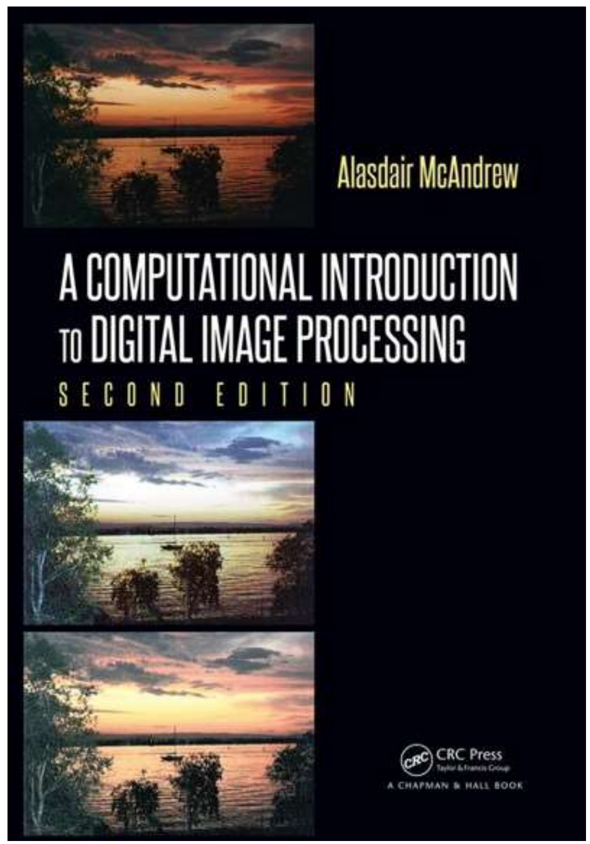
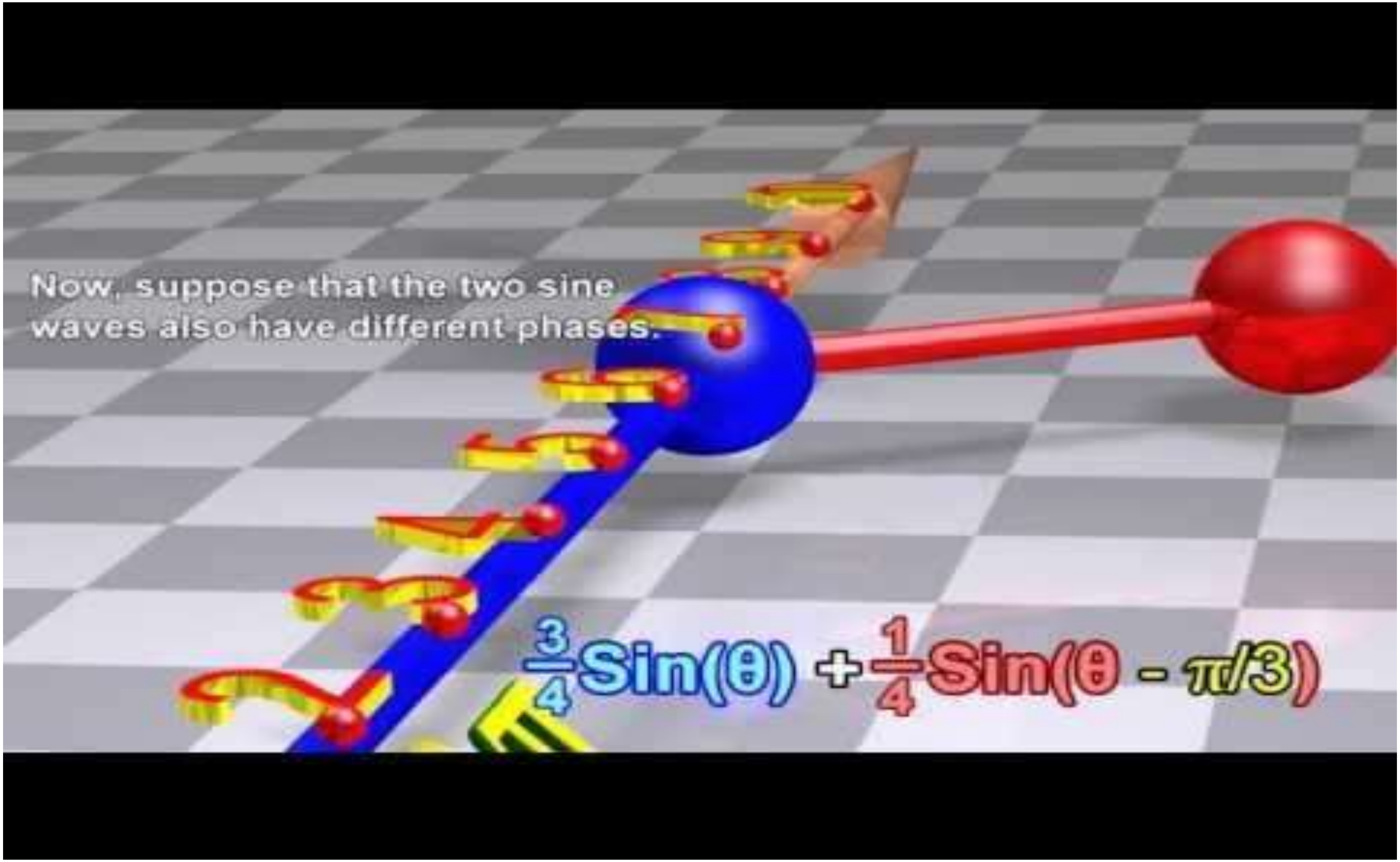
Chapter 7

