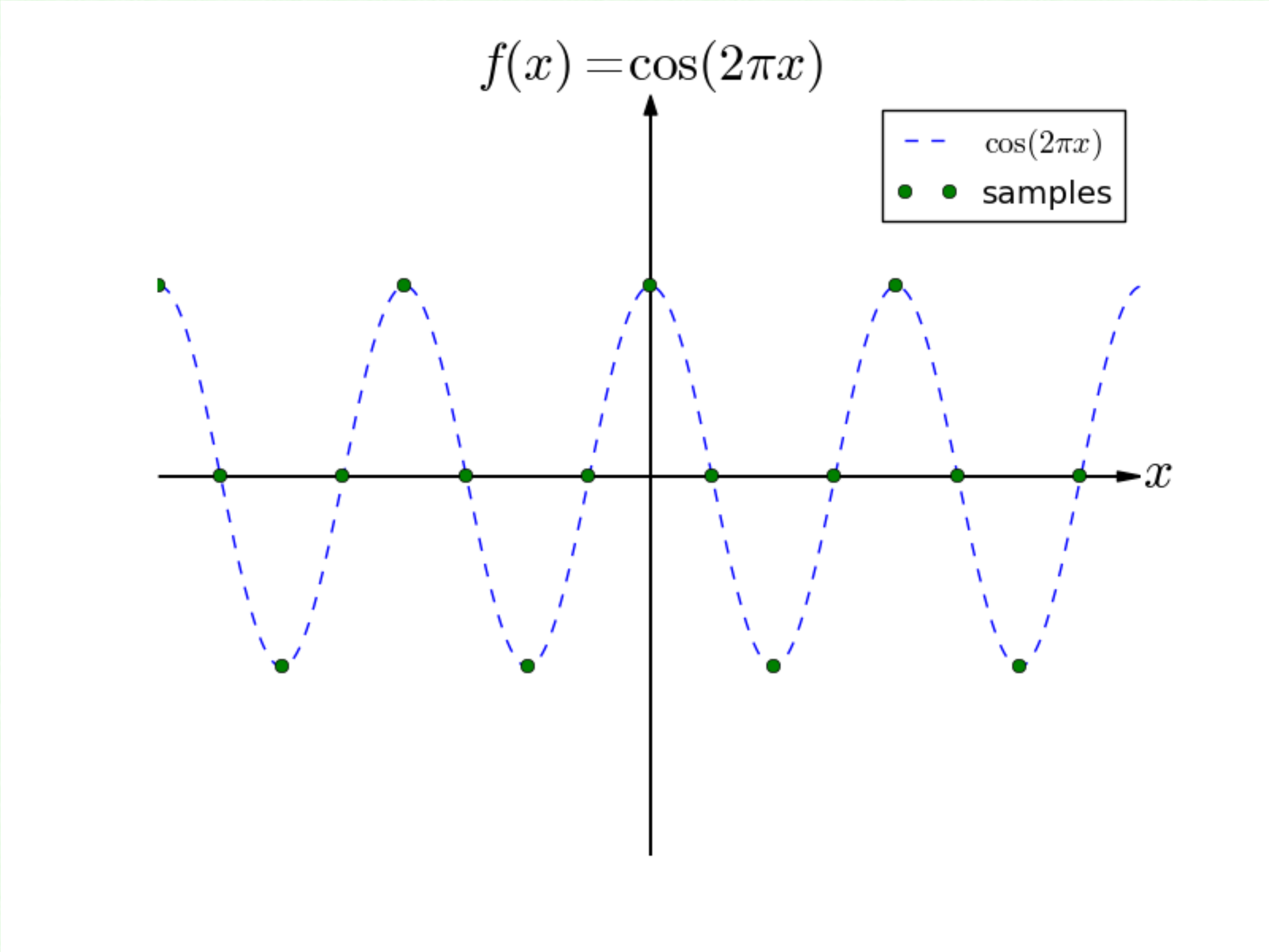
4 samples per period



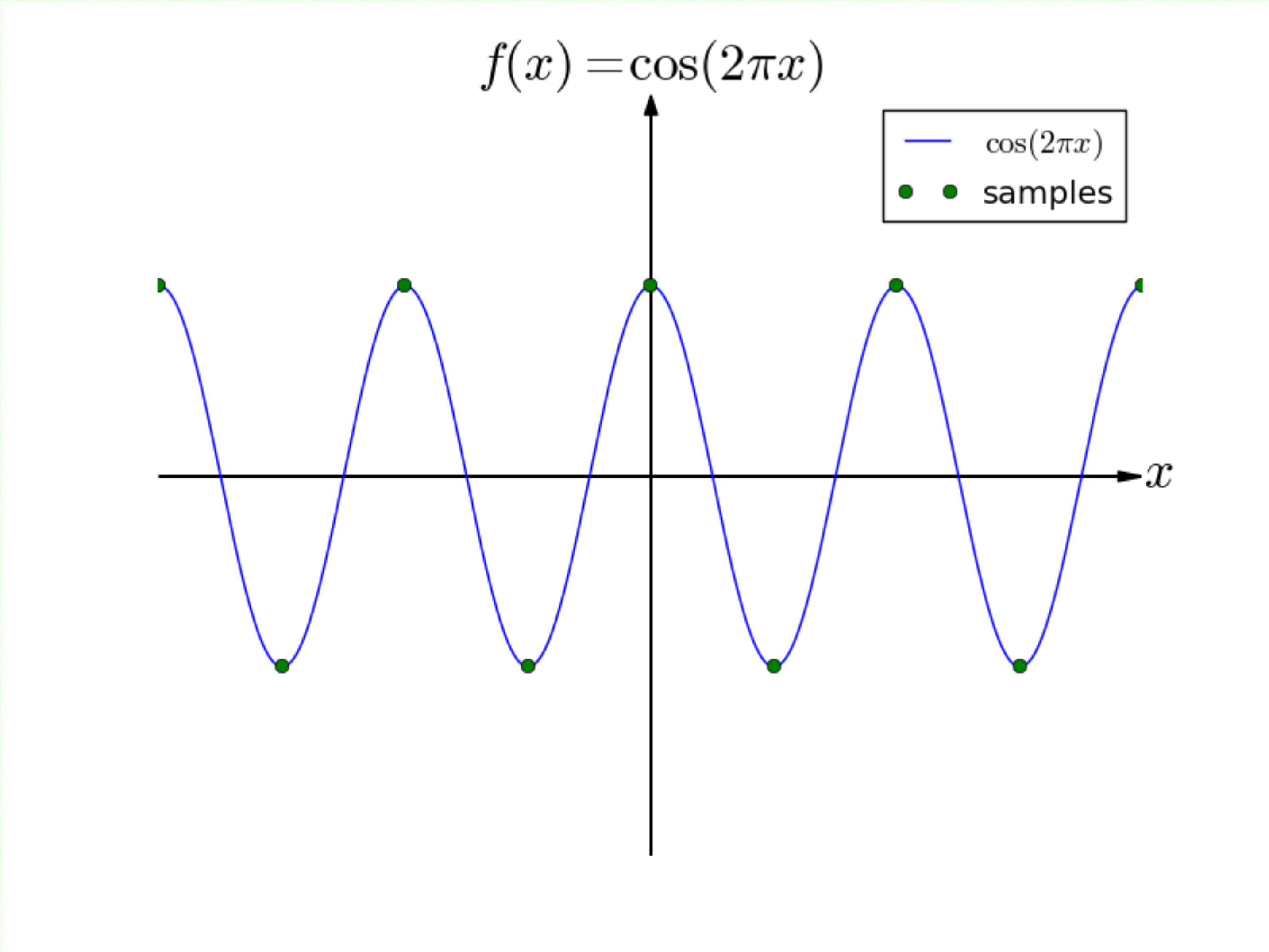
samples



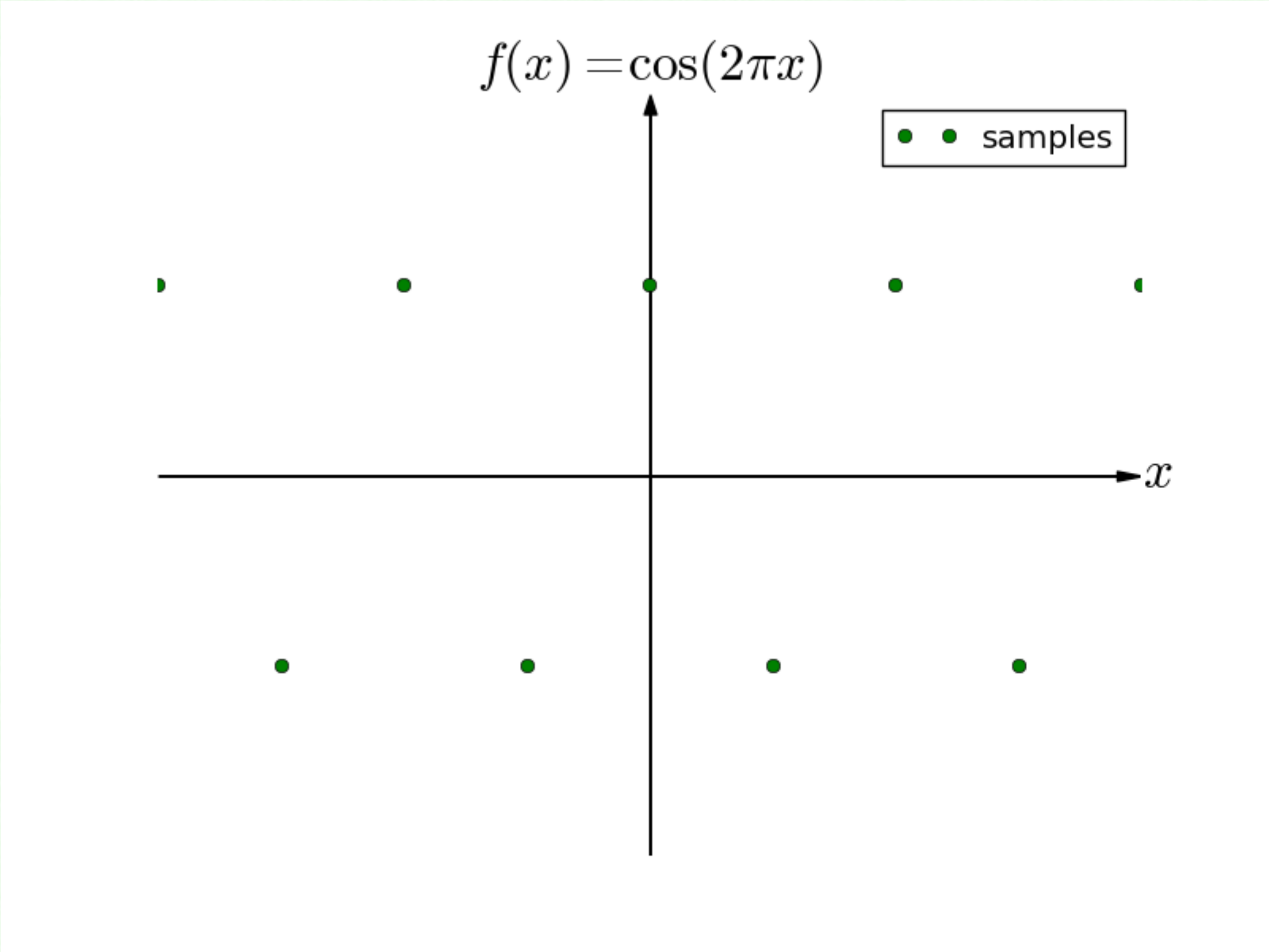
reconstruction



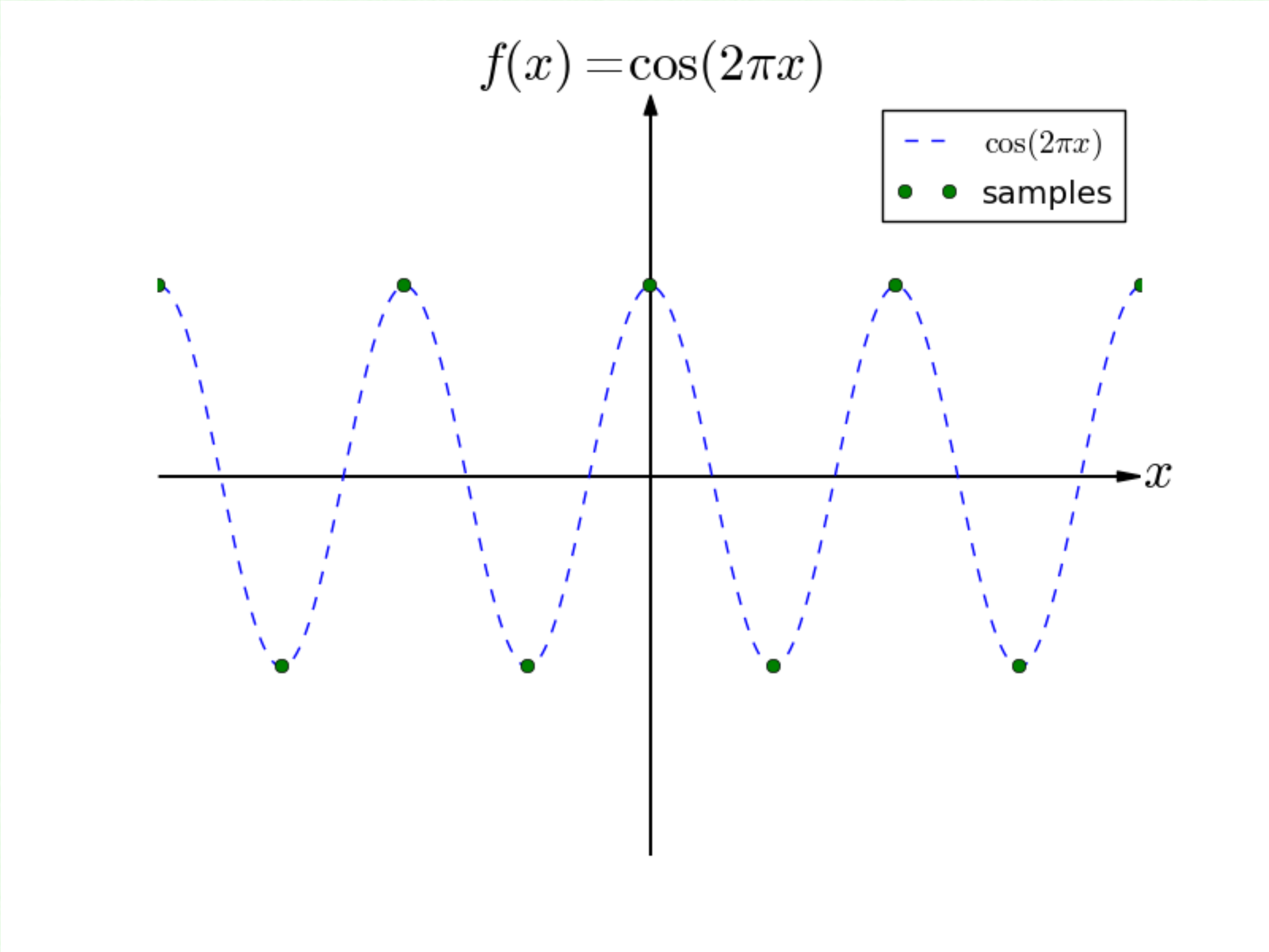
2 samples per period



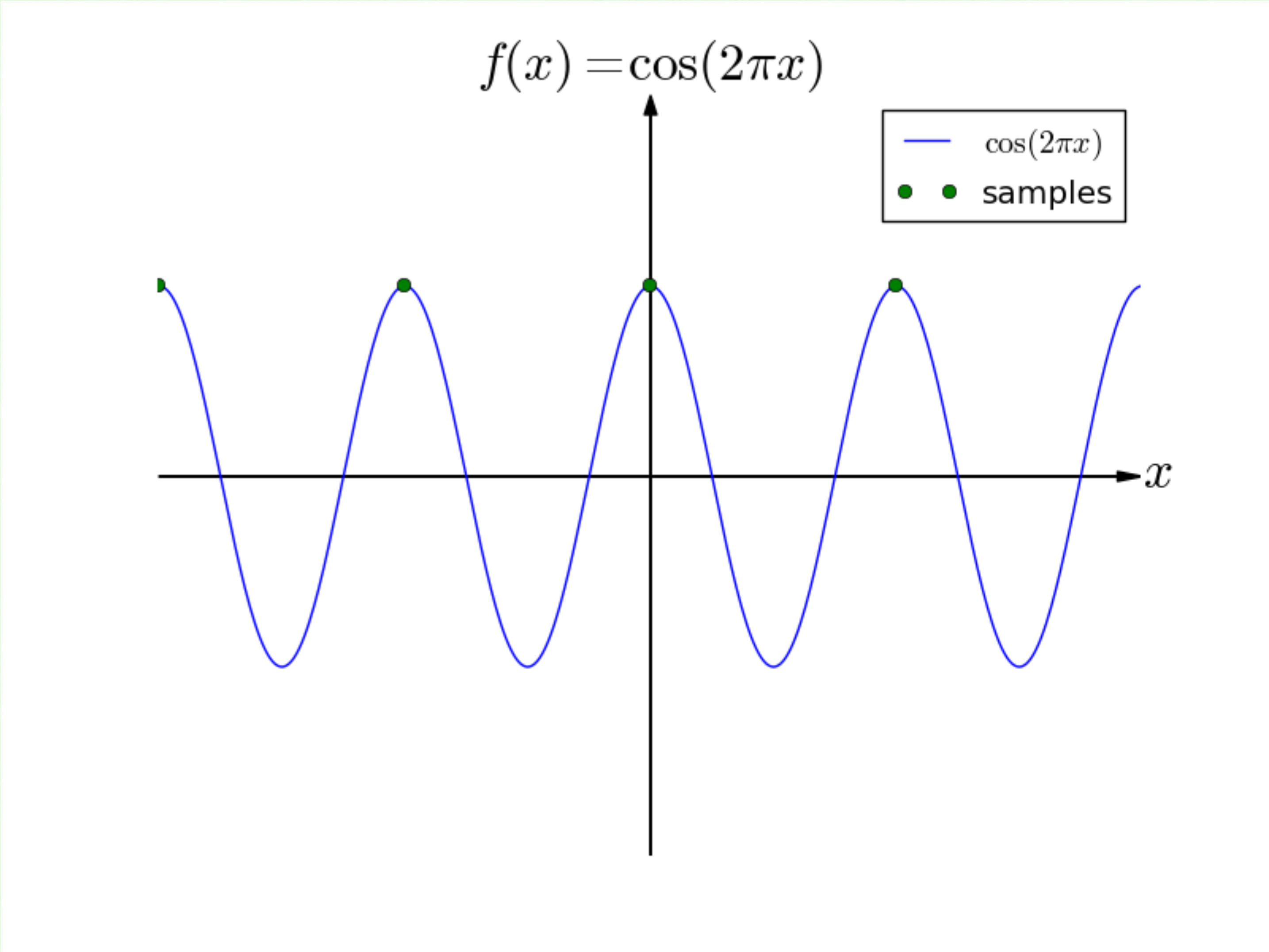
samples



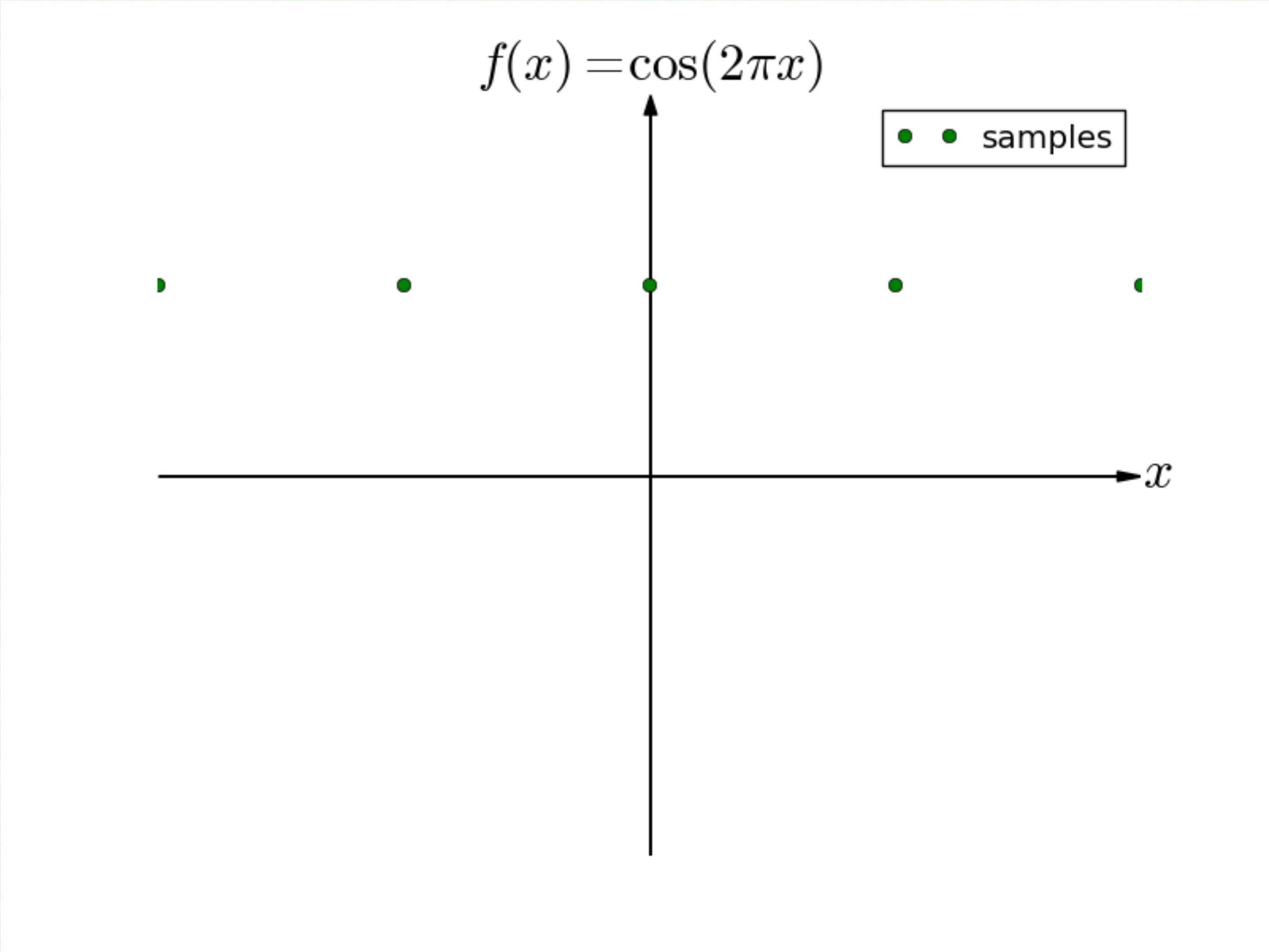
reconstruction



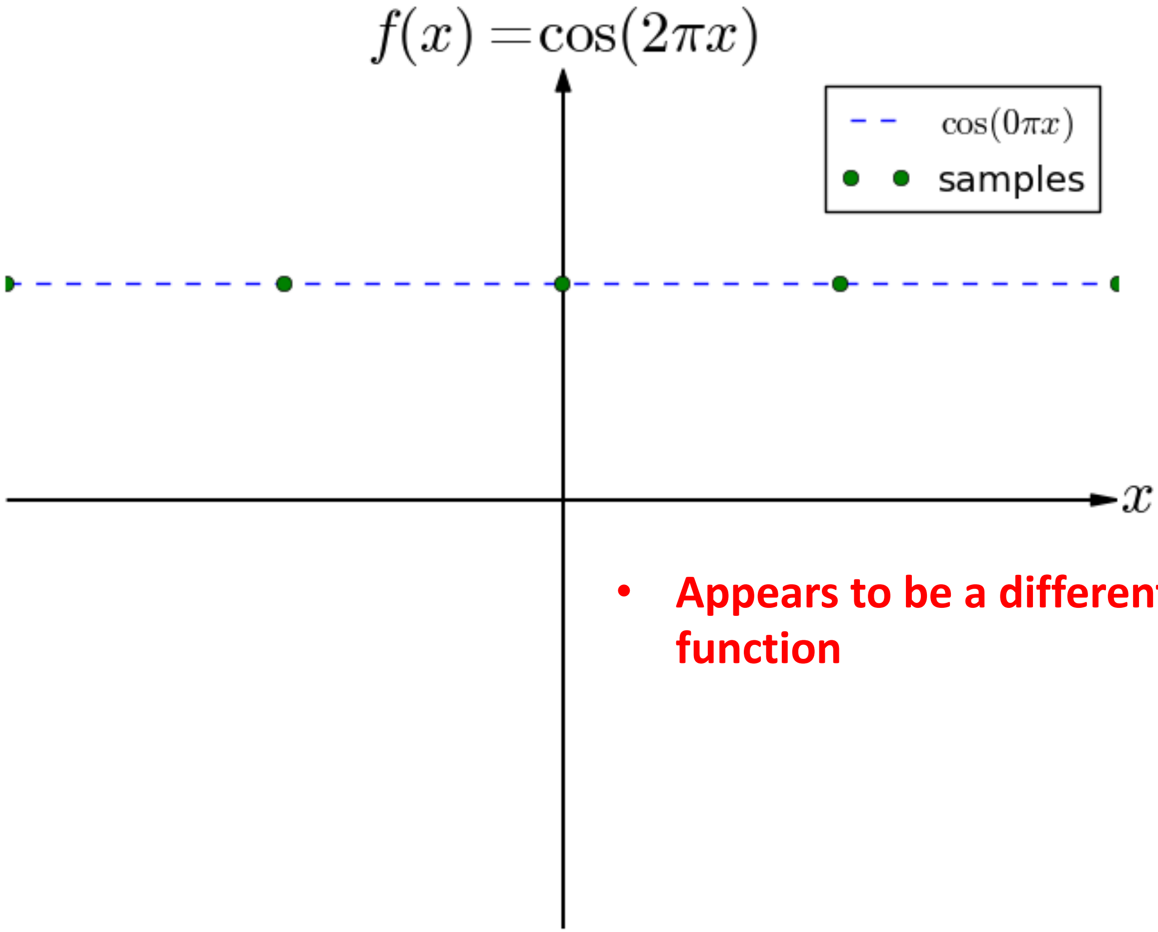
1 sample per period



samples

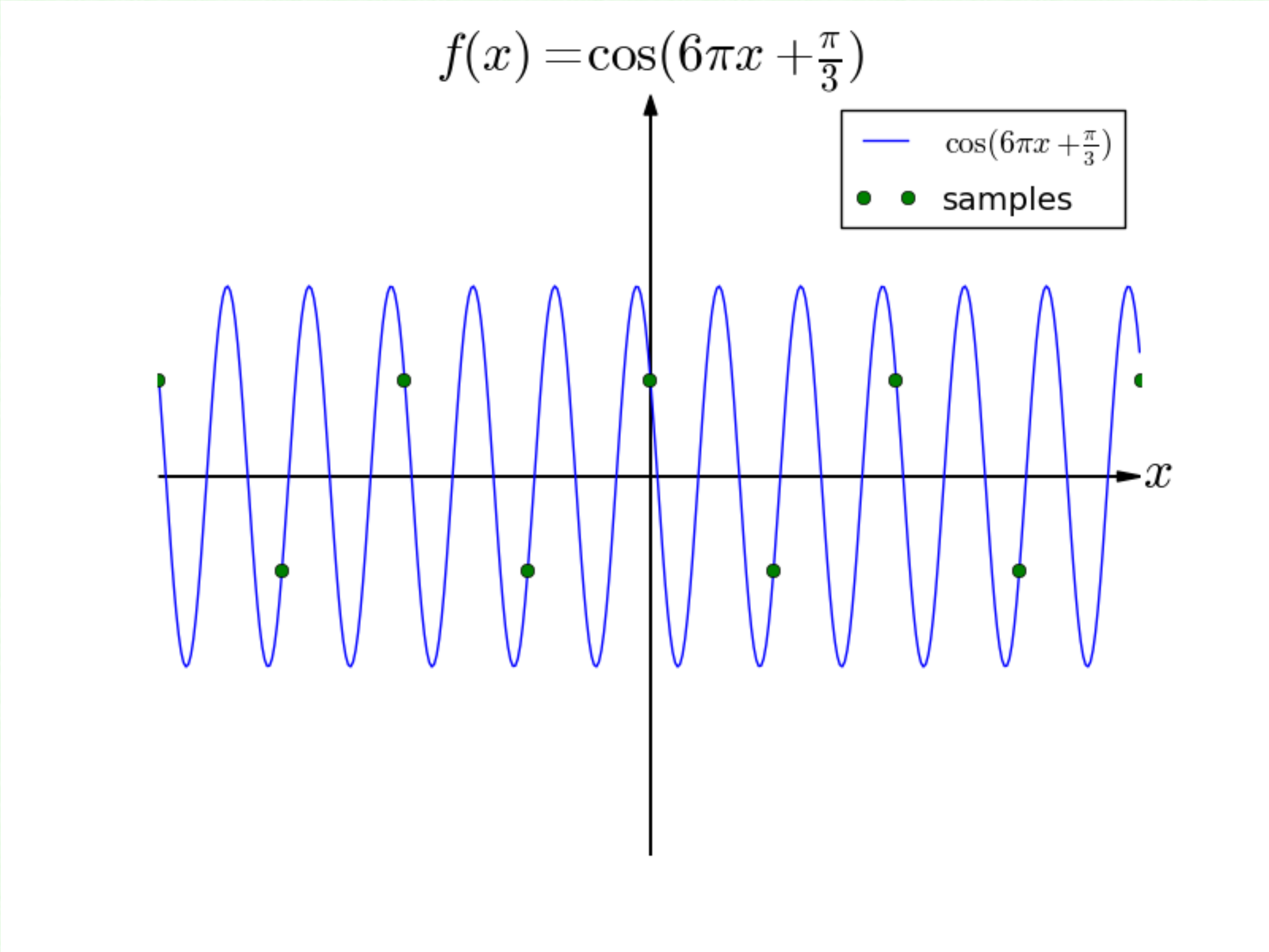


reconstruction

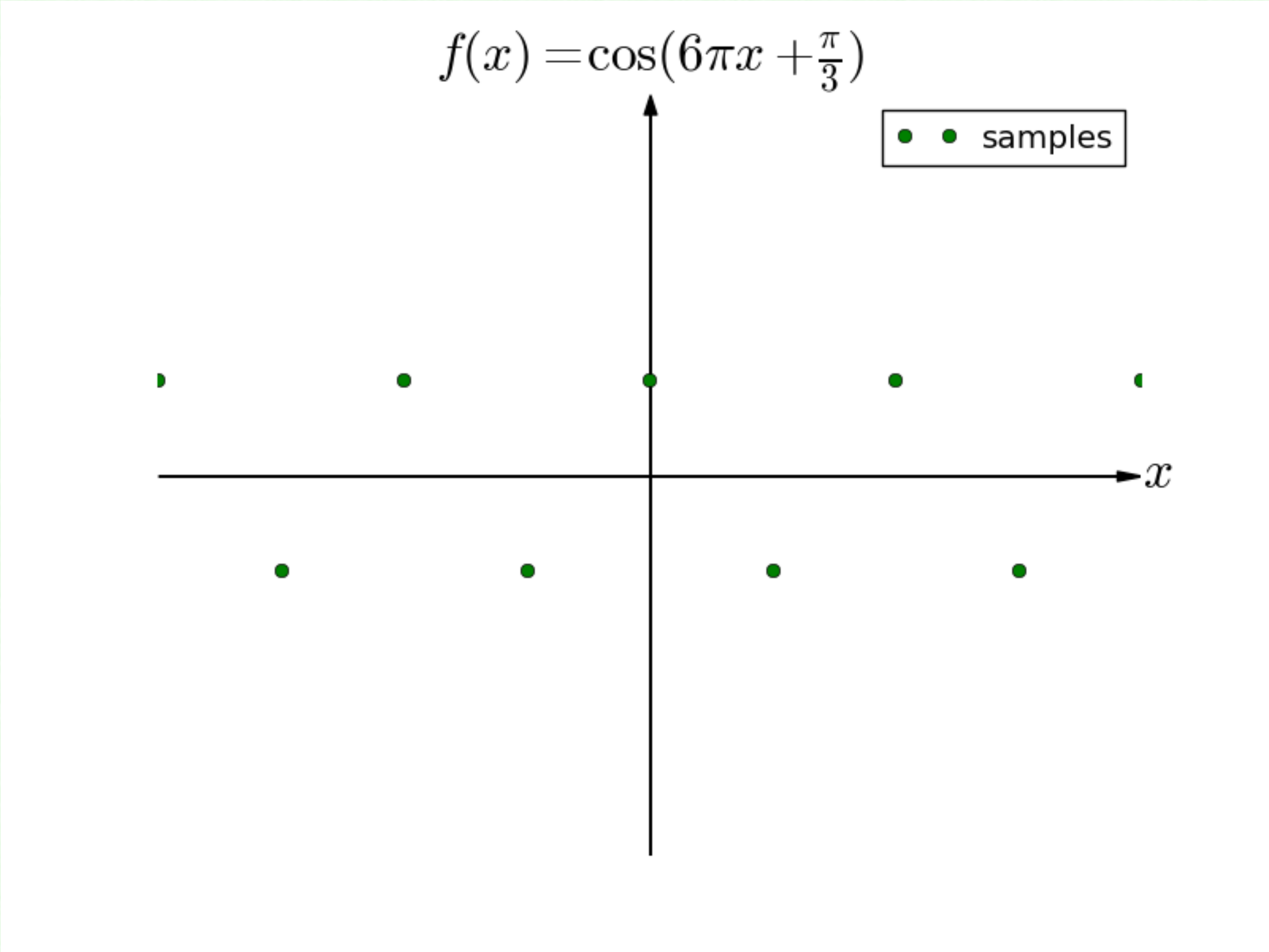


- **Appears to be a different function**

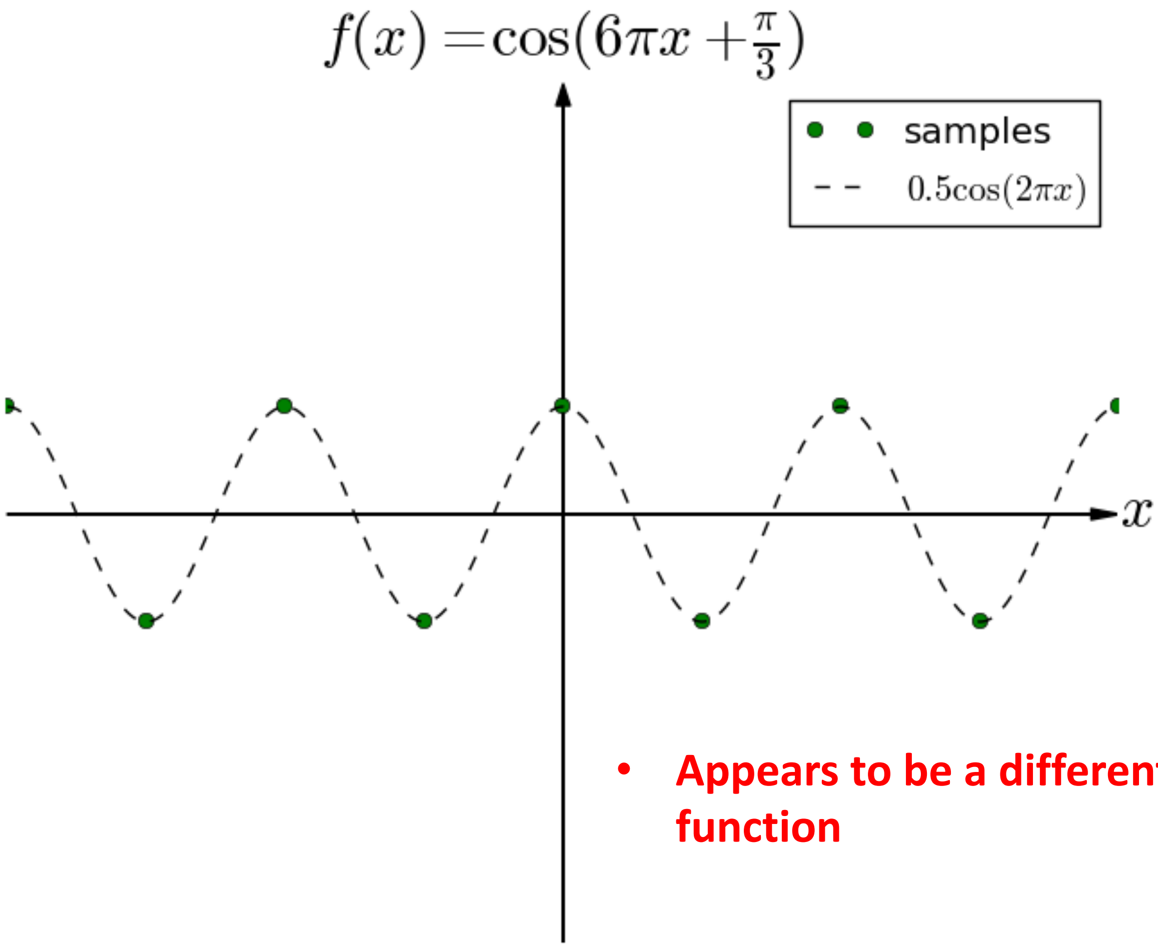
