Fourier Transform: part 2
SCC0251/5830 – Image Processing

Prof. Moacir Ponti

Instituto de Ciências Matemáticas e de Computação – USP

2021/1

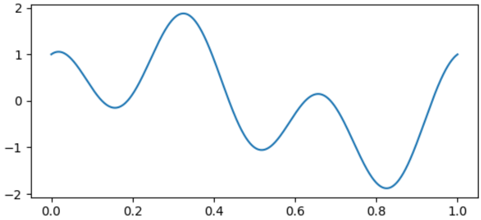
Agenda

1. Big Picture
2. Discrete Fourier Transform
   - Convolution Theorem
3. Filtering in frequency domain
4. Fast Fourier Transform (FFT)

Fourier Transform

It allows us to see the frequency content of a given signal.

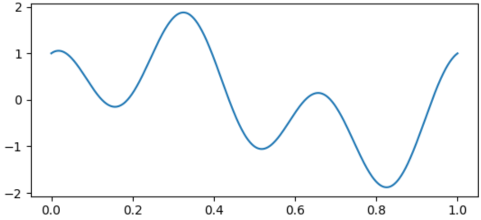
$$
\begin{aligned}
F(\omega) &= \int_{-\infty}^{\infty} f(t)\, \boxed{e^{-j2\pi t\omega}}\, dt \\
&= \int_{-\infty}^{\infty} f(t)\cos(2\pi t\omega)dt + j\int_{-\infty}^{\infty} f(t)\sin(2\pi t\omega)dt
\end{aligned}
$$



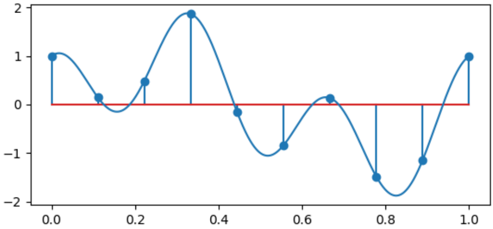
Fourier Transform

It allows us to see the frequency content of a given signal.

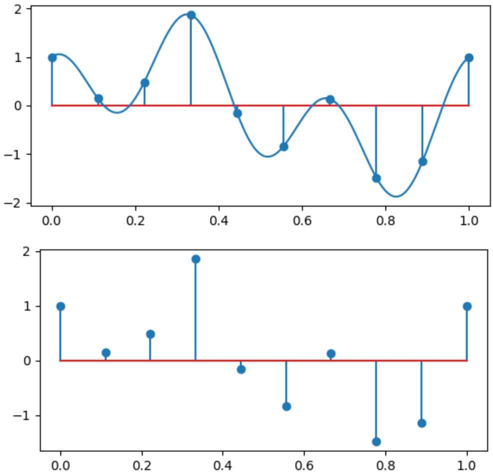
$$
\begin{aligned}
F(\omega) &= \int_{-\infty}^{\infty} f(t)\, \boxed{e^{-j2\pi t\omega}}\, dt \\
&= \int_{-\infty}^{\infty} f(t)\, \boxed{\cos(2\pi t\omega)}\, dt + j \int_{-\infty}^{\infty} f(t)\, \boxed{\sin(2\pi t\omega)}\, dt
\end{aligned}
$$



Fourier Transform



Fourier Transform



Discrete Fourier Transform

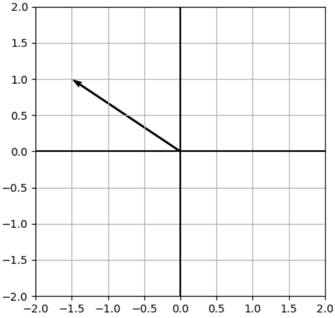
for N samples:

$$F[k] = \sum_{n=0}^{N-1} f[n] e^{-j2\pi n \frac{k}{N}}$$
$$= A[k] + j\, B[k]$$

Fourier Transform

For each frequency a complex exponential with:

- the relative amplitude of the cosine (real part) and of the sine (imaginary part) as a function of different frequencies,
- the representation of the signal in the **frequency domain**:
  - $A[k] = Re(F[k])$
  - $B[k] = Im(F[k])$

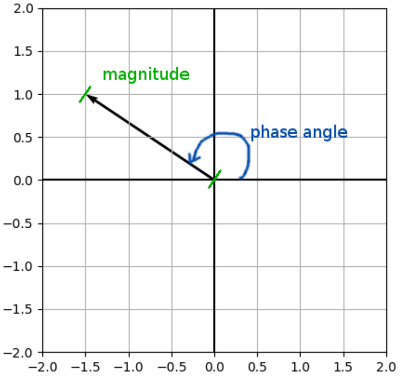


Fourier Transform

$$F[k] = \sum_{n=0}^{N-1} f[n] e^{-j2\pi n \frac{k}{N}}$$

- evaluating $F[k]$ on different k, we obtain the **amplitudes of** cosines (real part) and sines (imaginary part) so that we can reconstruct $f[n]$ if needed.

