Computer Vision

Lecture 8: Frequential Analysis

November 12, 2021

Prof. Sébastien Lefèvre
sebastien.lefevre@univ-ubs.fr

Reminder

An image can be seen as a sum of sinusoids at different frequencies & amplitudes.

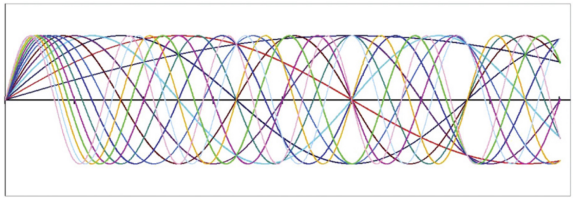


Fig. 4.1 An image can be considered as an assembly of spatial information at various frequencies.

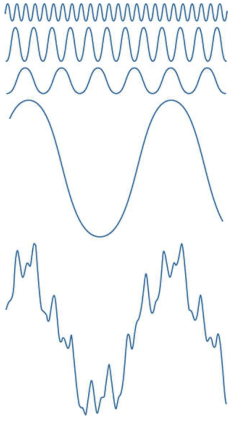


FIGURE 4.1

The function at the bottom is the sum of the four functions above it. Fourier's idea in 1807 that periodic functions could be represented as a weighted sum of sines and cosines was met with skepticism.

Image processing and GIS for remote Sensing: Techniques and applications; Liu & Mason; Wiley Blackwell

Digital image processing; Gonzalez, Woods; Pearson

Fourier theory

Any periodic function can be expressed as the sum of sines and cosines of different frequencies, each multiplied by a different coefficient.

Non-periodic functions (but with area under the curve being finite) can be expressed as the integral of sines and/or cosine multiplied by a weighting function. This is the Fourier transform.

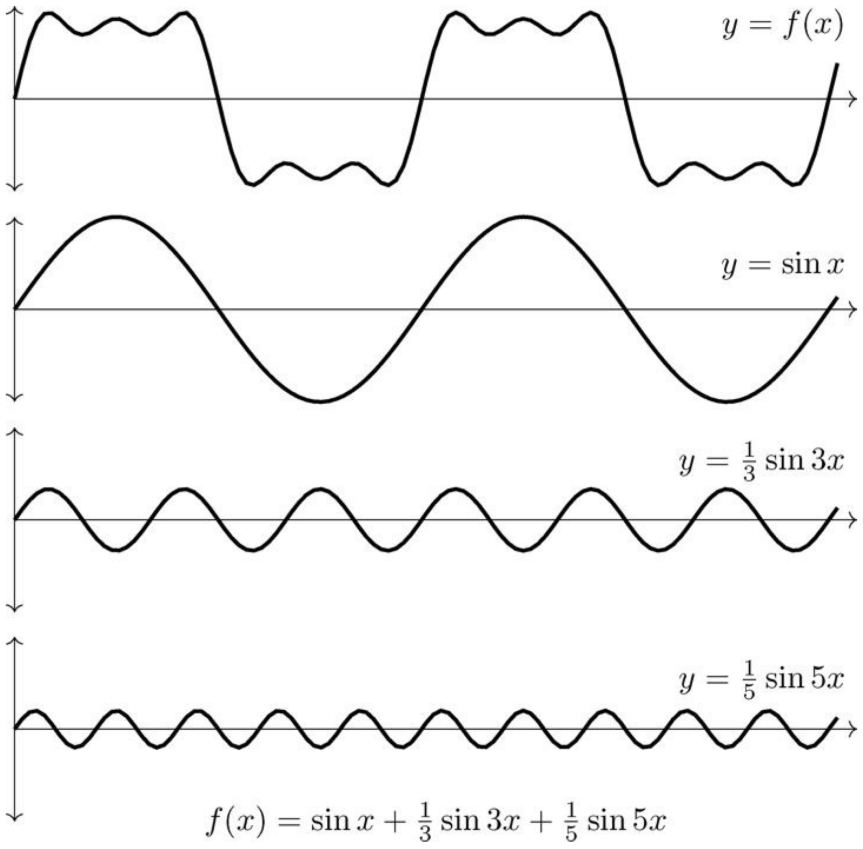
A function, expressed in a Fourier series/transform, can be reconstructed (recovered) completely via an inverse process, with no loss of information.

Fast Fourier Transform (FFT) in the early 1960s revolutionized the field of signal processing.

A note on complexity: for an image of size $M \times N$ and a kernel of size $m \times n$, spatial filtering complexity is $MNmn$, $MN(m+n)$ if the kernel is separable; it becomes $2MN \log_2 MN$ in the frequency domain.

