Analysis: TextonBoost and Semantic Texton Forests

Daniel Munoz
16-721
February 9, 2009
Papers

Problem

- Ultimate goal for both these papers:
  - Simultaneous segmentation and recognition of objects in images

[shotton-eccv-06] [shotton-cvpr-08]

- Simultaneous segmentation and recognition of objects in images
Data and Classes

- Goal: assign every pixel to a label

- MSRC-21 database ("void" label ignored for training and testing)
Claimed contributions

- Discriminative model capable of fusing
  - shape
  - appearance
  - context
  information to efficiently recognize and accurately segment the object classes present in an image

- New texton-based features which are capable of modeling object shape, appearance and context.

- Efficient training of model on large dataset with many labels
  - Piece-wise CRF training with boosting
Outline

- High-level description of approach:
  - Learn classifier based on relative texture locations for each class
  - Refine classification with Conditional Random Field (CRF)
  - Improve classification with additional pixel information

- Review of CRFs….
Conditional Random Fields

Main idea:

- Local classifiers (SVM, LR, etc.) classify each pixel individually

- Markov Random Field (MRF) framework classifies all pixels jointly
  - Each pixel is a node in a undirected graph
  - Interactions/dependencies indicated by linked nodes

Why?
Conditional Random Fields

- Discriminative MRF for jointly estimating the label assignments to random variables ($c$), given all data observations ($x$)

```
P(c|x, \theta) = \frac{1}{Z(\theta)} \prod_{i \in V} \Psi_i^{(1)}(c_i, x; \theta) \prod_{(i,j) \in E} \Psi_{i,j}^{(2)}(c_i, c_j, x; \theta)
```

- $\Psi^{(1)}$ models the local score in the label assignment
- $\Psi^{(2)}$ models the score for the pairwise assignment
- $Z$ costs exponentially to explicitly compute ($|L|^|V|$)
Inference

- **Inference** = finding the best joint labeling
  - NP-complete problem in general
- **Two options:** 1) argmax labeling  2) labeling + confidences
- **Argmax labeling with usually Graph-Cut inference**
  - Edge potentials need to satisfy submodularity constraints
    - ✓ Pott’s model satisfies this (more on this later)
    - ✓ High-order potentials possible
  - Recent research with non-submodular potentials
    - ✓ Quadratic Pseudo-Boolean Optimization (QPBO)

- **Labeling + confidences**
  - Estimate the marginal probabilities
  - Usually done with Belief Propagation (or one of its variants)
  - Approximate solution if loops present
    - ✓ Computation exponential in size of smallest clique (tree-width)
    - ✓ Hence, most models are *pairwise* (maximal clique size of 2)
Back to TextonBoost…
Learning a local classifier

- The TextonBoost CRF model
  \[
  \log P(c|x, \theta) = \sum_i \psi_i(c_i, x; \theta) \cdot \text{shape-texture}
  \]

- Shape-texture Potential
  - Function based on new features called *shape filters*

- Trained using boosting to produce multi-class logistic classifier
  - See [torralba-pami-07], Yuandong’s upcoming analysis (Week 11)

- Most important potential in the model
Capturing context

- **Shape-texture Potential** \( \psi_i(c_i, x; \theta_\psi) = \log \tilde{P}_i(c_i | x) \)
  - Main idea: capture the context of relative texton locations for certain classes

- **Step 1: Texton Map generation** (17 filters, K=400)

- **Step 2: Shape Filter**
  - For each texton \( t \)
    - Inputs
      - Texton Map
      - (Rectangle mask \( r \), texton query \( t \))
      - Pixel location \( i \)
    - Output
      - Area in rectangle mask that match \( t \)
  - End result is a texton histogram of area responses

- **How does this capture shape?**
Shape Filters

- Pair: \((i, r, t)\)
  - rectangle \(r\)
  - texton \(t\)

- Feature responses \(v(i, r, t)\)

- Large bounding boxes enable long range interactions

- Integral images
Shape as Texton Layout

\[(r_1, t_1) = \begin{pmatrix} \square \end{pmatrix}, \quad (r_2, t_2) = \begin{pmatrix} \square \end{pmatrix} \]

Slides from Shotton’s ECCV talk

- Texton map
- Ground truth
- Texton map
- Feature response image

\[v(i, r_2, t_2)\]
Shape as Texton Layout

\( (r_1, t_1) = (\text{ground truth}, \text{ground truth}) \)

\( (r_2, t_2) = (\text{summed response images}, \text{summed response images}) \)

\( v(i, r, t_1) + v(i, r, t_2) \)

