Computer vision – Textures

Elad Osherov Dec 2015
Problem:

• Objects do not appear as uniform color/intensity regions due to variable surface geometry, lighting, albedo etc.
• Sometimes the “variation is uniform” - Texture
  • Variation
  • Repetitiveness
Texture

Computer Vision

Textures
How texture affects perception

• Texture generates edges inside objects

• Texture helps to identify objects

• Texture helps to segment objects (can overcome significant intensity change, shadows)

• Texture helps to find 3D surface shape

• Human vision can detect texture differences, pre-attentively
Texture tasks

Texture Analysis
Take two patches
Are they the same “stuff”? 
Is this a wall picture?

Texture Synthesis
Take a patch
Synthesize a different patch of the same texture

“Same” or “different”
### Texture analysis

**How to decide that two patches are the same texture?**

1. Find a descriptor for each texture patch
2. Compare descriptors

**But which descriptors?**

**How to quantify the type of variation?**

**Some naïve approaches:**

- Use gray levels vector
- Measure “busyness” – How many edges in a patch?
- Histogram of gradient directions
A **co-occurrence matrix** is a matrix in which:

- Both the rows and columns represent a set of possible image values.
- \(C(i,j)\) indicates how many times value \(i\) co-occurs with value \(j\) in a particular spatial relationship \(d\).
- The spatial relationship is specified by a vector \(d = (dr,dc)\).
- Co-occurrence matrix = probability distributions for intensity pairs.
The co-occurrence matrix - example

\[
\begin{array}{cccc}
1 & 1 & 0 & 0 \\
1 & 1 & 0 & 0 \\
0 & 0 & 2 & 2 \\
0 & 0 & 2 & 2 \\
0 & 0 & 2 & 2 \\
0 & 0 & 2 & 2 \\
\end{array}
\]

\[d = (3,1)\]

From \(C_d\) we can compute \(N_d\), the normalized co-occurrence matrix, where each value is divided by the sum of all the values.
The co-occurrence matrix - features

\[
\text{Energy} = \sum_i \sum_j N_d^2(i, j) \\
\text{Entropy} = -\sum_i \sum_j N_d(i, j) \log_2 N_d(i, j) \\
\text{Contrast} = \sum_i \sum_j (i - j)^2 N_d(i, j) \\
\text{Homogeneity} = \sum_i \sum_j \frac{N_d(i, j)}{1 + |i - j|} \\
\text{Correlation} = \frac{\sum_i \sum_j (i - \mu_i)(j - \mu_j)N_d(i, j)}{\sigma_i \sigma_j}
\]

(7.7) (7.8) (7.9) (7.10) (7.11)

where \( \mu_i, \mu_j \) are the means and \( \sigma_i, \sigma_j \) are the standard deviations of the row and column

Energy measures uniformity of the normalized matrix.
The co-occurrence matrix - problems

• How to choose d?
• This is actually a critical question with all the statistical texture methods.

• Are the “texels” tiny, medium, large, all three ...?

• Not really a solved problem.

A possible solution: Zucker and Terzopoulos suggested using a $\chi^2$ statistical test to select the value(s) of d that have the most structure for a given class of images. Meaning maximizing over:

$$\chi^2(d) = \left( \sum_i \sum_j \frac{N_d^2(i,j)}{N_d(i)N_d(j)} - 1 \right)$$
The autocorelation function

\[ \rho(dr, dc) = \frac{\sum_{r=0}^{N} \sum_{c=0}^{N} I[r,c]I(r+dr,c+dc)}{\sum_{r=0}^{N} \sum_{c=0}^{N} I^2[r,c]} \]

\[ = \frac{I[r,c] \circ I_d[r,c]}{I[r,c] \circ I[r,c]} \]

- Related to co-occurrence
- Compare the dot product (energy) of non shifted image with a shifted image
- Autocorrelation function can detect repetitive patterns of texels
- Also defines fineness/coarseness of the texture
The autocorrelation function - interpretation

\[ \rho(dr, dc) = \frac{\sum_{r=0}^{N} \sum_{c=0}^{N} I[r,c]I(r+dr,c+dc)}{\sum_{r=0}^{N} \sum_{c=0}^{N} I^2[r,c]} \]

- **Random texture:**
  
  only peak at \([0, 0]\); breadth of peak gives the size of the texture.

