Convolutional Oriented Boundaries: Perceptual Grouping beyond the BSDS

Anonymous ECCV submission

Paper ID 179

Abstract. This paper brings the potential of deep Convolutional Neural Network (CNN) architectures to the field of perceptual grouping. We present Convolutional Oriented Boundaries (COB), a technique that produces multiscale oriented contours and region hierarchies starting from generic image classification CNNs. COB is computationally efficient, because it requires a single CNN forward pass for contour detection and uses a novel sparse boundary representation for hierarchical segmentation; it gives a significant leap in performance over the state-of-the-art, and it generalizes very well to unseen categories and datasets. Particularly, we show that learning to estimate not only contour strength but also contour orientation provides more accurate results. We also conclude that our architectures do not require contour globalization, which was one of the speed bottlenecks in existing approaches. We perform an extensive experimental validation on BSDS, PASCAL Context, PASCAL Segmentation, and MS-COCO, showing that COB provides state-of-the-art oriented contours, region hierarchies, and object proposals.

Keywords: Contour detection, contour orientation estimation, hierarchical segmentation, object proposals

1 Introduction

For the last fifteen years, the Berkeley Segmentation Dataset and Benchmark (BSDS) [1] has been the experimental testbed of choice for the study of boundary detection and image segmentation. The existence of that empirical framework, whose significance to the computer vision community was acknowledged by a Helmholtz award at ICCV 2015, has allowed quantifiable progress in the field of perceptual grouping. The formulation of boundary detection as a supervised binary pixel classification task, proposed by Martin et al. [?], marked the beginning of the current era, in which progress in machine learning has continuously permeated to the tasks of boundary detection and generic image segmentation. Among the milestones in grouping in the last decade, one can cite the application of boosting [2], of sparse coding [3–5], of SIFT features [6], and of correlation clustering [7], the effective use of spectral graph partitioning [8, 9], the joint study of contours and hierarchical regions [9], the design of classifier cascades [10], the use of structured forests [11], of combinatorial grouping [12], and of planar ultrametrics [13].

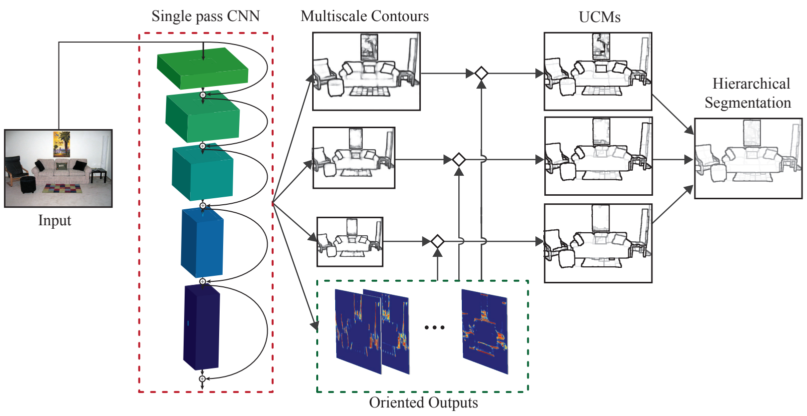


Fig. 1. Overview of our approach: From a single pass of a base CNN, we obtain multiscale oriented contours. We combine them to build ultrametric contour maps (UCMs) at different scales and in turn they are combined into a single hierarchical segmentation structure.

Recently, the community-wide adoption of Convolutional Neural Networks (CNNs) has caused a profound change in the study of perceptual grouping and a large leap forward in performance, as it has fostered the appearance of systems [14–19] relying on large-scale category-specific information in the form of deep architectures pre-trained on Imagenet for image classification [20, 21] into a traditionally category-agnostic field. However, larger capacity and more accurate models have also underlined the limitations of the BSDS as the primary benchmark for grouping. Its 300 train images are inadequate for training systems with tens of millions of parameters and, critically, current state-of-the-art techniques are reaching human performance for boundary detection on its 200 test images.

The next natural frontier for perceptual grouping is the PASCAL VOC dataset [22], an influential benchmark for image classification, object detection and semantic segmentation which has a trainval set with more than 10 000 challenging and varied images. A first step in that direction was taken by Hariharan et al. [23], who annotated the VOC dataset for category-specific boundary detection on the foreground objects. More recently, the Pascal Context dataset [24] extended this annotation effort to all the background categories, providing thus fully-parsed images which are a direct VOC counterpart to the human ground-truth of the BSDS.

In this paper, we investigate the transition from the BSDS to PASCAL in the study of perceptual grouping. For this purpose, we develop a generic CNN architecture, which allows end-to-end learning of multiscale oriented contours, and we show how it translates top performing input networks into high-quality contours. We then propose a sparse boundary representation for efficient construction of hierarchical regions from our contour signal. Our overall approach is both efficient and highly accurate, and produces state-of-the-art contours and regions on PASCAL and on the BSDS. Fig. 1 shows an overview of our system.

We derive valuable insights from studying perceptual grouping in a larger and more challenging empirical framework. Among them, we observe that our system leverages increasingly deeper state-of-the-art architectures, such as the recent Residual Networks [21], to produce improved results. This indicates that our approach is generic and can directly benefit from future advances in CNNs. We also observe that, in PASCAL, the globalization strategy of contour strength by spectral graph partitioning proposed in [9] and used in state-of-the-art methods [12, 14] is unnecessary in the presence of the high-level knowledge conveyed by pre-trained CNNs, removing thus a significant computational bottleneck for high-quality contours.

We conduct comprehensive experiments demonstrating the interest of our technique for downstream recognition applications. We use our hierarchical regions as input to the combinatorial grouping algorithm of [12] and obtain state-of-the-art segmented object proposals on PASCAL. Furthermore, we provide empirical evidence for the generalization power of our approach by evaluating our object proposals without any retraining in the even larger and more challenging MS-COCO dataset, where we report also a large improvement in performance with respect to the state-of-the-art.