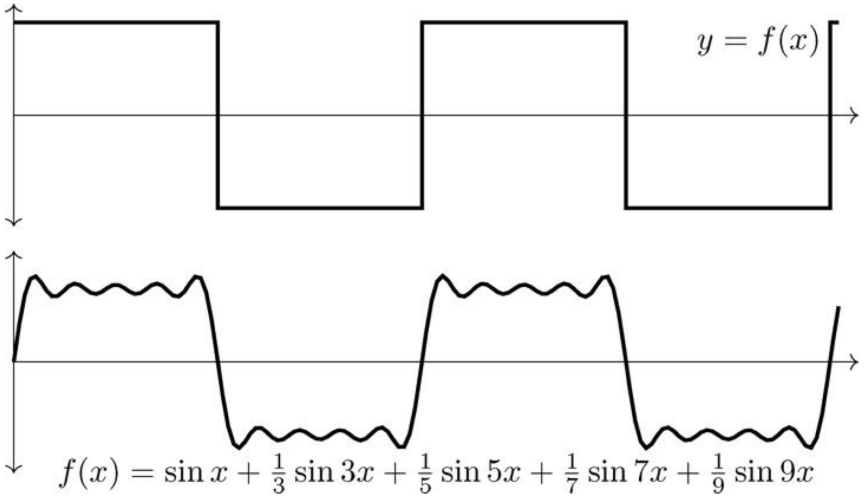
Examples of decompositions

Figure 7.1: A function and its trigonometric decomposition



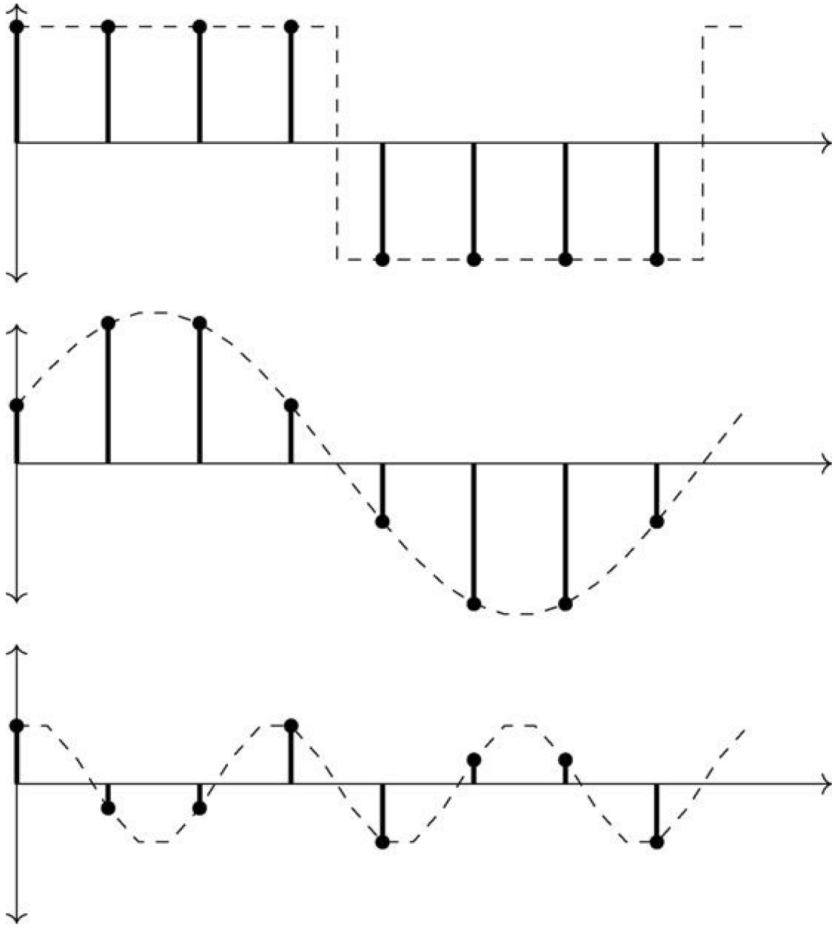
Examples of decompositions

Figure 7.2: A square wave and its trigonometric approximation



Examples of decompositions

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



Complex numbers

A complex number C is defined as $C = R + iI$ where R and I are real numbers and $i = \sqrt{-1}$: R is the real part and I its imaginary part.

The conjugate of a complex number is $C^* = R - iI$, i.e. the symetric over the imaginary axis of the complex plane.

Complex numbers can be represented in polar coordinates:
$C = |C|(\cos\phi + i\sin\phi)$ where $|C| = \sqrt{R^2 + I^2}$ is the length of the vector extending from the origin of the complex plane to point (R, I) and ϕ is the angle between the vector and the real axis.

The Euler's formula $e^{i\phi} = \cos\phi + i\sin\phi$ gives $C = |C|e^{i\phi}$.

These definitions are applicable to complex functions:
$F(u) = R(u) + iI(u) = |F(u)|e^{i\phi(u)}$
with $|F(u)|$ the magnitude and $\phi(u)$ the phase.

