

# High-for-Low and Low-for-High: Efficient Boundary Detection from Deep Object Features and its Applications to High-Level Vision

Gedas Bertasius  
University of Pennsylvania  
gberta@seas.upenn.edu

Jianbo Shi  
University of Pennsylvania  
jshi@seas.upenn.edu

Lorenzo Torresani  
Dartmouth College  
lt@dartmouth.edu

## Abstract

*Most of the current boundary detection systems rely exclusively on low-level features, such as color and texture. However, perception studies suggest that humans employ object-level reasoning when judging if a particular pixel is a boundary. Inspired by this observation, in this work we show how to predict boundaries by exploiting object-level features from a pretrained object-classification network. Our method can be viewed as a High-for-Low approach where high-level object features inform the low-level boundary detection process. Our model achieves state-of-the-art performance on an established boundary detection benchmark and it is efficient to run.*

*Additionally, we show that due to the semantic nature of our boundaries we can use them to aid a number of high-level vision tasks. We demonstrate that by using our boundaries we improve the performance of state-of-the-art methods on the problems of semantic boundary labeling, semantic segmentation and object proposal generation. We can view this process as a Low-for-High scheme, where low-level boundaries aid high-level vision tasks.*

*Thus, our contributions include a boundary detection system that is accurate, efficient, generalizes well to multiple datasets, and is also shown to improve existing state-of-the-art high-level vision methods on three distinct tasks.*

## 1. Introduction

In the vision community, boundary detection has always been considered a low-level problem. However, psychological studies suggest that when a human observer perceives boundaries, object level reasoning is used [13, 25, 17]. Despite these findings, most of the boundary detection methods rely exclusively on low-level color and gradient features. In this work, we present a method that uses object-level features to detect boundaries. We argue that using object-level information to predict boundaries is more sim-

	Low-Level Task	High-Level Tasks					
		SBL		SS	OP		
		BD	ODS	MF	AP	PI-IOU	MR
SotA	0.76 [27]			28.0 [11]	19.9 [11]	45.8 [5]	0.88 [31]
<b>HFL</b>	<b>0.77</b>			<b>62.5</b>	<b>54.6</b>	<b>48.8</b>	<b>0.90</b>

Table 1: Summary of results achieved by our proposed method (HFL) and state-of-the-art methods (SotA). We provide results on four vision tasks: Boundary Detection (BD), Semantic Boundary Labeling (SBL), Semantic Segmentation (SS), and Object Proposal (OP). The evaluation metrics include ODS F-score for BD task, max F-score (MF) and average precision (AP) for SBL task, per image intersection over union (PI-IOU) for SS task, and max recall (MR) for OP task. Our method produces better results in each of these tasks according to these metrics.

ilar to how humans reason. Our boundary detection scheme can be viewed as a *High-for-Low* approach where we use high-level object features as cues for a low-level boundary detection process. Throughout the rest of the paper, we refer to our proposed boundaries as *High-for-Low* boundaries (HFL).

We present an efficient deep network that uses object-level information to predict the boundaries. Our proposed architecture reuses features from the sixteen convolutional layers of the network of Simonyan et al. [29], which we refer to as VGG net. The VGG net has been trained for object classification, and therefore, reusing its features allows our method to utilize high-level object information to predict HFL boundaries. In the experimental section, we demonstrate that using object-level features produces semantically meaningful boundaries and also achieves above state-of-the-art boundary detection accuracy.

Additionally, we demonstrate that we can successfully apply our HFL boundaries to a number of high-level vision tasks. We show that by using HFL boundaries we improve the results of three existing state-of-the-art methods on the tasks of semantic boundary labeling, semantic segmenta-

tion and object proposal generation. Therefore, using HFL boundaries to boost the results in high level vision tasks can be viewed as a *Low-for-High* scheme, where boundaries serve as low-level cues to aid high-level vision tasks.

We present the summarized results for the boundary detection and the three mentioned high-level vision tasks in Table 1. Specifically, we compare our proposed method and an appropriate state-of-the-art method for that task. As the results indicate, we achieve better results in each of the tasks for each presented evaluation metric. We present more detailed results for each of these tasks in the later sections.

In summary, our contributions are as follows. First, we show that using object-level features for boundary detection produces perceptually informative boundaries that outperform prior state-of-the-art boundary detection methods. Second, we demonstrate that we can use HFL boundaries to enhance the performance on the high-level vision tasks of semantic boundary labeling, semantic segmentation and object proposal. Finally, our method can detect boundaries in near-real time. Thus, we present a boundary detection system that is accurate, efficient, and is also applicable to high level vision tasks.

## 2. Related Work

Most of the contour detection methods predict boundaries based purely on color, text, or other low-level features. We can divide these methods into three broad categories: spectral methods, supervised discriminative methods and deep learning based methods.

Spectral methods formulate contour detection problem as an eigenvalue problem. The solution to this problem is then used to reason about the boundaries. The most successful approaches in this genre are the MCG detector [2], gPb detector [1], PMI detector [14], and Normalized Cuts [28].

Some of the notable discriminative boundary detection methods include sketch tokens (ST) [18], structured edges (SE) [6] and sparse code gradients (SCG) [23]. While SCG use supervised SVM learning [4], the latter two methods rely on a random forest classifier and models boundary detection as a classification task.

Recently there have been attempts to apply deep learning to the task of boundary detection. SCT [20] is a sparse coding approach that reconstructs an image using a learned dictionary and then detect boundaries. Both  $N^4$  fields [10] and DeepNet [16] approaches use Convolutional Neural Networks (CNNs) to predict edges.  $N^4$  fields rely on dictionary learning and the use of the Nearest Neighbor algorithm within a CNN framework while DeepNet uses a traditional CNN architecture to predict contours.

