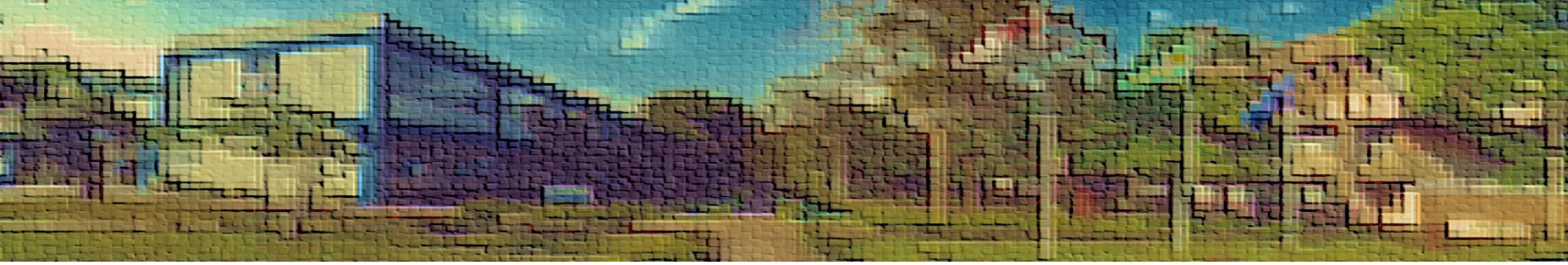


MAC317
Introdução ao Processamento de Sinais Digitais

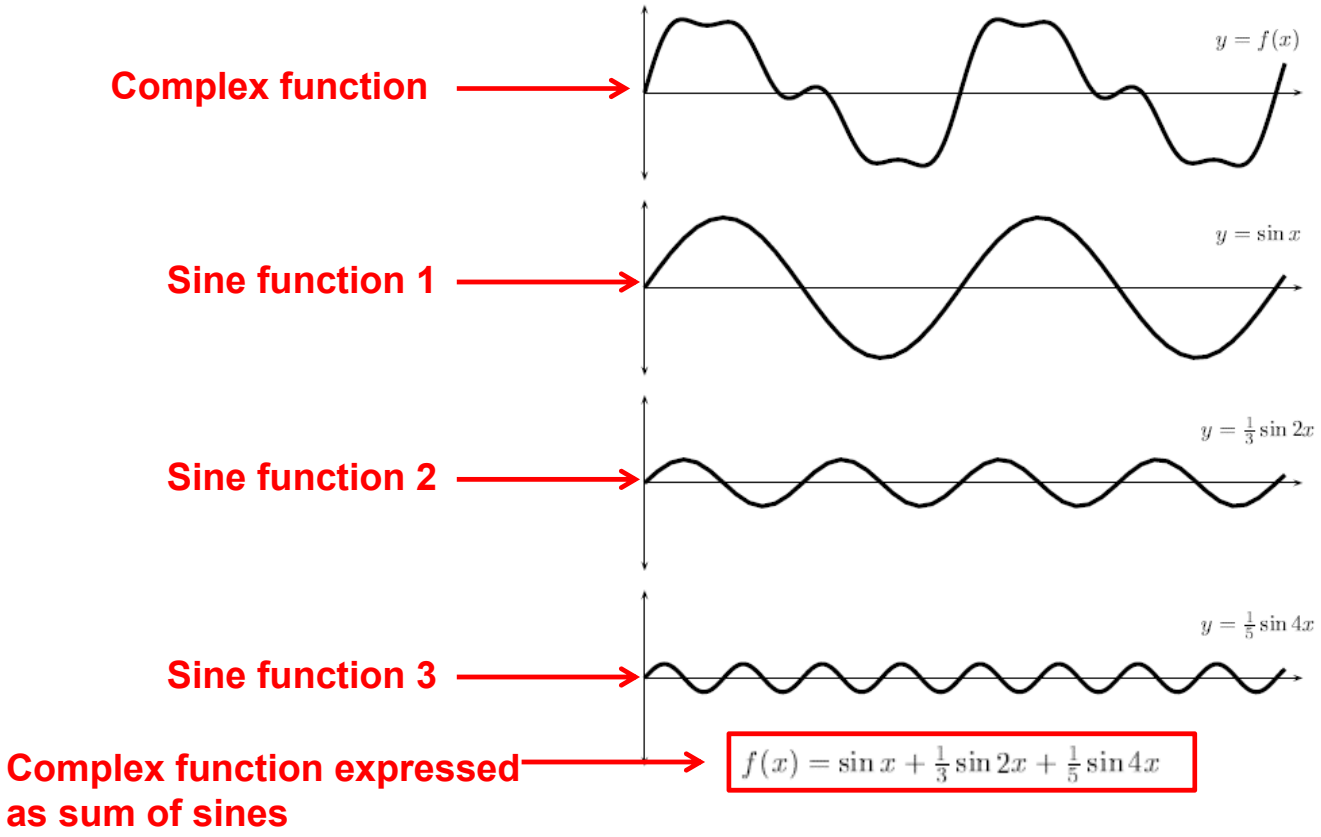
Prof. Marcel P. Jackowski

mjack@ime.usp.br

Aula #8: Transformada discreta de Fourier (Parte 3)

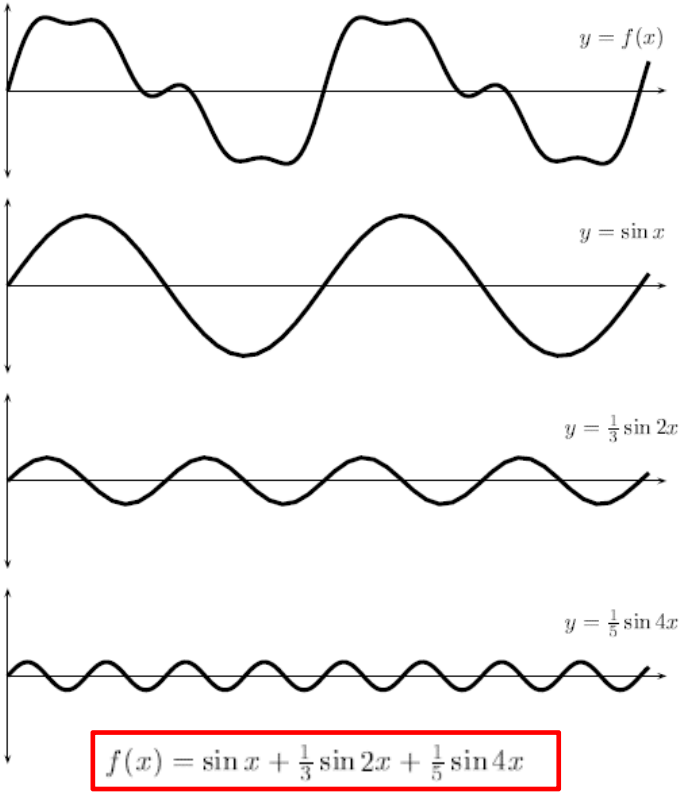
Fourier Transform

- **Main idea:** Any periodic function can be decomposed into a summation of sines and cosines

