Chapter 15
Semantic Texton Forests for Image Categorization and Segmentation

M. Johnson, J. Shotton, and R. Cipolla

Semantic texton forests (STFs) are a form of random decision forest that can be employed to produce powerful low-level codewords for computer vision. Each decision tree acts directly on image pixels, resulting in a codebook that bypasses the expensive computation of filter-bank responses or local descriptors. Further, STFs are extremely fast to both train and test, especially when compared with k-means clustering and nearest-neighbor assignment of feature descriptors. The nodes in the STFs provide both an implicit hierarchical clustering into semantic textons, and also an explicit pixel-wise local classification estimate. In this chapter we (i) investigate STFs as learned visual dictionaries; (ii) show how STFs can be used for both image categorization and semantic segmentation by aggregating hierarchical *bags of semantic textons*; (iii) demonstrate that STFs allow us to exploit semantic context in segmentation; and (iv) show how a global image-level categorization can be used as a prior to improve the accuracy of semantic segmentation. We also see that the efficient tree structures of STFs allow at least a five-fold increase in execution speed over competing techniques.

This work was undertaken while the first two authors were at the University of Cambridge and Toshiba Corporate Research and Development Center respectively.

M. Johnson (✉)
Unicorn Media, Temple, USA

J. Shotton
Microsoft Research, Cambridge, UK

R. Cipolla
University of Cambridge, Cambridge, UK

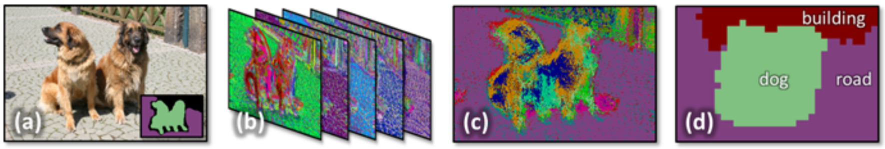


Fig. 15.1 Semantic texton forests. (**a**) Test image, with ground truth in-set. Semantic texton forests very efficiently compute (**b**) a set of semantic textons per pixel and (**c**) a rough per-pixel classification (a prior for the subsequent segmentation). Our algorithm uses both the textons and priors as features to give coherent semantic segmentation (**d**). Colors show texton indices in (**b**), but categories corresponding to the ground truth in (**c**) and (**d**)

15.1 Introduction

This chapter discusses *semantic texton forests*, and demonstrates their use for image categorization and semantic segmentation; see Fig. 15.1. Our aim is to show 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. Semantic texton forests (STFS) achieve both these goals. STFS are randomized decision forests that use only simple pixel comparisons on local image patches, and output both an implicit hierarchical clustering into semantic textons and an explicit local classification of the patch category.

We 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). To these ends, we propose the *bag of semantic textons* (BOST). The BOST is computed over a given image region, and extends the bag of words model [81] by combining a histogram of the hierarchical semantic textons with a region prior category distribution. For categorization, we obtain a highly discriminative descriptor by considering the image as a whole. For segmentation, we use many local rectangular regions and build a second randomized decision forest that achieves efficient and accurate segmentation.

Inferring the correct segmentation depends on local image information that can often be ambiguous. The global statistics of the image, however, are more discriminative and may be sufficient to accurately estimate the image categorization. We therefore investigate how an SVM-based image categorization can act as an *image-level prior* to improve segmentation: the classification output of the SVM is used as a prior to emphasizing the categories most likely to be present given the global appearance of the image.

To summarize, the main topics in this chapter are: (i) semantic texton forests which efficiently provide both a hierarchical clustering into semantic textons and a local classification; (ii) the bag of semantic textons model, and its applications in categorization and segmentation; (iii) how STFS allow us to exploit semantic context for segmentation; and (iv) the use of the image-level prior to improve segmentation accuracy.

15.1.1 Related Work

Textons [176, 229, 381] and visual words [348] have proven powerful discrete image representations for categorization and segmentation [81, 342, 404, 416]. Filter-bank responses (derivatives of Gaussians, wavelets, *etc.*) or invariant descriptors (*e.g.* SIFT [225]) are computed across a training set, either at sparse interest points (*e.g.* [248]) or more densely; results in [267] suggest that densely sampling visual words improves categorization accuracy. 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 system, even with optimizations such as kd-trees, the triangle inequality [97], or hierarchical clusters [266, 321].

The work of Moosmann *et al.* [253] proposed a more efficient alternative, in which training examples are recursively divided using a randomized decision forest [5, 44, 128] and where the splits in the decision trees are comparisons of a descriptor dimension to a threshold. With semantic texton forests, we extend [253] in three ways: (i) we learn a codebook that acts directly on image pixels, bypassing the expensive step of computing image descriptors; (ii) while [253] use the learned decision forest only for clustering, we also use it as a classifier, which enables us to use semantic context for image segmentation; and (iii) in addition to the leaf nodes used in [253], we include the split nodes as hierarchical clusters. A related method, the pyramid match kernel (PMK) [141], exploits a hierarchy in descriptor space, though the PMK requires the computation of feature descriptors and is primarily applicable only to kernel-based classifiers. The pixel-based features we use are similar to those in [214], but our forests are trained to recognize object categories, not to match particular feature points.