The most similar to our approach is DeepEdge [3], which uses a multi-scale bifurcated network to perform contour detection using object-level features. However, we show that our method achieves better results even without

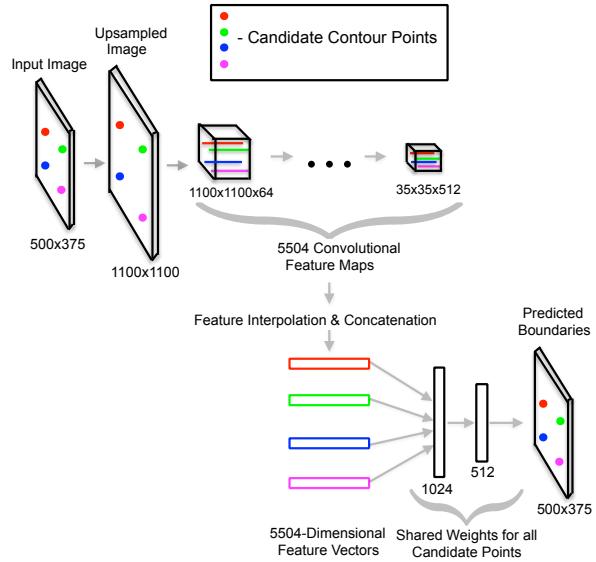


Figure 1: An illustration of our architecture (best viewed in color). First we extract a set of candidate contour points. Then we upsample the image and feed it through 16 convolutional layers pretrained for object classification. For each candidate point, we find its correspondence in each of the feature maps and perform feature interpolation. This yields a 5504-dimensional feature vector for each candidate point. We feed each of these vectors to two fully connected layers and store the predictions to produce a final boundary map.

the complicated multi-scale and bifurcated architecture of DeepEdge. Additionally, unlike DeepEdge, our system can run in near-real time.

In comparison to prior approaches, we offer several contributions. First, we propose to use object-level information to predict boundaries. We argue that such an approach leads to semantic boundaries, which are more consistent with humans reasoning. Second, we avoid feature engineering by learning boundaries from human-annotated data. Finally, we demonstrate excellent results for both low-level and high-level vision tasks. For the boundary detection task, our proposed HFL boundaries outperform all of the prior methods according to both F-score metrics. Additionally, we show that because HFL boundaries are based on object-level features, they can be used to improve performance in the high-level vision tasks of semantic boundary labeling, semantic segmentation, and object proposal generation.

## 3. Boundary Detection

In this section, we describe our proposed architecture and the specific details on how we predict HFL boundaries using our method. The detailed illustration of our architecture is presented in Figure 1.



Figure 2: A visualization of selected convolutional feature maps from VGG network (resized to the input image dimension). Because VGG was optimized for an object classification task, it produces high activation values on objects and their parts.

### 3.1. Selection of Candidate Contour Points

We first extract a set of candidate contour points with a high recall. Due to its efficiency and high recall performance, we use the SE edge detector [6]. In practice, we could eliminate this step and simply try to predict boundaries at every pixel. However, selecting a set of initial candidate contour points, greatly reduces the computational cost. Since our goal is to build a boundary detector that is both accurate and efficient, we use these candidate points to speed up the computation of our method.

### 3.2. Object-Level Features

After selecting candidate contour points, we up-sample the original input image to a larger dimension (for example  $1100 \times 1100$ ). The up-sampling is done to minimize the loss of information due to the input shrinkage caused by pooling at the different layers. Afterwards, we feed the up-sampled image through 16 convolutional layers of the VGG net.

We use the VGG net as our model because it has been trained to recognize a large number of object classes (the 1000 categories of the ImageNet dataset [24]) and thus encodes object-level features that apply to many classes. To preserve specific location information we utilize only the 16 convolutional layers of the VGG net. We don't use fully connected layers because they don't preserve spatial information, which is crucial for accurate boundary detection.

We visualize some of the selected convolutional maps in Figure 2. Note the high activation values around the various objects in the images, which confirms our hypothesis that the VGG net encodes object specific information in its convolutional feature maps.

### 3.3. Feature Interpolation

Similarly to [26, 12, 19], we perform feature interpolation in deep layers. After the up-sampled image passes through all 16 convolutional layers, for each selected candidate contour point we find its corresponding point in the feature maps. Due to the dimension differences in conve-

lutional maps these correspondences are not exact. Thus we perform feature interpolation by finding the four nearest points and averaging their activation values. This is done in each of the 5504 feature maps. Thus, this results in a 5504-dimensional vector for each candidate point.

We note that the interpolation of convolutional feature maps is the crucial component that enables our system to predict the boundaries efficiently. Without feature interpolation, our method would need to independently process the candidate edge points by analyzing a small image patch around each point, as for example done in DeepEdge [3] which feeds one patch at a time through a deep network. However, when the number of candidate points is large (e.g., DeepEdge considers about 15K points at each of 4 different scales), their patches overlap significantly and thus a large amount of computation is wasted by recalculating filter response values over the same pixels. Instead, we can compute the features for all candidate points with a single pass through the network by performing deep convolution over the *entire* image (i.e., feeding the entire image rather than one patch at a time) and then by interpolating the convolutional feature maps at the location of each candidate edge point so as to produce its feature descriptor. Thanks to this speedup, our method has a runtime of 1.2 seconds (using a K40 GPU), which is better than the runtimes of prior deep-learning based edge detection methods [27, 10, 16, 3].

### 3.4. Learning to Predict Boundaries

After performing feature interpolation, we feed the 5504-dimensional feature vectors corresponding to each of the candidate contour points to two fully connected layers that are optimized to the human agreement criterion. To be more precise, we define our prediction objective as a fraction of human annotators agreeing on the presence of the boundary at a particular pixel. Therefore, a learning objective aims at mimicking the judgement of the human labelers.

Finally, to detect HFL boundaries, we accumulate the predictions from the fully connected layers for each of the

candidate points and produce a boundary probability map as illustrated in Figure 1.

