INTRODUCTION TO
DIGITAL IMAGE PROCESSING

Lecture #8

Niels Volkmann
Professor

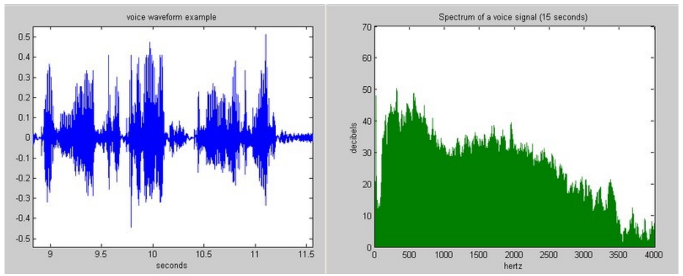
ECE Department
Department of Bioengineering
Quantitative Bioscience Program

Fourier Transform: Why?

- Mathematially easier to analyze effects of transmission medium, noise, etc on simple sine functions, then add to get effect on complex signal

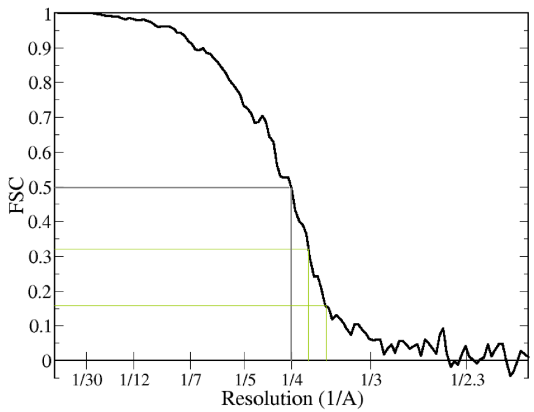
Example: Music

- We think of music in terms of frequencies at different magnitudes.



Other signals

- We can also think of all kinds of other signals the same way



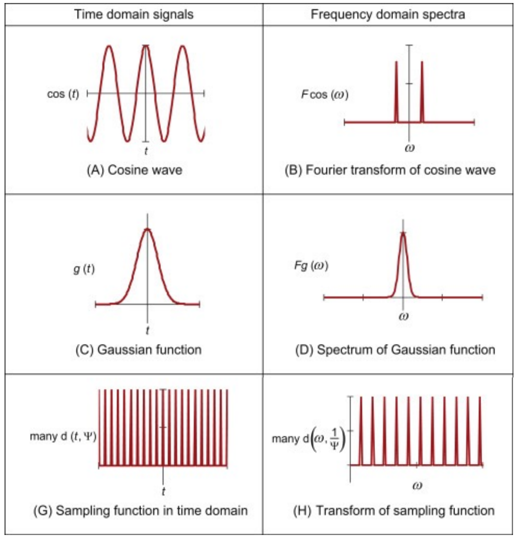
Fourier Series of Periodic Functions

- (Almost) any periodic function *g(x)* with fundamental frequency ω_0 can be described as a sum of sinusoids

$$g(x) = \underbrace{\sum_{k=0}^{\infty}}_{\text{Infinite sum of}} [\underbrace{A_k \cos(k\omega_0 x)}_{\text{Cosines}} + \underbrace{B_k \sin(k\omega_0 x)}_{\text{Sines}}]$$

- This infinite sum is called a **Fourier Series**
- Summed sines and cosines are multiples of the fundamental frequency (harmonics)
- A_k and B_k called **Fourier coefficients**
  - Not known initially but derived from original function *g(x)* during **Fourier analysis**

Fourier Series Examples



Nonperidoc Functions - Fourier Transform

- **Fourier Transform:** Transition of function *g(x)* to its Fourier spectrum *G(ω)*

$$\boxed{\begin{aligned} G(\omega) &= \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{\infty} g(x) \cdot \Big[\cos(\omega x) - \mathrm{i} \cdot \sin(\omega x)\Big] \, \mathrm{d}x \\ &= \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{\infty} g(x) \cdot e^{-\mathrm{i}\omega x} \, \mathrm{d}x. \end{aligned}}$$

- **Inverse Fourier Transform:** Reconstruction of original function *g(x)* from its Fourier spectrum *G(ω)*

$$\boxed{\begin{aligned} g(x) &= \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{\infty} G(\omega) \cdot \Big[\cos(\omega x) + \mathrm{i} \cdot \sin(\omega x)\Big] \, \mathrm{d}\omega \\ &= \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{\infty} G(\omega) \cdot e^{\mathrm{i}\omega x} \, \mathrm{d}\omega. \end{aligned}}$$

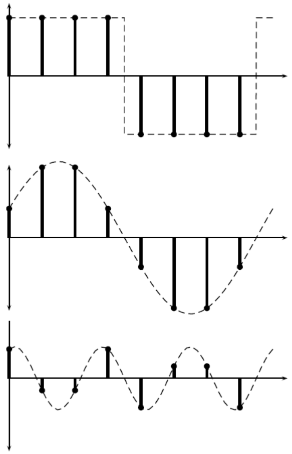
- ***G(ω) + g(ω)*** called **Fourier transform pair**

