Lecture 5: Multi-scale pyramids
• PS1 due at midnight
• PS2 out tonight, due next Weds.
• Reading: Szeliski 3.5
• If you joined class late, make sure you have access to Gradescope and Canvas
Today

- Image pyramids
- Sampling
- Texture
We want *scale* invariance.

Source: Torralba, Freeman, Isola
Image pyramids

Source: Torralba, Freeman, Isola
Image pyramid

Source: Torralba, Freeman, Isola
Subsampling and aliasing

103×128  52×64  26×32

Idea #1: Throw away every other pixel.

Source: Torralba, Freeman, Isola
What’s happening?

Consider a sinusoid:
Undersampling

- What if we “missed” things between the samples?
- As expected, information is lost
- Unexpectedly: indistinguishable from low-frequency sinusoid!
- Also indistinguishable from higher frequencies
- **Aliasing:** signals “traveling in disguise” as other frequencies

Source: S. Marschner
Removing aliasing

- Remove the high frequencies first!
- Blur the image before downsampling

Source: Torralba, Freeman, Isola
Recall: blurring removes high frequencies

\[ F\{f \ast g\} = F\{f\} \cdot F\{g\} \]

FT of Gaussian is another Gaussian

\[ F\{\text{Gaussian } \sigma = 4\} \]

\[ f \circ g \]
Gaussian pyramid

For each level:

1. Blur input image with a Gaussian (or binomial) filter

Source: Torralba, Freeman, Isola
Gaussian pyramid

For each level:

1. Blur input image with a Gaussian (or binomial) filter
2. Downsample (throw away every other pixel)
Gaussian pyramid

256×256 → 128×128 → 64×64 → 32×32

Source: Torralba, Freeman, Isola
Gaussian pyramid

512×512 256×256 128×128 64×64 32×32

(original image)

Source: Torralba, Freeman, Isola. Image from Forsyth & Ponce
Gaussian pyramid

\[
G_0 = \begin{bmatrix}
1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\
\end{bmatrix}
\]

\[
g_1 = G_0 g_0
\]
Gaussian pyramid

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

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

\[
\begin{bmatrix}
1, 4, 6, 4, 1 \\
\end{bmatrix}
\]

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

Source: Torralba, Freeman, Isola

\[
g_1 = G_0 g_0 \quad G_0 = \frac{1}{16}
\]

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

\[ g_0 \rightarrow 256 \times 256 \]

\[ g_1 = G_0 g_0 \]

\[ g_1 \rightarrow 128 \times 128 \]

\[ g_2 = G_1 g_1 \]

\[ g_2 \rightarrow 64 \times 64 \]

\[ g_2 = G_1 G_0 g_0 \]

Source: Torralba, Freeman, Isola
For each level

1. Blur input image with a Gaussian filter
2. Downsample image
Laplacian Pyramid

Compute the difference between upsampled Gaussian pyramid level $k+1$ and Gaussian pyramid level $k$. Recall that this approximates the blurred Laplacian.
Laplacian Pyramid

Gaussian pyramid

Source: Torralba, Freeman, Isola
Laplacian Pyramid

Gaussian pyramid

\[ g_0 \rightarrow G_0 \rightarrow g_1 \rightarrow G_1 \rightarrow g_2 \rightarrow G_2 \rightarrow g_3 \]

\[ + \rightarrow F_0 \rightarrow + \rightarrow F_1 \rightarrow + \rightarrow F_2 \]

\[ l_0 \rightarrow l_1 \rightarrow l_2 \]

Laplacian pyramid

Source: Torralba, Freeman, Isola
Laplacian Pyramid

Blurring and downsampling:

\[ G_0 = \begin{bmatrix}
1 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 1 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 1 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 1 & 0 \\
\end{bmatrix} \cdot \frac{1}{16} \]

(Downsampling by 2)

Upsampling and blurring:

\[ F_0 = \]
Upsampling

64x64

Insert zeros

128x128

128x128

Source: Torralba, Freeman, 2011

\[
\begin{bmatrix}
0.25 & 0.5 & 0.25 \\
0.5 & 1 & 0.5 \\
0.25 & 0.5 & 0.25
\end{bmatrix}
\]
Laplacian Pyramid

Blurring and downsampling:

\[ G_0 = \begin{bmatrix}
1 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 1 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 1 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 1 \\
\end{bmatrix} \frac{1}{16} \]

(Downsampling by 2)

