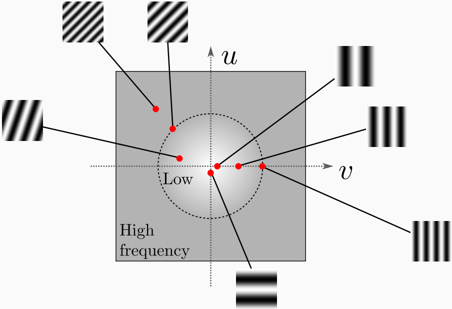
ECE 285

Image and video restoration

Chapter III – Basics of filtering II

Charles Deledalle

May 30, 2019

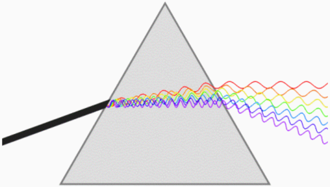


Basics of filtering

Standard filters

Two main approaches:

- **Spatial domain:** use the pixel grid / spatial neighborhoods
- **Spectral domain:** use Fourier transform, cosine transform, . . .

Spatial  Spectral

Spectral filtering

Spectral filtering – Periodical functions

A sine wave (or sinusoidal) $f(t) = a\cos(2\pi u t + \varphi)$ is periodical

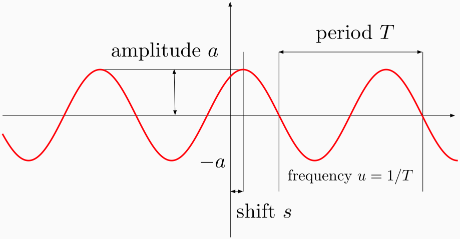
$$f(t+T) = f(t) \quad \text{for} \quad T = 1/u, \quad \text{for all} \quad t \in \mathbb{R}$$

and characterized by

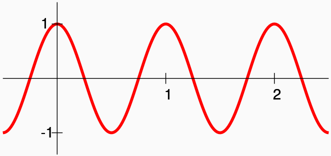
- u : frequency ($u = 1/T$)
- a : amplitude
- φ : phase ($\varphi = -2\pi u s$)

where

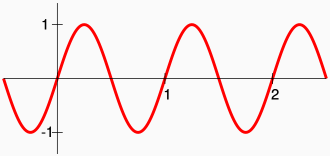
- T : period
- s : shift



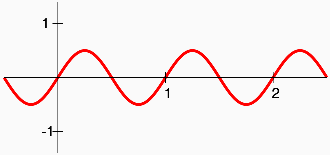
Spectral filtering – Periodical functions



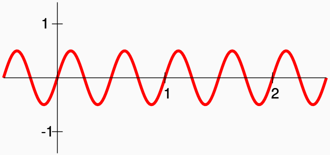
(a) $a = 1$, $u = 1$, $\varphi = 0$



(b) $a = 1$, $u = 1$, $\varphi = 3\pi/2$

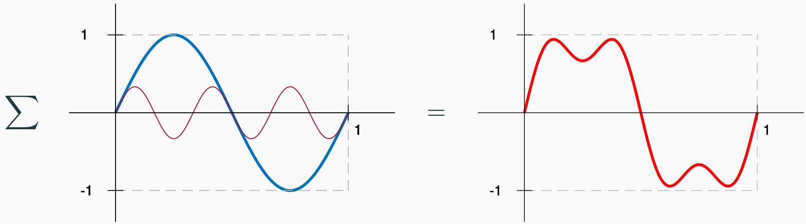


(c) $a = 1/2$, $u = 1$, $\varphi = 3\pi/2$



(d) $a = 1/2$, $u = 2$, $\varphi = 3\pi/2$

Figure 1 – Simple periodical signals



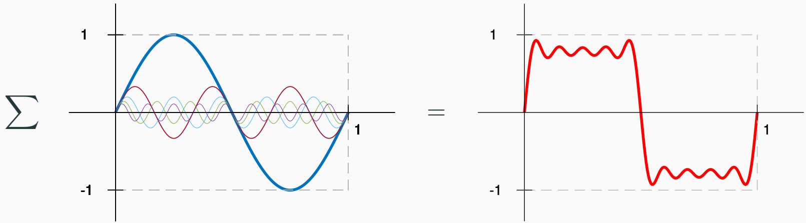
$u_1 = 1, a_1 = 1, \varphi_1 = 3\pi/2$ $\qquad$ $u_2 = 3, a_2 = 1/3, \varphi_2 = 3\pi/2$

Figure 2 – A complex periodical signal as the sum of simple ones

---

$$f(t) = a_1 \cos(2\pi u_1 t + \varphi_1) + a_2 \cos(2\pi u_2 t + \varphi_2)$$

Spectral filtering – Periodical functions



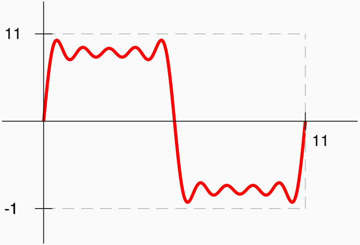
$$u_1 = 1, a_1 = 1, \varphi_1 = 3\pi/2 \qquad u_2 = 3, a_2 = 1/3, \varphi_2 = 3\pi/2$$
$$u_3 = 5, a = 1/5, \varphi_1 = 3\pi/2 \qquad u_4 = 7, a_2 = 1/7, \varphi_2 = 3\pi/2$$
$$u_5 = 9, a_2 = 1/9, \varphi_2 = 3\pi/2$$

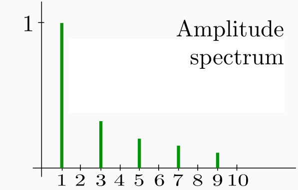
Figure 2 – A complex periodical signal as the sum of simple ones

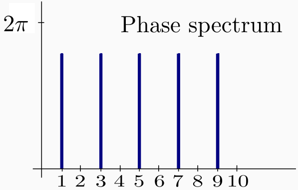
---

$$f(t) = \sum_{k=1}^{5} a_k \cos(2\pi u_k t + \varphi_k)$$

Spectral filtering – Periodical functions

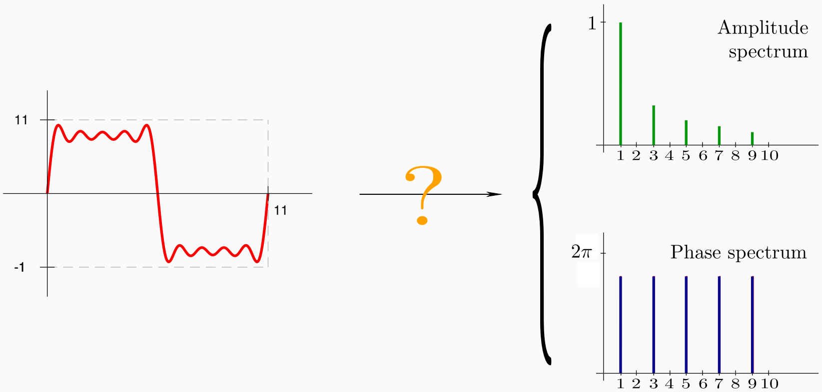






$1 \mapsto (1, 3\pi/2)$	$2 \mapsto (0, 0)$
$3 \mapsto (1/3, 3\pi/2)$	$4 \mapsto (0, 0)$
$5 \mapsto (1/5, 3\pi/2)$	$6 \mapsto (0, 0)$
$7 \mapsto (1/7, 3\pi/2)$	$8 \mapsto (0, 0)$
$9 \mapsto (1/9, 3\pi/2)$	$10 \mapsto (0, 0)$

The function $u \mapsto (a_u, \varphi_u)$ characterizes f



How to change representation?

Spectral filtering – Fourier transform

Jean Baptiste Joseph Fourier

Figure 3 – (left) Sketch of Fourier by Julien Léopold Boilly. (right) Bust of Fourier at Musée de l'Ancien Évêché in Grenoble, France.

Fourier series

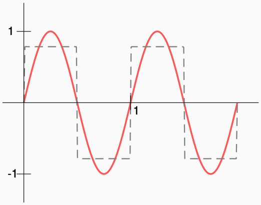
- Let $f : \mathbb{R} \to \mathbb{R}$ be a T-periodical function, i.e.,

$$f(t+T) = f(t), \quad \text{for all} \quad t \in \mathbb{R}$$

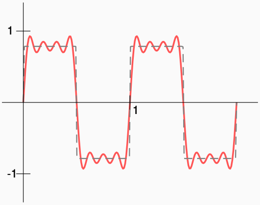
  with $T > 0$ as small as possible.
- Denote by $u = 1/T$ the fundamental frequency.
- Then, under only mild assumptions on f, we have

$$f(t) = \frac{a_0}{2} + \sum_{k=1}^{\infty} a_k \cos\left(2\pi u_k t + \varphi_k\right) \quad \text{with} \quad u_k = u \cdot k$$

- The frequencies $u_k = u \cdot k$ are called harmonics.
- The coefficients (a_k, φ_k) associated to the harmonic u_k characterize f.



(a) $k = 1$

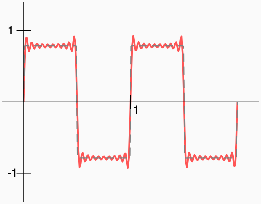


(b) $k = 1$ to 4

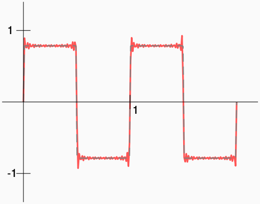
Discontinuity

$\equiv$

Infinite number of harmonics



(c) $k = 1$ to 10



(d) $k = 1$ to 20

$$f(t) = \frac{a_0}{2} + \sum_{k=1}^{\infty} a_k \cos\left(2\pi ukt + \varphi_k\right)$$

Complex formulation

- Using Euler's formula: $\cos(x) = \frac{e^{ix}+e^{-ix}}{2}$ (i imaginary number: $i^2 = -1$)

$$\begin{aligned} f(t) &= \frac{a_0}{2} + \sum_{k=1}^{\infty} \frac{a_k}{2}\left(e^{i(2\pi ukt+\varphi_k)} + e^{-i(2\pi ukt+\varphi_k)}\right) \\ &= \sum_{k=-\infty}^{-1} \underbrace{\frac{a_{|k|}e^{-i\varphi_{|k|}}}{2}}_{c_k} e^{i2\pi ukt} + \underbrace{\frac{a_0}{2}e^{i2\pi u0t}}_{c_0} + \sum_{k=1}^{\infty} \underbrace{\frac{a_{|k|}e^{i\varphi_{|k|}}}{2}}_{c_k} e^{i2\pi ukt} \\ &= \sum_{k=-\infty}^{+\infty} c_k e^{i2\pi ukt} \quad \text{with } \varphi_0 = 0. \end{aligned}$$

- Coefficients $c_k = \frac{1}{2}a_{|k|}e^{\mathrm{sign}(k)i\varphi_{|k|}} \in \mathbb{C}$ encode a_k and φ_k

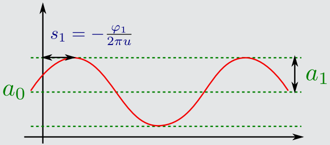
  $\Rightarrow$ They characterize f.

