Visual Computing:
Pyramids and Wavelets

Prof. Marc Pollefeys
Last lecture

- **PCA (or KL transform)**

  \[
  I_i \rightarrow \hat{I} = I - \bar{I} = U \Sigma V^T = U_k V_{k}^T
  \]

- **SSD matching vs. Eigenspace matching**

  \[
  \| I_i - I \| = \| \hat{I}_i - \hat{I} \| = \| c_i - c \| \quad \text{with} \quad c = U_k^T \hat{I}
  \]

  Eigenspace matching will typically work better because only main characteristics are preserved and irrelevant details are discarded.
Fisherfaces / LDA (Belhumeur et al. 1997)

KEY IDEAS:
• Find directions where ratio of between:within individual variance are maximized
• Linearly project to basis where dimension with good signal:noise ratio are maximized
Differences due to varying illumination can be much larger than differences between faces!

[Belhumeur, Hespanha, Kriegman, 1997]
KLT/PCA on natural image patches

from Aapo Hyvärinen et al.
JPEG image compression

Lenna, 256x256 RGB
Baseline JPEG: 4572 bytes
Campbell-Robson contrast sensitivity curve

We don’t resolve high frequencies too well...
... let’s use this to compress images... JPEG!
Lossy Image Compression (JPEG)

Block-based Discrete Cosine Transform (DCT)
JPEG Encoding and Decoding

Encoding

Decoding

www.jpeg.org
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>
Entropy Coding (Huffman code)

The code words, if regarded as a binary fractions, are pointers to the particular interval being coded.

In Huffman code, the code words point to the base of each interval.

The average code word length is $H = -\sum p(s) \log_2 p(s) \rightarrow$ optimal
JPEG compression comparison

89k

12k
Scale-space representations

- From an original signal $f(x)$ generate a parametric family of signals $f^t(x)$ where fine-scale information is successively suppressed

\[ f(x) = f_0(x) \]

[\text{scale } t]

- Family of signals generated by successive smoothing with a Gaussian filter

- Zero-crossings of 2\textsuperscript{nd} derivative: Fewer features at coarser scales

[\text{Witkin 1983}]
Image pyramid
Applications of scaled representations

• Search for correspondence
  – look at coarse scales, then refine with finer scales

• Edge tracking
  – a “good” edge at a fine scale has parents at a coarser scale

• Control of detail and computational cost in matching
  – e.g. finding stripes
  – terribly important in texture representation
Example: CMU face detection
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 representation is redundant
GAUSSIAN PYRAMID

\[ g_0 = \text{IMAGE} \]

\[ g_L = \text{REDUCE} [g_{L-1}] \]
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
LoG vs. DoG

Laplacian of Gaussian

Difference of Gaussians
Application for compression
Oriented pyramids

- Laplacian pyramid is orientation independent
- Apply an oriented filter to determine orientations at each layer
  - by clever filter design, we can simplify synthesis
  - this represents image information at a particular scale and orientation
Laplacian Pyramid Layer → \( B_1 \) → \( B_2 \) → \( B_3 \) → \( B_4 \) → Oriented Pyramid Levels

Analysis
Oriented Pyramid Levels

B_1 → B_2 → B_3 → B_4 → Laplacian Pyramid Layer

synthesis
1D Discrete Wavelet Transform

- Recursive application of a two-band filter bank to the lowpass band of the previous stage yields octave band splitting:
Haar Transform

- Haar transform $H$
  - Sample $h_k(x)$ at $\{m/N\}$
    - $m = 0, ..., N-1$
  - Real and orthogonal
  - Transition at each scale $p$ is localized according to $q$

- Basis images of 2-D (separable) Haar transform
  - Outer product of two basis vectors
Compare Basis Images of DCT and Haar

See also: Jain’s Fig.5.2 pp136
Summary on Haar Transform

- Two major sub-operations
  - Scaling captures info. at different frequencies
  - Translation captures info. at different locations
- Can be represented by filtering and downsampling
- Relatively poor energy compaction
Cascade analysis/synthesis filterbanks
Successive lowpass/highpass filtering and downsampling

- on different level: capture transitions of different frequency bands
- on the same level: capture transitions at different locations

Figure from Matlab Wavelet Toolbox Documentation
...etc
two-filterbanks with perfect reconstruction

- Impulse responses, analysis filters:
  - **Lowpass**: \( \left( \frac{-1}{4}, \frac{1}{2}, \frac{3}{2}, \frac{1}{2}, \frac{-1}{4} \right) \)
  - **highpass**: \( \left( \frac{1}{4}, \frac{-1}{2}, \frac{1}{2}, \frac{-1}{4} \right) \)

- Impulse responses, synthesis filters
  - **Lowpass**: \( \left( \frac{1}{4}, \frac{1}{2}, \frac{1}{2}, \frac{1}{4} \right) \)
  - **highpass**: \( \left( \frac{1}{4}, \frac{1}{2}, \frac{-3}{2}, \frac{1}{2}, \frac{1}{4} \right) \)

- Mandatory in JPEG2000
- Frequency responses:

  - \( |g_0| \)
  - \( |g_1| \)
  - \( |h_0| \)
  - \( |h_1| \)

“Biorthogonal 5/3 filters”
“LeGall filters”
Lifting

- **Analysis filters**

  even samples $x[2n]$  
  odd samples $x[2n+1]$

  ![Diagram of lifting process]

  - $\lambda_1$  
  - $\lambda_2$  
  - $\lambda_{L-1}$  
  - $\lambda_L$

  $K_0$  
  $K_1$

  low band $y_0$  
  high band $y_1$

- $L$ “lifting steps”

- First step can be interpreted as prediction of odd samples from the even samples

[Sweldens 1996]
Lifting (cont.)

- **Synthesis filters**
  
  Even samples $x[2n]$

  ![Diagram showing synthesis filters](image)

  Odd samples $x[2n+1]$

- **Perfect reconstruction (biorthogonality)** is directly built into lifting structure

- **Powerful for both implementation and filter/wavelet design**
Example: Lifting implementation of 5/3 filter

Verify by considering response to unit impulse in even and odd input channel.
Figure 2 Operation flow of the JPEG 2000 standard.
JPEG vs. JPEG2000

Lenna, 256x256 RGB
Baseline JPEG: 4572 bytes

Lenna, 256x256 RGB
JPEG-2000: 4572 bytes
Examples

JPEG2K vs. JPEG

From Christopoulos (IEEE Trans. on CE 11/00)

Fig. 20. Reconstructed images compressed at 0.125 bpp by means of (a) JPEG and (b) JPEG2000

Fig. 21. Reconstructed images compressed at 0.25 bpp by means of (a) JPEG and (b) JPEG2000
Next week:
Optical flow and video compression