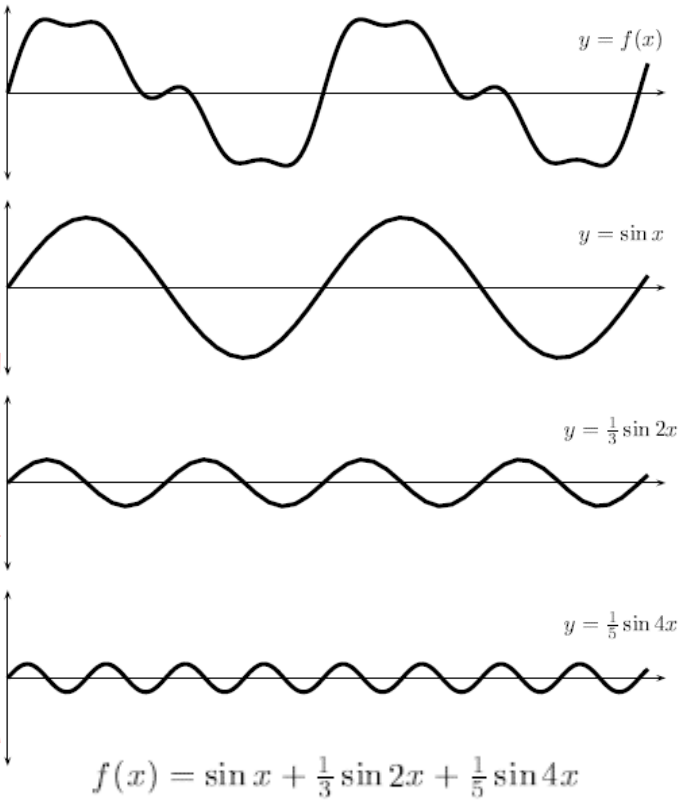


Fourier Transform: Why?

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



Fourier Transform: Some Observations



Observation 2: Frequencies of sines are multiples of each other (called harmonics)

Frequency = 1x

Frequency = 2x

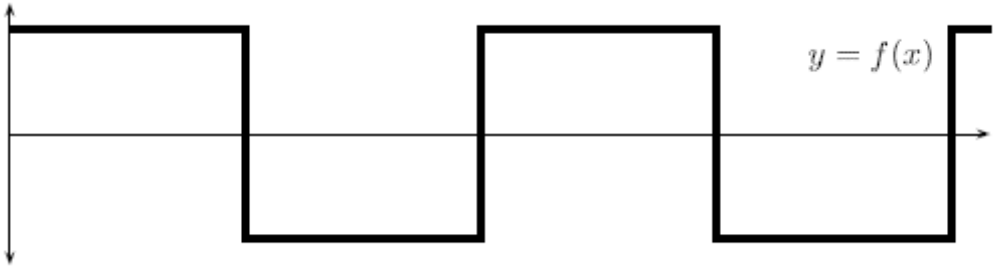
Frequency = 4x

Observation 1: The sines have different frequencies (not same)

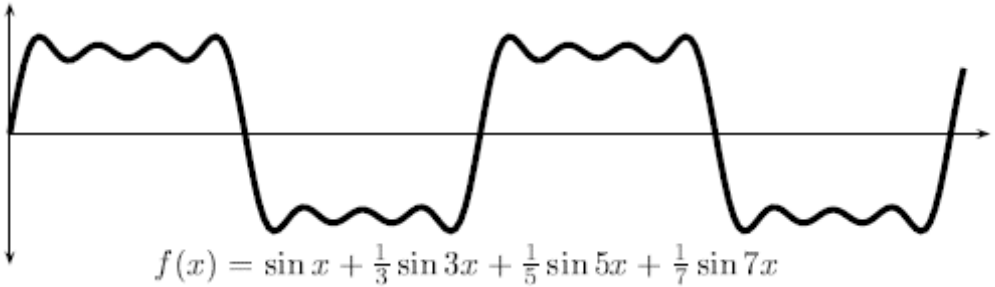
Observation 3: Different amounts of different sines added together (e.g. 1/3, 1/5, etc)

Fourier Transform: Another Example

Square wave



Approximation
Using sines



$$f(x) = \sin x + \tfrac{1}{3}\sin 3x + \tfrac{1}{5}\sin 5x + \tfrac{1}{7}\sin 7x$$

Observation 4: The sine terms go to infinity. The more sines we add, the closer the approximation of the original.

Fourier Series Expansion

- If *f(x)* is **periodic function** of period *2T*
- **Fourier series expansion**

$$f(x) = a_0 + \sum_{n=1}^{\infty}\left(a_n \cos\frac{n\pi x}{T} + b_n \sin\frac{n\pi x}{T}\right)$$

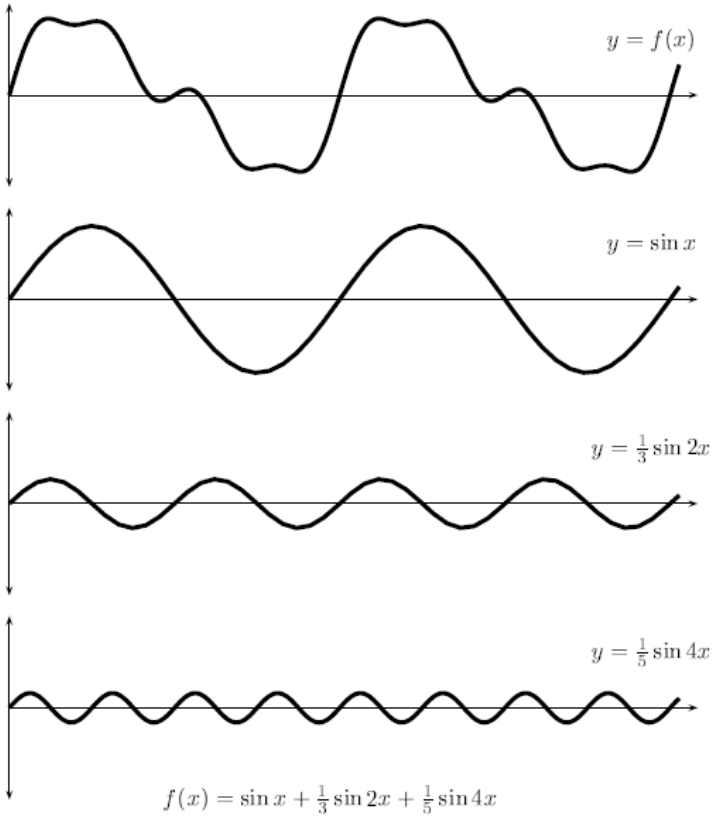
Where

$$a_0 = \frac{1}{T}\int_{-T}^{T} f(x)dx$$

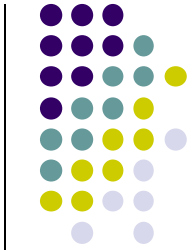
$$a_n = \frac{1}{T}\int_{-T}^{T} f(x)\cos\frac{n\pi x}{T}dx,\ n = 1,2,3,\ldots$$