\( v(i, r_1, t_1) + v(i, r_2, t_2) \)
Learning context

- What do we do with these histograms of shape filters?
  - Boosting over the shape-filter counts of texton \( t \) in rectangle \( r \)
    \[
    \tilde{P}_i(c_i|x) = \frac{\exp(H(c_i))}{\sum_{c_i'} \exp(H(c_i'))}
    \]
  - Ideal algorithm:
    - For each pixel in the Texton Map
      - For each possible rectangle mask orientation
        - For each texton
          » Augment shape-filter to training set
    - Actual algorithm
      - For each pixel in the sub-sampled Texton Map
        - For 10 random rectangle masks
          - For each texton (K=400)
            » Augment shape-filter to training set with 0.3% probability

- 42 hours for 5,000 rounds on 276 images
Initial result

- Cumulative Results

Shape-texture potentials only: 69.6%

- pixel-wise
- segmentation
- accuracies
Refining classification

- Let’s smooth the borders

$$\log P(c|x, \theta) = \sum_i \psi_i(c_i, x; \theta_\psi) + \sum_{(i, j) \in \mathcal{E}} \phi(c_i, c_j, g_{ij}(x); \theta_\phi)$$

- **Edge Potential**
  - Use neighborhood to find and enforce boundaries
  $$\phi(c_i, c_j, g_{ij}(x); \theta_\phi) = -\theta_\phi^T g_{ij}(x) \delta(c_i \neq c_j).$$

- **Main idea:**
  - If class is the **same**, then the pixel **difference** should be **small**
  - If class is **different**, then the pixel **difference** should be **big**

- This is a Pott’s model
  - Efficient inference on CRF with graph-cuts

- $\theta_\phi$ hand tuned with validation data
Cumulative Results

- Shape-texture potentials only: 69.6%
- + edge potentials: 70.3%

pixel-wise segmentation accuracies
Augmenting the model

- Can we improve?
  - Add pixel color information and a prior on class locations in the image

- Final TextonBoost CRF model

\[
\log P(c|x, \theta) = \sum_i \underbrace{\psi_i(c_i, x; \theta_\psi)}_{\text{shape–texture}} + \underbrace{\pi(c_i, x_i; \theta_\pi)}_{\text{color}} + \underbrace{\lambda(c_i, i; \theta_\lambda)}_{\text{location}} + \sum_{(i,j) \in \mathcal{E}} \underbrace{\phi(c_i, c_j, g_{ij}(x); \theta_\phi)}_{\text{edge}} - \log Z(\theta, x)
\]
A prior on class location

- **Location Potential** $\lambda_i(c, i; \theta)$ = $\log \theta(c, i)$
- Create normalized image coordinates for all images
- Lookup the count of queried class at normalize location in training set

$$\theta(c, i) = \left( \frac{N_{c, i} + \alpha \lambda}{N_i + \alpha \lambda} \right)^{w \lambda}$$

Think Naïve Bayes

- $N_{cow, ⭐} = 1$, $N_{⭐} = 3$
Modeling color

- **Color potential**
  \[ \pi \left( c_i, x_i ; \theta_\pi \right) \]

- **Motivation**: hard to learn model for color across many images due to illumination variances
  - Solution: learn potential independently on each image

- **Main idea**:
  - Use the classification from other potentials as a prior
  - Examine the distribution of color with respect to classes
  - **Keep the classification color-consistent**
    - Ex: Pixels associated with cows are black \( \rightarrow \) remaining black pixels in the image should be a cow

- **(Convoluted) Approach**:
  - Gaussian Mixture Model over image CIELab
    - (Distribution of color)
  - Iteratively weight components using EM-like approach
    - Inference to get initial image labeling
    - Weight components so similar color components have same class
    - Repeat
Putting it together

- **Cumulative Results**

  - Shape-texture potentials only: **69.6%**
  - + edge potentials: **70.3%**
  - + colour potentials: **72.0%**
  - + location potentials: **72.2%**

  pixel-wise segmentation accuracies
Learning reminder

- The TextonBoost CRF model

\[
\log P(c | x, \theta) = \sum_i \left( \psi_i(c_i, x; \theta_\psi) + \pi(c_i, x_i; \theta_\pi) + \lambda(c_i, i; \theta_\lambda) \right)
\]

\[
+ \sum_{(i,j) \in E} \phi(c_i, c_j, g_{ij}(x); \theta_\phi) - \log Z(\theta, x)
\]

