Object Detection and Localization Using Local and Global Features

Kevin Murphy[1], Antonio Torralba[2], Daniel Eaton[1], and William Freeman[2]

[1] Department of Computer Science, University of British Columbia
[2] Computer Science and AI Lab, MIT

Abstract. Traditional approaches to object detection only look at local pieces of the image, whether it be within a sliding window or the regions around an interest point detector. However, such local pieces can be ambiguous, especially when the object of interest is small, or imaging conditions are otherwise unfavorable. This ambiguity can be reduced by using global features of the image — which we call the "gist" of the scene — as an additional source of evidence. We show that by combining local and global features, we get significantly improved detection rates. In addition, since the gist is much cheaper to compute than most local detectors, we can potentially gain a large increase in speed as well.

1 Introduction

The most common approach to generic[1] object detection/ localization is to slide a window across the image (possibly at multiple scales), and to classify each such local window as containing the target or background. This approach has been succesfully used to detect rigid objects such as faces and cars (see e.g., [26,24,27,35]), and has even been applied to articulated objects such as pedestrians (see e.g., [20,36]). A natural extension of this approach is to use such sliding window classifiers to detect object parts, and then to assemble the parts into a whole object (see e.g., [19,20]). Another popular approach is to extract local interest points from the image, and then to classify each of the regions around these points, rather than looking at all possible subwindows (see e.g., [5,11]).

A weakness shared by all of the above approaches is that they can fail when local image information is insufficient e.g. because the target is very small or highly occluded. In such cases, looking at parts of the image outside of the patch to be classified — that is, by using the context of the image as a whole — can help. This is illustrated in Figure 1.

[1] By generic detection, we mean detecting classes (categories) of objects, such as any car, any face, etc. rather than finding a specific object (class instance), such as a particular car, or a particular face. For one of the most succesful approaches to the instance-level detection problem, see [18]. The category-level detection problem is generally considered harder, because of the need to generalize over intra-class variation. That is, approaches which memorize idiosyncratic details of an object (such as particular surface pattern or texture) will not work; rather, succesful techniques need to focus on generic object properties such as shape.

J. Ponce et al. (Eds.): Toward Category-Level Object Recognition, LNCS 4170, pp. 382–400, 2006.

Fig. 1. An image blob can be interpreted in many different ways when placed in different contexts. The blobs in the circled regions have identical pixel values (except for rotation), yet take on different visual appearances depending on their context within the overall image. (This image is best viewed online.)

An obvious source of context is other objects in the image (see e.g., [10,30], [31,7,15] for some recent examples of this old idea), but this introduces a chicken-and-egg situation, where objects are mutually dependent. In this chapter, we consider using global features of the image — which we call the "gist" of the image — as a source of context. There is some psychological evidence [22,3,28] that people use such global scene factors before analysing the image in detail.

In [23,34], Oliva and Torralba showed how one can use the image gist to predict the likely location and scale of an object. without running an object detector. In [21], we showed that combining gist-based priming with standard object detection techniques based on local image features lead to better accuracy, at negligible extra computational cost. This chapter is an extension of [21]: we provide a more thorough experimental comparison, and demonstrate much improved performance.[2]

[2] These improvements are due to various changes: first, we use better local features; second, we prepare the dataset more carefully, fixing labeling errors, ensuring the objects in the test set are large enough to be detected, etc; finally, we have subsantially simplified the model, by focusing on single-instance object localization, rather than pixel labeling i.e., we try to estimate the location of one object, $P(X = i)$, rather than trying to classify every pixel, $P(C_i = 1)$; thus we replace N binary variables with one N-ary variable. Note that in this chapter, in order to focus on the key issue of local vs global features, we do not address the scene categorization problem; we therefore do not need the graphical model machinery used in [21].

We consider two closely related tasks: Object-presence detection and object localization. Object-presence detection means determining if one or more instances of an object class are present (at any location or scale) in an image. This is sometimes called "image classification", and can be useful for object-based image retrieval. Formally we define it as estimating $P(O = 1|f(I))$, where $O = 1$ indicates the presence of class O and $f(I)$ is a set of features (either local or global or both) extracted from the image.

Object localization means finding the location and scale of an object in an image. Formally we define this as estimating $P(X = i|f(I))$, where $i \in \{1, \ldots, N\}$ is a discretization of the set of possible locations/ scales, so $\sum_i P(X = i|\cdot) = 1$. If there are multiple instances of an object class in an image, then $P(X|\cdot)$ may have multiple modes. We can use non-maximal suppression (with radius r, which is related to the expected amount of object overlap) to find these, and report back all detections which are above threshold. However, in this chapter, we restrict our attention to single instance detection.

Table 1. Some details on the dataset: number of positive (+) and negative (-) images in the training, validation and testing sets (each of which had 668, 132 and 537 images respectively). We also show the size of the bounding box which was used for training the local classifier.

	Train +	Train -	Valid +	Valid -	Test +	Test -	Size (hxw)
Screen	247	421	49	84	199	337	30x30
Keyboard	189	479	37	95	153	384	20x66
CarSide	147	521	29	104	119	417	30x80
Person	102	566	20	113	82	454	60x20

For training/testing, we used a subset of the MIT-CSAIL database of objects and scenes[3], which contains about 2000 images of indoor and outdoor scenes, in which about 30 different kinds of objects have been manually annotated. We selected images which contain one of the following 4 object classes: computer screens (front view), keyboards, pedestrians, and cars (side view). (These classes were chosen because they had enough training data.) We then cropped and scaled these so that each object's bounding box had the size indicated in Table 1. The result is about 668 training images and 537 testing images, most of which are about 320x240 pixels in size.

