Open access • Book Chapter • DOI:10.1007/978-3-319-46448-0_35

Convolutional Oriented Boundaries — Source link

Kevis-Kokitsi Maninis, Jordi Pont-Tuset, Pablo Arbeláez, Luc Van Gool ...+1 more authors

Institutions: ETH Zurich, University of Los Andes, Katholieke Universiteit Leuven

Published on: 08 Oct 2016 - European Conference on Computer Vision

Topics: Convolutional neural network

Convolutional Oriented Boundaries

Kevis-Kokitsi Maninis[1(✉)], Jordi Pont-Tuset[1], Pablo Arbeláez[2], and Luc Van Gool[1,3]

[1] ETH Zürich, Zürich, Switzerland
kmaninis@vision.ee.ethz.ch
[2] Universidad de Los Andes, Bogotá, Colombia
[3] KU Leuven, Leuven, Belgium

Abstract. We present Convolutional Oriented Boundaries (COB), which produces multiscale oriented contours and region hierarchies starting from generic image classification Convolutional Neural Networks (CNNs). COB is computationally efficient, because it requires a single CNN forward pass for contour detection and it 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 orientation provides more accurate results. We perform extensive experiments on BSDS, PASCAL Context, PASCAL Segmentation, and MS-COCO, showing that COB provides state-of-the-art contours, region hierarchies, and object proposals in all datasets.

Keywords: Contour detection · Contour orientation estimation · Hierarchical image segmentation · Object proposals

1 Introduction

The adoption of Convolutional Neural Networks (CNNs) has caused a profound change and a large leap forward in performance throughout the majority of fields in computer vision. In the case of a traditionally category-agnostic field such as contour detection, it has recently fostered the appearance of systems [1–6] that rely on large-scale category-specific information in the form of deep architectures pre-trained on Imagenet for image classification [7–10].

This paper proposes Convolutional Oriented Boundaries (COB), a generic CNN architecture that allows end-to-end learning of multiscale oriented contours, and we show how it translates top performing base CNN networks into high-quality contours; allowing to bring future improvements in base CNN architectures into semantic grouping. We then propose a sparse boundary representation for efficient construction of hierarchical regions from the contour signal. Our overall approach is both efficient (it runs in 0.8 seconds per image) and highly accurate (it produces state-of-the-art contours and regions on PASCAL and on the BSDS). Figure 1 shows an overview of our system.

B. Leibe et al. (Eds.): ECCV 2016, Part I, LNCS 9905, pp. 580–596, 2016.
DOI: 10.1007/978-3-319-46448-0_35

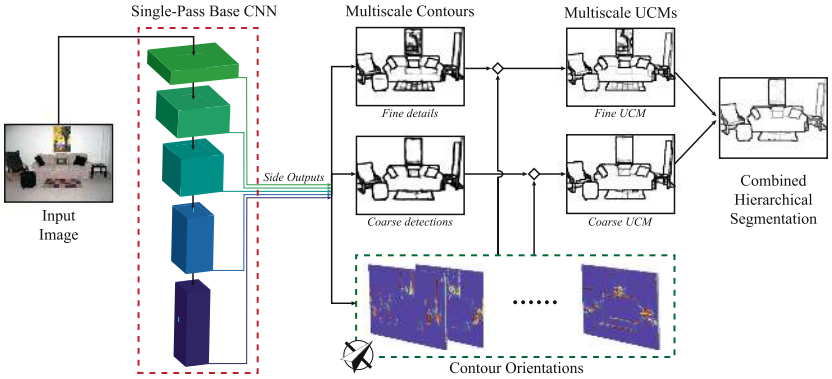


Fig. 1. Overview of COB: From a single pass of a base CNN, we obtain multiscale oriented contours. We combine them to build Ultrametric Contour Maps (UCMs) at different scales and fuse them into a single hierarchical segmentation structure.

For the last fifteen years, the Berkeley Segmentation Dataset and Benchmark (BSDS) [11] has been the experimental testbed of choice for the study of boundary detection and image segmentation. However, the current large-capacity and very accurate models have 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.

In terms of scale and difficulty, the next natural frontier for perceptual grouping is the PASCAL VOC dataset [12], 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. [13], who annotated the VOC dataset for category-specific boundary detection on the foreground objects. More recently, the PASCAL Context dataset [14] 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 direction, this paper investigates the transition from the BSDS to PASCAL Context in the evaluation of image segmentation.