### 3.5. Implementation Details

In this section, we describe the details behind the training procedure of our model. We use the Caffe library [15] to implement our network architecture.

In the training stage, we freeze the weights in all of the convolutional layers. To learn the weights in the two fully connected layers we train our model to optimize the least squares error of the regression criterion that we described in the previous subsection. To enforce regularization we set a dropout rate of 0.5 in the fully connected layers.

Our training dataset includes  $80K$  points from the BSDS500 dataset [22]. As described in the previous subsection, the labels represent the fraction of human annotators agreeing on the boundary presence. We divide the label space into four quartiles, and select an equal number of samples for each quartile to balance the training dataset. In addition to the training dataset, we also sample a hold-out dataset of size 40,000. We use this for the hard-positive mining [21] in order to reduce the number of false-negative predictions.

For the first 25 epochs we train the network on the original 80,000 training samples. After the first 25 epochs, we test the network on the hold-out dataset and detect false negative predictions made by our network. We then augment the original 80,000 training samples with the false negatives and the same number of randomly selected true negatives. For the remaining 25 epochs, we train the network on this augmented dataset.

### 3.6. Boundary Detection Results

In this section, we present our results on the BSDS500 dataset [22], which is the most established benchmark for boundary detection. The quality of the predicted boundaries is evaluated using three standard measures: fixed contour threshold (ODS), per-image best threshold (OIS), and average precision (AP).

We compare our approach to the state-of-the-art based on two different sets of BSDS500 ground truth boundaries. First, we evaluate the accuracy by matching each of the predicted boundary pixels with the ground truth boundaries that were annotated by *any* of the human annotators. This set of ground truth boundaries is referred to as “any”. We present the results for “any” ground truth boundaries in the lower half of Table 2. As indicated by the results, HFL boundaries outperform all the prior methods according to both F-score measures.

Recently, there has been some criticism raised about the procedure for boundary detection evaluation on the BSDS500 dataset. One issue with the BSDS500 dataset involves the so called “orphan” boundaries: the bound-

<i>Consensus GT</i>	<i>ODS</i>	<i>OIS</i>	<i>AP</i>	<i>FPS</i>
SCG [23]	0.6	0.64	0.56	1/280
DeepNet [16]	0.61	0.64	0.61	1/5 <sup>‡</sup>
PMI [14]	0.61	<b>0.68</b>	0.56	1/900
DeepEdge [3]	0.62	0.64	0.64	1/1000 <sup>‡</sup>
$N^4$ -fields [10]	0.64	0.67	0.64	1/6 <sup>‡</sup>
<b>HFL</b>	<b>0.65</b>	<b>0.68</b>	<b>0.67</b>	5/6 <sup>‡</sup>
<i>Any GT</i>	<i>ODS</i>	<i>OIS</i>	<i>AP</i>	<i>FPS</i>
SE [6]	0.75	0.77	0.80	<b>2.5</b>
MCG [2]	0.75	0.78	0.76	1/24
$N^4$ -fields [10]	0.75	0.77	0.78	1/6 <sup>‡</sup>
DeepEdge [3]	0.75	0.77	<b>0.81</b>	1/1000 <sup>‡</sup>
MSC [30]	0.76	0.78	0.79	-
DeepContour [27]	0.76	0.77	0.8	1/30 <sup>‡</sup>
<b>HFL</b>	<b>0.77</b>	<b>0.79</b>	0.8	5/6 <sup>‡</sup>

Table 2: Boundary detection results on BSDS500 benchmark. Upper half of the table illustrates the results for “consensus” ground-truth criterion while the lower half of the table depicts the results for “any” ground-truth criterion. In both cases, our method outperforms all prior methods according to both ODS (optimal dataset scale) and OIS (optimal image scale) metrics. We also report the run-time of our method (<sup>‡</sup> GPU time) in the FPS column (frames per second), which shows that our algorithm is faster than prior approaches based on deep learning [27, 10, 16, 3].

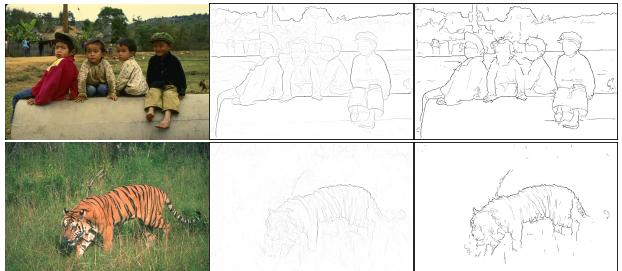


Figure 5: Qualitative results on the BSDS benchmark. The first column of images represent input images. The second column illustrates SE [6], while the third column depicts HFL boundaries. Notice that SE boundaries are predicted with low confidence if there is no significant change in color between the object and the background. Instead, because our model is defined in terms of object-level features, it can predict object boundaries with high confidence even if there is no significant color variation in the scene.

aries that are marked by only one or two human annotators. These “orphan” boundaries comprise around 30% of BSDS500 dataset but most of them are considered uninformative. However, the standard evaluation benchmark rewards the methods that predict these boundaries. To resolve

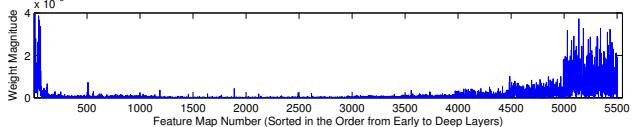


Figure 6: We train a linear regression model and visualize its weight magnitudes in order to understand which features are used most heavily in the boundary prediction (this linear regression is used only for the visualization purposes and not for the accuracy analysis). Note how heavily weighted features lie in the deepest layers of the network, i.e., the layers that are most closely associated with object information.

this issue we also evaluate our HFL boundaries on the so called “consensus” set of ground truth boundaries. These “consensus” boundaries involve only boundaries that are marked by *all* of the human annotators and hence, are considered perceptually meaningful. In the upper half of Table 2, we present the results achieved by our method on the “consensus” set of the ground truth boundaries. Our HFL boundaries outperform or tie all the prior methods in each of the three evaluation metrics, thus suggesting that HFL boundaries are similar to the boundaries that humans annotated. We also report the runtimes in Table 2 and note that our method runs faster than previous deep-learning based edge detection systems [27, 10, 16, 3].

