BIL 719 - Computer Vision
Mar. 02, 2012

Aykut Erdem
Dept. of Computer Engineering
Hacettepe University

Lecture #3
frequency domain, texture, image pyramids and scale space
Why does the Gaussian give a nice smooth image, but the square filter give edgy artifacts?
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
A sum of sines

Our building block:

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

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

\[ f(\text{target}) = f_1 + f_2 + f_3 + \ldots + f_n + \ldots \]

Slide credit: A. Efros
Fourier Transform

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

\[
\begin{align*}
f(x) & \quad \rightarrow \quad \text{Fourier Transform} \quad \rightarrow \quad F(w) \\
A \sin(\omega x + \phi)
\end{align*}
\]

For every \( w \) from 0 to inf, \( F(w) \) holds the amplitude \( A \) and phase \( f \) of the corresponding sine

- How can \( F \) hold both? Complex number trick!

\[
F(\omega) = R(\omega) + iI(\omega)
\]

\[
A = \pm \sqrt{R(\omega)^2 + I(\omega)^2} \quad \phi = \tan^{-1} \frac{I(\omega)}{R(\omega)}
\]

We can always go back:

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

Slide credit: A. Efros
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
Frequency Spectra

Slide credit: A. Efros
Frequency Spectra

Slide credit: A. Efros
Frequency Spectra

Slide credit: A. Efros
Frequency Spectra

\[ A \sum_{k=1}^{\infty} \frac{1}{k} \sin(2\pi kt) \]
Example: Music

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

- Forward transform

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

- Inverse transform

\[ f[k,l] = \frac{1}{MN} \sum_{m=0}^{M-1} \sum_{n=0}^{N-1} F[m,n] e^{\pi i \left( \frac{km}{M} + \frac{ln}{N} \right)} \]
How to interpret 2D Fourier Spectrum

- **Horizontal orientation**
- **Vertical orientation**
- **45 deg.**
- **0**
- **f_{max}**
- **fx in cycles/image**

**Log power spectrum**

**Low spatial frequencies**

**High spatial frequencies**

Slide credit: B. Freeman and A. Torralba
Some important Fourier Transforms

Slide credit: B. Freeman and A. Torralba
Some important Fourier Transforms

Image

Magnitude FT

Slide credit: B. Freeman and A. Torralba
The Fourier Transform of some important images

Image

Log(1+Magnitude FT)

Slide credit: B. Freeman and A. Torralba
Fourier Amplitude Spectrum

Slide credit: B. Freeman and A. Torralba
Fourier transform magnitude
Masking out the fundamental and harmonics from periodic pillars
Signals can be composed

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

- The Fourier transform of the convolution of two functions is the product of their Fourier transforms

$$F[g \ast h] = F[g]F[h]$$

- The inverse Fourier transform of the product of two Fourier transforms is the convolution of the two inverse Fourier transforms

$$F^{-1}[gh] = F^{-1}[g] \ast F^{-1}[h]$$

- Convolution in spatial domain is equivalent to multiplication in frequency domain!
Filtering in spatial domain

Slide credit: D. Hoiem
Filtering in frequency domain

Slide credit: D. Hoiem
2D convolution theorem example

\[ f(x, y) \]

\[ h(x, y) \]

\[ g(x, y) \]

\[ |F(s_x, s_y)| \]

\[ |H(s_x, s_y)| \]

\[ |G(s_x, s_y)| \]

Slide credit: A. Efros
Filtering

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

Gaussian
Filtering

Box Filter
Fourier Transform pairs

\[ F(s) = \int_{-\infty}^{\infty} f(x) e^{-i2\pi sx} dx \]
Low-pass, Band-pass, High-pass filters

low-pass:

High-pass / band-pass:
Edges in images

Slide credit: A. Efros
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), imagesc(log(abs(fftshift(im_fft)))), axis image, colormap jet
```
• 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.
This is the magnitude transform of the zebra picture.
Reconstruction with zebra phase, cheetah magnitude

Slide credit: B. Freeman and A. Torralba
Reconstruction with cheetah phase, zebra magnitude

Slide credit: B. Freeman and A. Torralba
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.
Analyzing local image structures

Too much

Too little

Slide credit: B. Freeman and A. Torralba
The image through the Gaussian window

Too much

$$h(x,y) = e^{-\frac{x^2+y^2}{2\sigma^2}}$$

Too little

Probably still too little...

...but hard enough for now

Slide credit: B. Freeman and A. Torralba
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:
\[ \psi_c(x,y) = e^{-\frac{x^2 + y^2}{2\sigma^2}} \cos(2\pi u_0 x) \]
\[ \psi_s(x,y) = e^{-\frac{x^2 + y^2}{2\sigma^2}} \sin(2\pi u_0 x) \]

Slide credit: B. Freeman and A. Torralba
Gabor wavelets

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

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

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
Fig. 5. Top row: illustrations of empirical 2-D receptive field profiles measured by J. P. Jones and L. A. Palmer (personal communication) in simple cells of the cat visual cortex. Middle row: best-fitting 2-D Gabor elementary function for each neuron, described by (10). Bottom row: residual error of the fit, indistinguishable from random error in the Chi-squared sense for 97 percent of the cells studied.
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:

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

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

Slide credit: B. Freeman and A. Torralba
Quadrature filter pairs

edge

energy response to an edge

Slide credit: B. Freeman and A. Torralba
Quadrature filter pairs

Slide credit: B. Freeman and A. Torralba
How quadrature pair filters work

(a) Frequency response of even filter, $G$ (real)

(b) Frequency response of odd filter, $H$ (imaginary)

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 sign 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
How quadrature pair filters work

Figure 3-6: Derivation of energy measure frequency content for the filters of Fig. 3-5. (a) Fourier transform of $G \ast G$. (b) Fourier transform of $H \ast H$. Each squared response has 3 lobes in the frequency domain, arising from convolution of the frequency domain responses. The center lobe is modulated down in frequency while the two outer lobes are modulated up. (There are two sign changes which combine to give the signs shown in (b). To convolve $H$ with itself, we flip it in $f_x$ and $f_y$, which interchanges the + and – lobes of Fig. 3-5 (b). Then we slide it over an unflipped version of itself, and integrate the product of the two. That operation will give positive outer lobes, and a negative inner lobe. However, $H$ has an imaginary frequency response, so multiplying it by itself gives an extra factor of $-1$, which yields the signs shown in (b)). (c) Fourier transform of the energy measure, $G \ast G + H \ast H$. The high frequency lobes cancel, leaving only the baseband spectrum, which has been demodulated in frequency from the original bandpass response. This spectrum is proportional to the sum of the auto-correlation functions of either lobe of Fig. 3-5 (a) and either lobe of Fig. 3-5 (b).
Oriented Filters

- Gabor wavelet:
  \[ \psi(x, y) = e^{-\frac{x^2+y^2}{2\sigma^2}} e^{j2\pi m_0 x} \]

- Tuning filter orientation:
  \[ x' = \cos(\alpha)x + \sin(\alpha)y \]
  \[ y' = -\sin(\alpha)x + \cos(\alpha)y \]
Simple example

