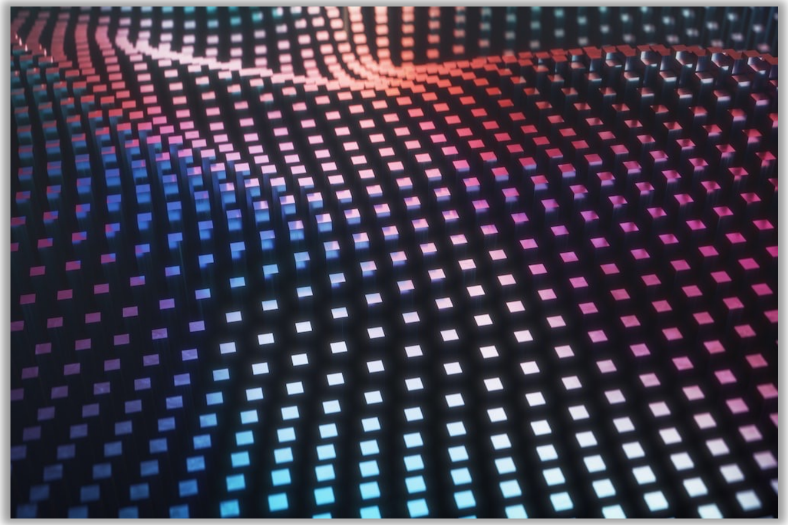
Review of Sampling, Deconvolution, Linear Systems



CSC2529

David Lindell

University of Toronto

cs.toronto.edu/~lindell/teaching/2529

*slides adapted from Gordon Wetzstein, Yannis Gkioulekas, and Fredo Durand

Announcements

- HW 2 due Wednesday 4/10
- HW3 is out, due 18/10
- Problem session for HW3 tomorrow

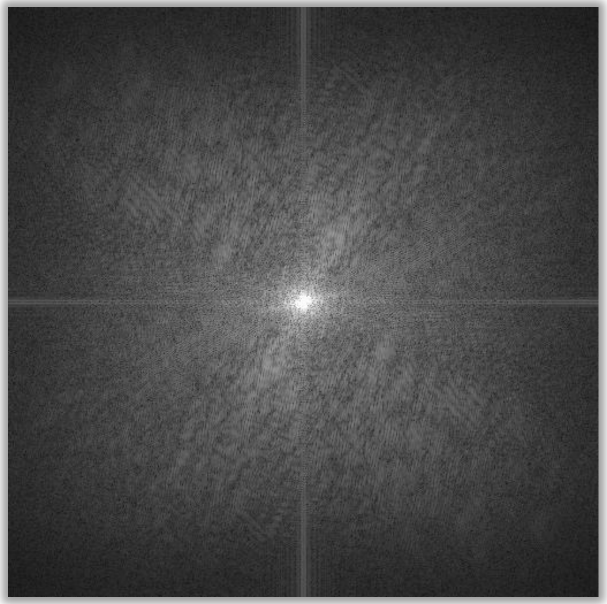
- See website for all office hours/problem session dates

Outline

- Fourier transform review
- Fourier transforms in imaging
- Image filtering, anti-aliasing, and deconvolution
- Linear systems review

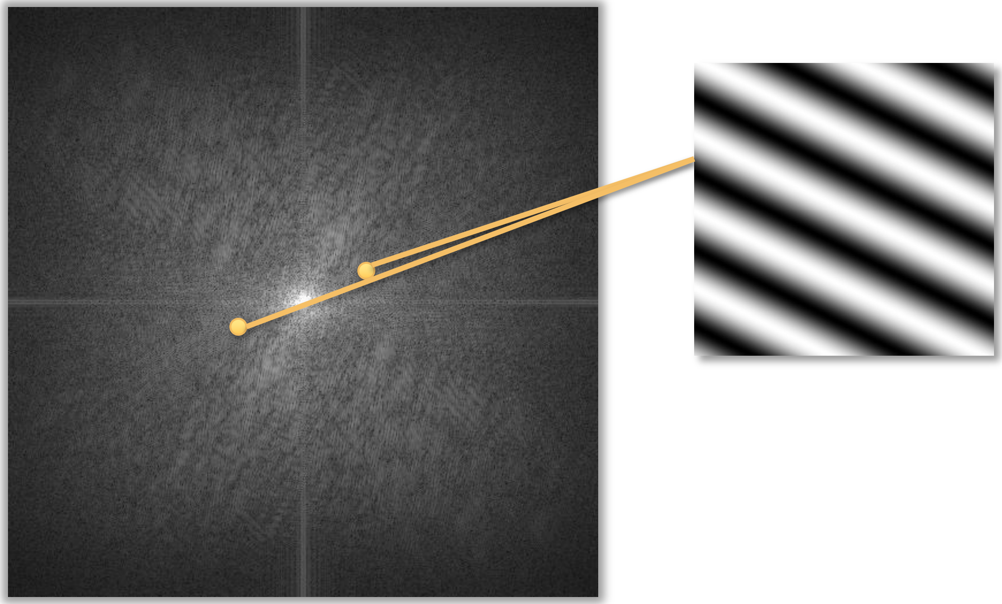
Fourier Transform

- What is this?



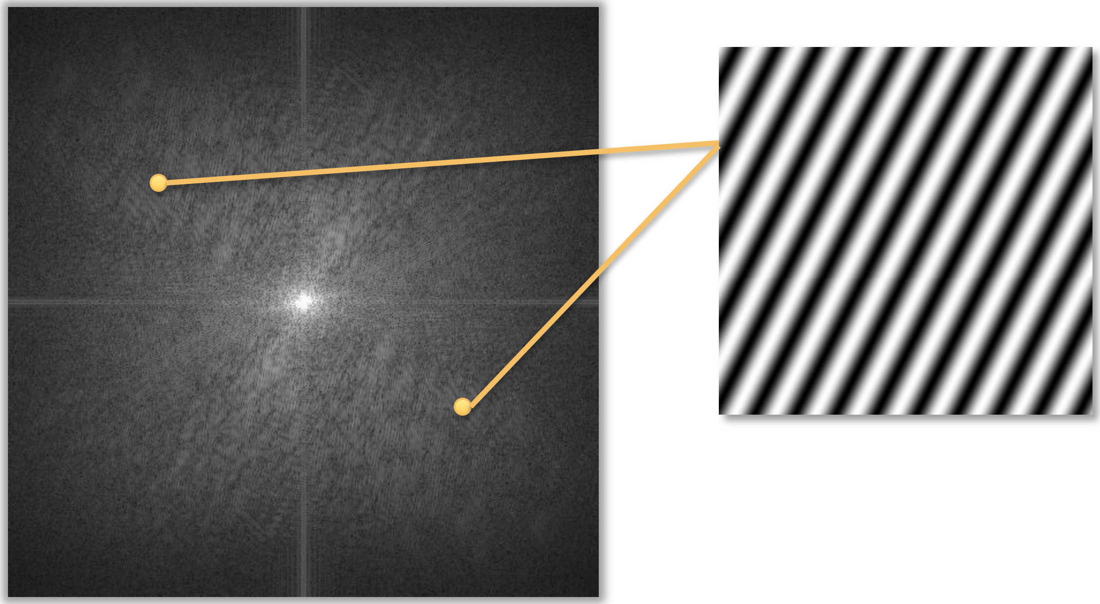
Fourier Transform

- What is this?



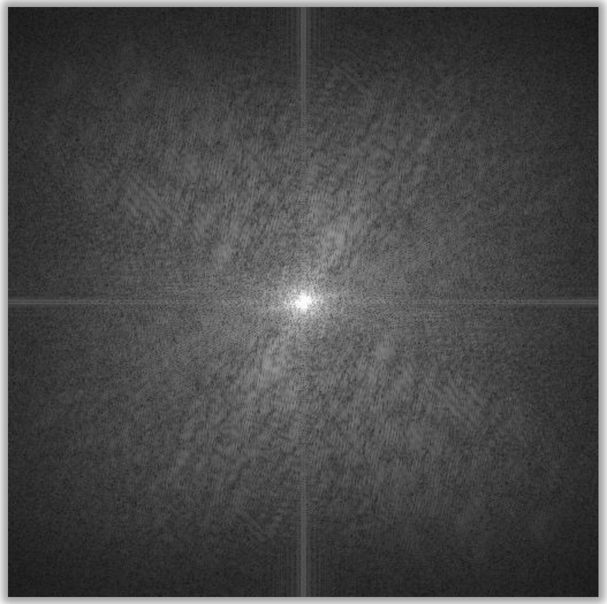
Fourier Transform

- What is this?



Fourier Transform

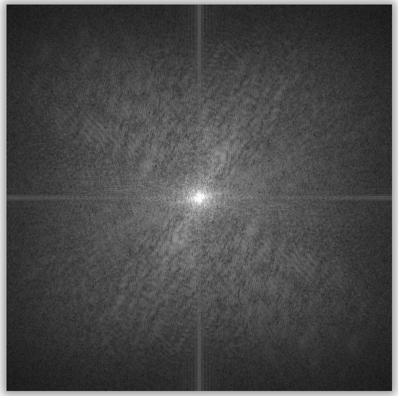
- What is this?



Fourier Transform

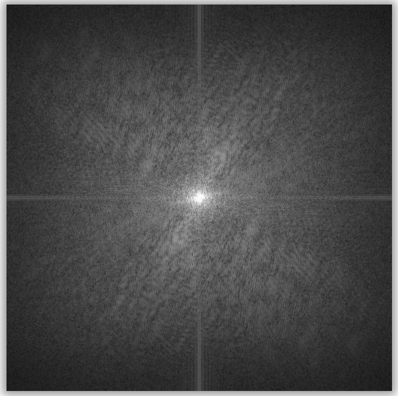
- any continuous, integrable function can be represented as an infinite sum of sines and cosines:

$$f(x) = \int_{-\infty}^{\infty} \hat{f}(\xi) e^{2\pi i \xi x}\, d\xi \quad \longleftrightarrow \quad \hat{f}(\xi) = \int_{-\infty}^{\infty} f(x) e^{-2\pi i \xi x}\, dx$$



Fourier Transform

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

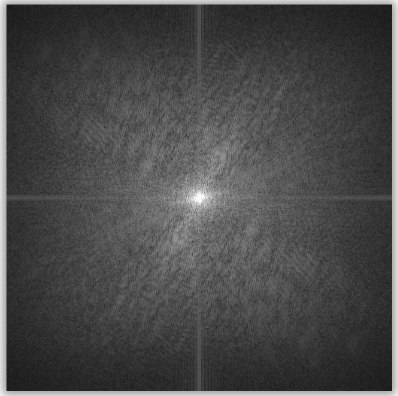


Fourier Transform

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

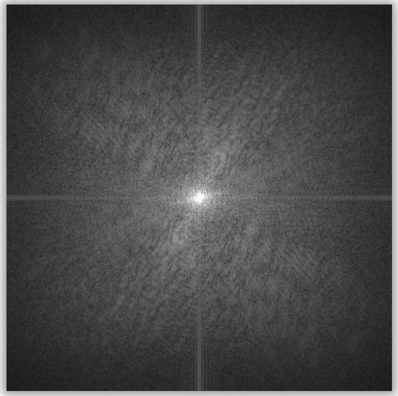
$$\cos(2\pi[k_x x + k_y y]) + j\sin(2\pi[k_x x + k_y y])$$

Fourier Transform



$$f(x,y) = \int_{-\infty}^{\infty} \underbrace{F(k_x, k_y)}_{Ae^{j\phi}} e^{2\pi i (k_x x + k_y y)} \mathrm{d}k_x \mathrm{d}k_y$$

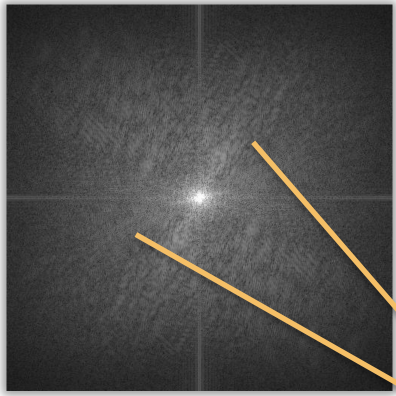
Fourier Transform



$$f(x,y) = \int_{-\infty}^{\infty} \underbrace{F(k_x,k_y)e^{2\pi i(k_x x + k_y y)}}_{} \mathrm{d}k_x \mathrm{d}k_y$$

$$A\cos(2\pi[k_x x + k_y y] + \phi) + jA\sin(2\pi[k_x x + k_y y] + \phi)$$

Fourier Transform

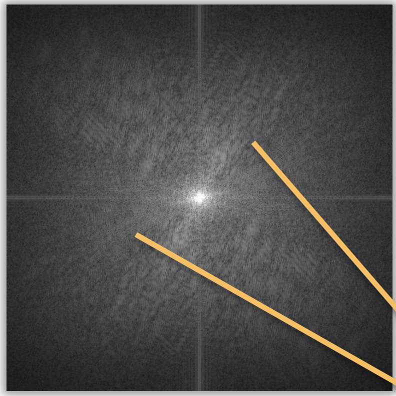


$$f(x,y) = \int_{-\infty}^{\infty} F(k_x,k_y)e^{2\pi i(k_x x + k_y y)}\mathrm{d}k_x\mathrm{d}k_y$$

$$A\cos(2\pi[k_x x + k_y y] + \phi) + jA\sin(2\pi[k_x x + k_y y] + \phi)$$

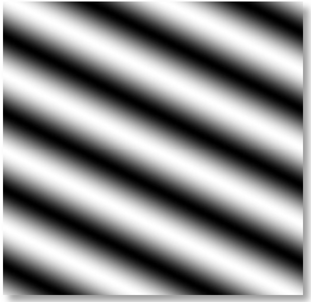
Fourier coefficients of real signals are conjugate symmetric

Fourier Transform



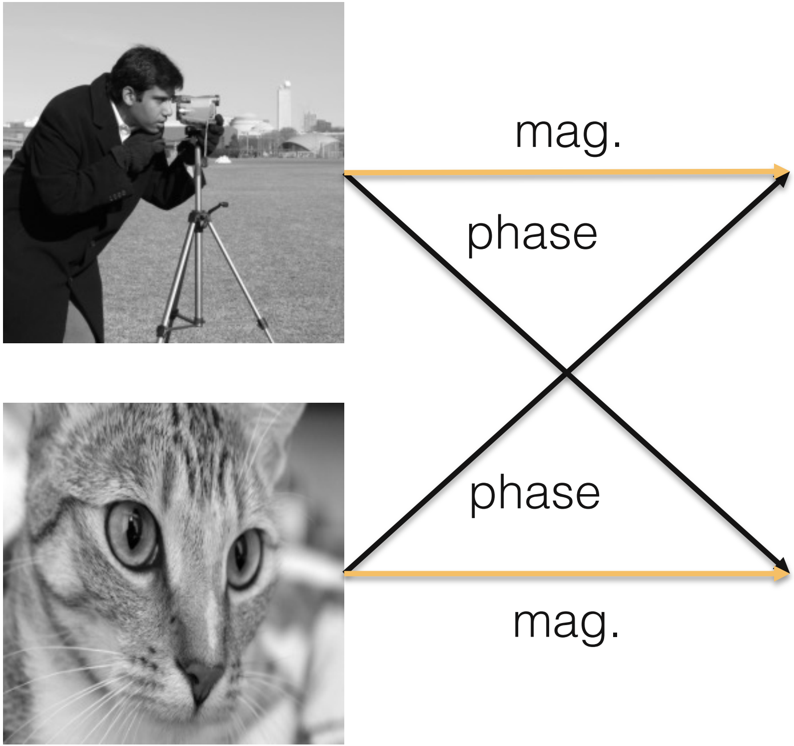
$$f(x,y) = \int_{-\infty}^{\infty} \underbrace{F(k_x, k_y) e^{2\pi i (k_x x + k_y y)}} \mathrm{d}k_x \mathrm{d}k_y$$

$$A\cos(2\pi[k_x x + k_y y] + \phi) + jA\sin(2\pi[k_x x + k_y y] + \phi)$$

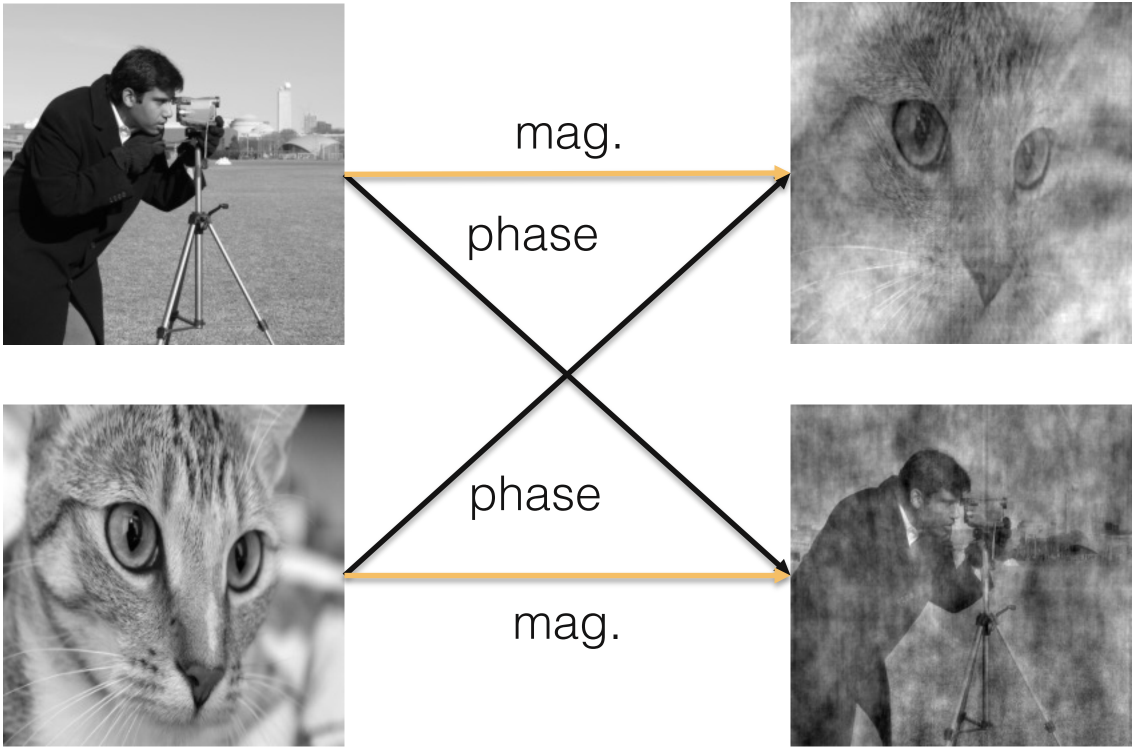


Images are sums of cosines at different amplitudes, phases, spatial frequencies

Magnitude vs Phase



Magnitude vs Phase



Fourier Transform

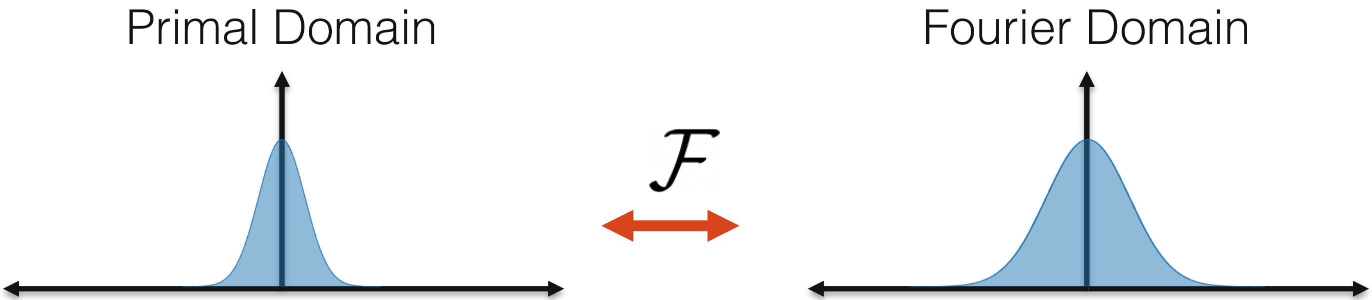
- any continuous, integrable, periodic function can be represented as an infinite sum of sines and cosines:

$$f(x) = \int_{-\infty}^{\infty} \hat{f}(\xi) e^{2\pi i \xi x} \, d\xi \quad \longleftrightarrow \quad \hat{f}(\xi) = \int_{-\infty}^{\infty} f(x) e^{-2\pi i \xi x} \, dx$$

- convolution theorem (critical):

$$x * g = F^{-1}\left\{ F\{x\} \cdot F\{g\} \right\}$$

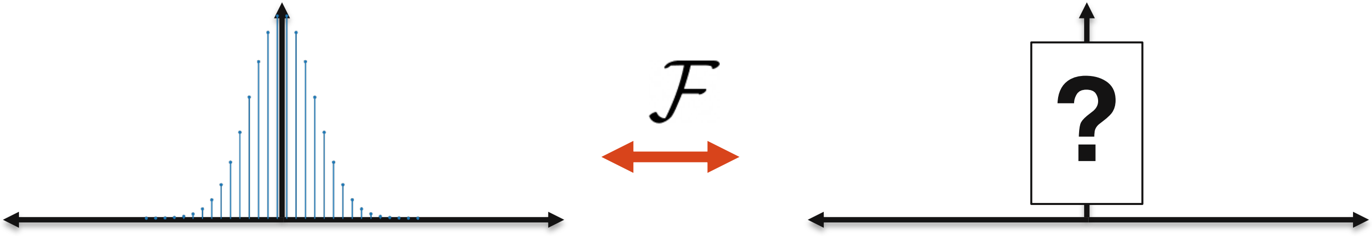
Discrete vs Continuous Fourier Transform



