Semantic Contours from Inverse Detectors*

Bharath Hariharan[1], Pablo Arbeláez[1], Lubomir Bourdev[1,2], Subhransu Maji[1] and Jitendra Malik[1]
[1]EECS, U.C. Berkeley, Berkeley, CA 94720
[2]Adobe Systems, Inc., 345 Park Ave, San Jose, CA 95110
{bharath2, arbelaez, lbourdev, smaji, malik}@eecs.berkeley.edu

Abstract

We study the challenging problem of localizing and classifying category-specific object contours in real world images. For this purpose, we present a simple yet effective method for combining generic object detectors with bottom-up contours to identify object contours. We also provide a principled way of combining information from different part detectors and across categories. In order to study the problem and evaluate quantitatively our approach, we present a dataset of semantic exterior boundaries on more than $20,000$ *object instances belonging to* 20 *categories, using the images from the VOC2011 PASCAL challenge [7].*

1. Introduction

Consider Figure 1. We are interested in identifying which contours belong to each of the objects in the image: the boy, the bicycle and the cars. The top-central panel displays the output of the contour detector [2], which uses multiple low-level cues (brightness, color, texture) to estimate the probability of having a boundary at each location in the image. Such a detector is unable to discriminate among contours of different objects, because it does not have access to any category-specific information. We developed a new method for detecting class-specific contours, whose results are shown in the bottom row. The top-right panel shows the annotations we present here in order to study this task, which delineate the exterior outline of each instance of objects for 20 semantic categories.

The inputs of our approach are the output of a bottom-up contour detector such as [2] and the top down detections of an object detector. We present an approach that can be applied to any generic object detector; the only requirement is that it outputs a set of activation windows and corresponding scores. Our approach assigns weights to bottom-up contours based on where they occur in relation to the activations of the detectors. The final strength of the contour is its bottom-up contrast modulated by these weights.

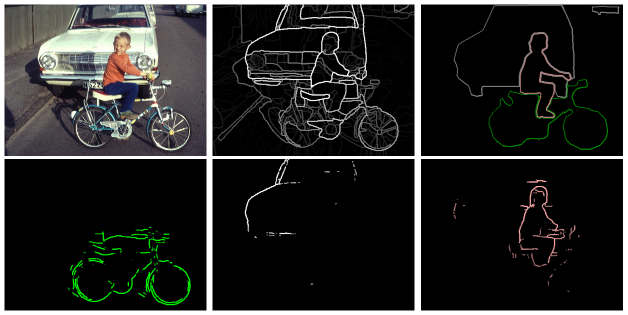
Figure 1. **Top:** Original image, low-level contours using the detector of [2] and ground-truth from our new annotated dataset. **Bottom:** Result of our semantic contour detector for the categories bicycle (green), car (gray) and person (pink).

For generic object detectors, these weights can be learnt. For the specific case of quasi-linear detectors such as HOG, these weights can be determined analytically.

We call the task of localizing class-specific contours *semantic contour detection*. Like in the case of low-level contour detection, we allow semantic contours to be open curves and don't impose the restriction of forming regions. Detecting semantic contours appears to be a novel problem in the field, and only a handful of methods, which we review below, have addressed it explicitly. However, the dual problem, semantic segmentation, has received a lot of attention from the community. Thus, one may wonder, why would semantic contours be useful? One can debate the relative merits of contours and regions. However, as low-level edges have found application in many computer vision problems, we regard semantic contours as an important intermediate representation which is worth studying. For instance, carrying the analogy with segmentation, a semantic contour detector can be thought of as a unary potential in a probabilistic framework for contour-based recognition.

For the purpose of studying semantic contour detection, we present a large-scale annotated dataset of precisely located outlines of $20,000$ objects from 20 categories which we make public. We defined the annotation task as marking closed exterior boundaries in order to make our ground-

*This work was supported by a Berkeley Fellowship, Adobe Systems, Inc., Google, Inc., as well as ONR MURI N00014-10-10933

truth also useful for the evaluation of semantic segmentation.

2. Related Work

The problem of recovering full-object boundaries without category-specific information has been extensively studied in the past. Several research directions have been explored towards that goal. For example, grouping edge pixels based on mid-level Gestalt cues, *e.g*. [18, 23, 12], or recovering occlusion boundaries from a single image [11], or from a sequence [21].

However, the problem of class-specific contour detection in real world images has not yet received the attention it deserves from the community, mainly because of the inherent complexity of the task and the lack of a standardized evaluation framework. Although a minority trend, a seminal line of work has addressed the problem within the paradigm of representing and classifying local image patches. In [20], a boosting framework is proposed for aggregating weak texture classifiers for the purpose of tracking object boundaries. Dollar *et al*. [5] propose an extension of probabilistic boosting trees to combine a rich set of local features. Prasad *et al*. [17] regularize the problem by restricting the set of pixels under consideration to those detected by a low-level edge detector, and use simpler local features and a linear SVM classifier. Mairal *et al*. [13] also reason on low-level edges, but learn dictionaries on multiscale RGB patches with sparse coding and use the reconstruction error curves as features for a linear logistic classifier.