The rest of the chapter is structured as follows. In Section 2, we will discuss our implementation of the standard technique of object detection using sliding window classifiers applied to local features. In Section 3, we will discuss our implementation of the ideas in [34] concerning the use of global image features for object priming. In Section 4, we discuss how we tackle the object presence detection problem, using local and global features. In Section 5, we discuss how we tackle the object localization problem, using local and global features. Finally, in Section 6, we conclude.

[3] `http://web.mit.edu/torralba/www/database.html`

2 Object Detection Using Local Image Features

The standard approach to object detection is to classify each image patch/ window as foreground (containing the object) or background. There are two main decisions to be made: what kind of local features to extract from each patch, and what kind of classifier to apply to this feature vector. We discuss both of these issues below.

2.1 Feature Dictionary

Following standard practice, we first convolve each image with a bank of filters (shown in Figure 2). These filters were chosen by hand, but are similar to what many other groups have used. After filtering the images, we then extract image fragments from one of the filtered outputs (chosen at random). The size and location of these fragments is chosen randomly, but is constrained to lie inside the annotated bounding box. (This approach is similar to the random intensity patches used in [37], and the random filtered patches used in [29].) We record the location from which the fragment was extracted by creating a spatial mask centered on the object, and placing a blurred delta function at the relative offset of the fragment. This process is illustrated in Figure 3. We repeat this process for multiple filters and fragments, thus creating a large ($N \sim 150$) dictionary of features. Thus the i'th dictionary entry consists of a filter, f_i, a patch fragment P_i, and a Gaussian mask g_i. We can create a feature vector for every pixel in the image in parallel as follows:

$$v_i = [(I * f_i) \otimes P_i] * g_i$$

where $*$ represents convolution, $\otimes$ represents normalized cross-correlation and $v_i(x)$ is the i'th component of the feature vector at pixel x. The intuition behind this is as follows: the normalized cross-correlation detects places where patch P_i occurs, and these "vote" for the center of the object using the g_i masks (c.f., the Hough transform). Note that the $D \sim 10$ positive images used to create the dictionary of features are not used for anything else.

Fig. 2. The bank of 13 filters. From left to right, they are: a delta function, 6 oriented Gaussian derivatives, a Laplace of Gaussian, a corner detector, and 4 bar detectors.

2.2 Patch Classifier

Popular classifiers for object detection include SVMs [20], neural networks [26], naive Bayes classifiers [22], boosted decision stumps [26], etc. We use boosted

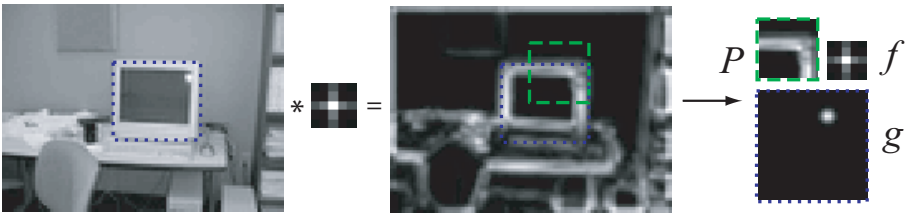

Fig. 3. Creating a random dictionary entry consisting of a filter f, patch P and Gaussian mask g. Dotted blue is the annotated bounding box, dashed green is the chosen patch. The location of this patch relative to the bounding box is recorded in the g mask.

decision stumps[4], since they have been shown to work well for object detection [26,17] they are easy to implement, they are fast to train and to apply, and they perform feature selection, thus resulting in a fairly small and interpretable classifier.

We used the gentleBoost algorithm [9], because we found that it is more numerically stable than other confidence-rated variants of boosting; a similar conclusion was reached in [14].

Our training data for the classifier is created as follows. We compute a bank of features for each labeled image, and then sample the resulting filter "jets" at various locations: once near the center of the object (to generate a positive training example), and at about 20 random locations outside the object's bounding box (to generate negative training examples): see Figure 4. We repeat this for each training image. These feature vectors and labels are then passed to the classifier.

We perform 50 rounds of boosting (this number was chosen by monitoring performance on the validation set). It takes 3–4 hours to train each classifier (using about 700 images); the vast majority of this time is spent computing the feature vectors (in particular, performing the normalized cross correlation).[5] The resulting features which are chosen for one of the classes are shown in Figure 5. (Each classifier is trained independently.)

Once the classifier is trained, we can apply it to a novel image at multiple scales, and find the location of the strongest response. This takes about 3 seconds for an image of size 240x320.

The output of the boosted classifier is a score b_i for each patch, that approximates $b_i \approx \log P(C_i = 1|I_i)/P(C_i = 0|I_i)$, where I_i are the features extracted from image patch I_i, and C_i is the label (foreground vs background) of patch i. In order to combine different information sources, we need to convert the output of the discriminative classifier into a probability. A standard way to do this [25] is by taking a sigmoid transform: $s_i = \sigma(w^T[1 \;\; b_i]) = \sigma(w_1 + w_2 b_i)$,

[4] A decision stump is a weak learner of the form $h(v) = a\delta(v_i > \theta) + b$, where v_i is the i'th dimension (feature) of v, θ is a threshold, a is a regression slope and b an offset.