Sampling

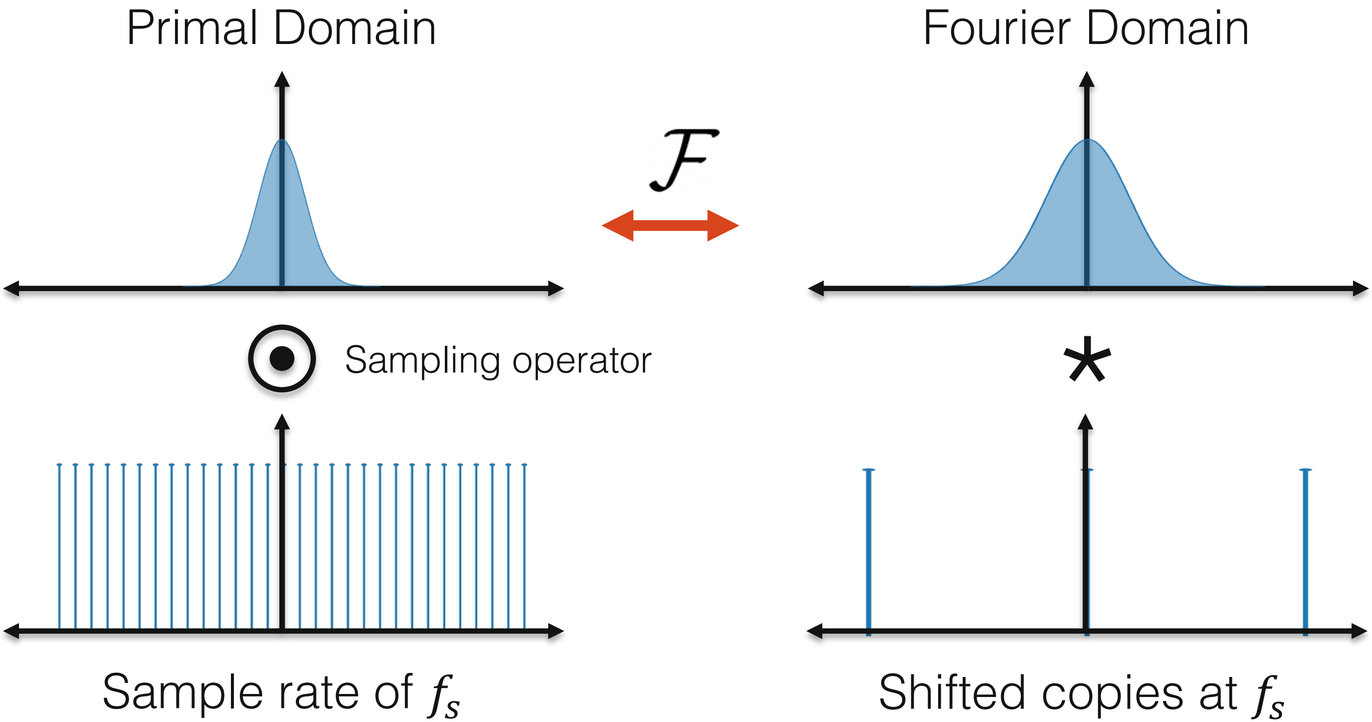
Primal Domain

Fourier Domain

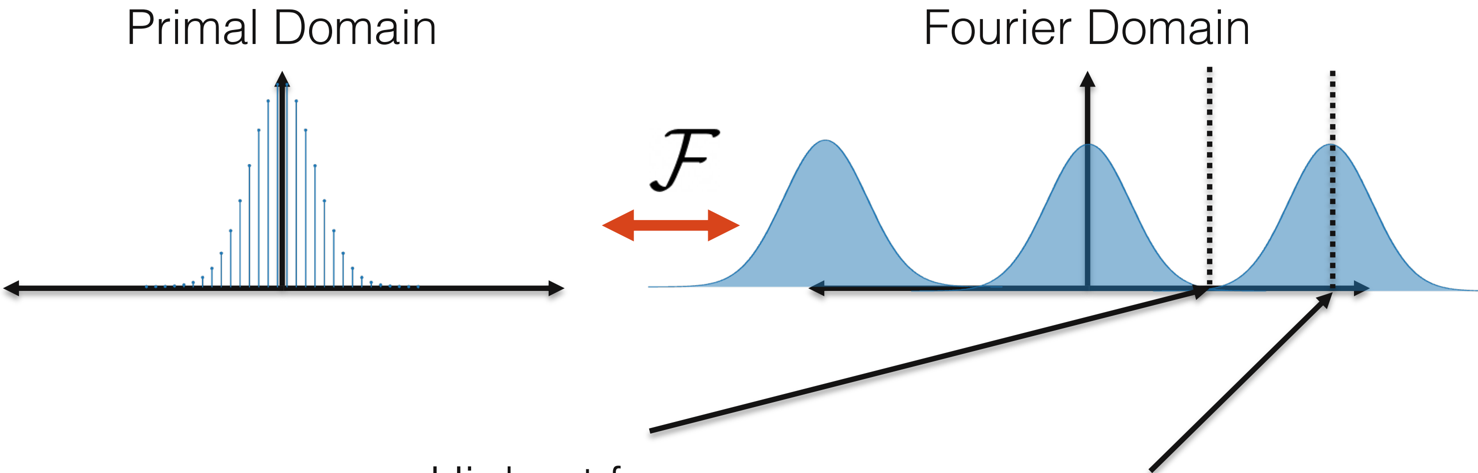


discrete sampled signal

Sampling


Primal Domain
Fourier Domain
$\mathcal{F}$
Sampling operator
*
Sample rate of f_s
Shifted copies at f_s

Sampling

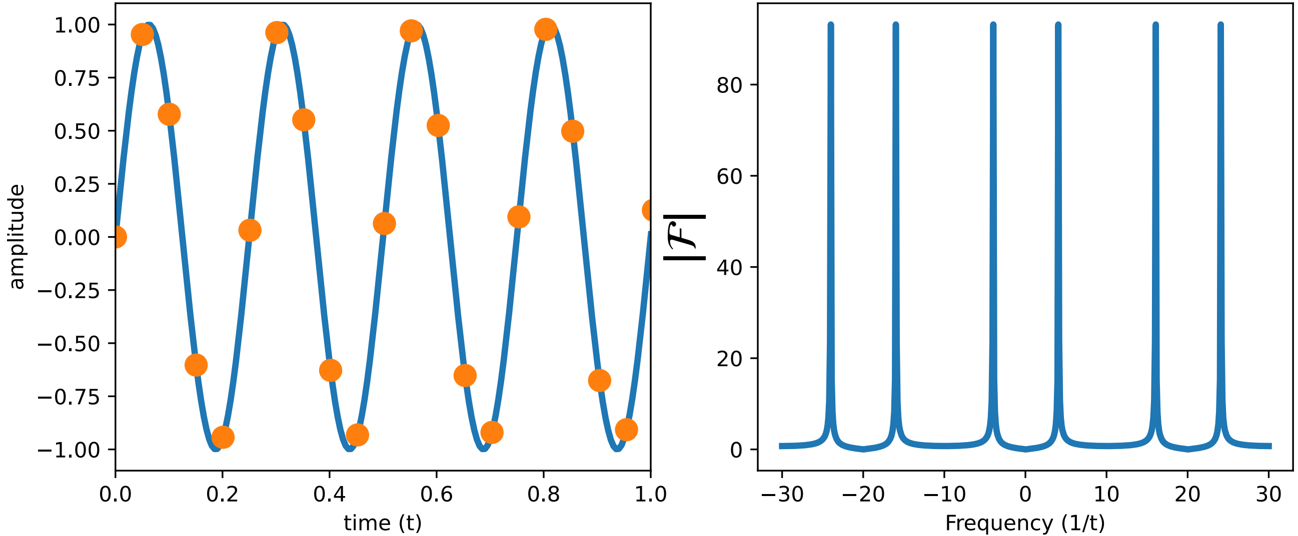


Highest frequency

Sample rate should be twice the highest frequency to avoid aliasing!

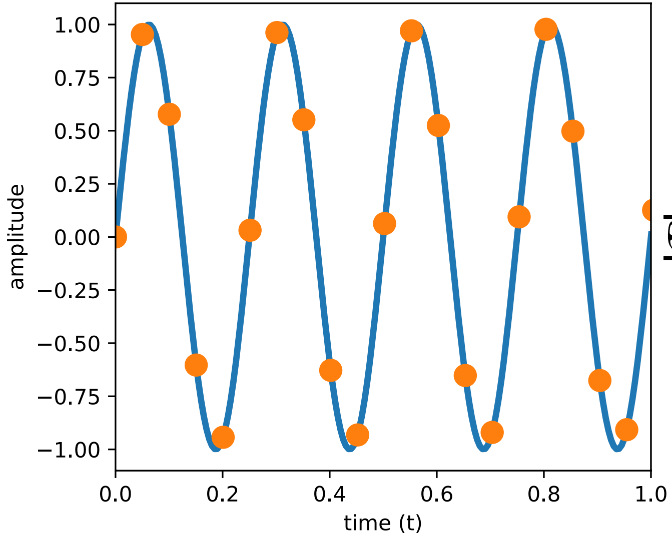
Sampling exercise

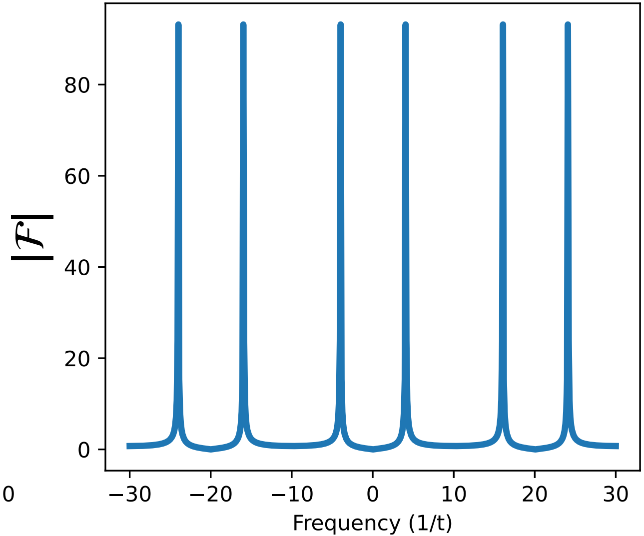
Sample frequency: ?
Signal frequency: ?



Sampling exercise

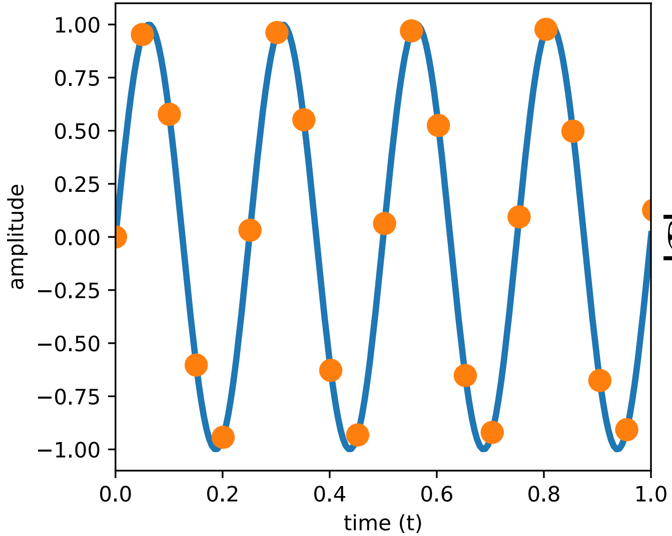
Sample frequency: 20 Hz
Signal: 4 Hz

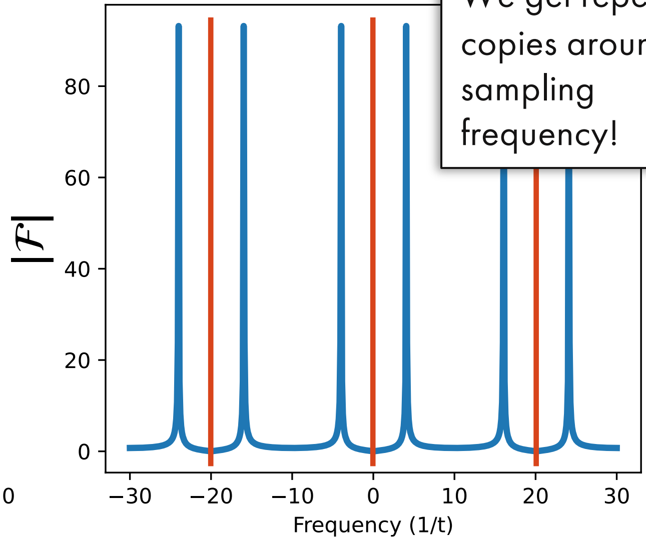




Sampling exercise

Sample frequency: 20 Hz
Signal: 4 Hz

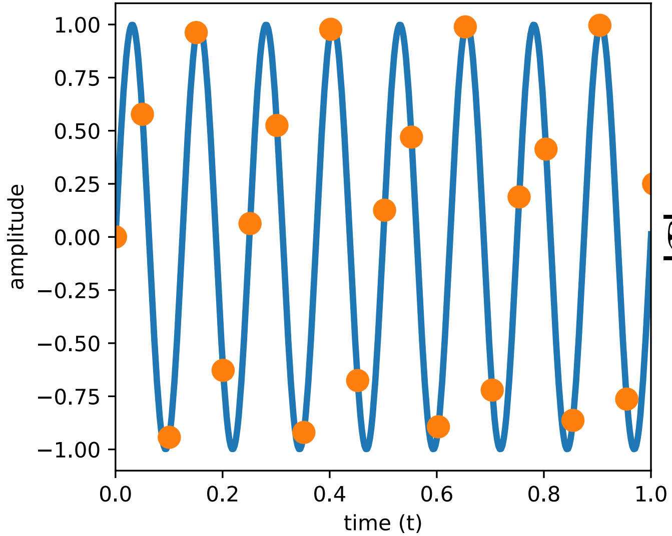


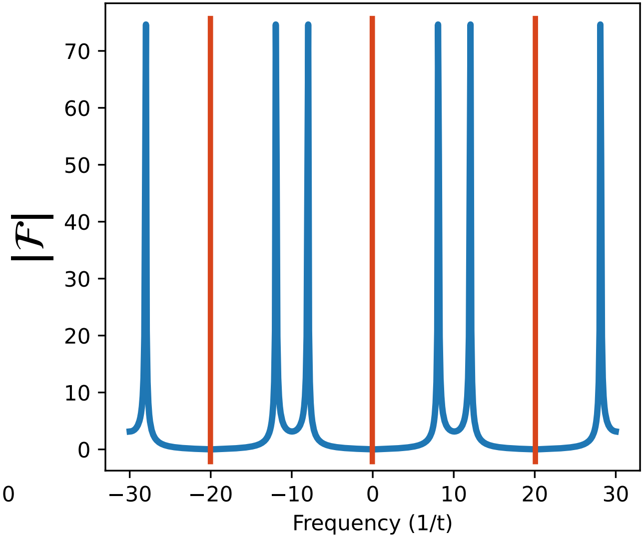


We get repeated copies around the sampling frequency!

