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

Fourier Transform of p

$i^2 = -1$

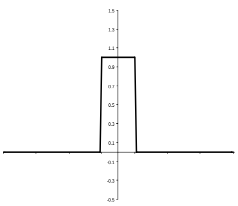
$$\mathrm{Fp}(\omega) = \int_{-\infty}^{\infty} \mathrm{p}(t) e^{-i\omega t} dt$$

$e^{-i\omega t} = \cos(\omega t) - i \sin(\omega t)$

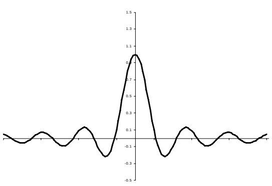
Angular frequency

Continuous function

FT of a pulse function



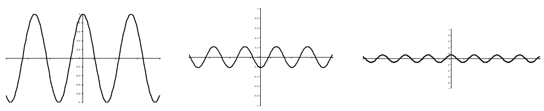
Spatial: Pulse



Frequency: sinc

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

What is FT ?

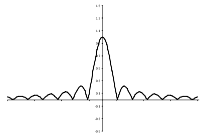


FT decomposes a function into a weighted sum of sinusoidal functions
=> We can reconstruct the original function:

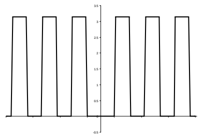
$$\mathrm{p}(t) = \frac{1}{2\pi} \int_{-\infty}^{\infty} \mathrm{Fp}(\omega) e^{i\omega t} d\omega$$

Representing FT

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

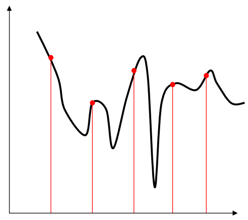


Magnitude

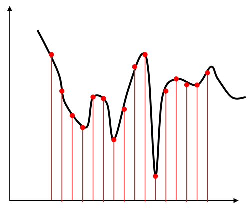


Phase

Discreet Sampling



Sampling at low frequency



Sampling at high frequency

1-D Discreet Fourier Transform

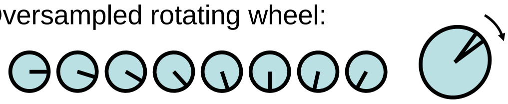
- Assumptions:
  - Sampling criterion satisfied
  - Sampled function replicates to infinity

$$\text{Forward DFT}: \; \mathrm{Fp}_u = \frac{1}{\sqrt{N}} \sum_{x=0}^{N-1} \mathrm{p}_x \, e^{-i\left(\frac{2\pi}{N}\right)xu}$$

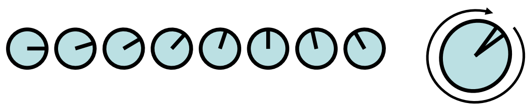
$$\text{Inverse DFT}: \; \mathrm{p}_x = \sum_{x=0}^{N-1} \mathrm{Fp}_u \, e^{i\left(\frac{2\pi}{N}\right)ux}$$

Sampling a rotating wheel

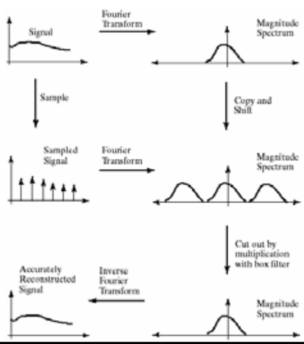
- Oversampled rotating wheel:



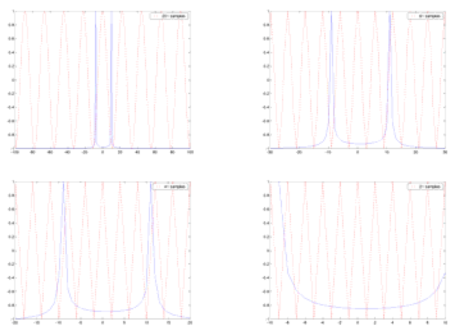
- Same wheel, undersampled:



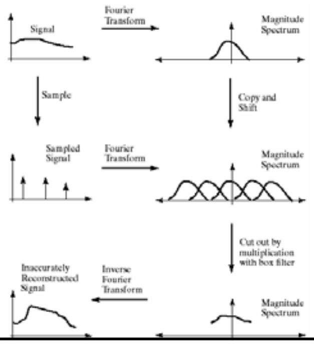
SUFFICIENT SAMPLING RATE



SAMPLING ARTIFACTS



INSUFFICIENT SAMPLING RATE



NYQUIST THEOREM

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

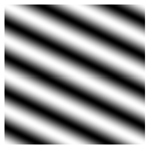
2-D Discrete Fourier Transform

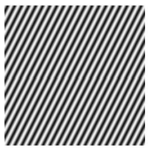
$$\mathrm{Fp}_{\mu,\nu} = \frac{1}{\sqrt{MN}} \sum_{x=0}^{M-1} \sum_{y=0}^{N-1} \mathrm{p}_{x,y}\, e^{-i2\pi(\mu x/M + \nu y/N)}$$

$$\mathrm{p}_{x,y} = \frac{1}{\sqrt{MN}} \sum_{\mu=0}^{M-1} \sum_{\nu=0}^{N-1} \mathrm{Fp}_{\mu,\nu}\, e^{+i2\pi(\mu x/M + \nu y/N)}$$

Bracewell, chap. 11

Decomposition into sinusoidal functions





Real part of

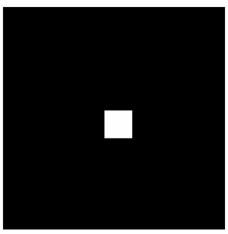
$$e^{+i2\pi(ux+vy)}$$

where

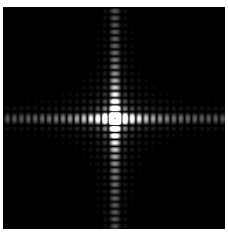
$\sqrt{u^2+v^2}$ represents the frequency

$a\tan(v,u)$ represents the orientation

2D Pulse FT



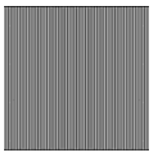
FT ->

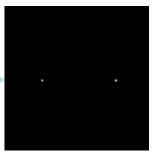


Square Pulse

2D sinc function

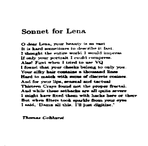
HORIZONTAL AND VERTICAL STUCTURES

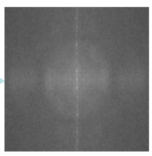




2 pixel wide stripes:

- Vertical structures
- Half the max freq.

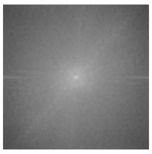




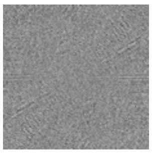
Horizontal text:

- Horizontal structures
- Line spacing

PHASE AND MAGNITUDE

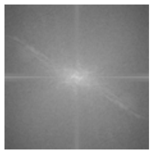


Magnitude of the transform

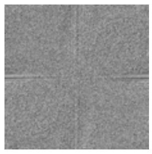


Phase of the transform

PHASE AND MAGNITUDE

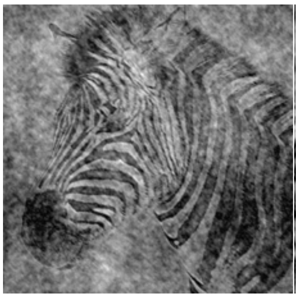


Magnitude of the transform

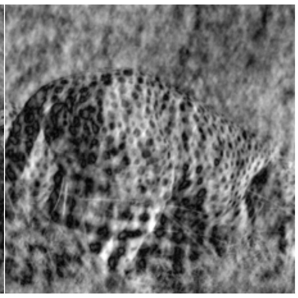


Phase of the transform

SWITCHING PHASE AND MAGNITUDE



- Zebra phase
- Cheetah magnitude



- Cheetah phase
- Zebra magnitude

FT is Shift Invariant

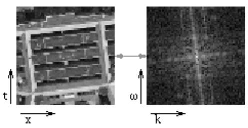
After shifting:

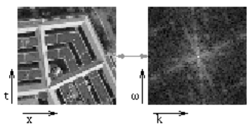
- Magnitude stay constant
- Phase changes

Rotation

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

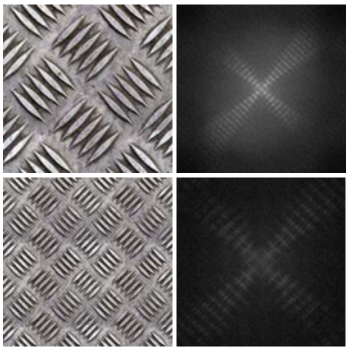
Spectral Analysis





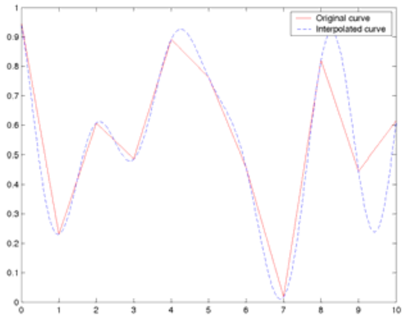
The magnitude of the DFT captures the main orientations in the image.

