Texture
Reading: Chapter 9 (skip 9.4)

- **Key issue:** How do we represent texture?
- **Topics:**
  - Texture segmentation
  - Texture-based matching
  - Texture synthesis
    - Can be based on simpler representations than analysis
  - Shape from texture (we will skip)
Objectives: 1) Discrimination/Analysis

The Goal of Texture Analysis

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

Slide credit: Freeman
2) Synthesis

The Goal of Texture Synthesis

input image

SYNTHESIS

True (infinite) texture

generated image

Slide credit: Freeman
Representing textures

Observation: textures are made up of subelements, repeated over a region with similar statistical properties

Texture representation:
- find the subelements, and represent their statistics
- What filters can find the subelements?
  - Human vision suggests spots and oriented filters at a variety of different scales
- What statistics?
  - Mean of each filter response over region
  - Other statistics can also be useful
Human texture perception

Bergen and Adelson, Nature 1988

Learn size-tuned filter responses.

Fig. 1 Top row: Textures consisting of Xs within a texture composed of Ls. The micropatterns are placed at random orientations on a randomly perturbed lattice. a. The bars of the Xs have the same length as the bars of the Ls. b. The bars of the Ls have been lengthened by 25%, and the intensity adjusted for the same mean luminance. Discriminability is enhanced. c. The bars of the Ls have been shortened by 25%, and the intensity adjusted for the same mean luminance. Discriminability is impaired. Bottom row: the responses of a size-tuned mechanism. d, response to image a; e, response to image b; f, response to image c.
Derivative of Gaussian Filters

Measure the image gradient and its direction at different scales (use a pyramid).
Threshold squared, blurred responses, then categorize texture based on those two bits.
Add more oriented filters
(Malik & Perona, 1990)
Alternative: Gabor filters

**Gabor filters**: Product of a Gaussian with sine or cosine

Top row shows anti-symmetric (or odd) filters, bottom row the symmetric (or even) filters.

No obvious advantage to any one type of oriented filters.
The Laplacian Pyramid

• **Building a Laplacian pyramid:**
  – Create a Gaussian pyramid
  – Take the difference between one Gaussian pyramid level and the next (before subsampling)

• **Properties**
  – Also known as the difference-of-Gaussian function, which is a close approximation to the Laplacian
  – It is a band pass filter - each level represents a different band of spatial frequencies

• **Reconstructing the original image:**
  – Reconstruct the Gaussian pyramid starting at top layer
Gaussian pyramid
Laplacian Pyramid

(note top image is from Gaussian)
Oriented pyramids

• Laplacian pyramid is orientation independent
• Apply an oriented filter to determine orientations at each layer
  – This represents image information at a particular scale and orientation.
  – We will not study details in this course.
Creating oriented pyramid

Laplacian Pyramid Layer → B₁ → B₂ → B₃ → B₄ → Oriented Pyramid Levels

Creating oriented pyramid
Final texture representation

• Form a Laplacian and oriented pyramid (or equivalent set of responses to filters at different scales and orientations).
• Square the output (makes values positive)
• Average responses over a neighborhood by blurring with a Gaussian
• Take statistics of responses
  – Mean of each filter output
  – Possibly standard deviation of each filter output
Application: Texture-based Image Matching

Query image

Ordered list of best matches

Decreasing response vector similarity
The texture synthesis problem

Generate new examples of a texture.

- **Original approach:** Use the same representation for analysis and synthesis
  - This can produce good results for random textures, but fails to account for some regularities

- **Recent approach:** Use an image of the texture as the source of a probability model
  - This draws samples directly from the actual texture, so can account for more types of structure
  - Very simple to implement
  - However, depends on choosing a correct distance parameter
This is like copying, but not just repetition
Efros and Leung method

For each new pixel $p$ (select $p$ on boundary of texture):
- Match a window around $p$ to sample texture, and select several closest matches
  - Matching minimizes sum of squared differences of each pixel in the window (Gaussian weighted)
  - Give zero weight to empty pixels in the window
- Select one of the closest matches at random and use its center value for $p$
Initial conditions for growing texture

- If no initial conditions are specified, just pick a patch from the texture at random.
- To fill in an empty region within an existing texture:
  - Grow away from pixels that are on the boundary of the existing texture.
Window size parameter
More Synthesis Results

Increasing window size
Failures
Texturing a sphere

Sample image

2D

3D

Image Extrapolation
Further issues in texture synthesis

• How to improve efficiency
  – Use fast nearest-neighbor search
• How to select region size automatically
• How to edit textures to modify them in natural ways