Sampling exercise

Sample frequency: 20 Hz
Signal: 8 Hz

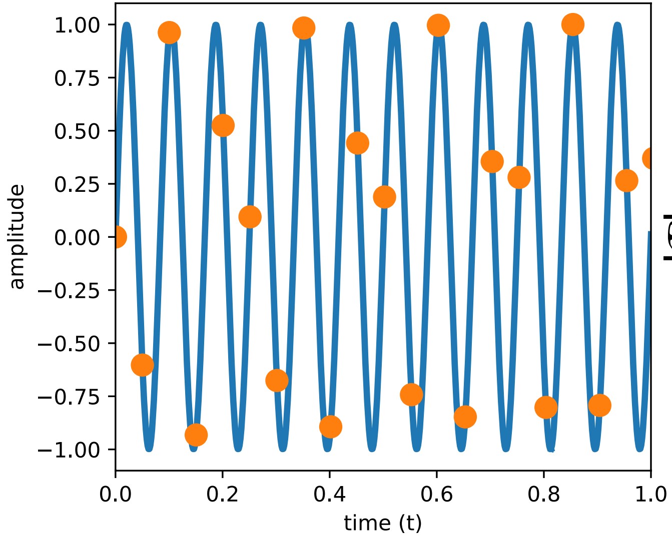


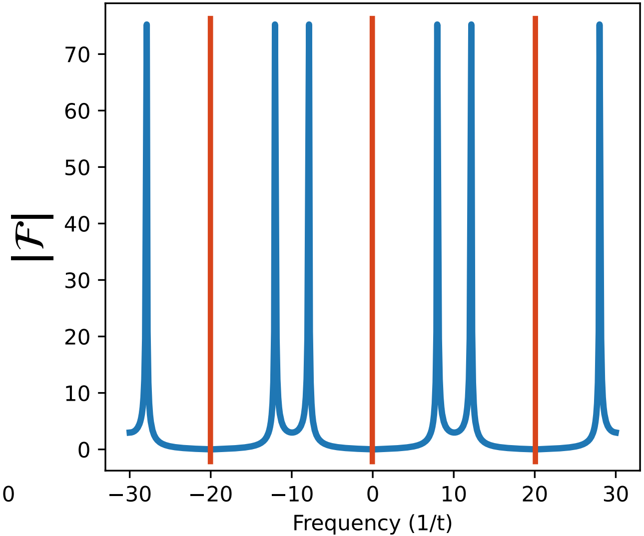


Sampling exercise

Sample frequency: 20 Hz

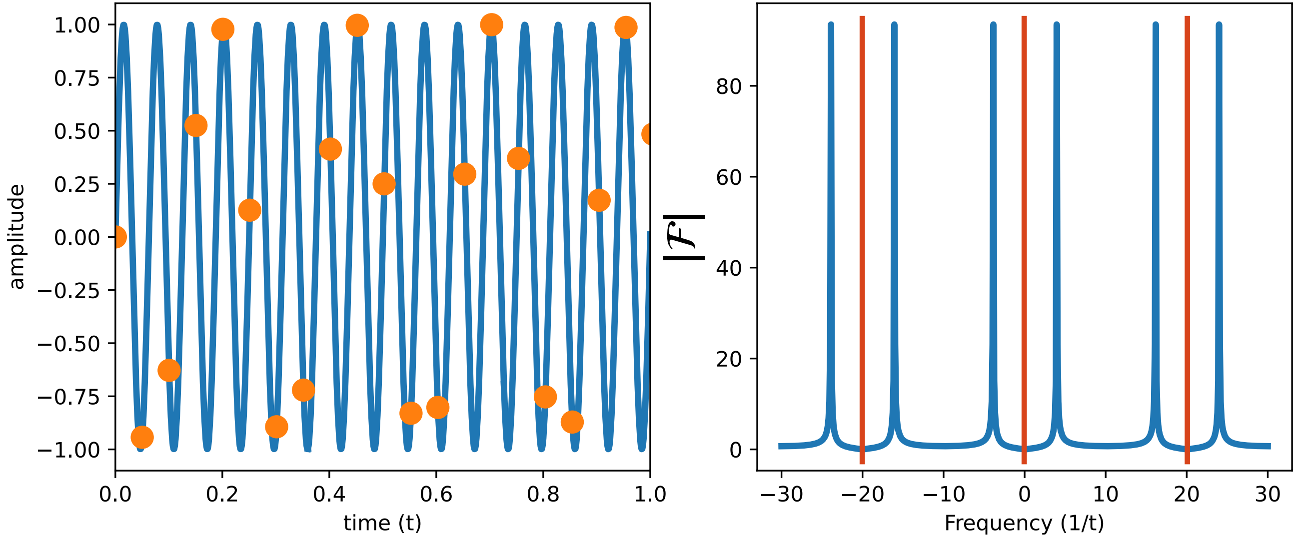
Signal: 12 Hz





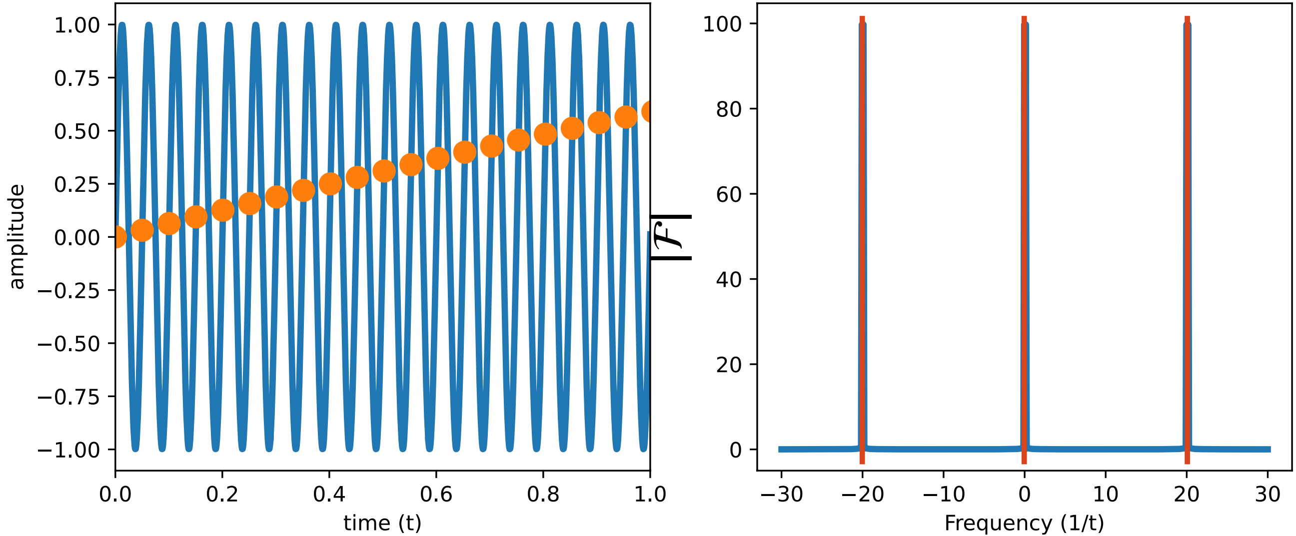
Sampling exercise

Sample frequency: 20 Hz
Signal: 16 Hz



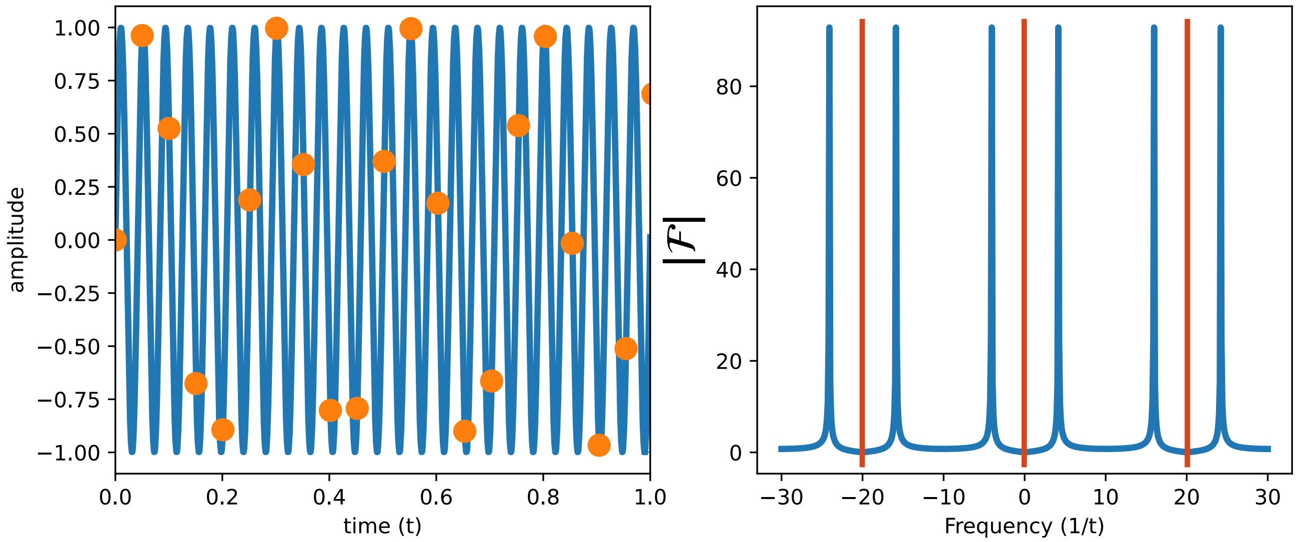
Sampling exercise

Sample frequency: 20 Hz
Signal: 20 Hz



Sampling exercise

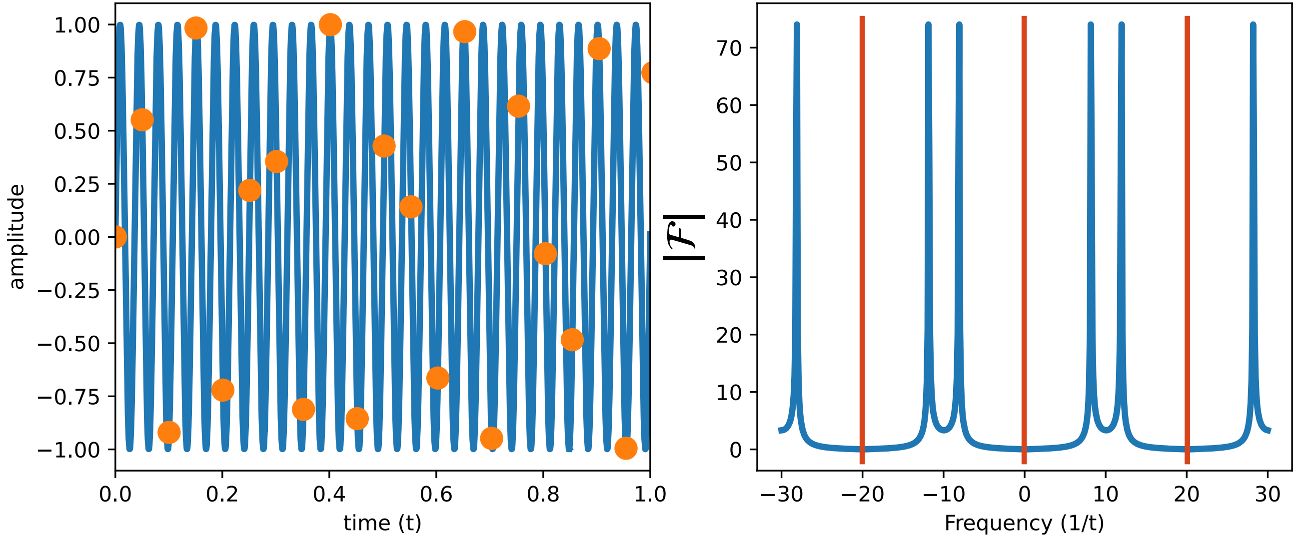
Sample frequency: 20 Hz
Signal: 24 Hz



Sampling exercise

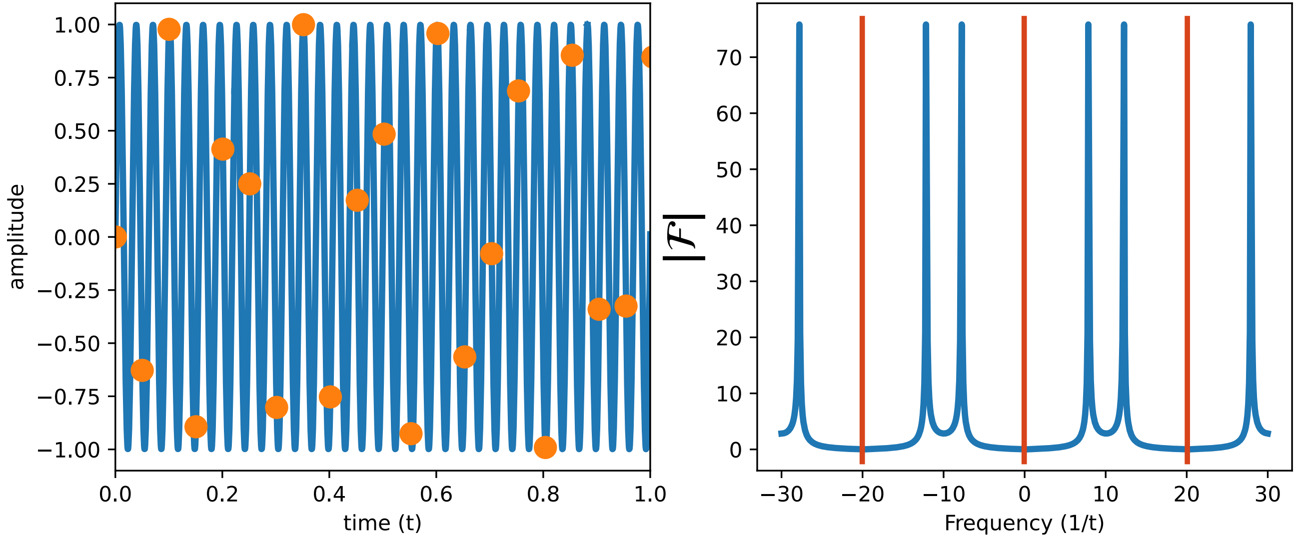
Sample frequency: 20 Hz

Signal: 28 Hz

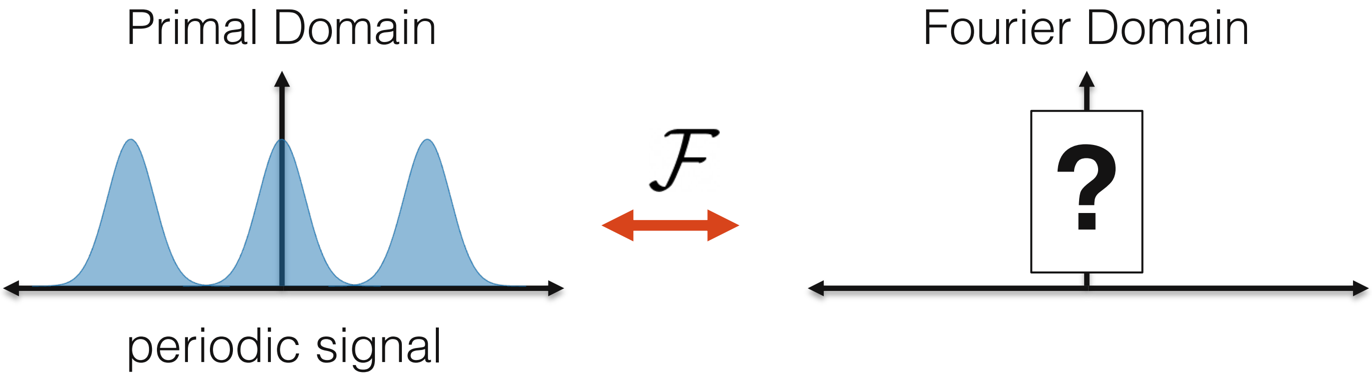


Sampling exercise

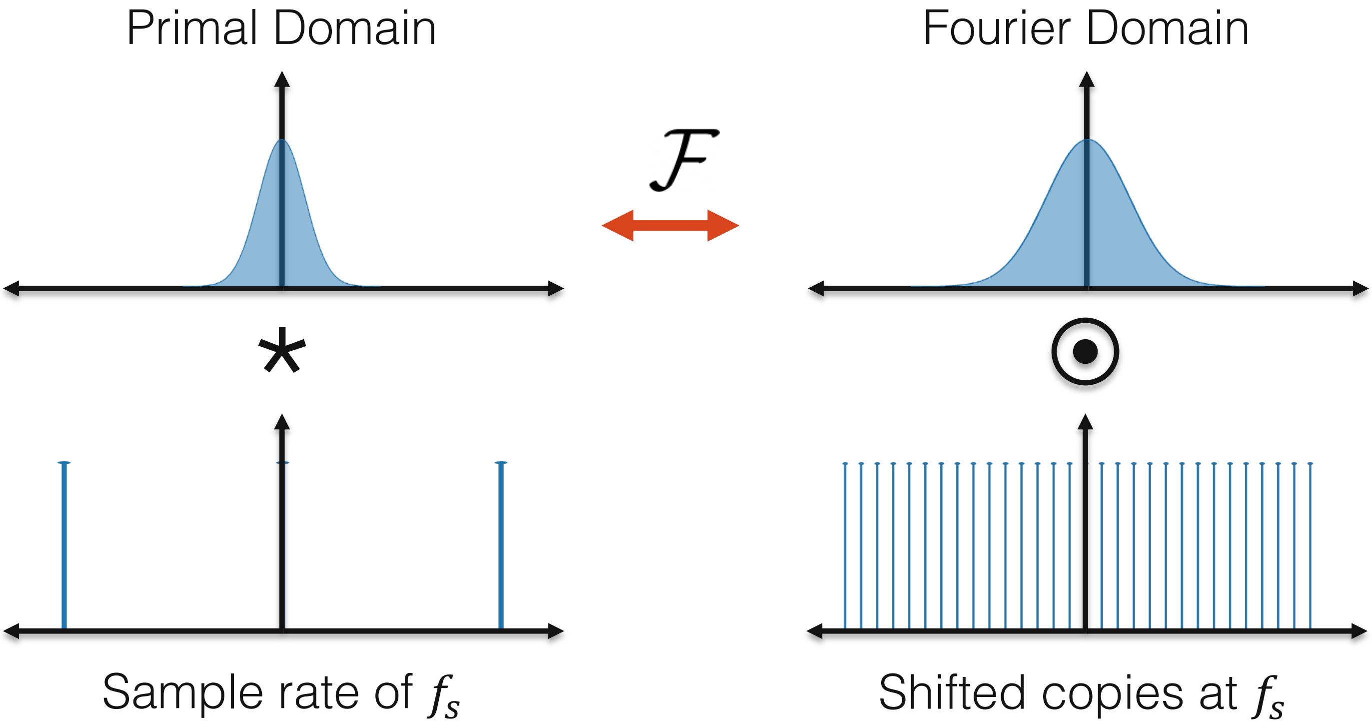
Sample frequency: 20 Hz
Signal: 32 Hz



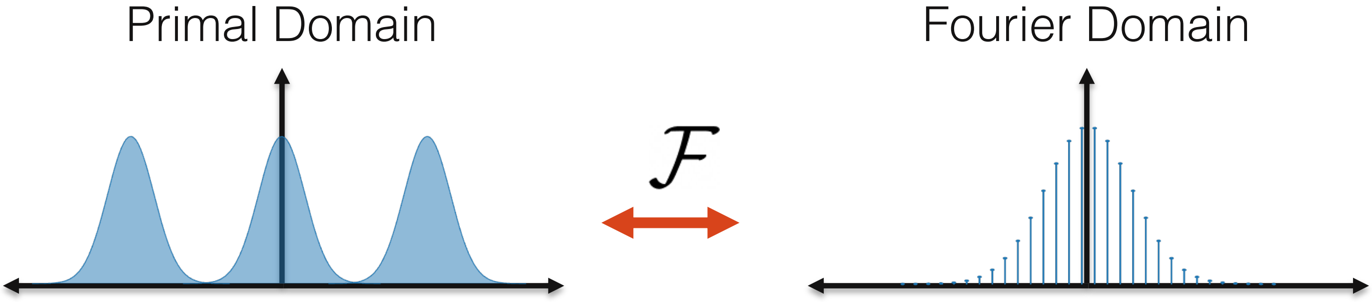
Periodicity



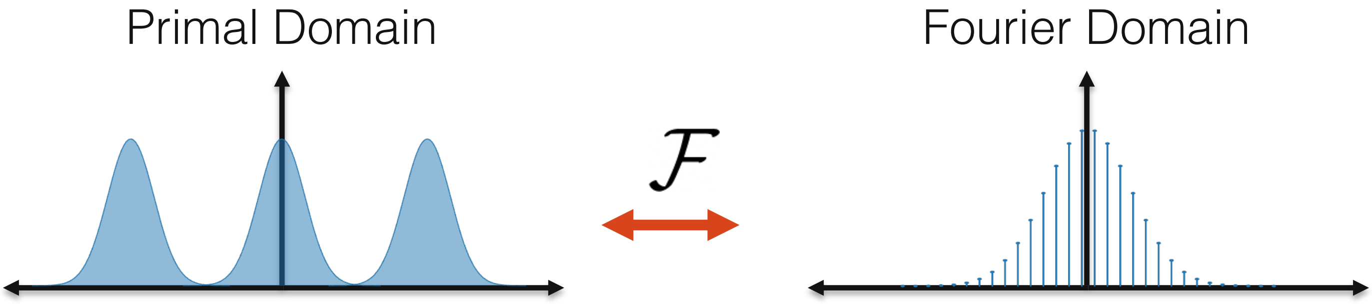
Periodicity



Periodicity



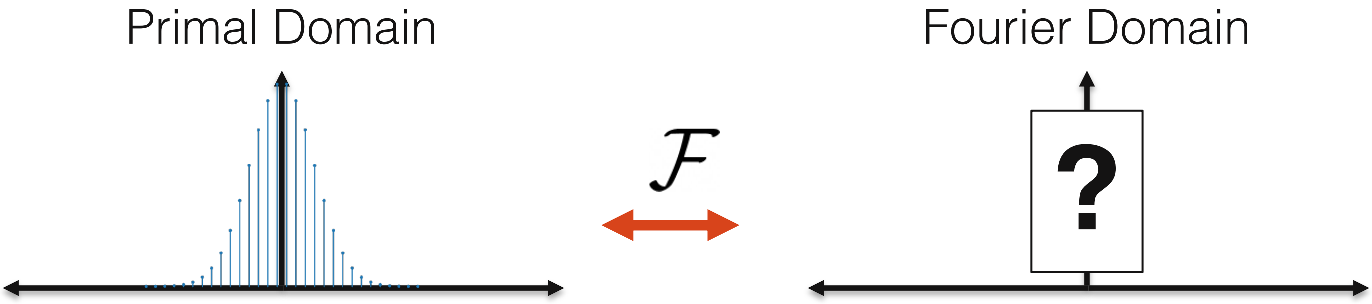
Periodicity



A periodic signal can be represented by a discrete set of Fourier coefficients

- These are called the “Fourier series coefficients”

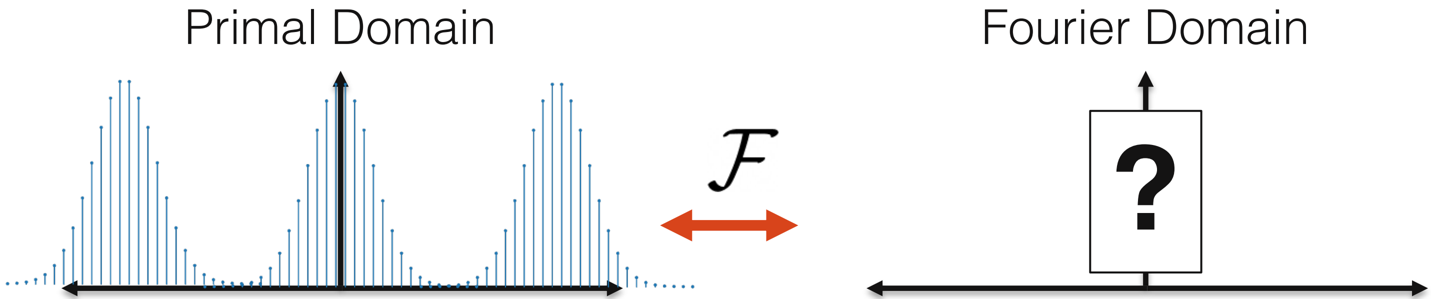
Discrete Fourier Transform



In practice, we wish to take the Fourier transform of discrete signals.

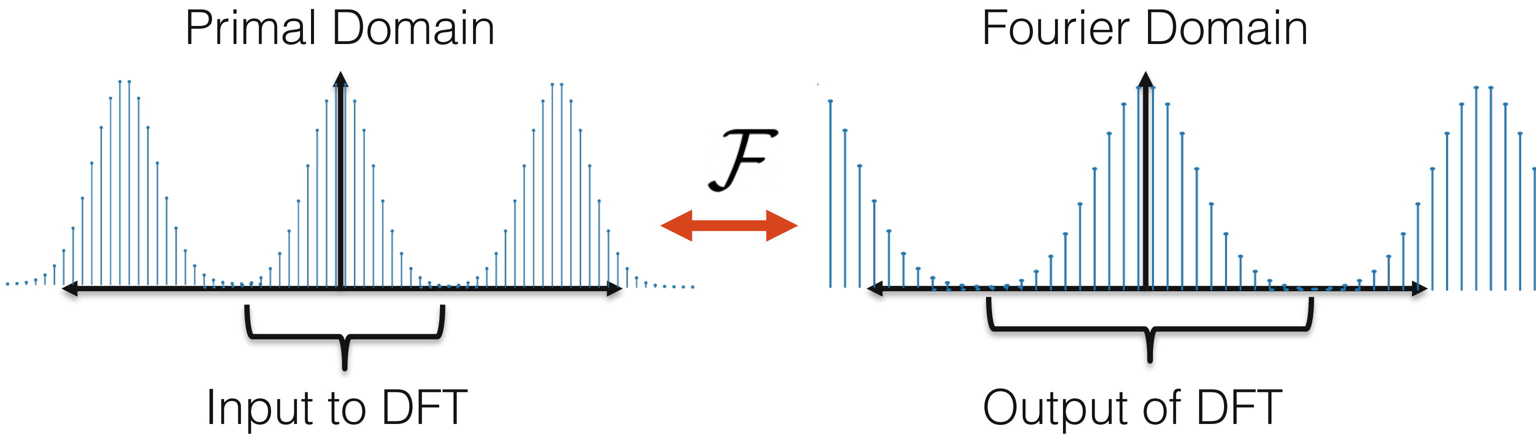
But we need to represent the Fourier domain with discrete values, too!

Discrete Fourier Transform



Assume the primal domain signal is periodic

Discrete Fourier Transform



Assume the primal domain signal is periodic

Discrete Fourier Transform

- most important for us: discrete Fourier transform

$$x[n] = \frac{1}{N}\sum_{k=0}^{N-1} \hat{x}[k] e^{2\pi ikn/N} \quad \longleftrightarrow \quad \hat{x}[k] = \sum_{n=0}^{N-1} x[n] e^{-2\pi ikn/N}$$

Discrete Fourier Transform

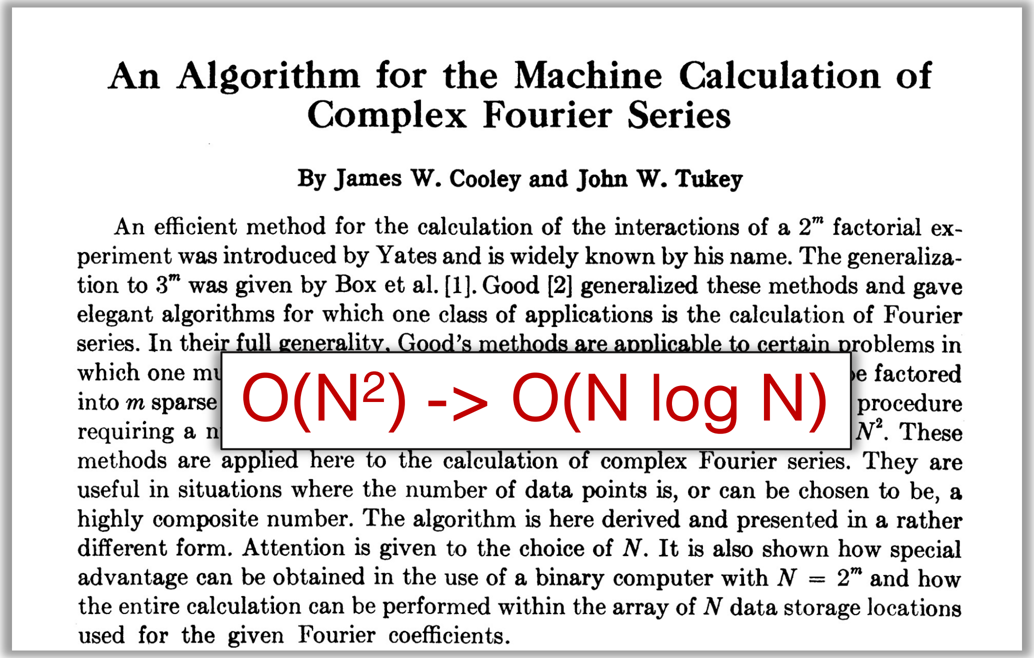
An Algorithm for the Machine Calculation of Complex Fourier Series

By James W. Cooley and John W. Tukey

An efficient method for the calculation of the interactions of a 2^m factorial experiment was introduced by Yates and is widely known by his name. The generalization to 3^m was given by Box et al. [1]. Good [2] generalized these methods and gave elegant algorithms for which one class of applications is the calculation of Fourier series. In their full generality, Good's methods are applicable to certain problems in which one must multiply an N-vector by an $N \times N$ matrix which can be factored into m sparse matrices, where m is proportional to log N. This results in a procedure requiring a number of operations proportional to $N \log N$ rather than N^2. These methods are applied here to the calculation of complex Fourier series. They are useful in situations where the number of data points is, or can be chosen to be, a highly composite number. The algorithm is here derived and presented in a rather different form. Attention is given to the choice of N. It is also shown how special advantage can be obtained in the use of a binary computer with $N = 2^m$ and how the entire calculation can be performed within the array of N data storage locations used for the given Fourier coefficients.

Fast Fourier Transform: Cooley & Tukey 1965

Discrete Fourier Transform



An Algorithm for the Machine Calculation of Complex Fourier Series

By James W. Cooley and John W. Tukey

An efficient method for the calculation of the interactions of a 2^m factorial experiment was introduced by Yates and is widely known by his name. The generalization to 3^m was given by Box et al. [1]. Good [2] generalized these methods and gave elegant algorithms for which one class of applications is the calculation of Fourier series. In their full generality, Good's methods are applicable to certain problems in which one mu ... e factored into m sparse ... procedure requiring a n ... N^2. These methods are applied here to the calculation of complex Fourier series. They are useful in situations where the number of data points is, or can be chosen to be, a highly composite number. The algorithm is here derived and presented in a rather different form. Attention is given to the choice of N. It is also shown how special advantage can be obtained in the use of a binary computer with $N = 2^m$ and how the entire calculation can be performed within the array of N data storage locations used for the given Fourier coefficients.

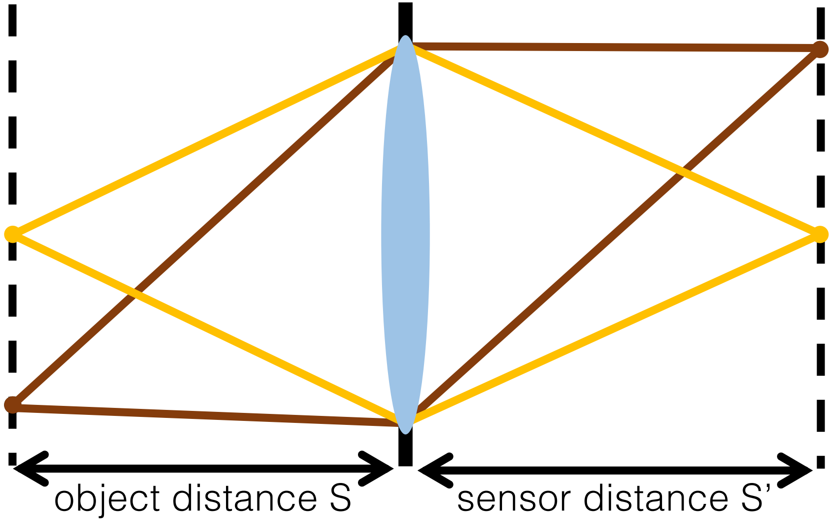
Fast Fourier Transform: Cooley & Tukey 1965

Fourier Transforms in Imaging

Lens imperfections

- Ideal lens: A point maps to a point at a certain plane.

