Lecture 3:

- Multi-scale Image Representations
- Gaussian/Laplacian Pyramids
- QMF/Wavelets
- Steerable Filters
- Image statistics

Readings: F&P Chapter 7.7, 9.2; Simoncelli et al. handout
<table>
<thead>
<tr>
<th>Lecture</th>
<th>Date</th>
<th>Description</th>
<th>Readings</th>
<th>Assignments</th>
<th>Materials</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>2/3</td>
<td>Course Introduction Cameras and Lenses</td>
<td>Req: FP 1.1, 2.1, 2.2, 2.3, 3.1, 3.2</td>
<td>PSo out</td>
<td>Lecture 1 Lecture 1 (6 slides/page)</td>
</tr>
<tr>
<td>2</td>
<td>2/5</td>
<td>Image Filtering</td>
<td>Req: FP 7.1 - 7.6</td>
<td></td>
<td>Lecture 2 Lecture 2 (6 slides/page)</td>
</tr>
<tr>
<td>3</td>
<td>2/10</td>
<td>Image Representations: pyramids</td>
<td>Req: FP 7.7, 9.2</td>
<td></td>
<td>Handout 1</td>
</tr>
<tr>
<td>4</td>
<td>2/12</td>
<td>Texture</td>
<td>Req: FP 9.1, 9.3, 9.4</td>
<td>PSo due</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>2/17</td>
<td>Monday Classes Held (NO LECTURE)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>2/19</td>
<td>Color</td>
<td>Req: FP 6.1-6.4</td>
<td>PSo out</td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>2/24</td>
<td>Local Features</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>2/26</td>
<td>Multiview Geometry</td>
<td>Req: FP 10</td>
<td>PSo due</td>
<td></td>
</tr>
<tr>
<td>9</td>
<td>3/2</td>
<td>Affine Reconstruction</td>
<td>Req: FP 12</td>
<td></td>
<td></td>
</tr>
<tr>
<td>10</td>
<td>3/4</td>
<td>Projective Reconstruction</td>
<td>Req: FP 13</td>
<td>PSo out</td>
<td></td>
</tr>
<tr>
<td>11</td>
<td>3/9</td>
<td>Scene Reconstruction</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>12</td>
<td>3/11</td>
<td>Non-Rigid Motion</td>
<td></td>
<td>PSo due</td>
<td></td>
</tr>
<tr>
<td>13</td>
<td>3/16</td>
<td>Morphable and Active Appearance Models</td>
<td></td>
<td>EX1 out</td>
<td></td>
</tr>
<tr>
<td>14</td>
<td>3/18</td>
<td>Model-Based Object Recognition</td>
<td></td>
<td>EX1 due</td>
<td></td>
</tr>
<tr>
<td>15</td>
<td>3/23-3/25</td>
<td>Spring Break (NO LECTURE)</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Last time: Linear Filters

- Convolution kernels
- Edges and contrast
- Fourier transform
- Sampling and Aliasing
Linear image transformations

- In analyzing images, it’s often useful to make a change of basis.

\[ \vec{F} = U \vec{f} \]

- Fourier transform, or
- Wavelet transform, or
- Steerable pyramid transform
An example of such a transform: the Fourier transform

discrete domain

Forward transform

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

Inverse transform

\[ f[k, l] = \frac{1}{MN} \sum_{k=0}^{M-1} \sum_{l=0}^{N-1} F[m, n] e^{+\pi i \left( \frac{km}{M} + \frac{ln}{N} \right)} \]
To get some sense of what basis elements look like, we plot a basis element --- or rather, its real part --- as a function of $x, y$ for some fixed $u, v$. We get a function that is constant when $(ux+vy)$ is constant. The magnitude of the vector $(u, v)$ gives a frequency, and its direction gives an orientation. The function is a sinusoid with this frequency along the direction, and constant perpendicular to the direction.

\[
\begin{array}{c|c|c}
\v & e^{-\pi i(ux+vy)} & e^{\pi i(ux+vy)} \\
\hline
\end{array}
\]

\[u\]
Here $u$ and $v$ are larger than in the previous slide.

\[
\begin{array}{c|c|c}
& e^{-\pi i (ux + vy)} & \\
\hline
\bullet & \cdot & u \\
\hline
\cdot & e^{\pi i (ux + vy)} & \cdot \\
\end{array}
\]
And larger still...
Thought problem

Analyze crossed gratings…
Thought problem

Analyze crossed gratings…
Analyze crossed gratings...