Other work has also looked at alternatives to k-means. The work of [376] quantized feature space into a hyper-grid, but required descriptor computation and can result in very large visual word codebooks. Winder and Brown [402] learned the parameters of generic image descriptors for 3D matching, though did not address visual word clustering. Jurie and Triggs [177] proposed building codebooks using mean shift, but did not incorporate semantic supervision in the codebook generation.

15.2 Randomized Decision Forests

We begin with a brief review of randomized classification forests [5, 128]. We follow the notation and terminology introduced in Chaps. 3 and 4 as closely as possible. A decision forest is an ensemble of T decision trees. Associated with each node j in the tree is a learned class distribution $p_j(c)$. A decision tree works by recursively branching left or right down the tree according to a series of learned binary functions computed at 2D pixel position $\mathbf{u} = (u_x, u_y)$, until a leaf node l is reached. The whole forest achieves an accurate and robust classification by averaging the class distributions over the leaf nodes $\mathcal{L}(\mathbf{u}) = \{l_t(\mathbf{u})\}_{t=1}^{T}$ reached for all T trees:

$$p(c|\mathbf{u}) = \frac{1}{T} \sum_{l \in \mathcal{L}(\mathbf{u})} p_l(c). \tag{15.1}$$

Existing work has shown the power of decision forests as either classifiers [36, 214] or a fast means of clustering descriptors [253]. In this chapter we show how to simultaneously exploit *both* classification and clustering. Furthermore, we generalize [253] to use the tree hierarchies as hierarchical clusters.

We use the standard randomized learning algorithm described in Chap. 4 to learn binary forests. Each tree is trained separately on a small random subset $\mathcal{S}' \subseteq \mathcal{S}$ of the training data $\mathcal{S}$ (here we employ the bagging randomness model). We will denote the weak learner decision function as

$$h(\mathbf{u}; \boldsymbol{\theta}_j) = \left[f(\mathbf{u}; \boldsymbol{\phi}_j) \geq \tau_j\right] \tag{15.2}$$

which is governed by node-specific parameters $\boldsymbol{\theta}_j = (\boldsymbol{\phi}_j, \tau_j)$ consisting of offset parameters $\boldsymbol{\phi}$ (see below) and a threshold τ. Learning proceeds as described in Chap. 4, using the standard entropy-based information gain objective. The training continues to a maximum depth D or until no further information gain is possible. The class distributions $p_j(c)$ are estimated empirically as a histogram of the class labels $c(\mathbf{u})$ of the training examples $\mathbf{u} \in \mathcal{S}_j$ that reached node j.

The amount of training data may be significantly biased towards certain classes in some datasets. A classifier learned on these data will have a corresponding prior preference for those classes. We weight each training example by the inverse class frequency as $w(\mathbf{u}) = \xi_{c(\mathbf{u})}$ with $\xi_c = (\sum_{\mathbf{u} \in \mathcal{S}}[c = c(\mathbf{u})])^{-1}$. This weight is applied to each example when accumulating the histograms used to compute the information gain. The classifiers trained using this weighting tend to give a better class average accuracy.

Using ensembles of trees trained on only small random subsets of the data helps to speed up training time and reduce over-fitting [5]. The trees are fast to learn and extremely fast to evaluate since only a small portion of the tree is traversed for each data point. After training, an improved estimate of the class distributions is obtained using *all* pixels in the training data $\mathbf{u} \in \mathcal{S}$, not just the subset $\mathcal{S}'$. We found this to improve the generalization of the classifiers slightly, especially for classes with few training examples.

15.3 Semantic Texton Forests

Semantic texton forests (STFs) are a specific form of randomized decision forests that can be used for both clustering and classification. The features $f(\mathbf{u}; \boldsymbol{\phi})$ in STFs act on small image patches centered at pixel $\mathbf{u}$ of size $\Delta \times \Delta$ pixels, as illustrated in Fig. 15.2(a). The feature parameters $\boldsymbol{\phi}$ denote one of the following functions:

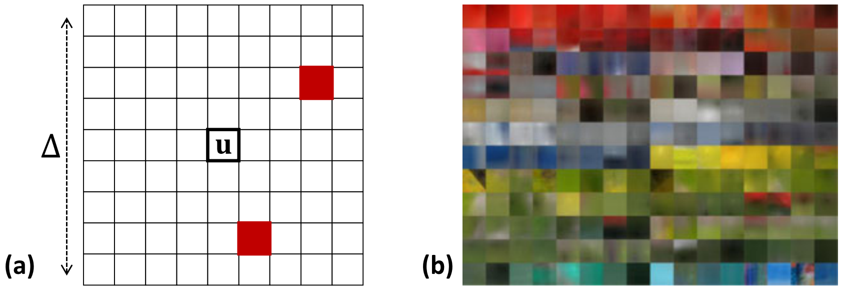


Fig. 15.2 (**a**) Semantic texton forests features. The split nodes in semantic texton forests use simple functions of raw image pixels within a $\Delta \times \Delta$ patch: either the raw value of a single pixel, or the sum, difference, or absolute difference of a pair of pixels (*red*). (**b**) Semantic textons. A visualization of leaf nodes from one tree (distance $\Delta = 21$ pixels). Each patch is the average of all patches in the training images assigned to a particular leaf node l. We can observe distinct patterns of color, horizontal, vertical, and diagonal edges, blobs, ridges, and corners. This visualization also allows a simple image reconstruction; see Fig. 15.8. Note that also associated with each semantic texton is a learned distribution $p_l(c)$ (not shown) which is used as the rough local segmentation of Fig. 15.1(c)