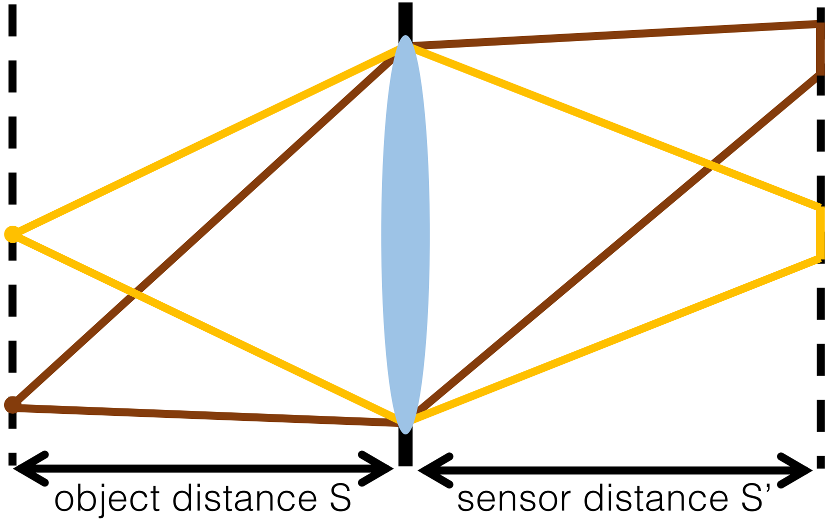
$$\frac{1}{S'} + \frac{1}{S} = \frac{1}{f}$$



Lens imperfections

- Ideal lens: A point maps to a point at a certain plane.
- Real lens: A point maps to a circle that has non-zero minimum radius among all planes.

$$\frac{1}{S'} + \frac{1}{S} = \frac{1}{f}$$

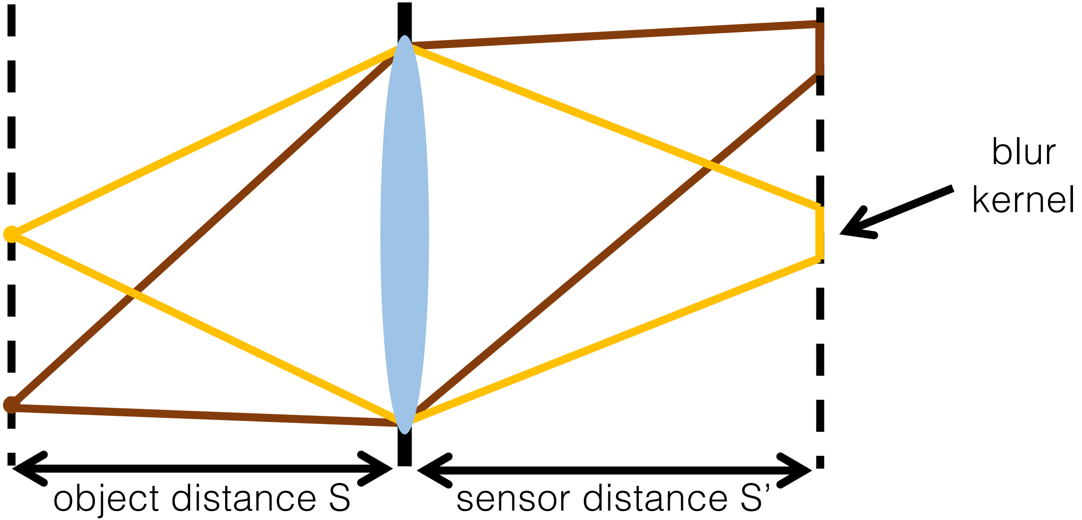


What is the effect of this on the images we capture?

Lens imperfections

- Ideal lens: A point maps to a point at a certain plane.
- Real lens: A point maps to a circle that has non-zero minimum radius among all planes.

$$\frac{1}{S'} + \frac{1}{S} = \frac{1}{f}$$



Shift-invariant blur.

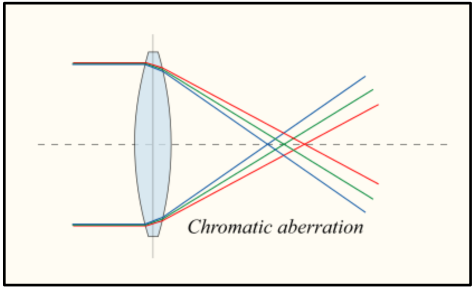
Lens imperfections

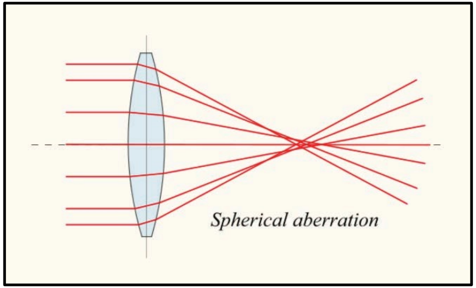
What causes lens imperfections?

Lens imperfections

What causes lens imperfections?

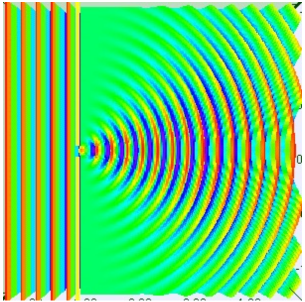
- Aberrations.



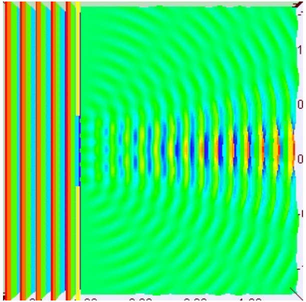


(Important note: Oblique aberrations like coma and distortion <u>are not shift-invariant</u> blur and we do not consider them here!)

- Diffraction.



small aperture



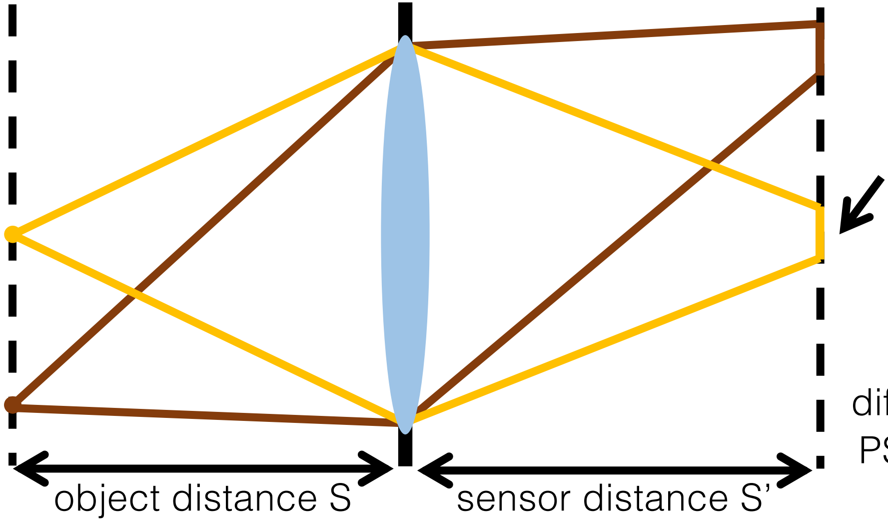
large aperture

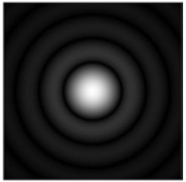
Lens as an optical low-pass filter

Point spread function (PSF): The blur kernel of a lens.

- "Diffraction-limited" PSF: No aberrations, only diffraction. Determined by aperture shape.

$$\frac{1}{S'} + \frac{1}{S} = \frac{1}{f}$$





diffraction-limited PSF of a circular aperture (Airy pattern)

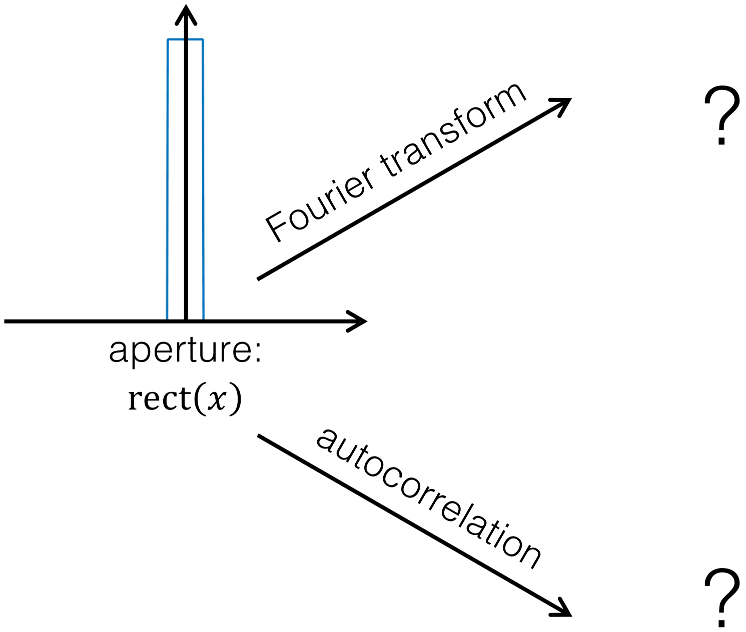
Some basics of diffraction theory

We will assume that we can use:

- Fraunhofer diffraction (i.e., distance of sensor and aperture is large relative to wavelength).
- incoherent illumination (i.e., the light we are measuring is not laser light).

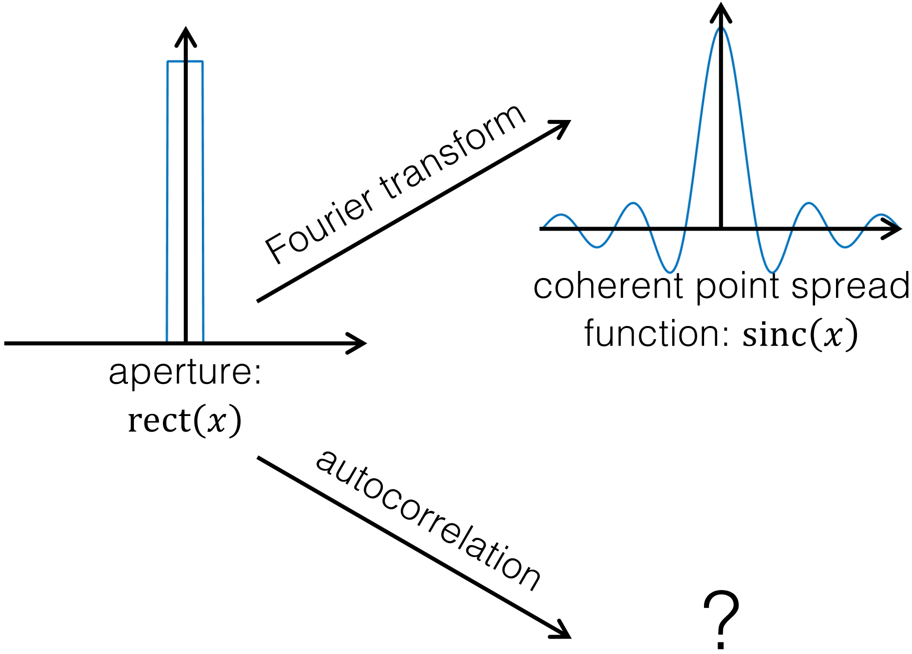
We will also be ignoring various scale factors. Different functions are <u>not</u> drawn to scale.

Some basics of diffraction theory



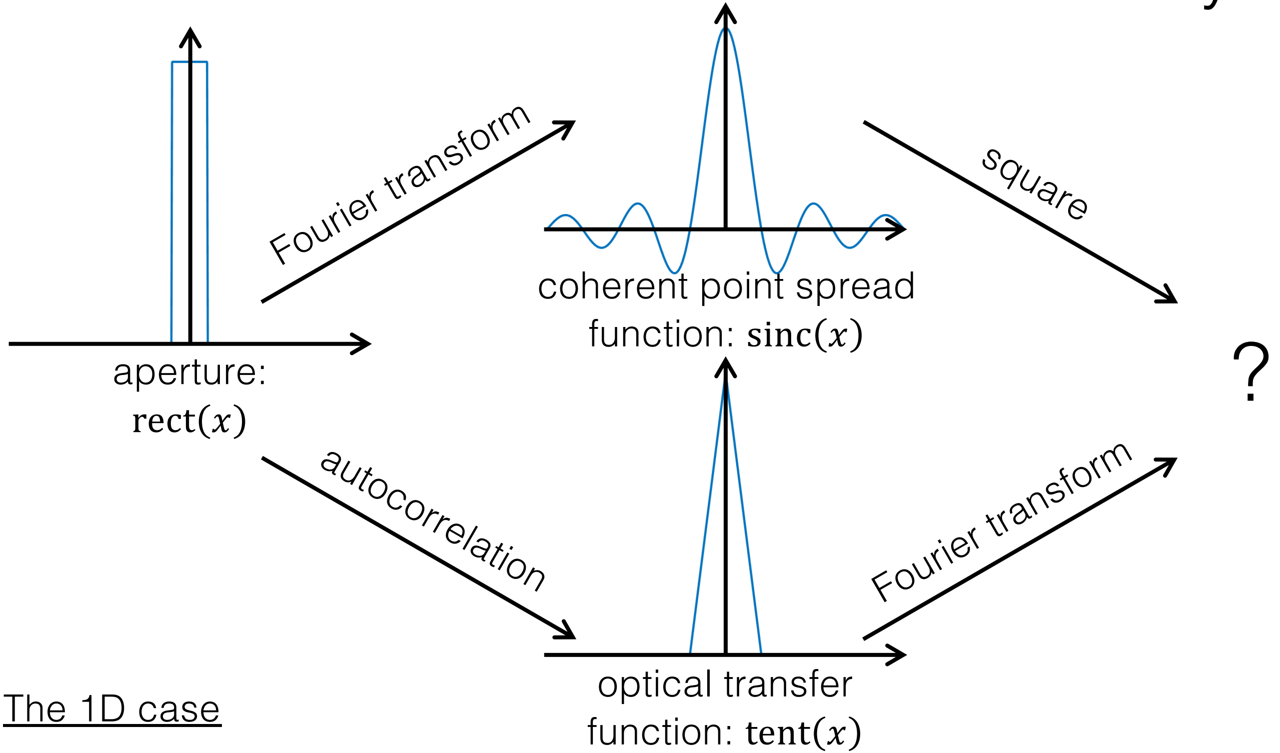
The 1D case

Some basics of diffraction theory



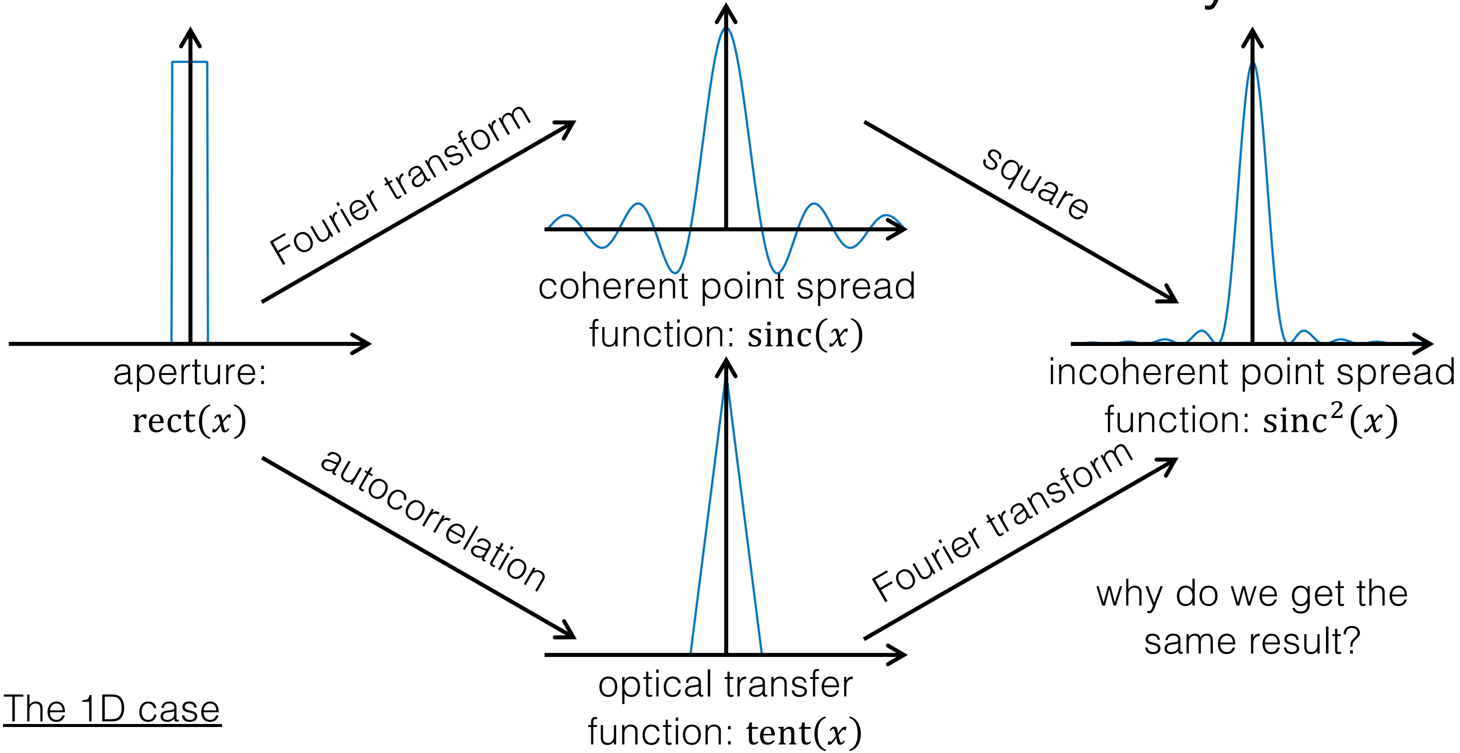
The 1D case

Some basics of diffraction theory



The 1D case

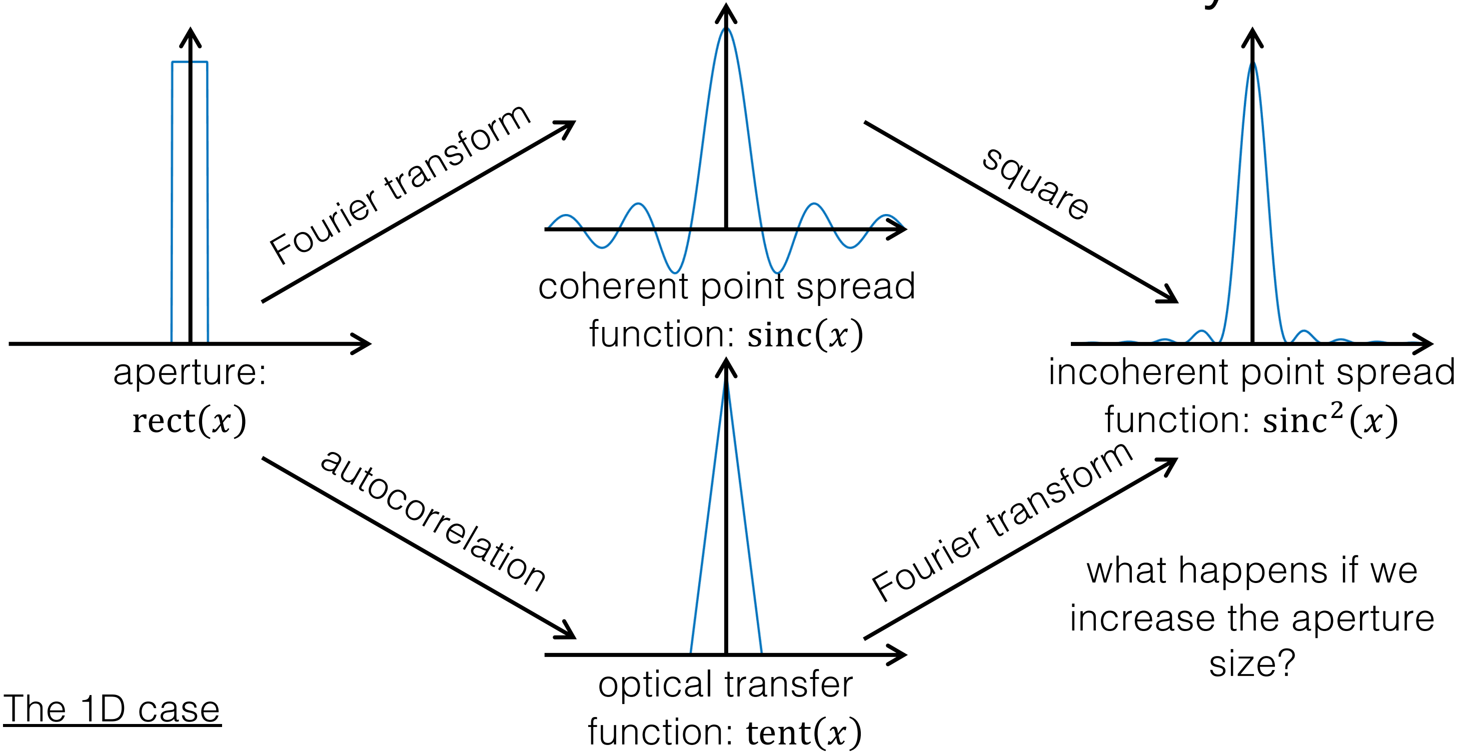
Some basics of diffraction theory



why do we get the same result?

The 1D case

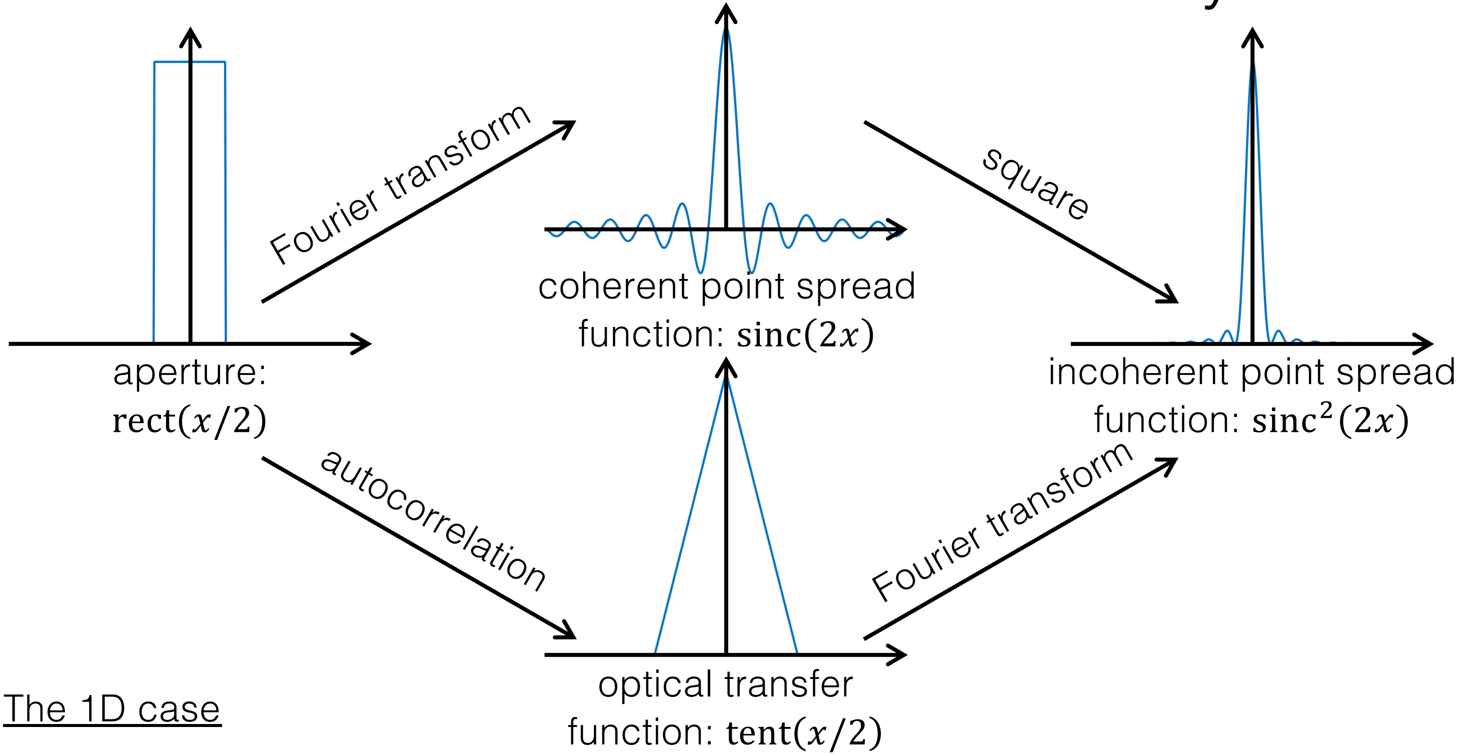
Some basics of diffraction theory



what happens if we increase the aperture size?