Thought problem
Thought problem

Analyze crossed gratings…

Where does perceived near horizontal grating come from?
A*B

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

$F(A) \ast F(B)$
\text{Lowpass}(F(A)**F(B)) \\
\sim F(C)
Today

- Image pyramids
- Image statistics
- Color and spatial frequency effects
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.
- Should naturally represent objects across varying scale.
Scaled representations

- Big bars (resp. spots, hands, etc.) and little bars are both interesting
  - Stripes and hairs, say
- Inefficient to detect big bars with big filters
  - And there is superfluous detail in the filter kernel

- Alternative:
  - Apply filters of fixed size to images of different sizes
  - Typically, a collection of images whose edge length changes by a factor of 2 (or root 2)
  - This is a pyramid (or Gaussian pyramid) by visual analogy
Example application: CMU face detector

From: http://www.ius.cs.cmu.edu/IUS/har2/har/www/CMU-CS-95-158R/
Image pyramids

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

- Smooth with gaussians, because
  - a gaussian*gaussian=another gaussian
- Synthesis
  - smooth and sample
- Analysis
  - take the top image
- Gaussians are low pass filters, so repn 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. 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.

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.
Linear image transforms

\[ \vec{F} = U \vec{f} \]

- Transformed image
- Vectorized image
- Fourier transform, or
- Wavelet transform, or
- Steerable pyramid transform
Convolution and subsampling as a matrix multiply (1-d case)

\[ U_1 = \]

\[
\begin{bmatrix}
1 & 4 & 6 & 4 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 1 & 4 & 6 & 4 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 1 & 4 & 6 & 4 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 1 & 4 & 6 & 4 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 1 & 4 & 6 & 4 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 4 & 6 & 4 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 4 & 6 & 4 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 4 & 6 & 4 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 4 & 6 & 4 & 1 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 4 & 6 & 4 & 1 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 4 & 6 & 4 & 1 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 4 & 6 & 4 & 1 & 0 & 0 \\
\end{bmatrix}
\]
Next pyramid level

\[ U_2 = \]

<table>
<thead>
<tr>
<th>1</th>
<th>4</th>
<th>6</th>
<th>4</th>
<th>1</th>
<th>0</th>
<th>0</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>4</td>
<td>6</td>
<td>4</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>4</td>
<td>6</td>
<td>4</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>4</td>
</tr>
</tbody>
</table>
b * a, the combined effect of the two pyramid levels

>> U2 * U1

ans =

1  4  10  20  31  40  44  40  31  20  10  4  1  0  0  0  0  0  0  0
0  0  0  0  1  4  10  20  31  40  44  40  31  20  10  4  1  0  0  0
0  0  0  0  0  0  0  0  1  4  10  20  31  40  44  40  30  16  4  0
0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  1  4  10  20  25  16  4  0
The Laplacian Pyramid

• Synthesis
  – preserve difference between upsampled Gaussian pyramid level and Gaussian pyramid level
  – band pass filter - each level represents spatial frequencies (largely) unrepresented at other levels

• Analysis
  – reconstruct Gaussian pyramid, take top layer
Fig 5. First four levels of the Gaussian and Laplacian pyramid. Gaussian images, upper row, were obtained by expanding pyramid arrays (Fig. 4) through Gaussian interpolation. Each level of the Laplacian pyramid is the difference between the corresponding and next higher levels of the Gaussian pyramid.
Application to image compression

Fig. 10. A summary of the steps in Laplacian pyramid coding and decoding. First, the original image $g_0$ (lower left) is used to generate Gaussian pyramid levels $g_1, g_2, \ldots$ through repeated local averaging. Levels of the Laplacian pyramid $L_0, L_1, \ldots$ are then computed as the differences between adjacent Gaussian levels. Laplacian pyramid elements are quantized to yield the Laplacian pyramid code $C_0, C_1, C_2, \ldots$. Finally, a reconstructed image $r_0$ is generated by summing levels of the code pyramid.

Oriented pyramids

Laplacian pyramid is
multi-scale
band-pass

but is over-complete

Is this a problem?
maybe

Wavelets/QMFs are multi-scale, band-pass, complete…
Wavelets/QMF’s

High and low bandpass analysis filters…

\[
U = \begin{bmatrix}
1 & 1 \\
1 & -1
\end{bmatrix} \quad \text{>> inv}(U)
\]

\[
\text{ans} = \begin{bmatrix}
0.5000 & 0.5000 \\
0.5000 & -0.5000
\end{bmatrix}
\]

