Image Processing

Cosimo Distanté

Lecture 5: Pyramids
Last time: Image Filters

• Filters allow local image neighborhood to influence our description and features
  – Smoothing to reduce noise
  – Derivatives to locate contrast, gradient

• Filters have highest response on neighborhoods that “look like” it; can be thought of as template matching.

• Convolution properties will influence the efficiency with which we can process images.
  – Associative
  – Filter separability

• Edge detection processes the image gradient to find curves, or chains of edgels.
Today

• Review of Fourier Transform
• Sampling and Aliasing
• Image Pyramids
• Applications: Blending and noise removal
Background: Fourier Analysis

Figure 3.23: The Fourier Transform as the response of a filter $h(x)$ to an input sinusoid $s(x) = e^{j\omega x}$ yielding an output sinusoid $o(x) = h(x) * s(x) = Ae^{j\omega x + \phi}$.

Note symmetry in magnitude $F(w) = F(-w)$
Background: Fourier Analysis

<table>
<thead>
<tr>
<th>Property</th>
<th>Signal</th>
<th>Transform</th>
</tr>
</thead>
<tbody>
<tr>
<td>superposition</td>
<td>$f_1(x) + f_2(x)$</td>
<td>$F_1(\omega) + F_2(\omega)$</td>
</tr>
<tr>
<td>shift</td>
<td>$f(x - x_0)$</td>
<td>$F(\omega)e^{-j\omega x_0}$</td>
</tr>
<tr>
<td>reversal</td>
<td>$f(-x)$</td>
<td>$F^*(\omega)$</td>
</tr>
<tr>
<td>convolution</td>
<td>$f(x) * h(x)$</td>
<td>$F(\omega)H(\omega)$</td>
</tr>
<tr>
<td>correlation</td>
<td>$f(x) \otimes h(x)$</td>
<td>$F(\omega)H^*(\omega)$</td>
</tr>
<tr>
<td>multiplication</td>
<td>$f(x)h(x)$</td>
<td>$F(\omega) * H(\omega)$</td>
</tr>
<tr>
<td>differentiation</td>
<td>$f'(x)$</td>
<td>$j\omega F(\omega)$</td>
</tr>
<tr>
<td>domain scaling</td>
<td>$f(ax)$</td>
<td>$1/aF(\omega/a)$</td>
</tr>
<tr>
<td>real images</td>
<td>$f(x) = f^*(x)$</td>
<td>$F(\omega) = F(-\omega)$</td>
</tr>
<tr>
<td>Parseval’s Thm.</td>
<td>$\sum_x [f(x)]^2 = \sum_\omega [F(\omega)]^2$</td>
<td></td>
</tr>
</tbody>
</table>

Table 3.1: Some useful properties of Fourier transforms. The original transform pair is $F(\omega) = \mathcal{F}\{f(x)\}$.
<table>
<thead>
<tr>
<th>Name</th>
<th>Signal</th>
<th>Transform</th>
</tr>
</thead>
<tbody>
<tr>
<td>impulse</td>
<td>$\delta(x)$</td>
<td>$1$</td>
</tr>
<tr>
<td>shifted impulse</td>
<td>$\delta(x-u)$</td>
<td>$e^{-j\omega u}$</td>
</tr>
<tr>
<td>box filter</td>
<td>box($x/a$)</td>
<td>$\text{asinc}(a\omega)$</td>
</tr>
<tr>
<td>tent</td>
<td>tent($x/a$)</td>
<td>$\text{asinc}^2(a\omega)$</td>
</tr>
<tr>
<td>Gaussian</td>
<td>$G(x;\sigma)$</td>
<td>$\frac{\sqrt{2\pi}}{\sigma} G(\omega;\sigma^{-1})$</td>
</tr>
<tr>
<td>Lapl. of Gauss.</td>
<td>$(\frac{x^2}{\sigma^2} - \frac{1}{\sigma^2})G(x;\sigma)$</td>
<td>$-\frac{\sqrt{2\pi}}{\sigma} \omega^2 G(\omega;\sigma^{-1})$</td>
</tr>
<tr>
<td>Gabor</td>
<td>$\cos(\omega_0 x)G(x;\sigma)$</td>
<td>$\frac{\sqrt{2\pi}}{\sigma} G(\omega \pm \omega_0;\sigma^{-1})$</td>
</tr>
</tbody>
</table>

