Image Classification using Random Forests and Ferns

Anna Bosch, Andrew Zisserman and Xavier Munoz
ICCV 2007

Yair Movshovitz
15.6.2009

http://users.soe.ucsc.edu/~taoswap/GroupMeeting/
Outline

- Randomized methods
  - Randomized tree
  - Randomized forests
  - Randomized Ferns

- “Image Classification using Random Forests and Ferns” Anna Bosch, Andrew Zisserman, Xavier Munoz. *ICCV 2007*
Randomized methods

Trees
ferns
Randomized Tree

- Each internal node contains a simple test to split the space,
- Tree leaves store the posterior probabilities,
- Randomized means the intermediate node tests are randomly chosen,
- Easily handle multi–class problems and are robust and fast.
Building the tree

- Greedy algorithm to select the partition that gives the best separation of the training samples
  - Maximize the information gain by partition the set of $S$ samples into subsets $S_i$ according to a given test:
    \[ \Delta E = - \sum_i \frac{|S_i|}{|S|} E(S_i) \]
    \[ E(s) = - \sum_{j=1}^{N} p_j \log_2(p_j) \]
  - Select most relevant features
Building the tree

- Random selection

\[ T = \begin{cases} 
  \text{if } n^T x + b \leq 0 & \text{go to the right child} \\
  \text{otherwise} & \text{go to the left child} 
\end{cases} \]

- Choose random number of elements and set them to \([-1, 1]\), the rest are 0
Random Forests

A random forest is a classifier consisting of a collection of tree structured classifiers \( \{h(x, \Theta_k), k=1, \ldots\} \) where the \( \{\Theta_k\} \) are independent identically distributed random vectors and each tree casts a unit vote for the most popular class at input \( x \).
Building a Random Forest

- Subsample the data so that each tree is grown using a different subset
- Growing each tree
  - Random
  - Greedy – entropy optimization
- Learning the posteriors – count
Classifcation
Features of Random Forests

- High accuracy – compare favorably to adaboost
- Capable of handling large datasets
- Capable of handling a very large number of input variables.
- Effectively handles missing values
A New Classifier: Ferns
The tests compare the intensities of two pixels around the keypoint:

\[ f_i = \begin{cases} 
  1 & \text{if } I(m_{i,1}) \leq I(m_{i,2}) \\
  0 & \text{otherwise}
\end{cases} \]

Invariant to light change by any raising function.

Posterior probabilities:

\[ P(f_1, f_2, \cdots f_n \mid C = c_j) \]
Ferns: Training
Ferns: Training
Ferns: Training Results
Ferns: Recognition
Justification

We are looking for

\[ P(C = c_i \mid f_1, f_2, \cdots f_n, f_{n+1}, \cdots f_N) \]

proportional to

\[ P(f_1, f_2, \cdots f_n, f_{n+1}, \cdots f_N \mid C = c_i) \]

but complete representation of the joint distribution infeasible.

Naive Bayesian ignores the correlation:

\[ \approx \prod_j P(f_j \mid C = c_i) \]

Compromise:

\[ \approx P(f_1, f_2, \cdots, f_n \mid C = c_i) \times P(f_{n+1}, \cdots \mid C = c_i) \]

i.e. probabilities stored by the leaves.
Pros & Cons

Fast, easy to implement; No parameters to tune.

Takes a lot of memory:
\[ 2^{\text{depth}} \times \text{number of structures} \times \text{number of classes} \]
floating point values to store.
Ferns Implementation

1: for(int i = 0; i < H; i++) P[i] = 0.;
2: for(int k = 0; k < M; k++) {
3:   int index = 0, * d = D + k * 2 * S;
4:   for(int j = 0; j < S; j++) {
5:     index <<= 1;
6:     if (*(K + d[0]) < *(K + d[1]))
7:       index++;
8:     d += 2;
9:   }
10:  p = PF + k * shift2 + index * shift1;
11: for(int i = 0; i < H; i++) P[i] += p[i];
}

Very simple to implement;
Very fast.
Building the ferns takes no time (except for the posterior probabilities estimation);
Allows incremental learning;
Simplifies the classifier structure.
### Comparison with Randomized Trees

<table>
<thead>
<tr>
<th></th>
<th>Trees</th>
<th>Fern</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Structure</strong></td>
<td>Hierarchical</td>
<td>Flat</td>
</tr>
<tr>
<td><strong>Posterior</strong></td>
<td>Additive</td>
<td>Multiplicative</td>
</tr>
</tbody>
</table>

Similar performance when training set size is small

*Insight: empirical prob. are only rude estimates of true ones, no much difference*

Ferns outperform Trees when the training set is sufficiently large

*Insight: multiplicative nature -> one small leads to all small*
Image Classification using Random Forests and Ferns

Anna Bosch, Andrew Zisserman and Xavier Munoz
ICCV 2007
Contributions

- Shape and appearance representation that support spatial pyramid matching over a region of interest
- Automatic selection of the regions of interest in training
- The use of random forests and ferns as a multi-way classifier
Features

- **Appearance**
  - Regularly sampled points (10 pixels)
  - SIFT computed at 4 scales
  - \( \text{App}_{\text{Gray}} I = 0.3R + 0.59G + 0.11B \)  \( \text{App}_{\text{Color}} \) – HSV space
  - Bag of visual words – 300 visual words