All our code, pre-trained results, and benchmark environment will be made publicly available.

2 Related Work

The latest wave of contour detectors takes advantage of deep learning to obtain state of the art results [14, 15, 17, 16, 18, 19]. Ganin and Lempitsky [19] use a deep architecture to extract features of image patches. They approach contour detection as a classification task, and in a second step they match the features to their nearest neighbor among predefined ground-truth features. The authors of [16, 17] make use of features generated by pre-trained CNNs to regress contours. They prove that object-level information provides powerful cues for the prediction of contours. Shen et al. [18] learn deep features using shape information. Xie and Tu [15] provide an end-to-end deep framework to boost the efficiency and accuracy of contour detection, using convolutional feature maps and a novel loss function. Kokkinos [14] builds upon [15] and improves the results by tuning the loss function, running the detector at multiple scales, and adding globalization. Our approach is different from previous work in that we obtain multiscale information in a single pass of the network, we combine the per-pixel classification with contour orientation estimation, our output is richer than a linear combination of cues at different scales and is merged at the segmentation stage.

At the core of all these deep learning approaches, lies a *base network architecture*, starting from the seminal AlexNet [25] (8 layers), to the more complex VGGNet [20] (16 layers), to the inception architecture of GoogLeNet [26] (22 layers), to the very recent and very deep ResNets [21] (50, 101 and 152 layers). Image classification results, which originally motivated these architectures, have been continuously improved by exploring deeper and more complex networks. In this work, we present results both using VGGNet and ResNet, showing that our method is modular and can incorporate and benefit from future improvements in the base network architectures.

Our approach exploits the duality between contour detection and segment extraction, initially studied by Najman and Schmitt [27]. Arbeláez et al. [9] showed its usefulnes for jointly optimizing contours and regions, and later proved its interest also for generating object proposals [12]. We differentiate from these approaches in two aspects. First, our sparse boundary representation translates into a clean and highly efficient implementation of hierarhcical segmentation. Second, by leveraging high-level knowledge from the CNNs in the estimation of contour strength and orientation, our method benefits naturally from global information, which allows bypassing the normalized cuts step, a bottleneck in terms of computational cost, but a cornerstone for their good results.

3 Deep Multiscale Oriented Contours

CNNs are by construction multi-scale feature extractors. If one examines the standard architecture of a CNN consisting of convolutional and spatial pooling layers, it becomes clear that as we move deeper, feature maps capture more global information due to the decrease in scale. For contour detection, this architecture implies local and fine-scale contours at shallow levels, coarser spatial resolution and larger receptive fields for the units when going deeper into the network and, consequently, more global information for predicting boundary strength and orientation. CNNs have therefore a built-in globalization strategy for contour detection, analogous to the hand-engineered globalization of contour strength through spectral graph partitioning in [9, 12].

In this section, we present our approach to multiscale oriented contour detection. Figure 2 depicts how we make use of information provided by the intermediate layers of a CNN to detect contours and their orientations at multiple scales. Different groups of feature maps contain different, scale-specific information, which we combine to build a multiscale oriented contour detector.

The remainder of this section is devoted to introducing the recent approaches to contour detection using deep learning (Section 3.1), to presenting our CNN architecture to produce contour detection at different scales (Section 3.2), and to explaining how we estimate the orientation of the edges (Section 3.3; all in a single CNN forward pass).

3.1 Training deep contour detectors

The recent success of [15] is based on a CNN to accurately regress the contours of an image. Within this framework, the idea of employing a neural network in an image-to-image fashion without any post-processing has proven successful and serves right now as the state-of-the-art for the task of contour detection. Their network, HED, produces scale-specific contour images (side outputs) for different scales of a network, and combines their activations linearly to produce

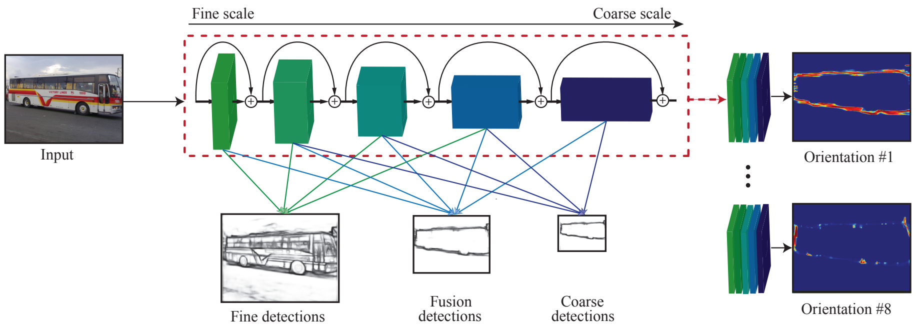


Fig. 2. Our deep learning architecture (best viewed in color). The connections show the different stages that are used to generate the multiscale contours. Orientations further require additional convolutional layers in multiple stages of the network

a contour probability map. Using the notation of the authors, we denote the training dataset by $S = \{(X_n, Y_n), n = 1, ..., N\}$, with X_n being the input image and $Y_n = \{y_j^{(n)}, j = 1, ..., |X_n|\}, y_j^{(n)} \in \{0, 1\}$ the predicted pixelwise labels. For simplicity, we drop the subscript n. Each of the M side outputs minimizes the objective function:

$$\ell_{side}^{(m)}\left(\mathbf{W},\mathbf{w}^{(m)}\right) = -\beta \sum_{j \in Y_+} \log P\left(y_j = 1 | X; \mathbf{W}, \mathbf{w}^{(m)}\right) - (1-\beta) \sum_{j \in Y_-} \log P\left(y_j = 0 | X; \mathbf{W}, \mathbf{w}^{(m)}\right) \quad (1)$$

