

Chapter 7

Semantic Texton Forests

Matthew Johnson and Jamie Shotton

Abstract. The *semantic texton forest* is an efficient and powerful low-level feature which can be effectively employed in the semantic segmentation of images. As ensembles of decision trees that act directly on image pixels, semantic texton forests do not need the expensive computation of filter-bank responses or local descriptors. They are extremely fast to both train and test, especially compared with k -means clustering and nearest-neighbor assignment of feature descriptors. The nodes in the trees provide (i) an implicit hierarchical clustering into semantic textons, and (ii) an explicit local classification estimate. The *bag of semantic textons* combines a histogram of semantic textons over an image region with a region prior category distribution. The bag of semantic textons can be used by an SVM classifier to infer an image-level prior over categories, allowing the segmentation to emphasize those categories that the SVM believes to be present. We will examine the segmentation performance of semantic texton forests on two datasets including the VOC 2007 segmentation challenge.

7.1 Introduction

In this chapter we examine *semantic texton forests*, and evaluate their use for image categorization and semantic segmentation. Semantic texton forests (STFs) demonstrate that one can build powerful texton codebooks without computing expensive filter-banks or descriptors, and without performing costly k -means clustering and nearest-neighbor assignment. They are randomized decision forests that use only simple pixel comparisons on local image patches, performing both an implicit

Matthew Johnson
Nokia, San Francisco, USA
e-mail: matthew.3.johnson@nokia.com

Jamie Shotton
Microsoft Research, Cambridge, UK
e-mail: jamiesho@microsoft.com

hierarchical clustering into semantic textons and an explicit local classification of the patch category. STFs provide advantages over other algorithms in both quantitative performance and execution speed.

We will look at two applications of STFs: image categorization (inferring the object categories present in an image) and semantic segmentation (dividing the image into coherent regions and simultaneously categorizing each region). The tool that will be used for both of these is the *bag of semantic textons*. This is computed over a given image region, and extends the bag of words model [6] by combining a hierarchical histogram of the semantic textons with a prior category distribution. By considering the image as a whole, a highly discriminative descriptor for categorization can be obtained. For segmentation, a bag of semantic textons can be computed for many local rectangular regions and then a second randomized decision forest can be built which achieves efficient and accurate segmentation by drawing on appearance and semantic context.

The segmentation algorithm depends on image information that even with semi-local context can often be ambiguous. The global statistics of the image, however, can be more discriminative and may be sufficient to accurately estimate the image categorization. It is therefore useful to use categorization as an *image-level prior* to improve segmentation by emphasizing the categories most likely to be present.

7.2 Related Work

Textons [17, 35] and visual words [32] have proven powerful discrete image representations for categorization and segmentation [6, 30, 39, 40]. We will treat the terms texton and visual word synonymously. Filter-bank responses (derivatives of Gaussians, wavelets, etc.) or invariant descriptors (e.g., SIFT [16]) are computed across a training set, either at sparse interest points [19] or more densely; recent results in [22] suggest that densely sampling visual words improves categorization performance. The collection of descriptors are then clustered to produce a codebook of visual words, typically with the simple but effective k -means, followed by nearest-neighbor assignment. Unfortunately, this three-stage process is extremely slow and often the most time consuming part of the whole algorithm, even with optimizations such as k -d trees, the triangle inequality [7], or hierarchical clusters [21, 29].

The recent work of Moosmann *et al* [20] proposed a more efficient alternative, in which training examples are recursively divided using a randomized decision forest [1, 10] and where the splits in the decision trees are comparisons of a descriptor dimension to a threshold. Semantic texton forests extend [20] in three ways: (i) the model is learned *directly* from the image pixels, bypassing the expensive step of computing image descriptors; (ii) while [20] use the learned decision forest only for clustering, here it is used as a classifier, which enables the algorithm to use semantic context for image segmentation; and (iii) in addition to the leaf nodes used in [20], split nodes as hierarchical clusters are included. A related method, the pyramid match kernel [11], exploits a hierarchy in descriptor space, though it

requires the computation of feature descriptors and is only applicable to kernel based classifiers.

The pixel-based features used are similar to those in [15], but the forests are trained to recognize object categories, not match particular feature points.

Other work has also looked at alternatives to k -means. Recent work [34] quantizes feature space into a hyper-grid, but requires descriptor computation and can result in very large visual word codebooks. Winder & Brown [38] learned the parameters of generic image descriptors for 3D matching, though did not address visual word clustering. Jurie & Triggs [13] proposed building codebooks using mean shift, but did not incorporate semantic information in the codebook generation.

7.3 Randomized Decision Forests

The randomized decision forest is a fast and accurate classifier [1, 10] that is an ensemble of decision trees. Decision trees are a construct used extensively in data mining [5] and machine learning [4], and consist of a hierarchy of questions, as illustrated in Figure 7.1. A tree is traversed, starting at the root, by repeatedly asking questions and branching to the relevant child node until a leaf node is reached. At the leaf, the stored decision is output. In this chapter, decision trees are used to categorize individual image pixels, and so the questions, or “split tests”, at each node are based on image information. The specific tests used in this work are shown in Figure 7.2.

A randomized decision forest combines the output of many different decision trees, each of which has a different structure and split tests. The term “randomized” refers to the training algorithm in two ways. Firstly, each tree is trained on a random subset of the data following the method outlined in Section 7.3.1.1. Secondly, when building the tree, several candidate split tests are chosen at random from a large

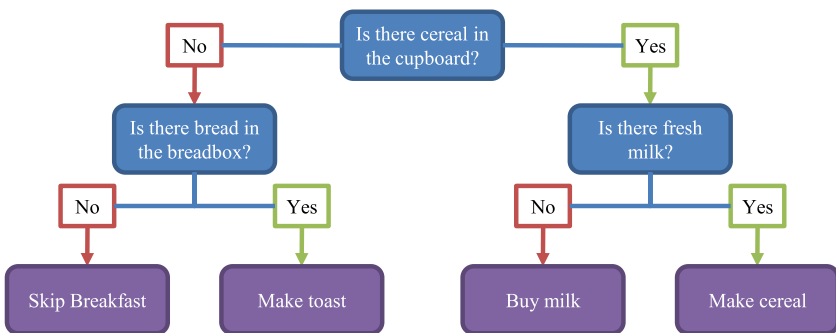


Fig. 7.1 Example Decision Tree. This is an example decision tree for determining what to do about breakfast. At each node a simple split test is performed, and the result of that test is used to determine which child to choose. This process is repeated until a leaf node is reached, with each leaf encoding a particular decision to be made that is based upon all of the tests performed to reach that node.

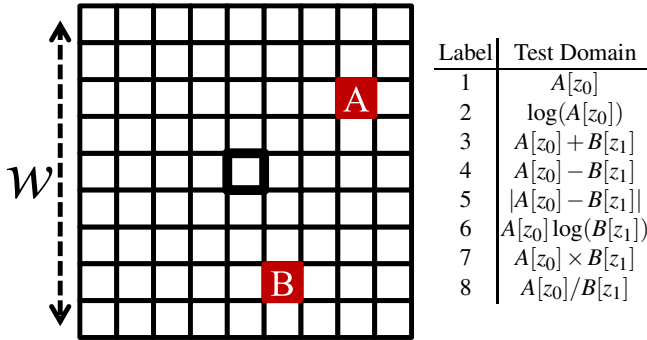


Fig. 7.2 Pixel Comparison Split Tests. The split tests in a semantic texton forest consist of pixel combinations within a square neighborhood centered on the pixel to be categorized of size $w \times w$. z_0 and z_1 are channels in the image, e.g., R, G and B in RGB images. It is not necessary that $z_0 = z_1$.

pool of potential features, and the test that optimally splits the data (under some optimization criterion) is taken. These two forms of randomization help generalization by ensuring that no two trees in the forest can overfit to the whole training set.

Let us formalize notation. For the forests described in this chapter, the goal is to determine the category c of a pixel p , given the context around that pixel. We assume a labeled training set, such as that in Figure 7.4. Each forest contains trees with nodes n , and leaf nodes l . Associated with each node is a learned category distribution $P(c|n)$. An example semantic texton tree can be seen in Figure 7.3, in which a tree has been trained on sheep and grass images and can effectively segment an image according to these two semantic categories.

When a new pixel is to be classified, the whole forest achieves an accurate and robust classification by averaging the class distributions over the leaf nodes $L(p) = (l_1, \dots, l_T)$ reached by the pixel p for all T trees:

$$P(c|L(p)) = \sum_{t=1}^T P(c|l_t)P(t). \quad (7.1)$$

An example of the overall structure of the forest can be seen in Figure 7.5.

Existing work has shown the power of decision forests as either classifiers [3, 15, 18] or a fast means of clustering descriptors [20]. In this chapter we will examine an extended model in which the forest is used both for classification and for a hierarchical clustering.

