Introduction to Medical Imaging

Lecture 4: Sampling

Klaus Mueller

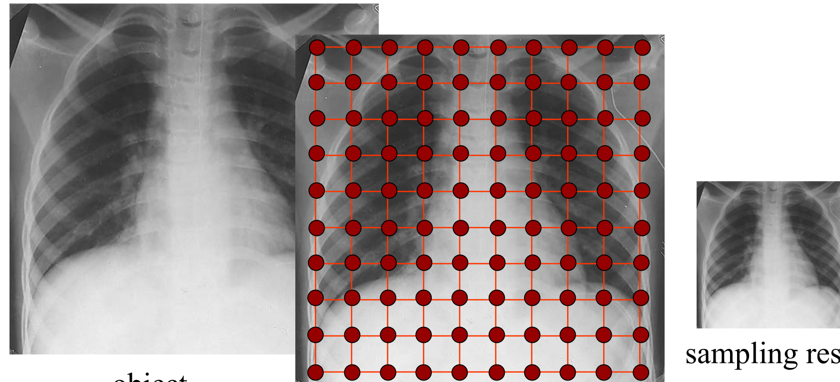
Computer Science Department

Stony Brook University

Introduction

Sampling is the process of discretizing a continuous function into an array/matrix of data points

- the matrix values are some function of the sampled real-life object
- this function is given by the *sampling filter* (more to follow)



Importance of the Fourier Domain

Visual artifacts are also often easier understood in the Fourier domain

We can use the Fourier domain to:

- gain insight into the spatial / temporal frequency content of the data (see last lecture)
- from this, gain insight into how much a continuous signal must be sampled when it is discretized
- design proper filters to avoid an important phenomenon: *aliasing*

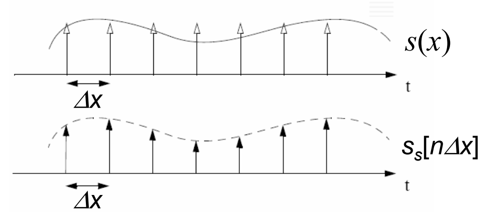
We usually do not use the Fourier domain to:

- perform the actual signal filtering, sampling, resampling, reconstruction (there are exceptions, however)
- these real operations are usually performed in the original signal domain (spatial, temporal)

Sampling: Spatial Domain

Definition:

- a continuous signal $s(x)$ is measured at fixed instances spaced apart by an interval Δx
- the data points so obtained form a discrete signal $s_s[n\Delta x] = s_s(n\Delta x)$
- here, Δx is called the *sampling period (distance)*, and $K = 1/\Delta x$ the *sampling frequency*



Sampling is the multiplication of the signal with an impulse train:

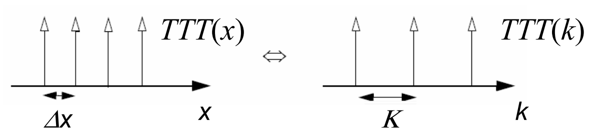
$$s_s(x) = s(x) \cdot \text{TTT}(x)$$

$$\text{TTT}(x) = \sum_{n=-\infty}^{+\infty} \delta(x - n\Delta x), \quad \text{TTT}(x) \text{ is the comb function}$$

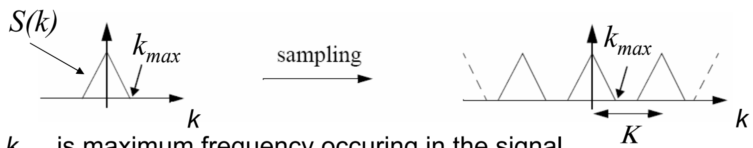
Sampling: Frequency Domain

Using the convolution theorem of the Fourier transform:

$$S_s(k) = S(k) * F\{\text{TTT}(x)\}, \quad \text{where } F\{\text{TTT}(x)\} = K \sum_{l=-\infty}^{+\infty} \delta(k - lK)$$



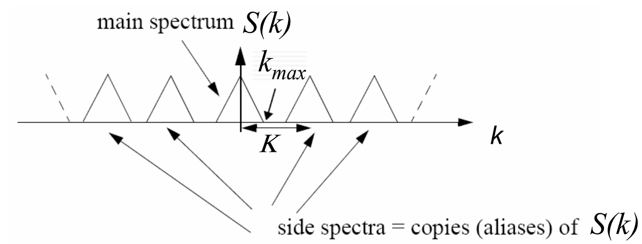
- the smaller Δx the wider K (recall the Fourier scaling theorem)
- sampling (the convolution of $TTT(k)$ and $S(k)$) replicates the signal spectrum $S(k)$ at integer multiples of sampling frequency K