Fourier series

The decomposition (a.k.a. Fourier series) of a function f of continuous variable x, that is periodic with a period T, is:

$$f(x) = \sum_{n=-\infty}^{\infty} c_n e^{i\frac{2\pi n}{T}x}$$

with c_n the Fourier coefficients (for $n = 0, \pm 1, \pm 2, \ldots$):

$$c_n = \frac{1}{T}\int_{-T/2}^{T/2} f(x) e^{-i\frac{2\pi n}{T}x} dx$$

If the function is non-periodic, we can obtain similar formulation with letting $T \to \infty$.

Fourier Transform

The Fourier transform maps a spatial domain function $f(x)$ into a frequency domain function $F(u)$:

$$F(u) = \int_{-\infty}^{\infty} f(x) e^{-2\pi i u x} dx$$

The spatial domain function $f(x)$ can be recovered from $F(u)$ by the inverse Fourier transform:

$$f(x) = \int_{-\infty}^{\infty} F(u) e^{2\pi i u x} du$$

Extension to 2D leads to:

$$F(u,v) = \iint_{-\infty}^{\infty} f(x,y) e^{-2\pi i (ux+vy)} dx\ dy$$

Discrete Fourier Transform (DFT)

Digital images require a discrete definition for an image of size $M \times N$:

$$F(u,v) = \sum_{x=0}^{M-1} \sum_{y=0}^{N-1} f(x,y) e^{-2\pi i (\frac{ux}{M} + \frac{vy}{N})}$$

The image can be reconstructed using the inverse transform:

$$f(x,y) = \frac{1}{MN} \sum_{u=0}^{M-1} \sum_{v=0}^{N-1} F(u,v) e^{-2\pi i (\frac{ux}{M} + \frac{vy}{N})}$$

The magnitude $|F(u,v)| = \sqrt{R(u,v)^2 + I(u,v)^2}$ (a.k.a. Fourier spectrum), or the power spectrum $|F(u,v)|^2$, can be displayed as a 2D image, showing direction of frequency components in f. Note that $|F(u,v)| = |F(-u,-v)|$.

The phase $\phi(u,v) = \arctan(\frac{I(u,v)}{R(u,v)})$ needs to be computed with a 4-quadrant arctangent function (e.g. Python `atan2`), and is mandatory to recover the signal. Note that $\phi(u,v) = -\phi(-u,-v)$

Efficient implementation

Let us note that the 2D DFT is separable:

$$F(u,v) = \sum_{x=0}^{M-1} F(x,v) e^{-2\pi i \frac{ux}{M}}$$

with

$$F(x,v) = \sum_{y=0}^{N-1} f(x,y) e^{-2\pi i \frac{vy}{N}}$$

The 2D IDFT is also separable, and can even be computed with 1D DFT.

In practice, the DFT is implemented through a Fast Fourier Transform (FFT) algorithm, reducing the complexity from $O(N^2)$ to $O(N \log N)$ with N the data size.

Among the various FFT algorithms, the most popular is from Cooley and Tukey (1965) and consists in dividing the transform in several of smaller size.

Examples of Fourier spectra

a b
c d

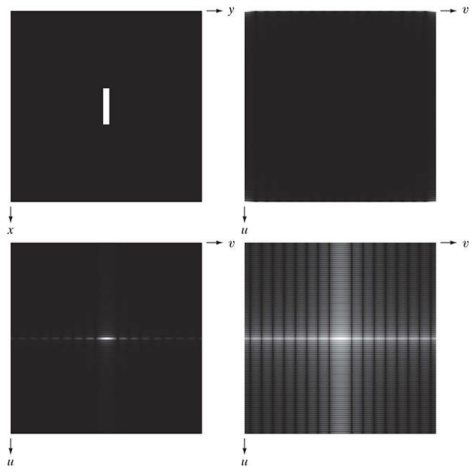


FIGURE 4.23
(a) Image. (b) Spectrum, showing small, bright areas in the four corners (you have to look carefully to see them). (c) Centered spectrum. (d) Result after a log transformation. The zero crossings of the spectrum are closer in the vertical direction because the rectangle in (a) is longer in that direction. The right-handed coordinate convention used in the book places the origin of the spatial and frequency domains at the top left (see Fig. 2.19).

a b
c d

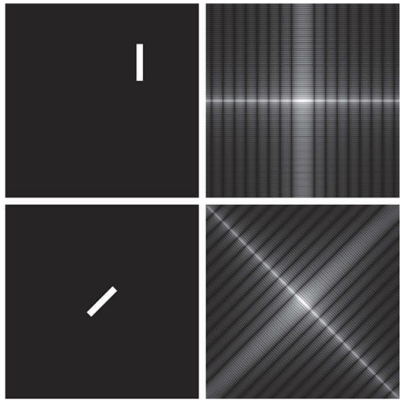


FIGURE 4.24
(a) The rectangle in Fig. 4.23(a) translated. (b) Corresponding spectrum. (c) Rotated rectangle. (d) Corresponding spectrum. The spectrum of the translated rectangle is identical to the spectrum of the original image in Fig. 4.23(a) .