2/3 sample per period



samples

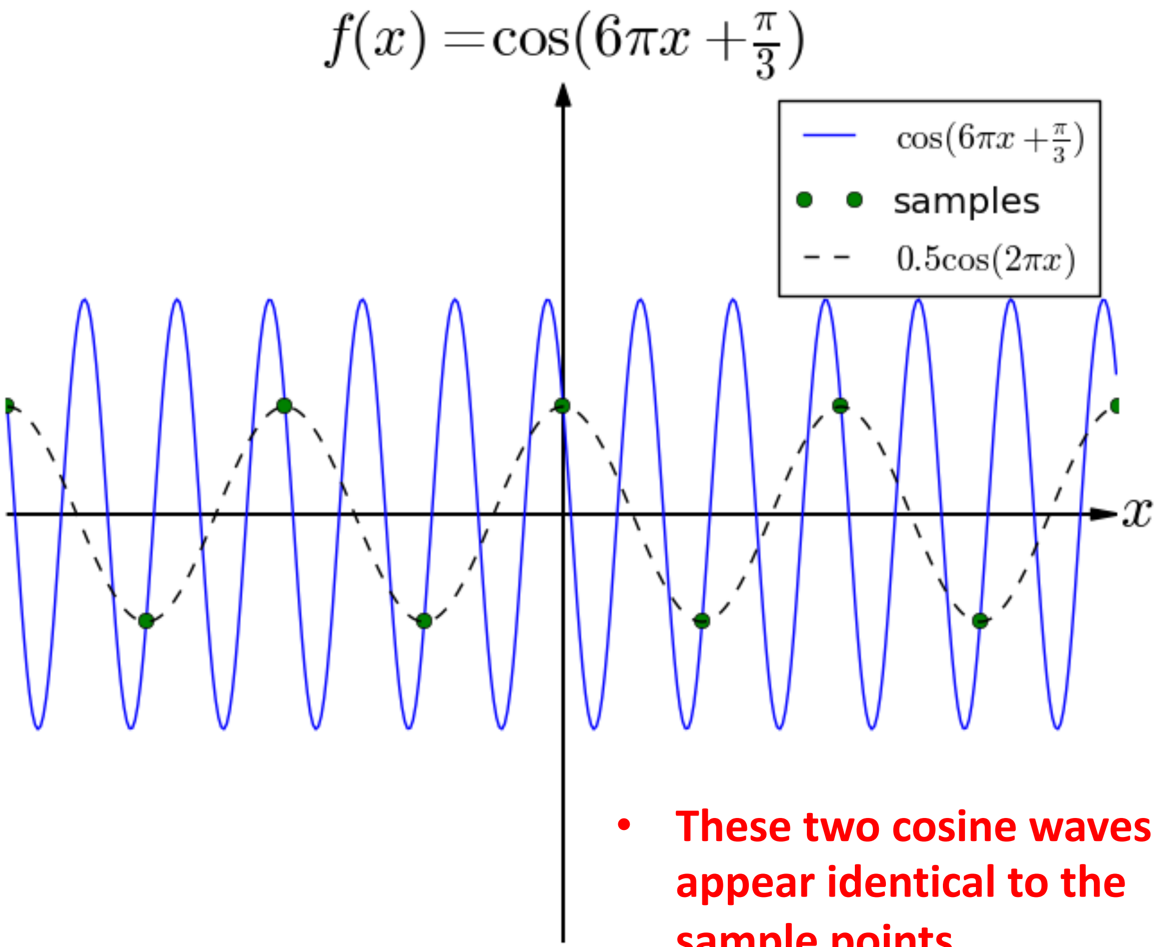


reconstruction



- **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 <u>Nyquist frequency</u> is defined as <u>half</u> the sampling frequency
- If the function being sampled has no frequencies above the Nyquist frequency, then no aliasing occurs

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

Fourier Transform

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

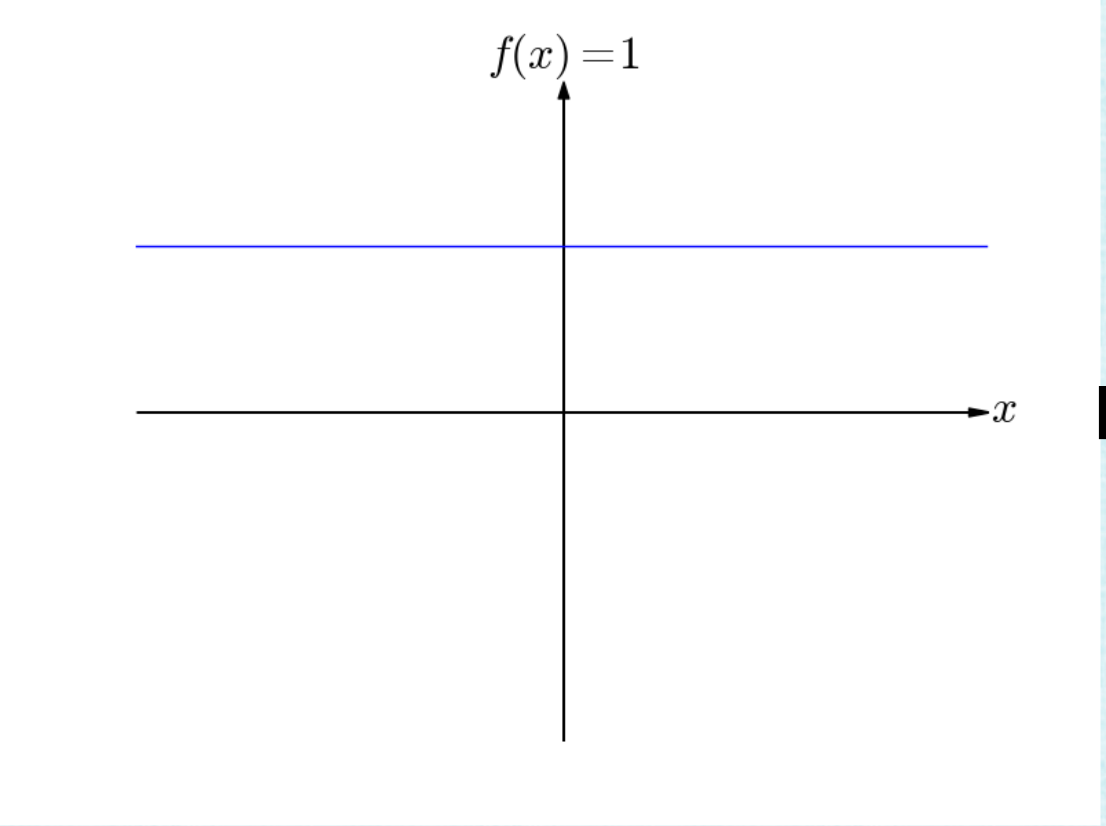
Frequency Domain: $F(k) = \int_{-\infty}^{\infty} f(x) e^{-2\pi i k x} dx$

