

---

# 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

We consider the problem of detecting several types of objects (cars, desks, bookshelves, pedestrians, etc.) simultaneously. We treat the objects as if they were parts of a scene, and perform joint scene and object recognition. We show that this results in better performance on both tasks. We also introduce a new set of features that can be used for recognizing both objects and scenes.

## 1 Introduction

A goal of computer vision researchers is to develop systems that can recognize many different objects under general viewing conditions in unconstrained settings. Progress has been made for restricted cases of the general problem: restriction to one object and one pose leads to reliable algorithms for detecting, for example, frontal-view faces (e.g. [14, 15, 17]). Viewing isolated objects on uniform backgrounds allows systems that can identify many objects (e.g. [6, 5]). But the general problem is difficult and unsolved.

A simple example illustrates one reason why the object detection problem can be so difficult: finding a table in an office. A table is typically covered with other objects; indeed almost none of the table itself may be visible, and the parts that may be visible, such as its edge, are fairly generic features that may occur in many images. The table can only be identified using the context of the office and other objects as guide.

We compare two different approaches to this kind of problem. One approach is to model the dependencies between objects explicitly, and hope that objects that are easy to detect and recognize can help simplify the problem for the harder ones. Unfortunately, if all detectors are poor, we show that such an approach is unable to “lift itself by its own bootstraps”. The second approach is to use a holistic representation of the overall image [7] to provide a top-down, global context. This holistic representation, or “gist” of the image, can be used to estimate the category of the visual scene, e.g. a street scene, a kitchen scene, etc. We show that using global scene-level knowledge improves detection rates for a variety of everyday objects.

A further contribution of this work is a novel set of features which provides a common framework for both our global and local image analysis. This includes as special cases the features used in many previous successful object recognition systems. As in [15], we use boosting to select the useful features.

## 2 Features for objects and scenes

For object detection there are at least three families of approaches: region based (a region of the image is segmented from the background and is described by a set of features that provide texture information and shape information [1]), part based (an object is defined as a specific arrangement of small parts [2]) and patch based (a patch assumes that a rectangular image region is enough to recognize the object).

Here we use a patch-based approach. For objects with well-defined shapes (screens, 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 (similar to segmentation based approaches). To classify the scene, we treat the whole image as a single patch.

We would like to construct a set of features that is useful for cases in which the patch contains the whole object or just part of it. We therefore create a large set of features, and then use a (supervised) learning algorithm to choose the best subset for each class of objects. We discuss the learning algorithm in Section 3; here we focus on the features.

We can compute a single feature  $k$  for image patch  $i$  as follows. First we convolve the whole (monochrome) image  $I(x)$  with a filter  $g_k$ , next we compute the magnitude of the filter response raised to some power  $\gamma_k$ , then we extract the part corresponding to the patch, and finally we average the result using the weights given by a spatial template  $w_k$ :

$$f_i(k) = \sum_x w_k(x) (|I(x) * g_k(x)|^{\gamma_k})_i \quad (1)$$

The set of 12 filters is shown in Figure 1(a). This set includes oriented edges, a Laplacian filter, corner detectors and long edge detectors. We consider two exponents: by setting  $\gamma_k = 2$  we can compute the variance, and by setting  $\gamma_k = 4$ , we can compute the kurtosis (using  $f_i(\gamma = 4)/f_i(\gamma = 2)^2$ ). These two statistics are sufficient to describe the textural properties of each patch region, since it is well known that the histogram of responses to Gabor-like wavelets (as used in e.g., [10]) produced by natural images can be encoded by these two measures.

We use 30 different spatial templates, which are shown in Figure 1(b). The use of a spatial template allows us to encode, in a crude way, the spatial arrangement of features or “parts”; similar approaches are used in object recognition [2, 8] and scene recognition [7, 12, 16]. Rectangular features applied to the raw image have also been successful in face detection [15]; we can simulate this by choosing a filter that is a delta function.