where $\ell_{side}^{(m)}$ is the loss function for scale $m \in \{1, ..., M\}$, $\mathbf{W}$ denotes the standard set of parameters of the CNN, and $\{\mathbf{w}^{(m)}, m = 1, ..., M\}$ the corresponding weights of the the m-th side output. The multiplier β is used to handle the imbalance of the substantially greater number of background compared to contour pixels. Y_+ and Y_- denote the contour and background sets of the groundtruth Y, respectively. The probability $P(.)$ is obtained by applying a sigmoid $\sigma(.)$ to the activations of the side outputs $\hat{A}_{side}^{(m)} = \{a_j^{(m)}, j = 1, ..., |Y|\}$. The activations are finally fused linearly, as: $\hat{Y}_{fuse} = \sigma\left(\Sigma_{m=1}^{M} h_m \hat{A}_{side}^{(m)}\right)$ where $\mathbf{h} = \{h_m, m = 1, ..., M\}$ are the fusion weights. The fusion output is also trained to resemble the ground-truth applying the same loss function of Eq. 1, by optimizing the complete set of parameters, including the fusion weights $\mathbf{h}$. We denote the fusion loss function as $\mathcal{L}_{fuse}(\mathbf{W}, \mathbf{w}, \mathbf{h})$. The contour detector optimizes the overall objective function:

$$\mathcal{L}(\mathbf{W}, \mathbf{w}, \mathbf{h}) = \sum_{m=1}^{M} \ell_{side}^{(m)}\left(\mathbf{W}, \mathbf{w}^{(m)}\right) + \mathcal{L}_{fuse}(\mathbf{W}, \mathbf{w}, \mathbf{h}) \quad (2)$$

In the rest of the paper, whenever we train a part of our CNN architecture, we use the class balancing cross entropy loss function of Equation 1.

3.2 Multiscale contours

We finetune ResNet [21] for the task of contour detection. The fully connected layers used for classification are removed, and so are the batch normalization layers, since we operate on one image per iteration. So the network consists mainly of convolutional layers coupled with ReLU activations, divided into 5 stages. We will refer at this architecture as the "base network" of our implementation. Each stage is handled as a different scale, since it contains feature maps of a similar size. At the end of a stage, there is a max pooling layer, which reduces the dimensions of the produced feature maps to half. Therefore, the CNN contains multiscale information which we exploit to build a multiscale contour regressor.

We separately supervise the last layer of each stage, so that each scale produces a contour map. We associate each contour map with a loss function of Eq. 1. The idea of supervising intermediate parts of a CNN has successfully been used in previous approaches, for a variety of tasks [26, 28, 15]. In the 5-scale base network illustrated in Fig. 2, we append 3 convolutional layers. The first one acts on the 4 finest scales, combining their side activations to a finer scale output, $\hat{Y}_{fine}$. The second layer combines all 5 side activations into $\hat{Y}_{all}$, and the third one acts on the 4 coarsest scales, producing $\hat{Y}_{coarse}$. The three combinations lead to scale-specific detections. The finer scale contains better localized contours, whereas the coarse scale leads to less noisy detections. An example of the scale-specific outputs are illustrated in Fig.2. At training time, we freeze the weights of the base network, and we only train the three additional convolutional layers needed for merging the supervised output activations.

Contour detection at different scales is key for how a grouping algorithm like [12] combines cues from different scales into accurate segmentations. The intuition behind our framework is that a grouping algorithm can perform better than combining the different scales linearly.

3.3 Estimation of Contour Orientations

In order to predict accurate contour orientations, we propose an extension of the CNN that we use as multiscale contour detector. We define the task as pixel-wise image-to-image multiscale classification into K bins. We connect K different branches (sub-networks) to the base network. Each of the branches is associated with one orientation bin, and has access to feature maps that are generated from the intermediate convolutional layers at M different scales. We assign the parts of the CNN associated with each orientation a different task than the base network: classify the pixels of the contours that match a specific orientation. In order to design these orientation specific subtasks, we further divide the human annotations of the contours into K classes, depending on the orientation. Such division is obtained by approximating the ground-truth boundaries with lines, and classifying their orientation into K bins. The parts of a contour between the sections that are approximated by a line, get assigned the orientation of the line. Figure 5 illustrates this concept. As in the case of multiscale contours, the weights of the base network remain frozen, meaning that for the layers associated with the contour detector the learning rate is set to zero.

Each sub-network consists of M convolutional layers, each of them appended on different scales of the base network. Thus we need $M * K$ additional layers, namely `conv_scale_m_orient_k`, with `k`$= 1, ..., K$ and `m`$= 1, ..., M$. In our setup, we use $K = 8$ and $M = 5$. All K orientations are regressed in parallel, and since they are associated with a certain angle, we post-process them to obtain the orientation map. Specifically, the orientation map is obtained as:

$$O(x, y) = \mathcal{T}\left(arg \max_k B_k(x, y)\right), k = 1, ..., K \tag{3}$$

where $B_k(x, y)$ denotes the response of the n-th orientation bin of the CNN at the pixels with coordinates (x, y) and $\mathcal{T}(.)$ is the transformation function which associates each bin with its central angle. For the cases where two neighboring bins lead to strong responses, we compute the angle as their weighted average. At pixels where there is no response for any of the orientations, we assign random values between 0 and π. The different orientations as well as the resulting orientation map (colorcoded) are illustrated in Fig. 3. In [9, 11, 12] the orientations are computed by applying local gradient filters and obtaining the angles from their responses. In Section 5 we show that substituting the orientation map obtained by gradient filters with our trained orientation map leads to more accurate region segmentations.