- 4-neighborhood graph

- Parameters learned independently
Results

- Successes
Results

- Failures
### Results

#### Quantitative results on MSRC-21

<table>
<thead>
<tr>
<th>True class</th>
<th>inferred class</th>
<th>building</th>
<th>grass</th>
<th>tree</th>
<th>cow</th>
<th>sheep</th>
<th>sky</th>
<th>aeroplane</th>
<th>water</th>
<th>face</th>
<th>car</th>
<th>bike</th>
<th>flower</th>
<th>sign</th>
<th>bird</th>
<th>book</th>
<th>chair</th>
<th>road</th>
<th>cat</th>
<th>dog</th>
<th>body</th>
<th>boat</th>
</tr>
</thead>
<tbody>
<tr>
<td>building</td>
<td>61.6 0.3 1.7 9.7 0.3 2.5 0.6 1.3 2.0 2.6 2.1 0.6 0.2 4.8 6.3 0.4 0.5</td>
<td>61.6</td>
<td>0.3</td>
<td>97.6</td>
<td>0.5</td>
<td>0.1</td>
<td>0.1</td>
<td>0.1</td>
<td>0.1</td>
<td>0.1</td>
<td>0.1</td>
<td>0.1</td>
<td>0.1</td>
<td>0.1</td>
<td>0.1</td>
<td>0.1</td>
<td>0.1</td>
<td>0.1</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>grass</td>
<td>0.3 97.6 0.5</td>
<td>1.2</td>
<td>86.3</td>
<td>0.5</td>
<td>2.9</td>
<td>1.4</td>
<td>1.9</td>
<td>0.8</td>
<td>0.1</td>
<td>0.1</td>
<td>0.1</td>
<td>0.1</td>
<td>0.1</td>
<td>0.1</td>
<td>0.1</td>
<td>0.1</td>
<td>0.1</td>
<td>0.1</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>tree</td>
<td>1.2 4.4</td>
<td>86.3 0.5</td>
<td>2.9</td>
<td>1.4</td>
<td>1.9</td>
<td>0.8</td>
<td>0.1</td>
<td>0.1</td>
<td>0.1</td>
<td>0.1</td>
<td>0.1</td>
<td>0.1</td>
<td>0.1</td>
<td>0.1</td>
<td>0.1</td>
<td>0.1</td>
<td>0.1</td>
<td>0.1</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>cow</td>
<td>30.9 0.7 58.3</td>
<td>0.9</td>
<td>0.4</td>
<td>0.4</td>
<td>4.2</td>
<td>4.1</td>
<td>4.1</td>
<td>4.1</td>
<td>4.1</td>
<td>4.1</td>
<td>4.1</td>
<td>4.1</td>
<td>4.1</td>
<td>4.1</td>
<td>4.1</td>
<td>4.1</td>
<td>4.1</td>
<td>4.1</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>sheep</td>
<td>16.5 25.5 4.8</td>
<td>1.9</td>
<td>50.4</td>
<td>0.9</td>
<td>0.4</td>
<td>0.4</td>
<td>4.2</td>
<td>4.1</td>
<td>4.1</td>
<td>4.1</td>
<td>4.1</td>
<td>4.1</td>
<td>4.1</td>
<td>4.1</td>
<td>4.1</td>
<td>4.1</td>
<td>4.1</td>
<td>4.1</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>sky</td>
<td>3.4 0.2 1.1</td>
<td>82.6</td>
<td>7.5</td>
<td>82.6</td>
<td>7.5</td>
<td>82.6</td>
<td>7.5</td>
<td>82.6</td>
<td>7.5</td>
<td>82.6</td>
<td>7.5</td>
<td>82.6</td>
<td>7.5</td>
<td>82.6</td>
<td>7.5</td>
<td>82.6</td>
<td>7.5</td>
<td>82.6</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>aeroplane</td>
<td>21.5 7.2</td>
<td>3.0 59.6</td>
<td>8.5</td>
<td>3.0 59.6</td>
<td>8.5</td>
<td>3.0 59.6</td>
<td>8.5</td>
<td>3.0 59.6</td>
<td>8.5</td>
<td>3.0 59.6</td>
<td>8.5</td>
<td>3.0 59.6</td>
<td>8.5</td>
<td>3.0 59.6</td>
<td>8.5</td>
<td>3.0 59.6</td>
<td>8.5</td>
<td>3.0 59.6</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>water</td>
<td>8.7 7.5 1.5</td>
<td>0.2</td>
<td>4.5</td>
<td>52.9</td>
<td>0.7</td>
<td>4.9</td>
<td>0.2 4.2</td>
<td>14.1 0.4</td>
<td>14.1 0.4</td>
<td>14.1 0.4</td>
<td>14.1 0.4</td>
<td>14.1 0.4</td>
<td>14.1 0.4</td>
<td>14.1 0.4</td>
<td>14.1 0.4</td>
<td>14.1 0.4</td>
<td>14.1 0.4</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>face</td>
<td>4.1 1.1</td>
<td>73.5</td>
<td>7.1</td>
<td>73.5</td>
<td>7.1</td>
<td>73.5</td>
<td>7.1</td>
<td>73.5</td>
<td>7.1</td>
<td>73.5</td>
<td>7.1</td>
<td>73.5</td>
<td>7.1</td>
<td>73.5</td>
<td>7.1</td>
<td>73.5</td>
<td>7.1</td>
<td>73.5</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>car</td>
<td>10.1 1.7</td>
<td>62.5</td>
<td>3.8</td>
<td>5.9</td>
<td>0.2</td>
<td>15.7</td>
