Figure-ground segmentation using a hierarchical conditional random field

Jordan Reynolds
University of British Columbia
jordan.reynolds@gmail.com

Kevin Murphy
University of British Columbia
murphyk@cs.ubc.ca

Abstract

We propose an approach to the problem of detecting and segmenting generic object classes that combines three "off the shelf" components in a novel way. The components are a generic image segmenter that returns a set of "super pixels" at different scales; a generic classifier that can determine if an image region (such as one or more super pixels) contains (part of) the foreground object or not; and a generic belief propagation (BP) procedure for tree-structured graphical models. Our system combines the regions together into a hierarchical, tree-structured conditional random field, applies the classifier to each node (region), and fuses all the information together using belief propagation. Since our classifiers only rely on color and texture, they can handle deformable (non-rigid) objects such as animals, even under severe occlusion and rotation. We demonstrate good results for detecting and segmenting cows, cats and cars on the very challenging Pascal VOC dataset.

1. Introduction

Recognizing, localizing and segmenting generic object classes is a challenging problem in computer vision, with many important applications. We propose a system for solving this which uses three "off the shelf" components: a generic image segmenter that returns a set of "super pixels" at different scales; a generic classifier that can determine if an image region (such as one or more super pixels) contains (part of) the foreground object or not; and a generic belief propagation (BP) procedure for tree-structured graphical models. Our system combines the regions together into a multi-scale tree, applies the classifier to each node (region), fuses all the information together using BP, and classifies the pixels using the beliefs at the leaves of the tree; this can be used to segment out the objects (if present).

By segmenting the image at multiple scales, we increase the chance that we will find some regions that can be confidently classified. (Regions that are too small will be ambiguous, because they have insufficient image data, but regions which are too big may also be ambiguous, because they contain a mixture of foreground and background.) By using BP on a tree, we can ensure that the unambigious regions can send information to their neighbors to help disambiguate them. Note that the tree structure, and strength of the tree edges, are determined dynamically from the segmentations, as we explain below. Also, the local evidence at each node is computed using a discriminative classifier. Thus the whole model is a tree-structured conditional random field (CRF).

The basic idea is illustrated in Figure 1. We see that regions that are too small to be classified accurately can "inherit" the labels of their parents. This allows us to estimate the object boundaries fairly accurately, using small regions, while simultaneously benefiting from the lower ambiguity afforded by larger regions.

Since our classifiers only rely on color and texture, they can handle deformable (non-rigid) objects such as animals. In this paper, we focus on detecting and segmenting cows and cats. However, to demonstrate the generality of the technique, we also use it to detect and segment cars. We use data from the very challenging Pascal VOC dataset.[1] for our experiments.

2. Related work

There are many papers that discuss generic object detection. These can roughly be grouped into approaches that use sliding window classifiers (e.g., [18, 22]), and those that use parts-based models, applied to sparse features (e.g., [7]) or dense features (e.g., [15, 6]). We use boosted decision stumps as in [22], and graphical models, as in [7, 15, 6]. However, parts-based models use graphical models where the nodes represent locations of the parts, whereas we use graphical models where the nodes represent labels of image patches. This will allow us to detect multiple objects and to perform object segmentation.

[1] Specifically, we used the 2006 training and validation dataset from http://www.pascal-network.org/challenges/VOC/. Note that the test set was not available at the time that this work was done.