We derive valuable insights from studying perceptual grouping in a larger and more challenging empirical framework. Among them, we observe that COB leverages increasingly deeper state-of-the-art architectures, such as the recent Residual Networks [10], 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 [15] and used in state-of-the-art methods [1,16] is unnecessary in the presence of the high-level knowledge conveyed by pre-trained CNNs and oriented contours, thus removing a significant computational

bottleneck for high-quality contours. Overall, COB generates state-of-the-art contours and regions on PASCAL Context and on the BSDS while being computationally very efficient: it runs in 0.8 seconds per image.

We also conduct comprehensive experiments demonstrating the interest of COB for downstream recognition applications. We use our hierarchical regions as input to the combinatorial grouping algorithm of [16] and obtain state-of-the-art segmented object proposals on PASCAL Segmentation 2012 by a significant margin. Furthermore, we provide empirical evidence for the generalization power of COB by evaluating our object proposals without any retraining in the even larger and more challenging MS-COCO dataset, where we also report a large improvement in performance with respect to the state of the art. Our efforts on segmentation through CNNs have also found application in retinal image segmentation [17], obtaining state-of-the-art and super-human performance in vessel and optic disc segmentation, which further highlights their generality.

The COB code, pre-computed results, pre-trained models, and benchmarks are publicly available at www.vision.ee.ethz.ch/~cvlsegmentation/.

2 Related Work

The latest wave of contour detectors takes advantage of deep learning to obtain state-of-the-art results [1–6,18]. Ganin and Lempitsky [6] use a deep architecture to extract features of image patches. They approach contour detection as a multiclass classification task, by matching the extracted features to predefined ground-truth features. The authors of [3,4] 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. [5] learn deep features using shape information. Xie and Tu [2] 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 [1] builds upon [2] and improves the results by tuning the loss function, running the detector at multiple scales, and adding globalization. COB is different from this previous work in that we obtain multiscale information in a single pass of the network on the whole image, it combines the per-pixel classification with contour orientation estimation, and its output is richer than a linear combination of cues at different scales.

At the core of all these deep learning approaches, lies a *base CNN*, starting from the seminal AlexNet [7] (8 layers), through the more complex VGGNet [9] (16 layers) and inception architecture of GoogLeNet [8] (22 layers), to the very recent and very deep ResNets [10] (up to 1001 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 COB is modular and can incorporate and benefit from future improvements in the base CNN.

Recent work has also explored the weakly supervised or unsupervised learning of contours: Khoreva et al. [19] learn from the results of generic contour

detectors coupled with object detectors; and Li et al. [20] train contour detectors from motion boundaries acquired from video sequences. Yang et al. [21] use conditional random fields to refine the inaccurately localized boundary annotations of PASCAL. Our approach uses full supervision from BSDS and PASCAL Context for contour localization and orientation.

COB exploits the duality between contour detection and segmentation hierarchies, initially studied by Najman and Schmitt [22]. Arbeláez et al. [15] showed its usefulness for jointly optimizing contours and regions. Pont-Tuset et al. [16] leveraged multi-resolution contour detection and proved its interest also for generating object proposals. We differentiate from these approaches in two aspects. First, our sparse boundary representation translates into a clean and highly efficient implementation of hierarchical 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 globalization step (output of normalized cuts), a bottleneck in terms of computational cost, but a cornerstone of previous aproaches.

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 resolution. 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 [15,16].

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, to presenting our CNN architecture to produce contour detection at different scales, and to explain how we estimate the orientation of the edges; all in a single CNN forward pass at the image level.

Training Deep Contour Detectors: The recent success of [2] 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 a contour probability map. Using the notation of the authors, we denote the training dataset by $S = \{(X_n, Y_n), n = 1, \ldots, N\}$,

