TextonBoost for Image Understanding: Multi-Class Object Recognition and Segmentation by Jointly Modeling Texture, Layout, and Context

Jamie Shotton*
Machine Intelligence Laboratory, University of Cambridge
jamie@shotton.org

John Winn, Carsten Rother, Antonio Criminisi
Microsoft Research Cambridge, UK
[jwinn,carrot,antcrim]@microsoft.com

July 2, 2007

Abstract

This paper details a new approach for learning a discriminative model of object classes, incorporating texture, layout, and context information efficiently. The learned model is used for automatic visual understanding and semantic segmentation of photographs. Our discriminative model exploits *texture-layout filters*, novel features based on textons, which jointly model patterns of texture and their spatial layout. Unary classification and feature selection is achieved using shared boosting to give an efficient classifier which can be applied to a large number of classes. Accurate image segmentation is achieved by incorporating the unary classifier in a conditional random field, which (i) captures the spatial interactions between class labels of neighboring pixels, and (ii) improves the segmentation of specific object instances. Efficient training of the model on large datasets is achieved by exploiting both random feature selection and piecewise training methods.

High classification and segmentation accuracy is demonstrated on four varied databases: (i) the MSRC 21-class database containing photographs of real objects viewed under general lighting conditions, poses and viewpoints, (ii) the 7-class Corel subset and (iii) the 7-class Sowerby database used in [19], and (iv) a set of video sequences of television shows. The proposed algorithm gives competitive and visually pleasing results for objects that are highly textured (grass, trees, etc.), highly structured (cars, faces, bicycles, airplanes, etc.), and even articulated (body, cow, etc.).

*Now working at at Toshiba Corporate Research & Development Center, Kawasaki, Japan.

1 Introduction

This paper investigates the problem of achieving automatic detection, recognition, and segmentation of object classes in photographs. Precisely, given an image, the system should automatically partition it into semantically meaningful regions each labeled with a specific object class, as illustrated in Figure 1.

The challenge is to model the visual variability of a large number of both structured and unstructured object classes, to be invariant to viewpoint and illu-

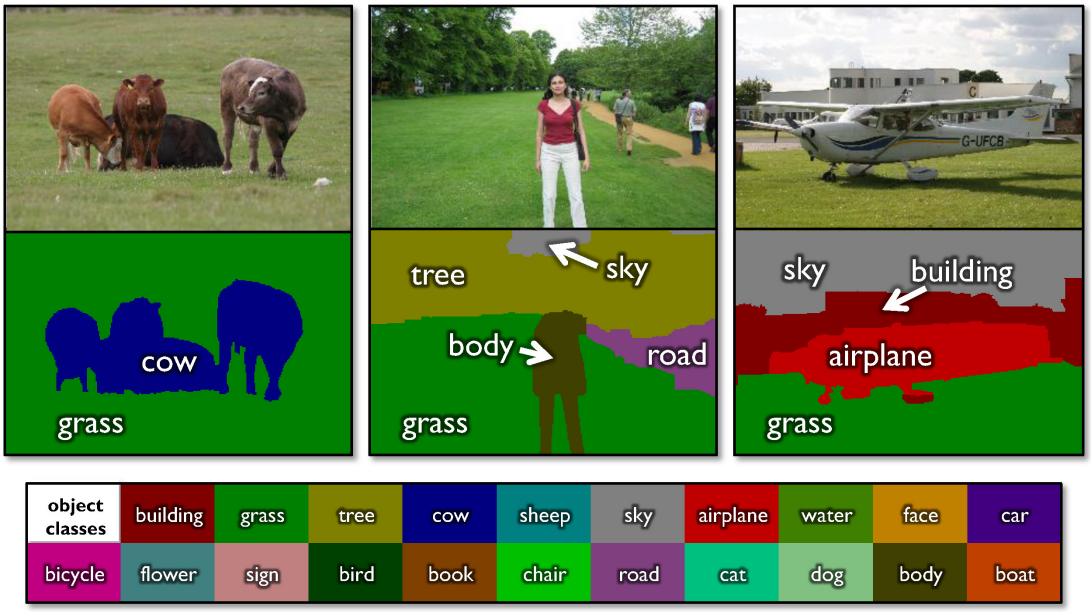

Figure 1: **Example results of our new simultaneous object class recognition and segmentation algorithm.** Up to 21 object classes (color-coded in the key) are recognized, and the corresponding object instances segmented in the images. For clarity, textual labels have been superimposed on the resulting segmentations. Note, for instance, how the airplane has been correctly recognized and separated from the building, the sky, and the grass lawn. In these experiments only one *single* learned multi-class model has been used to segment all the test images. Further results from this system are given in Figure 18.

mination, and to be robust to occlusion. Our focus is not only the accuracy of segmentation and recognition, but also the efficiency of the algorithm, which becomes particularly important when dealing with large image collections or video sequences.

At a local level, the *appearance* of an image patch leads to ambiguities in its class label. For example, a window could be part of a car, a building or an airplane. To overcome these ambiguities, it is necessary to incorporate longer range information such as the spatial *layout* of an object and also *contextual* information from the surrounding image. To achieve this, we construct a discriminative model for labeling images which exploits all three types of information: textural appearance, layout, and context. Our technique can model very long-range contextual relationships extending over half the size of the image.

Additionally, our technique overcomes several problems typically associated with object recognition techniques that rely on sparse features (such as [33, 36]). These problems are mainly related to textureless or very highly textured image regions. Figure 2 shows some examples of images with which those techniques would very likely struggle. In contrast, our technique based on dense features is capable of coping with both textured and untextured objects, and with multiple objects which inter- or self-occlude, while retaining high efficiency.

The main contributions in this paper are threefold. The most significant is a novel type of feature, which we call the texture-layout filter. These features record patterns of textons, and exploit the textural appearance of the object, its layout, and its textural context. Our second contribution is a new discriminative model that combines texture-layout filters with lower-level image features, in order to pro-

Figure 2: **Example images for which sparse features are insufficient.** Techniques based on sparse features struggle with textureless and very highly textured regions, and multiple objects, especially those that severely inter-occlude.

vide a near pixel-perfect segmentation of the image. Finally, we demonstrate how to train this model efficiently on a very large dataset by exploiting both boosting and piecewise training methods.

To stimulate further research and the development of new applications, source code has been made publicly available.[1] Parts of this paper originally appeared as [43], but note that some terms have been renamed: 'shape filters' are now called 'texture-layout filters', and 'shape-texture potentials' are now called 'texture-layout potentials'. The new names are more intuitive, and we encourage their use.

The paper is organized as follows. Immediately below, we discuss related work. In Section 2, we describe the image databases used in our experiments. Section 3 introduces the high-level discriminative model, a conditional random field (CRF). Readers most interested in the powerful texture-layout filters may jump to Section 4, where we also discuss their combination in a boosted classifier. We evaluate and compare with related work in Section 5, suggest some novel applications of semantic segmentation in Section 6, and conclude in Section 7.

1.1 Related Work

Whilst the fields of object recognition and segmentation have been extremely active in recent years, many authors have considered these two tasks separately. For example, recognition of particular object classes has been achieved using the constellation models of Fergus *et al.* [17], the deformable shape models of Berg *et al.* [6] and the texture models of Winn *et al.* [49]. None of these methods leads to a pixel-wise segmentation of the image. Conversely, other authors have considered only the segmentation task, for example [28, 9]. In [39], image regions are classified into figure and ground using a conditional random field model. Our technique goes further, providing a full image segmentation and recognizing the object class of each resulting region.

Joint detection and segmentation of a *single* object class has been achieved by several authors [50, 27, 31]. Typically, these approaches exploit a global layout model and are therefore unable to cope with arbitrary viewpoints or severe occlusion. Additionally, only highly structured object classes are addressed.

Several authors such as [16, 45, 30] have considered recognition for up to 101 object classes, but these techniques only address image classification and object localization in fairly constrained images. In contrast, our technique achieves both recognition and segmentation on natural scenes, and copes accurately with 21 classes, which to our knowledge is state of the art for this task.

In [14], a classifier, trained from images associated with textual class labels, was used to label regions found by bottom-up segmentation. However, such segmentations often do not correlate with semantic objects, and so our proposed solution performs segmentation and recognition in the same unified framework rather than in two separate steps. Such a unified approach was presented in [46], but only text and faces were recognized, and at a high computational cost. Konishi and Yuille [26] labeled images using only a unary classifier, and hence did not achieve spatially coherent segmentations.

The most similar work to ours by He *et al.* [19] incorporates region and global label features to model layout and context in a conditional random field. Their work uses Gibbs sampling for both the parameter learning and label inference and is therefore limited in the size of dataset and number of classes which can be handled efficiently. Our focus on the speed of training and inference allows the use of larger datasets with many more object classes: up to 21 classes in our evaluation, compared to 7 classes in [19].

[1] `http://jamie.shotton.org/work/code.html`

More recent work by the same group presented a related technique [20], where images are first segmented with a bottom-up algorithm to produce 'super-pixels' which are then merged together and semantically labeled using a combination of several scene-specific CRF models. Their technique improves the quantitative results from [19], but is not demonstrated on more than 11 classes. Additionally, their model cannot recover from mistakes where the inferred super-pixels cross a semantic object boundary, such as where a dark object meets its shadow. These problems, while perhaps infrequent, are side-stepped by our technique which instead works efficiently at a per-pixel level.

Building on TextonBoost is a system called 'LayoutCRF' by Winn & Shotton [51]. This incorporates an additional constraint to recognize configurations of object parts and hence individual instances of structured objects whilst allowing for deformation and occlusion.

2 Image Databases

Our object class models are learned from a set of labeled training images. In this paper we consider four different labeled image databases. The Microsoft Research Cambridge (MSRC) database[2] is composed of 591 photographs of the following 21 object classes: building, grass, tree, cow, sheep, sky, airplane, water, face, car, bicycle, flower, sign, bird, book, chair, road, cat, dog, body, and boat. Examples are shown in Figure 3. The training images were hand-labeled by means of a 'paint' interface, with the assigned colors acting as indices into the list of object classes. One could instead use one of the novel 'user-centric computation' methods such as [42] or Peekaboom[3]. Note that we consider general lighting conditions, camera viewpoint, scene geometry, object pose and articulation. The database is split randomly into roughly 45% training, 10% validation and 45% test sets, while ensuring approximately proportional contributions from each class.

[2] `http://research.microsoft.com/vision/cambridge/recognition/`

[3] `http://www.peekaboom.org/`

Note that the ground truth labeling of the 21-class database contains pixels labeled as 'void'. This label was used both to cope with pixels that do not belong to a class in the database, and also to allow for a rough and quick hand-segmentation which need not align exactly with the object boundaries. Due to this semantic ambiguity, it was not sensible to learn a background class based on these regions, and hence void pixels are ignored for both training and testing.

For comparison with previous work [19], we also used the 7-class Corel database subset (where images are 180×120 pixels) and the 7-class Sowerby database (96×64 pixels). For those two databases, the numbers of images in the training and test sets we used are exactly as for [19], although their precise train-test split was not known. Neither of these data sets include the void label.

The final evaluation we present was performed on a set of nine 20-minute video sequences of television programs: modern drama, news, golf, soccer, cooking, variety, music, historical drama, and business news. The set of classes used for this evaluation was as for the MSRC evaluation, though with sign, book, and chair removed, and the new classes hand, table, and headgear added. For speed of evaluation, video frames were down-sampled to 336x224 pixel resolution, but were otherwise not preprocessed. A total of about 120 frames (one every 300 frames) in each sequence were labeled by hand for training and evaluation.

Real scenes contain many more object classes than the 21 evaluated in this paper. It proved impractical to hand-label more classes for evaluation, but we believe TextonBoost could readily be extended to many more classes without major algorithmic changes. In our conclusions, we discuss this extensibility in more depth.