Table 3.2: Some useful (continuous) Fourier transforms pairs. The dashed line in the Fourier transform of the shifted impulse indicates its (linear) phase. All other transforms have zero phase (they are real-valued). Note that the figures are not necessarily drawn to scale, but are rather drawn to illustrate the general shape and characteristics of the filter or its response. In particular, the Laplacian of a Gaussian is drawn inverted because it resembles more the “Mexican Hat” it is sometimes called.
<table>
<thead>
<tr>
<th>Name</th>
<th>Kernel</th>
<th>Transform</th>
<th>Plot</th>
</tr>
</thead>
<tbody>
<tr>
<td>box-3</td>
<td>$\frac{1}{3} \begin{bmatrix} 1 &amp; 1 &amp; 1 \end{bmatrix}$</td>
<td>$\frac{1}{3}(1 + 2 \cos \omega)$</td>
<td><img src="image" alt="Plot" /></td>
</tr>
<tr>
<td>box-5</td>
<td>$\frac{1}{5} \begin{bmatrix} 1 &amp; 1 &amp; 1 &amp; 1 &amp; 1 \end{bmatrix}$</td>
<td>$\frac{1}{5}(1 + 2 \cos \omega + 2 \cos 2\omega)$</td>
<td><img src="image" alt="Plot" /></td>
</tr>
<tr>
<td>linear</td>
<td>$\frac{1}{4} \begin{bmatrix} 1 &amp; 2 &amp; 1 \end{bmatrix}$</td>
<td>$\frac{1}{2}(1 + \cos \omega)$</td>
<td><img src="image" alt="Plot" /></td>
</tr>
<tr>
<td>binomial</td>
<td>$\frac{1}{16} \begin{bmatrix} 1 &amp; 4 &amp; 6 &amp; 4 &amp; 1 \end{bmatrix}$</td>
<td>$\frac{1}{4}(1 + \cos \omega)^2$</td>
<td><img src="image" alt="Plot" /></td>
</tr>
<tr>
<td>Sobel</td>
<td>$\frac{1}{2} \begin{bmatrix} -1 &amp; 0 &amp; 1 \end{bmatrix}$</td>
<td>$\sin \omega$</td>
<td><img src="image" alt="Plot" /></td>
</tr>
<tr>
<td>“Laplacian”</td>
<td>$\frac{1}{2} \begin{bmatrix} -1 &amp; 2 &amp; -1 \end{bmatrix}$</td>
<td>$\frac{1}{2}(1 - \cos \omega)$</td>
<td><img src="image" alt="Plot" /></td>
</tr>
</tbody>
</table>

Table 3.3: Fourier transforms of the separable kernels shown in Figure 3.13.
Background: high/low pass

Original signal

Magnitude of Fourier transform signal

Low passed signal

High passed signal

Image credit: Sandberg, UC Boulder
Background: 2D FT Example

Image credit: Sandberg, UC Boulder
Background: high/low pass

Image credit: Sandberg, UC Boulder
Background: more examples

- [http://mathworld.wolfram.com/FourierTransform.html](http://mathworld.wolfram.com/FourierTransform.html)
- [http://www.cs.unm.edu/~brayer/vision/fourier.html](http://www.cs.unm.edu/~brayer/vision/fourier.html)
- ...

Magnitude vs Phase...?

- Mostly considered Magnitude spectra so far
- Sufficient for many vision methods:
  - high-pass/low-pass channel coding later in lecture.
  - simple edge detection, focus/defocus models
  - certain texture models
- May discard perceptually significant structure!
Phase and Magnitude

- Fourier transform of a real function is complex
  - difficult to plot, visualize
  - instead, we can think of the phase and magnitude of the transform
- Phase is the phase of the complex transform
- Magnitude is the magnitude of the complex transform

- 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 pic
This is the phase transform of the cheetah pic
This is the magnitude transform of the zebra pic.
This is the phase transform of the zebra pic
Reconstruction with zebra phase, cheetah magnitude
Reconstruction with cheetah phase, zebra magnitude
1D D.O.G.

