Review – Frequency Domain Techniques

- Thinking images in terms of frequency.
- Treat images as infinite-size, continuous periodic functions.

Review - Fourier Transform

We want to understand the frequency \( w \) of our signal. So, let’s reparametrize the signal by \( w \) instead of \( x \):

\[
\begin{align*}
    f(x) & \quad \xrightarrow{\text{Fourier Transform}} \quad F(w) \\

    \text{For every } w \text{ from 0 to inf, } F(w) & \text{ holds the amplitude } A \text{ and phase } \phi \text{ of the corresponding sine } \\
    \quad \quad A \sin(\omega x + \phi)
\end{align*}
\]

- How can \( F \) hold both? Complex number trick!

\[
F(\omega) = R(\omega) + iI(\omega) \\
A = \pm \sqrt{R(\omega)^2 + I(\omega)^2} \\
\phi = \tan^{-1} \frac{I(\omega)}{R(\omega)}
\]

We can always go back:

\[
\begin{align*}
    F(w) & \quad \xrightarrow{\text{Inverse Fourier Transform}} \quad f(x) \\

    \text{Amplitude: } A = \pm \sqrt{R(\omega)^2 + I(\omega)^2} & \quad \text{Phase: } \phi = \tan^{-1} \frac{I(\omega)}{R(\omega)}
\end{align*}
\]

Slide credit: A. Efros
Review - Discrete Fourier transform

- Forward transform
  \[ F(u,v) = \frac{1}{MN} \sum_{x=0}^{M-1} \sum_{y=0}^{N-1} f(x,y) e^{-j2\pi(ux/M + vy/N)} \]
  for \( u = 0, 1, 2, \ldots, M - 1, \) \( v = 0, 1, 2, \ldots, N - 1 \)

- Inverse transform
  \[ f(x,y) = \sum_{u=0}^{M-1} \sum_{v=0}^{N-1} F(u,v) e^{j2\pi(ux/M + vy/N)} \]
  for \( x = 0, 1, 2, \ldots, M - 1, \) \( y = 0, 1, 2, \ldots, N - 1 \)

\( u, v \) : the transform or frequency variables
\( x, y \) : the spatial or image variables

Euler's definition of \( e^{j\theta} = \cos \theta + j\sin \theta \)

Review - The Fourier Transform

- Represent function on a new basis
  - Think of functions as vectors, with many components
  - We now apply a linear transformation to transform the basis
    - dot product with each basis element

- In the expression, \( u \) and \( v \) select the basis element, so a function of \( x \) and \( y \) becomes a function of \( u \) and \( v \)

- basis elements have the form \( e^{-j2\pi(ux + vy)} \)

Review - The Convolution Theorem

- The Fourier transform of the convolution of two functions is the product of their Fourier transforms
  \[ F[g * h] = F[g]F[h] \]

- The inverse Fourier transform of the product of two Fourier transforms is the convolution of the two inverse Fourier transforms
  \[ F^{-1}[gh] = F^{-1}[g] * F^{-1}[h] \]

- **Convolution** in spatial domain is equivalent to **multiplication** in frequency domain!
Review - Filtering in frequency domain

Today

- Sampling
- Gabor wavelets, Steerable filters

Why does a lower resolution image still make sense to us?
What do we lose?

Sampling

Image: http://www.flickr.com/photos/igorms/136916757/
**Sampled representations**

- How to store and compute with continuous functions?
- Common scheme for representation: samples
  - write down the function’s values at many points

**Reconstruction**

- Making samples back into a continuous function
  - for output (need realizable method)
  - for analysis or processing (need mathematical method)
  - amounts to “guessing” what the function did in between

**Sampling in digital audio**

- Recording: sound to analog to samples to disc
- Playback: disc to samples to analog to sound again
  - how can we be sure we are filling in the gaps correctly?

