Using the Forest to See the Trees: A Graphical Model Relating Features, Objects, and Scenes*

Kevin Murphy
MIT AI lab
Cambridge, MA 02139
`murphyk@ai.mit.edu`

Antonio Torralba
MIT AI lab
Cambridge, MA 02139
`torralba@ai.mit.edu`

William T. Freeman
MIT AI lab
Cambridge, MA 02139
`wtf@ai.mit.edu`

Abstract

Standard approaches to object detection focus on local patches of the image, and try to classify them as background or not. We propose to use the *scene context* (image as a whole) as an extra source of (global) information, to help resolve local ambiguities. We present a conditional random field for jointly solving the tasks of object detection and scene classification.

1 Introduction

Standard approaches to object detection (e.g., [25, 15]) usually look at local pieces of the image in isolation when deciding if the object is present or not at a particular location/ scale. However, this approach may fail if the image is of low quality (e.g., [24]), or the object is too small, or the object is partly occluded, etc. In this paper we propose to use the image as a whole as an extra global feature, to help overcome local ambiguities.

There is some psychological evidence that people perform rapid global scene analysis before conducting more detailed local object analysis [4, 2]. The key computational question is how to represent the whole image in a compact, yet informative, form. [22] suggests a representation, called the "gist" of the image, based on PCA of a set of spatially averaged filter-bank outputs. The gist acts as an holistic, low-dimensional representation of the whole image. They show that this is sufficient to provide a useful prior for what types of objects may appear in the image, and at which locations/scale.

We extend [22] by combining the prior suggested by the gist with the outputs of bottom-up, local object detectors, which are trained using boosting (see Section 2). Note that this is quite different from approaches that use joint spatial constraints between the locations of objects, such as [11, 21, 20, 8]. In our case, the spatial constraints come from the image as a whole, not from other objects. This is computationally much simpler.

Another task of interest is detecting if the object is present anywhere in the image, regardless of location. (This can be useful for object-based image retrieval.) In principle, this is straightforward: we declare the object is present iff the detector fires (at least once) at any location/scale. However, this means that a single false positive at the patch level can cause a 100% error rate at the image level. As we will see in Section 4, even very good

*This is an extended version of our NIPS 2003 paper, and was last updated on 17 January 2004.

detectors can perform poorly at this task. The gist, however, is able to perform quite well at suggesting the presence of types of objects, without using a detector at all. In fact, we can use the gist to decide if it is even “worth” running a detector, although we do not explore this here.

Often, the presence of certains types of objects is correlated, e.g., if you see a keyboard, you expect to see a screen. Rather than model this correlation directly, we introduce a hidden common cause/ factor, which we call the “scene”. In Section 5, we show how we can reliably determine the type of scene (e.g., office, corridor or street) using the gist. Scenes can also be defined in terms of the objects which are present in the image. Hence we combine the tasks of scene classification and object-presence detection using a tree-structured graphical model: see Section 6. We perform top-down inference (scenes to objects) and bottom-up inference (objects to scenes) in this model. Finally, we conclude in Section 7.

2 Object detection and localization

2.1 General approach

For object detection there are at least three families of approaches: parts-based (an object is defined as a specific spatial arrangement of small parts e.g., [6]), patch-based (we classify each rectangular image region as object or background), and region-based (a region of the image is segmented from the background and is described by a set of features that provide texture and shape information e.g., [5]).

Here we use a patch-based approach. For objects with rigid, well-defined shapes (screens, keyboards, people, cars), a patch usually contains the full object and a small portion of the background. For the rest of the objects (desks, bookshelves, buildings), rectangular patches may contain only a piece of the object. In that case, the region covered by a number of patches defines the object. In such a case, the object detector will rely mostly on the textural properties of the patch.

The main advantage of the patch-based approach is that object-detection can be reduced to a binary classification problem. Specifically, we compute $P(O_i^c = 1|v_i^c)$ for each class c and patch i (ranging over location and scale), where $O_i^c = 1$ if patch i contains (part of) an instance of class c, and $O_i^c = 0$ otherwise; v_i^c is the feature vector (to be described below) for patch i computed for class c.

To detect an object, we slide our detector across the image pyramid and classify all the patches at each location and scale (20% increments of size and every other pixel in location). After performing non-maximal suppression [1], we report as detections all locations for which $P(O_i^c|v_i^c)$ is above a threshold, chosen to given a desired trade-off between false positives and missed detections.

The overall detector takes about 2 s/frame/object using Matlab on 320x240 monochrome images. We could get substantial speedups if we trained a cascade, as in [25], and if we re-implemented in C.