Fourier Transform



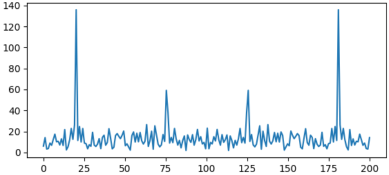
$$|F[k]| = \sqrt{A[k]^2 + B[k]^2}$$

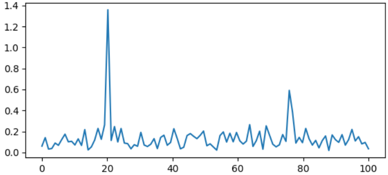
$$\phi(F[k]) = \arctan\left(\frac{A[k]}{B[k]}\right)$$

Fourier Transform: practical considerations

- Cosine is an odd (anti-symmetric) function
- Sine is an even (symmetric) function

Because of that, values of k after $N/2$ start to repeat. That is why we just show half the values.



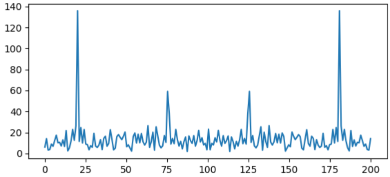


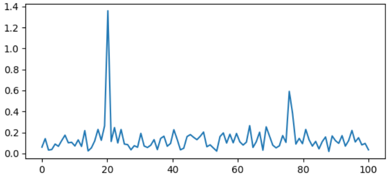
.

Fourier Transform: practical considerations

- Cosine is an odd (anti-symmetric) function
- Sine is an even (symmetric) function

Because of that, values of k after $N/2$ start to repeat. That is why we just show half the values.

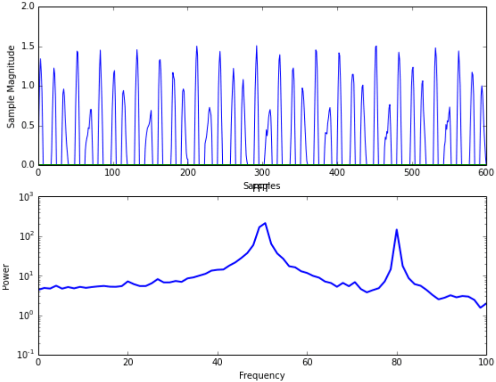




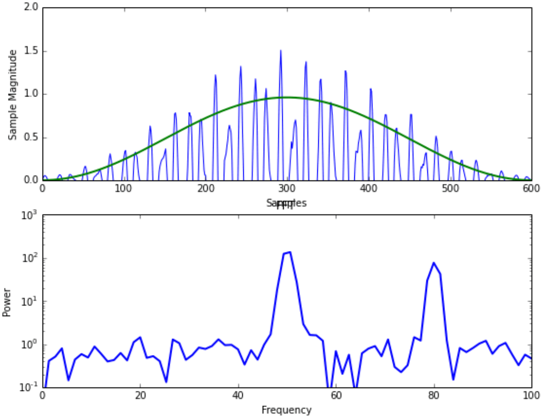
The maximum frequency $N/2$ has a important relationship with the Nyquist-Shannon Theorem of sampling.

Fourier Transform: noise and abrupt signals

- Because we start to measure at $k = 0$ and stop at $k = N - 1$, this can be considered high frequency content, hampering the analysis
- To prevent that, it is possible to use a windowing function, to obtain a signal that increases its amplitude more slowly



Fourier Transform: noise and abrupt signals



Agenda

Discrete Fourier Transform 2D (DFT)

$$F(u,v) = \frac{1}{\sqrt{mn}} \sum_{x=0}^{n-1} \sum_{y=0}^{m-1} f(x,y) e^{-j2\pi(ux/n + vy/m)},$$

$f(x,y)$ is a function representing an image with size $n \times m$.

Discrete Fourier Transform 2D (DFT)

$$F(u, v) = \frac{1}{\sqrt{mn}} \sum_{x=0}^{n-1} \sum_{y=0}^{m-1} f(x, y) e^{-j2\pi(ux/n + vy/m)},$$

$f(x, y)$ is a function representing an image with size $n \times m$.

- considering the 2-D case: x, y are coordinates, u, v are frequencies in each direction.

Discrete Fourier Transform 2D (DFT)

$$F(u,v) = \frac{1}{\sqrt{mn}} \sum_{x=0}^{n-1} \sum_{y=0}^{m-1} f(x,y) e^{-j2\pi(ux/n + vy/m)},$$

$f(x,y)$ is a function representing an image with size $n \times m$.