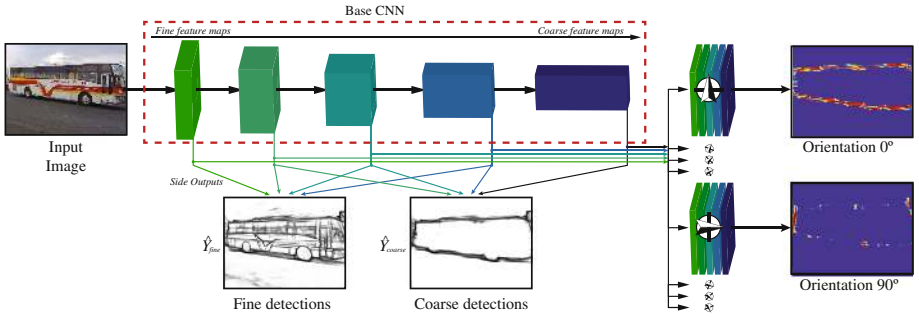


Fig. 2. Our deep learning architecture (best viewed in color). The connections show the different stages that are used to generate the multiscale contours. Orientations further require additional convolutional layers in multiple stages of the network. (Color figure online)

with X_n being the input image and $Y_n = \{y_j^{(n)}, j = 1, \ldots, |X_n|\}, y_j^{(n)} \in \{0, 1\}$ the predicted pixelwise labels. For simplicity, we drop the subscript n. Each of the M side outputs minimizes the objective function:

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

where $\ell_{side}^{(m)}$ is the loss function for scale $m \in \{1, \ldots, M\}$, $\mathbf{W}$ denotes the standard set of parameters of the CNN, and $\{\mathbf{w}^{(m)}, m = 1, \ldots, M\}$ 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 ground-truth Y, respectively. The probability $P(\cdot)$ is obtained by applying a sigmoid $\sigma(\cdot)$ to the activations of the side outputs $\hat{A}_{side}^{(m)} = \{a_j^{(m)}, j = 1, \ldots, |Y|\}$. The activations are finally fused linearly, as: $\hat{Y}_{fuse} = \sigma\left(\Sigma_{m=1}^{M} h_m \hat{A}_{side}^{(m)}\right)$ where $\mathbf{h} = \{h_m, m = 1, \ldots, M\}$ 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 $\mathbf{h}$. In the rest of the paper we use the class-balancing cross-entropy loss function of Eq. 1.

Multiscale Contours: We finetune the 50-layer ResNet [10] 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. Therefore, the network consists mainly of convolutional layers coupled with ReLU activations, divided into 5 stages. We will refer to this architecture as the "base CNN" 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 a half. As discussed before, the CNN naturally contains multiscale information, which we exploit to build a multiscale contour regressor.

We separately supervise the output of the last layer of each stage (side activation), comparing it to the ground truth using the loss function of Eq. 1. This way, we enforce each side activation to produce an intermediate contour map at different resolution. The idea of supervising intermediate parts of a CNN has successfully been used in previous approaches, for a variety of tasks [2,8,23]. In the 5-scale base CNN illustrated in Fig. 2, we linearly combine the side activations of the 4 finest and 4 coarsest scales to a fine-scale and a coarse-scale output ($\hat{Y}_{fine}$ and $\hat{Y}_{coarse}$, respectively) with trainable weights. The finer scale contains better localized contours, whereas the coarse scale leads to less noisy detections. To train the two sets of weights of the linear combinations, we freeze the pre-trained weights of the base CNN.

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 which 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 classify each pixel of the human contour annotations into K different orientations. The orientation of each contour pixel is obtained by approximating the ground-truth boundaries with polygons, and assigning each pixel the orientation of the closest polygonal segment, as shown in Fig. 5. As in the case of multiscale contours, the weights of the base network remain frozen when training these sub-networks.

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`$= 1, \ldots, K$ and `m`$= 1, \ldots, M$. 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:

$$O(x, y) = \mathcal{T}\left(\arg\max_k B_k(x, y)\right), k = 1, \ldots, K \tag{2}$$

where $B_k(x, y)$ denotes the response of the k-th orientation bin of the CNN at the pixels with coordinates (x, y) and $\mathcal{T}(\cdot)$ 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, not to bias the orientations. The different orientations as well as the resulting orientation map (color-coded) are illustrated in Fig. 3.

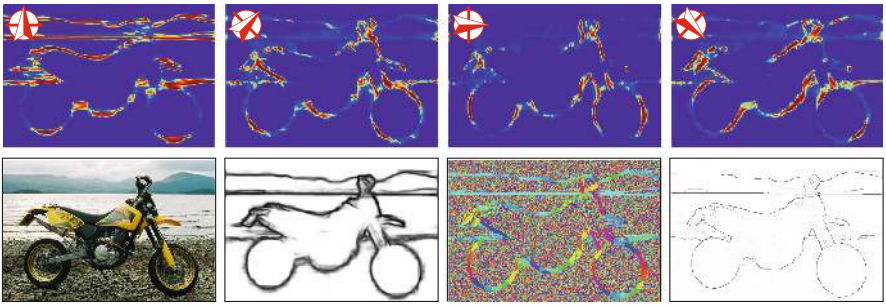


Fig. 3. Illustration of contour orientation learning. Row 1 shows the responses B_k for 4 out of the 8 orientation bins. Row 2, from left to right: original image, contour strength, learned orientation map into 8 orientations, and hierarchical boundaries.