(i) the value $v(\mathbf{u} + \boldsymbol{\delta}, b)$ of a single pixel at $\mathbf{u} + \boldsymbol{\delta}$ in color channel b; (ii) the sum $v(\mathbf{u} + \boldsymbol{\delta}_1, b_1) + v(\mathbf{u} + \boldsymbol{\delta}_2, b_2)$; (iii) the difference $v(\mathbf{u} + \boldsymbol{\delta}_1, b_1) - v(\mathbf{u} + \boldsymbol{\delta}_2, b_2)$; or (iv) the absolute difference $|v(\mathbf{u}+\boldsymbol{\delta}_1, b_1) - v(\mathbf{u}+\boldsymbol{\delta}_2, b_2)|$. Here, function v denotes a look-up into the image pixel colors. The color channels b_1 and b_2 need not be the same.

To textonize an image, the $\Delta \times \Delta$ patch centered at each pixel $\mathbf{u}$ is passed down the STF resulting in semantic texton leaf nodes $\mathcal{L} = \{l_t\}_{t=1}^{T}$ and the averaged class distribution $p(c|\mathbf{u})$. Examples are shown in Fig. 15.1 and Fig. 15.3(b). A pixel-level classification based on the local distributions $P(c|\mathcal{L})$ gives poor but still surprisingly good accuracy (see Sect. 15.6.1). We will shortly describe in Sect. 15.3.3 how the *bag of semantic textons* can pool the statistics of semantic textons $\mathcal{L}$ and distributions $P(c|\mathcal{L})$ over an image region to form a much more powerful feature for image categorization and semantic segmentation.

Examples of the appearance clusters learned in STFs are given in Fig. 15.2(b).

15.3.1 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. To avoid this loss, we augment the training data with image copies that are artificially transformed geometrically and photometrically [214]. This augmentation allows one to *learn* the right degree of invariance required by suitably designing these transformations for a particular problem. In our experiments we explored small rotations and scalings, and left-right flipping as geometric transformations, and affine photometric transformations.

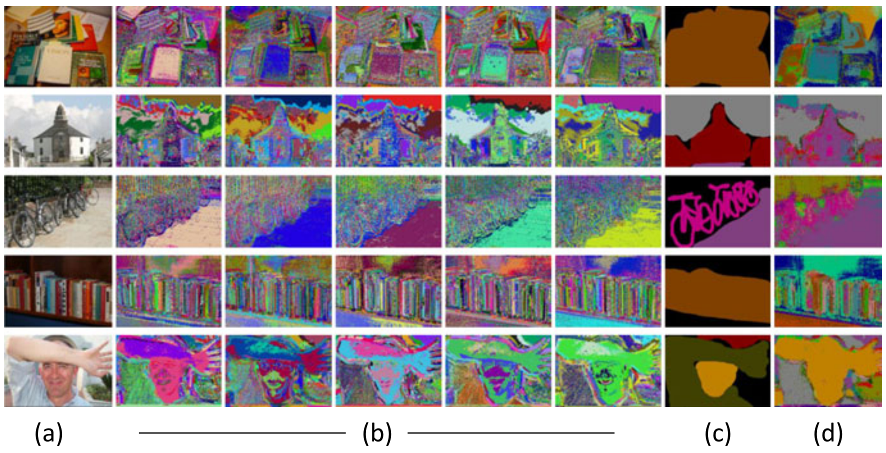


Fig. 15.3 Example semantic textonizations. (**a**) Test image. (**b**) One texton map per tree in the STF. Colors represent tree leaf nodes. (**c**) Ground truth classification. (**d**) Inferred rough local segmentation, showing the most likely class per pixel. Colors in (**c**) and (**d**) represent category labels. Both the textons and the rough segmentation are used as features for whole image categorization and higher-level segmentation. A further example is given in Fig. 15.1

15.3.2 Implementation Details

As discussed in Part I of this book, forests can be trained both in a supervised or unsupervised manner. Similarly, an STF can be trained using (i) pixel-level supervision, (ii) weak supervision, in which the members of the set of classes present in the whole image are used as training labels for all pixels, or (iii) no supervision, where the split function that most evenly divides the data is used. In the unsupervised case, the STF forest acts only as a hierarchical clusterer, not a classifier, similar to the density forests of Chap. 6. We examine the effect of different levels of supervision in Sect. 15.6.

We found the CIELab color space to generalize better than RGB, and it is used in all experiments. Training example pixels are taken on a regular grid (every 5×5 pixels) in the training images, excluding a narrow band of $\frac{\Delta}{2}$ pixels around the image border to avoid artifacts; at test time, the image is extended to ensure a smooth estimate of the semantic textons near the border.