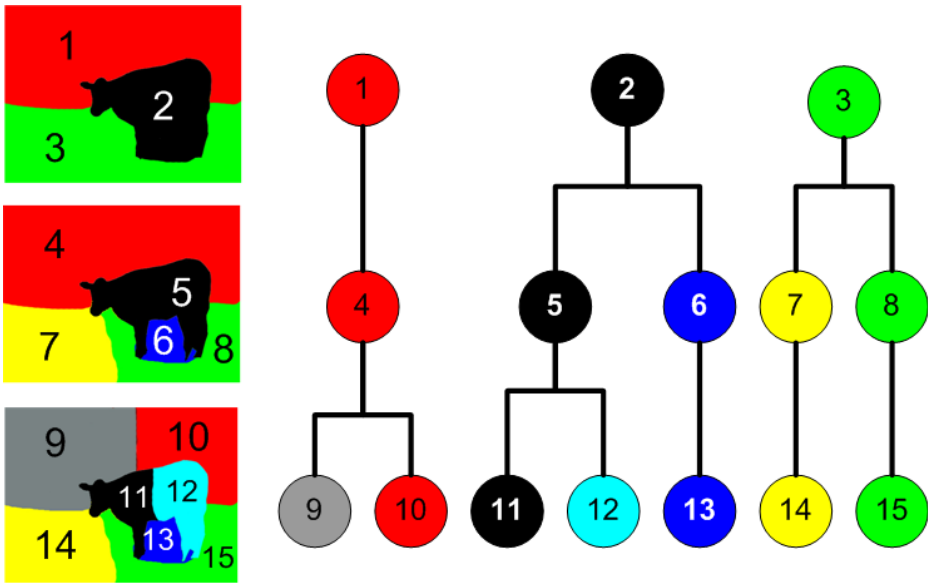


Figure 1. Simulated segmentations at multiple scales (left), together with a set of trees (right), with one node per region. Level 1 is at the bottom, level 3 at the top. Regions 11 and 12 may be too small to reliably classify in isolation, but when they inherit a message from their parent (region 5), they can be correctly classified as "cow". Similar reasoning applies to other regions.

The problem with sliding window classifiers is that they usually assume the object is rigid and mostly fills the bounding box. While one can learn to ignore areas with the bounding box (using fragments or masks [3]), this requires a lot of training data, and cannot easily handle out-of-plane rotation. The problem with parts-based approaches is that for some kinds of objects, such as cats, it is hard to detect distinct parts, and even when parts are detectable, they may have complex spatial relationships which are hard to model.

In [23], it was shown that a simple color and texture based classifier can do remarkably well at detecting 9 different kinds of objects, ranging from cows to bicycles. However, their system could only classify individual points, or regions chosen by a user. We also use color and texture features, but we use an image segmenter to automatically detect regions, which often coincide with object boundaries.

There has been a lot of work on image segmentation, but only a few papers look at it in the context of object detection. Borenstein et al [3, 2, 1] have proposed to combine bottom up segmentation with top down object detection. Their system is similar to ours in that they construct a hierarchical tree structured CRF, and use BP to infer figure-ground labels. However, they only have image evidence at the leaves, and their classifier is based on image patches, which is not robust to object rotation. (For instance, they only show results on side views of horses, whereas we use the much harder Pascal VOC dataset.)

There have been several papers on grid-structured CRFs for object detection and segmentation (e.g., [13, 4, 12, 14, 20]). Some of these models can model contextual relations between object classes. However, all these models only work at a single scale. Also, note that grid-structured CRFs are computationally intractable, whereas trees support linear time inference.

Schiele and students have produced several papers (e.g., [10]) describing methods for object detection and segmentation. These approaches work by building a dictionary of image patches and masks, together with offset vectors relative to the centroid of an object; at test time, the patches are detected and they vote for the centroid; the peaks in voting space are back-projected and masks super-imposed to yield a segmentation, which can be verified by a discriminative classifier. While very successful, this approach requires a lot of memory to store the large dictionary of patches. In contrast, our system, which is based on boosted decision stumps, has very few parameters (about 300, chosen by cross validation); this should enable it to scale to more classes, especially since such features can easily be shared across classes [21].

3 Our approach

We first give a brief outline of the steps that we follow when given a test image.

1. Perform histogram equalization to deal with images with low contrast.
2. Run a segmenter at multiple scales to return a set of segmented images.
3. Compute feature vectors for each region.
4. Apply a probabilistic binary classifier to each region.
5. Build a tree where nodes represent regions. Attach weights to the edges according to how similar the regions are. Attach weights to the nodes based on the output of the binary classifier.
6. Run belief propagation.
7. Threshold the beliefs at the leaves to segment out the object(s).

We explain the steps in more detail below.

3.1 Segmentation

We run a segmenter at multiple scales to return a set of segmented regions, v_i^ℓ, where ℓ indexes levels $\ell = 1 : L$, and i indexes regions. We use $L = 3$ levels, and typically have ~ 100 regions at the bottom (finest) level and ~ 10 regions at the top (coarsest) level. We use the segmentation method of Felzenszwalb and Huttenlocher [5], because it is very fast, and seems to work as well as more complex schemes, such as normalized cuts [19]. The method has 3 free parameters: σ controls the amount of image smoothing, k controls how similar regions have to be in order to be merged, and m controls the minimum region size. See Figure 2 for an illustration of the effect of changing these parameters.

