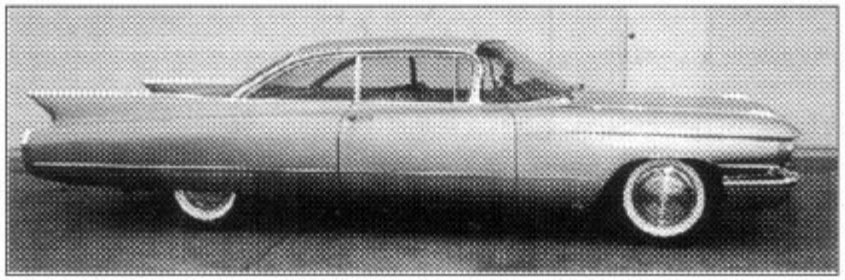
- If earthquake vibrations can be separated into "ingredients" (vibrations of different speeds & amplitudes), buildings can be designed to avoid interacting with the strongest ones.
- If sound waves can be separated into ingredients (bass and treble frequencies), we can boost the parts we care about, and hide the ones we don't. The crackle of random noise can be removed. Maybe similar "sound recipes" can be compared (music recognition services compare recipes, not the raw audio clips).
- If computer data can be represented with oscillating patterns, perhaps the least-important ones can be ignored. This "lossy compression" can drastically shrink file sizes (and why JPEG and MP3 files are much smaller than raw .bmp or .wav files).
- If a radio wave is our signal, we can use filters to listen to a particular channel. In the smoothie world, imagine each person paid attention to a different ingredient: Adam looks for apples, Bob looks for bananas, and Charlie gets cauliflower (sorry bud).

Joseph Fourier

Fourier, Joseph (1822). Théorie analytique de la chaleur. Paris: Firmin Didot Père et Fils.

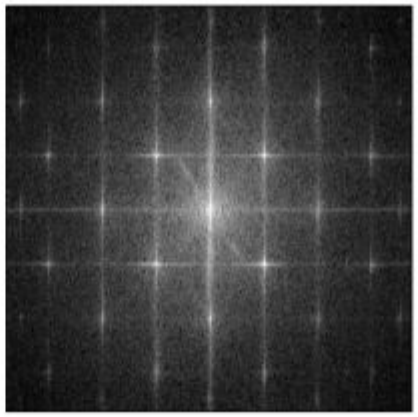
2D Fourier Transform Examples: Printed Patterns

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



(a)

(b)



(c)

Continuous Fourier Transform (FT)

- Transforms a signal (i.e., function) from the **spatial** (x) domain to the **frequency** (u) d

Forward FT: $F(f(x)) = F(u) = \int_{-\infty}^{\infty} f(x)e^{-j2\pi ux}\,dx$

Inverse FT: $F^{-1}(F(u)) = f(x) = \int_{-\infty}^{\infty} F(u)e^{j2\pi ux}\,du$

Extending FT in 2D

- Forward FT

$$F(f(x,y)) = F(u,v) = \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} f(x,y)e^{-j2\pi(ux+vy)}dxdy$$

- Inverse FT

$$F^{-1}(F(u,v)) = f(x,y) = \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} F(u,v)e^{j2\pi(ux+uy)}dudv$$

Discrete Fourier Transform (DFT) (cont'd)

- Forward DFT

$$F(u) = \frac{1}{N} \sum_{x=0}^{N-1} f(x) e^{\frac{-j2\pi ux}{N}}, u = 0, 1, \ldots, N-1$$

- Inverse DFT

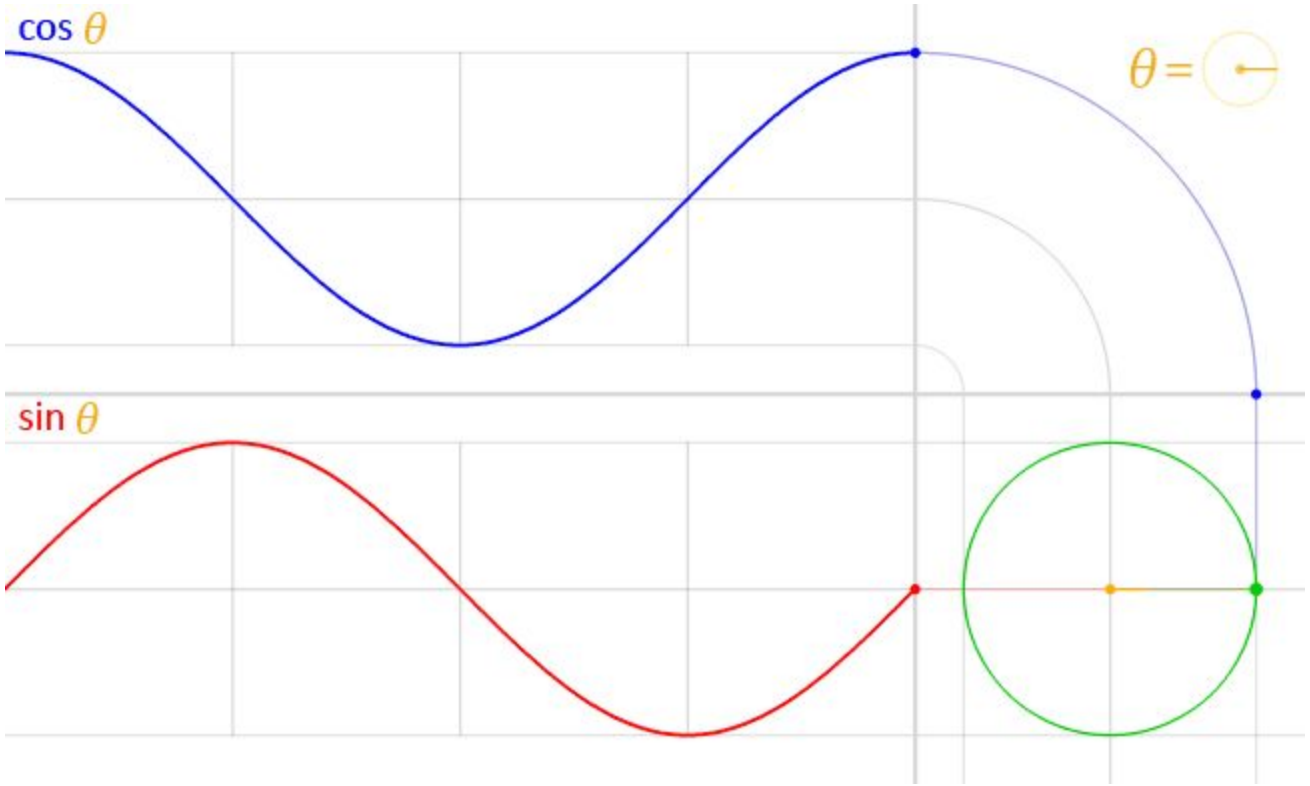
$$f(x) = \sum_{u=0}^{N-1} F(u) e^{\frac{j2\pi ux}{N}}, x = 0, 1, \ldots, N-1$$

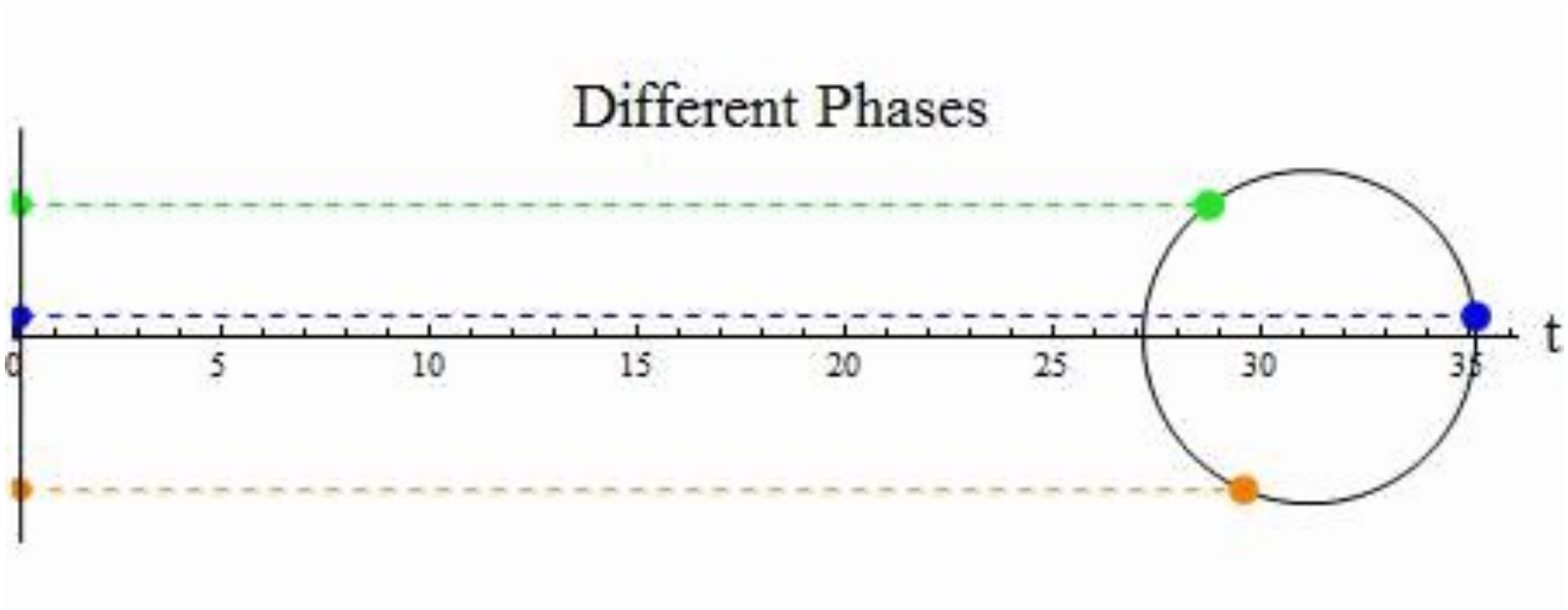
$F(u)$ is discrete: $F(u) = F(u\Delta u), u = 0, 1, \ldots, N-1, \Delta u =$ 1/(NΔx)