Figure 3.37: The difference of two low-pass filters results in a band-pass filter. The dashed blue lines show the close fit to a half-octave Laplacian of Gaussian.
Sampling and aliasing
Sampling in 1D takes a continuous function and replaces it with a vector of values, consisting of the function’s values at a set of sample points. We’ll assume that these sample points are on a regular grid, and can place one at each integer for convenience.
\[
\int_{-\infty}^{\infty} a\delta(x)f(x)dx = a \lim_{\epsilon \to 0} \int_{-\infty}^{\infty} d(x; \epsilon)f(x)dx \\
= a \lim_{\epsilon \to 0} \int_{-\infty}^{\infty} \frac{bar(x; \epsilon)}{\epsilon}(f(x))dx \\
= a \lim_{\epsilon \to 0} \sum_{i=-\infty}^{\infty} \frac{bar(x; \epsilon)}{\epsilon}(f(i\epsilon)bar(x - i\epsilon; \epsilon))\epsilon \\
= af(0)
\]

**Figure 8.8.** Convolving a $\delta$-function with an arbitrary function can be thought of in terms of convolving a $d_{\epsilon}$ function, and taking the limit. The shaded region contributes to the integral, and as $\epsilon$ gets smaller — and so the region averaged gets narrower — the result limits to the original function.
Sampling in 2D does the same thing, only in 2D. We’ll assume that these sample points are on a regular grid, and can place one at each integer point for convenience.

\[
sample(f) = \sum_{i=-\infty}^{\infty} \sum_{j=-\infty}^{\infty} f(i, j) \delta(x - i, y - j)
\]

\[
= f(x, y) \left\{ \sum_{i=-\infty}^{\infty} \sum_{j=-\infty}^{\infty} \delta(x - i, y - j) \right\}
\]
The Fourier transform of a sampled signal

\[
F(\text{Sample}_{2D}(f(x,y))) = F\left(f(x,y) \sum_{i=-\infty}^{\infty} \sum_{j=-\infty}^{\infty} \delta(x-i,y-j)\right)
\]

\[
= F(f(x,y)) \ast F\left(\sum_{i=-\infty}^{\infty} \sum_{j=-\infty}^{\infty} \delta(x-i,y-j)\right)
\]

\[
= \sum_{i=-\infty}^{\infty} \sum_{j=-\infty}^{\infty} F(u-i,v-j)
\]
Aliasing

• Can’t shrink an image by taking every second pixel
• If we do, characteristic errors appear
  – In the next few slides
  – Typically, small phenomena look bigger; fast phenomena can look slower
  – Common phenomenon
    • Wagon wheels rolling the wrong way in movies
    • Checkerboards misrepresented in ray tracing
Space domain explanation of Nyquist sampling

You need to have at least two samples per sinusoid cycle to represent that sinusoid.
Resample the checkerboard by taking one sample at each circle. In the case of the top left board, new representation is reasonable. Top right also yields a reasonable representation. Bottom left is all black (dubious) and bottom right has checks that are too big.
Smoothing as low-pass filtering

- The message of the FT is that high frequencies lead to trouble with sampling.
- Solution: suppress high frequencies before sampling
  - multiply the FT of the signal with something that suppresses high frequencies
  - or convolve with a low-pass filter
- A filter whose FT is a box is bad, because the filter kernel has infinite support
- Common solution: use a Gaussian
  - multiplying FT by Gaussian is equivalent to convolving image with Gaussian.
Sampling without smoothing. Top row shows the images, sampled at every second pixel to get the next.
Sampling with smoothing. Top row shows the images. We get the next image by smoothing the image with a Gaussian with sigma 1 pixel, then sampling at every second pixel to get the next.
Sampling with smoothing. Top row shows the images. We get the next image by smoothing the image with a Gaussian with sigma 1.4 pixels, then sampling at every second pixel to get the next.
Sampling example

Analyze crossed gratings...
Sampling example

Analyze crossed gratings...
Sampling example

Analyze crossed gratings...
Analyze crossed gratings...

Where does perceived near horizontal grating come from?
B

F(B)
(using Szeliski notation, ‘*’ is convolution)
AB

F(A) * F(B)
\text{Lowpass}( F(A) \ast F(B) ) \sim F(C)
Control test

• If our analysis is correct, if we *add* those two sinusoids (or square waves), and if there is no non-linearity in the display of the sum, then there should only be summing, not convolution, in the frequency domain.
Low-pass filtered

$A \times B$

$F(A) \ast F(B)$

$F(A) + F(B)$

$A + B$
Image information occurs at all spatial scales
Image pyramids

- Gaussian pyramid
- Laplacian pyramid
- Wavelet/QMF pyramid
- Steerable pyramid
Image pyramids

- Gaussian pyramid
- Laplacian pyramid
- Wavelet/QMF pyramid
- Steerable pyramid
The Gaussian pyramid

- Smooth with gaussians, because
  - a gaussian*gaussian=another gaussian
- Gaussians are low pass filters, so representation is redundant.
The computational advantage of pyramids