3 A Conditional Random Field Model of Object Classes

We use a conditional random field (CRF) model [29] to learn the conditional distribution over the class labeling given an image. The use of a conditional

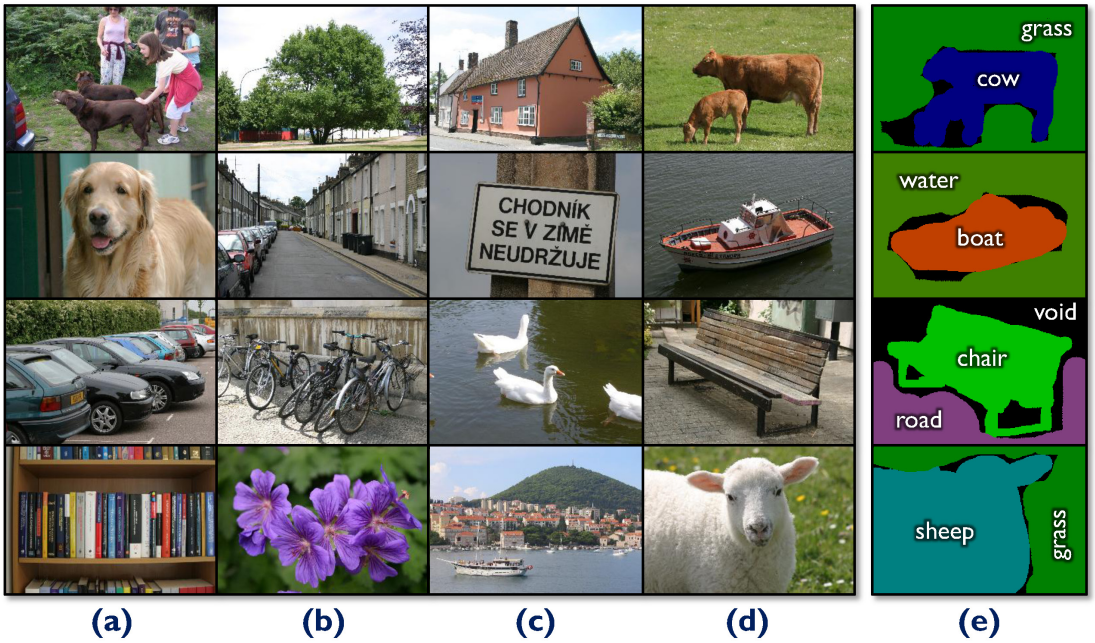

Figure 3: **The MSRC labeled image database. (a-d)** A selection of images in the 21-class database. **(e)** The ground truth annotations corresponding to column (d), where each color maps uniquely to an object class label. All images are approximately 320×240 pixels.

random field allows us to incorporate texture, layout, color, location, and edge cues in a single unified model. We define the conditional probability of the class labels $\mathbf{c}$ given an image $\mathbf{x}$ as

$$\log P(\mathbf{c}|\mathbf{x}, \boldsymbol{\theta}) =$$

$$\sum_i \overbrace{\psi_i(c_i, \mathbf{x}; \boldsymbol{\theta}_\psi)}^{\text{texture-layout}} + \overbrace{\pi(c_i, x_i; \boldsymbol{\theta}_\pi)}^{\text{color}} + \overbrace{\lambda(c_i, i; \boldsymbol{\theta}_\lambda)}^{\text{location}}$$

$$+ \sum_{(i,j)\in\mathcal{E}} \overbrace{\phi(c_i, c_j, \mathbf{g}_{ij}(\mathbf{x}); \boldsymbol{\theta}_\phi)}^{\text{edge}} - \log Z(\boldsymbol{\theta}, \mathbf{x}) \quad (1)$$

where $\mathcal{E}$ is the set of edges in a 4-connected grid structure, $Z(\boldsymbol{\theta}, \mathbf{x})$ is the partition function which normalizes the distribution, $\boldsymbol{\theta} = \{\boldsymbol{\theta}_\psi, \boldsymbol{\theta}_\pi, \boldsymbol{\theta}_\lambda, \boldsymbol{\theta}_\phi\}$ are the model parameters, and i and j index pixels in the image, which correspond to sites in the graph. Note that our model consists of three *unary* potentials which depend only on one node i in the graph, and one *pairwise* potential depending on pairs of neighboring nodes in the graph. We next define the form of the four potential functions and their parameters, before detailing inference and learning.

Texture-layout potentials: The unary texture-layout potentials ψ are defined as

$$\psi_i(c_i, \mathbf{x}; \boldsymbol{\theta}_\psi) = \log P(c_i|\mathbf{x}, i) \ , \quad (2)$$

where $P(c_i|\mathbf{x}, i)$ is the normalized distribution given by a boosted classifier. This classifier models the texture, layout, and textural context of the object classes, by combining novel discriminative features called texture-layout filters. These features and the boosted classifier are detailed in Section 4. As we show in the evaluation, the texture-layout potentials ψ prove to be the most powerful term in the CRF.

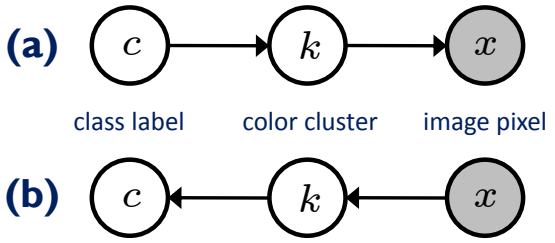

Figure 4: **Graphical model of the color potentials.** **(a)** The color models use Gaussian Mixture Models where the mixture coefficients $P(k|c)$ are conditioned on the class label c. **(b)** For discriminative inference, the arrows in the graphical model are reversed using Bayes rule.

Color potentials: The unary color potentials exploit the color distribution of objects in a *particular image*. Individual objects tend to have a relatively compact color distribution, and exploiting an accurate instance color model can therefore dramatically improve segmentation results [8]. Note that the texture-layout potentials *also* implicitly use color information, but, in contrast, model the much broader color distribution across the whole class.

Our color models are represented as Gaussian Mixture Models (GMMs) in CIELab color space where the mixture coefficients depend on the class label. As illustrated in the graphical model of Figure 4, the conditional probability of the color x of a pixel is given by

$$P(x|c) = \sum_k P(x|k)P(k|c) \tag{3}$$

with color clusters (mixture components)

$$P(x|k) = \mathcal{N}(x \,|\, \mu_k, \Sigma_k) \tag{4}$$

where μ_k and Σ_k are the mean and variance respectively of color cluster k. Notice that the clusters are shared between different classes, and that only the coefficients $P(k|c)$ depend on the class label, making the model more efficient to learn than a separate GMM for each class.

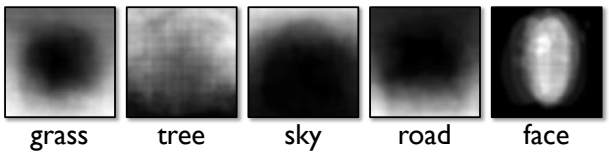

Figure 5: **Example location potentials.** Note how, for example, tree and sky pixels tend to occur in the upper half of images, while road pixels tends to occur at the bottom of the image. (White indicates increased frequency).

For pixel x_i, a soft assignment[4] to the color clusters $P(k|x_i) \propto P(x_i|k)$ is computed using a uniform prior $P(k)$, and used in the color potentials, which we define as

$$\pi(c_i, x_i; \boldsymbol{\theta}_\pi) = \log \sum_k \boldsymbol{\theta}_\pi(c_i, k) P(k|x_i) \ . \tag{5}$$

Learned parameters $\boldsymbol{\theta}_\pi(c_i, k)$ represent the distribution $P(c_i|k)$. We discuss learning the color clusters (4) and parameters $\boldsymbol{\theta}_\pi$ in Section 3.2.3.

Location potentials: The unary location potentials capture the (relatively weak) dependence of the class label on the absolute location of the pixel in the image. The potential takes the form of a look-up table with an entry for each class and pixel location:

$$\lambda_i(c_i, i; \boldsymbol{\theta}_\lambda) = \log \boldsymbol{\theta}_\lambda(c_i, \hat{i}) \ . \tag{6}$$

The index $\hat{i}$ is the normalized version of the pixel index i, where the normalization allows for images of different sizes: the image is mapped onto a canonical square and $\hat{i}$ indicates the pixel position within this square. Some learned location potentials are illustrated in Figure 5.

Edge potentials: The pairwise edge potentials ϕ have the form of a contrast sensitive Potts model [10],

$$\phi(c_i, c_j, \mathbf{g}_{ij}(\mathbf{x}); \boldsymbol{\theta}_\phi) = -\boldsymbol{\theta}_\phi^T \mathbf{g}_{ij}(\mathbf{x})[c_i \neq c_j] \ , \tag{7}$$

[4] A soft color cluster assignment gave a marginal improvement over a hard assignment, at negligible extra cost.

Figure 6: **Edge potentials for an example image.** The edge potentials in the CRF explicitly penalize neighboring nodes in the graph having different class labels, *except* where there is a corresponding edge in the image. Darker pixels in this image represent stronger edge responses and therefore lower costs.

with $[\cdot]$ the zero-one indicator function. In this work, the edge feature $\mathbf{g}_{ij}$ measures the difference in color between the neighboring pixels, as suggested by [41],

$$\mathbf{g}_{ij} = \begin{bmatrix} \exp(-\beta\|x_i - x_j\|^2) \\ 1 \end{bmatrix} \tag{8}$$

where x_i and x_j are three-dimensional vectors representing the colors of pixels i and j respectively. Including the unit element allows a bias to be learned, to remove small, isolated regions. The quantity β is an image-dependent contrast term, and is set separately for each image to $(2\langle\|x_i - x_j\|^2\rangle)^{-1}$, where $\langle\cdot\rangle$ denotes an average over the image. An example is shown in Figure 6.

We next describe inference in our CRF model, and then detail in Section 3.2 how the model parameters are learned.

3.1 Inference in the CRF Model

Given the CRF model and its learned parameters, we wish to find the most probable labeling $\mathbf{c}^\star$, i.e. the labeling that maximizes the conditional probability of (1). The optimal labeling is found by applying the alpha-expansion graph-cut algorithm of [10].[5]

[5]Our energy is alpha-expansion *submodular* [25].

The idea of the expansion move algorithm is to reduce the problem of maximizing a function $f(\mathbf{c})$ (corresponding to (1)) with *multiple* labels to a sequence of *binary* maximization problems. These subproblems are called 'alpha-expansions', and can be described as follows (see [10] for details). Suppose that we have a current configuration (set of labels) $\mathbf{c}$ and a fixed label $\alpha \in \{1, \ldots, C\}$, where C is the number of classes. In the alpha-expansion operation, each pixel i makes a binary decision: it can either keep its old label or switch to label α. Therefore, we introduce a binary vector $\mathbf{s} \in \{0,1\}^{\mathcal{P}}$ which defines the auxiliary configuration $\mathbf{c}[\mathbf{s}]$ as

$$c_i[\mathbf{s}] = \begin{cases} c_i & \text{if } s_i = 0 \\ \alpha & \text{if } s_i = 1. \end{cases} \tag{9}$$

This auxiliary configuration $\mathbf{c}[\mathbf{s}]$ transforms the function f with multiple labels into a function of binary variables $f'(\mathbf{s}) = f(\mathbf{c}[\mathbf{s}])$. Because our edge potentials are attractive, the global maximum of this binary function can be found exactly using graph cuts.

The expansion move algorithm starts with an initial configuration $\mathbf{c}^0$, given by the mode of the texture-layout potentials. It then computes optimal alpha-expansion moves for labels α in some order, accepting the moves only if they increase the objective function.[6] The algorithm is guaranteed to converge, and its output is a strong local maximum, characterized by the property that no further alpha-expansion can increase the function f.

3.2 Learning the CRF

Learning the numerous CRF parameters proved difficult. As described next, MAP training was at first attempted. Poor results, however, encouraged a more pragmatic approach based on piecewise training, justified in Section 3.2.2. The final learning solution is given in Section 3.2.3.

3.2.1 MAP Training

Ideally, the parameters of the model should be learned using maximum *a-posteriori* (MAP) learn-

[6]The final MAP solution was not in practice sensitive to the initial configuration or to the order of expansion moves.

ing [28]. This maximizes the conditional likelihood of the labels given the training data,

$$L(\boldsymbol{\theta}) = \sum_n \log P(\mathbf{c}_n|\mathbf{x}_n, \boldsymbol{\theta}) + \log P(\boldsymbol{\theta}) \ , \qquad (10)$$

incorporating a prior $P(\boldsymbol{\theta})$ to prevent overfitting. The maximization of $L(\boldsymbol{\theta})$ with respect to $\boldsymbol{\theta}$ can be achieved using a gradient ascent algorithm, iteratively computing the gradient of the conditional likelihood with respect to each parameter, $\frac{\partial}{\partial \theta_i} L(\boldsymbol{\theta})$, and moving up the gradient. This requires the evaluation of marginal probabilities over the class labels at each pixel for all training images.

Exact computation of these marginals is sadly infeasible, since it would require vast computational effort to perform the numerous marginalizations of (1). Instead, we approximated the label marginals by the mode, the most probable labeling, inferred using alpha-expansion graph cuts as described above. Alternative, slower but more accurate approximations such as loopy belief propagation (BP) [37, 52] or variational methods [3] could also be investigated.

We attempted MAP learning of the several thousand texture-layout potential parameters $\boldsymbol{\theta}_\psi$ and the two edge potential parameters $\boldsymbol{\theta}_\phi$. For the $\boldsymbol{\theta}_\psi$, the optimization was performed over the a and b parameters of the weak learners in (18), initialized at the values given by boosting.

However, the modal approximation used proved insufficient for estimating such a large number of parameters. Conjugate gradient ascent did eventually converge to a solution, but evaluating the learned parameters against validation data gave poor results with almost no improvement on unary classification alone. Additionally, for the learning of the edge potential parameters, the lack of alignment between object edges and label boundaries in the roughly labeled training set forced the learned parameters to tend toward zero.

3.2.2 Piecewise Training

Given these problems with directly maximizing the conditional likelihood, we turned to a more pragmatic solution, based on the *piecewise training* method of [44]. The terms in the CRF are first trained separately, and then recombined with powers (weighting functions) that alleviate problems due to overcounting.

Piecewise training involves dividing the CRF model into pieces corresponding to the different terms in (1). Each of these pieces is then trained independently, as if it were the only term in the conditional model. For example, if we apply piecewise training to the CRF model of Figure 7(a), the parameter vectors $\boldsymbol{\theta}_\phi$, $\boldsymbol{\theta}_\psi$ and $\boldsymbol{\theta}_\pi$ are learned by maximizing the conditional likelihood in each of the three models of Figure 7(b). In each case, only the factors in the model that contain the relevant parameter vector are retained.

As discussed in [44], this training method minimizes an upper bound on the log partition function, as follows: if we define the logarithm of the partition function $z(\boldsymbol{\theta}, \mathbf{x}) = \log Z(\boldsymbol{\theta}, \mathbf{x})$ and index the terms in the model by r, then

$$z(\boldsymbol{\theta}, \mathbf{x}) \leq \sum_r z_r(\boldsymbol{\theta}_r, \mathbf{x}) \qquad (11)$$

where $\boldsymbol{\theta}_r$ are the parameters of the rth term and $z_r(\boldsymbol{\theta}_r)$ is the partition function for a model containing only the rth term. Replacing $z(\boldsymbol{\theta}, \mathbf{x})$ with $\sum_r z_r(\boldsymbol{\theta}_r)$ in (1) then gives a lower bound on the conditional likelihood. It is this bound which is maximized during piecewise learning.