$$b_n = \frac{1}{T}\int_{-T}^{T} f(x)\sin\frac{n\pi x}{T}dx,\ n = 1,2,3,\ldots$$

- a_n and b_n called **Fourier coefficients**



Fourier Series of Periodic Functions



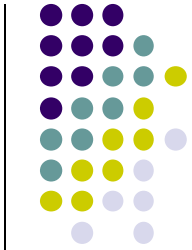
- (Almost) any periodic function *g(x)* with fundamental frequency ω_0 can be described as a sum of sinusoids

$$g(x) = \sum_{k=0}^{\infty} [A_k \cos(k\omega_0 x) + B_k \sin(k\omega_0 x)]$$

Infinite sum of — Cosines — 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 Integral



- For non-periodic functions we can get similar results by letting period T → ∞ similar ideas yield **Fourier Integral**

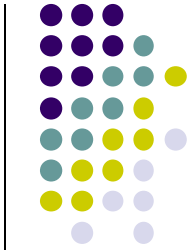
$$g(x) = \int_0^\infty A_\omega \cos(\omega x) + B_\omega \sin(\omega x)\, \mathrm{d}\omega$$

where coefficients can be found as

$$A_\omega = A(\omega) = \frac{1}{\pi} \int_{-\infty}^{\infty} g(x) \cdot \cos(\omega x)\, \mathrm{d}x$$

$$B_\omega = B(\omega) = \frac{1}{\pi} \int_{-\infty}^{\infty} g(x) \cdot \sin(\omega x)\, \mathrm{d}x$$

Definition of 1D DFT



- Suppose

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

is a sequence of length *N*

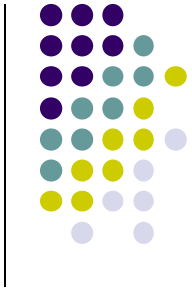
$$\mathbf{F} = [F_0, F_1, F_2, \ldots, F_{N-1}]$$

where

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

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

Inverse 1D DFT



- Formula for inverse

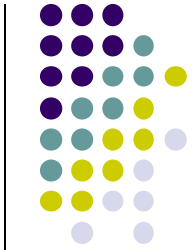
$$x_u = \sum_{x=0}^{N-1} \exp\left[2\pi i \frac{xu}{N}\right] F_u$$

DFT equation

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

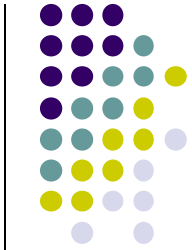
- Compared to DFT equation,
  - The inverse has no scaling factor 1/N
  - The sign inside the square bracket has been changed

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

2D DFT



- Thus if the matrix *F* is the Fourier Transform of *f* we can write

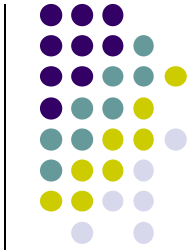
$$F = \mathcal{F}(f)$$

- The original matrix *f* is the Inverse Fourier Transform of *F*

$$f = \mathcal{F}^{-1}(F)$$

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

2D Fourier Transform



- For *M x N* matrix, forward and inverse fourier transforms can be written

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

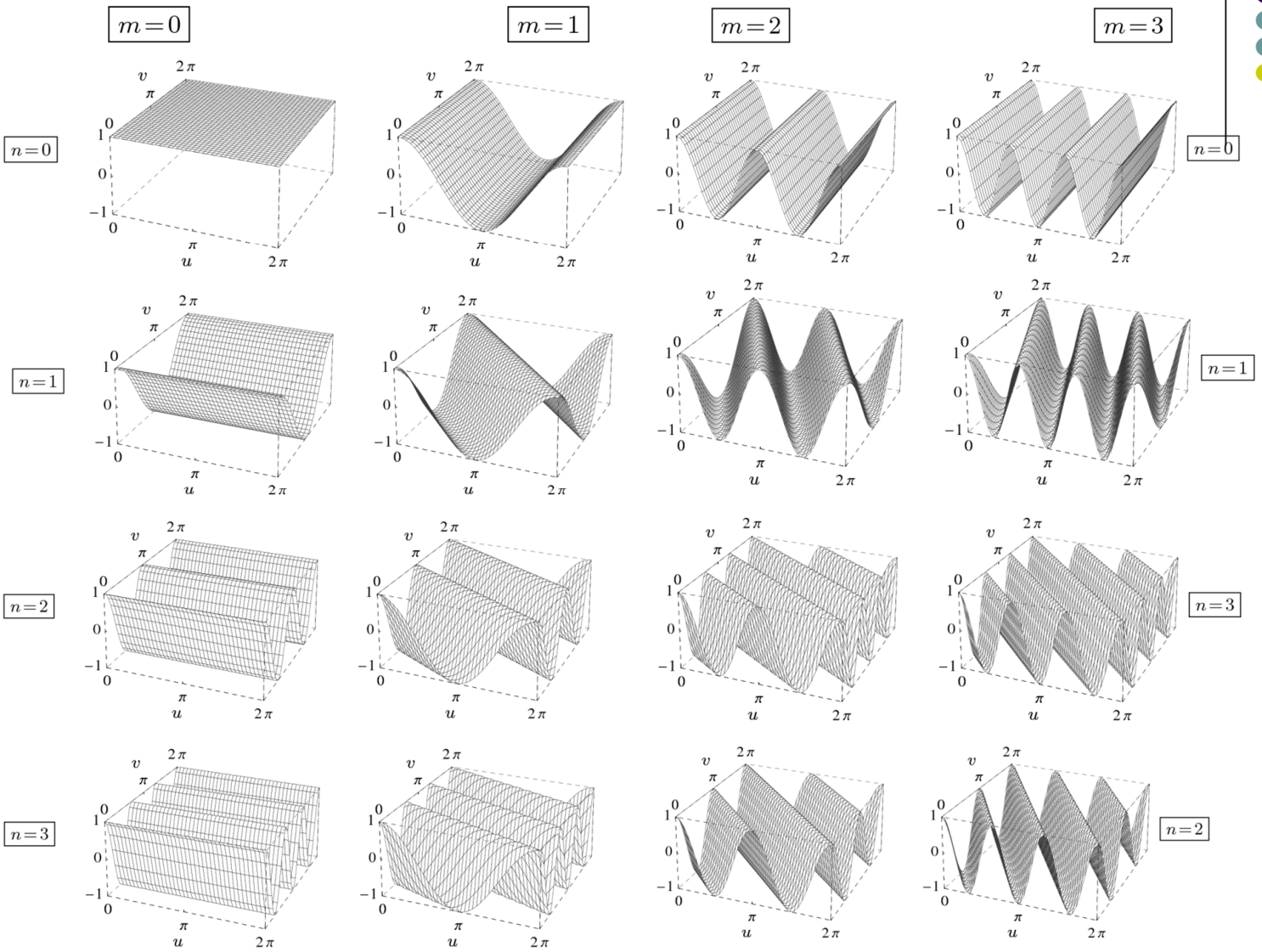
$$f(x,y) = \frac{1}{MN}\sum_{u=0}^{M-1}\sum_{v=0}^{N-1} F(u,v)\exp\left[2\pi i\left(\frac{xu}{M}+\frac{yv}{N}\right)\right].$$

where

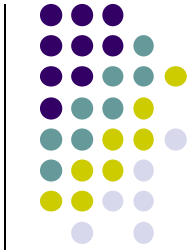
- *x* indices go from *0… M – 1 (x* cycles over distance *M*)
- *y* indices go from *0… N – 1 (*y cycles over distance *N*)