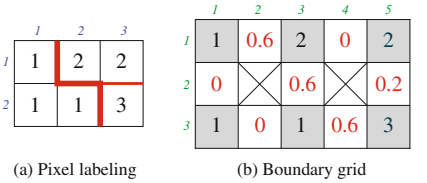
In [15,16,24] the orientations are computed by means of local gradient filters. In Sect. 5 we show that our learned orientations are significantly more accurate and lead to more better region segmentations.

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) [15], which transforms a contour detection probability map into a hierarchical boundary map, which gets partitions at different granularities when thresholding at various contour strength values. Despite the success of UCMs, their low speed significantly limits their applicability.

In the remainder of this section we first describe an alternative representation of an image partition that allows us to reduce the computation time of multiscale UCMs by an order of magnitude, to less than one second. Then, we present the global algorithm to build a hierarchy of regions from the multiscale contours and the orientations presented in Sect. 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 [16].

Sparse Boundary 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 Fig. 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), 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, markers of the boundary positions. (c) Sparse boundaries, lists of boundary coordinates between neighboring regions. (Color figure online)

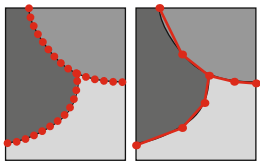


Fig. 5. Polygon simplification: from all boundary points (left) to simplified polygons (right). (Color figure online)

These boundaries are usually stored in the *boundary grid* (Fig. 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* over the entire matrix to perform a modification on a single boundary piece.

Inspired by how sparse matrices are handled, we designed the *sparse boundaries* representation (Fig. 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.

Fast Hierarchies from Multiscale Oriented Contours: The deep CNN presented in Sect. 3 provides different levels of detail for the image contours. A linear combination of the layers is the straightforward way of providing a single contour signal [2]. 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 [16], 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.

A drawback of the original framework [16] is that the manipulation of the hierarchies is very slow (in the order of seconds), so the operations on the UCMs

had to be discretized and performed at a low number of contour strengths. By using the fast sparse boundary representation, we can 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 Oriented Watershed Transform (OWT), further boosting performance.

5 Experiments

This section presents the empirical evidence that supports our approach. First, Sect. 5.1 explores the ablated and baseline techniques studied to isolate and quantify the improvements due to different components of our system. Then Sects. 5.2, 5.3, and 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, Sect. 5.5 analyzes the gain in speed achieved mainly by the use of our sparse boundaries representation.

We extend the main BSDS benchmarks to the PASCAL Context dataset [14], 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 Context train with 4 998 images, *VOC val* corresponds to half the official PASCAL Context 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 different semantic categories and not their parts, in contrast to the BSDS.

We used the publicly available *Caffe* [25] 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.

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 [26]) and regions (F_{op} [27]).

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 [16], which uses Structured Edges [24] as input contour signal, and denote it MCG [16]. We then substitute SE by the newer HED [2], trained on *VOC train* as input contours and denote it MCG-HED.

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

Method	Global.	Orient.	Boundaries - F_b ODS	OIS	AP	Regions - F_{op} ODS	OIS	AP
MCG [16]	✓	✗	0.548	0.594	0.519	0.355	0.419	0.263
MCG-HED	✓	✗	0.691	0.727	0.693	0.459	0.520	0.374
VGGNet-Side	✓	✗	0.644	0.683	0.664	0.439	0.505	0.351
ResNet50-Side	✓	✗	0.676	0.711	0.681	0.456	0.521	0.374
Ours (VGGNet)	✗	✓	0.705	0.735	0.741	0.466	0.533	0.384
Ours (ResNet50)	✗	✗	0.734	0.767	0.757	0.475	0.545	0.405
Ours (ResNet50)	✓	✗	0.726	0.759	0.725	0.461	0.531	0.395
Ours (ResNet50)	✓	✓	0.732	0.763	0.731	0.481	**0.554**	**0.418**
Ours (ResNet50)	✗	✓	**0.737**	**0.768**	**0.758**	**0.483**	0.553	0.417

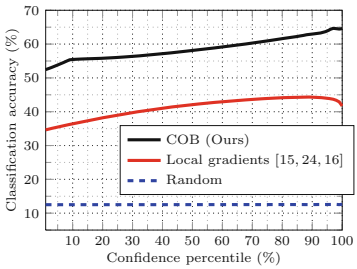


Fig. 6. Contour orientation: classification accuracy into orientations quantized in 8 bins.