Fig 1. A one-dimensional graphic representation of the process which generates a Gaussian pyramid. Each row of dots represents nodes within a level of the pyramid. The value of each node in the zero level is just the gray level of a corresponding image pixel. The value of each node in a high level is the weighted average of node values in the next lower level. Note that node spacing doubles from level to level, while the same weighting pattern or "generating kernel" is used to generate all levels.
Fig. 4. First six levels of the Gaussian pyramid for the "Lady" image. The original image, level 0, measures 257 by 257 pixels and each higher level array is roughly half the dimensions of its predecessor. Thus, level 5 measures just 9 by 9 pixels.
Gaussian Pyramids

Level 1: 1x1
Level 2: 2x2
Level 3: 4x4
Level 4: 8x8

Level 10: 512x512

Freeman
Image Pyramids

Idea: Represent NxN image as a “pyramid” of 1x1, 2x2, 4x4,..., 2^k x 2^k images (assuming N=2^k)

Known as a **Gaussian Pyramid** [Burt and Adelson, 1983]

- In computer graphics, a *mip map* [Williams, 1983]
- A precursor to *wavelet transform*
Reduce (1D)

\[ g_l(i) = \sum_{m=-2}^{2} \hat{w}(m) g_{l+1}(2i + m) \]

\[ g_l(2) = \hat{w}(-2) g_{l+1}(4 - 2) + \hat{w}(-1) g_{l+1} \hat{w}(4 - 1) + \hat{w}(0) g_{l+1}(4) + \hat{w}(1) g_{l+1}(4 + 1) + \hat{w}(2) g_{l+1}(4 + 2) \]

\[ g_l(2) = \hat{w}(-2) g_{l+1}(2) + \hat{w}(-1) g_{l+1} \hat{w}(3) + \hat{w}(0) g_{l+1}(4) + \hat{w}(1) g_{l+1}(5) + \hat{w}(2) g_{l+1}(6) \]
Convolution Kernel

$$[w(-2), w(-1), w(0), w(1), w(2)]$$

- Symmetric

$$w(i) = w(-i) \Rightarrow [c, b, a, b, c]$$

- Sum of mask should be 1

$$a + 2b + 2c = 1$$
Convolution Kernel

- All nodes at a given level must contribute the same total weight to the nodes at the next higher level

\[
a + 2c = 2b \quad \text{Constrained with}
\]

\[
a + 2b + 2c = 1
\]
Convolution Kernel

\[
\begin{align*}
w(0) &= a \\
w(-1) &= w(1) = \frac{1}{4} = b \\
w(-2) &= w(2) = \frac{1}{4} - \frac{a}{2} = c
\end{align*}
\]

\(a=0.4 \text{ GAUSSIAN}, \ a=0.5 \text{ TRIANGULAR}\)
Triangular
Approximate Gaussian
What about 2D?

- Separability of Gaussian

\[ \hat{I}(x, y) = I(x, y) * G(x, y) \]

Requires \( n^2 k^2 \) multiplications for \( n \times n \) image and \( k \times k \) kernel.

\[ \hat{I}(x, y) = I(x, y) * G(x) * G(y) \]

Requires \( 2kn^2 \) multiplications for \( n \times n \) image and \( k \times k \) kernel.
Algorithm

• Apply 1D mask to alternate pixels along each row of image.
• Apply 1D mask to alternate pixels along each column of resultant image from previous step.
Gaussian Pyramids Expand

\[ g_{l,n}(i) = \sum_{p=-2}^{2} \sum_{q=-2}^{2} \hat{w}(p,q) g_{l,n-1}\left(\frac{i-p}{2}, \frac{j-q}{2}\right) \]

\[ g_{l,n} = \text{EXPAND}[g_{l,n-1}] \]
Gaussian Pyramids Expand (1D)

\[ g_{l,n}(i) = \sum_{p=-2}^{2} \hat{w}(p) g_{l,n-1} \left( \frac{i-p}{2} \right) \]

\[ g_{l,n}(4) = \hat{w}(-2) g_{l,n-1} \left( \frac{4+2}{2} \right) + \hat{w}(-1) g_{l,n-1} \left( \frac{4+1}{2} \right) + \hat{w}(0) g_{l,n-1} \left( \frac{4}{2} \right) + \hat{w}(1) g_{l,n-1} \left( \frac{4-1}{2} \right) + \hat{w}(2) g_{l,n-1} \left( \frac{4-2}{2} \right) \]