- k_{max} is maximum frequency occuring in the signal

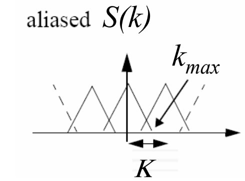
Aliasing

Terminology:



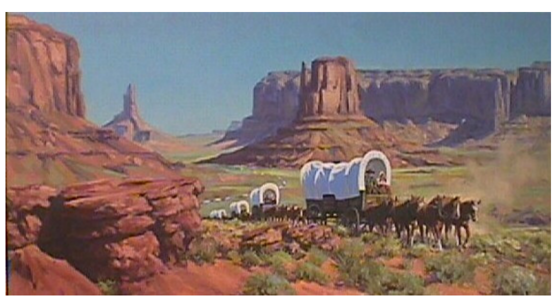
However, if we choose $K < 2\,k_{max}$ the aliases overlap and we get *aliasing*

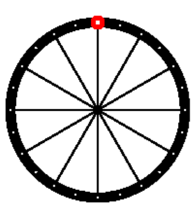
- what does aliasing look like?
- let's see some examples



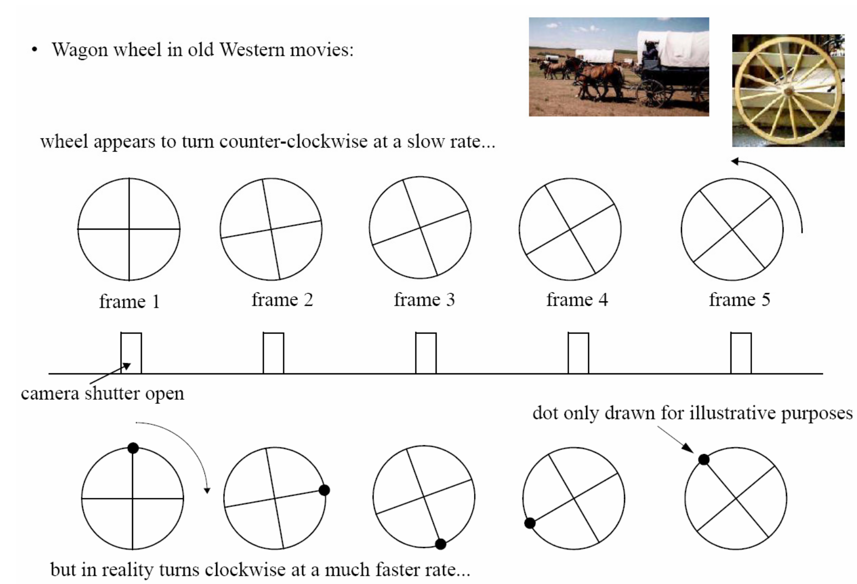
Aliasing: A Commonly Observed Phenomenon

Ever wondered about the wagon wheels in old Western movies:



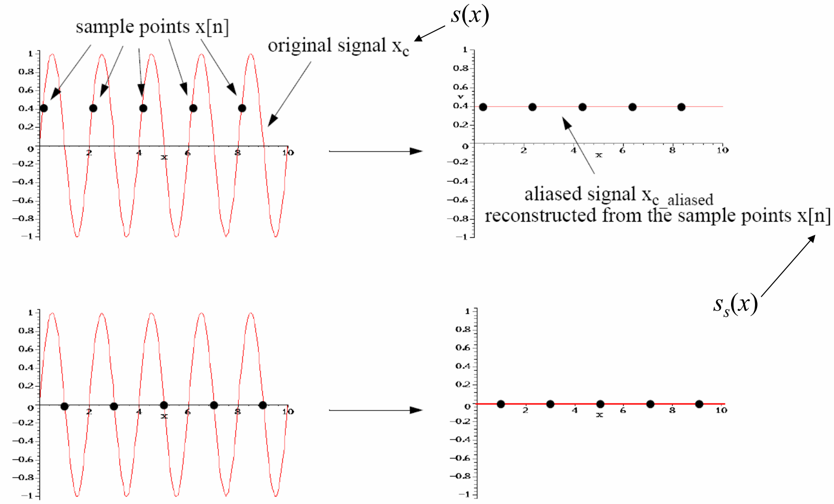


Aliasing: A Commonly Observed Phenomenon



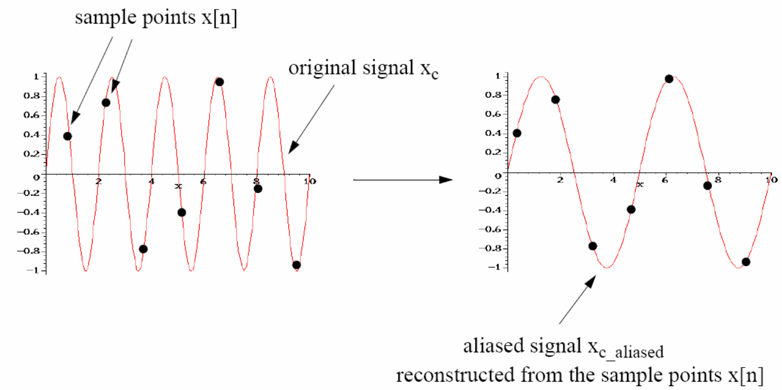
Aliasing: A More Analytical Example (1)

- Frequency of original signal: 0.5 (oscillations per time unit)
- Sampling frequency: 0.5 (samples per time unit) → original signal can not be recovered



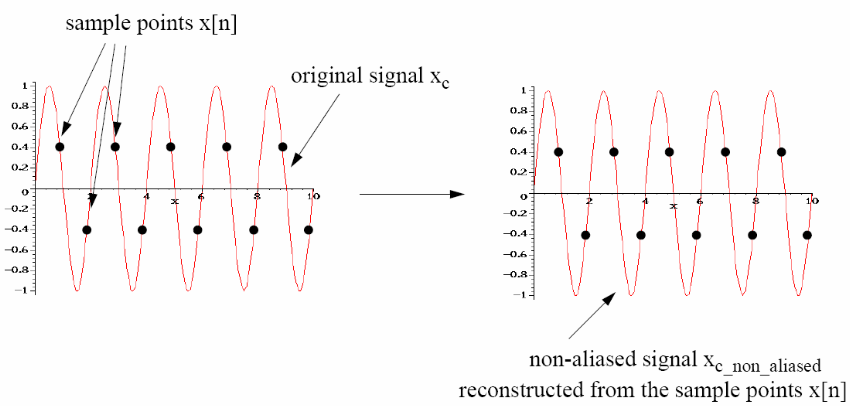
Aliasing: A More Analytical Example (2)

- Frequency of original signal: 0.5 (oscillations per time unit)
- Sampling frequency: 0.7 (samples per time unit)
- Looking at the sample points x[n], they appear to originate from a sine wave $x_{c\_aliased}$ of much lower frequency → again, the original sine wave is lost and can not be recovered



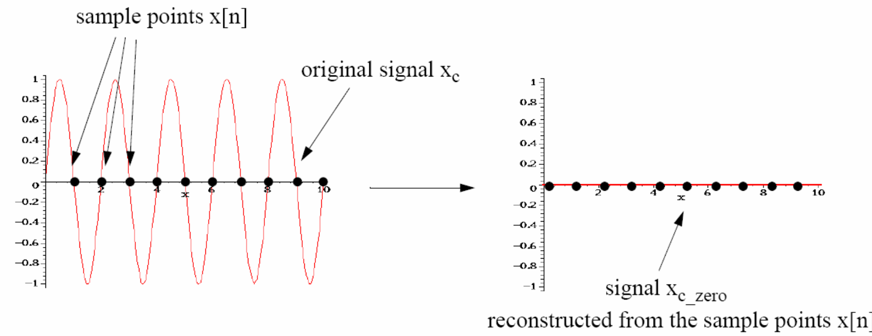
Aliasing: A More Analytical Example (3)

- Frequency of original signal: 0.5 (oscillations per time unit)
- Sampling frequency: 1.0 (sample per time unit) → original signal can be recovered
- We learn that we need to sample each oscillation period twice for good reconstruction



Aliasing: A More Analytical Example (4)

- In practice, it is best to use more than 2 samples per oscillation period
  - else one may get wrong reconstructions for some special sample alignments



- Thus, to be on the safe side:
  - sample each oscillation period more than twice
- Next: a closer look onto the whole process

Aliasing: Prevention

So must choose:

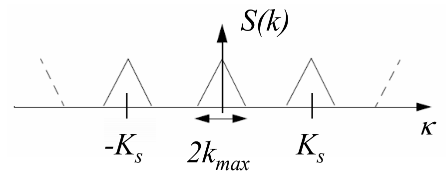
$$K > K_s = 2 \cdot k_{\max}, \quad K_s \text{ is the } \textit{Nyquist rate}$$

In other words:

- the samples only uniquely define the signal if:

$$S(k) = 0 \qquad \forall |k| > k_{\max}$$

$$\frac{1}{\Delta x} > 2k_{\max} = K_s$$



- this assumes that the signal is band-limited ($S(k)$=0 above K_s

Anti-Aliasing

Usually signals are not band-limited

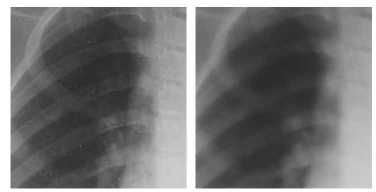
- recall the infinite spectrum of a sharp edge (for example: a bone)

To prevent the inevitable aliasing we must perform anti-aliasing before sampling the signal

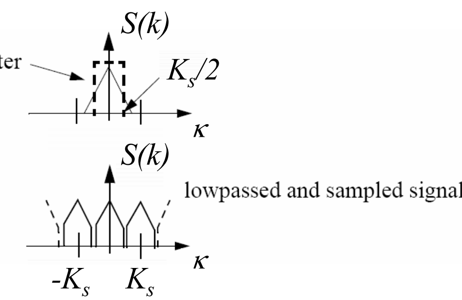
- for example: when digitizing a radiograph of a bone or a chest

Anti-aliasing is done by low-pass filtering (blurring)

- band-limit the signal *prior* to sampling
- we shall see later, how

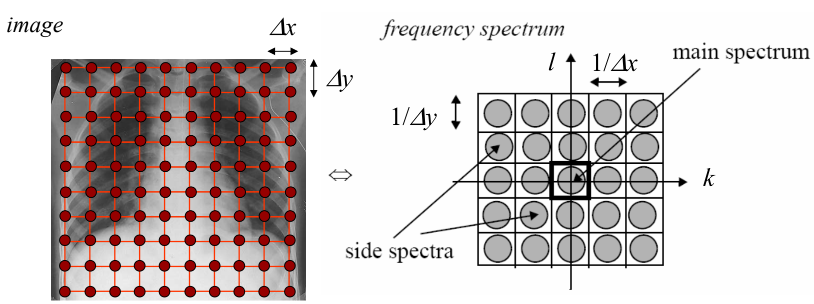


original blurred



Higher Dimensions

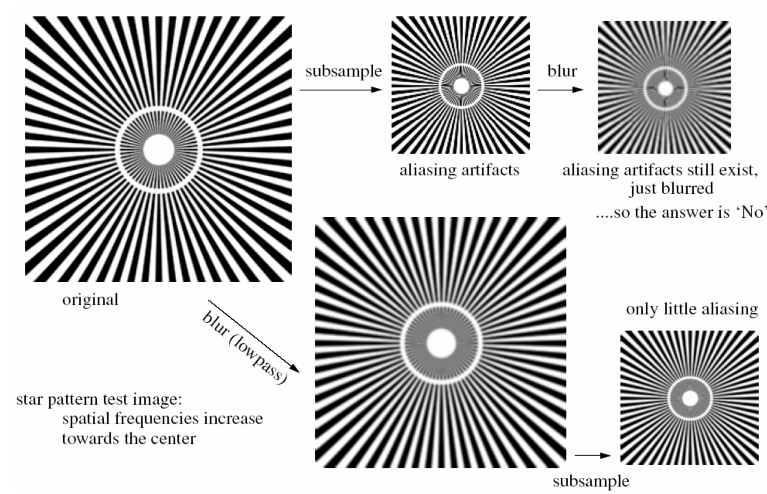
All of these concepts readily extend to higher dimensions