A second factor that has hindered the study of semantic contours is the lack of a large-scale annotated dataset and a standard evaluation protocol. In low-level contour detection, the Berkeley Segmentation Data Set (BSDS) [16] and the Precision-Recall methodology of [15] have served that purpose, and were used in [5, 13] to evaluate the proposed methods without object-specific knowledge. An alternative approach is to measure the improvement obtained in a given application when switching from low-level to category-specific contours, as was done in [17, 13], with an image classification algorithm based on contour matching. However, such a strategy measures only indirectly the accuracy of contours.

The usefulness of annotating at the boundary level has been acknowledged in other vision problems, for instance, Hoeim *et al*. [11] select a subset of contours in the LabelMe dataset [19] to evaluate occlusion boundary detection from a single image; Stein *et al*. [21] released the CMU motion dataset for a similar task, but using a sequence of images. In [6], the annotations of the BSDS are extended to the level of complete objects. Wang *et al*. [22] proposed a dataset with instance-level segmentations, but only for a single category (pedestrians) and at a reduced scale (345 instances). Recently, Ferrari *et al*. [10] have evaluated class-specific boundaries on the ETHZ-shape dataset [9]. However, ETHZ-shape was designed to study object categories that can be represented by a single global shape. Therefore, its 5 object classes are either rigid (apple logo, bottle, mug) or appear in the same canonical pose in all the images (giraffes, swans). Furthermore, the total number of images is reduced (255), each image contains objects of a single category and often only one instance. The largest and most complex recognition dataset possessing instance-level segmentations is the PASCAL VOC2011 challenge [7], with object masks for 20 categories in 2223 images. However, since its exclusive purpose is the evaluation of semantic segmentation, only interior pixels are marked and a bordering region with a width of five pixels labeled *void* hides the exact boundary locations.

In this work, we provide a large-scale annotated dataset of semantic boundaries in real world images, which we call Semantic Boundaries Dataset (SBD). This dataset has object instance boundaries on almost $10,000$ images containing more than $20,000$ objects from 20 categories, using the trainval set of the PASCAL VOC2011 challenge. Thus, we introduce the study of a new task in the most challenging recognition dataset currently available, while providing a natural experimental testbed for the development and evaluation of contour-based matching techniques.

3. The inverse detector

In our approach to detecting semantic contours we build on two sources of information. The first is the output of a bottom-up contour detector. The contours output by such a detector are highly localized since they are based on low level gradient and texture information, but have no class-specificity. The second source of information is the set of activations of various object detectors in the image. The output of an object detector is merely a set of windows where the object is likely to occur. Thus this information, though category specific, is very coarse. Our contribution is to combine these two signals to get the best of both worlds: localized, class-specific contours.

Consider first a simplified situation: suppose we want to detect the contours of a particular category, and suppose that we also have a single monolithic detector ϕ for this category. The central component of our system is a function that takes an image I, the detector ϕ and produces a contour image $S(I, \phi)$. We call this function the inverse detector for ϕ because while ϕ goes from the image to activations, S goes from the activations back to the image (in particular a contour image). In this section we motivate and describe this "inverse detector". In the next section we use this formulation of inverse detectors as a building block within a complete system that handles more sophisticated object detectors, and leverages information from other categories.

A detector when run on the image produces a set of activation windows. One naive approach would be to highlight all contours that lie inside the activation windows and suppress everything else. However, detectors often include considerable context around the object and hence these windows can contain spurious contours. The problem will be exacerbated when the object suffers from heavy occlusion. Hence we need to extract more fine-grained information.

Our intuition is that, given an activation of an object detector, it is possible to guess the rough locations and orientations of the object contours in the activation window. For instance, given a window in which a pedestrian detector has fired, we can predict the rough location of the head and shoulders, and hence we can predict the rough locations and orientations of the corresponding contours. The location of a pixel in the activation window, and the strength and rough orientation of the contour at that pixel, are useful cues in deciding if the pixel lies on the contour of the object.

We now formalize this intuition. Given an image I denote the output of the contour detector by $\mathbf{G}$, where G_{ij} scores the likelihood of a pixel (i, j) lying on a contour. Denote the l activation windows of the detector ϕ on I by $R_1, \ldots, R_l$. Each activation window R_k has a corresponding score s_k.

For each pixel (i, j) we construct a feature vector as follows. Each activation window is divided into S spatial bins or cells. The contours are also binned into O orientation bins, giving rise to a total of $N = SO$ bins. For a pixel (i, j), for an activation window R_k, we assign the pixel into one of the bins, thus encoding the rough location of the pixel relative to R_k and the rough orientation of the contour at (i, j). Let $n(R_k, i, j)$ be the index of the bin into which the pixel (i, j) falls. Define a vector that encodes $n(R_k, i, j)$:

$$\mathbf{x}_\phi(i, j, R_k) = \begin{cases} G_{ij}\mathbf{e}_{n(R_k,i,j)} & \text{if } (i, j) \in R_k \\ \mathbf{0} & \text{otherwise} \end{cases} \quad (1)$$

where $\mathbf{e}_n$ is an N-dimensional vector with 1 in the nth position and 0 otherwise. The feature vector for pixel (i, j) is the weighted sum of all the vectors $\mathbf{x}_\phi(i, j, R_k)$, with the scores s_k as the weights. That is, define the feature vector:

$$\mathbf{x}_\phi(i, j) = \sum_{k=1}^{l} s_k \mathbf{x}_\phi(i, j, R_k) \quad (2)$$