- considering the 2-D case: x, y are coordinates, u, v are frequencies in each direction.
- as in 1-D, we evaluate it for a range of u and v:
    - $u = 0, 1, ..., n-1$ and $v = 0, 1, ..., m-1$

Inverse Discrete Fourier Transform 2D (IDFT)

$$f(x,y) = \frac{1}{\sqrt{nm}} \sum_{u=0}^{n-1} \sum_{v=0}^{m-1} F(u,v) e^{j2\pi(ux/n + vy/m)},$$

for $x = 0, 1, ...n-1$ and $y = 0, 1, ..., m-1$

Spectrum and phase

- Writing the DFT-2D in the polar form

$$F(u,v) = |F(u,v)|e^{j\phi(u,v)},$$

Spectrum and phase

- Writing the DFT-2D in the polar form

$$F(u, v) = |F(u, v)|e^{j\phi(u,v)},$$

- the magnitude is often called *Fourier spectrum* (or frequency spectrum):

$$|F(u, v)| = \left[R^2(u, v) + I^2(u, v)\right]^{1/2}$$

Spectrum and phase

- Writing the DFT-2D in the polar form

$$F(u,v) = |F(u,v)|e^{j\phi(u,v)},$$

- the magnitude is often called *Fourier spectrum* (or frequency spectrum):

$$|F(u,v)| = \left[R^2(u,v) + I^2(u,v)\right]^{1/2}$$

- the *phase angle* is:

$$\phi(u,v) = \arctan\left[\frac{I(u,v)}{R(u,v)}\right]$$

Spectrum and phase

- Writing the DFT-2D in the polar form

$$F(u,v) = |F(u,v)|e^{j\phi(u,v)},$$

- the magnitude is often called *Fourier spectrum* (or frequency spectrum):

$$|F(u,v)| = \left[R^2(u,v) + I^2(u,v)\right]^{1/2}$$

- the *phase angle* is:

$$\phi(u,v) = \arctan\left[\frac{I(u,v)}{R(u,v)}\right]$$

- the *power spectrum* is:

$$P(u,v) = |F(u,v)|^2$$

Properties

- From Fourier Transform

$$F(0,0) = \frac{1}{\sqrt{nm}} \sum_{x=0}^{n-1} \sum_{y=0}^{m-1} f(x,y),$$

the frequency term zero is proportional to the sum of all values of $f(x,y)$ (normalised by $1/\sqrt{nm}$).

Properties

- From Fourier Transform

$$F(0,0) = \frac{1}{\sqrt{nm}} \sum_{x=0}^{n-1} \sum_{y=0}^{m-1} f(x,y),$$

  the frequency term zero is proportional to the sum of all values of $f(x,y)$ (normalised by $1/\sqrt{nm}$).

- The Fourier spectrum of a real function is even (symmetric with respect to the origin):

$$|F(u,v)| = |F(-u,-v)|$$

Properties

- From Fourier Transform

$$F(0,0) = \frac{1}{\sqrt{nm}} \sum_{x=0}^{n-1} \sum_{y=0}^{m-1} f(x,y),$$

  the frequency term zero is proportional to the sum of all values of $f(x,y)$ (normalised by $1/\sqrt{nm}$).
- The Fourier spectrum of a real function is even (symmetric with respect to the origin):

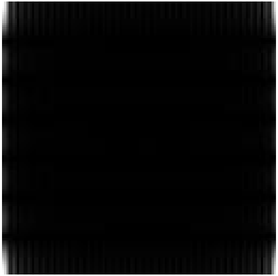
$$|F(u,v)| = |F(-u,-v)|$$

- the phase angle is odd (anti-symmetric):

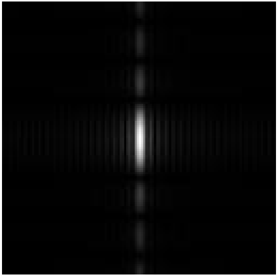
$$\phi(u,v) = -\phi(-u,-v)$$

Plotting the Fourier spectrum

- Re-positioning of the quadrants (*FFT shift*) by using coordinates $(-1)^{x+y}$



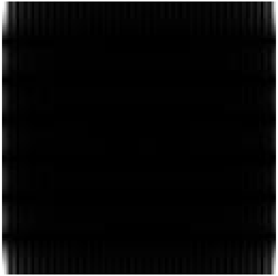
$|F(u,v)|$



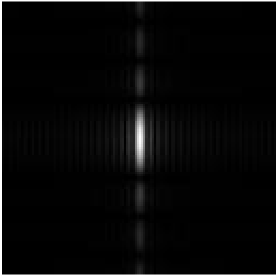
$Shift(|F(u,v)|)$

Plotting the Fourier spectrum

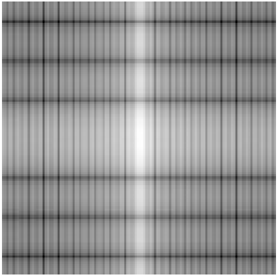
- Re-positioning of the quadrants (*FFT shift*) by using coordinates $(-1)^{x+y}$
- Applying the logarithm: $(\log(1 + |F(u,v)|)$ to increase the dynamic range of frequencies with small amplitude.