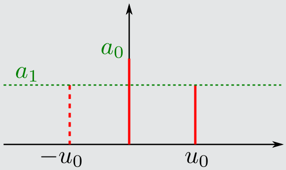
- They are called **Fourier coefficients**.

$$f(t) = \sum_{k=-\infty}^{+\infty} c_k e^{i2\pi ukt}$$

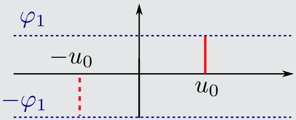
Negative frequencies

- Introduction of negative frequencies





- As $c_k = \frac{1}{2} a_{|k|} e^{\mathrm{sign}(k) i \varphi_{|k|}}$
  - We have $c_k = c^*_{-k}$
  - Amplitude spectrum: symmetrical
  - Phase spectrum: anti-symmetrical
  - Complex spectrum: Hermitian



- f as complex values: $f(t) \in \mathbb{C} \backslash \mathbb{R} \Leftrightarrow$ non-Hermitian complex spectrum.

Spectral filtering – Fourier transform – Periodical functions

$$f(t) = \sum_{k=-\infty}^{+\infty} c_k e^{i2\pi ukt}$$

Why the complex formulation?

$$\begin{aligned} f(t) &= (\alpha f_1 + \beta f_2)(t) \\ &= \alpha f_1(t) + \beta f_2(t) \\ &= \alpha \sum_{k=-\infty}^{+\infty} (c_1)_k e^{i2\pi ukt} + \beta \sum_{k=-\infty}^{+\infty} (c_2)_k e^{i2\pi ukt} \\ &= \sum_{k=-\infty}^{+\infty} (\alpha c_1 + \beta c_2)_k e^{i2\pi ukt} \end{aligned}$$

As the coefficients c characterized f, by identification:

$$c = \alpha c_1 + \beta c_2$$

Linear change of representation $\Rightarrow$ Change of basis

$$f(t) = \sum_{k=-\infty}^{+\infty} c_k e^{i2\pi ukt} = \sum_{k=-\infty}^{+\infty} c_k a_k(t)$$

Fourier atoms

- Functions: $a_k(t) = e^{i2\pi ukt}$, for $k \in \mathbb{Z}$.
- They are orthogonal to each other, for $k \neq l$:

$$\underbrace{\langle a_k,\, a_l \rangle}_{\text{scalar product for periodical functions}} = \int_{-T/2}^{T/2} a_k(t) a_l^*(t) \mathrm{d}t = 0$$

- They have the same finite norm:

$$\|a_k\|_2^2 = \int_{-T/2}^{T/2} a_k(t) a_k^*(t) \mathrm{d}t = T$$

- In particular: $a_k \neq 0$

Proof.

- Remark that, for $k \neq l$, a_k and a_l satisfy

$$\begin{aligned}
\underbrace{\langle a_k,\, a_l \rangle}_{\substack{\text{scalar product for} \\ \text{periodical function}}} &= \int_{-T/2}^{T/2} a_k(t) a_l^*(t) \mathrm{d}t \\
&= \int_{-T/2}^{T/2} e^{i2\pi ukt} e^{-i2\pi ult} \mathrm{d}t \\
&= \int_{-T/2}^{T/2} e^{i2\pi u(k-l)t} \mathrm{d}t \\
&= \left[\frac{e^{i2\pi u(k-l)t}}{i2\pi u(k-l)} \right]_{-T/2}^{T/2} \\
&= \frac{e^{i\pi(k-l)} - e^{-i\pi(k-l)}}{i2\pi u(k-l)} && \text{(Since } T = 1/u\text{)} \\
&= \frac{\sin(\pi(k-l))}{\pi u(k-l)} = 0 && \text{(Since } k - l \in \mathbb{Z}\text{)}
\end{aligned}$$

$\square$

Proof.

- Moreover for all k

$$\begin{aligned}\langle a_k,\, a_k\rangle &= \int_{-T/2}^{T/2} a_k(t)a_k^*(t)\mathrm{d}t \\ &= \int_{-T/2}^{T/2} e^{i2\pi ukt}e^{-i2\pi ukt}\mathrm{d}t \\ &= \int_{-T/2}^{T/2} \mathrm{d}t \\ &= T\end{aligned}$$

□

$$f(t) = \sum_{k=-\infty}^{+\infty} c_k e^{i2\pi ukt} = \sum_{k=-\infty}^{+\infty} c_k a_k(t)$$

Fourier basis

(1) Complex Fourier series:

all T-periodical functions are linear combinations of Fourier atoms a_k.

(2) Fourier atoms satisfy:

$$a_k \neq 0 \text{ and } \langle a_k, a_l \rangle = 0 \text{ for } k \neq l$$

(1)+(2) $\Rightarrow$

Fourier atoms form an orthogonal basis for T-periodical functions called **Fourier basis**.

What are the consequences?

We can compute the coefficient c_k

- Since (a_k) form an orthogonal basis for T-periodical functions:

$$f(t) = \sum_{k=-\infty}^{+\infty} \frac{\langle f, a_k \rangle}{\|a_k\|_2^2} a_k(t) = \sum_{k=-\infty}^{+\infty} \left(\frac{1}{T} \int_{-T/2}^{+T/2} f(t') e^{-i2\pi ukt'} \mathrm{d}t' \right) e^{i2\pi ukt}$$

- By identification

$$c_k = \underbrace{\frac{1}{T} \int_{-T/2}^{+T/2} f(t) e^{-i2\pi ukt} \mathrm{d}t}_{\mathcal{F}[f]_k} \qquad \text{(Fourier transform)}$$

- and the operation is invertible and corresponds to the Fourier series

$$f(t) = \underbrace{\sum_{k=-\infty}^{+\infty} c_k e^{i2\pi ukt}}_{\mathcal{F}^{-1}[c_k](t)} \qquad \text{(inverse Fourier transform)}$$

Non-periodical functions

- If f is non-periodical: no more fundamental frequency
- Cannot be characterized only by the harmonics: $\ldots, -2u, -u, 0, u, 2u, \ldots$
- Require a continuum of frequencies: all possible $u \in \mathbb{R}$
- Under mild assumptions on f, we get similar transforms

$$\underbrace{\hat{f}(u) = \mathcal{F}[f](u) = \int_{-\infty}^{+\infty} f(t) e^{-i2\pi ut} \mathrm{d}t}_{\text{Fourier transform}}$$

and $$\underbrace{f(t) = \mathcal{F}^{-1}[\hat{f}](t) = \int_{-\infty}^{+\infty} \hat{f}(u) e^{i2\pi ut} \mathrm{d}u}_{\text{inverse Fourier transform}}$$

Why does it matter?

It helps at simplifying calculus,

e.g., eases to find solutions of differential equations.

Spectral filtering – Discrete Fourier Transform (DFT)

Discrete signals

- Let $f \in \mathbb{R}^n$ be a discrete signal
- Consider it to be periodical: $f_{k+n} = f_k$
- It can be characterized only by its n harmonics of the form:

$$\frac{-\lceil n/2 \rceil + 1}{n}, \ldots, -\frac{2}{n}, -\frac{1}{n}, 0, \frac{1}{n}, \frac{2}{n}, \ldots, \frac{\lfloor n/2 \rfloor}{n}$$

- The discrete Fourier transforms (DFT) is thus given by

$$\underbrace{\hat{f}_u = \mathcal{F}[f]_u = \sum_{k=0}^{n-1} f_k e^{-i2\pi \frac{uk}{n}},}_{\text{Discrete Fourier transform}} \quad u = 0 \ldots n-1$$

and $$\underbrace{f_k = \mathcal{F}^{-1}[\hat{f}]_k = \frac{1}{n} \sum_{u=0}^{n-1} \hat{f}_u e^{i2\pi \frac{uk}{n}},}_{\text{inverse DFT}} \quad k = 0 \ldots n-1$$

Why does it matter? **It allows us to do signal processing**.

Spectral filtering – 2d DFT

Discrete images

- Let $f \in \mathbb{R}^{n_1 \times n_2}$ be a discrete image
- Consider it to be periodical: $f_{k+n_1, l+n_2} = f_{k,l}$
- The 2d discrete Fourier transforms (DFT) is thus given by

$$\underbrace{\hat{f}_{u,v} = \mathcal{F}[f]_{u,v} = \sum_{k=0}^{n_1-1} \sum_{l=0}^{n_2-1} f_{k,l} e^{-i2\pi\left(\frac{uk}{n_1} + \frac{vl}{n_2}\right)}}_{\text{2D DFT}}$$

and $$\underbrace{f_{k,l} = \mathcal{F}^{-1}[\hat{f}]_{k,l} = \frac{1}{n_1 n_2} \sum_{u=0}^{n_1-1} \sum_{v=0}^{n_2-1} \hat{f}_{u,v} e^{i2\pi\left(\frac{uk}{n_1} + \frac{vl}{n_2}\right)}}_{\text{inverse 2D DFT}}$$

- The pair (u, v) represents a two-dimensional frequency.

What does it look like?

Spectral filtering – 2d DFT

- Each point (u, v) in the Fourier domain corresponds to a sine "wave" of frequency $\sqrt{u^2 + v^2}$ along the axis Δ directed by the vector (u, v)

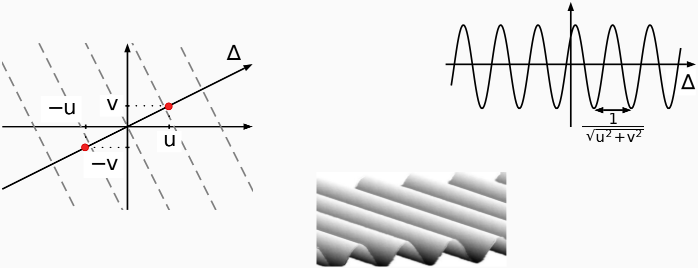


Figure 4 – 2D signals with spectrum limited only to frequencies (u, v) and $(-u, -v)$

$$= \frac{1}{n} \quad \hat{f}_{u_1,v_1} \cdot e^{i2\pi\left(\frac{u_1 k}{n_1}+\frac{v_1 l}{n_2}\right)} + \hat{f}_{u_1,v_2} \cdot e^{i2\pi\left(\frac{u_1 k}{n_1}+\frac{v_2 l}{n_2}\right)} + \hat{f}_{u_i,v_j} \cdot e^{i2\pi\left(\frac{u_i k}{n_1}+\frac{v_j k}{n_l}\right)} \vdots$$

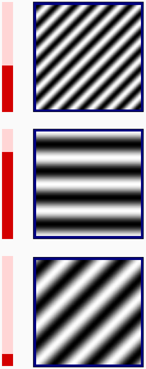
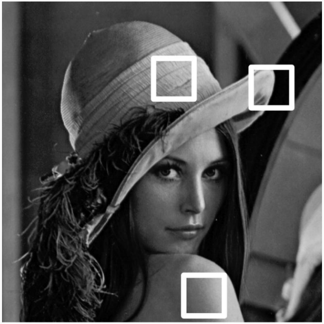


Image = weighted sum of sine waves

Spectral filtering – 2d DFT

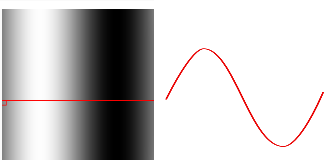
- In practice: all frequencies are more or less used in different regions



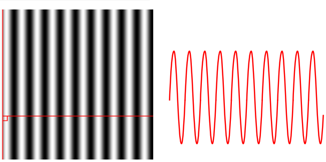
Which kinds of frequencies are used in the white squares?