15.3.3 Bags of Semantic Textons

A popular and powerful method for categorizing images and detecting objects is the bag of words model [81, 348, 416]. A histogram of visual words is created over the whole image or a region of interest [67], either discarding spatial layout or using a spatial hierarchy [204]. The histogram is used as input to a classifier

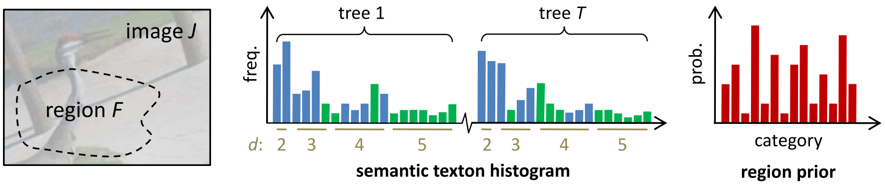


Fig. 15.4 Bags of semantic textons. Within a region F of image J we generate the semantic texton histogram and region prior. The histogram incorporates the implicit hierarchy of clusters in the STF, containing both STF leaf nodes (*green*) and split nodes (*blue*). The depth d of the nodes in the STF 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 class distributions $p_l(c)$

to recognize object categories. We propose the localized bag of semantic textons (BoST), illustrated in Fig. 15.4. This extends the bag of words to be hierarchical and to include low-level semantic information, as follows.

Given the leaf nodes $\mathcal{L}(\mathbf{u}) = \{l_t\}_{t=1}^{T}$ and the inferred class distribution $p(c|\mathbf{u})$ for each pixel $\mathbf{u}$, one can compute the following over an image region F: (i) a non-normalized histogram $G_F(j)$ that concatenates the occurrences of tree nodes j across the different trees [253]; and (ii) a prior over the region given by the average class distribution $p(c|F) = \sum_{\mathbf{u}\in F} p(c|\mathbf{u})$. In contrast to [253], we include both leaf nodes l and split nodes j in the histogram, noting that $G_F(j) = \sum_{j'\in\text{child}(j)} G_F(j')$. The histogram therefore uses the hierarchy of clusters implicit in each tree. Each $p(c|\mathbf{u})$ is already averaged across trees, and hence there is a single region prior $p(c|F)$ for the whole forest.

Our results in Sect. 15.6 show that the histograms and region priors are complementary, and that the hierarchical clusters are better than the leaf node clusters alone. For categorization (Sect. 15.4), we use BoSTs where the region is the whole image. For segmentation (Sect. 15.5), we use a combination of BoSTs over many local rectangular regions to model layout and context.

Implementation Details The counts of tree *root* nodes hold no useful information and are not included in the histograms. The histograms are sparse near the leaves, and can be stored efficiently since the histogram counts at the parent split node can be quickly computed on-the-fly. If the region F is rectangular, the histograms and class distributions can be calculated very efficiently using integral histograms [295, 342].

15.4 Image Categorization

The task of image categorization is to determine those categories (*e.g.* dog images, beach images, indoor images) to which an image belongs. For our purposes, every image belongs to those categories for which there exists a pixel in the image that has been labeled with that category. Thus, an image with a sheep eating grass will belong to both the 'grass' and 'sheep' categories. Example previous approaches have used global image information [276], bags of words [104] or textons [404].

We propose an image categorization algorithm that exploits the hierarchy of semantic textons and the node prior distributions $p_j(c)$. This algorithm uses a non-linear support vector machine (SVM), though of course decision forests could also be used instead. The SVM 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 pyramid match kernel [141] to act on a pair of BoST histograms computed across the whole image.

Consider first the BoST histogram computed for just one tree in the STF. The kernel function (based on [141]) is then

$$K(P,Q) = \frac{1}{\sqrt{Z}}\tilde{K}(P,Q), \tag{15.3}$$

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

$$Z = \tilde{K}(P,P)\tilde{K}(Q,Q). \tag{15.4}$$

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

$$\tilde{K}(P,Q) = \sum_{d=1}^{D} \frac{1}{2^{D-d+1}}(\mathcal{G}_d - \mathcal{G}_{d+1}), \tag{15.5}$$

using the histogram intersection $\mathcal{G}$

$$\mathcal{G}_d = \sum_j \min\bigl(P_d[j], Q_d[j]\bigr). \tag{15.6}$$

In the above, D is the maximum depth of the tree, P and Q are the hierarchical histograms, 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 $\mathcal{G}_{D+1} = 0$. If the tree is not full depth, missing nodes j are simply assigned $P_d[j] = Q_d[j] = 0$.

The kernel over all trees in the STF is calculated as $K = \sum_t \gamma_t K_t$ with mixture weights γ_t. Similarly to [416], we found $\gamma_t = \frac{1}{T}$ to result in the best categorization results. While effective, this kernel can be improved by using the learned 'prior' distributions $p_j(c)$ from the STF. We build a 1-*vs.*-all SVM kernel K_c per category, in which the count for node j in the BoST histogram is weighted by the value $p_j(c)$.[1] This weighting helps balance the categories, by selectively down-weighting those that cover large image areas (*e.g.* grass, water) and thus have inappropriately strong

[1]At training time, we compute and store the distributions $p_j(c)$ for all nodes j in the tree, not just for leaf nodes.

influence on the pyramid match, masking the signal of smaller classes (*e.g.* cat, bird).