Discrete Fourier Transform (DFT)

- **Image is a discrete 2D function!!**
- For discrete functions we need only finite number of functions
- For example, consider the discrete sequence

1, 1, 1, 1, -1, -1, -1, -1

- … which is a discrete approximation to a square wave
- Can use DFT to express as sum of 2 sine functions



Definition of 1D DFT

For a sequence of length *N*

$$\mathbf{f} = [f_0, f_1, f_2, \ldots, f_{N-1}]$$

The DFT is

$$\mathbf{F} = [F_0, F_1, F_2, \ldots, F_{N-1}]$$

where

$$F_u = \frac{1}{N} \sum_{x=0}^{N-1} \exp\left[-2\pi i \frac{xu}{N}\right] f_x$$

Compare with complex form of coefficients

$$c_n(x) = \frac{1}{2T}\int_{-T}^{T} f(x)\exp\left(\frac{-in\pi x}{T}\right)dx$$

- Similar to Fourier series expansion
- Instead of integral, we now have a finite sum

Inverse 1D DFT

- Formula for inverse DFT

$$x_u = \sum_{x=0}^{N-1} \exp\left[2\pi i \frac{xu}{N}\right] F_u$$

DFT equation

$$F_u = \frac{1}{N} \sum_{x=0}^{N-1} \exp\left[-2\pi i \frac{xu}{N}\right] f_x$$

- Compared to DFT equation,
  - The inverse has no scaling factor 1/N
  - The sign inside the square bracket has been changed from negative to positive

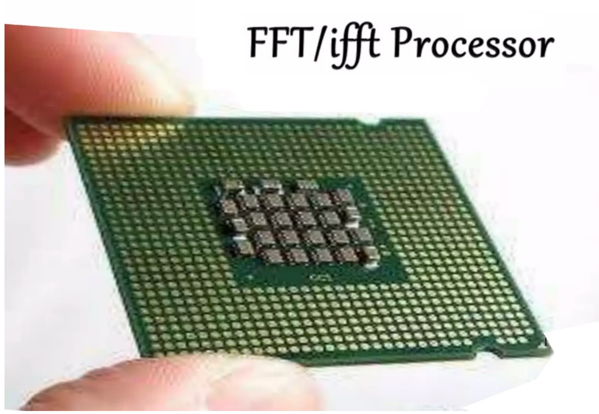
Fast Fourier Transform (FFT)

- Many ways to compute DFT quickly
- **Fast Fourier Transform (FFT)** algorithm is one such way
- One FFT computation method
  - Divides original vector into 2
  - Calculates FFT of each half recursively
  - Merges results

FFT Computation Time Savings

2^n	Direct arithmetic	FFT	Increase in speed
4	16	8	2.0
8	84	24	2.67
16	256	64	4.0
32	1024	160	6.4
64	4096	384	10.67
128	16384	896	18.3
256	65536	2048	32.0
512	262144	4608	56.9
1024	1048576	10240	102.4

FFT Computation Time Savings



2D DFT

- We have seen that a 1D function can be written as a sum of sines and cosines
- Images can be thought of as 2D function f that can be expressed as a sum of **sines and cosines along 2 dimensions**

2D DFT

- For $M \times N$ matrix (image), forward and inverse fourier transforms can be written

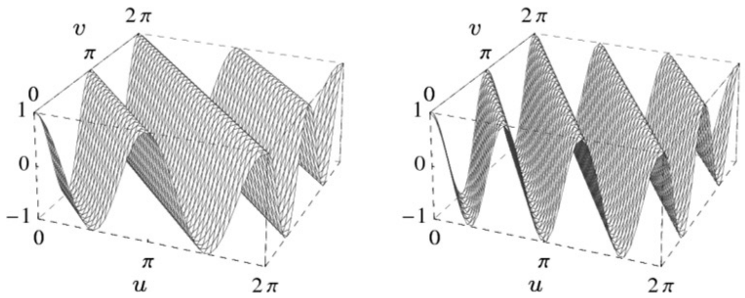
$$F(u,v) = \sum_{x=0}^{M-1}\sum_{y=0}^{N-1} f(x,y)\exp\left[-2\pi i\left(\frac{xu}{M}+\frac{yv}{N}\right)\right].$$

$$f(x,y) = \frac{1}{MN}\sum_{u=0}^{M-1}\sum_{v=0}^{N-1} F(u,v)\exp\left[2\pi i\left(\frac{xu}{M}+\frac{yv}{N}\right)\right].$$

where

- x indices go from $0 \ldots M-1$ (
- y indices go from $0 \ldots N-1$

2D Cosines functions



Properties of 2D Fourier Transform

- All properties of 1D Fourier transform apply + additional properties
- **Similarity:** Forward and inverse transforms are similar except
  1. scale factor $1/MN$ in inverse transform
  2. Negative sign in exponent of forward transform

$$F(u,v) = \sum_{x=0}^{M-1}\sum_{y=0}^{N-1} f(x,y)\exp\left[-2\pi i\left(\frac{xu}{M}+\frac{yv}{N}\right)\right].$$

$$f(x,y) = \frac{1}{MN}\sum_{u=0}^{M-1}\sum_{v=0}^{N-1} F(u,v)\exp\left[2\pi i\left(\frac{xu}{M}+\frac{yv}{N}\right)\right].$$

