Richer Convolutional Features for Edge Detection

Yun Liu, Ming-Ming Cheng, Xiaowei Hu, Jia-Wang Bian, Le Zhang, Xiang Bai, and Jinhui Tang

Abstract—Edge detection is a fundamental problem in computer vision. Recently, convolutional neural networks (CNNs) have pushed forward this field significantly. Existing methods which adopt specific layers of deep CNNs may fail to capture complex data structures caused by variations of scales and aspect ratios. In this paper, we propose an accurate edge detector using richer convolutional features (RCF). RCF encapsulates all convolutional features into more discriminative representation, which makes good usage of rich feature hierarchies, and is amenable to training via backpropagation. RCF fully exploits multiscale and multilevel information of objects to perform the image-to-image prediction holistically. Using VGG16 network, we achieve state-of-the-art performance on several available datasets. When evaluating on the well-known BSDS500 benchmark, we achieve ODS F-measure of **0.811** while retaining a fast speed (**8** FPS). Besides, our fast version of RCF achieves ODS F-measure of **0.806** with **30** FPS. We also demonstrate the versatility of the proposed method by applying RCF edges for classical image segmentation.

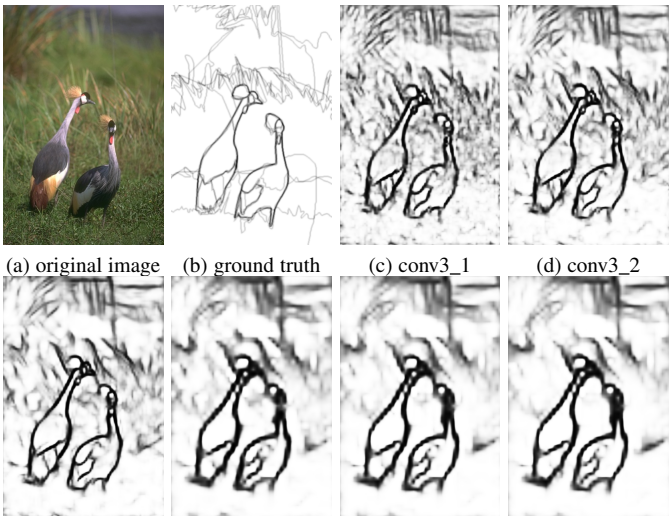
Index Terms—Edge detection, deep learning, richer convolutional features.

1 Introduction

Edge detection can be viewed as a method to extract visually salient edges and object boundaries from natural images. Due to its far-reaching applications in many high-level applications including object detection [2], [3], object proposal generation [4], [5], and image segmentation [6], [7], edge detection is a core low-level problem in computer vision.

The fundamental scientific question here is what is the appropriate representation which is rich enough for a predictor to distinguish edges/boundaries from the image data. To answer this, traditional methods first extract the local cues of brightness, color, gradient and texture, or other manually designed features like Pb [8] and gPb [9], then sophisticated learning paradigms [10] are used to classify edge and non-edge pixels. Although low-level features based edge detectors are somehow promising, their limitations are obvious as well. For example, edges and boundaries are often defined to be semantically meaningful, however, it is difficult to use low-level cues to represent high-level information. Recently, *convolutional neural networks* (CNNs) have become popular in computer vision [11], [12]. Since CNNs have a strong capability to automatically learn the high-level representations for natural images, there is a recent trend of using CNNs to perform edge detection. Some well-known CNN-based methods have pushed forward this field significantly, such as DeepEdge [13], N^4-Fields [14], DeepContour [15], and HED [16]. Our algorithm falls into this category as well.