\[ g_{l,n}(4) = \hat{w}(-2) g_{l,n-1}(3) + \hat{w}(0) g_{l,n-1}(2) + \hat{w}(2) g_{l,n-1}(1) \]
Gaussian Pyramids Expand (1D)

\[ g_{l,n}(i) = \sum_{p=-2}^{2} \hat{w}(p) g_{l,n-1}(\frac{i-p}{2}) \]

\[ g_{l,n}(3) = \hat{w}(-2) g_{l,n-1}(\frac{3+2}{2}) + \hat{w}(-1) g_{l,n-1}(\frac{3+1}{2}) + \]

\[ \hat{w}(0) g_{l,n-1}(\frac{3}{2}) + \hat{w}(1) g_{l,n-1}(\frac{3-1}{2}) + \hat{w}(2) g_{l,n-1}(\frac{3-2}{2}) \]

\[ g_{l,n}(3) = \hat{w}(-1) g_{l,n-1}(2) + \hat{w}(1) g_{l,n-1}(1) \]
Expand (1D)

Gaussian Pyramid

\[ g_{1,1} = \text{EXPAND}[g_{1} ] \]
Fig. 2. The equivalent weighting functions $h_i(x)$ for nodes in levels 1, 2, 3, and infinity of the Gaussian pyramid. Note that axis scales have been adjusted by factors of 2 to aid comparison. Here the parameter $a$ of the generating kernel is 0.4, and the resulting equivalent weighting functions closely resemble the Gaussian probability density functions.

Gaussian pyramids used for

• up- or down-sampling images.
• Multi-resolution image analysis
  – Look for an object over various spatial scales
  – Coarse-to-fine image processing: form blur estimate or the motion analysis on very low-resolution image, upsample and repeat. Often a successful strategy for avoiding local minima in complicated estimation tasks.
Image pyramids

- Gaussian
- Laplacian
- Wavelet/QMF
- Steerable pyramid
Image pyramids

- Gaussian
- Laplacian
- Wavelet/QMF
- Steerable pyramid
The Laplacian Pyramid

• Synthesis
  – Compute the difference between upsampled Gaussian pyramid level and Gaussian pyramid level.
  – band pass filter - each level represents spatial frequencies (largely) unrepresented at other level.
Constructing Laplacian Pyramid

- Compute Gaussian pyramid

\[ g_k, g_{k-1}, g_{k-2}, \ldots, g_2, g_1 \]

- Compute Laplacian pyramid as follows:

\[
L_k = g_k - \text{EXPAND}(g_{k-1}) \\
L_{k-1} = g_{k-1} - \text{EXPAND}(g_{k-2}) \\
L_{k-2} = g_{k-2} - \text{EXPAND}(g_{k-3}) \\
\vdots \\
L_1 = g_1
\]
Reconstructing Image

\[ g_1 = L_1 \]
\[ g_2 = \text{EXPAND}(g_1) + L_2 \]
\[ g_3 = \text{EXPAND}(g_2) + L_3 \]
\[ \vdots \]
\[ g_k = \text{EXPAND}(g_{k-1}) + L_k \]
Laplacian pyramid algorithm

\[ x_1 \]

\[ G_1 x_1 = x_2 \]

\[ F_1 G_1 x_1 \]

\[ (I - F_1 G_1) x_1 \]

\[ x_2 \]

\[ (I - F_2 G_2) x_2 \]

\[ x_3 \]

\[ (I - F_3 G_3) x_3 \]
Showing, at full resolution, the information captured at each level of a Gaussian (top) and Laplacian (bottom) pyramid.

Laplacian pyramid reconstruction algorithm: recover $x_1$ from $L_1$, $L_2$, $L_3$ and $x_4$

$G#$ is the blur-and-downsample operator at pyramid level #
$F#$ is the blur-and-upsampling operator at pyramid level #

Laplacian pyramid elements:
$L_1 = (I - F_1 G_1) x_1$
$L_2 = (I - F_2 G_2) x_2$
$L_3 = (I - F_3 G_3) x_3$
$x_2 = G_1 x_1$
$x_3 = G_2 x_2$
$x_4 = G_3 x_3$

Reconstruction of original image ($x_1$) from Laplacian pyramid elements:
$x_3 = L_3 + F_3 x_4$
$x_2 = L_2 + F_2 x_3$
$x_1 = L_1 + F_1 x_2$
Laplacian pyramid reconstruction algorithm: recover $x_1$ from $L_1$, $L_2$, $L_3$ and $g_3$
Gaussian pyramid