The total set of features has size  $12 \times 30 \times 2 = 760$  (the factor of 2 arises because we consider  $\gamma_k = 2$  or 4). This set of 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  $[1 \ 1 \ 1 \ 1 \ 1]$ ) or as linear combinations of the other filter outputs (e.g., the first six filters are steerable). For the spatial templates, we use rectangular shapes as they can be easily computed using the integral image [15] of the output of each wavelet filter.

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

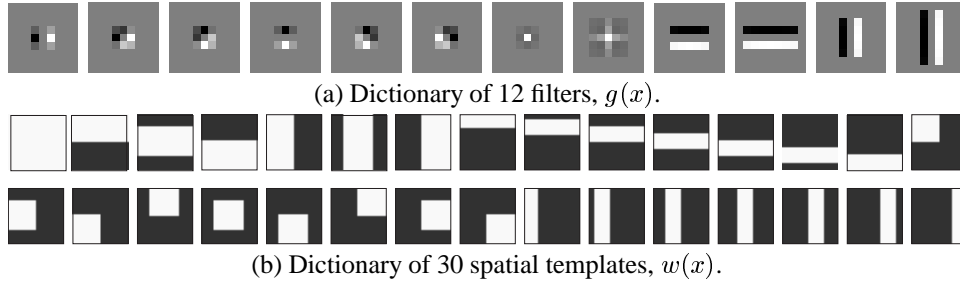


Figure 1: Dictionary of filters and spatial templates.

### 3 Isolated object recognition

In this section, we consider the problem of classifying an isolated image patch into one of about 25 different kinds of objects, including: cars, pedestrians, steps, desks, computer screens, bookshelves, etc. To train and to evaluate the system we use images acquired using a head-mounted camera. Images were captured while walking through the environment (including 63 different locations, indoors and outdoors). The system captures 4 frames/second at a resolution of 120x160 pixels. The resulting sequences contain many low quality images, due to motion blur, saturation or low-contrast (when suddenly changing lighting conditions), non-informative views (e.g., a close-up view of a door or wall), unusual camera angles, etc. For training, we labeled a subset of the images by indicating the name of the location, the category of the scene and the bounding box around a subset of the objects that are in each image. The training database contains about 1000 images.

For each object, we extract a set of patches with an aspect ratio that is average for that class, ranging from  $38 \times 40$  for screens to  $16 \times 60$  for pedestrians. The training patches are normalized in scale by scaling the image before applying the filters and spatial templates. From each training patch we extract the set of 760 features described in the previous section. We train each classifier with a set of 100-200 positive examples, and a large set of negative examples ( $> 10000$ ) chosen randomly from the regions of the training set known not to contain the object of interest.

The detectors are based on a classifier trained using boosting. The boosting procedure learns a weighted combination of base classifiers, or “weak learners”:  $h = \sum_t \alpha_t h_t(\vec{f})$ , where  $\vec{f}$  is the feature vector of the patch,  $h_t$  is the base classifier used at round  $t$ , and  $\alpha_t$  is its corresponding weight. The final classification result is gotten by thresholding this weighted sum of outputs; varying this threshold changes the hit rate/ false alarm rate.

For the weak classifiers we use decision stumps [12], i.e., each  $h_t$  picks a single component of the feature vector  $\vec{f}$  and chooses the corresponding optimal threshold. In this way, boosting can be used for feature selection. For most of the objects we used between 100 and 500 rounds of boosting (allowing to use the same features several times). The number of times that a features is used, and the weights given to it, provide an indication of the features that are most discriminative of each object: see Figure 2).

