**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 a *intensity transformation* function

**Review – Spatial Filtering**

\[
\begin{align*}
  f[.,.] & \quad h[.,.] \\
  g[.,.] & = \sum_{k,l} g[k,l] f[m+k,n+l]
\end{align*}
\]

Slide credit: S. Seitz
$h[m,n] = \sum_{k,l} g[k,l] \cdot f[m+k,n+l]$
**Review – Spatial Filtering**

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

- **Gaussian**
- **Box filter**

Slide credit: D. Hoiem

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


Slide credit: D. Hoiem

**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

Slide credit: A. Efros
**A sum of sines**

Our building block:

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

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

**Fourier Transform**

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

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

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!

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

We can always go back:

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

**Frequency Spectra**

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

**Frequency Spectra**
Frequency Spectra

Slide credit: A. Efros

Frequency Spectra

Slide credit: A. Efros
**Example: Music**

- We think of music in terms of frequencies at different magnitudes.
The Discrete Fourier transform

- Forward transform
  \[ F[m,n] = \sum_{k=0}^{M-1} \sum_{l=0}^{N-1} f[k,l] e^{-\frac{2\pi i (km + ln)}{MN}} \]

- Inverse transform
  \[ f[k,l] = \frac{1}{MN} \sum_{m=0}^{M-1} \sum_{n=0}^{N-1} F[m,n] e^{\frac{2\pi i (km + ln)}{MN}} \]

Some important Fourier Transforms

- Image
- Magnitude FT

How to interpret 2D Fourier Spectrum

- Log power spectrum
- Euler’s definition of \( e^{i\theta} \)
- How to interpret 2D Fourier Spectrum
- Image
- Magnitude FT
The Fourier Transform of some well-known images

Image

Log(1+Magnitude FT)

Fourier Amplitude Spectrum

A

B

C

Fourier Amplitude Spectrum

1

2

3

Masking out the fundamental and harmonics from periodic pillars

What in the image causes the dots?
Signals can be composed

\[ \begin{array}{c}
\text{Image 1} \\
\text{Image 2} \\
\end{array} \]

\[ + \]

\[ = \]

\[ \text{Image 3} \]

http://sharp.bu.edu/~slehar/fourier/fourier.html#filtering
More: http://www.cs.unm.edu/~brayer/vision/fourier.html

Slide credit: A. Efros

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!

Filtering in spatial domain

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

Filtering in frequency domain

Slide credit: D. Hoiem
2D convolution theorem example

\[ f(x, y) \]

\[ h(x, y) \]

\[ g(x, y) \]

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

Filtering

Gaussian

Box filter

Filtering

Gaussian

Box Filter
Fourier Transform pairs

Spatial domain

Frequency domain

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

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

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

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

\[ x \]

\[ s \]

Low-pass, Band-pass, High-pass filters

Low-pass:

High-pass / band-pass:

Edges in images

FFT in Matlab

- 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);
fil_fft = fft2(fil, fftsize, fftsize); % 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), imshowc(log(abs(fftshift(im_fft)))), axis image, colormap jet
```
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

Image with cheetah phase (and zebra magnitude)

Image with zebra phase (and cheetah magnitude)

Slide credit: B. Freeman and A. Torralba
This is the magnitude transform of the zebra picture.

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

Slide credit: S. Marschner

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

Slide credit: S. Marschner

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?

Slide credit: S. Marschner

Subsampling by a factor of 2

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

Slide credit: D. Hoiem
**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

---

**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)

---

**Aliasing in graphics**

---

**Sampling and aliasing**

<table>
<thead>
<tr>
<th>256x256</th>
<th>128x128</th>
<th>64x64</th>
<th>32x32</th>
<th>16x16</th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="image" alt="Disintegrating textures" /></td>
<td><img src="image" alt="Sampling and aliasing" /></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
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 us 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)

Algorithm for downsampling by factor of 2

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

256x256  |  128x128  |  64x64  |  32x32  |  16x16

---

256x256  |  128x128  |  64x64  |  32x32  |  16x16

Slide credit: Forsyth and Ponce

---

**Subsampling without pre-filtering**

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

Slide credit: S. Seitz

---

**Subsampling with Gaussian pre-filtering**

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

Slide credit: S. Seitz

---

1000 pixel width

Slide credit: S. Marschner
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.

The image through the Gaussian window

\[ b(x,y) = e^{-\frac{x^2 + y^2}{2\sigma^2}} \]

Analyze local image structures

Too much

Too little

Too much

Too little

Probably still too little...

...but hard enough for now
**Analysis of local frequency**

Fourier basis:

\[ e^{j2\pi u_0 x} \]

Gabor wavelet:

\[ \psi(x, y) = e^{-\frac{x^2+y^2}{2\sigma^2}} e^{j2\pi u_0 x} \]

We can look at the real and imaginary parts:

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

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

Slide credit: B. Freeman and A. Torralba

---

**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:
\[ g(x, y) = e^{-x^2 + y^2} e^{j2\pi u_0 x} \]

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

How quadrature pair filters work

- Gabor wavelet: $\psi(x, y) = e^{-\frac{x^2 + y^2}{2\sigma^2}} e^{i2\pi u_0 x}$
- Tuning filter orientation:
  - Space: $x' = \cos(\alpha)x + \sin(\alpha)y$
  - Space: $y' = -\sin(\alpha)x + \cos(\alpha)y$

Oriented Filters

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.

Slide credit: B. Freeman and A. Torralba
**Simple example**

“Steerability” -- the ability to synthesize a filter of any orientation from a linear combination of filters at fixed orientations.

\[ G_\theta = \cos(\theta)G_{00} + \sin(\theta)G_{90} \]

Filter Set:

<table>
<thead>
<tr>
<th>0°</th>
<th>90°</th>
<th>Synthesized 30°</th>
</tr>
</thead>
</table>

Response:

Raw Image

[Image of raw images with orientations and synthesized 30° filter]

**Steerable filters**

Derivatives of a Gaussian:

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

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

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

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

The representation is “shiftable” on orientation. We can interpolate any other orientation from a finite set of basis functions.

**Local image representations**

- A pixel \([r,g,b]\)
- An image patch
- Gabor filter pair in quadrature
- Gabor jet
- V1 sketch: hypercolumns

[Diagram showing local image representations]

Slide credit: B. Freeman and A. Torralba
Gabor Filter Bank

\[ \text{or} = [12 \ 6 \ 3 \ 2]; \]
\[ \text{or} = [4 \ 4 \ 4 \ 4]; \]

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

Announcements

• There will be no classes next week!
• You will take your midterm exams on 29\textsuperscript{th} of November.
• The exam will cover all the topics we covered in the class.

Slide credit: B. Freeman and A. Torralba