Our proposed model computes a highly nonlinear function of the 5504-dimensional feature vector of each candidate point. Thus, it is difficult to assess which features are used most heavily by our edge predictor. However, we can gain a better insight by replacing the nonlinear function with a simple linear model. In Fig. 6 we show the weight magnitudes of a simple linear regression model (we stress that this linear model is used only for feature visualization purposes). From this Figure, we observe that many important features are located in the deepest layers of the VGG network. As shown in [7], these layers encode high-level object information, which confirms our hypothesis that high-level information is useful for boundary detection.

Finally, we present some qualitative results achieved by our method in Figure 5. These examples illustrate the effective advantage that HFL boundaries provide over another state-of-the-art edge detection system, the SE system [6]. Specifically, observe the parts of the image where there is a boundary that separates an object from the background but where the color change is pretty small. Notice that because the SE boundary detection is based on low-level color and texture features, it captures these boundaries with very low confidence. In comparison, because HFL boundaries rely on object-level features, it detects these boundaries with high confidence.

## 4. High-Level Vision Applications

In this section, we describe our proposed *Low-for-High* pipeline: using low-level boundaries to aid a number of high-level vision tasks. We focus on the tasks of semantic boundary labeling, semantic segmentation and object proposal generation. We show that using HFL boundaries improves the performance of state-of-the-art methods in each of these high-level vision tasks.

### 4.1. Semantic Boundary Labeling

The task of semantic boundary labeling requires not only to predict the boundaries but also to associate a specific object class to each of the boundaries. This implies that given our predicted boundaries we also need to label them with object class information. We approach this problem by adopting the ideas from the recent work on Fully Convolutional Networks (FCN) [19]. Given an input image, we concurrently feed it to our boundary-predicting network (described in Section 3), and also through the FCN that was pretrained for 20 Pascal VOC classes and the background class. While our proposed network produces HFL boundaries, the FCN model predicts class probabilities for each of the pixels. We can then merge the two output maps as follows. For a given boundary point we consider a  $9 \times 9$  grid around that point from each of the 21 FCN object-class probability maps. We calculate the maximum value inside each grid, and then label the boundary at a given pixel with the object-class that corresponds to the maximum probability across these 21 maps. We apply this procedure for each of the boundary points, in order to associate object-class labels to the boundaries. Note that we consider the grids around the boundary pixel because the output of the FCN has a poor localization, and considering the grids rather than individual pixels leads to higher accuracy.

We can also merge HFL boundaries with the state-of-the-art DeepLab-CRF segmentation [5] to obtain higher accuracy. We do this in a similar fashion as just described. First, around a given boundary point we extract a  $9 \times 9$  grid from the DeepLab-CRF segmentation. We then compute the mode value in the grid (excluding the background class), and use the object-class corresponding to the mode value as a label for the given boundary point. We do this for each of the boundary points. By merging HFL boundaries and the output of FCN or DeepLab-CRF, we get semantic boundaries that are highly localized and also contain object-specific information.

#### 4.1.1 Semantic Boundary Labeling Results

In this section, we present semantic boundary labeling results on the SBD dataset [11], which includes ground truth boundaries that are also labeled with one of 20 Pascal VOC classes. The boundary detection accuracy for each class is

Method (Metric)	aero	bike	bird	boat	bottle	bus	car	cat	chair	cow	table	dog	horse	mbike	person	plant	sheep	sofa	train	tv	mean
InvDet (MF)	42.6	49.5	15.7	16.8	36.7	43.0	40.8	22.6	18.1	26.6	10.2	18.0	35.2	29.4	48.2	14.3	26.8	11.2	22.2	32.0	28.0
<b>HFL-FC8 (MF)</b>	71.6	59.6	68.0	54.1	57.2	68.0	58.8	69.3	43.3	65.8	33.3	67.9	67.5	62.2	69.0	43.8	68.5	33.9	57.7	54.8	58.7
<b>HFL-CRF (MF)</b>	<b>73.9</b>	<b>61.4</b>	<b>74.6</b>	<b>57.2</b>	<b>58.8</b>	<b>70.4</b>	<b>61.6</b>	<b>71.9</b>	<b>46.5</b>	<b>72.3</b>	<b>36.2</b>	<b>71.1</b>	<b>73.0</b>	<b>68.1</b>	<b>70.3</b>	<b>44.4</b>	<b>73.2</b>	<b>42.6</b>	<b>62.4</b>	<b>60.1</b>	<b>62.5</b>
InvDet (AP)	38.4	29.6	9.6	9.9	24.2	33.6	31.3	17.3	10.7	16.4	3.7	12.1	28.5	20.4	45.7	7.6	16.1	5.7	14.6	22.7	19.9
<b>HFL-FC8 (AP)</b>	66.0	50.7	58.9	40.6	47.1	62.9	51.0	59.0	25.6	54.6	15.3	57.8	57.3	55.9	62.2	27.5	55.6	18.0	50.1	40.6	47.8
<b>HFL-CRF (AP)</b>	<b>71.2</b>	<b>55.2</b>	<b>69.3</b>	<b>45.7</b>	<b>48.9</b>	<b>71.1</b>	<b>56.8</b>	<b>65.7</b>	<b>29.1</b>	<b>65.9</b>	<b>17.7</b>	<b>64.5</b>	<b>68.3</b>	<b>64.7</b>	<b>65.9</b>	<b>29.1</b>	<b>66.5</b>	<b>25.7</b>	<b>60.0</b>	<b>49.8</b>	<b>54.6</b>

Table 3: Results of semantic boundary labeling on the SBD benchmark using the Max F-score (MF) and Average Precision (AP) metrics. Our method (HFL) outperforms Inverse Detectors [11] for all 20 categories according to both metrics. Note that using the CRF output to label the boundaries produces better results than using the outputs from the FC8 layer of FCN.

evaluated using the maximum F-score (MF), and average precision (AP) measures.