Spectral filtering – 2d DFT

- Spatial frequency: measures how fast the image varies in a given direction



Low frequency (slow variations)

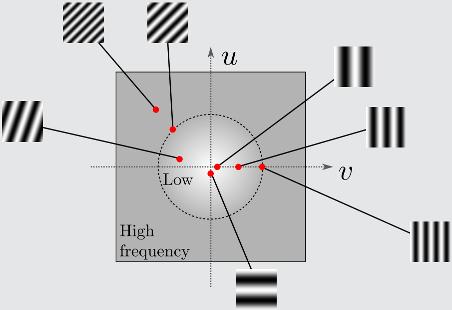


High frequency (fast variations)

How do we represent the Fourier coefficients?

Spectral filtering – 2d DFT

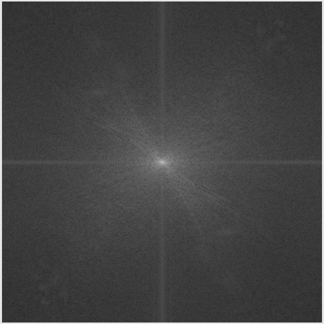
- Represent each Fourier coefficients on a 2d grid



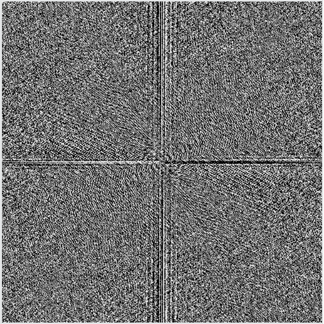
- $|\hat{f}_{u,v}|$: contribution of frequency $\sqrt{u^2+v^2}$ in the direction (u,v).
- $\arg \hat{f}_{u,v}$: phase shift of frequency $\sqrt{u^2+v^2}$ in the direction (u,v).
- Center $\equiv$ low frequencies
- Periphery $\equiv$ high frequencies

Example

$\rightarrow$



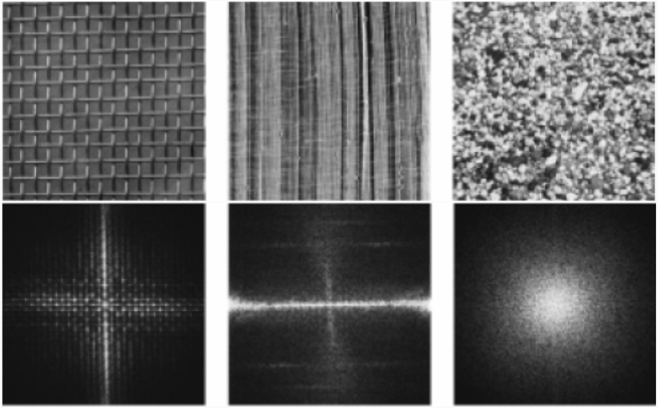
Amplitude



Phase

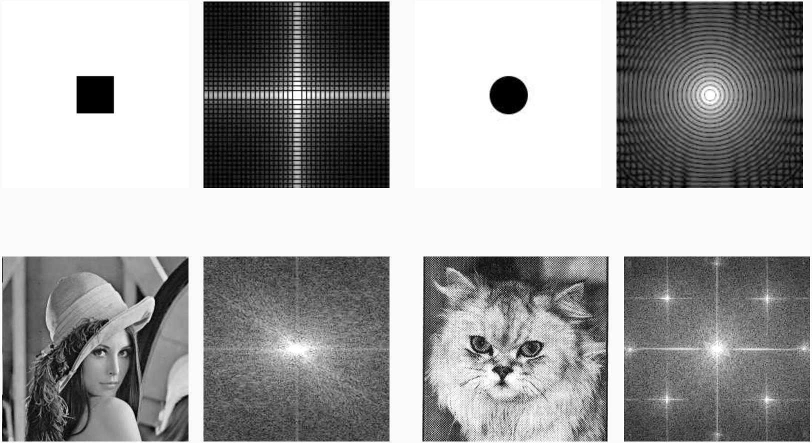
How to interpret it?

Spectral filtering – 2d DFT

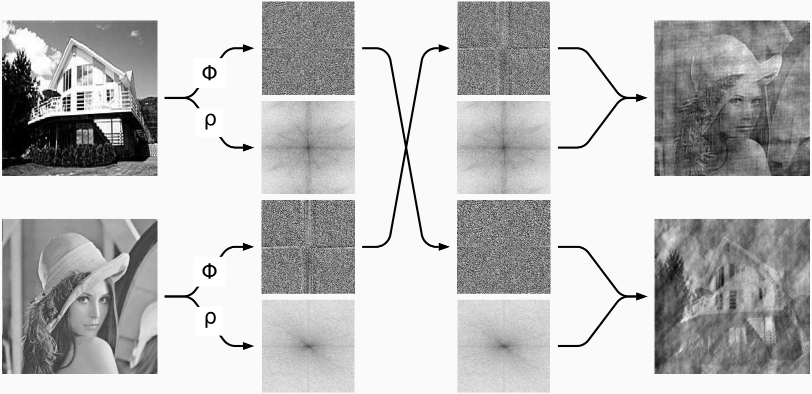


- Amplitude spectrum highlights the "directions" of a pattern
- Edge is represented by all harmonics in its orthogonal direction
- i.e., a line in the orthogonal direction (passing through the origin)

Spectral filtering – 2d DFT

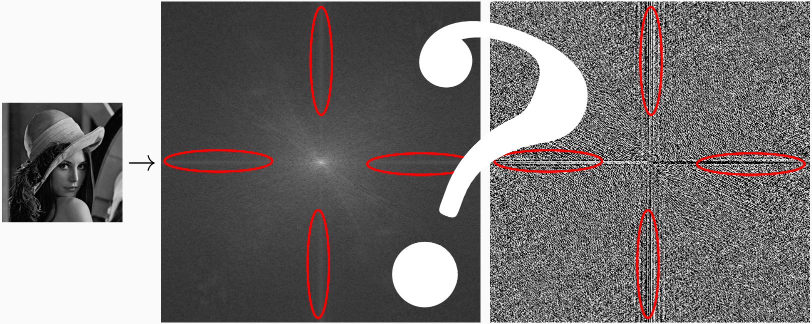


Spectral filtering – 2d DFT



- In general, we only represent the modulus
- Nevertheless, the phase encodes a large amount of information

Spectral filtering – 2d DFT



Amplitude Phase

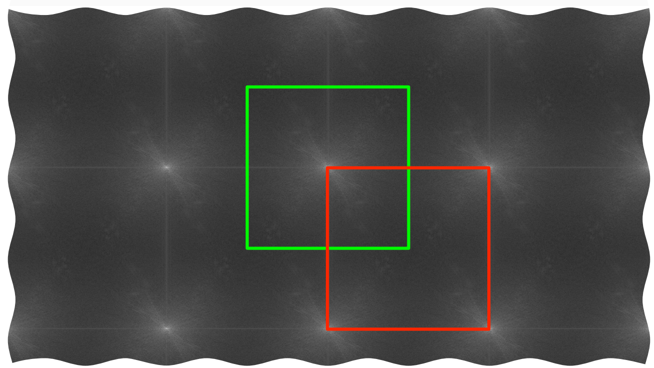
Why do the vertical and horizontal directions appear so strong?

$\longrightarrow$

Periodization

- It is assumed that the image is periodical
- Image borders may create strong edges
- Strong vertical and horizontal directions

Spectral filtering – 2d DFT

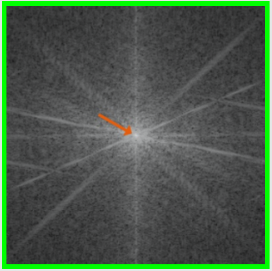


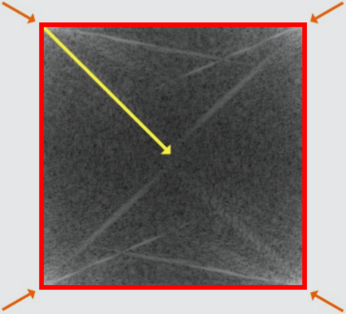
Periodization

- The spectrum is also periodical
- Different ways to represent it

Spectral filtering – 2d DFT

Recenter / Shift





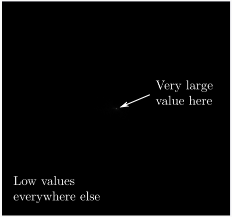
- Option 1: place the zero-frequency in the middle
  - Good way to visualize it
- Option 2: place the zero-frequency at top left location
  - Good way to manipulate it
  - Representation used by `Python`, `Matlab`, `fftw3`, ...

Visualization of the amplitude spectrum

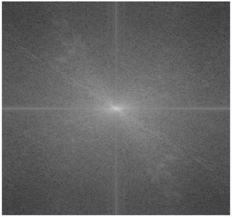
- Recall that $\hat{f}_{u,v} = \sum_{k=0}^{n_1-1} \sum_{l=0}^{n_2-1} f_{k,l} e^{-i2\pi\left(\frac{uk}{n_1}+\frac{vl}{n_2}\right)}$
- Then $\hat{f}_{0,0} = \sum_{k=0}^{n_1-1} \sum_{l=0}^{n_2-1} f_{k,l} = \sum$ of all intensities

  $\longleftarrow$ Can be very large!

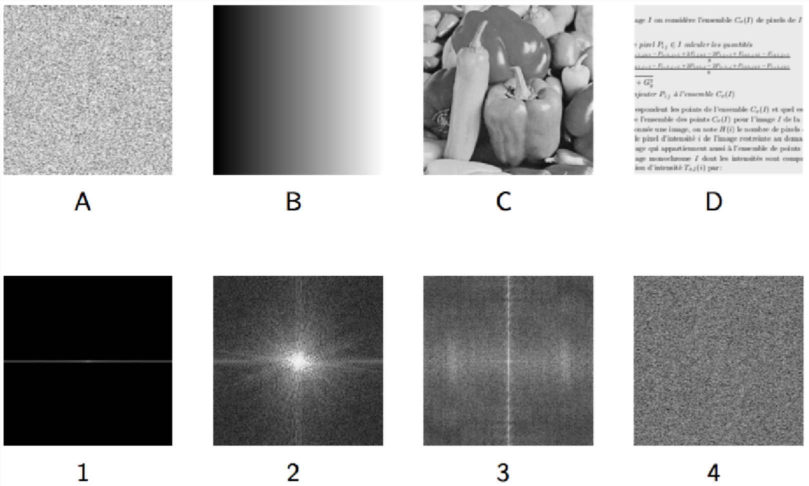
- **Consequence:** the dynamic is too large to be displayed correctly
- **Solution:** perform a punctual non-linear transform
- **Classical one:** use $\log(|\hat{f}_{u,v}| + \varepsilon)$, $\varepsilon > 0$



$\rightarrow$



Spectral filtering – 2d DFT



Which one is which?