$|F(u,v)|$

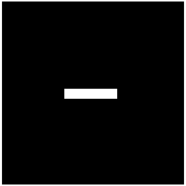
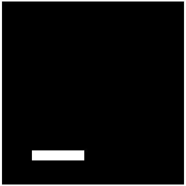
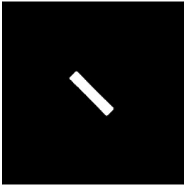


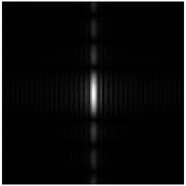
$Shift(|F(u,v)|)$



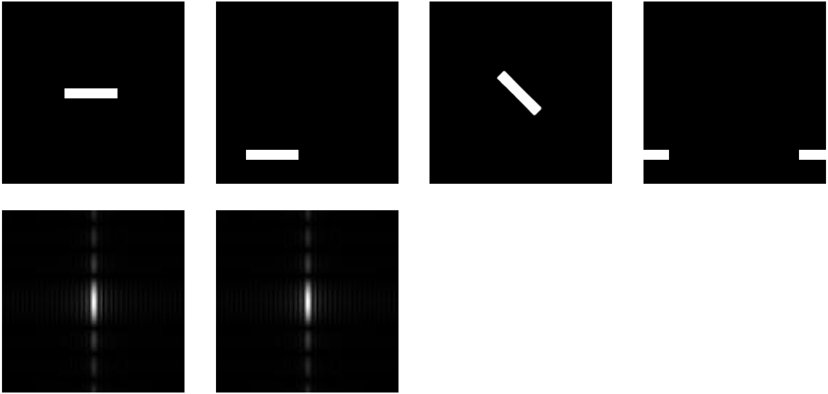
$\log(1 + Shift(|F(u,v)|))$

Examples

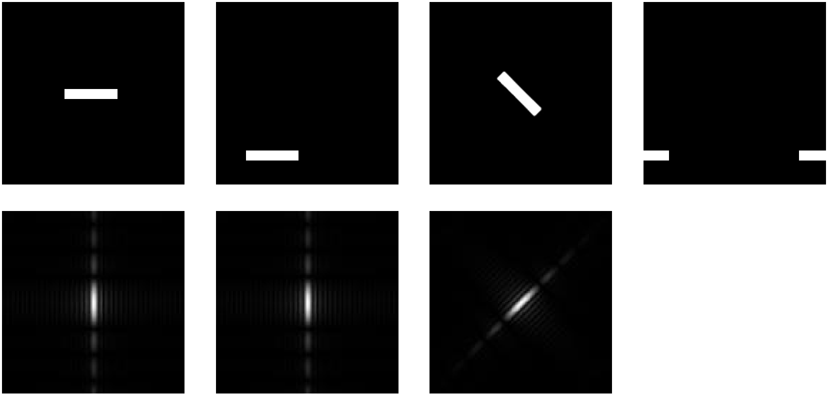






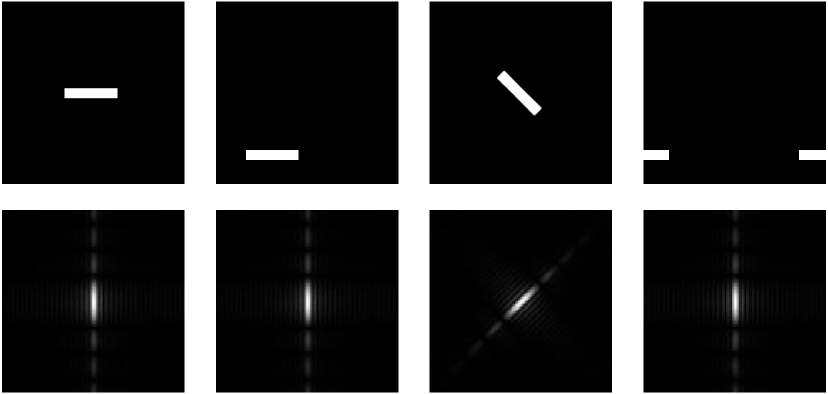
Examples



Examples



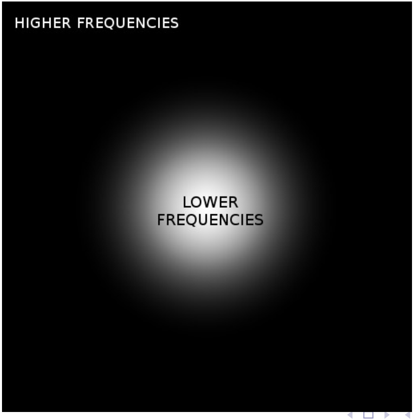
Examples



Visualizing the Fourier Spectrum

After FFT-Shift:

- In the central point of the image we have lower frequencies (with respect to u, v)
- Higher frequencies are further from the centre



Agenda

1. Big Picture
2. Discrete Fourier Transform
   - Convolution Theorem
3. Filtering in frequency domain
4. Fast Fourier Transform (FFT)

Convolution Theorem

- A circular convolution in 2D is given by:

$$f(x, y) * h(x, y) = \sum_{i=0}^{n-1} \sum_{j=0}^{m-1} f(i, j) h(x - i, y - j),$$

Convolution Theorem

- The 2D convolution theorem has the following result :

$$f(x,y) * h(x,y) \Leftrightarrow F(u,v) \cdot H(u,v),$$

$$f(x,y) \cdot h(x,y) \Leftrightarrow F(u,v) * H(u,v).$$

F and H are the Fourier Transforms of f and h, respectively

Agenda

Filtering in frequency domain

- Modify the Fourier transform of an image and calculate its inverse:

$$g(x, y) = \mathfrak{F}^{-1}\left[F(u, v)H(u, v)\right]$$

Filtering in frequency domain: steps

1. Input: $f(x, y)$ with size $m \times n$

Filtering in frequency domain: steps

1. Input: $f(x, y)$ with size $m \times n$
2. Obtain optimal values to perform the transform: P and Q. Options:
   - use the next power of two, or
   - use $P = 2n$ and $Q = 2m$.