a b c

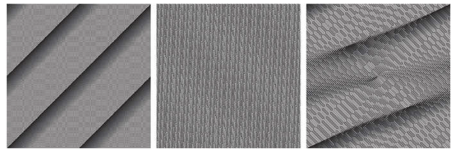
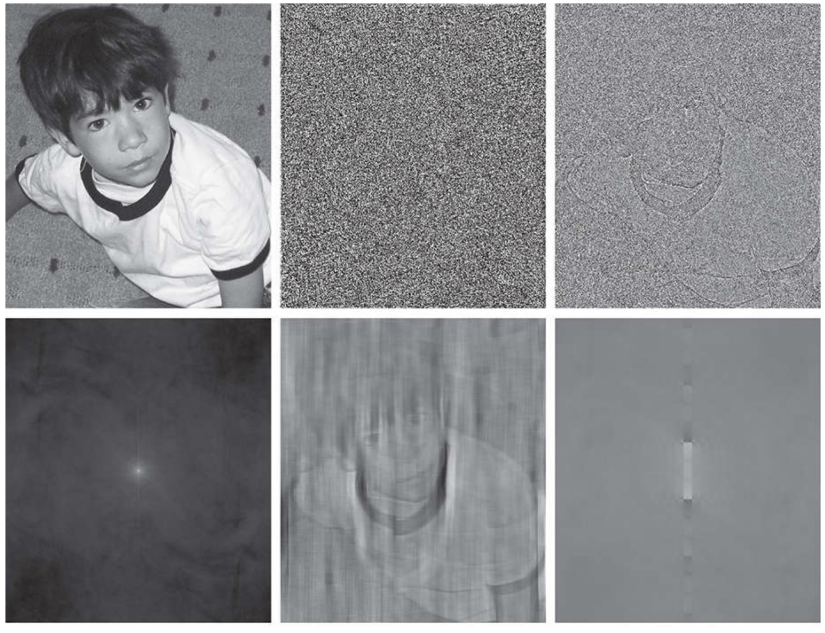


FIGURE 4.25
Phase angle images of (a) centered, (b) translated, and (c) rotated rectangles.

Examples of Fourier spectra

Spectrum contains intensity information, phase contains shape information.

$F(0,0) = \sum_x \sum_y f(x,y)e^{-2\pi i 0} = \sum_x \sum_y f(x,y)$ is proportional to the average intensity of the image (a.k.a. DC coefficient).



a b c
d e f

FIGURE 4.26
(a) Boy image. (b) Phase angle. (c) Boy image reconstructed using only its phase angle (all shape features are there, but the intensity information is missing because the spectrum was not used in the reconstruction). (d) Boy image reconstructed using only its spectrum. (e) Boy image reconstructed using its phase angle and the spectrum of the rectangle in Fig. 4.23(a) . (f) Rectangle image reconstructed using its phase and the spectrum of the boy's image.

Filtering in the frequency domain

Filtering is achieved in 3 steps:

1. Compute the Fourier transform F of an image f
2. Modify the Fourier transform, leading to F'
3. Compute the inverse Fourier transform of F', its real part leading to a modified image f'

The modification in the Fourier domain is done with a filter transfer function H, such as $F'(u,v) = H(u,v)F(u,v)$ (using elementwise multiplication).

Defining H is easier if we consider functions that are symetric about their center, requiring F to be centered too (easing displaying F too). This can be obtained by multiplying the input image by $(-1)^{x+y}$ prior computing the DFT.

Since $F(0,0)$ is usually much larger than other coefficient, the spectrum can be shown using $\log(1 + |F(u,v)|)$.

Low-pass Filtering

Let us consider centered coordinates: $D(u,v) = \sqrt{\left(u - \frac{M}{2}\right)^2 + \left(v - \frac{N}{2}\right)^2}$

Main filters:

- Ideal filter: $H(u,v) = \begin{cases} 1 & \text{if } D(u,v) \leq D_0 \\ 0 & \text{otherwise} \end{cases}$
- Butterworth filter (of order n): $H(u,v) = \frac{1}{1+\left(\frac{D(u,v)}{D_0}\right)^{2n}}$
- Gaussian filter: $H(u,v) = e^{-\frac{D^2(u,v)}{2\sigma^2}}$, with $\sigma = D_0$: $H(u,v) = e^{-\frac{D^2(u,v)}{2D_0^2}}$

High-pass Filtering

High-pass filters are opposed to low-pass filters, $H_{\text{high}}(u,v) = 1 - H_{\text{low}}(u,v)$