Unfortunately, this bound can be loose, especially if the terms in the model are correlated. In this case, performing piecewise parameter training leads to over-counting during inference in the combined model. To understand this, consider the case where we duplicate a term of the model $\psi(\boldsymbol{\theta}_\psi)$, so that there is an additional term $\psi(\boldsymbol{\theta}'_\psi)$ which has the same functional form but new parameters $\boldsymbol{\theta}'_\psi$. A model with duplicated terms is shown in Figure 7(c). As the duplicate terms have the same form and are based on the same features, these terms are perfectly correlated.

Piecewise training will learn that $\boldsymbol{\theta}_\psi$ and $\boldsymbol{\theta}'_\psi$ are the same and equal to the parameters $\boldsymbol{\theta}_\psi^{\text{old}}$ learned for this term in the original model. Since the log potential function $\log \psi$ is linear in the parameters,

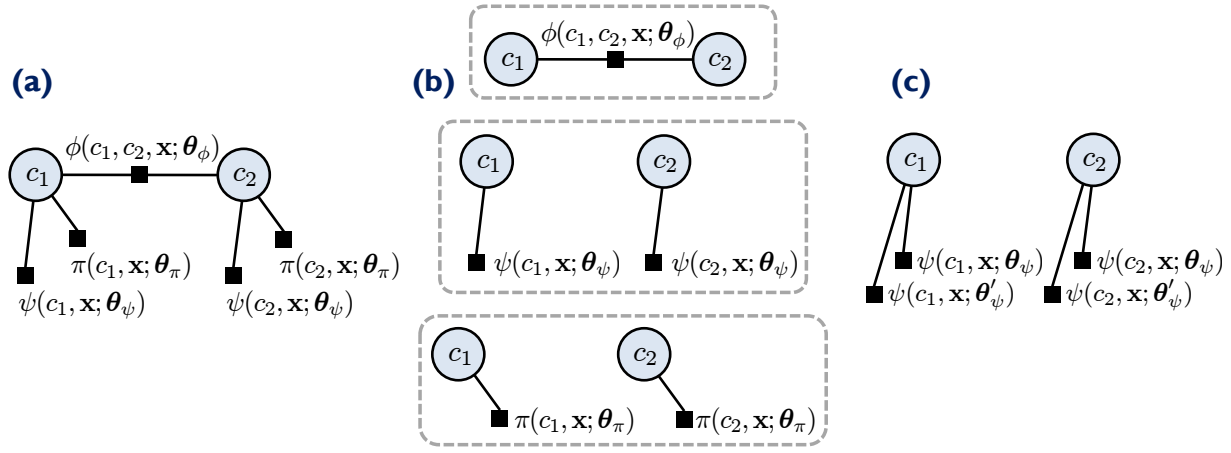

Figure 7: **Piecewise training of the CRF parameters. (a)** The factor graph (see e.g. [7]) for a simplified CRF model. Each black square represents a term in (1) and each circle represents a latent variable. Terms are connected to all variables that they depend on. **(b)** When piecewise training is applied to the CRF model of **(a)**, the parameters $\boldsymbol{\theta}_\phi$, $\boldsymbol{\theta}_\psi$ and $\boldsymbol{\theta}_\pi$ are learned by maximizing conditional likelihood in the top, middle and bottom models respectively. In each model, only the terms relating to the parameter being learned are retained. **(c)** A model in which the term ψ has been duplicated. In this case, piecewise training will learn model parameters which are twice those learned in the original non-duplicated model. Hence, piecewise training will lead to over-counting errors when terms in the model are correlated. See text for more details.

the duplicate model will be equivalent to the original model but with $\boldsymbol{\theta}_\psi^{\text{new}} = 2\boldsymbol{\theta}_\psi^{\text{old}}$, i.e. twice the correct value. To offset this over-counting effect and recover the original parameters, it would be necessary to weight the logarithm of each duplicate term by a factor of 0.5, or equivalently raise the term to the power of 0.5.

It is difficult to assess analytically the degree of over-counting introduced by dependencies between the different terms in our CRF model. Instead, we introduce scalar powers for each term and optimize these powers discriminatively using holdout validation. We found that it was only necessary to introduce powers for the location and color potentials. This use of weights means that unfortunately we are no longer able to claim the rigorous bound on the partition function given by piecewise training. Nonetheless, we introduce them for pragmatic reasons since they give significant improvement in performance and consist of only a few additional parameters which can tractably be chosen by optimizing on holdout data.

3.2.3 Learning the Potentials

Piecewise training with powers is therefore used to train the parameters of each of the CRF potentials separately as follows.

Texture-layout potential parameters: The texture-layout potential parameters are learned during boosting, described in Section 4.

Color potential parameters: The color potentials are learned at *test time* for each image independently, using the approach of [41]. First, the color clusters (4) are learned in an unsupervised manner using K-means. An iterative algorithm, reminiscent of EM [12], then alternates between inferring class labeling $\mathbf{c}^\star$ (Section 3.1), and updating the color potential parameters as

$$\boldsymbol{\theta}_\pi(c_i, k) = \left(\frac{\sum_i [c_i = c_i^\star] P(k|x_i) + \alpha_\pi}{\sum_i P(k|x_i) + \alpha_\pi} \right)^{w_\pi}, \tag{12}$$

which counts the class frequencies for each cluster. On the first iteration, inference is performed without the color potentials. In practice, the Dirichlet prior parameter α_π was set to 0.1, the power parameter was set as $w_\pi = 3$, and fifteen color components and two iterations gave good results. Because we are training in pieces, the color parameters do not need to be learned for the training set.

Location potential parameters: We train the location potential parameters by maximizing the likelihood of the normalized model containing just that potential and raising the result to a fixed power w_λ to compensate for over-counting. This corresponds to

$$\boldsymbol{\theta}_\lambda(c, \hat{i}) = \left(\frac{N_{c,\hat{i}} + \alpha_\lambda}{N_{\hat{i}} + \alpha_\lambda} \right)^{w_\lambda} \tag{13}$$

where $N_{c,\hat{i}}$ is the number of pixels of class c at normalized location $\hat{i}$ in the training set, $N_{\hat{i}}$ is the total number of pixels at location $\hat{i}$ and α_λ is a small integer (we use $\alpha_\lambda = 1$) corresponding to a weak Dirichlet prior on $\boldsymbol{\theta}_\lambda$. The parameter w_λ was changed between the different datasets; the relevant values are given in Section 5.

Edge potential parameters: The values of the two contrast-related parameters were manually selected to minimize the error on the validation set.

4 Boosted Learning of Texture, Layout, and Context

The most important term in the CRF energy is the unary texture-layout potential ψ (2), which is based on a novel set of features which we call *texture-layout filters*. These features are capable of jointly capturing texture, spatial layout, and textural context. In this section, we describe texture-layout filters and the boosting process used for automatic feature selection and learning the texture-layout potentials.

4.1 Textons

Efficiency demands a compact representation for the range of different appearances of an object. For this we use *textons* [34] which have been proven effective in categorizing materials [47] as well as generic object classes [49]. The term texton was coined by [23] for describing human textural perception, and is somewhat analogous to phonemes used in speech recognition.

The process of textonization is illustrated in Figure 8, and proceeds as follows. The training images are convolved with a 17-dimensional filter-bank at scale κ.[7] The 17D responses for all training pixels are then whitened (to give zero mean and unit covariance), and an unsupervised clustering is performed. We employ the Euclidean-distance K-means clustering algorithm, which can be made dramatically faster by using the techniques of [15]. Finally, each pixel in each image is assigned to the nearest cluster center, producing the *texton map*. We will denote the texton map as T where pixel i has value $T_i \in \{1, \ldots, K\}$. Note that for efficiency one can use the kd-tree algorithm [4] to perform the nearest neighbor search; without any approximation, textonization using kd-trees with leaf-node bins containing 30 cluster centers gave a speed up of about 5 times over simple linear search.

4.2 Texture-Layout Filters

Each texture-layout filter is a pair (r, t) of an image region, r, and a texton t. Region r is defined in coordinates relative to the pixel i being classified. For efficiency, we only investigate rectangular regions, though an arbitrary region could be used. For simplicity, a set $\mathcal{R}$ of candidate rectangles are chosen at random, such that their top-left and bottom-

[7] The choice of filter-bank is somewhat arbitrary, as long as it is sufficiently representative. We use the same filter-bank as [49], which consists of Gaussians at scales κ, 2κ and 4κ, x and y derivatives of Gaussians at scales 2κ and 4κ, and Laplacians of Gaussians at scales κ, 2κ, 4κ and 8κ. The Gaussians are applied to all three color channels, while the other filters are applied only to the luminance. The perceptually uniform CIELab color space is used. This filter-bank was determined to have full rank in a singular-value decomposition (see [22]), and therefore no redundant elements.

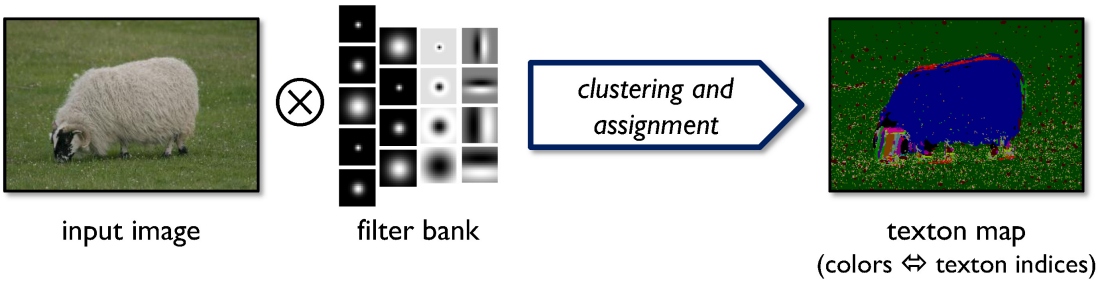

Figure 8: **The process of image textonization.** An input image is convolved with a filter-bank. The filter responses for all pixels in training images are clustered. Finally, each pixel is assigned a texton index corresponding to the nearest cluster center to its filter responses.

right corners lie within a fixed bounding box covering about half the image area.[8] As illustrated in Figure 9, the feature response at location i is the proportion of pixels under the offset region $r + i$ that have texton index t

$$v_{[r,t]}(i) = \frac{1}{\text{area}(r)} \sum_{j \in (r+i)} [T_j = t] \ . \qquad (14)$$

Outside the image boundary there is zero contribution to the feature response.

The filter responses can be efficiently computed over a whole image with integral images [48]. Figure 10 illustrates this process: the texton map is separated into K channels (one for each texton) and then, for each channel, a separate integral image is calculated. This can be thought of as an integral histogram [38] with one bin for each texton. The integral images can later be used to compute the texture-layout filter responses in constant time: if $\hat{T}^{(t)}$ is the integral image of T for texton channel t, then the feature response is computed as:

$$v_{[r,t]}(i) = \hat{T}^{(t)}_{r_{\text{br}}} - \hat{T}^{(t)}_{r_{\text{bl}}} - \hat{T}^{(t)}_{r_{\text{tr}}} + \hat{T}^{(t)}_{r_{\text{tl}}} \qquad (15)$$

where r_{br}, r_{bl}, r_{tr} and r_{tl} denote the bottom right, bottom left, top right and top left corners of rectangle r.

[8] For the evaluations in this paper, the bounding box was ± 100 pixels in x and y. This allows the model to exploit textural context over a long range, despite the CRF having connections only to pixel-wise neighbors. The CRF is still very important however: it allows us additionally to exploit the edge, color, and location potentials to achieve near pixel-perfect segmentation.

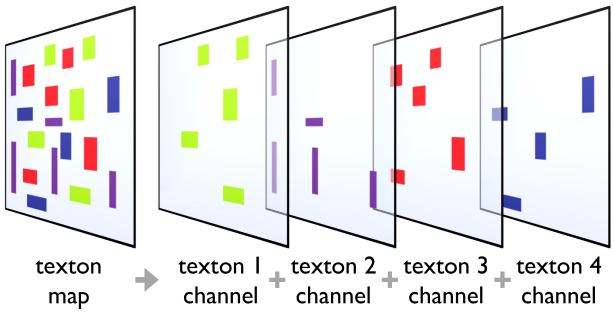

Figure 10: **Separating the texton map into multiple channels.** The texton map of an image, containing K textons, is split into K channels. An integral image [48] is built for each channel and used to compute texture-layout filter responses in constant time.