7.3.1 Training the Forest

Each tree in the forest is build separately on a subset of the training images. We describe below how each tree is built greedily, and the parameter settings that will

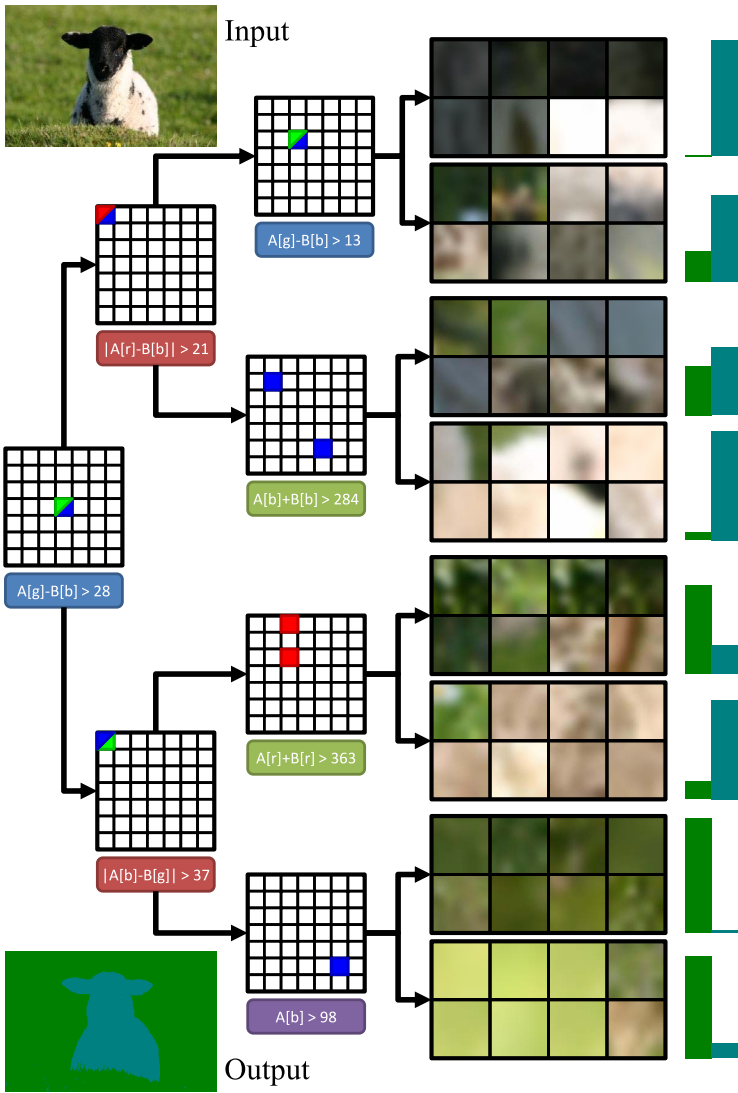


Fig. 7.3 Sample Semantic Texton Tree. This is an actual semantic texton tree, trained on 23 images of grass and sheep as described in Section 7.3.1.1. The split tests are shown at each split node as the image patch ($w = 7$). The two triangles indicate the offset pixel location (relative to the center of the grid) and their colors indicate the image color channel used for this split test. The leaf nodes are represented by 8 patches sampled from the training pixels which reached those nodes and the distribution $P(x|c)$ for that leaf node where green represents grass and blue represents sheep. The input image is an unseen test image, with the resulting semantic segmentation shown below.



Fig. 7.4 Training Data. A tree is trained on ground-truth labeled images like these above from the MSRC dataset [30], in which each pixel is labelled with an object category (indicated by colors).

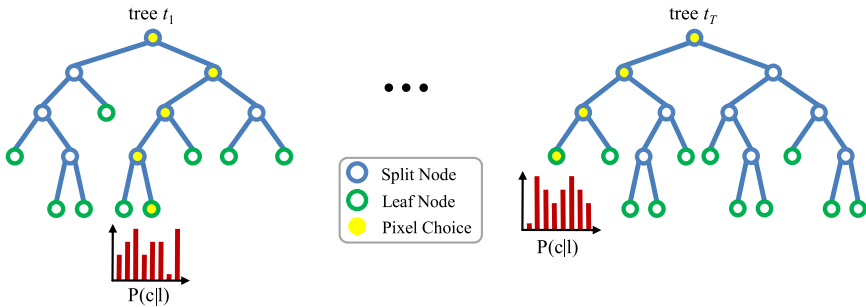


Fig. 7.5 Semantic Texton Forest Structure. The forest is made up of T binary trees. Each split node n in a tree (blue circles) has a test associated with it, and based upon the result of that test one or the other child is chosen. When a leaf node l in a tree t is reached (green circles), the $P(c|l_t)$ distribution for that leaf is used as a soft category decision for the test pixel. In this figure, a sample decision path for each tree is denoted by a series of yellow circles. The final decision is a combination of $P(c|l_t)$ for all $t \in T$.

affect this. We also discuss how to learn invariances to rotation, scale and other fundamental image transformations.

7.3.1.1 Building a Tree

The training data consists of a set P of pixels sampled from training images at every 4th pixel, and ignoring pixels marked as background. This sub-sampling decreases the time required for tree construction. To ensure good estimates of the tree class distributions, *all* pixels are used later to “fill” the tree after construction as described below in Section 7.3.1.3.

Each tree is constructed by recursively partitioning P into two subsets P_{left} and P_{right} based upon a split test. P_{left} is used to create the left subtree and P_{right} is used for the right, repeating the process until a stopping condition is met. The split test used to partition P is chosen in the same manner as [15]: by searching over a set of

possible tests and choosing that which maximizes the expected gain in information about the node categories. The information gain is calculated as

$$\Delta E = -\frac{|P_{left}|}{|P|}E(P_{left}) - \frac{|P_{right}|}{|P|}E(P_{right}), \quad (7.2)$$

where $E(I)$ is the Shannon entropy of the classes in the set of examples P .

7.3.1.2 Parameters

Training a randomized decision forest involves several parameters, including:

Type of Split Tests. The types of split tests used can play a significant role in tree training and performance, with different tests acting in a complimentary manner.

Number of Trees. The number of trees in the forest is a tradeoff between accuracy and speed.

Information Channels. The number and type of information channels in the image and how they are processed can have a large impact on which tests should be chosen and on model generalization. In the case of color images, this takes the form of the encoding method chosen for the color at each pixel.

Maximum Tree Depth. Deeper trees are more powerful classifiers, but more prone to overfitting.

Value of Window Size. The size of the window around each pixel w can effect whether the tree learns local image characteristics or contextual information. Larger windows provide more discriminative features but ones that are less likely to generalize.

The choice of these parameters depends heavily on the nature of the dataset, and should be optimized against a validation set. Smaller datasets will tend to need many shallower trees, whereas larger datasets make generalization easier and so fewer deeper trees may work well. We discuss below the results of several experiments designed to discover the best parameters for the task of pixel-level category inference. The cost of performing a full exploration of the parameter space is prohibitive, and so our exploration varies one parameter while holding the others constant.

Of particular interest are the types of split tests made available to the training algorithm. The pixel tests listed in Figure 7.2 are those used in the experiments below, where $A[z_0]$ and $B[z_1]$ are the values of pixels within a patch of size $w \times w$ centered on the training pixel. The channels z_0 and z_1 do not have to be the same. While some test types have a basis in image structure (the difference and absolute difference of pixels is invariant to a global intensity shift, and signifies edges in the image), others are just possible discriminative combinations of pixels. In addition to these pixel tests, we evaluate two rectangle-based tests: the Haar-like features of [37] and the rectangle sum features of [30].

7.3.1.3 Supervision

Labeled image data is used to train a semantic texton forest, consisting pairs (p, c) of pixels and category labels. In the case of *full supervision* each pixel is given a

training label as shown in Figure 7.4. During training, the distribution $P(c|n)$ is computed as a normalized histogram of the training tuples which reached a particular node n :

$$P(c|n) = \frac{H_n[c]}{\sum_c H_n[c]}, \quad (7.3)$$

where $H_n[c]$ is the number of pixels of class c that passed through a node n during training. The process of computing this histogram at each node will be referred to as “filling” the forest. Filling is performed using all of the pixels in the training data, by passing each pixel down a tree and incrementing the relevant histogram bin $H_n[c]$. If desired, a small Dirichlet prior corresponding to a extra constant count added to all classes can be used to smooth the distributions.

In the case of *partial supervision*, we do not have pixel labels, but rather the set of categories present somewhere in the image. In other words, we have just the distribution $P(c|x)$ where x is an observed *topic* for the image. In this circumstance, the topic can be thought of as an underlying meaning generating the categories in the image, for example a “forest” topic is more likely to produce “tree” pixels, whereas a “city” topic would be more likely to produce “building” pixels. As we have no data about $P(p|c)$, it is modeled as a uniform distribution. Thus, to create training points to use in a partially supervised forest we first sample a category using $P(c|x)$ and then sample a pixel using $P(p|c)$. The forest is subsequently trained on these points, and the result has a fairly low pixel accuracy (though still greater than random chance).

7.3.1.4 Learning Invariances

Although using raw pixels as features is much faster than first computing descriptors or filter-bank responses, one risks losing their inherent invariances. Thus, as the distribution $P(c|n)$ is estimated with the training images these images are augmented with copies that are artificially transformed geometrically and photometrically as done by Lepetit in [15]. This allows the forest to *learn* the right degree of invariance required for a particular problem. In these experiments the transformations used were rotation, scaling, and left-right flipping as geometric transformations, as well as affine photometric transformations.