In Sect. 15.6.2, we show 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. We also obtain an improvement using the per-category kernels K_c instead of a global kernel K. Finally, we show how this categorization can act as an image-level prior for segmentation in Sect. 15.5.1.

15.5 Semantic Segmentation

To demonstrate the power of the BoSTs as features for segmentation, we adapt the TextonBoost algorithm [342]. The goal is to segment an image into coherent regions and simultaneously infer the class label of each region (see Sect. 15.6.3.1).

Appearance Context *vs.* Semantic Context In [342], a boosting algorithm selected features based on localized counts of textons to model patterns of texture, layout, and context. The context modeled in [342] was appearance-based, for example: sheep often stand on something green. We adapt the rectangle count features of [342] 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* [303], not just texture. Continuing the example, our model can capture the notion that sheep often stand on *grass*. This concept of basing the output of one classifier as the input to another was proposed concurrently by [341] (the original version of this chapter) and [375]. The related idea of entanglement is explored in Chap. 19.

The segmentation algorithm works as follows. For speed we use a second classification forest in place of the boosting classifier used by [342]. We train this forest to act at image pixels $\mathbf{u}$, using pixels on a regular grid as training examples. At test time, the segmentation forest is applied at each pixel $\mathbf{u}$ densely or, for more speed, on a grid. The most likely class in the averaged category distribution (15.1) gives the final segmentation for each pixel. The split node functions f now compute either the count $G_{F+\mathbf{u}}(j)$ of semantic texton j, or the probability $p(c \mid F + \mathbf{u})$ of class c, within rectangle F translated relative to pixel $\mathbf{u}$. By translating rectangle F relative to the pixel $\mathbf{u}$ being classified, and uniformly sampling rectangles F within a box offset from $\mathbf{u}$ by up to half the image size, such features can exploit texture, layout and context information (see [342] for more details). Our extension to these features exploits semantic context by using the region prior probabilities $p(c|F+\mathbf{u})$ inferred by the semantic textons. We show the benefit this brings in Sect. 15.6.3.

15.5.1 Image-Level Prior

We could embed the above segmentation forest in a conditional random field model to achieve more coherent results or to refine the grid segmentation to a per-pixel segmentation [157, 342]. Instead, we decided to investigate a simpler and more efficient approach using the image categorizer we built in Sect. 15.4. For each test image we separately run the categorization and segmentation algorithms. This gives an image-level prior (ILP) distribution $p(c)$ and a per-pixel segmentation distribution $p(c|\mathbf{u})$ respectively. We use the ILP to emphasize the likely categories and discourage unlikely categories, by multiplying the somewhat independent distributions as $p'(c|\mathbf{u}) = p(c|\mathbf{u})p(c)^{\alpha}$, using parameter α to soften the prior. We show in Sect. 15.6.3 and Sect. 15.6.3.1 how the addition of the ILP gives a considerable improvement to the resulting segmentations. Li and Fei-Fei [219] proposed a related idea that uses scene categorization as priors for object detection.

15.6 Experiments

We performed experiments on the following two datasets:

	# classes	# training images	# test images
MSRC [342]	21	276	256
VOC 2007 (Seg) [99]	21	422	210

We use the standard train/test splits where available, and the hand-labeled ground truth to train the classifiers. Image categorization accuracy is measured as mean average precision [99]. Segmentation accuracy is measured as the category average accuracy (the average proportion of pixels correct in each category). We also report the global accuracy (total proportion of pixels correct), but note that the category average is fairer and more rigorous as it normalizes for category bias in the test set. Training and test times are reported using an unoptimized C# implementation on a single 2.7 GHz core.

15.6.1 Learning the Semantic Texton Forest Vocabulary

Before presenting in-depth results for categorization and segmentation, let us look briefly at the STFs themselves. In Figs. 15.1 and 15.3 we visualize the inferred leaf nodes $\mathcal{L}(\mathbf{u})$ for each pixel $\mathbf{u}$ and the most likely category $c^{\star}(\mathbf{u}) = \arg\max_c p(c|\mathbf{u})$. Observe that the textons in each tree capture different aspects of the underlying texture and that even at such a low level the distribution $p(c|\mathbf{u})$ contains significant semantic information. Table 15.1 gives a naïve segmentation baseline on the MSRC dataset by comparing $c^{\star}(\mathbf{u})$ to the ground truth, with either fully or weakly supervised training pixels (see Sect. 15.3.2).

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 un-

Table 15.1 Naïve segmentation baseline on MSRC

	Global	Average
supervised	49.7 %	34.5 %
weakly supervised	14.8 %	24.1 %

Table 15.2 Image categorization results. (Mean AP)

	Global kernel K	Per-category kernel K_c
RBF	49.9	52.5
PMK	76.3	**78.3**

der 5 %). We show below 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, we used STFs with the following parameters, hand-optimized on the MSRC validation set: distance $\Delta = 21$, $T = 5$ trees, maximum depth $D = 10$, 500 feature tests and 10 threshold tests per split node, and bagging using $\frac{1}{4}$ of the data per tree, resulting in approximately 500 leaves per tree. Training the STF on the MSRC dataset took only 15 minutes.