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

\[ G^1_\theta = \cos(\theta)G^1_0 + \sin(\theta)G^1_{90} \]

Filter Set:

<table>
<thead>
<tr>
<th>0°</th>
<th>90°</th>
<th>Synthesized 30°</th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="image1.png" alt="Filter 0°" /></td>
<td><img src="image2.png" alt="Filter 90°" /></td>
<td><img src="image3.png" alt="Synthesized 30°" /></td>
</tr>
</tbody>
</table>

Response:

Raw Image

![Raw Image](image4.png)


Slide credit: B. Freeman and A. Torralba
Steerable filters

Derivatives of a Gaussian:

\[ h_x(x, y) = \frac{\partial h(x, y)}{\partial x} = \frac{-x}{2\pi\sigma^4} e^{-\frac{x^2+y^2}{2\sigma^2}} \]

\[ h_y(x, y) = \frac{\partial h(x, y)}{\partial y} = \frac{-y}{2\pi\sigma^4} e^{-\frac{x^2+y^2}{2\sigma^2}} \]

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.
Fig. 3. Steerable filter system block diagram. A bank of dedicated filters process the image. Their outputs are multiplied by a set of gain maps that adaptively control the orientation of the synthesized filter.
The Design and Use of Steerable Filters

William T. Freeman and Edward H. Adelson

- William T. Freeman Edward H. Adelson,
The Design and Use of Steerable Filters,
Texture

What defines a texture?

Slide credit: K. Grauman
Texture and Material

http://www-cvr.ai.uiuc.edu/ponce_grp/data/texture_database/samples/

Slide credit: D. Hoiem
Texture and Scale

http://www-cvr.ai.uiuc.edu/ponce_grp/data/texture_database/samples/

Slide credit: D. Hoiem
What is texture?

- Regular or stochastic patterns caused by bumps, grooves, and/or markings

Images: B. Freeman and A. Torralba

Slide credit: D. Hoiem
Texture-related tasks

• Texture segmentation
  – Analyze, represent texture
  – Group image regions with consistent texture

• Texture synthesis
  – Generate new texture patches/images given some examples

• Shape from texture
  – Estimate surface orientation or shape from image texture
When are two textures similar?

All these images are different instances of the same texture. We can differentiate between them, but they seem generated by the same process.

Slide credit: B. Freeman and A. Torralba
Texture Analysis

Compare textures and decide if they’re made of the same “stuff”.

Slide credit: B. Freeman and A. Torralba
Given a finite sample of some texture, the goal is to synthesize other samples from that same texture. The sample needs to be "large enough".
Why analyze texture?

Importance to perception:
• Often indicative of a material’s properties
• Can be important appearance cue, especially if shape is similar across objects
• Aim to distinguish between shape, boundaries, and texture

Technically:
• Representation-wise, we want a feature one step above “building blocks” of filters, edges.
Psychophysics of texture

• Some textures distinguishable with preattentive perception without scrutiny, eye movements [Julesz 1975]

Textons, the elements of texture perception, and their interactions

Bela Julesz
Bell Laboratories, Murray Hill, New Jersey 07974, USA

Research with texture pairs having identical second-order statistics has revealed that the pre-attentive texture discrimination system cannot globally process third- and higher-order statistics, and that discrimination is the result of a few local conspicuous features, called textons. It seems that only the first-order statistics of these textons have perceptual significance, and the relative phase between textons cannot be perceived without detailed scrutiny by focal attention.

Pre-attentive texture discrimination


Slide credit: B. Freeman and A. Torralba
Pre-attentive texture discrimination


Slide credit: B. Freeman and A. Torralba
Pre-attentive texture discrimination

This texture pair is pre-attentively indistinguishable. Why?


