BBM 413
Fundamentals of Image Processing

Erkut Erdem
Dept. of Computer Engineering
Hacettepe University

Frequency Domain Techniques – Part I

Review - Point Operations

- Smallest possible neighborhood is of size 1x1
- Process each point independently of the others
- Output image \( g \) depends only on the value of \( f \) at a single point \((x,y)\)
- Transformation function \( T \) remaps the sample's value:
  \[ s = T(r) \]
  where
  - \( r \) is the value at the point in question
  - \( s \) is the new value in the processed result
  - \( T \) is an intensity transformation function

Review - Spatial Filtering

\[ f[\cdot,\cdot] \]
\[ g[\cdot,\cdot] \]

\[ h[m,n] = \sum_{k,l} g[k,l] f[m+k,n+l] \]

\[ h[m,n] = \sum_{k,l} g[k,l] f[m+k,n+l] \]

Slide credit: S. Seitz
\[ h[m,n] = \sum_{k,l} g[k,l] f[m+k,n+l] \]
Today

- Frequency domain techniques
- Images in terms of frequency
- Fourier Series
- Convolution Theorem

Why does the Gaussian give a nice smooth image, but the square filter give edgy artifacts?

Fill in the blanks:

a) \( \_ = D \ast B \)
b) \( A = \_ \ast \_ \)
c) \( F = D \ast \_ \)
d) \( \_ = D \ast D \)

Slide credit: J. Hays

Slide credit: D. Hoiem
Why does a lower resolution image still make sense to us? What do we lose?

How is it that a 4MP image can be compressed to a few hundred KB without a noticeable change?

Answer to these questions?

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

Jean Baptiste Joseph Fourier (1768-1830)

had crazy idea (1807):

Any univariate function can be rewritten as a weighted sum of sines and cosines of different frequencies.
Jean Baptiste Joseph Fourier (1768-1830)

had crazy idea (1807):
Any univariate function can be rewritten as a weighted sum of sines and cosines of different frequencies.

• Don’t believe it?
  – Neither did Lagrange, Laplace, Poisson and other big wigs
  – Not translated into English until 1878!

• But it’s (mostly) true!
  – called Fourier Series
  – there are some subtle restrictions

A sum of sines

Our building block:
\[ A \sin(\omega x + \phi) \]

Add enough of them to get any signal \( f(x) \) you want!

Frequency Spectra

• example: \( g(t) = \sin(2\pi f t) + (1/3)\sin(2\pi(3f) t) \)
Frequency Spectra

Slide credit: A. Efros
Frequency Spectra

$$\sum_{k=1}^{\infty} \frac{1}{k} \sin(2\pi kt)$$

Image credit: Lucas V. Barbosa
**Example: Music**

We think of music in terms of frequencies at different magnitudes.

**Other signals**

We can also think of all kinds of other signals the same way.

---

**Fourier Transform**

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

\[
 f(x) \longrightarrow \text{Fourier Transform} \longrightarrow F(w)
\]

For every \( w \) from 0 to inf, \( F(w) \) holds the amplitude \( A \) and phase \( f \) of the corresponding sine

\[
 A \sin(\omega x + \phi)
\]

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

\[
 F(\omega) = R(\omega) + iI(\omega)
\]

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

We can always go back:

\[
 F(w) \longrightarrow \text{Inverse Fourier Transform} \longrightarrow f(x)
\]

---

**Fourier Transform**

- Fourier transform stores the magnitude and phase at each frequency
  - Magnitude encodes how much signal there is at a particular frequency
  - Phase encodes spatial information (indirectly)
  - For mathematical convenience, this is often notated in terms of real and complex numbers

Amplitude: \( A = \pm \sqrt{R(\omega)^2 + I(\omega)^2} \)

Phase: \( \phi = \tan^{-1} \frac{I(\omega)}{R(\omega)} \)
**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

**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 \( e^{-j2\pi(ux+vy)} \)

---

**How to interpret 2D Fourier Spectrum**

- **Vector** \( (u,v) \)
  - **Magnitude** gives frequency
  - **Direction** gives orientation.

- **Fourier basis element**
  \( e^{-j2\pi(ux+vy)} \)

- **example, real part**
  \( F^a(x,y) \)

- **Log power spectrum**
  \( F^a(x,y) = \text{const. for } (ux+vy) = \text{const.} \)

- **Vector** \( (u,v) \)
  - **Magnitude** gives frequency
  - **Direction** gives orientation.
Here \( u \) and \( v \) are larger than in the previous slide.

\[
e^{-\pi \left( \omega x + \omega y \right)}
\]

\[
e^{-\pi \left( \omega x + \omega y \right)}
\]

Slide credit: S. Thrun

And larger still...

\[
e^{-\pi \left( \omega x + \omega y \right)}
\]

\[
e^{-\pi \left( \omega x + \omega y \right)}
\]

Slide credit: S. Thrun

2D FFT

Sinusoid with frequency = 1 and its FFT

Slide credit: M. H. Yang

2D FFT

Sinusoid with frequency = 3 and its FFT

Slide credit: M. H. Yang
2D FFT

Sinusoid with frequency = 5 and its FFT
Slide credit: M. H. Yang

Sinusoid with frequency = 10 and its FFT
Slide credit: M. H. Yang

Sinusoid with frequency = 15 and its FFT
Slide credit: M. H. Yang

Sinusoid with varying frequency and their FFT
Slide credit: M. H. Yang
Rotation

Sinusoid rotated at 30 degrees and its FFT

Slide credit: M. H. Yang

2D FFT

Sinusoid rotated at 60 degrees and its FFT

Slide credit: M. H. Yang

2D FFT

$$F(u, v) = \frac{1}{MN} \sum_{x=0}^{M-1} \sum_{y=0}^{N-1} f(x, y) e^{-j2\pi(ux/M + vy/N)}$$