3.2 Feature vectors

We compute feature vectors for each region as follows. We use color and texture histograms, since they are simple and work well for discriminating animal fur etc. from background. Also, they are easy to compute from an arbitrary-shaped region. Specifically, we use a 100-bin histogram for hue and saturation, and a 10-bin histogram for value (intensity). For texture, we convolve the image with a bank of Gabor filters at 6 orientations and 4 scales, and compute the average energy of the response in each region; we bin this response into 10 intervals. Thus we have $100 + 10 + 6 \times 4 \times 10 = 350$ features in total.

3.3 Classifier

We apply a probabilistic binary classifier to compute $p(y_i^\ell = 1|x_i^\ell, \theta)$, where x_i^ℓ is the feature vector, $y_i^\ell \in \{1, 0\}$ is the label (object or background), and θ are the parameters of the classifier. We use boosted decision stumps as our classifier [8, 9, 22], since they work well, and are fast to train and apply. That is, each weak learner is a function of the form

$$f(i) = \text{sgn}(w_1 x_k(i) - w_0)$$

where $x_k(i)$ is the k'th feature (histogram entry) evaluated at super-pixel i; the parameters of the weak learner are the index k, the weight w_1 and the offset w_0. We used $T = 300$ rounds of boosting for cows and cats, and 150 for cars this value was chosen using a validation set.

Boosted decision stumps implicitly perform feature selection, so the final classifier only uses at most T features (corresponding to $3T$ parameters), chosen from the original 350. However, since boosting is a greedy procedure that cannot go back and adjust weights of the weak learners, it often adds multiple copies of the same weak learner with different weights. These can all be combined at the end of training, to save time and space at runtime. The total number of unique features was 121 for cows, 131 for cats and 98 for cars.

The training procedure was as follows. We ran the segmenter on the training images, and then labeled every region as positive if it contained 75% or more overlap with the ground truth foreground; otherwise it was labeled negative. The original labeled data in the Pascal dataset consisted of bounding boxes, but we found that this resulted in a lot of regions being labeled positive even if they only contained a tiny fraction of the foreground. Hence we relabeled the data to get silhouettes (using Photoshop's flood-fill tool), as shown in Figure 3. This resulted in an improvement of the cow object base classifier (area under ROC (AUC) score) from approximately 73% to 89%.

The output of a boosted classifier is a number c which represents the "confidence" that the label is class 1. Following [17], we rescale these confidence values to the $[0, 1]$ interval using logistic regression: $p(y = 1|x) = \sigma(ac(x)+b)$, where $\sigma(u) = 1/(1+e^{-u})$ is the sigmoid function. The parameters a and b were fit by maximum likelihood on a separate validation set (to avoid overconfidence). The resulting values can be interpreted as probabilities (albeit poorly calibrated ones). This makes it easier to combine the detector results with other sources of information. To this end, let us define the local evidence at node i to be the vector

$$\phi_i = \left(p(y_i^\ell = 1|x_i^\ell, \theta),\ \ p(y_i^\ell = 0|x_i^\ell, \theta)\right).$$

The training algorithm (implemented in Matlab) takes about 3 hours to run on 100 images. However, most of this

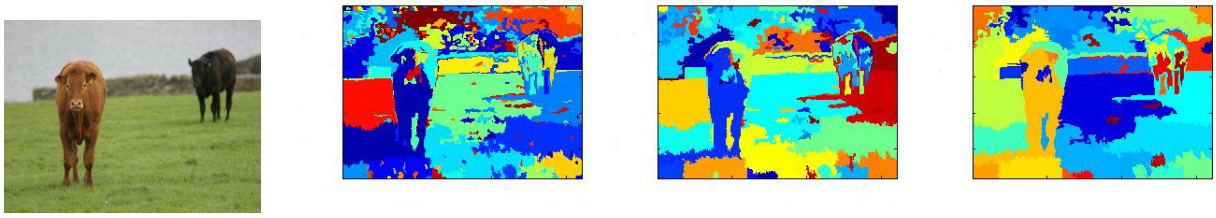
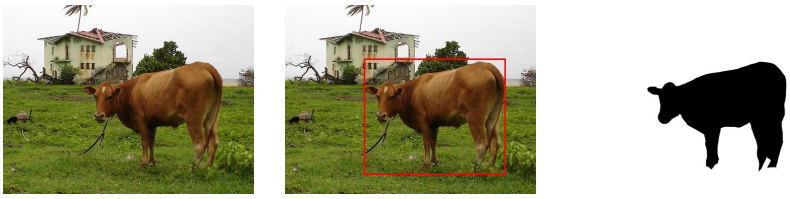