Thus $\mathbf{x}_\phi(i, j)$ encodes the orientation and typical location of (i, j) in a detection template. For example, if (i, j) lies on the top of a person's head, $\mathbf{x}_\phi(i, j)$ will be highly spiked around the bin corresponding to ”location=top, orientation=horizontal”. Our “inverse detector” then has the form:

$$S(I, \phi)_{ij} = \mathbf{w}_\phi^t \mathbf{x}_\phi(i, j) \quad (3)$$

The important task now is to define the weights $\mathbf{w}_\phi$. In the next section we describe how we can learn the weights $\mathbf{w}_\phi$ for any generic detector. The binning process that we have used here is similar to that used in HOG detectors [4] and so when ϕ is a HOG detector, or a similar quasi-linear detector, we can “look inside” the detector and come up with weights $\mathbf{w}$ analytically, a method we explore in Appendix A.

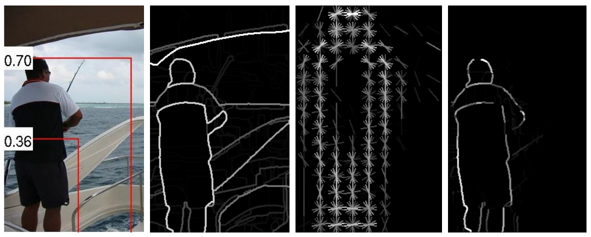

Figure 2. The first panel from the left shows an image and the detections of a pedestrian detector. The activation windows R_k are shown in red boxes and the scores s_k are indicated on the top of the boxes. Some of these activation windows, such as the smaller one in the figure, might be false positives. The second panel shows the bottom-up contours $\mathbf{G}$. The third panel shows the weights $\mathbf{w}_\phi$ and the final panel shows the output of the inverse detector. Note how only the relevant contours of the person are highlighted.

4. Localizing semantic contours using inverse detectors

Armed with this formulation of inverse detectors, we now describe our complete system.

As a bottom-up contour detector, we use as is the contour detector of[2], which has shown state of the art performance on benchmarks such as the BSDS [16]. The object detection framework we consider is Poselets [3], although our approach can easily be applied to other systems such as [8]. In the poselets framework, each object category has roughly 100-200 poselet types. Each poselet type can be thought of as a detector for a part of the object: for instance there might be a poselet corresponding to the head and shoulders of a person. The final detector for the category combines the activations of all the poselets.

Our system consists of two stages. In the first stage, we train inverse detectors for each poselet type. In the second stage we combine the output of these inverse detectors to produce category-specific contours for each category. Finally, we also consider ways of combining information across classes in order to improve performance.

4.1. Inverse detectors for each poselet

Let $\phi_1^C, \phi_2^C, \ldots \phi_P^C$ be the P poselet types for category C. Each of these poselet types provides information about where the contours of C can be, and so we train separate inverse detectors for each of these poselets.

To train each inverse detector, we pose the task of detecting contours of C as a problem of classifying pixels: each pixel must be classified as belonging to a contour of C or not. We then train a linear SVM using the feature vector $\mathbf{x}_\phi(i,j)$ described in (2):

$$f(\mathbf{x}_\phi(i,j)) = \text{sign}(\mathbf{w}^t\mathbf{x}_\phi(i,j)) \tag{4}$$

We use the weight vector of the SVM as the weights $\mathbf{w}_\phi$ to define the inverse detector (3). (In other words, the inverse detector output for a pixel is simply the score of the SVM)

The positive training examples for each of the inverse detectors are pixels that lie on the contours of C, while all other pixels are negative examples. Because human annotation is in general noisy, we do not take the human annotated boundaries directly. Instead, we threshold the bottom-up contour image at a low threshold, and match it to the human annotated boundaries using the bipartite matching of [15]. The pixels that are matched form the positive examples, and those that are not matched form the negative examples.

4.2. Combining the inverse detectors

After we have trained inverse detectors for each poselet, we are faced with the task of combining the outputs of each of these inverse detectors. Again we frame this as a pixel classification task, and train a linear SVM. The features we use in this case are the outputs of the inverse detectors corresponding to each of the poselets:

$$\mathbf{x}_C(i,j) = [S(I,\phi_1^C)_{ij}, \ldots S(I,\phi_P^C)_{ij}]^t \tag{5}$$

Our contour detector then outputs, for each pixel, the score of the linear SVM:

$$S(I,C)_{ij} = \mathbf{w}_C^T\mathbf{x}_C(i,j) \tag{6}$$

4.3. Combining information across categories

Our contour detector as we have described it considers each category independently. However, this disregards crucial information. For instance, cow heads might be frequently mistaken for sheep heads, but if we have both the cow and sheep head detectors, we might be able to tell the difference.