7.3.2 Experiments

In the following experiments, accuracy was measured as the mean percentage of pixels labeled correctly over all categories. A confusion matrix M was computed for each image over pixels $p \in P_R$, where P_R is the set of test pixels. Thus, the individual cells of M are computed as

$$M[i, j] = |\{p : p \in P_R, G(p) = c_i, \operatorname{argmax}_c P(c|L_p) = c_j\}|, \quad (7.4)$$

using the image ground-truth G . For these experiments the mean category accuracy μ is reported, calculated as

$$\mu = \frac{1}{Z} \sum_{i=1}^Z \alpha_i \quad (7.5)$$

where Z is the number of categories and

$$\alpha_i = \frac{M[i, i]}{\sum_j M[i, j]} . \quad (7.6)$$

A second metric is the overall accuracy α , calculated as

$$\alpha = \frac{\sum_i M[i, i]}{\sum_i \sum_j M[i, j]} . \quad (7.7)$$

The mean category accuracy μ ensures a fair balance across categories which potentially have very different numbers of pixels in the data. The overall accuracy α , on the other hand, tells us what proportion of the image the system can reliably segment. Both are important to get a sense of accuracy of the system, e.g., a high α and low μ indicates overfitting to a particular category which is disproportionately represented in the dataset.

The control training scenario was set as the following parameters:

Parameter	Value
<i>Number of Trees</i>	5
<i>Maximum Depth</i>	10
<i>Type of Split Tests</i>	$A, A + B, A \log(B), A - B $
	w
<i>Color Channels</i>	CIE Lab
<i>Data % Per Tree</i>	25

In each experiment, a single training parameter was changed while keeping all others constant to see the effect it had on test performance. Experiments were performed on the MSRC21 dataset [30]. In each experiment, ten trees were trained on a subset of the data and then filled with all of the data points as described in Section 7.3.1.4. Ten forests of five trees (with the exception of the number of trees experiment) were then created from permutations of these ten trees. The reported values are the mean α and μ of 10 segmentation trials run with those forests over the test data after being trained on the training and validation data with the specified parameters. Error bars indicating the standard error are omitted due to the remarkably low error on the values making them indiscernible. It is quite possibly due to the fact that the different forests being used for each trial are 5 trees chosen from the same set of 10, but even so it is intriguing to note that 10 different trees trained independently on different subsets of the data but with the same parameters achieved very close to the same accuracy on the test data.

In Figure 7.6, one can see the effect different combinations of feature tests has on segmentation performance. In each of the five trials a different random order of five feature tests was chosen, shown below in Table 7.2, and in each step an additional feature was added for the tree to use, accompanied by an increase in the feature pool

Table 7.1 Test Proportions for MSRC21 Dataset. A semantic texton forest was trained on the MSRC21 dataset, making all of the tests in the table available to it during training. The counts for each test were recorded over the entire forest to get a sense of which tests were most useful for categorization. As can be seen, the rectangle-based features performed best and were chosen at a much higher rate than the others. Note that the Haar features are a superset of the additive and subtractive pixel features.

	Counts	%
Rectangle	670930	39.89%
Haar	316368	18.81%
A	179816	10.69%
$A + B$	118638	7.05%
$A \times B$	107146	6.37%
$ A - B $	105794	6.29%
$A \log(B)$	92274	5.49%
$A - B$	45968	2.73%
A/B	44954	2.67%
<i>Total</i>	1681888	

size. One trial was done with every test and a pool size of 1000 (the rightmost bar on the graph) showing the practical limit on accuracy. Every additional test made available to the algorithm will usually improve performance, but there are certain groups of tests which work together better than others. So it is that performance can actually drop, sometimes dramatically, when a new test is added that elsewhere when added improved performance. The ideal mixture of tests depends to a certain extent on the data, and these experiments should only be considered in so far as they show the rate at which adding different feature tests improves performance. One good method of finding out which tests work best for a dataset is to train one forest with a large pool size and every available test. As each tree will choose tests based purely on information gain, this will give a very good indicator of which tests work best for the dataset, and a subset can be chosen either for quicker training or to optimize tree performance depending on the situation at hand. As an example, Table 7.1 gives the proportions of tests for the forest which had every test available to it, showing that for the evaluation dataset rectangle features like the rectangle sum [30] and Haar-like features [37] perform very well.

In Figure 7.7 the effect of the number of trees in the forest can be seen. It is clear that there are diminishing returns as forest size increases. In order to investigate the effects of different methods of representing color on performance forests were trained on grayscale, RGB, CIELab, and HSV images, the results of which can be seen in Figure 7.8. CIELab clearly results in the best performance, most likely due to its useful features (e.g., meaningful perceptual distances, device independence) for computer vision as described in [12]. The effect of different tree depths is shown in Figure 7.9. The values chosen for this trial were dependent on the dataset size, as the amount of data needed to fill a tree increases exponentially with tree depth, but it can be seen that as the number of nodes in the tree increases so does its classification ability, which is to be expected due to the way in which the trees are trained

Table 7.2 Test Domain Experimental Setup. The size of the feature pool and the number of different feature domains those features could be drawn from increases from right to left, with 25 total trials being performed. Results are shown in Figure 7.6.

Trial	1	2	3	4	5
1	A	$A \times B$	$A - B$	Haar	$ A - B $
2	$ A - B $	$A - B$	A/B	A	$\log(A)$
3	A/B	$A - B$	$\log(A)$	$A \times B$	A
4	$\log(A)$	Rectangle	$A - B$	$ A - B $	$A + B$
5	$A \times B$	A	$A + B$	$A \log(B)$	A/B
Pool Size	100	200	300	400	500

(i.e. nodes will stop splitting if there is no expected information gain). Finally, the effect of the w parameter (the size of the window test pixels can be chosen from) can be seen in Figure 7.10. The effect here of a steady increase in average accuracy coupled with an increase and then decline in overall accuracy is very interesting. This is likely due to the fact that the easier categories which take up many of the pixels in the dataset (e.g., grass and sky) do not require the context of nearby pixels to be classified, but the smaller and more complex categories can take advantage of nearby information to aid in classification that is only possible when the window of possible pixels is significantly increased.

7.4 Image Categorization

The bag of words histogram has been extensively used in recent years for object categorization in computer vision [32, 6, 9, 25, 31, 28, 40]. As an alternative to the typical method for creating these histograms (using interest points, descriptors and a vector quantized patch dictionary) one can use the localized bag of semantic textons (BoST), illustrated in Figure 7.11. This extends the bag of words representation with low-level semantic information, as follows.

Given for each pixel p the leaf nodes $L(p) = (l_1, \dots, l_T)$ and inferred class distribution $P(c|L(p))$, one can compute over image region r :

1. A non-normalized histogram $H_r(n)$ that concatenates the occurrences of tree nodes n across the different trees [20], and
2. A conditional distribution over the region given by the average class distribution $P(c|r) = \sum_{p \in r} P(c|L_p)P(p)$.

Experiments were performed with tree histograms (unlike the leaf histograms of [20]) where both leaf nodes l and split nodes n are included in the histogram, such that

$$H_r(n) = \sum_{n' \in \text{child}(n)} H_r(n'). \quad (7.8)$$

This histogram therefore uses the hierarchy of clusters implicit in each tree. Each $P(c|L(p))$ is already averaged across trees, and hence there is a single region prior $P(c|r)$ for the whole forest.

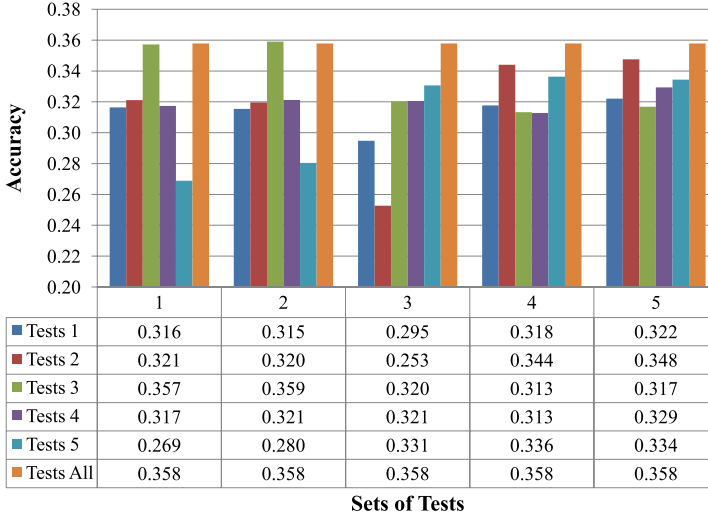


Fig. 7.6 Effect of Different Test Domains. The values shown are the mean over ten trials of the mean category accuracy (μ). The computation for μ is described in Section 7.3.2. This graph tracks the effect of increasing the test domains on accuracy, with the overall trend being that the larger the domain of tests the more accurate the system becomes, though there is quite a lot of variation with some compositional changes, particularly in set 3. For reference, a system trained with all possible tests and a pool size of 1000 is shown as the rightmost bar. The meanings of the numbers in the graph are explained in Table 7.2.