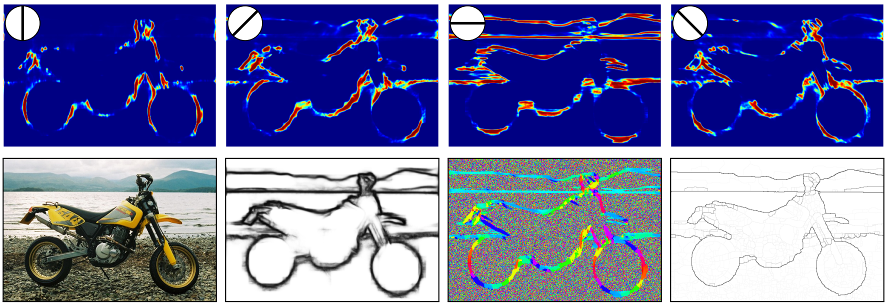


Fig. 3. Illustration of contour orientation learning. Row 1 shows the responses B_n for four out of the eight orientation bins. Pixels with different contour orientations show high responses in the respective bins. Row 2, from left to right: original image, contour strength, learned orientation map into 8 orientations, and hierarchical boundaries.

4 Fast Hierarchical Regions

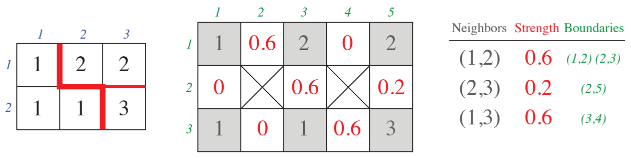
This section is devoted to building an efficient hierarchical image segmentation algorithm from the multiscale contours and the orientations extracted in the previous section. We build on the concept of Ultrametric Contour Map (UCM) [9], which transforms a contour detection probability map into a hierarchical boundary map. Thus, we get partitions at different granularities when thresholding at various strength values.

Despite the success of UCMs, their low efficiency significantly limits their applicability, particularly in multi-scale settings. Section 4.1 presents an alternative representation of an image partition that allows us to reduce the computation time of multi-scale UCMs by an order of magnitude, to less than one second. Then, Section 4.2 presents the algorithm to build a hierarchy of regions from the multiscale contours and the orientations presented in Section 3. As we will show in the experimental section, the resulting algorithm improves the state of the art significantly, at a fraction of the computational time of [12].

4.1 Boundary-Based Representation of Hierarchies of Regions

An image partition is a clustering of the set of pixels into different sets, which we call regions. The most straightforward way of representing it in a computer is by a matrix of labels, as in the example in Figure 4(a), with three regions on an image of size 2×3. The boundaries of this partition are the edge elements, or *edgels*, between the pixels with different labels (highlighted in red).

We can assign different *strengths* to these boundaries (thicknesses of the red lines in Figure 4(a)), which indicate the *confidence* of that piece of being a true boundary. By iteratively *erasing* these boundaries in order of increasing strength we obtain different partitions, which we call *hierarchy of regions*, or Ultrametric Contour Maps.



(a) Pixel labeling (b) Boundary grid (c) Sparse boundaries

Fig. 4. Image Partition Representation: (a) Pixel labeling, each pixel gets assigned a region label. (b) Boundary grid, binary markers of the boundary positions. (c) Sparse boundaries, lists of boundary coordinates between neighboring regions.

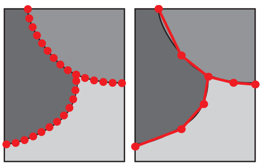


Fig. 5. Polygon simplification: From all boundary points (left) to simplified polygons (right).

These boundaries are usually stored in the *boundary grid* (Figure 4(b)), a matrix of double the size of the image (minus one), in which the odd coordinates represent pixels (gray areas), and the positions in between represent boundaries (red numbers) and junctions (crossed positions).

UCMs use this representation to store their boundary *strength* values, that is, each boundary position stores the threshold value beyond which that edgel *disappears* and the two neighboring regions merge. This way, simply *binarizing* a UCM we have a partition represented as a boundary grid.

This representation, while useful during prototyping, becomes very inefficient at run time, where the percentage of *activated* boundaries is very sparse. Not only are we wasting memory by storing those *empty* boundaries, but it also makes operating on them very inefficient by having to *sweep* all the matrix to perform a modification on a single boundary piece.

Inspired by how sparse matrices are handled, we designed the *sparse boundaries* representation (Figure 4(c)). It stores a look-up table for pairs of neigboring regions, their boundary strength, and the list of coordinates the boundary occupies. Apart from being more compact in terms of memory, this representation enables efficient operations on specific pieces of a boundary, since one only needs to perform a search in the look-up table and scan the activated coordinates; instead of having to sweep the whole boundary grid.

4.2 Fast Hierarchies from Multiscale Oriented Contours

The deep CNN presented in Section 3 provides different levels of detail for the image contours. A linear combination of the layers would be the straightforward way of providing a single contour signal (as is done in [15]).

The approach in this work is to combine the region hierarchies extracted from the contour signals at each layer instead of the contours directly. We were inspired by the framework proposed in [12], in which a UCM is obtained from contours computed at different image scales and then combined in a single hierarchy; but instead we use the different contour outputs that are computed in a single pass of the proposed CNN architecture.

The drawback of this framework is that the manipulation of the hierarchies is very slow (in the order of seconds). In [12], the operations on the UCMs and the Oriented Watershed Transform (OWT) therefore had to be discretized and performed at a low number of contour strengths. By using the fast sparse boundary representation of the previous section, we can afford to operate on all contour strengths, yielding better results at a fraction of the original cost. Moreover, we use the learned contour orientations for the computation of the OWT, instead of the local gradient estimations, further boosting the performance.

5 Experiments

This section presents the empirical evidence that supports our approach. First, Section 5.1 explores the ablated and baseline techniques studied to isolate and quantify the improvements due to different components of our system. Then Section 5.2, Section 5.3, and Section 5.4 compare our results against the state-of-the-art in terms of contour orientation estimation, generic image segmentation, and the application to object proposals, respectively. In all three cases, we obtain the best results to date by a significant margin. Finally, Section 5.5 analyzes the gain in speed by the use of our sparse boundaries representation.