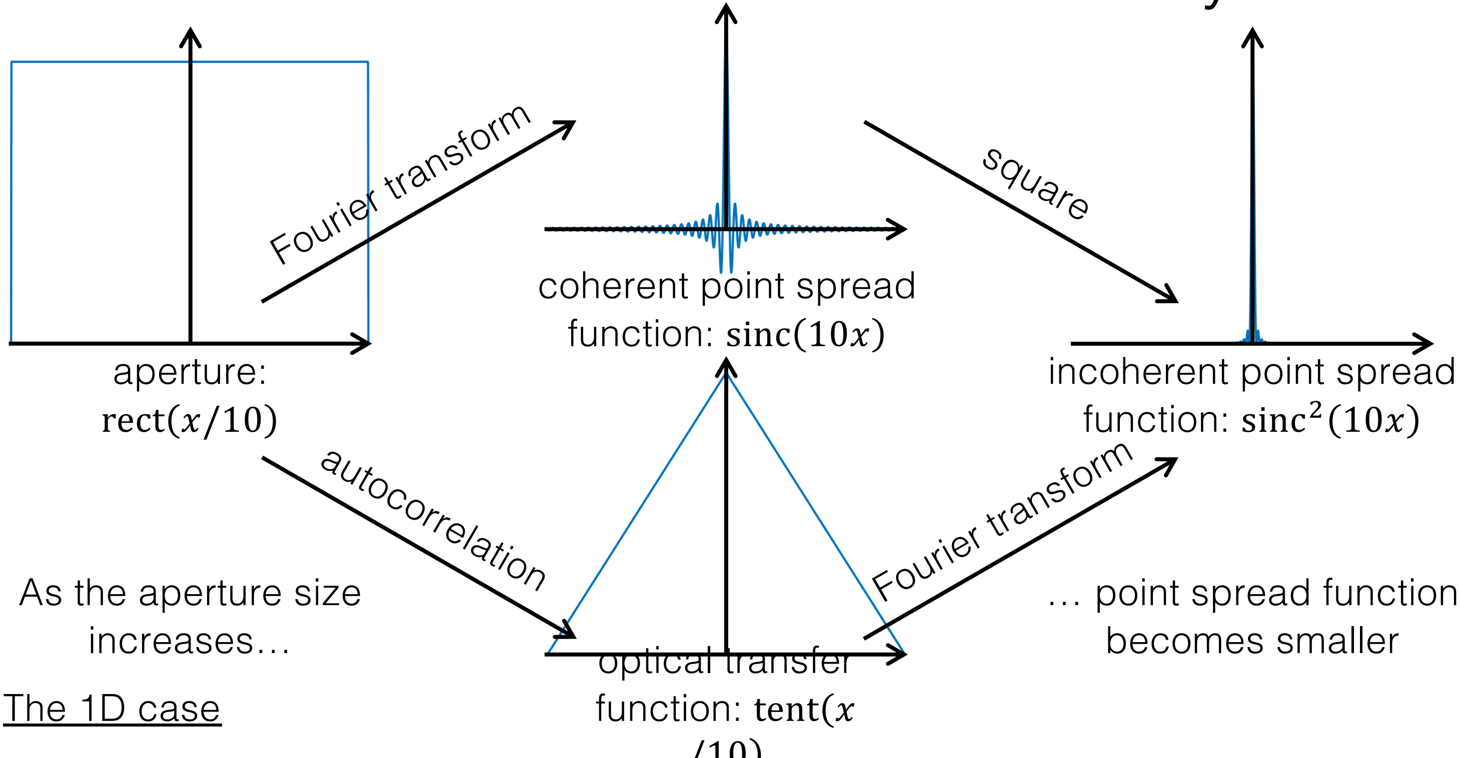
<u>The 1D case</u>

Some basics of diffraction theory



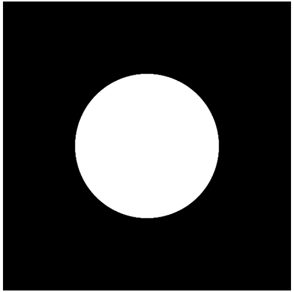
The 1D case

Some basics of diffraction theory



The 1D case

Some basics of diffraction theory

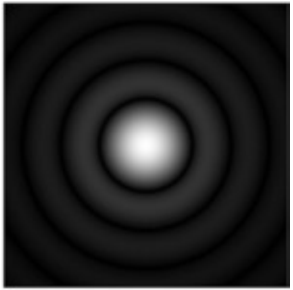


aperture

autocorrelation

optical transfer function

Fourier transform



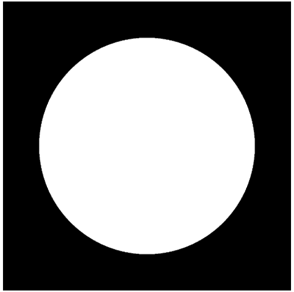
incoherent point spread function

As the aperture size increases…

… point spread function becomes smaller

The 2D case

Some basics of diffraction theory

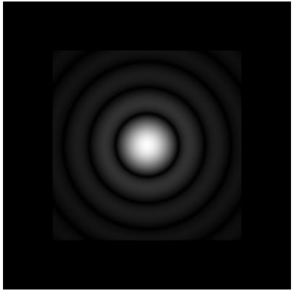


aperture

autocorrelation

optical transfer function

Fourier transform



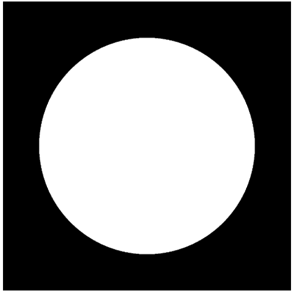
incoherent point spread function

As the aperture size increases…

… point spread function becomes smaller

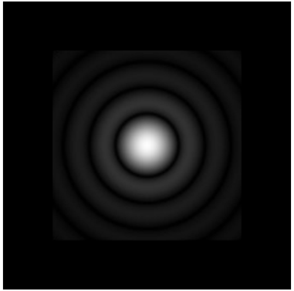
The 2D case

Some basics of diffraction theory



aperture

Why do we prefer circular apertures?



incoherent point spread function

autocorrelation

optical transfer function

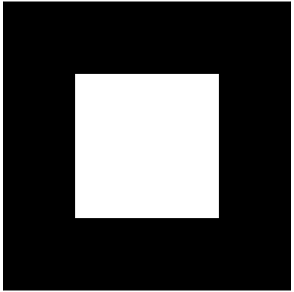
Fourier transform

As the aperture size increases…

… point spread function becomes smaller

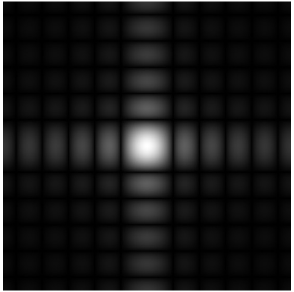
The 2D case

Some basics of diffraction theory



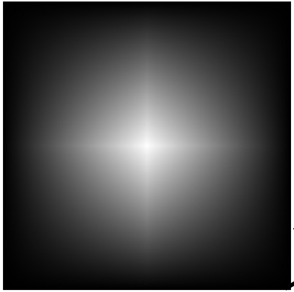
aperture

Other shapes produce very anisotropic blur.



incoherent point spread function

autocorrelation



Fourier transform

As the aperture size increases…

… point spread function becomes smaller

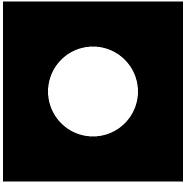
optical transfer function

<u>The 2D case</u>

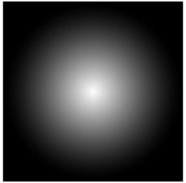
Lens as an optical low-pass filter

Point spread function (PSF): The blur kernel of a lens.

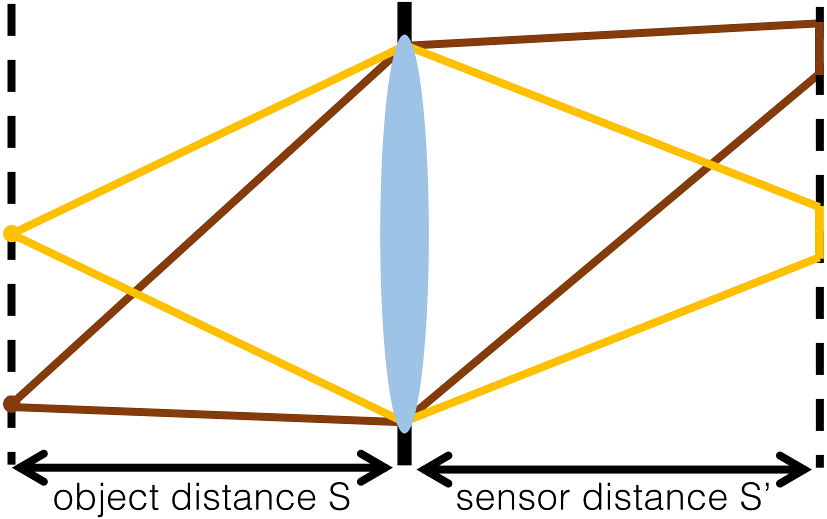
- “Diffraction-limited” PSF: No aberrations, only diffraction. Determined by aperture shape.



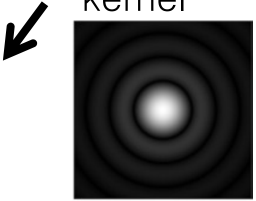
aperture



optical transfer function (OTF)



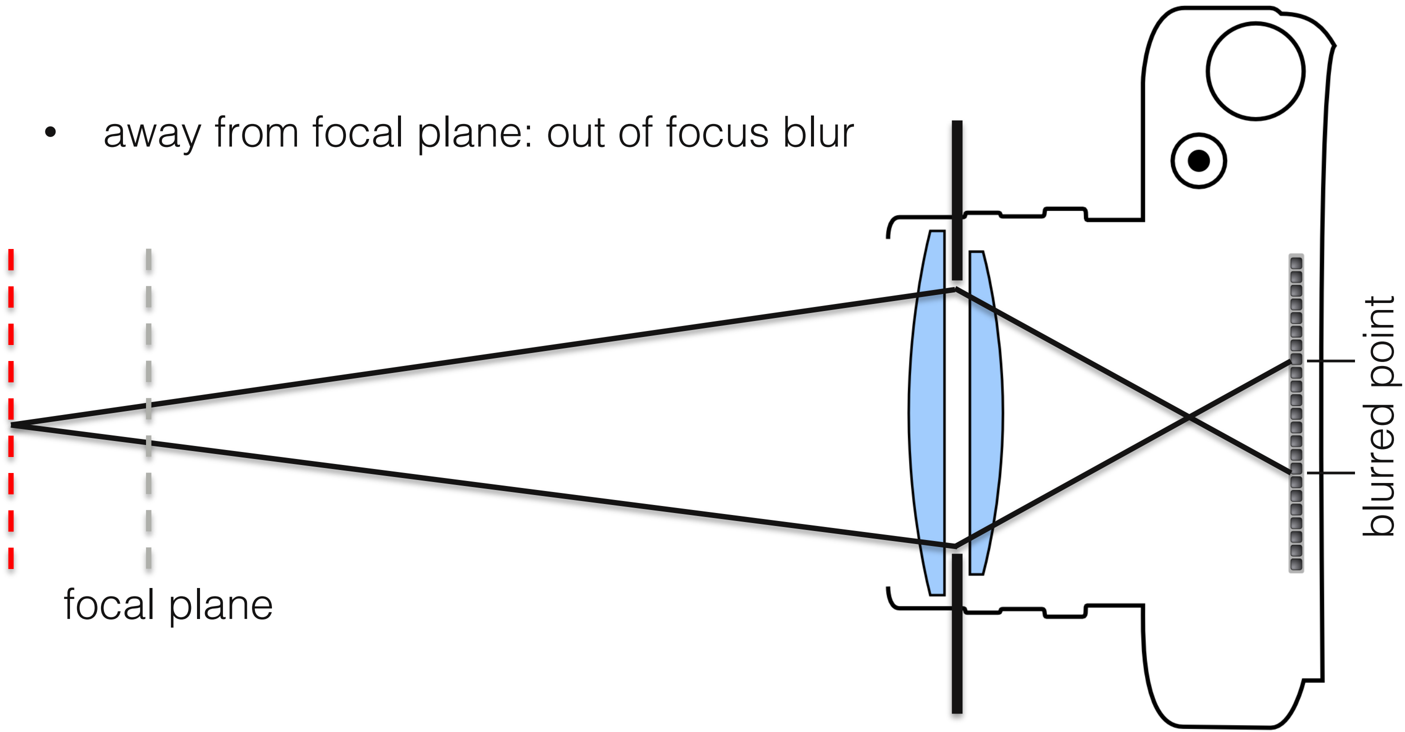
blur kernel



diffraction-limited PSF of a circular aperture (Airy pattern)

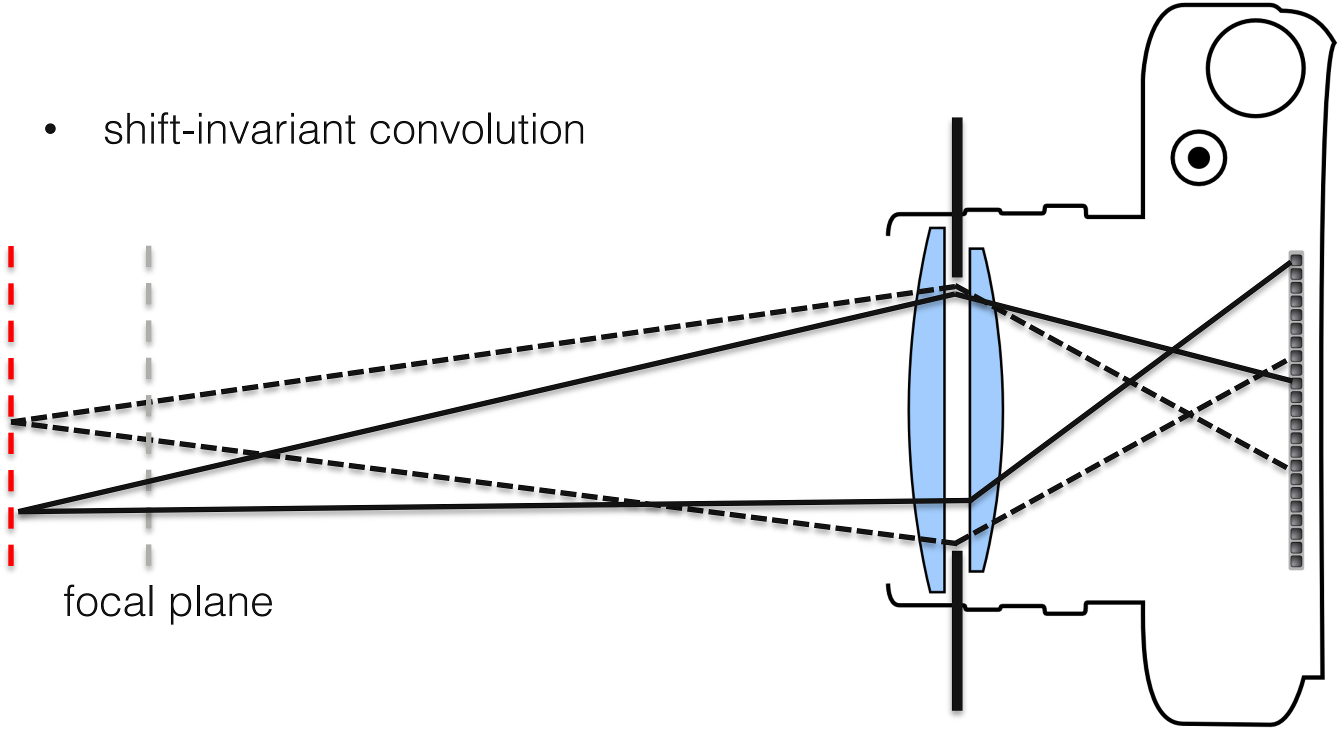
Lens as Optical Low-pass Filter

- away from focal plane: out of focus blur



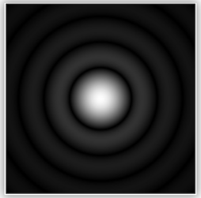
Lens as Optical Low-pass Filter

- shift-invariant convolution

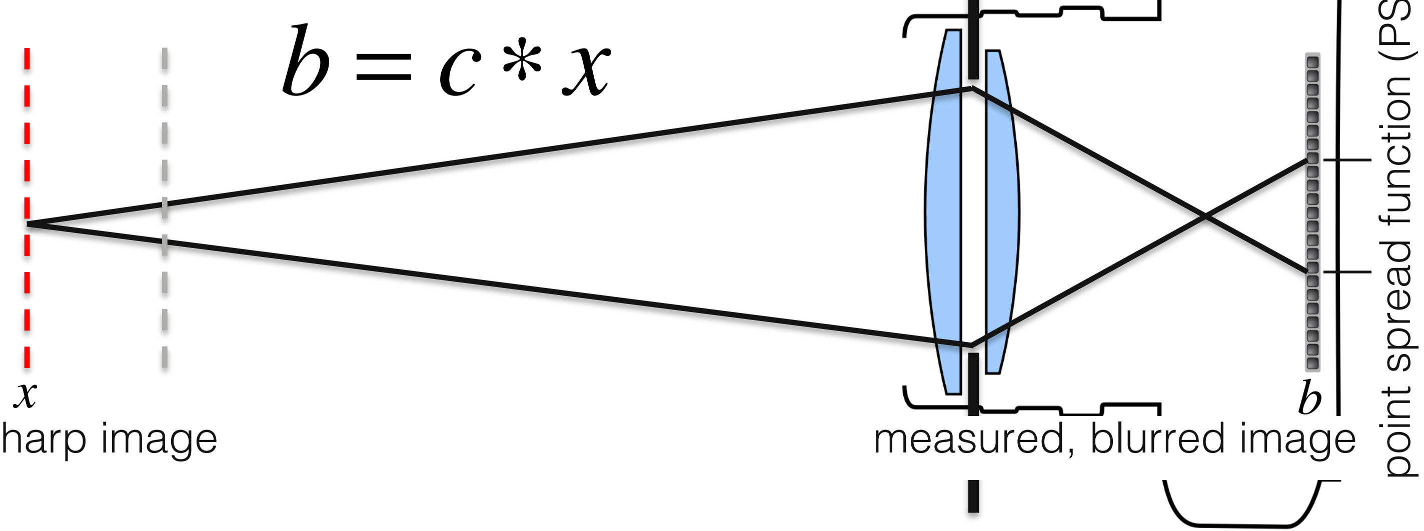


Lens as Optical Low-pass Filter

diffraction-limited PSF of circular aperture (aka “Airy” pattern):



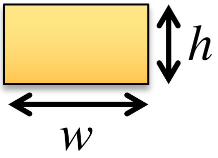
$$b = c * x$$



What's a Discrete Image?

- continuous 2D visual signal on sensor: $i(x,y)$
- integration over pixels: $\tilde{i}(x,y) = i(x,y) * \left(rect\left[\frac{x}{w}\right] \cdot rect\left[\frac{y}{h}\right] \right)$

sensor pixel:



What's a Discrete Image?

- continuous 2D visual signal on sensor: $i(x,y)$

- integration over pixels: $\tilde{i}(x,y) = i(x,y) * \left(rect\left[\frac{x}{w}\right] \cdot rect\left[\frac{y}{h}\right] \right)$

- discrete sampling: (in irradiance $\frac{W}{m^2}$) $E[i,j] = sample\left(\tilde{f}(x,y)\right) = \tilde{f}(x,y) \cdot \sum_m \sum_n \delta(i,j)$

What’s a Discrete Image?

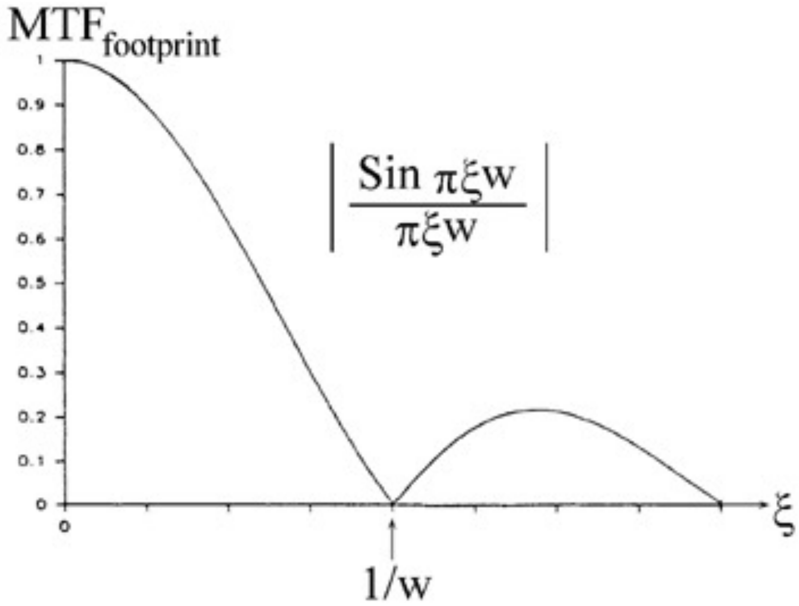
- continuous 2D visual signal on sensor: $i(x,y)$

- integration over pixels: $\tilde{i}(x,y) = i(x,y) * \left(rect\left[\frac{x}{w}\right] \cdot rect\left[\frac{y}{h}\right] \right)$

- discrete sampling: (in irradiance $\frac{W}{m^2}$) $E[i,j] = sample\left(\tilde{f}(x,y)\right) = \tilde{f}(x,y) \cdot \sum_m \sum_n \delta(i,j)$

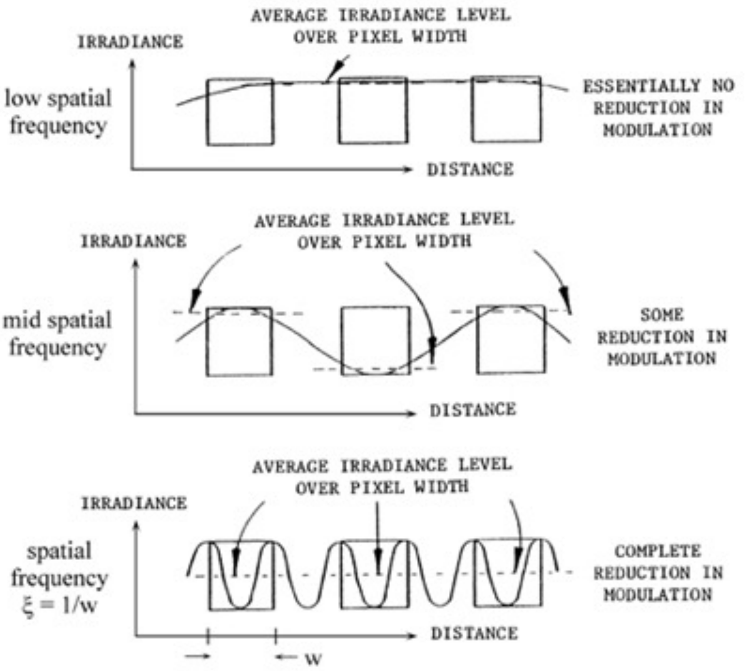
What does this mean for image frequencies we can capture?

