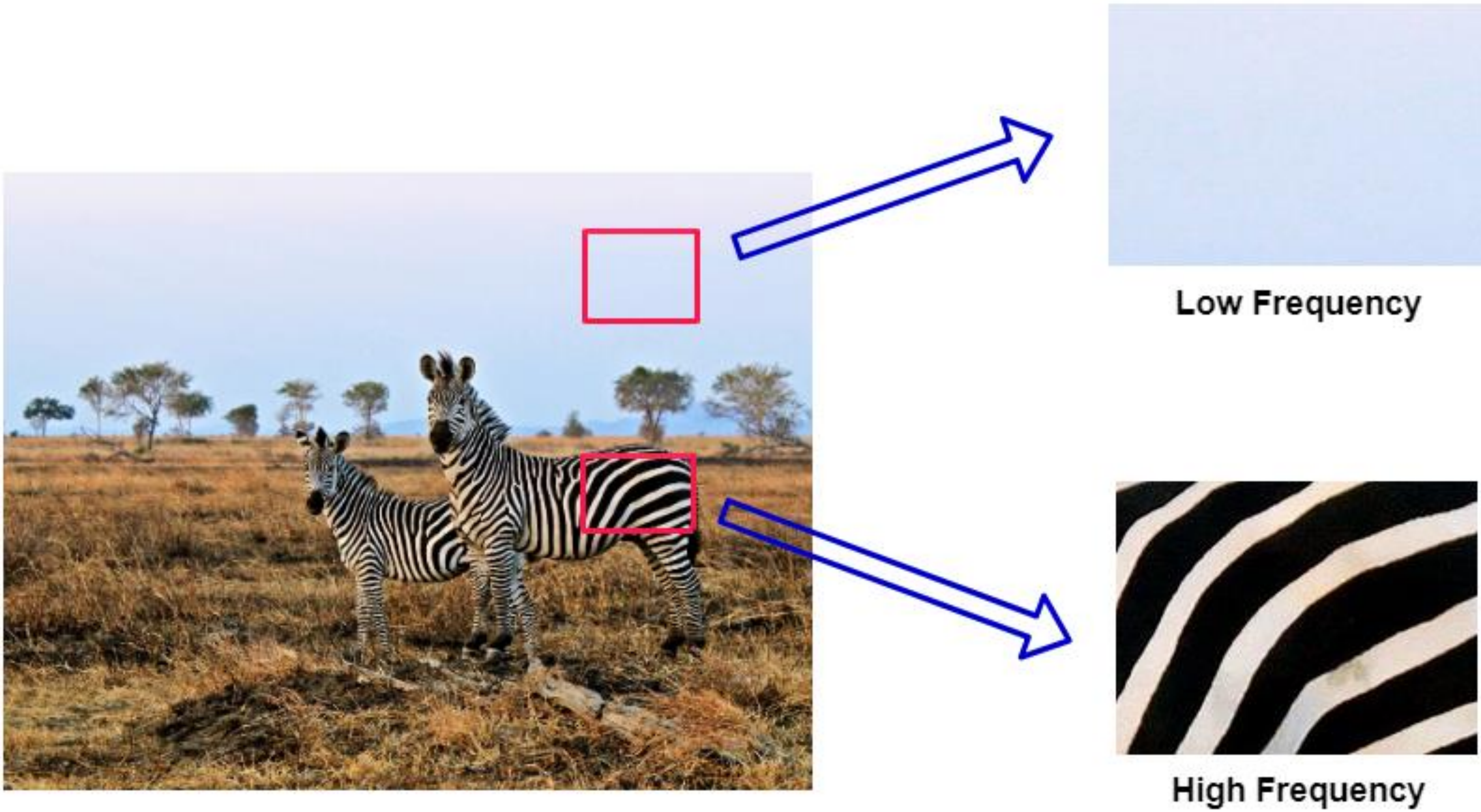
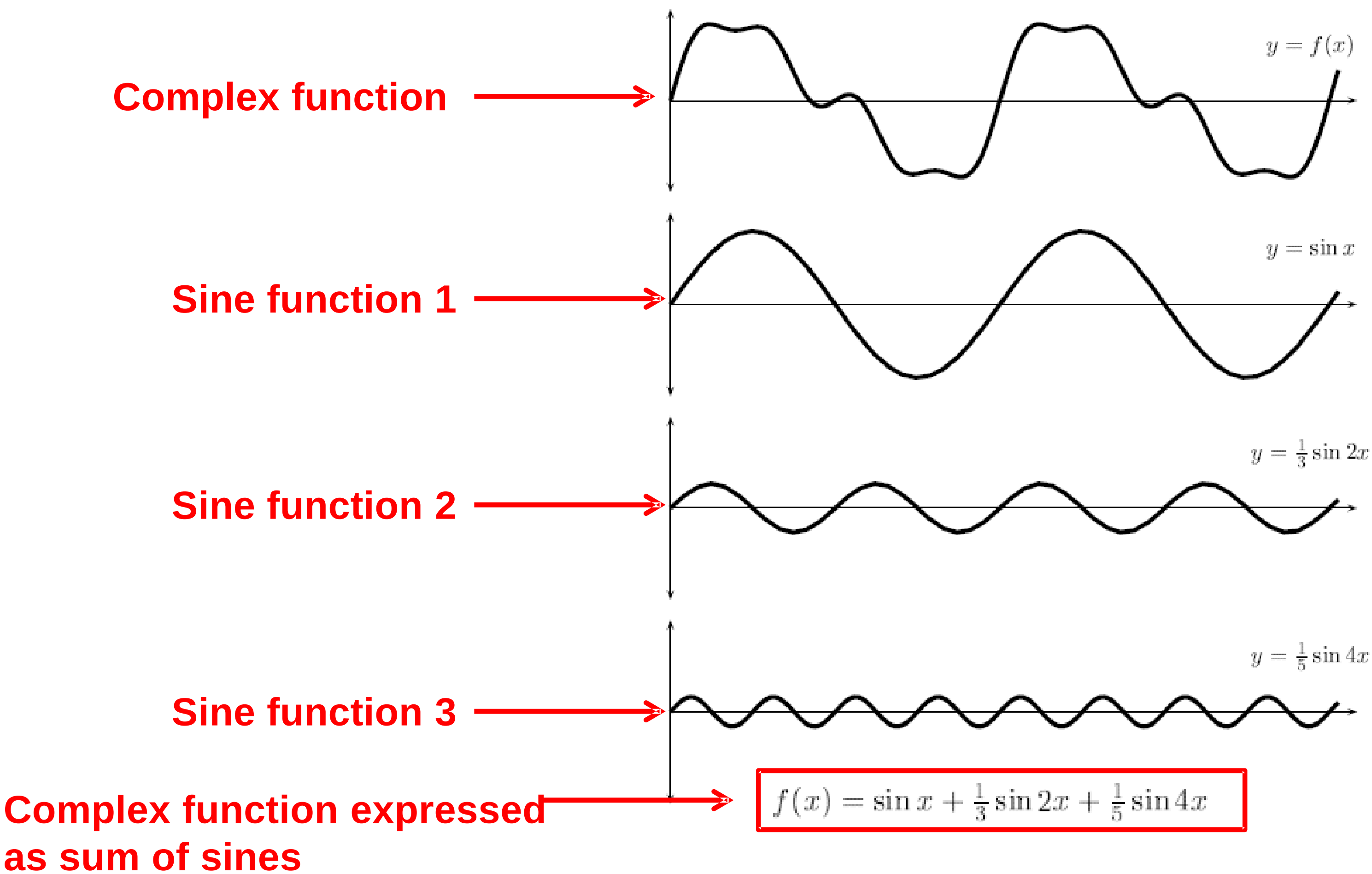
Discrete Fourier Transform (DFT)

- DFT as Discrete Fourier Transform is used as a transform from pixel-domain into frequency-domain.
- Practically, the most frequent pixels will be put in one corner and the least frequent pixels will be in the opposing corner.
- DFT has real and imaginary components
- DFT is used to decompose an image into its sine and cosine components.