2.2 Features for objects and scenes

We would like to use the same set of features for detecting a variety of object types, as well as for classifying scenes. Hence we will create a large set of features and use a feature selection algorithm (Section 2.3) to select the most discriminative subset.

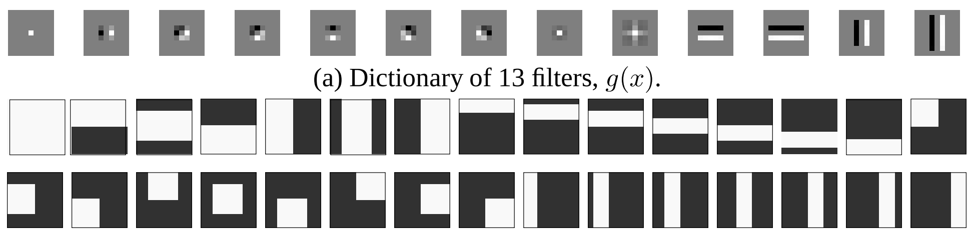
We compute a single feature k for image patch i in three steps, as follows. First we convolve the (monochrome) patch $I_i(x)$ with a filter $g_k(x)$, chosen from the set of 13 (zero-mean) filters shown in Figure 1(a). This set includes oriented edges, a Laplacian filter, corner detectors and long edge detectors. These features can be computed efficiently: The filters used can be obtained by convolution of 1D filters (for instance, the long edge filters are obtained by the convolution of the two filters $[-1\ 0\ 1]^T$ and $[1\ 1\ 1\ 1\ 1\ 1]$) or as linear combinations of the other filter outputs (e.g., the first six filters are steerable).

We can summarize the response of the patch convolved with the filter, $|I_i(x) * g_k(x)|$, using a histogram. For natural images, we can further summarize this histogram using just two statistics, the variance and the kurtosis [7]. Hence in step two, we compute $|I_i(x) * g_k(x)|^{\gamma_k}$, for $\gamma_k \in \{2, 4\}$. (The kurtosis is useful for characterizing texture-like regions.)

Often we are only interested in the response of the filter within a certain region of the patch. Hence we can apply one of 30 different spatial templates, which are shown in Figure 1(b). The use of a spatial template provides a crude encoding of "shape" inside the rectangular patch. We use rectangular masks because we can efficiently compute the average response of a filter within each region using the integral image [25].[1]

Summarizing, we can compute feature k for patch i as follows: $f_i(k) = \sum_x w_k(x) \left(|I(x) * g_k(x)|^{\gamma_k}\right)_i$. (To achieve some illumination invariance, we also standardize each feature vector on a per-patch basis.) The feature vector has size $13 \times 30 \times 2 = 780$ (the factor of 2 arises because we consider $\gamma_k = 2$ or 4).

Figure 2 shows some of the features selected by the learning algorithm for different kinds of objects. For example, we see that computer monitor screens are characterized by long horizontal or vertical lines on the edges of the patch, whereas buildings, seen from the outside, are characterized by cross-like texture, due to the repetitive pattern of windows.



(a) Dictionary of 13 filters, $g(x)$.

(b) Dictionary of 30 spatial templates, $w(x)$.

Figure 1: **(a) Dictionary of filters**. Filter 1 is a delta function, 2–7 are 3x3 Gaussian derivatives, 8 is a 3x3 Laplacian, 9 is a 5x5 corner detector, 10–13 are long edge detectors (of size 3x5, 3x7, 5x3 and 7x3). **(b) Dictionary of 30 spatial templates**. Template 1 is the whole patch, 2–7 are all sub-patches of size 1/2, 8–30 are all sub-patches of size 1/3.

2.3 Classifier

Following [25], our detectors are based on a classifier trained using boosting. There are many variants of boosting [10, 9, 18], which differ in the loss function they are trying to optimize, and in the gradient directions which they follow. We, and others [14], have found that GentleBoost [10] gives higher performance than AdaBoost [18], and requires fewer iterations to train, so this is the version we shall (briefly) present below.

The boosting procedure learns a (possibly weighted) combination of base classifiers, or "weak learners": $\alpha(v) = \sum_t \alpha_t h_t(v)$, where v is the feature vector of the patch, h_t is the

[1] The Viola and Jones [25] feature set is equivalent to using these masks plus a delta function filter; the result is like a Haar wavelet basis. This has the advantage that objects of any size can be detected without needing an image pyramid, making the system very fast. By contrast, since our filters have fixed spatial support, we need to down-sample the image to detect large objects.