As illustrated in Fig. 1, we build a simple network to produce side outputs of intermediate layers using VGG16 [11] with HED architecture [16]. We can see that the information obtained by different *convolution* (*i.e. conv*) layers gradually becomes coarser. More importantly, intermediate *conv* layers contain essential fine details. However, previous CNN architectures only use the final *conv* layer or the layers before the pooling layers of neural networks, but ignore the intermediate layers. On the other hand, since richer convolutional features are highly effective for many vision tasks, many researchers make efforts to develop deeper networks [17]. However, it is difficult to get the networks to converge when going deeper because of vanishing/exploding gradients and training data



(a) original image (b) ground truth (c) conv3_1 (d) conv3_2
(e) conv3_3 (f) conv4_1 (g) conv4_2 (h) conv4_3

Fig. 1: We build a simple network based on VGG16 [11] to produce side outputs (c-h). One can see that convolutional features become coarser gradually, and the intermediate layers (c,d,f,g) contain essential fine details that do not appear in other layers.

shortage (*e.g.* for edge detection). So why don't we make full use of the CNN features we have now? Based on these observations, we propose richer convolutional features (RCF), a novel deep structure fully exploiting the CNN features from all the *conv* layers, to perform the pixel-wise prediction for edge detection in an image-to-image fashion. RCF can automatically learn to combine complementary information from all layers of CNNs and thus can obtain accurate representations for objects or object parts in different scales. The evaluation results demonstrate RCF performs very well on edge detection.

After the publication of the conference version [1], our proposed RCF edges have been widely used in weakly supervised semantic segmentation [18], style transfer [19], and stereo matching [20]. Besides, the idea of utilizing all the *conv* layers in a unified framework can be potentially generalized to other vision tasks. This has been demonstrated in skeleton detection [21], medial axis detection [22], people detection [23], and surface fatigue crack identification [24].

When evaluating our method on BSDS500 dataset [9] for edge detection, we achieve a good trade-off between effectiveness and efficiency with the ODS F-measure of **0.811** and the speed of **8** FPS.

* *M.M. Cheng is the corresponding author. URL: http://mmcheng.net/rcfedge/*

- *Y. Liu, M.M. Cheng, and J.W. Bian, are with the College of Computer Science, Nankai University, Tianjin 300350, China.*
- *L. Zhang is with the Advanced Digital Sciences Center.*
- *X. Bai is with Huazhong University of Science and Technology.*
- *J. Tang is with School of Computer Science and Engineering, Nanjing University of Science and Technology, Nanjing 210094, China.*
- *A preliminary version of this work has been published in CVPR 2017 [1].*

It even outperforms human perception (ODS F-measure 0.803). In addition, a fast version of RCF is also presented, which achieves ODS F-measure of **0.806** with **30** FPS. When applying our RCF edges to classic image segmentation, we can obtain high-quality perceptual regions as well.

2 Related Work

As one of the most fundamental problem in computer vision, edge detection has been extensively studied for several decades. *Early pioneering methods* mainly focus on the utilization of intensity and color gradients, such as Canny [25]. However, these early methods are usually not accurate enough for real-life applications. To this end, *feature learning based methods* have been proposed. These methods, such as Pb [8], gPb [9], and SE [10], usually employ sophisticated learning paradigms to predict edge strength with low-level features such as intensity, gradient, and texture. Although these methods are shown to be promising in some cases, these handcrafted features have limited ability to represent high-level information for semantically meaningful edge detection.

Deep learning based algorithms have made vast inroads into many computer vision tasks. Under this umbrella, many deep edge detectors have been introduced recently. Ganin *et al.* [14] proposed N^4-Fields that combines CNNs with the nearest neighbor search. Shen *et al.* [15] partitioned contour data into subclasses and fitted each subclass by learning the model parameters. Recently, Xie *et al.* [16] developed an efficient and accurate edge detector, HED, which performs image-to-image training and prediction. This holistically-nested architecture connects their side output layers, which is composed of one *conv* layer with kernel size 1, one *deconv* layer, and one softmax layer, to the last *conv* layer of each stage in VGG16 [11]. Moreover, Liu *et al.* [26] used relaxed labels generated by bottom-up edges to guide the training process of HED. Wang *et al.* [27] leveraged a top-down backward refinement pathway to effectively learn crisp boundaries. Xu *et al.* [28] introduced a hierarchical deep model to robustly fuse the edge representations learned at different scales. Yu *et al.* [29] extended the success in edge detection to semantic edge detection which simultaneously detected and recognized the semantic categories of edge pixels.