Low Frequency
High Frequency

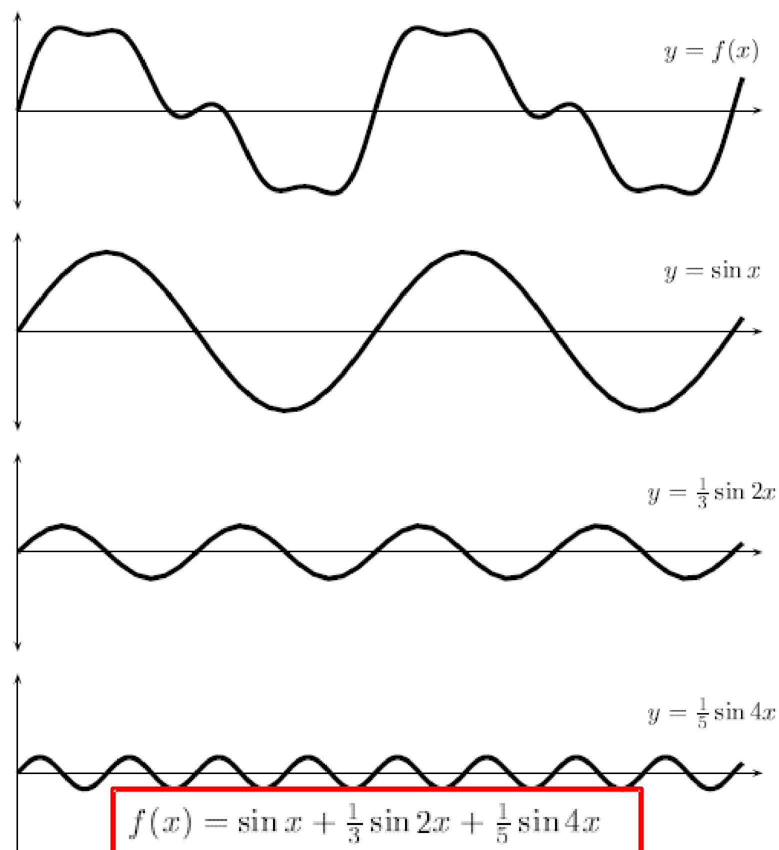
Fourier Transform

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



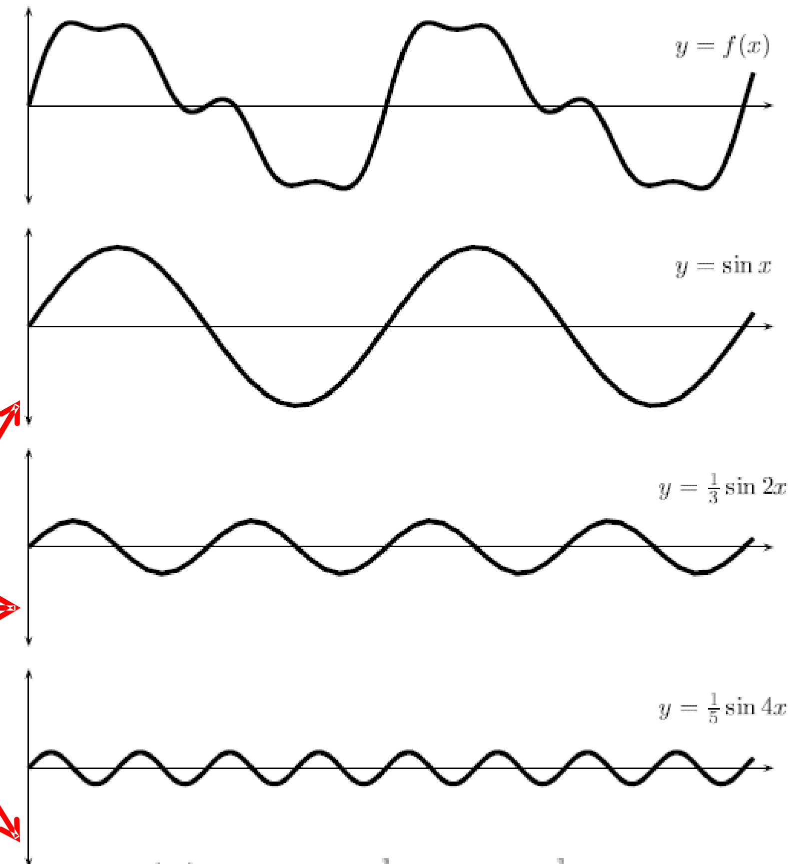
Fourier Transform: Why?

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



$f(x) = \sin x + \frac{1}{3}\sin 2x + \frac{1}{5}\sin 4x$

Fourier Transform: Some Observations



$$f(x) = \sin x + \frac{1}{3}\sin 2x + \frac{1}{5}\sin 4x$$

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

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

Frequency = 1x

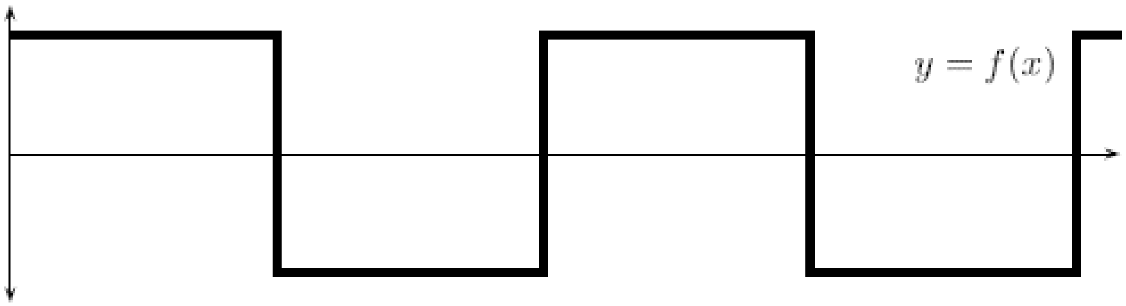
Frequency = 2x

Frequency = 4x

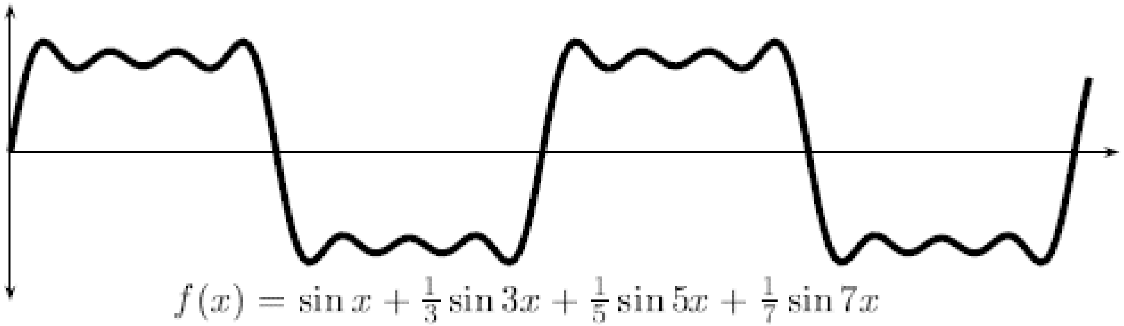
Observation 3: Different amounts of different sines added together (e.g. 1/3, 1/5, etc)

Fourier Transform: Another Example

Square wave



Approximation
Using sines



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

Who is Fourier?