Figure 2. Hierarchy of segmentations produced by varying the parameters $(\sigma, k, m) = \{0.5, 500, 50\}, \{0.75, 500, 200\}, \{0.75, 500, 500\}$**.**



(a). (b). (c).

Figure 3. Example of training data (a). Input image, (b). Bounding box annotation, (c). Pixel level labeling.

time is spent computing the texture features. We are confident that with a faster implementation of Gabor filtering (e.g., in C++), we could reduce the training time by at least a factor of 10.

3.4 Tree-construction

We build a tree where node i at level ℓ connects to a single node j at level $\ell+1$, where j is the region with maximal pixel overlap with i:

$$j = \arg\max_j \frac{|V_i^\ell \cap V_j^{\ell+1}|}{|V_i^\ell|}$$

The result is a forest of trees, since the nodes at level L have no parents. We do not merge these top levels into a single tree, since the resulting super regions would be a mix of foreground and background and hence hard to classify.

The topology of the tree depends on the location of the regions, but not on the image data per se. This can result in a child region being connected to a parent which contains it, even though they are visually dissimilar. In such a case, we do not want to force the labels of the parent and child to be the same (which would be overconstraining them). So we weight the edges based on how similar the corresponding regions are. Specifically, let χ^2_{ij} be the χ^2 distance between the feature vectors (histograms) for regions i and j. Let $\lambda_{ij} = e^{-\chi^2_{ij}}$, so similar regions (with small χ^2) will have $\lambda_{ij} \approx 1$, and dissimilar regions (with large χ^2) will have $\lambda_{ij} \approx 0$. We define the following edge potentials (c.f., an Ising model):

$$\psi_{i,j} = \begin{bmatrix} e^{\lambda_{i,j}\cdot\gamma} & e^{-\lambda_{i,j}\cdot\gamma} \\ e^{-\lambda_{i,j}\cdot\gamma} & e^{\lambda_{i,j}\cdot\gamma} \end{bmatrix}$$

We used $\gamma = 100$, chosen by cross validation. Thus some edges in the tree are effectively "disconnected" if the corresponding regions are too dissimilar.

The overall model defines a CRF:

$$p(y|x,\theta) = \frac{1}{Z(x,\theta)} \prod_{<ij>} \psi_{ij}(y_j, y_j) \prod_i \phi_i(y_i|\theta)$$

where $\prod_{<ij>}$ is a product over adjacent nodes i and j (which must be at different levels). Since the graph is tree structured, we can efficiently compute Z using belief propagation. Hence we could learn the parameters in ϕ and ψ jointly. However, in this work we learn them separately, for simplicity.

3.5 Belief propagation

We use the sum-product algorithm (see e.g., [11]), sending messages from the leaves to the root and back. This takes time linear in the number of nodes, and is exact, since the graph is a tree. We then threshold the beliefs at the leaves to choose the pixels that are considered foreground and use this binary mask to segment out the object.

4 Results

We show some sample results of our system on some of the Pascal test images in Figure 4. The base classifier does remarkably well, given its simplicity. However, there are some regions about which the classifier is uncertain, and it is here that the tree structure can help. We see that BP tends to push some false positive regions below threshold, such as the top left background region in the car image, while simultaneously raising some false negative regions above threshold, such as the top right horse. However, the system is not fool-proof: since the base detector essentially failed to detect the bottom cat, adding BP does not help.