MIDWESTERN
STATE UNIVERSITY

The Fourier Transform

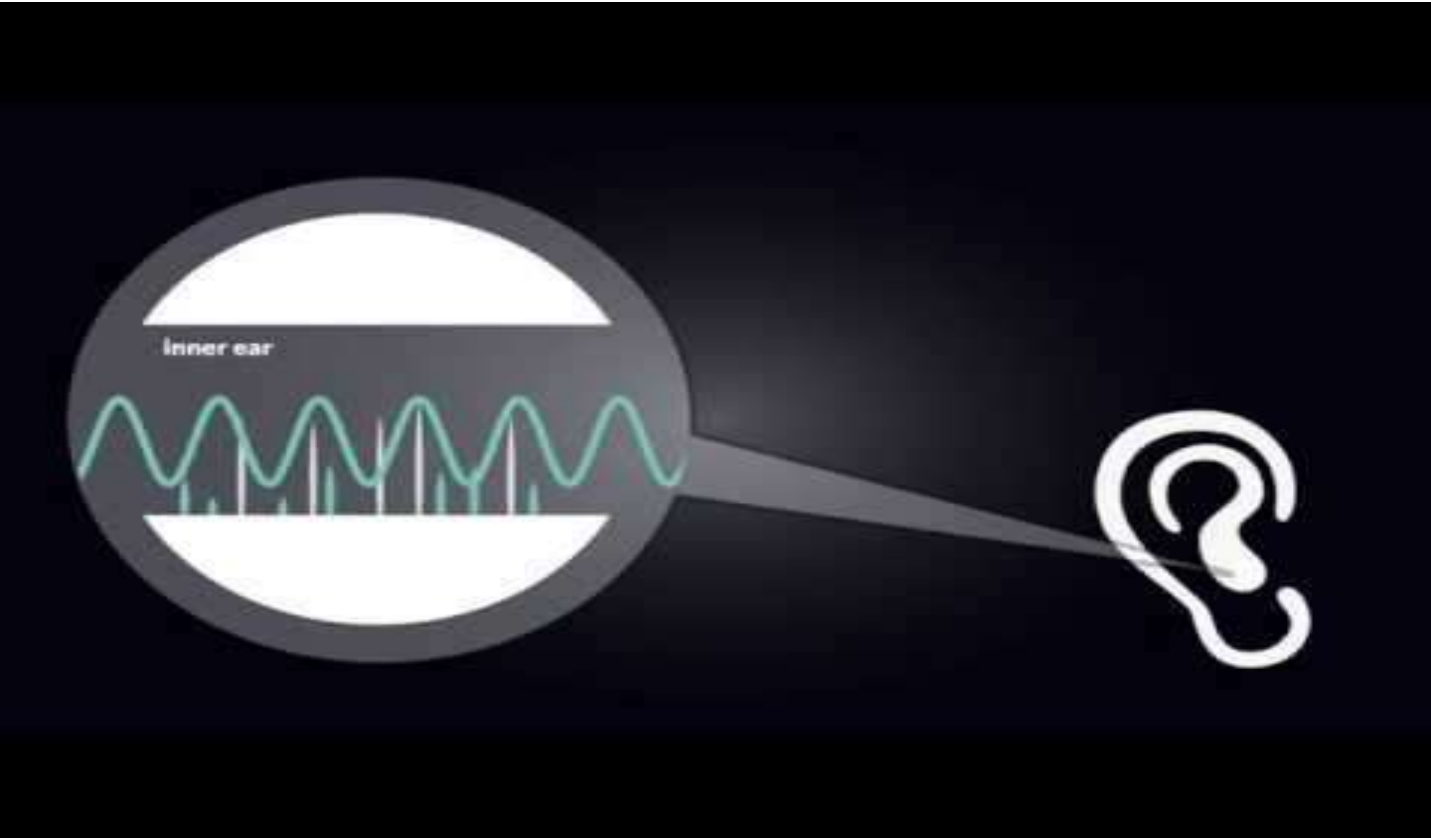


Fourier Transform Video



Fourier Transform Video

Fourier Transform Video



Introduction

Why Fourier Transforms?