To apply the detector to an image, we filter the image with all the filters of the dictionary, we then compute the output for  $\gamma = 2$  and  $\gamma = 4$ , and then we evaluate the integral images of all the outputs. Finally we extract the features at all locations by extracting patches of the appropriate size. To detect the object at multiple scales, we scale down the image in steps of 20% of reduction until we reach the size of the patch. This process is very efficient since all the objects share most of the computations. Figure 3 summarizes the performances of the detectors for a set of objects on the test set. The results vary in quality since some

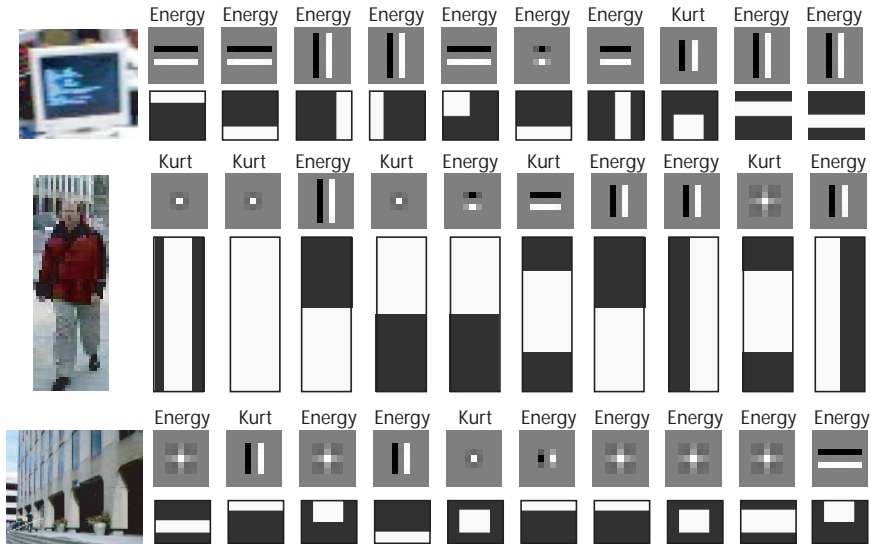


Figure 2: Some of the features chosen after 100 rounds of boosting for recognizing screens, pedestrians and buildings. Features are sorted in order of decreasing weight, which is a rough indication of importance. “Energy” means  $\gamma_k = 2$  and “Kurt” (kurtosis) means  $\gamma_k = 4$ .

objects are harder to recognize than others (e.g., their is more intra-class variability), and because some objects have less training data.

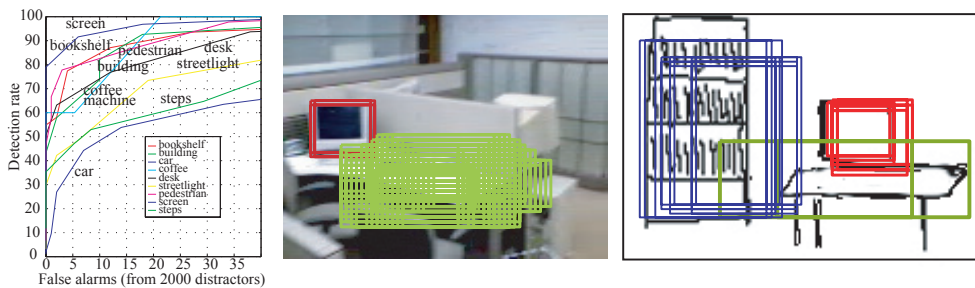


Figure 3: a) ROC curves for 9 objects; we plot hit rate vs number of false alarms. b) Example of the detector output on one of the test set images. c) Example of the detector output on a line drawing of a typical office scene. The system correctly detects the screen, the desk and the bookshelf.

## 4 Joint object recognition

So far we have considered the problem of recognizing individual objects in isolation. However, this is suboptimal, because objects often co-occur in statistically reliable ways. The optimal approach is to compute the joint probability of the presence of all the objects in the image given all the detector outputs:  $P(o_1, \dots, o_n | \vec{h})$ . Unfortunately, this quantity is intractable to compute (or even represent), because the number of detector outputs can be very large, and can vary from image to image. In this section, we will outline a variety of different approaches to this problem, and compare them empirically in Section 4.5.