[5] All the code is written in matlab, except for normalized cross-correlation, for which we use OpenCV, which is written in C++.

$$[(I * f) \otimes P] * g$$

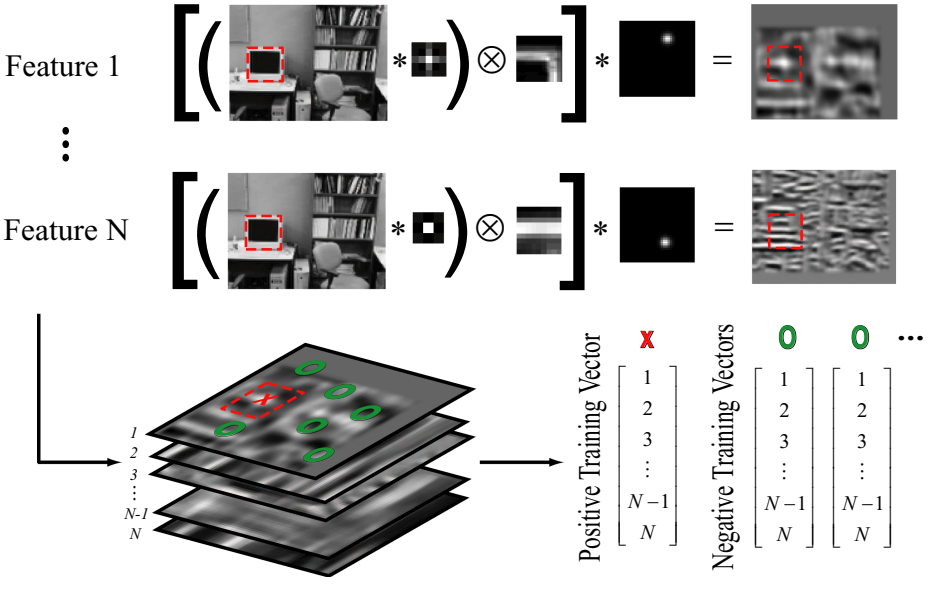

Fig. 4. We create positive (X) and negative (O) feature vectors from a training image by applying the whole dictionary of $N = 150$ features to the image, and then sampling the resulting "jet" of responses at various points inside and outside the labeled bounding box

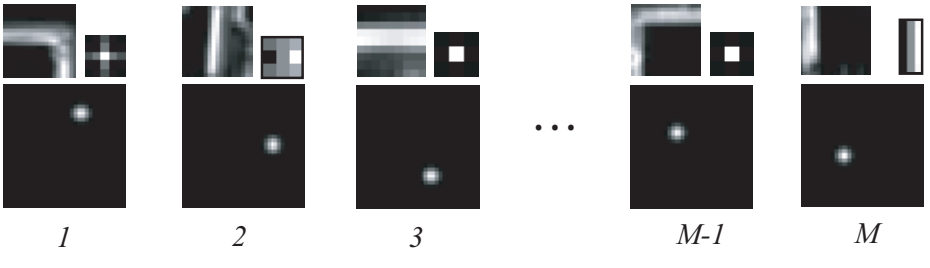

Fig. 5. Some of the $M = 50$ features which were chosen from the dictionary for the screen classifier. Within each group, there are 3 figures, representing (clockwise from upper left): the filtered image data; the filter; the location of that feature within the analysis region.

where the weights w are fit by maximum likelihood on the validation set, so that $s_i \approx P(C_i = 1|I_i)$. This gives us a per-patch probability; we will denote this vector of local scores computed from image I as $L = L(I)$. Finally we convert this into a probability distribution over possible object locations by normalizing:

$P(X = i|L) = s_i/(\sum_{j=1}^{N} s_j)$, so that $\sum_i P(X = i|L) = 1$. (None of these operations affect the performance curves, since they are monotonic transformations of the original classifier scores b_i, but they will prove useful later.)

2.3 Results

To illustrate performance of our detector, we applied it to a standard dataset of side views of cars[6]. In Figure 6, we show the performance of our car detector on the single scale dataset used in [2] and the multiscale dataset used in [1]. (Note that the multiscale dataset is much harder, as indicated by the decreased performance of both methods.) This shows that our local features, and our boosted classifier, provide a high quality baseline, which we will later extend with global features.

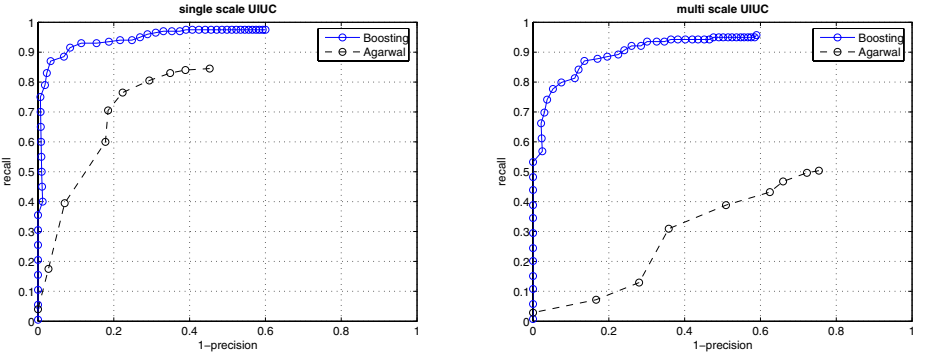

Fig. 6. Localization performance features on UIUC carSide data. (Left) Single scale. (Right) Multi scale. Solid blue circles (upper line): our approach based on boosted classifiers applied to local features; Dashed black circles (lower line): Agarwal's approach, based on a different kind of classifier and different local features. See Section 5.1 for an explanation of precision-recall curves.

3 Object Detection Using Global Image Features

3.1 The Gist of an Image

We compute the gist of an image using the procedure described in [32]. First we compute a steerable pyramid transformation, using 4 orientations and 2 scales; second we divide the image into a 4x4 grid, and compute the average energy of each channel in each grid cell, giving us $4 \times 2 \times 4 \times 4 = 128$ features; finally, we reduce dimensionality by performing PCA, and taking the first 80 dimensions. We will denote the resulting global feature vector derived from image I by $G = G(I)$. Note that this procedure is similar to the one used in [34], except in that chapter, Torralba used Gabor filters instead of steerable pyramids.