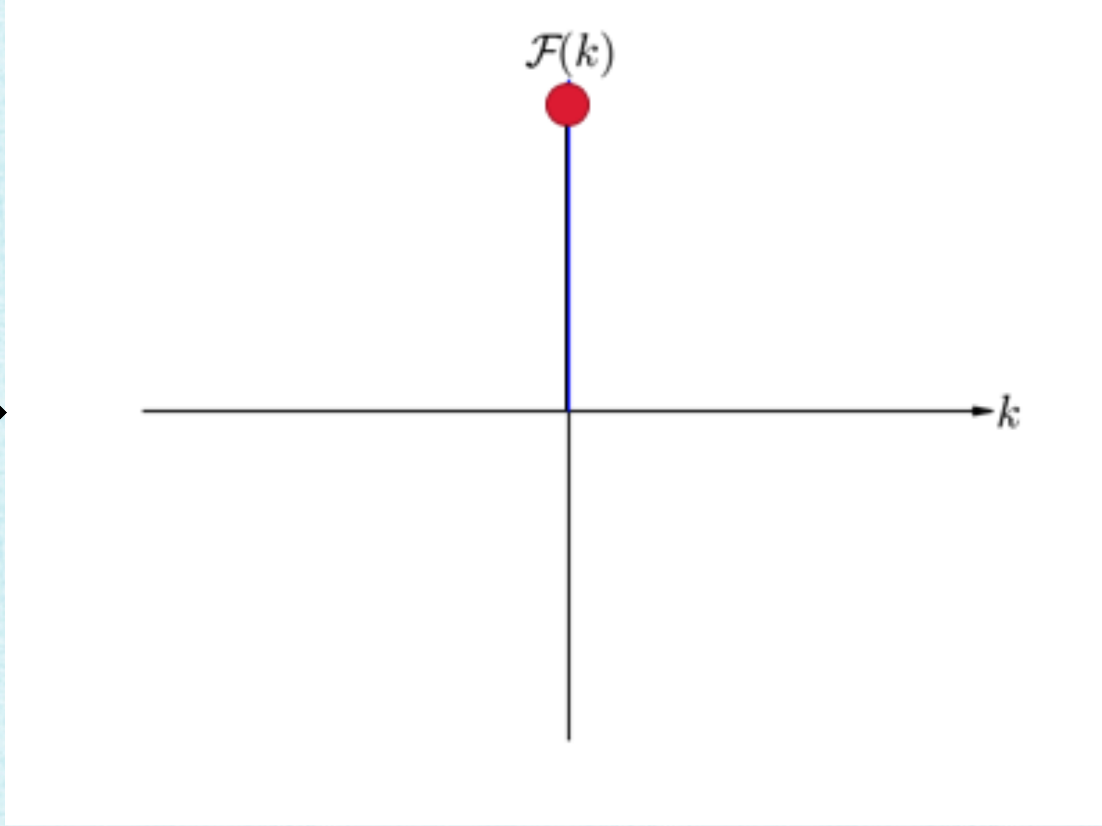
Spatial Domain: $f(x) = \int_{-\infty}^{\infty} F(k) e^{2\pi i k x} dk$

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

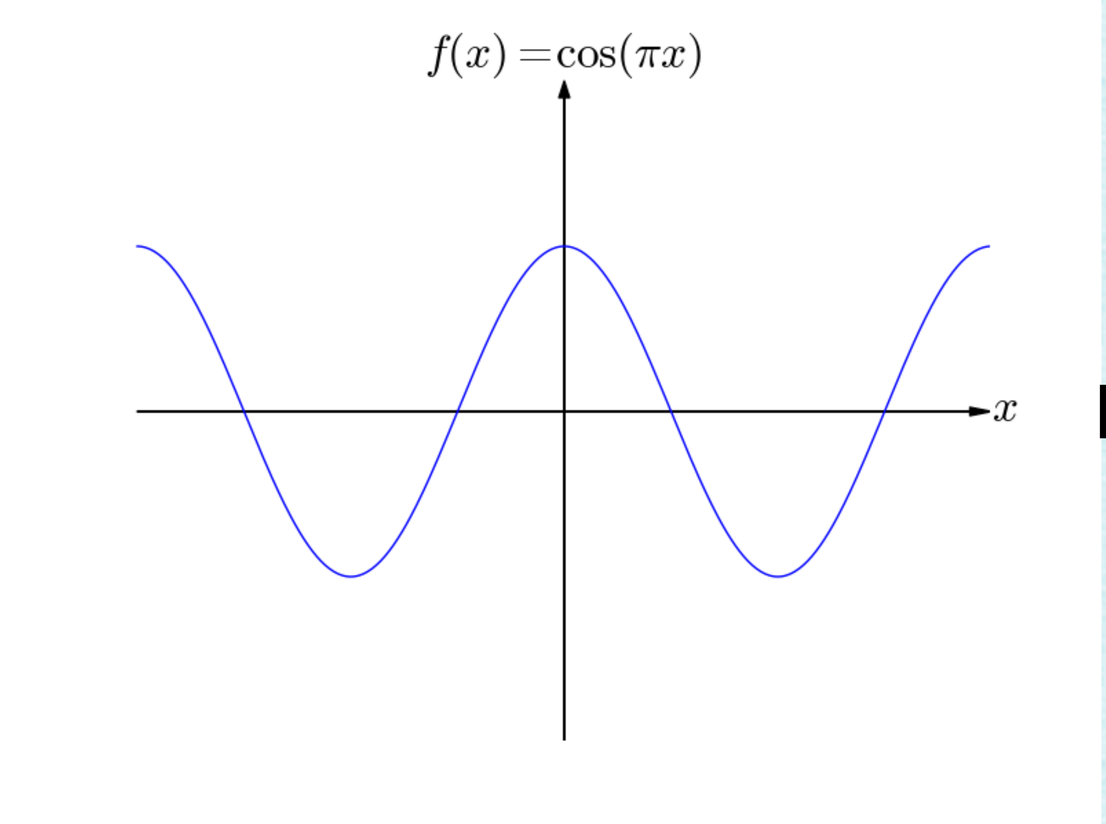
$$\cos\theta = \frac{e^{i\theta} + e^{-i\theta}}{2} \qquad \sin\theta = \frac{e^{i\theta} - e^{-i\theta}}{2i}$$