<td>0.4 0.2</td>
<td>5.2</td>
<td>0.4 0.2</td>
<td>5.2</td>
<td>0.4 0.2</td>
<td>5.2</td>
<td>0.4 0.2</td>
<td>5.2</td>
<td>0.4 0.2</td>
<td>5.2</td>
<td>0.4 0.2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>bike</td>
<td>9.3 1.3</td>
<td>1.0</td>
<td>74.5</td>
<td>2.5</td>
<td>3.9</td>
<td>5.9</td>
<td>1.6</td>
<td>3.9</td>
<td>5.9</td>
<td>1.6</td>
<td>3.9</td>
<td>5.9</td>
<td>1.6</td>
<td>3.9</td>
<td>5.9</td>
<td>1.6</td>
<td>3.9</td>
<td>5.9</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>flower</td>
<td>6.6 19.3</td>
<td>3.0</td>
<td>62.8</td>
<td>7.3</td>
<td>1.0</td>
<td>1.0</td>
<td>1.0</td>
<td>1.0</td>
<td>1.0</td>
<td>1.0</td>
<td>1.0</td>
<td>1.0</td>
<td>1.0</td>
<td>1.0</td>
<td>1.0</td>
<td>1.0</td>
<td>1.0</td>
<td>1.0</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>sign</td>
<td>31.5 0.2</td>
<td>11.5 2.1</td>
<td>0.5</td>
<td>6.0</td>
<td>1.5</td>
<td>2.5</td>
<td>35.1</td>
<td>3.6</td>
<td>8.3</td>
<td>0.3</td>
<td>0.3</td>
<td>0.3</td>
<td>0.3</td>
<td>0.3</td>
<td>0.3</td>
<td>0.3</td>
<td>0.3</td>
<td>0.3</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>bird</td>
<td>16.9 18.4</td>
<td>9.8</td>
<td>6.3 8.9</td>
<td>1.8</td>
<td>9.4</td>
<td>19.4</td>
<td>4.6</td>
<td>4.5</td>
<td>4.5</td>
<td>4.5</td>
<td>4.5</td>
<td>4.5</td>
<td>4.5</td>
<td>4.5</td>
<td>4.5</td>
<td>4.5</td>
<td>4.5</td>
<td>4.5</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>book</td>
<td>2.6 0.6</td>
<td>0.4</td>
<td>2.0</td>
<td>91.9</td>
<td>15.4</td>
<td>4.5</td>
<td>1.1</td>
<td>1.1</td>
<td>1.1</td>
<td>1.1</td>
<td>1.1</td>
<td>1.1</td>
<td>1.1</td>
<td>1.1</td>
<td>1.1</td>
<td>1.1</td>
<td>1.1</td>
<td>1.1</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>chair</td>
<td>20.6 24.8</td>
<td>9.6</td>
<td>18.2</td>
<td>0.2</td>
<td>3.7</td>
<td>1.3</td>
<td>15.4</td>
<td>4.5</td>
<td>1.1</td>
<td>1.1</td>
<td>1.1</td>
<td>1.1</td>
<td>1.1</td>
<td>1.1</td>
<td>1.1</td>
<td>1.1</td>
<td>1.1</td>
<td>1.1</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>road</td>
<td>5.0 1.1</td>
<td>0.7</td>
<td>3.4</td>
<td>0.3</td>
<td>0.7</td>
<td>0.6</td>
<td>0.1 0.1</td>
<td>1.1</td>
<td>86.0</td>
<td>0.7</td>
<td>86.0</td>
<td>0.7</td>
<td>86.0</td>
<td>0.7</td>
<td>86.0</td>
<td>0.7</td>
<td>86.0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>cat</td>
<td>5.0 1.1</td>
<td>8.9</td>
<td>0.2</td>
<td>2.0</td>
<td>0.6</td>
<td>20.4</td>
<td>53.6</td>
<td>0.2</td>
<td>20.4</td>
<td>53.6</td>
<td>0.2</td>
<td>20.4</td>
<td>53.6</td>
<td>0.2</td>
<td>20.4</td>
<td>53.6</td>
<td>0.2</td>
<td>20.4</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>dog</td>
<td>29.0 2.2</td>
<td>12.9 7.1</td>
<td>9.7</td>
<td>8.1</td>
<td>11.7</td>
<td>19.2</td>
<td>20.4</td>
<td>53.6</td>
<td>0.2</td>
<td>20.4</td>
<td>53.6</td>
<td>0.2</td>
<td>20.4</td>
<td>53.6</td>
<td>0.2</td>
<td>20.4</td>
<td>53.6</td>
<td>0.2</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>body</td>
<td>4.6 2.8</td>
<td>2.0</td>
<td>2.1</td>
<td>1.3</td>
<td>0.2</td>
<td>6.0</td>
<td>1.1</td>
<td>9.9</td>
<td>1.7</td>
<td>4.0</td>
<td>2.1</td>
<td>62.1</td>
<td>62.1</td>
<td>62.1</td>
<td>62.1</td>
<td>62.1</td>
<td>62.1</td>
<td>62.1</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>boat</td>
<td>25.1 11.5</td>
<td>3.8</td>
<td>30.6</td>
<td>2.0</td>
<td>8.6</td>
<td>6.4</td>
<td>5.1</td>
<td>0.3</td>
<td>0.3</td>
<td>0.3</td>
<td>0.3</td>
<td>0.3</td>
<td>0.3</td>
<td>0.3</td>
<td>0.3</td>
<td>0.3</td>
<td>0.3</td>
<td>0.3</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