Note however, that at the pixel level a particular pixel can belong to the contours of two categories, because pixels on the boundary between two categories belong to both categories. Hence we will still train a separate contour detector per category. The difference however is that the features we consider here combine information across categories. We consider two choices:

1. We first train independent contour detectors for each category as in the previous subsection. Then we use as features the outputs of these contour detectors to train a second level of contour detectors. The feature vector for this second level of contour detectors becomes:

$$\mathbf{x}(i,j) = [S(I,C_1)_{ij}, \ldots, S(I,C_M)_{ij}]^t \tag{7}$$

where $C_1, \ldots C_M$ are the various categories.

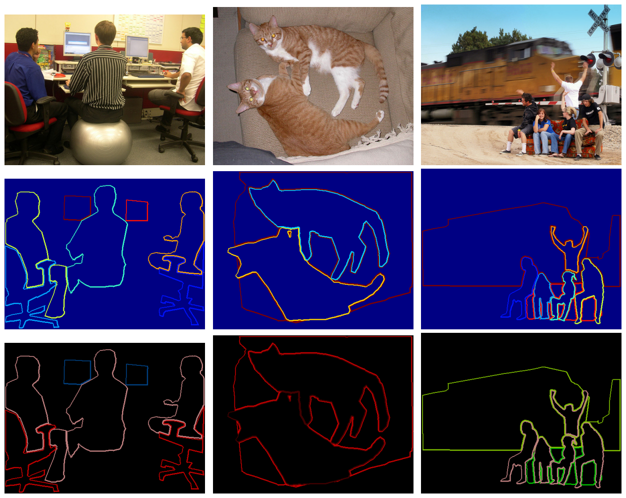
Figure 3. Example of annotations in the Semantic Boundaries Dataset. **Top:** Original Images. **Middle:** Per instance boundaries. **Bottom:** Per category boundaries. We display the semantic boundaries using the same color code for object classes as the PASCAL dataset.

2. We train only one level of contour detectors. But as features we use the inverse detectors corresponding to the poselets of all categories and not just the category in question. The feature vector in this case is merely the concatenation of the feature vectors $\mathbf{x}_C$ defined in (5) :

$$\mathbf{x}(i,j) = [\mathbf{x}_{C_1}(i,j), \ldots, \mathbf{x}_{C_M}(i,j)]^t \tag{8}$$

Again, we train a linear SVM and use the score of the SVM at each pixel as the output of our semantic contour detector:

$$S^*(I,C)_{ij} = \mathbf{w}_C^t\mathbf{x}(i,j) \tag{9}$$

where $\mathbf{x}(i,j)$ is either as in (8) or (7).

We evaluate empirically these choices, and other aspects of our system, in section 6.

5. Semantic Boundaries Dataset (SBD)

5.1. Ground-truth

Binary figure-ground segmentations were collected for all the objects and categories in the images of the trainval set of the VOC2011 PASCAL challenge. In order to obtain precisely located boundaries, the cropped bounding box of each instance was rescaled to a standard size of 500×500 pixels and presented to human observers, who outlined the object boundary by marking vertices of a polygon. We implemented the task using Amazon Mechanical Turk [1] and retrieved an average of 5 annotations made by different subjects per object instance, for a total number of initial figure/ground object masks above $100,000$[14].

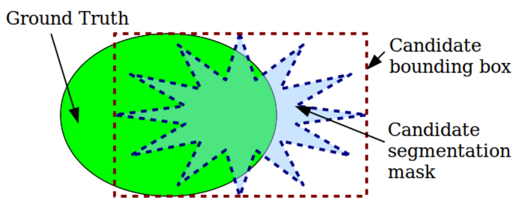

Figure 4. Comparison of our benchmark with standard recognition evaluation metrics from PASCAL. The bounding box(red) overlaps by more than 50% with that of the ground truth and is considered correct. The region(blue) obtains a segmentation score of 0.5, the value of its overlap with the ground-truth mask. In contrast, both Precision and Recall in our evaluation are practically zero in this case, because the boundary localization is inaccurate.

For each instance, we counted as object pixels those marked by the majority (three or more) subjects. The instances where less than three humans agreed were labeled as *void*. This provides consolidated object masks for all the instances, from which we extract boundaries. However, in the presence of partial occlusion between different objects, human annotations may vary, as some subjects may mark only the visible parts while others the full object. To solve this ambiguity, we considered all the pairs of object masks that conflict spatially and deleted manually the semantic boundaries that were occluded. Figure 3 presents some examples of our Semantic Boundaries dataset.

Note that our annotations provide ground-truth for the study of category-specific boundary detection and semantic segmentation and also for recognition methods based on contour matching . The SBD can be seen as a scaling of the current PASCAL segmentation subset to the full dataset, an increase by a factor of 5 in the number of images and objects. The 11318 images in the SBD are divided into 8498 training images and 2820 test images. The test images form a subset of the VOC2011 validation set.