**Sampling Theorem**

Continuous signal: (Real world signal)

Shah function (Impulse train): (What the image measures)

Sampled function:

\[ f_s(x) = f(x) \delta(x - nx_0) = f(x) \sum_{n=-\infty}^{\infty} \delta(x - nx_0) \]
**Sampling Theorem**

Sampled function:

\[ f_s(x) = f(x) = f(x) \sum_{n=-\infty}^{\infty} \delta(x - nx_0) \]

\[ F_s(u) = F(u) \ast S(u) = F(u) \ast \frac{1}{x_0} \sum_{n=-\infty}^{\infty} \delta(u - \frac{n}{x_0}) \]

\[ u_{\text{max}} \]

Note that these are derived using angular frequency \( e^{-j\omega t} \).

Slide credit: S. Narasimhan
**Subsampling by a factor of 2**

Throw away every other row and column to create a 1/2 size image

---

**Undersampling**

- What if we “missed” things between the samples?
- Simple example: undersampling a sine wave
  - unsurprising result: information is lost
  - surprising result: indistinguishable from lower frequency
  - also was always indistinguishable from higher frequencies
  - **aliasing**: signals “traveling in disguise” as other frequencies

---

**Undersampling**

- What if we “missed” things between the samples?
- Simple example: undersampling a sine wave
  - unsurprising result: information is lost
  - surprising result: indistinguishable from lower frequency
  - also was always indistinguishable from higher frequencies
  - **aliasing**: signals “traveling in disguise” as other frequencies
Undersampling

- What if we “missed” things between the samples?
- Simple example: undersampling a sine wave
  - unsurprising result: information is lost
  - surprising result: indistinguishable from lower frequency
  - also was always indistinguishable from higher frequencies
  - aliasing: signals “traveling in disguise” as other frequencies

Slide credit: S. Marschner
Aliasing problem

- Sub-sampling may be dangerous....
- Characteristic errors may appear:
  - "Wagon wheels rolling the wrong way in movies"
  - "Checkerboards disintegrate in ray tracing"
  - "Striped shirts look funny on color television"

More examples

Check out Moire patterns on the web.

Aliasing in video

Imagine a spoked wheel moving to the right (rotating clockwise). Mark wheel with dot so we can see what’s happening.

If camera shutter is only open for a fraction of a frame time (frame time = 1/30 sec. for video, 1/24 sec. for film):

Without dot, wheel appears to be rotating slowly backwards! (counterclockwise)
**Sampling Theorem**

Sampled function:

\[ f_s(x) = f(x) = f(x) \sum_{n=-\infty}^{\infty} \delta(x-nx_0) \]

\[ F_3(u) = F(u) * S(u) = F(u) * \frac{1}{x_0} \sum_{n=-\infty}^{\infty} \delta\left( u - \frac{n}{x_0} \right) \]

\[ F(u) \quad u \quad F_3(u) \quad u \]

\[ u_{\text{max}} \quad \frac{A}{x_0} \quad u_{\text{max}} \]

- **Aliasing in graphics**
- **Sampling and aliasing**

**Nyquist Frequency**

If \( u_{\text{max}} > \frac{1}{2x_0} \)

When can we recover \( F(u) \) from \( F_3(u) \)?

Only if \( u_{\text{max}} \leq \frac{1}{2x_0} \) (Nyquist Frequency)

We can use

\[ C(u) = \begin{cases} x_0 & |u| < \frac{1}{2x_0} \\ 0 & \text{otherwise} \end{cases} \]

Then \( F(u) = F_3(u) C(u) \) and \( f(x) = \text{IFT}[F(u)] \)

Sampling frequency must be greater than \( 2u_{\text{max}} \)
Nyquist-Shannon Sampling Theorem

- When sampling a signal at discrete intervals, the sampling frequency must be $\geq 2 \times f_{\text{max}}$
- $f_{\text{max}}$ = max frequency of the input signal
- This will allow to reconstruct the original perfectly from the sampled version

Anti-aliasing

Solutions:
- Sample more often
- Get rid of all frequencies that are greater than half the new sampling frequency
  - Will lose information
  - But it’s better than aliasing
  - Apply a smoothing filter