*What about for synthesis?*
$$U =$$

\[
\begin{array}{cccccccccc}
1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
1 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 1 & -1 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 1 & -1 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \\
0 & 0 & 0 & 0 & 0 & 1 & -1 & 1 & 1 & -1 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
\end{array}
\]


```
>> inv(U)

ans =

0.5000   0.5000   0   0   0   0   0   0   0   0
0.5000  -0.5000   0   0   0   0   0   0   0   0
    0   0   0.5000  0.5000   0   0   0   0   0
    0   0   0.5000  -0.5000   0   0   0   0   0
    0   0   0   0   0  0.5000  0.5000   0   0
    0   0   0   0   0  0.5000 -0.5000   0   0
    0   0   0   0   0   0   0   0  0.5000  0.5000
    0   0   0   0   0   0   0   0  0.5000 -0.5000
```

<table>
<thead>
<tr>
<th>n</th>
<th>QMF-5</th>
<th>QMF-9</th>
<th>QMF-13</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0.8593118</td>
<td>0.7973934</td>
<td>0.7737113</td>
</tr>
<tr>
<td>1</td>
<td>0.3535534</td>
<td>0.41472545</td>
<td>0.42995453</td>
</tr>
<tr>
<td>2</td>
<td>-0.0761025</td>
<td>-0.073386624</td>
<td>-0.057827797</td>
</tr>
<tr>
<td>3</td>
<td>-0.060944743</td>
<td>0.02807382</td>
<td>0.039045125</td>
</tr>
<tr>
<td>4</td>
<td></td>
<td>0.021651438</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td></td>
<td></td>
<td>0.021651438</td>
</tr>
<tr>
<td>6</td>
<td></td>
<td></td>
<td>-0.014556438</td>
</tr>
</tbody>
</table>

**Table 4.1**: Odd-length QMF kernels. Half of the impulse response sample values are shown for each of the normalized lowpass QMF filters (All filters are symmetric about $n = 0$). The appropriate highpass filters are obtained by delaying by one sample and multiplying with the sequence $(-1)^n$.  

41

Figure 4.2: An analysis/synthesis filter bank.

Figure 4.3: A non-uniformly cascaded analysis/synthesis filter bank.
Figure 4.4: Octave band splitting produced by a four-level pyramid cascade of a two-band A/S system. The top picture represents the splitting of the two-band A/S system. Each successive picture shows the effect of re-applying the system to the lowpass subband (indicated in grey) of the previous picture. The bottom picture gives the final four-level partition of the frequency domain. All frequency axes cover the range from 0 to π.
To create 2-d filters, apply the 1-d filters separably in the two spatial dimensions

Figure 4.1.2: Idealized diagram of the partition of the frequency plane resulting from a 4-level pyramid cascade of separable 2-band filters. The top plot represents the frequency spectrum of the original image, with axes ranging from $-\pi$ to $\pi$. This is divided into four subbands at the next level. On each subsequent level, the lowpass subband (outlined in bold) is subdivided further.
Wavelet/QMF representation
Good and bad features of wavelet/QMF filters

• Bad:
  – Aliased subbands
  – Non-oriented diagonal subband

• Good:
  – Not overcomplete (so same number of coefficients as image pixels).
  – Good for image compression (JPEG 2000)
Steerable pyramids

• Good:
  – Oriented subbands
  – Non-aliased subbands
  – Steerable filters

• Bad:
  – Overcomplete
  – Have one high frequency residual subband, required in order to form a circular region of analysis in frequency from a square region of support in frequency.
But we need to get rid of the corner regions before starting the recursive circular filtering.

**Figure 1.** Idealized illustration of the spectral decomposition performed by a steerable pyramid with $k = 4$. Frequency axes range from $-\pi$ to $\pi$. The basis functions are related by translations, dilations and *rotations* (except for the initial highpass subband and the final lowpass subband). For example, the shaded region corresponds to the spectral support of a single (vertically-oriented) subband.
<table>
<thead>
<tr>
<th>Property</th>
<th>Laplacian Pyramid</th>
<th>Dyadic QMF/Wavelet</th>
<th>Steerable Pyramid</th>
</tr>
</thead>
<tbody>
<tr>
<td>self-inverting (tight frame)</td>
<td>no</td>
<td>yes</td>
<td>yes</td>
</tr>
<tr>
<td>overcompleteness</td>
<td>4/3</td>
<td>1</td>
<td>4k/3</td>
</tr>
<tr>
<td>aliasing in subbands</td>
<td>perhaps</td>
<td>yes</td>
<td>no</td>
</tr>
<tr>
<td>rotated orientation bands</td>
<td>no</td>
<td>only on hex lattice [9]</td>
<td>yes</td>
</tr>
</tbody>
</table>

**Table 1:** Properties of the Steerable Pyramid relative to two other well-known multi-scale representations.
Image pyramids

- **Gaussian**
  
  Progressively blurred and subsampled versions of the image. Adds scale invariance to fixed-size algorithms.