Labeling boundaries with the semantic object information is a novel and still relatively unexplored problem. Therefore, we found only one other approach (Inverse Detectors) that tried to tackle this problem [11]. The basic idea behind Inverse Detectors consists of several steps. First, generic boundaries in the image are detected. Then, a number of object proposal boxes are generated. These two sources of information are then used to construct the features. Finally, a separate classifier is used to label the boundaries with the object-specific information.

Table 3 shows that our approach significantly outperforms Inverse Detectors according to both the maximum F-score and the average precision metrics for all twenty categories. As described in Section 4.1 we evaluate the two variants of our method. Denoted by HFL-FC8 is the variant for which we label HFL boundaries with the outputs from the last layer (FC8) of the pretrained FCN. We denote with HFL-CRF the result of labeling our boundaries with the output from the DeepLab-CRF [5]. Among these two variants, we show that the latter one produces better results. This is expected since the CRF framework enforces spatial coherence in the semantic segments.

In Figure 7, we present some of the qualitative results produced by our method. We note that even with multiple objects in the image, our method successfully recognizes and localizes boundaries of each of the classes.

## 4.2. Semantic Segmentation

For the semantic segmentation task, we propose to enhance the DeepLab-CRF [5] with our predicted HFL boundaries. DeepLab-CRF is a system comprised of a Fully Convolutional Network (described in Section 4.1) and a dense CRF applied on top of FCN predictions.

Specifically, in the CRF, the authors propose to use a Gaussian kernel and a bilateral term including position and color terms as the CRF features (see [5]). While in most cases the proposed scheme works well, DeepLab-CRF sometimes produces segmentations that are not spatially coherent, particularly for images containing small object re-

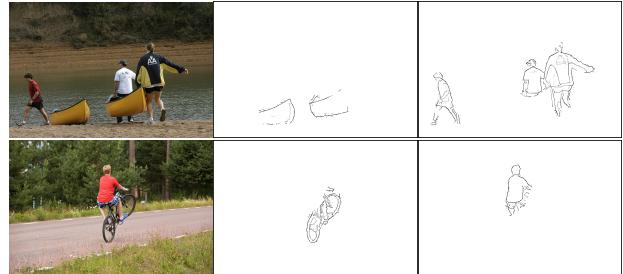


Figure 7: A visualization of the predicted semantic boundary labels. Images in the first column are input examples. Columns two and three show semantic HFL boundaries of different object classes. Note that even with multiple objects appearing simultaneously, our method outputs precise semantic boundaries.

gions.

We propose to address this issue by adding features based on our predicted HFL boundaries in the CRF framework. Note that we use predicted boundaries from Section 3 and not the boundaries labeled with the object information that we obtained in Section 4.1. We use the Normalized Cut [28] framework to generate our features.

First, we construct a pixel-wise affinity matrix  $\mathbf{W}$  using our HFL boundaries. We measure the similarity between two pixels as:

$$W_{ij} = \exp\left(-\max_{p \in \bar{ij}}\left\{\frac{M(p)^2}{\sigma^2}\right\}\right)$$

where  $W_{ij}$  represents the similarity between pixels  $i$  and  $j$ ,  $p$  denotes the boundary point along the line segment  $\bar{ij}$  connecting pixels  $i$  and  $j$ ,  $M$  depicts the magnitude of the boundary at pixel  $p$ , and  $\sigma$  denotes the smoothness parameter, which is usually set to 14% of the maximum boundary value in the image.

The intuitive idea is that two pixels are similar (i.e.  $W_{ij} = 1$ ) if there is no boundary crossing the line connecting these two pixels (i.e.  $M(p) = 0 \quad \forall p \in \bar{ij}$ ) or if the boundary strength is low. We note that it is not necessary to build a full affinity matrix  $\mathbf{W}$ . We build a sparse affin-