#### 4.1 Isolated object detection

The simplest approach is to assume all object detections are independent:  $P(o_1, \dots, o_n | h_1, \dots, h_n) = \prod_{i=1}^n p(o_i | h_i)$ . This is essentially the approach adopted in the previous section, where we convert the output of the boosted classifier to a probability. One way of doing this is discussed in [3]. Here we adopt the simpler strategy of fitting one Gaussian to the weighted output of the classifier when the object is present, another when it is absent, and then using Bayes' rule, i.e.,

$$p(O_i | h_{i,o}) = \frac{p(h_{i,o} | O_i) p(O_i)}{p(h_{i,o} | O_i) p(O_i) + p(h_{i,o} | \bar{O}_i) p(\bar{O}_i)} \quad (2)$$

where  $p(h_{i,o} | O_i) = N(h_{i,o}; \mu_o, \sigma_o)$  (and similarly for  $p(h_{i,o} | \bar{O}_i)$ ).

#### 4.2 Modeling joint dependencies between objects directly

Another approach is to assume that detections of the same type (e.g., all outputs of the car detector) are conditionally independent given a latent variable, representing the presence or absence of that class (category) in the image. We can then model the joint constraints between objects at the category level, c.f., [4]. This lets us encode the fact that e.g., cars and desks do not usually co-occur in the same image, although it does not let us model constraints (e.g., spatial) between the individual detections themselves.

More formally, the model is as follows. Conditioned on the presence/absence of the class  $C_j$ , we assume the individual detections of type  $j$ ,  $O_{i,j}$ , are conditionally independent, i.e.,

$$P(C_1, \dots, C_N, O_1, \dots, O_n | \vec{h}) \propto P(C_1, \dots, C_N) \prod_{j=1}^N \prod_{i=1}^{n_j} P(O_{i,j} | C_j) P(h_{i,j} | O_{i,j})$$

where  $N$  is the (fixed) number of object types (5-10 in this paper), and  $n_j$  is the (fixed) number of patches of type  $j$ . The number of patches of each type is equal to the size of the image divided by the size of the training patch, multiplied by the number of scales; roughly,  $n_j \sim 2000$  for each class. To represent  $P(C_1, \dots, C_N)$ , it is natural to use a Markov random field with pairwise potentials:

$$P(C_1, \dots, C_N) = \frac{1}{Z} \prod_{i,j} \phi_{i,j}(C_i, C_j)$$

where  $Z$  is the normalizing constant (partition function) and  $\phi_{i,j}(C_i, C_j) \stackrel{\text{def}}{=} P(C_j | C_i)$  represents the probability of  $C_j$  being present/absent in an image given that  $C_i$  is present/absent. (We can estimate these parameters from labeled data by counting.) The resulting model, which we will call the "joint object model", is shown as a graphical model in Figure 4(b). We assume the graph between the  $C_j$  nodes is fully connected, but one could imagine learning a sparse structure.

The advantage of the joint model is that the detectors can "talk to each other" to help overcome ambiguities. However, this requires that at least some of the detectors be reasonably good. To see this, note that to correctly infer the status of the class-level variables (e.g., to answer the question "is there a car anywhere in this image?"), we must have fewer than one false alarm per image, otherwise we cannot distinguish the true positives from the false positives. This task is therefore more demanding in some ways than detecting specific object instances.

#### 4.3 Objects are conditionally independent given scenes

An alternative approach to multiple object recognition exploits the intuition that although there is correlation amongst the presence of objects, conditional on the global context, they

may be treated as independent: see Figure 4(c).

One kind of global context is the current scene [7], which encodes what kinds of objects typically co-occur (as well as their typical size and location, etc.). For example, in an outdoor urban scene, we expect to see cars and buildings, and in an indoor office scene, we expect to see desks and chairs. The scene can be used as a top-down prior to predict which kinds of objects are likely to occur, and in which (image) locations. This prior can be used in two alternative ways: either to reduce computation by selectively gathering/ processing data (e.g., we don't bother to run our car detector if we know we are indoors, and we don't bother to run our desk detector if we know we are looking at the ceiling), or to disambiguate the results of the detectors applied to the whole image. (Ambiguities are often inherent to the problem, and cannot be overcome by local, bottom-up approaches alone.)