We extend the main BSDS benchmarks to the PASCAL Context dataset [24], which contains carefully localized pixelwise semantic annotations for the entire image on the PASCAL VOC 2010 detection trainval set. This results in 459 semantic categories across 10 103 images, which is an order of magnitude (20×) larger than the BSDS. In order to allow training and optimization of large capacity models, we split the data into train, validation, and test sets as follows: *VOC train* corresponds to the official PASCAL train with 4 998 images, *VOC* *val* corresponds to half the official PASCAL validation set with 2 607 images and *VOC test* corresponds to the second half with 2 498 images. In the remainder of the paper, we refer to this dataset division. Note that, in this setting, the notion of boundary is defined as separation between objects of different semantic categories and does not consider part annotations, in contrast to the BSDS.

We used the publicly available *Caffe* [29] framework for training and testing CNNs, and all the state-of-the-art results are computed using the publicly-available code provided by the respective authors. We will make all the source code of our approach, dataset splits, pre-computed results, and benchmarking code publicly available.

5.1 Control Experiments/Ablation Analysis

This section presents the control experiments and ablation analysis to assess the performance of all subsystems of our method. We train on *VOC train*, and evaluate on *VOC val* set. We report the standard F-measure at Optimal Dataset Scale (ODS) and Optimal Image Scale (OIS), as well as the Average Precision (AP), both evaluating boundaries (F_b [30]) and regions (F_{op} [31]).

Table 1 shows the evaluation results of the different variants, highlighting whether we include globalization and/or trained orientations. As a first baseline, we test the performance of MCG [12], which uses Structured Edges [11] as input contour signal, and denote it MCG (SE). We then substitute SE by the newer HED [15], trained on *VOC train* as input contours and denote it MCG (HED). Note that the aforementioned baselines require multiple passes of the contour detector (3 different scales).

In the direction of using the side outputs of the base CNN architecture as multiscale contour detections in one pass, we tested the baseline of naively taking the 5 side outputs directly as the contour detections. We trained both VGGNet [20] and ResNet50 [21] on *VOC train* and combined the 5 side outputs with our fast hierarchical regions of Section 4 (VGGNet-Side and ResNet50-Side).

We finally evaluate different variants of our system, as presented in Section 3. We first compare our system with two different base architectures: Ours(VGGNet) and Ours(ResNet50). We train the base networks for 30000 iterations, with stochastic gradient descent and a momentum of 0.9. We observe that the deeper architecture translates into significantly better boundaries and regions.

We then evaluate the influence of our trained orientations and globalization, by testing the four possible combinations (the orientations are further evaluated in next section). Our method using ResNet50 together with trained orientations leads to the best results both for boundaries and for regions. The experiments also show that, when coupled with trained orientations, globalization even decreases performance, so we can safely remove it and get a significant speed up. Our technique with trained orientations and without globalization is therefore selected as our final system and will be referred to in the sequel as Convolutional Oriented Boundaries (COB). We plot the precision recall evaluation for all the different variants of our technique in the supplemental material.

Method	Global.	Orient.	Boundaries - F_b ODS	OIS	AP	Regions - F_{op} ODS	OIS	AP
MCG (SE)	✓	✗	0.573	0.630	0.572	0.355	0.419	0.264
MCG (HED)	✓	✗	0.699	0.735	0.705	0.459	0.520	0.374
VGGNet-Side	✓	✗	0.659	0.710	0.688	0.439	0.505	0.351
ResNet50-Side	✓	✗	0.692	0.735	0.711	0.456	0.521	0.374
Ours (VGGNet)	✗	✓	0.717	0.754	0.753	0.466	0.533	0.384
Ours (ResNet50)	✗	✗	0.735	0.774	0.757	0.475	0.545	0.405
Ours (ResNet50)	✓	✗	0.727	0.764	0.742	0.461	0.531	0.395
Ours (ResNet50)	✓	✓	0.736	0.776	0.773	0.481	**0.554**	**0.418**
Ours (ResNet50)	✗	✓	**0.740**	**0.779**	**0.780**	**0.483**	0.553	0.417

Table 1. Ablation analysis on *VOC val*: Comparison of different ablated versions of our system.

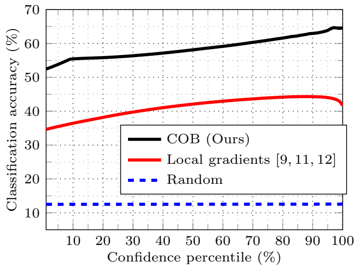


Fig. 6. Contour orientation: Classification accuracy into orientations quantized in 8 bins.

5.2 Contour Orientation

We evaluate contour orientation results by the classification accuracy into 8 different orientations, to isolate their performance from the global system. We compute the ground truth orientations as depicted in Figure 5 by means of the sparse boundaries representation. We then sweep all ground-truth boundary pixels and compare the estimated orientation with the ground-truth one. Since the orientations are not well-balanced classes (much more horizontal and vertical contours), we compute the classification accuracy per each of the 8 classes and then compute the mean.

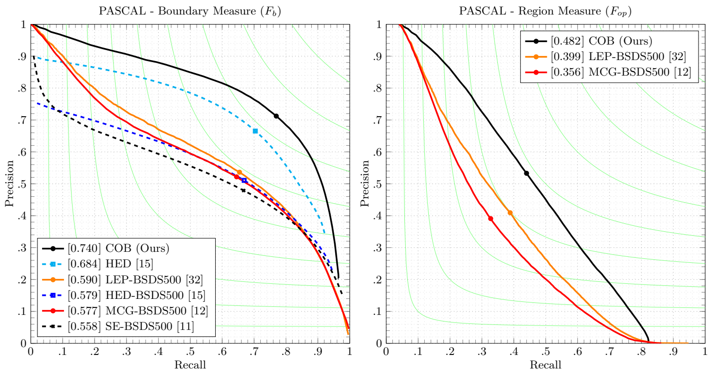
Figure 6 shows the classification accuracy with respect to the confidence of the estimation. We compare our proposed technique against the local gradient estimation used in previous literature [9, 11, 12]. As a baseline, we plot the result a random guess of the orientations would get. We observe that our estimation is significantly better than the previous approach. As a summary measure, we compute the area under the curve of the accuracy (ours 58.6%, local gradients 41.2%, random 12.5%), which corroborates the superior results from our technique.