Texture-layout filters, as pairs of rectangular regions and textons, can be seen as an extension of the features used in [48]. Our features are sufficiently general to allow us to automatically *learn* layout and context information, in contrast to techniques such as shape context [5] which utilize a hand-picked descriptor. Similar features were proposed in [13], although textons were not used, and responses were not aggregated over a spatial region. Figure 9 illustrates how texture-layout filters are able to model textural context, and a toy example in Figure 11 demonstrates

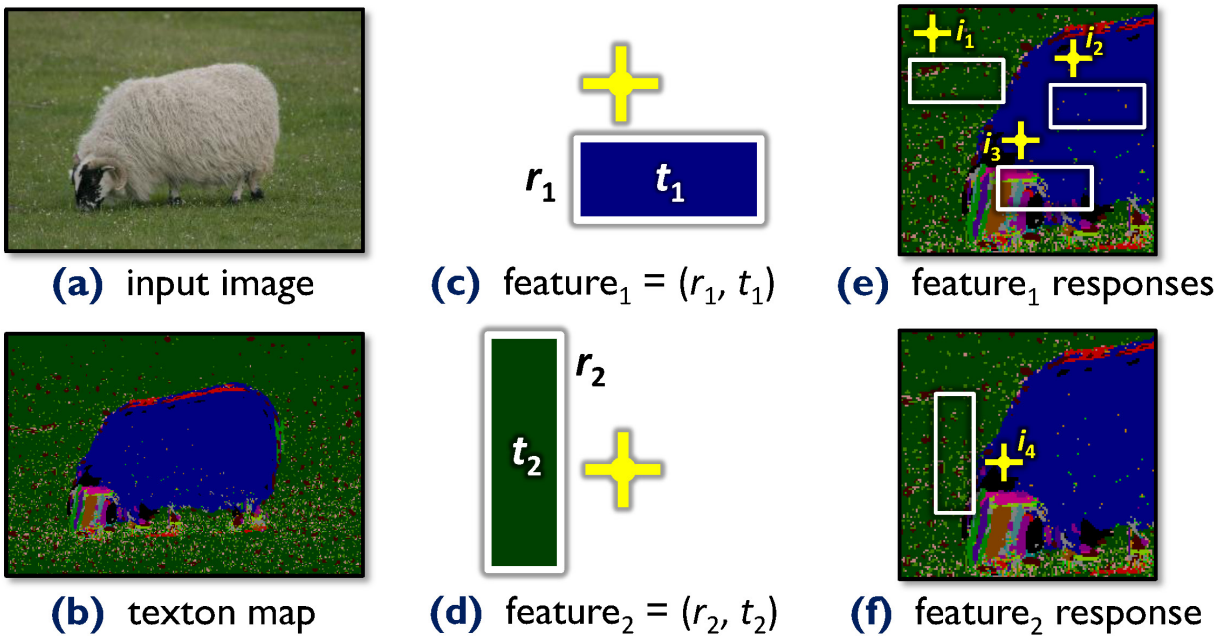

Figure 9: **Calculating feature responses and capturing textural context. (a, b)** An image and its corresponding texton map (colors map uniquely to texton indices). **(c)** Texture-layout filters are defined relative to the point i being classified (yellow cross). In this first example feature, rectangular region r_1 is paired with texton t_1 (mapping to the blue color). **(d)** A second feature where region r_2 is paired with texton t_2 (green). To improve readability, (c) and (d) are shown double size compared with (e) and (f). **(e)** The response $v_{[r_1,t_1]}(i)$ of the first feature is calculated at three positions in the texton map (magnified). In this example $v_{[r_1,t_1]}(i_1) \approx 0$, $v_{[r_1,t_1]}(i_2) \approx 1$, and $v_{[r_1,t_1]}(i_3) \approx \frac{1}{2}$. **(f)** For the second feature (r_2, t_2), where t_2 corresponds to 'grass', our algorithm can learn that points i (such as i_4) belonging to sheep regions tend to produce large values of $v_{[r_2,t_2]}(i)$, and hence can exploit the contextual information that sheep pixels tend to be surrounded by grass pixels.

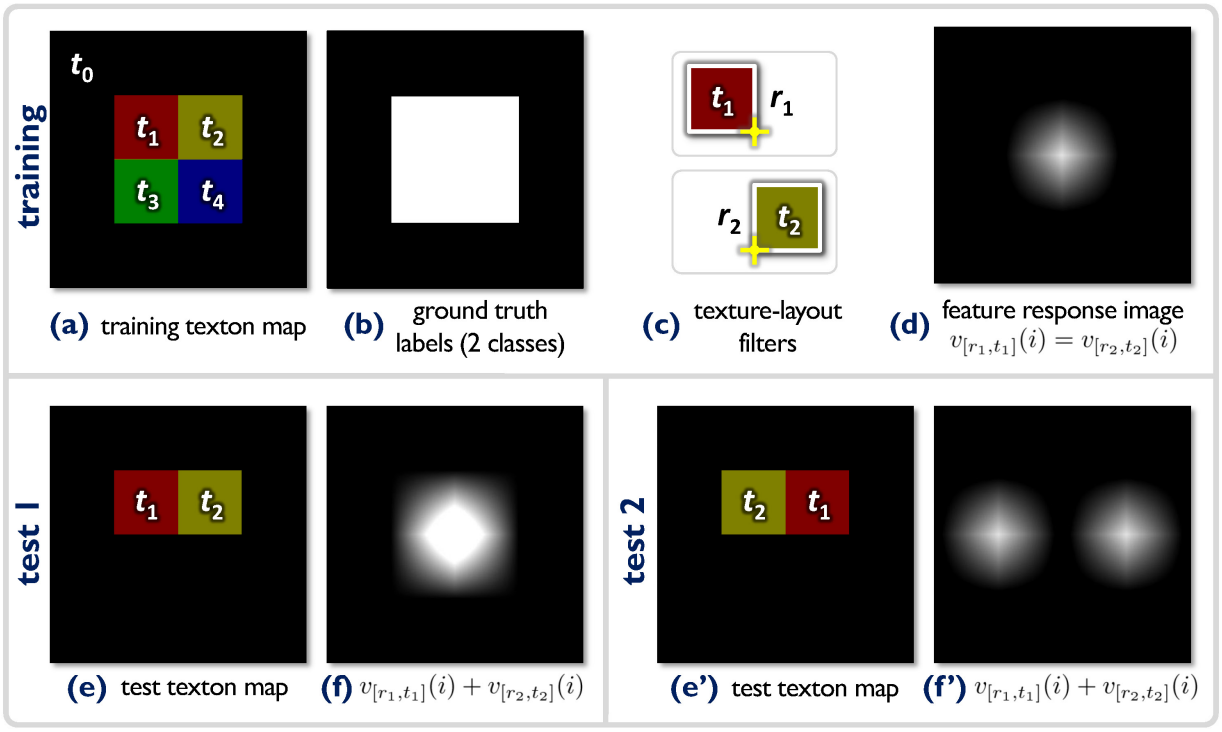

Figure 11: **Capturing local layout information.** This toy example illustrates how our texture-layout filters capture the layout of textons. **(a)** Input texton map. **(b)** Input binary ground truth label map (foreground=white, background=black). **(c)** Example texture-layout filters (r_1, t_1) and (r_2, t_2). **(d)** The feature response image $v_{[r_1,t_1]}(i)$ shows a positive response within the foreground region and zero in the background. An identical response image is computed for feature (r_2, t_2). Boosting would pick both these discriminative features. **(e)** A test input with textons t_1 and t_2 in the *same* relative position as that of training. **(f)** Illustration that the two feature responses *reinforce* each other. **(e')** A second test with t_1 and t_2 swapped. **(f')** The summed feature responses do not reinforce, giving a weaker signal for classification. Note that **(f)** and **(f')** are illustrative only, since boosting actually combines thresholded feature responses, though the principle still applies.

how texture-layout filters model layout.

4.3 Variations on Texture-Layout Filters

We investigated other region shapes r beyond simple rectangles. In particular we evaluated rectangles rotated by 45°, and pairs of rectangles with the texton responses either added ($v_{[r_1,t]}(i) + v_{[r_2,t]}(i)$) or subtracted ($v_{[r_1,t]}(i) - v_{[r_2,t]}(i)$). However, despite considerable extra computational expense (since these new combinations of features must be tested at each round of boosting; see below), the more complicated features did not produce noticeably better results. We believe this to be due to overfitting.

We also tried modeling appearance using the learned visual words of [49]. Unsurprisingly, the classification results were worse than using the raw K-means clusters, since the learning algorithm in [49] performs clustering so as to be invariant to the spatial layout of the textons – exactly the opposite of what is required here.

4.3.1 Texton-Dependent Layout Filters

We propose a further *texton-dependent* layout filter. Some classes may have large within-class textural differences, but a repeatable layout of texture within a particular object instance. As a concrete example, shirts have many different colors and textures, but all have roughly the same shape, and a particular shirt will often have a relatively uniform texture. The texton-dependent layout filters capture this intuition as follows. A texton-dependent layout filter acts identically to standard texture-layout filters, except that it uses the texton at pixel i being classified, T_i, rather than a particular learned texton. The feature response is therefore calculated as $v_{[r,T_i]}(i)$ (cf. (14)). The addition of this texton-dependent layout filter is evaluated in Section 5.3.

4.3.2 Separable Texture-Layout Filters

For speed critical applications, e.g. processing video sequences or large images, we suggest a final variation. As illustrated in Figure 12, separable texture-layout filters use spans instead of rectangles: horizontal spans count the proportion of textons agreeing in texton index that lie in a horizontal strip relative to the y coordinate being classified; vertical spans do similarly for a vertical strip.

Two boosted classifiers are then trained (see below) to act on the two separate Cartesian axes, using as target values the *set* of all classes present in column x or row y. A horizontal classifier $P(\mathbf{c}_\mathrm{x}|\mathbf{x}, \boldsymbol{\theta}_\mathrm{x})$, representing the class probabilities for each column, and a vertical classifier $P(\mathbf{c}_\mathrm{y}|\mathbf{x}, \boldsymbol{\theta}_\mathrm{y})$, representing the class probabilities for each row, are combined as the outer product

$$P(\mathbf{c}|\mathbf{x}, \boldsymbol{\theta}) \approx P(\mathbf{c}_\mathrm{x}|\mathbf{x}, \boldsymbol{\theta}_\mathrm{x}) \times P(\mathbf{c}_\mathrm{y}|\mathbf{x}, \boldsymbol{\theta}_\mathrm{y}) \ . \qquad (16)$$

This factorized approximation is clearly less powerful than learning the full joint classifier, but as shown in Section 5.4, gives acceptable quantitative performance and a considerable speed-up.

4.4 Learning Texture-Layout Filters using Joint Boost

We employ an adapted version of the Joint Boost algorithm [45], which iteratively selects discriminative texture-layout filters as 'weak learners', and combines them into a powerful classifier $P(c|\mathbf{x}, i)$, used by the texture-layout potentials ψ (2). Joint Boost *shares* each weak learner between a set of classes $\mathcal{C}$, so that a single weak learner classifies for several classes at once. This allows for classification with cost sublinear in the number of classes, and leads to improved generalization [45].

The learned 'strong' classifier is an additive model of the form $H(c, i) = \sum_{m=1}^{M} h_i^m(c)$, summing the classification confidence of M weak learners. The confidence value $H(c, i)$ can be reinterpreted as a probability distribution over c using the soft-max or multiclass logistic transformation [18] to give the texture-layout potentials:

$$P(c|\mathbf{x}, i) \propto \exp H(c, i) \ . \qquad (17)$$

Each weak learner is a decision stump based on

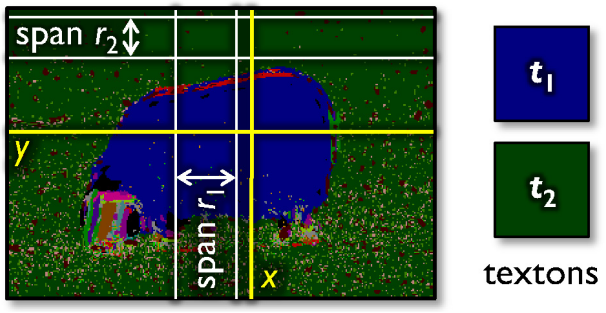

Figure 12: **Separable texture-layout filters.** For speed, separable texture-layout filters may be used. Horizontal spans are defined relative to the y coordinate being classified, and vertical spans relative to the x coordinate. The response of separable texture-layout filter (r, t) is computed as the proportion of pixels within the span r that have texton index t (cf. Figure 9). The classifiers for the two separate axes are combined as (16). In this example, separable texture-layout filter (r_1, t_1) uses the presence of texton t_1 in span r_1 as evidence that sheep is present at coordinate x. Feature (r_2, t_2) exploits textural context, looking for regions of grass texton t_2 in span r_2 above the sheep at coordinate y.

feature response $v_{[r,t]}(i)$ of the form

$$h_i(c) = \begin{cases} a[v_{[r,t]}(i) > \theta] + b & \text{if } c \in \mathcal{C} \\ k^c & \text{otherwise,} \end{cases} \tag{18}$$

with parameters $(a, b, \{k^c\}_{c \notin \mathcal{C}}, \theta, \mathcal{C}, r, t)$. The region r and texton index t together specify the texture-layout filter feature, and $v_{[r,t]}(i)$ denotes the corresponding feature response at position i. For those classes that share this feature ($c \in \mathcal{C}$), the weak learner gives $h_i(c) \in \{a + b, b\}$ depending on the comparison of $v_{[r,t]}(i)$ to a threshold θ. For classes not sharing the feature ($c \notin \mathcal{C}$), the constant k^c ensures that unequal numbers of training examples of each class do not adversely affect the learning procedure.

An excellent detailed treatment of the learning algorithm is given in [45], but we briefly describe it here for completeness. Each training example i (a pixel in a training image) is paired with a target value $z_i^c \in \{-1, +1\}$ ($+1$ if example i has ground truth class c, -1 otherwise) and assigned a weight w_i^c specifying its classification accuracy for class c after $m-1$ rounds of boosting. Round m chooses a new weak learner by minimizing an error function J_{wse} incorporating the weights:

$$J_{\text{wse}} = \sum_c \sum_i w_i^c (z_i^c - h_i^m(c))^2. \tag{19}$$

The training examples are then re-weighted

$$w_i^c := w_i^c e^{-z_i^c h_i^m(c)} \tag{20}$$

to reflect the new classification accuracy and maintain the invariant that $w_i^c = e^{-z_i^c H_i(c)}$. This procedure emphasizes poorly classified examples in subsequent rounds, and ensures that over many rounds, the classification for each training example approaches the target value.

