The Fourier Transform

Fourier Transform: Overview
- Why FT is useful
- 1D FT, DFT, 2D DFT
- FT properties
- Linear Filters

Why Fourier Transform?
- FT helps to analyze
  - Sampling artifacts
  - Linear Filters
- Some interesting image transformation
- Nice properties for pattern matching or classification

FT maps a function to its frequencies

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

Angular frequency

Continuous function

FT of a pulse function

Spatial: Pulse

Frequency: sinc

$sinc(x) = \frac{\sin(x)}{x}$

What is FT?

FT decomposes a function into a weighted sum of sinusoidal functions

$\Rightarrow$ We can reconstruct the original function:

$p(t) = \frac{1}{2\pi} \int_{-\infty}^{\infty} F_p(\omega)e^{i\omega t} \, d\omega$
Representing FT

- FT is complex
- Representation:
  - Real / Imaginary
  - Magnitude / Phase

Discrete Sampling

- Sampling at low frequency
- Sampling at high frequency

1-D Discreet Fourier Transform

- Assumptions:
  - Sampling criterion satisfied
  - Sampled function replicates to infinity

Forward DFT: \( \hat{F}_p = \frac{1}{\sqrt{N}} \sum_{x=0}^{N-1} f_x e^{-j \frac{2\pi}{N} px} \)

Inverse DFT: \( f_x = \sum_{x=0}^{N-1} \hat{F}_p e^{j \frac{2\pi}{N} px} \)

Sampling a rotating wheel

- Oversampled rotating wheel:
- Same wheel, undersampled:

SUFFICIENT SAMPLING RATE

- Oversampled rotating wheel:
- Same wheel, undersampled:

SAMPLING ARTIFACTS
INSUFFICIENT SAMPLING RATE

563x25]3

NYQUIST THEOREM

• The sample frequency must be at least twice the highest frequency present for a signal to be reconstructed from a sampled version.

2-D Discrete Fourier Transform

\[
F_{p,u,v} = \frac{1}{\sqrt{MN}} \sum_{x=0}^{M-1} \sum_{y=0}^{N-1} p_{x,y} e^{i2\pi(u/M + v/N)}
\]

\[
p_{x,y} = \frac{1}{\sqrt{MN}} \sum_{u=0}^{M-1} \sum_{v=0}^{N-1} F_{p,u,v} e^{-i2\pi(u/M + v/N)}
\]

Bracewell, chap. 11

Decomposition into sinusoidal functions

Real part of

\[e^{i2\pi((u+\nu) + vy)}\]

where

\[\sqrt{u^2 + v^2}\]

represents the frequency

\[a \tan(v, u)\]

represents the orientation

2D Pulse FT

Square Pulse 2D sinc function

HORIZONTAL AND VERTICAL STRUCTURES

2 pixel wide stripes:
• Vertical structures
• Half the max freq.

Horizontal text:
• Horizontal structures
• Line spacing
**Phase and Magnitude**

- Magnitude of the transform
- Phase of the transform

**Phase and Magnitude**

- Magnitude of the transform
- Phase of the transform

**Switching Phase and Magnitude**

- Zebra phase
- Cheetah magnitude
- Cheetah phase
- Zebra magnitude

**FT is Shift Invariant**

- After shifting:
  - Magnitude stay constant
  - Phase changes

**Rotation**

- FT of a rotated image also rotates
- Image replication do not replicate for every angle

**Spectral Analysis**

The magnitude of the DFT captures the main orientations in the image.
Frequency Scaling
- Spatial compression
- Frequency increase

FT Interpolation
1. Compute DFT
2. Add zeros at both ends
3. Inverse DFT

Superposition
\[ F[p_1 + p_2] = F[p_1] + F[p_2] \]

Removing Noise

Frequency Cut
\[ = \]

Reconstruction
Multiplication In Fourier Domain

Multiplication in Fourier Domain can suppress unwanted frequencies.
Removing high freq = smoothing

Fast Fourier Transform (FFT)

\[ F(\mu) = \sum_{n=0}^{N-1} f(n)e^{-i2\pi n\mu/N} \] with \( N = 2^a \)
\[ = \frac{1}{2} \left( \sum_{n=0}^{N/2-1} f(2n)e^{-i2\pi n\mu/2} + \sum_{n=0}^{N/2-1} f(2n+1)e^{-i2\pi n\mu/2} \right) \]
\[ = \frac{1}{2} \left( F_{even}(\mu) + F_{odd}(\mu) \right) \]
where \( F_{even} \) and \( F_{odd} \) are DFTs over \( N/2 \) points from 0 to \( M-1 \).