5.3 Generic Image Segmentation

We present our results for contour detection and generic image segmentation on Pascal Context [24] as well as on the BSDS500 [1], which is the most established benchmark for perceptual grouping.

PASCAL Context: We train COB in the *VOC train*, and perform parameter selection on *VOC val*. We report the final results on the unseen *VOC test*, using the previously tuned parameters. Figure 7 shows the evaluation of our method compared to the state-of-the-art. We compare our approach to several methods trained on the BSDS [11, 12, 32, 15] and we also retrain the current state-of-the-art contour detection method HED [15] on *VOC train*. Results show that our method outperforms all others by a considerable margin both in terms of boundaries and in terms of regions. The lower performance of methods trained on the BSDS database quantifies the difficulty of the task when moving to a larger and more challenging dataset.

In the supplemental material we compare COB to the pre-computed evaluation of [33], as provided by the authors in the dataset split of that work. We also present some qualitative results of our technique.

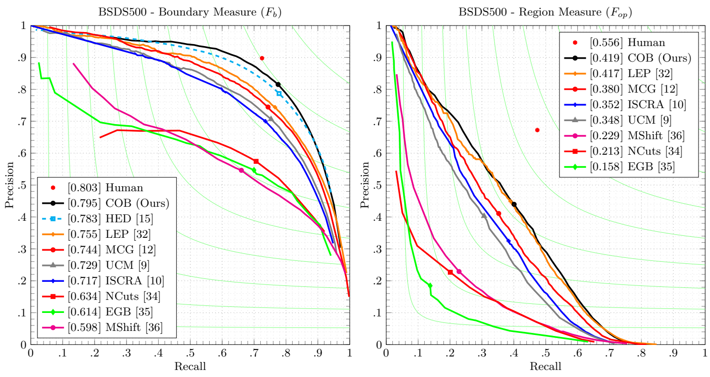


Method	Boundaries - F_b ODS	OIS	AP
COB	**0.740**	**0.781**	**0.780**
HED [15]	0.684	0.702	0.678
HED-B [15]	0.579	0.605	0.532
LEP-B [32]	0.590	0.656	0.597
MCG-B [12]	0.577	0.634	0.578
SE-B [11]	0.558	0.583	0.525

Method	Regions - F_{op} ODS	OIS	AP
COB	**0.482**	**0.557**	**0.428**
LEP-B [32]	0.399	0.461	0.304
MCG-B [12]	0.356	0.421	0.265

Fig. 7. PASCAL *VOC Test* Evaluation: Precision-recall curves for evaluation of boundaries (F_b [30]), and regions (F_{op} [31]). Contours in dashed lines and boundaries (from segmentation) in solid lines. ODS, OIS, and AP summary measures.

BSDS500: We retrained COB using only the 300 images of the *trainval* set of the BSDS, after data augmentation as suggested in [15], keeping the architecture decided in Section 5.1. For comparison to HED [15], we used the model that the authors provide online. Figure 8 presents the evaluation results, which show that we also obtain state-of-the-art results in this dataset. The smaller margins are in all likelihood due to the fact that we almost reach human performance for the task of contour detection on the BSDS, which motivates the shift to PASCAL to achieve further progress in the field.



Method	Boundaries - F_b ODS	OIS	AP
COB (Ours)	**0.795**	**0.820**	**0.857**
HED [15]	0.783	0.798	0.821
LEP [32]	0.755	0.790	0.821
MCG [12]	0.744	0.777	0.760
UCM [9]	0.729	0.755	0.745
ISCRA [10]	0.717	0.752	0.770
NCuts [34]	0.634	0.664	0.422
EGB [35]	0.614	0.658	0.564
MShift [36]	0.598	0.645	0.497

Method	Regions - F_{op} ODS	OIS	AP
COB (Ours)	**0.419**	**0.478**	**0.343**
LEP [32]	0.417	0.468	0.334
MCG [12]	0.380	0.433	0.271
ISCRA [10]	0.352	0.418	0.275
UCM [9]	0.348	0.385	0.235
MShift [36]	0.229	0.292	0.122
NCuts [34]	0.213	0.270	0.096
EGB [35]	0.158	0.240	0.080

Fig. 8. BSDS500 Test Evaluation: Precision-recall curves for evaluation of boundaries (F_b [30]), and regions (F_{op} [31]). ODS, OIS, and AP summary measures.

5.4 Object Proposals

Object proposals are an integral part of current object detection and semantic segmentation pipelines [37–39], as they provide a reduced search space on locations, scales, and shapes over the image. This section evaluates COB as a segmented proposal technique, when using our high-quality region hierarchies in conjunction with the combinatorial grouping framework of [12]. We compare against the more recent techniques POISE [40], MCG and SCG [12], LPO [41], GOP [42], SeSe [43], GLS [44], and RIGOR [45]. Recent thorough comparisons of object proposal generation methods can be found in [46, 47].

We perform experiments on the PASCAL 2012 segmentation dataset [22] and on the bigger and more challenging MS-COCO [48] (val set). The hierarchies and combinatorial grouping are trained on PASCAL. To assess the generalization capability, we evaluate on MS-COCO, which contains a large number of previously unseen categories, without further retraining.

