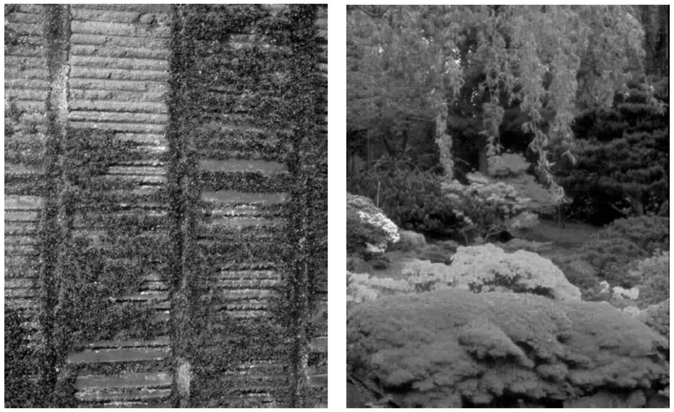
Texture

- Texture always has a scale (leaf -> bush -> forest)
- Key issue: representing texture
- Texture based matching
  - obvious thing to do, little is known
- Texture segmentation
  - key issue: representing texture
- Texture synthesis
  - useful; also gives some insight into quality of representation
- Shape from texture
  - cover superficially



Representing textures

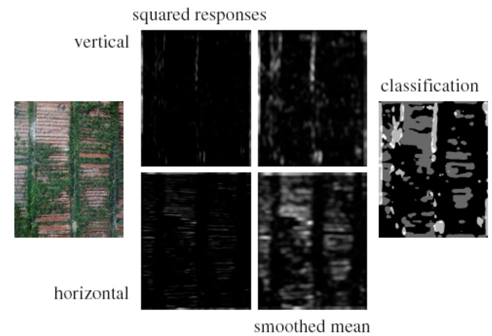
- Textures are made up of quite stylized sub-elements
  - e.g. polka-dots
- Representation:
  - find the sub-elements, and represent their **statistics**
- But what are the sub-elements, and how do we find them?
  - recall normalized correlation
  - find sub-elements by applying filters, looking at the magnitude of the response

Representing textures

- Begin with collections of responses to a variety of filters
- Generally need a collection of spots and bars at various scales and orientations (for the bars), but it is not so critical how one gets the spots and bars.
- Thus the filter banks are typically chosen based on other (often relatively arbitrary) considerations.

Representing textures

- Associate texture with **statistics** of the conglomerate of responses over some scale (window size)
- Simplest statistic is mean (square) response for each filter
  - So, N filters gives a vector of dimension N
- Including standard deviation helps
  - Now, N filters gives a vector of dimension 2*N
- These simple methods ignore spatial correlation
  - Including spatial correlation increases data by a factor of the number of pixels in a window
  - Too many, and too noisy ==> cluster point data in "textons"
  - Texture in a window is a histogram of texton popularity



A typical filter bank

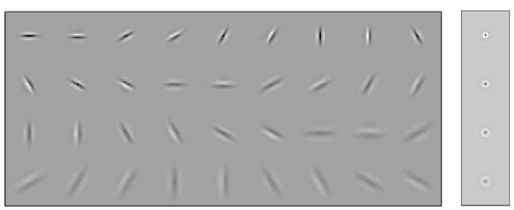
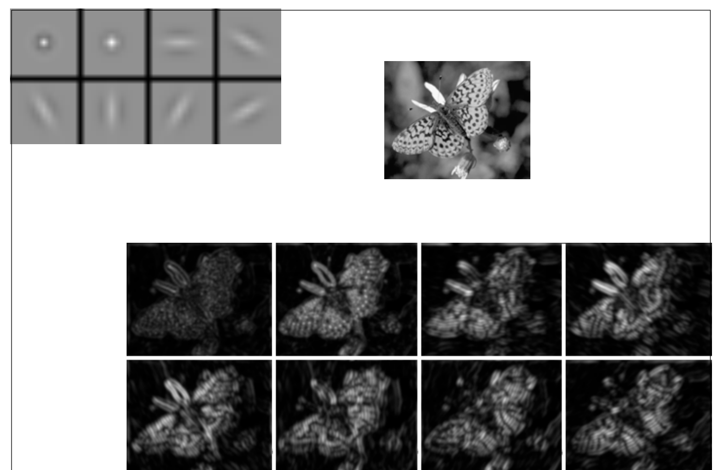
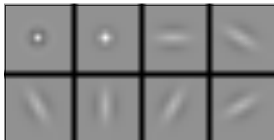


Figure 4. Left: Filter set f_i consisting of 2 phases (even and odd), 3 scales (spaced by half-octaves), and 6 orientations (equally spaced from 0 to π). The basic filter is a difference-of-Gaussian quadrature pair with $3:1$ elongation. Right: 4 scales of center-surround filters. Each filter is L_1-normalized for scale invariance.

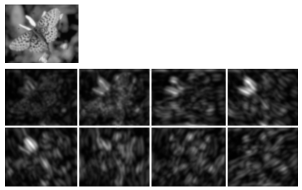
From Malik et al., "Contour and texture analysis for image segmentation"

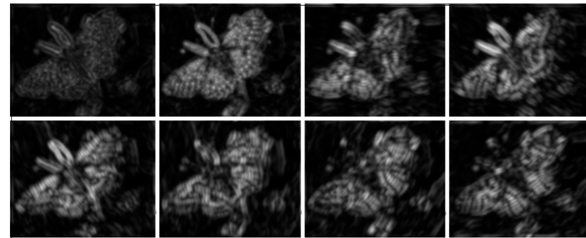
(We have an implementation for this filter bank, as part of the N-cuts software from Berkeley).



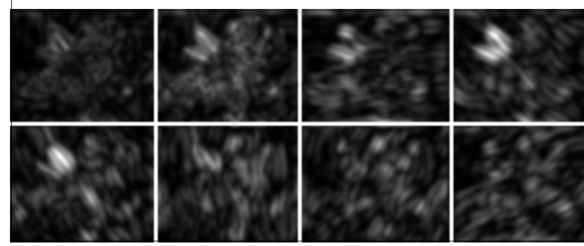


Next scale up





Smaller Scale



Larger Scale (Image from previous slide made larger to compare)

Final texture representation

- Form an oriented pyramid (or equivalent set of responses to filters at different scales and orientations).
- Square the output
- Take statistics of responses in a window (sets texture scale)
  - simplest is mean of each filter output (are there lots of spots?)
  - next most convenient enhancement is to look at standard deviation of each filter output
  - more complicates schemes are important in practice

Optional

Texture synthesis

- Use image as a source of probability model
- Grab a section of the image at random for seeding
- Expand from unfilled edges by matching boundary sections to the image, and randomly sampling the unfilled value from the blocks matched
- (For details see pages 206-207 in text)

Optional

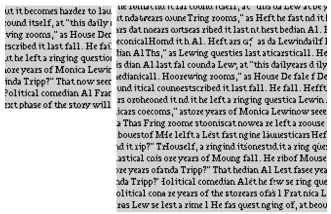


Figure from Texture Synthesis by Non-parametric Sampling, A. Efros and T.K. Leung, Proc. Int. Conf. Computer Vision, 1999 copyright 1999, IEEE