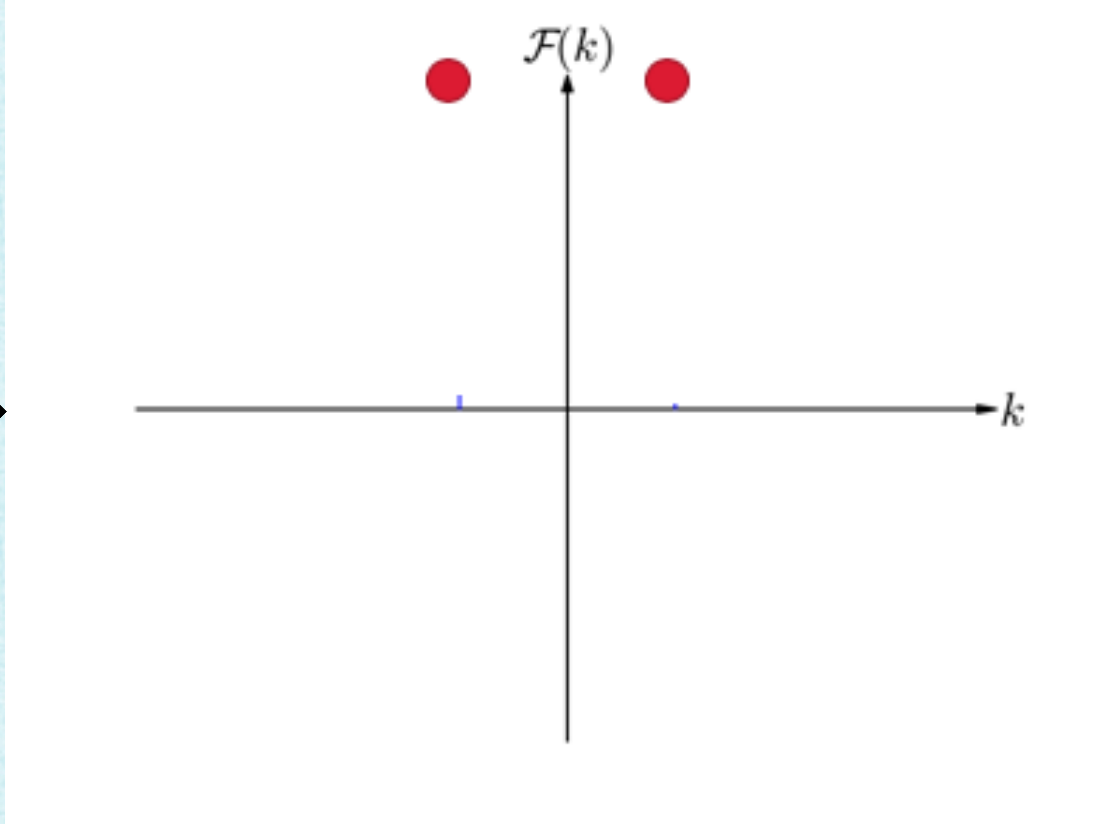
Constant Function



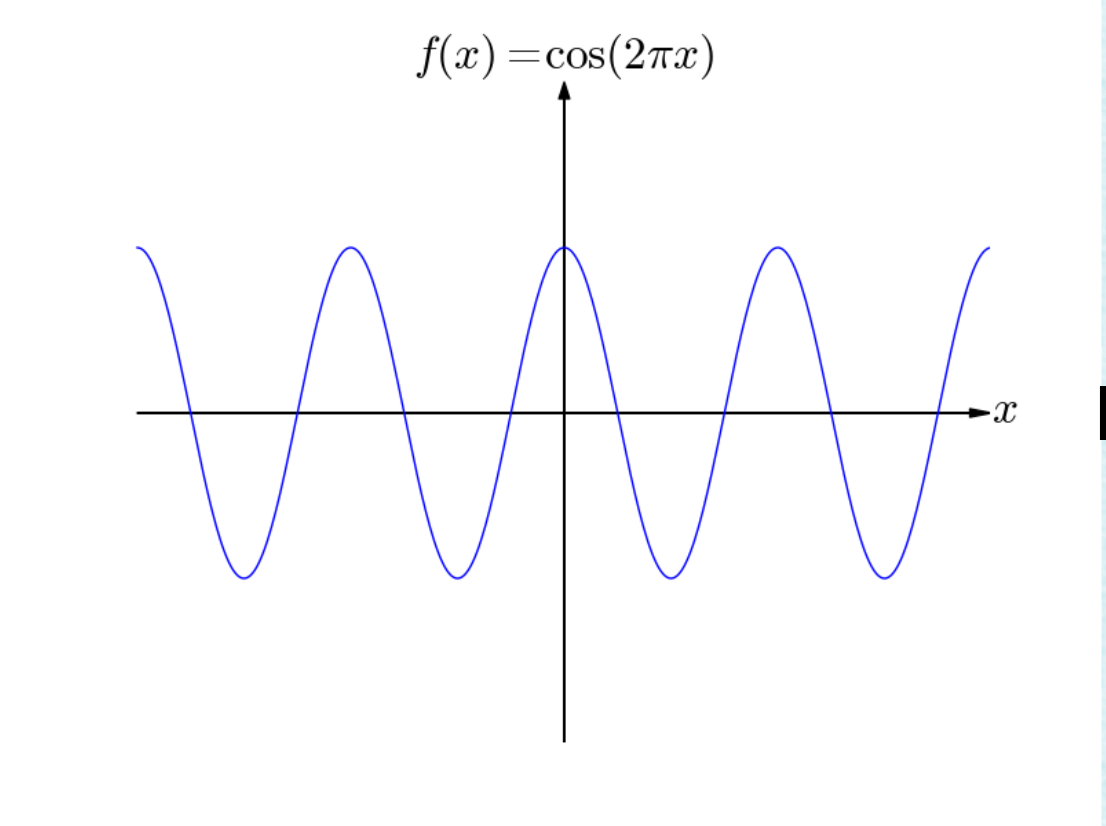


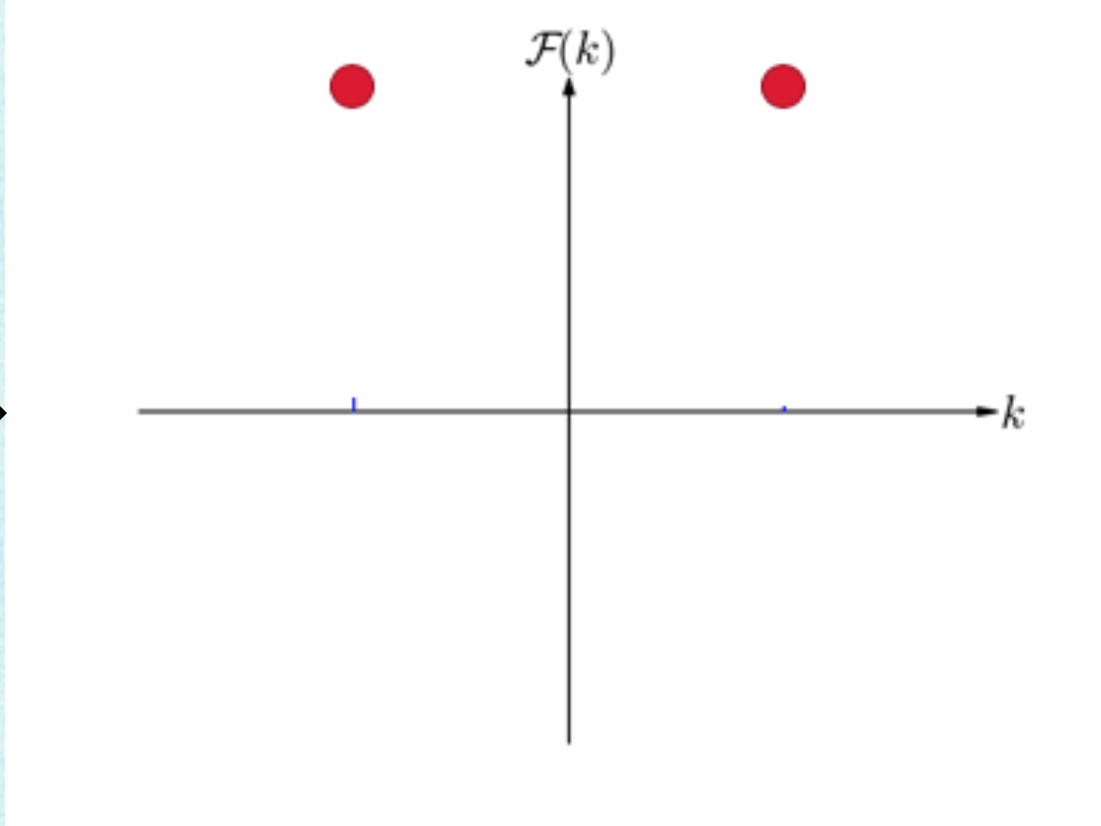
Low Frequency Cosine



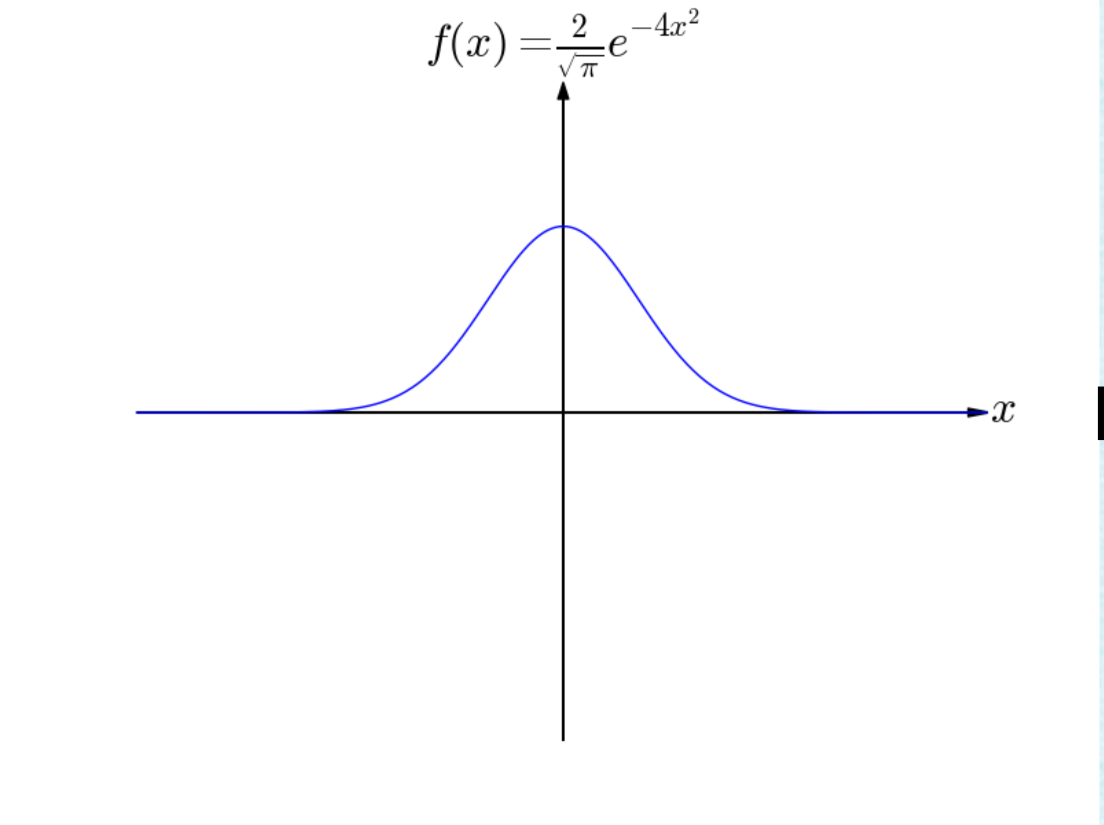


High Frequency Cosine

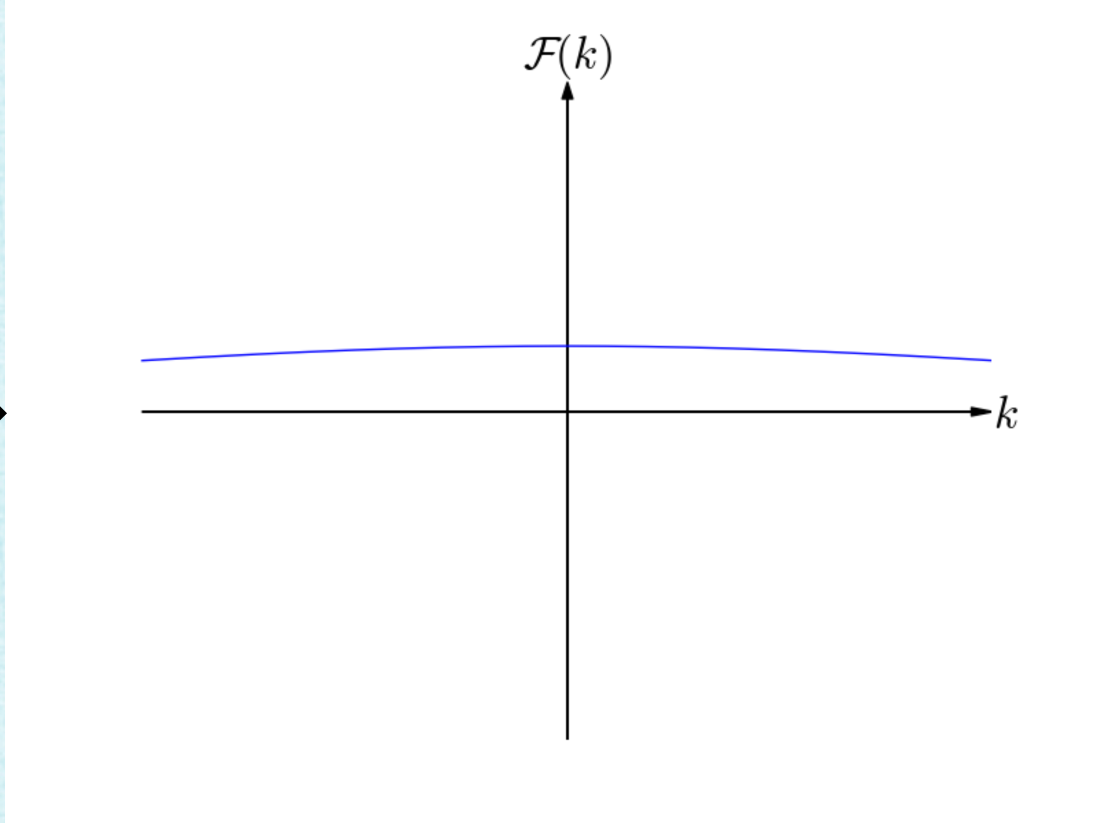




Narrow Gaussian

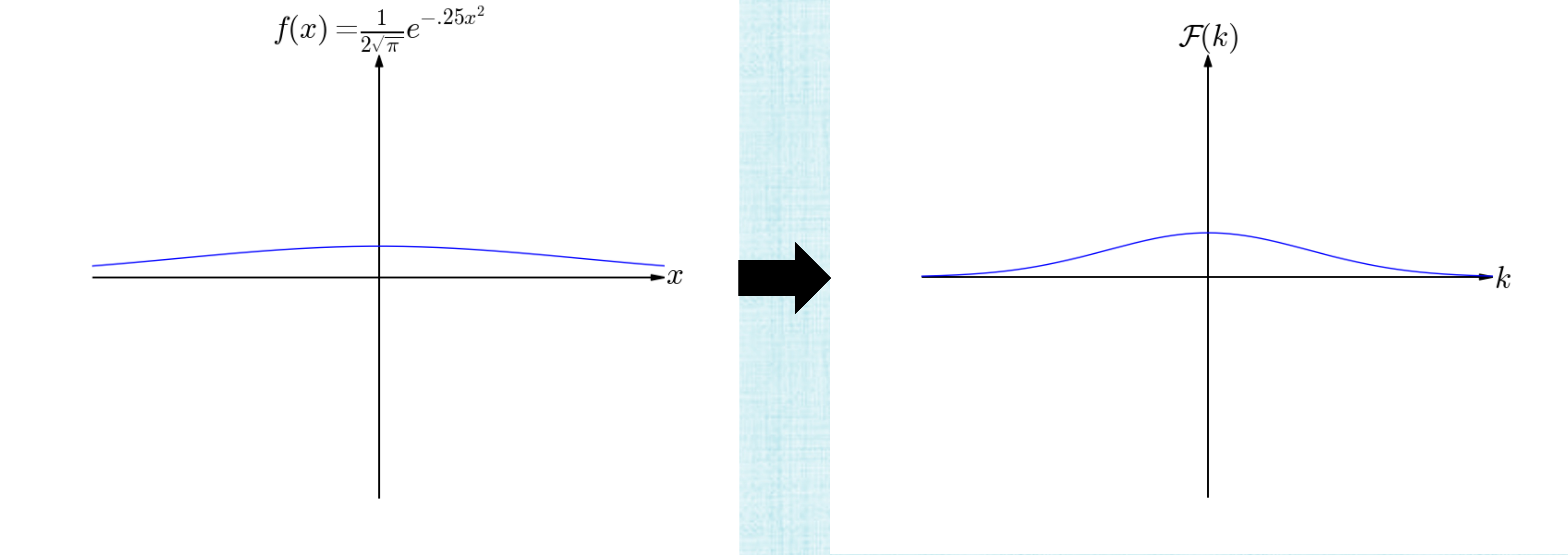


Narrow



Wide

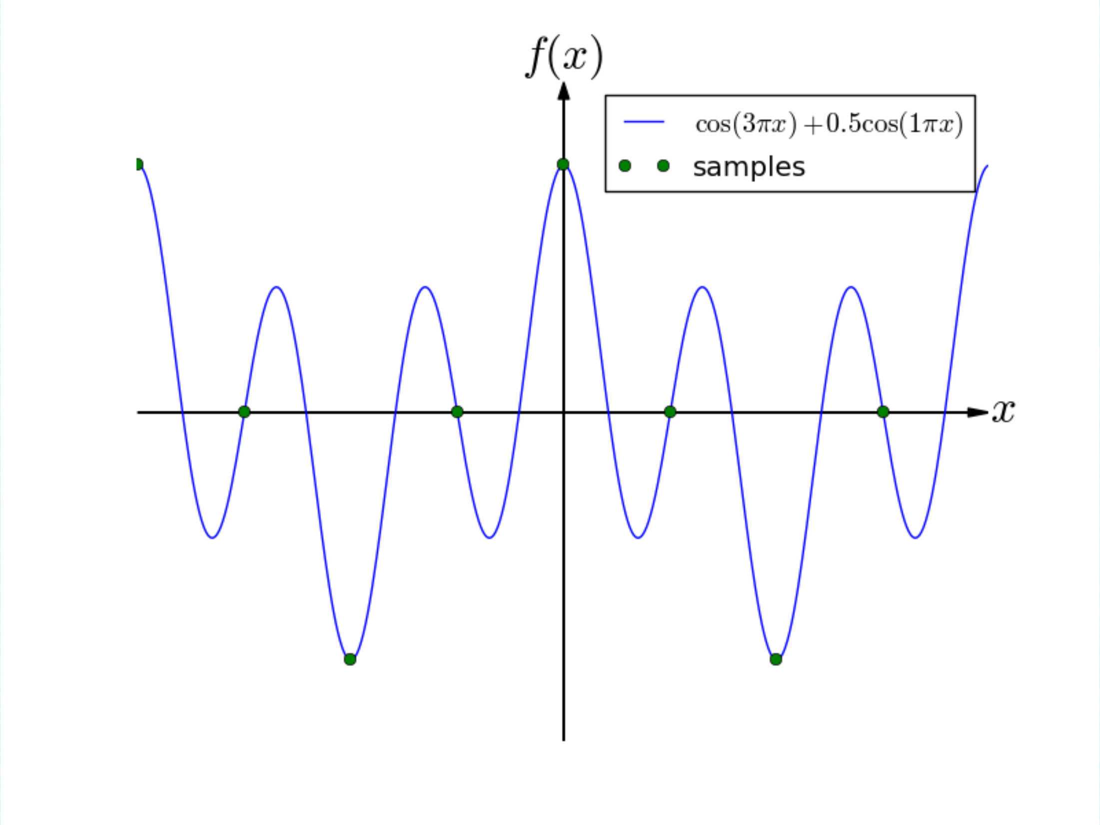
Wider Gaussian



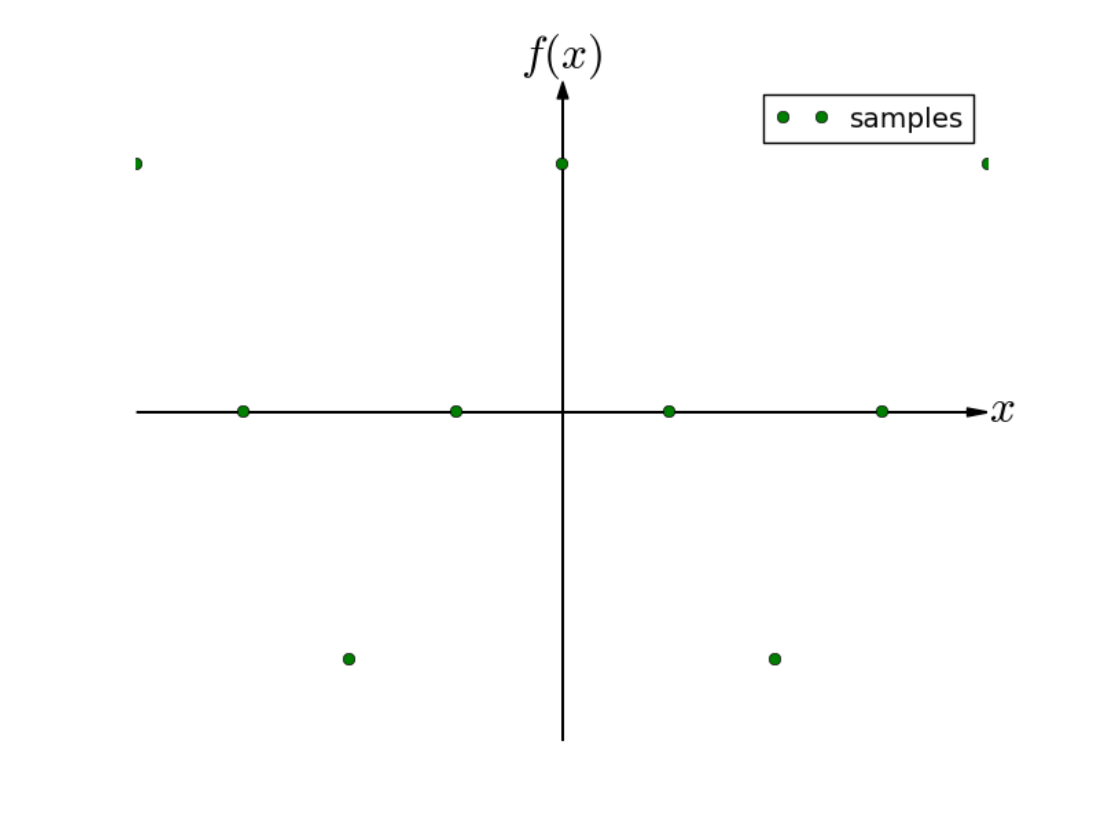
Wider

Narrower

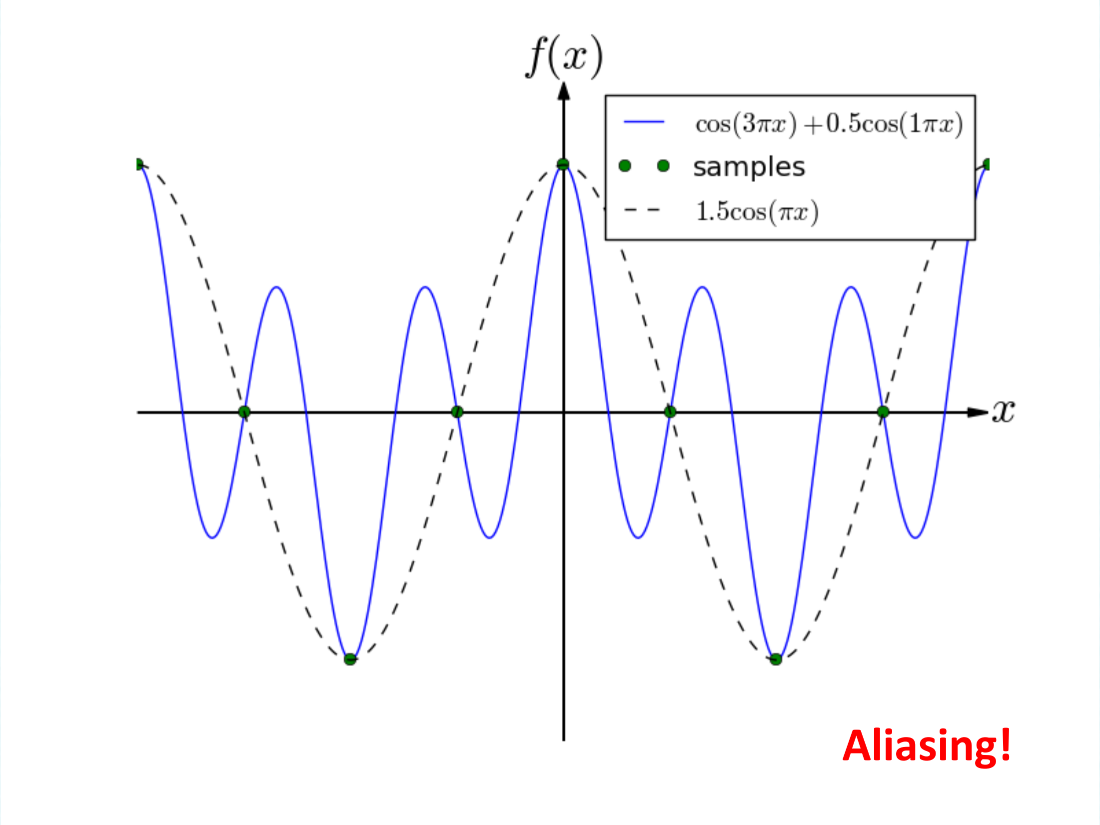
sum of two different cosine functions



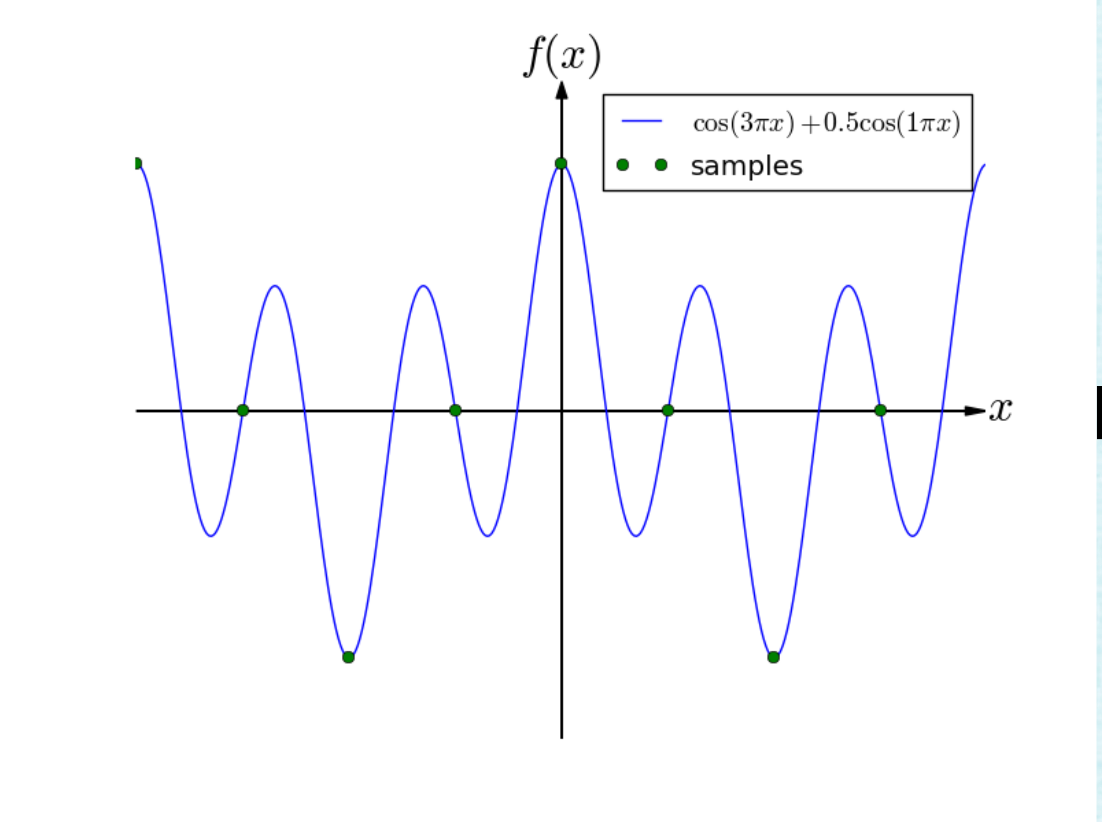
samples

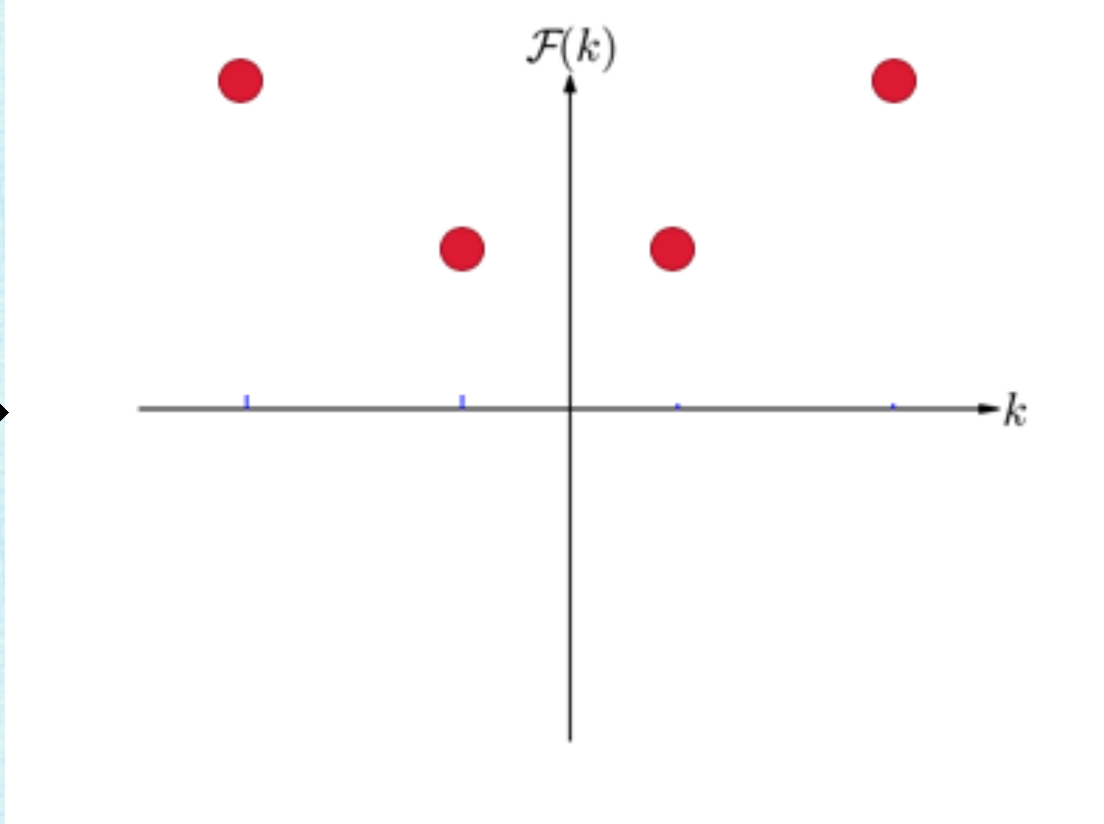