In this paper, we consider seven kinds of indoor scene categories, and one outdoors: kitchen, office, lab, conference room, open area, corridor, elevator and street. We have trained several classifiers (in particular, mixtures of Gaussians and boosted decision stumps) to predict the current scene category given the features of the image (the "visual gist"). We have tried various sets of features, including the ones described in Section 2, as well as PCA applied to the output of a (steerable pyramid or Gabor) filter bank; all approaches work equally well. The main point is that, in this framework, scene recognition *precedes* object recognition.

It is often hard to tell what the current scene is from a single frame (e.g., if you have a close up view of something which you do not recognize). A simple approach is to average the result of the classifier over the last few frames. A better approach is to use an HMM, where the states represent scene categories and the observation model is the mixture of Gaussians classifier referred to above [13]. This model can encode general knowledge of the environment, such as the fact that the most likely route from a street to an office is via a lobby or corridor. (Of course, it is also possible to use the same approach to learn to recognize specific places, as in a topological map [13]; this acts as an even stronger prior, but such knowledge is not transferable to new environments.)

#### 4.4 Inference in the full model

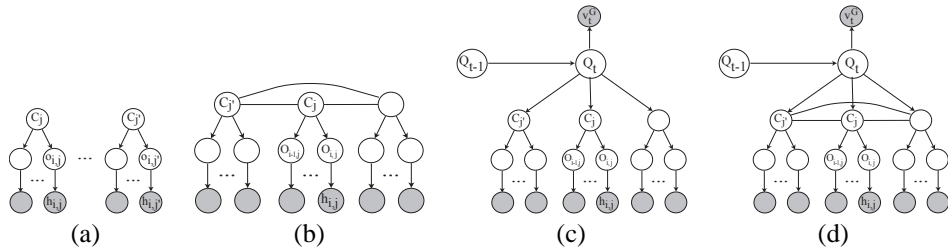


Figure 4: Graphical models for object detection. (a) Independent (forest of trees). (b) Joint modeling at the class level. (c) Conditionally independent given the current scene,  $Q_t$ . (d) The full model.  $v_t^G$  are the current global scene features,  $Q_t$  is the current scene,  $C_j$  represents the presence/absence of class  $j$  in the image,  $O_{i,j}$  represents whether patch  $i$  contains an object of type  $j$ , and  $h_{i,j}$  is the output of the  $j$ 'th detector on the  $i$ 'th patch.

The final model, which combines the HMM for scene recognition, joint object modeling at the class level, and individual detections, is shown in Figure 4(d). We now briefly describe how to perform inference in this model. In the case in which we do not model the dependencies between object classes (no horizontal arcs between the  $C_j$  nodes), the graph is a tree, and hence we can perform exact inference using Pearl's algorithm (belief propagation) [9]. When we have dependencies between the object classes, we can either collapse all the

$C_j$  nodes into a large “mega” node (which is necessary since the graph is fully connected), or we can run use loopy belief propagation to do approximate inference.

We send messages in the following order. First, the global scene context,  $Q_t$ , absorbs evidence from the previous context,  $Q_{t-1}$ , and the current visual gist,  $V_t^G$  (the HMM predict-update step). Then the scene node  $Q_t$  sends a (top-down) message to the class nodes,  $C_j$ , so they can perform object priming. Next, we run all detectors on all patches (we could use the prior to make this more efficient); i.e., the object detection nodes,  $O_{i,j}$ , send messages (reflecting their likelihoods) up to their parents,  $C_j$ . We are now able to compute the class posteriors,  $P(C_j|v_{1:t}^G, \vec{h}_j)$ . In the joint model, the  $C_j$  nodes now perform message passing to compute their posterior given all the detectors,  $P(C_j|\vec{h})$ . Next, the class nodes  $C_j$  send messages back up to the scene,  $Q_t$  (e.g., if you think you saw a desk, you are probably in an office), ready for the next time step. Finally, the class nodes  $C_j$  send messages back to the detection nodes  $O_{i,j}$ .