2D Cosines functions

Orientation depend on *m* and *n*



Properties of 2D Fourier Transform



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

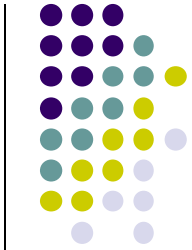
$$f(x,y) = \frac{1}{MN}\sum_{u=0}^{M-1}\sum_{v=0}^{N-1} F(u,v)\exp\left[2\pi i\left(\frac{xu}{M}+\frac{yv}{N}\right)\right].$$

- **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!)
  - **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)

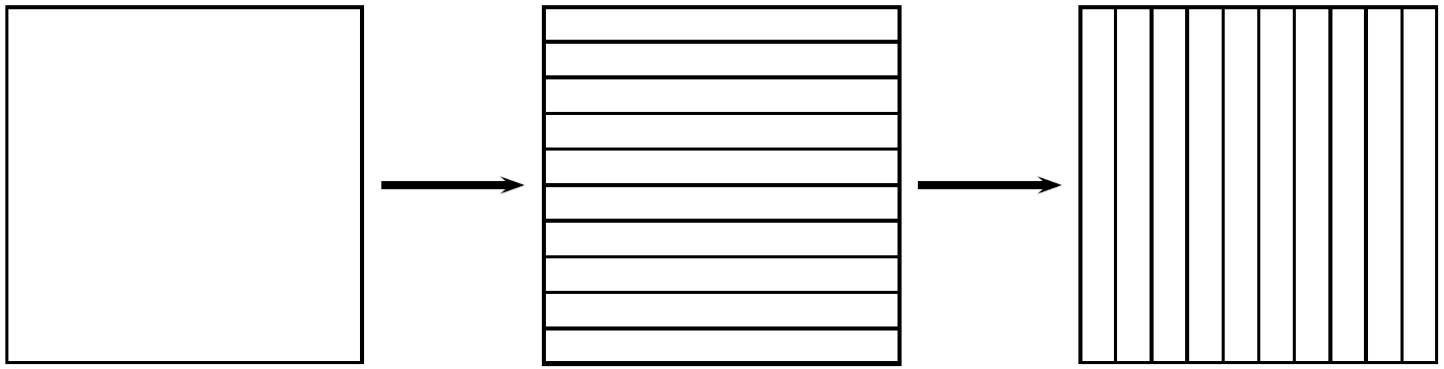
- Formula above can be broken down into simpler formulas for 1D DFT

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

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

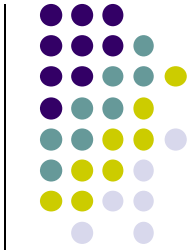
Properties: Separabilty of 2D DFT

- 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)

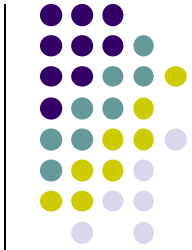
Implementation of 2D DFT



- Can use separability to implement 2D DFT as sequence of 1D DFTs on rows and columns

1: SEPARABLE 2D-DFT (g) $\triangleright$ $g(u,v) \in \mathbb{C}$, $0 \leq u < M$, $0 \leq v < N$
2: **for** $v \leftarrow 0 \ldots N{-}1$ **do**
3: Let $g(\cdot, v)$ be the vth row vector of g:
Replace $g(\cdot, v)$ by $\mathsf{DFT}\big(g(\cdot, v)\big)$
4: **for** $u \leftarrow 0 \ldots M{-}1$ **do**
5: Let $g(u, \cdot)$ be the uth column vector of g:
Replace $g(u, \cdot)$ by $\mathsf{DFT}\big(g(u, \cdot)\big)$
Remark: $g(u,v) \equiv G(u,v) \in \mathbb{C}$ now contains the discrete 2D spectrum.
6: **return** g

Properties of 2D DFT



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

$$\mathcal{F}(f+g) = \mathcal{F}(f) + \mathcal{F}(g)$$
$$\mathcal{F}(kf) = k\mathcal{F}(f)$$ ***k* is a scalar**

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

$$d = f + n$$

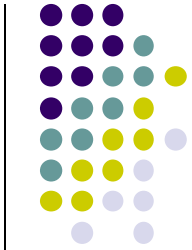
Where *f* is original image, *n* is the noise, *d* is degraded image

- We can find fourier transform as:

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

- Noise can be removed/reduced by modifying 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

$$\mathcal{F}(M) \cdot \mathcal{F}(S')$$

1. Take inverse transform of result

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

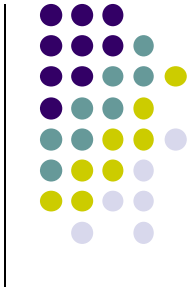
- Essentially

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

Or equivalently the convolution *M * S*

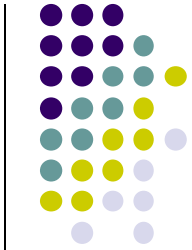
$$\mathcal{F}(M * S) = \mathcal{F}(M) \cdot \mathcal{F}(S')$$

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
  - Total multiplications for entire image = 512 x 512 x 1024 = **268,435,456** multiplications
- Using DFT:
  - Each row requires 4608 multiplications
  - Multiplications for rows = 4608 x 512 = 2,359,296 multiplications
  - Repeat for columns, DFT of image = 4718592 multiplications
  - Need same for DFT of filter and for inverse DFT.
  - Also need 512 x 512 multiplications for product of 2 transforms
  - Total multiplications = 4718592 x 3 + 262144 = **14,417,920**

DC Component



- Recall that:

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

- The value *F(0,0)* of the DFT is called the **dc coefficient**
- If we put $u = v = 0$, then