- French mathematician and physicist (1768 - 1830)

Fourier Series Expansion

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

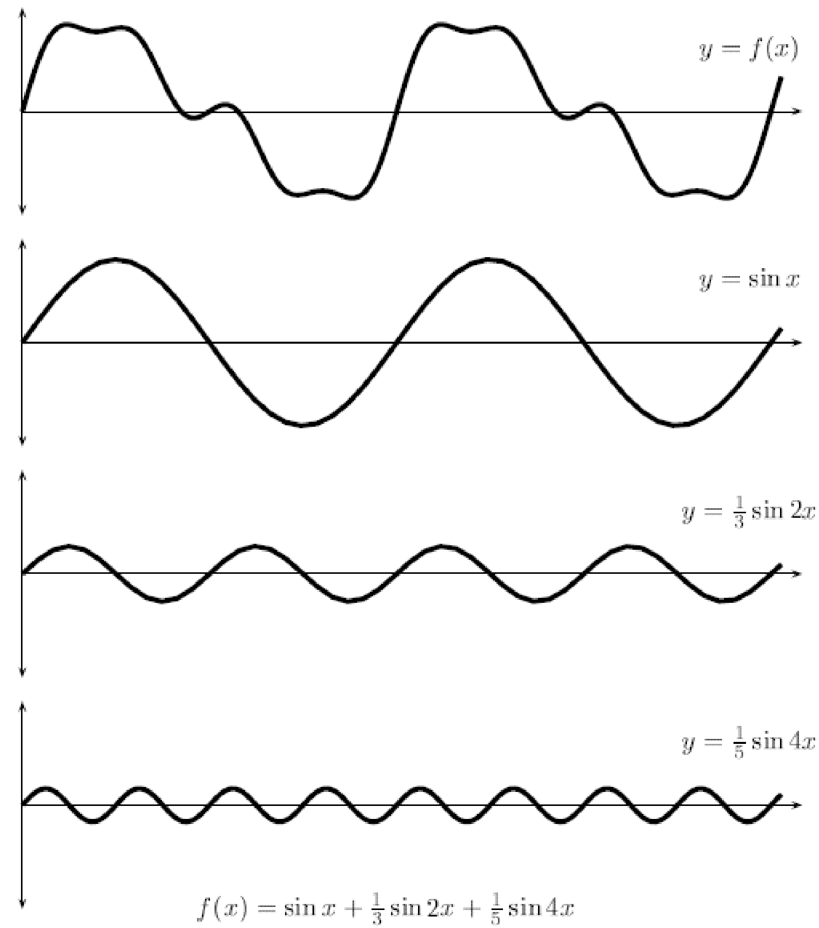
$$f(x) = a_0 + \sum_{n=1}^{\infty}(a_n \cos\frac{n\pi x}{T} + b_n \sin\frac{n\pi x}{T})$$

Where

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

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

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

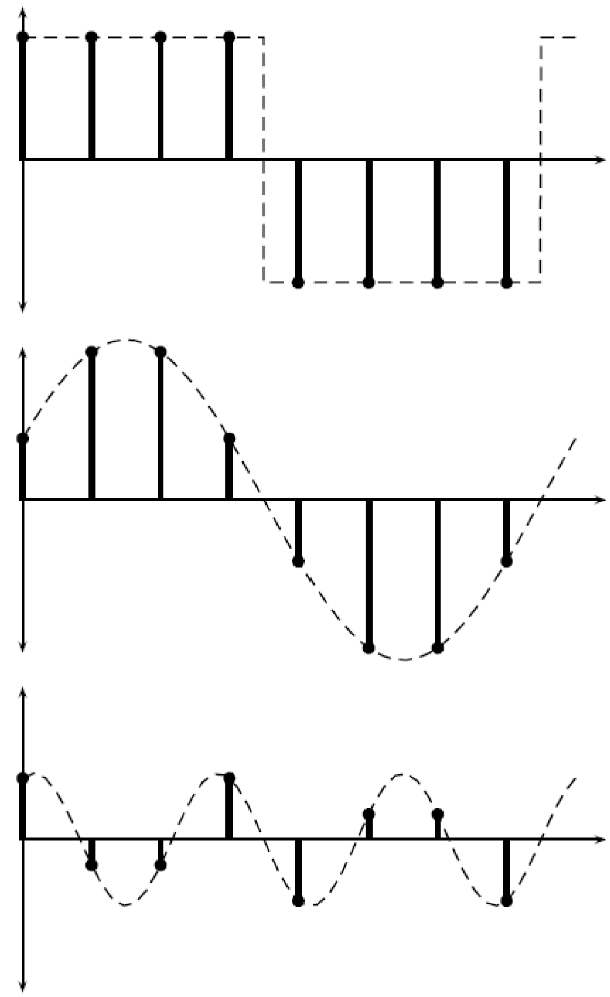


- a_n and b_n called **Fourier coefficients**

1D Discrete Fourier Transform

- **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



- Above is discrete approximation to square wave
- Can use Fourier transform to express as sum of 2 sine functions

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

- **Direct computation takes time:** 2^{2n} multiplications
- **FFT method takes:** $n2^n$ multiplications
- **Time savings:** $2^n/n$

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

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 (Formal Eq)

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

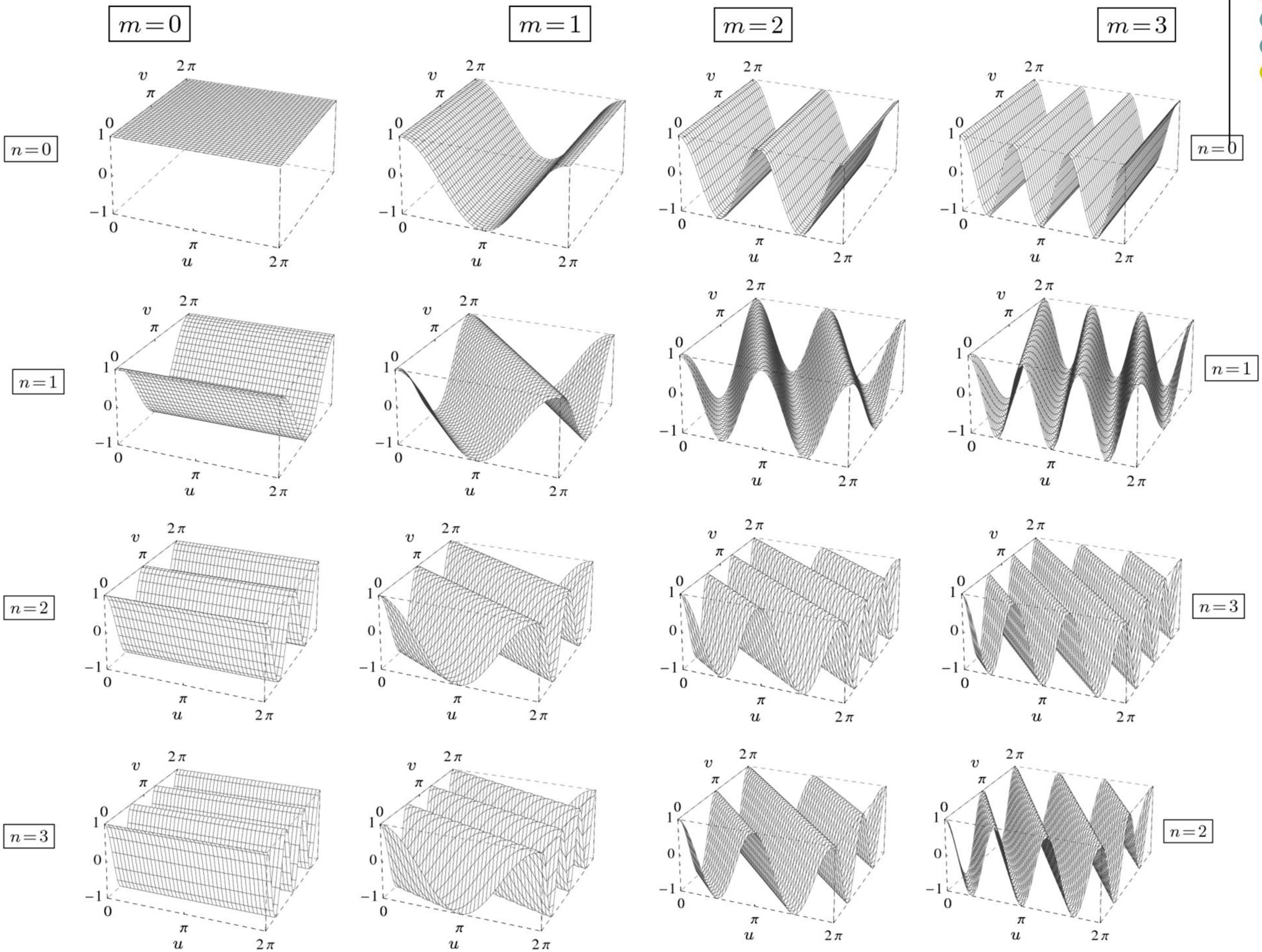
$$
\begin{aligned}
F(u,v) &= \sum_{x=0}^{M-1}\sum_{y=0}^{N-1} f(x,y)\exp\left[-2\pi i\left(\frac{xu}{M}+\frac{yv}{N}\right)\right].\\
f(x,y) &= \frac{1}{MN}\sum_{u=0}^{M-1}\sum_{v=0}^{N-1} F(u,v)\exp\left[2\pi i\left(\frac{xu}{M}+\frac{yv}{N}\right)\right].
\end{aligned}
$$