Minimizing the error function J_{wse} unfortunately requires an expensive brute-force search over the possible weak learners h_i^m to find the optimal combination of the sharing set N, features (r, t), and thresholds θ. However, given these parameters, a closed form solution does exist for a, b and $\{k^c\}_{c \notin N}$:

$$b = \frac{\sum_{c \in N} \sum_i w_i^c z_i^c [v(i, r, t) \leq \theta]}{\sum_{c \in N} \sum_i w_i^c [v(i, r, t) \leq \theta]}, \tag{21}$$

$$a + b = \frac{\sum_{c \in N} \sum_i w_i^c z_i^c [v(i, r, t) > \theta]}{\sum_{c \in N} \sum_i w_i^c [v(i, r, t) > \theta]}, \tag{22}$$

$$k^c = \frac{\sum_i w_i^c z_i^c}{\sum_i w_i^c}. \tag{23}$$

We conclude our discussion of the learning algorithm with important and novel insights into efficient implementation of the Joint Boost algorithm.

Weak learner optimization: Several optimizations are possible to speed up the search for the optimal weak learner h_i^m. Since the set of all possible sharing sets is exponentially large, we employ the quadratic-cost greedy approximation of [45]. To speed up the minimization over features we employ a random feature selection procedure, described shortly. Optimization over $\theta \in \Theta$ for a discrete set Θ can

be made efficient by careful use of histograms of weighted feature responses: by treating Θ as an ordered set, histograms of values $v_{[r,t]}(i)$, weighted appropriately by $w_i^c z_i^c$ and w_i^c, are built over bins corresponding to the thresholds in Θ; these histogram are accumulated to give the thresholded sums necessary for the direct calculation of a and b in (21) and (22) for all values of θ at once.

Random feature selection: Exhaustive search over all features (r, t) at each round of boosting is prohibitive. For efficiency therefore, our algorithm examines only a randomly chosen fraction $\zeta \ll 1$ of the possible features [2]. All results in this paper use $\zeta = 0.003$ so that, over several thousand rounds, there is high probability of testing all features at least once, and hence good features should eventually get selected.

To analyze the effect of random feature selection, we compared the results of boosting on a toy data set of ten images with $|\mathcal{R}| = 10$ candidate regions, and $K = 400$ textons. The results in Figure 13 show that random feature selection improves the training time by two orders of magnitude whilst having only a small impact on the training error. Additionally, although we have no formal experiments to prove this, our experience with randomization has been that decreasing ζ occasionally gives an overall gain in test performance. This perhaps suggests that randomization is not only speeding up learning, but also improving generalization by preventing overfitting to the training data. This effect was also observed in [53].

Sub-sampling: The considerable memory and processing requirements of this procedure make training with an example at every pixel impractical. Hence we take training examples i only at pixels lying on a $\Delta_{\text{ss}} \times \Delta_{\text{ss}}$ grid ($\Delta_{\text{ss}} = 3$ for the Corel and Sowerby datasets, which contain smaller images, and $\Delta_{\text{ss}} = 5$ for the other datasets with larger images). The texture-layout filter responses are still calculated at full resolu-

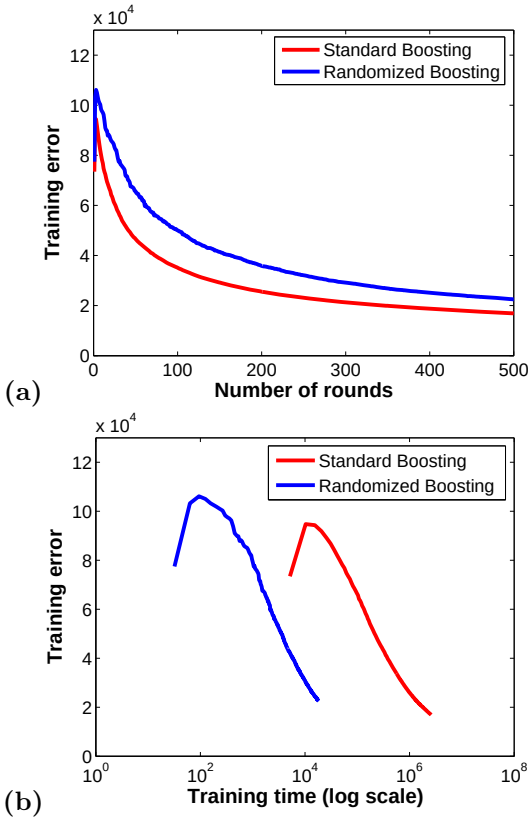

Figure 13: **Effect of random feature selection on a toy example. (a)** Training error as a function of the number of rounds (axis scales are unimportant). **(b)** Training error as a function of time. Randomization makes learning two orders of magnitude faster here, with very little increase in training error for the same number of rounds. The initial peak in error is due to the imbalance in the number of training examples for each class; on the log scale this appears quite significant, but in fact affects at most the first five rounds of boosting.

tion to allow for per-pixel accurate classification at test time; we simply train from fewer examples.

One consequence of this sub-sampling is that a small degree of shift-invariance is learned. On its own, this might lead to inaccurate segmentation at object boundaries. However, when applied in the context of the CRF, the edge and color potentials come into effect to locate the object boundary accurately.

Forests of boosted classifiers: A further possible efficiency, though not evaluated here, is the use of a forest of classifiers. In a similar manner to [1, 32], several texture-layout potential classifiers can be trained on random subsets of the image data and combined by averaging:

$$P(c|\mathbf{x}, i) = \frac{1}{W} \sum_{w=1}^{W} P^{[w]}(c|\mathbf{x}, i) \ . \qquad (24)$$

This allows infeasibly large datasets to be partitioned into smaller, manageable subsets, and has the potential to improve the generalization ability of the texture-layout potentials.

5 Results and Comparisons

In this section we investigate the performance of our system on several challenging datasets, and compare our results with existing work. We first investigate the effect of different aspects of the model, and then give the full quantitative and qualitative results.

5.1 Boosting Accuracy

In Figure 14 we illustrate how boosting gradually selects new texture-layout filters to improve classification accuracy. Initially, after 30 rounds of boosting (i.e. 30 texture-layout filters), a very poor classification is given, with low confidence. As more texture-layout filters are added, the classification accuracy improves greatly, and after 2000 rounds a very accurate classification is given. Note that this illustrates only the texture-layout potentials, and not the full CRF model.

Figure 15(a) illustrates the effect of training the texture-layout potentials using boosting on the MSRC dataset. As expected, the training error J_{wse} (19) decreases non-linearly as the number of weak learners increases. Furthermore, Figure 15(b) shows the accuracy of classification with respect to the validation set, which after about 5000 rounds flattens out to a value of approximately 73%. The accuracy against the validation set is measured as the pixel-wise segmentation accuracy, in other words the percentage of pixels that are assigned the correct class label.

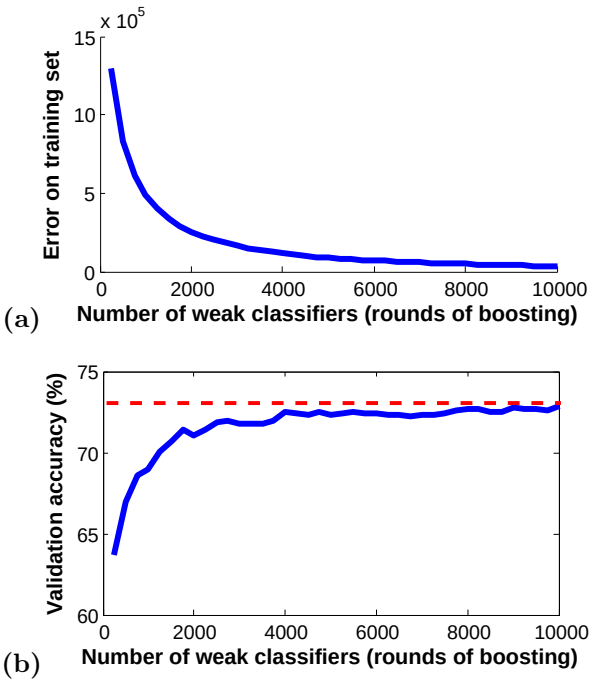

Figure 15: **Training and validation error. (a)** Training error, and **(b)** accuracy on the validation set, as functions of the number of weak learners. While the training error decreases almost to zero, the validation set accuracy rises to a maximum of about 73%.

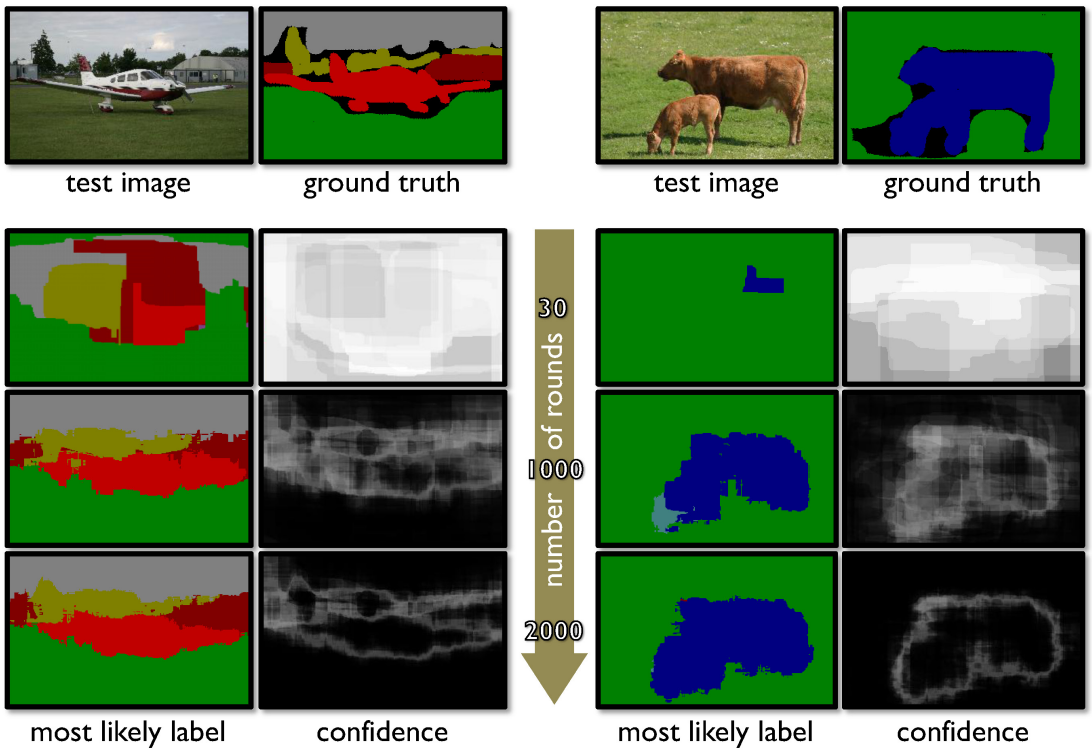

Figure 14: **Adding more texture-layout filters improves classification. Top row**: Unseen test images and the corresponding ground truth. **Lower three rows**: classification output of the texture-layout potentials trained by boosting, as more texture-layout filters are used. The three rows show the most likely label maps and the confidence maps with 30, 1000 and 2000 weak learners. Colors represent class labels (see Figure 18 for color key). White represents low confidence, black high confidence, computed as the entropy of the inferred class label distribution.

5.2 The Effect of Different Model Potentials

Figure 16 shows results for variations of the CRF model (1) with different potentials included. The parameter settings are given below in Section 5.4. It is evident that imposing spatial coherency (c) as well as an image dependent color model (d) improves the results considerably.

The percentage accuracies in Figure 16 (evaluated over the whole dataset) show that each term in our model captures essential information. Note that the improvement given by the full model over just the texture-layout potentials, while numerically small, corresponds to a significant increase in perceived accuracy (compare (b) with (d)), since the object contour is more accurately delineated. However, these results do suggest that, depending on the application, the texture-layout potentials may be sufficient without the full and relatively expensive CRF inference. One such application could be multi-class object tracking, and others are presented in Section 6.

5.3 Texton-Dependent Layout Filters

As described in Section 4.3.1, we proposed texton-dependent layout filters. An experiment was performed on the Corel dataset with 1000 rounds of

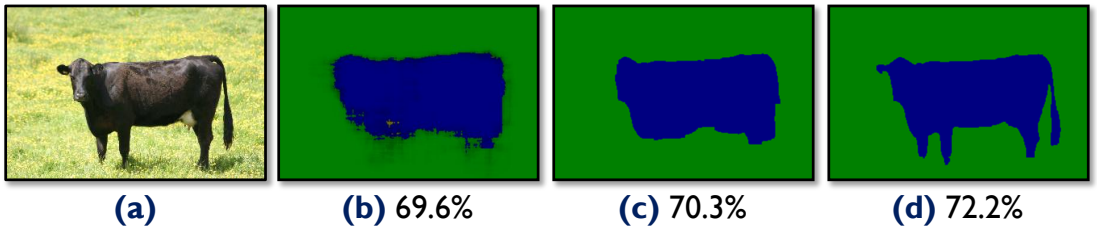

Figure 16: **Effect of different model potentials**. The original input image **(a)**, and the result using only the texture-layout potentials ψ **(b)**, with no explicit spatial coherency; at the boundary between blue and green, darker pixels correspond to higher entropy. **(c)** Results for the CRF model without color modeling, i.e. $\psi + \lambda + \phi$, and **(d)** for the full CRF model (1). Segmentation accuracy figures are given over the whole dataset. Observe the marked improvement in perceived segmentation accuracy of the full model over the texture-layout potentials alone, despite a seemingly small numerical improvement.

boosting, $\kappa = 0.45$, and using just the texture-layout potentials. We compared the performance on the test set of models trained (i) using just standard texture-layout filters, and (ii) using both standard and texton-dependent layout filters. The graph in Figure 17 plots the resulting pixel-wise segmentation accuracy of the boosted classifier as a function of K, the number of textons. As one would expect, the extra flexibility accorded by the additional texton-dependent layout filters gave a small but significant improvement. Also of note is the definite peak in performance as a function of K: with too many textons the boosting algorithm starts to overfit.