[6] `http://l2r.cs.uiuc.edu/~cogcomp/Data/Car/`

We have found both methods to work equally well, and mainly chose steerable pyramids because there is a good implementation available in Matlab/C.[7] We have also performed some preliminary experiments where we replace the steerable pyramids with the 13 filters in Figure 2; this has the advantage that some of the work involved in computing L and G can be shared. Performance seems to be comparable; however, the results in this chapter are based on the steerable pyramid version of gist.

The gist captures coarse texture and spatial layout of an image. This is illustrated in Figure 7, where we show a real image I and a noise image J, both of which have roughly the same gist, i.e., $G(I) \approx G(J)$. (J was created by initializing it to a random image, and then locally perturbing it until $||G(I) - G(J)||$ was minimized.)

Fig. 7. An illustration of the gist of an image. Top row: original image I; bottom row: noise image J for which $\text{gist}(I) = \text{gist}(J)$. We see that the gist captures the dominant textural features of the overall image, and their coarse spatial layout. (This figure is best viewed online.)

3.2 Location Priming Using the Gist

As shown in [34], it is possible to predict the rough location and scale of objects based on the gist, before applying a local detector. We will denote this as $P(X|G)$. (Note that this is typically much more informative than the unconditional prior marginal, $P(X)$: see Figure 9.) This information can be used in

[7] `http://www.cns.nyu.edu/~eero/STEERPYR/`

two ways: We can either threshold $P(X|G)$ and apply the local detector only to the locations deemed probable by the gist (to increase speed), or we can can apply the detector to the whole image and then combine $P(X|L)$ and $P(X|G)$ (to increase accuracy). In this chapter, we adopt the latter approach. We will discuss how we perform the combination in Section 5.2, but first we discuss how to compute $P(X|G)$.

Location priming learns a conditional density model of the form $p(X = x, y, s|G)$. Following [34], we assume scale and location are conditionally independent, and learn separate models for $p(x, y|G)$ and $p(s|G)$. As shown in [34], we can predict the y value of an object class from the gist reasonably well, but it is hard to predict the x value; this is because the height of an object in the image is correlated with properties which can be inferred from the gist, such as the depth of field [33], location of the ground plane, etc., whereas the horizontal location of an object is essentially unconstrained by such factors. Hence we take $p(x|G)$ to be uniform and just learn $p(y|G)$ and $p(s|G)$.

In [34], Torralba used cluster weighted regression to represent $p(X, G)$:

$$p(X,G) = \sum_q P(q)P(G|q)P(X|G,q) = \sum_q \pi(q)\mathcal{N}(G;\mu_q^{(1)},\Sigma_q^{(1)})\mathcal{N}(X;W_qG+\mu_q^{(2)},\Sigma_q^{(2)})$$

where $\pi(q)$ are the mixing weights, W_q is the regression matrix, $\mu_q^{(i)}$ are mean (offset) vectors, and $\Sigma_q^{(i)}$ are (diagonal) covariance matrices for cluster q. These parameters can be estimated using EM. A disadvantage of this model is that it is a generative model of X and G. An alternative is a mixture of experts model [16], which is a conditional model of the form

$$p(X|G) = \sum_q P(q|G)P(X|G,q) = \sum_q \text{softmax}(q; w_q^T G)\mathcal{N}(X; W_qG, \Psi_q)$$

This can also be fit using EM, although now the M step is slightly more complicated, because fitting the softmax (multinomial logistic) function requires an iterative algorithm (IRLS). In this chapter, we use a slight variant of the mixture of experts model called mixture density networks (MDNs) [4]. MDNs use a multilayer perceptron to represent $P(q|G)$, $E[X|G, q]$ and $\text{Cov}[X|G, q]$, and can be trained using gradient descent. The main reason we chose MDNs is because they are implemented in the netlab software package.[8] Training (using multiple restarts) only takes a few minutes, and application to a test image is essentially instantaneous. When applied to the dataset used in [34], we get essentially the same results using MDN as those achieved with cluster weighted regression. (We have also performed some preliminary experiments using boosted stumps for regression [12]; results seem comparable to MDNs.)

To evaluate performance, the prior $P(X|G)$ can be evaluated on a grid of points for each scale: $G_i = P(x, y|G)P(s|G)$, where $i = (x, y, s)$. We then normalize to get $P(X = i|G) = G_i/(\sum_j G_j)$ so that $\sum_i P(X = i|G) = 1$. We can visualize this density by multiplying it elementwise by the image: see Figure 8

[8] `http://www.ncrg.aston.ac.uk/netlab`

for an example. We can evaluate the performance of the density estimate more quantitatively by comparing the predicted mean, $EX = \int Xp(X|G)dX$, with the empirical mean $\hat{X}$: see Figure 9. We see that we can predict the y value reasonably well, but the scale is harder to predict, especially for pedestrians.

Fig. 8. Example of location priming for screens, keyboards, cars and people using global features. In each group, the image on the left is the input image, and the image on the right is the input image multiplied by the probability, given the gist, of the object being present at a given location, i.e., $I. * P(x|G(I))$.

4 Object Presence Detection

Object presence detection means determining if one or more instances of an object class are present (at any location or scale) in an image. A very successful approach to this problem, pioneered by [6], is as follows: first, extract patches around the interest points in an image; second, convert them to codewords using vector quantization; third, count the number of occurrences of each possible codeword; finally, classify the resulting histogram. Unfortunately, if this method determines the object is present, it is not able to say what its location is, since all spatial information has been lost. We therefore use the more straightforward approach of first running a local object detector, and then using its output as input to the object presence classifier. More specifically, we define $L_m = \max_i L_i$ as the largest local detection score, and take $P(O = 1|L) = \sigma(w^T \ [1 \ \ L_m])$, so that $0 \leq P(O = 1|L) \leq 1$.