Upsampling and blurring:

\[ F_0 = \frac{1}{8} \begin{bmatrix}
6 & 4 & 1 & 0 & 0 & 0 & 0 \\
4 & 6 & 4 & 1 & 0 & 0 & 0 \\
1 & 4 & 6 & 4 & 1 & 0 & 0 \\
0 & 1 & 4 & 6 & 4 & 1 & 0 \\
0 & 0 & 1 & 4 & 6 & 4 & 1 \\
0 & 0 & 0 & 1 & 4 & 6 & 4 \\
0 & 0 & 0 & 0 & 1 & 4 & 6 \\
\end{bmatrix} \]

(blur)

\[ l_0 = (I_0 - F_0 G_0) g_0 \]

Source: Torralba, Freeman, Isola
Laplacian Pyramid

Gaussian pyramid

\[ l_0 = g_0 \rightarrow G_0 \rightarrow g_1 \rightarrow G_1 \rightarrow g_2 \rightarrow G_2 \rightarrow g_3 \]

Source: Torralba, Freeman, Isola
Laplacian Pyramid

Can we invert the Laplacian Pyramid?

Source: Torralba, Freeman, Isola
Laplacian Pyramid

Gaussian pyramid

Laplacian pyramid

Source: Torralba, Freeman, [28]
Laplacian Pyramid

Analysis/Encoder

Synthesis/Decoder

Source: Torralba, Freeman, Isola
Laplacian pyramid applications

• Texture synthesis
• Image compression
• Noise removal
• Computing image “keypoints”

Source: Torralba, Freeman, l30la
Image Blending

Source: Torralba, Freeman, Isola
Image Blending

Source: Torralba, Freeman, Isola
Image Blending

\[ I = m \cdot I^A + (1 - m) \cdot I^B \]

Source: Torralba, Freeman, 18
Image Blending with the Laplacian Pyramid

\[ l_k = l_k^A \cdot m_k + l_k^B \cdot (1 - m_k) \]
Image Blending with the Laplacian Pyramid

Source: Torralba, Freeman, Isola
Simple blend
With Laplacian pyr.

Source: A. Efros
Image Blending (PS2 problem)

- Build Laplacian pyramid for both images: $L_A, L_B$
- Build Gaussian pyramid for mask: $G$
- Build a combined Laplacian pyramid
- Collapse $L$ to obtain the blended image

Source: Torralba, Freeman, Isola
Image pyramids

Gaussian Pyramid

Laplacian Pyramid

And many more: steerable filters, wavelets, … (and later) convolutional networks!

Source: Torralba, Freeman, Isola
Orientations

Source: Torralba, Freeman, Isola
Steerable Pyramid

Oriented gradient

Source: Torralba, Freeman, Isola
Linear Image Transforms

Fourier transform

Gaussian pyr.  Laplacian pyr.  Steerable pyr.

Source: Torralba, Freeman, Isola
Texture

Stationary

Stochastic

Source: Torralba, Freeman, Isola
Texture analysis

What we’d like: are they made of the same “stuff”. Are these textures similar?
How can we represent texture in natural images?

**Idea #1**: Record simple statistics (e.g., mean, std.) of absolute filter responses
Can you match the texture to the response?

Filters

1

2

3

Mean abs. responses

A

B

C

Source: A. Efros
How can we represent texture?

- Generalize this to “orientation histogram”

- **Idea #2**: Histograms of filter responses
  - One histogram per filter

Source: A. Efros
Steerable filter decomposition

Filter bank

Input image

Source: A. Efros
Texture synthesis

Start with a noise image as output.

**Iterative algorithm** [Heeger & Bergen, 95]:
- Match pixel histogram of output image to input
- Decompose input/output images using a Steerable Pyramid
- Match histograms of input and output pyramids
- Reconstruct image and repeat

Later in the class we’ll see a simpler optimization method on neural nets [Gatys et al. 2015]

Source: A. Efros
Failure cases

Figure 7: (Left pair) Inhomogenous input texture produces blotchy synthetic texture. (Right pair) Homogenous input.

Figure 8: Examples of failures: wood grain and red coral.

Figure 9: More failures: hay and marble.
Nonparametric texture synthesis: who needs pyramids or filters?
Modeling local neighborhoods

Model $p(p \mid N(p))$, Probability of pixel given its neighbors
Efros & Leung Algorithm

- Synthesize one pixel at a time. Want to sample: $P(p|N(p))$, where $N(p)$ are the already filled-in neighbors
  - Building explicit probability tables is hard
- Instead, we search the input image for all similar neighborhoods — that’s our distribution for $p$
- To sample from this distribution, just pick one match at random

Synthesizing a pixel

Most similar neighborhoods

Source: A. Efros

Input image
Neighborhood Window

input

Source: A. Efros
Synthesis Results

french canvas

rafia weave

Source: A. Efros
More Results

white bread

brick wall

Source: A. Efros
Homage to Shannon

Source: A. Efros
Hole Filling (a.k.a. Inpainting)

Source: A. Efros