Spectral filtering – Principle

Principle of spectral filtering

1. Apply the **Fourier transform**: $\hat{f} = \mathcal{F}[f]$
2. **Extract** the amplitude and phase

$$a_{u,v} = |\hat{f}_{u,v}| = \sqrt{\mathrm{Re}[\hat{f}_{u,v}]^2 + \mathrm{Im}[\hat{f}_{u,v}]^2}$$

$$\text{and} \quad \varphi_{u,v} = \arg \hat{f}_{u,v} = \mathrm{atan2}(\mathrm{Im}[\hat{f}_{u,v}], \mathrm{Re}[\hat{f}_{u,v}])$$

3. **Modify** the amplitude spectrum (and eventually the phase spectrum)

$$a_{u,v} \leftarrow a'_{u,v} \quad \text{and} \quad \varphi_{u,v} \leftarrow \varphi'_{u,v}$$

4. **Reconstruct** a complex spectrum

$$\hat{f}'_{u,v} = a'_{u,v} e^{i\varphi'_{u,v}}$$

5. Apply the **inverse Fourier transform**: $f' = \mathcal{F}^{-1}[\hat{f}']$

Useful only if we have a fast implementation of the Fourier transform

Spectral filtering – Fast Fourier Transform

Discrete Fourier Transform (DFT)

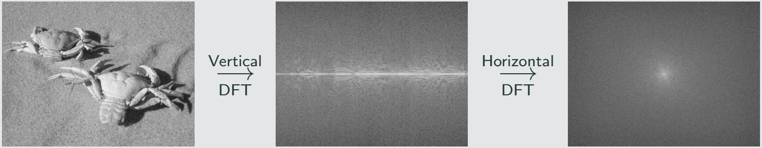
$$\hat{f}_u = \sum_{k=0}^{n-1} f_k e^{-i2\pi \frac{uk}{n}}$$

$\rightarrow$ Perform one loop for $u = 0$ to $n-1$

$\rightarrow$ Direct computation in $O(n^2)$

2d Discrete Fourier Transform (DFT2)

- The discrete Fourier transform is directionally separable



- Complexity in: $O(n_1 n_2^2 + n_2 n_1^2) = O(n(n_1 + n_2))$
- Best scenario $n_1 = n_2 = \sqrt{n}$: $O(n^{3/2})$

Spectral filtering – Fast Fourier Transform

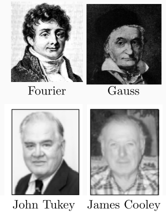
Fast Fourier Transform (FFT) [Cooley & Tukey, 1965]

- $\sim$1805: first described by Gauss (Fourier's paper: 1807)
- Exploits symmetry of DFT for faster computation
- Computation of the discrete Fourier transform can be done in

$$O(n \log n)$$

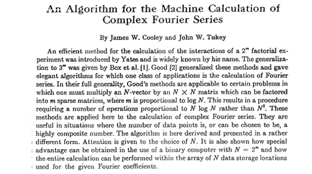
- Same for images thanks to directional separability

$$O(n_1 n_2 \log n_2 + n_2 n_1 \log n_1) = O(n(\log n_2 + \log n_1)) = O(n \log n)$$



Fourier Gauss

John Tukey James Cooley

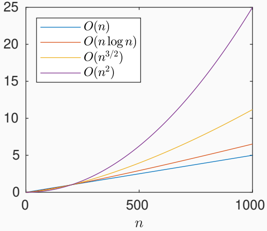


An Algorithm for the Machine Calculation of Complex Fourier Series

By James W. Cooley and John W. Tukey

An efficient method for the calculation of the interactions of a 2^m factorial experiment was introduced by Yates and is widely known by his name. The generalization to 3^m was given by Box et al. [1]. Good [2] generalized these methods and gave elegant algorithms for which one class of applications is the calculation of Fourier series. In their full generality, Good's methods are applicable to certain problems in which one must multiply an N-vector by an $N \times N$ matrix which can be factored into m sparse matrices, where m is proportional to $\log N$. This results in a procedure requiring a number of operations proportional to $N \log N$ rather than N^2. These methods are applied here to the calculation of complex Fourier series. They are useful in situations where the number of data points is, or can be chosen to be, a highly composite number. The algorithm is here derived and presented in a rather different form. Attention is given to the choice of N. It is also shown how special advantage can be obtained in the use of a binary computer with $N = 2^m$ and how the entire calculation can be performed within the array of N data storage locations used for the given Fourier coefficients.

J. W. Cooley and J. W. Tukey, Mathematics of Computation, Vol. 19, pp. 297-301, 1965.



(Source: Iasonas Kokkinos)

FFT: Top 10 Algorithms of 20th Century!

Society for Industrial and Applied Mathematics (SIAM)
The Best of the 20th Century: Editors NameTop 10 Algorithms
May 16, 2000 Barry A Cipra

- 1946: The Metropolis Algorithm for Monte Carlo. Through the use of random processes, this algorithm offers an efficient way to stumble toward answers to problems that are too complicated to solve exactly.
- 1947: Simplex Method for Linear Programming. An elegant solution to a common problem in planning and decision-making.
- 1950: Krylov Subspace Iteration Method. A technique for rapidly solving the linear equations that abound in scientific computation.
- 1951: The Decompositional Approach to Matrix Computations. A suite of techniques for numerical linear algebra.
- 1957: The Fortran Optimizing Compiler. Turns high-level code into efficient computer-readable code.
- 1959: QR Algorithm for Computing Eigenvalues. Another crucial matrix operation made swift and practical.
- 1962: Quicksort Algorithms for Sorting. For the efficient handling of large databases.
- 1965: Fast Fourier Transform. Perhaps the most ubiquitous algorithm in use today, it breaks down waveforms (like sound) into periodic components.
- 1977: Integer Relation Detection. A fast method for spotting simple equations satisfied by collections of seemingly unrelated numbers.
- 1987: Fast Multipole Method. A breakthrough in dealing with the complexity of n-body calculations, applied in problems ranging from celestial mechanics to protein folding.

(Source: Ioannos Kokkinos)

Spectral filtering – Low-pass filter

Python demo – Low-pass filter

```
import numpy.fft as nf
import imagetools as im

▶ f      = plt.imread('butterfly.png')
n1, n2 = f.shape
tf     = nf.fft2(f, axes=(0, 1))
a      = np.abs(tf)
phi    = np.angle(tf)
u, v   = im.fftgrid(n1, n2)
dist2  = u**2 + v**2
mask   = dist2 <= r**2
ap     = mask * a
tfp    = ap * np.exp(1j * phi)
fp     = np.real(nf.ifft2(tfp, axes=(0, 1)))
```

f

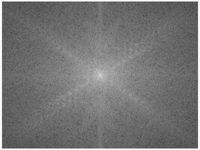
Spectral filtering – Low-pass filter

Python demo – Low-pass filter

```
import numpy.fft as nf
import imagetools as im

f      = plt.imread('butterfly.png')
n1, n2 = f.shape
tf     = nf.fft2(f, axes=(0, 1))
▶ a    = np.abs(tf)
phi    = np.angle(tf)
u, v   = im.fftgrid(n1, n2)
dist2  = u**2 + v**2
mask   = dist2 <= r**2
ap     = mask * a
tfp    = ap * np.exp(1j * phi)
fp     = np.real(nf.ifft2(tfp, axes=(0, 1)))
```

f



a

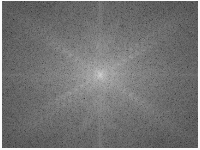
Spectral filtering – Low-pass filter

Python demo – Low-pass filter

```
import numpy.fft as nf
import imagetools as im

f      = plt.imread('butterfly.png')
n1, n2 = f.shape
tf     = nf.fft2(f, axes=(0, 1))
a      = np.abs(tf)
phi    = np.angle(tf)
▶ u, v   = im.fftgrid(n1, n2)
dist2  = u**2 + v**2
mask   = dist2 <= r**2
ap     = mask * a
tfp    = ap * np.exp(1j * phi)
fp     = np.real(nf.ifft2(tfp, axes=(0, 1)))
```

f



a

u

v

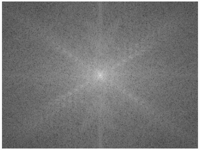
Spectral filtering – Low-pass filter

Python demo – Low-pass filter

```
import numpy.fft as nf
import imagetools as im

f      = plt.imread('butterfly.png')
n1, n2 = f.shape
tf     = nf.fft2(f, axes=(0, 1))
a      = np.abs(tf)
phi    = np.angle(tf)
u, v   = im.fftgrid(n1, n2)
dist2  = u**2 + v**2
mask   = dist2 <= r**2
ap     = mask * a
tfp    = ap * np.exp(1j * phi)
fp     = np.real(nf.ifft2(tfp, axes=(0, 1)))
```

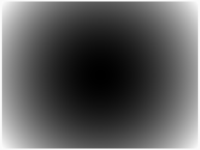
f



a

u

v



dist2

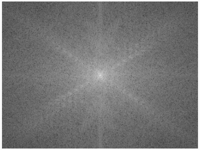
Spectral filtering – Low-pass filter

Python demo – Low-pass filter

```
import numpy.fft as nf
import imagetools as im

f      = plt.imread('butterfly.png')
n1, n2 = f.shape
tf     = nf.fft2(f, axes=(0, 1))
a      = np.abs(tf)
phi    = np.angle(tf)
u, v   = im.fftgrid(n1, n2)
dist2  = u**2 + v**2
mask   = dist2 <= r**2
ap     = mask * a
tfp    = ap * np.exp(1j * phi)
fp     = np.real(nf.ifft2(tfp, axes=(0, 1)))
```

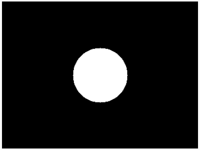
f



a

u

v



mask

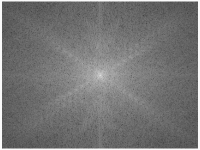
Spectral filtering – Low-pass filter

Python demo – Low-pass filter

```
import numpy.fft as nf
import imagetools as im

f      = plt.imread('butterfly.png')
n1, n2 = f.shape
tf     = nf.fft2(f, axes=(0, 1))
a      = np.abs(tf)
phi    = np.angle(tf)
u, v   = im.fftgrid(n1, n2)
dist2  = u**2 + v**2
mask   = dist2 <= r**2
▶ ap   = mask * a
tfp    = ap * np.exp(1j * phi)
fp     = np.real(nf.ifft2(tfp, axes=(0, 1)))
```

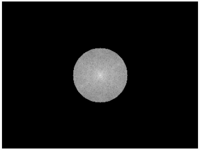
f



a

u

v



ap

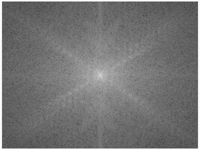
Spectral filtering – Low-pass filter

Python demo – Low-pass filter

```
import numpy.fft as nf
import imagetools as im

f      = plt.imread('butterfly.png')
n1, n2 = f.shape
tf     = nf.fft2(f, axes=(0, 1))
a      = np.abs(tf)
phi    = np.angle(tf)
u, v   = im.fftgrid(n1, n2)
dist2  = u**2 + v**2
mask   = dist2 <= r**2
ap     = mask * a
tfp    = ap * np.exp(1j * phi)
▶ fp   = np.real(nf.ifft2(tfp, axes=(0, 1)))
```

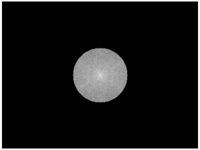
f



a

u

v



ap

fp

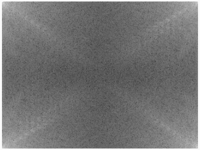
Spectral filtering – Low-pass filter

Python demo – Low-pass filter

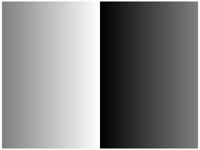
```python
import numpy.fft as nf
import imagetools as im

f      = plt.imread('butterfly.png')
n1, n2 = f.shape
tf     = nf.fft2(f, axes=(0, 1))
a      = np.abs(tf)
phi    = np.angle(tf)
u, v   = im.fftgrid(n1, n2)
dist2  = u**2 + v**2
mask   = dist2 <= r**2
ap     = mask * a
tfp    = ap * np.exp(1j * phi)
fp     = np.real(nf.ifft2(tfp, axes=(0, 1)))
```

f



a



u

v

ap

fp

nf.fftshift

Spectral filtering – Low-pass filter

Shorter version

```
f      = plt.imread('butterfly.png')
n1, n2 = f.shape
u, v   = im.fftgrid(n1, n2)

tfp    = nf.fft2(f, axes=(0, 1))               # Transform
tfp[u**2 + v**2 > r**2] = 0                    # Modify
fp     = np.real(mpf.ifft2(tfp, axes=(0, 1)))   # Transform back
```

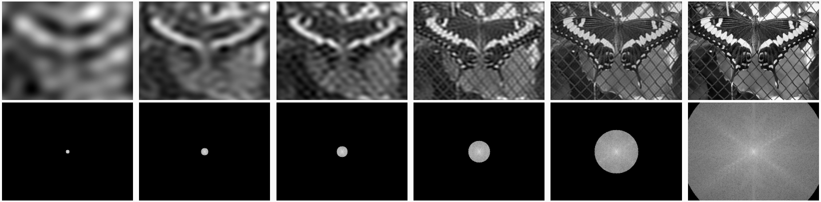
Spectral filtering – Low-pass filter

Shorter version

```
f      = plt.imread('butterfly.png')
n1, n2 = f.shape
u, v   = im.fftgrid(n1, n2)

tfp    = nf.fft2(f, axes=(0, 1))                    # Transform
tfp[u**2 + v**2 > r**2] = 0                         # Modify
fp     = np.real(mpf.ifft2(tfp, axes=(0, 1)))       # Transform back
```

What is the influence of the radius `r`?