Frequency Scaling



- Spacial compression
- Frequency increase

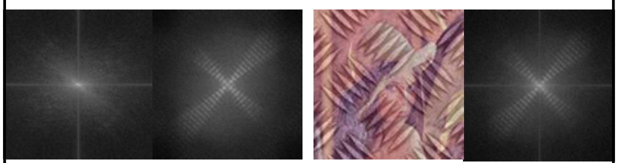
FT Interpolation



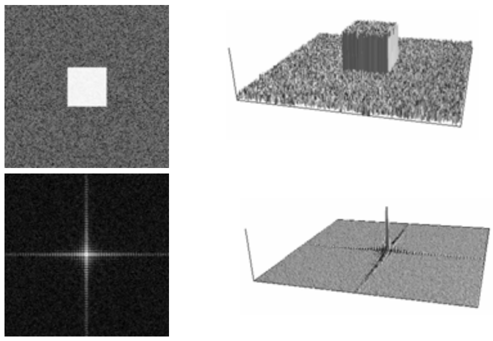
1. Compute DFT
2. Add zeros at both ends
3. Inverse DFT

Superposition

$$\mathrm{F}[p_1 + p_2] = \mathrm{F}[p_1] + \mathrm{F}[p_2]$$

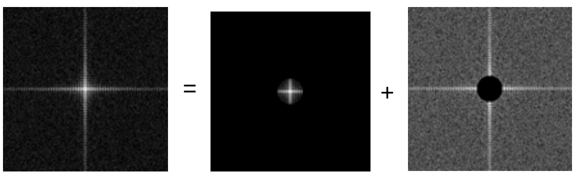


Removing Noise

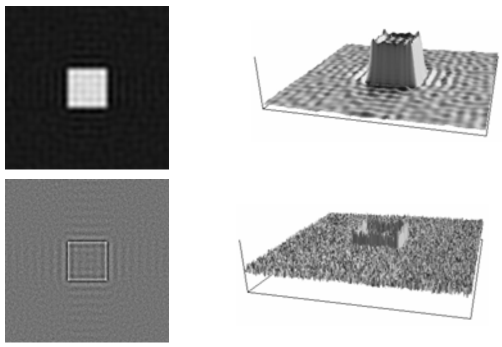


http://local.wasp.uwa.edu.au/~pbourke/other/imagefilter/index.html

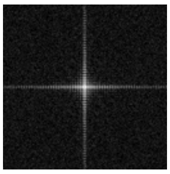
Frequency Cut

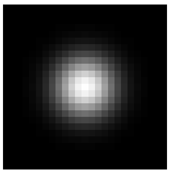


Reconstruction



Multiplication In Fourier Domain





Multiplication in Fourier Domain can suppress unwanted frequencies.

Removing high freq = smoothing

Fast Fourier Transform

$$\{8,7,6,5,4,3,2,1\} = \{8,0,6,0,4,0,2,0\} + \{0,7,0,5,0,3,0,1\}$$

$$\{8,6,4,2\} \rightarrow \{A,B,C,D\}$$

$$\{8,0,6,0,4,0,2,0\} \rightarrow \{A,B,C,D,A,B,C,D\} \text{ (Stretching Theorem)}$$

$$\{7,5,3,1\} \rightarrow \{P,Q,R,S\}$$

$$\{7,0,5,0,3,0,1,0\} \rightarrow \{P,Q,R,S,P,Q,R,S\} \text{ (Stretching Theorem)}$$

$$\{0,7,0,5,0,3,0,1\} \rightarrow \{P,WQ,W^2R,W^3S,W^4P,W^5Q,W^6R,W^7S\}$$

$$\text{with } W = \exp(-2i\pi/8) \text{ (Shift Theorem)}$$

Bracewell, Chap. 11

Fast Fourier Transform (FFT)

$$F(\mu) \propto \sum_{x=0}^{N-1} f(x) e^{-i2\pi(\mu x/N)} \text{ with } N = 2^n$$