What’s a Discrete Image?



(detector footprint modulation transfer function, Boreman 2001)

What’s a Discrete Image?



(detector footprint modulation transfer function, Boreman 2001)

Image filtering & anti-aliasing

Filtering – Low-pass Filter

- low-pass filter: convolution in primal domain $b = x * c$
- convolution kernel c is also known as point spread function (PSF)

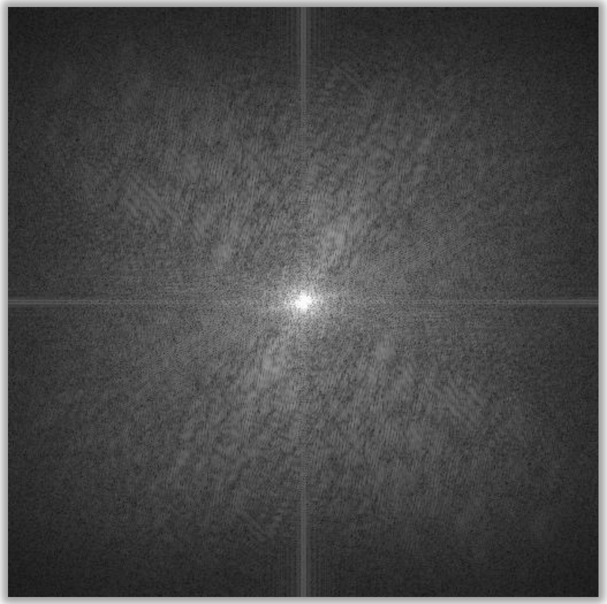
*

small kernel

=

Filtering – Low-pass Filter

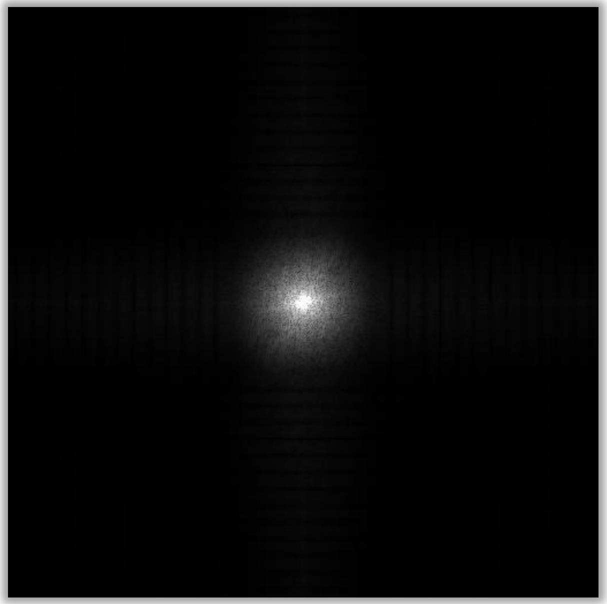
- low-pass filter: multiplication in frequency domain $F\{b\} = F\{x\} \cdot F\{c\}$



·

big

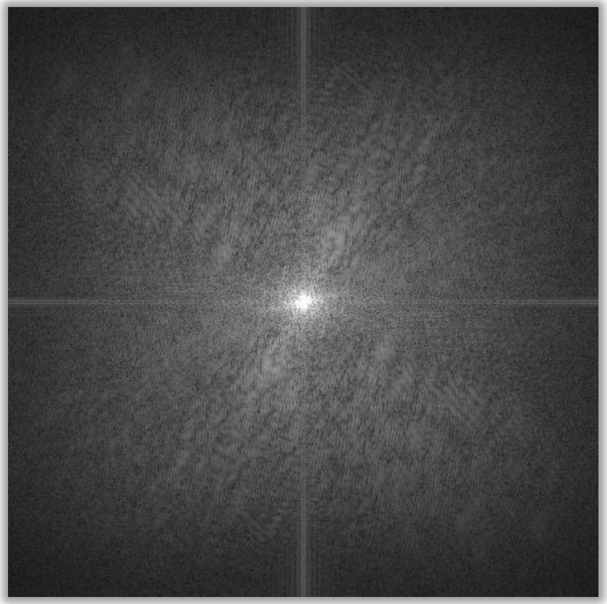
=



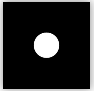
Filtering – Low-pass Filter

$$F\{b\} = F\{x\} \cdot F\{c\}$$

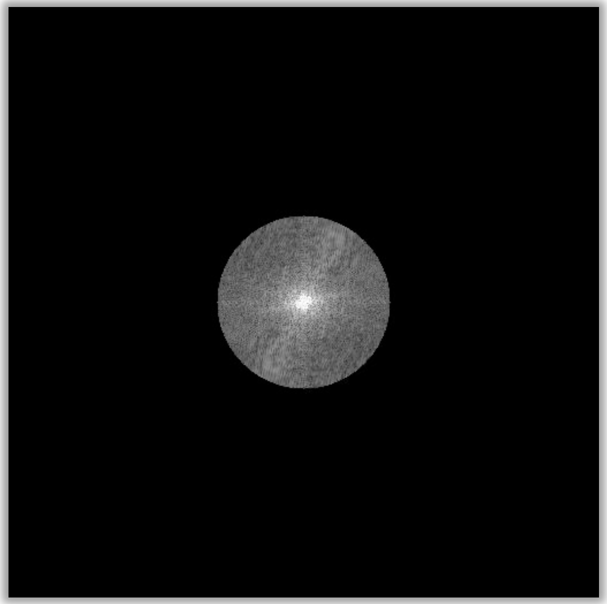
- low-pass filter: hard cutoff



$\cdot$

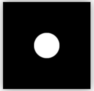


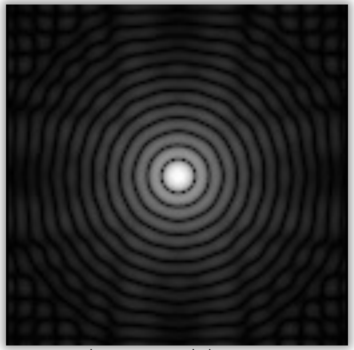
$=$



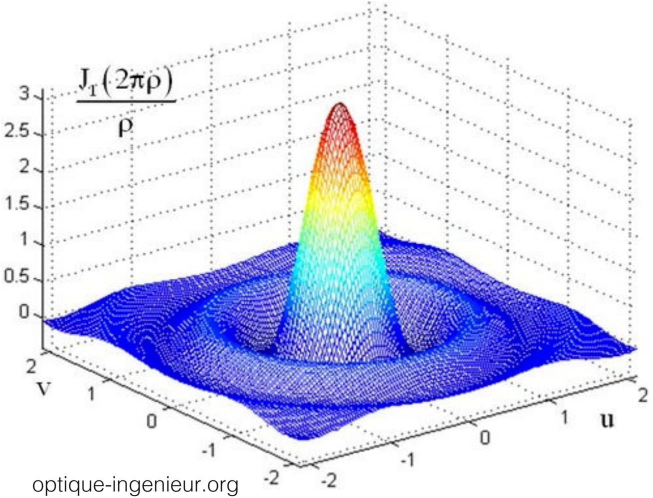
Filtering – Low-pass Filter

- Bessel function of the first kind or “jinc”

$F^{-1}\{$  $\}$



imagemagick.org

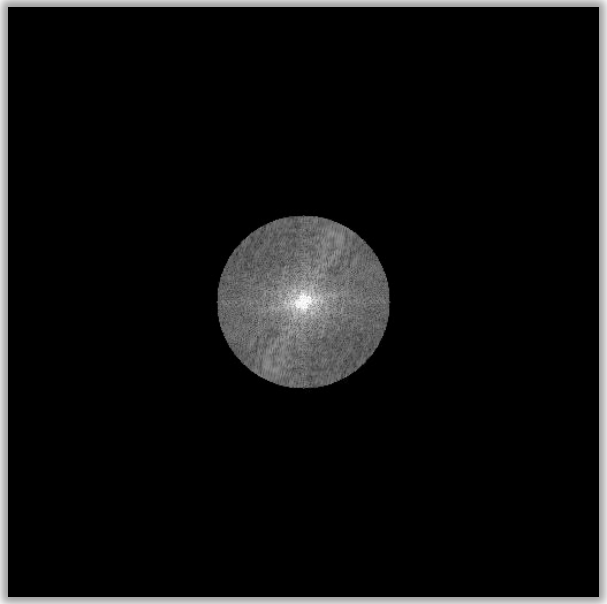


optique-ingenieur.org

Filtering – Low-pass Filter

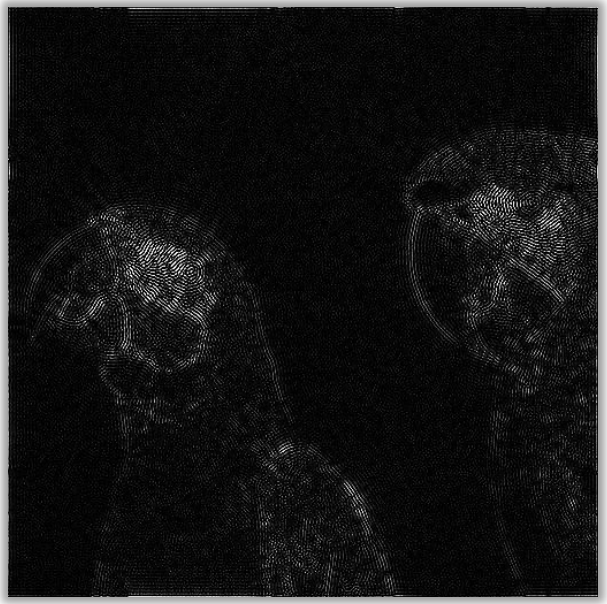
- hard frequency filters often introduce ringing

←

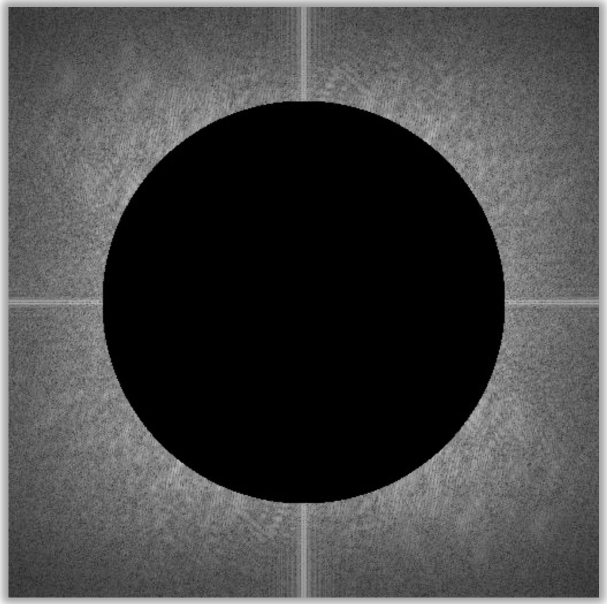


Filtering – High-pass Filter

- sharpening (possibly with ringing)



←



Filtering – Unsharp Masking

- sharpening (without ringing): unsharp masking, e.g. in Photoshop

$$b = x * (\delta - c_{lowpass\_gauss}) = x - x * c_{lowpass\_gauss}$$

or

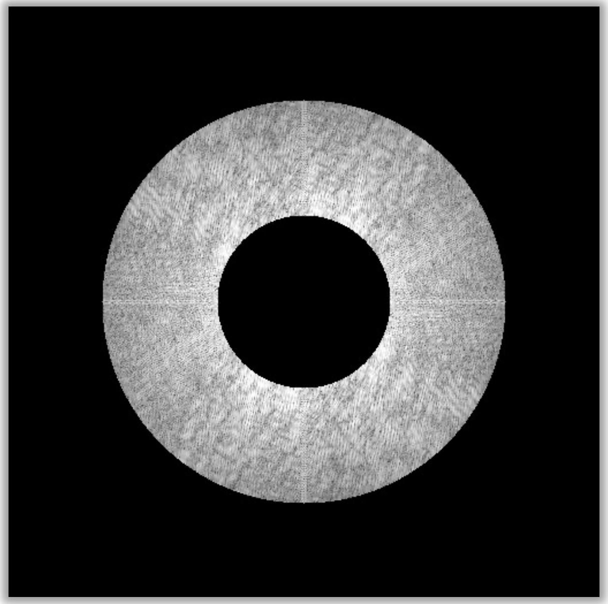
$$b = x * (\delta + c_{highpass}) = x + x * c_{highpass}$$

Filtering – Unsharp Masking

- sharpening (without ringing): unsharp masking, e.g. in Photoshop

Filtering – Band-pass Filter

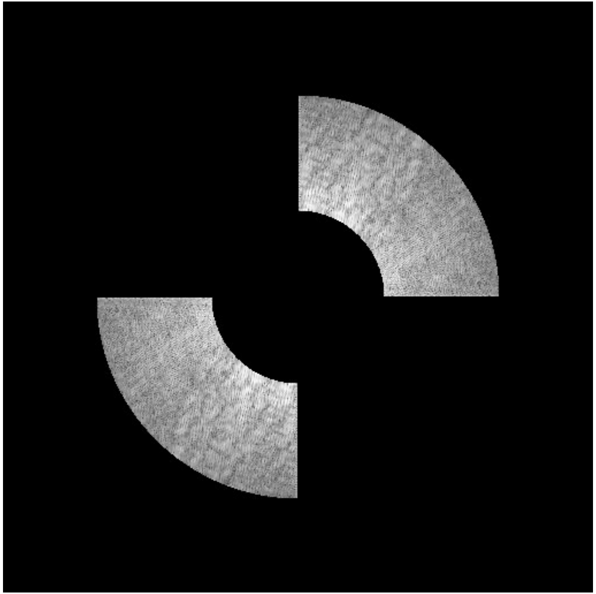
←



Filtering – Oriented Band-pass Filter

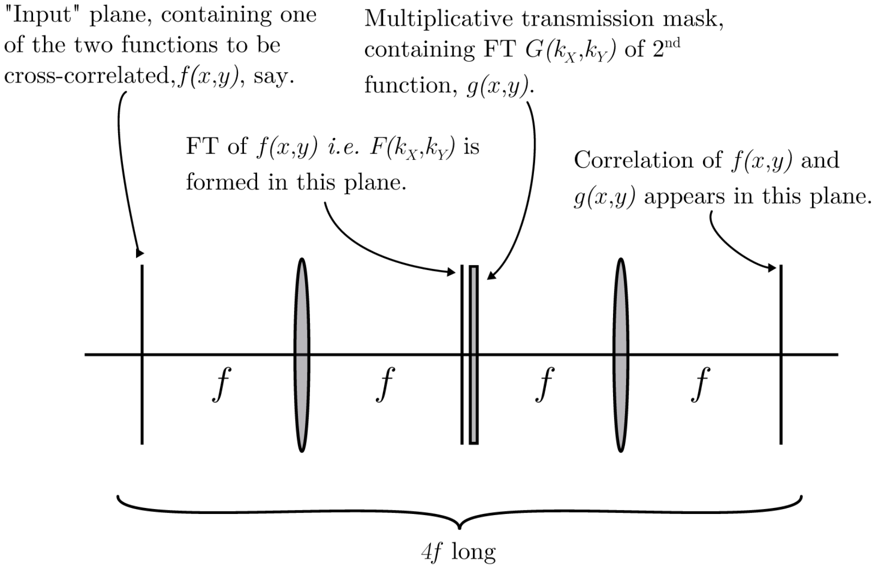
- edges with specific orientation (e.g., hat) are gone!

←



Optical Filtering with Fourier Optics

- can do all of this optically



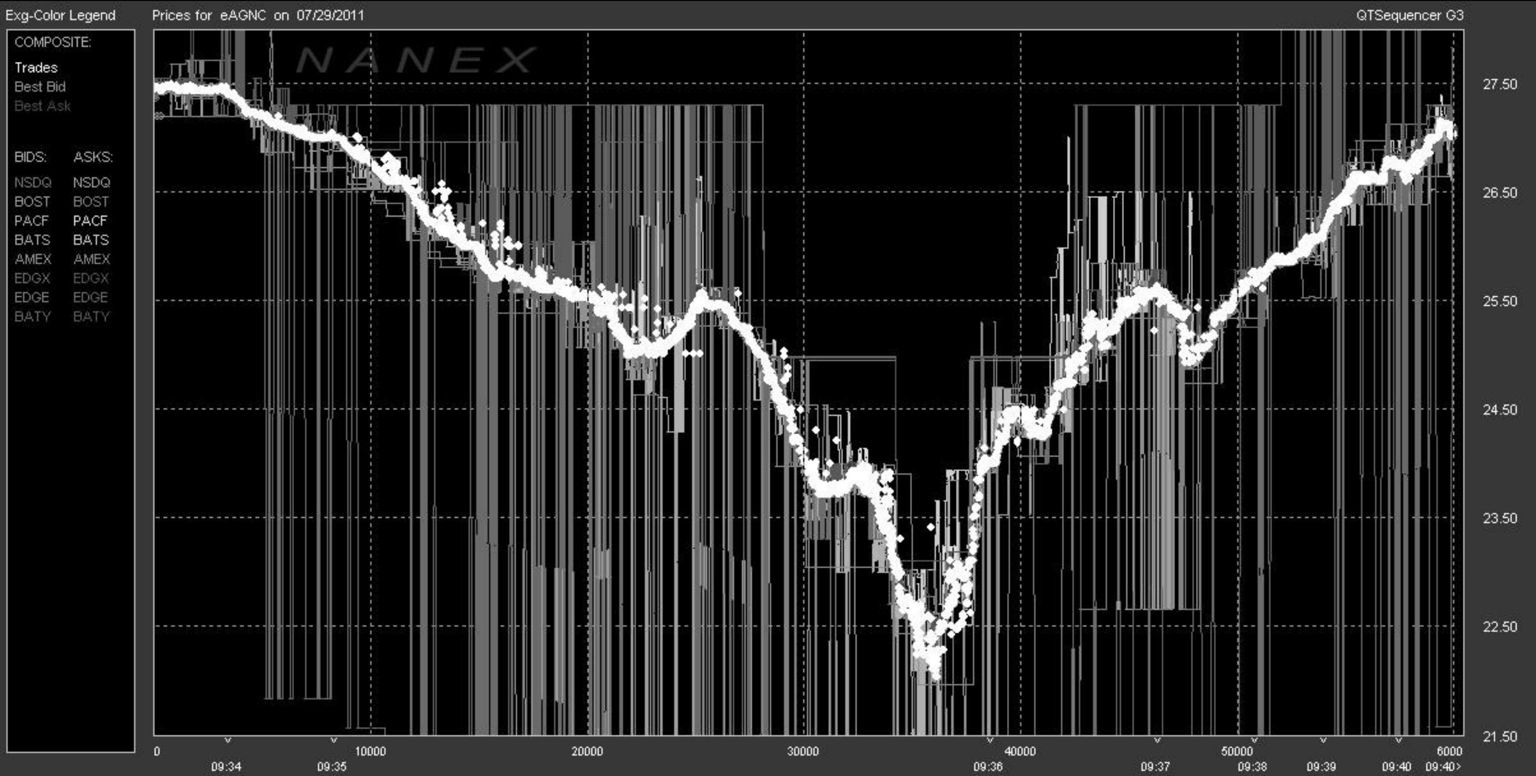
http://en.wikipedia.org/wiki/Fourier_optics

Image Downsampling (& Upsampling)

- best demonstrated with “high-frequency” image
- that’s just resampling, right?

pocketfullofliberty.com/high-frequency-trading

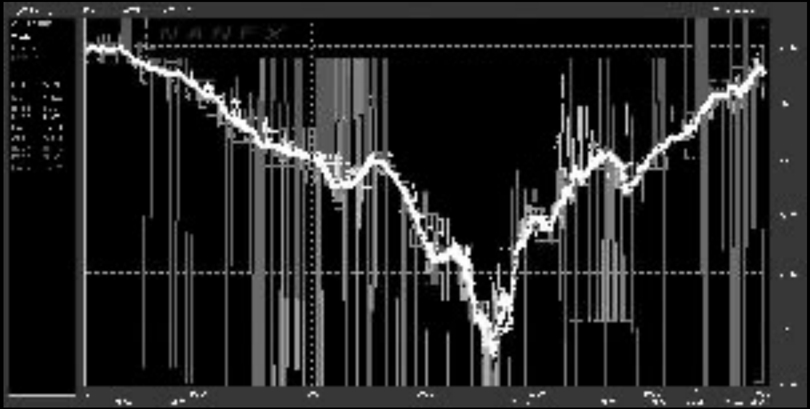
original image: I



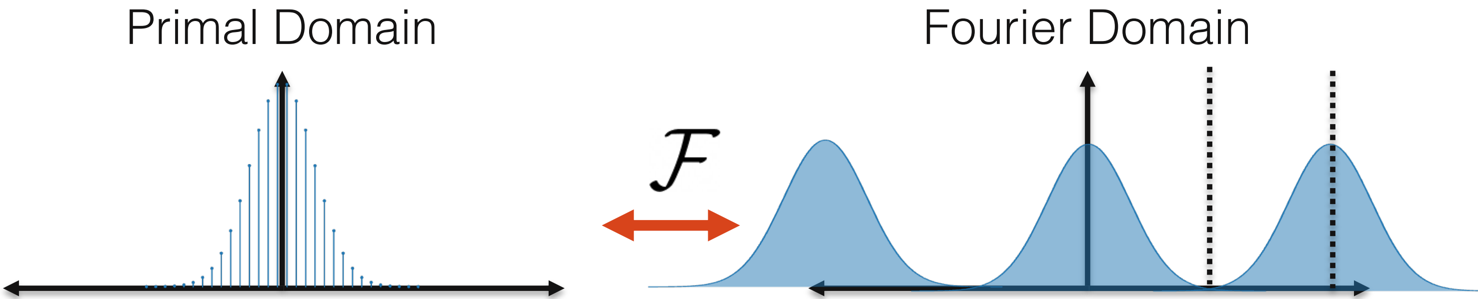
pocketfullofliberty.com/high-frequency-trading

re-sample image: I(1:4:end,1:4:end) in Matlab

something is wrong - aliasing!

