Texture and materials

Subhransu Maji

CMPSCI 670: Computer Vision

December 1, 2016

What does texture tell us?

- Indicator of materials properties, e.g. brick vs wooden
- Complementary to shape

correlated with identity but not the same

Lecture outline

- Texture perception
  - Texture attributes
  - Describing textures from images
- Texture representation
  - Filter-banks and bag-of-words
  - CNN filter-banks for texture
Pre-attentive texture segmentation

- Phenomena in which two regions of texture quickly (i.e., in less than 250 ms) and effortlessly segregate

Led to early models of texture representation “textons”


High-level attributes of texture

- Early works include:
  - Orientation, contrast, size, spacing, location [Bajcsy 1973]
  - Coarseness, contrast, directionality, line-like, regularity, roughness [Tamura et al., 1978]
  - Coarseness, contrast, busyness, complexity and texture strength [Amadusen and King, 1989]

- These attributes can be measured reasonably well from images using low-level statistics of pixel intensities

Towards a texture lexicon


56 images from Brodatz

Descrivable texture dataset

- From human perception to computer vision
- 47 attributes (after accounting for synonyms, etc)
- 120+ images per attribute (crowdsourced)


56 images from Brodatz
Human centric applications

Properties complementary to materials

Find striped wallpaper
or describing patterns in clothing

Retrieving fabrics and wallpapers

Automatic predictions using computer vision (more later…)

Talk outline

- Texture perception
  - Texture attributes
  - Describing textures in the wild [CVPR 14]
- Texture representation
  - Filter-banks and bag-of-words
  - CNN filter-banks for texture [CVPR 15, IJCV 16]

Texture representation

- Textures are made up of repeated local patterns
  - Use filters that look like patterns — spots, edges, bars

Describe their statistics within each image/region
Filter bank response

[r1, r2, ..., r38]

“Bag of words” for texture

- Absolute positions of local patterns don’t matter as much
- Bag of words approach:
  - Inspired by text representation, i.e., document ~ word counts
  - In vision we don’t have a pre-defined dictionary
    - Learn words by clustering local responses (Vector quantization)
  - Computational basis of “textons” [Julesz, 1981]

Bag of words is only counting the number of local descriptors assigned to each word (Voronoi cell)

- Why not include other statistics? For instance:
  - Mean of local descriptors \( \bar{x} \)

Learning attributes on DTD

<table>
<thead>
<tr>
<th>Local descr.</th>
<th>Kernel</th>
<th>Linear</th>
<th>Hellinger</th>
<th>add-( \chi^2 )</th>
<th>exp-( \chi^2 )</th>
</tr>
</thead>
<tbody>
<tr>
<td>MR8</td>
<td>15.9 ( \pm ) 0.8</td>
<td>19.7 ( \pm ) 0.8</td>
<td>24.1 ( \pm ) 0.7</td>
<td>30.7 ( \pm ) 0.7</td>
<td></td>
</tr>
<tr>
<td>LM</td>
<td>18.8 ( \pm ) 0.5</td>
<td>25.8 ( \pm ) 0.8</td>
<td>31.6 ( \pm ) 1.1</td>
<td>39.7 ( \pm ) 1.1</td>
<td></td>
</tr>
<tr>
<td>Patch(_3\times3)</td>
<td>14.6 ( \pm ) 0.6</td>
<td>22.3 ( \pm ) 0.7</td>
<td>26.0 ( \pm ) 0.8</td>
<td>30.7 ( \pm ) 0.9</td>
<td></td>
</tr>
<tr>
<td>Patch(_7\times7)</td>
<td>18.0 ( \pm ) 0.4</td>
<td>26.8 ( \pm ) 0.7</td>
<td>31.6 ( \pm ) 0.8</td>
<td>37.1 ( \pm ) 1.0</td>
<td></td>
</tr>
<tr>
<td>LBP(_8)</td>
<td>8.2 ( \pm ) 0.4</td>
<td>9.4 ( \pm ) 0.4</td>
<td>14.2 ( \pm ) 0.6</td>
<td>24.8 ( \pm ) 1.0</td>
<td></td>
</tr>
<tr>
<td>LBP-VQ</td>
<td>21.1 ( \pm ) 0.8</td>
<td>23.1 ( \pm ) 1.0</td>
<td>28.5 ( \pm ) 1.0</td>
<td>34.7 ( \pm ) 1.3</td>
<td></td>
</tr>
<tr>
<td>SIFT</td>
<td><strong>34.7 ( \pm ) 0.8</strong></td>
<td><strong>45.5 ( \pm ) 0.9</strong></td>
<td><strong>49.7 ( \pm ) 0.8</strong></td>
<td><strong>53.8 ( \pm ) 0.8</strong></td>
<td></td>
</tr>
</tbody>
</table>

Bag of words (~1k words) representations on DTD dataset

SIFT works quite well

David Lowe, ICCV 99

Dealing with quantization error

- Bag of words is only counting the number of local descriptors assigned to each word (Voronoi cell)
- Why not include other statistics? For instance:
  - Mean of local descriptors \( \bar{x} \)
Dealing with quantization error

- Bag of words is only **counting** the number of local descriptors assigned to each word (Voronoi cell)
- Why not include other statistics? For instance:
  - Mean of local descriptors $x$
  - Covariance of local descriptors