As shown in [34], it is possible to use the gist to predict the presence of objects, without needing to use a detector, since gists are correlated with object presence. Torralba used a mixture of (diagonal) Gaussians as a classifier:

$$P(O = 1|G) = \frac{P(G|O = 1)}{P(G|O = 1) + P(G|O = 0)} = \frac{\sum_q \pi_q^+ N(G; \mu_q^+, \Sigma_q^+)}{\sum_q \pi_q^+ N(G; \mu_q^+, \Sigma_q^+) + \sum_q \pi_q^- N(G; \mu_q^-, \Sigma_q^-)}$$

where each class-conditional density $P(G|O = \pm, q)$ is modeled as a Gaussian with diagonal covariance. We have found that using a single mixture component

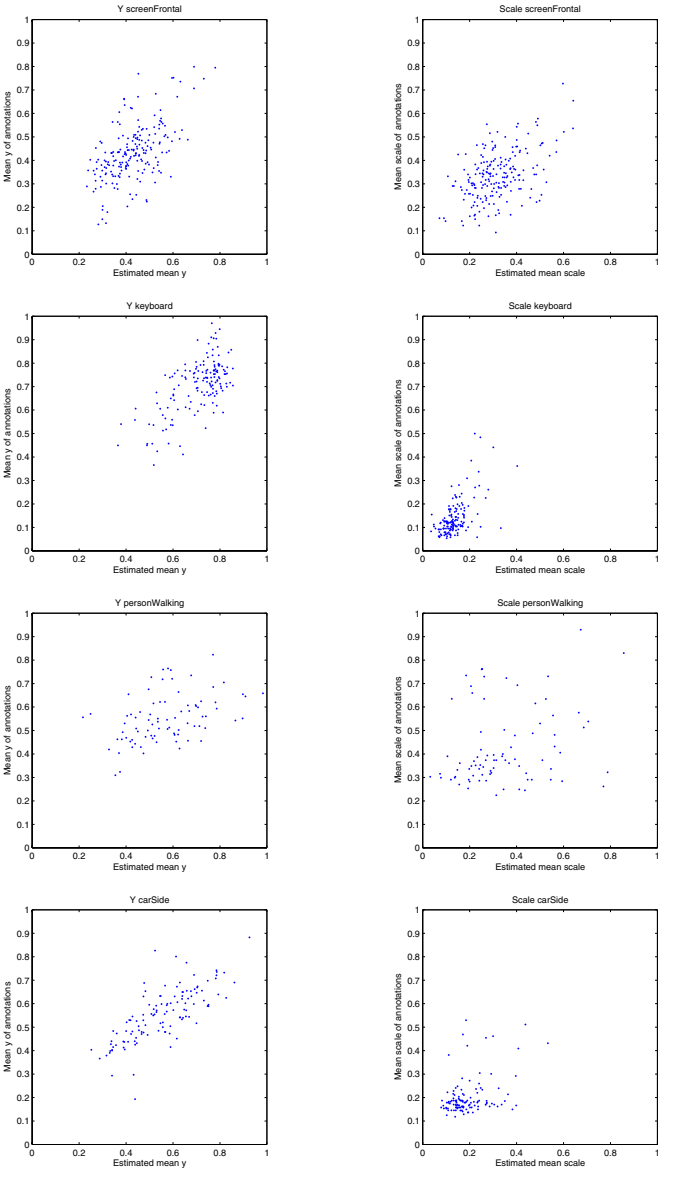

Fig. 9. Localization performance of screens, keyboards, people and cars using global features. Left column: vertical location of object; right column: scale of object. Vertical axis = truth, horizontal axis = prediction. We see that the gist provides a coarse localization of the object in vertical position and scale c.f., [34].

(i.e., a naive Bayes classifier) is sufficient, and has the advantage that EM is not necessary for learning. (We have also performed preliminary experiments using

boosted decisions stumps; results were slightly better, but in this chapter, we stick to the naive Bayes classifier for simplicity.)

To combine the local and global features, we treat $P(O = 1|G)$ and $P(O = 1|L)$ as scalar features and combine them with logistic regression:

$$P(O = 1|L, G) = \sigma(w^T \ [1 \ \ P(O = 1|L) \ \ P(O = 1|G)])$$

We estimate the weights w using maximum likelihood on the validation set, just in case either $P(O = 1|L)$ or $P(O = 1|G)$ is overconfident. Applying one classifier inside of another is a standard technique called "stacking". Other approaches to combining the individual $P(O = 1|L)$ and $P(O = 1|G)$ "experts" will be discussed in Section 5.2.

We compare the performance of the 3 methods (i.e., $P(O|L)$, $P(O|G)$ and $P(O|L, G)$) using ROC curves, as shown in Figure 10. We summarize these results using the area under the curve (AUC), as shown in Table 2. We see that the combined features always work better than either kind of feature alone. What is perhaps surprising is that the global features often perform as well as, and sometimes even better than, the local features. The reason for this is that in many of the images, the object of interest is quite small; hence it is hard to detect using a local detector, but the overall image context is enough to suggest the object presence (see Figure 11).

Table 2. AUC for object presence detection

	Screen	Kbd	Car	Ped
L	0.93	0.81	0.85	0.78
G	0.93	0.90	0.79	0.79
L,G	0.96	0.91	0.88	0.85

5 Object Localization

Object localization means finding the location and scale of an object in an image. Formally, we can define the problem as estimating $P(X = i|\cdot)$. We will compare 3 methods: $P(X|L)$, $P(X|G)$ and $P(X|L, G)$.