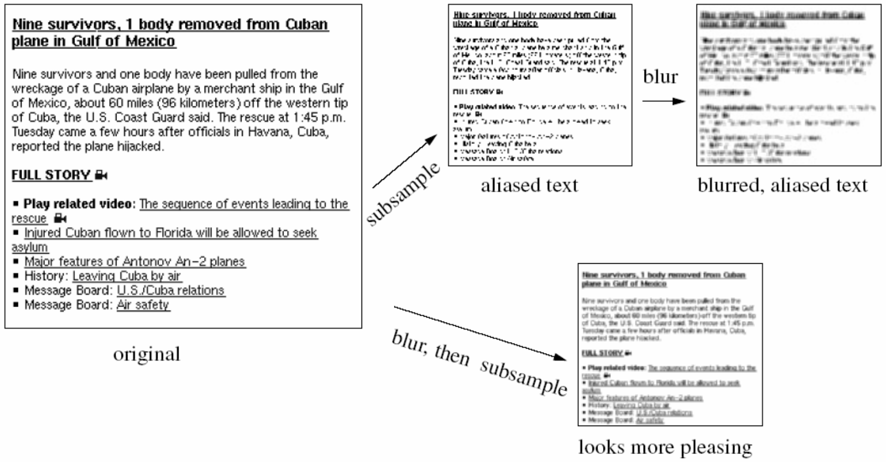
Main spectrum ($S(k,l)$) must fit into the center box to prevent overlap with side-spectra (and aliasing)

$$\frac{1}{\Delta x} > 2 \cdot k_{x\max} \qquad \frac{1}{\Delta y} > 2 \cdot k_{y\max}$$

Anti-Aliasing: Practical Examples (1)



Anti-Aliasing: Practical Examples (2)

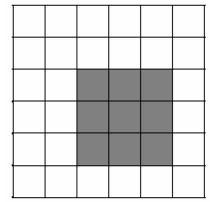


We observe: Anti-aliasing (i.e., blurring, lowpassing) must be applied before sampling

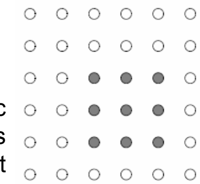
Image Representation

We know that a discrete image is a matrix of pixels

- do keep this in mind, however:

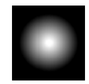


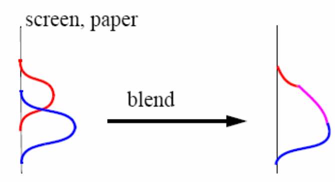
an image is NOT a matrix of solid squares



rather, each pixel is a Dirac impulse, with the pixel's value as its height

So, why do we not see isolated dots on the screen or paper?

- a monitor or printer "splats" the pixels onto the screen or paper.
- each pixels assumes the shape of a Gaussian 
- the Gaussians blend together and form a continuous image

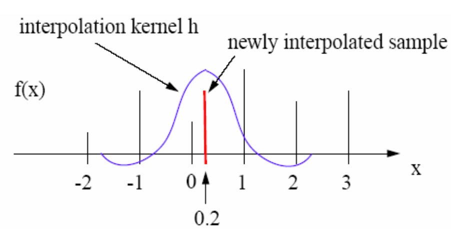


Interpolation

Often we want to estimate the formerly continuous function from the discretized function represented by the matrix of sample points

This is done via *interpolation*

Concept:

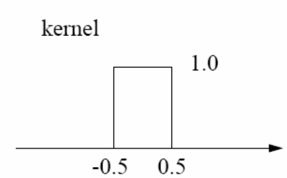


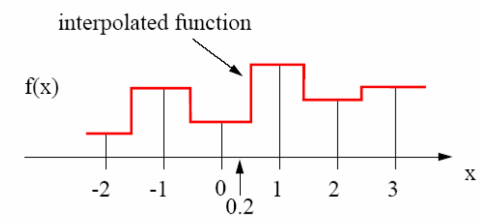
- center the interpolation kernel (filter) h at the sample position and superimpose it onto the grid
- multiply the values of the grid samples with the kernel value at the superimposed position
- add all the products → this gives the value of the newly interpolated sample
- in the shown case:

$$f(0.2) = h(-0.2)\, f(0) + h(-1.2)\, f(-1) + h(0.8)\, f(1) + h(1.8)\, f(2)$$

Interpolation Kernels (1)

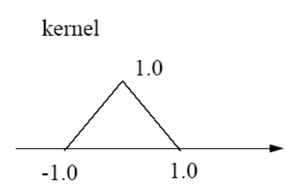
- Nearest Neighbor:

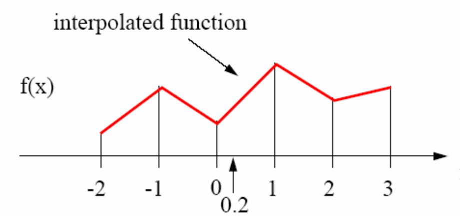




- simply pick the value of the nearest grid point: f(0.2) = f(trunc(0.2+0.5) = f(round(0.2))