In Figure 5, we present some quantitative results. By varying the threshold on the beliefs at the leaves, we can produce a ROC curve, which measures the tradeoff between detection rate and the false alarm rate. We see that the base classifiers do quite well, and that adding BP helps in each case, albeit sometimes by a rather modest amount. The performance of the car detector is rather low, and BP cannot help it much. However, our technique was not really designed for rigid objects such as cars, for which standard sliding window classifiers work very well [18]. Note that our framework can be used to potentially improve the performance of any set of features or any classifier, and is not tied to the simple ones we used here.

5 Conclusions and future work

We have presented a simple technique for detecting and segmenting deformable objects such as cows and cats. The system can be easily assembled from standard parts and trained very quickly.

There are many ways to improve performance, including: improving the base features, e.g. by including shape features [1, 14]; improving the classifier, e.g., by using boosted decision trees instead of stumps [9]; learning all the parameters of the CRF jointly [13, 14]; modeling correlation of the labels within a level of the tree [14, 20]; adding contextual priors (e.g., to a global root node) about the objects that are likely to be present [16]; etc. We leave these extensions to future work.

Object	Base classifier			Post BP
	Color	Texture	Combined	
Cow	0.818	0.809	0.894	0.916
Cat	0.734	0.792	0.799	0.819
Car	N/A	N/A	.736	0.744

Table 1. Area under the ROC curve for 3 different object classes using the base classifier, with color, texture, and combined color and texture features, and using belief propagation (BP) with the combined features.

Acknowledgements

We would like to thank Pedro Felzenszwalb for putting the source code for his segmentation algorithm on the web `http://people.cs.uchicago.edu/~pff/segment/`, and David Lowe, Jim Little and Daniel Eaton for their comments on the paper.

References

[1] E. Borenstein and J. Malik. Shape guided object segmentation. In *CVPR*, 2006.

[2] E. Borenstein, E. Sharon, and S. Ullman. Combining top-down and bottom-up segmentation. In *CVPR*, 2004.

[3] E. Borenstein and S. Ullman. Class-specific, top-down segmentation. In *ECCV*, 2002.

[4] P. Carbonetto, N. de Freitas, and K. Barnard. A statistical model for general contextual object recognition. In *ECCV*, 2004.

[5] P. Felzenszwalb and D. Huttenlocher. Efficient graph-based image segmentation. *Intl. J. Computer Vision*, 59(2):167–181, 2004.

[6] P. Felzenszwalb and D. Huttenlocher. Pictorial structures for object recognition. *Intl. J. Computer Vision*, 61(1), 2005.

[7] R. Fergus, P. Perona, and A. Zisserman. Object class recognition by unsupervised scale-invariant learning. In *CVPR*, 2003.

[8] Y. Freund and R. R. Schapire. Experiments with a new boosting algorithm. In *Intl. Conf. on Machine Learning*, 1996.

[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. Fritz, B. Leibe, B. Caputo, and B. Schiele. Integrating representative and discriminant models for object category detection. In *IEEE Conf. on Computer Vision and Pattern Recognition*, volume 2, pages 1363–1370, October 2005.

[11] F. Kschischang, B. Frey, and H.-A. Loeliger. Factor graphs and the sum-product algorithm. *IEEE Trans Info. Theory*, February 2001.

[12] M. P. Kumar, P. H. S. Torr, and A. Zisserman. OBJ CUT. In *CVPR*, 2005.

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

[14] A. Levin and Y. Weiss. Learning to combine bottom-up and top-down segmentation. In *ECCV*, 2006.

[15] G. Mori, X. Ren, A. Efros, and J. Malik. Recovering human body configurations: Combining segmentation and recognition. In *CVPR*, 2004.

[16] K. Murphy, A. Torralba, and W. Freeman. Using the forest to see the trees: a graphical model relating features, objects and scenes. In *NIPS*, 2003.