$$\propto \sum_{x=0}^{2M-1} f(x)\omega_{2M}^{x\mu} \text{ with } M = N/2 \text{ and } \omega_n = e^{-i2\pi n}$$

$$\propto \frac{1}{2}\left(\sum_{x=0}^{M-1} f(2x)\omega_{2M}^{(2x)\mu} + \sum_{x=0}^{M-1} f(2x+1)\omega_{2M}^{(2x+1)\mu}\right)$$

$$\propto \frac{1}{2}\left(\sum_{x=0}^{M-1} f(2x)\omega_{M}^{x\mu} + \sum_{x=0}^{M-1} f(2x+1)\omega_{M}^{x\mu}\omega_{2M}^{\mu}\right)$$

$$\propto \frac{1}{2}\left(F_{even}(\mu) + \omega_{2M}^{\mu} F_{odd}(\mu)\right)$$

where $= F_{even}$ and F_{odd} are DFTs over N/2 points from 0 to M -1.

Fast Fourier Transform

Since $\omega_M^{M+u} = \omega_M^u$ and $\omega_{2M}^{M+u} = -\omega_{2M}^u$ we can write

$$F(\mu) \propto \frac{1}{2}\left(F_{even}(\mu) + \omega_{2M}^{\mu} F_{odd}(\mu)\right)$$

$$F(\mu + M) \propto \frac{1}{2}\left(F_{even}(\mu) - \omega_{2M}^{\mu} F_{odd}(\mu)\right)$$

We can compute an *N*-point DFT by:

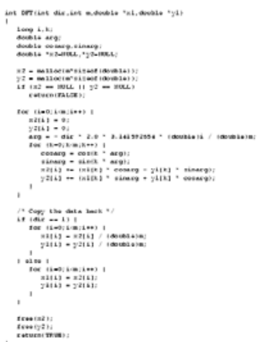
1. *Computing* F_{even} and F_{odd} for μ from 0..M-1,
2. Adding them to obtain F for m from 0..N-1.

→Total number of required multiplications is

$$T(n) = 2T(n-1) + 2^{(n-1)} = 2^{(n-1)}\log_2(2^n) = 1/2N\log_2(N)$$

with $N = 2^n$

C CODE FOR THE 1D CASE
DFT vs FFT



Computational Complexity in the 1D Case

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

Ordinary Fourier Transform :

$\rightarrow O(N^2)$ complexity

Fast Fourier Transform :

$\rightarrow O(N\log_2(N))$ complexity

2-D FFT Complexity

$$F(u,v)=\frac{1}{\sqrt{MN}}\sum_{x=0}^{M-1}\sum_{y=0}^{N-1}f(x,y)e^{-i2\pi(ux/M+vy/N)}$$
$$=\frac{1}{\sqrt{MN}}\sum_{x=0}^{M-1}\left(\sum_{y=0}^{N-1}f(x,y)e^{-i2\pi(vy/N)}\right)e^{-i2\pi(ux/M)}$$

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 2 N one dimensional transforms

$$\rightarrow O(N^2\log_2(N))\text{ complexity}$$

Filters

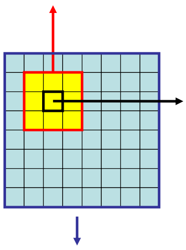
- A black box transforming an image

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.

$$\begin{bmatrix} \ddots & & \vdots & & \iddots \\ & 0 & 0 & 0 & \\ \cdots & 0 & 1 & 0 & \cdots \\ & 0 & 0 & 0 & \\ \iddots & & \vdots & & \ddots \end{bmatrix} \rightarrow \begin{bmatrix} \ddots & & & \vdots & & & \iddots \\ & 0 & 0 & 0 & 0 & 0 & \\ & 0 & 1 & 2 & 3 & 0 & \\ \cdots & 0 & 4 & 5 & 6 & 0 & \cdots \\ & 0 & 7 & 8 & 9 & 0 & \\ & 0 & 0 & 0 & 0 & 0 & \\ \iddots & & & \vdots & & & \ddots \end{bmatrix}$$

Convolution



- Weighted pixel sum within a neighborhood
- Convolution operator: $I**H$

$$\begin{bmatrix}9&8&7\\6&5&4\\3&2&1\end{bmatrix}**\begin{bmatrix}0&0&0\\0&1&0\\0&0&0\end{bmatrix}=\begin{bmatrix}1&2&3\\4&5&6\\7&8&9\end{bmatrix}$$