Although these aforementioned CNN-based models have pushed the state of the arts to some extent, they all turn out to be lacking in our view because that they are not able to fully exploit the rich feature hierarchies from CNNs. These methods usually adopt CNN features only from the last layer of each *conv* stage. To address this, we propose a fully convolutional network to combine features from all *conv* layers efficiently.

3 Richer Convolutional Features (RCF)

3.1 Network Architecture

We take inspirations from existing work [12], [16] and embark on the VGG16 network [11]. VGG16 network composes of 13 *conv* layers and 3 fully connected layers. Its *conv* layers are divided into five stages, in which a pooling layer is connected after each stage. The useful information captured by each *conv* layer becomes coarser with its receptive field size increasing. Detailed receptive field sizes of different layers can be found in [16]. The use of this rich hierarchical information is hypothesized to help edge detection. The starting point of our network design lies here.

The novel network introduced by us is shown in Fig. 2. Compared with VGG16, our modifications can be summarized as following:

- We cut all the fully connected layers and the *pool5* layer. On the one side, we remove the fully connected layers to have a fully convolutional network for an image-to-image prediction. On the other hand, adding *pool5* layer will increase the stride by two times, which usually leads to degeneration of edge localization.

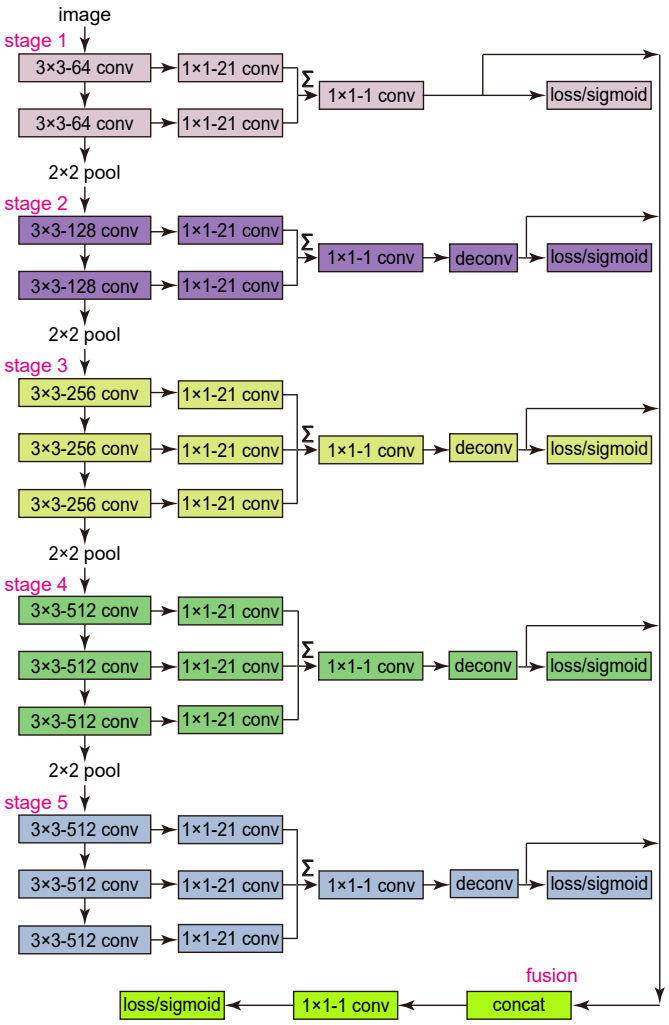


Fig. 2: Our RCF network architecture. The input is an image with arbitrary sizes, and our network outputs an edge possibility map in the same size.

- Each *conv* layer in VGG16 is connected to a *conv* layer with kernel size 1×1 and channel depth 21. And the resulting feature maps in each stage are accumulated using an *eltwise* layer to attain hybrid features.
- An $1 \times 1 - 1$ *conv* layer follows each *eltwise* layer. Then, a *deconv* layer is used to up-sample this feature map.
- A *cross-entropy loss/sigmoid* layer is connected to the up-sampling layer in each stage.
- All the up-sampling layers are concatenated. Then an 1×1 *conv* layer is used to fuse feature maps from each stage. At last, a *cross-entropy loss/sigmoid* layer is followed to get the fusion loss/output.