- **Laplacian**
  
  Shows the information added in Gaussian pyramid at each spatial scale. Useful for noise reduction & coding.

- **Wavelet/QMF**
  
  Bandpassed representation, complete, but with aliasing and some non-oriented subbands.

- **Steerable pyramid**
  
  Shows components at each scale and orientation separately. Non-aliased subbands. Good for texture and feature analysis.
Schematic pictures of each matrix transform

- Shown for 1-d images
- The matrices for 2-d images are the same idea, but more complicated, to account for vertical, as well as horizontal, neighbor relationships.
Fourier transform

Fourier bases are global: each transform coefficient depends on all pixel locations.

Pixel domain image

55
Gaussian pyramid

Overcomplete representation.
Low-pass filters, sampled appropriately for their blur.
Laplacian pyramid

Overcomplete representation. Transformed pixels represent bandpassed image information.
Wavelet (QMF) transform

Wavelet pyramid = Ortho-normal transform (like Fourier transform), but with localized basis functions.

pixel image
Steerable pyramid

Multiple orientations at one scale

Multiple orientations at the next scale

The next scale...

Over-complete representation, but non-aliased subbands.

Pixel image
Matlab resources for pyramids (with tutorial)
http://www.cns.nyu.edu/~eero/software.html

Eero P. Simoncelli

Associate Investigator,
Howard Hughes Medical Institute

Associate Professor,
Neural Science and Mathematics,
New York University
Matlab resources for pyramids (with tutorial)
http://www.cns.nyu.edu/~eero/software.html

Publicly Available Software Packages

- **Texture Analysis/Synthesis** - Matlab code is available for analyzing and synthesizing visual textures. README | Contents | ChangeLog | Source code (UNIX/PC, gzip'ed tar file)


- **matlabPyrTools** - Matlab source code for multi-scale image processing. Includes tools for building and manipulating Laplacian pyramids, OMF/Wavelets, and steerable pyramids. Data structures are compatible with the Matlab wavelet toolbox, but the convolution code (in C) is faster and has many boundary-handling options. README, Contents, Modification list, UNIX/PC source or Macintosh source.

- **The Steerable Pyramid**, an (approximately) translation- and rotation-invariant multi-scale image decomposition. MatLab (see above) and C implementations are available.

- **Computational Models of cortical neurons**, Macintosh program available.

- **EPIC** - Efficient Pyramid (Wavelet) Image Coder. C source code available.


Image statistics (or, mathematically, how can you tell image from noise?)
Pixel representation
image histogram
bandpass filtered image
bandpassed representation
image histogram
Pixel domain noise image and histogram
Bandpass domain noise image and histogram
Noise-corrupted full-freq and bandpass images
Bayes theorem

\[ P(x, y) = P(x|y) \ P(y) \]
so
\[ P(x|y) \ P(y) = P(y|x) \ P(x) \]
and
\[ P(x|y) = P(y|x) \ P(x) / P(y) \]

The parameters you want to estimate
What you observe
Likelihood function
Prior probability
Constant w.r.t. parameters x.
Bayesian MAP estimator for clean bandpass coefficient values

Let $x =$ bandpassed image value before adding noise.
Let $y =$ noise-corrupted observation.

By Bayes theorem

$$P(x|y) = k P(y|x) P(x)$$

$$P(x)$$

$$P(y|x)$$

$$P(x|y)$$
Bayesian MAP estimator

Let \( x \) = bandpassed image value before adding noise.
Let \( y \) = noise-corrupted observation.

By Bayes theorem

\[
P(x|y) = k \ P(y|x) \ P(x)
\]

\[
P(x)
\]

\[
P(y|x)
\]

\[
P(x|y)
\]
Bayesian MAP estimator

Let \( x = \) bandpassed image value before adding noise.
Let \( y = \) noise-corrupted observation.

By Bayes theorem

\[
P(x|y) = k \cdot P(y|x) \cdot P(x)
\]

\[
P(x)
\]

\[
P(y|x)
\]

\[
P(x|y)
\]
MAP estimate, $\hat{x}$, as function of observed coefficient value, $y$

Figure 2: Bayesian estimator (symmetrized) for the signal and noise histograms shown in figure 1. Superimposed on the plot is a straight line indicating the identity function.


Simoncelli and Adelson, Noise Removal via Bayesian Wavelet Coring
Figure 4: Noise reduction example. (a) Original image (cropped). (b) Image contaminated with additive Gaussian white noise (SNR = 9.00dB). (c) Image restored using (semi-blind) Wiener filter (SNR = 11.88dB). (d) Image restored using (semi-blind) Bayesian estimator (SNR = 13.92dB).

Simoncelli and Adelson, Noise Removal via Bayesian Wavelet Coring