Mask Pulse Response

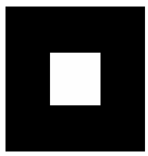
Smoothing by Averaging

Constant Kernel

$$R_{ij}=\frac{1}{(2k+1)^2}\sum_{u=i-k}^{u=i+k}\sum_{v=j-k}^{v=j+k}I_{uv}$$
$$=\sum_{u,v}H_{i-u,j-v}I_{uv}$$

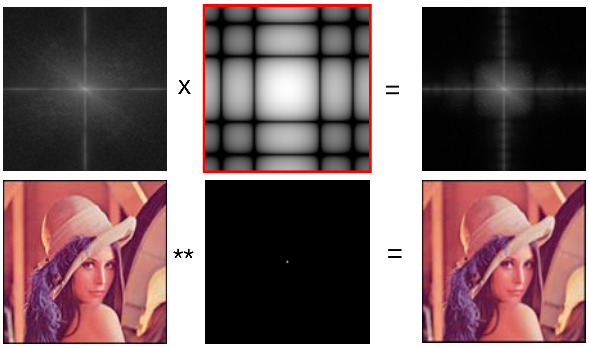
where :

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



Convolution kernel

Transfer Function



Convolution and Fourier Transform

- A convolution in spatial domain is a multiplication in Fourier domain

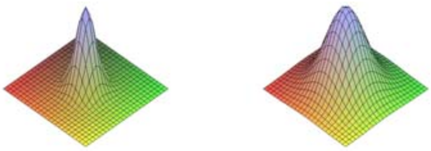
$$\text{1-D Convolution}: p_1(t) * p_2(t) = \int_{-\infty}^{\infty} p_1(\tau)\, p_2(t-\tau)\, d\tau$$

$$F[p_1(t) * p_2(t)] = \int_{-\infty}^{\infty}\left(\int_{-\infty}^{\infty} p_1(\tau)\, p_2(t-\tau)\, d\tau\right) e^{-i\omega t} dt =$$

$$= \int_{-\infty}^{\infty}\left(\int_{-\infty}^{\infty} p_2(t-\tau)\, e^{-i\omega t} dt\right) p_1(\tau)\, d\tau \;= \int_{-\infty}^{\infty} Fp_2(\omega)\, p_1(\tau)\, e^{-i\omega t} d\tau$$

$$= Fp_2(\omega)\int_{-\infty}^{\infty} p_1(\tau)\, e^{-i\omega t} d\tau \;= Fp_2(\omega)\, Fp_1(\omega)$$

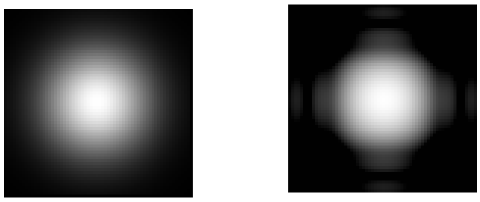
Gaussian Smoothing



$\sigma = 1$ $\qquad$ $\sigma = 2$

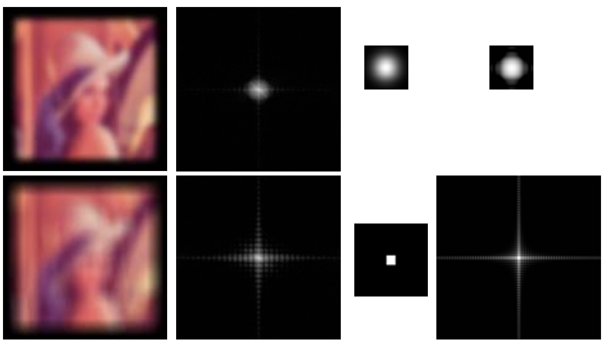
$$g_2(x, y) = \frac{1}{2\pi\sigma^2}\exp(-(x^2 + y^2)/2\sigma^2)$$