15.6.2 Image Categorization

We performed an experiment on the MSRC data to investigate our SVM categorization algorithm. The mean average precisions (AP) in Table 15.2 compare our 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 ('tf/idf', from standard information retrieval) 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 $p_j(c)$. As can be seen, the pyramid match kernel considerably improves on the RBF kernel. By training a per-category kernel, a small but noticeable improvement is obtained. For the image-level prior for segmentation, we thus use the PMK with per-category kernels. In Fig. 15.5 we plot the global kernel accuracy against the number T of STF trees, and see that categorization accuracy increases with more trees though it eventually levels out.

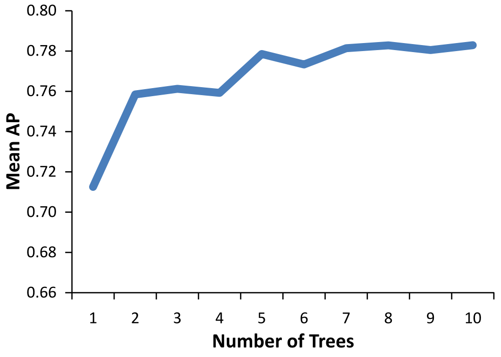


Fig. 15.5 Categorization accuracy *vs.* number of STF trees

Further categorization experiments are provided in Tables 15.3 and 15.4.

15.6.3 Semantic Segmentation

15.6.3.1 Experiments on the MSRC Dataset [342]

We first examine the influence of different aspects of our system on segmentation accuracy. We trained segmentation forests using (a) the histogram $G_F(l)$ of just leaf nodes l; (b) the histogram $G_F(j)$ of *all* tree nodes j; (c) just the region priors $p(c|F)$; (d) the full model using all nodes and region priors; (e) the full model trained without random geometric/photometric 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 15.5 with and without the image-level prior (ILP).

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),

Table 15.3 MSRC categorization results. The values shown are the average precision (AP) and the area under the ROC curve (AuC) for the MSRC21 dataset

	building	grass	tree	cow	sheep	sky	airplane	water	face	car	bicycle	flower	sign	bird	book	chair	road	cat	dog	body	boat	**Mean**
AP	71	92	81	91	97	95	97	86	83	87	79	85	47	43	89	61	77	62	60	85	76	**78**
AuC	85	96	94	99	100	98	100	94	95	99	98	99	88	86	99	93	92	95	94	97	94	**95**

Table 15.4 VOC2007 categorization results. The values shown are the average precision (AP) and the area under the ROC curve (AuC) for the VOC2007 segmentation dataset

	aeroplane	bicycle	bird	boat	bottle	bus	car	cat	chair	cow	table	dog	horse	motorbike	person	plant	sheep	sofa	train	tv/monitor	**Mean**
AP	69	15	11	26	11	24	35	12	15	18	22	15	33	40	67	29	19	10	55	16	**27**
AuC	93	67	70	68	71	73	79	72	77	66	80	72	88	83	72	78	80	79	92	75	**77**

Table 15.5 Comparative segmentation results on MSRC

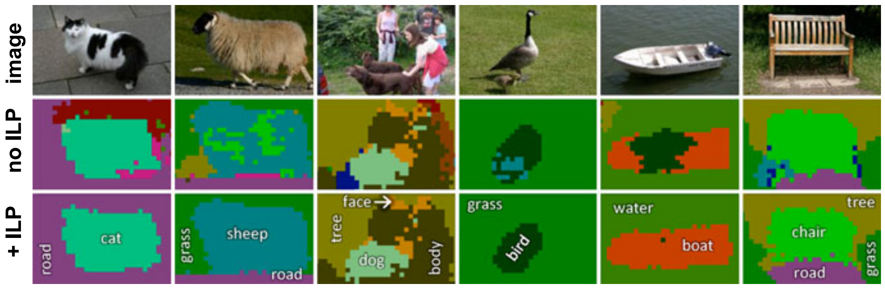
	Without ILP	With ILP
(a) only leaves	61.3 %	64.1 %
(b) all nodes	**63.5 %**	65.5 %
(c) only region priors	62.1 %	66.1 %
(d) **full model**	63.4 %	**66.9 %**
(e) no transformations	60.4 %	64.4 %
(f) unsupervised STF	59.5 %	64.2 %
(g) weakly supervised STF	61.6 %	64.6 %

adding in the ILP shows how the region priors and textons work together.[2] (5) Random transformations of the training images improve accuracy by adding invariance. (6) Accuracy increases with more supervision, but even unsupervised STFs allow good segmentations.