- **Regular texture:**
  
  function will have peaks and valleys; peaks can repeat far away from \([0, 0]\).

- **Coarse texture:**
  
  function drops off slowly.

- **Fine texture:**
  
  function drops off rapidly.

- Can drop differently for \(r\) and \(c\)
Filter based approaches

- Filter the image with several filters emphasizing different local properties of the variation
  - Laws filters: intensity, edge, LoG, ripple
  - Gabor filters: products of sin/cos function with Gaussians, similar to Gaussian derivatives.
  - Possibly normalize by intensity to get invariance.
- Average over neighborhoods to get a vector of descriptors
Laws filters

Signal-processing-based algorithms use texture filters applied to the image to create filtered images from which texture features are computed.

The Laws Algorithm:
1. Filter the input image using texture filters
2. Compute texture energy by summing the absolute value of filtering results in local neighborhoods around each pixel
3. Combine features to achieve rotational invariance
Laws filters

L5  (Level) = [  1  4  6  4  1  ]
E5  (Edge) = [ -1 -2  0  2  1  ]
S5  (Spot) = [ -1  0  2  0 -1  ]
R5  (Ripple) = [  1 -4  6 -4  1  ]

- (L5) (Gaussian) gives a center-weighted local average
- (E5) (gradient) responds to row or col step edges
- (S5) (LOG) detects spots
- (R5) (Gabor) detects ripples
Laws filters

- 1D Masks are “multiplied” to construct 2D masks:
  mask E5L5 is the “product” of E5 and L5 –

\[
\begin{bmatrix}
-1 \\
-2 \\
0 \\
2 \\
1
\end{bmatrix}
\times
\begin{bmatrix}
1 & 4 & 6 & 4 & 1
\end{bmatrix}
= \begin{bmatrix}
-1 & -4 & -6 & -4 & -1 \\
-2 & -8 & -12 & -8 & -1 \\
0 & 0 & 0 & 0 & 0 \\
2 & 8 & 12 & 8 & 2 \\
1 & 4 & 6 & 4 & 1
\end{bmatrix}
\]
Laws filters – 9D feature vector

1. Subtract mean neighborhood intensity from (center) pixel
2. Apply 16 5x5 masks to get 16 filtered images $F_k, k=1$ to 16
3. Produce 16 texture energy maps using 15x15 windows $E_k[r,c] = \sum |F_k[i,j]|$
4. Replace each distinct pair with its average map
5. 9 features (9 filtered images) defined as follows:

<table>
<thead>
<tr>
<th>L5E5/E5L5</th>
<th>L5S5/S5L5</th>
</tr>
</thead>
<tbody>
<tr>
<td>L5R5/R5L5</td>
<td>E5E5</td>
</tr>
<tr>
<td>E5S5/S5E5</td>
<td>E5R5/R5E5</td>
</tr>
<tr>
<td>S5S5</td>
<td>S5R5/R5S5</td>
</tr>
<tr>
<td>R5R5</td>
<td></td>
</tr>
</tbody>
</table>
Laws filters – 9D feature vector
Laws filters – example

- water
- tiger
- fence
- flag
- grass
- small flowers
- big flowers

(a) Original image
(b) Segmentation into 4 clusters
(c) Original image
(d) Segmentation into 4 clusters
(e) Original image
(f) Segmentation into 3 clusters
Gabor filters

- Similar approach to laws
- Wavelets at different frequencies and different orientations

Usage
- Construct a Gabor filter bank
- Filter the image and build a feature vector of responses
- Cluster/Classify accordingly
Gabor filters

Figure 4: Gabor filter composition: (a) 2D sinusoid oriented at 30° with the x-axis, (b) a Gaussian kernel, (c) the corresponding Gabor filter. Notice how the sinusoid becomes spatially localized.