- Overall pixel-wise accuracy is 72.2%
  - ~15 times better than chance if evenly guessing
  - What if guessing proportional to the distribution of pixels per class?
  - What are the precision rates?
Comparison with previous work

<table>
<thead>
<tr>
<th></th>
<th>Accuracy</th>
<th></th>
<th>Speed (Train/Test)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>Sowerby</td>
<td>Corel</td>
</tr>
<tr>
<td>This paper – Full CRF model</td>
<td>88.6%</td>
<td>74.6%</td>
<td></td>
</tr>
<tr>
<td>This paper – Unary classifier only</td>
<td>85.6%</td>
<td>68.4%</td>
<td></td>
</tr>
<tr>
<td>He et al. – mCRF model [1]</td>
<td>89.5%</td>
<td>80.0%</td>
<td></td>
</tr>
<tr>
<td>He et al. – unary classifier only</td>
<td>82.4%</td>
<td>66.9%</td>
<td></td>
</tr>
</tbody>
</table>

Table 1. Comparison of segmentation/recognition accuracy and efficiency.
Discussion

- What I like about this paper:
  - Classification of many classes
  - Publicly released database
  - Simple approach (minus color potential)

- What I dislike about this paper:
  - Training is ad-hoc
  - Multiple parameters are set by hand
  - Doesn’t improve on referenced work [he-cvpr-04]
Training data split (MSRC-21)

- Distribution of data over training split
  - 7 out of 21 classes > 5% of pixels

<table>
<thead>
<tr>
<th>Category</th>
<th>Percentage</th>
</tr>
</thead>
<tbody>
<tr>
<td>building</td>
<td>10.8</td>
</tr>
<tr>
<td>grass</td>
<td>19.0</td>
</tr>
<tr>
<td>tree</td>
<td>9.1</td>
</tr>
<tr>
<td>cow</td>
<td>3.2</td>
</tr>
<tr>
<td>sheep</td>
<td>2.2</td>
</tr>
<tr>
<td>sky</td>
<td>9.5</td>
</tr>
<tr>
<td>aeroplane</td>
<td>1.6</td>
</tr>
<tr>
<td>water</td>
<td>8.3</td>
</tr>
<tr>
<td>face</td>
<td>1.8</td>
</tr>
<tr>
<td>car</td>
<td>3.3</td>
</tr>
<tr>
<td>bicycle</td>
<td>2.8</td>
</tr>
<tr>
<td>flower</td>
<td>2.6</td>
</tr>
<tr>
<td>sign</td>
<td>1.9</td>
</tr>
<tr>
<td>bird</td>
<td>1.5</td>
</tr>
<tr>
<td>book</td>
<td>5.3</td>
</tr>
<tr>
<td>chair</td>
<td>1.8</td>
</tr>
<tr>
<td>road</td>
<td>9.3</td>
</tr>
<tr>
<td>cat</td>
<td>1.7</td>
</tr>
<tr>
<td>dog</td>
<td>1.5</td>
</tr>
<tr>
<td>body</td>
<td>2.3</td>
</tr>
<tr>
<td>boat</td>
<td>0.7</td>
</tr>
</tbody>
</table>
Testing data split (MSRC-21)

- Distribution of data over testing split
  - 7 out of 21 classes > 5% of pixels
  - Similar proportions to training split