Metric	Method (Dataset)	aero	bike	bird	boat	bottle	bus	car	cat	chair	cow	table	dog	horse	mbike	person	plant	sheep	sofa	train	tv	mean
PP-IOU	DL-CRF (VOC)	<b>78.6</b>	41.1	<b>83.5</b>	<b>75.3</b>	72.9	83.1	<b>76.6</b>	<b>80.8</b>	<b>37.8</b>	<b>72.1</b>	66.5	<b>64.7</b>	65.8	<b>75.7</b>	<b>80.5</b>	<b>34.4</b>	<b>75.9</b>	<b>47.4</b>	86.6	<b>77.9</b>	<b>68.9</b>
	<b>DL-CRF+HFL (VOC)</b>	77.9	<b>41.2</b>	83.1	74.4	<b>73.2</b>	<b>85.5</b>	76.1	80.6	35.7	71.0	<b>66.6</b>	64.3	<b>65.9</b>	75.2	80.2	32.8	75.2	47.0	<b>87.1</b>	<b>77.9</b>	68.5
	DL-CRF (SBD)	74.2	68.0	<b>81.9</b>	64.6	<b>71.8</b>	86.3	<b>78.3</b>	<b>84.3</b>	<b>41.6</b>	<b>78.0</b>	49.9	<b>82.0</b>	<b>78.5</b>	77.1	80.1	<b>54.3</b>	<b>75.6</b>	<b>49.8</b>	<b>79.5</b>	70.1	<b>71.4</b>
	<b>DL-CRF+HFL (SBD)</b>	<b>75.1</b>	<b>69.2</b>	81.6	<b>64.8</b>	71.3	<b>86.4</b>	78.1	84.1	41.2	77.8	<b>50.4</b>	81.6	78.2	<b>78.5</b>	<b>80.7</b>	53.8	74.9	49.1	<b>79.5</b>	<b>70.4</b>	<b>71.4</b>
PI-IOU	DL-CRF (VOC)	46.1	<b>28.0</b>	48.5	54.5	45.5	57.6	34.1	<b>47.3</b>	19.5	<b>61.4</b>	<b>41.6</b>	42.5	34.4	61.8	<b>62.1</b>	<b>22.1</b>	50.5	41.0	61.2	31.9	44.6
	<b>DL-CRF+HFL (VOC)</b>	<b>47.5</b>	27.6	<b>50.4</b>	<b>63.5</b>	<b>47.7</b>	<b>57.9</b>	<b>38.7</b>	47.2	<b>21.1</b>	57.3	41.2	<b>43.7</b>	<b>36.0</b>	<b>66.4</b>	61.1	21.3	<b>53.9</b>	<b>42.1</b>	<b>70.9</b>	<b>34.6</b>	<b>46.5</b>
	DL-CRF (SBD)	59.4	36.5	58.0	38.6	32.0	58.1	44.7	59.6	25.8	51.8	28.1	59.0	46.9	50.3	61.8	22.2	45.9	33.4	62.1	41.0	45.8
	<b>DL-CRF+HFL (SBD)</b>	<b>63.4</b>	<b>42.5</b>	<b>58.4</b>	<b>41.3</b>	<b>32.5</b>	<b>61.2</b>	<b>45.7</b>	<b>61.4</b>	<b>28.4</b>	<b>55.5</b>	<b>31.5</b>	<b>61.4</b>	<b>51.8</b>	<b>54.6</b>	<b>62.1</b>	<b>24.9</b>	<b>52.6</b>	<b>34.2</b>	<b>67.1</b>	<b>45.1</b>	<b>48.8</b>

Table 4: Semantic segmentation results on the SBD and VOC 2007 datasets. We measure the results according to PP-IOU (per pixel) and PI-IOU (per image) evaluation metrics. We denote the original DeepLab-CRF system and our proposed modification as DL-CRF and DL-CRF+HFL, respectively. According to the PP-IOU metric, our proposed features (DL-CRF+HFL) yield almost equivalent results as the original DeepLab-CRF system. However, based on PI-IOU metric, our proposed features improve the mean accuracy by 3% and 1.9% on SBD and VOC 2007 datasets respectively.

ity matrix connecting every pair of pixels  $i$  and  $j$  that have distance 5 or less from each other.

After building a boundary-based affinity matrix  $\mathbf{W}$  we set  $D_{ii} = \sum_{i \neq j} W_{ij}$  and compute eigenvectors  $\mathbf{v}$  of the generalized eigenvalue system:

$$(\mathbf{D} - \mathbf{W})\mathbf{v} = \lambda \mathbf{D}\mathbf{v}$$

We then resize the eigenvectors  $v$  to the original image dimensions, and use them as additional features to the CRF part of DeepLab-CRF system. In our experiments, we use the 16 eigenvectors corresponding to the smallest eigenvalues, which results in 16 extra feature channels.

Note that the eigenvectors contain soft segmentation information. Because HFL boundaries predict object-level contours with high confidence, the eigenvectors often capture regions corresponding to objects. We visualize a few selected eigenvectors in Figure 8. In the experimental section, we demonstrate that our proposed features make the output produced by DeepLab-CRF more spatially coherent and improve the segmentation accuracy according to one of the metrics.

We also note that our proposed features are applicable to any generic method that incorporates CRF. For instance, even if DeepLab-CRF used an improved DeepLab network architecture, our features would still be beneficial because they contribute directly to the CRF part and not the DeepLab network part of the system.

#### 4.2.1 Semantic Segmentation Results

In this section, we present semantic segmentation results on the SBD [11] and also Pascal VOC 2007 [8] datasets, which both provide ground truth segmentations for 20 Pascal VOC classes. We evaluate the results in terms of two metrics. The first metric measures the accuracy in terms of pixel intersection-over-union averaged per pixel (PP-IOU) across the 20 classes. According to this metric, the accuracy is computed on a per pixel basis. As a result, the images that contain large object regions are given more importance.

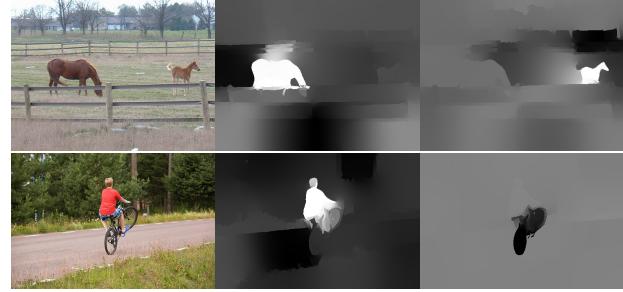


Figure 8: In this figure, the first column depicts an input image while the second and third columns illustrate two selected eigenvectors for that image. The eigenvectors contain soft segmentation information. Because HFL boundaries capture object-level boundaries, the resulting eigenvectors primarily segment regions corresponding to the objects.

We observe that while DeepLab-CRF works well on the images containing large object regions, it produces spatially disjoint outputs for the images with smaller and object regions (see Figure 9). This issue is often being overlooked, because according to the PP-IOU metric, the images with large object regions are given more importance and thus contribute more to the accuracy. However, certain applications may require accurate segmentation of small objects. Therefore, in addition to PP-IOU, we also consider the PI-IOU metric (pixel intersection-over-union averaged per image across the 20 classes), which gives equal weight to each of the images.

For both of the metrics we compare the semantic segmentation results of a pure DeepLab-CRF [5] and also a modification of DeepLab-CRF with our proposed features added to the CRF framework. We present the results for both of the metrics in Table 4.

