The Frequency Domain

Somewhere in Cinque Terre, May 2005

Many slides borrowed from Steve Seitz

15-463: Computational Photography
Alexei Efros, CMU, Fall 2007
Salvador Dali

“Gala Contemplating the Mediterranean Sea, which at 30 meters becomes the portrait of Abraham Lincoln”, 1976
A nice set of basis

Teases away fast vs. slow changes in the image.

This change of basis has a special name…
Jean Baptiste Joseph Fourier (1768-1830)

had crazy idea (1807):

*Any periodic 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 true!

- called Fourier Series
A sum of sines

Our building block:

$$A \sin(\omega x + \phi)$$

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

How many degrees of freedom?

What does each control?

Which one encodes the coarse vs. fine structure of the signal?

$$f(\text{target}) = f_1 + f_2 + f_3 + ... + f_n + ...$$
Fourier Transform

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

$$f(x) \xrightarrow{\text{Fourier Transform}} F(\omega)$$

For every $\omega$ from 0 to inf, $F(\omega)$ holds the amplitude $A$ and phase $\phi$ of the corresponding sine $A \sin(\omega x + \phi)$

- How can $F$ hold both? Complex number trick!

$$F(\omega) = R(\omega) + iI(\omega)$$

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

$$\phi = \tan^{-1} \frac{I(\omega)}{R(\omega)}$$

We can always go back:

$$F(\omega) \xrightarrow{\text{Inverse Fourier Transform}} f(x)$$
Time and Frequency

example: \( g(t) = \sin(2\pi f t) + \frac{1}{3}\sin(2\pi(3f) t) \)
Time and Frequency

example: \( g(t) = \sin(2\pi f t) + \left( \frac{1}{3} \right) \sin(2\pi (3f) t) \)
Frequency Spectra

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

Usually, frequency is more interesting than the phase
Frequency Spectra
Frequency Spectra
Frequency Spectra
Frequency Spectra
Frequency Spectra
Frequency Spectra

\[ A \sum_{k=1}^{\infty} \frac{1}{k} \sin(2\pi kt) \]
Frequency Spectra

(a) Graph showing $f(x)$.

(b) Graph showing $|F(u)|$.

(c) Graph showing $f(x)$ and $|F(u)|$ for a different function.
Extension to 2D

in Matlab, check out: imagesc(log(abs(fftshift(fft2(im)))));
Man-made Scene
Can change spectrum, then reconstruct
Low and High Pass filtering
The Convolution Theorem

The greatest thing since sliced (banana) bread!

- 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!
Fourier Transform pairs

Spatial domain

\[ f(x) \]

\[ \text{box}(x) \]

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

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

Frequency domain

\[ F(s) = \int_{-\infty}^{\infty} f(x)e^{-i2\pi sx}dx \]

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

\[ \text{gauss}(s; 1/\sigma) \]

\[ \text{box}(x) \]
2D convolution theorem example

$f(x,y)$

$h(x,y)$

$g(x,y)$

$|F(s_x,s_y)|$

$|H(s_x,s_y)|$

$|G(s_x,s_y)|$
Low-pass, Band-pass, High-pass filters

low-pass:

High-pass / band-pass:
Edges in images
What does blurring take away?
What does blurring take away?

smoothed (5x5 Gaussian)
High-Pass filter

smoothed – original
Band-pass filtering

Gaussian Pyramid (low-pass images)
Laplacian Pyramid

How can we reconstruct (collapse) this pyramid into the original image?
Why Laplacian?

Gaussian

delta function

Laplacian of Gaussian
Unsharp Masking

\[ \text{Original image} - \alpha \text{High-pass filtered image} = \text{Unsharp image} \]

\[ \text{Original image} + \alpha \text{High-pass filtered image} = \text{Enhanced image} \]
Image gradient

The gradient of an image:

\[ \nabla f = \left[ \frac{\partial f}{\partial x}, \frac{\partial f}{\partial y} \right] \]

The gradient points in the direction of most rapid change in intensity

\[ \nabla f = \left[ \frac{\partial f}{\partial x}, 0 \right] \]

\[ \nabla f = \left[ 0, \frac{\partial f}{\partial y} \right] \]

The gradient direction is given by:

\[ \theta = \tan^{-1} \left( \frac{\partial f}{\partial y} / \frac{\partial f}{\partial x} \right) \]

- how does this relate to the direction of the edge?