- Guess random, proportionally $\rightarrow$ ~9% chance

- TextonBoost is 8 times better than chance

<table>
<thead>
<tr>
<th>Class</th>
<th>Percentage</th>
</tr>
</thead>
<tbody>
<tr>
<td>building</td>
<td>10.4</td>
</tr>
<tr>
<td>grass</td>
<td>19.8</td>
</tr>
<tr>
<td>tree</td>
<td>8.4</td>
</tr>
<tr>
<td>cow</td>
<td>2.9</td>
</tr>
<tr>
<td>sheep</td>
<td>2.3</td>
</tr>
<tr>
<td>sky</td>
<td>9.8</td>
</tr>
<tr>
<td>aeroplane</td>
<td>1.3</td>
</tr>
<tr>
<td>water</td>
<td>7.8</td>
</tr>
<tr>
<td>face</td>
<td>1.8</td>
</tr>
<tr>
<td>car</td>
<td>3.4</td>
</tr>
<tr>
<td>bicycle</td>
<td>2.5</td>
</tr>
<tr>
<td>flower</td>
<td>3.5</td>
</tr>
<tr>
<td>sign</td>
<td>3.0</td>
</tr>
<tr>
<td>bird</td>
<td>1.3</td>
</tr>
<tr>
<td>book</td>
<td>5.3</td>
</tr>
<tr>
<td>chair</td>
<td>2.0</td>
</tr>
<tr>
<td>road</td>
<td>8.1</td>
</tr>
<tr>
<td>cat</td>
<td>1.4</td>
</tr>
<tr>
<td>dog</td>
<td>2.1</td>
</tr>
<tr>
<td>body</td>
<td>1.9</td>
</tr>
<tr>
<td>boat</td>
<td>1.0</td>
</tr>
</tbody>
</table>
Overview

- **Goal:** (same as before)

- **Motivation:**
  - 1) Visual words approach is slow
    - ✓ Compute feature descriptors
    - ✓ Cluster
    - ✓ Nearest-neighbor assignment
  - 2) CRF is even slower
    - ✓ Inference always a bottle-neck

- **Approach:** operate on pixel values
  - Simple & efficient

- **Result:** works well and efficiently
Overview

- Contributions
  - Semantic Texton Forests: local classification with hierarchical information
  - The Bag of Semantic Textons Model
  - Image-level prior to improve semantic segmentation

- Quick decision tree review…
Decision Trees

Who here has a car?

Structure of a decision tree

- Internal nodes correspond to attributes (features)
- Leaves correspond to classification outcome
- Edges denote assignment

Advantages?

Drawbacks?
Encoding decisions

- **Randomized Decision Forests**
  - Input: “features” describing pixel
  - Output: Predicted class distribution

- **Approach**
  - Each node $n$ in the decision tree contains an empirical class distribution $P(c|n)$
  - **Important**: Learn decision trees such that similar “features” should end up at the same leaf nodes

- ![Diagram of decision trees](image)
  - The leaves $L = \{l_i\}$ of a tree contain most discriminative information
    - Classify by averaging
    \[
    P(c|L) = \frac{1}{T} \sum_{t=1}^{T} P(c|l_t)
    \]

- Another histogram of texton-like per pixel!
Features?

- Think of the **simplest** features you can do.
- Center a \( d \)-by-\( d \) patch around a pixel (5x5)
- Possible features:
  - Feature #1: its value in a color channel (CIELab)
  - Feature #2: the sum of two points in the patch
  - Feature #3: the difference of two points in the patch
  - Feature #4: the absolute difference of two points in the patch
- Feature invariance accounted for by rotating, scaling, flipping, affine-ing training data

- Random Decision Tree training:
  - Take random subset of training data
  - Generate random features \( f \) from above
  - Generate random threshold \( t \)
  - Split data into left \( I_l \) and right \( I_r \) subsets according to
    \[
    I_l = \{ i \in I_n \mid f(v_i) < t \} \\
    I_r = I_n \setminus I_l.
    \]
  - Repeat for each side
- Does this actually work?
Filters found

- Yes

- Each **patch** represents one leaf node. It is the summation of all the patches from the training data that fell into that leaf.
- Learns colors, orientations, edges, blobs
Simple model results

- Semantic Texton Forests are better than chance (~5%)

<table>
<thead>
<tr>
<th></th>
<th>Global</th>
<th>Average</th>
</tr>
</thead>
<tbody>
<tr>
<td>supervised</td>
<td>49.7%</td>
<td>34.5%</td>
</tr>
<tr>
<td>weakly supervised</td>
<td>14.8%</td>
<td>24.1%</td>
</tr>
</tbody>
</table>

- MSRC-21 dataset

- Supervised = 1 label per pixel
  - Increase one bin in the histogram at a time

- Weakly-supervised = all labels in image per pixel
  - Increase multiple bins in the histogram at a time
Adding tricks to the model

- More extensions with this model: **Bags of Semantic Textons**

- How can we get a prior estimate for what is in region $r$?
  - 2 Options:
    - 1) Average leaf histograms in region $r$ together $P(c|r)$
      - Good for segmentation priors
    - 2) Create hierarchy histogram of node counts $H_r(n)$ visited in the tree for each classified pixel in region $r$
      - Want testing and training **decision paths** to match
Histogram-based Classification

- Main idea:
  - Have 2 vectors as features
    - (training-tree’s histograms, testing-tree’s histograms)
  - Want to measure similarity to do classification