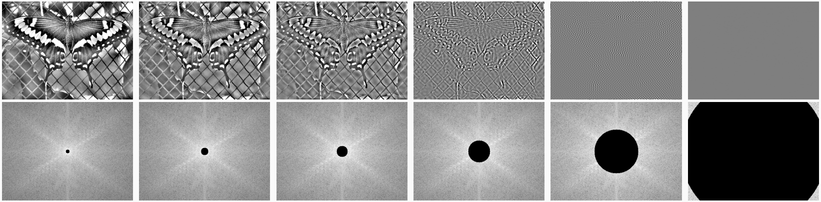


Acts similarly as a blur

Spectral filtering – High-pass filter

What if we do the opposite? (high-pass filter)

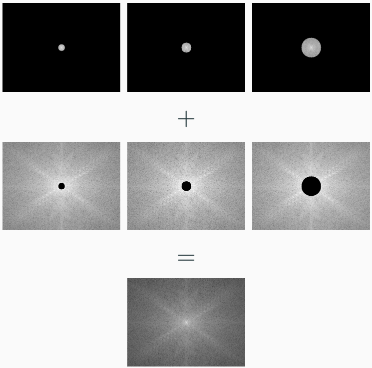
`u**2 + v**2 > r**2` $\rightarrow$ `u**2 + v**2 <= r**2`

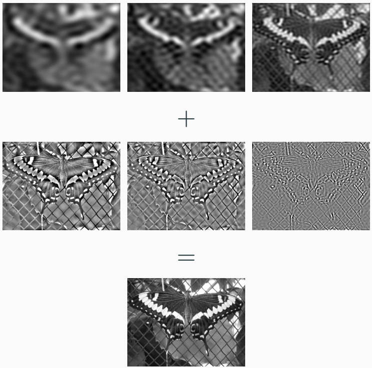


Acts similarly as an edge detector

Spectral filtering – High + Low -pass filters

What if we sum the two components?





$$\mathrm{M} \odot \hat{f} + (\mathrm{Id} - \mathrm{M}) \odot \hat{f} = \hat{f}$$

$$\mathcal{F}^{-1}[\mathrm{M} \odot \hat{f}] + \mathcal{F}^{-1}[(\mathrm{Id} - \mathrm{M}) \odot \hat{f}] = f$$

Image = Low frequencies + High frequencies

= Local averages + Edges/Textures

Spectral filtering – Low/High $\equiv$ Smooth/Edges

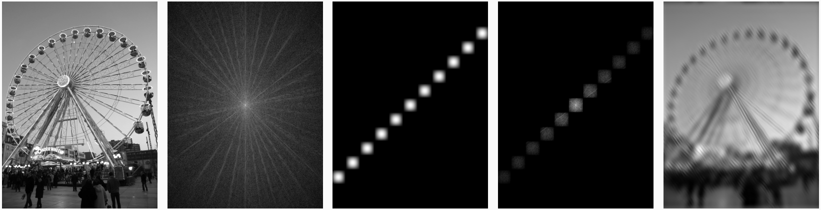
Standard spectral filters

- Accept or reject some frequencies
- Low-pass filter: smooth the image (accept low frequencies)
- High-pass filter: preserve edges (accept high frequencies)

Is there a connection with moving averages and derivative filters?

Spectral modulation

- Apply the Fourier transform $\hat{x} = \mathcal{F}[x]$
- Modulate each frequency individually $\hat{y}_{u,v} = \lambda_{u,v} \cdot \hat{x}_{u,v}$
- Apply the inverse Fourier transform $y = \mathcal{F}^{-1}[\hat{y}]$



(a) x (b) $\hat{x}$ (c) λ (d) $\hat{y}$ (e) y

Spectral filtering – DFT in matrix form

$$\hat{x} = \mathcal{F}[x] \qquad \hat{y}_u = \lambda_u \cdot \hat{x}_u \qquad y = \mathcal{F}^{-1}[\hat{y}]$$

Matrix form in 1d

- The Fourier transform can be written as

$$\underbrace{\hat{x}_u = \sum_{k=0}^{n-1} x_k e^{-i2\pi\frac{uk}{n}}}_{=\mathcal{F}[x]_u} \equiv \hat{x} = \underbrace{\begin{pmatrix} 1 & 1 & \dots & 1 \\ 1 & e^{-i2\pi\frac{1}{n}} & \dots & e^{-i2\pi\frac{n-1}{n}} \\ 1 & e^{-i2\pi\frac{2}{n}} & \dots & e^{-i2\pi\frac{2(n-1)}{n}} \\ \vdots & & & \\ 1 & e^{-i2\pi\frac{(n-1)}{n}} & \dots & e^{-i2\pi\frac{(n-1)^2}{n}} \end{pmatrix}}_{=\boldsymbol{F}} x$$

- The modulation as: $\hat{y} = \underbrace{\begin{pmatrix} \lambda_1 & & & \\ & \lambda_2 & & \\ & & \ddots & \\ & & & \lambda_n \end{pmatrix}}_{\boldsymbol{\Lambda}} \hat{x}$
- The inverse transform as $y = \boldsymbol{F}^{-1}\hat{y}$ with $\boldsymbol{F}^{-1} = \frac{1}{n}\boldsymbol{F}^*$.
- It follows that:

$$y = \frac{1}{n}\boldsymbol{F}^*\boldsymbol{\Lambda}\boldsymbol{F}x$$

Spectral filtering – DFT in matrix form

Link with circulant matrices

- Let $\boldsymbol{E} = \frac{1}{\sqrt{n}}\boldsymbol{F}^*$ and $\boldsymbol{E}^{-1} = \frac{1}{\sqrt{n}}\boldsymbol{F}$, and write

$$y = \frac{1}{n}\boldsymbol{F}^*\boldsymbol{\Lambda}\boldsymbol{F}x = \boldsymbol{E}\boldsymbol{\Lambda}\boldsymbol{E}^{-1}x$$

- The columns of $\boldsymbol{E}$ are of the form

$$e_k = \frac{1}{\sqrt{n}}\left(1, \exp\left(\frac{2\pi ik}{n}\right), \exp\left(\frac{4\pi ik}{n}\right), \ldots, \exp\left(\frac{2(n-1)\pi ik}{n}\right)\right)^T$$

  and are eigenvectors with unit norms of circulant matrices (see, last class)

- Then $\boldsymbol{E}\boldsymbol{\Lambda}\boldsymbol{E}^{-1}$ is the eigendecomposition of a circulant matrix $\boldsymbol{H}$
- And $y = \boldsymbol{H}x$ is nothing else as the convolution of x by some kernel ν.

Convolutions are diagonal in the Fourier domain

Spectral filtering – DFT in matrix form

Link with circulant matrices

- Let $\boldsymbol{E} = \frac{1}{\sqrt{n}}\boldsymbol{F}^*$ and $\boldsymbol{E}^{-1} = \frac{1}{\sqrt{n}}\boldsymbol{F}$, and write

$$y = \frac{1}{n}\boldsymbol{F}^*\boldsymbol{\Lambda}\boldsymbol{F}x = \boldsymbol{E}\boldsymbol{\Lambda}\boldsymbol{E}^{-1}x$$

- The columns of $\boldsymbol{E}$ are of the form

$$e_k = \frac{1}{\sqrt{n}}\left(1, \exp\left(\frac{2\pi ik}{n}\right), \exp\left(\frac{4\pi ik}{n}\right), \ldots, \exp\left(\frac{2(n-1)\pi ik}{n}\right)\right)^T$$

  and are eigenvectors with unit norms of circulant matrices (see, last class)
- Then $\boldsymbol{E}\boldsymbol{\Lambda}\boldsymbol{E}^{-1}$ is the eigendecomposition of a circulant matrix $\boldsymbol{H}$
- And $y = \boldsymbol{H}x$ is nothing else as the convolution of x by some kernel ν.

Convolutions are diagonal in the Fourier domain

Why is that important?

Spectral filtering – Fast convolutions with FFT

FFT $\Rightarrow$ Fast Convolutions

- Complexity of convolutions in spatial domain
- Limited support $s \times s$
  - Non separable: $O(s^2 n)$
  - Separable: $O(sn)$
- Unlimited support
  - Non separable: $O(n^2)$
  - Separable: $O(n^{3/2})$

- Complexity of convolutions through Fourier domain

$$\underbrace{\hat{x} = \mathcal{F}[x]}_{O(n \log n)} \qquad \underbrace{\hat{y}_u = \lambda_u \cdot \hat{x}_u}_{O(n)} \qquad \underbrace{y = \mathcal{F}^{-1}[\hat{y}]}_{O(n \log n)} \qquad \Rightarrow \quad \boldsymbol{O(n \log n)}$$

- Allows kernel functions to have a much larger support $s \times s$,
- Note: Spatial implementation can still be faster for small s.

Spectral filtering – Fast convolutions with FFT

FFT $\Rightarrow$ Fast Convolutions

- Complexity of convolutions in spatial domain

- Limited support $s \times s$
  - Non separable: $O(s^2 n)$
  - Separable: $O(sn)$

- Unlimited support
  - Non separable: $O(n^2)$
  - Separable: $O(n^{3/2})$

- Complexity of convolutions through Fourier domain

$$\underbrace{\hat{x} = \mathcal{F}[x]}_{O(n \log n)} \qquad \underbrace{\hat{y}_u = \lambda_u \cdot \hat{x}_u}_{O(n)} \qquad \underbrace{y = \mathcal{F}^{-1}[\hat{y}]}_{O(n \log n)} \qquad \Rightarrow \quad \boldsymbol{O(n \log n)}$$

- Allows kernel functions to have a much larger support $s \times s$,
- Note: Spatial implementation can still be faster for small s.

What is the link between the modulation λ and the convolution kernel ν?