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 [9] and ResNet50 [10] on *VOC train* and combined the 5 side outputs with our fast hierarchical regions of Sect. 4 (VGGNet-Side and ResNet50-Side).

We finally evaluate different variants of our system, as presented in Sect. 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 of ResNet translates into 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).

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 Fig. 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.

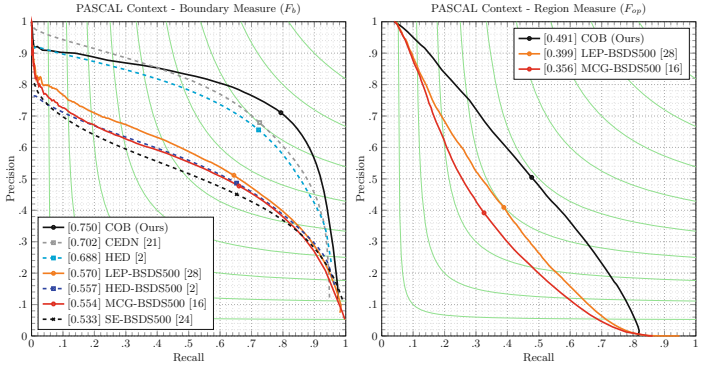
Figure 6 shows the classification accuracy with respect to the confidence of the estimation. We compare our proposed technique against the local gradient estimation used in previous literature [15,16,24]. As a baseline, we plot the result a random guess of the orientations would get. We observe that 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 [14] as well as on the BSDS500 [11], which is the most established benchmark for perceptual grouping.

PASCAL Context: We train COB in the *VOC train*, and perform hyper-parameter selection on *VOC val*. We report the final results on the unseen *VOC test* when trained on *VOC trainval*, using the previously tuned hyper-parameters. We compare our approach to several methods trained on the BSDS [2,16,24, 28] and we also retrain the current state-of-the-art contour detection methods HED [2] and the recent CEDN [21] on *VOC trainval* using the code provided by the respective authors.

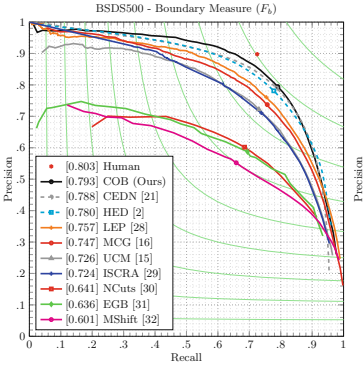
Figure 7 presents the evaluation results of our method compared to the state-of-the-art, which show that COB outperforms all others by a considerable margin both in terms of boundaries and in terms of regions. The lower performance of the methods trained on the BSDS quantifies the difficulty of the task when moving to a larger and more challenging dataset.

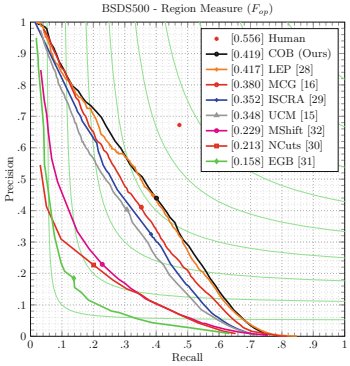


Method	Boundaries - F_b ODS	OIS	AP
COB	**0.750**	**0.781**	**0.773**
CEDN [21]	0.702	0.718	0.744
HED [2]	0.688	0.707	0.704
LEP-B [28]	0.570	0.636	0.547
HED-B [2]	0.557	0.588	0.518
MCG-B [16]	0.554	0.609	0.528
SE-B [24]	0.533	0.568	0.496

Method	Regions - F_{op} ODS	OIS	AP
COB	**0.491**	**0.565**	**0.439**
LEP-B [28]	0.399	0.461	0.304
MCG-B [16]	0.356	0.419	0.275

Fig. 7. PASCAL Context *VOC test* **Evaluation**: Precision-recall curves for evaluation of boundaries (F_b [26]), and regions (F_{op} [27]). Contours in dashed lines and boundaries (from segmentation) in solid lines. ODS, OIS, and AP summary measures.





