2D Fourier, Scale, and Cross-correlation

CS 510
Lecture #12
February 26th, 2014
Where are we?

• We can detect objects, but they can only differ in translation and 2D rotation
• Then we introduced Fourier analysis.
• Why?
  – Because Fourier analysis can help us with scale
  – Because Fourier analysis can make correlation faster
Review: Discrete Fourier Transform

- Problem: an image is not an analogue signal that we can integrate.
- Therefore for $0 \leq x < N$ and $0 \leq u < N/2$:

$$F(u) = \sum_{x=0}^{N-1} f(x) \left[ \cos \left( \frac{2\pi ux}{N} \right) - i \sin \left( \frac{2\pi ux}{N} \right) \right]$$

And the discrete inverse transform is:

$$f(x) = \frac{1}{N} \sum_{u=0}^{N-1} F(u) \left[ \cos \left( \frac{2\pi ux}{N} \right) + i \sin \left( \frac{2\pi ux}{N} \right) \right]$$
2D Fourier Transform

• So far, we have looked only at 1D signals
• For 2D signals, the continuous generalization is:
  \[ F(u, v) = \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} f(x, y) \left[ \cos(2\pi(ux + vy)) - i \sin(2\pi(ux + vy)) \right] \]

• Note that frequencies are now two-dimensional
  – \( u \) = freq in \( x \), \( v \) = freq in \( y \)
• Every frequency \((u,v)\) has a real and an imaginary component.
2D sine waves

• This looks like you’d expect in 2D

➤ Note that the frequencies don’t have to be equal in the two dimensions.
2D Discrete Fourier Transform

\[ F(u,v) = \sum_{x=-N/2}^{N/2} \sum_{y=-N/2}^{N/2} f(x,y) \left[ \cos\left(\frac{2\pi}{N}(ux + vy)\right) - i \sin\left(\frac{2\pi}{N}(ux + vy)\right) \right] \]

- What happened to the bounds on x & y?
- How big is the discrete 2D frequency space representation?
2D Frequency Space

• Remember that:
  – Cosine is an even function: \( \cos(x) = \cos(-x) \)
  – Sine is an odd function: \( \sin(x) = -\sin(-x) \)

• So
  – \( F(u,v) = a+ib \Rightarrow F(-u, -v) = a-ib \)

• And
  – \( F(-u,v) = a+ib \Rightarrow F(u, -v) = a-ib \)

• But
  – \( F(u,v) = a+ib \Rightarrow F(-u, v) = ??? \)
Size of 2D Frequency representation:

- One dimension must vary from \(-N/2\) to \(N/2\), while the other varies from 0 to \(N/2\)
  - Doesn’t matter which is which
- \(N \times (N/2) \times 2\) values per frequency = \(N^2\)
- Same as the source spatial representation
Showing Frequency Space

• To display a frequency space:
  – We plot it from \(-N/2\) to \(N/2\) in both dimensions
  – The result is symmetric about the origin (and therefore redundant)
  – We can’t plot a complex number, so we show the magnitude at every pixel \(\sqrt{a^2 + b^2}\)
    • Thus discarding the phase information
    • Phase plots are also possible \((\tan^{-1}(b/a))\)
Showing Frequency Space

http://www.brainflux.org/java/classes/FFT2DApplet.html
But Why?

• Reason 1: Fast Correlation

• Reason 2: Scale
“Slide” a mask over an image. At each window position, multiply the mask values by the image value under them. Sum the results for every pixel.

Think of this as a sliding dot product
Convolution (cont.)

• Why return to convolution after introducing the Fourier Transform?

• Because multiplying two signals in the frequency domain is the same as convolving them in the spatial domain! (trust me)
Computing Cross-Correlation

- In cross-correlation, the mask is convolved with the target image
  - zero-mean & unit length the mask
  - zero-mean & unit length the image
  - Convolve the image and mask
Fast correlation

• If we compute correlation in the spatial domain, the cost is $O(nm)$, where $n > m$.

• What if we use the frequency domain?
  – Convolution becomes point-wise multiplication
  – Convert to frequencies: $O(n \log n)$
  – Point-wise multiply: $O(n)$
  – Convert back to spatial: $O(n \log n)$

• Frequency domain is faster if $\log(n) < m$
Fast correlation (II)

• Is spatial convolution really the same as frequency point-wise multiplication?
• Yes, but…
  – Take the complex conjugate of the mask
  – Images must be the same size
    • Pad mask with zeroes
    • Doesn’t change the overall complexity
  – What happens at the image edges?
    • Frequency domain repeats
    • Values off the source image aren’t zero
    • Equivalent to convolution on a torus
Fourier Correlation

- Simple convolution, not Pearson’s correlation
  - The template can be zero mean & unit length
  - But the image windows won’t be
- No 2D rotation
- But fast! $O(n \log(n))$
**Using Fourier Correlation**

- Generate multiple templates at different rotations
- Pad to image size
- Multiply with target in frequency domain
- Find peak in spatial domain
  - Not true correlation
  - Only rough rotation
  - But fast
- Perform true rotation & correlation at peaks
But Why?