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*)

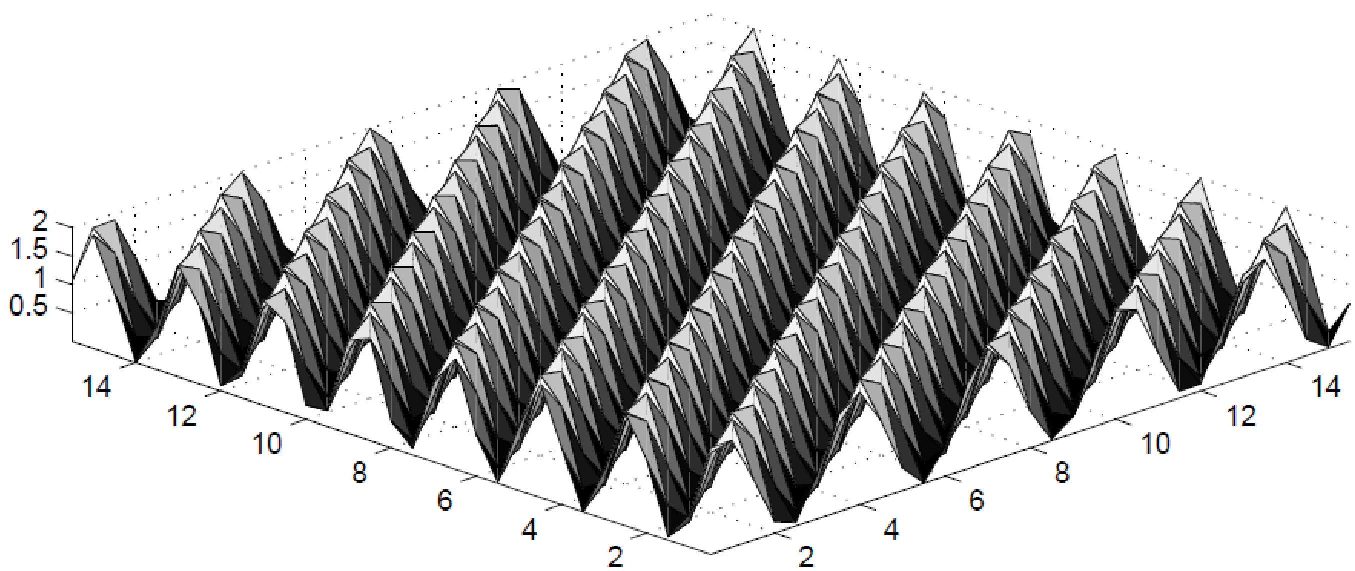
2D Cosines functions

Orientation depends on *m* and *n*



2D Fourier Transform: Corrugation of Functions

- Previous image just summed cosines
- Essentially, 2D Fourier Transform rewrites the original matrix by **summing sines and cosines in 2 direction = corrugations**



Corrugations result when sines and cosines are summed in 2 directions

Properties of 2D Fourier Transform

- All properties of 1D Fourier transform apply + additional properties
- **Similarity:** Forward and inverse transforms are similar except
  - scale factor 1/*MN* in inverse transform
  - 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 formulae 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)**

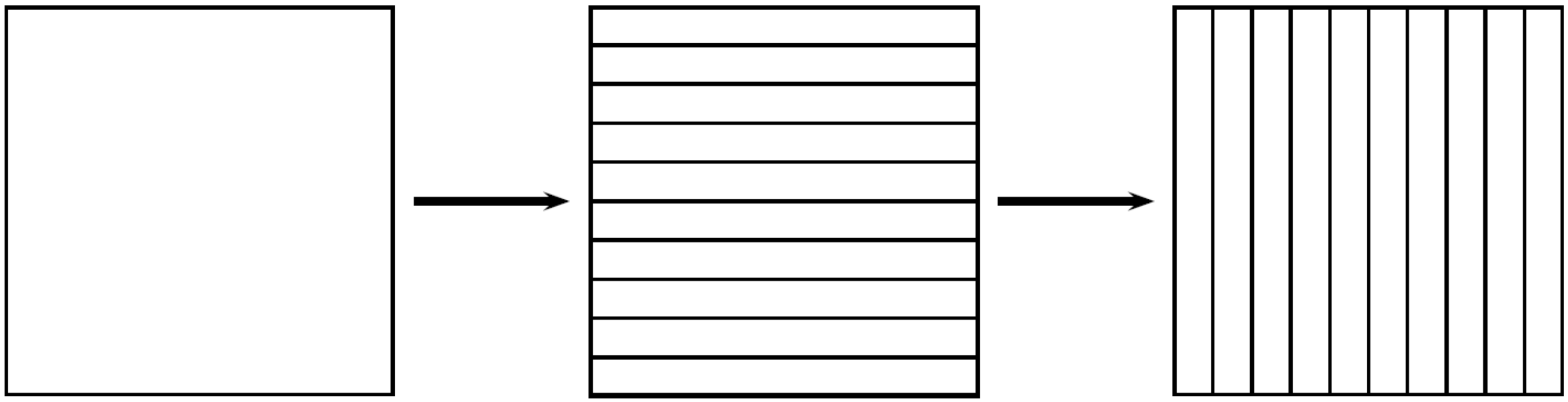
- Formula above can be broken down into simpler formulae for 1D DFT