Link between λ and ν

- The eigenvalues of a circulant matrix

$$\boldsymbol{H} = \begin{pmatrix} \nu_0 & \nu_{n-1} & \nu_{n-2} & \cdots & \nu_2 & \nu_1 \\ \nu_1 & \nu_0 & \nu_{n-1} & \nu_{n-2} & \cdots & \nu_2 \\ & & \ddots & & & \\ & & & \ddots & & \\ & & & & \ddots & \\ \nu_{n-1} & \nu_{n-2} & \cdots & \nu_2 & \nu_1 & \nu_0 \end{pmatrix}$$

  are

$$\lambda_u = \sum_{k=0}^{n-1} \nu_k \exp\left(-\frac{2\pi i u k}{n}\right)$$

Spectral filtering – Spectrum and convolution kernels

Link between λ and ν

- The eigenvalues of a circulant matrix

$$\boldsymbol{H} = \begin{pmatrix} \nu_0 & \nu_{n-1} & \nu_{n-2} & \cdots & \nu_2 & \nu_1 \\ \nu_1 & \nu_0 & \nu_{n-1} & \nu_{n-2} & \cdots & \nu_2 \\ & & \ddots & & & \\ & & & \ddots & & \\ & & & & \ddots & \\ \nu_{n-1} & \nu_{n-2} & \cdots & \nu_2 & \nu_1 & \nu_0 \end{pmatrix}$$

  are

$$\lambda_u = \sum_{k=0}^{n-1} \nu_k \exp\left(-\frac{2\pi i u k}{n}\right) = \mathcal{F}[\nu]_u$$

- Which means: $\boldsymbol{H} = \boldsymbol{F}^{-1}\boldsymbol{\Lambda}\boldsymbol{F}$ with $\boldsymbol{\Lambda} = \operatorname{diag}(\boldsymbol{F}\nu)$, and thus

$$\nu * x = \boldsymbol{F}^{-1}\operatorname{diag}(\boldsymbol{F}\nu)\boldsymbol{F}x$$

This is the **Convolution theorem**

Spectral filtering – Spectrum and convolution kernels

Theorem (Convolution theorem)

Vector form

$$h = f * g \quad \Leftrightarrow \quad \hat{h}_u = \hat{f}_u \cdot \hat{g}_u$$

Function form

$$(f * g)(t) = \mathcal{F}^{-1}(\mathcal{F}(f) \cdot \mathcal{F}(g))(t)$$

Matrix-vector form

$$f * g = \underbrace{\boldsymbol{F}^{-1} \operatorname{diag}(\boldsymbol{F} f) \boldsymbol{F}}_{\textit{circulant matrix}} g$$

Take home message

Convolution in spatial domain = Product in Fourier domain

Provides a new interpretation for LTI filters

- The convolution kernel ν characterizes the filter, (impulse response)
- Its Fourier transform $\lambda = \boldsymbol{F}\nu$ as well. (frequential response)

Spectral filtering – Properties of the Fourier transform

Main properties

	Time	Continuous	Discrete (periodic)
Linearity	$af + bg$	$a\hat{f} + b\hat{g}$	
Real/Hermitian	real	Hermitian	
Reverse/Conjugation	$f(-t)$	$\hat{f}^*$	
Convolution	$f * g$	$\hat{f} \cdot \hat{g}$	
Auto-correlation	$f \star g$	$\hat{f}^* \cdot \hat{g}$	
Zero frequency	$\int / \sum$	$\hat{f}(0)$	
Shift	$f(t-\delta)$	$e^{-i2\pi\delta u}\hat{f}(u)$	$e^{-i2\pi\delta u/n}\hat{f}_u$
Parseval	$\langle f, g\rangle$	$\langle \hat{f}, \hat{g}\rangle$	$\frac{1}{n}\langle \hat{f}, \hat{g}\rangle$
Plancherel	$\\|f\\|_2$	$\\|\hat{f}\\|_2$	$\frac{1}{n}\\|\hat{f}\\|_2$
Scaling	$f(at)$	$\frac{1}{\|a\|}\hat{f}(\frac{u}{a})$	–
Differentiation	$\frac{d^n f(t)}{dt^n}$	$(2\pi i u)^n \hat{f}(u)$	–

Similar properties for multi-dimensional signals

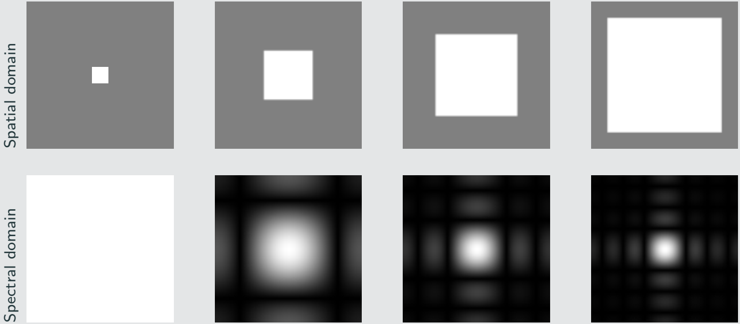
Spectral filtering – Moving averages = Low pass filters

Properties of moving average filters

- Low frequencies are preserved
- High frequencies are attenuated
- Zero-frequency is always one
- Preserves the mean of pixel values

Spectral filtering – Moving averages = Low pass filters

Boxcar filter

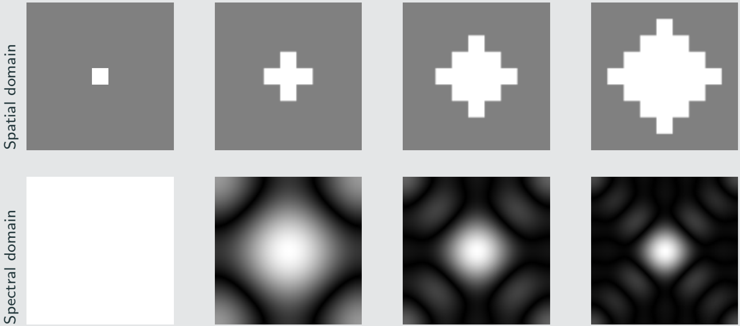


2d cardinal sines: $\frac{1}{\tau^2}\,\mathrm{sinc}(u/\tau)\,\mathrm{sinc}(v/\tau)$ $\quad$ $(\mathrm{sinc}(t) = \dfrac{\sin(t)}{t})$

- Bandwidth proportional to $1/\tau$
- Keep some high horizontal and vertical frequencies (side lobes)
- Explains horizontal and vertical artifacts of boxcar filters

Spectral filtering – Moving averages = Low pass filters

Diamond filter

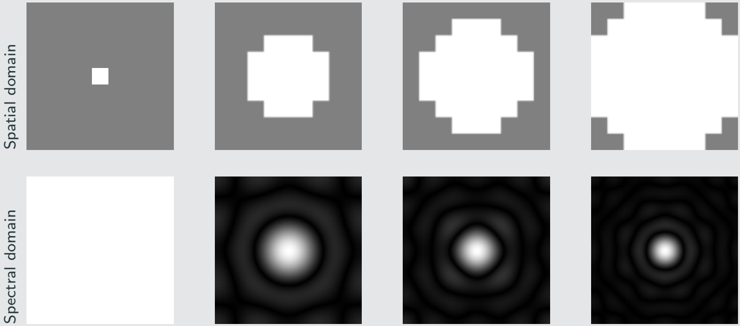


Similar to the box but rotated of 45°

- Bandwidth proportional to $1/\tau$
- Keep some high frequencies in diagonal directions (side lobes)
- Explains diagonal artifacts of diamond filters

Spectral filtering – Moving averages = Low pass filters

Diskcar filter

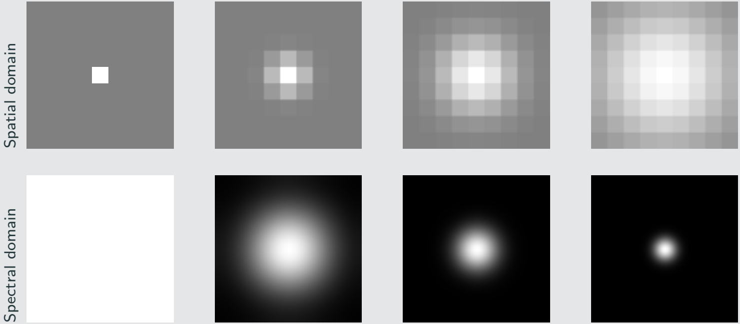


Cardinal sine in all directions

- Bandwidth proportional to $1/\tau$
- Keep some high frequencies (side lobes)
- No preferred direction (isotropic)

Spectral filtering – Moving averages = Low pass filters

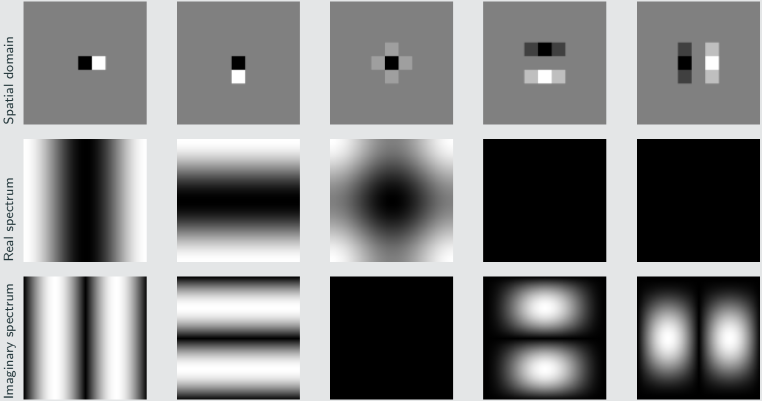
Gaussian filter



$$\mathcal{F}\left[\frac{1}{2\pi\tau^2}e^{-\frac{(s_1^2+s_2^2)}{2\tau^2}}\right] = e^{-4\pi^2\tau^2(u^2+v^2)} \quad \equiv \quad \mathcal{F}[\mathcal{G}_{\tau^2}] = \sqrt{2\pi\tau^2}^d \mathcal{G}_{1/4\pi^2\tau^2}$$

- Bandwidth proportional to $1/\tau$
- High frequencies are smoothly and monotonically removed
- No preferred direction (isotropic)

Derivative filters = High pass filters

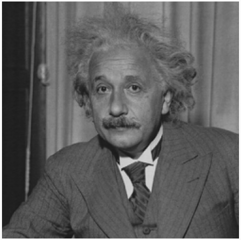


- Keep high frequencies only
- Often complex valued

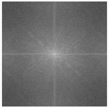
- Zero frequency is null
- Subtract the mean

Image sharpening

Spectral filtering – Image sharpening

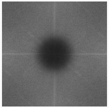


(a) x

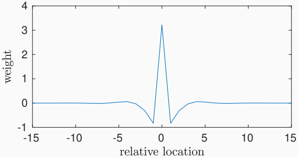


(b) $\hat{x}$

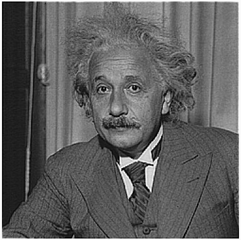
(c) $\lambda = \boldsymbol{F}\nu$



(d) $\hat{y}$



(e) ν



(f) y

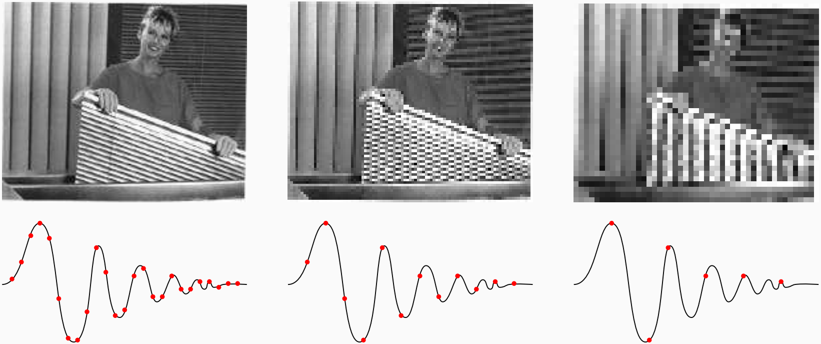
Image sharpening

- Goal: Re-enforce edges
- How: $\left\{ \right.$
  - Keep low frequencies
  - Amplify high frequencies
- Drawback: Amplify noise

$$y = x + \alpha D x$$

D: derivative filter, $\alpha > 0$

Image resizing

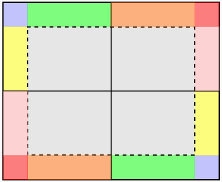


(a) $\times 1$ (b) $\times 2$ (c) $\times 4$

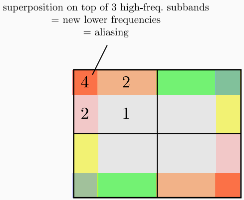
Spatial image resizing (sub-sampling by a factor a)

- Continuous image: $f^{\text{rescaled}}(t) = f(at)$
- Discrete image, ex: $f_k^{\text{rescaled}} = (1 - ak + \lfloor ak \rfloor) f_{\lfloor ak \rfloor} + (ak - \lfloor ak \rfloor) f_{\lceil ak \rceil}$ (linear interpolation)
- Aliasing: High frequencies lost, new frequencies created. Why?