Given this insight, we compare against [342] and [384]. We use the same train/test split as [342] (though not [384]). The results are summarized in Fig. 15.6 with further examples given in Fig. 15.9. Across the whole challenging dataset, using the full model with ILP achieved a class average accuracy of 66.9 %, a considerable improvement on both the 57.7 % of [342] and the 64 % of [384]. The global accuracy also improves slightly on [342]. The image-level prior improves accuracy for all but three classes, but even without it, results are still highly competitive with respect to other methods. Our use of balanced training has resulted in more consistent accuracy across classes, and significant improvements for certain difficult classes: cow, sheep, bird, chair, and cat. We do not use a Markov or conditional random field, which would likely further improve our accuracy [342].

These results used our learned and extremely fast STFs, without needing any hand-designed filter-banks or descriptors that are potentially slow to compute. Extracting the semantic textons at every pixel takes an average of only 275 millisec-

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



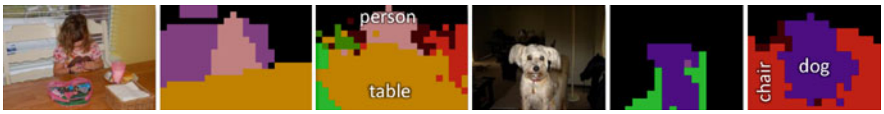
	building	grass	tree	cow	sheep	sky	airplane	water	face	car	bicycle	flower	sign	bird	book	chair	road	cat	dog	body	boat	**Global**	**Average**
[343]	**62**	**98**	**86**	58	50	83	60	53	74	63	**75**	63	35	19	92	15	86	54	19	62	7	71	58
[382]	52	87	68	73	84	**94**	**88**	**73**	70	68	74	**89**	33	19	78	34	**89**	46	**49**	54	**31**	-	64
Ours	41	84	75	89	93	79	86	47	**87**	65	72	61	**36**	26	91	50	70	72	31	61	14	68	63
Ours + ILP	49	88	79	**97**	**97**	78	82	54	**87**	**74**	72	74	**36**	24	**93**	**51**	78	**75**	35	**66**	18	**72**	**67**

Fig. 15.6 MSRC segmentation results. *Above*: 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. Further examples, including failure cases, in Fig. 15.9. (Note we do not use a Markov or conditional random field which could clean up the segmentations to precisely follow image edges [342]). *Below*: Segmentation accuracies (percent) over the whole dataset, without and with the ILP. Our highly efficient semantic textons achieve a significant improvement on previous work

onds per image, categorization takes 190 ms, and evaluating the segmentation forest only 140 ms. For comparison [342] took over 6 seconds per test image, and [384] took an average of over 2 seconds per image for feature extraction and between 0.3 to 2 seconds for estimating the segmentation. Our algorithm is well over 5 times faster *and* improves quantitative results. Minor optimizations have subsequently led to a real-time system that runs at over 8 frames per second.

15.6.3.2 Experiments on the VOC 2007 Segmentation Dataset [99]

This dataset contains 21 challenging categories including background. We trained a STF, a segmentation forest, and an ILP on these data, using the 'trainval' split and keeping parameters as for MSRC. The results in Fig. 15.7 compare with [99]. Our algorithm performs over twice as well as the only segmentation entry (Brookes), and the addition of the ILP further improves accuracy 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) we took the TKK entry output as the DLP. Our algorithm gave a 12 % improvement over the TKK segmentation by detection, highlighting the power of STFs as features for segmentation.



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

Fig. 15.7 VOC 2007 segmentation results. *Above*: Test images with ground truth and our inferred segmentations using the ILP (not the DLP). *Below*: Segmentation accuracies (percent) over the whole dataset. The top three results compare our method to the Brookes segmentation entry [99], and show that our method is over twice as accurate. The lower two results compare the best automatic segmentation-by-detection entry (see text) [99] with our algorithm using the TKK results as a detection-level prior (DLP). Our algorithm improves the accuracy of segmentation by detection by over 10 percentage points

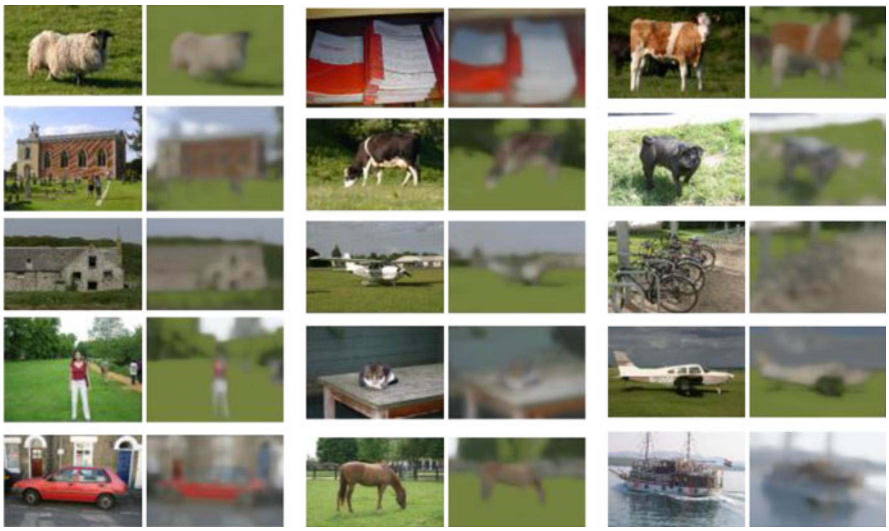


Fig. 15.8 Reconstruction from texton maps. By simply averaging the patches in Fig. 15.2(b) according to the texton maps in Fig. 15.3 one gets a blurry reconstruction of the original images. These reconstructions show that our semantic textons discretely capture significant image texture

Since the original publication of this work [341], there has been considerable progress in the field. Please see the latest PASCAL VOC challenge for the state of the art.