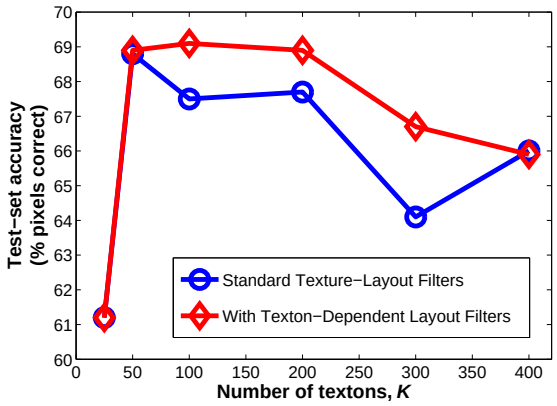

Figure 17: **Performance of texton-dependent layout filters.** Performance against a test set in terms of pixel-wise segmentation accuracy is plotted against the number of textons, K. See text for explanation.

5.4 MSRC 21-Class Database Results

This section presents results of object class recognition and image segmentation for the full CRF model on the MSRC 21-class database. Our unoptimized implementation takes approximately three minutes to segment each test image. The majority of this time is spent evaluating the texture-layout potentials $P(c|\mathbf{x}, i)$, involving a few thousand weak-classifier evaluations. Sub-sampling at test time by evaluating the texture-layout potentials on a $\Delta_{\text{ss}} \times \Delta_{\text{ss}}$ grid (with $\Delta_{\text{ss}} = 5$) produces results almost as good in about twenty-five seconds per test image.

Training the model took about 42 hours for 5000 rounds of boosting on the 21-class training set of 276

images on a 2.1 GHz machine with 2GB memory.[9] Without random feature selection, the training time would have been around 14000 hours.

Example results of simultaneous recognition and segmentation are shown in Figures 1 and 18. These show both the original photographs and the color-coded output labeling. Note the quality of both recognition and segmentation. For instance, despite large occlusions, bicycles are recognized and segmented correctly, and large variations in the appearance of grass and road are correctly modeled.

We present in Figure 19 some results where performance is poor, and highlight a few of examples in order to better understand the behavior of our algorithm. In (a) and (c), high segmentation accuracy is achieved due to the strong color model π in the CRF, but the semantic class labels are incorrect. In (b) a large wooden boat was incorrectly classified as tree, although the segmentation is not bad. In (d) the dog's shadow has been classified as building. Perhaps an explicit treatment of shadow would improve problems such as this. Finally, in (g) the model infers a bird on water. Examining the training data we observe several examples of white birds on water, but no white boats, suggesting that textural context has been overfit.

5.4.1 Quantitative Evaluation

Figure 20 shows the confusion matrix obtained by applying our algorithm to the test set. Accuracy values in the table are computed as the percentage of image pixels assigned to the correct class label, ignoring pixels labeled as void in the ground truth. The overall classification accuracy is 72.2%; random chance would give $1/21 = 4.76\%$, and thus our results are about 15 times better than chance. For comparison, the boosted classifier alone gives an overall accuracy of 69.6%, thus the color, edge and location potentials increase the accuracy by 2.6%. This seemingly small numerical improvement corresponds to a large perceptual improvement (cf. Figure 16). The parameter settings, learned against the validation set, were $M = 5000$ rounds, $K = 400$ textons, $|\mathcal{R}| = 100$ candidate regions, edge potential parameters $\boldsymbol{\theta}_\phi = [45, 10]^T$, filter-bank scale $\kappa = 1.0$, and location potential power $w_\lambda = 0.1$.

The greatest accuracies are for classes which have low visual variability and many training examples (such as grass, book, tree, road, sky and bicycle) whilst the lowest accuracies are for classes with high visual variability and fewer training examples (for example boat, chair, bird, dog). We expect more training data and the use of features with better lighting invariance properties to improve the recognition accuracy for these difficult classes considerably.

Let us now focus on some of the largest mistakes in the confusion matrix, to gather some intuition on how the algorithm may be improved. Structured objects such as airplanes, chairs, signs, and boats are sometimes incorrectly classified as buildings. This kind of problem may be ameliorated using a parts-based modeling approach, such as [51]. For example, detecting windows and roofs should resolve many such ambiguities. Furthermore, objects such as cows, sheep, and chairs, which in training are always seen standing on grass, can be confused with grass. This latter effect is partially attributable to inaccuracies in the manual ground truth labeling, where pixels are often mislabeled near object boundaries.

5.4.2 Separable Texture-Layout Filters

We suggested the use of separable texture-layout potentials in Section 4.3.2. We compare against the joint model, where we recall that 69.6% pixel-wise segmentation accuracy was achieved. Using the separable model, we obtain the remarkably accurate 64.9%. Part of the success is due to the separable potentials performing well in larger regions of classes such as sky and grass. Using the separable potentials, there is a very noticeable speed-up of over 5 times for both training and test time. With optimizations, this speed improvement could be increased dramatically since the critical path of the algorithm has been reduced from $O(NM)$ to $O(N + M)$ for an $N \times M$ image. Separable texture-layout potentials also re-

[9] Simple optimizations subsequent to these experiments have reduced test time to under 1 second per image for the texture-layout potentials, and 5 seconds per image for the CRF inference. Training time was also reduced drastically to about 1 hour.

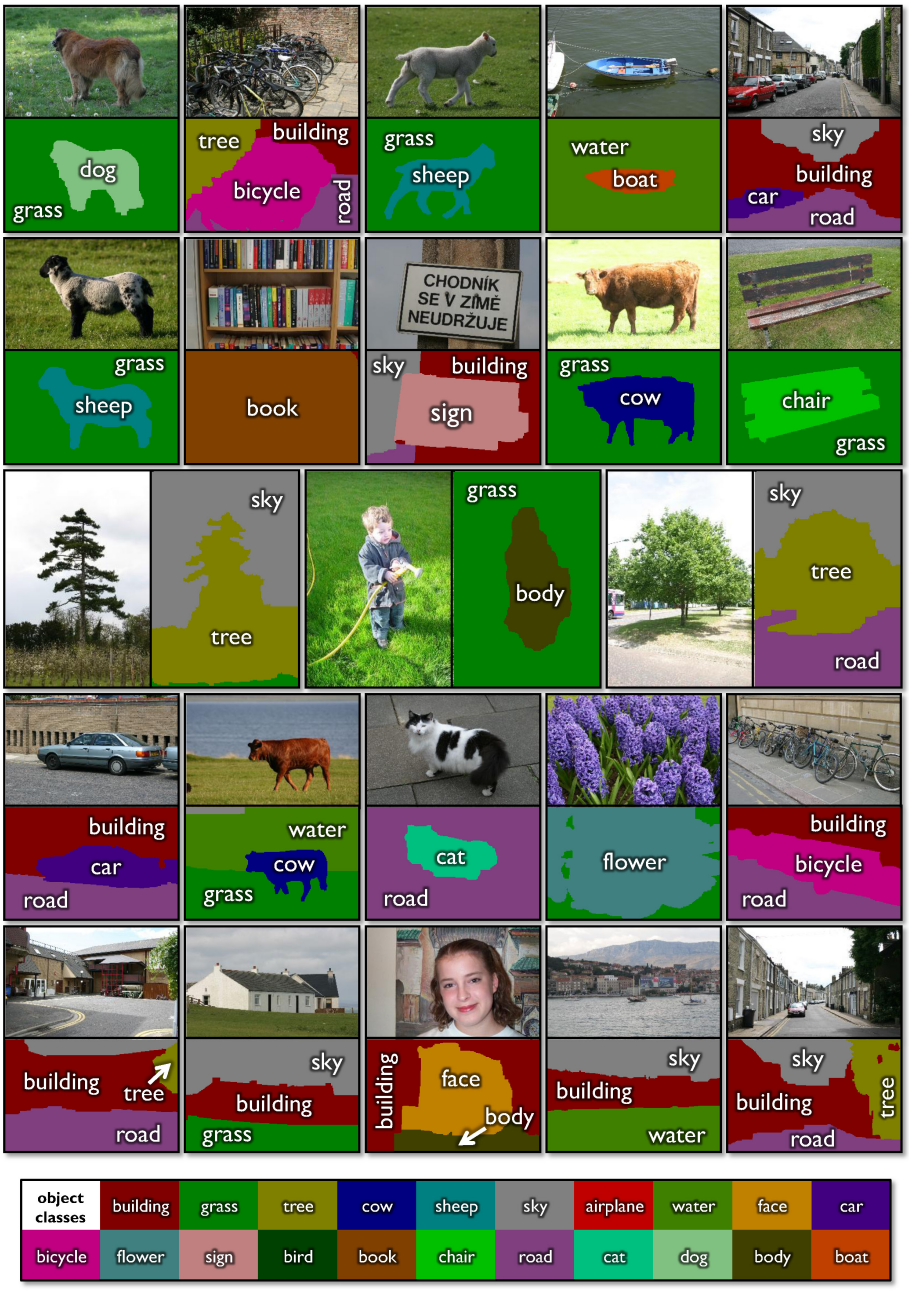

Figure 18: **Example results on the MSRC 21-class database.** Above, test images with inferred color-coded output object-class maps. Below, color-coding legend for the 21 object classes. For clarity, textual labels have also been superimposed on the resulting segmentations. Note that all images were processed using the *same* learned model. Further results from this system are given in Figure 1.

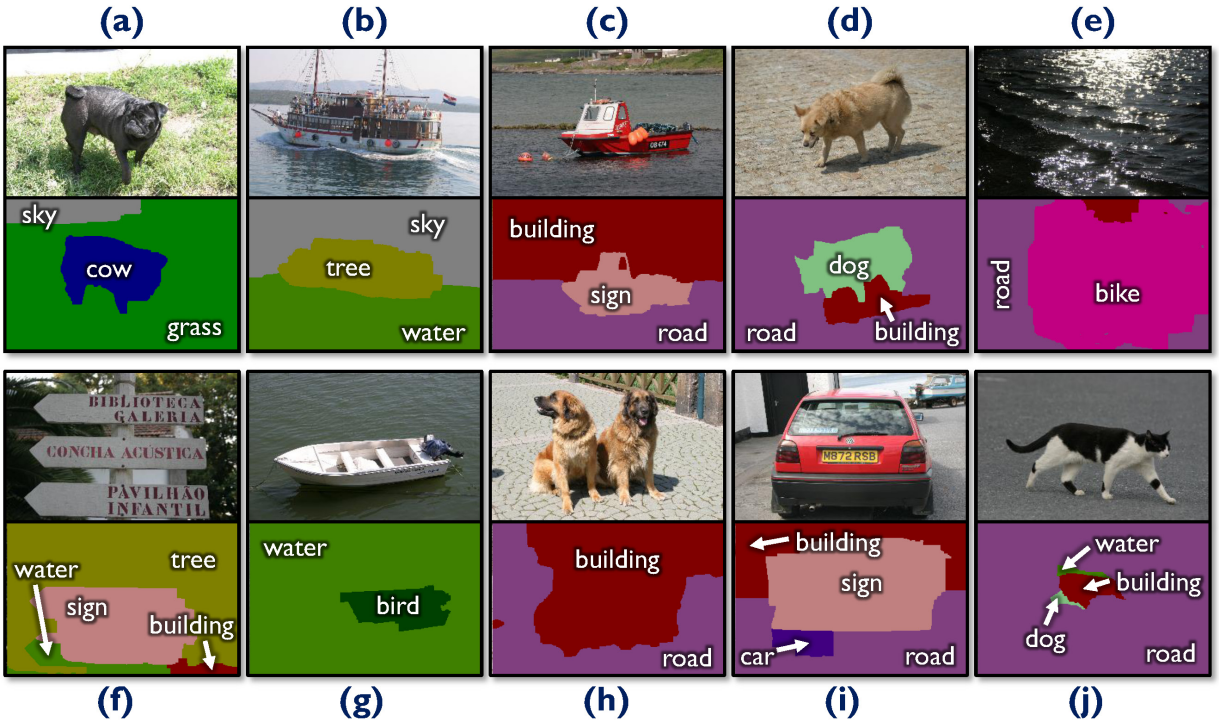

Figure 19: **Examples where recognition works less well.** Input test images with corresponding color-coded output object-class maps. Even when recognition fails, segmentation may still be quite accurate.

quires considerably less memory during training, and so more training data could be employed if available.

5.4.3 The Influence of Color and Luminance

We performed an experiment on a subset of the MSRC 21-class test set to investigate the importance of color and luminance. Three boosted classifiers with $M = 1000$ rounds were learned using images textonized using different filter-banks: (i) the original filter-bank that combines color and luminance information, (ii) the subset of the filter-bank that is applied only to the luminance channel, and (iii) the subset of the filter-bank that is applied only to the a and b color channels. All other parameters were kept the same. The pixel-wise segmentation accuracies obtained are as follows: 70.1% using the full filter-bank (both color and luminance), 61.4% using only color, and 50.9% using only luminance. These results suggest that, in this dataset, textural information from the color channels plays a more important role than from the luminance channel in accurately segmenting the images. However, information from the luminance channel gives additional strong clues to further improve results.