Figure 5: Example of Gabor filters with different frequencies and orientations. First column shows their 3D plots and the second one, the intensity plots of their amplitude along the image plane.
Gabor filters – segmentation example

Segmentation with Color and Gabor-Filter Texture (Smeulders)
What is Clustering

• Organizing data into classes such that:
  • high intra-class similarity
  • low inter-class similarity

• Finding the class labels and the number of classes directly from the data (in contrast to classification).
What is Clustering

• Natural grouping

Often there are several solutions
Partitional Clustering

• Nonhierarchical, each instance is placed in exactly one of K non-overlapping clusters.

• Since only one set of clusters is output, the user normally has to input the desired number of clusters K.
Objective Function

Squared error

\[ s_e K_i = \sum_{j=1}^{m} \| t_{ij} - C_k \|^2 \]

\[ s_e K = \sum_{j=1}^{k} s_e K_j \]
The KMeans algorithm

1. Decide on a value for $k$.

2. Initialize the $k$ cluster centers (randomly, if necessary).

3. Decide the class memberships of the $N$ objects by assigning them to the nearest cluster center.

4. Re-estimate the $k$ cluster centers, by assuming the memberships found above are correct.

5. If none of the $N$ objects changed membership in the last iteration, exit. Otherwise goto 3.
The Kmeans algorithm - Example

Distance Metric: Euclidean Distance

$\mathbf{k_1}$

$\mathbf{k_2}$

$\mathbf{k_3}$
The Kmeans algorithm - Example
The Kmeans algorithm - Example
The K-means algorithm - Example
Of all the sensory impressions proceeding to the brain, the visual experiences are the dominant ones. Our perception of the world around us is based essentially on the messages that reach the brain from our eyes. It was thought that the retinal image was transmitted point by point to visual centers in the brain; the cerebral cortex was a movie screen, so to speak, upon which the image was projected. Through the discoveries of Hubel and Wiesel we now know that behind the origin of the visual perception in the brain there is a considerably more complicated course of events. By following the visual impulses along their path to the various cell layers of the optical cortex, Hubel and Wiesel have been able to demonstrate that the message about the image falling on the retina undergoes a step-wise analysis in a system of nerve cells stored in columns. In this system each cell has its specific function and is responsible for a specific detail in the pattern of the retinal image.

China is forecasting a trade surplus of $90bn (£51bn) to $100bn this year, a threefold increase on 2004's $32bn. The Commerce Ministry said the surplus would be created by a predicted 30% jump in exports to $750bn, compared with a 18% rise in imports to $660bn. The ministry also said that China's imports would be raised to $1.3trn. The Commerce Ministry said the surplus would be created by a predicted 30% jump in exports to $750bn, compared with a 18% rise in imports to $660bn. The ministry also said that China's imports would be raised to $1.3trn.
BoW model - Basics

• Split into train and test sets

• Train:
  • Extract image patches and features (e.g. texture)
  • Learn a dictionary (of texture words) using Kmeans
  • Calculate histograms of each category
  • Learn a classifier accordingly

• Test:
  • Extract image patches and features
  • Associate each texture patch to a visual word
  • Calculate histograms of each category
  • Classify using the learned classifier
BoW model - Textures

• Texture is characterized by the repetition of basic elements or textons

• For stochastic textures, it is the identity of the textons, not their spatial arrangement, that matters
BoW model - Textures
BoW model - Textures

• Also called Histogram of local types

• Train
  • Classify the local neighborhood into “types”
  • The common method:
    • Characterize every local neighborhood by some vector, e.g. responses of
different (Gabor) filters.
  • Run K-means clustering
  • Assign every local neighborhood to the nearest cluster

• Test:
  • The texture descriptor is the histogram of types
  • The histogram is taken over a neighborhood
  • The size of the neighborhood is fixed or adaptive
Recap

- Texture is a repetitive variable pattern in an image.
- Multiple approaches to classify
  - Co-occurrence matrix
  - Auto correlation function
  - Laws filters
  - Gabor filters
  - Bag of texture words
  - Kmeans algorithm for dictionary learning