Figure 9 shows the average recall [46] with respect to the number of object proposals. In PASCAL, the absolute gap of improvement of COB is at least of +13% with the second-best technique, and consistent in all the range of number of proposals. In MS-COCO, even though the majority of categories of objects are not present in PASCAL and we did not train on any MS-COCO image, the percentage of absolute improvement is also consistently +13% at least. This shows that our contours, regions, and proposals are properly learning a generic concept of object rather than some specific categories. Qualitative results of the COB proposals can be found in the supplemental.

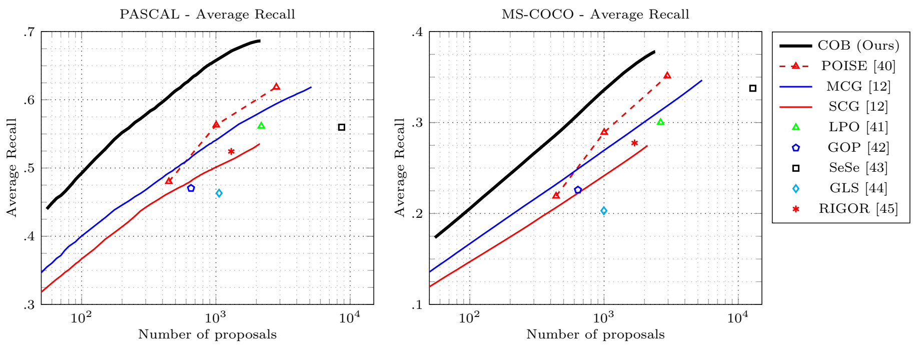


Fig. 9. Object proposals evaluation: Dashed lines refer to methods that do not provide a ranked set of proposals, but they need to be reparameterized.

5.5 Efficiency Analysis

Contour detection and image segmentation, as a preprocessing step towards high-level applications, need to be computationally efficient. The previous state-of-the-art in hierarchical image segmentation [12, 9] was of limited use in practice due to its computational load.

As a core in our system, the forward pass of our network to compute the contour strength and 8 orientations takes 0.259 seconds on a NVidia Titan X GPU. Table 2 shows the timing comparison between the full system COB (Ours) and some related baselines on PASCAL. We divide the timing into different relevant parts, namely, the contour detection step, the Oriented Watershed Transform (OWT) and Ultrametric Contour Map (UCM) computation, and the globalization (normalized cuts) step.

Steps	(1) MCG [12]	(2) MCG-HED	(3) Fast UCMs	(4) COB (Ours)
Contour Detection	3.08	0.39*	0.39*	0.26*
OWT and UCM	11.33	11.58	1.63	1.03
Globalization	9.96	9.97	9.92	0.00
Total Time	24.37	21.94	11.94	**1.34**

Table 2. Timing experiments: Comparing our approach to different baselines. Times computed using a GPU (NVidia Titan X) are marked with an asterisk.

Column (1) shows the timing for the original MCG [12], which uses Structured Edges (SE) [11]. As a first baseline, Column (2) displays the timing of MCG if we naively substitute SE by HED [15] at the three scales (running on a GPU). By applying the sparse boundaries representation we reduce the UCM and OWT time from 11.58 to 1.63 seconds (Column (3)). Our final technique COB, in which we remove the globalization step, computes the three scales in one pass and add contour orientations, takes 1.34 seconds in mean. Overall, comparing to previous state-of-the-art, we get a significant improvement at a fraction of the computation time (24.37 to 1.34 seconds).

6 Conclusions

In this work, we have developed an approach to detect contours at multiple scales, together with their orientations, in a single forward pass of a convolutional neural network. We provide a fast framework for generating region hierarchies by efficiently combining multiscale oriented contour detections, thanks to a new sparse boundary representation. We shift from the BSDS to PASCAL in the evaluation to unwind all the potential of data-hungry methods such as CNNs and by observing that the performance on the BSDS is close to saturation.

Our technique achieves state-of-the-art performance by a significant margin for contour detection, the estimation of their orientation, generic image segmentation, and object proposals. We show that our architecture is modular by using two different CNN base architectures, which suggests that it will be able to transfer further improvements in CNN base architectures to perceptual grouping. We also show that our method does not require globalization, which was a speed bottleneck in previous approaches.

All our code, pre-trained results, and benchmarks will be publicly released.