Based on these results, we observe that according to the first metric (PP-IOU), our proposed features yield almost equivalent results as the original DeepLab-CRF system. However, according to the second metric (PI-IOU) our features yield an average improvement of 3% and 1.9% in

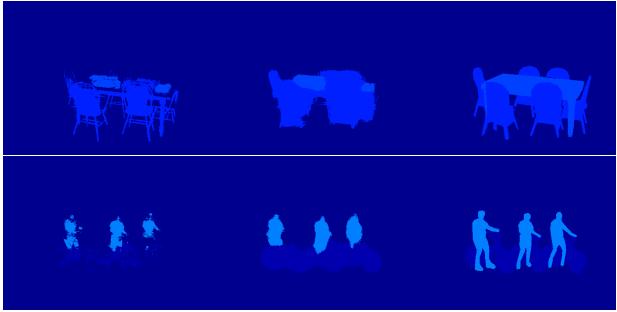


Figure 9: An illustration of the more challenging semantic segmentation examples. The first column depicts the predictions achieved by DeepLab-CRF, while the second column illustrates the results after adding our proposed features to the CRF framework. The last column represents ground truth segmentations. Notice how our proposed features render the predicted semantic segments more spatially coherent and overall more accurate.

SBD and VOC 2007 datasets respectively.

We also visualize the qualitative results produced by both approaches in Figure 9. Notice how our proposed features make the segmentations look smoother relative to the segmentations produced by the original DeepLab-CRF system.

Once again, we want to stress that our HFL features are applicable to any method that uses the CRF. Therefore, based on the results presented in this section, we believe that our proposed features could be beneficial in a wide array of problems that involve the use of the CRF framework.

### 4.3. Object Proposals

Finally, we show that our method produces object-level boundaries that can be successfully exploited in an object proposal scheme. Specifically we adopt the EdgeBoxes approach [31], which can be applied to any generic boundaries to produce a list of object proposal boxes. The original EdgeBoxes method uses SE boundaries to generate the boxes. However, SE boundaries are predicted using low-level color and texture features, rather than object-level features. Thus, here we validate the hypothesis that the EdgeBoxes proposals can be improved by replacing the SE boundaries with our HFL boundaries.

#### 4.3.1 Object Proposal Results

In this section, we present object proposal results on the Pascal VOC 2012 dataset [9]. We evaluate the quality of bounding-box proposals according to three metrics: area under the curve (AUC), the number of proposals needed to reach recall of 75%, and the maximum recall over 5000 object bounding-boxes. Additionally, we compute the accuracy for each of the metrics for three different intersection

Method	IoU 0.65			IoU 0.7			IoU 0.75		
	AUC	N@75%	Recall	AUC	N@75%	Recall	AUC	N@75%	Recall
SE	0.52	413	0.93	0.47	658	0.88	<b>0.41</b>	inf	0.75
<b>HFL</b>	<b>0.53</b>	<b>365</b>	<b>0.95</b>	<b>0.48</b>	<b>583</b>	<b>0.9</b>	<b>0.41</b>	<b>2685</b>	<b>0.77</b>

Table 5: Comparison of object proposal results. We compare the quality of object proposals using Structured Edges [6] and HFL boundaries. We evaluate the performance for three different IOU values and demonstrate that using HFL boundaries produces better results for each evaluation metric and for each IOU value.

over union (IOU) values: 0.65, 0.7, and 0.75. We present these results in Table 5. As described in Section 4.3, we use EdgeBoxes [31], a package that uses generic boundaries, to generate object proposals. We compare the quality of the generated object proposals when using SE boundaries and HFL boundaries. We demonstrate that for each IOU value and for each of the three evaluation metrics, HFL boundaries produce better or equivalent results. This confirms our hypothesis that HFL boundaries can be used effectively for high-level vision tasks such as generating object proposals.

## 5. Conclusions

In this work, we presented an efficient architecture that uses object-level information to predict semantically meaningful boundaries. Most prior edge detection methods rely exclusively on low-level features, such as color or texture, to detect the boundaries. However, perception studies suggest that humans employ object-level reasoning when deciding whether a given pixel is a boundary [13, 25, 17]. Thus, we propose a system that focuses on the semantic object-level cues rather than low level image information to detect the boundaries. For this reason we refer to our boundary detection scheme as a *High-for-Low* approach, where high-level object features inform the low-level boundary detection process. In this paper we demonstrated that our proposed method produces boundaries that accurately separate objects and the background in the image and also achieve higher F-score compared to any prior work.

Additionally, we showed that, because HFL boundaries are based on object-level features, they can be employed to aid a number of high level vision tasks in a *Low-for-High* fashion. We use our boundaries to boost the accuracy of state-of-the-art methods on the high-level vision tasks of semantic boundary labeling, semantic segmentation, and object proposals generation. We show that using HFL boundaries leads to better results in each of these tasks.

To conclude, our boundary detection method is accurate, efficient, applicable to a variety of datasets, and also useful for multiple high-level vision tasks. We plan to release the source code for HFL upon the publication of the paper .

## 6. Acknowledgements

We thank Mohammad Haris Baig for the suggestions and help with the software. This research was funded in part by NSF award CNS-1205521.

