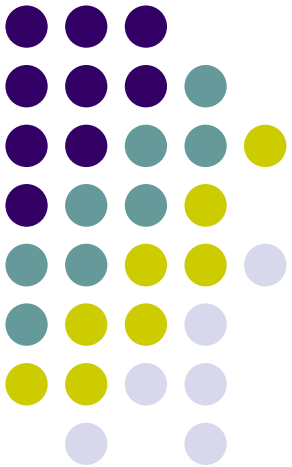
Digital Image Processing (CS/ECE 545) Lecture 10: Discrete Fourier Transform (DFT)

Prof Emmanuel Agu

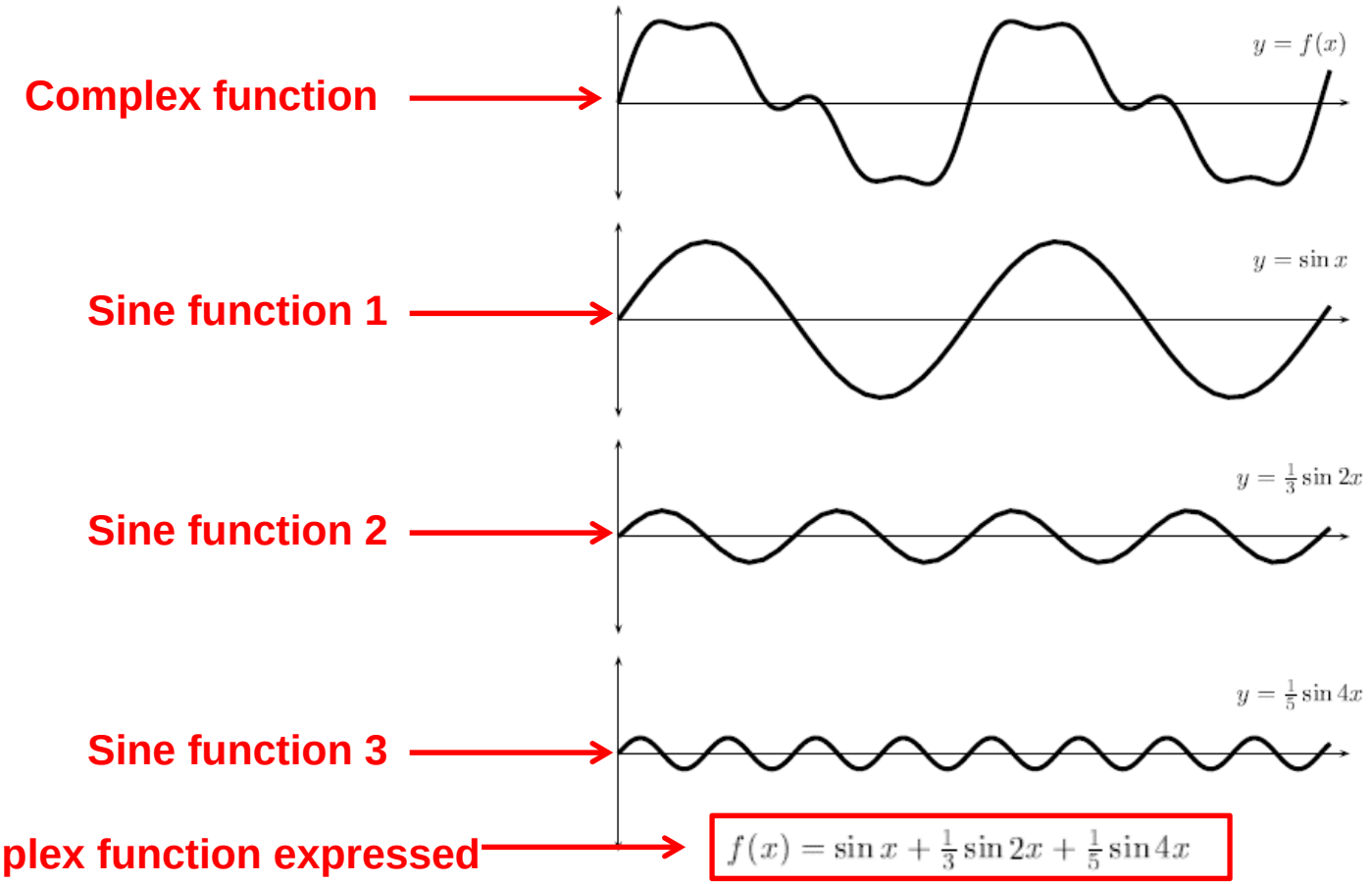
Computer Science Dept.

Worcester Polytechnic Institute (WPI)

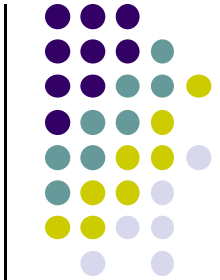


Fourier Transform

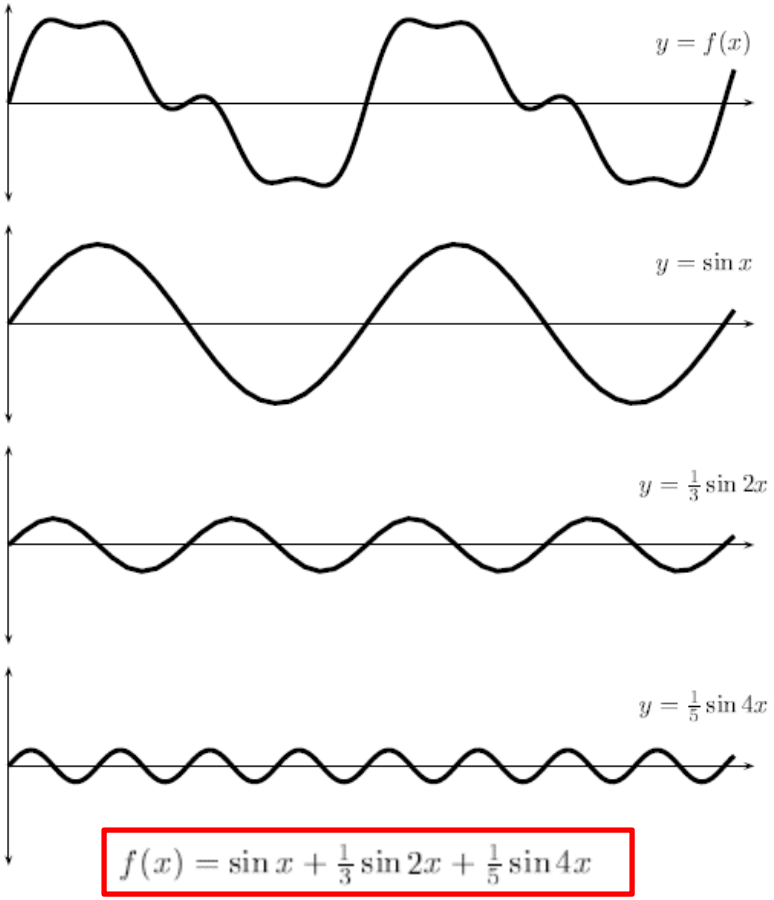
- **Main idea:** Any periodic function can be decomposed into a summation of sines and cosines



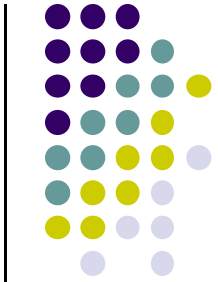
Fourier Transform: Why?

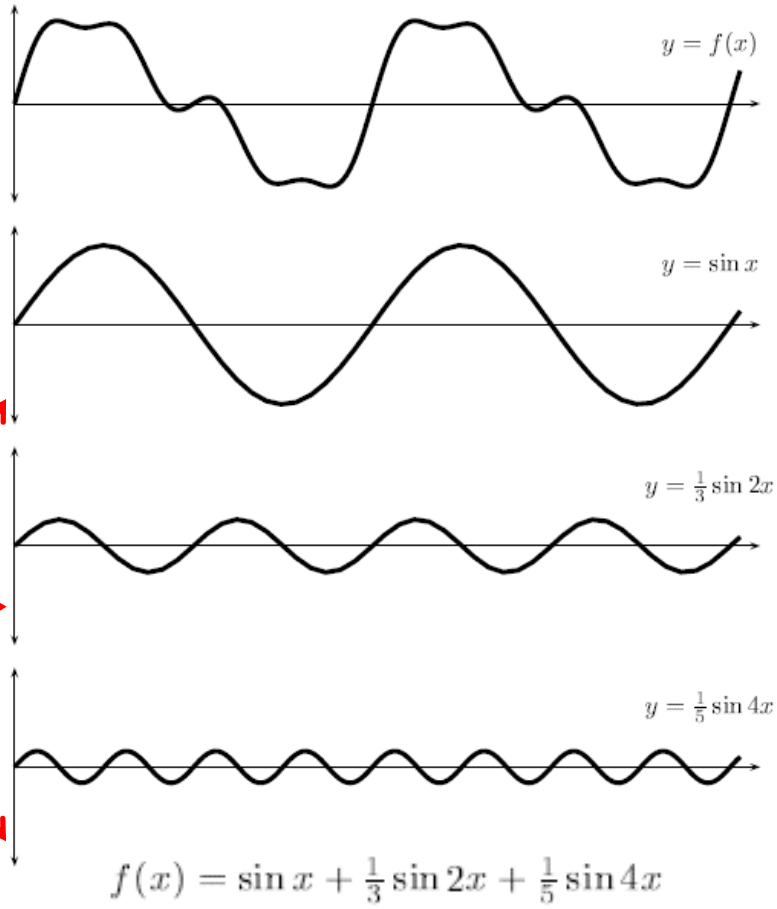


- Mathematially easier to analyze effects of transmission medium, noise, etc on simple sine functions, then add to get effect on complex signal



Fourier Transform: Some Observations





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

Frequency = 1x

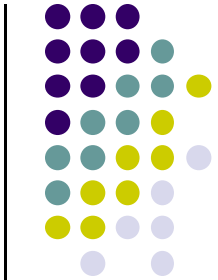
Frequency = 2x

Frequency = 4x

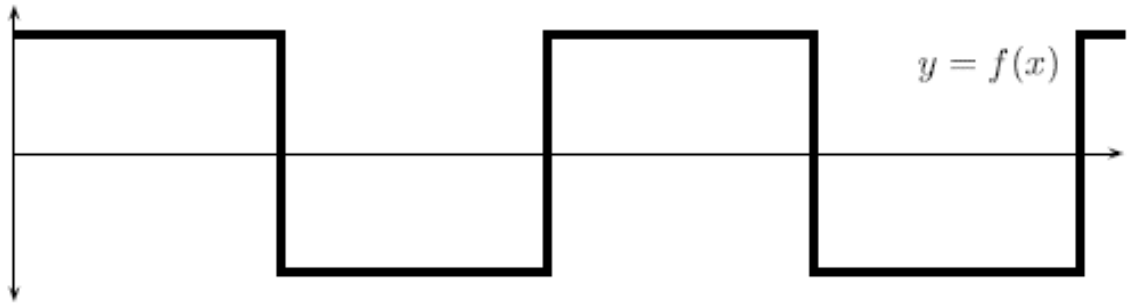
Observation 1: The sines have different frequencies (not same)

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

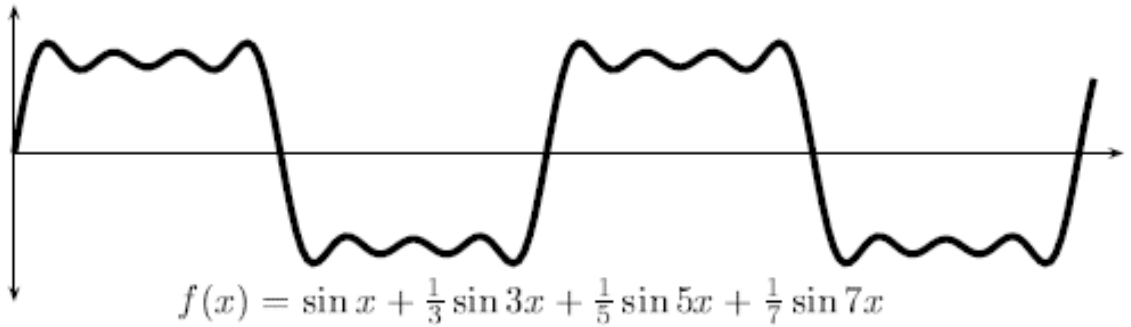
Fourier Transform: Another Example



Square wave



Approximation
Using sines



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
- Lived 1768 - 1830

Fourier Series Expansion

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

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

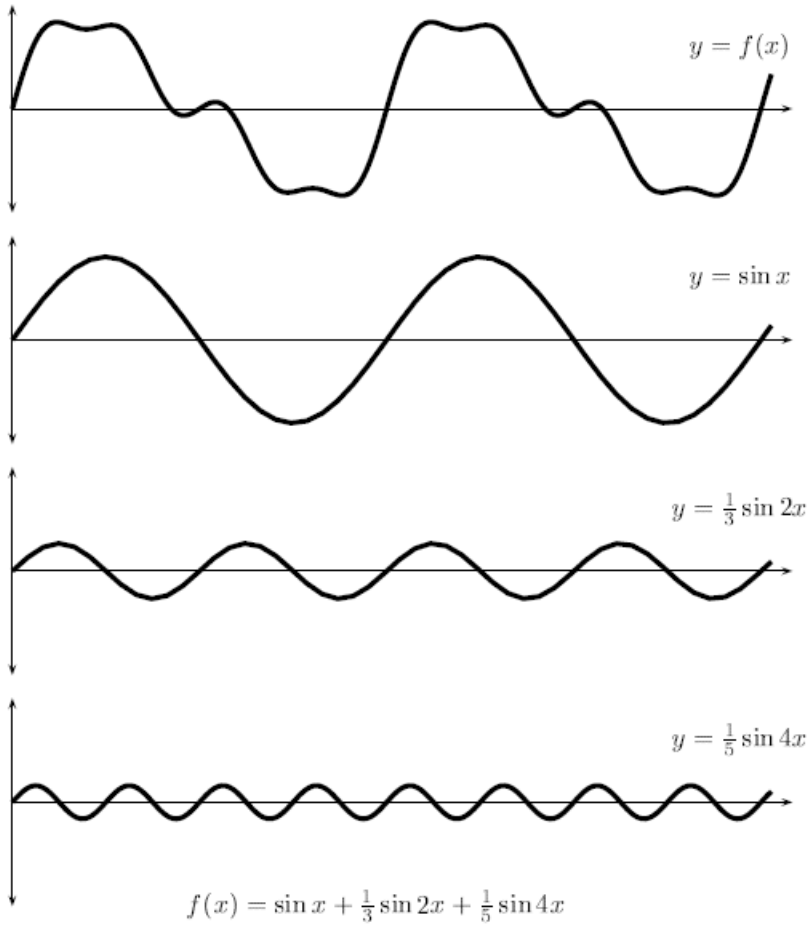
Where

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

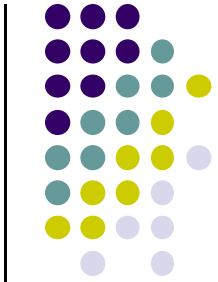
$$a_n = \frac{1}{T}\int_{-T}^{T} f(x)\cos\frac{n\pi x}{T}dx,\ n = 1,2,3,\ldots$$