5.1 Performance Evaluation

We evaluate performance by comparing the bounding box B_p corresponding to the most probable location, $i^* = \arg\max P(X = i|\cdot)$, to the "true" bounding box B_t in manually annotated data. We follow the procedure adopted in the Pascal VOC (visual object class) competition[9], and compute the area of overlap

$$a = \frac{\text{area}(B_p \cap B_t)}{\text{area}(B_p \cup B_t)} \tag{1}$$

[9] `http://www.pascal-network.org/challenges/VOC/`

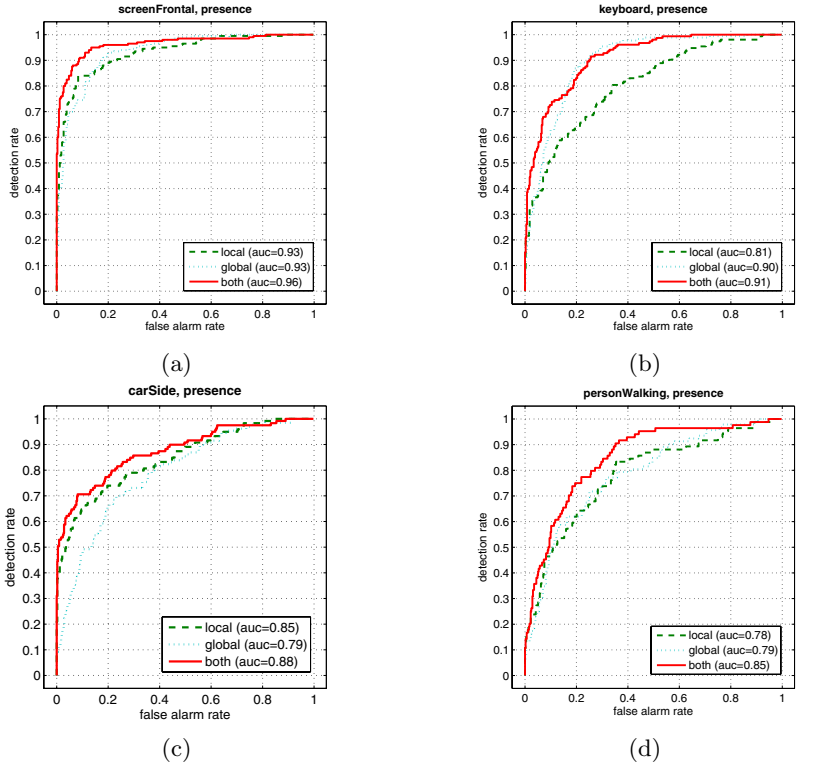

Fig. 10. Performance of object presence detection. (a) Screens, (b) keyboards, (c) cars, (d) pedestrains. Each curve is an ROC plot: $P(O|G)$: dotted blue, $P(O|L)$: dashed green, $P(O|L,G)$: solid red. We see that combining the global and local features improves detection performance.

If $a > 0.5$, then B_p is considered a true positive, otherwise it is considered a false positive.[10] (This is very similar to the criterion proposed in [1].)

We assign the best detection a score, $s(i^*) = P(O = 1|\cdot)$, which indicates the the probability that the class is present. By varying the threshold on this confidence, we can compute a precision-recall curve, where we define recall = TP/nP and precision = TP/ (TP+FP), where TP is the number of true positives (above threshold), FP is the number of false positives (above threshold), and nP is the number of positives (i.e., objects) in the data set.

We summarize performance of the precision-recall curves in a single number called the F1 score:

$$F = \frac{2 \cdot \text{Recall} \cdot \text{Precision}}{\text{Recall} + \text{Precision}}$$

[10] We used a slightly stricter criterion of $a > 0.6$ for the case of pedestrians, because the variation in their height was much less than for other classes, so it was too easy to meet the 0.5 criterion.

We use precision-recall rather than the more common ROC metric, since the latter is designed for binary classification tasks, not detection tasks. In particular, although recall is the same as the true positive rate, the false positive rate, defined as FP/nN where nN is the number of negatives, depends on the size of the state space of X (i.e., the number of patches examined), and hence is not a solution-independent performance metric. (See [1] for more discussion of this point.)

5.2 Combining Local and Global Features

We combine the estimates based on local and global features using a "product of experts" model [14]

$$P(X = i|L, G) = \frac{1}{Z} P(X = i|L)^{\gamma} P(X = i|G)$$

The exponent γ can be set by cross-validation, and is used to "balance" the relative confidence of the two detectors, given that they were trained independently (a similar technique was used in [15]). We use $\gamma = 0.5$.

Another way to interpret this equation is to realise that it is just a log-linear model:

$$P(X = i|L, G) \propto e^{\gamma \psi_L(X=i) + \psi_G(X=i)}$$

where the fixed potentials/features are $\psi_L(X) = \log P(X|L)$ and $\psi_G(X) = \log P(X|G)$. Since $Z = \sum_{i=1}^{N} P(X = i|L, G)$ is tractable to compute, we could find the optimal γ using gradient descent (rather than cross validation) on the validation set. A more ambitious approach would be to jointly learn the parameters inside the models $P(X = i|L)$ and $P(X = i|G)$, rather than fitting them independently and then simply learning the combination weight γ. We could optimize the contrastive divergence [14] instead of the likelihood, for speed. However, we leave this for future work.

Note that the product of experts model is a discriminative model, i.e. it defines $P(X|I)$ rather than $P(X, I)$. This gives us the freedom to compute arbitrary functions of the image, such as $G(I)$ and $L(I)$. Also, it does not make any claims of conditional independence: $P(X|L)$ and $P(X|G)$ may have features in common. This is in contrast to the generative model proposed in [34], where the image was partitioned into disjoint features, $I = I_G \cup I_L$, and the global features were used to define a "prior" $P(X|I_G)$ and the local features were used to define a "likelihood" $P(I_L|X)$:

$$\begin{aligned} P(X|I) &= \frac{P(I_L, I_G, X)}{P(I_L, I_G)} \\ &= \frac{P(I_L|X, I_G) P(X|I_G) P(I_G)}{P(I_L|I_G) P(I_G)} \\ &\propto P(I_L|X, I_G) P(X|I_G) \\ &\approx P(L(I)|X) P(X|G(I)) \end{aligned}$$

Fig. 11. Examples of localization of screens, keyboards, cars and pedestrians. Within each image pair, the top image shows the most likely location/scale of the object given local features ($\arg\max P(X|L)$), and the bottom image shows the most likely location/scale of the obhect given local and global features ($\arg\max P(X|L,G)$).

