Today

Early vision on a single image

• Textures
Texture

Patterns of structure from

• changes in surface albedo (e.g. printed cloth)
• changes in surface shape (e.g. bark)
• many small surface patches (e.g. leaves on a bush)

Hard to define; but texture tells us

• what a surface is like
• (sometimes) object identity
• (sometimes) surface shape
Texture

Core problems:
- Texture segmentation
- Texture based recognition
- Texture synthesis
- Shape from texture

Key issue: representing texture

Textures are made up of
- stylized subelements
- Spatially repeated in meaningful ways

Representation:
- find the subelements
- represent their statistics
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
  • e.g. mean of each filter output (are there lots of spots)
  • std of each filter output
Example based Texture Representations

How does one choose the filters?

Solution

• build a dictionary of subelements from pictures
• describe the image using this dictionary
Building a Dictionary

FIGURE 6.8: There are two steps to building a pooled texture representation for a texture in an image domain. First, one builds a dictionary representing the range of possible pattern elements, using a large number of texture patches. This is usually done in advance, using a training data set of some form.
Clustering the Examples

K-means

- represent patches with
  - intensity vector
  - vector of filter responses over patch

Choose $k$ data points to act as cluster centers
Until the cluster centers change very little
  Allocate each data point to cluster whose center is nearest.
  Now ensure that every cluster has at least one data point; one way to do this is by supplying empty clusters with a point chosen at random from points far from their cluster center.
  Replace the cluster centers with the mean of the elements in their clusters.

Algorithm 6.3: Clustering by K-Means.

Forsyth and Ponce, “Computer Vision – A Modern Approach 2e”
Representing a Region

Vector Quantization
• Represent a high-dimensional data item with a single number
• Find and use the index of the nearest cluster center in dictionary

Summarize the pattern of patches
• Cut region into patches
• Vector quantize - vector quantized image patches often called visual words
• Build histogram of resulting numbers

Forsyth and Ponce, “Computer Vision – A Modern Approach 2e”
Representing a Region

Build a dictionary:
Collect many training example textures
Construct the vectors $x$ for relevant pixels; these could be
a reshaping of a patch around the pixel, a vector of filter outputs
computed at the pixel, or the representation of Section 6.1.
Obtain $k$ cluster centers $c$ for these examples

Represent an image domain:
For each relevant pixel $i$ in the image
    Compute the vector representation $x_i$ of that pixel
    Obtain $j$, the index of the cluster center $c_j$ closest to that pixel
    Insert $j$ into a histogram for that domain

Algorithm 6.2: Texture Representation Using Vector Quantization.
Forsyth and Ponce, “Computer Vision – A Modern Approach 2e”
FIGURE 6.9: Pattern elements can be identified by vector quantizing vectors of filter outputs, using k-means. Here we show the top 50 pattern elements (or textons), obtained from all 1,000 images of the collection of material images described in Figure 6.2. These were filtered with the complete set of oriented filters from Figure 6.4. Each subimage here illustrates a cluster center. For each cluster center, we show the linear combination of filter kernels that would result in the set of filter responses represented by the cluster center. For some cluster centers, we show the 25 image patches in the training set whose filter representation is closest to the cluster center. *This figure shows elements of a database collected by C. Liu, L. Sharan, E. Adelson, and R. Rosenholtz, and published at [http://people.csail.mit.edu/lavanya/research_sharan.html](http://people.csail.mit.edu/lavanya/research_sharan.html). Figure by kind permission of the collectors.*
FIGURE 6.10: Pattern elements can also be identified by vector quantizing vectors obtained by reshaping an image window centered on each pixel. Here we show the top 50 pattern elements (or textons), obtained using this strategy from all 1,000 images of the collection of material images described in Figure 6.2. Each subimage here illustrates a cluster center. For some cluster centers, we show the closest 25 image patches. To measure distance, we first subtracted the average image intensity, and we weighted by a Gaussian to ensure that pixels close to the center of the patch were weighted higher than those far from the center. This figure shows elements of a database collected by C. Liu, L. Sharan, E. Adelson, and R. Rosenholtz, and published at http://people.csail.mit.edu/lavanya/research_sharan.html. Figure by kind permission of the collectors.
Texture Representations

A vector summarizing the trends in pattern elements
- either overall trend in filter responses
- or histogram of vector quantized patches

At a pixel
- compute representations for a patch centered on the pixel

For a region
- compute representations for the whole region
Texture Synthesis

Problem:
• Take a small example image of pure texture
• Use this to produce a large domain of “similar” texture

Why:
• Computer graphics demands lots of realistic texture
• Fill in holes in images created by removing objects
Texture Synthesis

Simple case -- Fill holes
- Synthesize a single pixel in a large image
- Approach:
  - Match the window around that pixel to other windows in the image
  - Choose a value from the matching windows
    - most likely, uniformly and at random

Expand to large images
- Start: take a piece of the example image
- Fill in pixels on the boundary
  - But these are missing more than the center
    - Discount missing pixels when matching
- Each time you fill in a pixel, you can use that to match
Choose a small square of pixels at random from the example image
Insert this square of values into the image to be synthesized
Until each location in the image to be synthesized has a value
   For each unsynthesized location on
       the boundary of the block of synthesized values
       Match the neighborhood of this location to the
       example image, ignoring unsynthesized
       locations in computing the matching score
       Choose a value for this location uniformly and at random
       from the set of values of the corresponding locations in the
       matching neighborhoods
   end
end

Algorithm 6.4: Non-parametric Texture Synthesis.
Small blocks are examples, large are synthesized. Notice how (for example) synthesized text looks like actual text.

Efros and Leung 1999
Neighborhood size
Fill in holes by looking for example patches in the image. If needed, rectify faces (lower images).