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

- a_n and b_n called **Fourier coefficients**



Complex Form of Fourier Series Expansion



- Fourier series expansion of *f(x)*

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

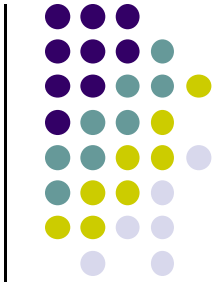
can be expressed in complex form as:

$$f(x) = \sum_{n=-\infty}^{\infty} c_n \exp\left(\cos\frac{in\pi x}{T}\right)dx$$

where

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

Fourier Series of Periodic Functions



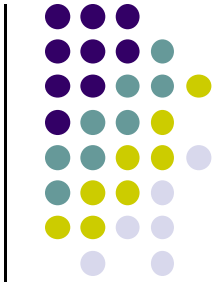
- (Almost) any periodic function *g(x)* with fundamental frequency ω_0 can be described as a sum of sinusoids

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

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** (integration of densely packed sines and cosines

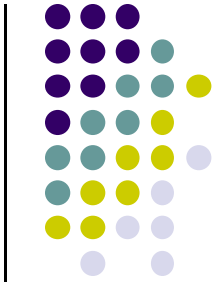
$$g(x) = \int_0^\infty A_\omega \cos(\omega x) + B_\omega \sin(\omega x)\ \mathrm{d}\omega$$

where coefficients can be found as

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

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

Fourier Transform



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

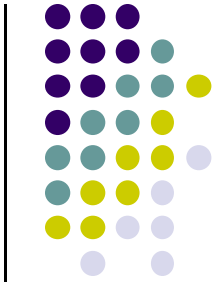
$$G(\omega) = \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{\infty} g(x) \cdot \Big[\cos(\omega x) - \mathrm{i} \cdot \sin(\omega x)\Big] \, \mathrm{d}x$$
$$= \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{\infty} g(x) \cdot e^{-\mathrm{i}\omega x} \, \mathrm{d}x.$$

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

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

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

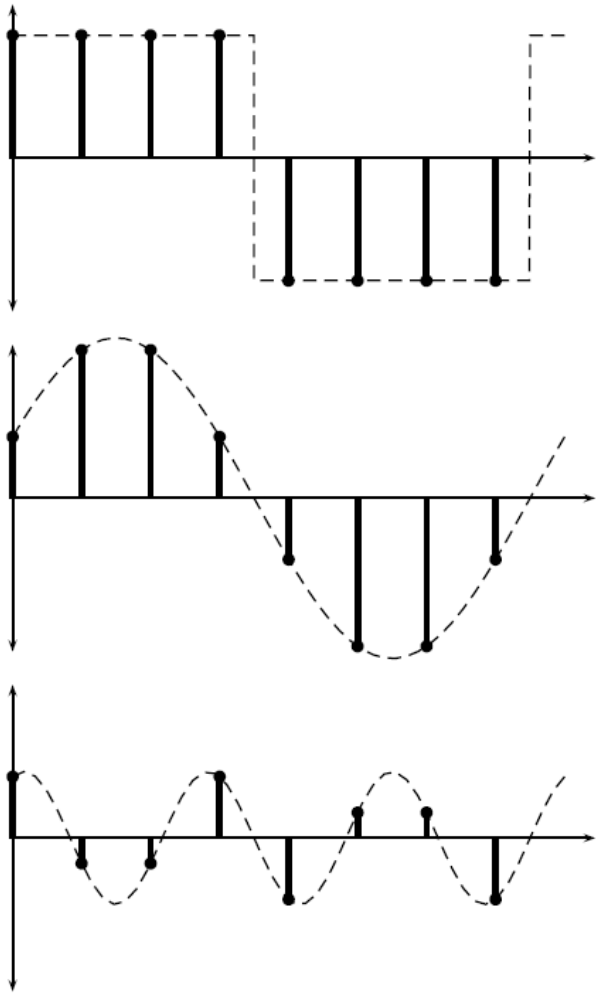
1D Discrete Fourier Transform



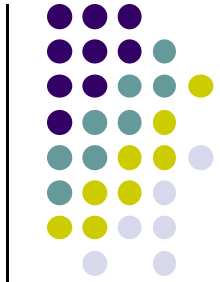
- **Image is a discrete 2D function!!**
- For discrete functions we need only finite number of functions
- For example, consider the discrete sequence

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



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

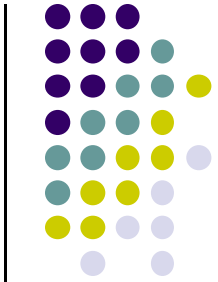
$$F_u = \frac{1}{N} \sum_{x=0}^{N-1} \exp\left[-2\pi i \frac{xu}{N}\right] f_x$$

Compare with complex form of coefficients

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

- **Interpretation:** Basis functions completes *x* cycles over distance *N*
- Similar to Fourier series expansion
- Instead of integral, we now have a finite sum

Inverse 1D DFT



- Formula for inverse

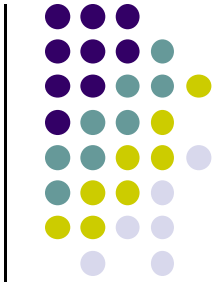
$$x_u = \sum_{x=0}^{N-1} \exp\left[2\pi i \frac{xu}{N}\right] F_u$$

DFT equation

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

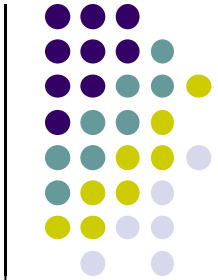
- Compared to DFT equation,
  - The inverse has no scaling factor 1/N
  - The sign inside the square bracket has been changed to +ve

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

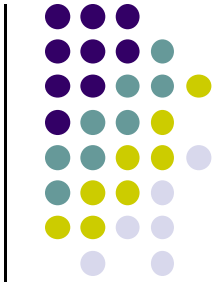
FFT Computation Time Savings



- **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

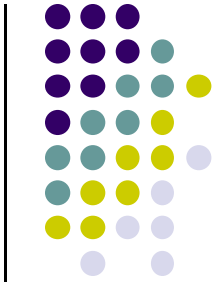
$$F = \mathcal{F}(f)$$

- 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



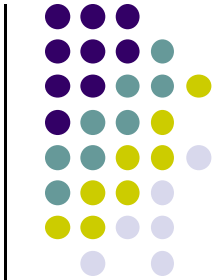
- Can be expressed as weighted sum of sines and cosines

$$e^{\mathrm{i}2\pi\left(\frac{mu}{M}+\frac{nv}{N}\right)} = e^{\mathrm{i}(\omega_m u+\omega_n v)}$$

$$= \underbrace{\cos\left[2\pi\left(\frac{mu}{M}+\frac{nv}{N}\right)\right]}_{\boldsymbol{C}_{m,n}^{M,N}(u,v)} + \mathrm{i}\cdot\underbrace{\sin\left[2\pi\left(\frac{mu}{M}+\frac{nv}{N}\right)\right]}_{\boldsymbol{S}_{m,n}^{M,N}(u,v)}$$

- *m* specifies how many cycles basis function performs over distance of *M* units
- *n* specifies how many cycles basis function performs over distance of *N* units

2D Fourier Transform



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

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

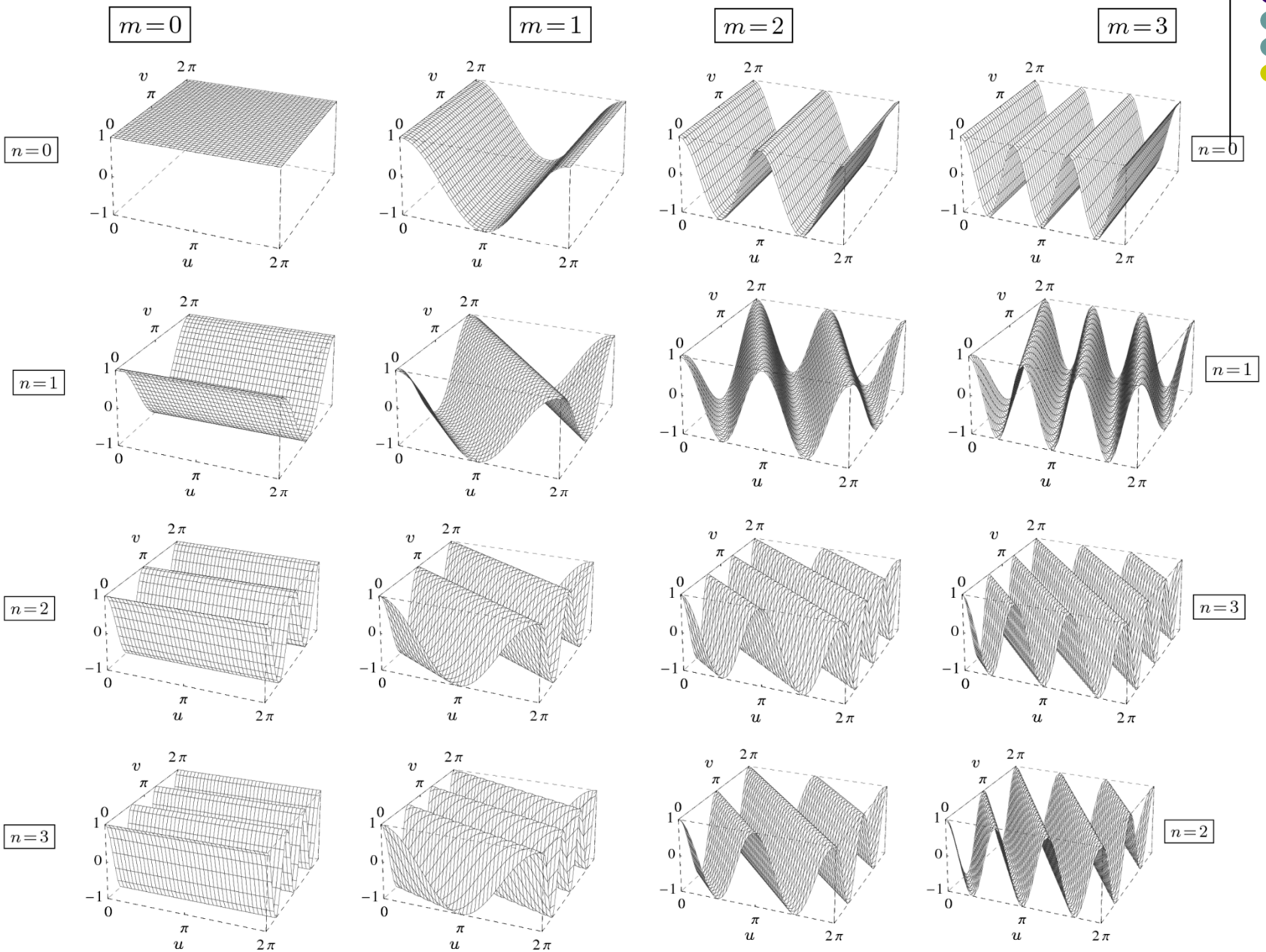
$$f(x,y) = \frac{1}{MN}\sum_{u=0}^{M-1}\sum_{v=0}^{N-1} F(u,v)\exp\left[2\pi i\left(\frac{xu}{M}+\frac{yv}{N}\right)\right].$$

where

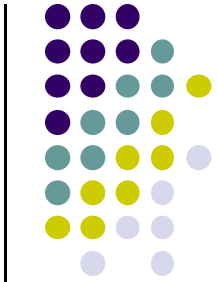
- *x* indices go from *0… M – 1 (x* cycles over distance *M*)
- *y* indices go from *0… N – 1 (*y cycles over distance *N*)

2D Cosines functions

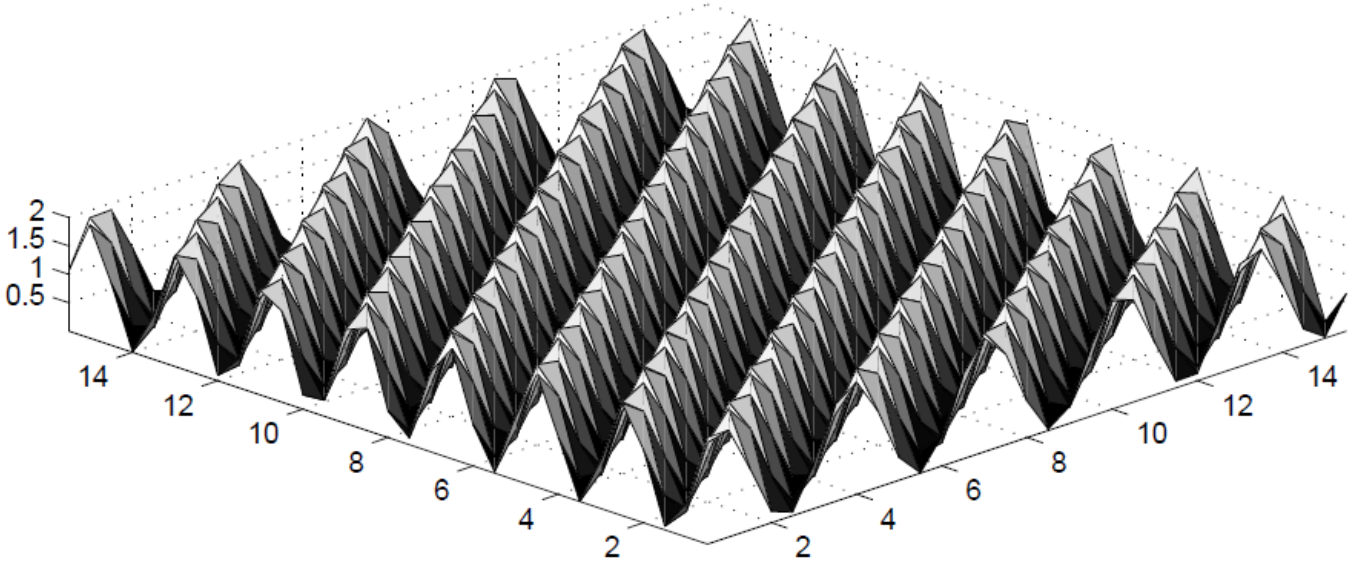
Orientation depend on *m* and *n*



2D Fourier Transform: Corrugation of Functions

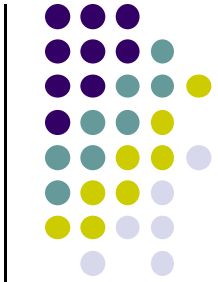


- Previous image just summed cosines
- Essentially, 2D Fourier Transform rewrites the original matrix by **summing sines and cosines in 2 direction = corrugations**



Corrugations result when sines and cosines are summed in 2 directions

Properties of 2D Fourier Transform

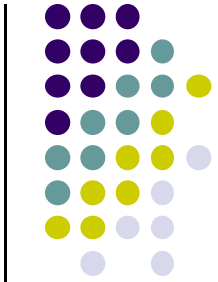


- All properties of 1D Fourier transform apply + additional properties
- **Similarity:** Forward and inverse transforms are similar except
  1. scale factor 1/*MN* in inverse transform
  2. Negative sign in exponent of forward transform

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

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

Properties of 2D Fourier Transform



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

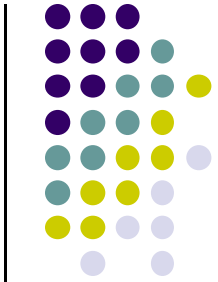
$$f(x,y) = \frac{1}{MN}\sum_{u=0}^{M-1}\sum_{v=0}^{N-1} F(u,v)\exp\left[2\pi i\left(\frac{xu}{M}+\frac{yv}{N}\right)\right].$$

- **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!)
  - **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

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

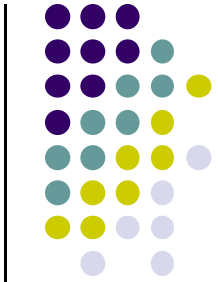
2D DFT | 1D DFT (row) | 1D DFT (column)