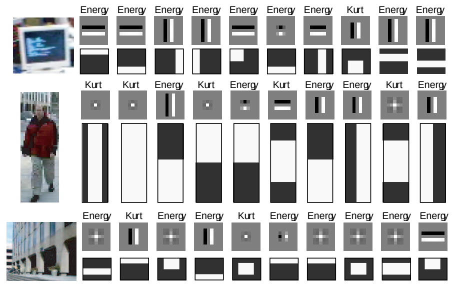


Figure 2: Some of the features chosen after 100 rounds of boosting for recognizing screens, pedestrians and buildings. Features are sorted in order of decreasing weight, which is a rough indication of importance. "Energy" means $\gamma_k = 2$ and "Kurt" (kurtosis) means $\gamma_k = 4$.

base classifier used at round t, and α_t is its corresponding weight. (GentleBoost, unlike AdaBoost, does not weight the outputs of the weak learners, so $\alpha_t = 1$.) For the weak classifiers we use regression stumps of the form $h(v) = a[v_f > \theta] + b$, where $[v_f > \theta] = 1$ iff component f of the feature vector v is above threshold θ. For most of the objects we used about 100 rounds of boosting. (We use a hold-out set to monitor overfitting.) See Figure 2 for some examples of the selected features.

As in decision tree induction, we can find the optimal feature f and threshold θ by exhaustive search (since the data has a finite number of split points). We can then find the corresponding optimal regression parameters a, b to minimize the weighted squared error on the training set:

$$E = \sum_i w_i \left(z_i - a[x_i > \theta] - b\right)^2$$

where $x_i = v_i(f)$ is component f of example $\vec{v}_i$, w_i is the normalized weight of example i at this round of boosting, and $z_i \in \{-1, +1\}$ is the target label. Partitioning the data into examples below and above the thresold, we get

$$E = \sum_{i \leq \theta} w_i (z_i - b)^2 + \sum_{i > \theta} w_i (z_i - a - b)^2$$

The optimal value of b is a weighted average of the labels for the examples below threshold, and the optimal value of $a + b$ is a weighted average of the labels for the examples above threshold:

$$b = \frac{\sum_{i \leq \theta} w_i z_i}{\sum_{i \leq \theta} w_i}, \quad a + b = \frac{\sum_{i > \theta} w_i z_i}{\sum_{i > \theta} w_i},$$

For most of the objects we used about 100 rounds of boosting. (We use a hold-out set to monitor overfitting.) The number of times that a features is used (and the weights given to it, in the case of AdaBoost) provide an indication of the features that are most discriminative of each object: see Figure 2 for some examples.

The output of a boosted classifier is a "confidence-rated prediction", α. We convert this to a probability using logistic regression: $P(O_i^c = 1 | \alpha(v_i^c)) = \sigma(w^T [1 \;\; \alpha])$, where $\sigma(x) = 1/(1 + \exp(-x))$ is the sigmoid function [17]. We compute the optimal maximum likelihood estimate of the weight vector w by a gradient method (IRLS) applied to a separate validation set (to avoid over confidence). We can then change the hit rate/false alarm rate of the detector by varying the threshold on $P(O = 1 | \alpha)$.

2.4 Training set

For training, we hand labeled about 2400 images (collected with a wearable webcam and with hand-held digital cameras) by drawing bounding boxes around a subset of the objects in each image; we also specified the scene label. (For pedestrians, we used the MIT-CBCL data set.[2]) The resulting data set has 5–500 examples of each class. We have trained detectors for the following classes, which have at least 100 examples each: screens, keyboards, bookshelves, desks, standing people, and cars. (Each detector is trained for a specific view, such as front, back or side of the object.) In addition, we created a large set of negative examples ($>$ 10,000) chosen randomly from the regions of the training set known not to contain the object of interest. (We did not use bootstrap selection [15], although this would undoubtedly improve performance.)

2.5 Performance of the classifier

Figure 3 summarizes the performances of the detectors for a set of objects on isolated patches (not whole images) taken from the test set. The results vary in quality since some objects are harder to recognize than others, and because some objects have less training data. When we trained and tested our detector on the training/testing sets of side-views of cars from UIUC[3], we outperformed the detector of [1] at every point on the precision-recall curve (results not shown), suggesting that our base-line detectors can match state-of-the-art detectors when given enough training data.