Filtering in frequency domain: steps

1. Input: $f(x, y)$ with size $m \times n$
2. Obtain optimal values to perform the transform: P and Q. Options:
   - use the next power of two, or
   - use $P = 2n$ and $Q = 2m$.
3. Create an image $f_p(x, y)$ of size $P \times Q$, padding with zeros,

Filtering in frequency domain: steps

1. Input: $f(x, y)$ with size $m \times n$
2. Obtain optimal values to perform the transform: P and Q. Options:
   - use the next power of two, or
   - use $P = 2n$ and $Q = 2m$.
3. Create an image $f_p(x, y)$ of size $P \times Q$, padding with zeros,
4. Compute DFT $F(u, v) = \mathfrak{F}[f_p(x, y)]$,

Filtering in frequency domain: steps

1. Input: $f(x, y)$ with size $m \times n$
2. Obtain optimal values to perform the transform: P and Q. Options:
   - use the next power of two, or
   - use $P = 2n$ and $Q = 2m$.
3. Create an image $f_p(x, y)$ of size $P \times Q$, padding with zeros,
4. Compute DFT $F(u, v) = \mathfrak{F}[f_p(x, y)]$,
5. Use a symmetric filter $H(u, v)$ of size $P \times Q$ centred in $(P/2, Q/2)$,

Filtering in frequency domain: steps

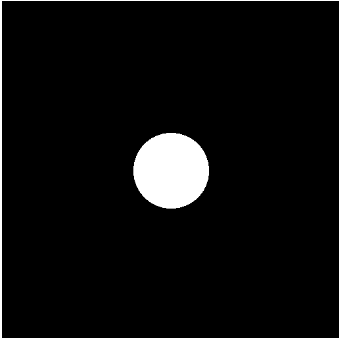
1. Input: $f(x, y)$ with size $m \times n$
2. Obtain optimal values to perform the transform: P and Q. Options:
   - use the next power of two, or
   - use $P = 2n$ and $Q = 2m$.
3. Create an image $f_p(x, y)$ of size $P \times Q$, padding with zeros,
4. Compute DFT $F(u, v) = \mathfrak{F}[f_p(x, y)]$,
5. Use a symmetric filter $H(u, v)$ of size $P \times Q$ centred in $(P/2, Q/2)$,
6. Compute processed image: $g_p(x, y) = real\left(\mathfrak{F}^{-1}[F(u, v)H(u, v)]\right)$,

Filtering in frequency domain: steps

1. Input: $f(x,y)$ with size $m \times n$
2. Obtain optimal values to perform the transform: P and Q. Options:
   - use the next power of two, or
   - use $P = 2n$ and $Q = 2m$.
3. Create an image $f_p(x,y)$ of size $P \times Q$, padding with zeros,
4. Compute DFT $F(u,v) = \mathfrak{F}[f_p(x,y)]$,
5. Use a symmetric filter $H(u,v)$ of size $P \times Q$ centred in $(P/2, Q/2)$,
6. Compute processed image: $g_p(x,y) = real\left(\mathfrak{F}^{-1}[F(u,v)H(u,v)]\right)$,
7. Output: $g(x,y)$ size $n \times m$, after removing the zero-padded region.

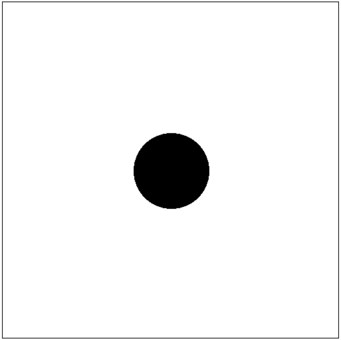
Filtering in frequency domain

- (*Low-Pass filter*):
  - Allow the low frequencies to "pass", retaining higher frequencies
  - An "ideal" filter removes all frequencies beyond a given threshold



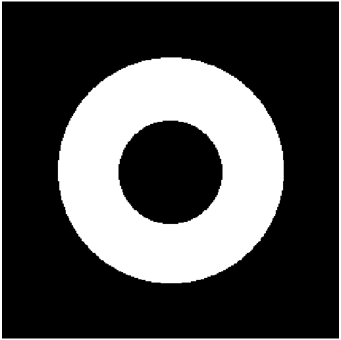
Filtering in frequency domain

- (*High-Pass filter*):
  - remove lower frequencies, while allowing to “pass” higher frequencies
  - An “ideal” high-pass filter removes all frequencies below a given threshold