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

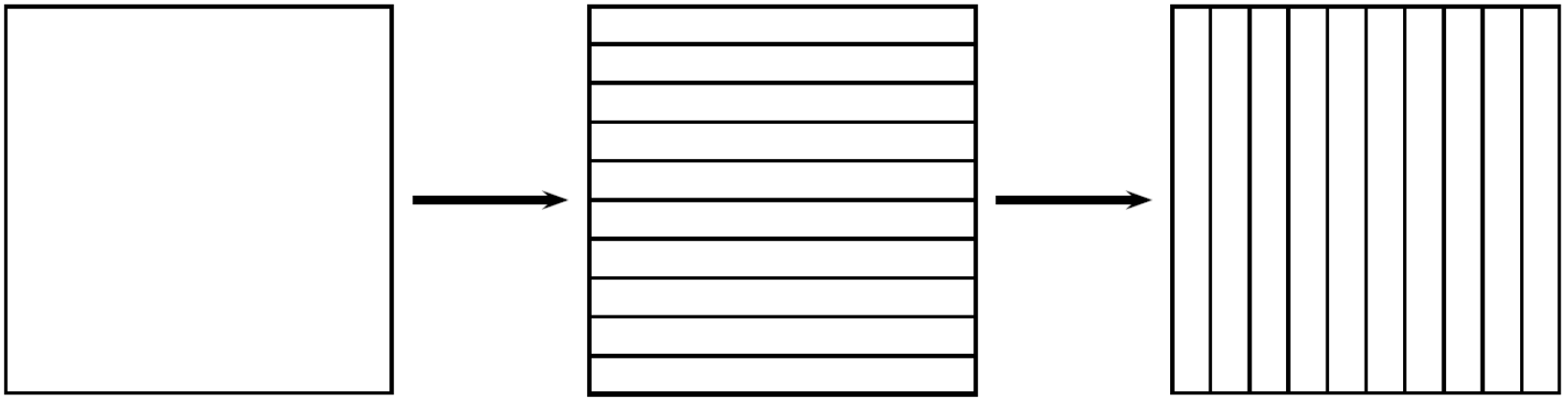
$$F(u) = \sum_{x=0}^{M-1} f(x)\exp\left[-2\pi i\frac{xu}{M}\right],$$

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

Properties: Separabilty of 2D DFT

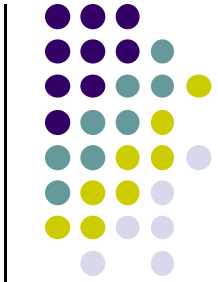


- Using their separability property, can use 1D DFTs to calculate rows then columns of 2D Fourier Transform



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

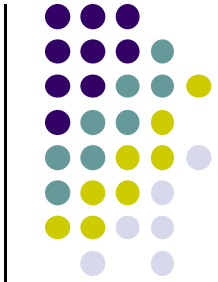
Implementation of 2D DFT



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

```
1: SEPARABLE 2D-DFT (g)        ▷ g(u,v) ∈ ℂ, 0 ≤ u < M, 0 ≤ v < N
2:     for v ← 0 ... N−1 do
3:         Let g(·,v) be the vth row vector of g:
               Replace g(·,v) by DFT(g(·,v))
4:     for u ← 0 ... M−1 do
5:         Let g(u,·) be the uth column vector of g:
               Replace g(u,·) by DFT(g(u,·))
       Remark: g(u,v) ≡ G(u,v) ∈ ℂ now contains the discrete 2D spec-
       trum.
6:     return g
```

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)

$$d = f + n$$

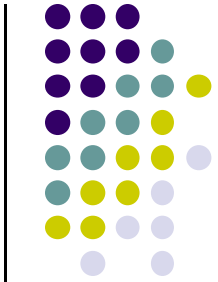
Where *f* is original image, *n* is the noise, *d* is degraded image

- 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

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

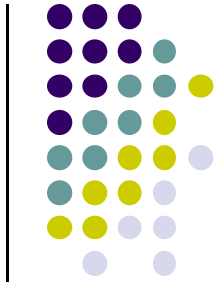
- Essentially

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

Or equivalently the convolution *M * S*

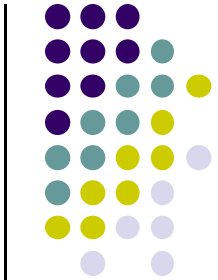
$$\mathcal{F}(M * S) = \mathcal{F}(M) \cdot \mathcal{F}(S')$$

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:

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

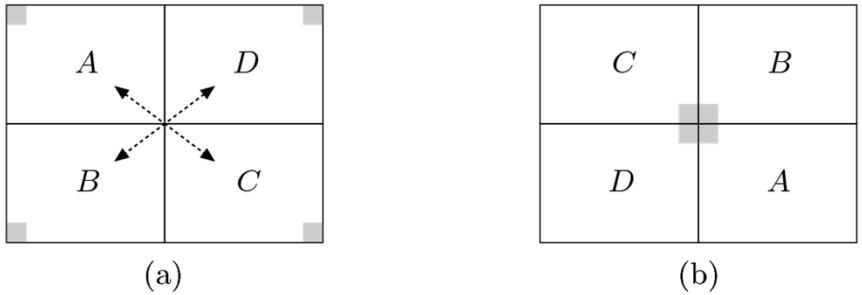
- The value *F(0,0)* of the DFT is called the **dc coefficient**
- If we put $u = v = 0$, then

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

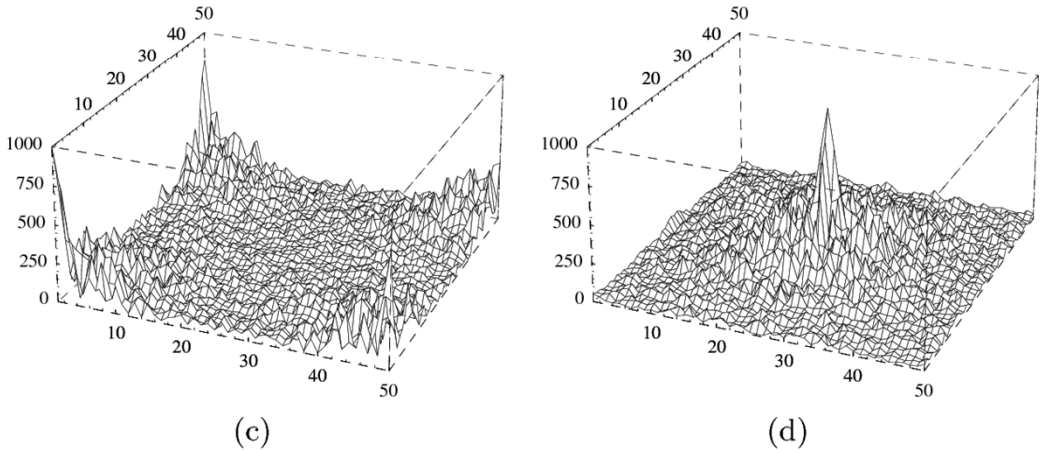
- Essentially *F(0,0)* is the sum of all terms in the original matrix

Centering DFT Spectrum

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



Swap 4 quadrants to center DC component

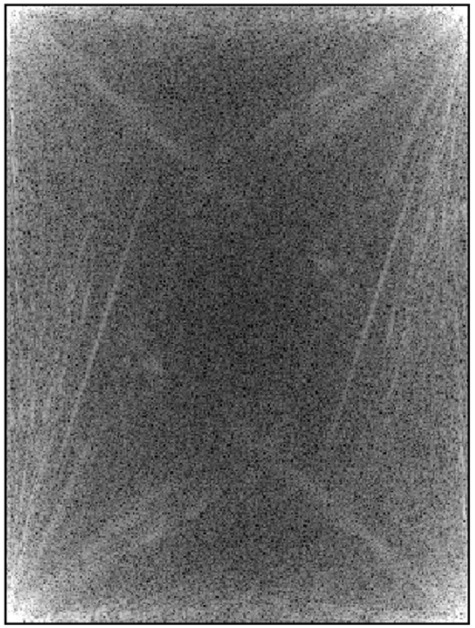


DFT spectrum after centering

Centering DFT Spectrum

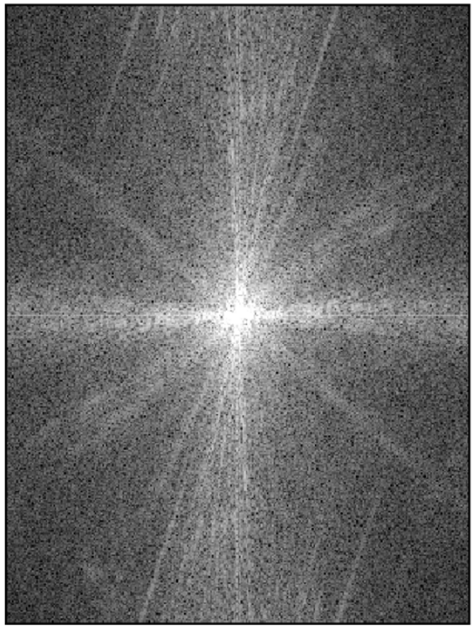
(a)

Original Image



(b)

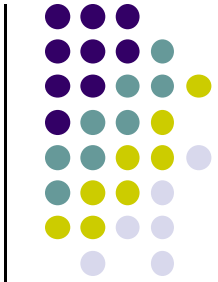
Non-centered spectrum



(c)

Centered spectrum

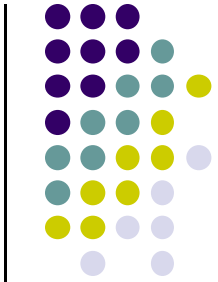
Conjugate Symmetry



- 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

	a		a^*
b^*	B^*	d^*	A^*
	c		c^*
b	A	d	B

Conjugate Symmetry



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

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

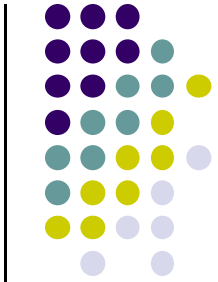
Then

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

for any integers *p* and *q*

	a		a^*
b^*	B^*	d^*	A^*
	c		c^*
b	A	d	B

Displaying Transforms

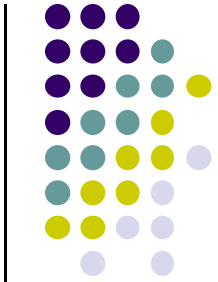


- As elements are complex numbers, we can view magnitudes $|F(u,v)|$ directly
- Displaying magnitudes of Fourier transforms called **spectrum** of the transform

- **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

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

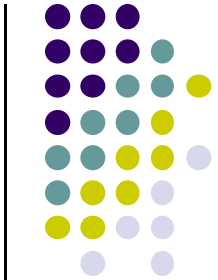
Examples of DFTs



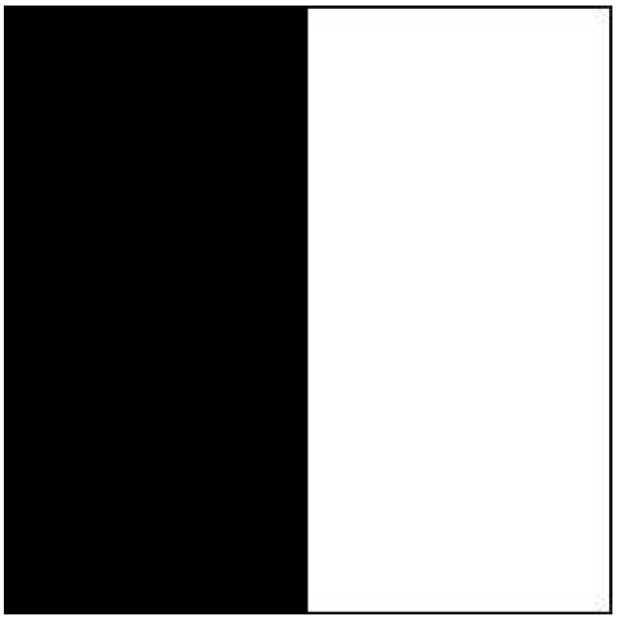
- Suppose we have as input a constant image of all 1's, $f(x,y) = 1$
- The DFT yields just a DC component, 0 everywhere else
- In this example, DC component is sum of all elements = 64