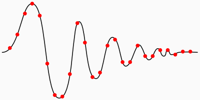
Spectral filtering – Image resizing / sub-sampling



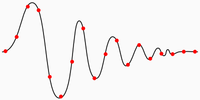
Spectrum before spatial subsampling



after spatial subsampling



(a) $\times 1$



(b) $\times 4/3$

Nyquist

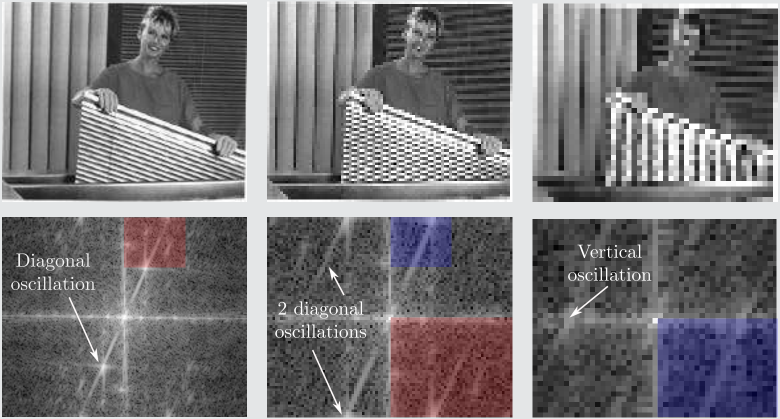
Shannon

Aliasing

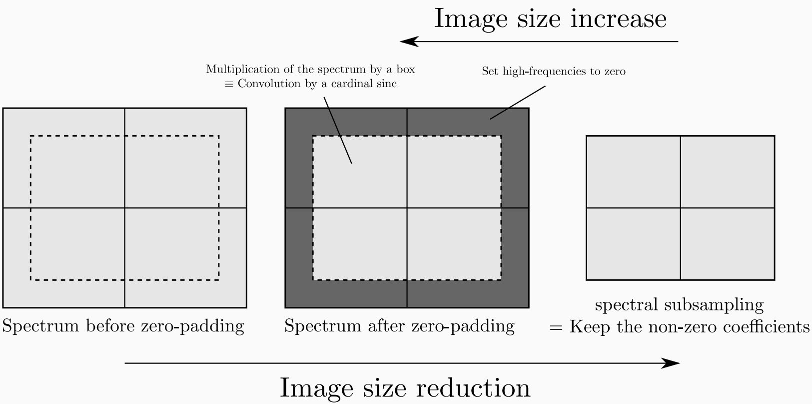
- Superposition of high frequency sub-bands in the new resized image
- Linked with Nyquist-Shannon's theorem:

  sampling frequency should be at least double the maximum frequency

Aliasing: how diagonal stripes become vertical...



How to avoid aliasing when resizing?

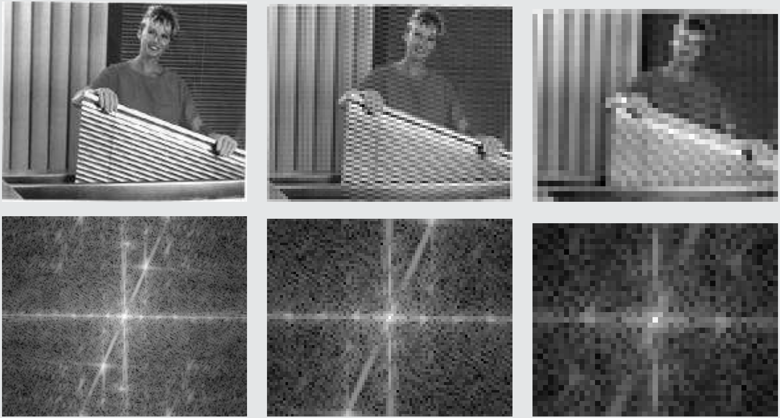


Spectral image resizing with zero-padding

- Reduction: set high frequencies to zero and reduce spectrum size
- Increase: increase spectrum size and fill new high frequencies by zeros

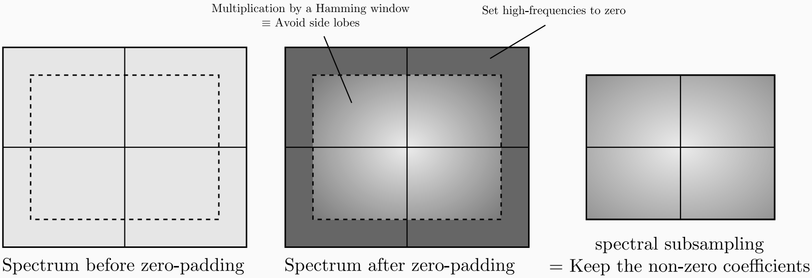
Spectral filtering – Image resizing / sub-sampling

Zero-padding: No more aliasing but unpleasant oscillations



How to avoid side lobes of the cardinal sine? (ringing/Gibbs artifacts)

Spectral filtering – Image resizing / sub-sampling



Zero-padding + windowing

- Not only set the high frequencies to zeros
- But modulate low frequencies by a weighting window, *i.e.*, a blur
- Choice of the window: trade-off between ringing vs blur

Spectral filtering – Image resizing / sub-sampling

Typical windows

- Hann window: to reduce all side lobes

$$w(u) = 0.5 - 0.5\cos\left(\frac{2\pi(u + \lceil n/2 \rceil - 1)}{n-1}\right)$$

- Hamming window: to reduce first side lobe

$$w(u) = 0.54 - 0.46\cos\left(\frac{2\pi(u + \lceil n/2 \rceil - 1)}{n-1}\right)$$

- Kaiser window: to choose a trade-off between blur and side lobes.

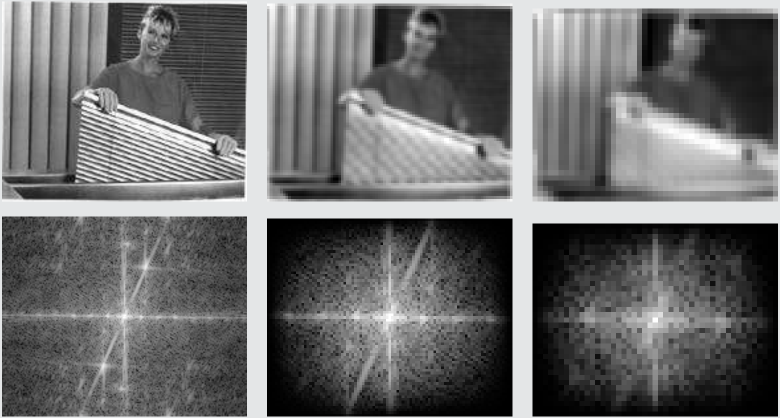
$$w(u) = \frac{I_0\left(\pi\alpha\sqrt{1 - \left(\frac{2(u+\lceil n/2 \rceil - 1)}{n-1} - 1\right)^2}\right)}{I_0(\pi\alpha)}, \quad \alpha > 0$$

for frequencies $u = -\lceil n/2 \rceil + 1$ to $\lfloor n/2 \rfloor$.

I_0: zero-order modified Bessel function.

Spectral filtering – Image resizing / sub-sampling

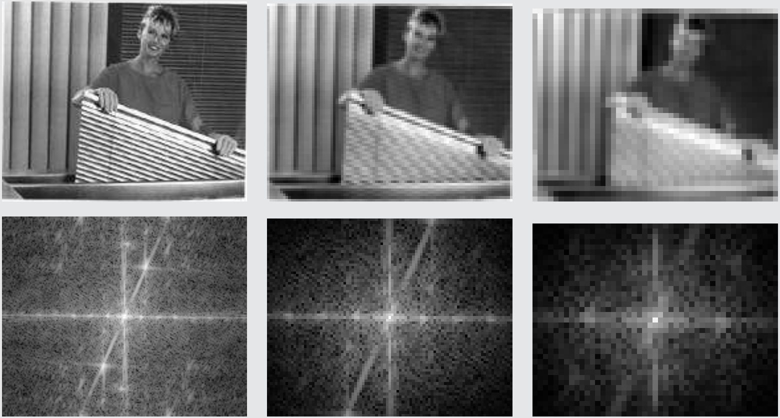
Hann window



Hann window: No more aliasing, no more ringing, but blur

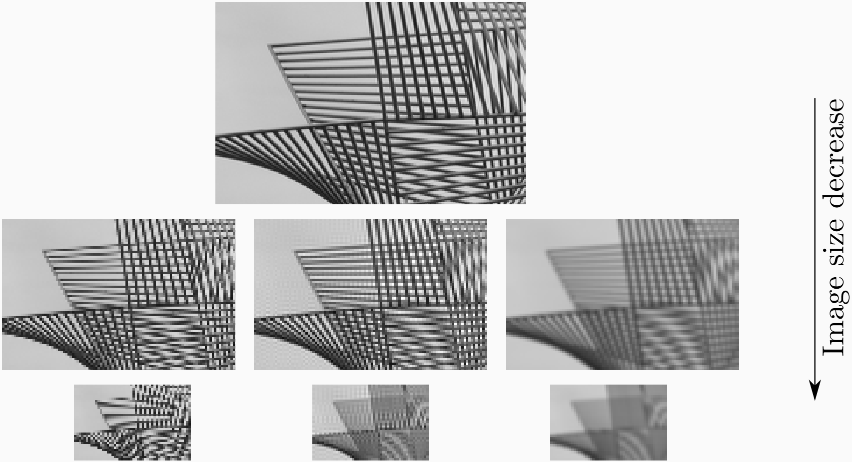
Spectral filtering – Image resizing / sub-sampling