- Efficiency
- Alternative to spatial filtering
- Perform low pass and high pass filtering with great degree of precision

One-Dimensional DFT

- Discrete function
  - only have to obtain finite number of values
  - only need finite number of functions

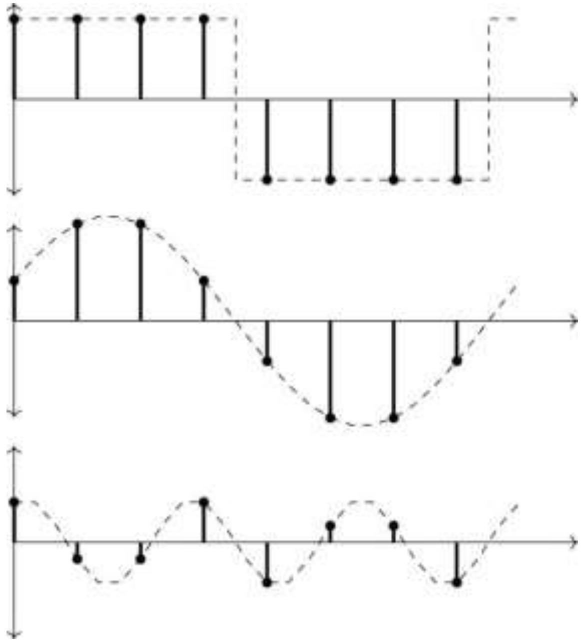


Figure 7.3: Expressing a discrete function as the sum of sines

One-Dimensional DFT

- Matlab or Octave
  - Functions
    - fft
    - ifft
- Python
  - Libraries
    - scipy
    - numpy

Two-Dimensional DFT

- Input: matrix
- Output: second matrix of same size
- F is the Fourier Transform of f and is written:

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

- Definition

$$F(u,v) = \sum_{x=0}^{M-1} \sum_{y=0}^{N-1} f(x,y) \exp\left[-2\pi i \left(\frac{xu}{M} + \frac{yv}{N}\right)\right]. \tag{7.4}$$

$$f(x,y) = \frac{1}{MN} \sum_{u=0}^{M-1} \sum_{v=0}^{N-1} F(u,v) \exp\left[2\pi i \left(\frac{xu}{M} + \frac{yv}{N}\right)\right]. \tag{7.5}$$

Two-Dimensional DFT: Similarity

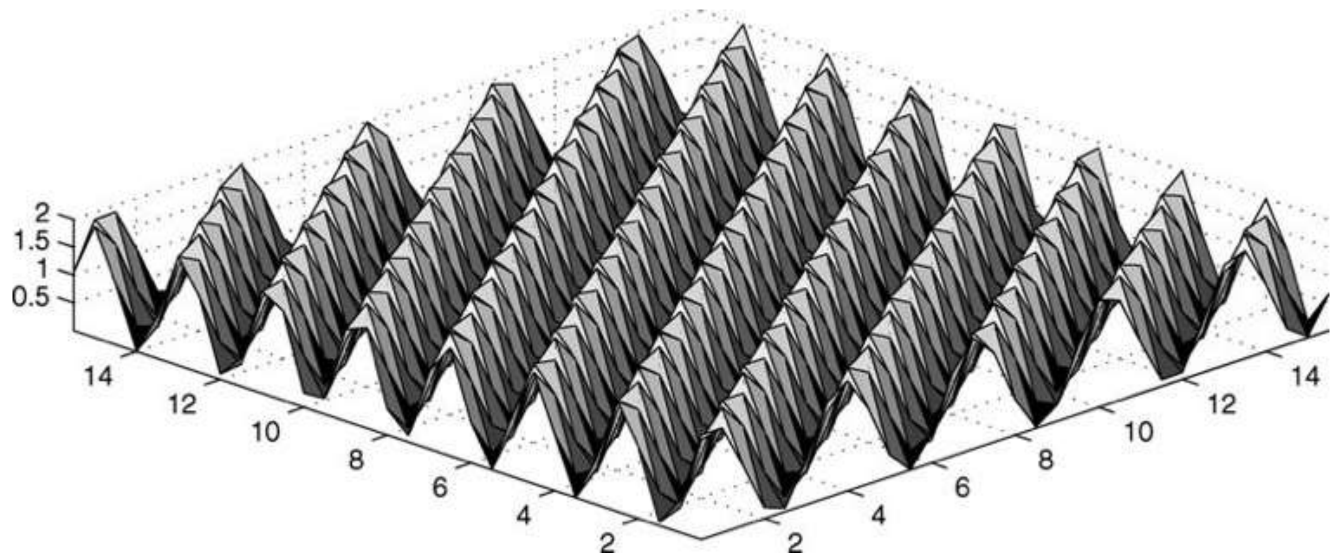
Forward and inverse transforms very similar

- Exceptions:
  - scale factor 1/MN in inverse transform,
  - negative sign in exponent of forward transform
- same algorithm, only very slightly adjusted, can be used for both the forward and inverse transforms.

Two-Dimensional DFT: Spatial Filter

- What is does:
  - multiplies all elements under a mask with fixed values, and adds them all up.
- Can consider DFT as linear spatial filter

Figure 7.5: A “corrugation” function



Two-Dimensional DFT: Convolution Theorem

The convolution theorem: One of the most powerful advantages of using the DFT.