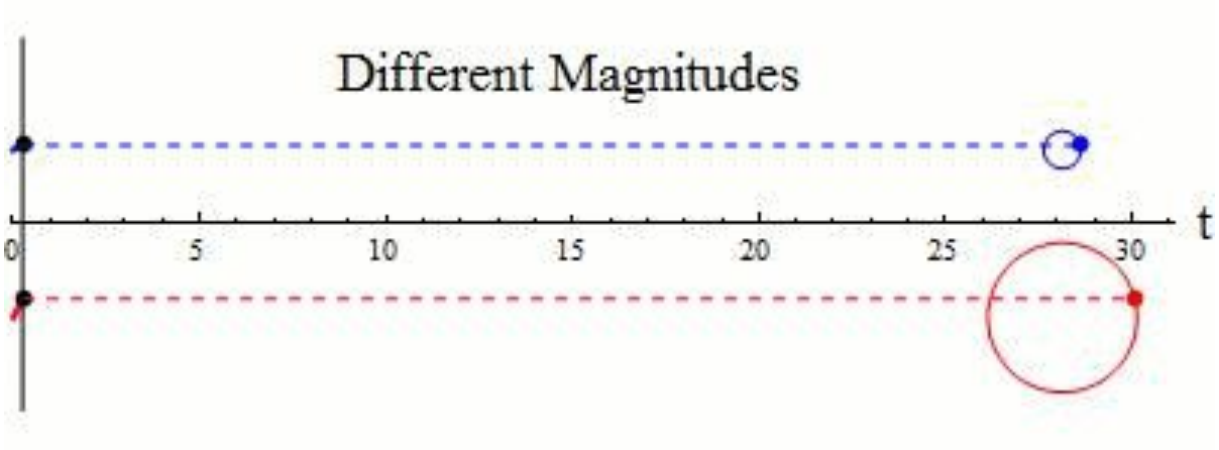
- For *M x N* matrix, forward and inverse fourier transforms can be written

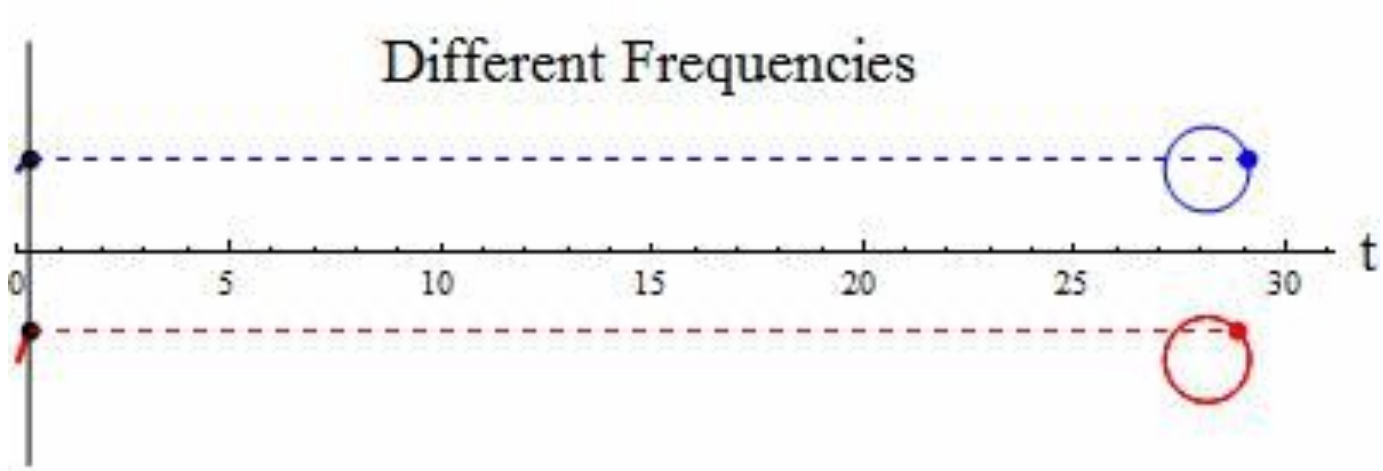
$$F(u,v) = \sum_{x=0}^{M-1}\sum_{y=0}^{N-1} f(x,y)\exp\left[-2\pi i\left(\frac{xu}{M}+\frac{yv}{N}\right)\right].$$

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


cos θ
θ =
sin θ

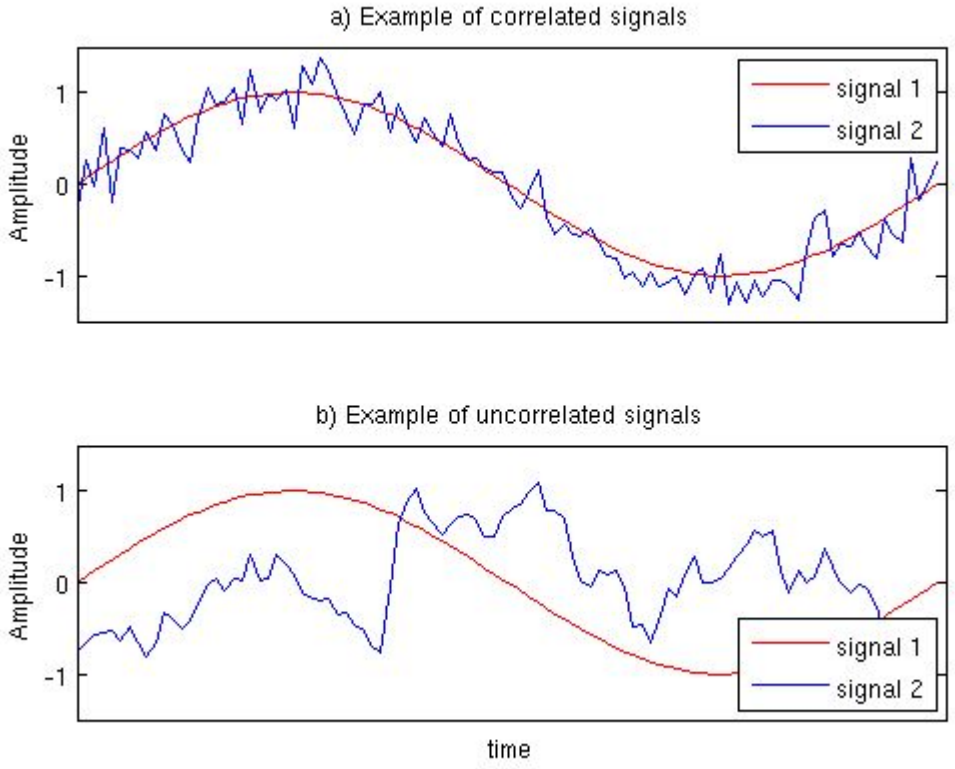

Different Phases
0
5
10
15
20
25
30
35
t


Different Magnitudes
0
5
10
15
20
25
30
t


Different Frequencies
t
0
5
10
15
20
25
30

$$\sum_{i=0}^{N} x(i)y(i)$$ Computes the correlation between x and y

If the correlation is high, then the signals are similar, if the correlation is near zero the signals are not very similar.