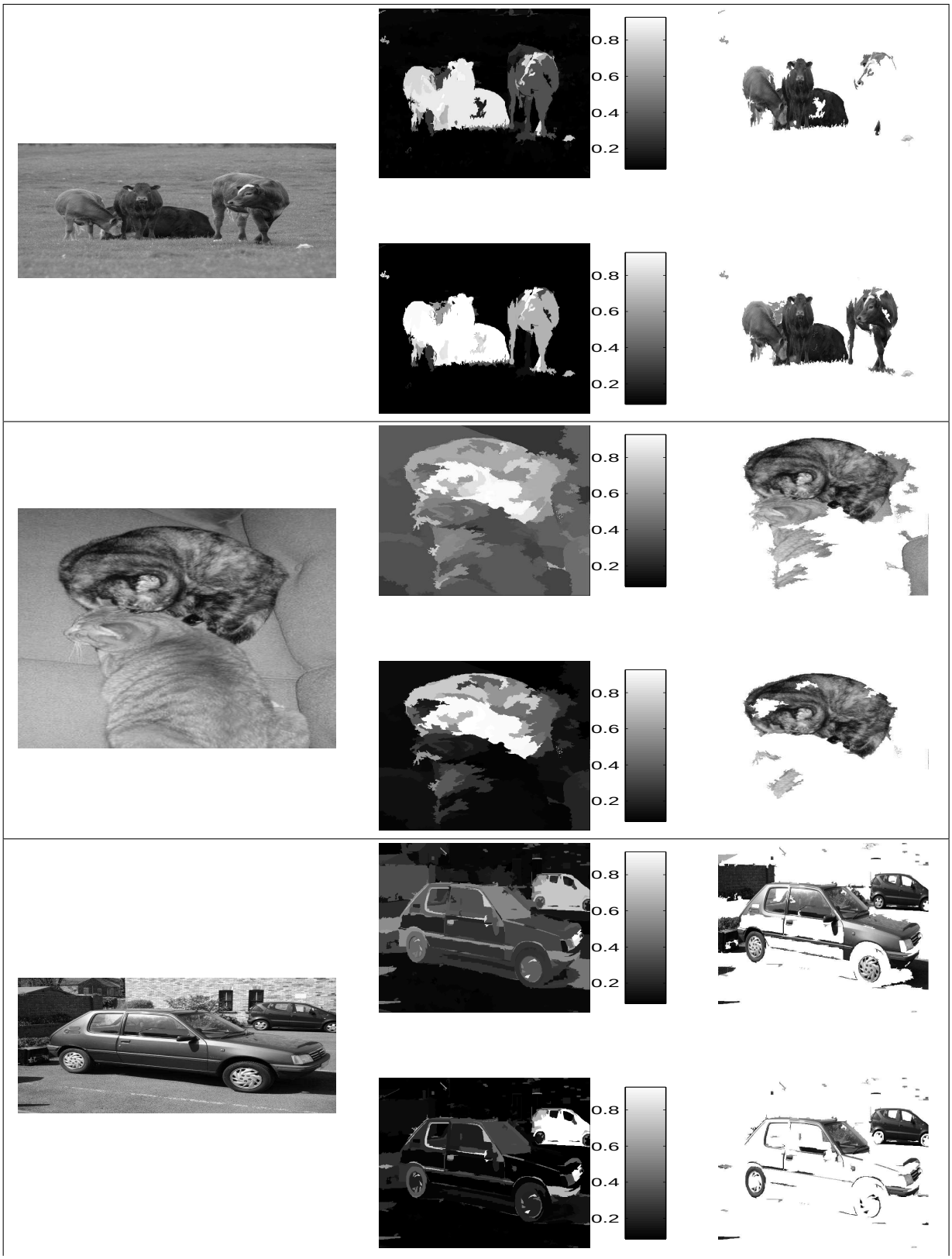


Figure 4. Some sample results for the cow, cat and car detectors. Left: input image. Middle: probability of object before (top) and after (bottom) BP. Right: pixels whose foreground probability is above 30% are shown in their original intensity.