Main filters:

- Ideal filter: $H(u,v) = \begin{cases} 0 & \text{if } D(u,v) \leq D_0 \\ 1 & \text{otherwise} \end{cases}$
- Butterworth filter (of order n): $H(u,v) = \frac{1}{1+\left(\frac{D_0}{D(u,v)}\right)^{2n}}$
- Gaussian filter: $H(u,v) = 1 - e^{-\frac{D^2(u,v)}{2D_0^2}}$

Laplacian can be defined as $H(u,v) = -(u^2+v^2)$,
or if F is centered $H(u,v) = -\left(\left(u-\frac{M}{2}\right)^2 + \left(v-\frac{N}{2}\right)^2\right)$

Contrast can then be enhanced with
$H(u,v) = 1 - \left(\left(u-\frac{M}{2}\right)^2 + \left(v-\frac{N}{2}\right)^2\right)$,
or with the high-boost filter $H(u,v) = (A-1) + H_{\text{high}}(u,v)$

Bandreject and bandpass Filtering

Instead of removing only low or high frequencies, we can focus on a specific range of frequencies $D_0 \pm W$.

Main bandreject filters H_{reject}:

- Ideal filter: $H(u,v) = \begin{cases} 1 & \text{if } D(u,v) < D_0 - \frac{W}{2} \\ 0 & \text{if } D_0 - \frac{W}{2} \leq D(u,v) \leq D_0 + \frac{W}{2} \\ 1 & \text{if } D(u,v) > D_0 + \frac{W}{2} \end{cases}$
- Butterworth filter (of order n): $H(u,v) = \frac{1}{1+\left(\frac{D(u,v)W}{D^2(u,v)-D_0^2}\right)^{2n}}$
- Gaussian filter: $H(u,v) = 1 - e^{-\frac{1}{2}\left(\frac{D^2(u,v)-D_0^2}{D(u,v)W}\right)^2}$

Bandpass filterings can be implemented straightforwardly:
$H_{\text{pass}}(u,v) = 1 - H_{\text{reject}}(u,v)$

Notch filters

Notch filters (or cut-band filters) are the most useful selective filters, rejecting (or passing) frequencies in a predefined neighbourhood of the frequency domain.

Let us consider (u_0, v_0) the frequency to reject, we have:

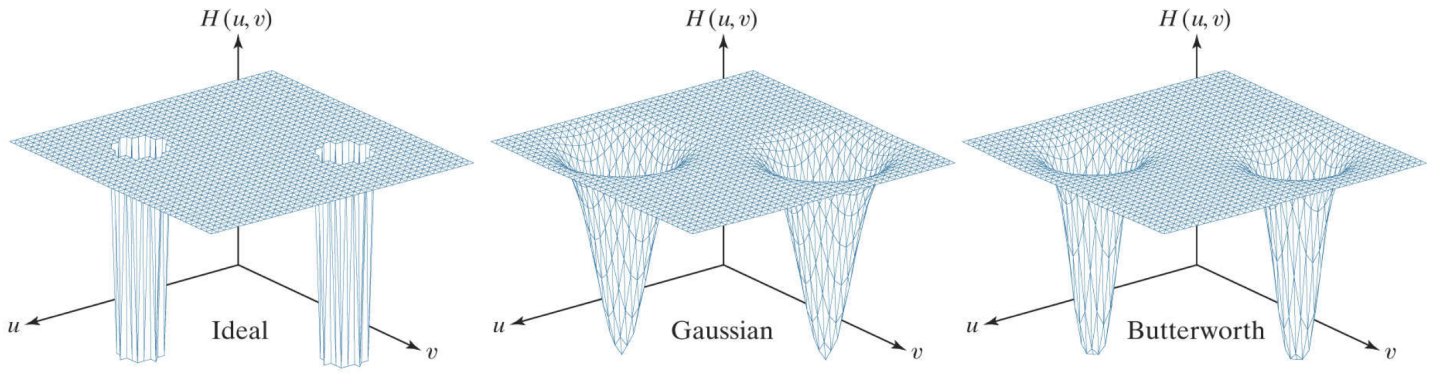
$$D_1(u,v) = \sqrt{\left(u - \frac{M}{2} - u_0\right)^2 + \left(v - \frac{N}{2} - v_0\right)^2}$$
$$D_2(u,v) = \sqrt{\left(u - \frac{M}{2} + u_0\right)^2 + \left(v - \frac{N}{2} + v_0\right)^2}$$

Main notch filters H_{notch}:

- Ideal filter: $H(u,v) = \begin{cases} 0 & \text{if } D_1(u,v) \leq D_0 \text{ or } D_2(u,v) \leq D_0 \\ 1 & \text{otherwise} \end{cases}$
- Butterworth filter (of order n): $H(u,v) = \frac{1}{1 + \left(\frac{D_0^2}{D_1(u,v) - D_2(u,v)}\right)^{2n}}$
- Gaussian filter: $H(u,v) = 1 - e^{-\frac{1}{2}\left(\frac{D_1(u,v) - D_2(u,v)}{D_0^2}\right)^2}$

Bandpass filterings are obtained through $H_{\text{pass}}(u,v) = 1 - H_{\text{notch}}(u,v)$

Notch filters (cont.)



a b c

FIGURE 5.15
Perspective plots of (a) ideal, (b) Gaussian, and (c) Butterworth notch reject filter transfer functions.