Slide credit: B. Freeman and A. Torralba
Julesz - Textons

Textons: fundamental texture elements.

Textons might be represented by features such as terminators, corners, and intersections within the patterns...

Slide credit: B. Freeman and A. Torralba
We note here that simpler, lower-level mechanisms tuned for size may be sufficient to explain this discrimination.

**Observation:** the Xs look smaller than the Ls.
Early vision and texture perception

James R. Bergen* & Edward H. Adelson**

Ls 25% larger
contrast adjusted to keep mean constant

Ls 25% shorter

Slide credit: B. Freeman and A. Torralba
Texture Segmentation

Texture representation

Edges and segmentation

Slide credit: B. Freeman and A. Torralba
Texture representation

- Textures are made up of repeated local patterns, so:
  - Find the patterns
    - Use filters that look like patterns (spots, bars, raw patches...)
    - Consider magnitude of response
  - Describe their statistics within each local window
    - Mean, standard deviation
    - Histogram
    - Histogram of “prototypical” feature occurrences
Texture representation: example

- Original image
- Derivative filter responses, squared
- Statistics to summarize patterns in small windows

<table>
<thead>
<tr>
<th>Window</th>
<th>mean ( \frac{d}{dx} ) value</th>
<th>mean ( \frac{d}{dy} ) value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Win. #1</td>
<td>4</td>
<td>10</td>
</tr>
</tbody>
</table>

Slide credit: K. Grauman
Texture representation: example

original image

derivative filter responses, squared

statistics to summarize patterns in small windows

<table>
<thead>
<tr>
<th>windows</th>
<th>mean $d/\ dx$ value</th>
<th>mean $d/\ dy$ value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Win. #1</td>
<td>4</td>
<td>10</td>
</tr>
<tr>
<td>Win.#2</td>
<td>18</td>
<td>7</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
</tbody>
</table>

Slide credit: K. Grauman
Texture representation: example

original image

derivative filter responses, squared

statistics to summarize patterns in small windows

<table>
<thead>
<tr>
<th></th>
<th>mean d/dx value</th>
<th>mean d/dy value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Win. #1</td>
<td>4</td>
<td>10</td>
</tr>
<tr>
<td>Win. #2</td>
<td>18</td>
<td>7</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
</tbody>
</table>

Slide credit: K. Grauman
Texture representation: example

Original image

Derivative filter responses, squared

Statistics to summarize patterns in small windows

<table>
<thead>
<tr>
<th>Window</th>
<th>$\text{mean } d/dx$</th>
<th>$\text{mean } d/dy$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Win. #1</td>
<td>4</td>
<td>10</td>
</tr>
<tr>
<td>Win. #2</td>
<td>18</td>
<td>7</td>
</tr>
<tr>
<td>Win. #9</td>
<td>20</td>
<td>20</td>
</tr>
</tbody>
</table>

Slide credit: K. Grauman
Texture representation: example

- Dimension 1 (mean d/dx value)
- Dimension 2 (mean d/dy value)

<table>
<thead>
<tr>
<th>Window</th>
<th>Mean (d/dx) Value</th>
<th>Mean (d/dy) Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Win. #1</td>
<td>4</td>
<td>10</td>
</tr>
<tr>
<td>Win. #2</td>
<td>18</td>
<td>7</td>
</tr>
<tr>
<td>Win. #9</td>
<td>20</td>
<td>20</td>
</tr>
</tbody>
</table>

Statistics to summarize patterns in small windows

Slide credit: K. Grauman
Texture representation: example

- Windows with primarily horizontal edges
- Windows with small gradient in both directions
- Windows with primarily vertical edges
- Both

<table>
<thead>
<tr>
<th>Windows</th>
<th>mean d/dx value</th>
<th>mean d/dy value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Win. #1</td>
<td>4</td>
<td>10</td>
</tr>
<tr>
<td>Win. #2</td>
<td>18</td>
<td>7</td>
</tr>
<tr>
<td>Win. #9</td>
<td>20</td>
<td>20</td>
</tr>
</tbody>
</table>