In RCF, features from all *conv* layers are well-encapsulated into a final representation in a holistic manner which is amenable to training by back-propagation. As receptive field sizes of *conv* layers in VGG16 are different from each other, RCF endows a better mechanism than existing ones to learn multiscale information coming from all levels of convolutional features which we believe are all pertinent for edge detection. In RCF, high-level features are coarser and can obtain strong response at the larger object or object part boundaries as illustrated in Fig. 1 while features from lower-part of CNNs are still beneficial in providing complementary fine details.

3.2 Annotator-robust Loss Function

Edge datasets in this community are usually labeled by several annotators using their knowledge about the presence of objects or object parts. Though humans vary in cognition, these human-labeled edges for the same image share high consistency [8] For each image, we average all the ground truth to generate an edge probability map, which ranges from 0 to 1. Here, 0 means no annotator labeled at this pixel, and 1 means all annotators have labeled at this pixel. We consider the pixels with edge probabilities higher than η as positive samples and the pixels with edge probabilities equal to 0 as negative samples. Otherwise, if a pixel is marked by fewer than η of the annotators, this pixel may be semantically controversial to be an edge point. Thus, regarding those pixels as either positive or negative samples may confuse the networks. Hence we ignore them, but HED tasks them as negative samples and uses a fix η of 0.5.

We compute the loss of each pixel with respect to its label as

$$l(X_i; W) = \begin{cases} \alpha \cdot log\ (1 - P(X_i; W)) & if\ y_i = 0 \\ 0 & if\ 0 < y_i \leq \eta \\ \beta \cdot log\ P(X_i; W) & otherwise, \end{cases} \quad (1)$$

in which

$$\begin{aligned} \alpha &= \lambda \cdot \frac{|Y^+|}{|Y^+| + |Y^-|} \\ \beta &= \frac{|Y^-|}{|Y^+| + |Y^-|}. \end{aligned} \quad (2)$$

Y^+ and Y^- denote the positive sample set and the negative sample set, respectively. The hyper-parameter λ is used to balance the number of positive and negative samples. The activation value (CNN feature vector) and ground truth edge probability at pixel i are presented by X_i and y_i, respectively. $P(X)$ is the standard *sigmoid* function, and W denotes all the parameters that will be learned in our architecture. Therefore, our improved loss function can be formulated as

$$L(W) = \sum_{i=1}^{|I|} \Big(\sum_{k=1}^{K} l(X_i^{(k)}; W) + l(X_i^{fuse}; W) \Big), \quad (3)$$

where $X_i^{(k)}$ is the activation value from stage k while X_i^{fuse} is from the fusion layer. $|I|$ is the number of pixels in image I, and K is the number of stages (equals to 5 here).

3.3 Multiscale Hierarchical Edge Detection

In single scale edge detection, we feed an original image into our fine-tuned RCF network, then, the output is an edge probability map. To further improve the quality of edges, we use image pyramids during the test phase. Specifically, we resize an image to construct an image pyramid, and each of these images is fed into our single-scale detector separately. Then, all resulting edge probability maps are resized to the original image size using bilinear interpolation. At last, these maps are fused to get the final prediction map. We adopt simple average fusion in this study although other advanced strategies are also applicable. In this way, our preliminary version [1] *firstly* demonstrates multiscale testing is still beneficial for edge detection although RCF itself is able to encode multiscale information. Considering the trade-off between accuracy and speed, we use three scales 0.5, 1.0, and 1.5 in this paper. When evaluating RCF on BSDS500 [9] dataset, this simple multiscale strategy improves the ODS F-measure from 0.806 to 0.811 with the speed of 8 FPS which we believe is good enough for real-life applications. See Sec. 4.1 for details.

3.4 Comparison With HED

The most obvious differences between our RCF and HED [16] lie in the three following aspects.