5.2. Benchmarks

In order to benchmark semantic contours, we adopt the widely used Precision-Recall framework. For a detector producing binary output, Precision (P) is defined as the fraction of true contours among detections and Recall (R) is the fraction of ground-truth contours detected. For a detector with soft output, a Precision-Recall curve, parametrized by the detection score, characterizes its performance across all operating regimes. We report two summary statistics: the Average Precision (**AP**) over the whole Recall range and the maximal F-measure (**MF**), defined as $F = 2PR/(P + R)$. Figure 4 compares our benchmark with standard recognition evaluation metrics.

We evaluate semantic contours for each category independently on the 2820 test images of the SBD and produce individual Precision-Recall curves. We consider as positives the exterior boundaries of the objects of that category and as negatives the background and all pixels belonging to objects from other categories. Pixels in the interior of objects from the category of interest are not taken into account in the evaluation because internal contours might prove useful for downstream applications such as recognition.

We follow the implementation of [15], based on bipartite matching of boundary pixels and used also in, *e.g.* [21, 10, 11]. In the experiments, we tolerate a localization error of 2% of the image diagonal for declaring a detected pixel as true positive.

Note that one could also consider evaluating all the categories jointly for the task of parsing the boundaries in the image. However, per-category evaluation provides more detailed information about the performance of each individual semantic boundary detector. Furthermore, assigning a single label to a boundary pixel is not always correct, as contact boundaries belong to the two objects they separate.

6. Experiments on Semantic Contour Detection

We evaluate our semantic contour detector on the SBD for the 20 PASCAL categories. For each category, we train a semantic contour detector on the train data and then measure its performance on all the images of the test set.

The low-level contour detector of [2] is an indicator of the inherent difficulty of the semantic contour detection task, as it does not have access to any category-specific information. It is also a natural baseline for comparison with our approach, because it provides the initial locations where we predict the probability of being an object contour. Both the low level contour detector and our contour detector fire exactly at the same locations in the image, and the difference is the strength of their response.

Table 1 and Figure 5 present the complete results. The performance of the bottom-up contour detector confirms the generality of the PASCAL images and the difficulty of the task of semantic contour detection. It obtains an average maximal F-measure of only 2% and a mean Average Precision of 4%. In $19/20$ categories, both statistics are below 10% and the curves lie flat at the bottom of the graph.

The results of our semantic contour detector validate empirically our approach. On average across categories, we improve both the MF and the AP by a factor of 5 with respect to bottom-up contour detection. Further, the single stage contour detector that combines the outputs of all inverse detectors across all categories does significantly better than the two stage contour detector. This is to be expected since the former operates on strictly more information and hence can make better choices. Our algorithm performs the best (MF $\geq 20\%$) on transportation means (aeroplane, bicycle, bus, car, motorbike, train), people and objects with simple shape (bottles and TV monitors). Our performance

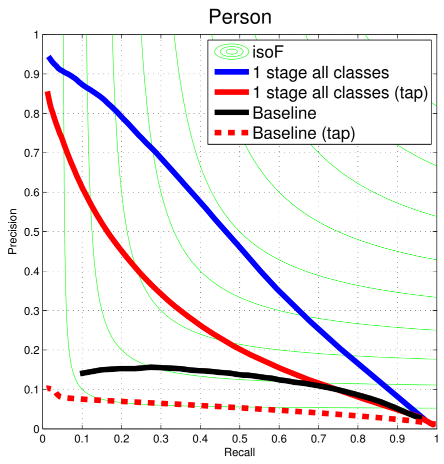

Figure 6. The effect of the choice of contour detector on our approach, evaluated on the person category. We compare tap filter outputs with the contour detector [2]

on animals (horse, cat, dog, sheep, cow) is lower but still much better than the baseline. We perform the worst on hard to detect categories, such as chairs, dining tables, potted plants, boats and birds. Note that our method preserves full recall for low-enough thresholds in all the curves of Fig. 5.

We also show in the table results for when the inverse detector is not learnt but constructed from the object detector, as explained in Appendix A. This method is slightly inferior to learning the inverse detector directly, which is to be expected since the weights are not optimized for the semantic contour detection task. However, because it requires no training, this method can potentially prove useful when training an inverse detector from scratch is infeasible either because of computational complexity or because of unavailability of training data.

Figure 6 and Table 2 show some more results on people. We compare our results with what we get when we replace the bottom-up contour detector with tap filter outputs. (The tap filter involves convolution with the matrix $[-1\ 0\ 1]$ for the gradient in x, and a transposed matrix for the gradient in y.) Tap outputs are very noisy and fire at many more locations. Hence, as expected, the results with tap filter are significantly worse. Nevertheless, our semantic contour detector still provides a significant boost compared to the baseline (MF of 32% as opposed to 11% for the baseline).