$$X(k) = \sum_{n=0}^{N-1} x(n) e^{-i 2\pi k n / N}$$

$$e^{-i\theta} = \cos\theta - i\sin\theta$$

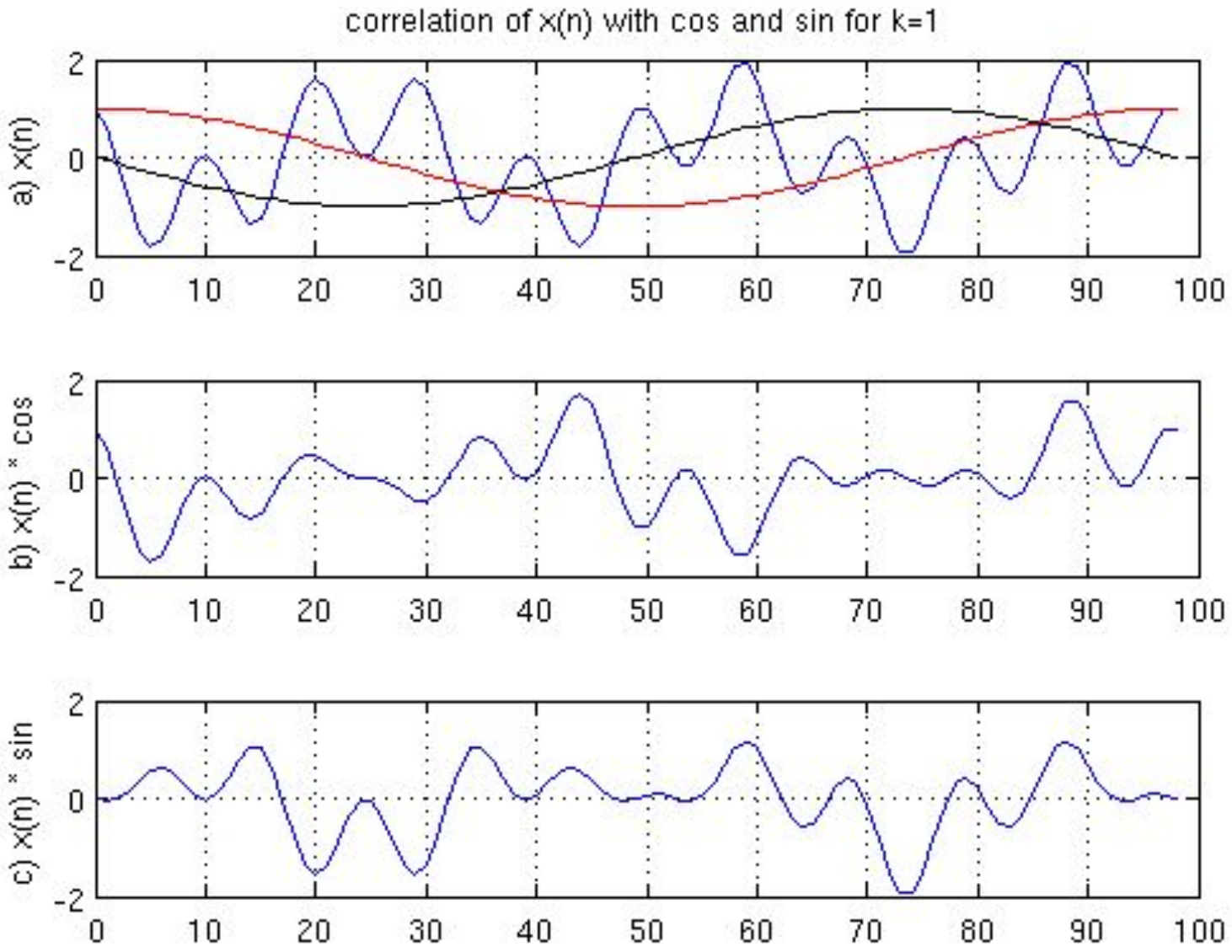
$$\theta = 2\pi k n / N$$

$$X(k) = \sum_{n=0}^{N-1} x(n) \left(\cos\left(2\pi k n / N\right) - i \sin\left(2\pi k n / N\right)\right)$$

$$X(k) = \boxed{\sum_{n=0}^{N-1} x(n) \cos\left(\frac{2\pi k n}{N}\right)} - i \boxed{\sum_{n=0}^{N-1} x(n) \sin\left(\frac{2\pi k n}{N}\right)}$$

$$f = \frac{k \times \mathrm{fs}}{N}$$

To convert these values of k into actual frequencies (measured in Hertz), we need the sampling rate of our original signal (fs) and the following formula:


correlation of x(n) with cos and sin for k=1
a) x(n)
b) x(n) * cos
c) x(n) * sin
2
0
-2
0
10
20
30
40
50
60
70
80
90
100

Separability

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

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

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

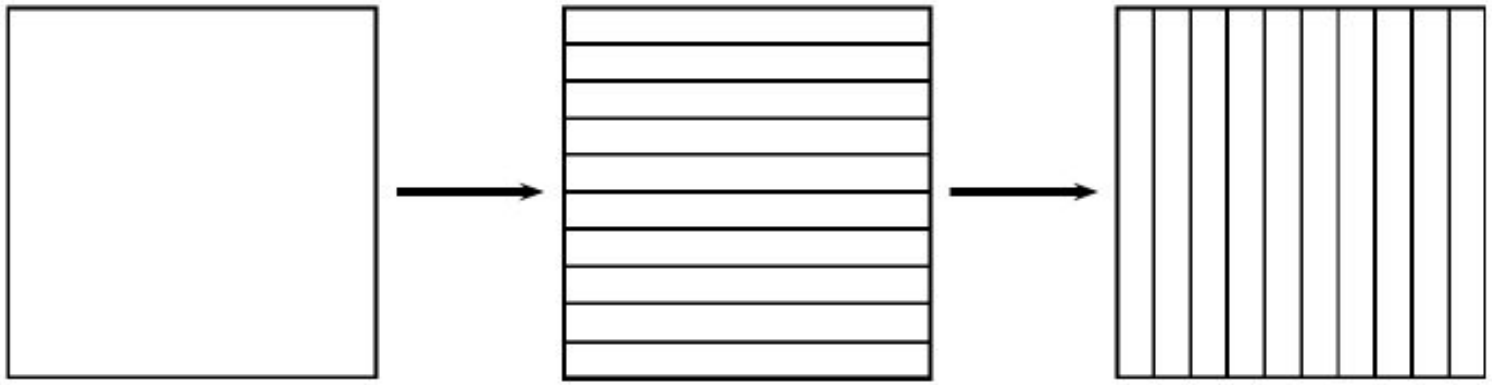
- Formula above can be broken down into simpler formulas for 1D DFT

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

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

Properties: Separabilty of 2D DFT

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



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

Properties of 2D DFT

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

$$\mathcal{F}(f+g) = \mathcal{F}(f) + \mathcal{F}(g)$$
$$\mathcal{F}(kf) = k\mathcal{F}(f)$$ *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:

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

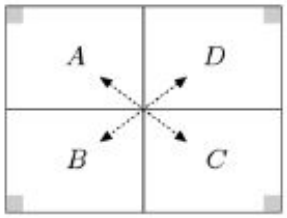
- The value *F(0,0)* of the DFT is called the **dc coefficient**
- If we put $u = v = 0$, then

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

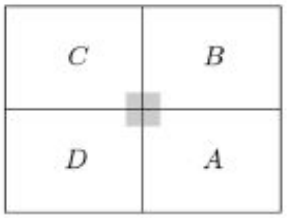
- Essentially *F(0,0)* is the sum of all terms in the original matrix

Centering DFT Spectrum

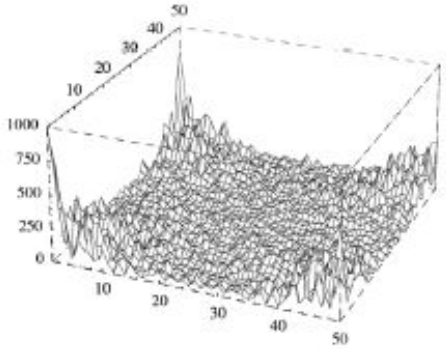
- *F(0,0)* at top left corner
- For display, convenient to have DC component in center
- Just swap four quadrants of Fourier transform



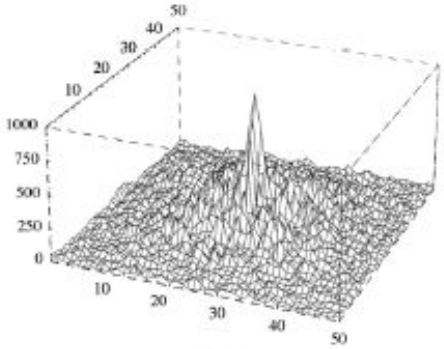
(a)