- Proposed approach: Kernalized SVM
  - Kernel = Pyramid Match Kernel (PMK)
  - Computes a histogram distance, using hierarchy information
  - Train 1-vs-all classifiers

- Review on Pyramid Match Kernel...
Example pyramid match

Level 0

$H_0(X)$

$H_0(Y)$

$I_0 = 2$

$N_0 = 2$

$w_0 = 1$
Example pyramid match

Level 1

\[ N_1 = 4 - 2 = 2 \]
\[ \omega_1 = \frac{1}{2} \]

\[ H_1(X) \]
\[ H_1(Y) \]

\[ I_1 = 4 \]
Example pyramid match

Level 2

\[N_2 = 5 - 4 = 1\]
\[w_2 = \frac{1}{4}\]

\[H_2(X)\]
\[H_2(Y)\]

\[I_2 = 5\]
Scene Categorization

- The whole image is one region
  - Using histogram matching approach
  - End result is an Image-level Prior

- Comparison with other similarity metric (radial basis function, RBF)
  - Unfair? RBF uses only leaf-level counts, PMK uses entire histogram

- Results

<table>
<thead>
<tr>
<th></th>
<th>Global kernel $K$</th>
<th>Per-category kernel $K_c$</th>
</tr>
</thead>
<tbody>
<tr>
<td>RBF</td>
<td>49.9</td>
<td>52.5</td>
</tr>
<tr>
<td>PMK</td>
<td>76.3</td>
<td>78.3</td>
</tr>
</tbody>
</table>

Table 2. Image categorization results. (Mean AP).

- $K_c = \text{trick to account for unbalanced classes}$
- Note Mean Average Precision reported here, but not elsewhere

- Number of trees has diminishing returns

Figure 5. Categorization accuracy vs number of STF trees.
Improving Semantic Segmentation

- Use idea of shape-filters to improve classification
- Main idea: After initial STF classification, learn how a pixel’s class interacts with neighboring regions’ classes

- Approach: Learn a second random decision forest (segmentation forest)
  - Use different weak features:
    - Histogram count at some level $H_{r+i}(?)$
    - Region prior probability of some class $P(\cdot \mid r+i)$

- Difference with shape filters:
  - Shape-filters learn: cow is adjacent to green-like texture
  - Segmentation forest learn: cow is adjacent to grass

- Trick: multiply with image-level prior for best results
  - Convert SVM decision to probability
Computation time

- **Fast**
  - STF feature extraction = 275 ms
  - Image categorization = 190 ms
  - Segmentation forest = 140 ms
  - Total ~ 605 ms

- **TextonBoost = 6000 ms**
### MSRC-21 Results

<table>
<thead>
<tr>
<th></th>
<th>building</th>
<th>grass</th>
<th>tree</th>
<th>cow</th>
<th>sheep</th>
<th>sky</th>
<th>airplane</th>
<th>water</th>
<th>face</th>
<th>car</th>
<th>bicycle</th>
<th>flower</th>
<th>sign</th>
<th>bird</th>
<th>book</th>
<th>chair</th>
<th>road</th>
<th>cat</th>
<th>dog</th>
<th>body</th>
<th>boat</th>
</tr>
</thead>
<tbody>
<tr>
<td>[27]</td>
<td>62</td>
<td>98</td>
<td>86</td>
<td>58</td>
<td>50</td>
<td>83</td>
<td>60</td>
<td>53</td>
<td>74</td>
<td>63</td>
<td>75</td>
<td>63</td>
<td>35</td>
<td>19</td>
<td>92</td>
<td>15</td>
<td>86</td>
<td>54</td>
<td>19</td>
<td>62</td>
<td>7</td>
</tr>
<tr>
<td>[32]</td>
<td>52</td>
<td>87</td>
<td>68</td>
<td>73</td>
<td>84</td>
<td>94</td>
<td>88</td>
<td>73</td>
<td>70</td>
<td>68</td>
<td>74</td>
<td>89</td>
<td>33</td>
<td>19</td>
<td>78</td>
<td>34</td>
<td>89</td>
<td>46</td>
<td>49</td>
<td>54</td>
<td>31</td>
</tr>
<tr>
<td>Ours</td>
<td>41</td>
<td>84</td>
<td>75</td>
<td>89</td>
<td>93</td>
<td>79</td>
<td>86</td>
<td>47</td>
<td>87</td>
<td>65</td>
<td>72</td>
<td>61</td>
<td>36</td>
<td>26</td>
<td>91</td>
<td>50</td>
<td>70</td>
<td>72</td>
<td>31</td>
<td>61</td>
<td>14</td>
</tr>
<tr>
<td>Ours + ILP</td>
<td>49</td>
<td>88</td>
<td>79</td>
<td>97</td>
<td>97</td>
<td>78</td>
<td>82</td>
<td>54</td>
<td>87</td>
<td>74</td>
<td>72</td>
<td>74</td>
<td>36</td>
<td>24</td>
<td>93</td>
<td>51</td>
<td>78</td>
<td>75</td>
<td>35</td>
<td>66</td>
<td>18</td>
</tr>
</tbody>
</table>