Method	Boundaries - F_b ODS	OIS	AP
COB (Ours)	**0.793**	**0.820**	**0.859**
CEDN [21]	0.788	0.804	0.834
HED [2]	0.780	0.796	0.834
LEP [28]	0.757	0.793	0.828
MCG [16]	0.747	0.779	0.759
UCM [15]	0.726	0.760	0.727
ISCRA [29]	0.724	0.752	0.783
NCuts [30]	0.641	0.674	0.447
EGB [31]	0.636	0.674	0.581
MShift [32]	0.601	0.644	0.493

Method	Regions - F_{op} ODS	OIS	AP
COB (Ours)	**0.419**	**0.478**	**0.343**
LEP [28]	0.417	0.468	0.334
MCG [16]	0.380	0.433	0.271
ISCRA [29]	0.352	0.418	0.275
UCM [15]	0.348	0.385	0.235
MShift [32]	0.229	0.292	0.122
NCuts [30]	0.213	0.270	0.096
EGB [31]	0.158	0.240	0.080

Fig. 8. BSDS500 test evaluation: precision-recall curves for evaluation of boundaries (F_b [26]), and regions (F_{op} [27]). ODS, OIS, and AP summary measures.

BSDS500: We retrain COB using only the 300 images of the *trainval* set of the BSDS, after data augmentation as suggested in [2], keeping the architecture decided in Sect. 5.1. For comparison to HED [2], we used the model that the authors provide online. We also compare with CEDN [21], by evaluating the results provided by the authors.

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 Context to achieve further progress in the field.

5.4 Object Proposals

Object proposals are an integral part of current object detection and semantic segmentation pipelines [33–35], 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 [16]. We compare against the more recent techniques POISE [36], MCG and SCG [16], LPO [37], GOP [38], SeSe [39], GLS [40], and RIGOR [41]. Recent thorough comparisons of object proposal generation methods can be found in [42,43].

We perform experiments on the PASCAL 2012 Segmentation dataset [12] and on the bigger and more challenging MS-COCO [44] (val set). The hierarchies and combinatorial grouping are trained on PASCAL Context. 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 [42] with respect to the number of object proposals. In PASCAL Segmentation, 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 we did not train on any

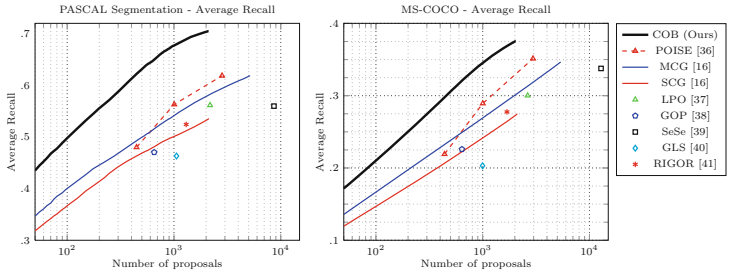


Fig. 9. Object proposals evaluation on PASCAL Segmentation val and MS-COCO val: dashed lines refer to methods that do not provide a ranked set of proposals, but they need to be reparameterized.

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.

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 [15,16] 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 0.28 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 Context. 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.

Column (1) shows the timing for the original MCG [16], which uses Structured Edges (SE) [24]. As a first baseline, Column (2) displays the timing of MCG if we naively substitute SE by HED [2] at the three scales (running on a GPU). By applying the sparse boundaries representation we reduce the UCM

Table 2. Timing experiments: comparing our approach to different baselines. Times computed using a GPU are marked with an asterisk.