7. Conclusions

This paper proposes three distinct contributions: A new task, a new annotated dataset with evaluation framework and a semantic contour detector that can be used for future reference. Our semantic contour detector is completely general and can be used with any object detector.

method	Bottom up	Bottom up(tap)	1-stage (all classes)	1-stage tap (all classes)
MF(%)	21.8	11.1	48.2	32.3
AP(%)	10.7	5.1	45.7	25.7

Table 2. Results on the person category

References

[1] Amazon. Amazon mechanical turk. https://www.mturk.com/mturk/welcome. 4
[2] P. Arbelaez, M. Maire, C. Fowlkes, and J. Malik. Contour detection and hierarchical image segmentation. *IEEE Trans. on PAMI*, 2011. 1, 3, 5, 6, 7
[3] L. Bourdev, S. Maji, T. Brox, and J. Malik. Detecting people using mutually consistent poselet activations. In *Proc. ECCV*, 2010. 3
[4] N. Dalal and B. Triggs. Histograms of oriented gradients for human detection. In *Proc. CVPR*, 2005. 3
[5] P. Dollar, Z. Tu, and S. Belongie. Supervised learning of edges and object boundaries. In *Proc. CVPR*, 2006. 2
[6] I. Endres and D. Hoiem. Category independent object proposals. In *Proc. ECCV*, 2010. 2
[7] M. Everingham, L. Van Gool, C. K. I. Williams, J. Winn, and A. Zisserman. The PASCAL Visual Object Classes Challenge 2010 (VOC2010) Results. http://www.pascal-network.org/challenges/VOC/voc2010/workshop/index.html. 1, 2
[8] P. Felzenszwalb, R. Girshick, D. McAllester, and D. Ramanan. Object detection with discriminatively trained part based models. *IEEE Trans. on PAMI*, 2010. 3
[9] V. Ferrari, F. Jurie, and C. Schmid. ETHZ shape classes v 1.2. http://www.vision.ee.ethz.ch/ calvin/datasets.html. 2
[10] V. Ferrari, F. Jurie, and C. Schmid. From images to shape models for object detection. *International Journal of Computer Vision*, 2010. 2, 5
[11] D. Hoiem, A. Efros, and M. Hebert. Recovering occlusion boundaries from an image. *IJCV*, 2011. 2, 5
[12] I. Kokkinos. Highly accurate boundary detection and grouping. In *Proc. CVPR*, 2010. 2
[13] J. Mairal, M. Leordeanu, F. Bach, M. Hebert, and J. Ponce. Discriminative sparse image models for class-specific edge detection and image interpretation. In *Proc. ECCV*, 2008. 2
[14] S. Maji. Large scale image annotations on amazon mechanical turk. Technical report, EECS Department, University of California, Berkeley, 2011. 4
[15] D. Martin, C. Fowlkes, and J. Malik. Learning to detect natural image boundaries using local brightness, color and texture cues. *IEEE Trans. on PAMI*, 2004. 2, 4, 5
[16] D. Martin, C. Fowlkes, D. Tal, and J. Malik. A database of human segmented natural images and its application to evaluating segmentation algorithms and measuring ecological statistics. In *Proc. ICCV*, 2001. 2, 3
[17] M. Prasad, A. Zisserman, M. Fitzgibbon, M. Kumar, and P. Torr. Learning class-specific edges for object detection and segmentation. In *Proc. CVPR*, 2006. 2
[18] X. Ren, C. Fowlkes, and J. Malik. Scale-invariant contour completion using conditional random fields. In *Proc. ICCV*, 2005. 2
[19] B. Russell, A. Torralba, K. Murphy, and W. Freeman. Labelme: A database and web-based tool for image annotation. *IJCV*, 2008. 2
[20] A. Shahrokni, F. Fleuret, and P. Fua. Classifier-based contour tracking for rigid and deformable objects. In *Proc. BMVC*, 2005. 2
[21] A. Stein, D. Hoiem, and M. Hebert. Learning to extract object boundaries using motion cues. In *Proc. ICCV*, 2007. 2, 5
[22] L. Wang, J. Shi, G. Song, and I. Shen. Object detection combining recognition and segmentation. In *Proc. ACCV*, 2007. 2
[23] Q. Zhu, G. Song, and J. Shi. Untangling cycles for contour grouping. In *Proc. ICCV*, 2007. 2