Convolution masks for different frequencies

Slide credit: M. H. Yang

Fourier analysis in images

Intensity Image

Fourier Image

More: http://www.cs.unm.edu/~brayer/vision/fourier.html

http://sharp.bu.edu/~slehar/fourier/fourier.html#filtering

Slide credit: A. Efros
Signals can be composed

$$\begin{align*}
\text{Image 1} & + \text{Image 2} = \text{Result Image} \\
\text{Magnitude FT Image 1} & + \text{Magnitude FT Image 2} = \text{Result Magnitude FT} \\
\text{Log(1+Magnitude FT Result Image)} & = \text{Result Log(1+Magnitude FT)}
\end{align*}$$

Some important Fourier Transforms

The Fourier Transform of some well-known images
**Fourier Amplitude Spectrum**

- Fourier Amplitude Spectrum images:
  - A
  - B
  - C

- Fourier transform magnitude images:
  - What in the image causes the dots?

**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!
Properties of Fourier Transforms

• Linearity \( \mathcal{F}[ax(t) + by(t)] = a\mathcal{F}[x(t)] + b\mathcal{F}[y(t)] \)

• Fourier transform of a real signal is symmetric about the origin

• The energy of the signal is the same as the energy of its Fourier transform

Filtering in spatial domain

\[
\begin{array}{c|c|c}
1 & 0 & -1 \\
2 & 0 & -2 \\
1 & 0 & -1 \\
\end{array}
\]

Filtering in frequency domain

\[
\begin{align*}
\mathcal{F}(s_x, s_y) & = |F(s_x, s_y)| \\
\mathcal{H}(s_x, s_y) & = |H(s_x, s_y)| \\
\mathcal{G}(s_x, s_y) & = |G(s_x, s_y)|
\end{align*}
\]

2D convolution theorem example

Slide credits: J. Hays, D. Hoiem, A. Efros
**Filtering**

Why does the Gaussian give a nice smooth image, but the square filter give edgy artifacts?

---

**Filtering**

Box Filter

---

**Fourier Transform pairs**

Spatial domain

\[ F(s) = \int_{-\infty}^{\infty} f(x)e^{-2\pi i sx}dx \]

Frequency domain

+box(x)

\[ \text{gauss}(x; \sigma) \]

\[ \text{sinc}(s) \]

Slide credit: A. Efros
Low-pass, Band-pass, High-pass filters

low-pass:

High-pass / band-pass:

FFT in Matlab

- Filtering with fft
  - Filtering with `fft`
    ```matlab
    im = ... % "im" should be a gray-scale floating point image
    [imh, imw] = size(im);
    fftsize = 1024; % should be order of 2 (for speed) and include padding
    im_fft = fft2(im, fftsize, fftsize); % 1) fft im with padding
    hs = 50; % filter half-size
    fil = fspecial('gaussian', hs*2+1, 10); % 2) fft fil, pad to same size as image
    im_fil_fft = im_fft .* fil_fft; % 3) multiply fft images
    im_fil = ifft2(im_fil_fft); % 4) inverse fft2
    im_fil = im_fil(1+hs:size(im,1)+hs, 1+hs:size(im, 2)+hs); % 5) remove padding
    ```

- Displaying with fft
  ```matlab
  figure(1), imagesc(log(abs(fftshift(im_fft)))), axis image, colormap jet
  ```

Edges in images

Phase and Magnitude

- Curious fact
  - all natural images have about the same magnitude transform
  - hence, phase seems to matter, but magnitude largely doesn’t

- Demonstration
  - Take two pictures, swap the phase transforms, compute the inverse - what does the result look like?
This is the magnitude transform of the cheetah picture.

This is the magnitude transform of the zebra picture.
Clues from Human Perception

- Early processing in humans filters for various orientations and scales of frequency
- Perceptual cues in the mid-high frequencies dominate perception
- When we see an image from far away, we are effectively subsampling it

Campbell-Robson contrast sensitivity curve

The higher the frequency the less sensitive human visual system is...
Lossy Image Compression (JPEG)

\[ X_{k_1,k_2} = \sum_{n_1=0}^{N_1-1} \sum_{n_2=0}^{N_2-1} x_{n_1,n_2} \cos \left( \frac{\pi}{N_1} (n_1 + \frac{1}{2}) k_1 \right) \cos \left( \frac{\pi}{N_2} (n_2 + \frac{1}{2}) k_2 \right). \]

Using DCT in JPEG

- The first coefficient \( B(0,0) \) is the DC component, the average intensity
- The top-left coeffs represent low frequencies, the bottom right – high frequencies

Image compression using DCT

- DCT enables image compression by concentrating most image information in the low frequencies
- Loose unimportant image info (high frequencies) by cutting \( B(u,v) \) at bottom right
- The decoder computes the inverse DCT – IDCT

JPEG compression comparison

89k  12k
Things to Remember

• Sometimes it makes sense to think of images and filtering in the frequency domain
  – Fourier analysis

• Can be faster to filter using FFT for large images (N logN vs. N² for auto-correlation)

• Images are mostly smooth
  – Basis for compression

Practice question

1. Match the spatial domain image to the Fourier magnitude image

Summary

• Frequency domain techniques
• Images in terms of frequency
• Fourier Series
• Convolution Theorem

Next Week

• Sampling
• Gabor wavelets
• Steerable filters