7.4.1 Tree Histograms and Pyramid Matching

Consider first the BoST histogram computed for just one tree in the STF. The kernel function (based on [11]) is then

$$K(P, Q) = \frac{1}{\sqrt{Z}} \tilde{K}(P, Q), \quad (7.9)$$

where Z is a normalization term for images of different sizes computed as

$$Z = \tilde{K}(P, P) \tilde{K}(Q, Q), \quad (7.10)$$

and \tilde{K} is the actual matching function, computed over levels of the tree as

$$\tilde{K}(P, Q) = \sum_{d=1}^D \frac{1}{2^{D-d+1}} (\mathcal{I}_d - \mathcal{I}_{d+1}), \quad (7.11)$$

using the histogram intersection \mathcal{I} [33]

$$\mathcal{I}_d = \sum_j \min(P_d[j], Q_d[j]), \quad (7.12)$$

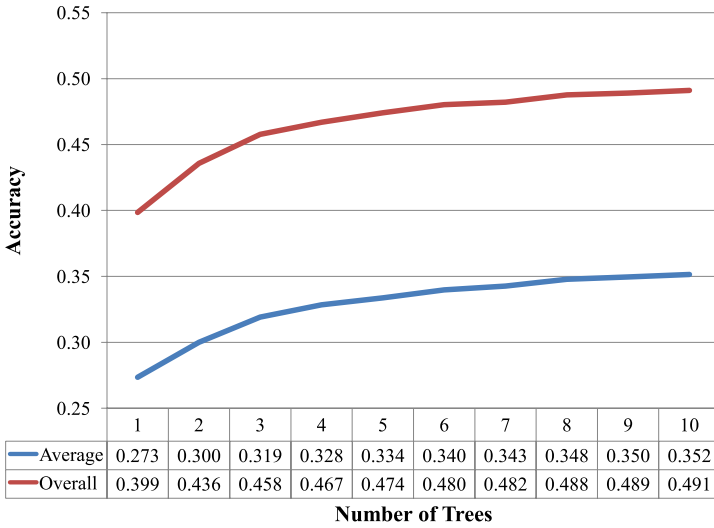


Fig. 7.7 Effect of Increasing the Number of Trees. The values shown are the mean over ten trials of the overall per-pixel accuracy (α) and the mean category accuracy (μ). The computations for α and μ are described in Section 7.3.2. We see here a clear logarithmic growth in performance with forest size, with the elbow of the graph occurring at 5 for this dataset.

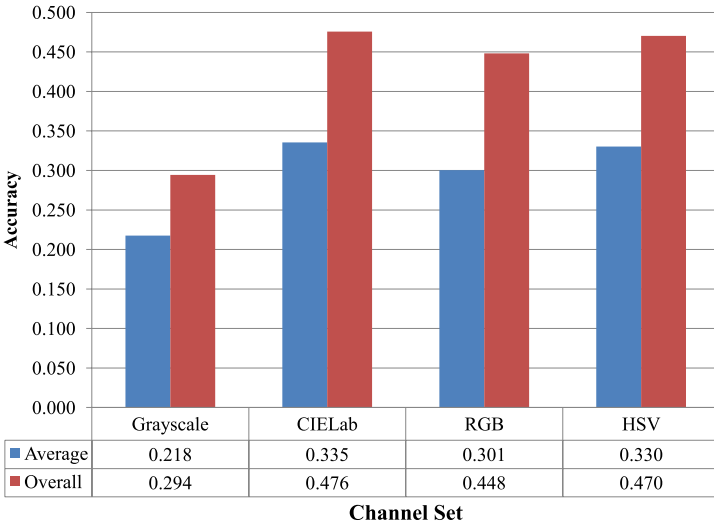


Fig. 7.8 Effect of Different Channels. The values shown are the mean over ten trials of the overall per-pixel accuracy (α) and the mean category accuracy (μ). The computations for α and μ are described in Section 7.3.2. There is a slight but consistent advantage to be gained by using orthogonal color spaces over RGB, and the addition of color gives a significant improvement over grayscale, as is to be expected.

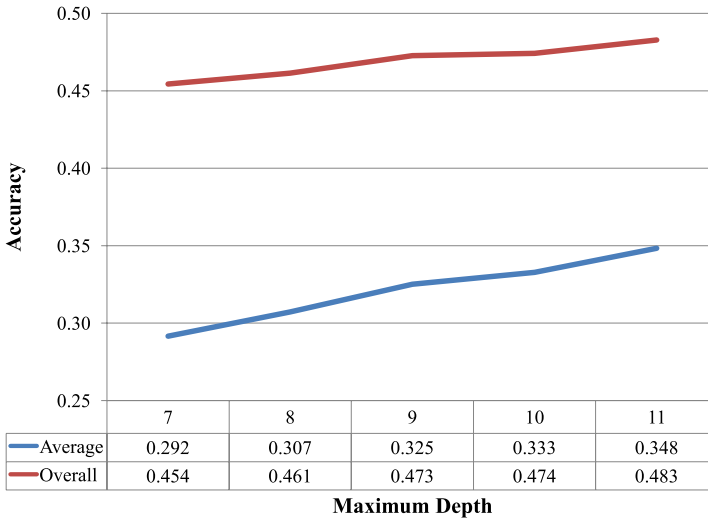


Fig. 7.9 Effect of Maximum Depth. The values shown are the mean over ten trials of the overall per-pixel accuracy (α) and the mean category accuracy (μ). The computations for α and μ are described in Section 7.3.2. We see here that tree depth results in a linear growth in accuracy, particularly for class average accuracy. While the discriminative power of the forest increases with maximum depth, so does the possibility of overfitting to the data. This can be avoided by only training on subsets of the data, but one is faced with the problem of having enough data to fill the tree so as to model the correct uncertainty (as the amount of data required increases exponentially with each additional level)

where D is the depth of the tree, P and Q are BOSTs, and P_d and Q_d are the portions of the histograms at depth d , with j indexing over all nodes at depth d . There are no nodes at depth $D + 1$, hence $\mathcal{I}_{D+1} = 0$. If the tree is not full depth, missing nodes j are simply assigned $P_d[j] = Q_d[j] = 0$.

7.4.2 Categorization Results

In this experiment, a forest was trained using the following parameters, selected based on the results of the experiments in Section 7.3.2:

Parameter	Value
Number of Trees	5
Maximum Depth	10
Feature Tests	$A, A + B, A \log(B), A - B , \text{Rectangle}, A - B$
Pool Size	600
w	10
Channels	CIE Lab
Data %	25

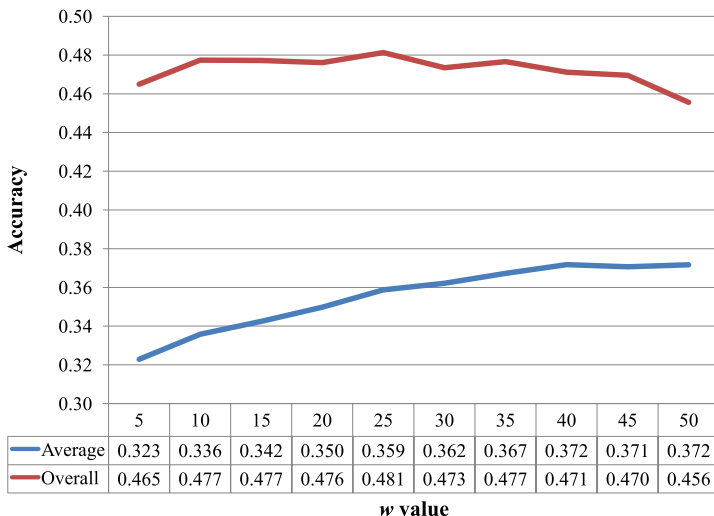


Fig. 7.10 Effect of w . The values shown are the mean over ten trials of the overall pixel accuracy (α) and the mean category accuracy (μ). The computations for α and μ are described in Section 7.3.2. We see here a general trend by which as the parameter w increases class average accuracy increases, but overall accuracy declines. Further experiments beyond the value of 55 were inconclusive due to the effect of less data being available to the training process, but seem to support this trend.

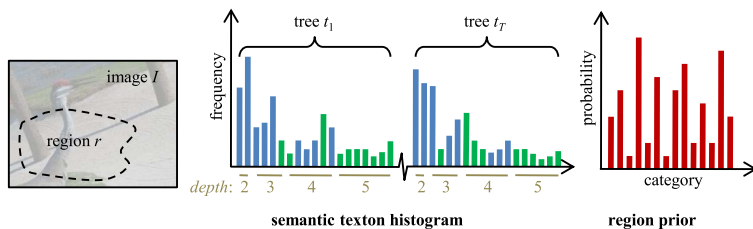


Fig. 7.11 Bags of semantic textons. Within a region r of image I we generate the semantic texton histogram and region prior. The histogram incorporates the implicit hierarchy of clusters in the STF, containing both STF leaf nodes (green) and split nodes (blue). The depth d of the nodes in the STF is shown. The STFs need not be to full depth, and empty bins in the histogram are not shown as the histogram is stored sparsely. The region prior is computed as the average of the individual leaf node category distributions $P(c|l)$.