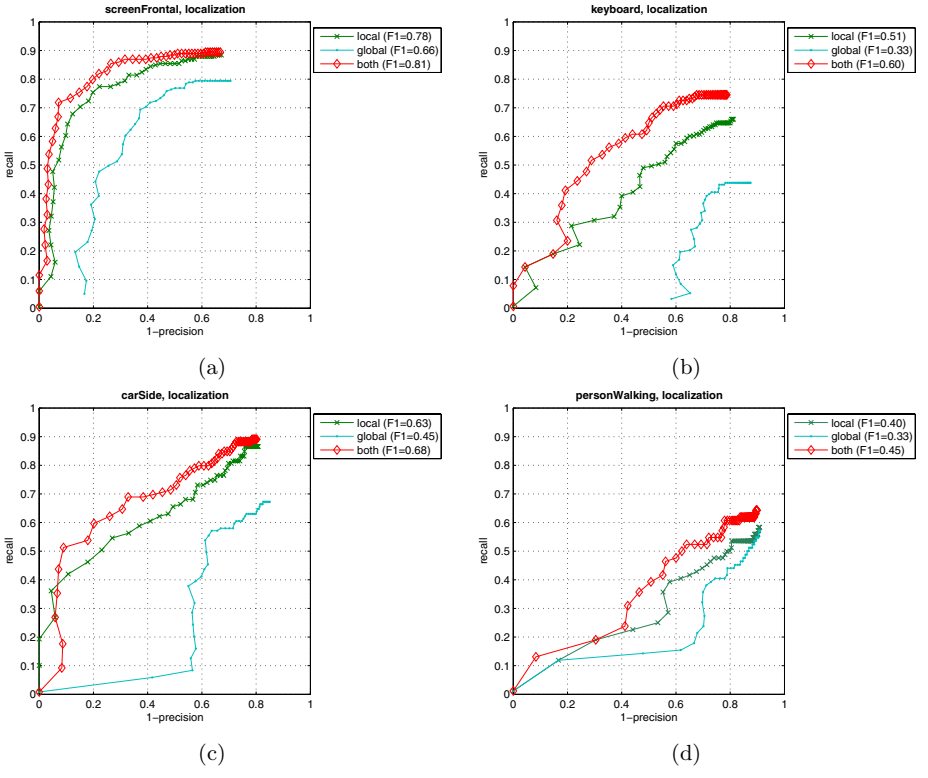

Fig. 12. Localization performance for (a) screens, (b) keyboards, (c) cars, (d) pedestrians. Each curve shows precision-recall curves for $P(X|G)$ (bottom blue line with dots), $P(X|L)$ (middle green line with crosses), and $P(X|L, G)$ (top red line with diamonds). We see significant performance improvement by using both global and local features.

The disadvantage of a product-of-experts model, compared to using Bayes' rule as above, is that the combination weights are fixed (since they are learned offline). However, we believe the advantages of a discriminative framework more than compensate.

The advantages of combining local and global features are illustrated qualitatively in Figure 11. In general, we see that using global features eliminates a lot of false positives caused by using local features alone. Even when the local detector has correctly detected the location of the object, sometimes the scale estimate is incorrect, which the global features can correct. We see a dramatic example of this in the right-most keyboard picture, and some less dramatic examples (too small to affect the precision-recall results) in the cases of screens and cars. The right-most pedestrian image is an interesting example of scale ambiguity. In this image, there are two pedestrians; hence both detections are considered correct. (Since we are only considering single instance detection in this chapter, neither method would be penalized for missing the second object.)

In Figure 12 we give a more quantitative assessement of the benefit using precision-recall curves.[11] (For the case of global features, we only measure the peformance of $P(y, s|G)$, since this method is not able to predict the x location of an object.) We summarize the results using F1 scores, as shown in Table 3; these indicate a significant improvement in performance when combining both local and global features.

Table 3. F1 scores for object localization

	Screen	Kbd	Car	Ped
G	0.66	0.33	0.45	0.33
L	0.78	0.51	0.63	0.40
L,G	0.81	0.60	0.68	0.45

6 Discussion and Future Work

We have shown how using global features can help to overcome the ambiguity often faced by local object detection methods. In addition, since global features are shared across all classes and locations, they provide a computationally cheap first step of a cascade: one only needs to invoke a more expensive local object detector for those classes that are believed to be present; furthermore, one only needs to apply such detectors in plausible locations/ scales.

A natural extension of this work is to model spatial correlations between objects. One approach would be to connect the X variables together, e.g., as a tree-structured graphical model. This would be like a pictorial structure model [8] for scenes. However, this raises several issues. First, the spatial correlations between objects in a scene are likely to be much weaker than between the parts of an object. Second, some object classes might be absent, so X^c will be undefined; we can set X^c to a special "absent" state in such cases (thus making the O^c nodes unnecessary), but the tree may still become effectively disconnected (since location information cannot propagate through the "absent" states). Third, some objects might occur more than once in an image, so multiple X^c nodes will be required for each class; this raises the problem of data association, i.e., which instance of class c should be spatially correlated with which instance of class c'.