5.5 Comparison with Winn *et al.*

To assess how much the layout and context modeling help with recognition we compared the accuracy of our system against the framework of [49], i.e. given a manually selected region, assign a single class label to it and then measure classification accuracy. On the 21-class database, our algorithm achieves 70.5% region-based recognition accuracy, beating [49] which achieves 67.6% using 5000 textons and their Gaussian class models. Moreover, the significant advantages of our proposed algorithm are that: (i) no regions

True class \ Inferred class	building	grass	tree	cow	sheep	sky	aeroplane	water	face	car	bike	flower	sign	bird	book	chair	road	cat	dog	body	boat
building	**61.6**	4.7	9.7	0.3		2.5	0.6	1.3	2.0	2.6	2.1		0.6	0.2	4.8		6.3	0.4		0.5	
grass	0.3	**97.6**	0.5								0.1									1.3	
tree	1.2	4.4	**86.3**	0.5		2.9	1.4	1.9	0.8	0.1							0.1		0.2	0.1	
cow		30.9	0.7	**58.3**				0.9	0.4			0.4			4.2					4.1	
sheep	16.5	25.5	4.8	1.9	**50.4**									0.6			0.2				
sky	3.4	0.2	1.1			**82.6**		7.5									5.2				
aeroplane	21.5	7.2				3.0	**59.6**	8.5													
water	8.7	7.5	1.5	0.2		4.5		**52.9**		0.7	4.9			0.2	4.2		14.1	0.4			
face	4.1		1.1						**73.5**	7.1					8.4			0.4	0.2	5.2	
car	10.1		1.7							**62.5**	3.8		5.9	0.2			15.7				
bike	9.3		1.3							1.0	**74.5**		2.5			3.9	5.9		1.6		
flower		6.6	19.3	3.0								**62.8**			7.3		1.0				
sign	31.5	0.2	11.5	2.1		0.5		6.0		1.5		2.5	**35.1**		3.6	2.7	0.8	0.3		1.8	
bird	16.9	18.4	9.8	6.3	8.9	1.8		9.4						**19.4**			4.6	4.5			
book	2.6		0.6						0.4			2.0			**91.9**					2.4	
chair	20.6	24.8	9.6	18.2		0.2					3.7				1.9	**15.4**	4.5		1.1		
road	5.0	1.1	0.7					3.4	0.3	0.7	0.6		0.1	0.1	1.1		**86.0**			0.7	
cat	5.0		1.1	8.9				0.2		2.0					0.6		28.4	**53.6**	0.2		
dog	29.0	2.2	12.9	7.1				9.7							8.1		11.7		**19.2**		
body	4.6	2.8	2.0	2.1	1.3	0.2			6.0	1.1					9.9		1.7	4.0	2.1	**62.1**	
boat	25.1		11.5			3.8		30.6		2.0	8.6		6.4	5.1			0.3				**6.6**

Figure 20: **Accuracy of segmentation for the 21-class database.** Confusion matrix with percentages row-normalized. The overall pixel-wise accuracy is 72.2%.

need to be specified manually, and (ii) a pixel-wise semantic segmentation of the image is obtained.

5.6 Comparison with He *et al.*

We also compared against [19] on their Corel and Sowerby databases, and give results in Table 1 and Figure 21. For both models we show the results of the unary classifier alone as well as results for the full model. For the Sowerby database the parameters were set as $M = 6500$, $K = 250$, $\kappa = 0.7$, $\boldsymbol{\theta}_\phi = [10, 2]^T$, and $w_\lambda = 2$. For the Corel database, all images were first automatically color and intensity normalized, and the training set was augmented by applying random affine intensity changes to give the classifier improved invariance to illumination. The parameters were set as $M = 5000$, $K = 400$, $\kappa = 0.7$, $\boldsymbol{\theta}_\phi = [20, 2]^T$, and $w_\lambda = 4$.

Our method gives comparable or better (with only unary classification) results than [19]. However, the careful choice of efficient features and learning techniques, and the avoidance of inefficient Gibbs sampling, enables our algorithm to scale much better with the number of training images and object classes. Incorporating *semantic* context information as in [19] is likely to further improve our performance.

In the Corel database, the ground truth labeling between the ground and vegetation classes was often quite ambiguous to human observers. The confusion matrix of our results also bore this out, and merging these two classes results in significantly improved performance: 75.9% with just the texture-layout potentials, and 82.5% with the full CRF model.

5.7 Television Sequences

A separate model was trained for each of the nine television video sequences. In each sequence, 120 frames spaced equally in time were selected, and di-

	Accuracy		Speed (Train/Test)	
	Sowerby	Corel	Sowerby	Corel
TextonBoost – full CRF model	88.6%	74.6%	20 m / 1.1 s	30 m / 2.5 s
TextonBoost – texture-layout potentials only	85.6%	68.4%		
He *et al.* – mCRF model	89.5%	80.0%	24 h / 30 s	24 h / 30 s
He *et al.* – unary classifier only	82.4%	66.9%		

Table 1: **Comparison of segmentation/recognition accuracy and efficiency.** Timings for [19] are from correspondence with authors. Training times are for the whole training set, test times are per image.

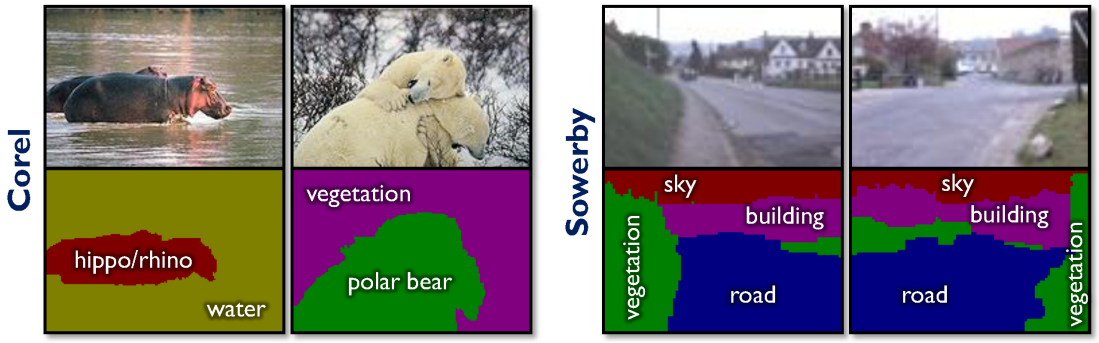

Figure 21: **Example results on the Corel and Sowerby databases.** A different set of object class labels and thus different color-coding is used here. Textual labels are superimposed for clarity.

vided randomly in half for training and testing.[10] The training data was combined with the MSRC training data, and the texture-layout potentials were learned by boosting. Only the texture-layout potentials, as the most significant part of the model, were used for evaluation. For more polished and visually pleasing results, the full CRF inference could be run, although as illustrated in Figure 16 only a small quantitative improvement would be seen. The parameters were set as $M = 700$, $K = 400$, and $\kappa = 0.7$.

Example results together with the overall segmentation accuracies are given in Figure 22. The numbers show considerable accuracy across very varied sets of images, with on average two-thirds of all pixels being correctly classified into one of 21 classes, indicating significant potential for the application of TextonBoost to automatic analysis of video sequences. As can be seen from the images, the technique works well across very varied sequences. One slight limitation is that the system tends to get the larger objects in the scene classified correctly, but smaller objects such as hands can get missed off. This is at least partially due to the filter-bank used during the textonization: the width of the Gaussian blur tends to oversmooth small objects. The particularly strong result on the soccer sequence is perhaps slightly skewed by the large amount of grass present in the images.

[10] Given the large time spacing (approximately 10 seconds between frames), we believe training and testing frames to be reasonably independent. With more hand-labeled data, using perhaps different episodes of the same television shows, an improved experiment on the generalization performance could be attempted.

6 Applications

The TextonBoost system has been applied in several exciting new areas.

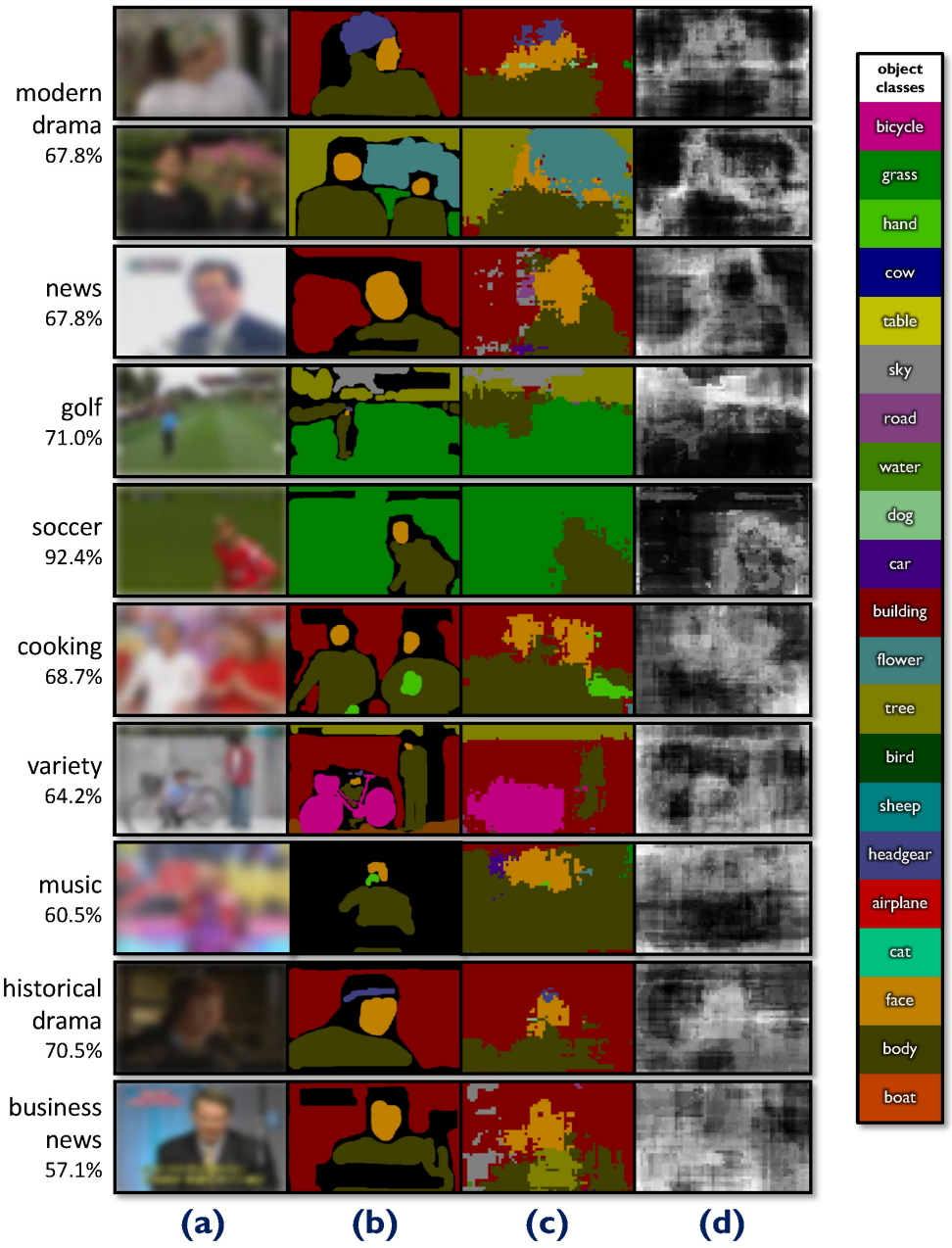

Figure 22: **Example results on the television sequences.** **(a)** Test images (blurred out for copyright reasons). **(b)** The hand-labeled ground truth. **(c)** The most likely labels inferred by the texture-layout potentials. **(d)** The entropy of the inferred class label distributions: white is high entropy, black low entropy. Class color key is given right, and pixel-wise segmentation accuracies for each dataset are shown.

AutoCollage: The work of [40] takes a collection of images and automatically blends them together to create a visually pleasing collage; by choosing image regions of particular interest (such as faces), detected through semantic segmentation, a more interesting collage could be generated. Additionally, images could be placed in suitable regions of the collage, so that for example, images with sky might be placed towards the top of the image.

Semantic Photo Synthesis: In [21], the user draws onto a canvas both particular objects (the Taj Mahal, for example) and regions assigned a particular semantic label (sky, water, car, etc.). The system then automatically queries a database containing images labeled by TextonBoost, to find relevant images that match the user-drawn query. Finally, it creates novel photo-realistic images by stitching together the image fragments that matched the individual parts of the query.

Interactive Semantic Segmentation: An optimized implementation of our system could be used as a complete interactive semantic segmentation tool, as demonstrated in Figure 23. With only one user click on the incorrectly labeled part of the building, a correct and accurate segmentation was achieved. Internally, the unary potential of pixels within a small radius of the clicked pixel is set to infinity for its initial label, tree. The result of the graph cut optimization for this new CRF energy is the correct labeling. A further speed-up can potentially be achieved by re-using the flow of the previous solution [10, 24].

Interactive Image Editing: We suggest one final exciting application: interactive image editing. Imagine that TextonBoost produces a perfect semantic segmentation of an image. It is then possible to tailor image editing tools presented to the user according to the semantic type: for example, tools for editing the sky could allow the user to tint it more blue or increase the contrast; for foreground objects, such as the person in Figure 24, options could be given to automatically erase the object from the image (using image in-painting, e.g. [11]), change the focus of background, fix red eye, or adjust the color balance just for that object.

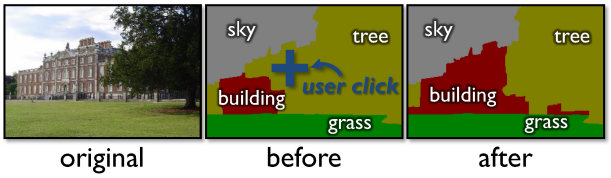

Figure 23: **Interactive object labeling**. **Left:** input test image. **Middle:** for this example, the automatic recognition failed to classify the building correctly. The user then clicks the mouse at the blue cross, indicating that this part of the image is currently misclassified. **Right:** inference is performed again with the additional user constraints. The building is now correctly classified and the segmentation is improved.