Statistics to summarize patterns in small windows

Slide credit: K. Grauman
Texture representation: example

original image

derivative filter responses, squared

visualization of the assignment to texture “types”

Slide credit: K. Grauman
Texture representation: example

Far: dissimilar textures

Close: similar textures

Dimension 1 (mean d/dx value)
Dimension 2 (mean d/dy value)

<table>
<thead>
<tr>
<th>Win. #1</th>
<th>Mean d/dx value</th>
<th>Mean d/dy value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Win. #2</td>
<td>18</td>
<td>7</td>
</tr>
<tr>
<td>Win. #9</td>
<td>20</td>
<td>20</td>
</tr>
</tbody>
</table>

statistics to summarize patterns in small windows

Slide credit: K. Grauman
Texture representation: example

\[ D(a, b) = \sqrt{(a_1 - b_1)^2 + (a_2 - b_2)^2} \]
Distance reveals how dissimilar texture from window a is from texture in window b.

Slide credit: K. Grauman
Texture representation: window scale

- We’re assuming we know the relevant window size for which we collect these statistics.

Possible to perform scale selection by looking for window scale where texture description not changing.
Filter banks

• Our previous example used two filters, and resulted in a 2-dimensional feature vector to describe texture in a window.
  – x and y derivatives revealed something about local structure.

• We can generalize to apply a collection of multiple \((d)\) filters: a “filter bank”

• Then our feature vectors will be \(d\)-dimensional.
  – still can think of nearness, farness in feature space

Slide credit: K. Grauman
Filter banks

- What filters to put in the bank?
  - Typically we want a combination of scales and orientations, different types of patterns.

Matlab code available for these examples:
http://www.robots.ox.ac.uk/~vgg/research/texclass/filters.html

Slide credit: K. Grauman
Multivariate Gaussian

\[ p(x; \mu, \Sigma) = \frac{1}{(2\pi)^{n/2}|\Sigma|^{1/2}} \exp \left( -\frac{1}{2} (x - \mu)^T \Sigma^{-1} (x - \mu) \right). \]

\[ \Sigma = \begin{bmatrix} 9 & 0 \\ 0 & 9 \end{bmatrix} \quad \Sigma = \begin{bmatrix} 16 & 0 \\ 0 & 9 \end{bmatrix} \quad \Sigma = \begin{bmatrix} 10 & 5 \\ 5 & 5 \end{bmatrix} \]
Filter bank
You try: Can you match the texture to the response?

Filters

A

B

C

Mean abs responses

Slide credit: D. Hoeim
Representing texture by mean abs response

Filters

Mean abs responses

Slide credit: D. Hoeim
We can form a feature vector from the list of responses at each pixel.

[r1, r2, …, r38]
$d$-dimensional features

$$D(a, b) = \sqrt{\sum_{i=1}^{d} (a_i - b_i)^2}$$

Euclidean distance ($L_2$)

Slide credit: K. Grauman
Texture synthesis

- Goal: create new samples of a given texture
- Many applications: virtual environments, hole-filling, texturing surfaces

Slide credit: K. Grauman
Texture synthesis

• 2 group of works:
  1. Parametric approaches
  2. Non-parametric approaches (example-based)
The Challenge

• Need to model the whole spectrum: from repeated to stochastic texture


Slide credit: K. Grauman
Texture synthesis: Intuition

Before, we inserted the next word based on existing nearby words...

Now we want to insert **pixel intensities** based on existing nearby pixel values.

Sample of the texture (“corpus”)

Distribution of a value of a pixel is conditioned on its neighbors alone.

Slide credit: K. Grauman
What is \( P(x|\text{neighborhood of pixels around } x) \)?

Find all the windows in the image that match the neighborhood.

To synthesize \( x \)

- pick one matching window at random
- assign \( x \) to be the center pixel of that window

An exact neighbourhood match might not be present, so find the best matches using SSD error and randomly choose between them, preferring better matches with higher probability.
Varying Window Size

Increasing window size