## References

- [1] P. Arbelaez, M. Maire, C. Fowlkes, and J. Malik. Contour detection and hierarchical image segmentation. *IEEE Trans. Pattern Anal. Mach. Intell.*, 33(5):898–916, May 2011. [2](#)
- [2] P. Arbelaez, J. Pont-Tuset, J. Barron, F. Marqués, and J. Malik. Multiscale combinatorial grouping. In *Computer Vision and Pattern Recognition (CVPR)*, 2014. [2](#), [4](#)
- [3] G. Bertasius, J. Shi, and L. Torresani. Deepedge: A multi-scale bifurcated deep network for top-down contour detection. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2015. [2](#), [3](#), [4](#), [5](#)
- [4] C. J. C. Burges. A tutorial on support vector machines for pattern recognition. *Data Mining and Knowledge Discovery*, 2:121–167, 1998. [2](#)
- [5] L. Chen, G. Papandreou, I. Kokkinos, K. Murphy, and A. L. Yuille. Semantic image segmentation with deep convolutional nets and fully connected crfs. *CoRR*, abs/1412.7062, 2014. [1](#), [5](#), [6](#), [7](#)
- [6] P. Dollár and C. L. Zitnick. Fast edge detection using structured forests. *PAMI*, 2015. [2](#), [3](#), [4](#), [5](#), [8](#)
- [7] J. Donahue, Y. Jia, O. Vinyals, J. Hoffman, N. Zhang, E. Tzeng, and T. Darrell. Decaf: A deep convolutional activation feature for generic visual recognition. *CoRR*, abs/1310.1531, 2013. [5](#)
- [8] M. Everingham, L. Van Gool, C. K. I. Williams, J. Winn, and A. Zisserman. The PASCAL Visual Object Classes Challenge 2007 (VOC2007) Results. <http://www.pascal-network.org/challenges/VOC/voc2007/workshop/index.html>. [7](#)
- [9] M. Everingham, L. Van Gool, C. K. I. Williams, J. Winn, and A. Zisserman. The PASCAL Visual Object Classes Challenge 2012 (VOC2012) Results. <http://www.pascal-network.org/challenges/VOC/voc2012/workshop/index.html>. [8](#)
- [10] Y. Ganin and V. S. Lempitsky.  $N^4$ -fields: Neural network nearest neighbor fields for image transforms. *ACCV*, 2014. [2](#), [3](#), [4](#), [5](#)
- [11] B. Hariharan, P. Arbelaez, L. Bourdev, S. Maji, and J. Malik. Semantic contours from inverse detectors. In *International Conference on Computer Vision (ICCV)*, 2011. [1](#), [5](#), [6](#), [7](#)
- [12] B. Hariharan, P. A. Arbeláez, R. B. Girshick, and J. Malik. Hypercolumns for object segmentation and fine-grained localization. *CoRR*, abs/1411.5752, 2014. [3](#)
- [13] P.-J. Hsieh, E. Vul, and N. Kanwisher. Recognition alters the spatial pattern of fmri activation in early retinotopic cortex. *Journal of Neurophysiology*, 103(3):1501–1507, 2010. [1](#), [8](#)
- [14] P. Isola, D. Zoran, D. Krishnan, and E. H. Adelson. Crisp boundary detection using pointwise mutual information. In *ECCV*, 2014. [2](#), [4](#)
- [15] Y. Jia, E. Shelhamer, J. Donahue, S. Karayev, J. Long, R. Girshick, S. Guadarrama, and T. Darrell. Caffe: Convolutional architecture for fast feature embedding. *arXiv preprint arXiv:1408.5093*, 2014. [4](#)
- [16] J. J. Kivinen, C. K. Williams, N. Heess, and D. Technologies. Visual boundary prediction: A deep neural prediction network and quality dissection. *AISTATS*, 1(2):9, 2014. [2](#), [3](#), [4](#), [5](#)
- [17] Z. Kourtzi and N. Kanwisher. Representation of perceived object shape by the human lateral occipital complex. *Science*, 293:1506–1509, 2001. [1](#), [8](#)
- [18] J. Lim, C. L. Zitnick, and P. Dollár. Sketch tokens: A learned mid-level representation for contour and object detection. In *CVPR*, 2013. [2](#)
- [19] J. Long, E. Shelhamer, and T. Darrell. Fully convolutional networks for semantic segmentation. *CVPR (to appear)*, Nov. 2015. [3](#), [5](#)
- [20] M. Maire, S. X. Yu, and P. Perona. Reconstructive sparse code transfer for contour detection and semantic labeling. In *Asian Conference on Computer Vision (ACCV)*, 2014. [2](#)
- [21] T. Malisiewicz, A. Gupta, and A. A. Efros. Ensemble of exemplar-svms for object detection and beyond. In *ICCV*, 2011. [4](#)
- [22] 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. 8th Int'l Conf. Computer Vision*, volume 2, pages 416–423, July 2001. [4](#)
- [23] X. Ren and L. Bo. Discriminatively Trained Sparse Code Gradients for Contour Detection. In *Advances in Neural Information Processing Systems*, December 2012. [2](#), [4](#)
- [24] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, A. C. Berg, and L. Fei-Fei. ImageNet Large Scale Visual Recognition Challenge, 2014. [3](#)
- [25] J. L. Sanguinetti, J. J. Allen, and M. A. Peterson. The ground side of an object perceived as shapeless yet processed for semantics. *Psychological science*, page 0956797613502814, 2013. [1](#), [8](#)
- [26] P. Sermanet, D. Eigen, X. Zhang, M. Mathieu, R. Fergus, and Y. LeCun. Overfeat: Integrated recognition, localization and detection using convolutional networks. *CoRR*, abs/1312.6229, 2013. [3](#)
- [27] W. Shen, X. Wang, Y. Wang, X. Bai, and Z. Zhang. Deep-contour: A deep convolutional feature learned by positive-sharing loss for contour detection. June 2015. [1](#), [3](#), [4](#), [5](#)
- [28] J. Shi and J. Malik. Normalized cuts and image segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22:888–905, 1997. [2](#), [6](#)
- [29] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. *CoRR*, abs/1409.1556, 2014. [1](#)
- [30] A. Sironi, V. Lepetit, and P. Fua. Multiscale centerline detection by learning a scale-space distance transform. June 2014. [4](#)
- [31] C. L. Zitnick and P. Dollár. Edge boxes: Locating object proposals from edges. In *ECCV*, 2014. [1](#), [8](#)