Suppose we wish to convolve an image M with a spatial filter S. Our method has been place S over each pixel of M in turn, calculate the product of all corresponding gray values of M and elements of S, and add the results. The result is called the *digital convolution* of M and S, and is denoted:

$$M * S$$

This method of convolution can be very slow, especially if S is large.

Two-Dimensional DFT: Convolution Theorem

Convolution Theorem states that M * S can be obtained by:

1. Pad S with zeros so that it is the same size as M; denote this padded result by S'.
2. Form the DFTs of both M and S', to obtain F(M) and F(S').
3. Form the element-by-element product of these two transforms: $\mathcal{F}(M) \cdot \mathcal{F}(S')$.
4. Take the inverse transform of the result:

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

Two-Dimensional DFT: Convolution Theorem

Put simply, the convolution theorem states:

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

Two-Dimensional DFT: Convolution Theorem

To convolve a 512x512 image with a 32x32 filter:

Directly: $32^2 = 1024$ multiplications for each pixel
Total 1024 x 512 x 512 = 268,435,456

DFT/FFT: 4608 multiplications per row & per col =
4608 x 512 x 2 = 4,718,592
Same operations for the DFT of filter and inverse DFT,
& 512 x 512 for product of two transforms:

4,718,592 x 3 + 262,144 = 14,417,920

Experimenting with Fourier Transforms

Relevant functions

- `fft`, takes the DFT of a vector
- `ifft`, takes the inverse DFT of a vector
- `fft2`, takes the DFT of a matrix
- `ifft2`, takes the inverse DFT of a matrix
- `fftshift`, shifts a transform

Filtering in the Frequency Domain

MIDWESTERN STATE UNIVERSITY

Why Use FFT in image processing?

Convolution theorem: Spatial convolution can be performed by element-wise multiplication of the Fourier Transform by a suitable "filter matrix"

Low Pass Filtering

MIDWESTERN STATE UNIVERSITY

Definition:

- Eliminate the outer values and keep the inner ones

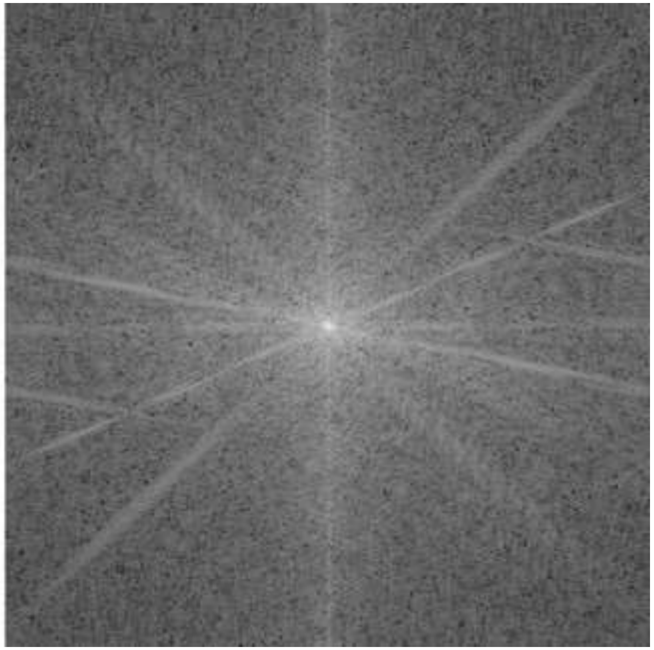
Ideal low pass matrix

- Binary matrix *m* defined by

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

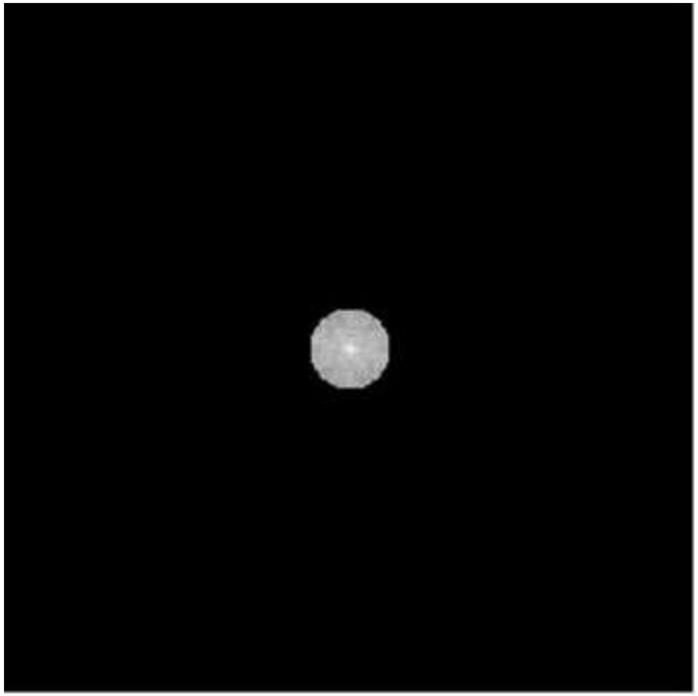
Low Pass Filtering

Figure 7.14: The “cameraman” image and its DFT



Low Pass Filtering

Figure 7.15: Applying ideal low pass filtering



(a) Ideal filtering on the DFT

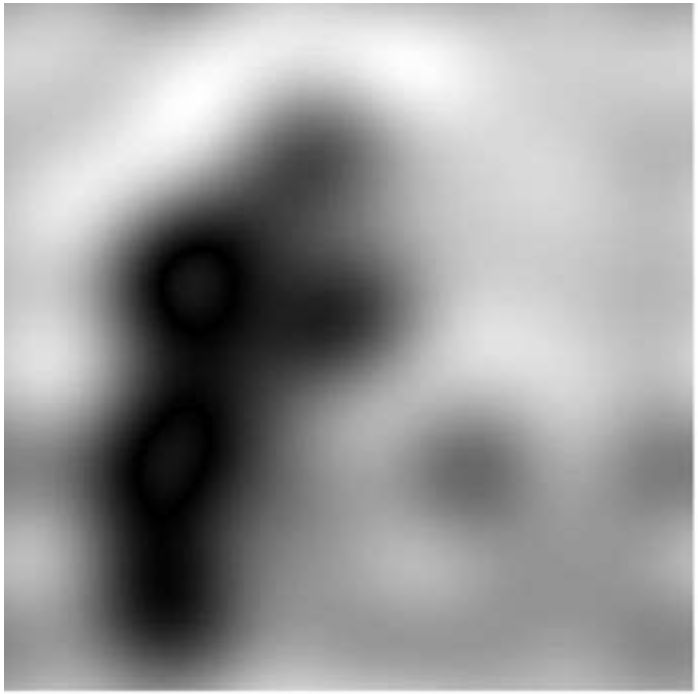
(b) After inversion

Low Pass Filtering

Expect:

- Smaller the circle, the more blurred the image
- Larger the circle, the less blurred

Figure 7.16: Ideal low pass filtering with different cutoffs



(a) Cutoff of 5

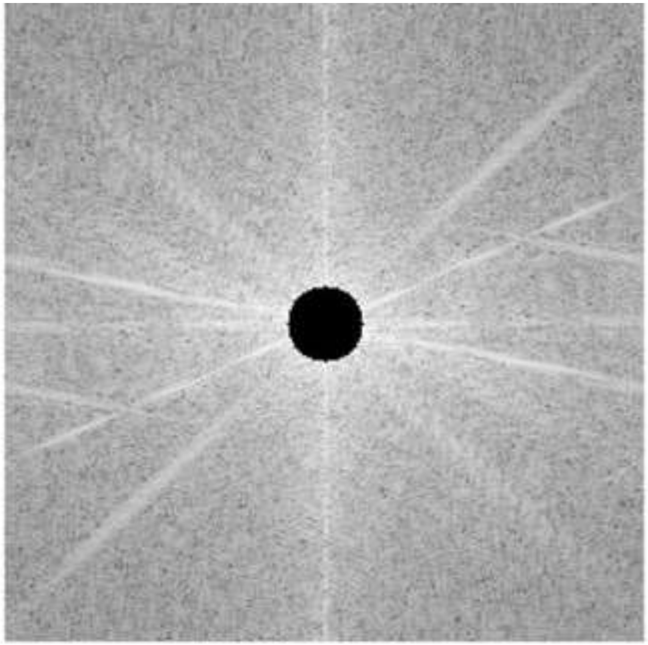
(b) Cutoff of 30

High Pass Filtering

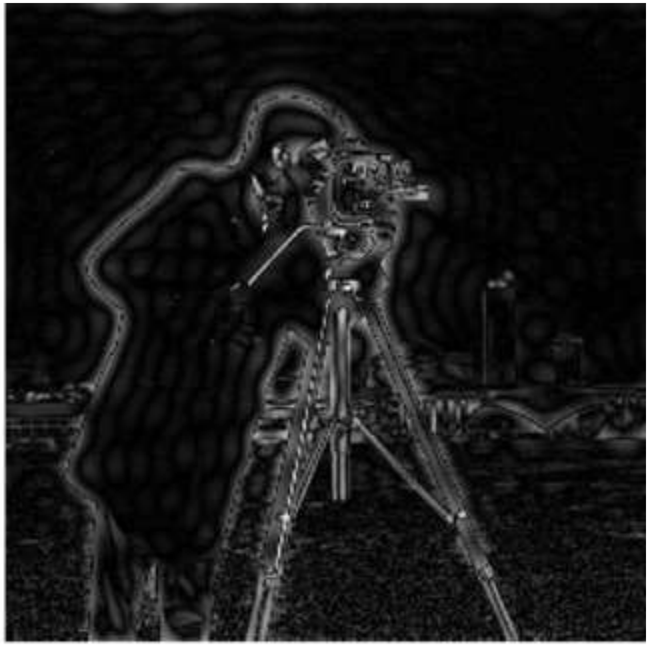
Definition:

– Eliminating center values and keeping the others

Figure 7.17: Applying an ideal high pass filter to an image



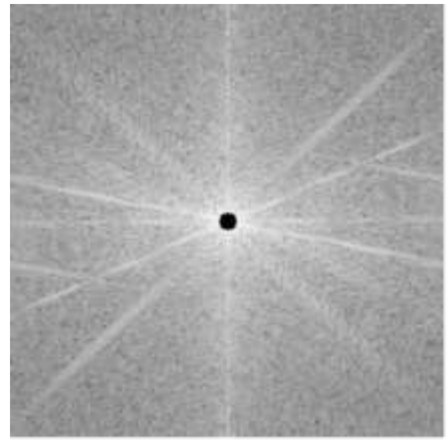
(a) The DFT after high pass filtering



(b) The resulting image

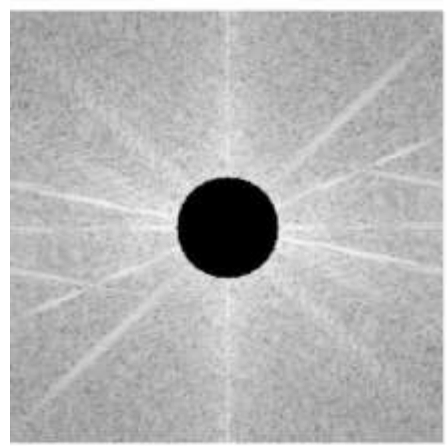
High Pass Filtering

Figure 7.18: Ideal high pass filtering with different cutoffs



(a) Cutoff of 5

(b) The resulting image



(a) Cutoff of 30

(b) The resulting image

Butterworth Filtering

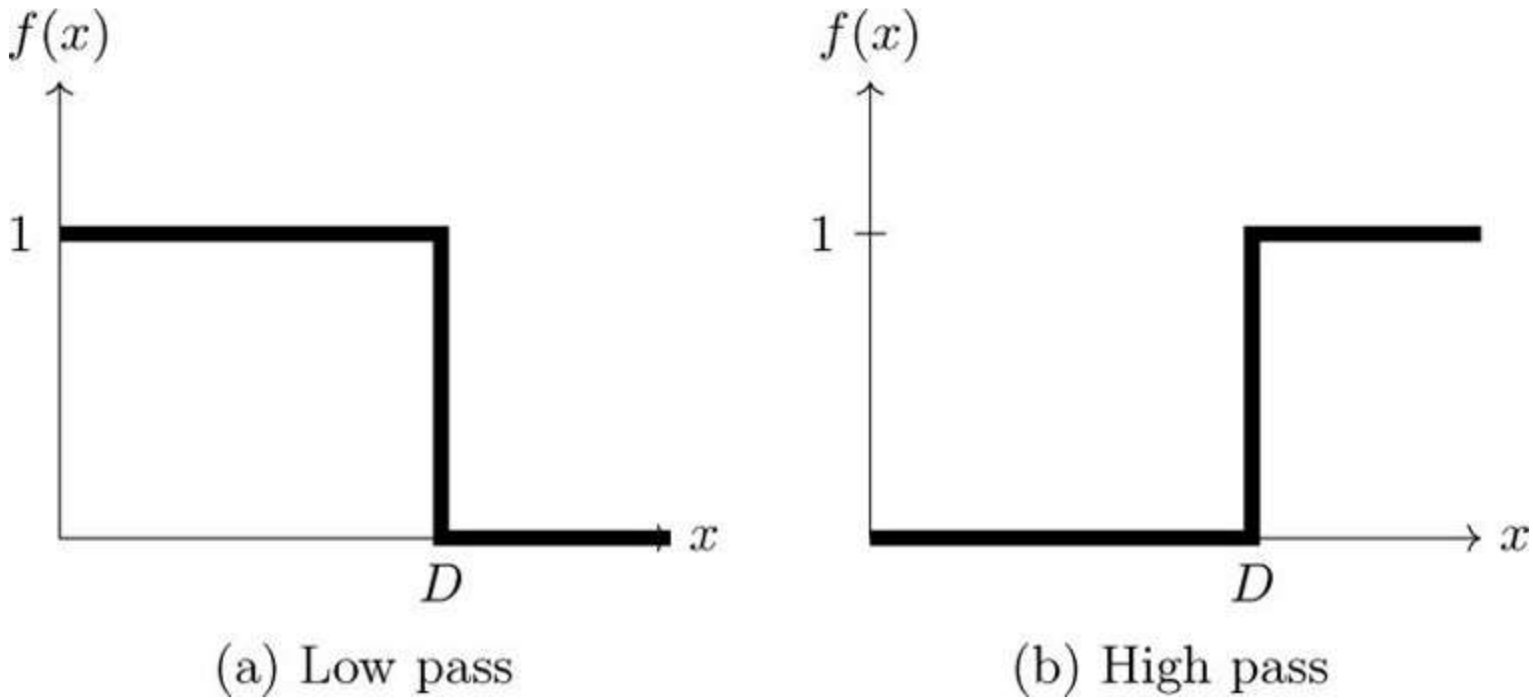
Butterworth filter functions are based on the following functions:

- Low pass filters: $f(x) = \frac{1}{1 + (x/D)^{2n}}$
- High pass filters: $f(x) = \frac{1}{1 + (D/x)^{2n}}$

Where in each case the parameter *n* is called the *order* of the filter

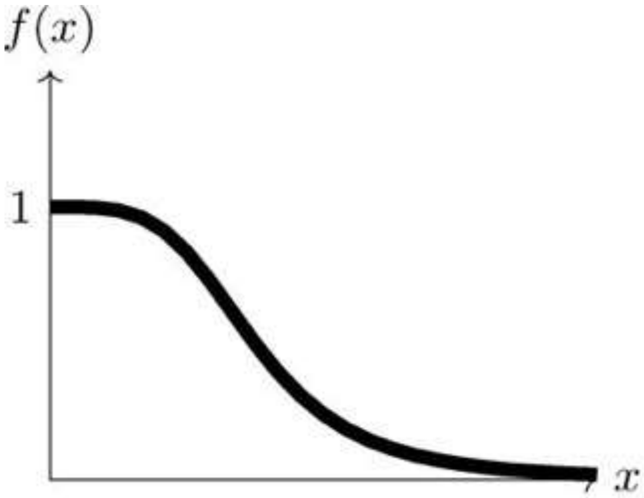
Butterworth Filtering

Figure 7.19: Ideal filter functions

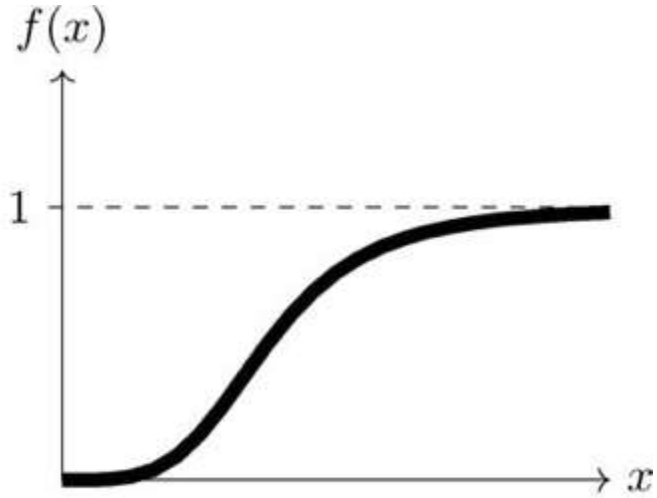


Butterworth Filtering

Figure 7.20: Butterworth filter functions with n = 2



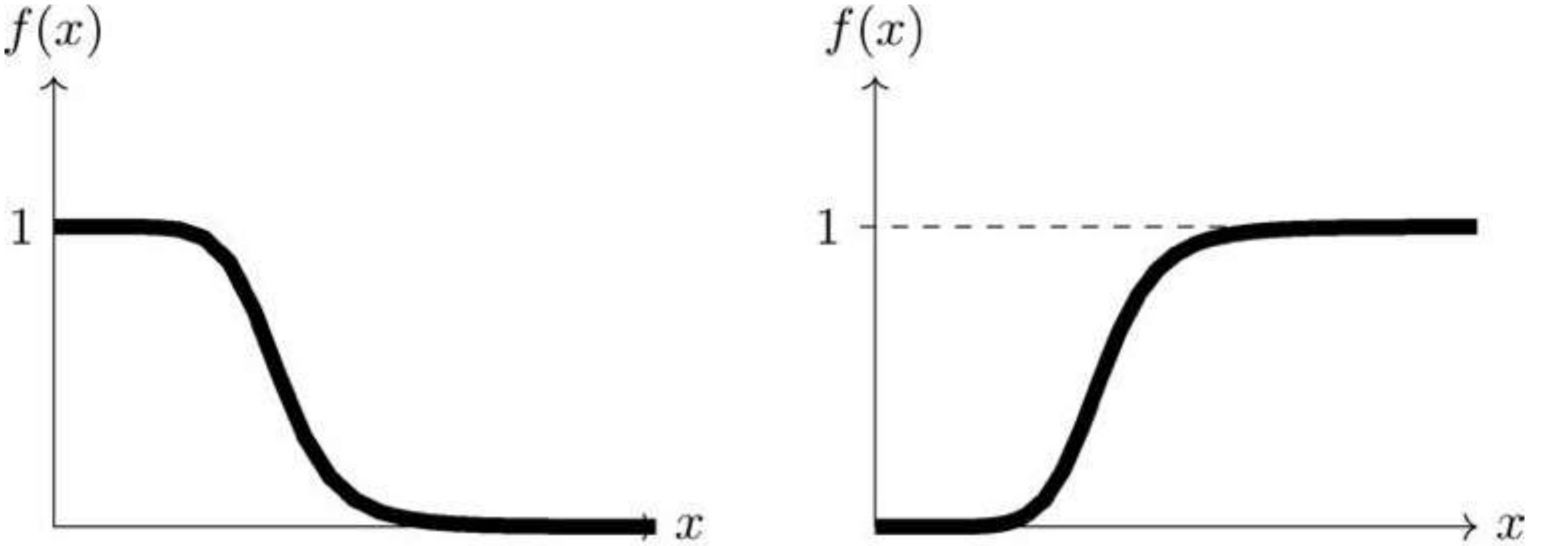
(a) Low pass



(b) High pass

Butterworth Filtering

Figure 7.21: Butterworth filter functions with n = 4

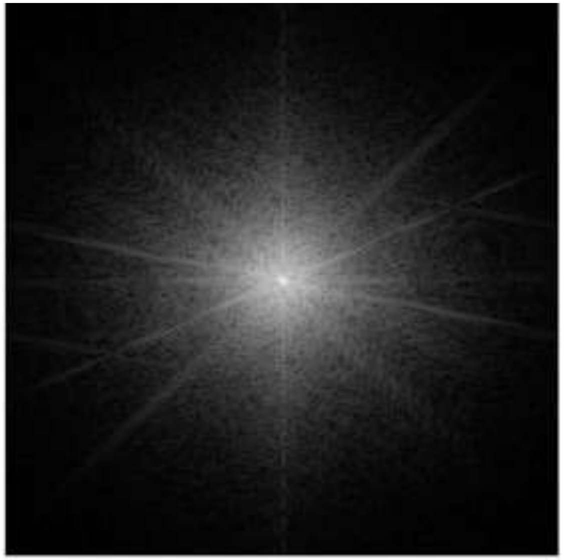


(a) Low pass

(b) High pass

Butterworth Filtering

Figure 7.22: Butterworth low pass filtering

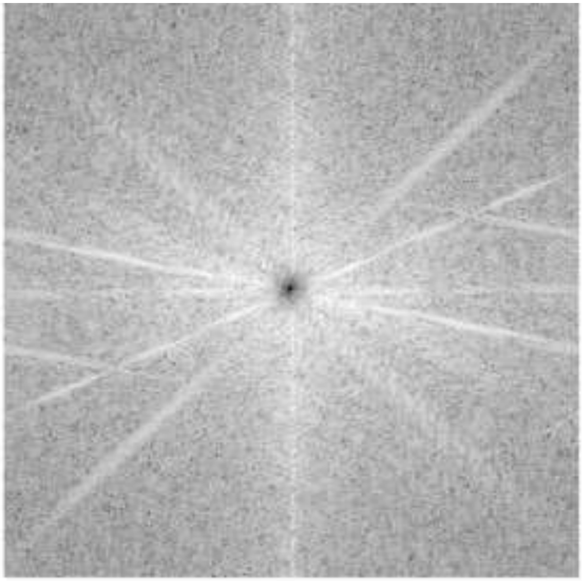