reconstruction

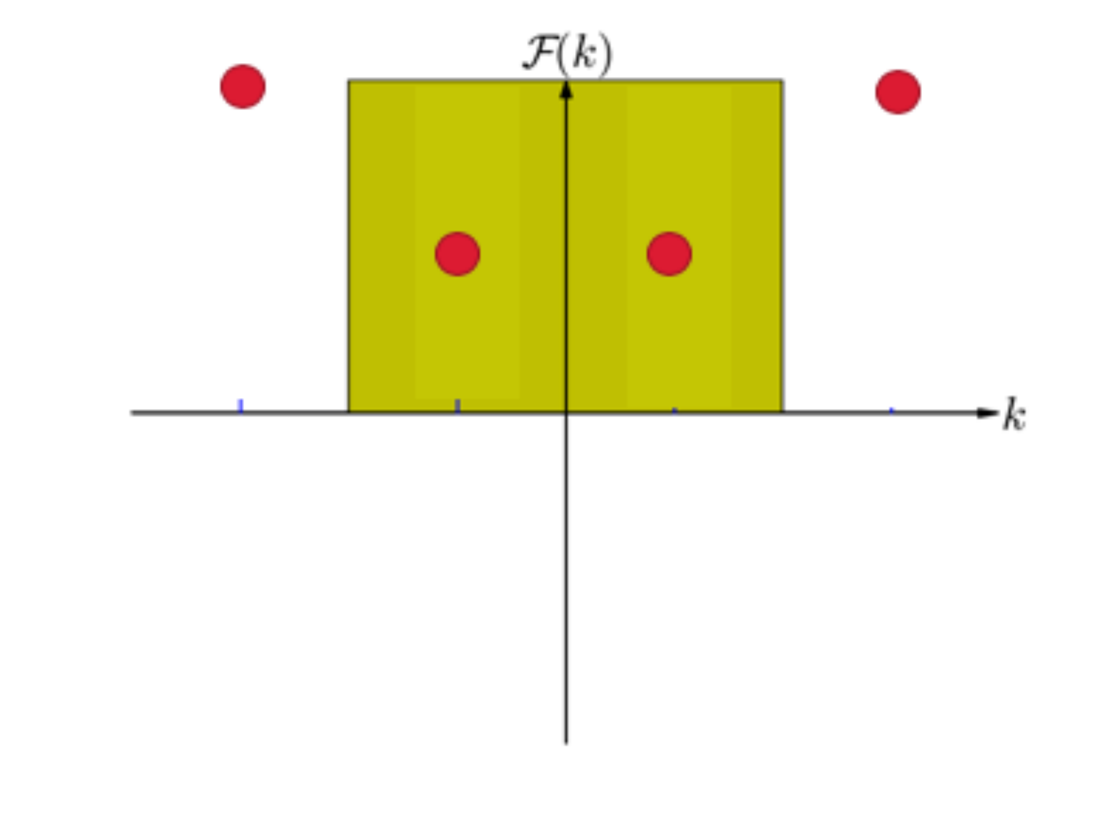


Fourier transform

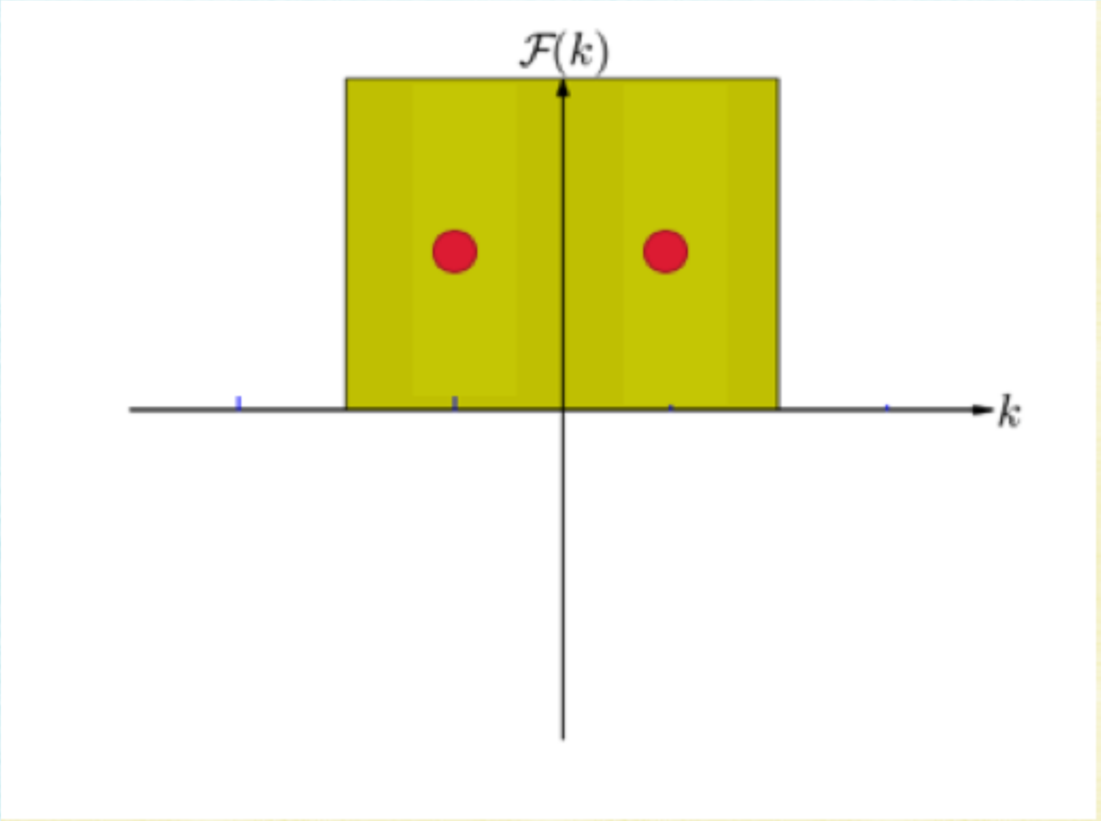




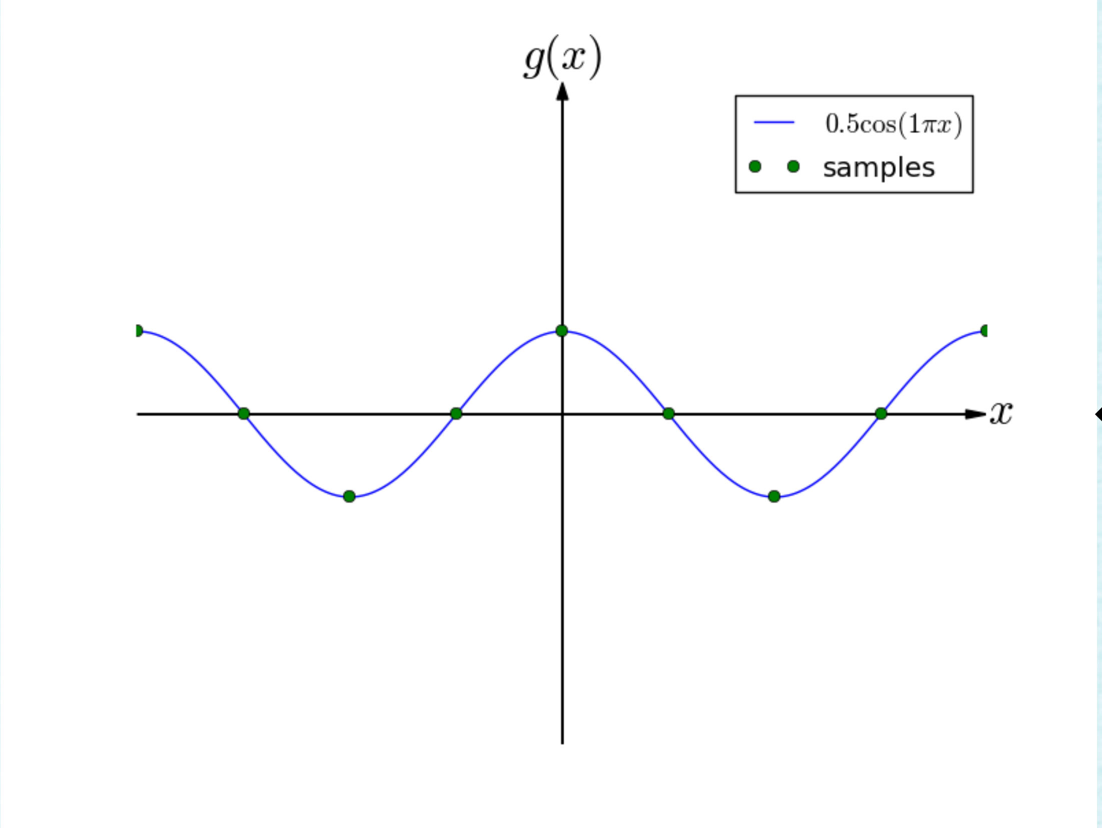
identify Nyquist frequency bounds

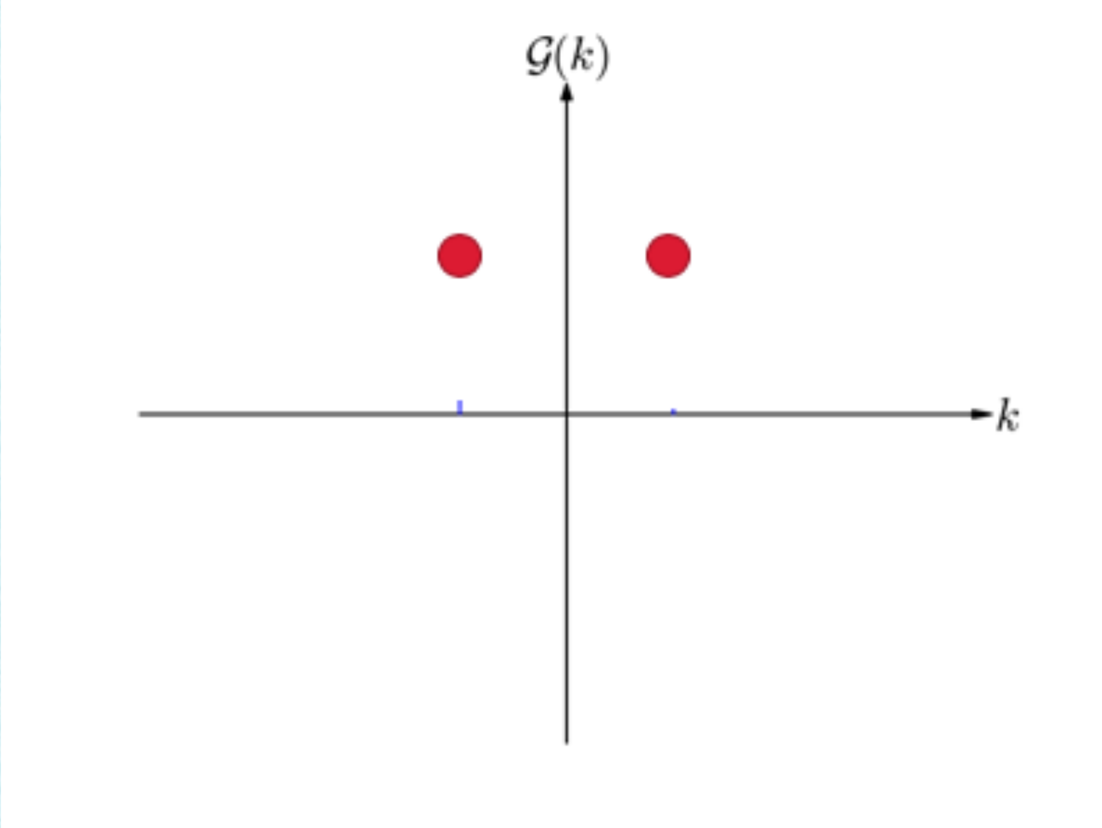


remove the high frequencies

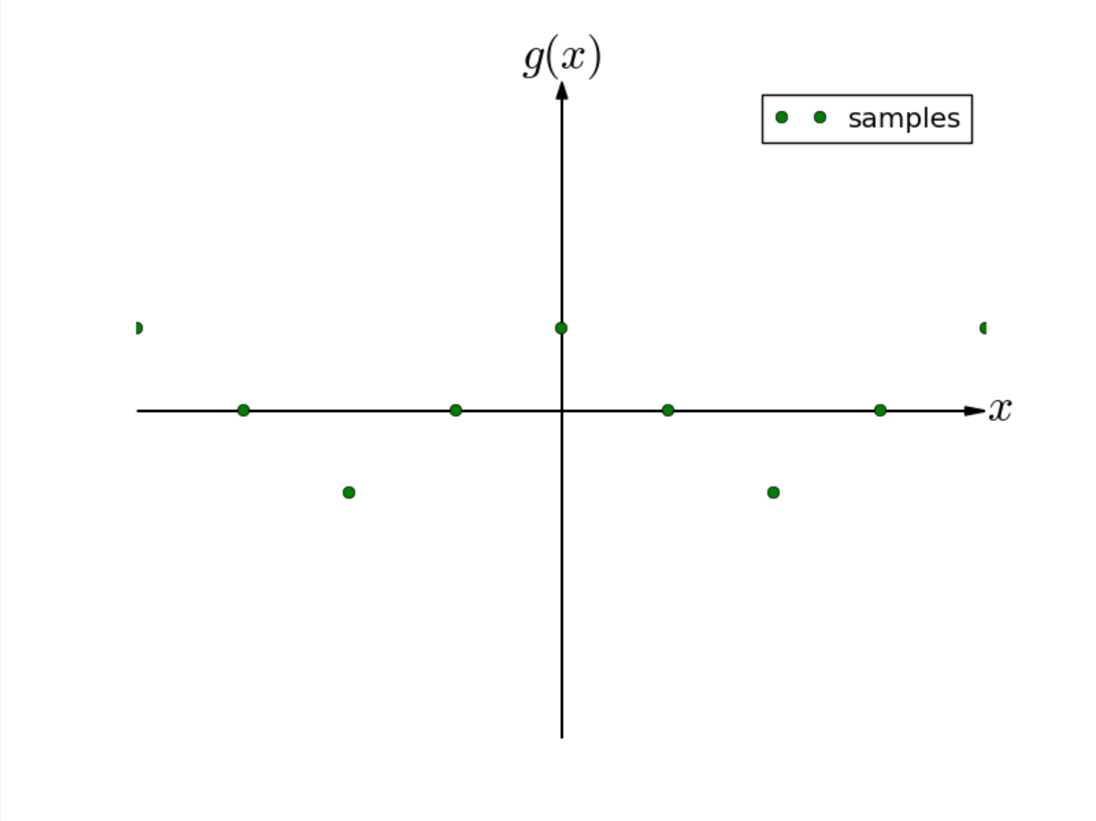


inverse Fourier transform

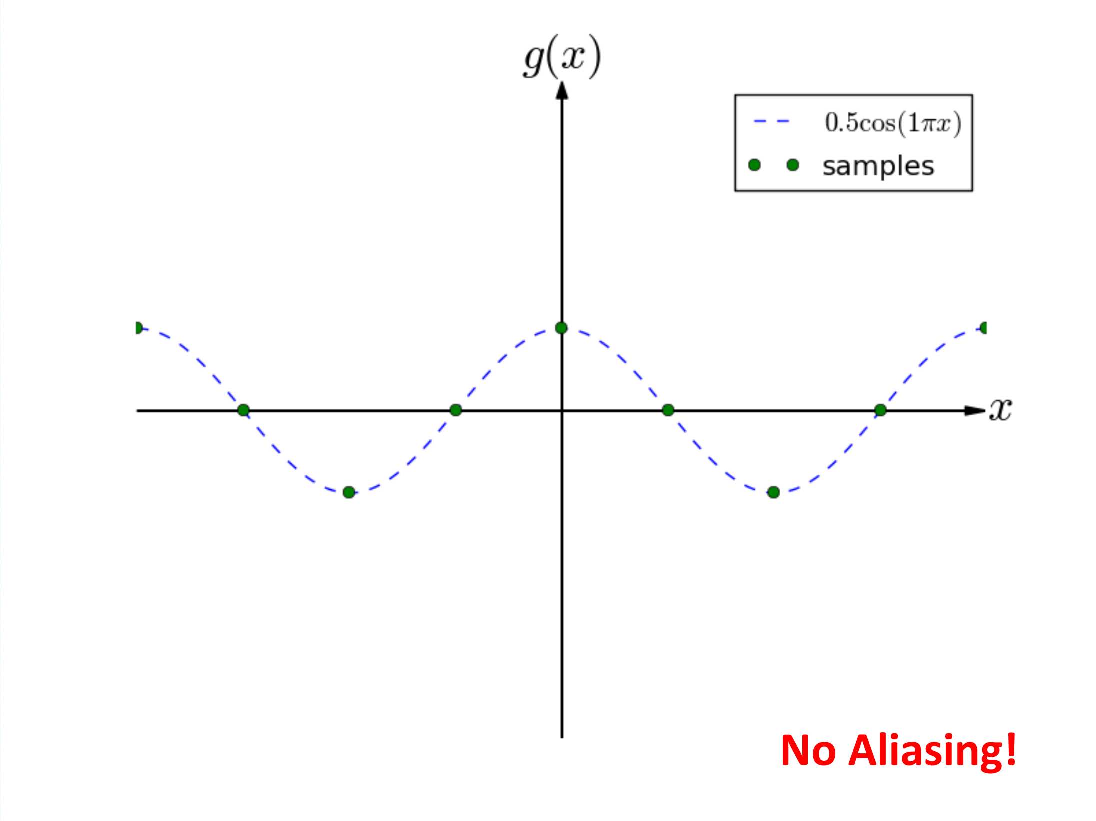




samples



reconstruction



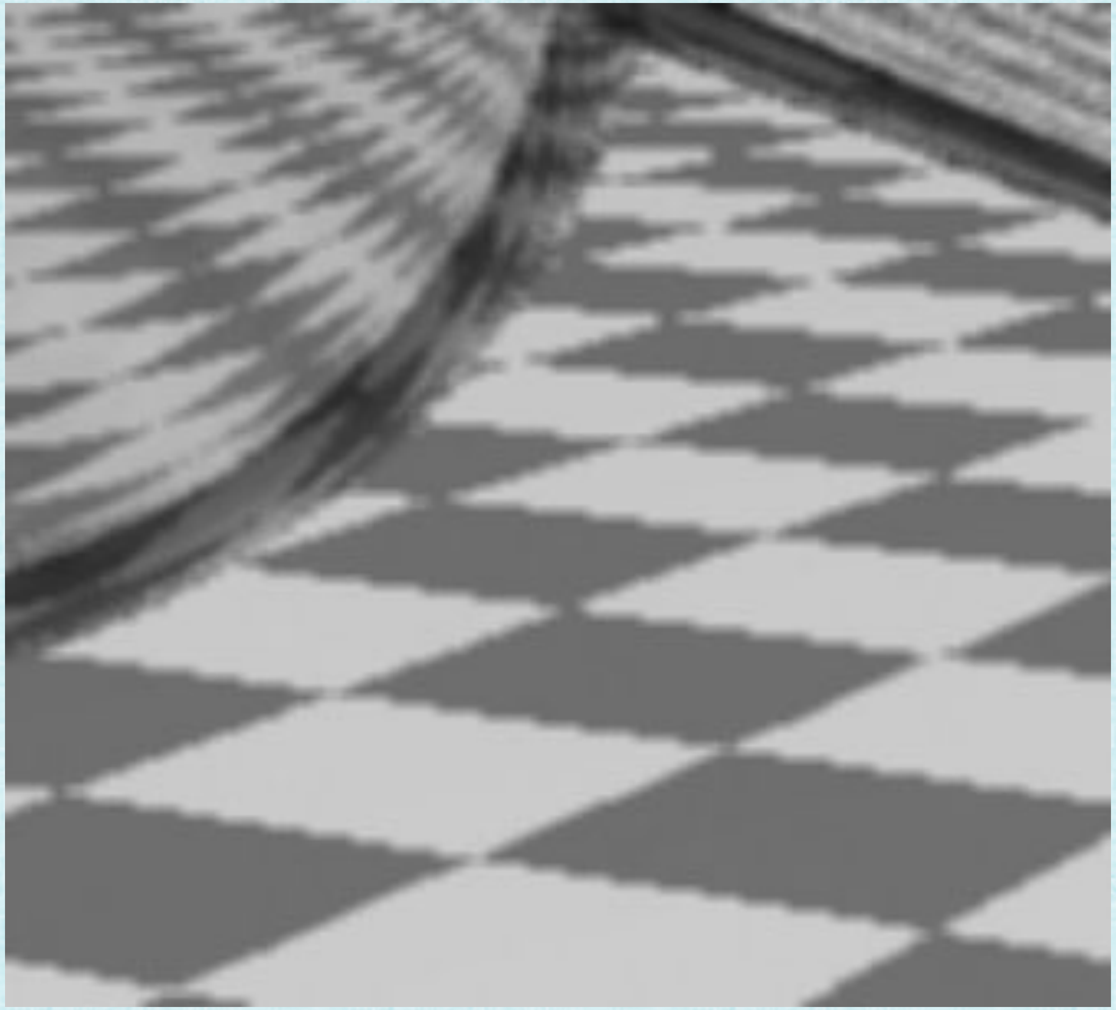
Anti-Aliasing

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

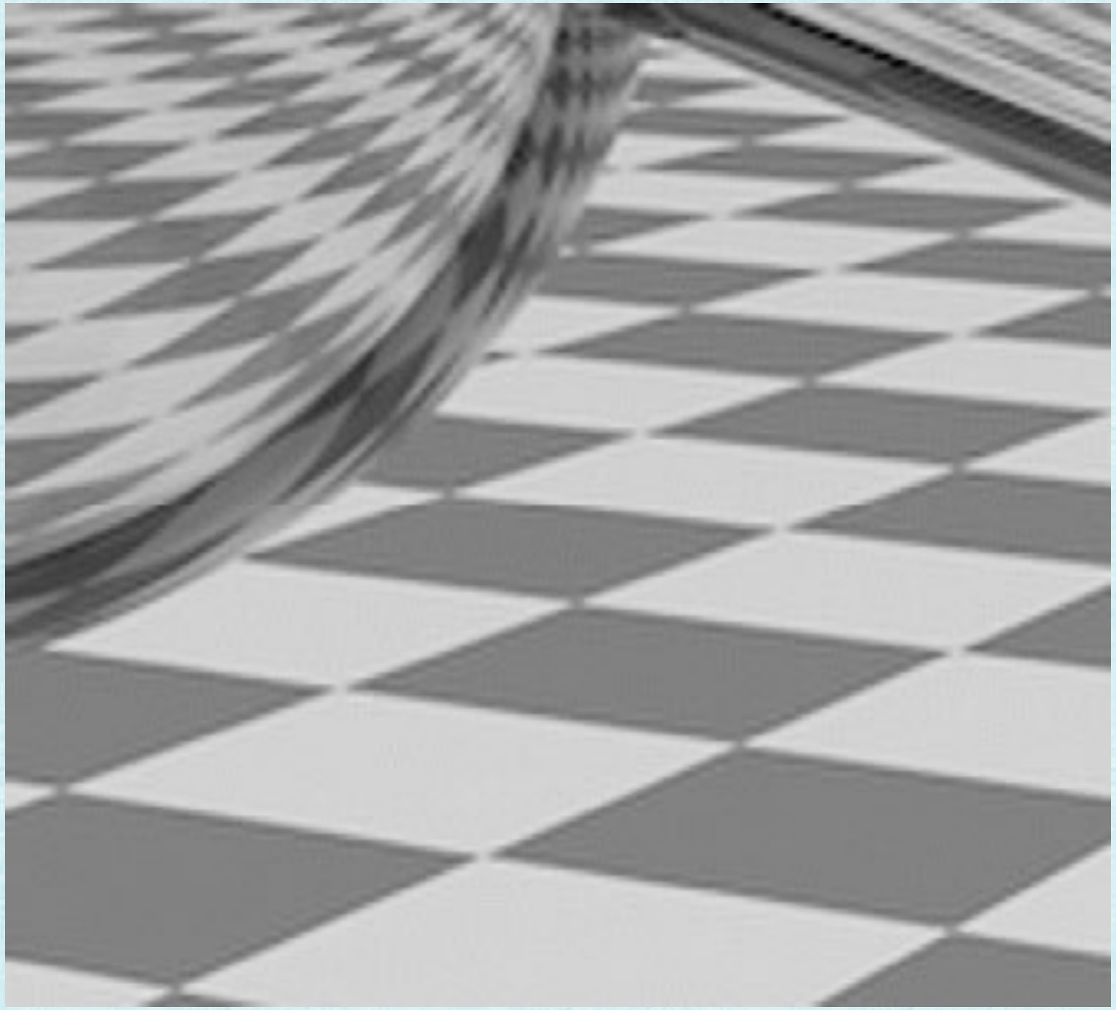
- Remove the high frequencies before sampling (in order to avoid aliasing)