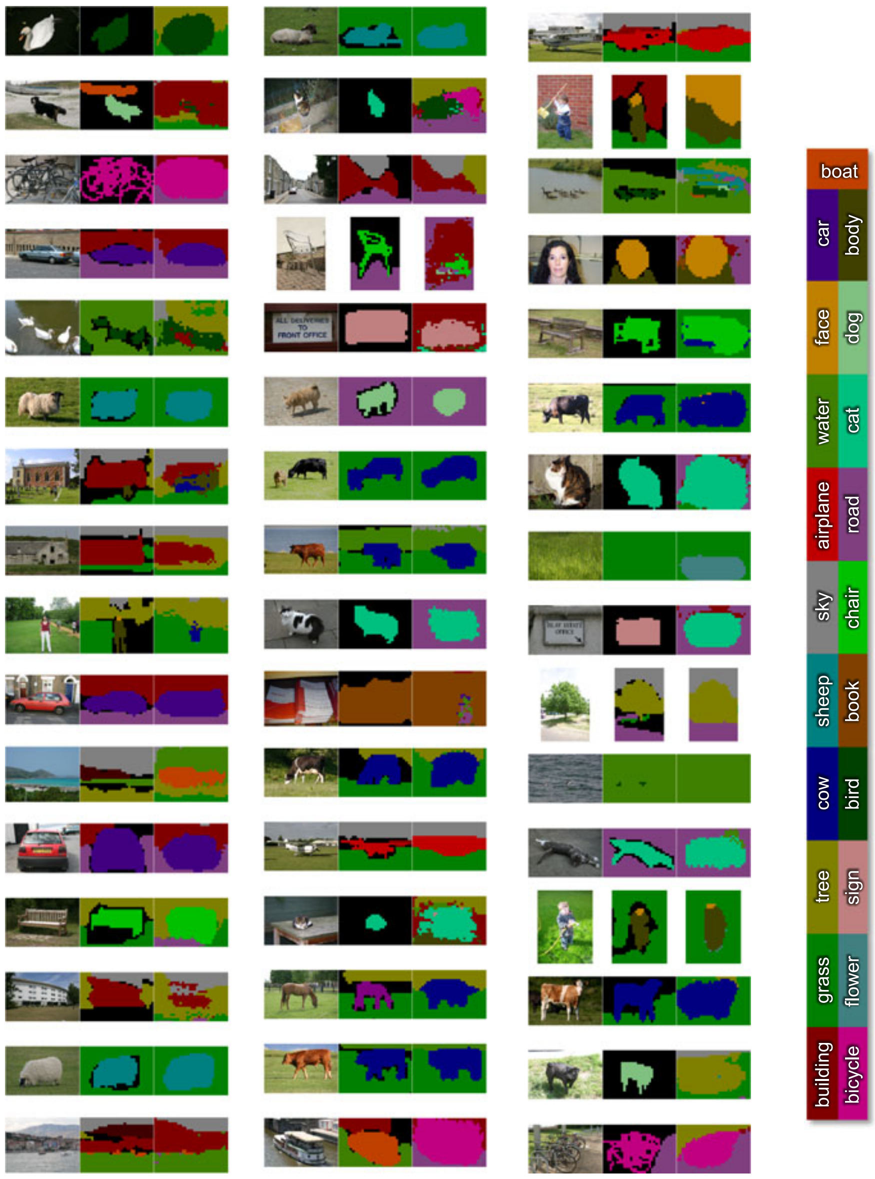


Fig. 15.9 A random selection of results on the MSRC dataset, including successes and failures. *Left*: Test image. *Middle*: Ground truth. *Right*: Our result. Image-level prior is used. Black pixels in the ground truth are 'void' and not used for evaluation

15.7 Conclusions

This chapter presented semantic texton forests as an efficient method for encoding textons. STFs do not depend on local descriptors or expensive k-means clustering, and when supervised during training they can infer a distribution over categories at each pixel. We showed how bags of semantic textons enabled state-of-the-art accuracy on challenging datasets for image categorization and semantic segmentation, and how the use of an inferred image-level prior significantly improves segmentation results. The substantial gains of our method over traditional textons are training and testing efficiency and improved quantitative performance.

The main limitation of our system is the large dimensionality of the bag of semantic textons, which necessitates a trade-off between the memory usage of the semantic texton integral images and the training time if they are computed at runtime. However, using just the region priors is more memory efficient.

As future work, it would be interesting to investigate how STFs might be used for image reconstruction. A few examples of a simple experiment are given in Fig. 15.8.

Acknowledgements We would like to thank J. Winn, B. Wenger, O. Yamaguchi, and V. Vitaniemi for helpful conversations and insights contributing to the work in this paper.