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

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

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

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

$$\mathcal{F}(kf) = k\mathcal{F}(f) \qquad \textbf{\textit{k} is a scalar}$$

- 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:
  - 32x32 = 1024 multiplications for each pixel
  - Total multiplications for entire image = 512 x 512 x 1024 = **26,84,35,456** 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 = **1,44,17,920**

DC Component

- Recall that:

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

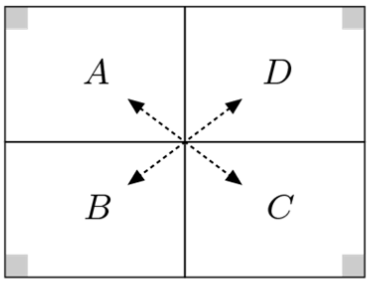
- The value *F(0,0)* of the DFT is called the **dc coefficient**
- If we put $u = v = 0$, then

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

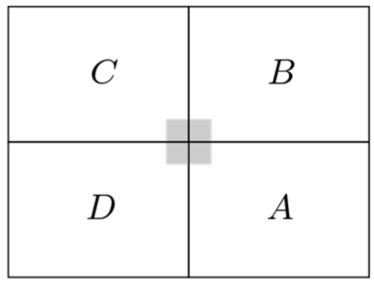
- Essentially ***F(0,0)* is the sum of all terms in the original matrix**

Centering DFT Spectrum

- *F(0,0)* at top left corner
- For display, convenient to have DC component in center
- Just swap four quadrants of Fourier transform

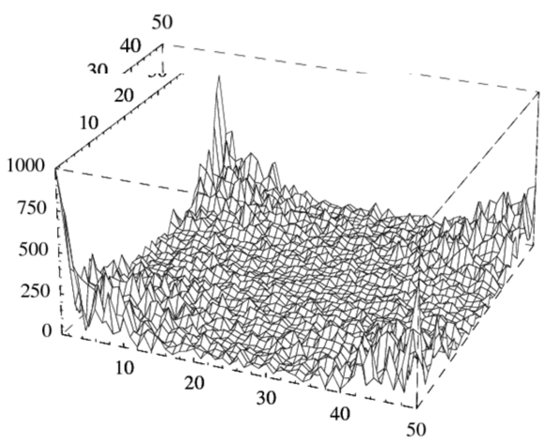


(a)

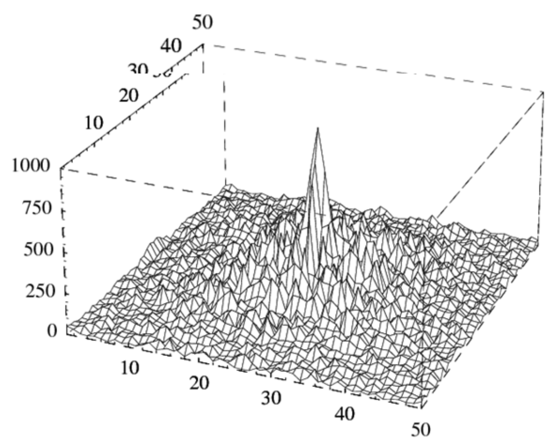


(b)

Swap 4 quadrants to center DC component



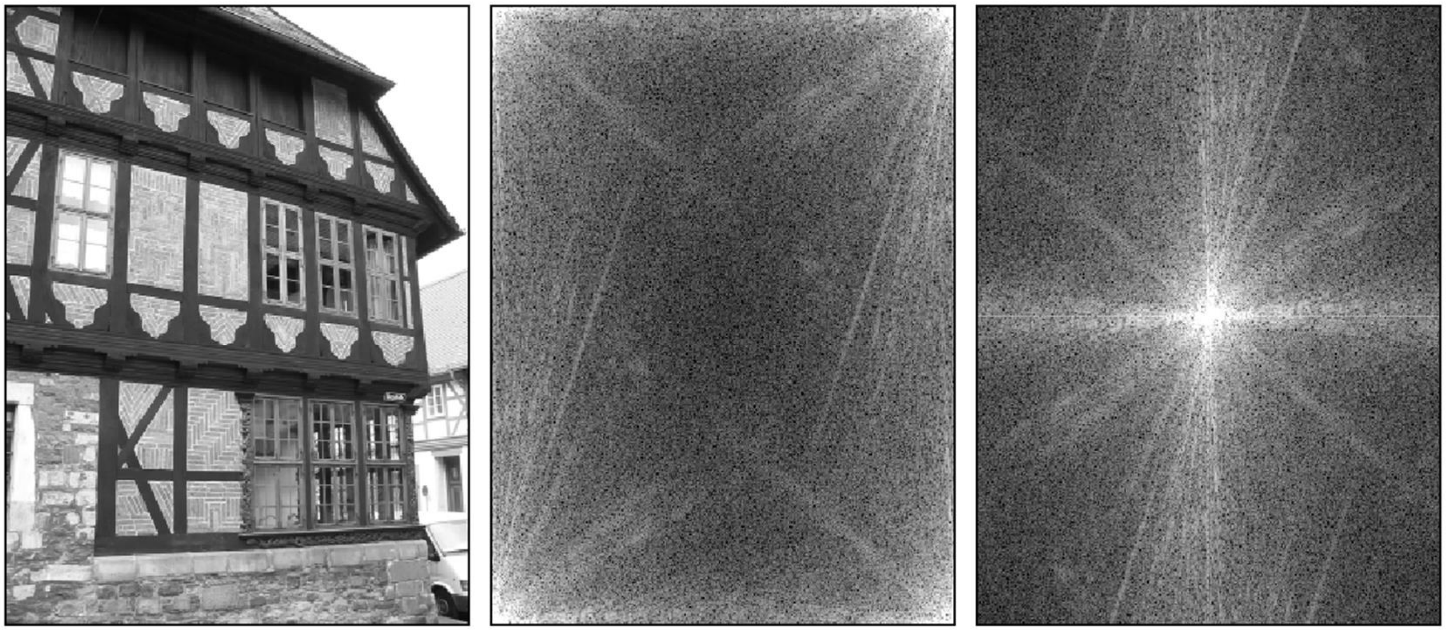
(c)



(d)

DFT spectrum after centering

Centering DFT Spectrum



(a) (b) (c)

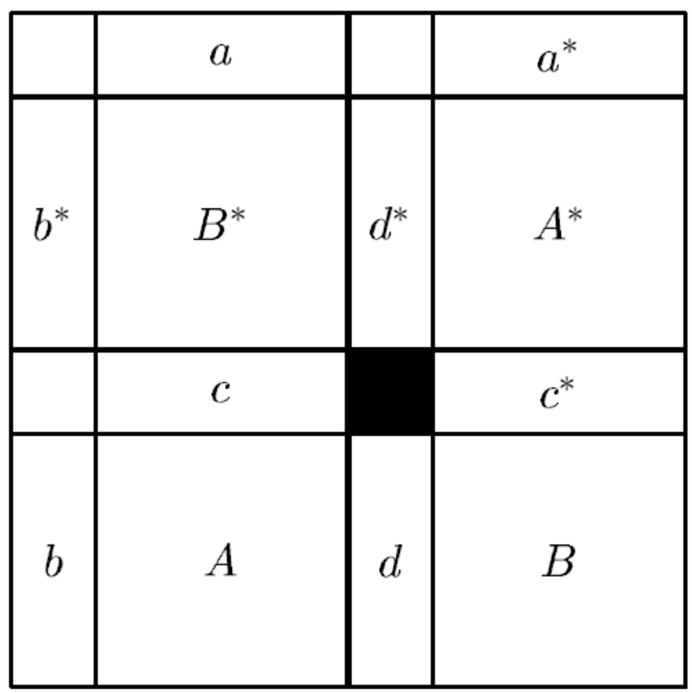
Original Image

Non-centered spectrum

Centered spectrum

Conjugate Symmetry

- DFT shows conjugate symmetry
- Half of the transform is mirror image of conjugate of other half
- **Implication:** information is contained in only half of a transform
- Other half is redundant