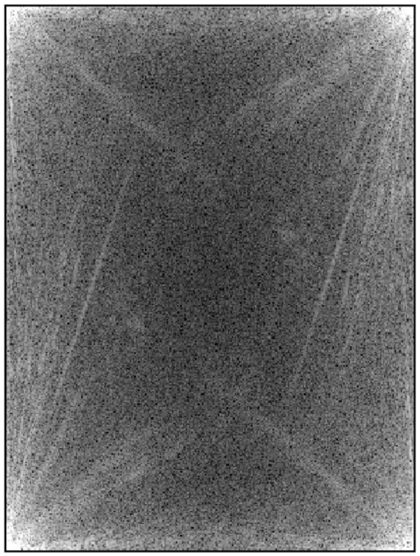
$$F(0,0) = \sum_{x=0}^{M-1}\sum_{y=0}^{N-1} f(x,y)\exp(0) = \sum_{x=0}^{M-1}\sum_{y=0}^{N-1} f(x,y)$$

- Essentially *F(0,0)* is the sum of all terms in the original matrix

Centering DFT Spectrum

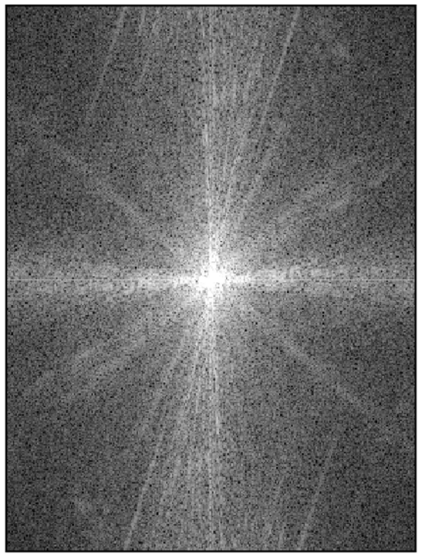
(a)

Original Image



(b)

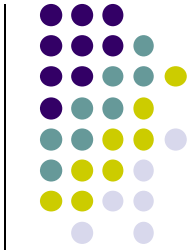
Non-centered spectrum



(c)

Centered spectrum

Displaying Transforms

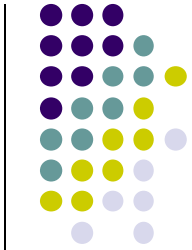


- As elements are complex numbers, we can view magnitudes $|F(u,v)|$ directly
- Displaying magnitudes of Fourier transforms called **spectrum** of the transform

- **Problem:** DC component much larger than other values
  - Displays white dot in middle surrounded by black
- So stretch transform values by displaying log of transform

$$\log(1 + |F(u,v)|)$$

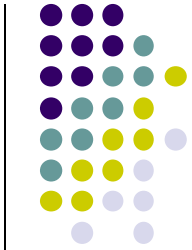
Examples of DFTs



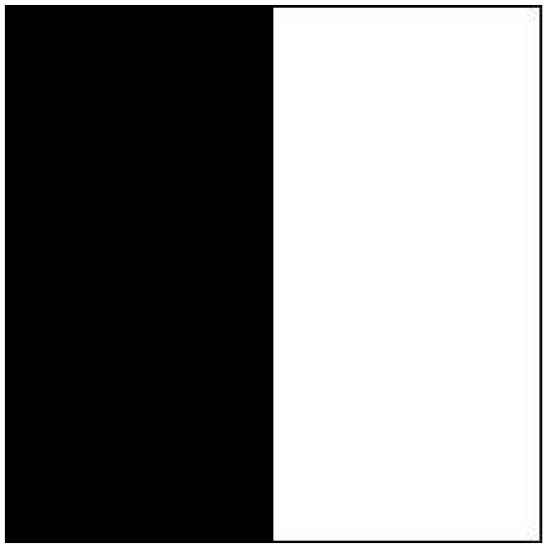
- Suppose we have as input a constant image of all 1's, $f(x,y) = 1$
- The DFT yields just a DC component, 0 everywhere else
- In this example, DC component is sum of all elements = 64

64	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0

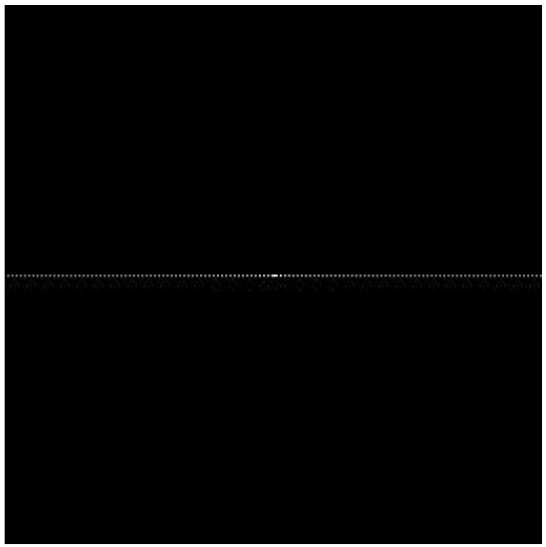
DFT of Image



- Consider DFT of image with single edge
- For display, DC component shifted to center
- Log of magnitudes of Fourier Transform displayed