Notch filters (cont.)

a b
c d

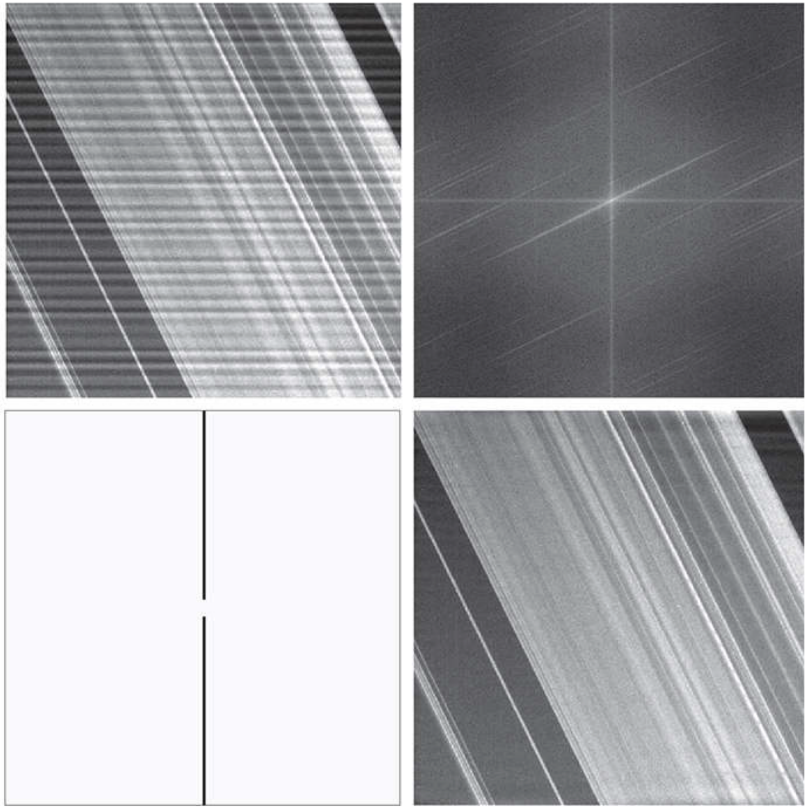


FIGURE 4.65
(a) Image of Saturn rings showing nearly periodic interference. (b) Spectrum. (The bursts of energy in the vertical axis near the origin correspond to the interference pattern). (c) A vertical notch reject filter transfer function. (d) Result of filtering. (The thin black border in (c) is not part of the data.)
(Original image courtesy of Dr. Robert A. West, NASA/JPL.)

Image denoising & restoration

Perfect images (noiseless) are neither available!

Most often the imaging sensor acquires a degraded/noisy representation of the observed scene.

Prior knowledge of the kind of degradation (a.k.a. noise model) is valuable to perform denoising (a.k.a. restoration).