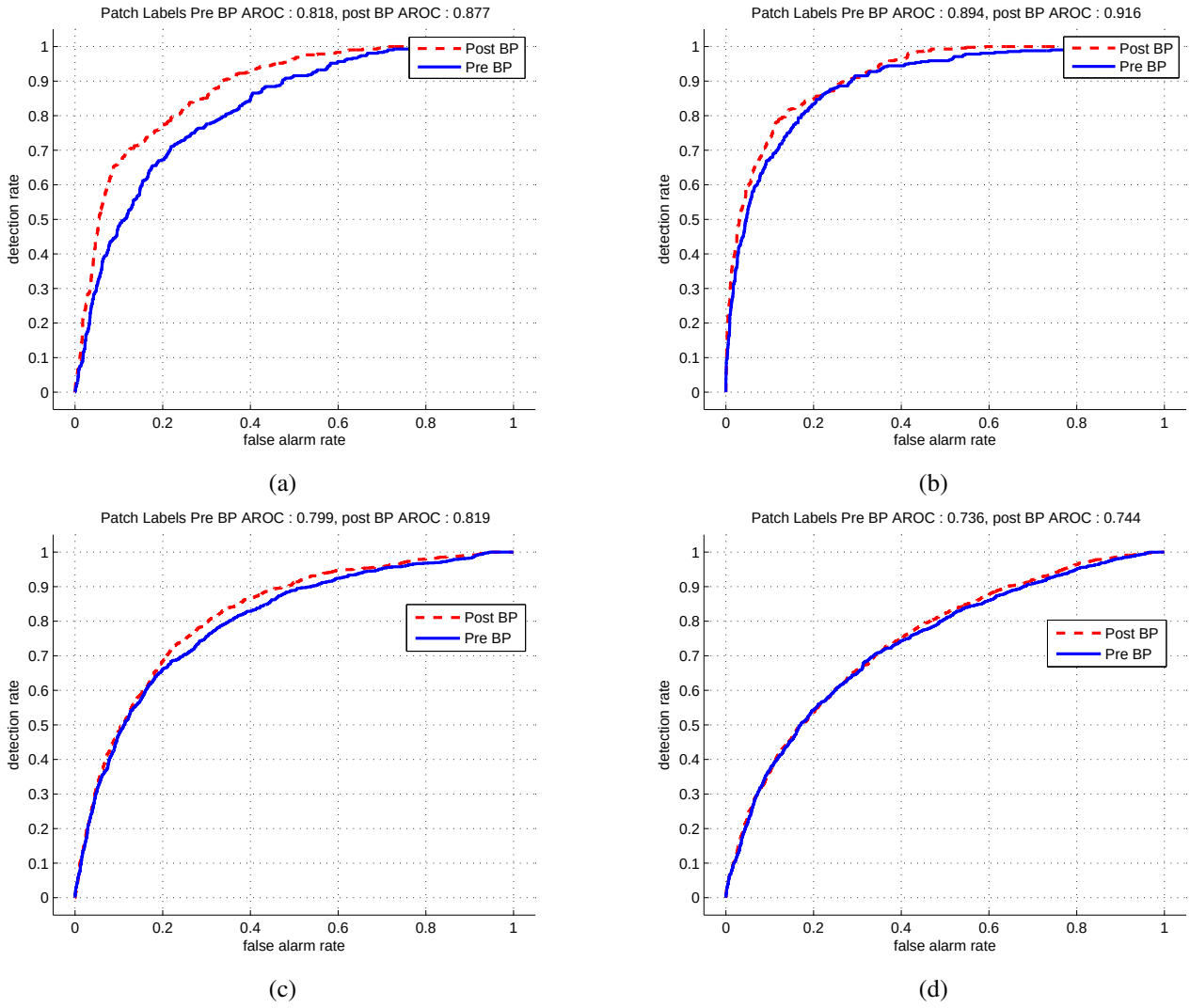


Figure 5. ROC curves for (a) cows with color features, (b) cows with color and texture features, (c) cats with color and texture, and (d) cars with color and texture. Solid blue line is the base level classifier (pre BP), dotted red line is the combined system (post BP).

[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] H. Schneiderman and T. Kanade. A statistical model for 3D object detection applied to faces and cars. In *CVPR*, 2000.

[19] J. Shi and J. Malik. Normalized cuts and image segmentation. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 2000.

[20] J. Shotton, J. Winn, C. Rother, and A. Criminisi. Textonboost: Joint appearance, shape and context modeling for multi-class object recognition and segmentation. In *ECCV*, 2006.

[21] A. Torralba, K. Murphy, and W. Freeman. Sharing visual features for multiclass and multiview object detection. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 2007. To appear.

[22] P. Viola and M. Jones. Rapid object detection using a boosted cascade of simple classifiers. In *CVPR*, 2001.

[23] J. Winn, A. Criminisi, and T. Minka. Object categorization by learned universal visual dictionary. In *IEEE Conf. on Computer Vision and Pattern Recognition*, 2005.