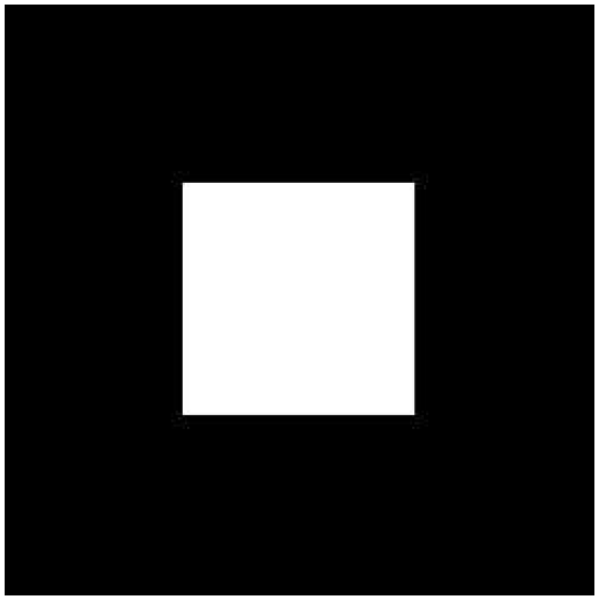


Image

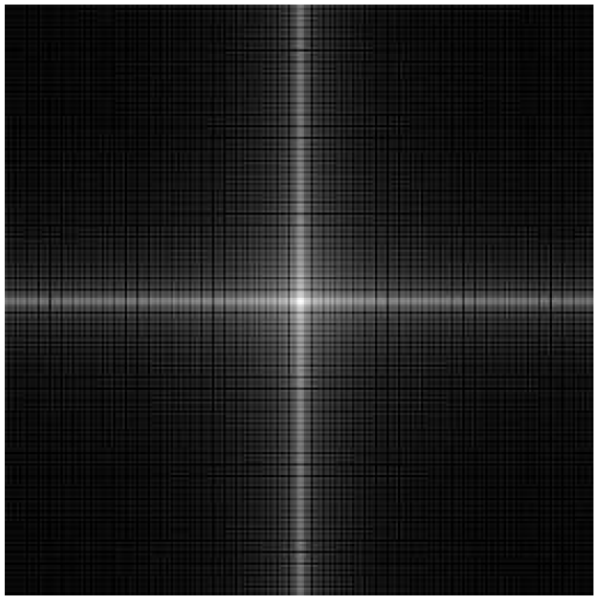


DFT

DFT Example: A Box

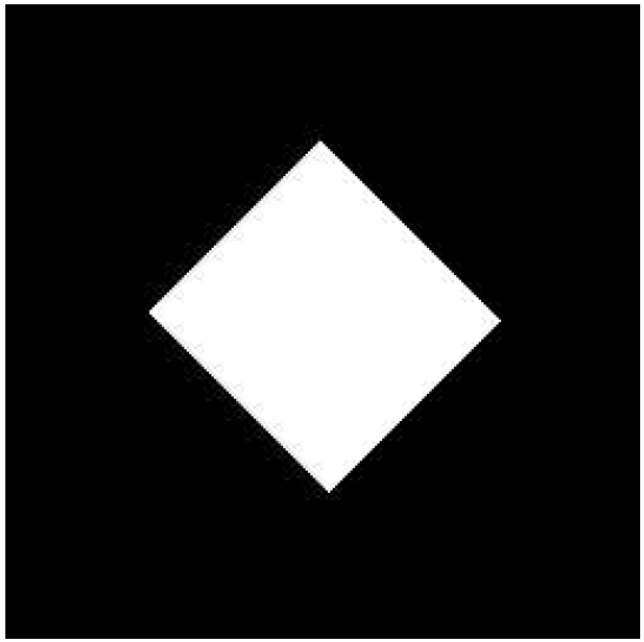


Box

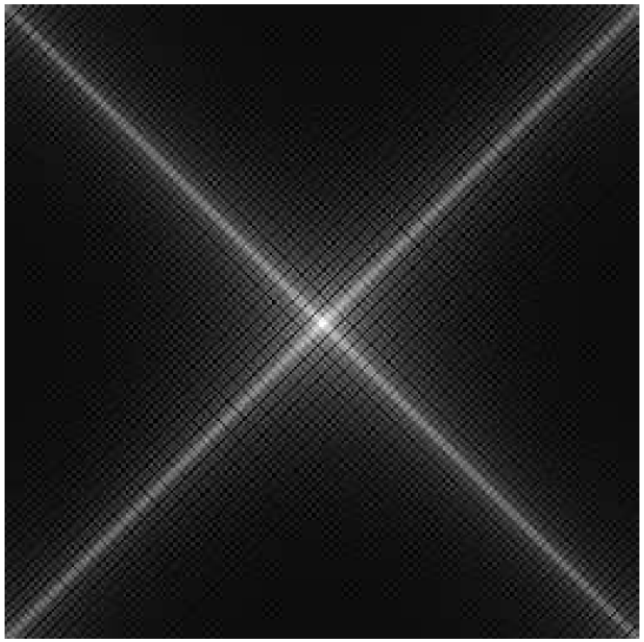


DFT

DFT Example: Rotated Box

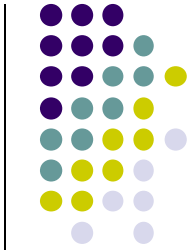


Rotated Box

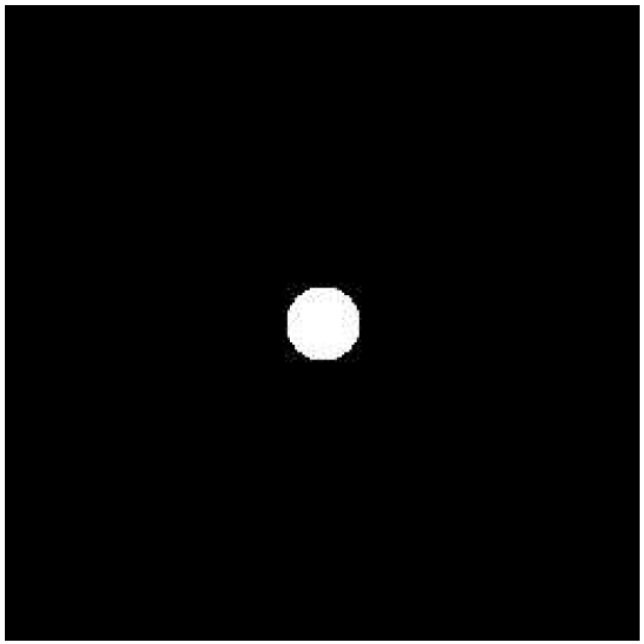


DFT

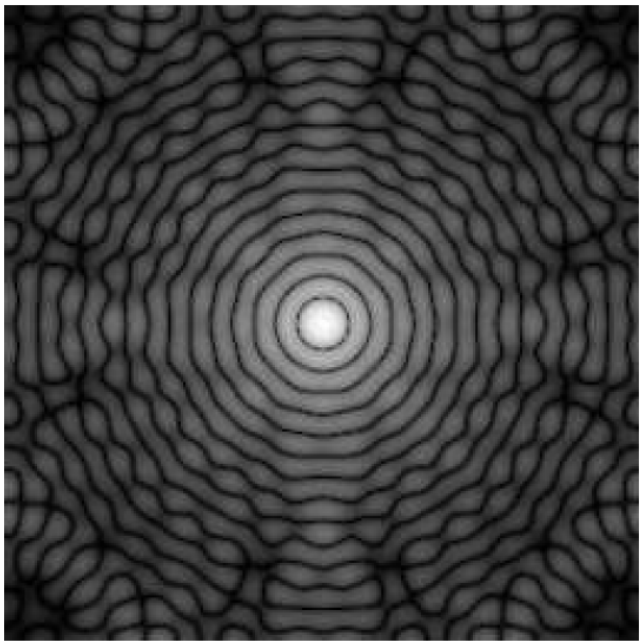
DFT Example: A Circle



- **Note:** Ringing caused by sharp cutoff of circle
- Ringing does not occur if circle cutoff is gentle



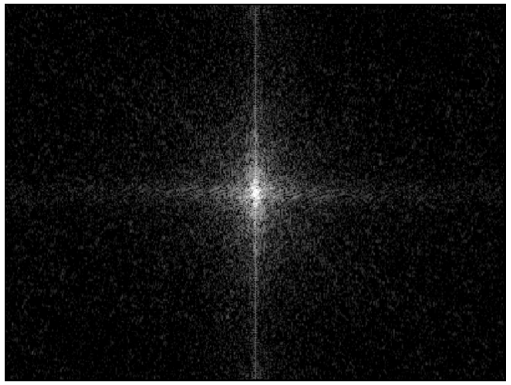
Circle



DFT

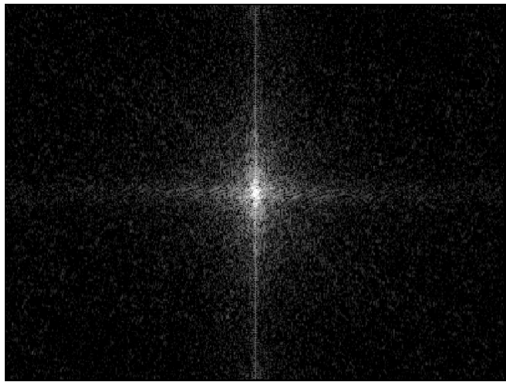
DFT Computation: Repeated Image

- DFT computation assumes image repeated horizontally and vertically
- Large intensity transition at edges = vertical and horizontal line in middle of spectrum after shifting
- Effects of sharp transitions affects many pixels