Fast Fourier Transform

Since \( o_{n\mu}^0 = o_{n\mu} \) and \( o_{n\mu}^0 = -o_{n\mu}^0 \), we can write
\[ F(\mu) = \frac{1}{2} \left( F_{even}(\mu) + o_{n\mu}^0 F_{odd}(\mu) \right) \]
\[ F(\mu + M) = \frac{1}{2} \left( F_{even}(\mu) - o_{n\mu}^0 F_{odd}(\mu) \right) \]
We can compute an \( N \)-point DFT by:
1. Computing \( F_{even} \) and \( F_{odd} \) for \( \mu \) from 0..M-1,
2. Adding them to obtain \( F \) for \( m \) from 0..N-1.
\[ \text{Total number of required multiplications is} \]
\[ T(n) = 2T(n-1) + 2^{n-2} \log_2(2^n) = 1/2N \log_2(N) \]
with \( N = 2^n \)

C CODE FOR THE 1D CASE

DFT vs FFT

Computational Complexity in the 1D Case

\[ F(\mu) = \frac{1}{\sqrt{N}} \sum_{n=0}^{N-1} f(n)e^{-i2\pi n\mu/N} \]

Ordinary Fourier Transform:
\[ \rightarrow O(N^2) \text{ complexity} \]
Fast Fourier Transform:
\[ \rightarrow O(N \log_2(N)) \text{ complexity} \]
2-D FFT Complexity

\[ F(\mu, \nu) = \frac{1}{\sqrt{MN}} \sum_{x=0}^{M-1} \sum_{y=0}^{N-1} f(x,y)e^{-2\pi i (\mu x/M + \nu y/N)} \]

\[ = \frac{1}{\sqrt{MN}} \sum_{x=0}^{M-1} \sum_{y=0}^{N-1} f(x,y)e^{-2\pi i (\nu y/N - \mu x/M)} \]

We can compute a two-dimensional FT by
1. performing a one-dimensional FFT for each column of \( f(x,y) \),
2. performing a one-dimensional FFT for each row on the resulting values.
This requires a total of \( 2N \) one dimensional transforms
\[ \rightarrow O(N^2 \log_2 (N)) \text{ complexity} \]

Filters

• A black box transforming an image

Linear Filters: Definition

• Does not depend on image location
• \( F(i+J) = F(i) + F(j) \)
• \( F(ki) = kF(i) \)
• How to define such a Filter?
  - With the impulse response.

Convolution

• Weighted pixel sum within a neighborhood
• Convolution operator: \( I**H \)

Smoothing by Averaging

\[ R_k = \frac{1}{(2k+1)^2} \sum_{i=-k}^{k} \sum_{j=-k}^{k} I_{ij} \]

\[ = \sum_{i=-k}^{k} \sum_{j=-k}^{k} H_{i,j} I_{ij} \]

where:

\[ H = \frac{1}{(2k+1)^2} \begin{bmatrix}
0 & 0 & 0 & 0 \\
0 & 1 & 1 & 0 \\
0 & 0 & 1 & 0 \\
0 & 0 & 0 & 0
\end{bmatrix} \]

Computed Pixel

Original Image \( I \)

Convolution Kernel

Mask Pulse Response

Constant Kernel

Convolution kernel
Transfer Function

\[ x \ast \ast = 1 \]

Convoltion and Fourier Transform

- A convolution in spatial domain is a multiplication in Fourier domain
- \( F[p_1(t) \ast p_2(t)] = \int_{-\infty}^{\infty} p_1(\tau) p_2(t - \tau) e^{-j\omega t} d\tau \)
- \( F[p_1(t) \ast p_2(t)] = \int_{-\infty}^{\infty} \left( \int_{-\infty}^{\infty} p_1(\tau) p_2(t - \tau) e^{-j\omega t} d\tau \right) e^{-j\omega t} d\tau = \int_{-\infty}^{\infty} F[p_2(\omega)] p_1(\tau) e^{-j\omega \tau} d\tau = \int_{-\infty}^{\infty} F[p_2(\omega)] F[p_1(\omega)] \)

Gaussian Smoothing

\( g_\sigma(x, y) = \frac{1}{2\pi\sigma^2} e^{-\left((x^2 + y^2) / 2\sigma^2 \right)} \)

A Gaussian FT is a Gaussian

63x63 Gaussian Kernel Its Fourier Transform

Gaussian Blur VS Averaging

Convolution & Fourier

- FT can compute a convolution:
- It is easier to understand a convolution kernel in frequency domain
Template Matching

Texture Classification

Subsampling Artifacts

- Particularly noticeable in high frequency areas, such as on the hair.

Sampling Without Smoothing

Problem:
- High frequencies lead to trouble with sampling.

Solution:
- Suppress high frequencies before sampling by
  1. multiplying DFT of the signal with something that suppresses high frequencies
  2. convolving with a low-pass filter

Sampling Using a Gaussian of Variance 2 to Smooth
LOSS OF DETAILS BUT NOT ARTIFACTS

No aliasing but details are lost as high frequencies are progressively removed.

Fourier Transform in Short

• Computation:
  – With Fast Fourier Transform
  – Complexity: $O(N^2 \log_2(N))$

• Applications:
  – Convolution computation
  – Linear Filters design
  – Correlation: template matching
  – Texture Classification

Exercises: Which is which?

Convolving $M$ with...

- A constant white image (1)?
- An image with only horizontal lines?
- A black image (0), except a single white pixel (1)?
- An black image (0), except a 5 by 5 white square (1)?

More Exercises...

• You can try in ImageJ:
  – Load an image
  – Duplicate it
  – Process/Filter/Gaussian Blur
  – Process/FFT/FFT or Inverse FFT
  – Compare original and blurred FT