- 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



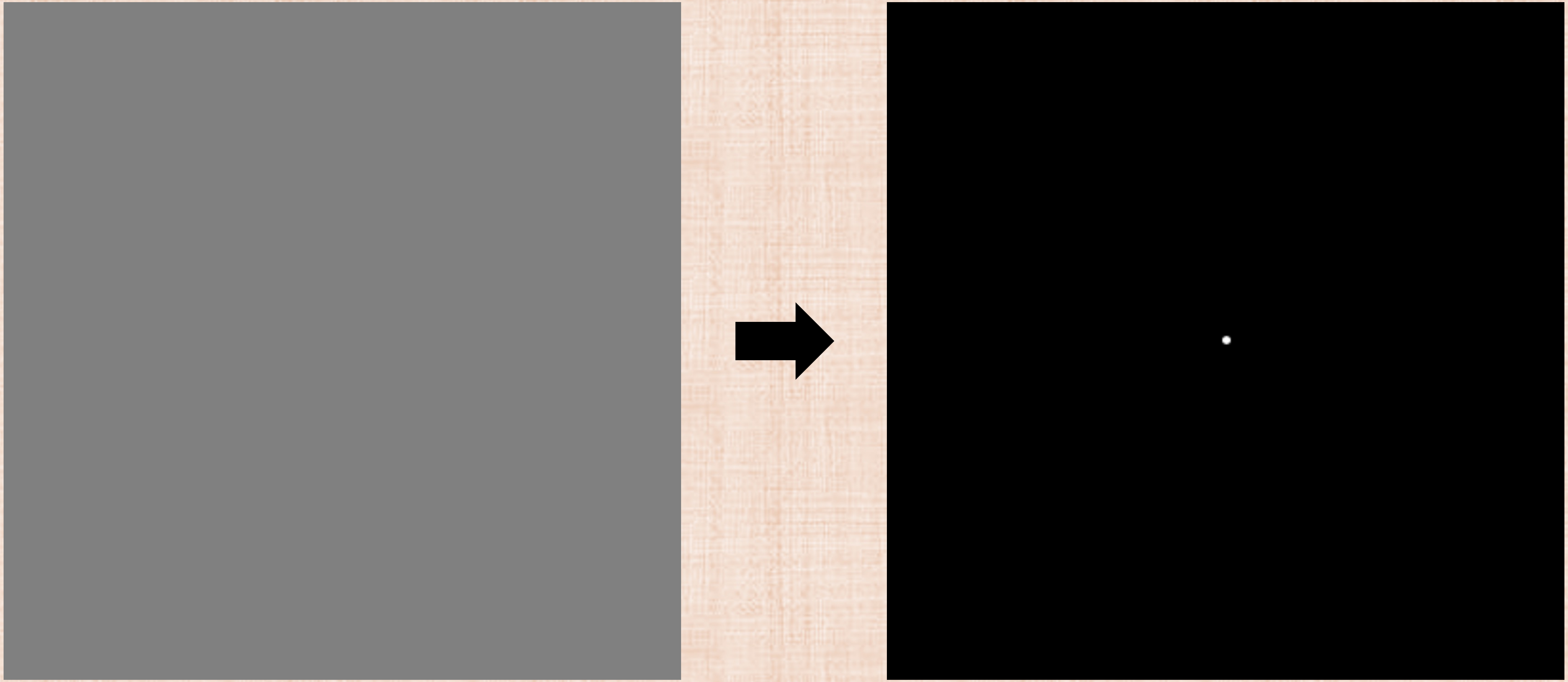
removing high frequencies before sampling

Images

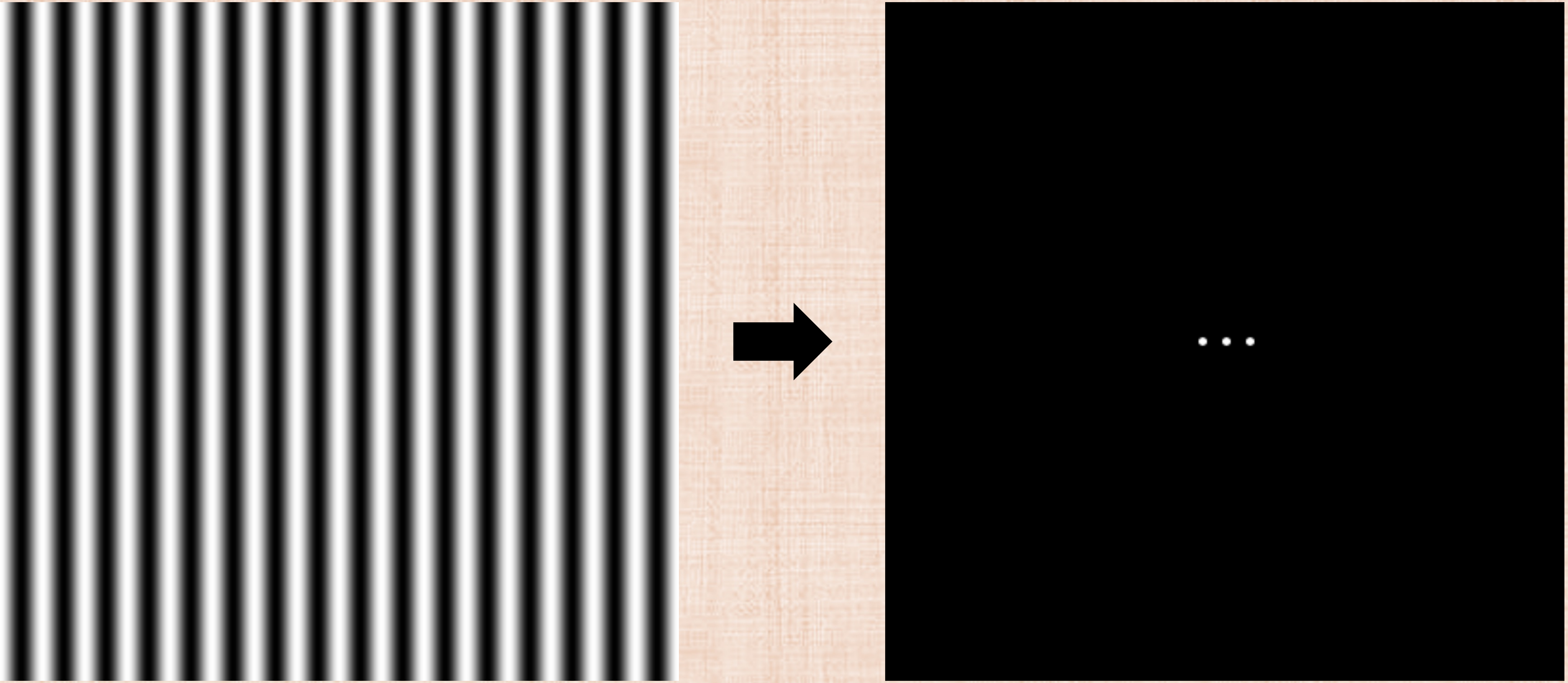
- Images have discrete values (and are not continuous functions)
  - Use a discrete version of the Fourier transform
  - The Fast Fourier Transform (FFT) computes the discrete Fourier transform (and its inverse) in $O(n \log n)$ complexity (where n is the number of samples)
- Images are 2D (not 1D)
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

1. Fourier transform (into the frequency domain)
   - Discrete image values are transformed into another array of discrete values
2. Remove high frequencies
3. Inverse Fourier transform (back out of the frequency domain)