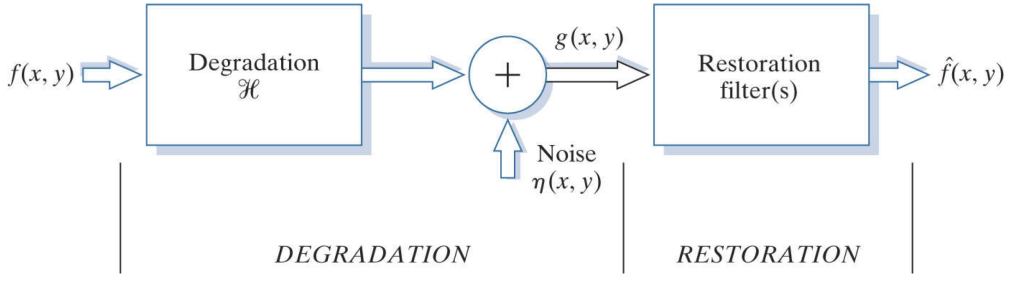
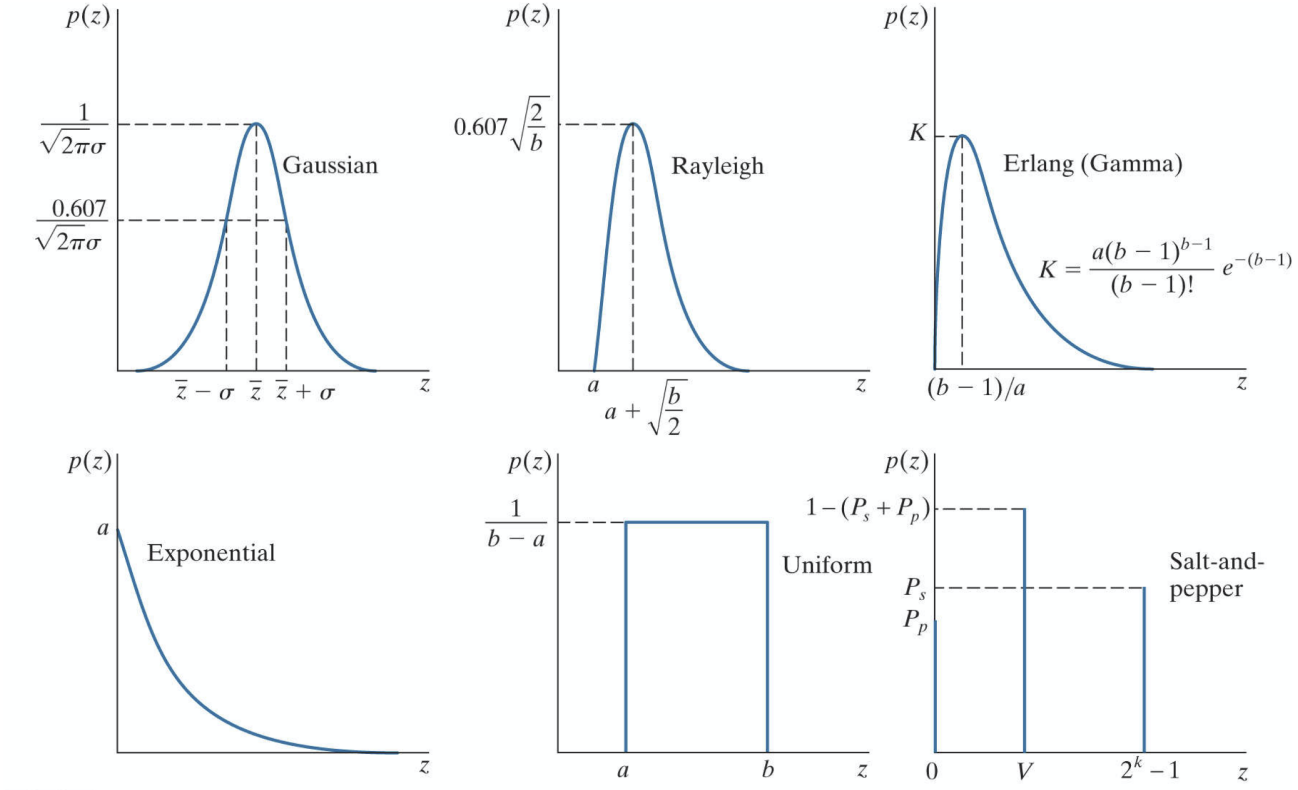


FIGURE 5.1
A model of the image degradation/restoration process.

Noise models



a b c
d e f

FIGURE 5.2
Some important probability density functions.

Noise models

- Gaussian noise: $p(z) = \frac{1}{\sqrt{2\pi}\sigma} e^{-(z-\mu)^2/2\sigma^2}$
- Rayleigh noise: $p(z) = \begin{cases} \frac{2}{b}(z-a)e^{-(z-a)^2/b} & \text{if } z \geq a \\ 0 & \text{otherwise} \end{cases}$
- Erlang (or Gamma) noise: $p(z) = \begin{cases} \frac{a^b z^{b-1}}{(b-1)!} e^{az} & \text{if } z \geq 0 \\ 0 & \text{otherwise} \end{cases}$
- Exponential noise: $p(z) = \begin{cases} ae^{az} & \text{if } z \geq 0 \\ 0 & \text{otherwise} \end{cases}$
- Uniform noise: $p(z) = \begin{cases} \frac{1}{b-a} & \text{if } a \leq z \leq b \\ 0 & \text{otherwise} \end{cases}$
- Impulse noise (a.k.a. salt-and-pepper): $p(z) = \begin{cases} P_a & \text{if } z = a \\ P_b & \text{if } z = b \\ 0 & \text{otherwise} \end{cases}$

From noise to denoising

Knowing the noise model (noise type & parameters) allows to select the appropriate denoising method and to tune it adequately:

- periodic noise is easily observable in the Fourier domain (see corresponding lecture)
- acquiring reference data (with no objects within) allows to obtain reference histogram to be compared with the noise pdf.
- then, compute the mean and the standard deviation, and possibly subsequent noise parameters.