The final step, where the class nodes  $C_j$  send messages back to the detection nodes  $O_{i,j}$ , is problematic. Although it can be implemented exactly (modulo any inaccuracies due to optionally ignores loops in the joint object model), it does not lead to good results. The reason is that the top down message from  $C_j$  to  $O_{i,j}$  contains information from the context and the other objects (which is fine), but also from all the other detectors,  $O_{i',j}$ ; the latter results in over confidence, since the model incorrectly assumes that each patch is independent given the class variable. The solution we adopt is to slightly modify Pearl’s algorithm in this final step, and to use  $P(C_j|v_{1:t}^G, \vec{h} \setminus \vec{h}_j)$  as the top-down  $\pi$  message, ignoring the other  $\lambda$  messages that  $O_{i',j}$  sent up to  $C_j$ . In other words, the final, context-weighted detections are computed as follows:

$$P(O_{i,j}|\vec{h}, v_{1:t}^G) \propto \sum_c p(h_{i,j}|O_{i,j})P(O_{i,j}|C_{i,j} = c)P(C_{i,j} = c|\vec{h} \setminus \vec{h}_j, v_{1:t}^G).$$

## 4.5 Results

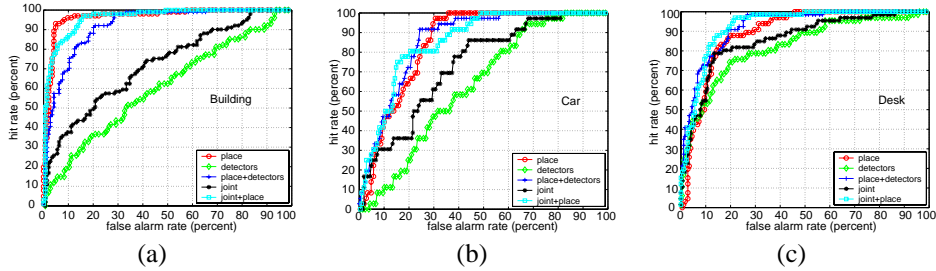


Figure 5: ROC curves for detecting the presence of object classes in the image: (a) building, (b) car, and (c) desks. Lowest curve is the detector alone, second best is the joint model; the remaining (indistinguishable) curves are results using the conditionally independent model without the detectors (to evaluate the strength of the prior), the conditionally independent model with the detectors, and the full model.

In this section, we compare the models described above on two different tasks. In the first task, the goal is to infer the class-level variables. (This can be useful for image retrieval, where one wants to find all images that contain a car or a person.) We measured performance on this task using a classical ROC detection paradigm by changing the threshold of  $P(C_j|\cdot)$ , where  $\cdot$  represents the different kinds of information we can condition on. In Figure 5, we see that just using the detector,  $P(C_j|\vec{h}_j)$ , results in the lowest performance; using the other objects,  $P(C_j|\vec{h})$ , helps a bit; but adding the scene,  $P(C_j|\vec{h}, v_{1:t}^G)$ , helps

the most. Figure 6c shows the results of retrieving images which are believed to contain a coffee machine, as sorted by  $P(C_j|\vec{h}_j)$  (top) and  $P(C_j|\vec{h}_j, v_{1:t}^G)$  (bottom). The second task is the traditional problem of detecting individual objects in an image. Figure 6(a-b) shows two ROC curves for screens and coffee machines, using just the detector,  $P(O_{I,j}|\vec{h}_j)$ , and using the detector plus scene context,  $P(O_{I,j}|\vec{h}_j, v_{1:t}^G)$ .