Windowing

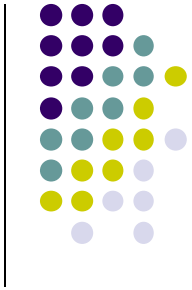
- Can multiply image by windowing function *w(u,v)* before DFT to reduce sharp transitions between ends of repeated images
- Ideally, causes image to drop off towards ends, reducing transitions



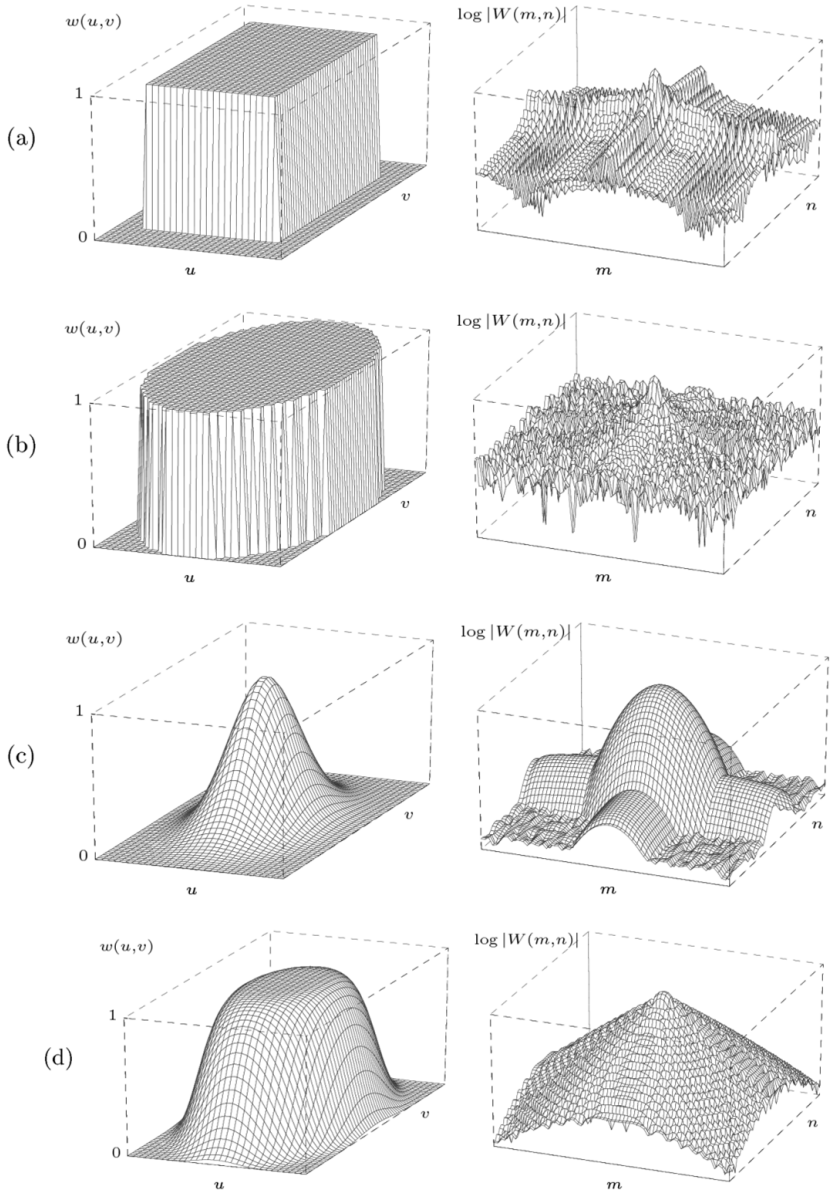
Definitions:

$$r_u = \frac{u - M/2}{M/2} = \frac{2u}{M} - 1 \qquad r_v = \frac{v - N/2}{N/2} = \frac{2v}{N} - 1 \qquad r_{u,v} = \sqrt{r_u^2 + r_v^2}$$