The VLAD descriptor

Given a codebook $\{\mu_i, i = 1 \ldots N\}$, e.g. learned with K-means, and a set of local descriptors $X = \{x_t, t = 1 \ldots T\}$:

1. assign: $\text{NN}(x_t) = \arg \min_{\mu_i} \| x_t - \mu_i \|
2. compute: $v_i = \sum_{x_t: \text{NN}(x_t) = \mu_i} x_t - \mu_i$
3. concatenate $v_i$'s + $\ell_2$ normalize

Very high dimensional: NxD

Fisher-vectors with SIFT

SIFT BoVW + linear SVM: mAP = **37.4** +27%

SIFT IFV on DTD mAP: 64.52

Descivable attributes as features

- Train classifiers to predict 47 attributes
  - SIFT + AlexNet features to make predictions
  - On a new dataset, learn classifiers on 47 features

<table>
<thead>
<tr>
<th>Features</th>
<th>KTH-2b</th>
<th>FMD</th>
</tr>
</thead>
<tbody>
<tr>
<td>DTD</td>
<td>73.8%</td>
<td>61.1%</td>
</tr>
<tr>
<td>Prev best</td>
<td>57.1%</td>
<td>66.3%</td>
</tr>
<tr>
<td>DTD + SIFT + DeCAF</td>
<td>77.1%</td>
<td>67.1%</td>
</tr>
</tbody>
</table>

47 dim

66K dim

- DTD attributes correlate well with material properties
The quest for better features …

- Early filter banks were based on simple linear filters - is there something better? Can we learn them from data?
- Slow progress for a while and performance plateaued on a number of benchmarks, e.g. PASCAL VOC

![Image of PASCAL VOC challenge dataset](source: http://pascallin.ecs.soton.ac.uk/challenges/VOC/voc2007/results/index.html)

**CNNs as feature extractors**

- Take the outputs of various layers: `conv5, fc6, fc7`
- State of the art on many datasets (Donahue et al, ICML 14)
- Regions with CNN features (Girshick et al., CVPR 14) achieves 41% ± 53.7% on PASCAL VOC 2007 detection challenge. Current best results 66%!
- A flurry of activity in computer vision; benchmarks are being shattered every few months! Great time for vision applications

**ImageNet classification breakthrough**

- "AlexNet" CNN
  - 60 million parameters trained on 1.2 million images
  - +1 for crowdsourcing

**CNNs for texture**

- CNN features from the last layer don’t seem to outperform SIFT on texture datasets
- Speculations on why:
  - Textures are different from categories on ImageNet which are mostly objects
  - Dense layers preserve spatial structure are not ideal for measuring orderless statistics

<table>
<thead>
<tr>
<th>Dataset</th>
<th>FV (SIFT)</th>
<th>AlexNet</th>
</tr>
</thead>
<tbody>
<tr>
<td>CUBeT</td>
<td>99.5</td>
<td>97.9</td>
</tr>
<tr>
<td>UMD</td>
<td>99.2</td>
<td>96.4</td>
</tr>
<tr>
<td>UIUC</td>
<td>97.0</td>
<td>94.2</td>
</tr>
<tr>
<td>KT</td>
<td>99.7</td>
<td>96.9</td>
</tr>
<tr>
<td>KT-2a</td>
<td>82.2</td>
<td>78.9</td>
</tr>
<tr>
<td>KT-2b</td>
<td>69.3</td>
<td>70.7</td>
</tr>
<tr>
<td>FMD</td>
<td>58.2</td>
<td>60.7</td>
</tr>
<tr>
<td>DTD</td>
<td>61.2</td>
<td>54.8</td>
</tr>
<tr>
<td><strong>mean</strong></td>
<td><strong>83.3</strong></td>
<td><strong>81.3</strong></td>
</tr>
</tbody>
</table>

Texture recognition accuracy

Flickr material dataset (10 categories)

http://people.csail.mit.edu/celiu/CVPR2010/FMD/
CNN layers are non-linear filter banks

Obtain filter banks by truncating the CNN

CNNs for texture

Texture recognition accuracy

<table>
<thead>
<tr>
<th>Dataset</th>
<th>FV (SIFT)</th>
<th>AlexNet (FC)</th>
<th>FV (conv5)</th>
</tr>
</thead>
<tbody>
<tr>
<td>KT-2b</td>
<td>69.3</td>
<td>70.7</td>
<td>71.0</td>
</tr>
<tr>
<td>FMD</td>
<td>58.2</td>
<td>60.7</td>
<td>72.6</td>
</tr>
<tr>
<td>DTD</td>
<td>61.2</td>
<td>54.8</td>
<td>66.7</td>
</tr>
</tbody>
</table>

Significant improvements over simply using CNN features

Using the model from Oxford VGG group that performed the best on LSVRC 2014 (ImageNet classification challenge)

http://www.robots.ox.ac.uk/~vgg/research/very_deep/
Scenes and objects as textures

- MIT Indoor dataset (67 classes)
  Prev. best: **70.8%**  D-CNN **81.7%**  
  Zhou et al., NIPS 14

- CUB 200 dataset (bird sub-category recognition)
  Prev. best: **76.4%** (w/ parts)  FV-CNN **72.1%** (w/o parts)
  Zhang et al., ECCV 14

SIFT vs. CNN filter banks

OpenSurfaces material segmentation

- MSRC segmentation dataset
  FV-CNN **87.0%** vs **86.5%** [Ladicky et al., ECCV 2010]