(b)



(c)

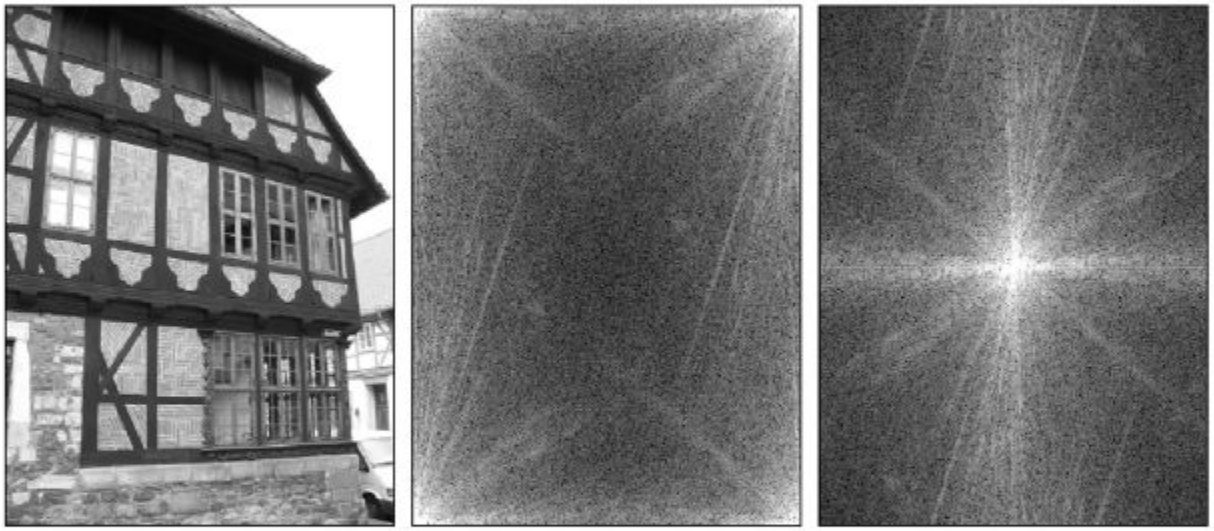


(d)

Swap 4 quadrants to center DC component

DFT spectrum after centering

Centering DFT Spectrum



(a) (b) (c)

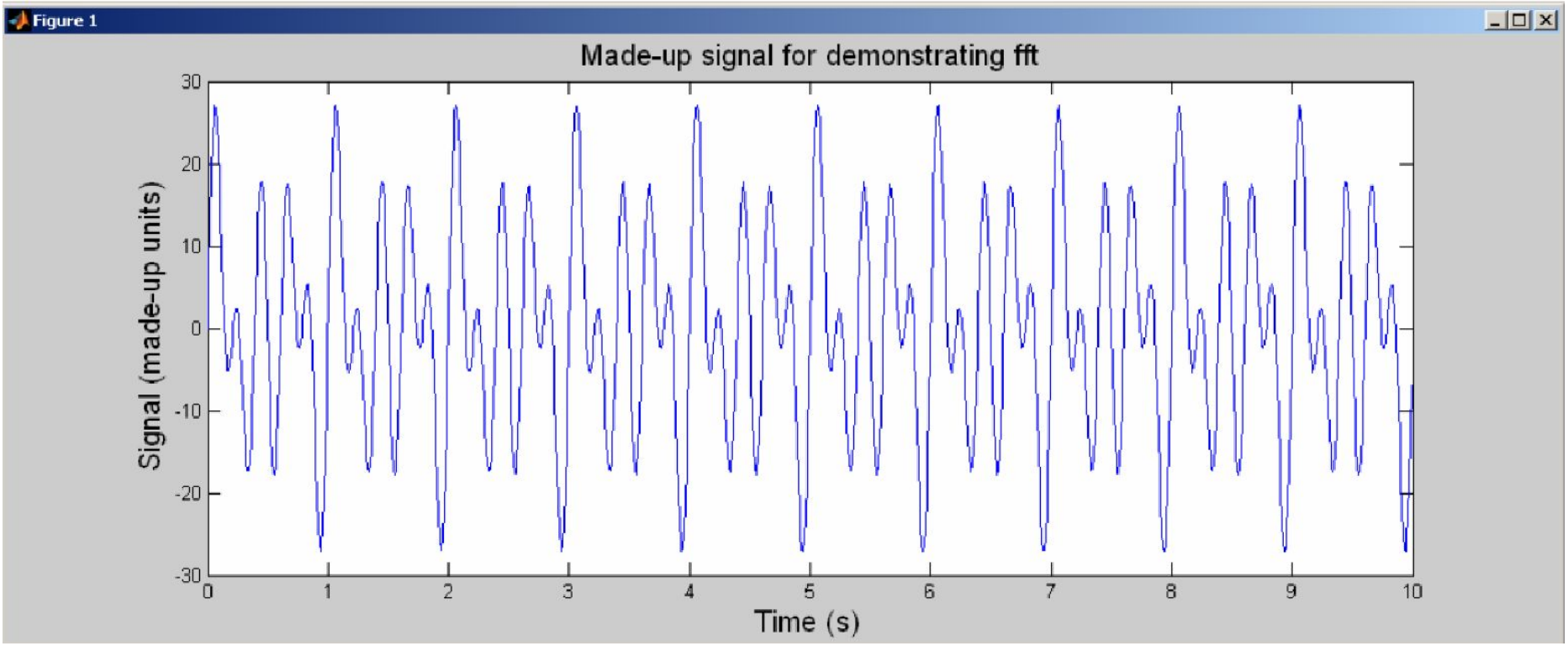
Original Image

Non-centered spectrum

Centered spectrum

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”



x(t) is a bunch of sinusoids added together. x(t) happens to be three sinusoids with frequencies of 2Hz, 3Hz, and 5Hz and amplitudes 7, 11, and 13

$$7\sin(2\pi \bullet 2t) + 11\sin(2\pi \bullet 3t) + 13\sin(2\pi \bullet 5t)$$

ω(rad/s) =
2π×f(Hz)

http://dmorris.net/projects/tutorials/fourier_tutorial.pdf

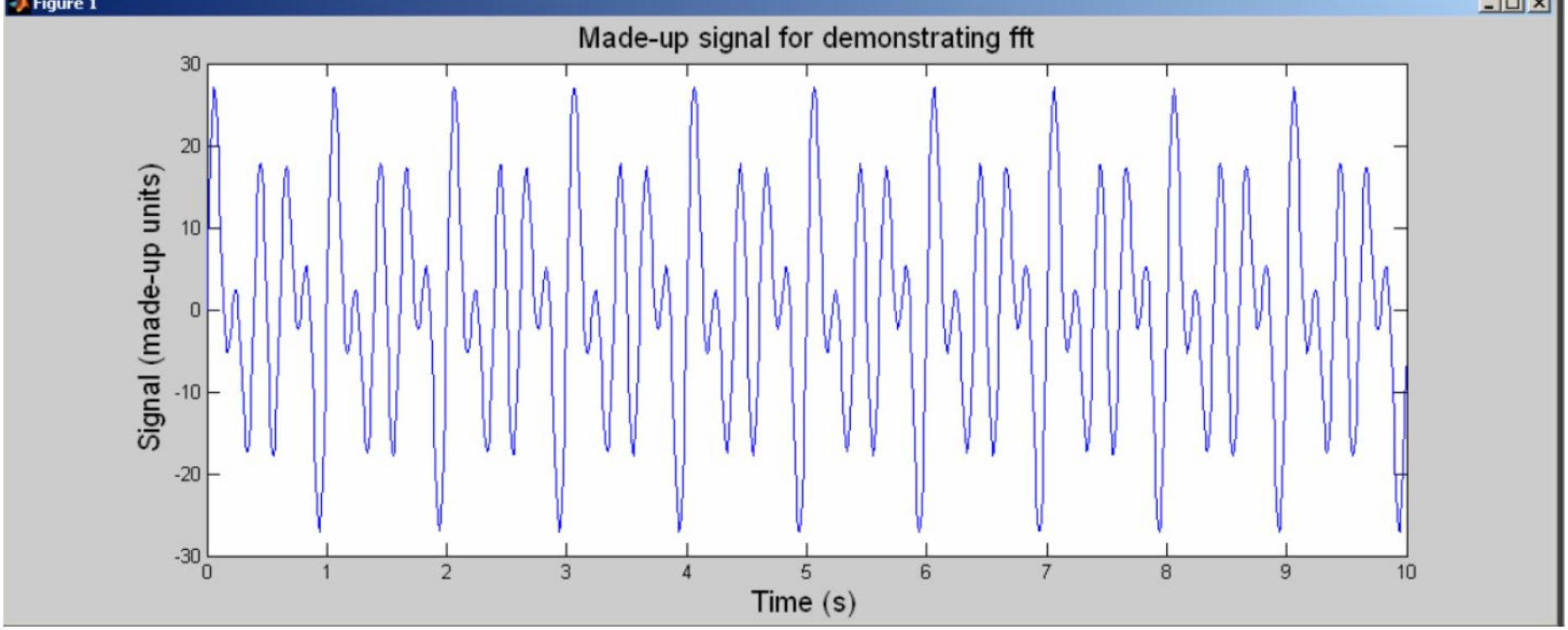


Figure 1
Made-up signal for demonstrating fft
Signal (made-up units)
Time (s)
30
20
10
0
-10
-20
-30
0
1
2
3
4
5
6
7
8
9
10