Steps	(1) MCG [16]	(2) MCG-HED	(3) Fast UCMs	(4) COB (Ours)
Contour detection	3.08	0.39*	0.39*	0.28*
OWT and UCM	11.33	11.58	1.63	0.51
Globalization	9.96	9.97	9.92	0.00
Total time	24.37	21.94	11.94	**0.79**

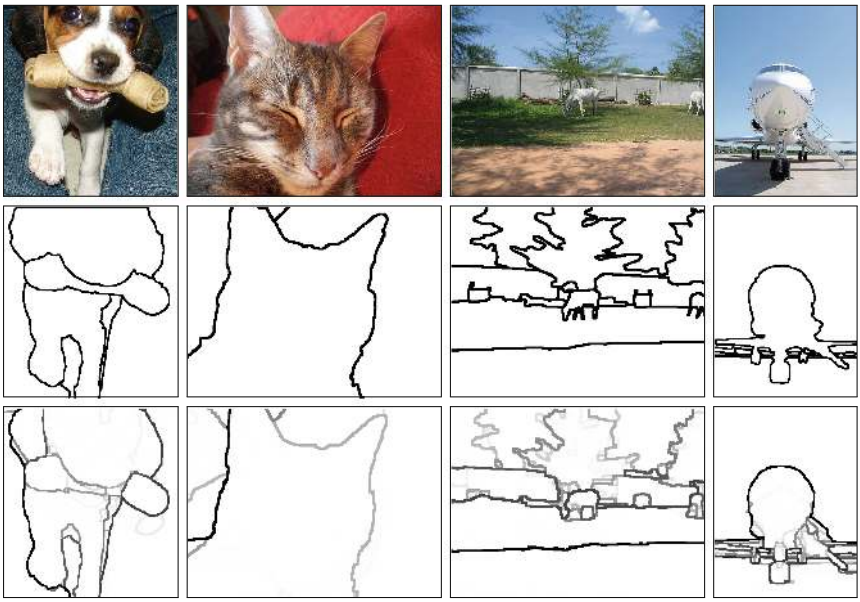


Fig. 10. Qualitative results on PASCAL - hierarchical regions. Row 1: original images, Row 2: ground-truth boundaries, Row 3: hierarchical regions with COB.

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 0.79 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 0.79 seconds).

Qualitative Results: Fig. 10 shows some qualitative results of our hierarchical contours. Please note that COB is capable of correctly distinguishing between internal contours (e.g. cat or dog) and external, semantical, object boundaries.

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, CNN models, pre-computed results, dataset splits, and benchmarks are publicly available at www.vision.ee.ethz.ch/~cvlsegmentation/.

Acknowledgements. Research funded by the EU Framework Programme for Research and Innovation - Horizon 2020 - Grant Agreement No. 645331 - EurEyeCase. The authors gratefully acknowledge support by armasuisse, and thank NVIDIA Corporation for donating the GPUs used in this project.

References

1. Kokkinos, I.: Pushing the boundaries of boundary detection using deep learning. In: ICLR (2016)
2. Xie, S., Tu, Z.: Holistically-nested edge detection. In: ICCV (2015)
3. Bertasius, G., Shi, J., Torresani, L.: Deepedge: a multi-scale bifurcated deep network for top-down contour detection. In: CVPR (2015)
4. 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)
5. 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)
6. Ganin, Y., Lempitsky, V.: N^4-Fields: Neural Network Nearest Neighbor Fields for Image Transforms. In: Cremers, D., Reid, I., Saito, H., Yang, M.-H. (eds.) ACCV 2014. LNCS, vol. 9004, pp. 536–551. Springer, Heidelberg (2015)
7. Krizhevsky, A., Sutskever, I., Hinton, G.E.: Imagenet classification with deep convolutional neural networks. In: NIPS (2012)
8. 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)
9. Simonyan, K., Zisserman, A.: Very deep convolutional networks for large-scale image recognition. In: ICLR (2015)
10. He, K., Zhang, X., Ren, S., Sun, J.: Deep residual learning for image recognition. In: CVPR (2016)
11. 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)
12. Everingham, M., Van Gool, L., Williams, C.K.I., Winn, J., Zisserman, A.: The PASCAL visual object classes challenge (VOC 2012) results (2012). http://www.pascal-network.org/challenges/VOC/voc2012/workshop/index.html
13. Hariharan, B., Arbeláez, P., Bourdev, L., Maji, S., Malik, J.: Semantic contours from inverse detectors. In: ICCV (2011)
14. 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)
15. Arbeláez, P., Maire, M., Fowlkes, C., Malik, J.: Contour detection and hierarchical image segmentation. TPAMI **33**(5), 898–916 (2011)