Conjugate Symmetry

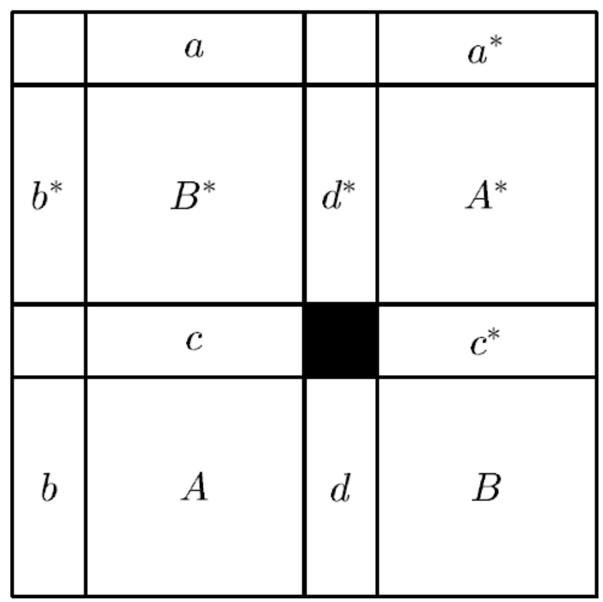
- Thus, if we put *u* = -*u,* and *v* = -*v* into the DFT equation

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

Then

$$\mathcal{F}(u,v) = \mathcal{F}^*(-u+pM, -v+qN)$$

for any integers *p* and *q*



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

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

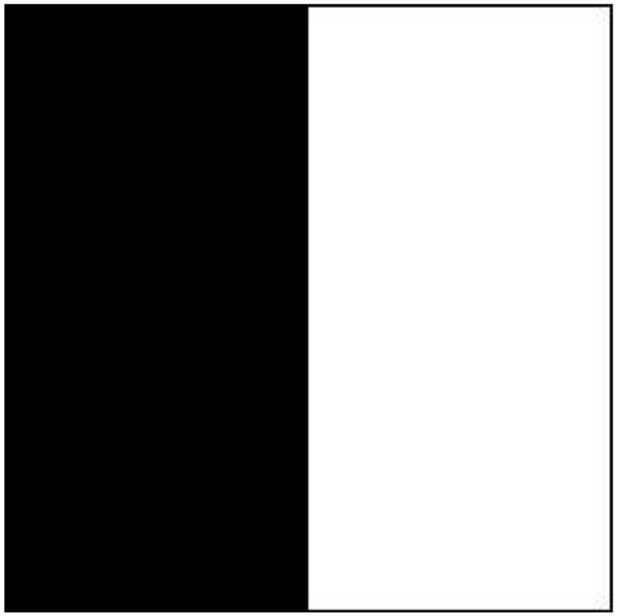
Examples of DFTs

- 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

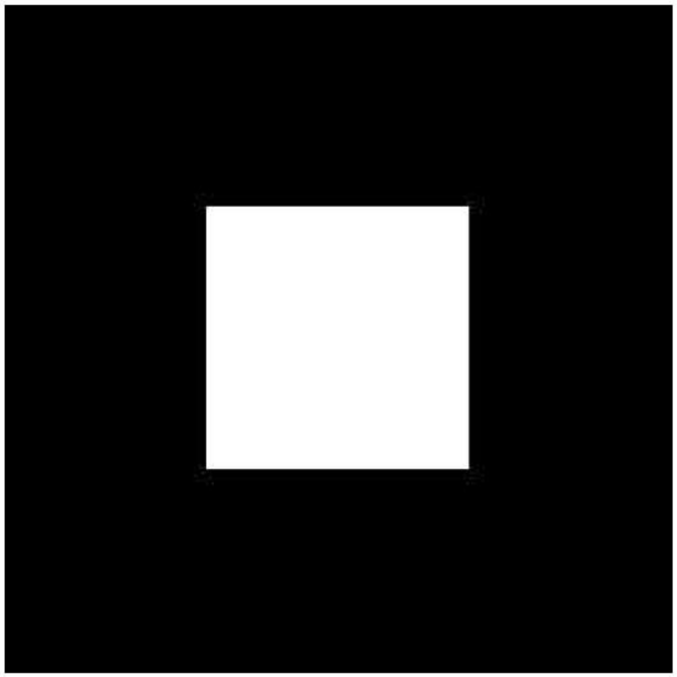
- Consider DFT of image with single edge
- For display, DC component shifted to center
- Log of magnitudes of Fourier Transform displayed