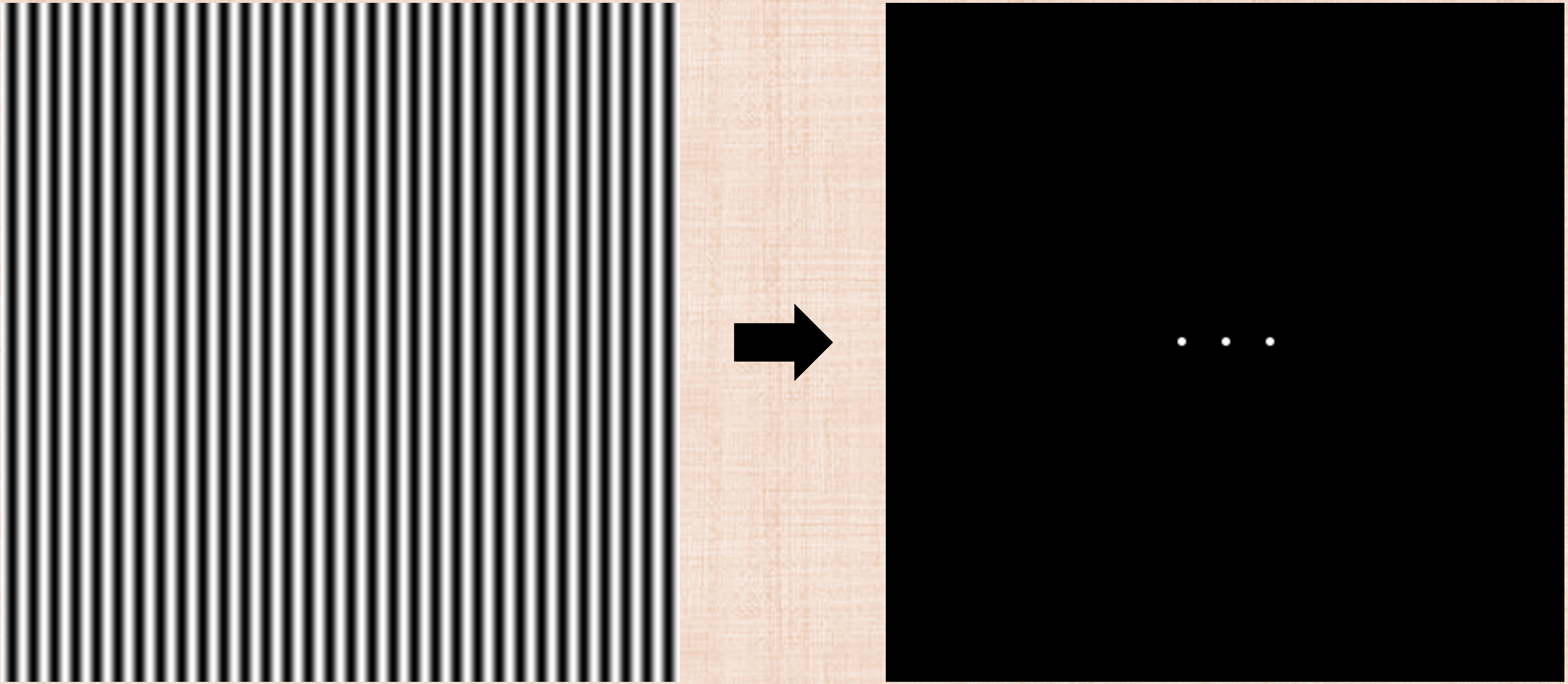
Constant Function



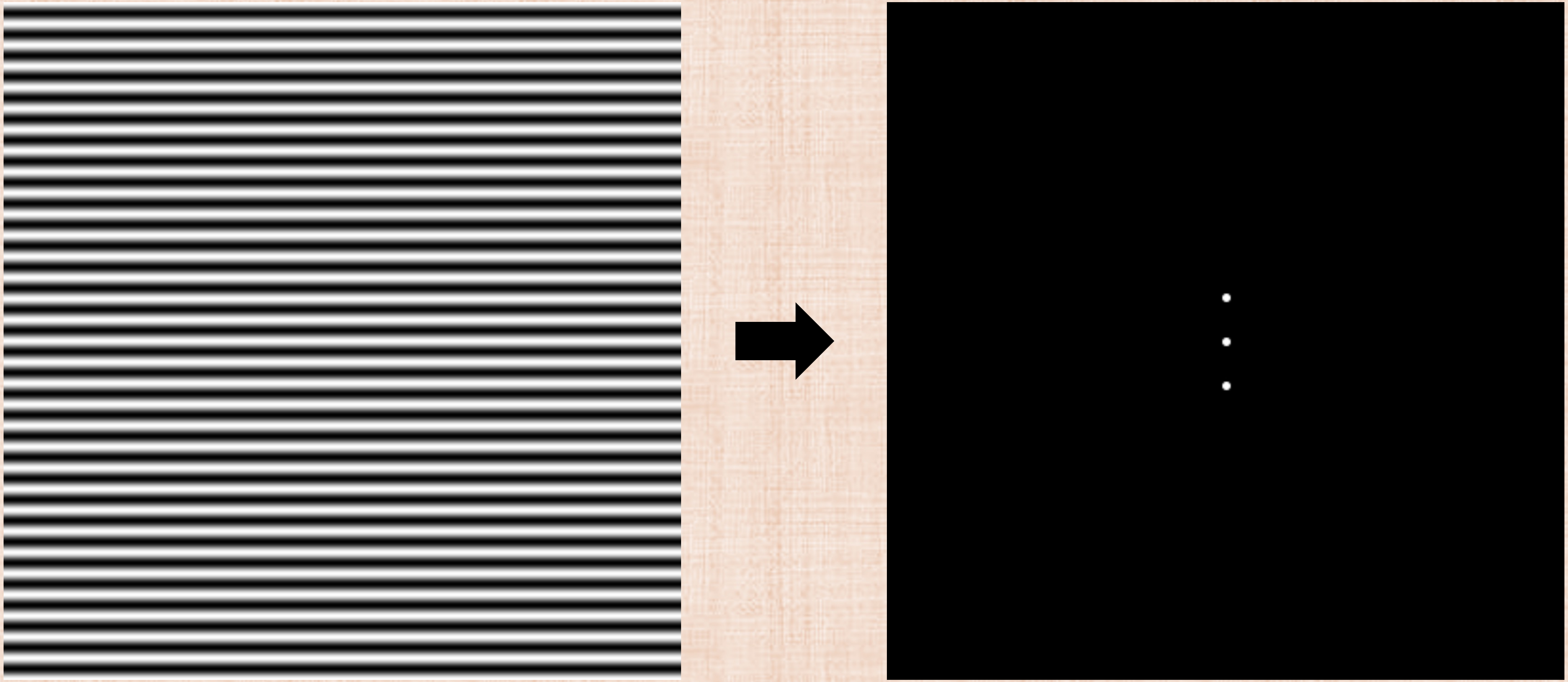
$$\sin(2\pi/32)\, x$$



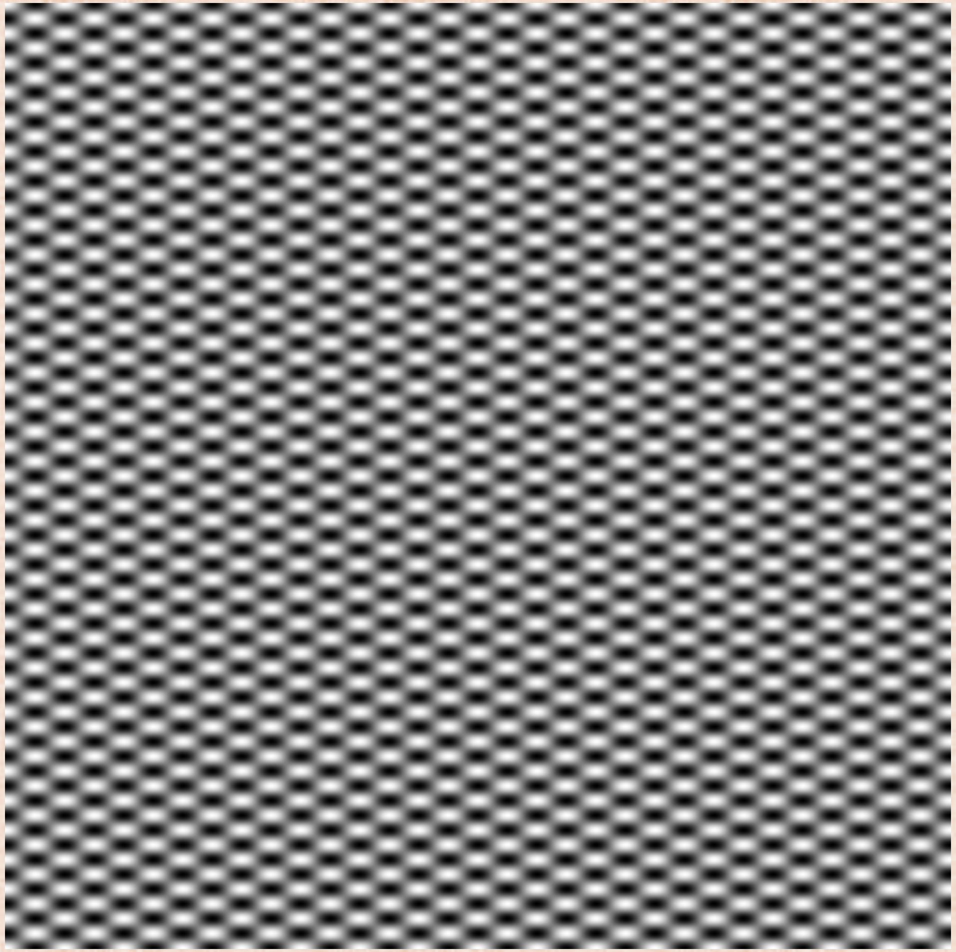
$$\sin(2\pi/16)\,x$$

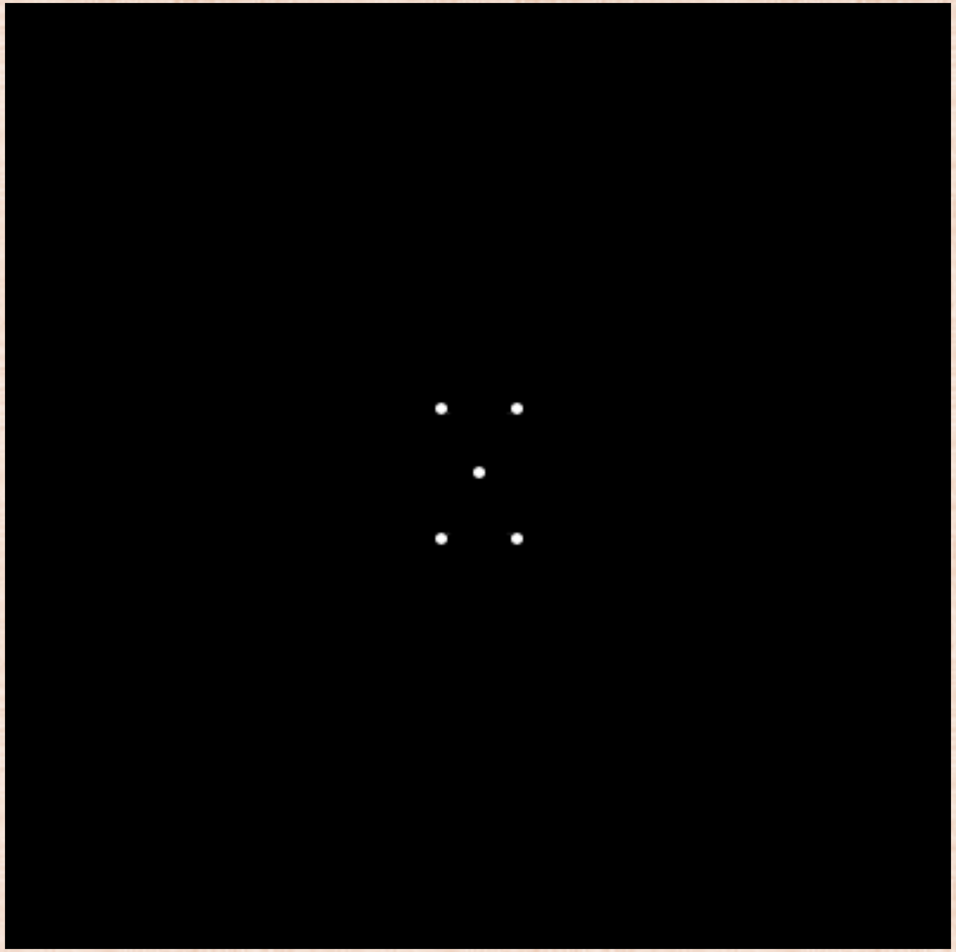


$$\sin(2\pi/16)\,y$$

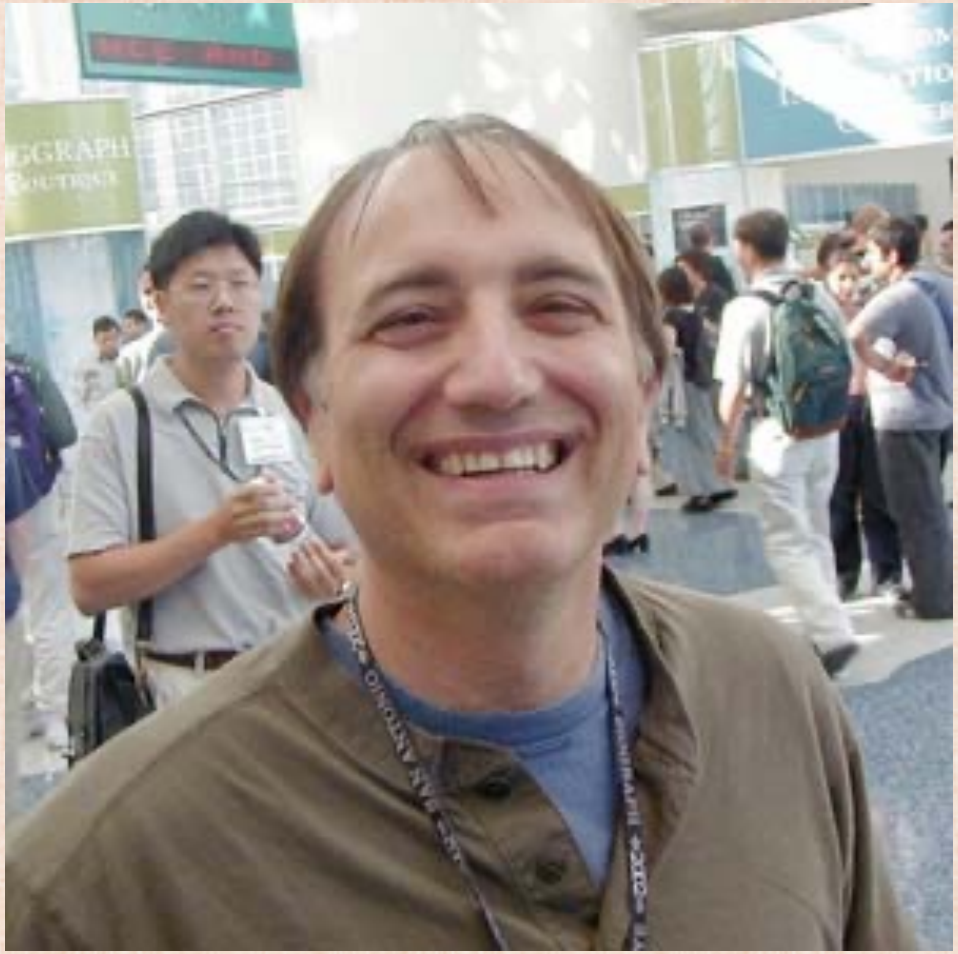


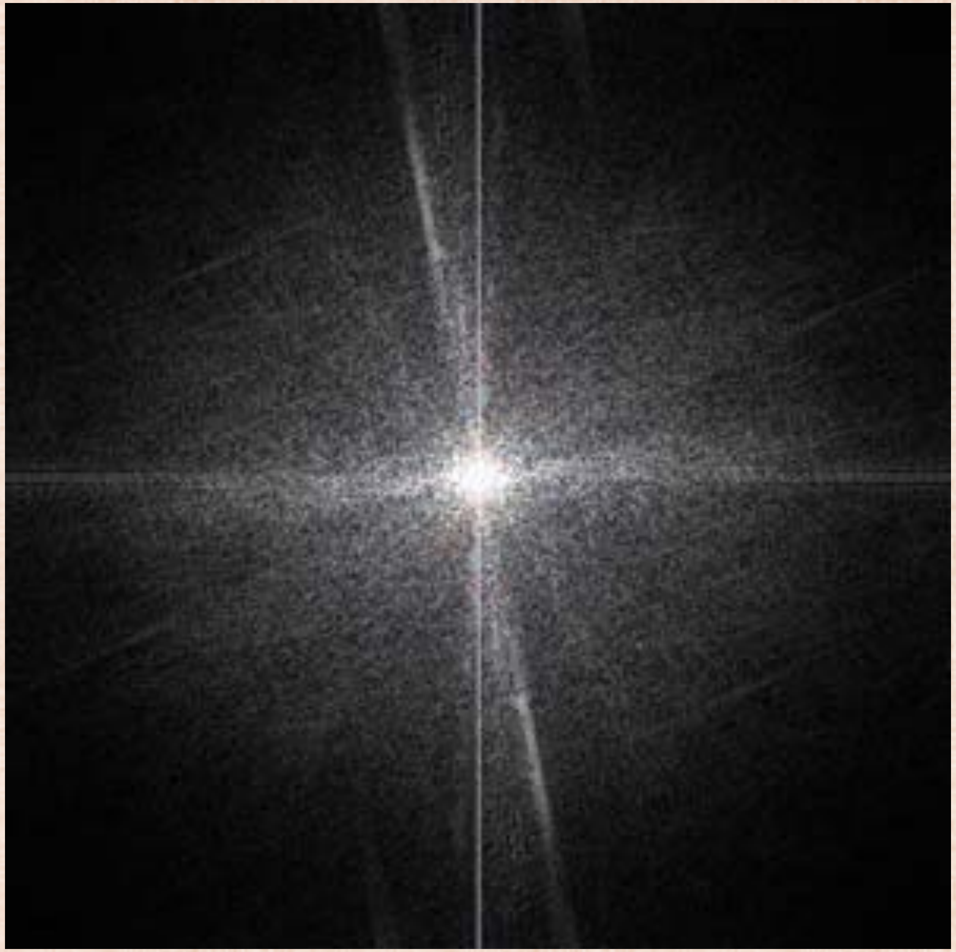
$$\sin(2\pi/32)\,x \;*\, \sin(2\pi/16)\,y$$



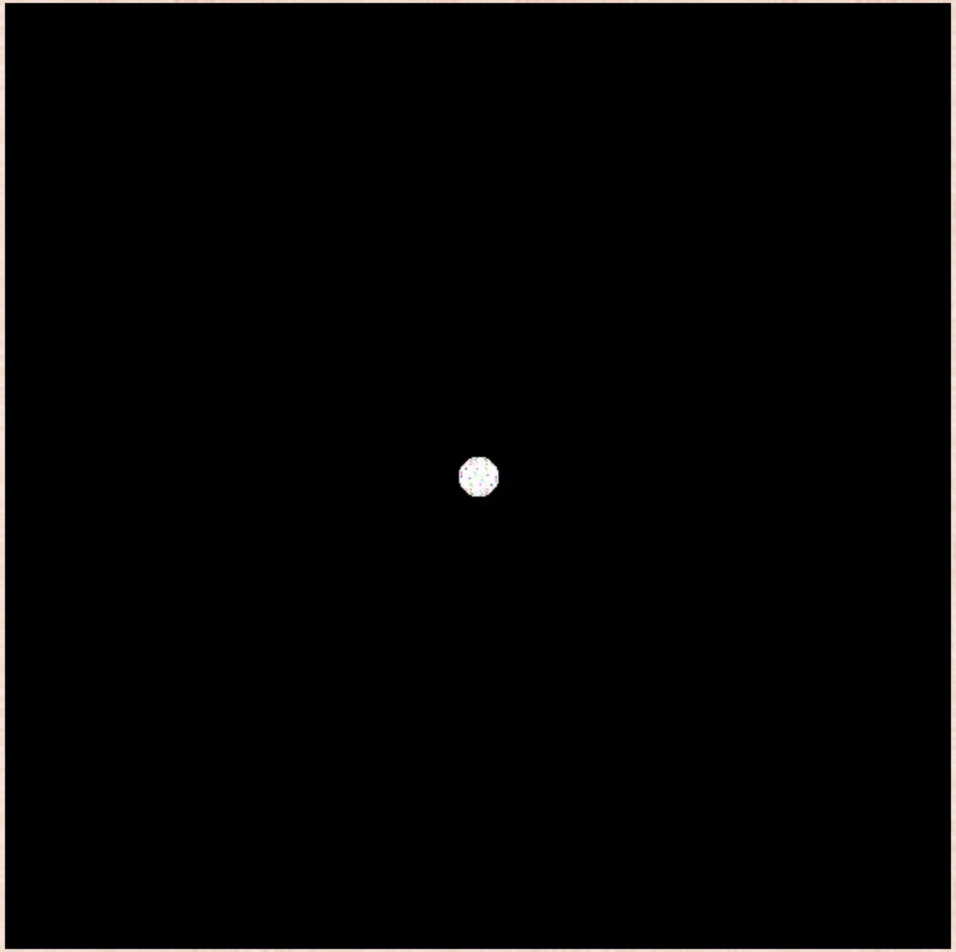


An obvious star!

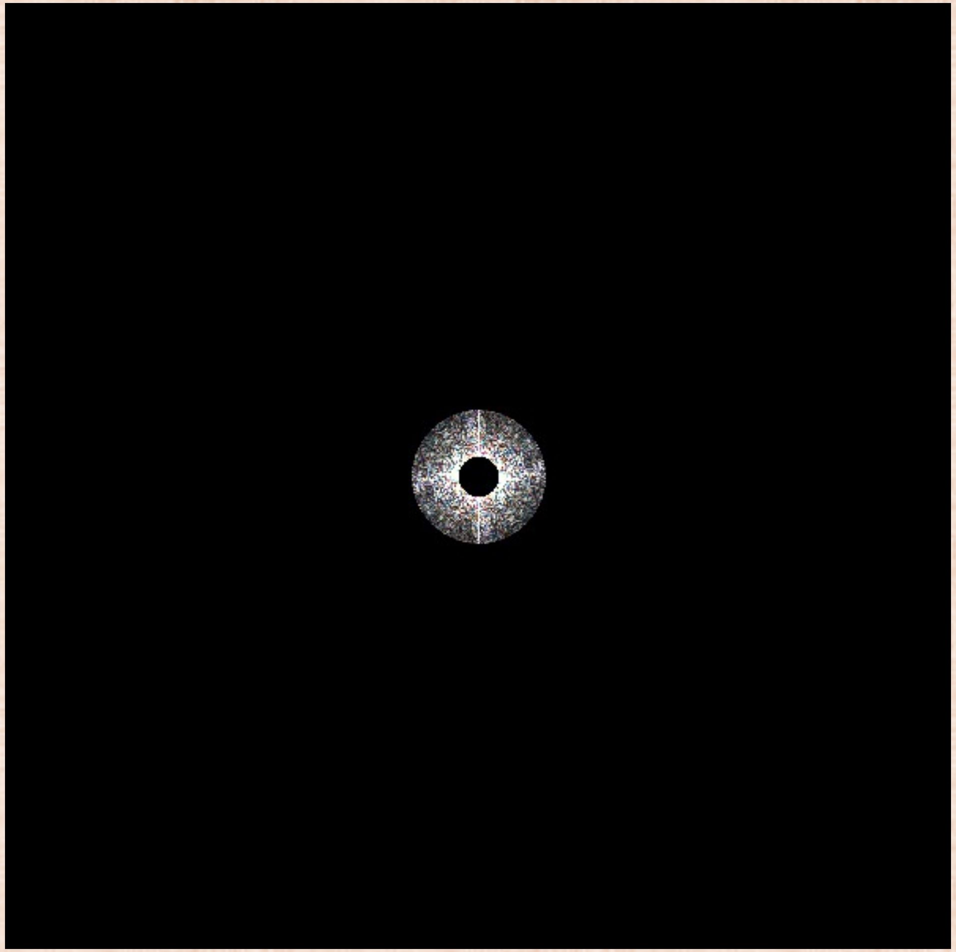




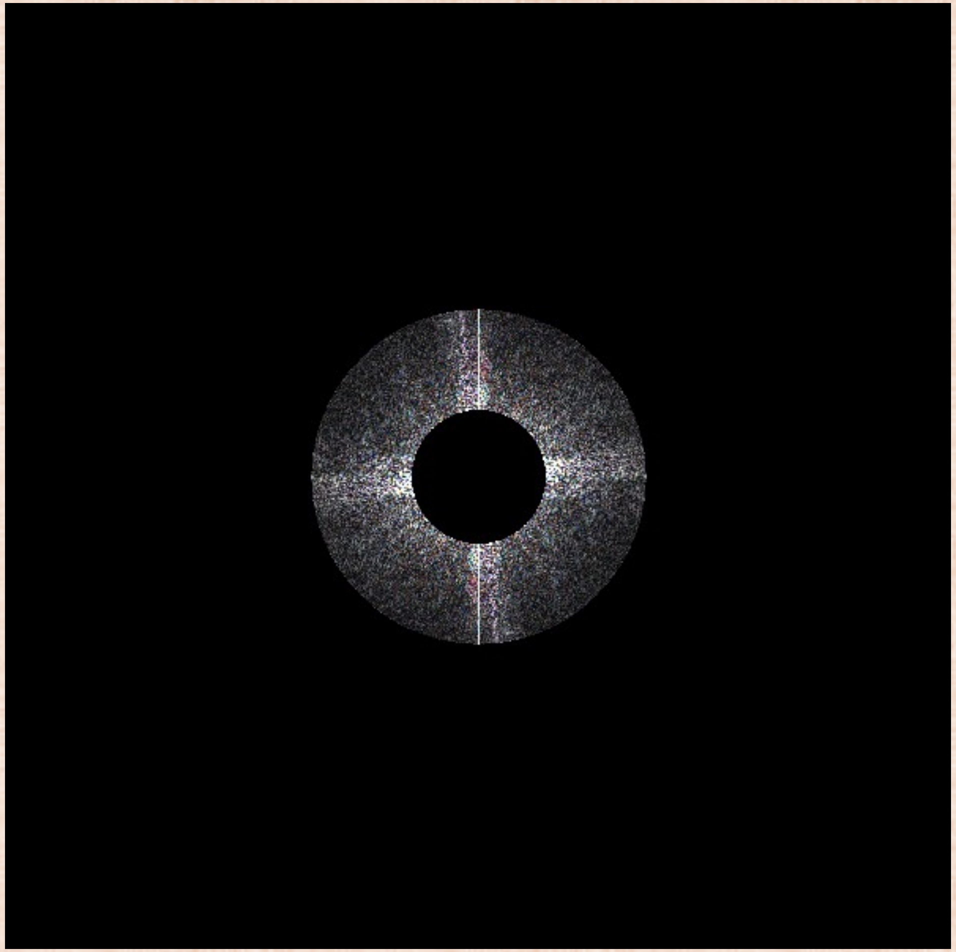
lowest frequencies



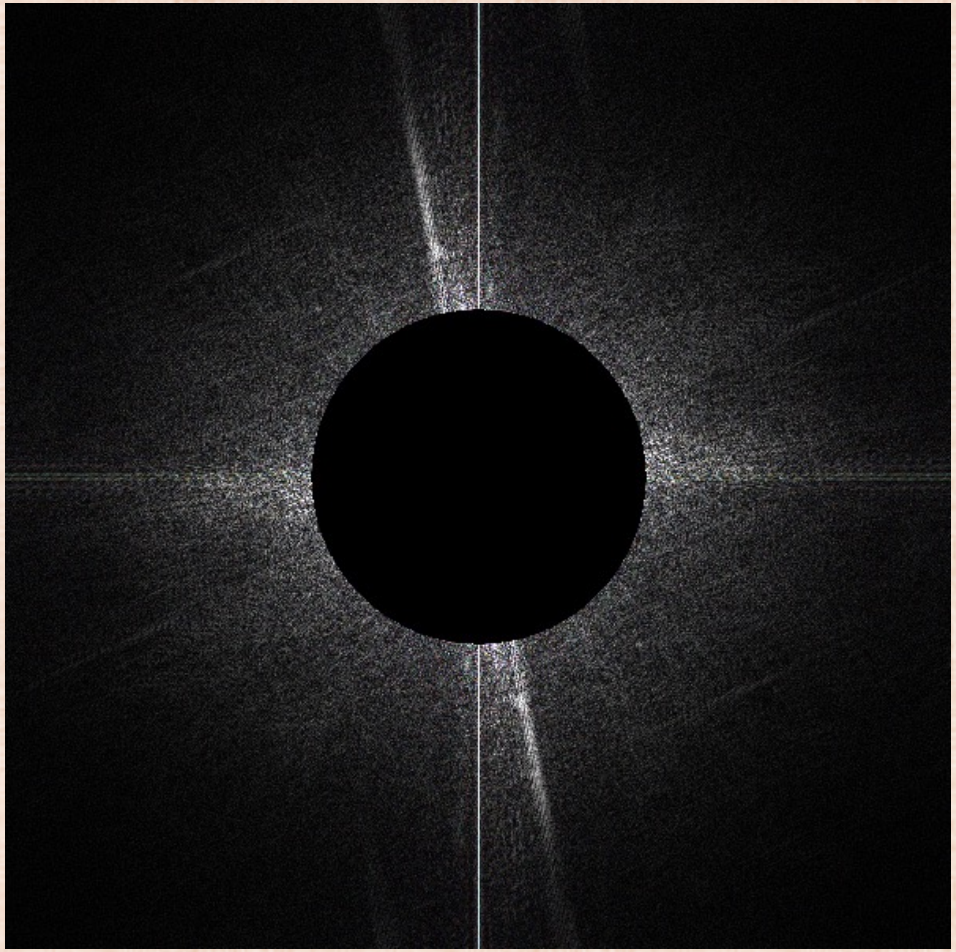
intermediate frequencies



(larger) intermediate frequencies



highest frequencies (edges)



Convolution

- Let f and g be functions in the spatial domain (e.g. images), and $F(f)$ and $F(g)$ be transformations of f and g into the frequency domain
  - In our prior examples: f was the image (to the left), $F(f)$ was the frequency domain version of the image (to the right)

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

- This entire process is called the convolution of f and g:

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

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

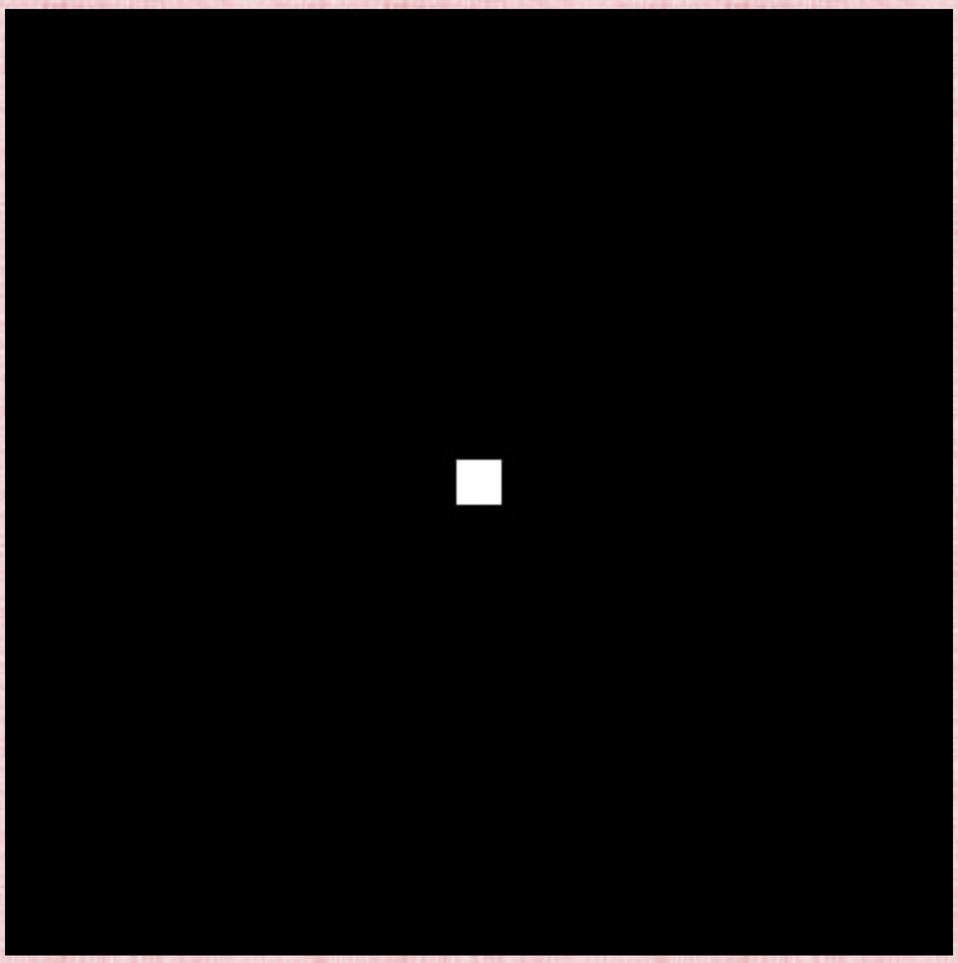
Box Filter

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

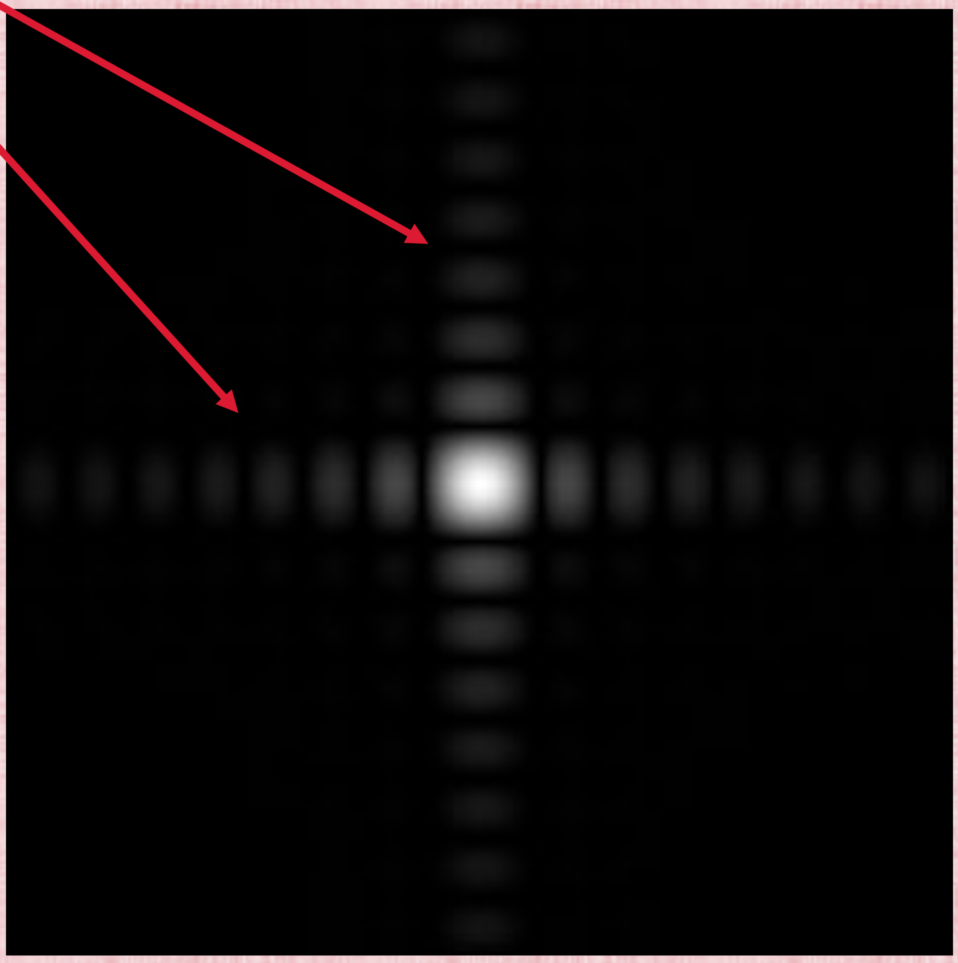
$^1/_{16}$	$^1/_{16}$	$^1/_{16}$	$^1/_{16}$
$^1/_{16}$	$^1/_{16}$	$^1/_{16}$	$^1/_{16}$
$^1/_{16}$	$^1/_{16}$	$^1/_{16}$	$^1/_{16}$
$^1/_{16}$	$^1/_{16}$	$^1/_{16}$	$^1/_{16}$