The experiments were performed on Oliva and Torralba’s scene recognition dataset from [23], in order compare the performance of SVM classifiers trained using semantic texton forests and a standard bag of words method (using Lowe’s SIFT detector and descriptor [16], a vector quantized dictionary and a radial basis function-based kernel). As can be seen in Figure 7.12, semantic texton forests achieve better performance than the naive bag of words technique. What is worth

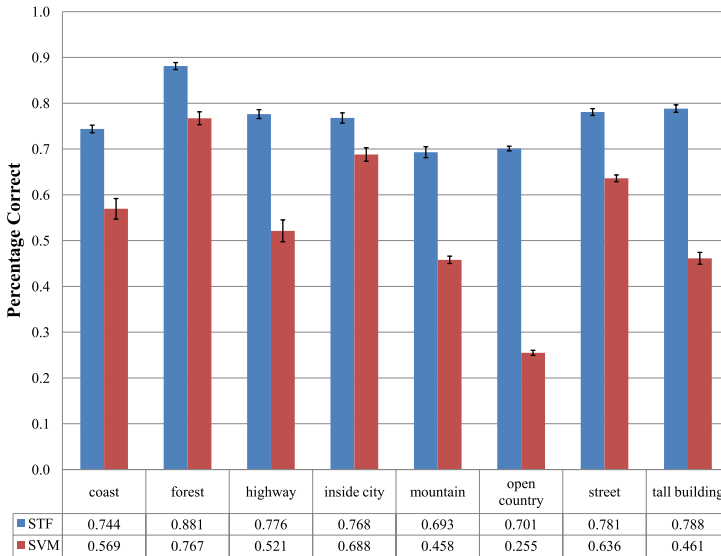


Fig. 7.12 Scene Categorization Results. The values shown are image accuracy per category. An SVM was trained using BoSTs, and its performance was compared against an SVM trained using a standard bags-of-words model using Oliva and Torralba’s scene recognition dataset from [23]. As can be seen, the technique is on par, if not slightly better in some cases, than the bag of words based algorithm.

mentioning is that STFs are able to combine many different cues in the image aside from interest points, yet are able to compute the BoST for an image very efficiently. Due to the complexity of the process ($O(n \log n)$) it is feasible for this system to perform categorization at frame rate, which would be difficult to achieve for many bag of words models.

7.5 Semantic Segmentation

The idea of the semantic segmentation of an image is built on a model of image generation which is based on dividing the real-valued and continuous visual signal of an image into *cells*, or a regular rectangular sample grid, each of which is sufficiently explained by an underlying label. To perform a semantic segmentation of an image is to infer the semantic label for every cell. For example, look at the image in Figure 7.13. Using simple semantic labels, the pixels in the image have been explained, each one generated by some unknown model for the category label. If such a segmentation can be achieved, then the image can be catalogued for image search, used for navigation, or any number of other tasks which require basic semantic understanding of arbitrary scenes.

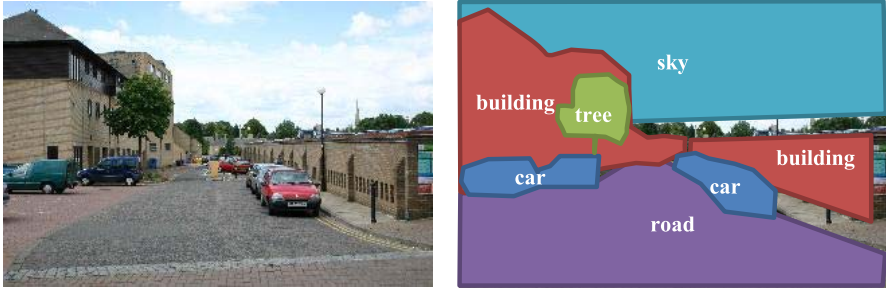


Fig. 7.13 Semantic Segmentation. A semantic segmentation of an image is one which groups the pixels together by common semantic meaning. Shown is one such segmentation of an image, using as pixel labels the objects in the scene.

7.6 Soft Classification of Pixels

Our formulation of semantic segmentation centers on the cell model, presented graphically in Figure 7.14 (graphical models are a common method of visualizing joint distributions over several random variables, see [2]). In this model, for every image there is a random variable X whose value represents the subject matter, or broad image-level category, of the image (e.g., the forest, an office, the moon). The likelihood of the various image-level categories are governed by some learned parameter χ . This topic generates cell labels c_i for each grid cell i . These are general semantic categories, that is categories of object or entity (e.g., trees, computers, rocks). The conditional probability of a semantic category given the image-level category is governed by the learned parameter γ . Finally, we have the appearance of a cell (e.g., a pixel patch, a descriptor, a histogram) which is generated by the

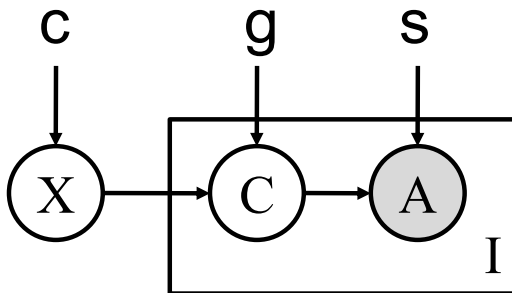


Fig. 7.14 Cell-based Image Generation Model. This figure is a graphical representation of the model [2]. For each image a particular topic is chosen, represented by the random variable x . These can be thought of as scenes (e.g., the forest, an office, the moon). For a particular topic, we generate I cells on a grid, where each cell has a semantic category c that generates the appearance a . To perform a semantic segmentation, we infer the semantic category for each grid cell.

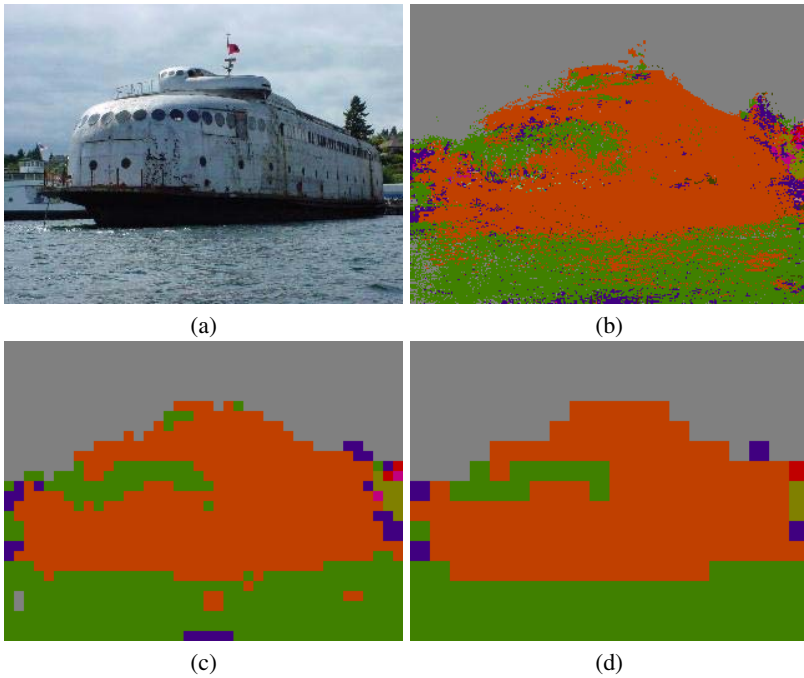


Fig. 7.15 Cell Segmentation. (a) is the original image. (b) is the image with pixel-level maximum *a posteriori* (MAP) labeling. In (c), the distributions are subsampled from (b) by 8, and in (d) the image in (b) has been subsampled by 16. As grid square size increases, detail is lost but the overall accuracy of the segmentation increases.

cell category using some process which adds in some noise (indicated by σ), e.g., a normal distribution over intensity values.

Naturally, a normal distribution over intensity values is not a sufficient generative patch model. While a general patch model may be out of reach in the near future, the promising jigsaw model [14] and other emerging techniques utilizing deep inference are showing great promise towards achieving this enviable goal. In the meantime, however, the best results have been obtained via discriminative methods. Since the appearance is observed, we can infer the category label from the appearance (essentially, reverse the arrow between c and A) and infer c by marginalizing over the topics and incorporating a discriminative model for $P(c|A)$. Previously we discussed semantic texton forests and their ability to estimate a distribution over a set of labels for arbitrary pixel regions in a discriminative manner. Now we will discuss how they can be used to infer $P(c|A)$ at the level of an image grid cell.

We have already established that we can infer $P(c|A)$ for cells at the level of a pixel with a semantic texton forest:

$$P(c|A_p) = P(c|L(p)) \propto \sum_{t=1}^T P(c|l_t)P(t). \quad (7.13)$$

As the size of a cell increases, we are faced with the problem of agglomerating the information held in individual pixels to regions. We can calculate this as

$$P(c|A_r) = P(c|r) \propto \sum_p P(c|L(p))P(p|r), \quad (7.14)$$

which leaves the conditional distribution $P(p|r)$ to be modeled. There are two practical choices for this. The first is a bivariate normal distribution with diagonal covariance and centered on the cell, and the second is

$$P(p|r) = \begin{cases} \frac{1}{|P_r|} & p \in P_r \\ 0 & \text{otherwise} \end{cases} \quad (7.15)$$

where P_r is the set of pixels in a region. We use the latter method here due to the easy of computation, but the former likely results in a better estimate. Figure 7.15 depicts both the pixel- and cell-level maximum *a posteriori* labelings for an image.

So far, we have discussed the conditional distribution $P(c|A)$, but we have yet to touch on $P(c|X)$ or $P(X)$. In the following section, we will discuss how it is possible to again use semantic texton forests to infer the parameter χ from the image data.