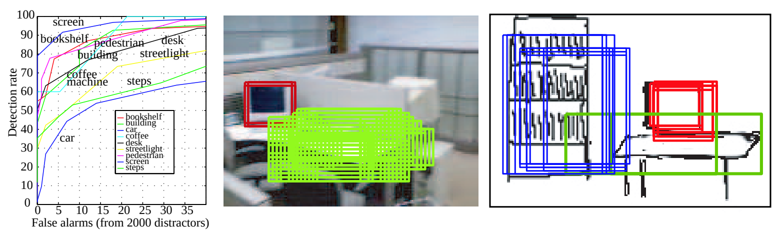


Figure 3: a) ROC curves for 9 objects; we plot hit rate vs number of false alarms, when the detectors are run on isolated test patches. in order to illustrate the range b) Example of the detector output on one of the test set images, before non-maximal suppression. c) Example of the detector output on a line drawing of a typical office scene. The system correctly detects the screen, the desk and the bookshelf.

3 Improving object localization by using the gist

One way to improve the speed and accuracy of a detector is to reduce the search space, by only running the detector in locations/ scales that we expect to find the object. The expected location/scale can be computed on a per image basis using the gist, as we explain below. (Thus our approach is more sophisticated than having a fixed prior, such as “keyboards always occur in the bottom half of an image”.)

If we only run our detectors in a predicted region, we risk missing objects. Instead, we run our detectors everywhere, but we penalize detections that are far from the predicted location/scale. Thus objects in unusual locations have to be particularly salient (strong local detection score) in order to be detected, which accords with psychophysical results of human observers.

[2]http://www.ai.mit.edu/projects/cbcl/software-datasets/

[3]http://l2r.cs.uiuc.edu/$\sim$cogcomp/Data/Car/

3.1 Computing the gist

We define the gist as a feature vector summarizing the whole image, and denote it by v_G. One way to compute this is to treat the whole image as a single patch, and to compute a feature vector for it as described in Section 2.2. If we use 4 image scales and 7 spatial masks, the gist will have size $13 \times 7 \times 2 \times 4 = 728$. Even this is too large for some methods, so we consider another variant that reduces dimensionality further by using PCA on the gist-minus-kurtosis vectors. Following [23, 22], we take the first 80 principal components; we call this the PCA-gist.

3.2 Predicting location/scale using the gist

We can predict the expected location/scale of objects of class c given the gist, $E[X^c|v^G]$, by using a regression procedure. We have tried linear regression, boosted regression [9], and cluster-weighted regression [22]; all approaches work about equally well.

Using the gist it is easy to distinguish long-distance from close-up shots (since the overall structure of the image looks quite different), and hence we might predict that the object is small or large respectively. We can also predict the expected height. However, we cannot predict the expected horizontal location, since this is typically unconstrained by the scene. (It *is* constrained by other objects, a fact which is exploited in works such as [11, 21, 20, 8]. However, these methods are much more complicated than what we are about to describe.)

We considered three different regression methods: (1) boosted regression stumps applied to the gist, where the goal is to minimize the squared error instead of the classification error [9]; (2) linear regression applied to PCA-gist; (3) cluster weighted regression (a varient of mixtures of experts) applied to PCA-gist, which was shown to be effective in [22]. (Note that linear regression is a special case of cluster weighted regression in which we only use one cluster; linear regression has the advantage that there is a closed form solution, so one does not need EM.)

We found that linear regression and boosting worked about equally well. The advantage of the boosting approach is that it is very similar to the strategy that we use for object detection, simplifying the code. Also, this method is capable of doing feature selection. The advantage of linear regression is that it is simple and fast to train, and it tends to work slightly better. Our experience with cluster-weighted regression was disappointing: the model needs to be fit using EM, which is prone to local minima and numerical problems caused by shrinking covariance matrices.

3.3 Combining the predicted location/scale with the detector outputs

To combine the local and global sources of information, we construct a feature vector f which combines the output of the boosted detector, $\alpha(v_i^c)$, and the vector between the location of the patch and the predicted location for objects of this class, $x_i^c - \hat{x}^c$. We then train another classifier to compute $P(O_i^c = 1|f(\alpha(v_i^c), x_i^c, \hat{x}^c))$ using either boosting or logistic regression. In Figure 4, we compare localization performance using just the detectors, $P(O_i^c = 1|\alpha(v_i^c)$, and using the detectors and the predicted location, $P(O_i^c = 1|f(\alpha(v_i^c), x_i^c, \hat{x}^c))$. For keyboards (which are hard to detect) we see that using the predicted location helps a lot, whereas for screens (which are easy to detect), the location information does not help.