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

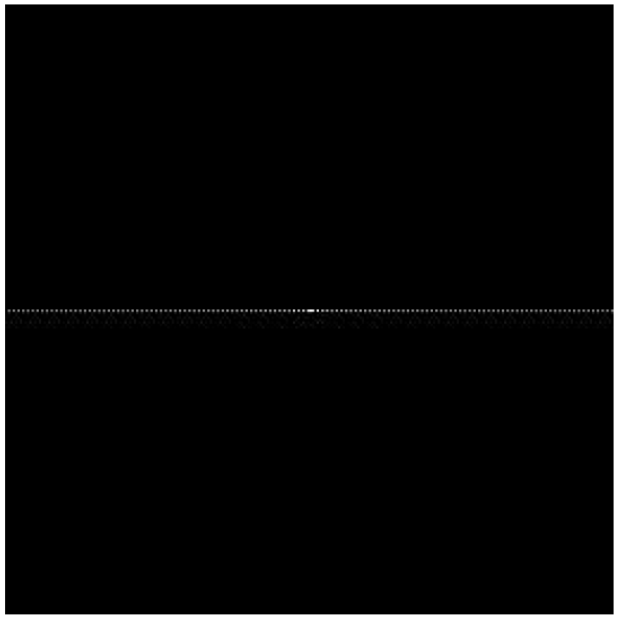
DFT of Image



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

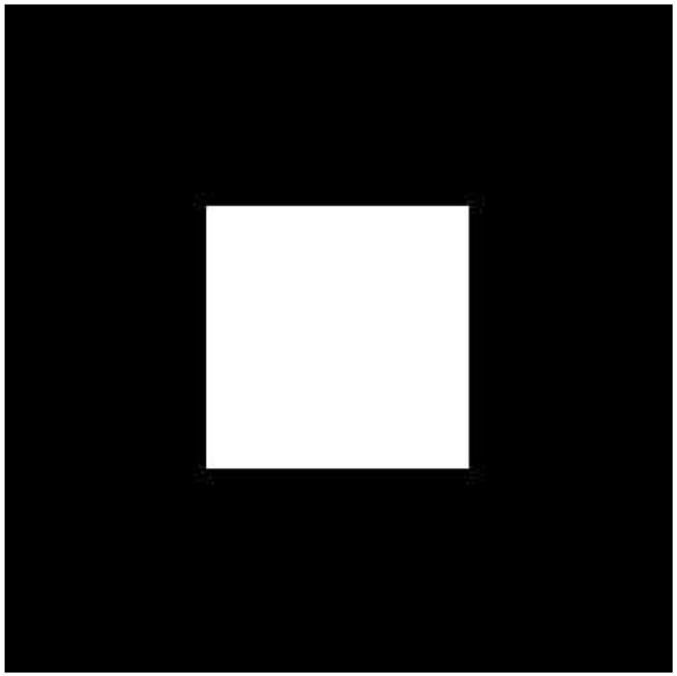


Image

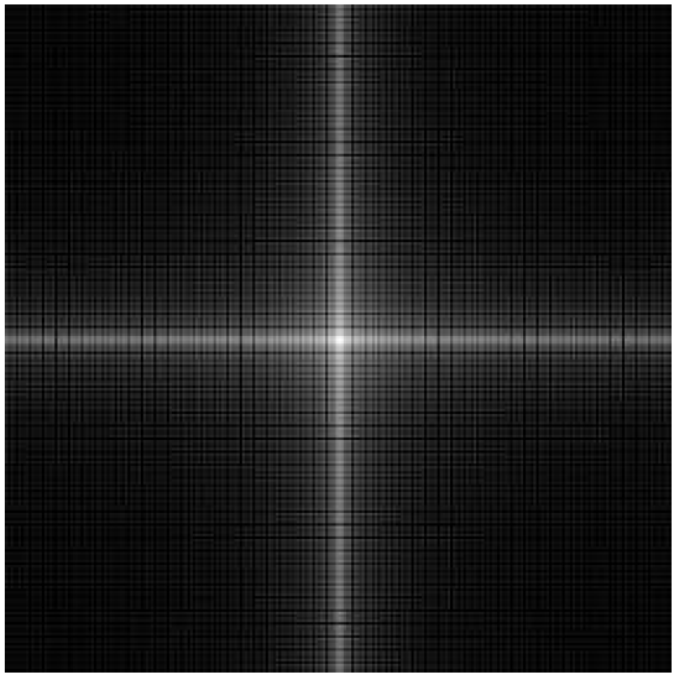


DFT

DFT Example: A Box

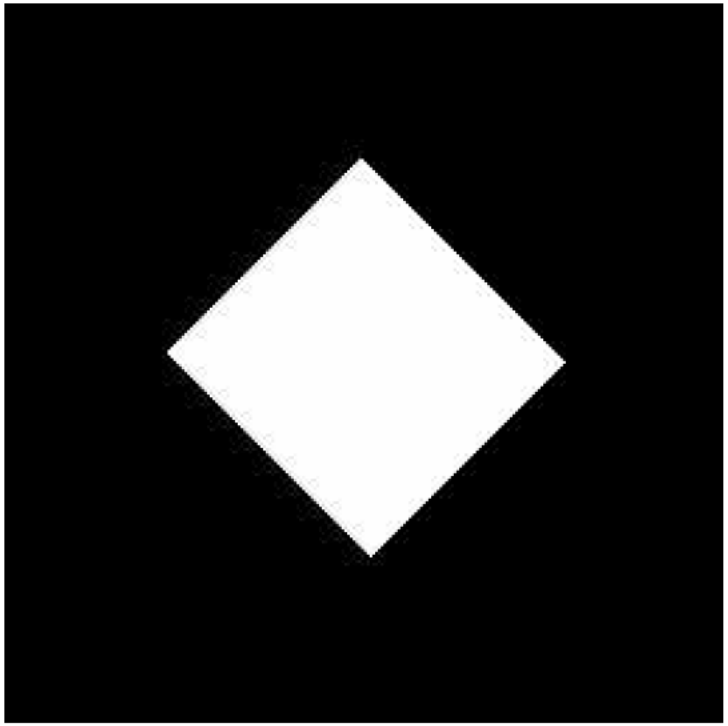


Box

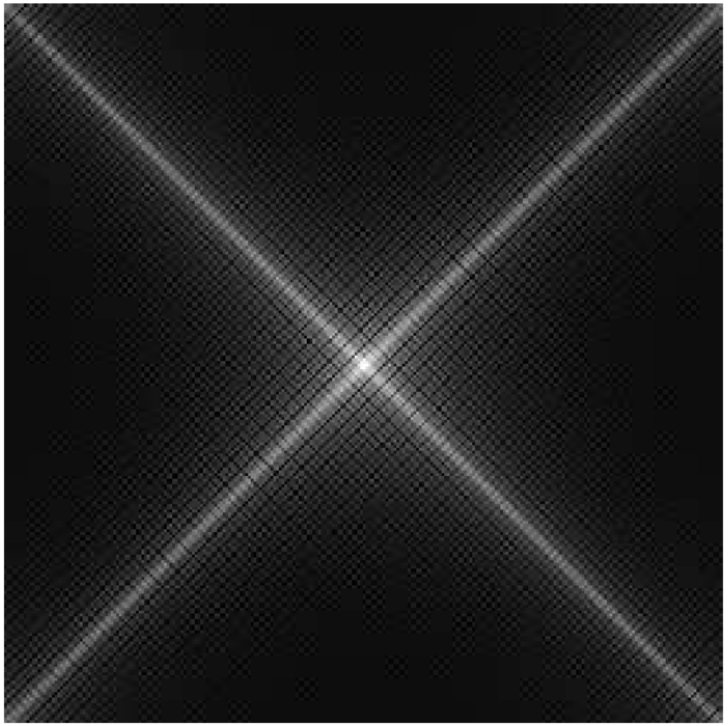


DFT

DFT Example: Rotated Box

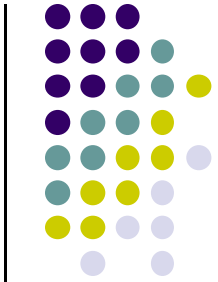


Rotated Box

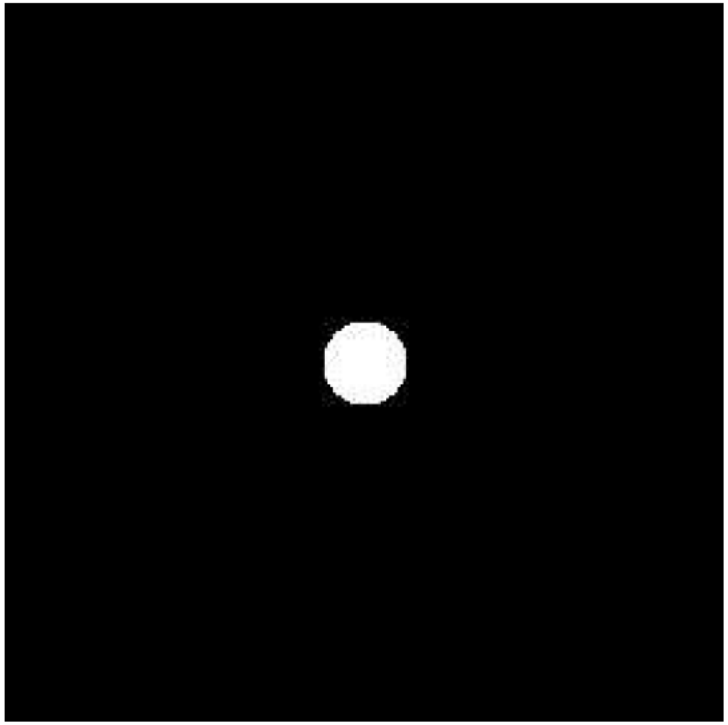


DFT

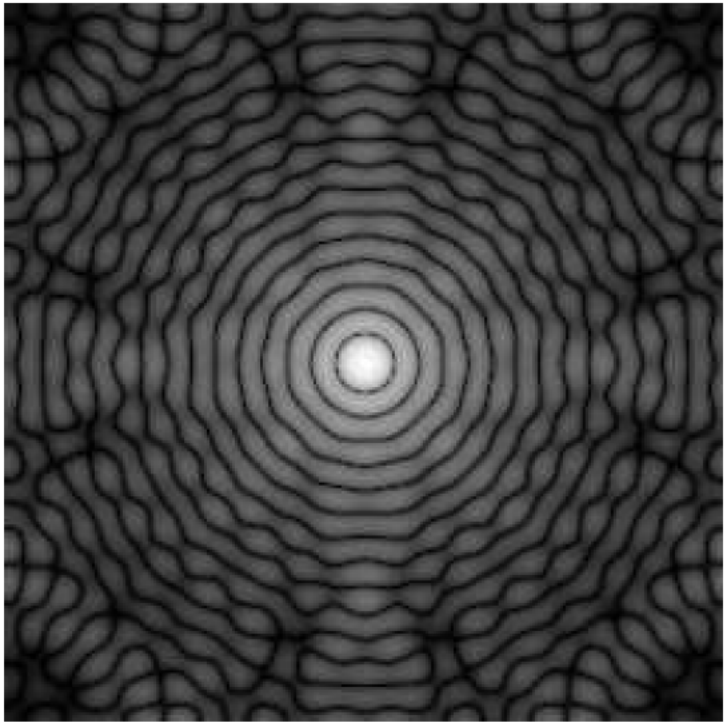
DFT Example: A Circle



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



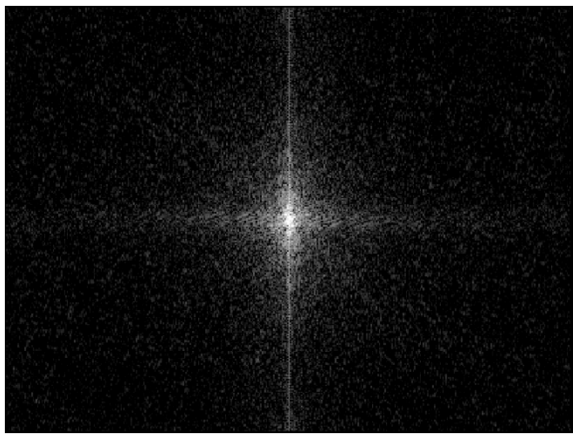
Circle



DFT

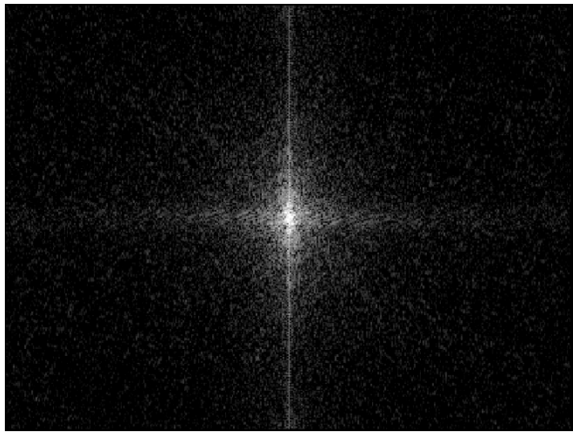
DFT Computation: Repeated Image

- 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

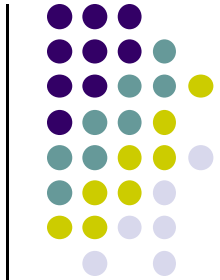
- Can multiply image by windowing function *w(u,v)* before DFT to reduce sharp transitions between ends of repeated images
- Ideally, causes image to drop off towards ends, reducing transitions



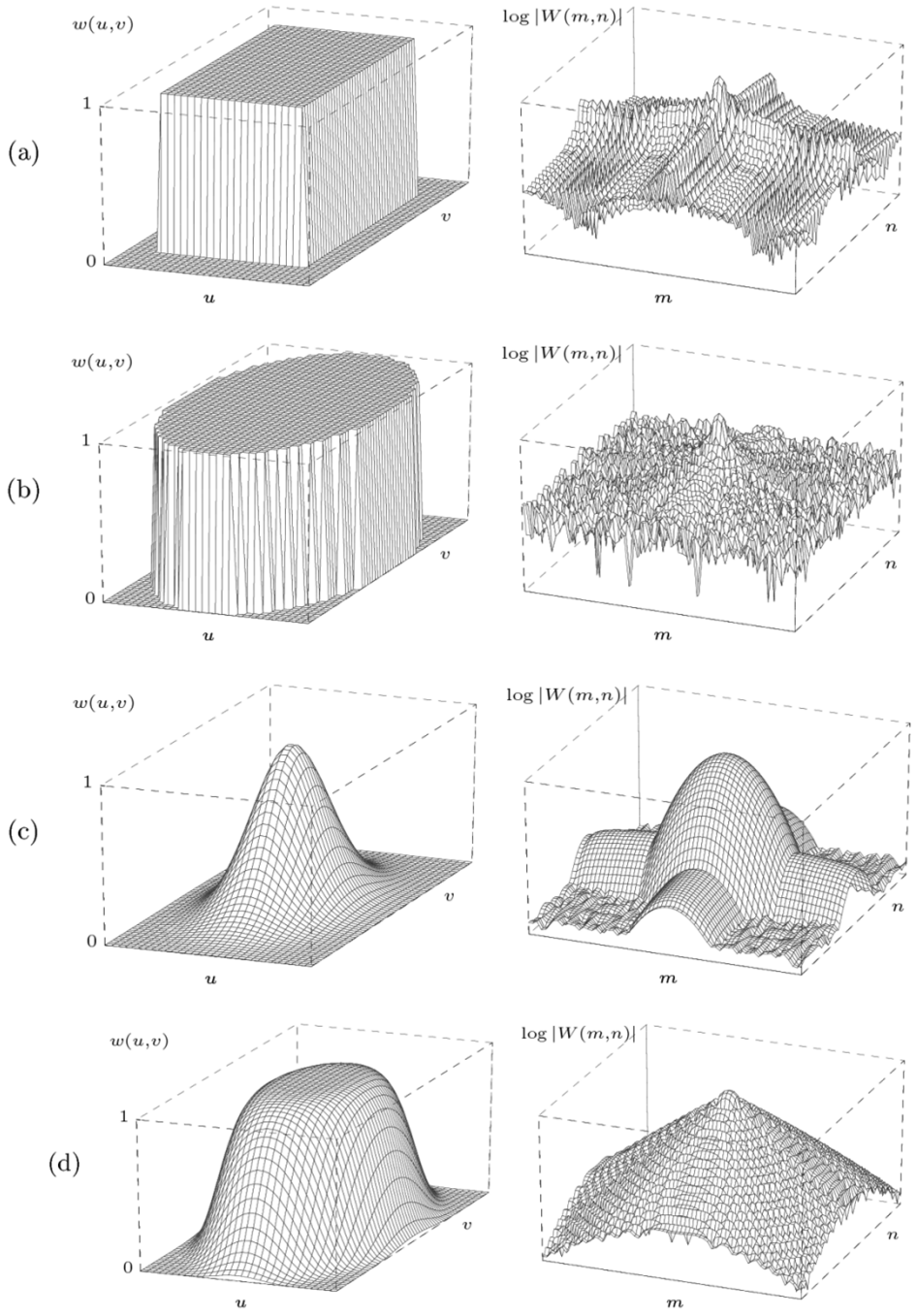
Definitions:

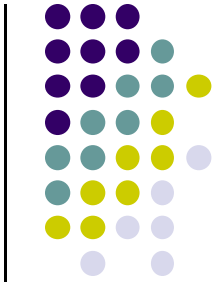
$$r_u = \frac{u - M/2}{M/2} = \frac{2u}{M} - 1 \qquad r_v = \frac{v - N/2}{N/2} = \frac{2v}{N} - 1 \qquad r_{u,v} = \sqrt{r_u^2 + r_v^2}$$

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:	$w(u,v) = \begin{cases} 1 - 6r_{u,v}^2 + 6r_{u,v}^3 & \text{for } 0 \le r_{u,v} < 0.5 \\ 2 \cdot (1 - r_{u,v})^3 & \text{for } 0.5 \le r_{u,v} < 1 \\ 0 & \text{otherwise} \end{cases}$