Slide credit: A. Efros
Growing Texture

- Starting from the initial image, “grow” the texture one pixel at a time
Synthesis results

french canvas

rafia weave

Slide credit: A. Efros
Synthesis results

white bread

brick wall

Slide credit: A. Efros
Synthesis results
Failure Cases

Growing garbage

Verbatim copying

Slide credit: A. Efros
Hole Filling
Extrapolation

Slide credit: A. Efros
Texture Synthesis by Non-parametric Sampling

Alexei A. Efros and Thomas K. Leung
Computer Science Division
University of California, Berkeley
Berkeley, CA 94720-1776, U.S.A.
{efros,leungt}@cs.berkeley.edu

Image Quilting [Efros & Freeman, 2001]

- **Observation:** neighbor pixels are highly correlated

**Idea:** unit of synthesis = block

- Exactly the same but now we want $P(B \mid N(B))$
- Much faster: synthesize all pixels in a block at once
- Not the same as multi-scale!
Input texture

Random placement of blocks

Neighboring blocks constrained by overlap

Minimal error boundary cut

Slide credit: A. Efros
Minimal error boundary

overlapping blocks

vertical boundary

2

overlap error

min. error boundary

Slide credit: A. Efros
Failures
(Chernobyl Harvest)

Slide credit: A. Efros
Image Pyramids
Image pyramids

Image information occurs at all spatial scales

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

Slide credit: B. Freeman and A. Torralba
The Gaussian pyramid

• Smooth with gaussians, because
  – a gaussian*gaussian=another gaussian

• Gaussians are low pass filters, so representation is redundant.

Slide credit: B. Freeman and A. Torralba
The computational advantage of pyramids

GAUSSIAN PYRAMID

\[ g_0 = \text{IMAGE} \]

\[ g_L = \text{REDUCE} [g_{L-1}] \]

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.

[Burt and Adelson, 1983]
The Gaussian Pyramid

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.

[Burt and Adelson, 1983]
Convolution and subsampling as a matrix multiply (1D case)

$$x_2 = G_1 x_1$$

$$G_1 =$$

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

(Normalization constant of 1/16 omitted for visual clarity.)

Slide credit: B. Freeman and A. Torralba
Next pyramid level

\[ x_3 = G_2 x_2 \]

\[ G_2 = \]

\[
\begin{array}{cccccccc}
1 & 4 & 6 & 4 & 1 & 0 & 0 & 0 \\
0 & 0 & 1 & 4 & 6 & 4 & 1 & 0 \\
0 & 0 & 0 & 0 & 1 & 4 & 6 & 4 \\
0 & 0 & 0 & 0 & 0 & 0 & 1 & 4 \\
\end{array}
\]
The combined effect of the two pyramid levels

\[ x_3 = G_2 G_1 x_1 \]

\[ G_2 G_1 = \]

\[
\begin{array}{cccccccccccccccccccc}
1 & 4 & 10 & 20 & 31 & 40 & 44 & 40 & 31 & 20 & 10 & 4 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 1 & 4 & 10 & 20 & 31 & 40 & 44 & 40 & 31 & 20 & 10 & 4 & 1 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 4 & 10 & 20 & 31 & 40 & 44 & 40 & 30 & 16 & 4 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 4 & 10 & 20 & 25 & 16 & 4 & 0
\end{array}
\]
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.
1D Gaussian pyramid matrix, for \([1 \ 4 \ 6 \ 4 \ 1]\) low-pass filter

full-band image, highest resolution

lower-resolution image

lowest resolution image

Slide credit: B. Freeman and A. Torralba
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.

Slide credit: B. Freeman and A. Torralba
The Laplacian Pyramid

\[ x_1 \]

Slide credit: B. Freeman and A. Torralba
The Laplacian Pyramid

\[ x_1 \rightarrow G_1 x_1 \rightarrow \cdot \cdot \cdot \rightarrow - \rightarrow \cdot \cdot \cdot \rightarrow \cdot \cdot \cdot \]
The Laplacian Pyramid

\[ x_1 \rightarrow G_1 x_1 \rightarrow - \rightarrow F_1 G_1 x_1 \]
The Laplacian Pyramid

\[ x_1 \rightarrow G_1 x_1 \rightarrow F_1 G_1 x_1 \rightarrow (I - F_1 G_1) x_1 \]

Slide credit: B. Freeman and A. Torralba
The Laplacian Pyramid

\[ x_1 \]

\[ G_1 x_1 = x_2 \]

\[ (I - F_1 G_1) x_1 \]

\[ F_1 G_1 x_1 \]
The Laplacian Pyramid