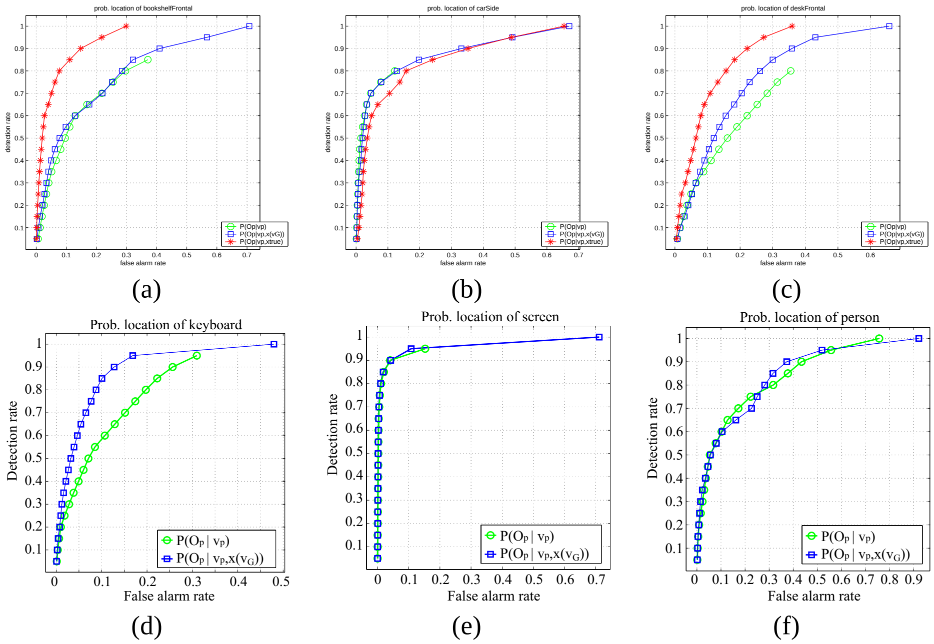


Figure 4: ROC curves for detecting the location of objects in the image: (a) bookshelf, (b) car, (c) desk, (d) keyboard, (e) person, (f) computer screen. The green circles are the local detectors alone, and the blue squares are the detectors and predicted location. (The red lines with stars can be ignored.)

4 Object presence detection

We can compute the probability that the object exists anywhere in the image (which can be used for e.g., object-based image retrieval) by taking the OR of all the detectors:

$$P(E^c = 1|v^c_{1:N}) = \vee_i P(O^c = 1|v^c_{1:N}).$$

Unfortunately, this leads to massive overconfidence, since the patches are not independent. As a simple approximation, we can use

$$P(E^c = 1|v^c_{1:N}) \approx \max_i P(E^c = 1|v^c_{1:N}) = P(E^c = 1|\max_i \alpha_i(v^c_i)) = P(E^c = 1|\alpha^c_{max}).$$

Unfortunately, even for good detectors, this can give poor results: the probability of error at the image level is $1 - \prod_i (1 - q_i) = 1 - (1 - q)^N$, where q is the probability of error at the patch level and N is the number of patches. For a detector with a reasonably low false alarm rate, say $q = 10^{-4}$, and $N = 5000$ patches, this gives a 40% false detection rate at the image level! For example, see the reduced performance at the image level of the screen detector (Figure 5(a)), which performs very well at the patch level (Figure 4(a)).

An alternative approach is to use the gist to predict the presence of the object, without using a detector at all. This is possible because the overall structure of the image can suggest what kind of scene this is (see Section 5), and this in turn suggests what kinds of objects are present (see Section 6). We trained another boosted classifier to predict $P(E^c = 1|v^G)$; results are shown Figure 5. For poor detectors, such as keyboards, the gist does a much better job than the detectors, whereas for good detectors, such as screens, the results are comparable. Finally, we can combine both approaches by constructing a feature vector from the output of the global and local boosted classifiers and using logistic regression:

$$P(E^c = 1|v^G, v^c_{1:N}) = \sigma(w^T[1 \;\; \alpha(v^G)\alpha^c_{max}]).$$

However, this seems to offer little improvement over the gist alone (see Figure 5), presumably because our detectors are not very good.