7.7 Image-Level Semantic Constraints

The inference of $P(X)$ is essentially the task of image categorization. The task of categorizing an image consists of determining those categories (e.g., forest images, office images, moon images) to which an image belongs. There has been much research performed on this problem, with the most successful of previous approaches using global image information [24], bags of words [9] or textons [39]. The STF categorization algorithm can be extended to exploit not just the hierarchy of semantic textons but also the node prior distributions $P(c|n)$. A non-linear support vector machine (SVM) is still used, which depends on a kernel function K that defines the similarity measure between images. To take advantage of the hierarchy in the STF, we adapt the innovative pyramid match kernel [11] to act on a pair of BOST histograms computed across the whole image, computed in the same way as described in Section 7.4.

The kernel over all trees in the STF is calculated as $K = \sum_t \kappa_t K_t$ with mixture weights κ_t . Similarly to [40], $\kappa_t = \frac{1}{T}$ results in the best categorization results. This method is very effective, but can be improved by using the learned distributions $P(c|n)$ in the STF. If $P(c|n)$ is large for class c and node n , then a large count of node n is a strong indicator that class c is present in the image. If $P(c|n)$ is small, less information is gained since the likely presence of other categories only helps as context. For example, if the target of a search is grass in an image, the count of nodes likely to be grass is more important than those likely to be motorbike. Following this intuition, a 1-vs-others SVM kernel K_c is built per category, in which the count for node n in the BOST histogram is weighted by the value $P(c|n)$. This helps balance the categories, by selectively down-weighting those that cover large

image areas (e.g., grass, water) and thus have inappropriately strong influence on the pyramid match, masking the signal of smaller classes (e.g., cat, bird).

In these experiments, the improvement that the pyramid match kernel on the hierarchy of semantic textons gives over a radial basis function on histograms of just leaf nodes is demonstrated. An improvement using the per-category kernels K_c instead of a global kernel K is also shown.

7.7.1 Categorization Results

The mean average precisions (AP) in Table 7.3 compare the modified pyramid match kernel (PMK) to a radial basis function (RBF) kernel, and compare the global kernel K to the per-category kernels K_c . In the baseline results with the RBF kernel, only the leaf nodes of the STF are used, separately per tree, using term frequency/inverse document frequency to normalize the histogram. The PMK results use the entire BoST which for the per-category kernels K_c are weighted by the prior node distributions $P(c|n)$. Note that the mean AP is a much harder metric and gives lower numbers than recall precision or AuC; the best result in the table shows very accurate categorization. As can be seen in Table 7.3, the pyramid match kernel considerably improves on the RBF kernel. By training a per-category kernel, a small but noticeable improvement is obtained. Due to its performance, the PMK with per-category kernels to train the SVM is used as χ .

Table 7.3 Image categorization results. (Mean AP).

	Global kernel K	Per-category kernel K_c
RBF	.499	.525
PMK	.763	.783

7.7.2 The Image Level Prior

To relate this system of image categorization back to the cell model for semantic segmentation, the task the SVM is performing is to infer the value of X for an image. In essence, instead of χ being a global prior distribution over topics, it is this system, and depends on the BoST computed for a particular image in order to compute $P(X)$. We can think of $P(X)$ as being related to χ thus:

$$P(X) = \chi(\text{BoST}_d)[x] \quad (7.16)$$

where χ is the SVM classifier.

Moreover, since X and c in the scenario just presented have the same domain, γ is not a learned distribution but instead a binary function:

$$\gamma[t, c] = \begin{cases} 1 & \text{if } x = c \\ 0 & \text{otherwise} \end{cases} \quad (7.17)$$

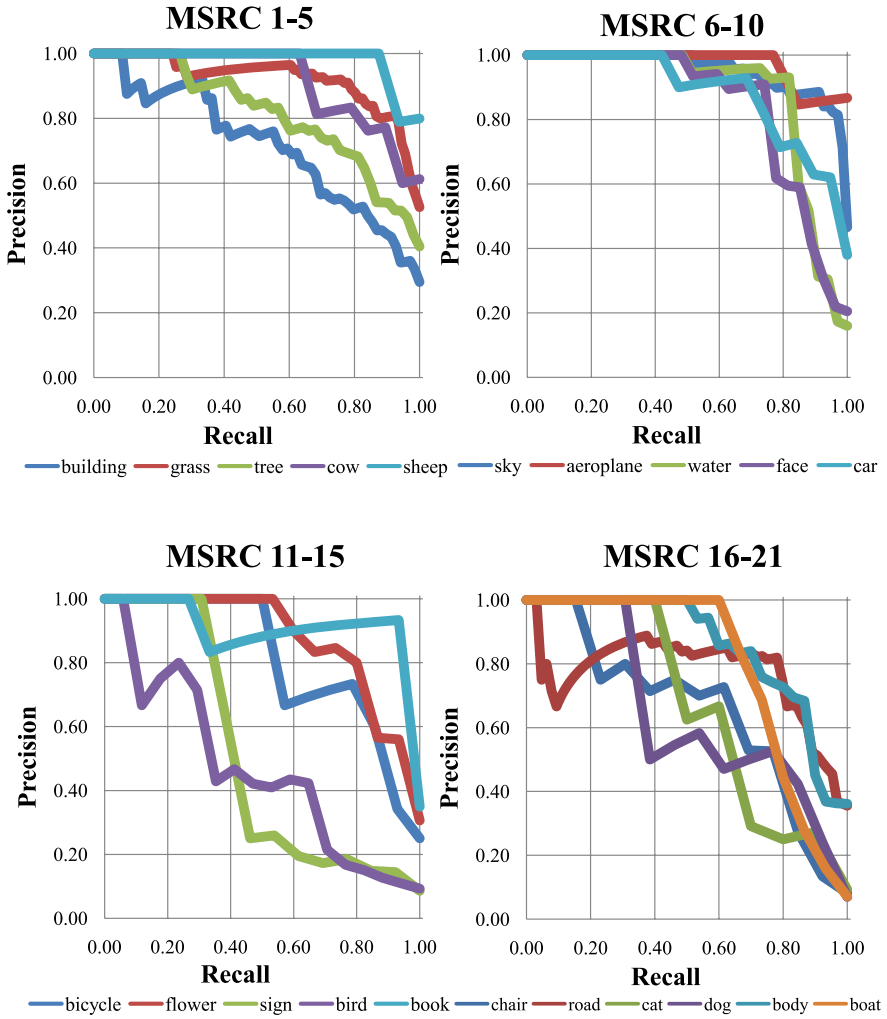


Fig. 7.16 MSRC Categorization Results Shown here are the precision recall curves for all of the categories in the MSRC21 dataset.

Thus, whereas one would usually marginalize over X when computing $P(c_i)$ in the following manner (substituting lower case letters for observed variables):

$$P(c_i) = P(c_i|a_i) \sum_X P(X) P(c_i|X) \prod_{j \neq i} \sum_{c_j} P(c_j|X) P(c_j|a_j) \tag{7.18}$$

this equation is derived instead :

$$P(c_i) = P(c_i|a_i) P(c_i|x) P(x) \prod_{j \neq i} \sum_{c_j} P(c_j|x) P(c_j|a_j) \tag{7.19}$$

which, given the binary nature of $P(c|T) = \lambda[t, c]$ and that in this situation $t \equiv c$, collapses further to:

$$P(c_i) = P(c_i|a_i)P(c_i) \prod_{j \neq i} P(c_j|a_j) \quad (7.20)$$

where $P(c_i)$ is the result of the SVM classifier and $P(c|A)$ is computed using the STF as described above. Furthermore, the effect of the right-hand product is estimated by a power α on $P(c)$ in the results presented in this chapter. Since this is no longer inference *per se*, the result of this process is referred to as an “image level prior”, or ILP. While these optimizations result in a system which is very efficient, they are made at the cost of a more powerful and expressive model.

7.8 Compositional Constraints

To demonstrate the power of the BOSTs as features for segmentation, they are integrated into the TextonBoost algorithm [30]. The goal is to segment an image into coherent regions and simultaneously infer the class label of each region. In [30], a boosting algorithm selected features based on localized counts of textons to model patterns of texture, layout and context. The context modeled in [30] was “textural”, for example: sheep often stand on something green. We adapt the rectangle count features of [30] to act on both the semantic texton histograms and the BOST region priors. The addition of region priors allows us to model context based on *semantics* [26], not just texture. Continuing the example, this new model can capture the notion that sheep often stand on *grass*.

The segmentation algorithm works as follows. For speed, a second randomized decision forest is used in place of boosting. This *segmentation forest* is trained to act at image cells i , using bags and priors computed using Equation 7.15 for $P(p|r = i)$. At test time, the segmentation forest is applied at each pixel p densely or, for more speed, on a grid. The most likely class in the averaged category distribution gives the final segmentation for each cell. The split tests compute either the count $H_{r+i}(n = n')$ of semantic texton n' , or the probability $P(c | r + i)$ of class c , within rectangle r translated relative to cell i . By translating rectangle r relative to the cell i being classified, and by allowing r to be a large distance away from i (up to half the image size), such features can exploit texture, layout and context information. This extension to their features exploits semantic context by using the region prior probabilities $P(c|r + i)$ inferred by the semantic textons.