Kaiser window



Kaiser window: No more aliasing and trade-off between ringing and blur

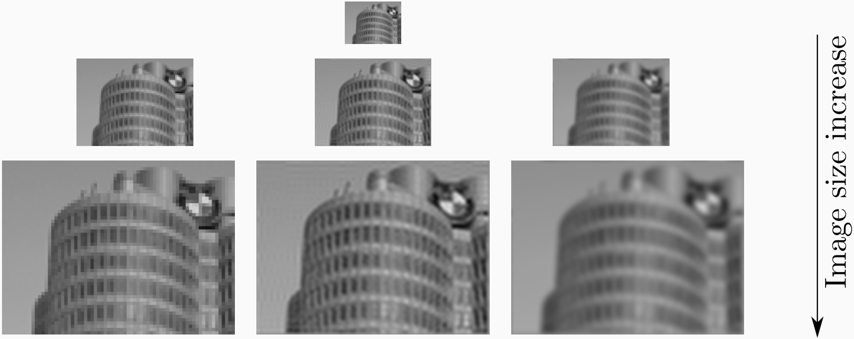
Spectral filtering – Image resizing / sub-sampling



Spatial sub-sampling
Linear interpolation

Spectral sub-sampling
Zero-padding

Spectral sub-sampling
Windowing



Spatial sub-sampling
Linear interpolation

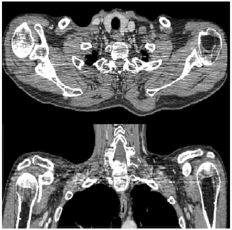
Spectral sub-sampling
Zero-padding

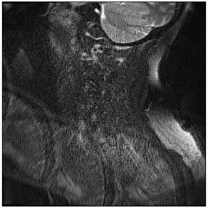
Spectral sub-sampling
Windowing

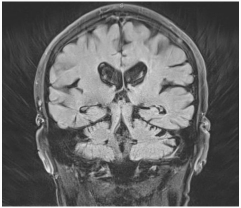
Streaking

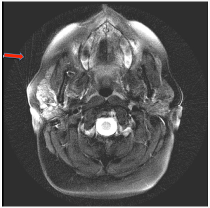
Spectral filtering – Streaking

What about streaking?









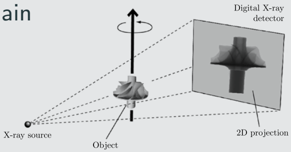
Computed tomography (CT)

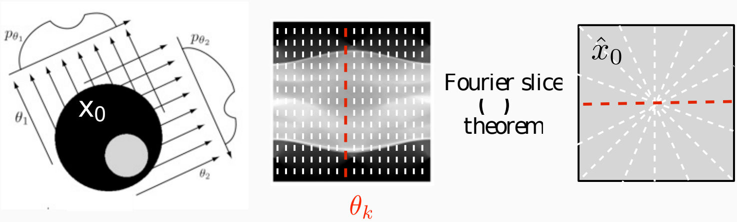
Fourier slice theorem:

One projection = one line in the Fourier domain

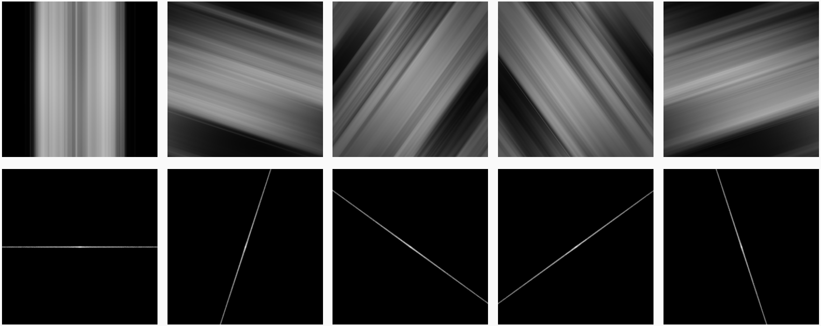
Radon transform:

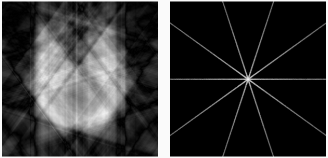
- K projections = K lines
- Capture frequencies along these lines
- Other frequencies are seen as being zero





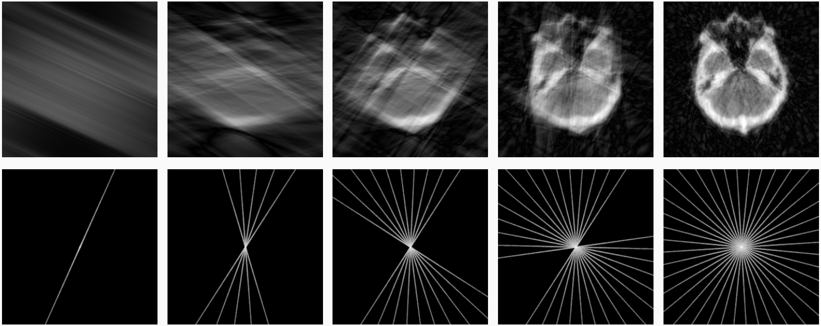
Spectral filtering – Streaking in CT / Radon transform



Fusion:  **What is that?**

Spectral filtering – Streaking in CT / Radon transform

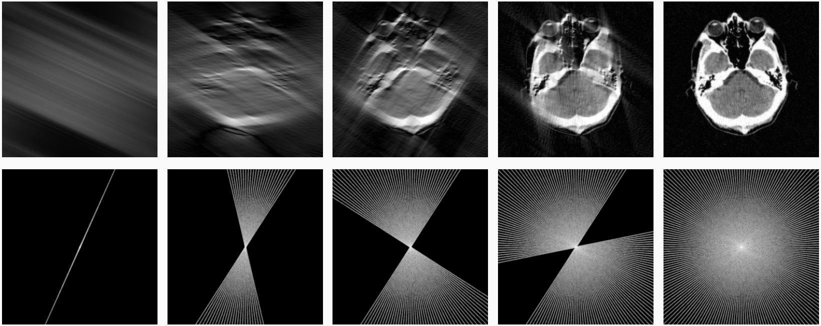
Use more projection angles



- As for sampling in spatial domain, there is a Nyquist barrier
- i.e., a threshold in the minimum number of lines to acquire
- below that threshold, image processing techniques must be used to fill the missing frequencies (a sort of inpainting problem in the Fourier domain)

Spectral filtering – Streaking in CT / Radon transform

Use more projection angles, . . . or even more

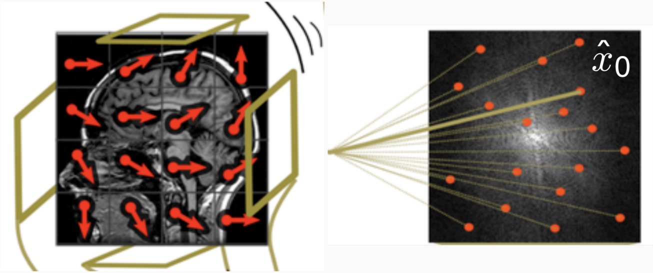


- As for sampling in spatial domain, there is a Nyquist barrier
- i.e., a threshold in the minimum number of lines to acquire
- below that threshold, image processing techniques must be used to fill the missing frequencies (a sort of inpainting problem in the Fourier domain)

Spectral filtering – Streaking in MRI

Magnetic Resonance Imaging (MRI)

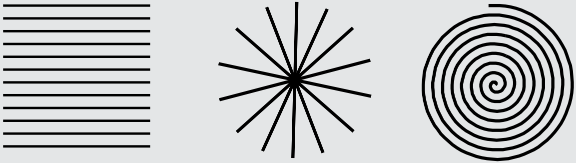
- Design/Setting of the MRI machine defines a path in the Fourier domain (called k-space)
- It captures frequencies along this path
- Other frequencies are seen as being zero



Spectral filtering – Streaking in MRI

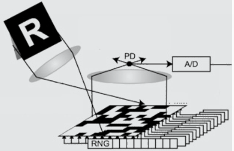
Feasible k-space trajectories

Cartesian Radial Spiral



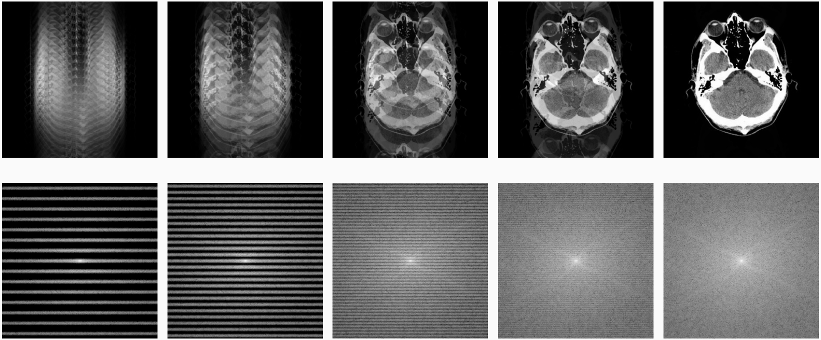
Ideal one

- Compressed sensing
- Select frequencies at random
- Incoherent measurements
- Not feasible yet



Spectral filtering – Streaking in MRI

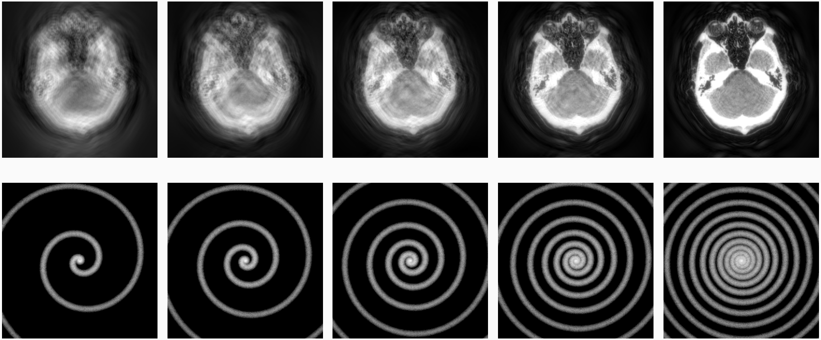
Cartesian path:



(a) 15%   (b) 25%   (c) 50%   (d) 75%   (e) 100%

Spectral filtering – Streaking in MRI

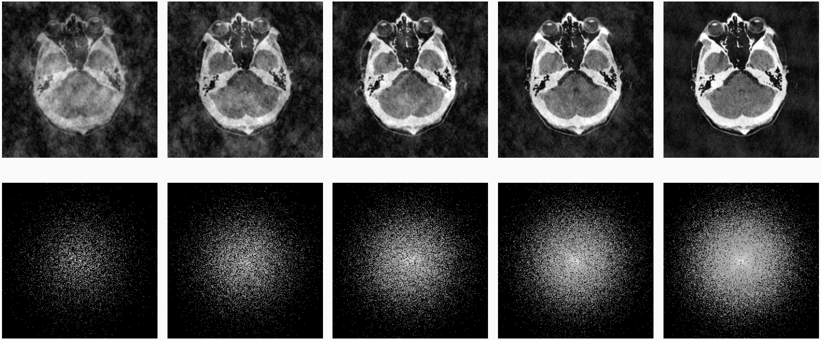
Spiral path:



(a) 20% (b) 28% (c) 40% (d) 57% (e) 80%

Spectral filtering – Streaking in MRI

Random path:



(a) 10% (b) 20% (c) 30% (d) 40% (e) 60%

Questions?

Next class: heat equation / anisotropic diffusion

Sources, images courtesy and acknowledgment

L. Condat

B. Denis de Senneville

A. Horodniceanu

I. Kokkinos

G. Peyré

R. Otazo

V.-T. Ta

Wikipedia