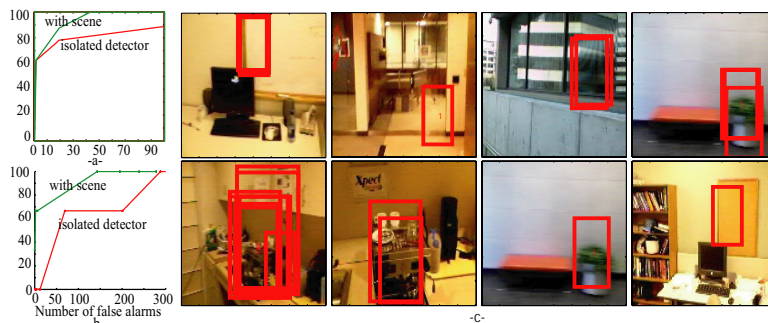


Figure 6: ROC curves for (a) screens and (b) coffee machine using Just the detector (lower curves) or with the scene. (c). The 4 images (from a test set of size 500) most likely to contain a coffee Machine, as judged by the detector alone (top) or the detector plus scene context (bottom). Sample detector outputs (above a fixed probability threshold) are also shown in red.

## 5 Conclusions and future work

We have shown how to construct a dictionary of features that is flexible enough to recognize a wide variety of objects and scenes. We have also shown how modeling objects and scenes together can result in improved recognition performance at both the class and instance level. In the future, we would like to better understand the overconfidence problem which arises when passing messages from the class level down to the individual detectors. Also, we would like to move beyond using rectangular patches, perhaps by using segmentation to derive more general regions upon which to apply a classifier.

## References

- [1] K. Barnard and D. A. Forsyth Learning the semantics of words and pictures ICCV 2:408–415, 2001
- [2] Burl, M.C., Weber, M., and Perona, P. 1998. A Probabilistic Approach to Object. Recognition using Local Photometry and Global Geometry. *Proc. 5th European Conf. Comp. Vision*, pp. 628-641.
- [3] J. Friedman and T. Hastie and R. Tibshirani Additive logistic regression: a statistical view of boosting Tech report, Dept. of Statistics, Stanford University, 1998
- [4] Haralick, R. M. 1983. Decision making in context. *IEEE Trans. Pattern Analysis and Machine Intelligence*, Vol. 5: 417–428.
- [5] Bartlett W. Mel SEEMORE: Combining Color, Shape and Texture Histogramming in a Neurally-Inspired Approach to Visual Object Recognition *Neural Computation* 9(4):777-804, 1997
- [6] H. Murase and S. Nayar Visual learning and recognition of 3-d objects from appearance IJCV 14:5–24, 1995
- [7] Oliva, A., and Torralba, A. (2001). Modeling the Shape of the Scene: A holistic representation of the spatial envelope. *International Journal of Computer Vision*, 42(3):145-175.
- [8] Papageorgiou, C., Oren, M., and Poggio, T. (1998). A general framework for object detection. *Int. Conference on Computer Vision*, pp. 555-562.
- [9] J. Pearl (1988). "Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference Morgan Kaufmann
- [10] Schiele, B., and Crowley, J.L. (2000). Recognition without Correspondence using Multidimensional Receptive Field Histograms. *Int. Journal of Computer Vision*, 36(1):31–50.
- [11] Strat, T. M., and Fischler, M. A. 1991. 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.
- [12] Tieu, K. and Viola, P. (2000). Boosting image retrieval. *Proc. of IEEE Conference on Computer Vision and Pattern Recognition*, pp. 1:228-235.
- [13] A. Torralba and K. Murphy and W. Freeman and M. Rubin Context-based vision system for place and object recognition AI Lab technical report, 2003
- [14] M. Turk and A. Pentland "Eigenfaces for recognition J. of Cognitive Neuroscience, 3(1):71–86, 1991
- [15] Viola, P. and Jones, M. (2001). Rapid object detection using a boosted cascade of simple features. *Proceedings of 2001 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, IEEE Computer Society Press, Jauai, Hawaii.
- [16] Vailaya, A., Jain, A., and Zhang, H. J. 1998. On image classification: city images vs. landscapes. *Pattern Recognition*, 31:1921–1935.
- [17] Z. Zhang and M. Li and S. Li and H. Zhang and T. Huang Multi-view face detection with float boost Workshop on applications of computer vision, 2002