| Global | Average |       |      |     |       |     |          |       |      |     |         |        |      |      |      |      |      |     |     |      |     |    |
|--------|---------|-------|------|-----|-------|-----|----------|-------|------|-----|---------|--------|------|------|------|------|------|-----|-----|------|-----|    |
| 71     | 58      |       |      |     |       |     |          |       |      |     |         |        |      |      |      |      |      |     |     |      |     |    |
| -      | 64      |       |      |     |       |     |          |       |      |     |         |        |      |      |      |      |      |     |     |      |     |    |
| 68     | 63      |       |      |     |       |     |          |       |      |     |         |        |      |      |      |      |      |     |     |      |     |    |
| 72     | 67      |       |      |     |       |     |          |       |      |     |         |        |      |      |      |      |      |     |     |      |     |    |
## VOC 2007 Segmentation

<table>
<thead>
<tr>
<th></th>
<th>background</th>
<th>aeroplane</th>
<th>bicycle</th>
<th>bird</th>
<th>boat</th>
<th>bottle</th>
<th>bus</th>
<th>car</th>
<th>cat</th>
<th>chair</th>
<th>cow</th>
<th>table</th>
<th>dog</th>
<th>horse</th>
<th>motorbike</th>
<th>person</th>
<th>plant</th>
<th>sheep</th>
<th>sofa</th>
<th>train</th>
<th>tv / monitor</th>
<th>Average</th>
</tr>
</thead>
<tbody>
<tr>
<td>Brookes</td>
<td>78</td>
<td>6</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>9</td>
<td>5</td>
<td>10</td>
<td>1</td>
<td>2</td>
<td>11</td>
<td>0</td>
<td>6</td>
<td>29</td>
<td>2</td>
<td>2</td>
<td>0</td>
<td>11</td>
<td>1</td>
<td>9</td>
<td></td>
</tr>
<tr>
<td>Ours</td>
<td>33</td>
<td>46</td>
<td>5</td>
<td>14</td>
<td>11</td>
<td>14</td>
<td>34</td>
<td>8</td>
<td>6</td>
<td>3</td>
<td>10</td>
<td>39</td>
<td>40</td>
<td>28</td>
<td>23</td>
<td>32</td>
<td>19</td>
<td>19</td>
<td>8</td>
<td>24</td>
<td>9</td>
<td>20</td>
</tr>
<tr>
<td>Ours + ILP</td>
<td>20</td>
<td>66</td>
<td>6</td>
<td>15</td>
<td>6</td>
<td>15</td>
<td>32</td>
<td>19</td>
<td>7</td>
<td>7</td>
<td>13</td>
<td>44</td>
<td>31</td>
<td>44</td>
<td>27</td>
<td>39</td>
<td>35</td>
<td>12</td>
<td>7</td>
<td>39</td>
<td>23</td>
<td>24</td>
</tr>
<tr>
<td>TKK</td>
<td>23</td>
<td>19</td>
<td>21</td>
<td>5</td>
<td>16</td>
<td>3</td>
<td>1</td>
<td>78</td>
<td>1</td>
<td>3</td>
<td>1</td>
<td>23</td>
<td>69</td>
<td>44</td>
<td>42</td>
<td>0</td>
<td>65</td>
<td>30</td>
<td>35</td>
<td>89</td>
<td>71</td>
<td>30</td>
</tr>
<tr>
<td>Ours + DLP</td>
<td>22</td>
<td>77</td>
<td>45</td>
<td>45</td>
<td>19</td>
<td>14</td>
<td>45</td>
<td>48</td>
<td>29</td>
<td>26</td>
<td>20</td>
<td>59</td>
<td>45</td>
<td>54</td>
<td>63</td>
<td>37</td>
<td>40</td>
<td>42</td>
<td>10</td>
<td>68</td>
<td>72</td>
<td>42</td>
</tr>
</tbody>
</table>

The table above compares the segmentation performance of different methods on the VOC 2007 dataset. The percentages represent the accuracy of each method for various categories, such as background, aeroplane, bicycle, and so on. The last row indicates the average accuracy across all categories.
Discussion

- What I like about this paper:
  - Simple concept
  - Good result
  - Works fast (testing & training)

- What I dislike about this paper:
  - More difficult to understand
  - Low-resolution classification
    - Segmentation forest operates at patches
  - Test-time inference is dependent on amount of training
    - Must iterate through all trees in the forest at test time
  - Many “Implementation Details” scattered through the paper.
    - What is the trick to get it to work?
  - How dependent is the performance on decision tree parameters?