First, HED only considers the last *conv* layer in each stage of VGG16, in which lots of helpful information for edge detection is missed. In contrast to it, RCF uses richer features from all the *conv* layers, making it more possible to capture more object or object-part boundaries across a larger range of scales.

Second, a novel loss function is proposed to treat training examples properly. We consider the edge pixels that η of the annotators labeled as positive samples and the pixels that no annotator labeled as negative samples. Besides, we ignore edge pixels that are marked by a few annotators because of their confusing attributes. In contrast, HED view edge pixels that are marked by less than half of the annotators as negative samples, which may confuse the network training because these pixels are not true non-edge points. Our new loss have been used in [27].

Thirdly, our preliminary version [1] first proposes the multiscale test for edge detection. Recent edge detectors such as HED usually use multiscale network features, but we demonstrate the simple multiscale test is still helpful to edge detection. This idea is also accepted by recent work [27].

4 Experiments on Edge Detection

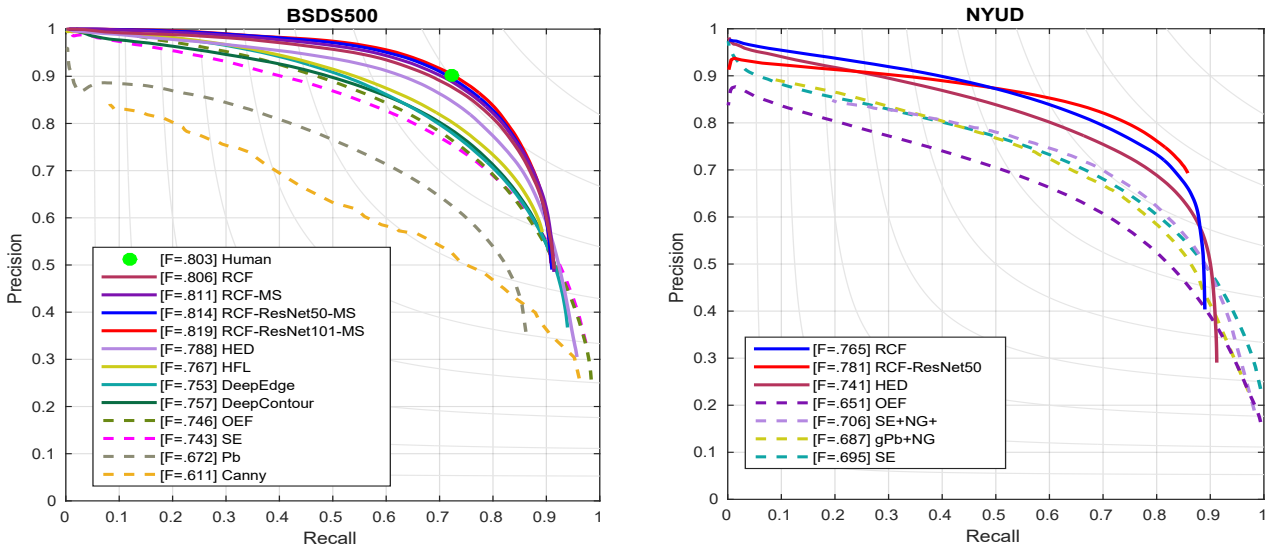
We implement our network using the Caffe framework [31]. The default setting using VGG16 [11] backbone net, and we also test ResNet [17] backbone net. In RCF training, the weights of 1×1 *conv* layers in stage 1-5 are initialized from zero-mean Gaussian distributions with standard deviation 0.01 and the biases are initialized to 0. The weights of the 1×1 *conv* layer in the fusion stage are initialized to 0.2 and the biases are initialized to 0. The weights of other layers are initialized using pre-trained ImageNet models. Stochastic gradient descent (SGD) minibatch samples 10 images randomly in each iteration. For other SGD hyper-parameters, the global learning rate is set to 1e-6 and will be divided by 10 after every 10k iterations. The momentum and weight decay are set to 0.9 and 0.0002, respectively. We run SGD for 40k iterations totally. The parameters η and λ in loss function are set depending on the training data. All experiments in this paper are finished using a NVIDIA TITAN X GPU.