method metric	Baseline [2] MF	1-stage (1 class) MF	2-stage MF	1-stage (all classes) MF	1-stage HOG (all classes) MF	Baseline [2] AP	1-stage (1 class) AP	2-stage AP	1-stage (all classes) AP	1-stage HOG (all classes) AP
Aeroplane	4.7	40.3	40.5	42.6	43.3	2.0	29.3	35.5	38.4	39.5
Bicycle	2.1	47.9	48.0	49.5	46.7	0.8	24.1	25.3	29.6	29.3
Bird	3.1	16.2	16.6	15.7	15.0	1.3	9.5	9.8	9.6	8.8
Boat	1.7	16.6	16.4	16.8	20.7	0.7	9.7	9.6	9.9	12.1
Bottle	3.1	35.9	35.9	36.7	35.5	1.0	24.8	24.1	24.2	23.4
Bus	3.7	42.0	42.0	43.0	39.4	1.3	33.9	29.1	33.6	32.2
Car	4.1	39.8	39.9	40.8	36.7	1.7	30.8	32.8	31.3	28.7
Cat	6.5	21.3	21.9	22.6	18.9	2.5	15.4	17.3	17.3	14.1
Chair	5.7	15.3	16.6	18.1	19.0	2.3	8.6	9.1	10.7	11.6
Cow	2.7	20.5	24.1	26.6	27.2	0.9	12.4	14.1	16.4	16.9
Dining Table	2.4	8.3	9.2	10.2	11.6	0.8	2.9	3.1	3.7	4.6
Dog	7.2	15.6	18.0	18.0	17.8	3.1	10.3	12.1	12.1	10.5
Horse	3.7	32.4	33.5	35.2	33.8	1.4	25.5	26.7	28.5	26.6
Motorbike	2.5	27.7	28.1	29.4	28.8	1.0	19.3	20.8	20.4	21.4
Person	21.8	47.8	47.6	48.2	44.9	11.1	46.0	46.5	45.7	41.2
Potted Plant	1.7	13.7	13.9	14.3	15.9	0.6	7.7	7.0	7.6	9.0
Sheep	1.5	23.8	25.0	26.8	26.6	0.6	13.1	14.0	16.1	16.0
Sofa	4.3	10.7	9.7	11.2	12.1	1.4	5.4	4.5	5.7	5.6
Train	2.6	20.8	20.6	22.2	22.6	1.0	13.4	13.3	14.6	14.3
TV Monitor	4.6	31.7	31.8	32.0	30.5	1.3	22.7	22.8	22.7	21.2
Average	4.5	26.4	27.0	**28.0**	27.3	1.8	18.2	18.9	**19.9**	19.3

Table 1. Comparison of our semantic contour detector with respect to the baseline given by low-level contour detector [2]. We report the Average Precision (AP) and the maximal F-measure (MF) in $\%$. "1-stage(1-class)" treats each class independently. "2-stage" combines all categories in two stages(see (8)) while "1-stage (all classes)" does so using only a single stage(see (7)). "1-stage HOG (all classes)" uses inverse detector weights that are determined analytically as explained in Appendix A.

A. Inverse detectors for HOG-based detectors

A HOG-based object detector works in the following way. Given a detection window R, the window is divided into spatial bins or cells. Orientations are also divided into bins. (Observe that this binning process is identical to the one we described in section 3.) Each pixel (i, j) in R contributes a vote proportional to g_{ij} to its bin $n(R, i, j)$ where $\mathbf{g}$ is the image gradient, thus giving a histogram $\mathbf{h}(\mathbf{g})$. In the simplest HOG detector, the final classifier is a linear classifier of the histogram:

$$score(R) = \mathbf{w}_{HOG}^{t}\mathbf{h}(\mathbf{g}) \tag{10}$$

The image gradients or edges that are discriminative for the object are those that fall in bins that have a high weight. We claim that the contours of an object are discriminative for the object and hence must fall in positively weighted bins. In other words, $\mathbf{w}_{HOG}$ gives a high weight to the probable locations of the contours. Hence we can take $\mathbf{w}_{\phi} = \mathbf{w}_{HOG}$ in (3).

Most common HOG based object detectors however involve a non-linear contrast normalization step before the linear classifier. The window is divided into overlapping blocks, and the histograms of each block are separately normalized and concatenated together to give the feature vector. Thus the detector is a non-linear function of the histogram:

$$score(R) = \psi(\mathbf{h}(\mathbf{g})) \tag{11}$$

One way to get around this non-linear function is to construct a linear approximation. To do so, we first observe that what we want is to find out which contour pixels have the most positive impact on the detection score. This suggests that we are concerned with the action of ψ on the contour image $\mathbf{G}$, and so we should linearize ψ about $\mathbf{G}$. Further, since $\mathbf{G}$ is sparse and almost 0, we can do a further approximation and linearize ψ around $\mathbf{0}$:

$$\tilde{\psi}(\mathbf{h}) \triangleq \psi(0) + \sum_{n} h_n \left.\frac{\partial\psi(\mathbf{h})}{\partial h_n}\right|_{\mathbf{h}=\mathbf{0}} \tag{12}$$

where n indexes the histogram bins and the partial derivative can be approximated by a finite difference. The linear approximation $\tilde{\psi}$ gives the weights $\mathbf{w}_{HOG}$, which we can then use. The results corresponding to this version of the inverse detector are shown in Table 1.