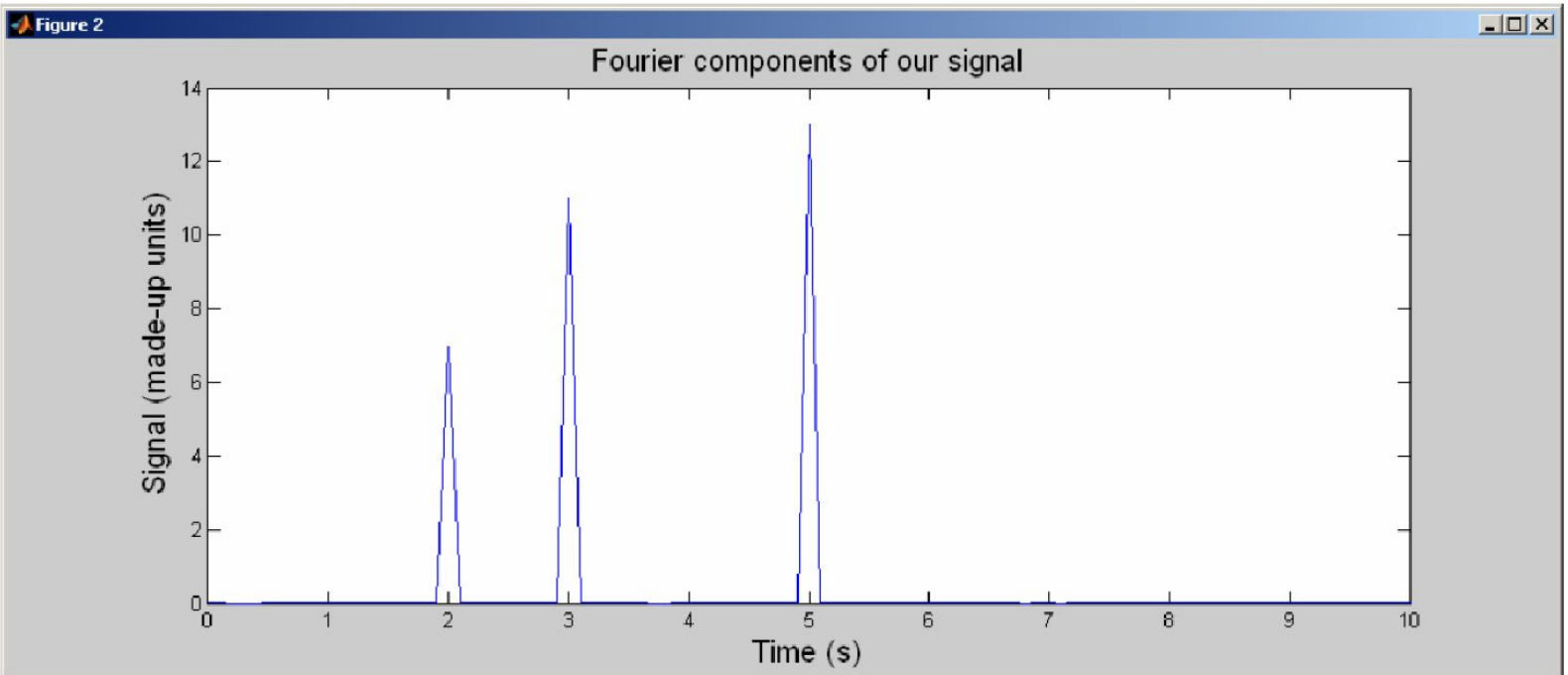
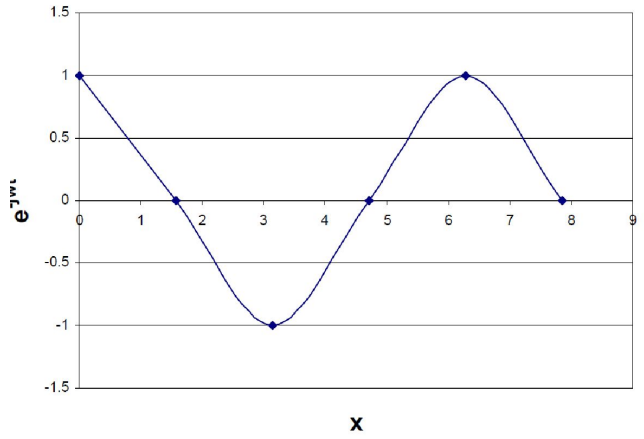


Figure 2
Fourier components of our signal
Signal (made-up units)
Time (s)
14
12
10
8
6
4
2
0
0
1
2
3
4
5
6
7
8
9
10

$$X(\omega) = \int_{-\infty}^{\infty} x(t) e^{-i\omega t}\, dt$$

0*π/2 , 1*π/2 , 2*π/2 , 3*π/2 , 4*π/2 , and 5*π/2

Value of ωt	Result (value of $e^{i\omega t}$)	Real part of result	Imaginary part of result
0π/2	1	1	0
1π/2	*i*	0	1*i
2π/2	-1	-1	0
3π/2	-*i*	0	-1*i
4π/2	1	1	0
5π/2	*i*	0	1*i



amount of signal with frequency ω in x(t) $= \dfrac{x(t)}{e^{i\omega t}} = x(t)e^{-i\omega t}$

amount of signal with frequency ω in x(t) $= \dfrac{x(t)}{e^{i\omega t}} = \int_{t=-\infty}^{t=\infty} x(t)e^{-i\omega t}\,dt$

$$X(\omega) = \int_{t=-\infty}^{t=\infty} x(t)e^{-i\omega t}\,dt$$

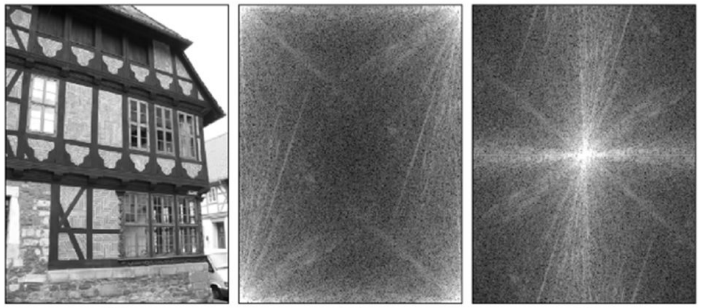
$$x(t) = \int_{\omega=-\infty}^{\omega=\infty} X(\omega)e^{i\omega t}\,d\omega$$

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

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

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

Centering DFT Spectrum



(a) (b) (c)

Original Image | Non-centered spectrum | Centered spectrum

The 2D Frequency Domain

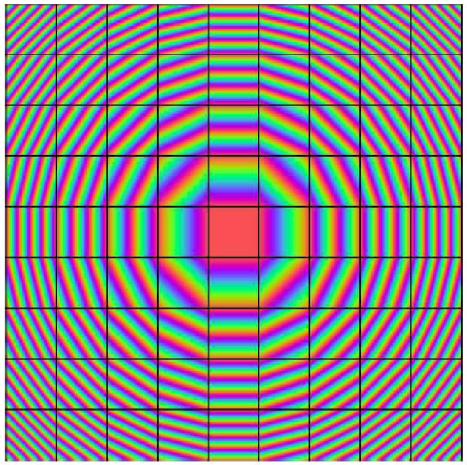


Figure 1: A 9×9 table of harmonic grating basis functions, $e^{j2\pi(ux+vy)}$, where $u \in \{-4 \ldots 4\}$ and $v \in \{-4 \ldots 4\}$. Color indicates phase with red signifying real positive, green signifying imaginary positive, blue signifying real negative and violet signifying imaginary negative.