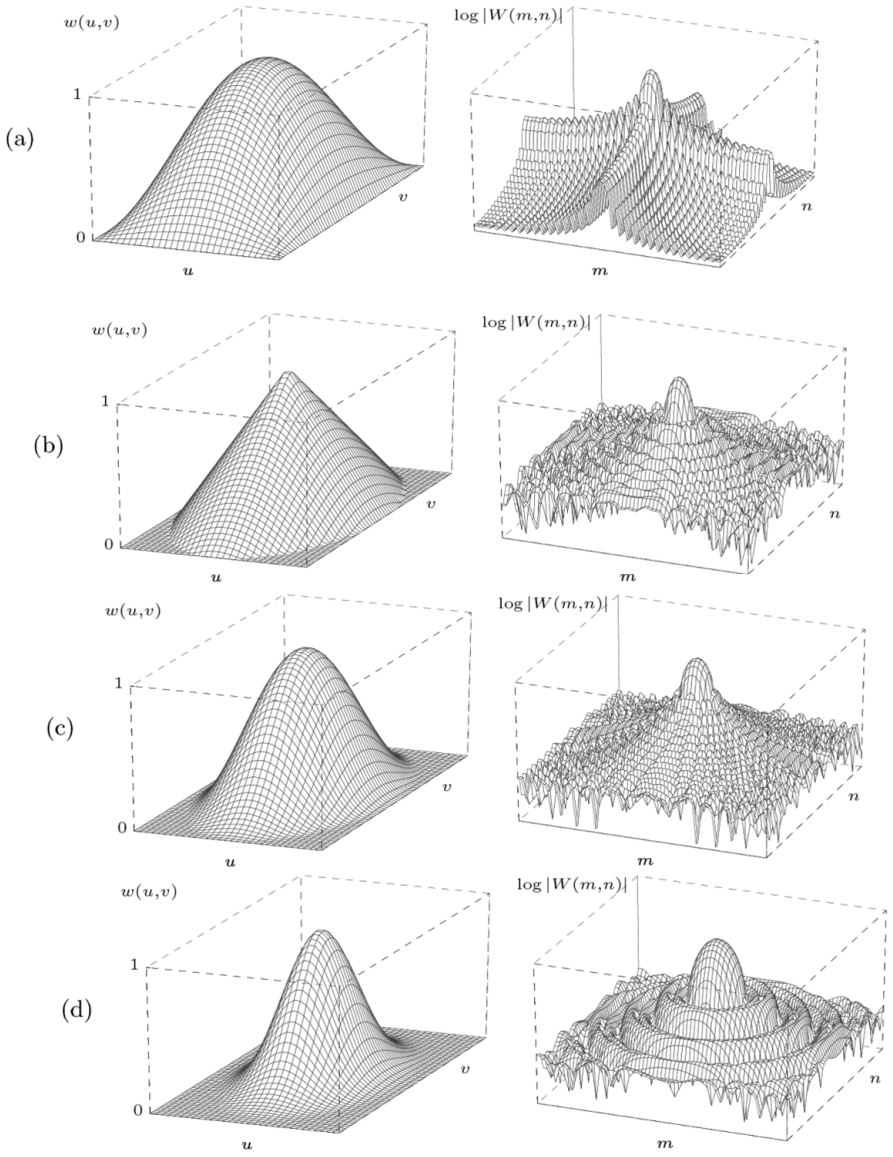
Elliptical window:	$w(u,v) = \begin{cases} 1 & \text{for } 0 \le r_{u,v} \le 1 \\ 0 & \text{otherwise} \end{cases}$
Gaussian window:	$w(u,v) = e^{\left(\frac{-r_{u,v}^2}{2\sigma^2}\right)}, \quad \sigma = 0.3 \ldots 0.4$
Super-Gaussian window:	$w(u,v) = e^{\left(\frac{-r_{u,v}^n}{\kappa}\right)}, \quad n = 6,\ \kappa = 0.3 \ldots 0.4$
Cosine^2 window:	$w(u,v) = \begin{cases} \cos\left(\frac{\pi}{2} r_u\right) \cdot \cos\left(\frac{\pi}{2} r_v\right) & \text{for } 0 \le r_u, r_v \le 1 \\ 0 & \text{otherwise} \end{cases}$
Bartlett window:	$w(u,v) = \begin{cases} 1 - r_{u,v} & \text{for } 0 \le r_{u,v} \le 1 \\ 0 & \text{otherwise} \end{cases}$
Hanning window:	$w(u,v) = \begin{cases} 0.5 \cdot \cos(\pi r_{u,v} + 1) & \text{for } 0 \le r_{u,v} \le 1 \\ 0 & \text{otherwise} \end{cases}$
Parzen window:	$w(u,v) = \begin{cases} 1 - 6r_{u,v}^2 + 6r_{u,v}^3 & \text{for } 0 \le r_{u,v} < 0.5 \\ 2 \cdot (1 - r_{u,v})^3 & \text{for } 0.5 \le r_{u,v} < 1 \\ 0 & \text{otherwise} \end{cases}$



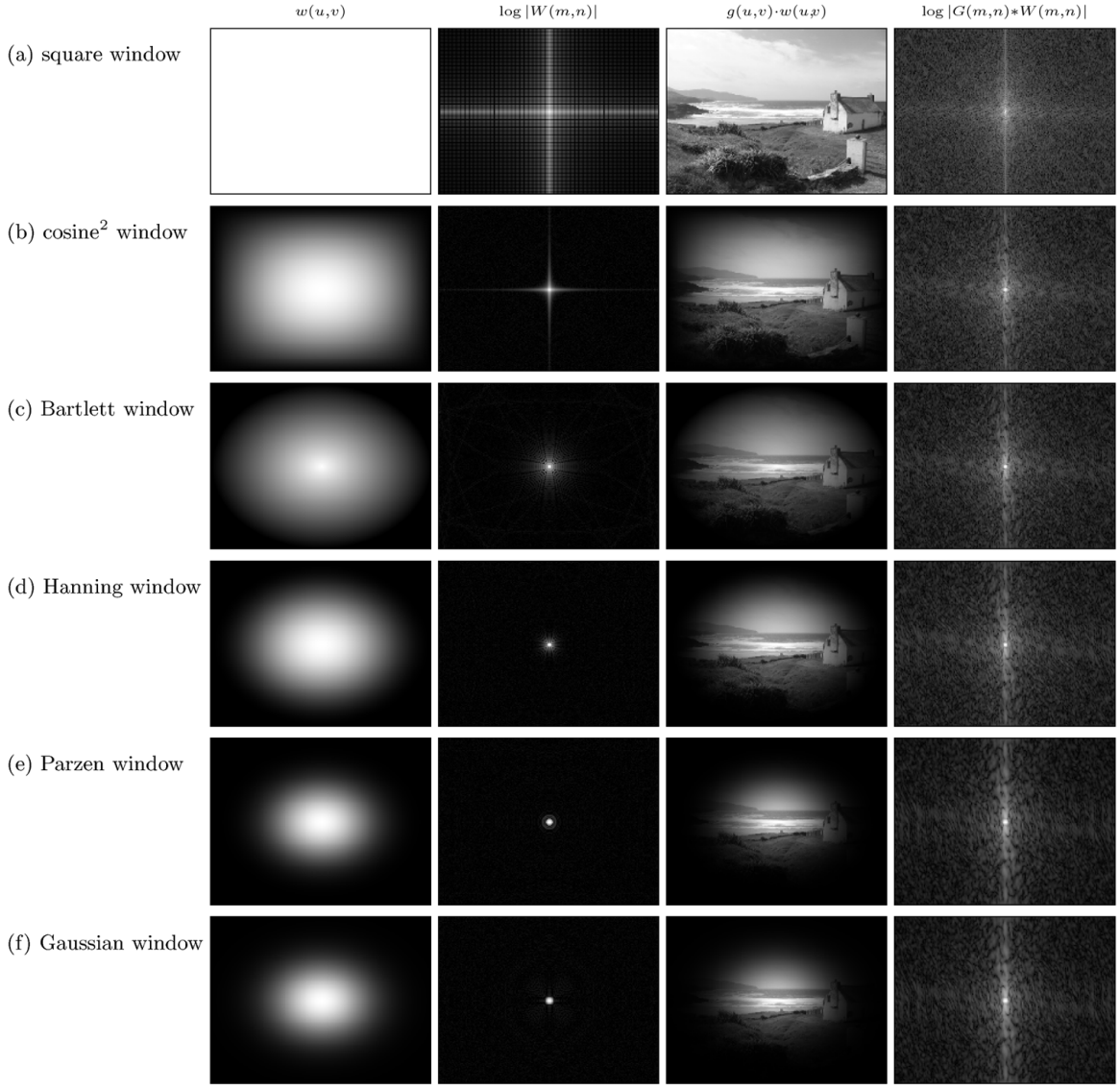
Some Proposed Windowing Functions



Some Proposed Windowing Functions



Some Proposed Windowing Functions



Application of Windowing Functions

Em sala

- Conversa Prof. Choukri
- Aula prática na próxima quinta-feira com imagens usando FFT em 2D