Spatial denoising filters (reminder)

Mean filters

- arithmetic mean filter
- geometric mean filter
- harmonic mean filter
- contraharmonic mean filter

Order-statistic filters

- median filter
- max and min filter
- midpoint filter
- alpha-trimmed mean filter

Adaptive filters

Frequency denoising filters (NEW)

A degraded image is often expressed as $g(i,j) = (f * h)(i,j) + \nu(i,j)$.

If the noise ν is not significant, then restoration equates inverse convolution (a.k.a. deconvolution), and we have $G = HF$ in the Fourier domain, e.g.

- relative motion of camera and object, $H(u,v) = \frac{\sin(\pi VTu)}{\pi Vu}$
  with T the shutter time and V the speed
- wrong lens focus, $H(u,v) = \frac{J_1(ar)}{ar}$
  with J_1 the Bessel function of the first order, $r^2 = u^2 + v^2$ and a the displacement
- atmospheric turbulence, $H(u,v) = e^{-c(u^2+v^2)^{5/6}}$
  with c a constant that depends on the type of turbulence.

Frequency denoising filters (NEW)

If the noise cannot be negliged, then the inverse convolution is solved as an overdetermined system of linear equations, using e.g. least square error (Wiener filtering) or Kalman filtering.

We have then $G(u,v) = F(u,v)H(u,v) + N(u,v)$, leading to inverse filtering $\hat{F}(u,v) = G(u,v)H^{-1}(u,v) - N(u,v)H^{-1}(u,v)$, raising two issues: N, additive noise, is possibly unknown; noise influence may become significant for frequencies where $H(u,v)$ has small magnitude.

If inverse filtering is not possible... then comes Wiener filtering aiming to minimize the MSE between the restored and original (unknown) image

$$\hat{F}(u,v) = \left(\frac{1}{H(u,v)} \frac{|H(u,v)|^2}{|H(u,v)|^2 + K(u,v)} \right) G(u,v)$$

with $K(u,v) = (|N(u,v)|^2 / |F(u,v)|^2)$ set to a constant scalar (or based on priors on $|N(u,v)|^2$ and $|F(u,v)|^2$)

Frequency denoising filters

More advanced filterings

- Constrained least square filtering

$$\hat{F}(u,v) = \left(\frac{H^*(u,v)}{|H(u,v)|^2 + \gamma |P(u,v)|^2} \right) G(u,v)$$

with γ to be found by an iterative algorithm

$P(u,v)$ the Fourier transform of $p(x,y) = \begin{pmatrix} 0 & -1 & 0 \\ -1 & 4 & -1 \\ 0 & -1 & 0 \end{pmatrix}$

H^* complex conjugate of H

- Geometric mean filter

$$\hat{F}(u,v) = \left(\frac{H^*(u,v)}{|H(u,v)|^2} \right)^{\alpha} \left(\frac{H^*(u,v)}{|H(u,v)|^2 + \beta (|N(u,v)|^2 / |F(u,v)|^2)} \right)^{1-\alpha} G(u,v)$$

with α, β nonnegative, real constants.

FFT beyond filtering?

30 years ago:

- Watson (1993) Processing remote sensing images using the 2-D FFT - Noise reduction and other applications

More recently:

- Tong (2019) Image Registration With Fourier-Based Image Correlation: A Comprehensive Review of Developments and Applications
- Teillet (2021) Fast Unsupervised Multi-Scale Characterization of Urban Landscapes Based on Earth Observation Data

Let us go through these papers together (stored on moodle)

Practice

FFT and 2D FFT are available in `numpy.fft`:

- `fft` and `ifft` for 1D signals
- `fft2` and `ifft2` for 2D signals (images)
- `fftshift` to shift the FFT in order $F(0,0)$ to be centered

Example 1

```
a=np.hstack([zeros((256,128)),ones((256,128))])
af=fftshift(fft2(a))
afl=abs(af)
io.imshow(afl)
```

Example 2

```
import skimage.exposure as ex
a=zeros((256,256))
a[77:177,77:177]=1
af=fftshift(fft2(a))
afl=ex.rescale_intensity(np.log(1+abs(af)),out_range=(0.0,1.0))
io.imshow(afl)
```

Labs

1. Load an image
2. Verify the invertibility of the FFT: $I = IFFT(FFT(I))$
3. Compare the effect of the main low-pass filters (w.r.t D_0) and show the FFT before and after filtering
4. Compare the effect of the main high-pass filters (w.r.t D_0) and show the FFT before and after filtering
5. Perform image sharpening in the frequency domain
6. Add some periodic noise in the frequential domain and apply notch filter to remove it

Labs (cont.)

1. Load a greyscale image
2. Generate some noisy versions by applying several noise models (with various parameters)
3. Perform noise reduction/image restoration

EDIT: Wiener filter available in `scipy.signal.wiener`