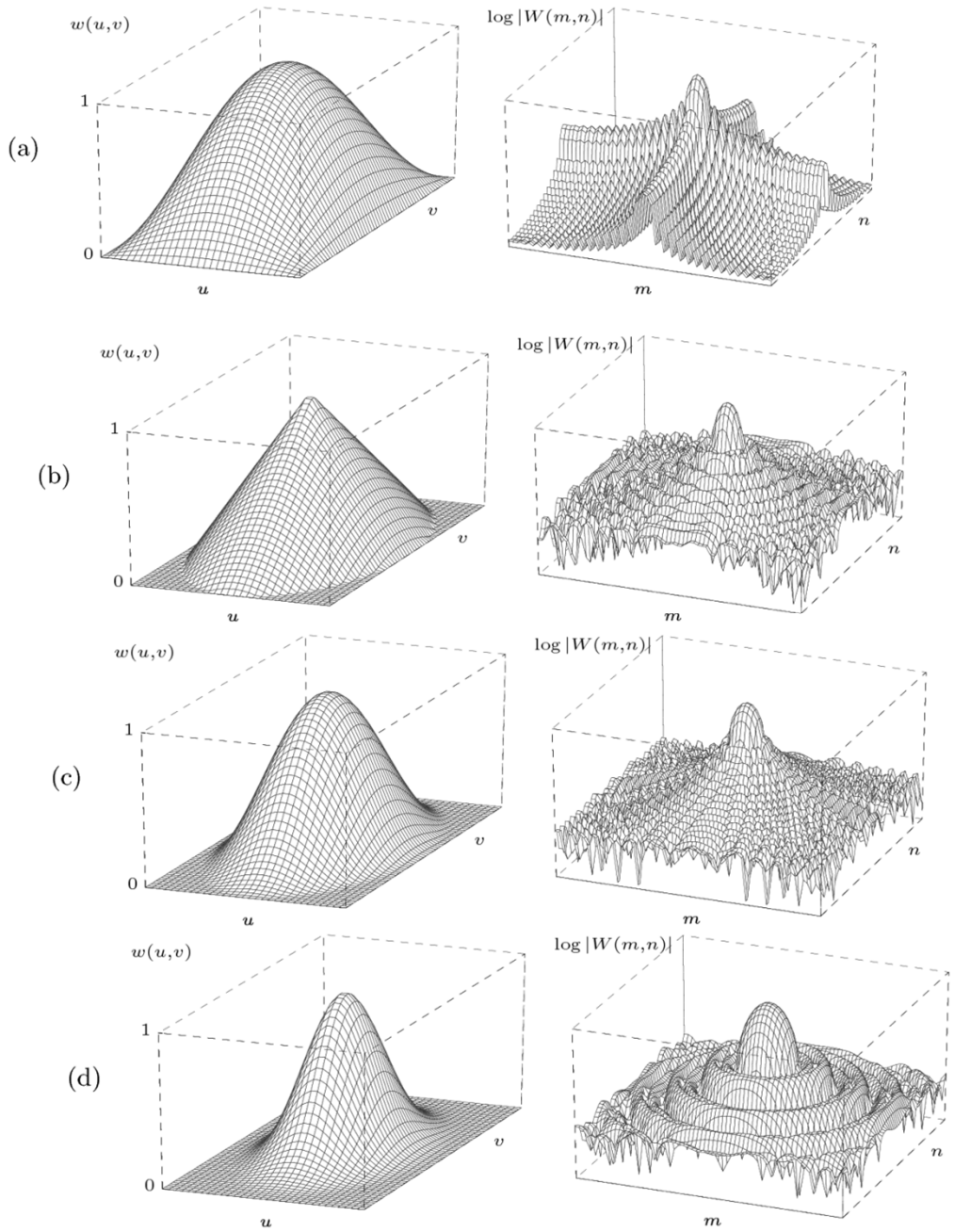


Some Proposed Windowing Functions

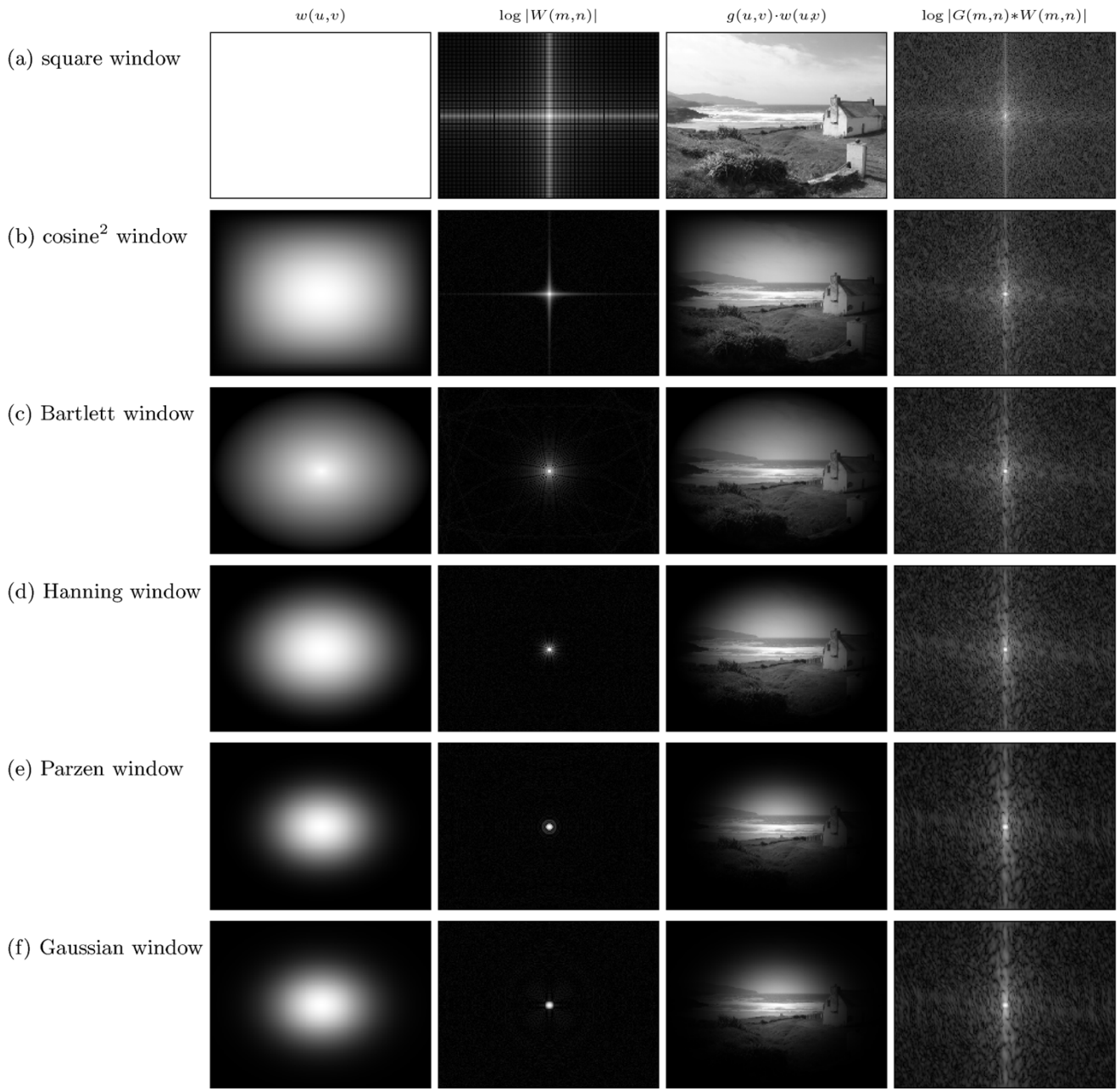




Some Proposed Windowing Functions

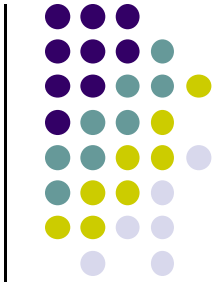


Some Proposed Windowing Functions



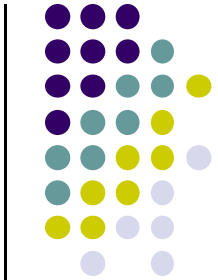
Application of Windowing Functions

Filtering in Frequency Domain

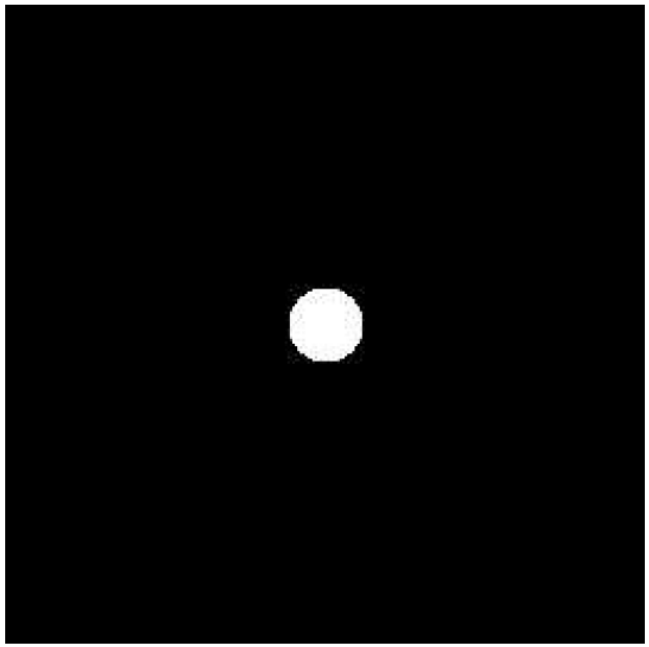


- one reason for using Fourier transform in image processing is due to convolution theorem
- Spatial convolution can be performed by element-wise multiplication of the Fourier transform by suitable “filter matrix”

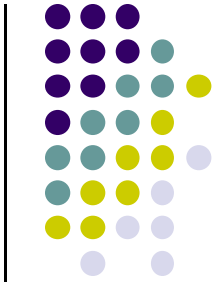
Ideal Low Pass Filtering



- **Low pass filter:** Keep frequencies **below** a certain frequency
- Low pass filtering causes **blurring**
- After DFT, DC components and low frequency components are towards center
- May specify frequency cutoff as circle c

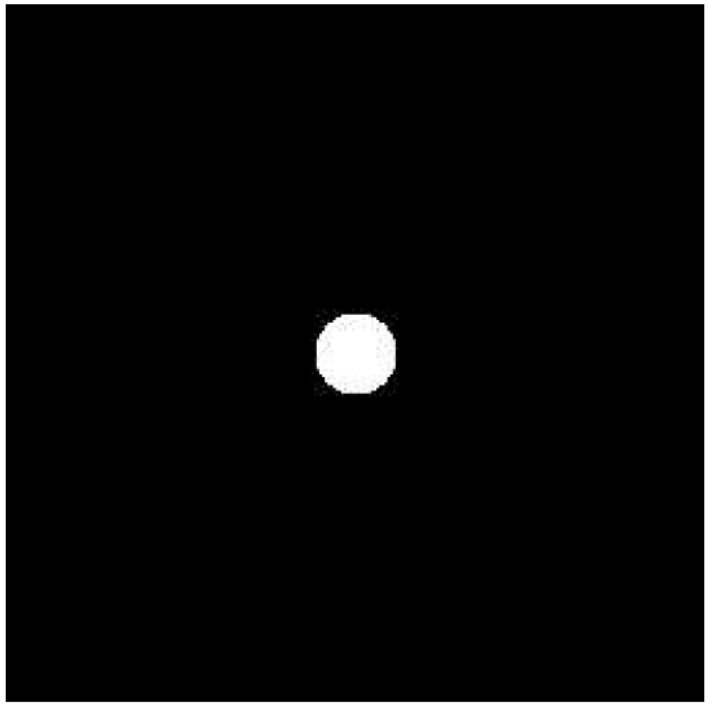


Ideal Low Pass Filtering



- Multiply Image Fourier Transform *F* by some filter matrix *m*

$$m(x,y) = \begin{cases} 1 & \text{if } (x,y) \text{ is closer to the center than some value } D, \\ 0 & \text{if } (x,y) \text{ is further from the center than } D. \end{cases}$$



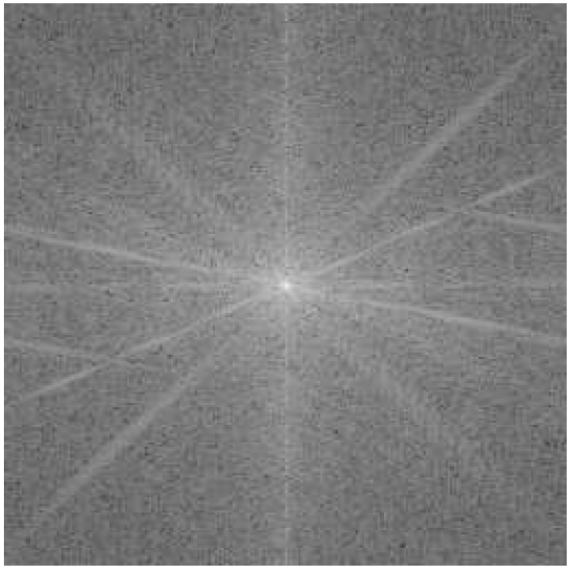
Ideal Low Pass Filtering

- Low pass filtered image is inverse Fourier Transform of product of *F* and *m*

$$\mathcal{F}^{-1}(F \cdot m)$$

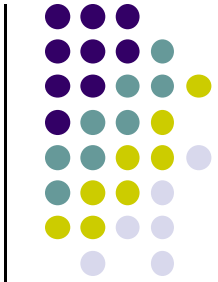
- Example: Consider the following image and its DFT

Image



DFT

Ideal Low Pass Filtering



Applying
low pass filter
to DFT
Cutoff D = 15

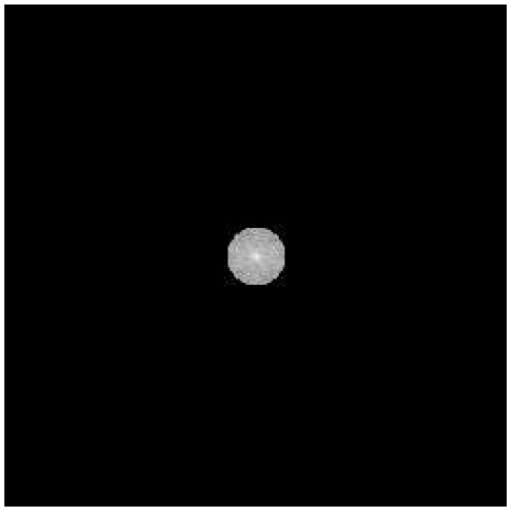
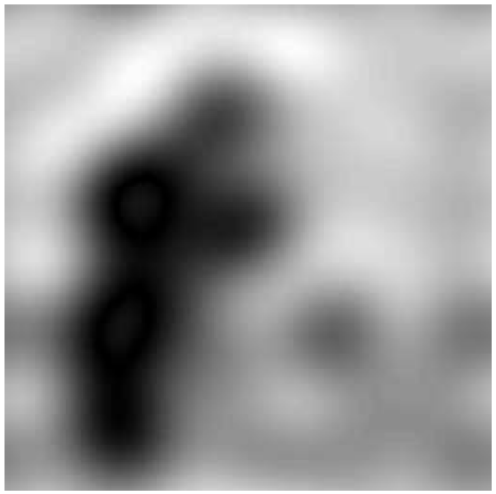