The edge strength is given by the gradient magnitude

\[ \| \nabla f \| = \sqrt{\left( \frac{\partial f}{\partial x} \right)^2 + \left( \frac{\partial f}{\partial y} \right)^2} \]
Effects of noise

Consider a single row or column of the image

- Plotting intensity as a function of position gives a signal

How to compute a derivative?

Where is the edge?
Solution: smooth first

Where is the edge? Look for peaks in $\frac{\partial}{\partial x}(h \ast f)$.
Derivative theorem of convolution

\[ \frac{\partial}{\partial x} (h \ast f) = \left( \frac{\partial}{\partial x} h \right) \ast f \]

This saves us one operation:

\( f \)

\( \frac{\partial}{\partial x} h \)

\( \left( \frac{\partial}{\partial x} h \right) \ast f \)
Laplacian of Gaussian

Consider \( \frac{\partial^2}{\partial x^2} (h \ast f) \)

Where is the edge? Zero-crossings of bottom graph
2D edge detection filters

\[ h_\sigma(u, v) = \frac{1}{2\pi\sigma^2} e^{-\frac{u^2+v^2}{2\sigma^2}} \]

\[ \frac{\partial}{\partial x} h_\sigma(u, v) \]

\[ \nabla^2 h_\sigma(u, v) \]

\[ \nabla^2 \text{ is the Laplacian operator}: \]

\[ \nabla^2 f = \frac{\partial^2 f}{\partial x^2} + \frac{\partial^2 f}{\partial y^2} \]
MATLAB demo

g = fspecial('gaussian',15,2);
imagesc(g)
surfl(g)
gclown = conv2(clown,g,'same');
imagesc(conv2(clown,[−1 1],'same'));
imagesc(conv2(gclown,[−1 1],'same'));
dx = conv2(g,[−1 1],'same');
imagesc(conv2(clown,dx,'same'));
lg = fspecial('log',15,2);
lclown = conv2(clown,lg,'same');
imagesc(lclown)
imagesc(clown + .2*lclown)
Campbell-Robson contrast sensitivity curve
 Depends on Color

R G B
Lossy Image Compression (JPEG)

Block-based Discrete Cosine Transform (DCT)
Using DCT in JPEG

A variant of discrete Fourier transform

• Real numbers
• Fast implementation

Block size

• small block
  – faster
  – correlation exists between neighboring pixels
• large block
  – better compression in smooth regions
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.

- Quantization Table

<p>| | | | | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>3</td>
<td>5</td>
<td>7</td>
<td>9</td>
<td>11</td>
<td>13</td>
<td>15</td>
<td>17</td>
</tr>
<tr>
<td>5</td>
<td>7</td>
<td>9</td>
<td>11</td>
<td>13</td>
<td>15</td>
<td>17</td>
<td>19</td>
</tr>
<tr>
<td>7</td>
<td>9</td>
<td>11</td>
<td>13</td>
<td>15</td>
<td>17</td>
<td>19</td>
<td>21</td>
</tr>
<tr>
<td>9</td>
<td>11</td>
<td>13</td>
<td>15</td>
<td>17</td>
<td>19</td>
<td>21</td>
<td>23</td>
</tr>
<tr>
<td>11</td>
<td>13</td>
<td>15</td>
<td>17</td>
<td>19</td>
<td>21</td>
<td>23</td>
<td>25</td>
</tr>
<tr>
<td>13</td>
<td>15</td>
<td>17</td>
<td>19</td>
<td>21</td>
<td>23</td>
<td>25</td>
<td>27</td>
</tr>
<tr>
<td>15</td>
<td>17</td>
<td>19</td>
<td>21</td>
<td>23</td>
<td>25</td>
<td>27</td>
<td>29</td>
</tr>
<tr>
<td>17</td>
<td>19</td>
<td>21</td>
<td>23</td>
<td>25</td>
<td>27</td>
<td>29</td>
<td>31</td>
</tr>
</tbody>
</table>
JPEG compression comparison

89k

12k