Separability of 2D Fourier Transform

The 2D analysis formula can be written as a 1D analysis in the x direction followed by a 1D analysis in the y direction:

$$F(u,v) = \int_{-\infty}^{\infty} \left[\int_{-\infty}^{\infty} f(x,y) e^{-j2\pi ux} dx \right] e^{-j2\pi vy} dy.$$

The 2D synthesis formula can be written as a 1D synthesis in the u direction followed by a 1D synthesis in v direction:

$$f(x,y) = \int_{-\infty}^{\infty} \left[\int_{-\infty}^{\infty} F(u,v) e^{j2\pi ux} du \right] e^{j2\pi vy} dv.$$

https://www.cs.unm.edu/~williams/cs530/ft2d.pdf

Separability Theorem

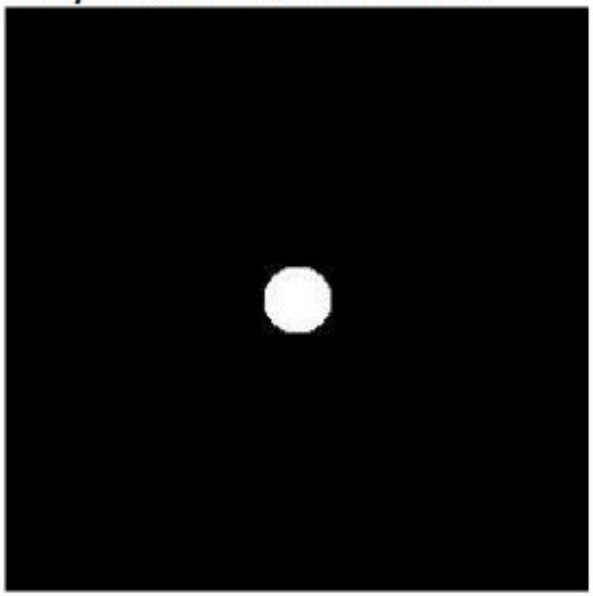
$$f(x,y) = f(x)g(y) \xrightarrow{\mathcal{F}} F(u,v) = F(u)G(v)$$

Proof:

$$\begin{aligned} & F(u,v) \\ &= \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} f(x,y)e^{-j2\pi(ux+vy)}dx\,dy \\ &= \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} f(x)g(y)e^{-j2\pi ux}e^{-j2\pi vy}\,dx\,dy \\ &= \int_{-\infty}^{\infty} f(x)e^{-j2\pi ux}\,dx \int_{-\infty}^{\infty} g(y)e^{-j2\pi vy}dy \\ &= F(u)\;G(v) \end{aligned}$$

Ideal Low Pass Filtering

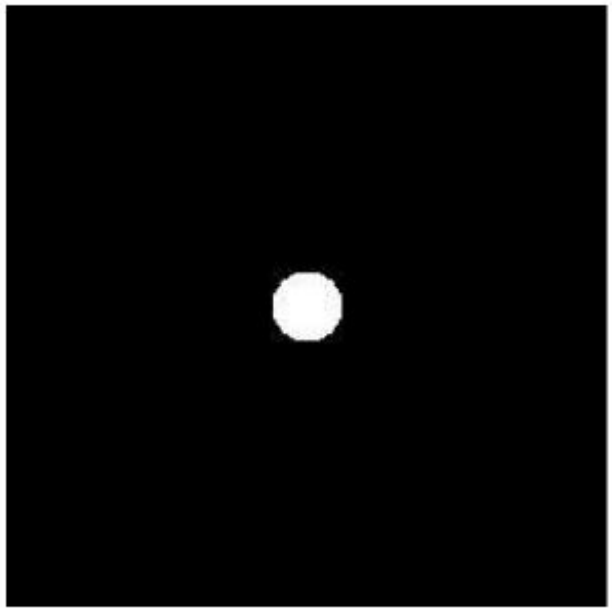
- **Low pass filter:** Keep frequencies **below** a certain frequency
- Low pass filtering causes **blurring**
- After DFT, DC components and low frequency components are towards center
- May specify frequency cutoff as circle c



Ideal Low Pass Filtering

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

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



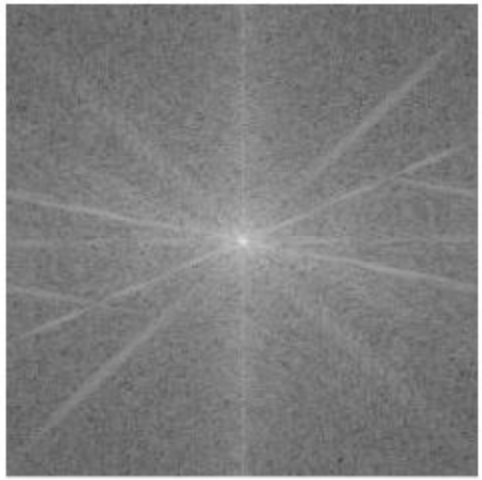
Ideal Low Pass Filtering

- 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

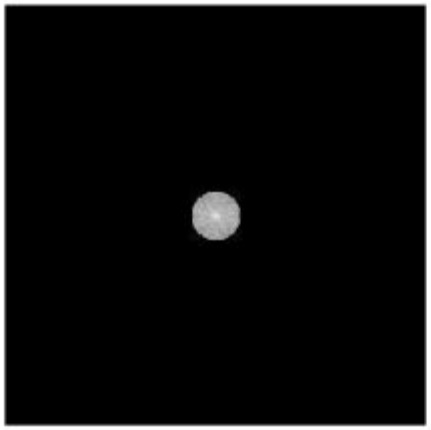
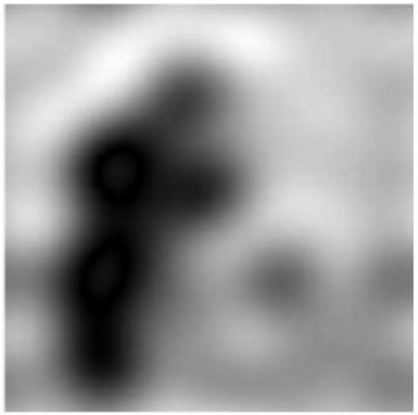


Image after
inversion

low pass filter
Cutoff D = 5

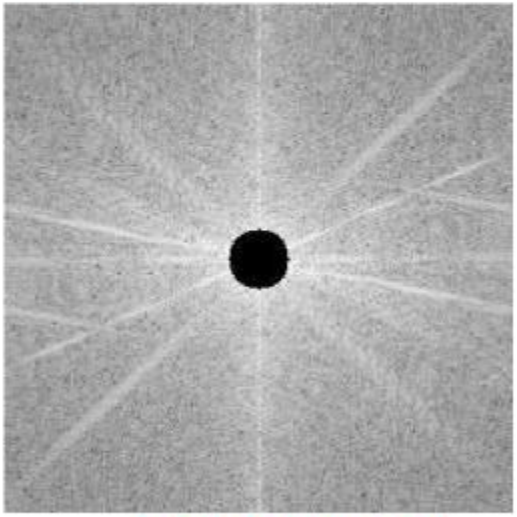


low pass filter
Cutoff D = 30

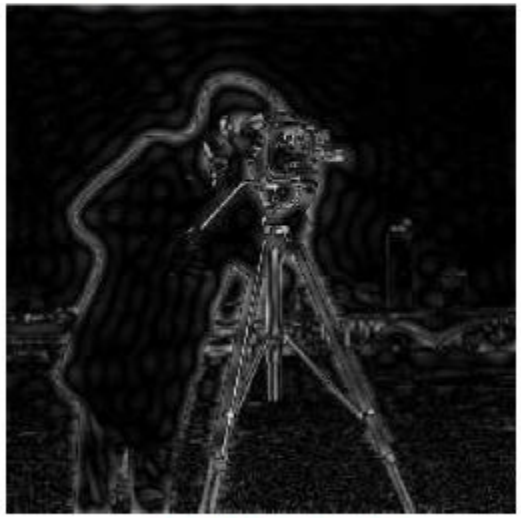
Note: Sharp filter
Cutoff causes
ringing

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



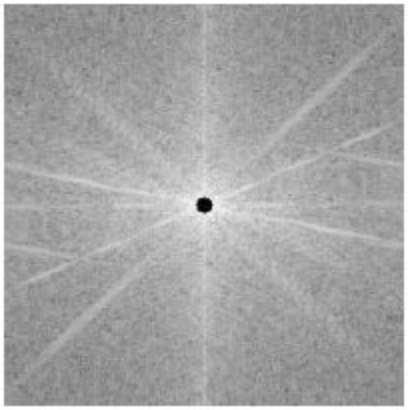
DFT of Image after high pass Filtering



Resulting image after inverse DFT

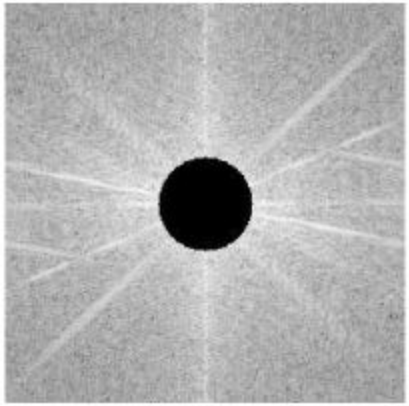
Ideal High Pass Filtering: Effect of Cutoffs