Preventing aliasing

- Introduce lowpass filters:
  - remove high frequencies leaving only safe, low frequencies
  - choose lowest frequency in reconstruction (disambiguate)
Impulse Train

- Define a comb function (impulse train) in 1D as follows

\[ \text{comb}_M(x) = \sum_{k=-\infty}^{\infty} \delta[x - kM] \]

where \( M \) is an integer

**Impulse Train in 1D**

- \( \text{comb}_2(x) \)
  - \( \frac{1}{2} \text{comb}_{\frac{1}{2}}(u) \)

- Remember:
  - Scaling: \( f(ax) \)
    - \( \frac{1}{|a|} F\left(\frac{u}{a}\right) \)

**Sampling low frequency signal**

- Convolve:
  - \( f(x) \times \text{comb}_M(x) \)
  - \( F(u) \times \text{comb}_{\frac{1}{M}}(u) \)

- Multiply:
  - \( f(x) \cdot \text{comb}_M(x) \)
  - \( F(u) \cdot \text{comb}_{\frac{1}{M}}(u) \)

**Impulse Train in 2D (bed of nails)**

\[ \text{comb}_{M,N}(x,y) = \sum_{k=-\infty}^{\infty} \sum_{l=-\infty}^{\infty} \delta[x - kM, y - lN] \]

- Fourier Transform of an impulse train is also an impulse train:

\[ \sum_{k=-\infty}^{\infty} \sum_{l=-\infty}^{\infty} \delta(x - kM, y - lN) \Leftrightarrow \frac{1}{MN} \sum_{k=-\infty}^{\infty} \sum_{l=-\infty}^{\infty} \delta\left(u - \frac{k}{M}, v - \frac{l}{N}\right) \]

\[ \text{comb}_{M,N}(x,y) \Leftrightarrow \text{comb}_{\frac{1}{M}, \frac{1}{N}}(u,v) \]

As the comb samples get further apart, the spectrum samples get closer together!
Sampling low frequency signal

If there is no overlap, the original signal can be recovered from its samples by low-pass filtering.

Overlap: The high frequency energy is folded over into low frequency. It is “aliasing” as lower frequency energy. And you cannot fix it once it has happened.
**Sampling high frequency signal**

- Without anti-aliasing filter:
  \[ f(x) \text{comb}_m(x) \]
  ![Diagram](image1)

- With anti-aliasing filter:
  \[ [f(x) \ast h(x)] \text{comb}_m(x) \]
  ![Diagram](image2)

---

**Algorithm for downsampling by factor of 2**

1. Start with image(h, w)
2. Apply low-pass filter
   \[ \text{im\_blur} = \text{imfilter}(	ext{image}, \text{fspecial('gaussian', 7, 1))} \]
3. Sample every other pixel
   \[ \text{im\_small} = \text{im\_blur}(1:2:end, 1:2:end); \]

---

**Anti-aliasing**

256x256  128x128  64x64  32x32  16x16

![Images](image3)

**Subsampling without pre-filtering**

1/2  1/4 (2x zoom)  1/8 (4x zoom)

![Images](image4)
**Subsampling with Gaussian pre-filtering**

Gaussian 1/2  
G 1/4  
G 1/8

1000 pixel width

**Up-sampling**

How do we compute the values of pixels at fractional positions?

by dropping pixels  
gaussian filter

250 pixel width
**Up-sampling**

How do we compute the values of pixels at fractional positions?

Bilinear sampling:

\[ f(x + a, y + b) = 
(1 - a)(1 - b) f(x, y) + 
a(1 - b) f(x + 1, y) + 
(1 - a)b f(x, y + 1) + 
ab f(x + 1, y + 1) \]

Bicubic sampling fits a higher order function using a larger area of support.

---

**Up-sampling Methods**

Nearest neighbor  Bilinear  Bicubic

---

**Up-sampling**

Nearest neighbor  Bilinear  Bicubic

---
Today

- Sampling
- Gabor wavelets, Steerable filters

Fourier Filtering

Multiply by a filter in the frequency domain => convolve with the filter in spatial domain.