Image after
inversion

low pass filter
Cutoff D = 5

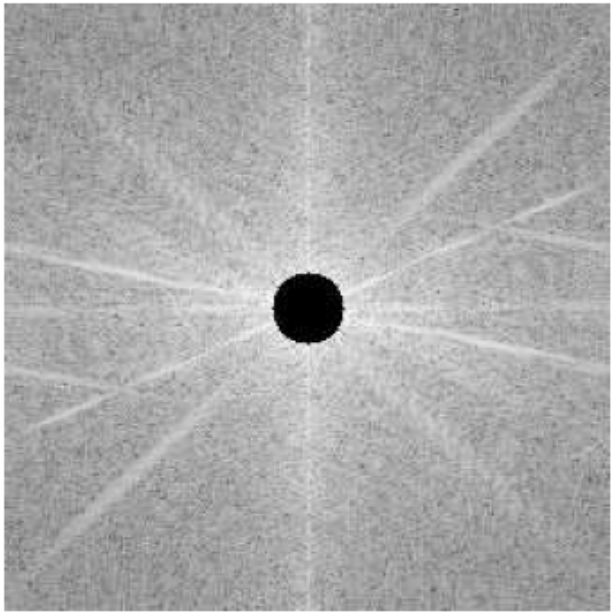


low pass filter
Cutoff D = 30

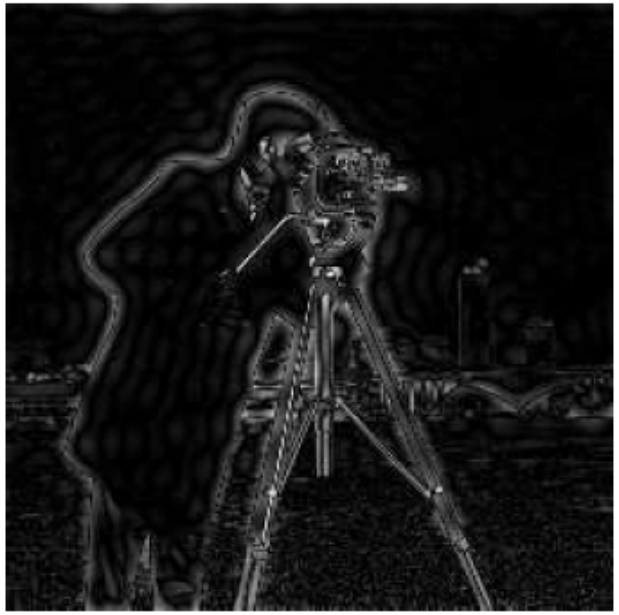
Note: Sharp filter
Cutoff causes
ringing

Ideal High Pass Filtering

- Opposite of low pass filtering: eliminate center (low frequency values), keeping others
- High pass filtering causes image **sharpening**
- If we use circle as cutoff again, size affects results
  - Large cutoff = More information removed

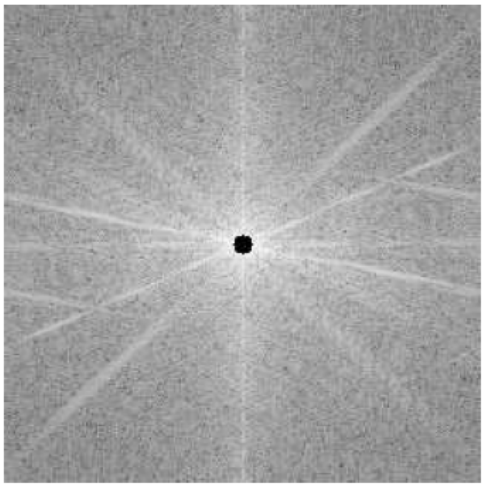


DFT of Image after high pass Filtering



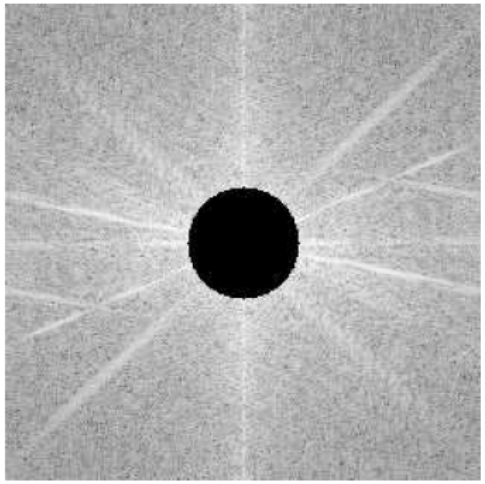
Resulting image after inverse DFT

Ideal High Pass Filtering: Effect of Cutoffs



High pass filtering of DFT with filter Cutoff D = 5

Low cutoff frequency removes Only few lowest frequencies



High pass filtering of DFT with filter Cutoff D = 30

High cutoff frequency removes many frequencies, leaving only edges

Highpass Filtering Example

Original image

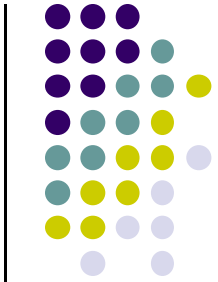
Highpass filtering result

High frequency emphasis result

After histogram equalisation

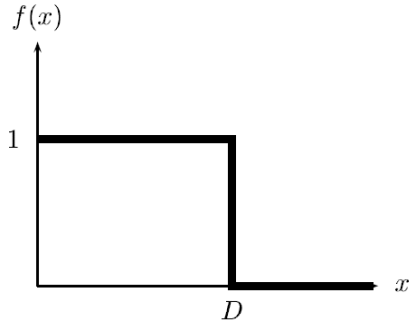
Images taken from Gonzalez & Woods, Digital Image Processing (2002)

Butterworth Filtering



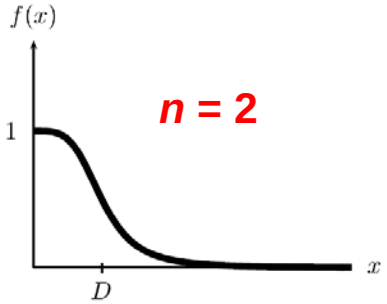
- Sharp cutoff leads to ringing
- To avoid ringing, can use circle with more gentle cutoff slope
- **Butterworth filters** have more gentle cutoff slopes
- Ideal low pass filter

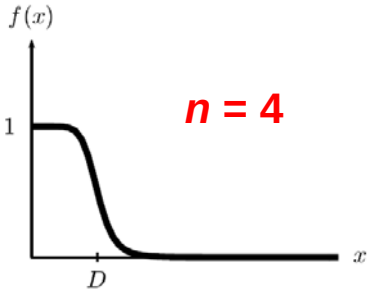
$$f(x) = \begin{cases} 1 & \text{if } x < D \\ 0 & \text{if } x \geq D \end{cases}$$



- Butterworth low pass filter

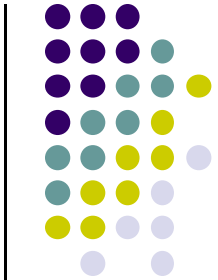
$$f(x) = \frac{1}{1 + (x/D)^{2n}}$$





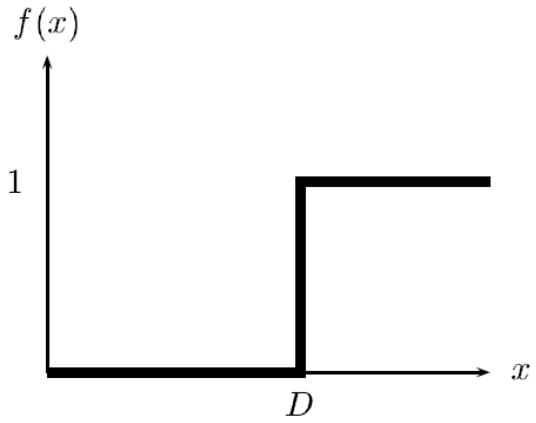
- *n* called **order** of the filter, controls sharpness of cutoff

Butterworth High Pass Filtering



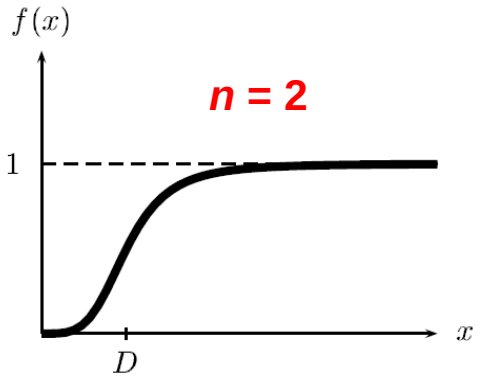
- Ideal high pass filter

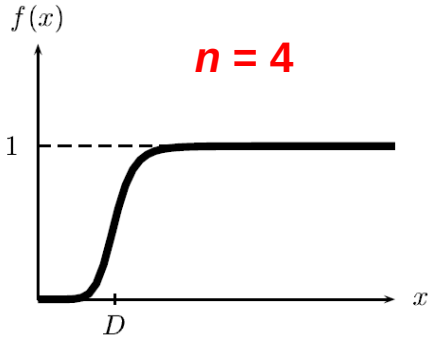
$$f(x) = \begin{cases} 1 & \text{if } x > D \\ 0 & \text{if } x \leq D \end{cases}$$



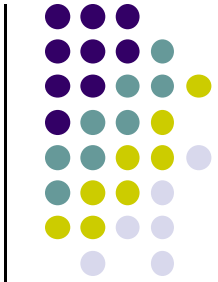
- Butterworth high pass filter

$$f(x) = \frac{1}{1 + (D/x)^{2n}}$$

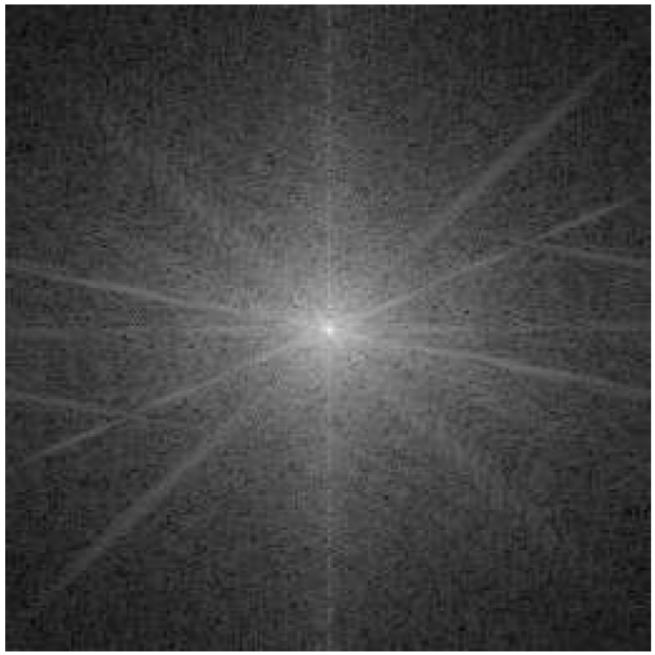




Low Pass Butterworth Filtering



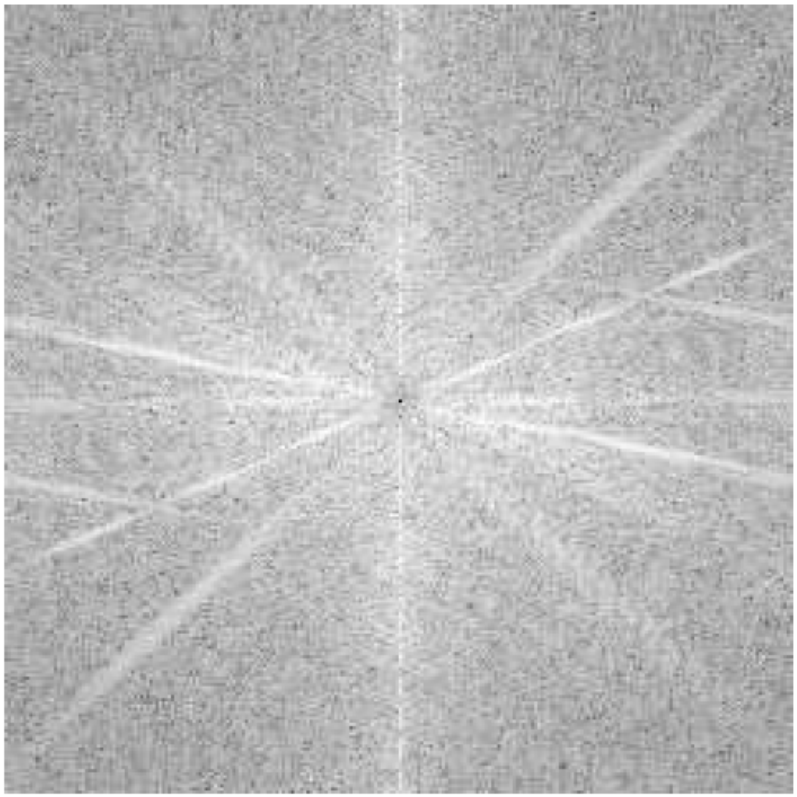
- Low pass filtering removes high frequencies, blurs image
- Gentler cutoff eliminates ringing artifact



DFT of Image after low pass Butterworth filtering

Resulting image after inverse DFT

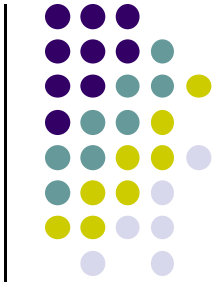
High Pass Butterworth Filtering



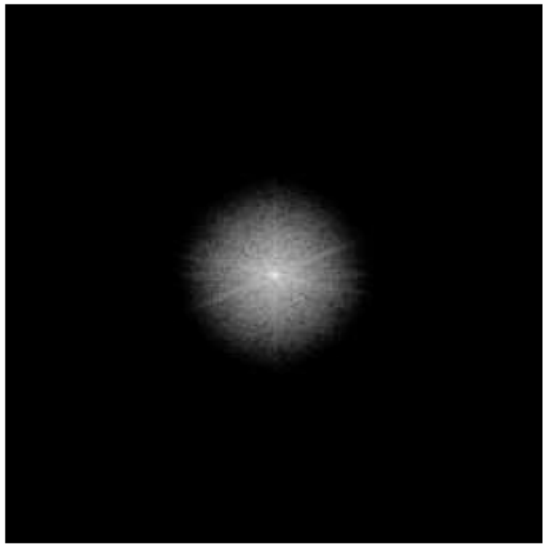
DFT of Image after high pass Butterworth filtering

Resulting image after inverse DFT

Gaussian Filtering

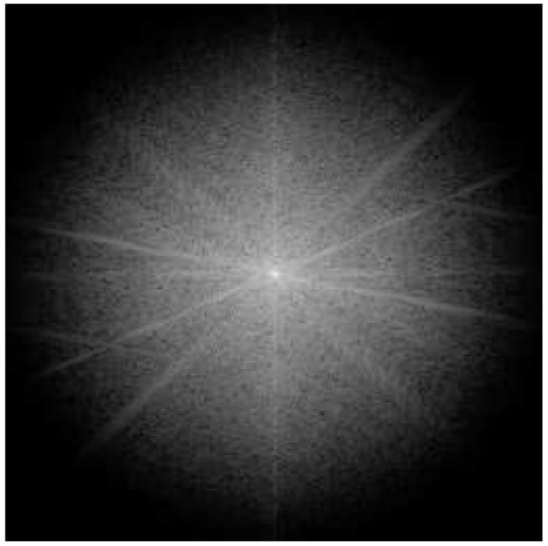


- Gaussian filters can be applied in frequency domain
- Same steps
  - Create gaussian filter
  - Multiply (**DFT of image**) by (**gaussian filter**)
  - Invert result
- **Note:** Fourier transform of gaussian is also a gaussian,
- Just apply gaussian multiply directly (no need to find Fourier transform)