Filters Most (but not all) High Frequencies

g

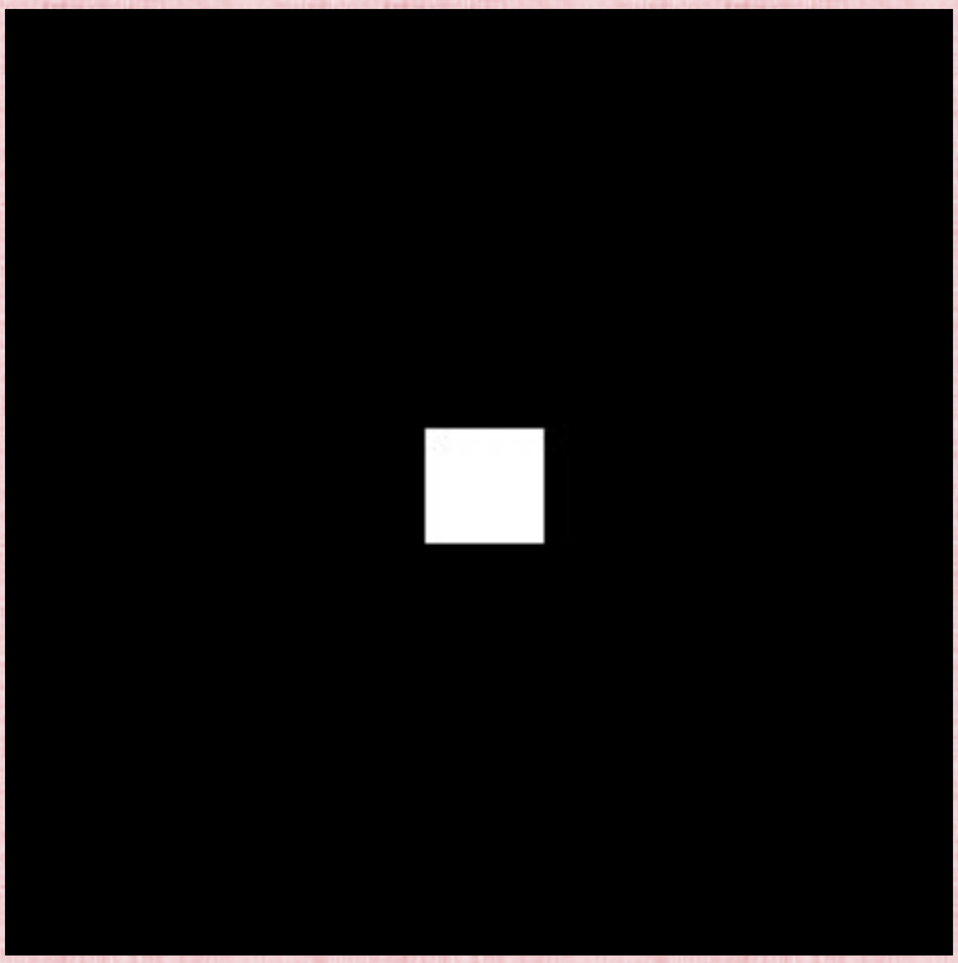


$F(g)$



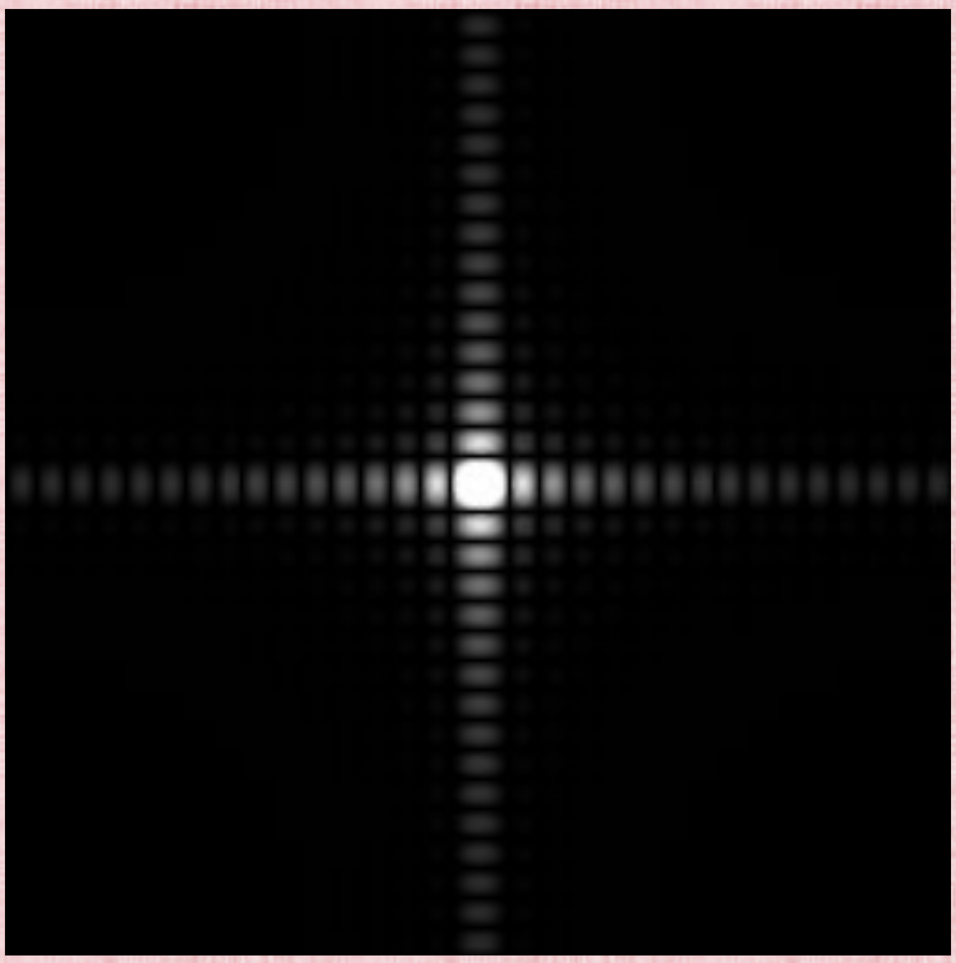
Wider Box Filter

g



more expensive convolution integral

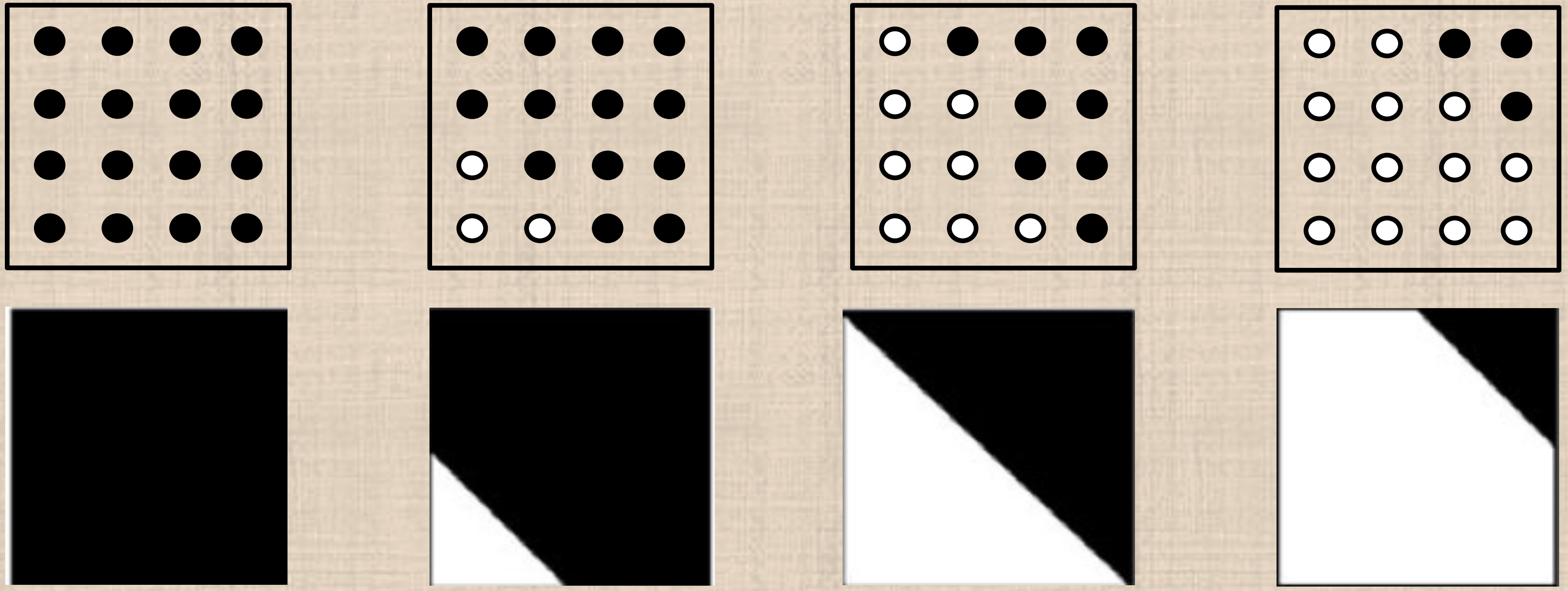
$F(g)$



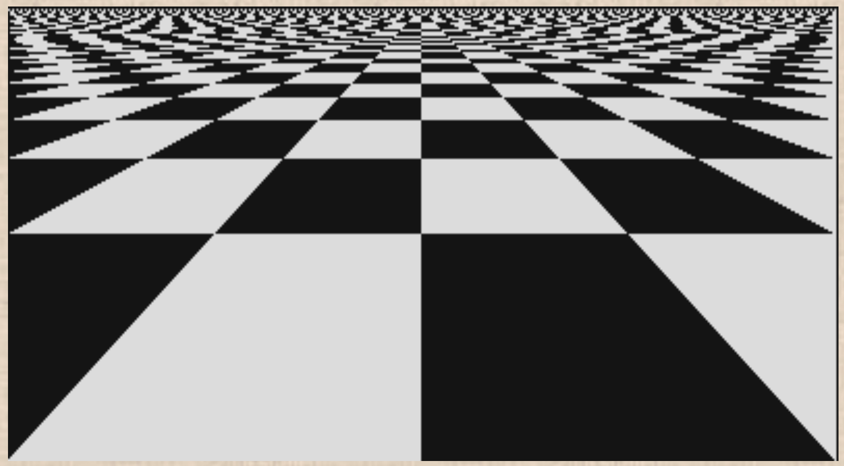
removes more of the high frequencies

Super-Sampling

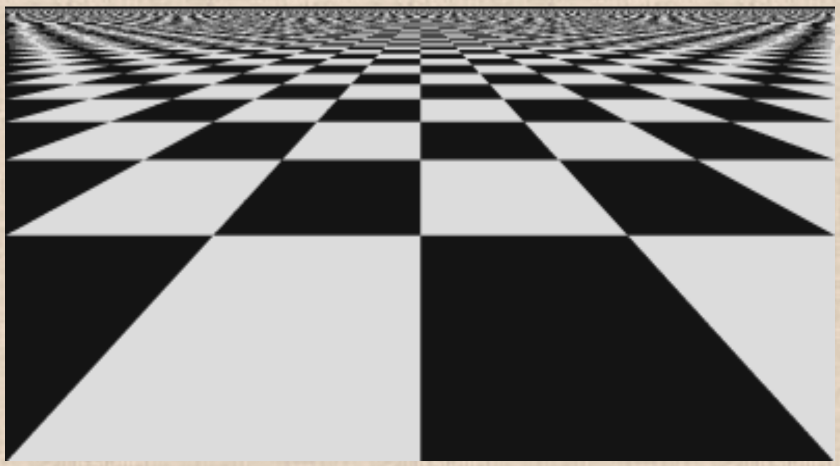
- Collect extra information/samples (in each pixel), and average the result (e.g. with a box filter)
  - E.g. render a 100 by 100 image with 4 by 4 super-sampling (equivalent to rendering a 400 by 400 image)
  - This properly represents (without aliasing) frequencies up to 4 times higher (than the original image could)
  - Apply a 4 by 4 box filter aiming to remove as much of those extra 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 strategies (instead of uniform super-sampling)



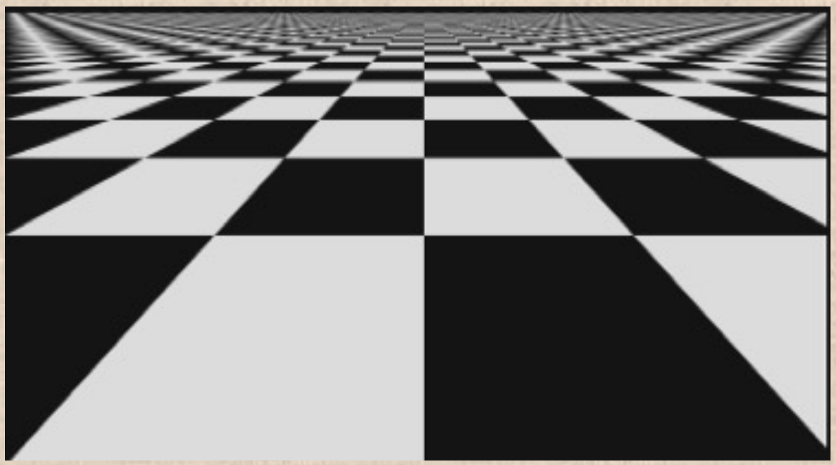
Super-Sampling



Point Sampling

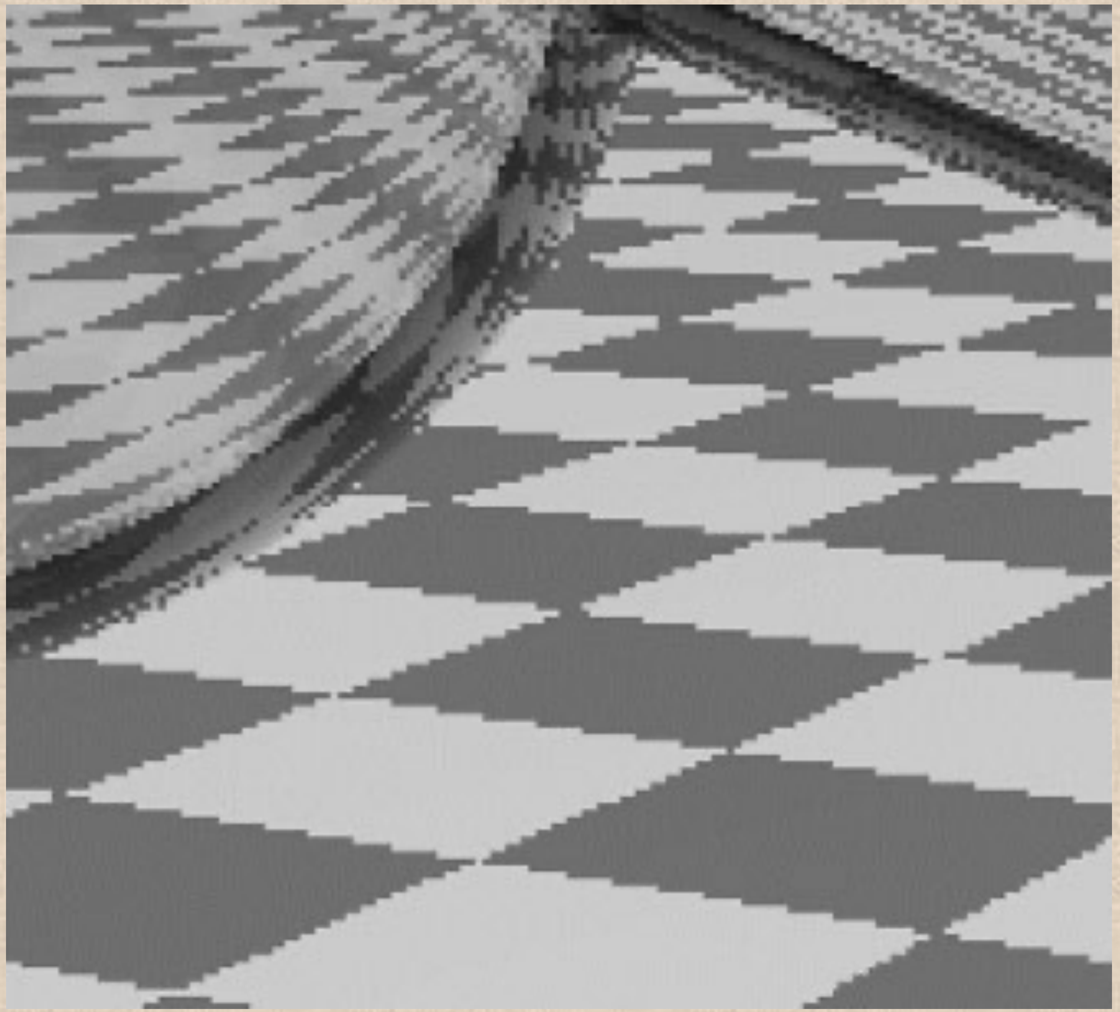


4 by 4 Super-Sampling

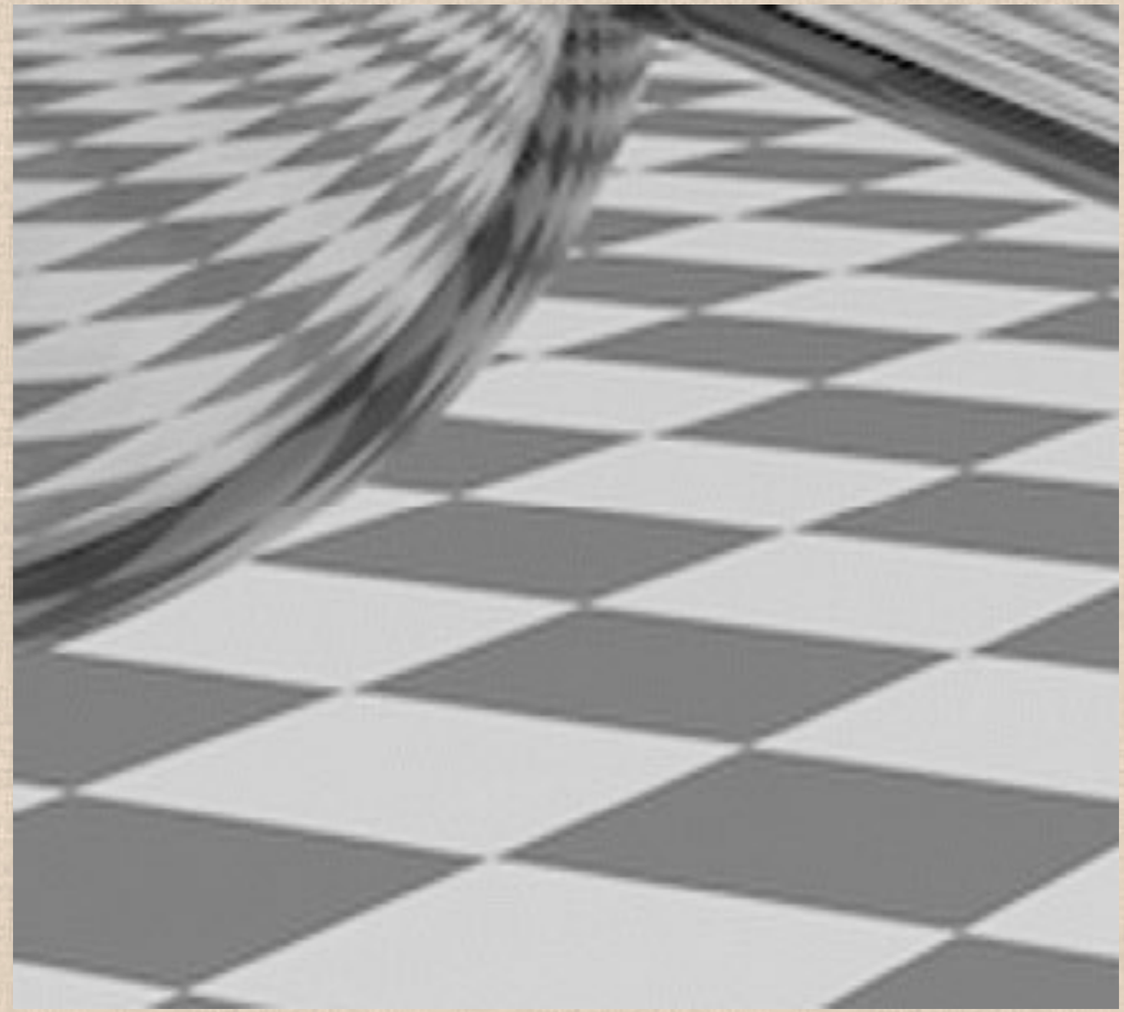


Exact Area Coverage

Super-Sampling



Jaggies



Anti-Aliased