Phase Carries More Information

Raw Images:

Magnitude and Phase:

Reconstruct (inverse FFT) mixing the magnitude and phase images

Phase “Wins”

What is a good representation for image analysis?

- Fourier transform domain tells you “what” (textural properties), but not “where”.
- Pixel domain representation tells you “where” (pixel location), but not “what”.
- Want an image representation that gives you a local description of image events—what is happening where.
Analyzing local image structures

We can look at the real and imaginary parts:

\[ u(x, y) = e^{\frac{x^2 + y^2}{2\sigma^2}} \cos(2\pi u_0 x) \]

\[ v(x, y) = e^{\frac{x^2 + y^2}{2\sigma^2}} \sin(2\pi u_0 x) \]

Gabor wavelets

\[ \psi_r(x, y) = e^{\frac{x^2 + y^2}{2\sigma^2}} \cos(2\pi u_0 x) \]

\[ \psi_i(x, y) = e^{\frac{x^2 + y^2}{2\sigma^2}} \sin(2\pi u_0 x) \]
**Gabor filters**

Gabor filters at different scales and spatial frequencies

Top row shows anti-symmetric (or odd) filters; these are good for detecting odd-phase structures like edges. Bottom row shows the symmetric (or even) filters, good for detecting line phase contours.

Slide credit: B. Freeman and A. Torralba

---

**Quadrature filter pairs**

- A quadrature filter is a complex filter whose real part is related to its imaginary part via a Hilbert transform along a particular axis through the origin.

\[
Gabor \ wavelet: \quad \psi(x, y) = e^{-\frac{x^2 + y^2}{2\sigma^2}} e^{j2\pi f_0 x}
\]

Slide credit: B. Freeman & A. Torralba
Quadrature filter pairs

Contrast invariance! (same energy response for white dot on black background as for a black dot on a white background).

How quadrature pair filters work

Figure 3.5: Frequency content of two bandpass filters in quadrature: (a) even phase filter, called $G$ in text, and (b) odd phase filter, $H$. Plus and minus signs illustrate relative sign of regions in the frequency domain. See Fig. 3.6 for calculation of the frequency content of the energy measure derived from these two filters.
Oriented Filters

- Gabor wavelet: 
  \[ y(x,y) = e^{-\frac{x^2 + y^2}{2\sigma^2}} e^{i2\omega_0 x} \]

- Tuning filter orientation: 
  \[
  x' = \cos(\alpha)x + \sin(\alpha)y \\
  y' = -\sin(\alpha)x + \cos(\alpha)y
  \]

Steerable filters

Derivatives of a Gaussian:

\[
  h_x(x,y) = \frac{\partial h(x,y)}{\partial x} = -\frac{x}{2\sigma^2} e^{-\frac{x^2 + y^2}{2\sigma^2}} \\
  h_y(x,y) = \frac{\partial h(x,y)}{\partial y} = -\frac{y}{2\sigma^2} e^{-\frac{x^2 + y^2}{2\sigma^2}}
\]

An arbitrary orientation can be computed as a linear combination of those two basis functions:

\[
  h_x(x,y) = \cos(\alpha) h_x(x,y) + \sin(\alpha) h_y(x,y)
\]

The representation is "shiftable" on orientation: We can interpolate any other orientation from a finite set of basis functions.
Steerable filters

Fig. 3. Steerable filter system block diagram. A bank of dedicated filters process the image. Their outputs are multiplied by a set of gain maps that adaptively control the orientation of the synthesized filter.

Gabor Filter Bank

or = [4 4 4 4];

or = [12 6 3 2];

Not for image reconstruction. It does NOT cover the entire space!

Local image representations

A pixel = [r,g,b]

An image patch

Gabor filter pair in quadrature

Gabor jet

V1 sketch: hypercolumns


Summary

• Sampling
• Gabor wavelets, Steerable filters
Next lecture

• Image pyramids