16. Pont-Tuset, J., Arbeláez, P., Barron, J., Marques, F., Malik, J.: Multiscale combinatorial grouping for image segmentation and object proposal generation. In: TPAMI (2016)
17. Maninis, K., Pont-Tuset, J., Arbeláez, P., Gool, L.V.: Deep retinal image understanding. In: MICCAI (2016)
18. Bertasius, G., Shi, J., Torresani, L.: Semantic segmentation with boundary neural fields. In: CVPR (2016)
19. Khoreva, A., Benenson, R., Omran, M., Hein, M., Schiele, B.: Weakly supervised object boundaries. In: CVPR (2016)
20. Li, Y., Paluri, M., Rehg, J.M., Dollár, P.: Unsupervised learning of edges. In: CVPR (2016)
21. Yang, J., Price, B., Cohen, S., Lee, H., Yang, M.H.: Object contour detection with a fully convolutional encoder-decoder network. In: CVPR (2016)
22. Najman, L., Schmitt, M.: Geodesic saliency of watershed contours and hierarchical segmentation. TPAMI **18**(12), 1163–1173 (1996)
23. Lee, C.Y., Xie, S., Gallagher, P., Zhang, Z., Tu, Z.: Deeply-supervised nets (2014). arXiv preprint arXiv:1409.5185
24. Dollár, P., Zitnick, C.L.: Structured forests for fast edge detection. In: ICCV (2013)
25. Jia, Y., Shelhamer, E., Donahue, J., Karayev, S., Long, J., Girshick, R., Guadarrama, S., Darrell, T.: Caffe: Convolutional architecture for fast feature embedding (2014). arXiv preprint arXiv:1408.5093
26. Martin, D., Fowlkes, C., Malik, J.: Learning to detect natural image boundaries using local brightness, color, and texture cues. TPAMI **26**(5), 530–549 (2004)
27. Pont-Tuset, J., Marques, F.: Supervised evaluation of image segmentation and object proposal techniques. TPAMI **38**(7), 1465–1478 (2016)
28. Ren, Z., Shakhnarovich, G.: Image segmentation by cascaded region agglomeration. In: CVPR (2013)
29. Ren, Z., Shakhnarovich, G.: Image segmentation by cascaded region agglomeration. In: CVPR (2013)
30. Shi, J., Malik, J.: Normalized cuts and image segmentation. TPAMI **22**(8), 888–905 (2000)
31. Felzenszwalb, P.F., Huttenlocher, D.P.: Efficient graph-based image segmentation. IJCV **59**, 167–181 (2004)
32. Comaniciu, D., Meer, P.: Mean shift: a robust approach toward feature space analysis. TPAMI **24**(5), 603–619 (2002)
33. Girshick, R., Donahue, J., Darrell, T., Malik, J.: Rich feature hierarchies for accurate object detection and semantic segmentation. In: CVPR (2014)
34. Girshick, R.: Fast R-CNN. In: ICCV. (2015)
35. Ren, S., He, K., Girshick, R., Sun, J.: Faster R-CNN: towards real-time object detection with regionproposal networks.In: NIPS (2015)
36. Humayun, A., Li, F., Rehg, J.M.: The middle child problem: revisiting parametric min-cut and seeds for object proposals. In: ICCV (2015)
37. Krähenbühl, P., Koltun, V.: Learning to propose objects. In: CVPR (2015)
38. Krähenbühl, P., Koltun, V.: Geodesic Object Proposals. In: Fleet, D., Pajdla, T., Schiele, B., Tuytelaars, T. (eds.) ECCV 2014, Part V. LNCS, vol. 8693, pp. 725–739. Springer, Heidelberg (2014)
39. Uijlings, J.R.R., Sande, K.E.A., Gevers, T., Smeulders, A.W.M.: Selective search for object recognition. IJCV **104**(2), 154–171 (2013)
40. Rantalankila, P., Kannala, J., Rahtu, E.: Generating object segmentation proposals using global and local search. In: CVPR (2014)

41. Humayun, A., Li, F., Rehg, J.M.: RIGOR: Recycling Inference in Graph Cuts for generating Object Regions. In: CVPR (2014)
42. Hosang, J., Benenson, R., Dollár, P., Schiele, B.: What makes for effective detection proposals? TPAMI **38**(4), 814–830 (2016)
43. Pont-Tuset, J., Van Gool, L.: Boosting object proposals: From Pascal to COCO. In: ICCV (2015)
44. 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 (2014). arXiv:1405.0312