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

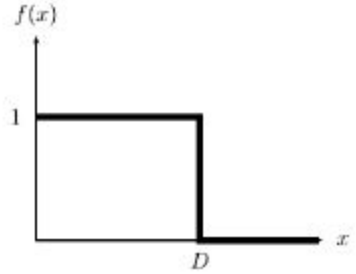
After histogram equalisation

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

Butterworth Filtering

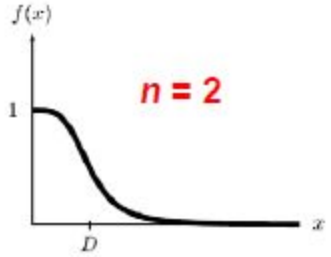
- Sharp cutoff leads to ringing
- To avoid ringing, can use circle with more gentle cutoff slope
- **Butterworth filters** have more gentle cutoff slopes
- Ideal low pass filter

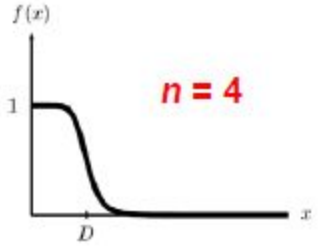
$$f(x) = \begin{cases} 1 & \text{if } x < D \\ 0 & \text{if } x \geq D \end{cases}$$



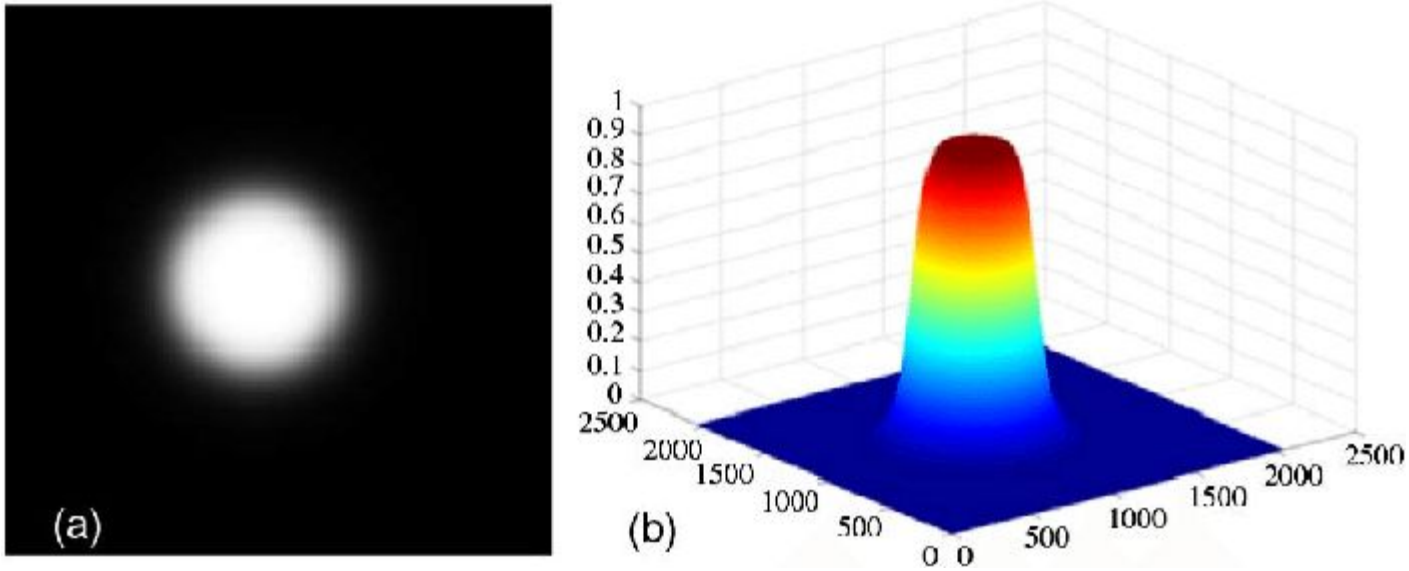
- Butterworth low pass filter

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





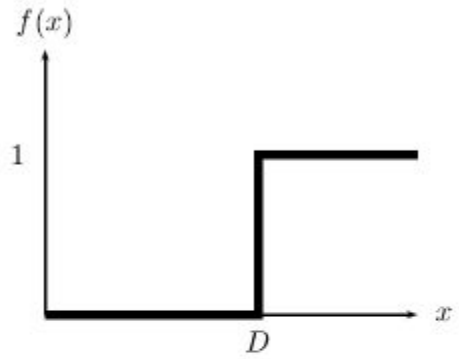
- *n* called **order** of the filter, controls sharpness of cutoff


(a)
(b)
1
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0
2500
2000
1500
1000
500
0
0
500
1000
1500
2000
2500

Butterworth High Pass Filtering

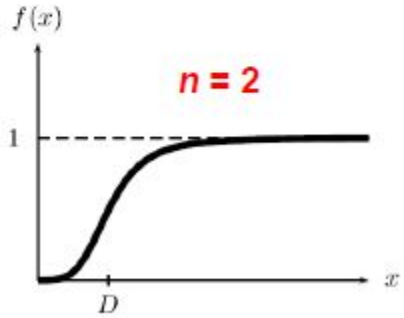
- Ideal high pass filter

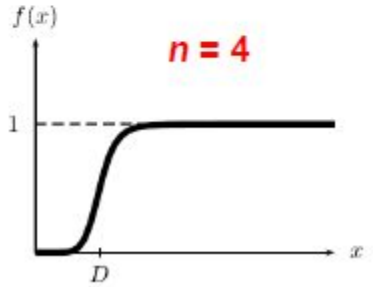
$$f(x) = \begin{cases} 1 & \text{if } x > D \\ 0 & \text{if } x \leq D \end{cases}$$



- Butterworth high pass filter

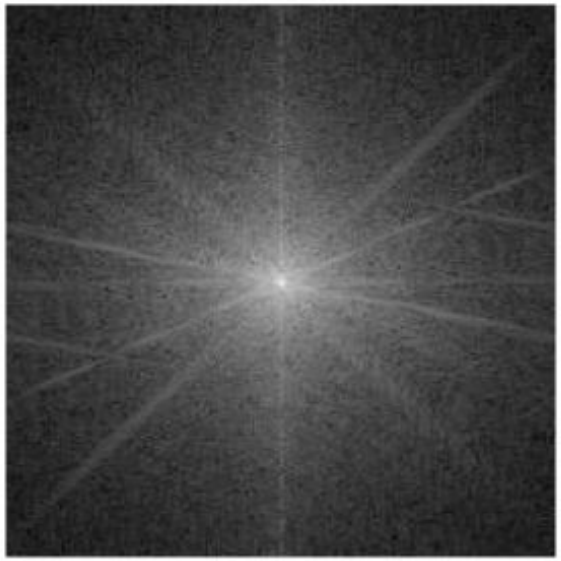
$$f(x) = \frac{1}{1 + (D/x)^{2n}}$$





Low Pass Butterworth Filtering

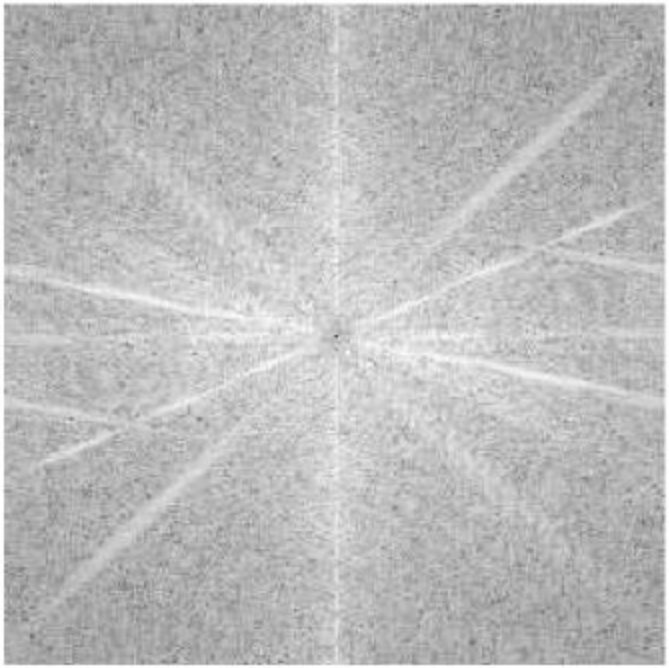
- Low pass filtering removes high frequencies, blurs image
- Gentler cutoff eliminates ringing artifact