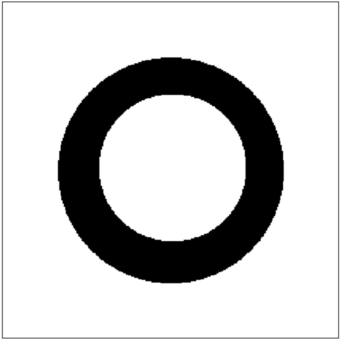
Filtering in frequency domain

- (*Bandpass filter*):
  - select a range of frequencies to be kept

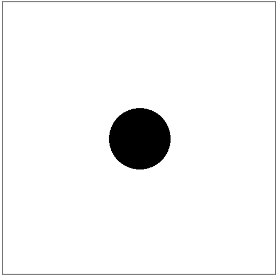


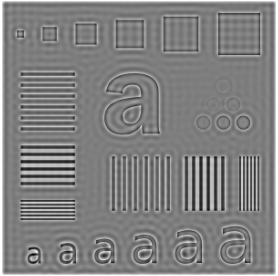
Filtering in frequency domain

- (*Band-stop filter*):
  - select a range of frequencies to be removed

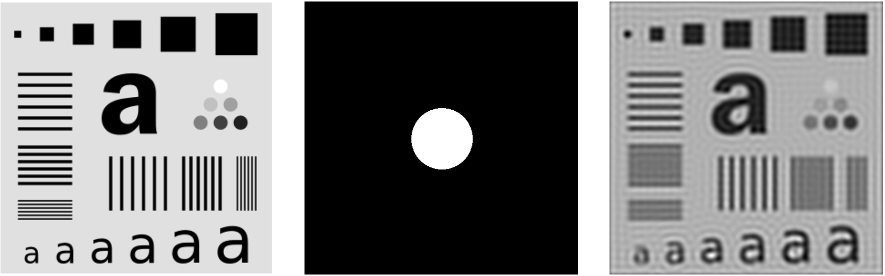


High-pass filter



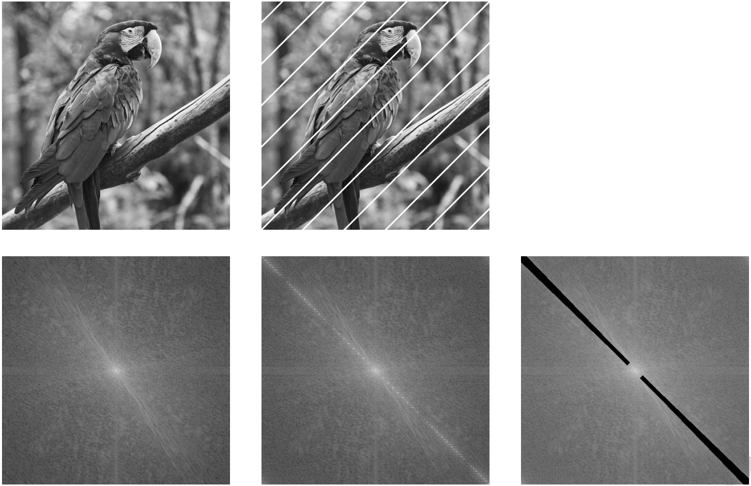


Low-pass filter

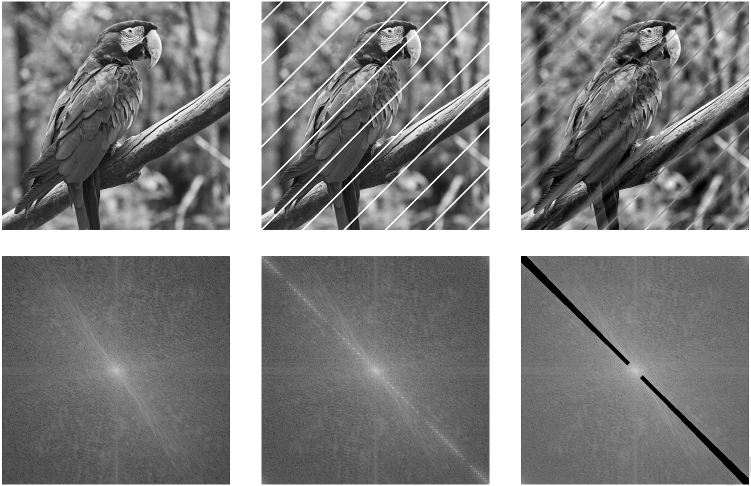


Band-stop filter

Band-stop filter



Band-stop filter



Agenda

Fast Fourier Transform (FFT)

- Efficient algorithm to compute DFT

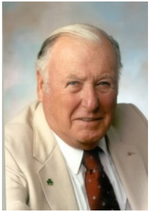
Fast Fourier Transform (FFT)

- Efficient algorithm to compute DFT
- Uses ideas from Gauss (1805)

Fast Fourier Transform (FFT)

- Efficient algorithm to compute DFT
- Uses ideas from Gauss (1805)
- The FFT Algorithm was published by Cooley and Tukey, 1965.