A Gaussian FT is a Gaussian



63x63 Gaussian Kernel $\qquad$ Its Fourier Transform

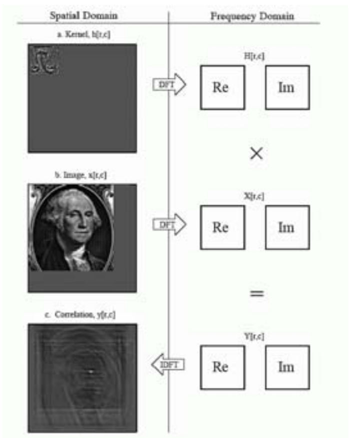
Gaussian Blur VS Averaging



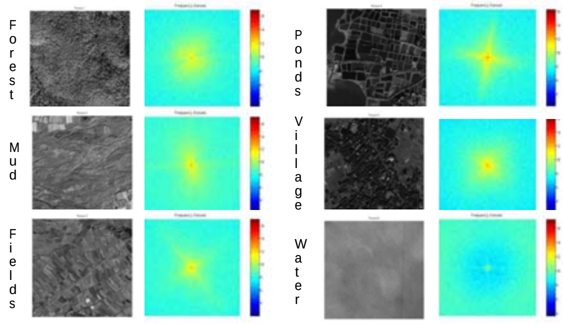
Convolution & Fourier

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

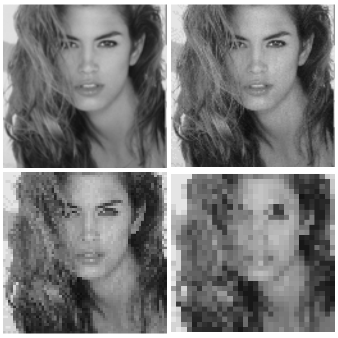
Template Matching



TEXTURE CLASSIFICATION

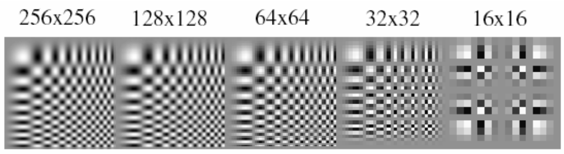


SUBSAMPLING ARTIFACTS

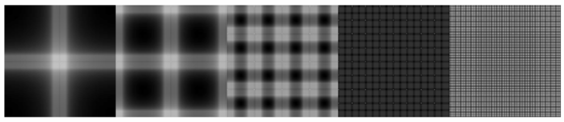


- Particularly noticeable in high frequency areas, such as on the hair.

SAMPLING WITHOUT SMOOTHING



Images sampled at every other pixel



Magnitude spectra of these images

SMOOTHING AS LOW-PASS FILTERING

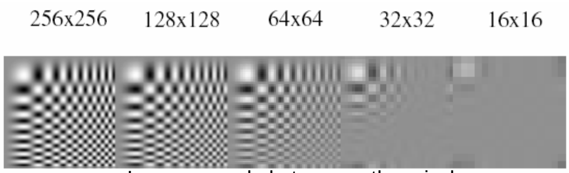
Problem:

- High frequencies lead to trouble with sampling.

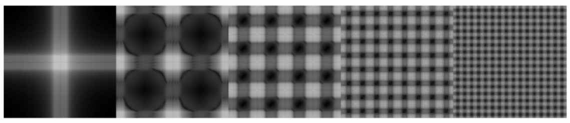
Solution:

- 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

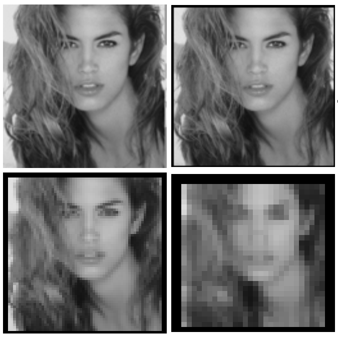


Images sampled at every other pixel



Magnitude spectra of these images

LOSS OF DETAILS BUT NOT ARTIFACTS

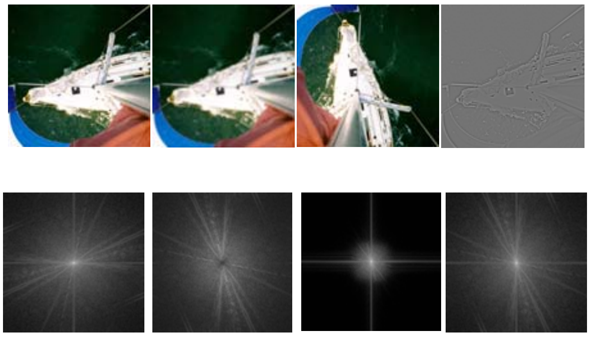


→No aliasing but details are lost as high frequencies are progressively removed.

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 ?



Convolution

$$\text{Consider the following mask : M} = \begin{bmatrix} -1 & 0 & 1 \\ -1 & 0 & 1 \\ -1 & 0 & 1 \end{bmatrix}$$

- What would give 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 blured FT