7.9 Experiments

Before presenting in-depth results for segmentation, let us look briefly at the STFs themselves. In Figure 7.17, we visualize the inferred leaf nodes $L = (l_1, \dots, l_T)$ for each pixel i and the most likely category $c_i = \arg \max_{c_i} P(c_i|L)$. Observe that

Fig. 7.17 Textonizations. Shown here are a large selection of textonizations performed by a semantic texton forest. The first column shows the image, the middle five are textonizations from the five trees in the forest, followed by the ground truth image and the combined textonization of the forest. In the textonizations, a separate color is given to each texton index to give a sense of leaf-membership for a pixel.

the textons in each tree capture different aspects of the underlying texture and that even at such a low level the distribution $P(c|L)$ contains significant semantic information. Table 7.4 gives a naïve segmentation baseline on the MSRC dataset by comparing c_i to the ground truth.

Clearly, this segmentation is poor, especially when trained in a weakly supervised manner, since only very local appearance and no context is used. Even so, the signal is remarkably strong for such simple features (random chance is under 5%). Below

Table 7.4 Naïve Segmentation Baseline on MSRC21. Using the parameters chosen as a result of the experiments in Section 7.3.2 we are able to obtain a solid baseline for segmentation.

	Global Average	
supervised	48.7%	41.5%
weakly supervised	17.7%	27.8%

is shown how using semantic textons as features in higher level classifiers greatly improves these numbers, even with weakly supervised or unsupervised STFs.

Except where otherwise stated, STFs were used with the following parameters, hand-optimized on the MSRC validation set: distance $w = 31$, $T = 5$ trees, maximum depth $D = 10$, 500 feature tests and 5 threshold tests per split, and $\frac{1}{4}$ of the data per tree, resulting in approximately 500 leaves per tree. Training the STF on the MSRC dataset took only 15 minutes. The tests used were $A + B$, $A - B$, $|A - B|$, and A , a combination motivated by the experiments in Section 7.3.2 and performance requirements.

7.9.1 MSRC21 Dataset

We first examine the influence of different aspects of our system on segmentation accuracy. Segmentation forests were trained using (a) the histogram $H_r(l)$ of just leaf nodes l , (b) the histogram $H_r(n)$ of *all* tree nodes n , (c) just the region priors $P(c|r)$, (d) the full model using all nodes and region priors, (e) the full model trained without random transformations, (f) all nodes using an unsupervised STF (no region priors are available), and (g) all nodes using a weakly-supervised STF (only image labels). The category average accuracies are given in Table 7.5 with and without the image-level prior.

There are several conclusions to draw. (1) In all cases the ILP improves results. (2) The hierarchy of clusters in the STF gives a noticeable improvement. (3) The region priors alone perform remarkably well. Comparing to the segmentation result using only the STF leaf distributions (34.5%) this shows the power of the localized BOSTs that exploit semantic context. (4) Each aspect of the BOST adds to the model. While, without the ILP, score (b) is slightly better than the full model (d), adding

Table 7.5 Comparative segmentation results on MSRC.

	Without ILP	With ILP
(a) only leaves	61.3%	64.1%
(b) all nodes	63.5%	65.5%
(c) only region priors	62.1%	66.1%
(d) full model	63.4%	66.9%
(e) no transformations	60.4%	64.4%
(f) unsupervised STF	59.5%	64.2%
(g) weakly supervised STF	61.6%	64.6%

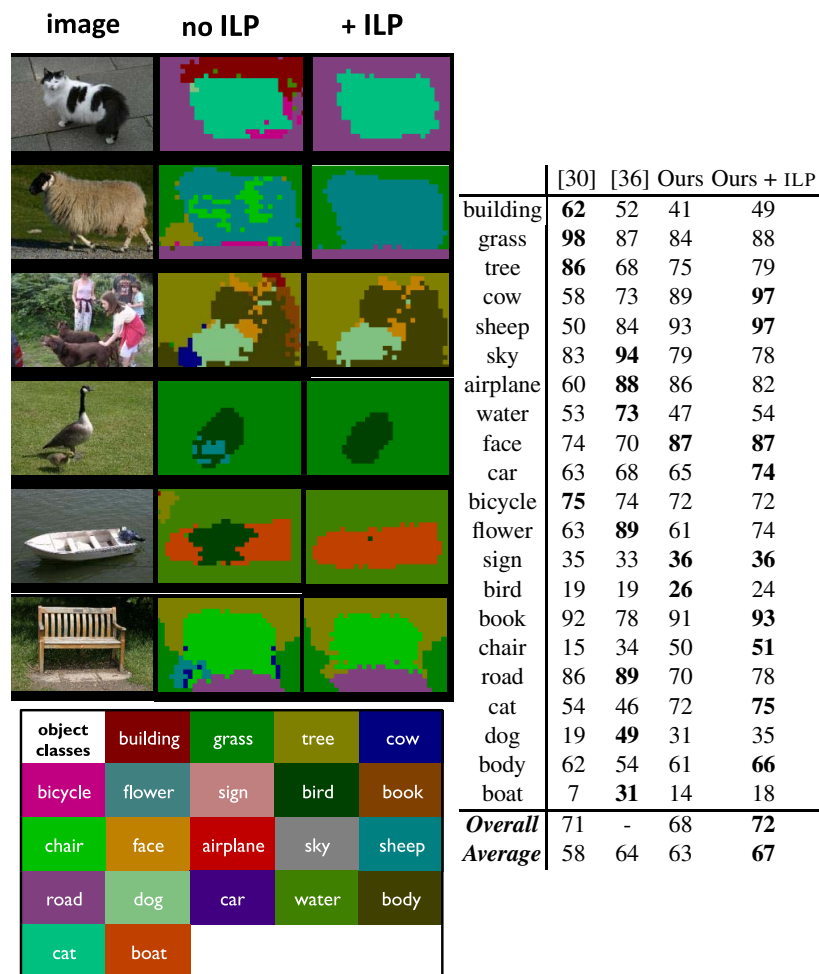


Fig. 7.18 MSRC21 segmentation results. Left: Segmentations on test images using semantic texton forests. Note how the good but somewhat noisy segmentations are cleaned up using our image-level prior (ILP) that emphasizes the categories likely to be present. (Note that neither a Markov nor conditional random field are used, which could clean up the segmentations to precisely follow image edges [30]). Right: Segmentation accuracies (percent) over the whole dataset, without and with the ILP. Highly efficient semantic textons achieve a significant improvement on previous work.

in the ILP shows how the region priors and textons work together.¹ (5) Random transformations of the training images improve performance by adding invariance.


¹ This effect may be due to segmentation forest (b) being over-confident: looking at the 5 most likely classes inferred for each pixel, (b) achieves 87.6% while (d) achieves a better 88.0%.



Fig. 7.19 Further MSRC segmentation results.

(6) Performance increases with more supervision, but even unsupervised STFs allow good segmentations.

Given this insight, the algorithm is compared against [30] and [36]. The same train/test split is used as in [30] (though not [36]). The results are summarized in Figure 7.18, with further segmentation results in Figure 7.19. Across the whole challenging dataset, using the full model with ILP achieved a class average performance of 66.9%, a significant improvement on both the 57.7% of [30] and the 64% of [36]. The global accuracy also improves slightly on [30]. The image-level prior improves



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

Fig. 7.20 VOC 2007 segmentation results. Above: Test images with ground truth and our inferred segmentations using the ILP (not the DLP). This dataset is extremely challenging and the resulting segmentations are thus slightly noisier. Below: Segmentation accuracies (percent) over the whole dataset. The left three results compare the method to the Brookes segmentation entry [8], and show that it is over twice as accurate. The two results on the right compare the best automatic segmentation-by-detection entry (see text) [8] with the STF algorithm using the TKK results as a detection-level prior (DLP). The algorithm improves the accuracy of segmentation-by-detection by over 10%.

performance for all but three classes, but even without it, results are still highly competitive with other methods. The use of balanced training has resulted in more consistent performance across classes, and significant improvements for certain difficult classes: cow, sheep, bird, chair, and cat. A Markov or conditional random field is not used here, which would likely further improve our performance [30].

These results used the learned and extremely fast STF, without needing any slow hand-designed filter-banks or descriptors. Extracting the semantic textons at every pixel takes an average of only 275 milliseconds per image, categorization takes 190 ms, and evaluating the segmentation forest only 140 ms. For comparison [30] took over 6 seconds per test image, and [36] took an average of over 2 seconds per image for feature extraction and between 0.3 to 2 seconds for estimating the segmentation.

This algorithm is well over 5 times faster *and* improves quantitative results. A real-time implementation of the STFs can be achieved on a standard PC, as the complexity is just $O(TD)$ per pixel.

The following parameter settings were used: $T = 50$ trees of depth $D = 14$, with training examples taken every 10 pixels in a random 50% of the training images, and using 1000 random feature tests and 20 random threshold tests at each split node. The ILP smoothing term $\alpha = 0.5$. To add invariance, the training set was augmented: the original plus 3 copies with random transformations of rotation up to 6° , scaling up to 1.2x, left-right flipping, and affine intensity changes up to $1.2I + 0.05$. Training took under 2 hours for the segmentation forest. The resulting forest used a total of 24805 node features and 107311 region prior features.

7.9.2 VOC 2007 Segmentation Dataset

The VOC object recognition challenge added a segmentation task to the competition in 2007 [8]. This dataset contains 21 extremely challenging categories including background. A STF, a segmentation forest, and an ILP were trained on this data, using the “trainval” split and keeping parameters as for MSRC. The results in Figure 7.20 compare with [8]. This algorithm performs over twice as well as the only *segmentation* entry (Brookes), and the addition of the ILP further improves performance by 4%. The actual winner of the segmentation challenge, the TKK algorithm, used *segmentation-by-detection* that fills in the detected object bounding boxes by category. To see if our algorithm could use a *detection*-level prior DLP (identical to the ILP but using the detected bounding boxes and varying with image position) the TKK entry output was used as the DLP. The STF algorithm gave a large 12% improvement over the TKK *segmentation-by-detection*, highlighting the power of STFs as features for segmentation.