- **Shape**
  - HOG
  - \( \text{Shape}_{180} = [0, 180] \) and \( \text{Shape}_{360} = [0, 360] \)
  - Discretized into 20 and 40 bins
Spatial pyramid representation

- Extension of a bag of features
- Locally orderless representation at several levels of resolution
- Based on *pyramid match kernels* Grauman & Darrell (2005)
Features

![Diagram showing various features and their corresponding histograms.](image-url)
Image matching

\[ K(D_I, D_J) = \exp\left\{ \frac{1}{\beta} \sum_{l \in L} \alpha_l d_l(D_I, D_J) \right\} \]

- Using both shape and appearance.
Selecting ROI in training images

- Intuition: object regions will tend to be consistent over subsets of the training images (of high visual similarity)
Selecting ROI in training images

\[ L_i = \max_{\{r_j\}} \sum_{j=1}^{s} K(D(r_i), D(r_j)) \]

- Too hard to optimize
- Approximation:
  - Fix \( r_j \) for all the other images, search over all regions \( r_i \)

Descriptor for each region (PHOW, PHOG)

Regions
Selecting ROI in training images

- Optimization process is done by testing groups \( s \) of size 1–4.
- Search is done using rectangles.
- Init size – entire image.
- Scale down by 0.1.
Classifiers

- Randomized Trees
  - Select $n_f$ number of features
  - Random test (RT)
  - Entropy optimization (EO)

- Randomized Ferns
  - Votes are averaged not multiplied

$$T = \begin{cases} 
  \text{if } n^T x + b \leq 0 & \text{go to the right child} \\
  \text{otherwise} & \text{go to the left child}
\end{cases}$$

Each test uses either shape or appearance
Results

Caltech 101
Caltech 256
### ROI detection

<table>
<thead>
<tr>
<th>Optimization</th>
<th>s = 1</th>
<th>s = 2</th>
<th>s = 3</th>
<th>s = 4</th>
</tr>
</thead>
<tbody>
<tr>
<td>no optimization</td>
<td>38.7</td>
<td>42.9</td>
<td>43.5</td>
<td>42.8</td>
</tr>
<tr>
<td></td>
<td>±1.3</td>
<td>±1.0</td>
<td>±1.0</td>
<td>±1.0</td>
</tr>
</tbody>
</table>

- Performance done using 100 randomized trees with depth=20. Node tests are using E.O.
Random Forests and Ferns

<table>
<thead>
<tr>
<th></th>
<th>Randomized Forests</th>
<th>Randomized Ferns</th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>$Shp_{180}$</td>
<td>$Shp_{360}$</td>
<td>$App_C$</td>
<td>$App_G$</td>
<td>$All$</td>
</tr>
<tr>
<td>RT</td>
<td>38.5</td>
<td>39.3</td>
<td>35.2</td>
<td>39.3</td>
<td>41.9</td>
</tr>
<tr>
<td></td>
<td>±0.8</td>
<td>±0.9</td>
<td>±0.9</td>
<td>±1.0</td>
<td>±1.2</td>
</tr>
<tr>
<td>EO</td>
<td>39.2</td>
<td>40.5</td>
<td>36.5</td>
<td>40.7</td>
<td>43.5</td>
</tr>
<tr>
<td></td>
<td>±0.8</td>
<td>±0.9</td>
<td>±0.8</td>
<td>±0.9</td>
<td>±1.1</td>
</tr>
</tbody>
</table>

7 hrs
20 hrs
1.5 hrs
4 hrs
# trees and their depth

a) Entropy Optimization V.S. Random Tests
b) The effect of depth of trees.
Number of features

![Graph showing performance against number of features with two strategies: random split and split with entropy. The graph indicates that the split with entropy generally leads to higher performance.]
Comparisons

<table>
<thead>
<tr>
<th>Dataset</th>
<th>C-101</th>
<th>C-256</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>15</td>
<td>30</td>
</tr>
<tr>
<td>$N_{Train}$</td>
<td></td>
<td></td>
</tr>
<tr>
<td>M-SVM</td>
<td>—</td>
<td>81.3</td>
</tr>
<tr>
<td></td>
<td>—</td>
<td>±0.8</td>
</tr>
<tr>
<td>R. Forests</td>
<td>70.4</td>
<td>80.0</td>
</tr>
<tr>
<td></td>
<td>±0.7</td>
<td>±0.6</td>
</tr>
<tr>
<td>R. Ferns</td>
<td>70.0</td>
<td>79.2</td>
</tr>
<tr>
<td></td>
<td>±0.7</td>
<td>±0.6</td>
</tr>
<tr>
<td>[5]</td>
<td>67.4</td>
<td>77.8</td>
</tr>
<tr>
<td>[13]</td>
<td>59.0</td>
<td>67.6</td>
</tr>
<tr>
<td>[26]</td>
<td>59.0</td>
<td>66.2</td>
</tr>
<tr>
<td>[11]</td>
<td>60.3</td>
<td>66.0</td>
</tr>
<tr>
<td>[16]</td>
<td>56.4</td>
<td>64.6</td>
</tr>
<tr>
<td>[18]</td>
<td>59.9</td>
<td>—</td>
</tr>
</tbody>
</table>
Number of training images
Thank You... Questions?