(a) $\sigma = 10$

(b) Resulting image

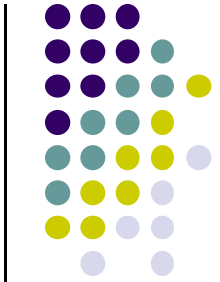


(c) $\sigma = 30$

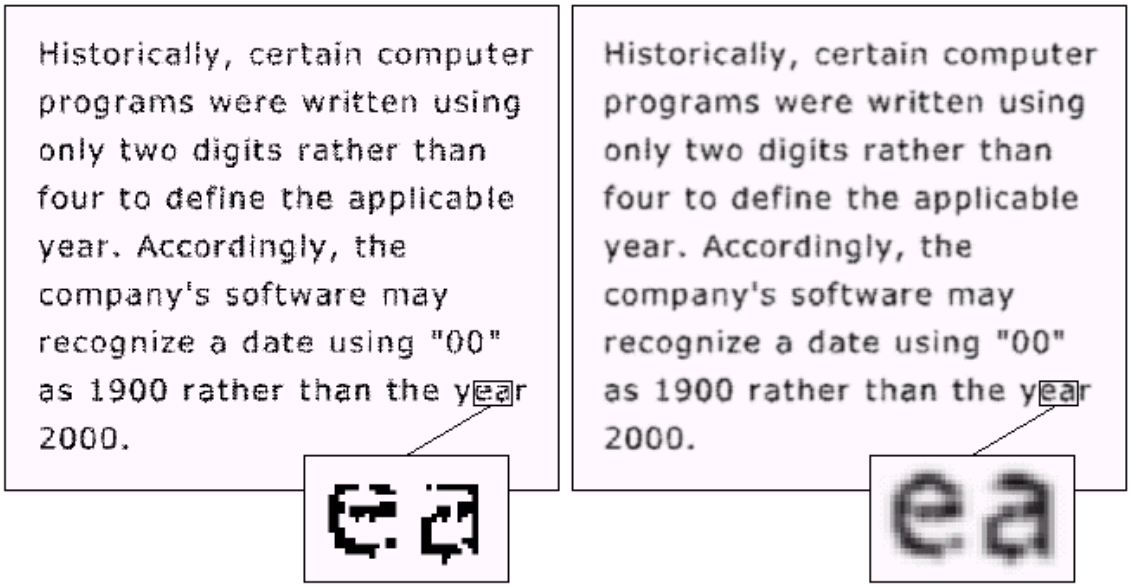
(d) Resulting image

Frequency Domain Low Pass Gaussian Filtering

Lowpass Filtering Examples

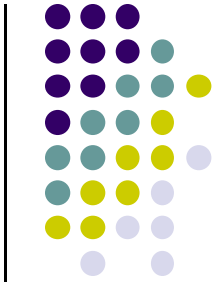


A low pass Gaussian filter can be used to connect broken text



Images taken from Gonzalez & Woods, Digital Image Processing (2002)

Lowpass Filtering Examples



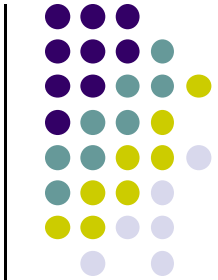
Different lowpass Gaussian filters used to remove blemishes in a photograph

Frequency Domain High Pass Gaussian Filtering

(a) Using $\sigma = 10$

(b) Using $\sigma = 30$

Frequency Domain
Removal of Periodic Noise



- **Recall:** periodic noise could not be removed in spatial domain
- Periodic noise can be removed in frequency domain
- Periodic noise shows up as spikes away from origin in DFT
- Higher frequency noise = further away from origin

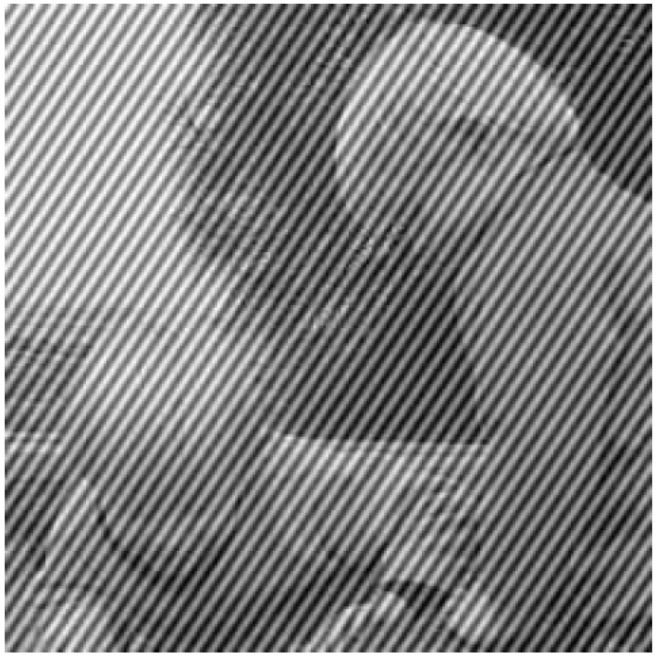
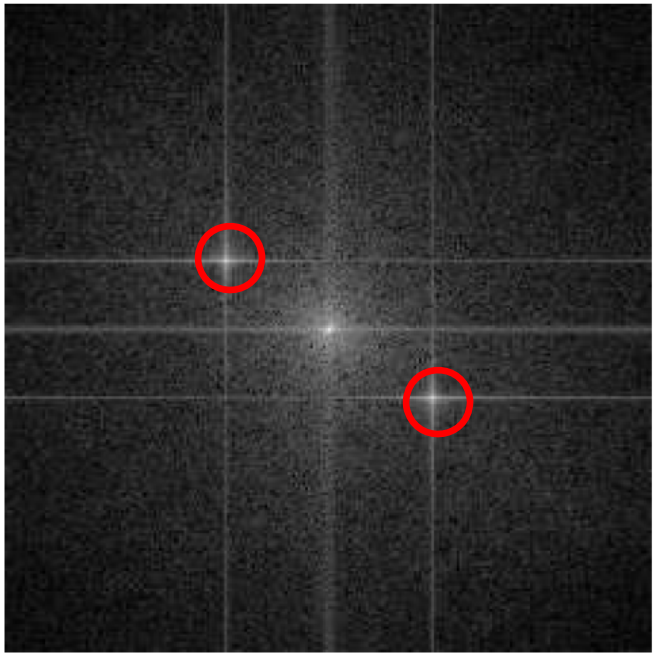
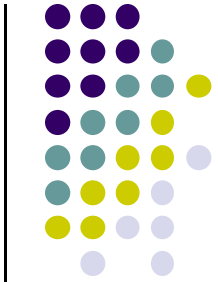


Image with periodic Noise

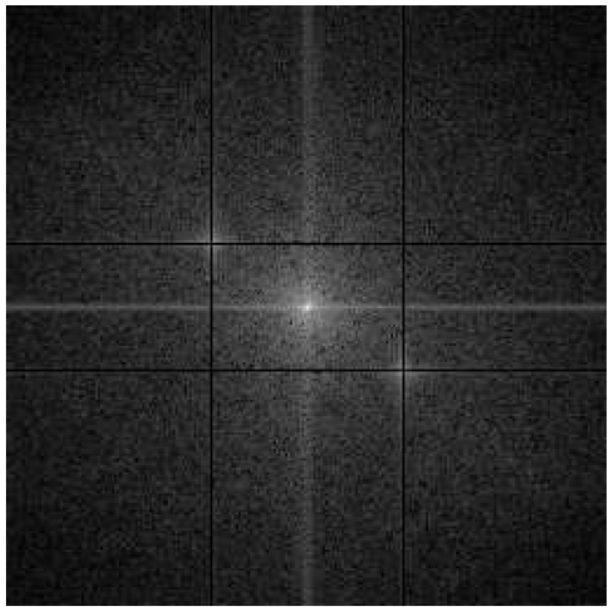


DFT of Image

Frequency Domain
Removal of Periodic Noise



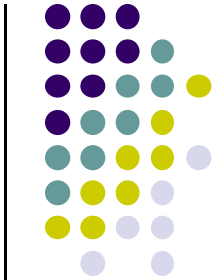
- 2 ways to remove periodic noise in frequency domain
  - Notch Filter
  - Band reject filter
- Notch filter: Set rows, columns of DFT corresponding to noise = 0
- Removes much of the periodic noise



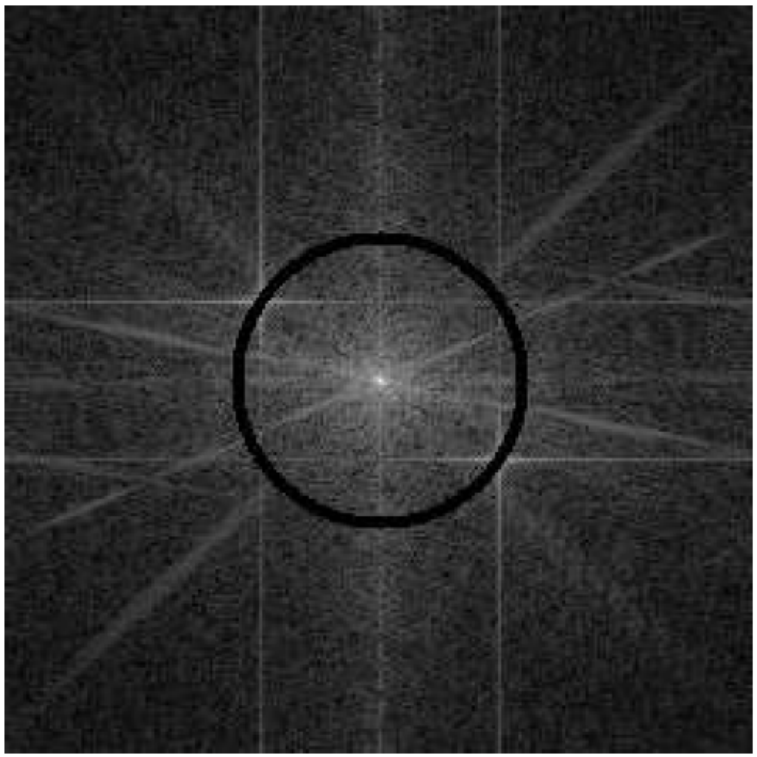
Notch Filter

Result after notch filter applied then inverted

Frequency Domain Removal of Periodic Noise: Band Reject Filter



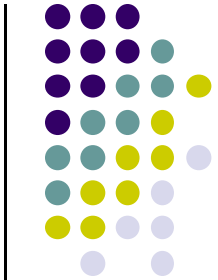
- Create filter with 0's at radius of noise from center, 1 elsewhere
- Apply filter to DFT



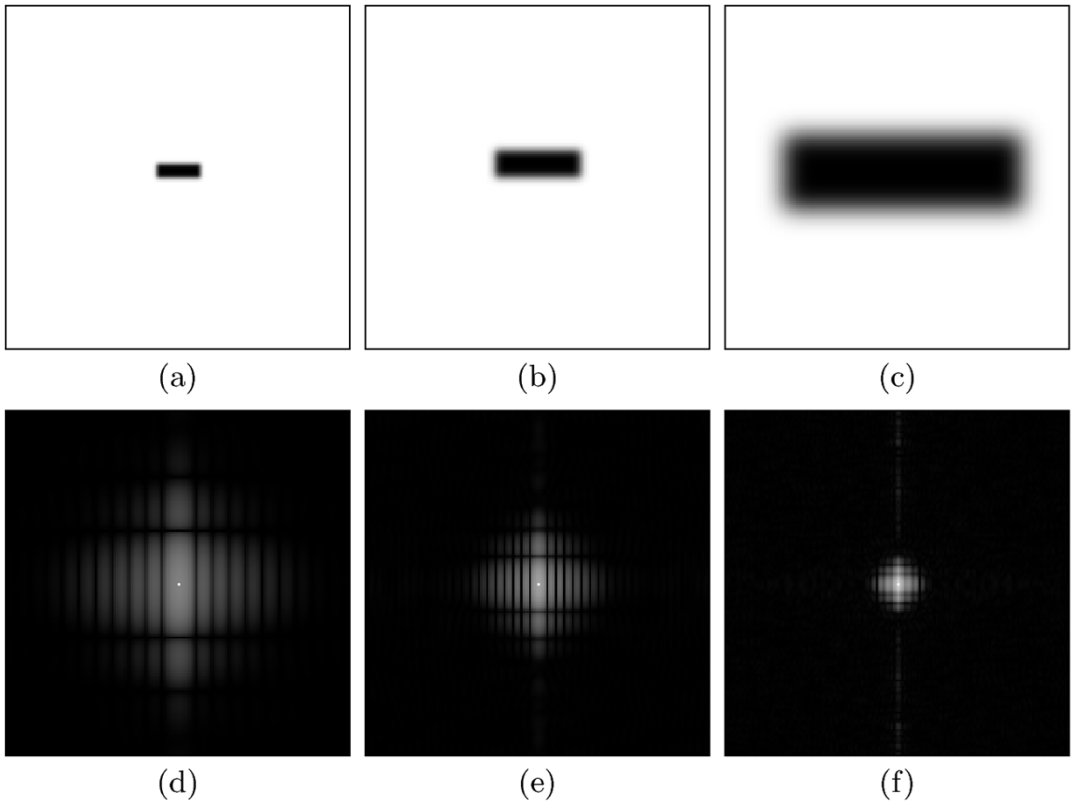
Band Reject Filter

Result after band reject filter applied then inverted

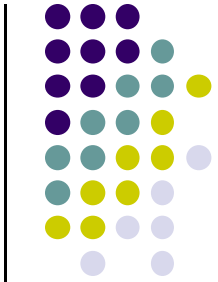
2D Fourier Transform Examples: Scaling



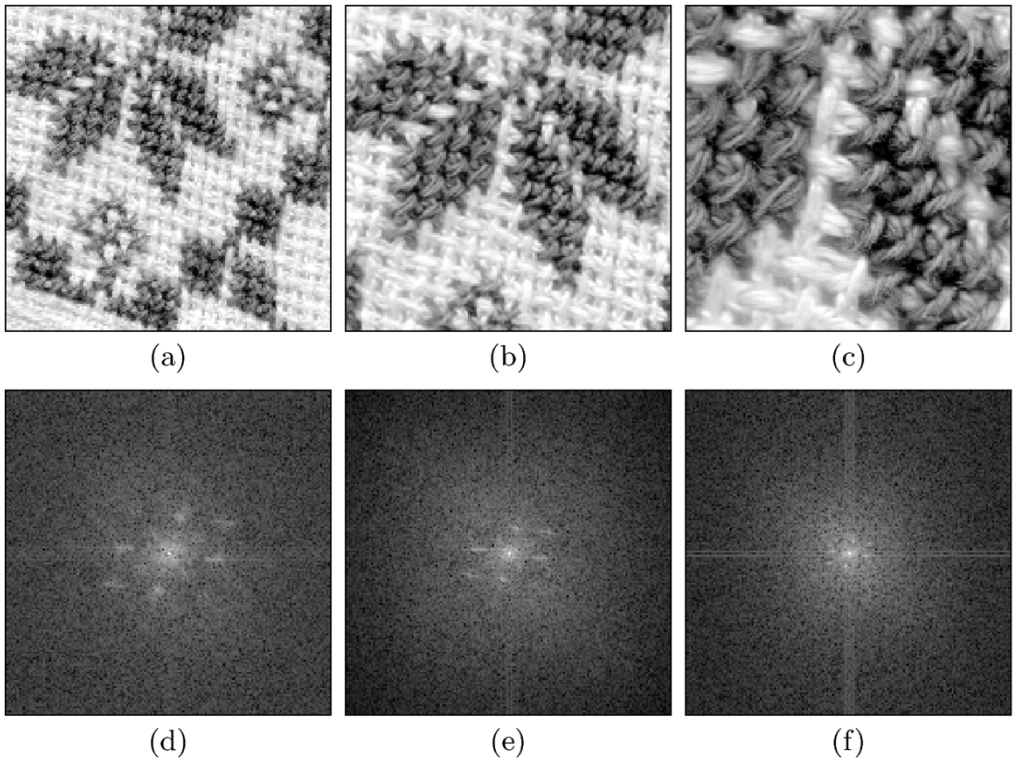
- Stretching image => Spectrum contracts
- And vice versa