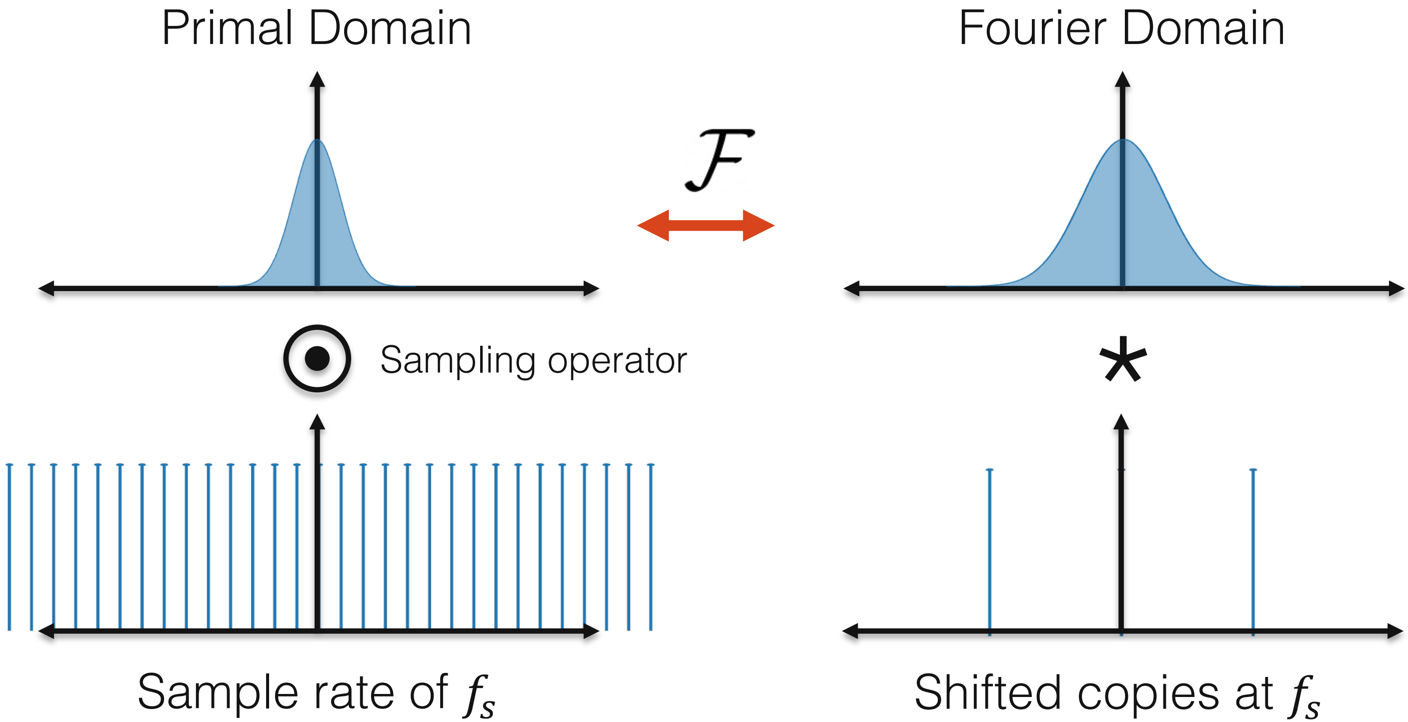


Sampling



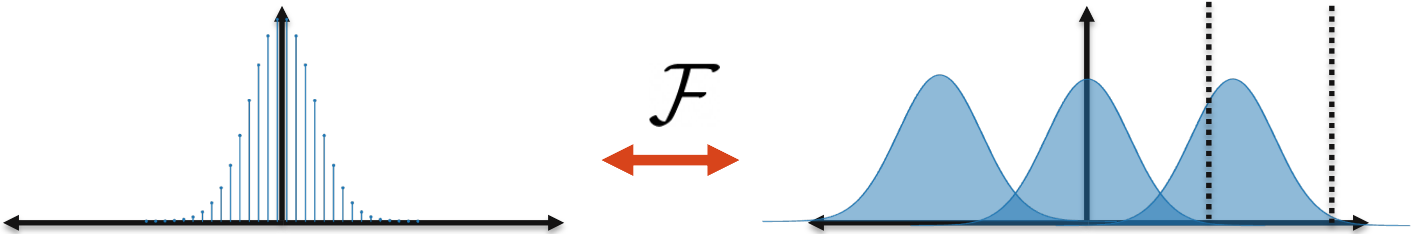
What happens if we subsample in the primal domain?

Sampling


Primal Domain
Fourier Domain
$\mathcal{F}$
Sampling operator
*
Sample rate of f_s
Shifted copies at f_s

Sampling

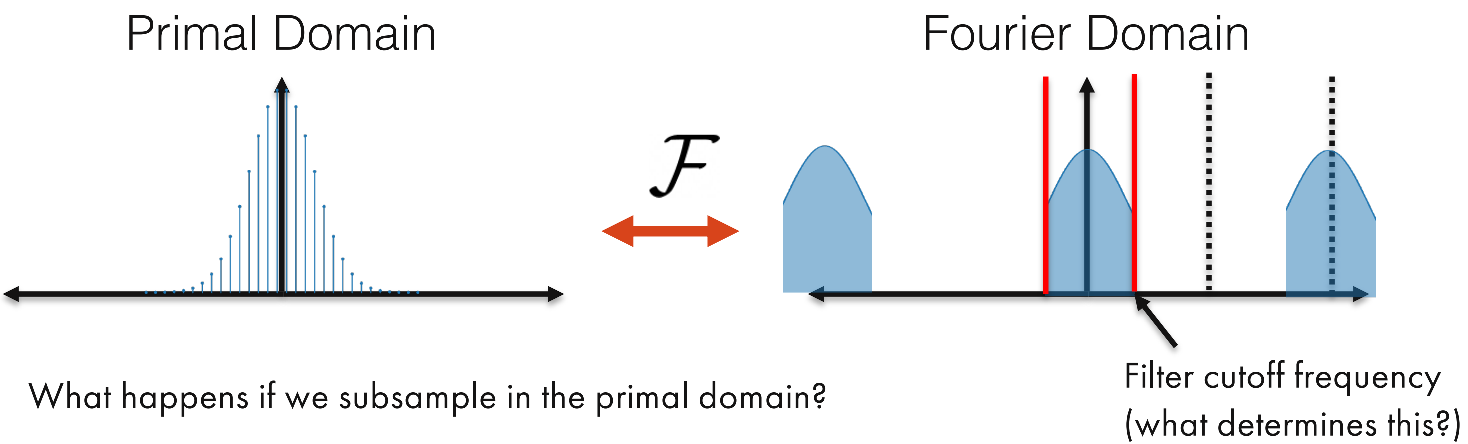
Primal Domain | Fourier Domain



What happens if we subsample in the primal domain?

- Shifted copies start to overlap! High frequencies *alias* into lower frequencies

Sampling



What happens if we subsample in the primal domain?

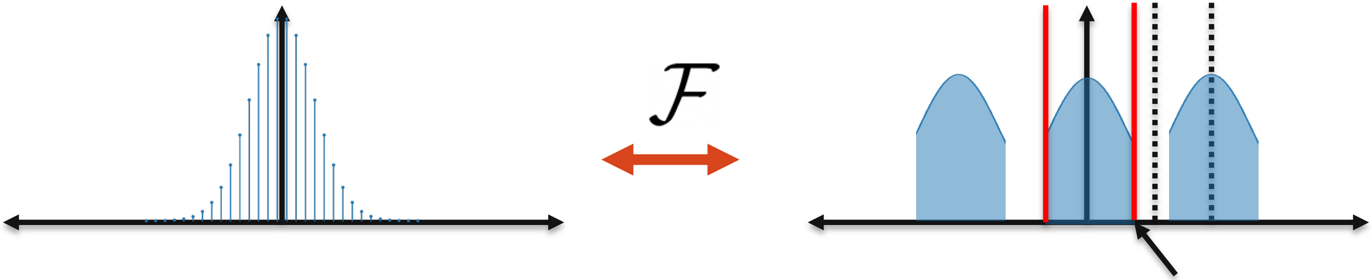
- **Shifted copies start to overlap! High frequencies *alias* into lower frequencies**

- To solve: first low-pass filter

Sampling

Primal Domain

Fourier Domain



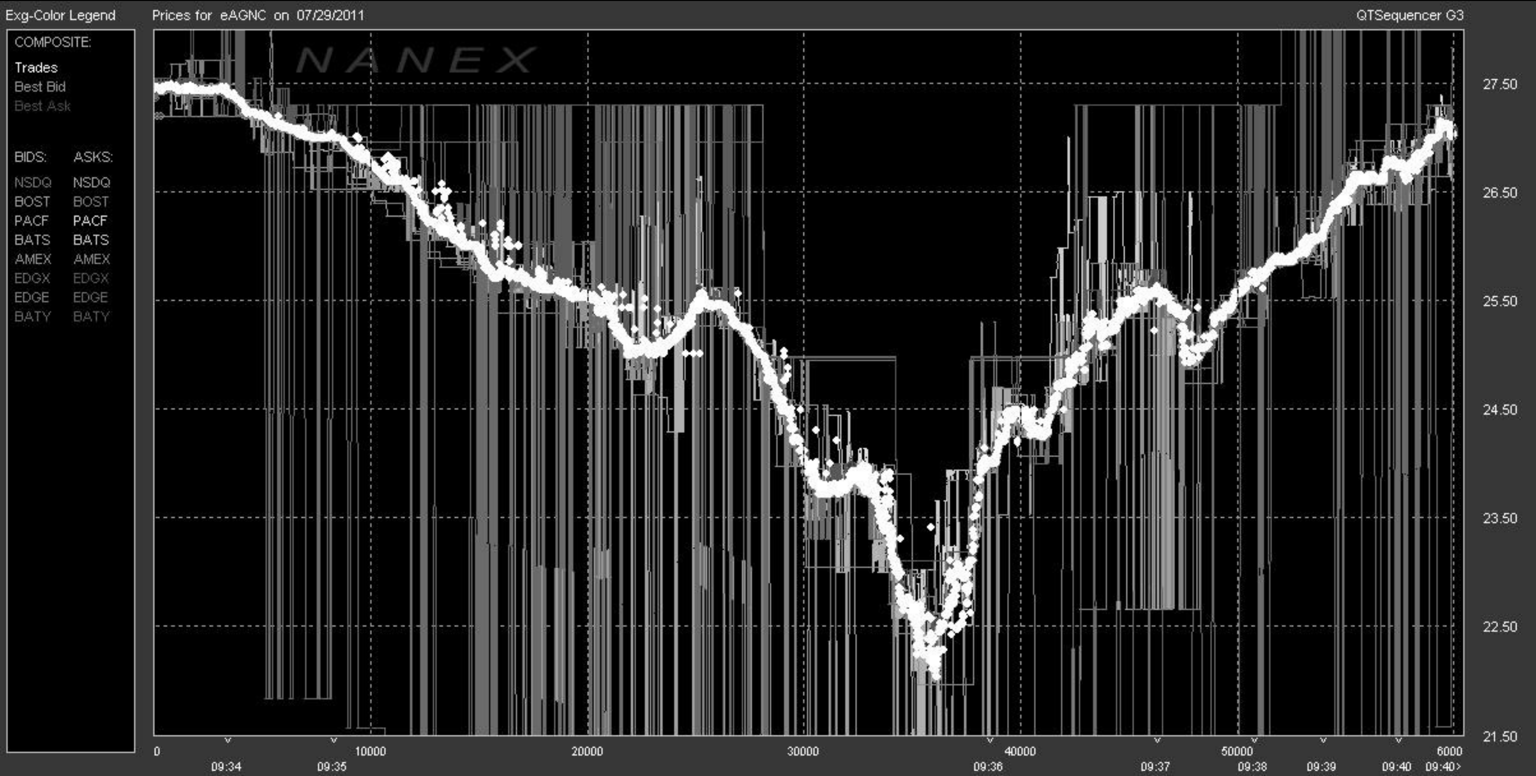
What happens if we subsample in the primal domain?

- **Shifted copies start to overlap! High frequencies *alias* into lower frequencies**

- To solve: first low-pass filter
- Then no aliasing after downsampling!

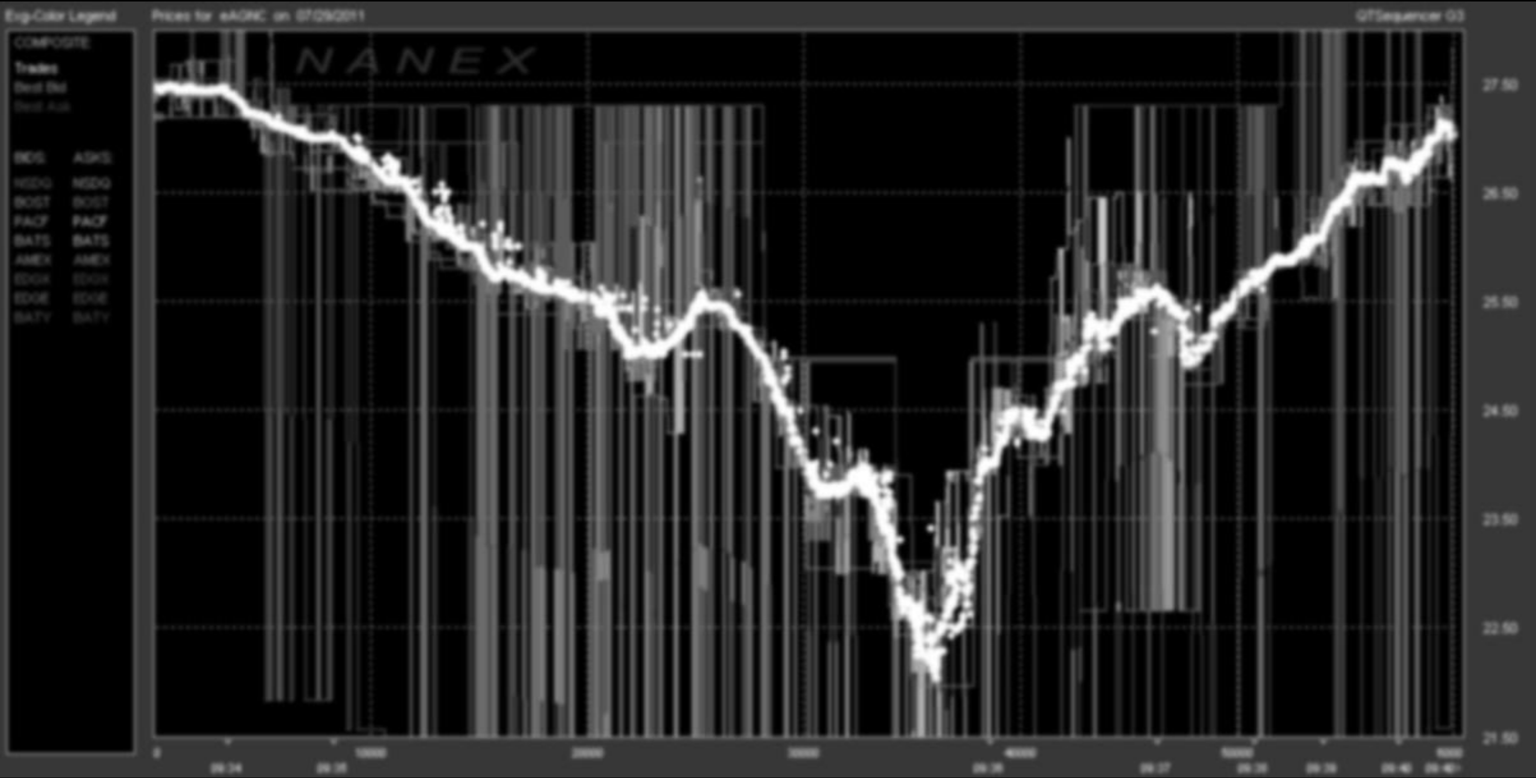
pocketfullofliberty.com/high-frequency-trading

need to low-pass filter image first!



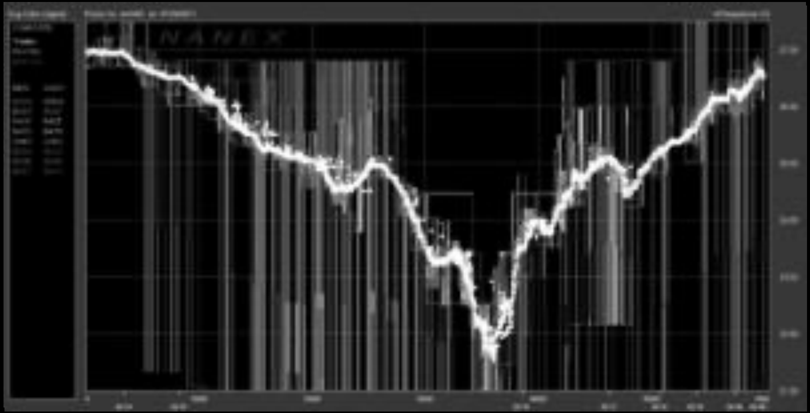
pocketfullofliberty.com/high-frequency-trading

need to low-pass filter image first!



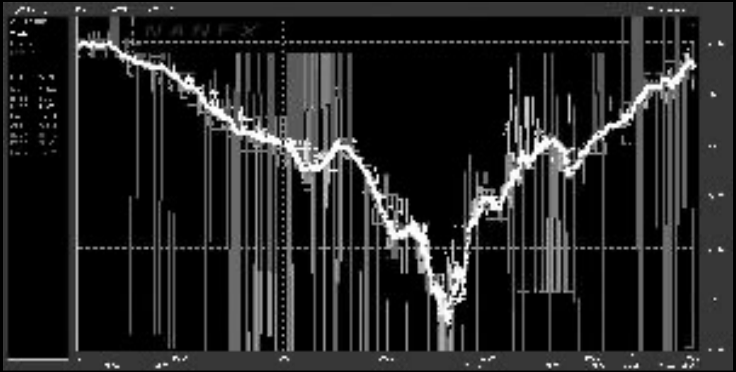
pocketfullofliberty.com/high-frequency-trading

first: filter out high frequencies (“anti-aliasing”)
then: then re-sample image: I(1:4:end,1:4:end)



pocketfullofliberty.com/high-frequency-trading

no anti-aliasing



with anti-aliasing

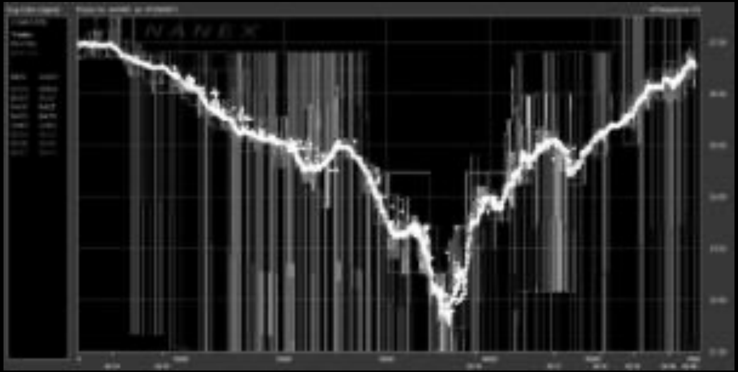
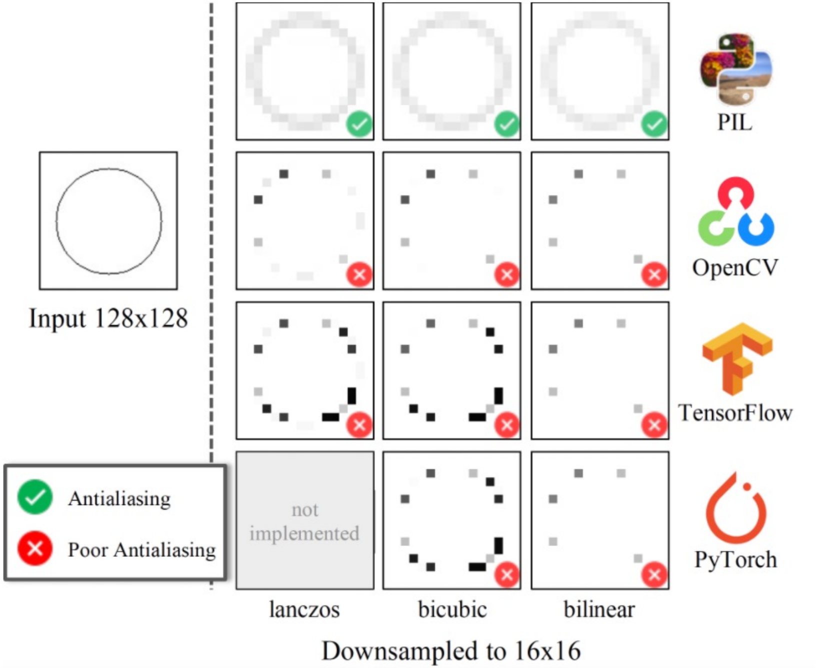


Image Downsampling (& Upsampling)

- "anti-aliasing" → **before** re-sampling, apply appropriate filter!