7.10 Discussion

We have examined semantic texton forests and their applications to image categorization and semantic segmentation. They act as efficient texton codebooks, which do not depend on local descriptors or expensive k -means clustering, and when supervised during training can infer a distribution over categories at each pixel. We examined how the training parameters affect performance, and how a hierarchical matching kernel can be used in a non-linear SVM classifier to achieve image categorization. Also, we saw how bags of semantic textons enabled state-of-the-art performance on challenging datasets for semantic segmentation, and how the use of an inferred image-level prior significantly improves segmentation results. The substantial gains of the method over traditional texton-based methods are training and testing efficiency and improved quantitative performance.

While semantic texton forests are quite powerful, there is much work left to be done. One limitation of the system is the large dimensionality of the bag of semantic textons. This necessitates a trade-off between the memory usage of the semantic

texton integral images and the training time if they are computed at runtime. While using just the region priors can be more memory efficient, this comes at some cost in pixel accuracy. Another weakness, mentioned previously, is the lack of a model of the inter-pixel relationships (like the conditional Markov random field from [30]) which incorporates an understanding of local structure in the image when arriving at a segmentation.

Perhaps the greatest limitation of this technique, however, is the fact that it is a discriminative learning algorithm and thus requires a fully-supervised training scenario. As the data required must be labeled on a per-pixel basis with a preset number of categories, this is a large barrier to the general use of semantic texton forests for image understanding. However, as the genesis of a generative patch model nears, it is possible that a hybridized version of the technique could be used in partially supervised and unsupervised cases. Perhaps most encouraging are the vast quantities of training data being created through projects like LabelMe [27], in which humans provide semantic segmentations of images of all kinds with arbitrary categories. While difficult to work with, such data could be exploited by discriminative learning algorithms like the semantic texton forest to understand a wider range of object categories.

References

1. Amit, Y., Geman, D.: Shape quantization and recognition with randomized trees. *Neural Computation* 9(7), 1545–1588 (1997)
2. Bishop, C.: *Pattern Recognition and Machine Learning*. Springer-Verlag New York, Inc. (2006)
3. Bosch, A., Zisermann, A., Muñoz, X.: Image classification using random forests and ferns. In: *Proceedings of the International Conference on Computer Vision* (2007)
4. Breiman, L.: Random forests. *Machine Learning* 45(1), 5–32 (2001)
5. Breiman, L., Friedman, J., Olshen, R.: *Classification and Regression Trees*. Wadsworth, Belmont (1984)
6. Csurka, G., Dance, C., Fan, L., Willamowski, J., Bray, C.: Visual categorization with bags of keypoints. In: *Proceedings of the International Workshop on Statistical Learning in Computer Vision, ECCV* (2004)
7. Elkan, C.: Using the triangle inequality to accelerate k -means. In: *Proceedings of the International Conference on Machine Learning*, pp. 147–153 (2003)
8. Everingham, M., Van Gool, L., Williams, C.K.I., Winn, J., Zisserman, A.: *The PASCAL VOC Challenge* (2007), <http://www.pascal-network.org/challenges/VOC/voc2007/workshop/index.html>
9. Fei-Fei, L., Perona, P.: A bayesian hierarchical model for learning natural scene categories. In: *Proceedings of the International Conference on Computer Vision and Pattern Recognition* (2005)
10. Geurts, P., Ernst, D., Wehenkel, L.: Extremely randomized trees. *Machine Learning* 36(1), 3–42 (2006)
11. Grauman, K., Darrell, T.: The pyramid match kernel: Discriminative classification with sets of image features. In: *Proceedings of the International Conference on Computer Vision* (2005)

12. Jain, A.K.: *Fundamentals of Digital Image Processing*. Prentice-Hall, New Jersey (1989)
13. Jurie, F., Triggs, B.: Creating efficient codebooks for visual recognition. In: *Proceedings of the International Conference on Computer Vision*, pp. 604–610 (2005)
14. Lasserre, J., Kannan, A., Winn, J.: Hybrid learning of large jigsaws. In: *Proceedings of the International Conference on Computer Vision and Pattern Recognition*, Minneapolis (2007)
15. Lepetit, V., Lagger, P., Fua, P.: Randomized trees for real-time keypoint recognition. In: *Proceedings of the International Conference on Computer Vision and Pattern Recognition*, pp. 775–781 (2005)
16. Lowe, D.: Distinctive image features from scale-invariant keypoints. *International Journal of Computer Vision* 60(2), 91–110 (2004)
17. Malik, J., Belongie, S., Leung, T., Shi, J.: Contour and texture analysis for image segmentation. *International Journal of Computer Vision* 43(1), 7–27 (2001)
18. Marée, R., Geurts, P., Piater, J., Wehenkel, L.: Random subwindows for robust image classification. In: *Proceedings of the International Conference on Computer Vision and Pattern Recognition*, pp. 34–40 (2005)
19. Mikolajczyk, K., Schmid, C.: Scale and affine invariant interest point detectors. *International Journal of Computer Vision* 60(1), 63–86 (2004)
20. Moosmann, F., Triggs, B., Jurie, F.: Fast discriminative visual codebooks using randomized clustering forests. In: *Proceedings of the International Conference on Neural Information Processing Systems* (2006)
21. Nistér, D., Stewénius, H.: Scalable recognition with a vocabulary tree. In: *Proceedings of the International Conference on Computer Vision and Pattern Recognition* (2006)
22. Nowak, E., Jurie, F., Triggs, B.: Sampling strategies for bag-of-features image classification. In: *Proceedings of the International Conference on Computer Vision* (2006)
23. Oliva, A., Torralba, A.: Modeling the shape of the scene: a holistic representation of the spatial envelope. *International Journal of Computer Vision* 42(3), 145–175 (2001)
24. Oliva, A., Torralba, A.: Building the gist of a scene: The role of global image features in recognition. *Visual Perception, Progress in Brain Research* 155(1), 23–26 (2006)
25. Quelhas, P., Monay, F., Odobez, J.M., Gatica, D., Tuytelaars, T.: Modeling scenes with local descriptors and latent aspects. In: *Proceedings of the International Conference on Computer Vision* (2005)
26. Rabinovich, A., Vedaldi, A., Galleguillos, C., Wiewiora, E., Belongie, S.: Objects in context. In: *Proceedings of the International Conference on Computer Vision* (2007)
27. Russell, B., Torralba, A., Murphy, K., Freeman, W.T.: Labelme: a database and web-based tool for image annotation. *Journal of Computer Vision* 77(1-3), 157–173 (2008)
28. Russell, B.C., Efros, A.A., Sivic, J., Freeman, W.T., Zisserman, A.: Using multiple segmentations to discover objects and their extent in image collections. In: *Proceedings of the International Conference on Computer Vision and Pattern Recognition* (2006)
29. Schindler, G., Brown, M., Szeliski, R.: City-scale location recognition. In: *Proceedings of the International Conference on Computer Vision and Pattern Recognition*, Minneapolis (2007)
30. Shotton, J., Winn, J., Rother, C., Criminisi, A.: Textonboost for image understanding: Multi-class object recognition and segmentation by jointly modeling texture, layout, and context. *International Journal of Computer Vision* 81(1) (2009)
31. Sivic, J., Russel, B., Efros, A., Zisserman, A., Freeman, W.: Discovering objects and their localization in images. In: *Proceedings of the International Conference on Computer Vision*, Beijing, China, pp. 370–377 (2005)
32. Sivic, J., Zisserman, A.: Video Google: A text retrieval approach to object matching in videos. In: *Proceedings of the International Conference on Computer Vision*, vol. 2, pp. 1470–1477 (2003)

33. Swain, M., Ballard, D.: Color indexing. *Int. J. Computer Vision* 7, 11–32 (1991)
34. Tuytelaars, T., Schmid, C.: Vector quantizing feature space with a regular lattice. In: *Proceedings of the International Conference on Computer Vision* (2007)
35. Varma, M., Zisserman, A.: A statistical approach to texture classification from single images. *International Journal of Computer Vision* 62(1-2), 61–81 (2005)
36. Verbeek, J., Triggs, B.: Region classification with markov field aspect models. In: *Proceedings of the International Conference on Computer Vision and Pattern Recognition* (2007)
37. Viola, P., Jones, M.: Rapid object detection using a boosted cascade of simple features. In: *Proceedings of the International Conference on Computer Vision and Pattern Recognition*, pp. 511–518 (2001)
38. Winder, S., Brown, M.: Learning local image descriptors. In: *Proceedings of the International Conference on Computer Vision and Pattern Recognition* (2007)
39. Winn, J., Criminisi, A., Minka, T.: Object categorization by learned universal visual dictionary. In: *Proceedings of the International Conference on Computer Vision, Beijing, China*, pp. 1800–1807 (2005)
40. Zhang, J., Marszałek, M., Lazebnik, S., Schmid, C.: Local features and kernels for classification of texture and object categories: A comprehensive study. *International Journal of Computer Vision* 73(2), 213–238 (2007)