2D Fourier Transform Examples: Periodic Patterns



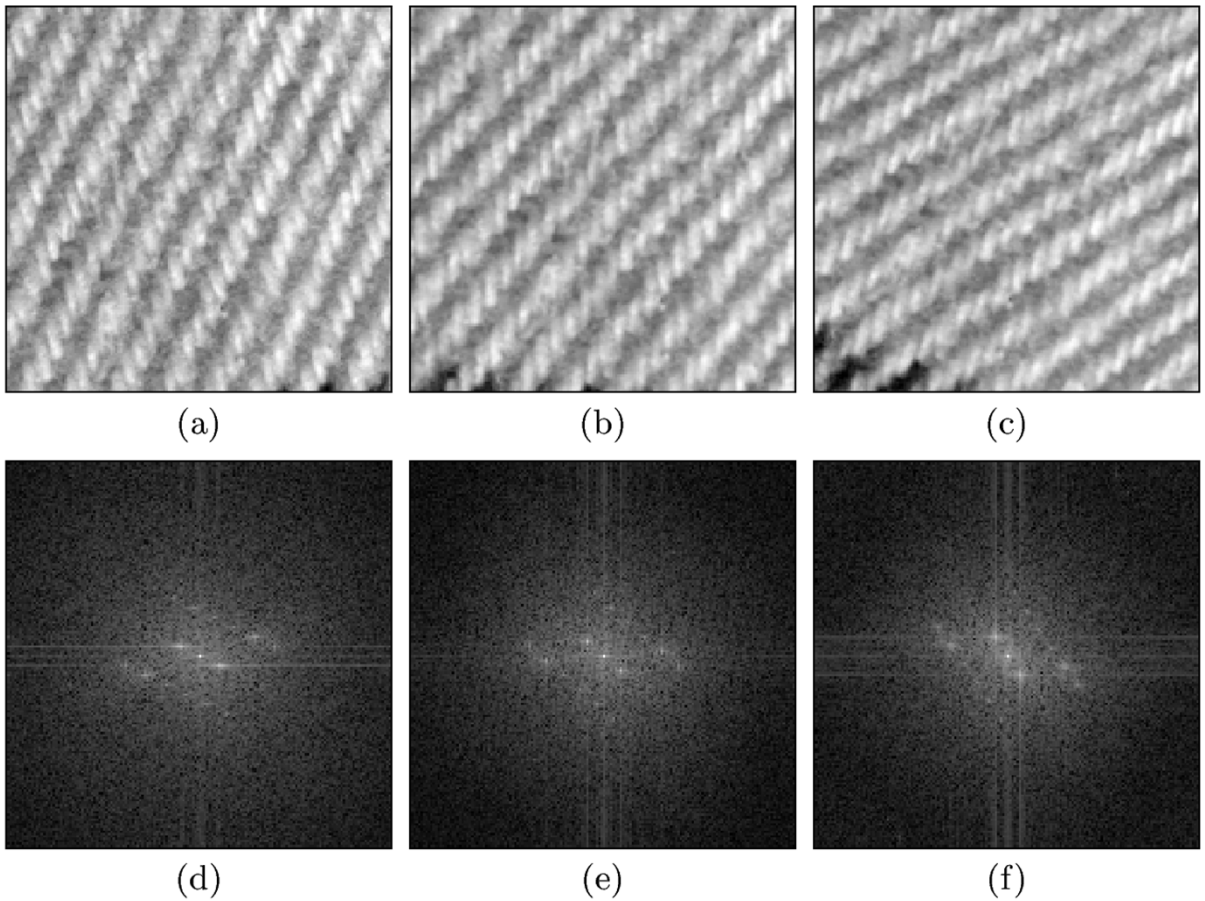
- Repetitive periodic patterns appear as distinct peaks at corresponding positions in spectrum



Enlarging image (c) causes Spectrum to contract (f)

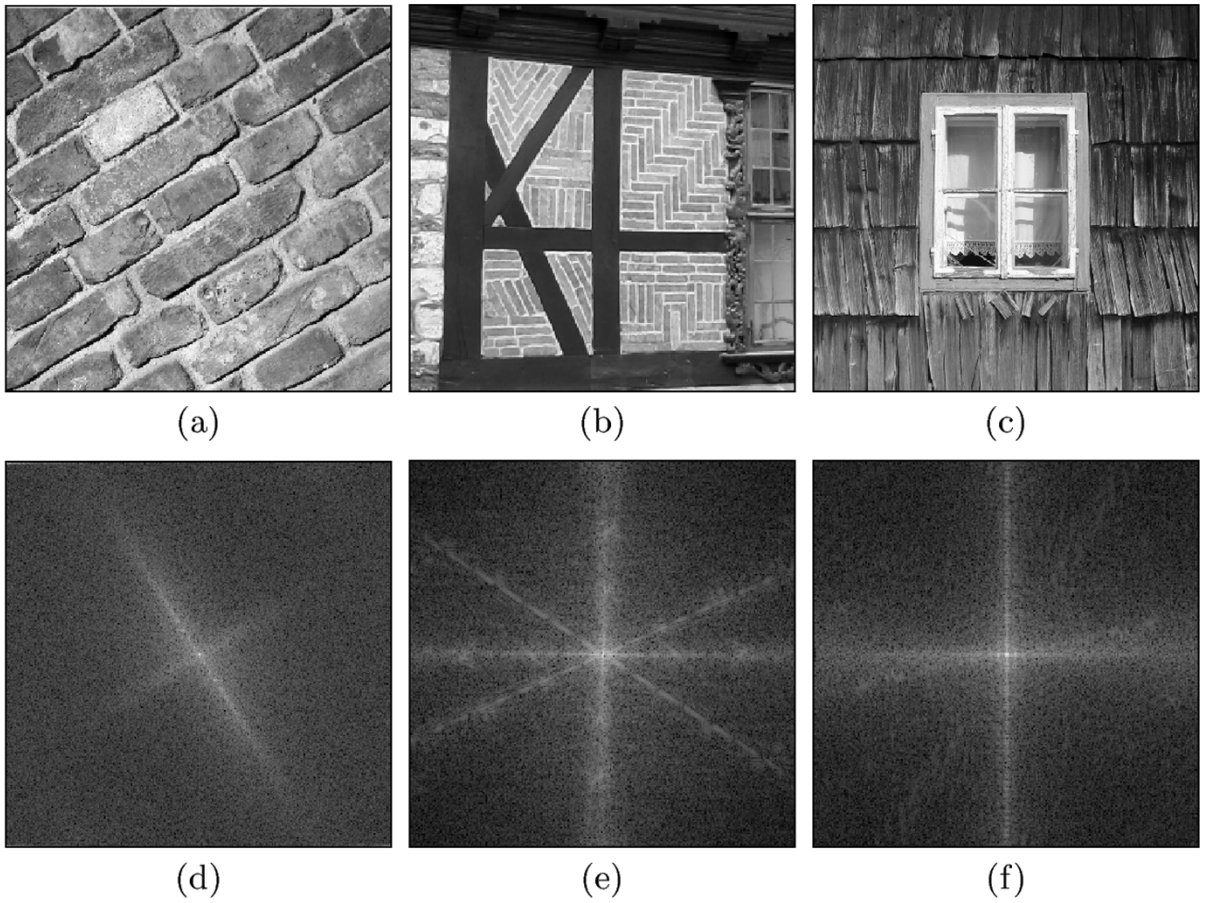
2D Fourier Transform Examples: Rotation

- Rotating image => Rotates spectra by same angle/amount

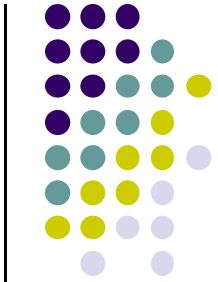


2D Fourier Transform Examples: Oriented, elongated Structures

- Man-made elongated regular patterns in image => appear dominant in spectrum

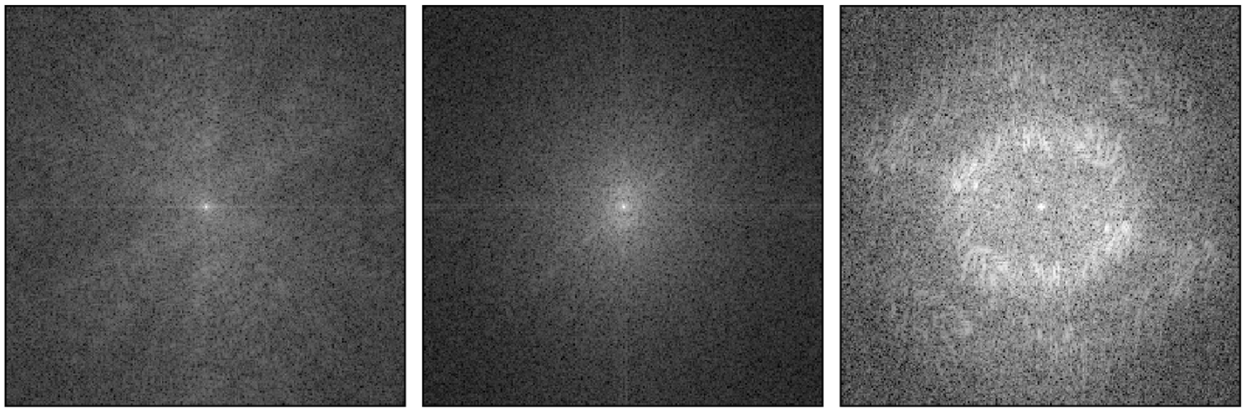


2D Fourier Transform Examples: Natural Images



- Repetitions in natural scenes => less dominant than man-made ones, less obvious in spectra

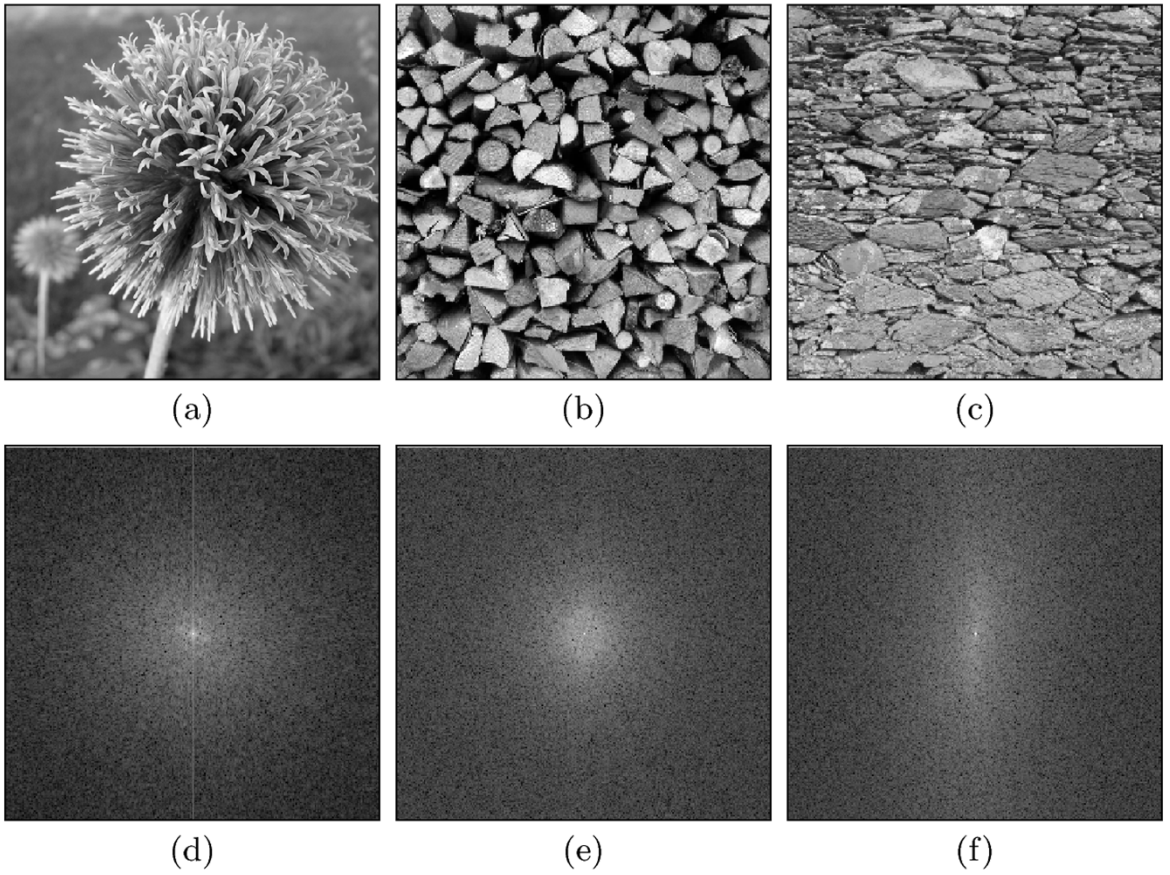
(a) (b) (c)



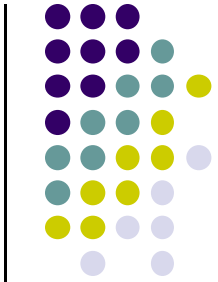
(d) (e) (f)

2D Fourier Transform Examples: Natural Images

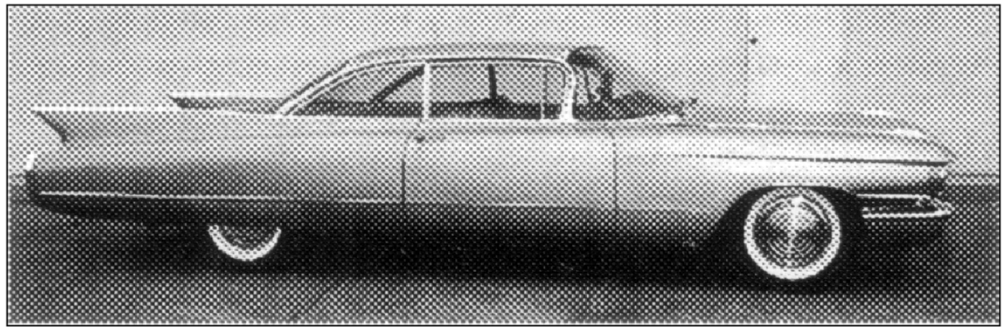
- Natural scenes with repetitive patterns but no dominant orientation => do not stand out in spectra



2D Fourier Transform Examples: Printed Patterns

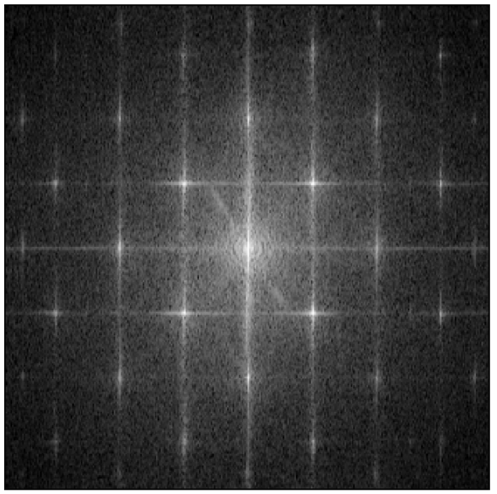


- Regular diagonal patterns caused by printing => Clearly visible/removable in frequency spectrum.



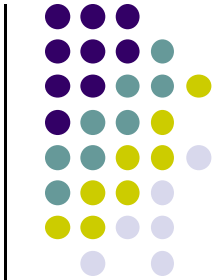
(a)

(b)



(c)

References



- Alasdair McAndrews, Introduction to Digital Image Processing using MATLAB
- Wilhelm Burger and Mark J. Burge, Digital Image Processing, Springer, 2008
- University of Utah, CS 4640: Image Processing Basics, Spring 2012
- Rutgers University, CS 334, Introduction to Imaging and Multimedia, Fall 2012
- Gonzales and Woods, Digital Image Processing (3rd edition), Prentice Hall