- Linear filter:

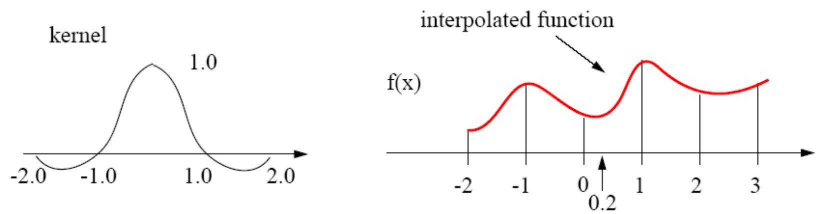




- use a linear combination of the two neighboring grid values: f(0.2) = 0.2·f(1) + 0.8·f(0)

Interpolation Kernels (2)

- Cubic filter:



An additional popular filter is the Gaussian function

Discussion:

- nearest neighbor is fastest to compute (just one add), gives sharp edges, but sometimes jagged lines
- linear interpolation takes 2 mults and 1 add and gives a piecewise smooth function
- cubic filter takes 4 mults and 3 adds, but gives an overall smooth interpolated function
- linear interpolation is most popular in many application

Interpolation in Higher Dimensions

- All interpolation kernels shown here are separable

  $h(x, y) = h(x) \cdot h(y)$ and $h(x, y, z) = h(x) \cdot h(y) \cdot h(z)$

- Linear interpolation

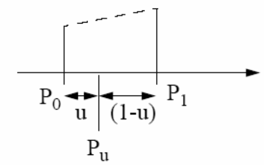
  assume: grid distance = 1.0

  P_u is the location of the sample value

  P_0 and P_1 are neighboring grid points

  then: $u = P_u - P_0$

  $f(x) = f(P_u) = (1 - u) \cdot f(P_0) + u \cdot f(P_1)$

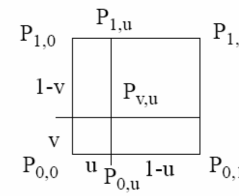


- Bilinear interpolation

  $f(P_{0,u}) = (1 - u) \cdot f(P_{0,0}) + u \cdot f(P_{0,1})$

  $f(P_{1,u}) = (1 - u) \cdot f(P_{1,0}) + u \cdot f(P_{1,1})$

  $f(P_{v,u}) = (1 - v) \cdot f(P_{0,u}) + v \cdot f(P_{1,u})$

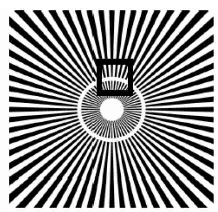


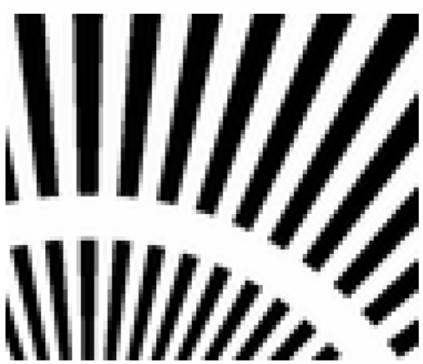
$\rightarrow f(x, y) = f(P_{v,u}) = (1-v)(1-u) f(P_{0,0}) + (1-v) u f(P_{0,1}) + v (1-u) f(P_{1,0}) + v u f(P_{1,1})$

Interpolation Quality

Example:

- resampling of a portion of the star image onto a high resolution grid
- magnification factor ~20





Computation of the Fourier Transform

The analytical form of the Fourier transform (and its laws) is convenient for theoretical, fundamental considerations

- examples: filter design, sampling rates, image resolutions

But in practical applications (for example, low-passing and other filtering) we require a means to compute a discretized signal's Fourier transform:

$$S(m\Delta k_x, n\Delta k_y) = \sum_{q=0}^{N-1} \sum_{p=0}^{M-1} s(p\Delta x, p\Delta y) e^{-2\pi i(\frac{mp}{M} + \frac{nq}{N})}$$

$$s(p\Delta x, q\Delta y) = \sum_{n=0}^{N-1} \sum_{m=0}^{M-1} S(m\Delta k_x, n\Delta k_y) e^{2\pi i(\frac{mp}{M} + \frac{nq}{N})}$$

Assume M=N, then this is an O(N^4) algorithm

- the Fast Fourier Transform (FFT) brings this down to O(N^2log$N$)