Fast Fourier Transform (FFT)



- James W. Cooley (1926–2016):
  - BSc., MSc., PhD. Mathematics;
  - Computer programmer with von Neumann at U.Princeton (1953-1956);
  - Researcher at IBM until 1991, his last position was professor at University of Rhode Island.
- John Tukey (1915–2000):
  - BSc., MSc. in Chemistry; PhD. Mathematics;
  - Professor at U.Princeton;
  - Researcher at AT&T Bell Labs;
  - Also known for developing (*boxplot*) and the *post-hoc* Tukey HSD test.

Fast Fourier Transform (FFT)

- Allows to compute Fourier Transform in $O(N \log_2 N)$

Fast Fourier Transform (FFT)

- Allows to compute Fourier Transform in $O(N \log_2 N)$
- There are several ways to implement. Originally assumed sample size in powers of two.

Fast Fourier Transform (FFT)

- Allows to compute Fourier Transform in $O(N \log_2 N)$
- There are several ways to implement. Originally assumed sample size in powers of two.
- The original algorithm uses the *successive duplication method*.
  - note that the DFT is a sequence of products and sums:

$$F(u) = \sum_{x=0}^{N-1} f(x) e^{-j \frac{2\pi u}{N} x},$$

Fast Fourier Transform (FFT)

- Allows to compute Fourier Transform in $O(N \log_2 N)$
- There are several ways to implement. Originally assumed sample size in powers of two.
- The original algorithm uses the *successive duplication method*.
    - note that the DFT is a sequence of products and sums:

$$F(u) = \sum_{x=0}^{N-1} f(x) e^{-j\frac{2\pi u}{N}x},$$

    - however, the term $e^{-j2\pi}$ is always the same, with different powers of $(u/N)x$

Fast Fourier Transform (FFT)

- Re-writing:

$$F(u) = \sum_{x=0}^{N-1} f(x) W_N^{ux},$$

where $W_N = e^{-j\frac{2\pi}{N}}$,

Properties:

Fast Fourier Transform (FFT)

- Re-writing:

$$F(u) = \sum_{x=0}^{N-1} f(x) W_N^{ux},$$

where $W_N = e^{-j\frac{2\pi}{N}}$,

Properties:

1. Complex conjugate **symmetry**:

$$W_N^{u(N-x)} = W_N^{-ux} = (W_N^{ux})^* \tag{1}$$

Fast Fourier Transform (FFT)

- Re-writing:

$$F(u) = \sum_{x=0}^{N-1} f(x) W_N^{ux},$$

where $W_N = e^{-j\frac{2\pi}{N}}$, and $W_N^{uN} = e^{-j2\pi u} = 1$

Properties:

1. Complex conjugate **symmetry**:

$$W_N^{u(N-x)} = W_N^{-ux} = (W_N^{ux})^* \tag{1}$$

Fast Fourier Transform (FFT)

- Re-writing:

$$F(u) = \sum_{x=0}^{N-1} f(x) W_N^{ux},$$

where $W_N = e^{-j\frac{2\pi}{N}}$, and $W_N^{uN} = e^{-j2\pi u} = 1$

Properties:

1. Complex conjugate **symmetry**:

$$W_N^{u(N-x)} = W_N^{-ux} = (W_N^{ux})^* \quad (1)$$

2. **Periodicity** in x, u:

$$W_N^{ux} = W_N^{u(x+N)} = W_N^{(u+N)x} \quad (2)$$

Fast Fourier Transform (FFT)

- Decimate: obtain DFT using smaller parts assuming $N = 2^m$
  - decompose $f(x)$ into indexes:
    even (index $x = 2r$) and **odd** (index $x = 2r + 1$):

$$F(u) = \sum_{r=0}^{(N/2)-1} f(2r)W_N^{u2r} + \sum_{r=0}^{(N/2)-1} f(2r+1)W_N^{u(2r+1)},$$

Fast Fourier Transform (FFT)

- Decimate: obtain DFT using smaller parts assuming $N = 2^m$
  - decompose $f(x)$ into indexes:
    even (index $x = 2r$) and **odd** (index $x = 2r + 1$):

$$
\begin{aligned}
F(u) &= \sum_{r=0}^{(N/2)-1} f(2r) W_N^{u2r} + \sum_{r=0}^{(N/2)-1} f(2r+1) W_N^{u(2r+1)}, \\
&= \sum_{r=0}^{(N/2)-1} f(2r) W_N^{u2r} + W_N^{u} \sum_{r=0}^{(N/2)-1} f(2r+1) W_N^{u2r},
\end{aligned}
$$

Fast Fourier Transform (FFT)