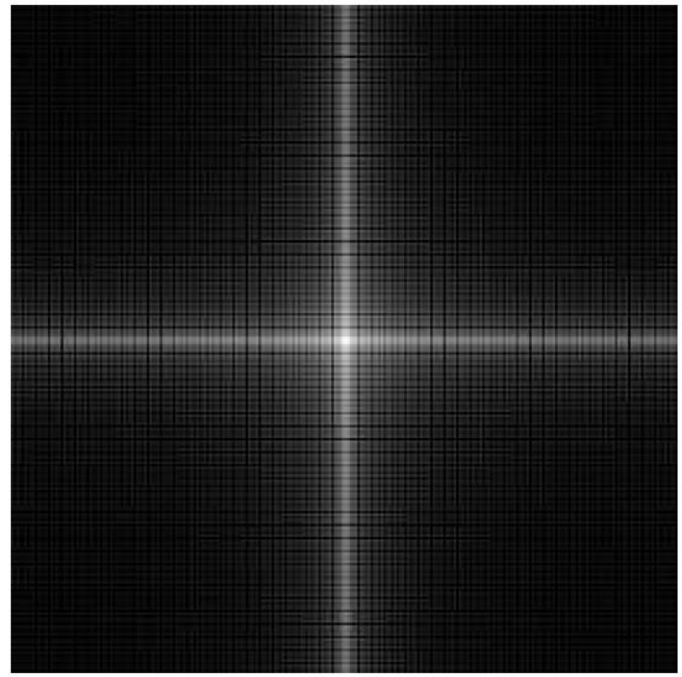
Image

DFT

DFT Example: A Box

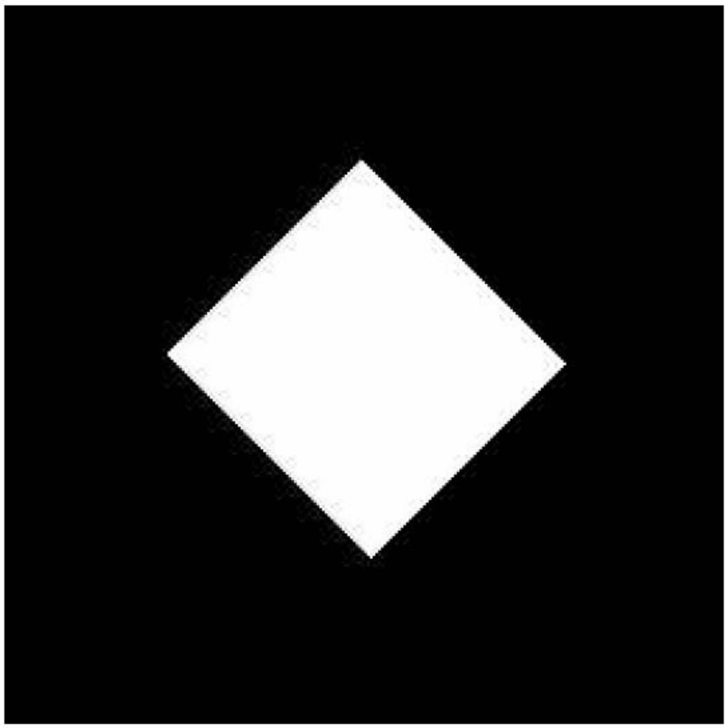


Box

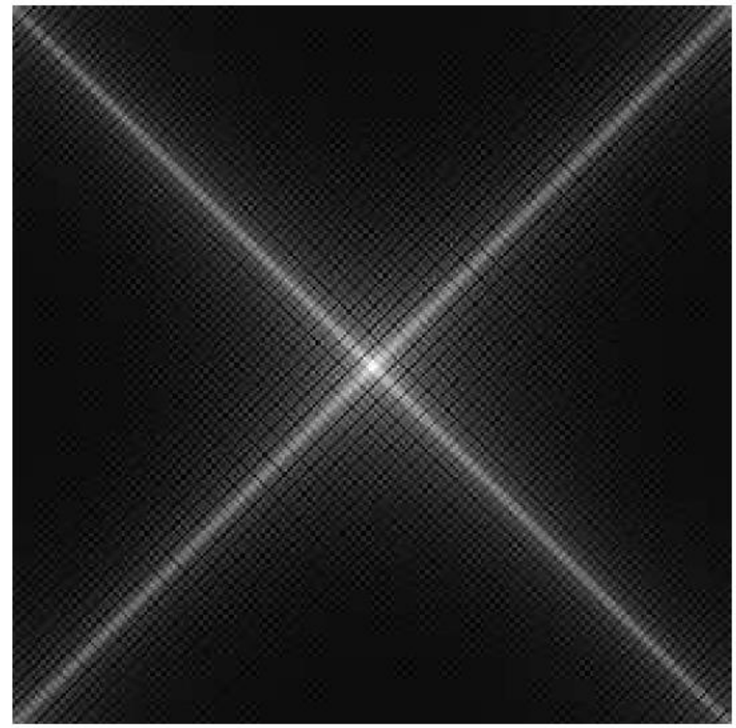


DFT

DFT Example: Rotated Box



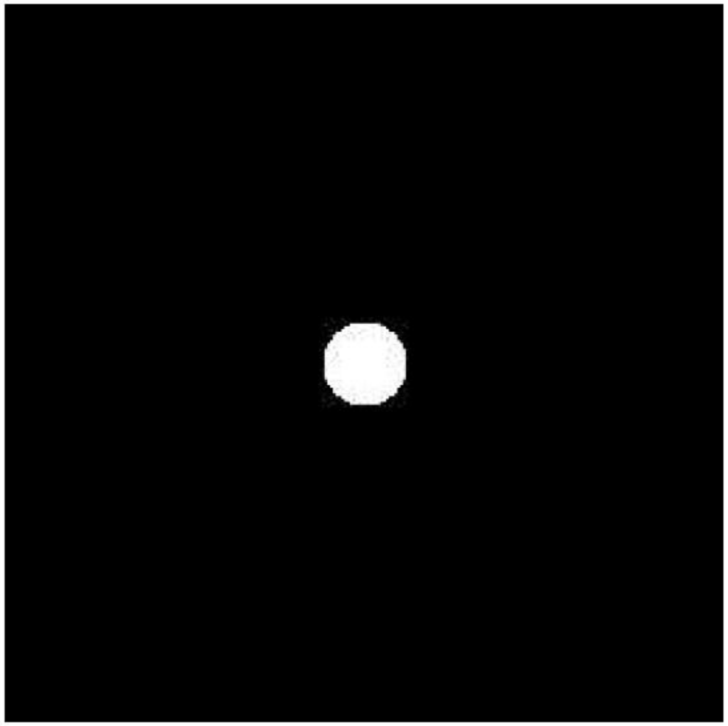
Rotated Box



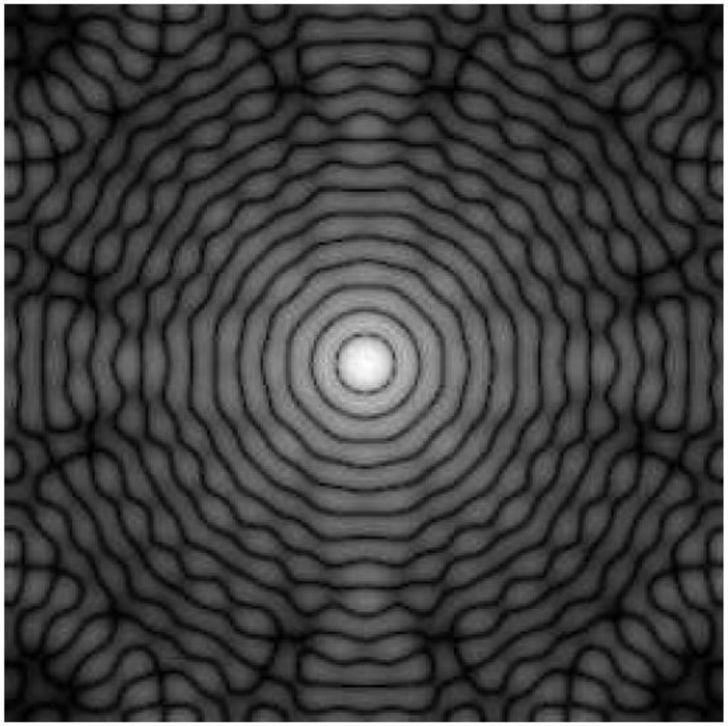
DFT

DFT Example: A Circle

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



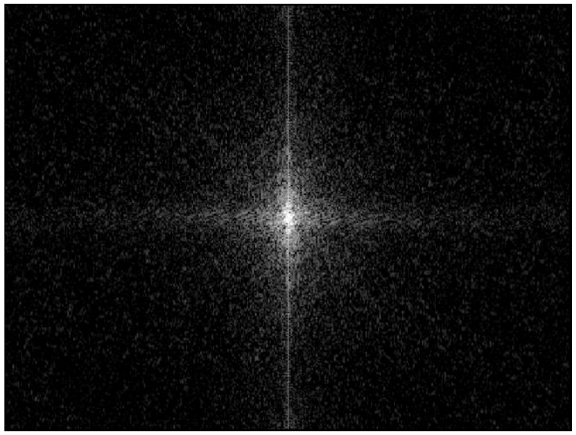
Circle



DFT

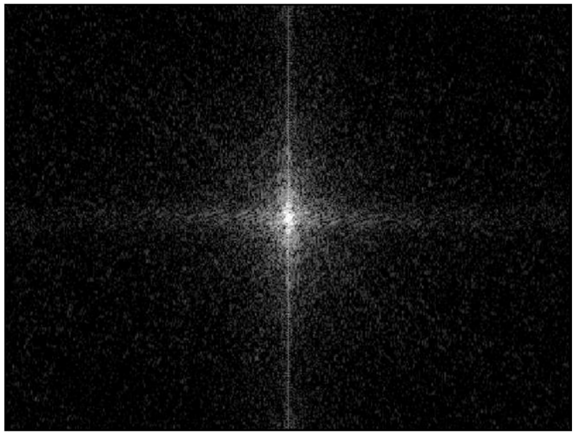
DFT Computation: Repeated Image

- 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 affect 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