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)

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

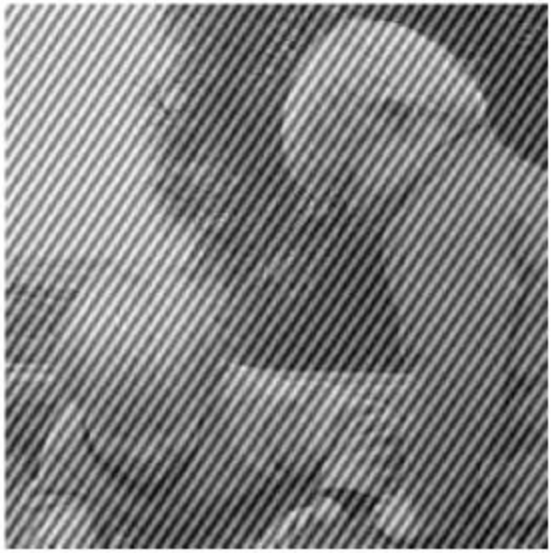
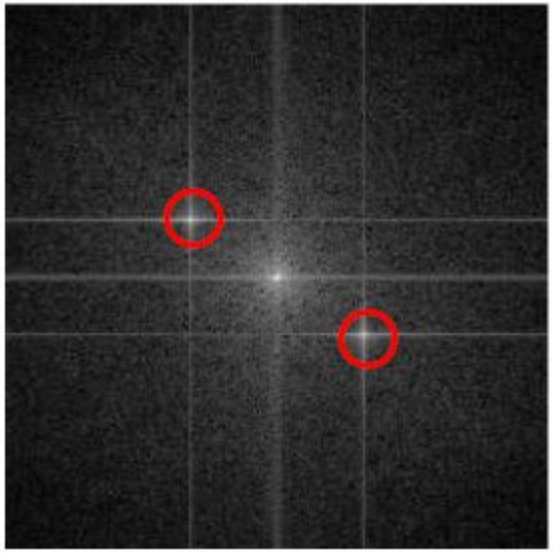


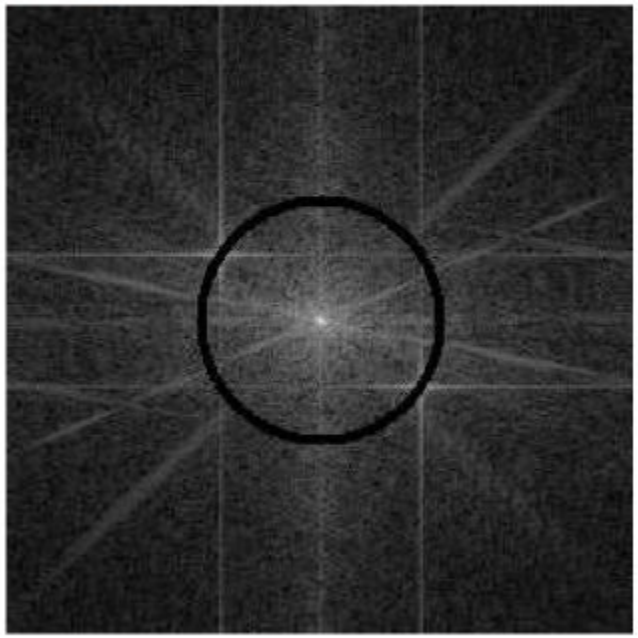
Image with periodic Noise



DFT of Image

Frequency Domain Removal of Periodic Noise: Band Reject Filter

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

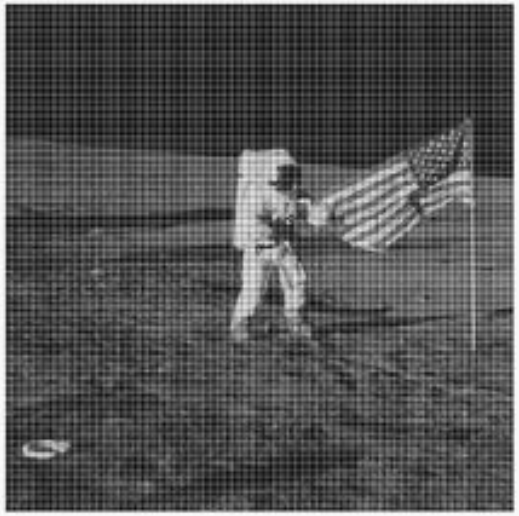


Band Reject Filter

Result after band reject filter applied then inverted

Original Image

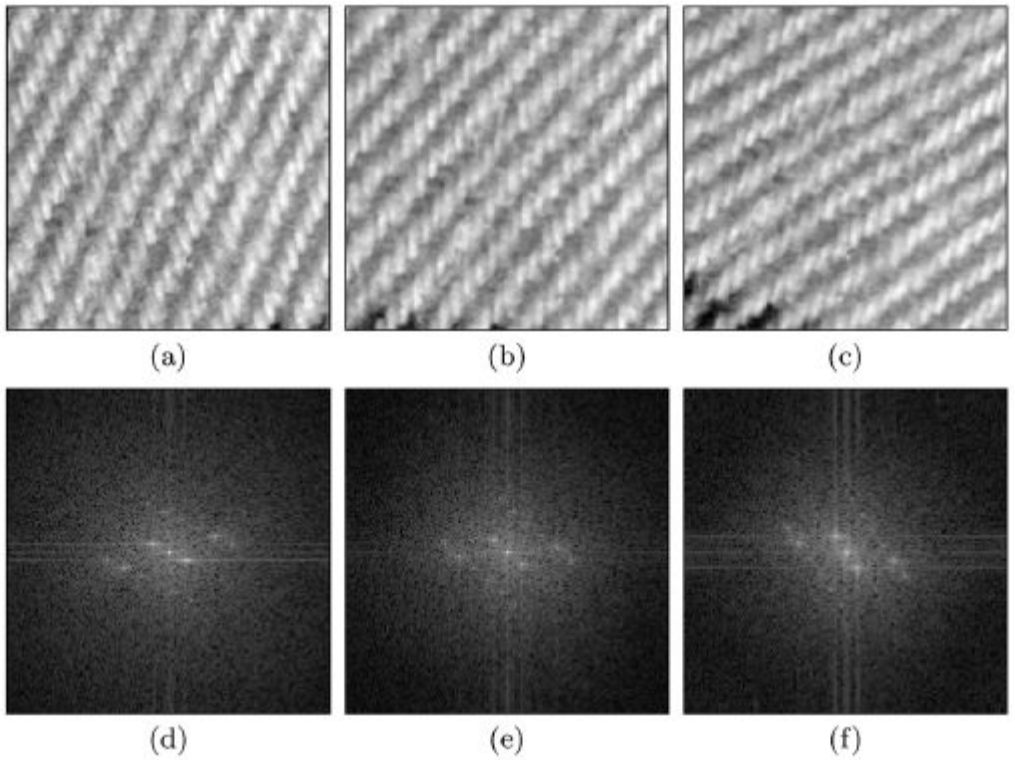
Added Noise



Band Reject Filtered

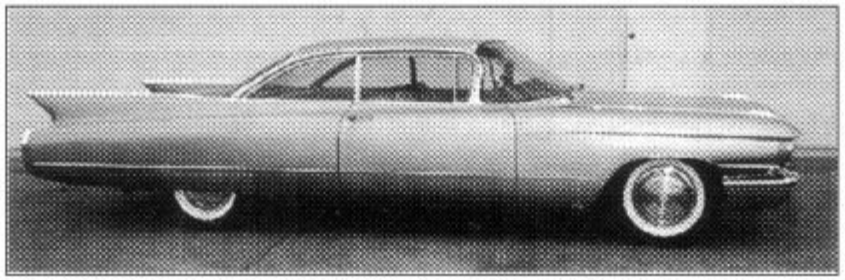
2D Fourier Transform Examples: Rotation

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



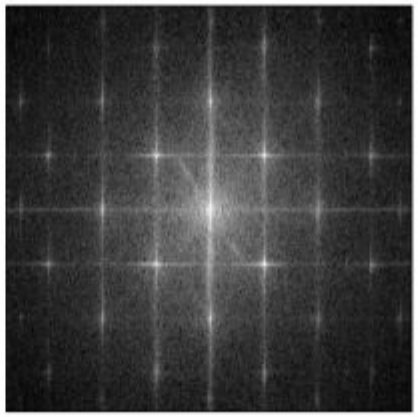
2D Fourier Transform Examples: Printed Patterns

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



(a)

(b)



(c)