\[ x_1 \xrightarrow{G_1} x_2 \]

\[ (I - F_1 G_1)x_1 \]

\[ (I - F_2 G_2)x_2 \]

Slide credit: B. Freeman and A. Torralba
The Laplacian Pyramid

\[ x_1 \stackrel{G_1}{\rightarrow} x_2 \stackrel{x_3}{\rightarrow} x_2 \rightarrow \frac{x_3}{(I - F_3 G_3)x_3} \]

\[ F_1 G_1 x_1 \rightarrow \frac{(I - F_2 G_2)x_2}{(I - F_1 G_1)x_1} \]

Slide credit: B. Freeman and A. Torralba
Upsampling

\[ y_2 = F_3 x_3 \]

Insert zeros between pixels, then apply a low-pass filter, \([1 \ 4 \ 6 \ 4 \ 1]\)

\[
F_3 = \begin{bmatrix}
6 & 1 & 0 & 0 \\
4 & 4 & 0 & 0 \\
1 & 6 & 1 & 0 \\
0 & 4 & 4 & 0 \\
0 & 1 & 6 & 1 \\
0 & 0 & 4 & 4 \\
0 & 0 & 1 & 6 \\
0 & 0 & 0 & 4 \\
\end{bmatrix}
\]
Showing, at full resolution, the information captured at each level of a Gaussian (top) and Laplacian (bottom) pyramid.

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.

Slide credit: B. Freeman and A. Torralba
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-upsample 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:
$x_3 = L_3 + F_3 x_4$
$x_2 = L_2 + F_2 x_3$
$x_1 = L_1 + F_1 x_2$

Slide credit: B. Freeman and A. Torralba
Laplacian pyramid reconstruction algorithm: recover $x_1$ from $L_1$, $L_2$, $L_3$ and $g_3$.

Slide credit: B. Freeman and A. Torralba
1D Laplacian pyramid matrix, for [1 4 6 4 1] low-pass filter
Laplacian pyramid applications

• Texture synthesis
• Image compression
• Noise removal

Slide credit: B. Freeman and A. Torralba
Image blending

(a)  
(b)
Figure 3.42 Laplacian pyramid blending details (Burt and Adelson 1983b) © 1983 ACM. The first three rows show the high, medium, and low frequency parts of the Laplacian pyramid.
Image blending

- Build Laplacian pyramid for both images: $L_A, L_B$
- Build Gaussian pyramid for mask: $G$
- Build a combined Laplacian pyramid: $L(j) = G(j) \cdot L_A(j) + (1 - G(j)) \cdot L_B(j)$
- Collapse $L$ to obtain the blended image

Slide credit: B. Freeman and A. Torralba
2D Haar transform

Basic elements:

\[
\begin{bmatrix}
1 \\
1 \\
\end{bmatrix} =
\begin{bmatrix}
1 & 1 \\
1 & 1 \\
\end{bmatrix}
\]

\[
\begin{bmatrix}
1 \\
1 \\
\end{bmatrix} =
\begin{bmatrix}
1 & -1 \\
1 & -1 \\
\end{bmatrix}
\]

Low pass

High pass vertical

High pass horizontal

High pass diagonal

Slide credit: B. Freeman and A. Torralba
2D Haar transform

Sketch of the Fourier transform

Horizontal low pass, vertical low-pass

Horizontal high pass, vertical low-pass

Horizontal low pass, vertical high-pass

Horizontal high pass, vertical high-pass

Slide credit: B. Freeman and A. Torralba
Figure 4.12: 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.

Slide credit: B. Freeman and A. Torralba
Wavelet/QMF representation

Same number of pixels!

Slide credit: B. Freeman and A. Torralba
Steerable Pyramid

2 Level decomposition of white circle example:

Images from: http://www.cis.upenn.edu/~eero/steerpyr.html

Slide credit: B. Freeman and A. Torralba
Steerable Pyramid

We may combine Steerability with Pyramids to get a Steerable Laplacian Pyramid as shown below

**Decomposition**

**Reconstruction**

Images from: http://www.cis.upenn.edu/~eero/steerpyr.html

Slide credit: B. Freeman and A. Torralba
Steerable Pyramid

We may combine Steerability with Pyramids to get a Steerable Laplacian Pyramid as shown below.

Decomposition  Reconstruction

Images from: http://www.cis.upenn.edu/~eero/steerpyr.html

Slide credit: B. Freeman and A. Torralba
Steerable Pyramid

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.

Simoncelli and Freeman, ICIP 1995
There is also a high pass residual...
Why use image pyramids?

- Handle real-world size variations with a constant-size vision algorithm.
- Remove noise
- Analyze texture
- Recognize objects
- Label image features
- Image priors can be specified naturally in terms of wavelet pyramids.