- Reason 1: Fast Correlation
- Reason 2: Scale
Reminder…

$$g(x) = a_1 \cos(f_1 x) + b_1 \sin(f_1 x)$$

$$+ a_2 \cos(f_2 x) + b_2 \sin(f_2 x)$$

$$+ a_3 \cos(f_3 x) + b_3 \sin(f_3 x)$$

$$+ \cdots$$

- Signal is reconstructed as a series of sine and cosine waves
Review: Fourier Magnitude & Phase

- The energy at a frequency is:

\[ |F(u)| = \sqrt{R^2(u) + I^2(u)} \]

- The phase at a frequency is:

\[ \tan^{-1}(u) = \frac{I(u)}{R(u)} \]
The Nyquist Rate

• What if the frequency is above N/2?
  – You have fewer than one sample per half-cycle
  – High frequencies look like lower frequencies
Example by Brent Locher - www.fourier-series.com
Low-Pass Filtering 101

• Drop high frequency Fourier coefficients.

To low-pass filter an image:
1) convert to frequency domain
2) discard all values for $u > \text{thresh}$
3) Convert back to spatial domain

**Brainflux Fourier Applet**

http://www.brainflux.org/java/classes/FFT2DApplet.html

But is there an easier way?
A more efficient way?

- Alternatively, convolve with a Gaussian.
Photoshop Gaussian Blur
Low-Pass Filter

- Low-Pass filter - multiply by a pulse in frequency space, or
- Convolve the image with the inverse Fourier transform of a pulse...

Sinc filter

Truncated sinc
The Gibbs Phenomenon  (ringing)

• The truncated sinc is no longer a pulse in frequency space
  – passes small amounts of some high frequencies
  – passes acceptable frequencies in uneven amounts
  – may create negative values in unusual circumstances
Alternative Filters

- Pulse/sinc
- Triangle/sinc²
- Gaussian/Gaussian

Image Reductions

- Anytime the target image has a lower resolution than the source image, prevent frequency aliasing by low-pass filtering.
  - In practice, convolve with a Gaussian
  - Determine Nyquist rate for target image
    - $\frac{1}{2}$ width and $\frac{1}{2}$ height
  - Select $\sigma$
  - Convolve source image with $g(\sigma)$
  - Apply geometric transformation to result
Image Reductions (II)

• Example: reduce 1Kx1K to 800x800 pixels
  – Select one (source) pixel as unit length
  – The Nyquist rate for source is 0.5 cycles/s_pixel
  – Nyquist rate for target is 0.4 cycles/s_pixel

• Problem: Gaussian is not a strict cut-off
  – Select “pass” value (2\(\sigma\) sounds good)
  – Select mask width to cover “most” of the area under the Gaussian curve
    • recommend 5\(\sigma\) (source: Trucco & Verri)
    • Covers 98.75% of the area under the Gaussian
Image Reduction (III)

- So $2\sigma$ is 0.4 cycles/pixel
  - The Fourier transform of $g(x, \sigma)$ is $g(\omega, 1/\sigma)$
  - The inverse of 0.4 cycles/pixel is 2.5 pixels/cycle
    - $2\sigma = 2.5$ pixels/cycle
    - $\sigma = 1.25$ pixels/cycle
  - (T&V): To include $5\sigma$ of the curve, $\sigma = w/5$,
    - $w$ is the width of the mask
    - $W = 6.25$

- Create a 7x7 Gaussian mask with sigma 1.25
  - $w$ should be odd, so don’t use 6x6
    - Why make $w$ odd? To avoid a geometric transformation…

- Smooth the image using this mask, then subsample.
**Image Transformation**

- What if we want to keep 1Kx1K size?
  - Target Nyquist rate is 0.5 cycles/pixel
  - In image space, $2\sigma = 2$ pixels/cycle, so $\sigma=1$
  - $\sigma = w/5$, so $w = 5$
  - Create a 5x5 mask with $\sigma=1$, smooth source image
  - Transform (rotate, etc.) the result.

- This is why most image processing packages includes predefined 5x5 Gaussian masks

- Other masks you build yourself.
Smoothing with $\sigma=1$

Original Image

Image with Gaussian Smoothing, $\sigma = 1.0$
Limits to Gaussians

• The Gaussian mask itself is a discrete sampling of a continuous signal.
• Gaussian signals with sigmas below 0.8 are too small to be sampled at pixel intervals.
• Generally not used for “up-sampling”
Implications of Smoothing

• All of this is based on the view that an image is a sum of sine waves.
• Physically, this assumption is absurd
  – Think of a ray tracer -- where would sine waves (or repeating signals) come from?
  – Occlusion edges lead to non-differentiable jumps
    • the signal content on the two sides are unrelated
    • violates the differentiability assumption underlying Fourier analysis
  – Edges are therefore very high frequency;
    • $G(x, \sigma=1)$ blurs the image
• Fourier analysis does describe the limitations of A/D conversion, and therefore of image manipulation