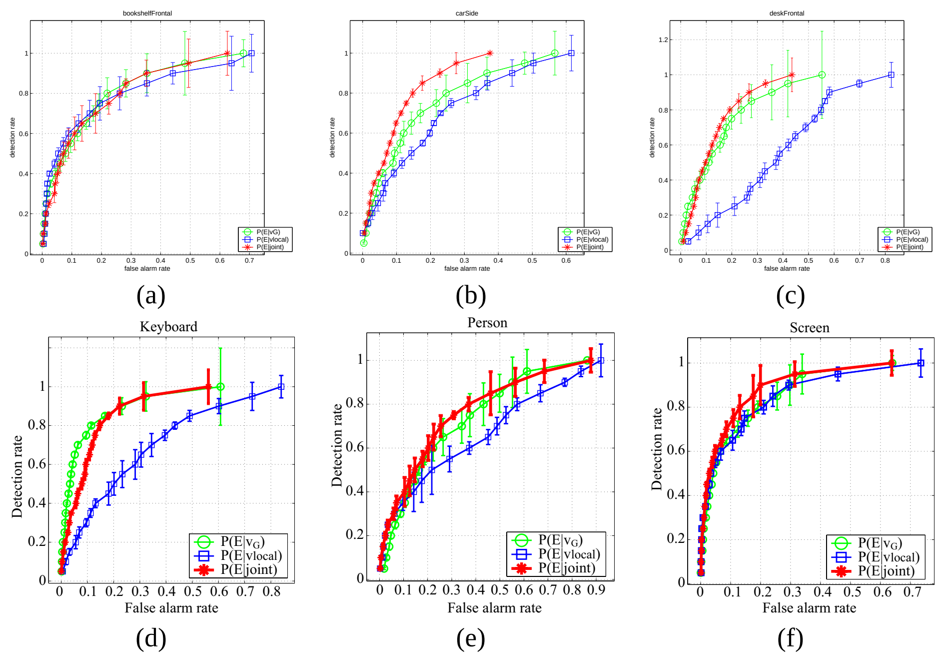


Figure 5: ROC curves for detecting the presence of object classes in the image: (a) bookshelf, (b) car, (c) desk, (d) keyboard, (e) person, (f) computer screen. The green circles use the gist alone, the blue squares use the detectors alone, and the red stars use the joint model, which uses the gist and all the detectors from all the object classes.

5 Scene classification

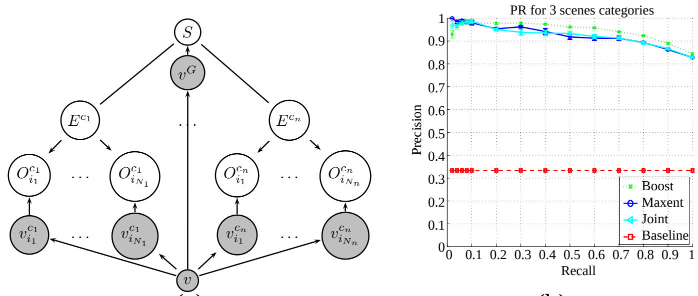
As mentioned in the introduction, the presence of many types of objects is correlated. Rather than model this correlation directly, we introduce a latent common "cause", which we call the "scene". We assume that object presence is conditionally independent given the scene, as explained in Section 6. But first we explain how we recognize the scene type, which in this paper can be office, corridor or street.

The approach we take to scene classification is simple. We train a one-vs-all binary classifier for recognizing each type of scene using boosting applied to the gist.[4] Then we normalize the results: $P(S = s|v^G) = \frac{P(S^s=1|v^G)}{\sum_{s'} P(S^{s'}=1|v^G)}$ where $P(S^s = 1|v^G)$ is the output of the s-vs-other classifier.[5] This approach works very well: see Figure 6(b).

In previous work [23], we considered the problem of classifying scenes collected with a wearable camera. We found that many of our images were ambiguous, e.g., a close-up of a white wall, as a person turns around in a narrow corridor. However, by using an HMM to exploit temporal context, we were able to resolve such ambiguities.

[4] An alternative would be to use the multi-class LogitBoost algorithm [10]. However, training separate one-vs-all classifiers allows them to have different internal structure (e.g., number of rounds).

[5] For scenes, it is arguably more natural to allow multiple labels, as in [3], rather than forcing each scene into a single category; this can be handled with a simple modification of boosting [19].



(a) (b)

Figure 6: **(a)** Graphical model for scene and object recognition. This is a conditional density models, of the form $P(\text{Hidden}|\text{obs})$, so the arcs are directed away from the observable variables (shaded) towards the hidden variables (clear). The visible variables are: v the whole image, v^G the gist, v_i^c a feature vector for class c and patch i. The hidden variables are: O_i^c true iff an instance (or part) of class c is in location i; E^c true iff an instance of class c is present anywhere in the image; S the scene type. The constants are: $n = 6$ is the number of object classes, $N_c \sim 5000$ is the number of patches for class c. Other terms are defined in the text. **(b)** Precision-recall curve for scene classification.

In this paper, we ignore temporal information, and we only try to classify "interesting" scenes (i.e., ones which have been annotated as containing objects). For long-distance shots, the gist is very good at classifying the scene, but for close-ups, it is the objects in the image, rather than the overall image "texture", that defines the scene. This suggests a model which uses both objects and the gist for scene classification, which we discuss below.