References

1. Martin, D., Fowlkes, C., Tal, D., Malik, J.: A database of human segmented natural images and its application to evaluating segmentation algorithms and measuring ecological statistics. In: ICCV. (2001)
2. Dollár, P., Tu, Z., Belongie, S.: Supervised learning of edges and object boundaries. In: CVPR. (2006)
3. Mairal, J., Leordeanu, M., Bach, F., Hebert, M., Ponce, J.: Discriminative sparse image models for class-specific edge detection and image interpretation. In: ECCV. (2008)
4. Xiaofeng, R., Bo, L.: Discriminatively trained sparse code gradients for contour detection. In: NIPS. (2012)
5. Maire, M., Yu, S.X., Perona, P.: Reconstructive sparse code transfer for contour detection and semantic labeling. In: ACCV. (2014)
6. Kokkinos, I.: Highly accurate boundary detection and grouping. In: CVPR. (2010)
7. Yarkony, J.E., Ihler, A., Fowlkes, C.: Fast planar correlation clustering for image segmentation. In: ECCV. (2012)
8. Maire, M.: Simultaneous segmentation and figure/ground organization using angular embedding. In: ECCV. (2010)
9. Arbeláez, P., Maire, M., Fowlkes, C., Malik, J.: Contour detection and hierarchical image segmentation. TPAMI **33**(5) (2011) 898–916
10. Ren, Z., Shakhnarovich, G.: Image segmentation by cascaded region agglomeration. In: CVPR. (2013)
11. Dollár, P., Zitnick, C.L.: Structured forests for fast edge detection. In: ICCV. (2013)
12. Arbeláez, P., Pont-Tuset, J., Barron, J., Marques, F., Malik, J.: Multiscale combinatorial grouping. In: CVPR. (2014)
13. Yarkony, J.E., Fowlkes, C.: Planar ultrametrics for image segmentation. In: NIPS. (2015)
14. Kokkinos, I.: Pushing the boundaries of boundary detection using deep learning. In: ICLR. (2016)
15. Xie, S., Tu, Z.: Holistically-nested edge detection. In: ICCV. (2015)
16. Bertasius, G., Shi, J., Torresani, L.: Deepedge: A multi-scale bifurcated deep network for top-down contour detection. In: CVPR. (2015)
17. Bertasius, G., Shi, J., Torresani, L.: High-for-low and low-for-high: Efficient boundary detection from deep object features and its applications to high-level vision. In: ICCV. (2015)
18. Shen, W., Wang, X., Wang, Y., Bai, X., Zhang, Z.: Deepcontour: A deep convolutional feature learned by positive-sharing loss for contour detection. In: CVPR. (2015)
19. Ganin, Y., Lempitsky, V.: N^4-fields: Neural network nearest neighbor fields for image transforms. In: ACCV. (2014)
20. Simonyan, K., Zisserman, A.: Very deep convolutional networks for large-scale image recognition. ICLR (2015)
21. He, K., Zhang, X., Ren, S., Sun, J.: Deep residual learning for image recognition. In: CVPR. (2016)
22. Everingham, M., Van Gool, L., Williams, C.K.I., Winn, J., Zisserman, A.: The PASCAL Visual Object Classes Challenge 2012 (VOC2012) Results. http://www.pascal-network.org/challenges/VOC/voc2012/workshop/index.html

23. Hariharan, B., Arbeláez, P., Bourdev, L., Maji, S., Malik, J.: Semantic contours from inverse detectors. In: ICCV. (2011)
24. Mottaghi, R., Chen, X., Liu, X., Cho, N.G., Lee, S.W., Fidler, S., Urtasun, R., Yuille, A.: The role of context for object detection and semantic segmentation in the wild. In: CVPR. (2014)
25. Krizhevsky, A., Sutskever, I., Hinton, G.E.: Imagenet classification with deep convolutional neural networks. In: NIPS. (2012)
26. Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S., Anguelov, D., Erhan, D., Vanhoucke, V., Rabinovich, A.: Going deeper with convolutions. In: CVPR. (2015)
27. Najman, L., Schmitt, M.: Geodesic saliency of watershed contours and hierarchical segmentation. TPAMI **18**(12) (1996) 1163–1173
28. Lee, C.Y., Xie, S., Gallagher, P., Zhang, Z., Tu, Z.: Deeply-supervised nets. arXiv preprint arXiv:1409.5185 (2014)
29. Jia, Y., Shelhamer, E., Donahue, J., Karayev, S., Long, J., Girshick, R., Guadarrama, S., Darrell, T.: Caffe: Convolutional architecture for fast feature embedding. arXiv preprint arXiv:1408.5093 (2014)
30. Martin, D., Fowlkes, C., Malik, J.: Learning to detect natural image boundaries using local brightness, color, and texture cues. TPAMI **26**(5) (2004) 530–549
31. Pont-Tuset, J., Marques, F.: Measures and meta-measures for the supervised evaluation of image segmentation. In: CVPR. (2013)
32. Zhao, Q.: Segmenting natural images with the least effort as humans. In: BMVC. (2015)
33. Premachandran, V., Bonev, B., Yuille, A.L.: Pascal boundaries: A class-agnostic semantic boundary dataset. arXiv preprint arXiv:1511.07951 (2015)
34. Shi, J., Malik, J.: Normalized cuts and image segmentation. TPAMI **22**(8) (2000)
35. Felzenszwalb, P.F., Huttenlocher, D.P.: Efficient graph-based image segmentation. IJCV **59** (2004) 2004
36. Comaniciu, D., Meer, P.: Mean shift: a robust approach toward feature space analysis. TPAMI **24**(5) (2002) 603 –619
37. Girshick, R., Donahue, J., Darrell, T., Malik, J.: Rich feature hierarchies for accurate object detection and semantic segmentation. In: CVPR. (2014)
38. Girshick, R.: Fast R-CNN. In: ICCV. (2015)
39. Ren, S., He, K., Girshick, R., Sun, J.: Faster R-CNN: Towards real-time object detection with region proposal networks. In: NIPS. (2015)
40. Humayun, A., Li, F., Rehg, J.M.: The middle child problem: Revisiting parametric min-cut and seeds for object proposals. In: ICCV. (2015)
41. Krähenbühl, P., Koltun, V.: Learning to propose objects. In: CVPR. (2015)
42. Krähenbühl, P., Koltun, V.: Geodesic object proposals. In: ECCV. (2014)
43. Uijlings, J.R.R., van de Sande, K.E.A., Gevers, T., Smeulders, A.W.M.: Selective search for object recognition. IJCV **104**(2) (2013) 154–171
44. Rantalankila, P., Kannala, J., Rahtu, E.: Generating object segmentation proposals using global and local search. In: CVPR. (2014)
45. Humayun, A., Li, F., Rehg, J.M.: RIGOR: Recycling Inference in Graph Cuts for generating Object Regions. In: CVPR. (2014)
46. Hosang, J., Benenson, R., Dollár, P., Schiele, B.: What makes for effective detection proposals? PAMI (2015)
47. Pont-Tuset, J., Van Gool, L.: Boosting object proposals: From Pascal to COCO. In: ICCV. (2015)
48. Lin, T., Maire, M., Belongie, S., Bourdev, L.D., Girshick, R.B., Hays, J., Perona, P., Ramanan, D., Dollár, P., Zitnick, C.L.: Microsoft COCO: common objects in context. arXiv:1405.0312 (2014)