- how much filtering? Shannon-Nyquist sampling theorem:

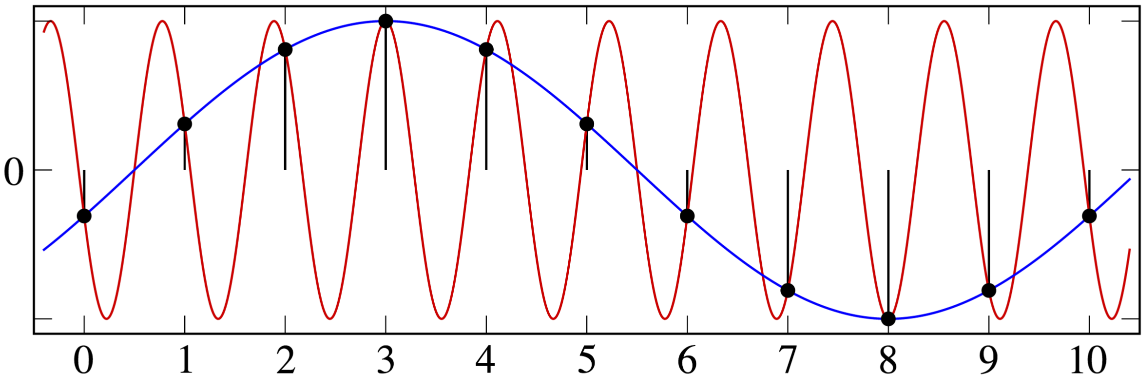
$$f_s \geq 2 f_{\max}$$



Parmar et al. 2021

Examples of Aliasing: Temporal Aliasing

- wagon wheel effect (temporal aliasing)
- sampling frequency was lower than $2f_{\text{max}}$



wikipedia

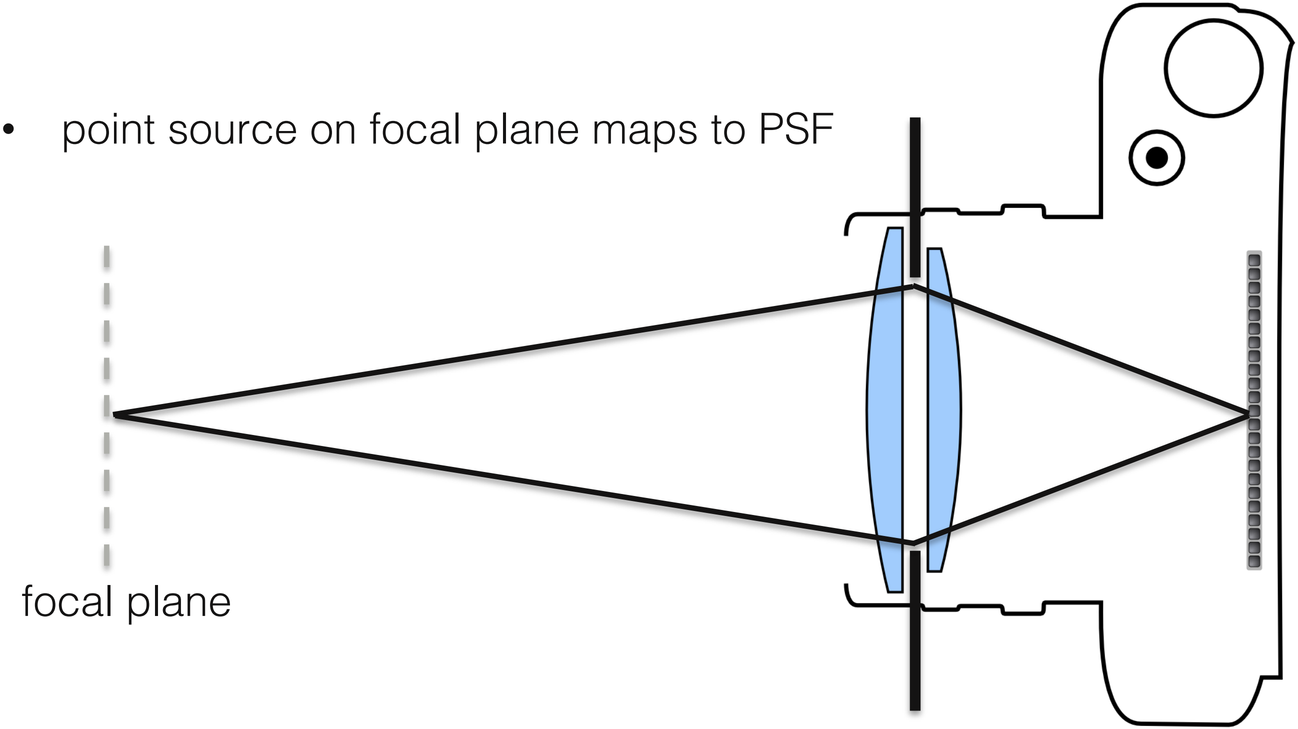
Examples of Aliasing: Temporal Aliasing

- wagon wheel effect

youtube.com/watch?v=jHS9JGkEOmA

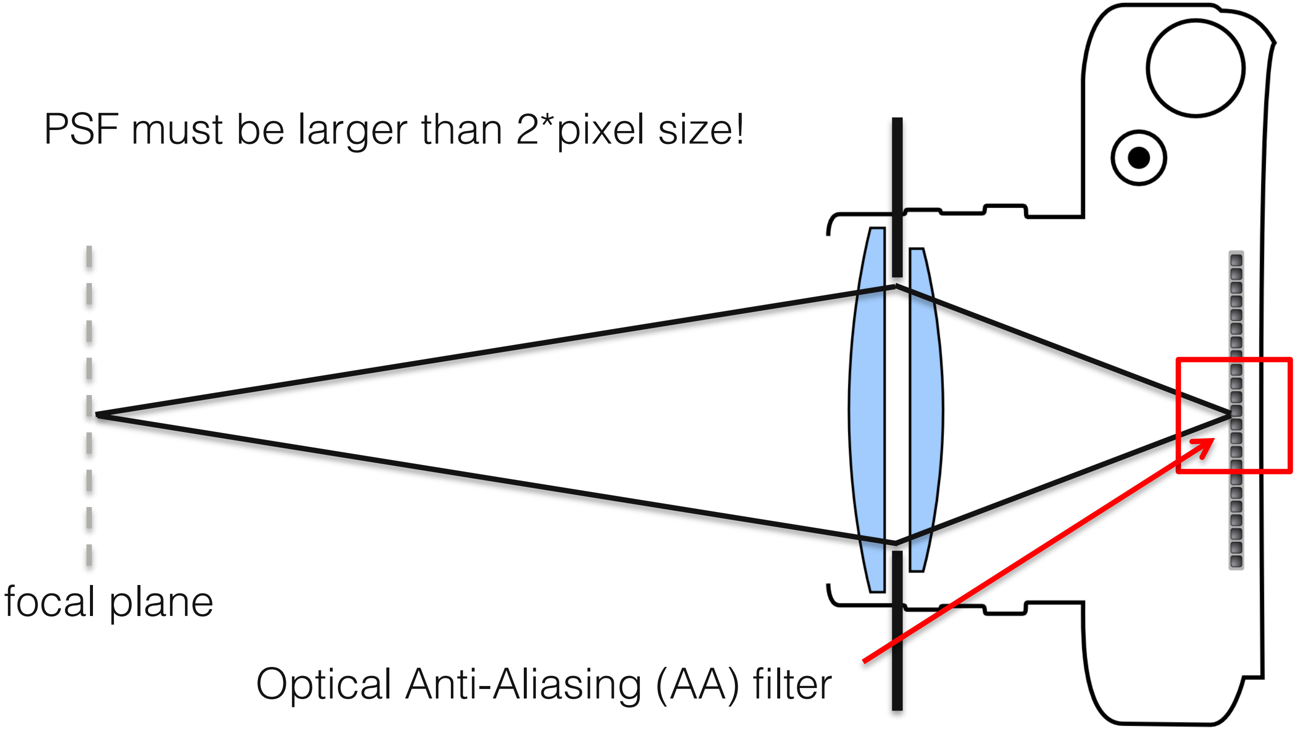
Examples of Aliasing: Sampling on Sensor

- point source on focal plane maps to PSF



Examples of Aliasing: Sampling on Sensor

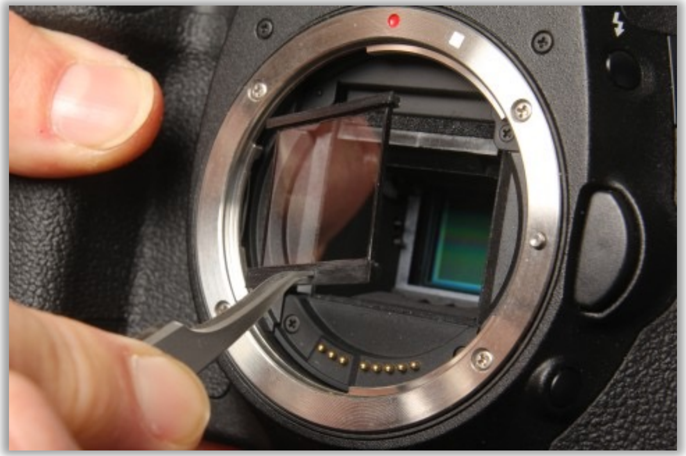
- PSF must be larger than 2*pixel size!



Other Forms of Aliasing

- photography – optical AA filter removed ("hot rodding" camera)

John Shafer

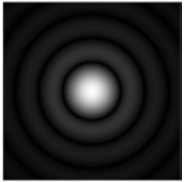


mosaicengineering.com

Deconvolution

Lens as an optical low-pass filter

*



=

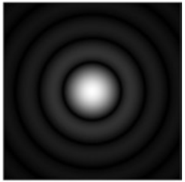
image from a perfect lens | imperfect lens PSF | image from imperfect lens

$$i * k = b$$

Lens as an optical low-pass filter

If we know b and k, can we recover i?

*



=

image from a perfect lens | imperfect lens PSF | image from imperfect lens

$$i \quad * \quad k \quad = \quad b$$

Deconvolution

$$i \quad * \quad k \quad = \quad b$$

If we know k and b, can we recover i?

Deconvolution

$$i \quad * \quad k \quad = \quad b$$

Reminder: convolution is multiplication in Fourier domain:

$$F(i) \quad \cdot \; F(k) = F(b)$$

If we know k and b, can we recover i?

Deconvolution

$$i \quad * \quad k \quad = \quad b$$

Reminder: convolution is multiplication in Fourier domain:

$$F(i) \cdot F(k) = F(b)$$

Deconvolution is division in Fourier domain:

$$F(i_{est}) = F(b) \setminus F(k)$$

After division, just do inverse Fourier transform:

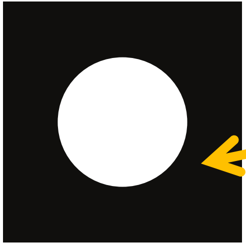
$$i_{est} = F^{-1}(\, F(b) \setminus F(k) \,)$$

Deconvolution

Any problems with this approach?

Deconvolution

- The OTF (Fourier of PSF) is a low-pass filter



zeros at high frequencies

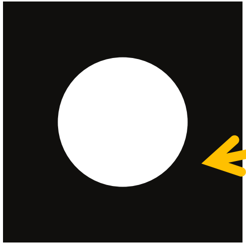
- The measured signal includes noise

$$b = k * i + n$$

noise term

Deconvolution

- The OTF (Fourier of PSF) is a low-pass filter



zeros at high frequencies

- The measured signal includes noise

$$b = k * i + n$$

noise term

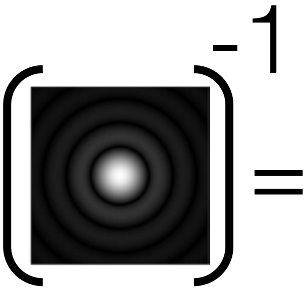
- When we divide by zero, we amplify the high frequency noise

Naïve deconvolution

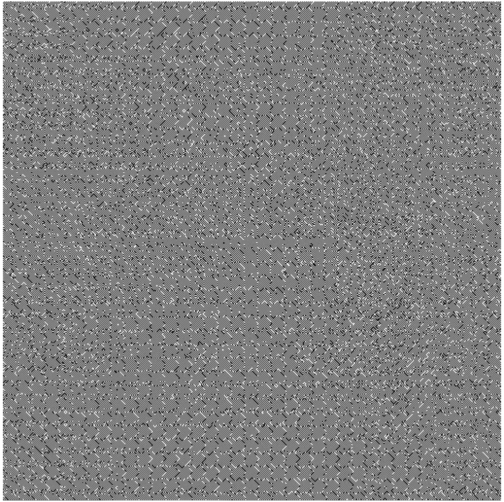
Even tiny noise can make the results awful.

- Example for Gaussian of σ = 0.05

$*$



$=$



$$b \quad * \quad k^{-1} \quad = \quad i_{est}$$

Wiener Deconvolution

Apply inverse kernel and do not divide by zero:

$$i_{est} = F^{-1}\left(\frac{|F(k)|^2}{|F(k)|^2 + 1/SNR(\omega)} \frac{F(b)}{F(k)} \right)$$

noise-dependent damping factor ↗

- Derived as solution to maximum-likelihood problem under Gaussian noise assumption
- Requires noise of signal-to-noise ratio at each frequency

$$SNR(\omega) = \frac{\text{signal variance at } \omega}{\text{noise variance at } \omega}$$

Wiener Deconvolution

Apply inverse kernel and do not divide by zero:

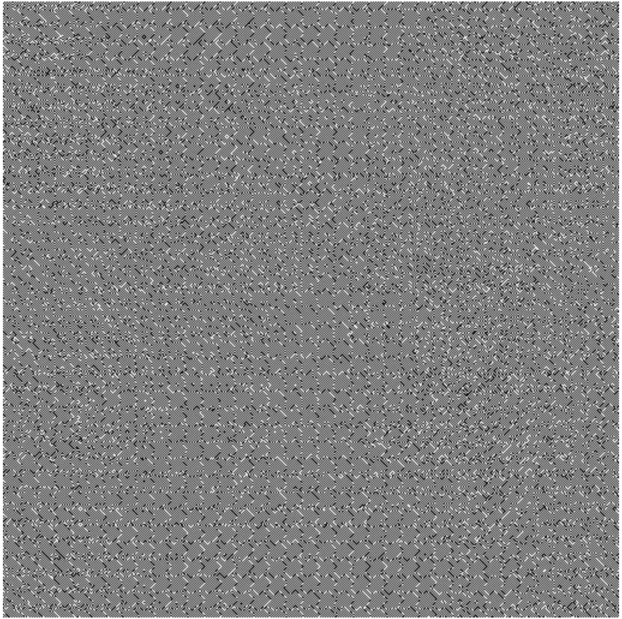
$$i_{est} = F^{-1}\left(\frac{|F(k)|^2}{|F(k)|^2 + 1/SNR(\omega)} \frac{F(b)}{F(k)}\right)$$

noise-dependent damping factor

Intuitively:

- When SNR is high (low or no noise), just divide by kernel.
- When SNR is low (high noise), just set to zero.

Deconvolution comparisons



naïve deconvolution

Wiener deconvolution

Deconvolution comparisons

σ = 0.01

σ = 0.05

σ = 0.01

Derivation

Sensing model:

$$b = k * i + n$$

Noise $\mathbf{n}$ is assumed to be zero-mean and independent of signal $\mathbf{i}$.

Derivation

Sensing model:

$$b = k * i + n$$

Noise $\mathbf{n}$ is assumed to be zero-mean and independent of signal $\mathbf{i}$.

Fourier transform:

$$B = K \cdot I + N$$

Why multiplication?

Derivation

Sensing model:

$$b = k * i + n$$

Noise $\mathbf{n}$ is assumed to be zero-mean and independent of signal $\mathbf{i}$.

Fourier transform:

$$B = K \cdot I + N$$

Convolution becomes multiplication.

Problem statement: Find function $\mathrm{H}(\omega)$ that minimizes expected error in Fourier domain.

$$\min_H E[\|I - HB\|^2]$$

Derivation

Replace B and re-arrange loss:

$$\min_H E[||(1 - HK)I - HN||^2]$$

Expand the squares:

$$\min_H ||1 - HK||^2 E[||I||^2] - 2H(1 - HK)E[IN] + ||H||^2 E[||N||^2]$$

Derivation

When handling the cross terms:

- Can I write the following?

$$E[IN] = E[I]E[N]$$

Derivation

When handling the cross terms:

- Can I write the following?

$$E[IN] = E[I]E[N]$$

Yes, because I and N are assumed independent.

- What is this expectation product equal to?

Derivation

When handling the cross terms:

- Can I write the following?

$$E[IN] = E[I]E[N]$$

Yes, because I and N are assumed independent.

- What is this expectation product equal to?

Zero, because N has zero mean.

Derivation

Replace B and re-arrange loss:

$$\min_{H} E[\|(1+HK)I - HN\|^2]$$

Expand the squares:

$$\min_{H} \|1 - HK\|^2 E[\|I\|^2] - 2H(1 - HK)E[IN] + \|H\|^2 E[\|N\|^2]$$

cross-term is zero

Simplify:

$$\min_{H} \|1 - HK\|^2 E[\|I\|^2] + \|H\|^2 E[\|N\|^2]$$

How do we solve this optimization problem?

Derivation

Differentiate loss with respect to H, set to zero, and solve for H:

$$\frac{\partial \mathrm{loss}}{\partial H} = 0$$

$$\Rightarrow -2K(1 - HK)E[\|I\|^2] + 2HE[\|N\|^2] = 0$$

$$\Rightarrow H = \frac{KE[\|I\|^2]}{K^2E[\|I\|^2] + E[\|N\|^2]}$$

Divide both numerator and denominator with $E[\|I\|^2]$, extract factor $1/\mathrm{K}$, and done!

Deconvolution with Wiener Filtering

- results: not too bad, but noisy
- need more advance image priors to solve this ill-posed inverse problem robustly → more in week 7&8

Sampling & Deconvolution – Summary

- Shannon-Nyquist theorem: always sample signal at a sampling rate >= 2*highest frequency of signal!
- if Shannon-Nyquist is violated, aliasing occurs
- aliasing cannot be corrected digitally in post-processing (see optical anti-aliasing filter)
- PSF is usually a low-pass filter, so deconvolution is an ill-posed inverse problem ☹

Linear systems review

Matrices and Linear Systems – Review

- basic linear algebra, review if necessary!
- see references for online resources
- brief review now

Matrices and Linear Systems – Review

- most computational imaging problems are linear
- geometric optics approximation of light is linear in intensity
- not necessarily true for wave-based models (e.g. interference, phase retrieval, …)

Matrices and Linear Systems – Review

- most computational imaging problems are linear

$$b = Ax$$

blurry, noisy, or otherwise corrupted measurements (b)

matrix modeling image formation, usually known (A)

unknown image (x)

Matrices and Linear Systems – Review

- common problem: given b, what can I hope to recover?
- answer: analyze matrix via condition number, rank, SVD → please review these concepts

$$b = Ax$$

blurry, noisy, or otherwise corrupted measurements

unknown image

matrix modeling image formation, usually known

Matrices and Linear Systems – Review

- other common problem: given b, what is x?
- answer: invert matrix?

$$b = Ax$$

$$x_{est} \stackrel{?}{=} A^{-1}b$$

Matrices and Linear Systems – Review

- other common problem: given b, what is x?
- answer: invert matrix – generally not!

$$b = Ax$$

$$x_{est} \stackrel{?}{=} A^{-1}b$$

Linear Systems

- problem 1: matrix inverse only defined for square, full-rank matrices – most imaging problems are NOT!

- problem 2: most imaging problems deal with really big matrices – couldn’t compute inverse, even if there was one!

- solution: iterative (convex) optimization

Linear Systems

- case 1: over-determined system = more measurements than unknowns
  $$A \in \mathbb{R}^{m \times n}, m > n$$
- case 2: under-determined system = fewer measurements than unknowns
  $$A \in \mathbb{R}^{m \times n}, m < n$$

Linear Systems

- case 1: over-determined system = more measurements than unknowns

$A \in \mathbb{R}^{m \times n}, m > n$

- formulate least-squared error objective function:

$$\underset{x}{\text{minimize}} \frac{1}{2} \left\| b - Ax \right\|_2^2 \qquad \left\| r \right\|_2^2 = \sum_i r_i^2, \quad r = b - Ax$$

ℓ_2 norm ↗   ↑ residual

Linear Systems

- least squares solution: gradient of objective = 0

- gradient:

$$\nabla_x \frac{1}{2}\|b - Ax\|_2^2 = \nabla_x \frac{1}{2}\left(b^T b - 2b^T Ax + x^T A^T Ax\right) = A^T Ax - A^T b$$

- equate to zero – normal equations:

$$A^T Ax = A^T b$$

Linear Systems

- least squares solution: gradient of objective = 0

- gradient:

$$\nabla_x \frac{1}{2}\|b - Ax\|_2^2 = \nabla_x \frac{1}{2}\left(b^T b - 2b^T Ax + x^T A^T Ax\right) = A^T Ax - A^T b$$

- equate to zero – normal equations:

$$A^T Ax = A^T b$$

$$A^T(Ax - b) = 0$$

The residual is “normal” to the columns of A

Linear Systems

- case 2: under-determined system = fewer measurements than unknowns $A \in \mathbb{R}^{m \times n}, m < n$

- $A^T A$ not invertible

- regularized solution $x_{est} = \left(A^T A + \lambda I\right)^{-1} A^T b$

(always full rank, but still too big to directly invert, equivalent to least norm solution)

Linear Systems – Gradient Descent

- solve with iterative method, easiest one: gradient descent

$$\left(\underbrace{A^T A + \lambda I}_{\tilde{A}}\right) x = \underbrace{A^T b}_{\tilde{b}}$$

- use the negative gradient of objective as descent direction at iteration *k*, with step length α

$$x^{(k+1)} = x^{(k)} - \alpha \nabla_x = x^{(k)} - \alpha \tilde{A}^T \left(\tilde{A} x^{(k)} - \tilde{b} \right)$$

Linear Systems – Gradient Descent

- use the negative gradient of objective as descent direction at iteration *k, with* step length α

$$x^{(k+1)} = x^{(k)} - \nabla_x = x^{(k)} - \alpha A^T\left(Ax^{(k)} - b\right)$$

- for large-scale problems, implement as function handles!

Linear Systems – Gradient Descent

- back to convolution example:

$$x^{(k+1)} = x^{(k)} - \nabla_x = x^{(k)} - \alpha A^T\left(Ax^{(k)} - b\right)$$
$$= x^{(k)} - \alpha\left(c^* * \left(c * x^{(k)} - b\right)\right)$$

- efficient implementation using convolution theorem:

$$x^{(k+1)} = x^{(k)} - \alpha\, F^{-1}\left\{F\{c\}^* \cdot \left(F\{c\} \cdot F\left\{x^{(k)}\right\} - F\{b\}\right)\right\}$$

Linear Systems – Stochastic Gradient Descent

$$b = Ax$$

- What if our measurements are too large to store in memory?
- Can happen for linear models—very common for nonlinear models (neural networks)!
- Will see more on this later…

Linear Systems – Stochastic Gradient Descent

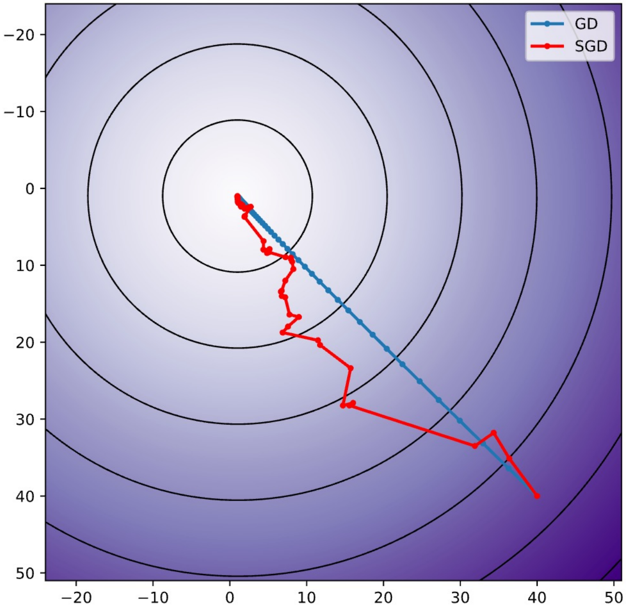
$$b = Ax$$

- Solution?
  - Stochastic optimization by sampling entries/rows from b and A at each iteration

$$\tilde{b} = \tilde{A}x$$

$$x^{(k+1)} = x^{(k)} - \alpha \tilde{A}^{(k)T} (\tilde{A}^{(k)} x^{(k)} - \tilde{b}^{(k)})$$

Linear Systems – Stochastic Gradient Descent



Tradeoffs

GD is expensive

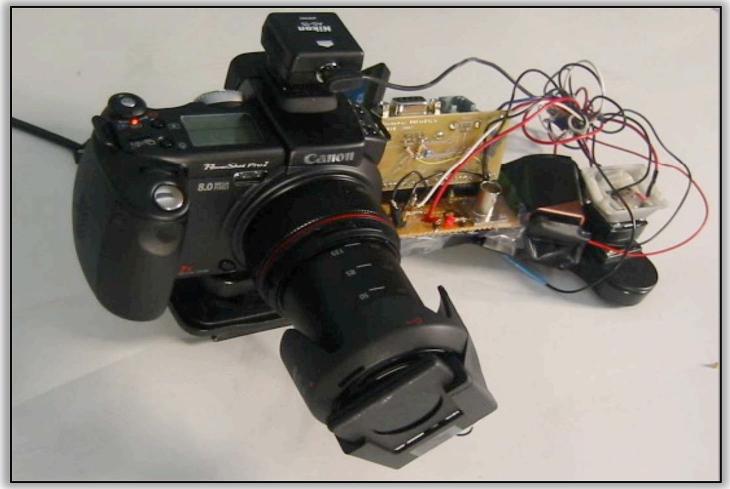
- but better convergence

SGD is more efficient

- works well far from minima
- but struggles close to minima
- can be good for non-convex problems!

Next: Computational Photography

HDR Imaging &
Tone Mapping



Coded Apertures

References and Further Reading

- Boreman, “Modulation Transfer Function in Optical and ElectroOptical Systems”, SPIE Publications, 2001
- http://www.imagemagick.org/Usage/fourier/
- Wikipedia
- Stanford EE263 lectures: https://www.youtube.com/playlist?list=PL06960BA52D0DB32B