(a) The DFT after Butterworth low pass filtering

(b) The resulting image

Butterworth Filtering

Figure 7.23: Butterworth high pass filtering



(a) The DFT after Butterworth high pass filtering

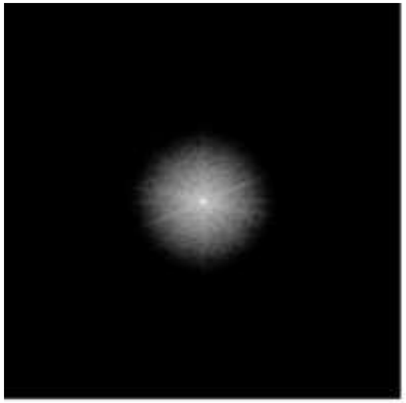
(b) The resulting image

Gaussian Filtering

- Implementation
  - Create Guassian filter
  - Multiply it by the image transform
  - Invert the result
- Considered the most “smooth”

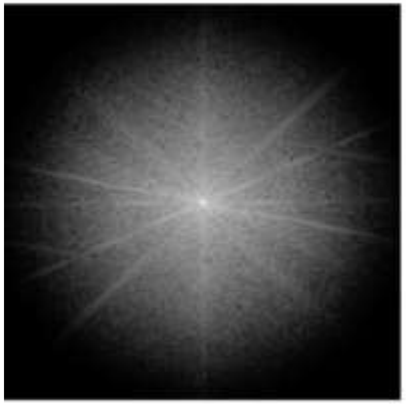
Gaussian Filtering

Figure 7.24: Applying a Gaussian low pass filter in the frequency domain



(a) $\sigma = 10$

(b) Resulting image



(c) $\sigma = 30$

(d) Resulting image

Gaussian Filtering

Figure 7.25: Applying a Gaussian high pass filter in the frequency domain

(a) Using $\sigma = 10$

(b) Using $\sigma = 30$