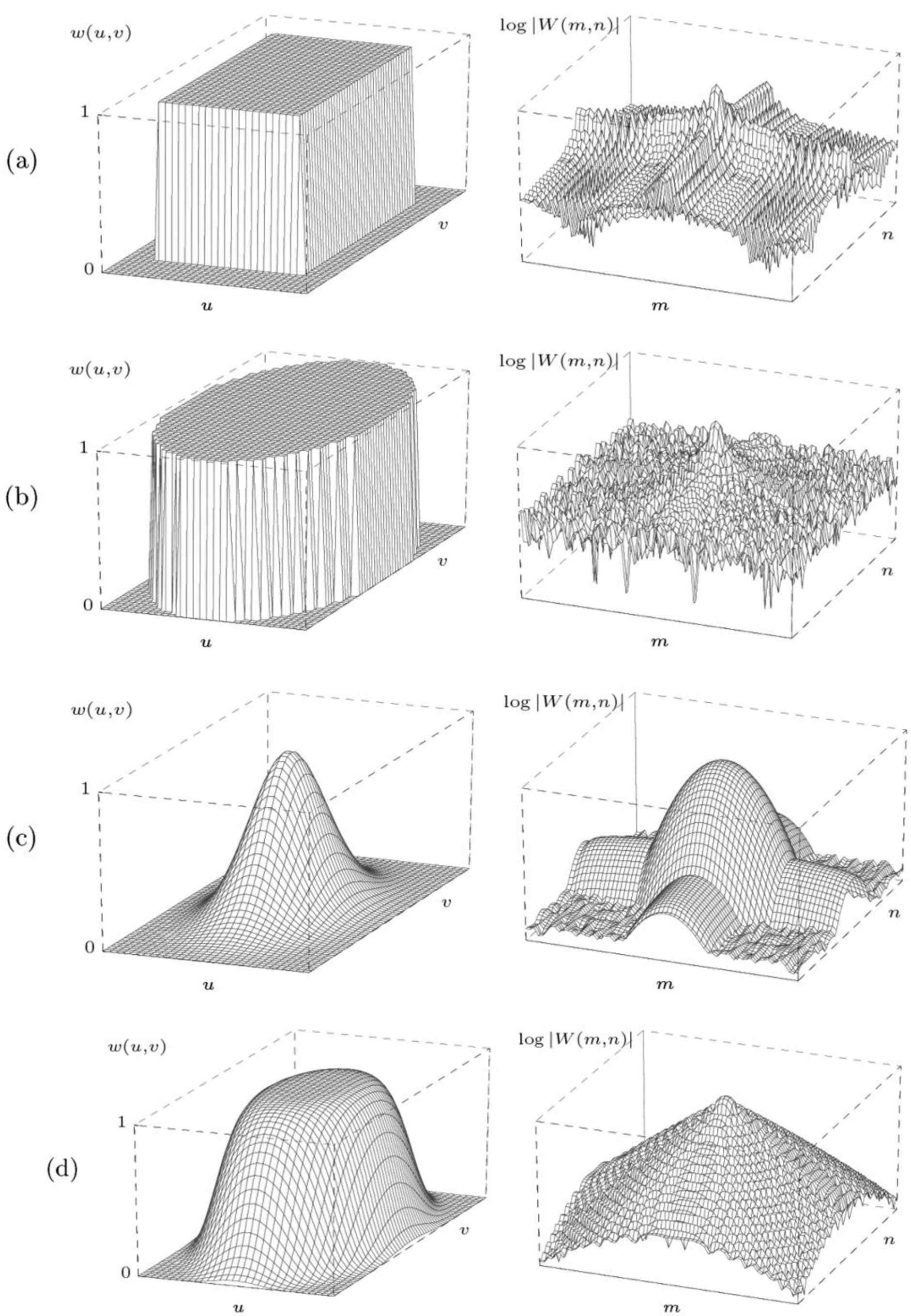


Some Proposed Windowing Functions

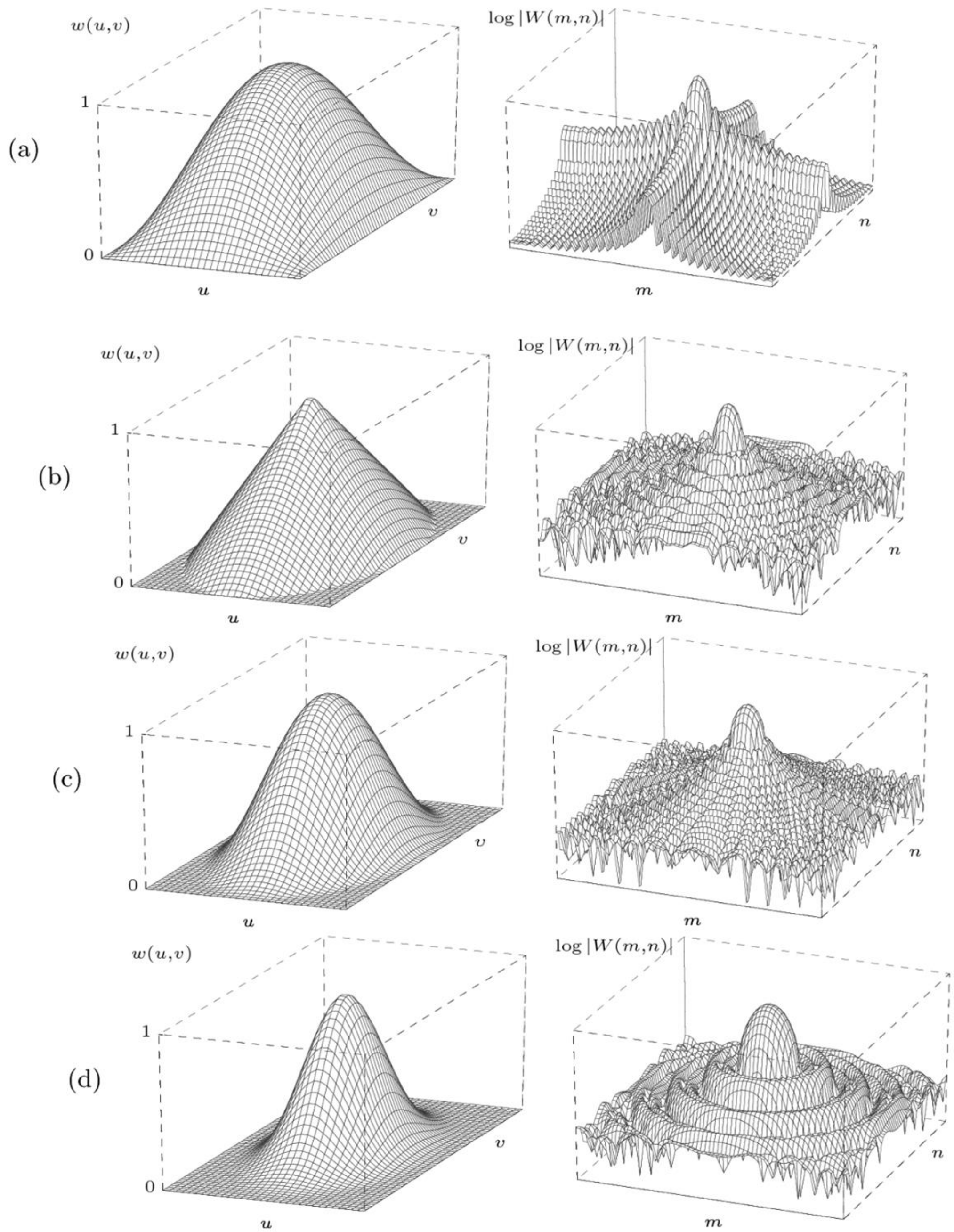
Definitions:

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

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

Application of Windowing Functions