An alternative approach to directly modeling correlations between objects is to recognize that many such correlations have a hidden common cause. This suggests the use of a latent variable model, where the objects are considered conditionally independent given the latent variable. In [21], we showed how we could introduce a latent scene category node to model correlations amongst the O variables (i.e., patterns of object co-occurrence). Extending this to model correlations amongst the X variables is an interesting open problem. One promising approach is to estimate the (approximate) underlying 3D geometry of the scene

[11] Note that the results for the car detector in Figure 12 are much worse than in Figure 6; this is because the MIT-CSAIL dataset is much harder than the UIUC dataset.

[13]. This may prove helpful, since e.g., the keyboard and screen appear close together in the image because they are both supported by a (potentially hidden) table surface. We leave this issue for future work.

References

1. S. Agarwal, A. Awan, and D. Roth. Learning to detect objects in images via a sparse, part-based representation. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 26(11):1475–1490, 2004.
2. S. Agarwal and D. Roth. Learning a sparse representation for object detection. In *ECCV*, 2002.
3. 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.
4. C. M. Bishop. Mixture density networks. Technical Report NCRG 4288, Neural Computing Research Group, Department of Computer Science, Aston University, 1994.
5. G. Bouchard and B. Triggs. A hierarchical part-based model for visual object categorization. In *CVPR*, 2005.
6. G. Csurka, C. Dance, C. Bray, L. Fan, and J. Willamowski. Visual categorization with bags of keypoints. In *ECCV workshop on statistical learning in computer vision*, 2004.
7. P. Carbonetto, N. de Freitas, and K. Barnard. A statistical model for general contextual object recognition. In *ECCV*, 2004.
8. P. Felzenszwalb and D. Huttenlocher. Pictorial structures for object recognition. *Intl. J. Computer Vision*, 61(1), 2005.
9. J. Friedman, T. Hastie, and R. Tibshirani. Additive logistic regression: a statistical view of boosting. *Annals of statistics*, 28(2):337–374, 2000.
10. M. Fink and P. Perona. Mutual boosting for contextual influence. In *Advances in Neural Info. Proc. Systems*, 2003.
11. R. Fergus, P. Perona, and A. Zisserman. A sparse object category model for efficient learning and exhaustive recognition. In *CVPR*, 2005.
12. J. Friedman. Greedy function approximation: a gradient boosting machine. *Annals of Statistics*, 29:1189–1232, 2001.
13. D. Hoiem, A.A. Efros, and M. Hebert. Geometric context from a single image. In *IEEE Conf. on Computer Vision and Pattern Recognition*, 2005.
14. G. Hinton. Training products of experts by minimizing contrastive divergence. *Neural Computation*, 14:1771–1800, 2002.
15. X. He, R. Zemel, and M. Carreira-Perpinan. Multiscale conditional random fields for image labelling. In *CVPR*, 2004.
16. M. I. Jordan and R. A. Jacobs. Hierarchical mixtures of experts and the EM algorithm. *Neural Computation*, 6:181–214, 1994.
17. 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.
18. D. G. Lowe. Distinctive image features from scale-invariant keypoints. *Intl. J. Computer Vision*, 60(2):91–110, 2004.
19. A. Mohan, C. Papageorgiou, and T. Poggio. Example-based object detection in images by components. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 23(4):349–361, 2001.

20. K. Mikolajczyk, C. Schmid, and A. Zisserman. Human detection based on a probabilistic assembly of robust part detectors. In *Proceedings of the 8th European Conference on Computer Vision, Prague, Czech Republic*, May 2004.
21. K. Murphy, A. Torralba, and W. Freeman. Using the forest to see the trees: a graphical model relating features, objects and scenes. In *Advances in Neural Info. Proc. Systems*, 2003.
22. D. Navon. Forest before the trees: the precedence of global features in visual perception. *Cognitive Psychology*, 9:353–383, 1977.
23. A. Oliva and A. Torralba. Modeling the shape of the scene: a holistic representation of the spatial envelope. *Intl. J. Computer Vision*, 42(3):145–175, 2001.
24. C. Papageorgiou and T. Poggio. A trainable system for object detection. *Intl. J. Computer Vision*, 38(1):15–33, 2000.
25. 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.
26. H. A. Rowley, S. Baluja, and T. Kanade. Human face detection in visual scenes. In *Advances in Neural Info. Proc. Systems*, volume 8, 1995.
27. H. Schneiderman and T. Kanade. A statistical model for 3D object detection applied to faces and cars. In *CVPR*, 2000.
28. P. Schyns and A. Oliva. From blobs to boundary edges: Evidence for time and spatial scale dependent scene recognition. *Psychological Science*, 5:195–200, 1994.
29. T. Serre, L. Wolf, and T. Poggio. A new biologically motivated framework for robust object recognition. In *CVPR*, 2005.
30. A. Singhal, J. Luo, and W. Zhu. Probabilistic spatial context models for scene content understanding. In *CVPR*, 2003.
31. A. Torralba, K. Murphy, and W. Freeman. Contextual models for object detection using boosted random fields. In *Advances in Neural Info. Proc. Systems*, 2004.
32. A. Torralba, K. Murphy, W. Freeman, and M. Rubin. Context-based vision system for place and object recognition. In *Intl. Conf. Computer Vision*, 2003.
33. A. Torralba and A. Oliva. Depth estimation from image structure. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 24(9):1225, 2002.
34. A. Torralba. Contextual priming for object detection. *Intl. J. Computer Vision*, 53(2):153–167, 2003.
35. P. Viola and M. Jones. Robust real-time object detection. *Intl. J. Computer Vision*, 57(2):137–154, 2004.
36. P. Viola, M. Jones, and D. Snow. Detecting pedestrians using patterns of motion and appearance. In *IEEE Conf. on Computer Vision and Pattern Recognition*, 2003.
37. M. Vidal-Naquet and S. Ullman. Object recognition with informative features and linear classification. In *IEEE Conf. on Computer Vision and Pattern Recognition*, 2003.