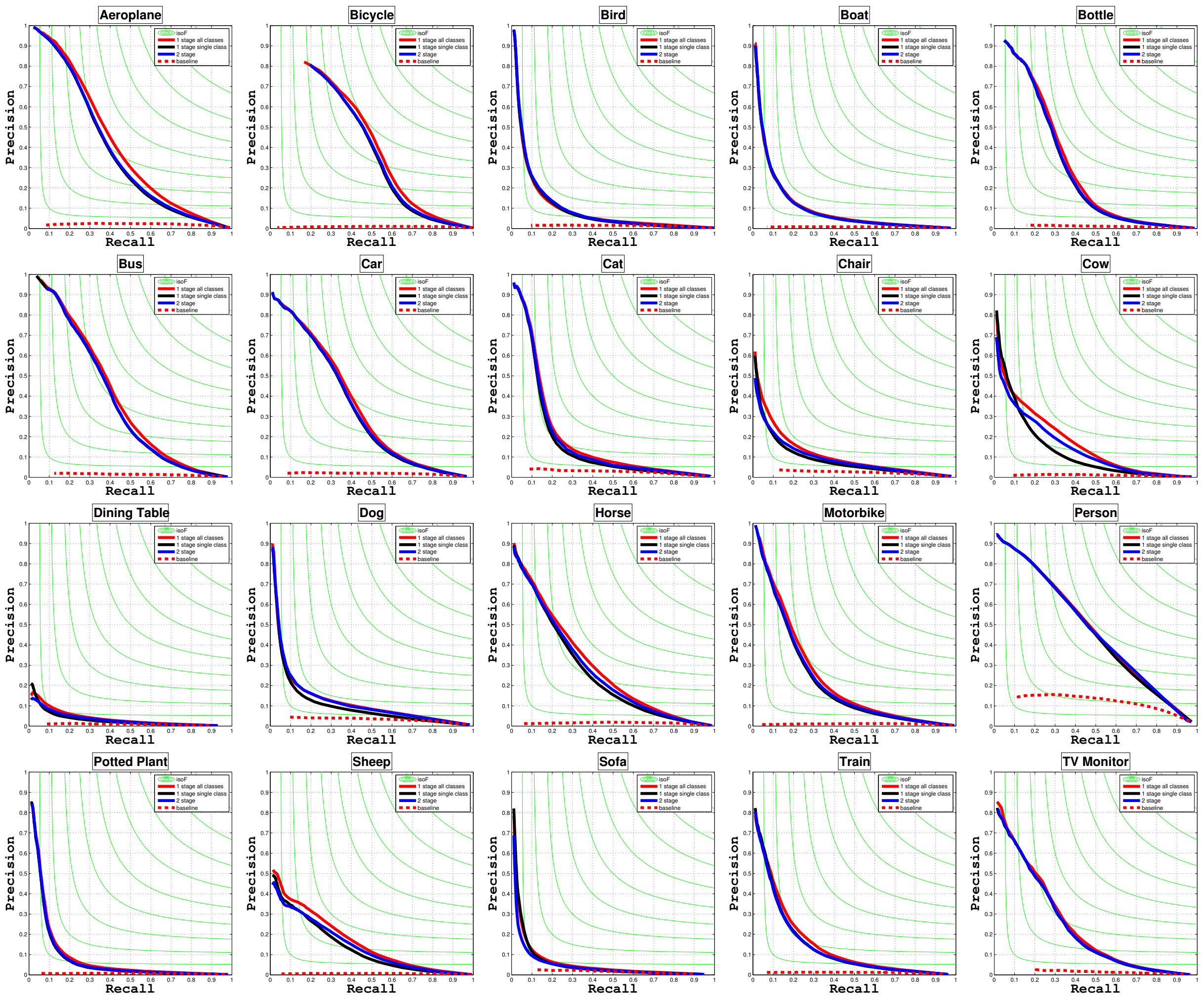

Figure 5. Evaluation results on the Semantic Boundaries Dataset.

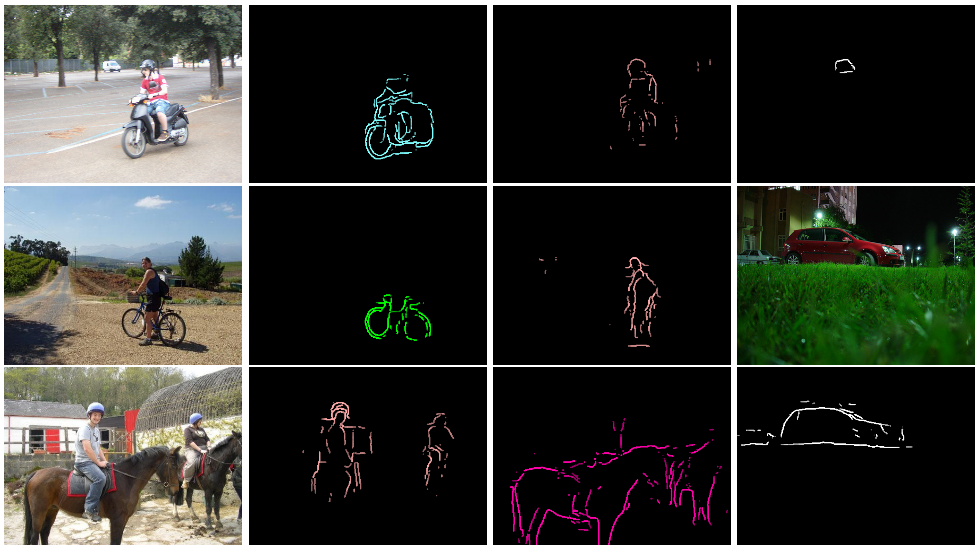

Figure 7. Results on semantic contour detection. The category is indicated by the color(motorbike: blue, car: gray, bicycle: green, horse: magenta, person: pink), following the PASCAL convention.