- Decimate: obtain DFT using smaller parts assuming $N = 2^m$
  - decompose $f(x)$ into indexes:
    even (index $x = 2r$) and **odd** (index $x = 2r + 1$):

$$
\begin{aligned}
F(u) &= \sum_{r=0}^{(N/2)-1} f(2r) W_N^{u2r} + \sum_{r=0}^{(N/2)-1} f(2r+1) W_N^{u(2r+1)}, \\
&= \sum_{r=0}^{(N/2)-1} f(2r) W_N^{u2r} + W_N^{u} \sum_{r=0}^{(N/2)-1} f(2r+1) W_N^{u2r}, \\
&= \sum_{r=0}^{(N/2)-1} f(2r) (W_N^{2})^{ux} + W_N^{u} \sum_{r=0}^{(N/2)-1} f(2r+1) (W_N^{2})^{ux},
\end{aligned}
$$

Fast Fourier Transform (FFT)

- Due to the periodicity of DFT:

$$F\left(u + {}^{N}/_{2}\right) = F(u)$$

- We can write the transform as:

$$F(u) = \begin{cases} F_e(u) + W_N^{ux} F_o(u), & \text{for } 0 \leq u < N/2 \\ F_e(u - N/2) + W_N^{ux} F_o(u - N/2), & \text{for } N/2 \leq u < N/2 \end{cases}$$

Fast Fourier Transform (FFT)

- By using the relationship between the terms W:

$$W_N^2 = e^{-j\frac{2\pi}{N}2} = e^{-j\frac{4\pi}{N}} = e^{-j\frac{2\pi}{N/2}} = W_{N/2}$$

  - Split the DFT into a sum of 2 DFTs with even and odd indices:

$$F(u) = \sum_{r=0}^{(N/2)-1} f(2r)W_{N/2}^{ux} + W_N^u \sum_{r=0}^{(N/2)-1} f(2r+1)W_{N/2}^{ux},$$

$$F(u) = F_e(u) + W_N^u \cdot F_o(u)$$

Fast Fourier Transform (FFT) – running time complexity

- For each $N/2$ we now have $2 \cdot N/4$
- Repeating the procedure

$$1 : \frac{N}{2} \rightarrow 2\left(\frac{N}{2}\right)^2 + N = \frac{N^2}{2} + N$$

Fast Fourier Transform (FFT) – running time complexity

- For each $N/2$ we now have $2 \cdot N/4$
- Repeating the procedure

$$1 : \frac{N}{2} \rightarrow 2\left(\frac{N}{2}\right)^2 + N = \frac{N^2}{2} + N$$

$$2 : \frac{N}{4} \rightarrow 2\left(2\left(\frac{N}{4}\right)^2 + \frac{N}{2}\right)^2 + N = \frac{N^2}{4} + 2N$$

Fast Fourier Transform (FFT) – running time complexity

- For each $N/2$ we now have $2 \cdot N/4$
- Repeating the procedure

$$1 : \frac{N}{2} \rightarrow 2\left(\frac{N}{2}\right)^2 + N = \frac{N^2}{2} + N$$

$$2 : \frac{N}{4} \rightarrow 2\left(2\left(\frac{N}{4}\right)^2 + \frac{N}{2}\right)^2 + N = \frac{N^2}{4} + 2N$$

$$3 : \frac{N}{8} \rightarrow 2\left(2\left(2\left(\frac{N}{8}\right)^2 + \frac{N}{4}\right) + \frac{N}{2}\right) + N = \frac{N^2}{8} + 3N$$

Fast Fourier Transform (FFT) – running time complexity

- For each $N/2$ we now have $2 \cdot N/4$
- Repeating the procedure

$$1 : \frac{N}{2} \rightarrow 2\left(\frac{N}{2}\right)^2 + N = \frac{N^2}{2} + N$$

$$2 : \frac{N}{4} \rightarrow 2\left(2\left(\frac{N}{4}\right)^2 + \frac{N}{2}\right)^2 + N = \frac{N^2}{4} + 2N$$

$$3 : \frac{N}{8} \rightarrow 2\left(2\left(2\left(\frac{N}{8}\right)^2 + \frac{N}{4}\right) + \frac{N}{2}\right) + N = \frac{N^2}{8} + 3N$$

$$\ldots\ldots$$

Fast Fourier Transform (FFT) – running time complexity

- For each $N/2$ we now have $2 \cdot N/4$
- Repeating the procedure

$$1 : \frac{N}{2} \rightarrow 2\left(\frac{N}{2}\right)^2 + N = \frac{N^2}{2} + N$$

$$2 : \frac{N}{4} \rightarrow 2\left(2\left(\frac{N}{4}\right)^2 + \frac{N}{2}\right)^2 + N = \frac{N^2}{4} + 2N$$

$$3 : \frac{N}{8} \rightarrow 2\left(2\left(2\left(\frac{N}{8}\right)^2 + \frac{N}{4}\right) + \frac{N}{2}\right) + N = \frac{N^2}{8} + 3N$$

$$\ldots\ldots$$

$$P : \frac{N}{2^P} \rightarrow \frac{N^2}{2^P} + pN = N + N\log_2 N$$

Bibliography I

GONZALEZ, R.C.; WOODS, R.E. ⋆
Processamento Digital de Imagens, 3.ed
Capítulo 4.
Pearson, 2010.