6 Joint scene classification and object-presence detection

We now discuss how we can use scene classification to facilitate object-presence detection, and vice versa. The approach is based on the tree-structured graphical model[6] in Figure 6(a), which encodes our assumption that the objects are conditionally independent given the scene.

6.1 Model definition

This graphical model encodes the following conditional joint density:

$$P(S, E^{1:n}, O^{c}_{1:N}, \ldots, O^{c_n}_{1:N}|v) = \frac{1}{Z} P(S|v^G) \prod_c \phi(E^c, S) \prod_i P(O_i^c|E^c, v_i^c)$$

where v^G and v_i^c are deterministic functions of the image v and Z is a normalizing constant. called the partition function (which is tracatable to compute, since the graph is a tree). By conditioning on the observations as opposed to generating them, we are free to incorporate arbitrary, possibly overlapping features (local and global), without having to make strong independence assumptions c.f., [13, 12].

We now define the individual terms in this expression. $P(S|v^G)$ is the output of boosting as described in Section 5. $\phi(E^c, S)$ is essentially a table which counts the number of times object type c occurs in scene type S. Finally, we define

$$P(O_i^c = 1|E^c = e, v_i^c) = \begin{cases} \sigma(w^T[1 \;\; \alpha(v_i^c)]) & \text{if } e = 1 \\ 0 & \text{if } e = 0 \end{cases}$$

[6] The graph is a tree once we remove the observed nodes.

This means that if we know the object is absent in the image ($E^c = 0$), then all the local detectors should be turned off ($O_i^c = 0$); but if the object is present ($E^c = 1$), we do not know where, so we allow the local evidence, v_i^c, to decide which detectors should turn on. We can find the maximum likelihood estimates of the parameters of this model by training it jointly using a gradient procedure; see the long version of this paper for details.

6.2 Inference

Since the detector nodes, O_i^c, are hidden nodes, and have no observed *children* (the v_i^c nodes are parents), they do not pass messages upwards to their parents E^c [16]. (Such upward messages would in any case cause overconfidence in E^c due to the fact that we did not model dependencies between the detectors.) The formal reason for this is that the O_i^c nodes are at the bottom of a hidden v-structure, so they block information from passing from the local patches v_i^c to the presence detector, E^c. However, we are free to pass information from the local detectors to E^c directly (e.g., via the $v_{max}^c = \max_i \alpha(v_i^c)$ node), since all hidden nodes are conditioned on all the visible ones.

The benefit of the d-separation discussed above is that we can perform inference on the undirected subgraph separately from the directed leaves. In particular, we can perform a bottom-up pass from the E^c nodes to the root, S, and then a top-down pass from the root back to the E^c nodes. Once we have established the probabilility of object presence given global factors, we can proceed to localize the object using

$$P(O_i^c = 1|v) = \sum_e P(O_i^c = 1|v_i^c, E^c = e)P(E^c = e|v) = P(O_i^c = 1|v_i^c, E^c 1)P(E^c = 1|v)$$

where $P(E^c = 1|v)$ is computed using inference in the tree. Thus global factors can serve to suppress over-active local detectors.

6.3 Learning

We can fit $P(O_i^c = 1|v_i^c, E_c = 1)$ using logistic regression applied to the boosted detectors, as described in Section 2.3. The remaining parameters ($\phi(S, E^c)$ and $P(S|\alpha(v^G))$) can be estimated jointly by performing gradient descent on the log-likelihood [13]; we have found quasi-Newton methods to be effective. Since this is a convex optimization problem, we are guaranteed to reach the global optimum.

6.4 Results

In Figure 5, we see that we can reliably detect In Figure 5, we see that we can reliably detect the presence of the object in an image without using the gist directly, providing we know what the scene type is (the red curve, derived from the joint model in this section, is basically the same as the green curve, derived from the gist model in Section 4). The importance of this is that it is easy to label images with their scene type, and hence to train $P(S|v^G)$, but it is much more time consuming to annotate objects, which is required to train $P(E^c|v^G)$.[7]

In Figure 6(b), we see that the joint model does not do any better at scene classification than just using the gist alone. This is probably because we just have 6 object classes, and their detectors are not very reliable, so the signal from the E^c nodes to S is less reliable than than from v^G. However, with a larger number of detectors, we would expect joint object-presence detection and scene recognition to outperform isolated classification.

[7] Since we do not need to know the location of the object in the image in order to train $P(E^c|v^G)$, we can use partially annotated data such as image captions, as used in [5].