Properties of 2D Fourier Transform

$$F(u,v) = \sum_{x=0}^{M-1}\sum_{y=0}^{N-1} f(x,y)\exp\left[-2\pi i\left(\frac{xu}{M}+\frac{yv}{N}\right)\right].$$

$$f(x,y) = \frac{1}{MN}\sum_{u=0}^{M-1}\sum_{v=0}^{N-1} F(u,v)\exp\left[2\pi i\left(\frac{xu}{M}+\frac{yv}{N}\right)\right].$$

- **DFT as spatial filter:** These values are just basis functions (are independent of f and F)

$$\exp\left[\pm 2\pi i\left(\frac{xu}{M}+\frac{yv}{N}\right)\right]$$

- Can be computed in advance, put into formulas later
- Implies each value *F(u,v)* obtained by multiplying every value of *f(x,y)* by a fixed value, then adding up all results
- Similar to a filter!
- **2D DFT can be considered a linear spatial filter as big as the image**

Separability

- Notice that Fourier transform "filter elements" can be expressed as products

$$\exp\left[2\pi i\left(\frac{xu}{M}+\frac{yv}{N}\right)\right] = \exp\left[2\pi i\frac{xu}{M}\right]\exp\left[2\pi i\frac{yv}{N}\right]$$

2D DFT **1D DFT (row)** **1D DFT (column)**

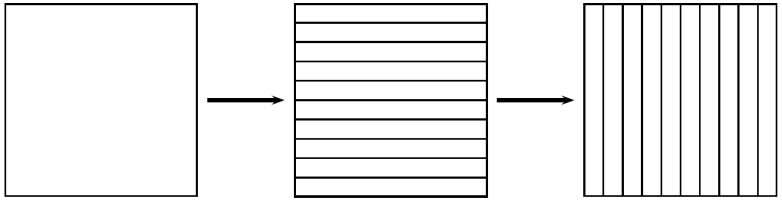
- Formula above can be broken down into simpler formulas for 1D DFT

$$F(u) = \sum_{x=0}^{M-1} f(x)\exp\left[-2\pi i\frac{xu}{M}\right],$$

$$f(x) = \frac{1}{M}\sum_{u=0}^{M-1} F(u)\exp\left[2\pi i\frac{xu}{M}\right]$$

Properties: Separabilty of 2D DFT

- Using their separability property, can use 1D DFTs to calculate rows then columns of 2D Fourier Transform



(a) Original image (b) DFT of each row of (a) (c) DFT of each column of (b)

Properties of 2D DFT

- **Linearity:** DFT of a sum is equal to sum (or multiplication) of the individual DFT's

$$\begin{aligned} \mathcal{F}(f+g) &= \mathcal{F}(f)+\mathcal{F}(g) \\ \mathcal{F}(kf) &= k\mathcal{F}(f) \end{aligned}$$

***k* is a scalar**

- Useful property for dealing with degradations that can be expressed as a sum (e.g. noise)

$$d = f + n$$

  where *f* is original image, *n* is the noise, *d* is degraded, noisy image
- We can find fourier transform as:

$$\mathcal{F}(d) = \mathcal{F}(f) + \mathcal{F}(n)$$

- Noise can be removed/reduced by modifying the **Transform** of *n*

Convolution using DFT

- DFT provides alternate method to do **convolution** of image *M* with spatial filter *S*
  1. Pad *S* to make it same size as *M*, yielding *S'*
  2. Form DFTs of both *M* and *S'*
  3. Multiply *M* and *S'* element by element

     $$\mathcal{F}(M)\cdot\mathcal{F}(S')$$
  4. Take the inverse Fourier transform of the result

     $$\mathcal{F}^{-1}(\mathcal{F}(M)\cdot\mathcal{F}(S')).$$

  Convolution can be expressed as elementwise multiplication in Fourier space

$$\begin{aligned} M * S &= \mathcal{F}^{-1}(\mathcal{F}(M)\cdot\mathcal{F}(S')) \\ \mathcal{F}(M * S) &= \mathcal{F}(M)\cdot\mathcal{F}(S') \end{aligned}$$

Convolution using DFT

- Large speedups if *S* is large
- Example: *M* = 512 x 512, *S* = 32 x 32
- Direct computation:
  - 32^2 = 1024 multiplications for each pixel
  - for entire image = 512 x 512 x 1024 = **268,435,456**

- Using DFT:
  - Each row requires 4608 multiplications
  - Multiplications for rows = 4608 x 512 = 2,359,296 multiplications
  - Repeat for columns, DFT of image = 4,718,592 multiplications
  - We need the same for DFT of filter and for inverse DFT.
  - Also need 512 x 512 multiplications for product of 2 transforms
  - Total multiplications = 4,718,592 x 3 + 262,144 = **14,417,920**

  Speed-up factor: ~20!

Displaying Transforms – Log Transform