7 Conclusions

This paper has presented a novel discriminative model for efficient and effective recognition and semantic segmentation of objects in images. We have: (i) introduced new features, texture-layout filters, which simultaneously capture texture, layout, and context information, and shown that they outperform other existing techniques for this problem; (ii) trained our model efficiently by exploiting both randomized boosting and piecewise training techniques; and (iii) achieved efficient labeling by a combination of integral image processing and feature sharing. The result is an algorithm which accurately recognizes and segments a large number of object classes in photographs much faster than existing systems. We have performed a thorough evaluation of the system on several varied image databases and have achieved accurate and competitive results.

Our comparison of the different terms in the CRF model suggests that the texture-layout potentials, based on the boosted combination of texture-layout

Figure 24: **Semantic-aware interactive image editing. Left:** the semantic segmentation given by TextonBoost is used to drive a semantic-aware user interface. Here the user has selected the person, and a context sensitive menu presents editing options specific for the 'person' class. **Right:** after the user clicks 'Erase...', an automatic system [11] removes the person from the image by filling in over the top of her.

filters, are by far the most significant contribution to accurate semantic segmentation of images. The CRF model does, however, significantly improve the perceived accuracy of results.

The evaluation was limited to 21 classes by the prohibitive cost of hand-labeling images, and the additional time and memory requirements for training. However, in principle, we believe TextonBoost could be applied to many more classes. The primary limitation is the performance of the texture-layout potentials learned by boosting. The use of Joint Boosting means that classification cost grows sub-linearly with the number of classes [45], although training time increases quadratically. The algorithm is highly parallelizable, which is becoming an important consideration as processors move towards multi-core architectures. One limitation when moving to more classes is the simple ontological model used, where each pixel is assigned only one class label. This can lead to semantic confusions, such as Figure 19(b). For datasets with more classes, improvements to the ontological model will start to have profound effects on accuracy. Already, new work [35] is showing promise in this direction.

In the future we hope to integrate explicit semantic context information such as in [19] to improve further the classification accuracy. We are interested in improving the detection of objects at smaller scales, which are sometimes poorly detected with the current system. We would like to investigate other forms of textonization: clustered SIFT [33] descriptors might provide more invariance to, for example, changes in lighting conditions than the filter-bank used in this work; in a similar vein, a soft assignment of pixels to textons might produce better results. Furthermore the features could be extended to incorporate motion-based cues to improve the technique when applied to video sequences.

Acknowledgements. In addition to the anonymous reviewers who provided excellent suggestions, the authors would like to thank Tom Minka, Martin Szummer, Björn Stenger, Giovanni Maria Farinella, Florian Schroff, Jitendra Malik, Andrew Zisserman, Roberto Cipolla, and Andrew Blake for their invaluable help.

References

[1] Y. Amit, D. Geman, and K. Wilder. Joint induction of shape features and tree classifiers. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 19:1300–1305, 1997.

[2] S. Baluja and H. A. Rowley. Boosting sex identification performance. In *AAAI*, pages 1508–1513, 2005.

[3] M.J. Beal. *Variational Algorithms for Approximate Bayesian Inference*. PhD thesis, Gatsby Computational Neuroscience Unit, University College London, 2003.

[4] J.S. Beis and D.G. Lowe. Shape indexing using approximate nearest-neighbour search in high-dimensional spaces. In *Proc. IEEE Conf. Computer Vision and Pattern Recognition*, pages 1000–1006, June 1997.

[5] S. Belongie, J. Malik, and J. Puzicha. Shape matching and object recognition using shape contexts. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 24(24):509–522, April 2002.

[6] A.C. Berg, T.L. Berg, and J. Malik. Shape matching and object recognition using low distortion correspondences. In *Proc. IEEE Conf. Computer Vision and Pattern Recognition*, volume 1, pages 26–33, June 2005.

[7] C.M. Bishop. *Pattern Recognition and Machine Learning*. Springer-Verlag New York, Inc., 2006.

[8] A. Blake, C. Rother, M. Brown, P. Perez, and P.H.S. Torr. Interactive image segmentation using an adaptive GMMRF model. In T. Pajdla and J. Matas, editors, *Proc. European Conf. on Computer Vision*, volume LNCS 3021, pages 428–441, Prague, Czech Republic, May 2004. Springer.

[9] E. Borenstein, E. Sharon, and S. Ullman. Combining top-down and bottom-up segmentations. In *IEEE Workshop on Perceptual Organization in Computer Vision*, volume 4, page 46, 2004.

[10] Y. Boykov and M.-P. Jolly. Interactive Graph Cuts for optimal boundary and region segmentation of objects in N-D images. In *Proc. Int. Conf. on Computer Vision*, volume 1, pages 105–112, Vancouver, Canada, July 2001.

[11] A. Criminisi, P. Perez, and K. Toyama. Region filling and object removal by exemplar-based inpainting. *IEEE Trans. on Image Processing*, 13(9):1200–1212, September 2004.

[12] A. Dempster, N. Laird, and D. Rubin. Maximum likelihood from incomplete data via the em algorithm. In *JRSS B*, pages 39:1–38, 1976.

[13] P. Dollár, Z. Tu, and S. Belongie. Supervised learning of edges and object boundaries. In *Proc. IEEE Conf. Computer Vision and Pattern Recognition*, volume 2, pages 1964–1971, 2006.

[14] P. Duygulu, K. Barnard, N. de Freitas, and D. Forsyth. Object recognition as machine translation: Learning a lexicon for a fixed image vocabulary. In A. Heyden, G. Sparr, and P. Johansen, editors, *Proc. European Conf. on Computer Vision*, volume LNCS 2353, pages 97–112. Springer, May 2002.

[15] C. Elkan. Using the triangle inequality to accelerate k-means. In *Proc. Int. Conf. on Machine Learning*, pages 147–153, 2003.

[16] L. Fei-Fei, R. Fergus, and P. Perona. Learning generative visual models from few training examples: an incremental bayesian approach tested on 101 object categories. *Proc. of CVPR 2004. Workshop on Generative-Model Based Vision.*, 2004.

[17] R. Fergus, P. Perona, and A. Zisserman. Object class recognition by unsupervised scale-invariant learning. In *Proc. IEEE Conf. Computer Vision and Pattern Recognition*, volume 2, pages 264–271, June 2003.

[18] J. Friedman, T. Hastie, and R. Tibshirani. Additive logistic regression: a statistical view of boosting. *Annals of Statistics*, 28(2):337–407, 2000.

[19] X. He, R.S. Zemel, and M.Á. Carreira-Perpiñán. Multiscale conditional random fields for image labeling. In *Proc. IEEE Conf. Computer Vision and Pattern Recognition*, volume 2, pages 695–702, June 2004.

[20] X. He, R.S. Zemel, and D. Ray. Learning and incorporating top-down cues in image segmentation. In A. Leonardis, H. Bischof, and A. Pinz, editors, *Proc. European Conf. on Computer Vision*, volume LNCS 3951, pages 338–351. Springer, May 2006.

[21] M. Johnson, G. Brostow, J. Shotton, O. Arandjelovic, V. Kwatra, and R. Cipolla. Semantic photo synthesis. *Computer Graphics Forum*, 25(3):407–413, September 2006.

[22] D.G. Jones and J. Malik. A computational framework for determining stereo correspondence from a set of linear spatial filters. In *Proc. European Conf. on Computer Vision*, pages 395–410, 1992.

[23] B. Julesz. Textons, the elements of texture perception, and their interactions. *Nature*, 290(5802):91–97, March 1981.

[24] P. Kohli and P.H.S. Torr. Efficiently solving dynamic Markov Random Fields using Graph Cuts. In *Proc. Int. Conf. on Computer Vision*, volume 2, pages 922–929, Beijing, China, October 2005.

[25] V. Kolmogorov and R. Zabih. What energy functions can be minimized via graph cuts? *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 26(2):147–159, February 2004.

[26] S. Konishi and A. L. Yuille. Statistical cues for domain specific image segmentation with performance analysis. In *Proc. IEEE Conf. Computer Vision and Pattern Recognition*, volume 1, pages 125–132, June 2000.

[27] M.P. Kumar, P.H.S. Torr, and A. Zisserman. OBJ CUT. In *Proc. IEEE Conf. Computer Vision and Pattern Recognition*, volume 1, pages 18–25, June 2005.

[28] S. Kumar and M. Hebert. Discriminative random fields: A discriminative framework for contextual interaction in classification. In *Proc. Int. Conf. on Computer Vision*, volume 2, pages 1150–1157, October 2003.

[29] J. Lafferty, A. McCallum, and F. Pereira. Conditional Random Fields: Probabilistic models for segmenting and labeling sequence data. In *Proc. Int. Conf. on Machine Learning*, pages 282–289, 2001.

[30] S. Lazebnik, C. Schmid, and J. Ponce. Beyond bags of features: spatial pyramid matching for recognizing natural scene categories. In *Proc. IEEE Conf. Computer Vision and Pattern Recognition*, 2006.

[31] B. Leibe and B. Schiele. Interleaved object categorization and segmentation. In *Proc. British Machine Vision Conference*, volume II, pages 264–271, 2003.

[32] V. Lepetit, P. Lagger, and P. Fua. Randomized trees for real-time keypoint recognition. In *Proc. IEEE Conf. Computer Vision and Pattern Recognition*, volume 2, pages 775–781, June 2005.

[33] D.G. Lowe. Distinctive image features from scale-invariant keypoints. *Int. J. Computer Vision*, 60(2):91–110, November 2004.

[34] J. Malik, S. Belongie, T. Leung, and J. Shi. Contour and texture analysis for image segmentation. *Int. J. Computer Vision*, 43(1):7–27, June 2001.

[35] M. Marszałek and C. Schmid. Semantic hierarchies for visual object recognition. In *Proc. IEEE Conf. Computer Vision and Pattern Recognition*, June 2007.

[36] K. Mikolajczyk and C. Schmid. An affine invariant interest point detector. In A. Heyden, G. Sparr, and P. Johansen, editors, *Proc. European Conf. on Computer Vision*, volume LNCS 2350, pages 128–142. Springer, May 2002.

[37] J. Pearl. *Probabilistic Reasoning in Intelligent Systems*. San Mateo: Morgan Kaufmann, 1988.

[38] F. M. Porikli. Integral Histogram: A fast way to extract histograms in cartesian spaces. In *Proc. IEEE Conf. Computer Vision and Pattern Recognition*, volume 1, pages 829–836, June 2005.

[39] X. Ren, C. Fowlkes, and J. Malik. Figure/ground assignment in natural images. In A. Leonardis, H. Bischof, and A. Pinz, editors, *Proc. European Conf. on Computer Vision*, volume 2, pages 614–627, Graz, Austria, May 2006. Springer.

[40] C. Rother, L. Bordeaux, Y. Hamadi, and A. Blake. AutoCollage. *ACM Transactions on Graphics*, 25(3):847–852, July 2006.

[41] C. Rother, V. Kolmogorov, and A. Blake. GrabCut - interactive foreground extraction using iterated graph cuts. *ACM Transactions on Graphics*, 23(3):309–314, August 2004.

[42] B.C. Russel, A. Torralba, K.P. Murphy, and W.T. Freeman. LabelMe: a database and web-based tool for image annotation. Technical Report 25, MIT AI Lab, September 2005.

[43] J. Shotton, J. Winn, C. Rother, and A. Criminisi. *TextonBoost*: Joint appearance, shape and context modeling for multi-class object recognition and segmentation. In A. Leonardis, H. Bischof, and A. Pinz, editors, *Proc. European Conf. on Computer Vision*, volume LNCS 3951, pages 1–15. Springer, May 2006.

[44] C. Sutton and A. McCallum. Piecewise training of undirected models. In *Proc. Conf. on Uncertainty in Artificial Intelligence*, 2005.

[45] A. Torralba, K. P. Murphy, and W. T. Freeman. Sharing visual features for multiclass and multiview object detection. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 19(5):854–869, May 2007.

[46] Z. Tu, X. Chen, A.L. Yuille, and S.C. Zhu. Image parsing: unifying segmentation, detection, and recognition. In *Proc. Int. Conf. on Computer Vision*, volume 1, pages 18–25, Nice, France, October 2003.

[47] M. Varma and A. Zisserman. A statistical approach to texture classification from single images. *Int. J. Computer Vision*, 62(1-2):61–81, April 2005.

[48] P. Viola and M.J. Jones. Rapid object detection using a boosted cascade of simple features. In *Proc. IEEE Conf. Computer Vision and Pattern Recognition*, volume 1, pages 511–518, December 2001.

[49] J. Winn, A. Criminisi, and T. Minka. Categorization by learned universal visual dictionary. In *Proc. Int. Conf. on Computer Vision*, volume 2, pages 1800–1807, Beijing, China, October 2005.

[50] J. Winn and N. Jojic. LOCUS: Learning Object Classes with Unsupervised Segmentation. In *Proc. Int. Conf. on Computer Vision*, volume 1, pages 756–763, Beijing, China, October 2005.

[51] J. Winn and J. Shotton. The layout consistent random field for recognizing and segmenting partially occluded objects. In *Proc. IEEE Conf. Computer Vision and Pattern Recognition*, volume 1, pages 37–44, June 2006.

[52] J.S. Yedidia, W.T. Freeman, and Y. Weiss. *Understanding belief propagation and its generalizations*. Morgan Kaufmann Publishers Inc., 2003.

[53] P. Yin, A. Criminisi, J. Winn, and I. Essa. Tree based classifiers for bilayer video segmentation. In *Proc. IEEE Conf. Computer Vision and Pattern Recognition*, 2007.