7 Conclusions and future work

We have shown how to combine global and local image features to solve the tasks of object detection and scene recognition. In the future, we plan to try a larger number of object classes. Also, we would like to investigate methods for choosing which order to run the detectors. For example, one can imagine a scenario in which we run the screen detector first (since it is very reliable); if we discover a screen, we conclude we are in an office, and then decide to look for keyboards and chairs; but if we don't discover a screen, we might be in a corridor or a street, so we choose to run another detector to disambiguate our belief state. This corresponds to a dynamic message passing protocol on the graphical model.

References

[1] S. Agarwal and D. Roth. Learning a sparse representation for object detection. In *Proc. European Conf. on Computer Vision*, 2002.

[2] I. Biederman. On the semantics of a glance at a scene. In M. Kubovy and J. Pomerantz, editors, *Perceptual organization*, pages 213–253. Erlbaum, 1981.

[3] M. Boutell, X. Shen, J. Luo, and C. Brown. Multi-label semantic scene classification. Technical report, Dept. Comp. Sci. U. Rochester, 2003.

[4] D. Davon. Forest before the trees: the precedence of global features in visual perception. *Cognitive Psychology*, 9:353–383, 1977.

[5] Pinary Duygulu, Kobus Barnard, Nando de Freitas, David Forsyth, and Michael I. Jordan. Object recognition as machine translation: Learning a lexicon for a fixed image vocabulary. In *Proc. European Conf. on Computer Vision*, 2002.

[6] R. Fergus, P. Perona, and A. Zisserman. Object class recognition by unsupervised scale-invariant learning. In *Proc. IEEE Conf. Computer Vision and Pattern Recognition*, 2003.

[7] D. Field. Relations between the statistics of natural images and the response properties of cortical cells. *J. Opt. Soc. Am.*, A4:2379–2394, 1987.

[8] M. Fink and P. Perona. Mutual boosting for contextual influence. In *Advances in Neural Info. Proc. Systems*, 2003.

[9] J. Friedman. Greedy function approximation: a gradient boosting machine. *Annals of Statistics*, 29:1189–1232, 2001.

[10] J. Friedman, T. Hastie, and R. Tibshirani. Additive logistic regression: a statistical view of boosting. *Annals of statistics*, 28(2):337–374, 2000.

[11] R. Haralick. Decision making in context. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 5:417–428, 1983.

[12] Sanjiv Kumar and Martial Hebert. Discriminative random fields: A discriminative framework for contextual interaction in classification. In *IEEE Conf. on Computer Vision and Pattern Recognition*, 2003.

[13] J. Lafferty, A. McCallum, and F. Pereira. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *Intl. Conf. on Machine Learning*, 2001.

[14] R. Lienhart, A. Kuranov, and V. Pisarevsky. Empirical analysis of detection cascades of boosted classifiers for rapid object detection. In *DAGM 25th Pattern Recognition Symposium*, 2003.

[15] C. Papageorgiou and T. Poggio. A trainable system for object detection. *Intl. J. Computer Vision*, 38(1):15–33, 2000.

[16] J. Pearl. *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*. Morgan Kaufmann, 1988.

[17] J. Platt. Probabilistic outputs for support vector machines and comparisons to regularized likelihood methods. In A. Smola, P. Bartlett, B. Schoelkopf, and D. Schuurmans, editors, *Advances in Large Margin Classifiers*. MIT Press, 1999.

[18] R. Schapire. The boosting approach to machine learning: An overview. In *MSRI Workshop on Nonlinear Estimation and Classification*, 2001.

[19] Robert E. Schapire and Yoram Singer. BoosTexter: A boosting-based system for text categorization. *Machine Learning*, 39(2/3):135–168, 2000.

[20] A. Singhal, Jiebo Luo, and Weiyu Zhu. Probabilistic spatial context models for scene content understanding. In *Proc. IEEE Conf. Computer Vision and Pattern Recognition*, 2003.

[21] T. M. Strat and M. A. Fischler. Context-based vision: recognizing objects using information from both 2-D and 3-D imagery. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 13(10):1050–1065, 1991.

[22] A. Torralba. Contextual priming for object detection. *Intl. J. Computer Vision*, 53(2):153–167, 2003.

[23] A. Torralba, K. Murphy, W. Freeman, and M. Rubin. Context-based vision system for place and object recognition. In *Intl. Conf. Computer Vision*, 2003.

[24] A. Torralba and P. Sinha. Detecting faces in impoverished images. Technical Report 028, MIT AI Lab, 2001.

[25] Paul Viola and Michael Jones. Robust real-time object detection. *International Journal of Computer Vision - to appear*, 2002.