Given an edge probability map, a threshold is needed to produce the binary edge map. There are two choices to set this threshold. The first one is referred as *optimal dataset scale* (ODS) which employs a fixed threshold for all images in a dataset. The second is called *optimal image scale* (OIS) which selects an optimal threshold for each image. We report the F-measure ($\frac{2 \cdot Precision \cdot Recall}{Precision + Recall}$) of both ODS and OIS in our experiments.

4.1 BSDS500 Dataset

BSDS500 [9] is a widely used dataset in edge detection. It is composed of 200 training, 100 validation and 200 test images, each of which is labeled by 4 to 9 annotators. We utilize the training and validation sets for fine-tuning, and test set for evaluation. Data augmentation is the same as [16]. Inspired by the previous work [26], [32], [33], we mix the augmented data of BSDS500 with flipped VOC Context dataset [34] as training data. When training, we set loss parameters η and λ to 0.5 and 1.1, respectively. When evaluating, standard non-maximum suppression (NMS) [10] is applied to thin detected edges. We compare our method with some non-deep-learning algorithms, including Canny [25], Pb [8], SE [10], and OEF [35], and some recent deep learning based approaches, including DeepContour [15], DeepEdge [13], HED [16], HFL [36], MIL+G-DSN+MS+NCuts [33], CASENet [29], AMH [28], CED [27] and *etc*.

Fig. 3a shows the evaluation results. The performance of human eye in edge detection is known as 0.803 ODS F-measure. Both single-scale and multiscale (MS) versions of RCF get better results than average human performance. When comparing with HED [16],



(a) (b)

Fig. 3: The evaluation results on the standard BSDS500 [9] and NYUD [30] datasets. The multiscale version of RCF is only evaluated on the BSDS500 dataset. Here, the solid lines represent CNN based methods, while the dotted lines represent non-deep algorithms.

ODS F-measures of our RCF-MS and RCF are 2.3% and 1.8% higher than it, respectively. Moreover, ResNet50 and ResNet101 can further improve the performance with more *conv* layers. These results demonstrate the effectiveness of the richer convolutional features.

Method	ODS	OIS	FPS
Canny [25]	0.611	0.676	28
Pb [8]	0.672	0.695	-
SE [10]	0.743	0.763	2.5
OEF [35]	0.746	0.770	2/3
DeepContour [15]	0.757	0.776	1/30†
DeepEdge [13]	0.753	0.772	1/1000†
HFL [36]	0.767	0.788	5/6†
N^4-Fields [14]	0.753	0.769	1/6†
HED [16]	0.788	0.808	**30**†
RDS [26]	0.792	0.810	**30**†
CEDN [32]	0.788	0.804	10†
MIL+G-DSN+VOC+MS +NCuts [33]	0.813	0.831	1†
CASENet [29]	0.767	0.784	18†
AMH-ResNet50 [28]	0.798	0.829	-
CED-VGG16 [27]	0.794	0.811	-
CED-ResNet50+VOC+MS [27]	0.817	0.834	-
RCF	0.806	0.823	**30**†
RCF-MS	0.811	0.830	8†
RCF-ResNet50	0.808	0.825	20†
RCF-ResNet50-MS	0.814	0.833	5.4†
RCF-ResNet101	0.812	0.829	12.2†
RCF-ResNet101-MS	**0.819**	**0.836**	3.6†

Fig. 4: The comparison with some competitors on the BSDS500 [9] dataset. † means GPU time.

We show statistic comparison in Fig. 4. From RCF to RCF-MS, the ODS F-measure increases from 0.806 to 0.811, though the speed drops from 30 FPS to 8 FPS. It proves the validity of our multiscale strategy. RCF with ResNet101 [17] achieves a state-of-the-art 0.819 ODS F-measure. We also observe an interesting phenomenon in which the RCF curves are not as long as other methods when evaluated using the default parameters in BSDS500 benchmark. It may suggest that RCF tends to only remain very confident edges. Our methods also achieve better results than recent edge detectors, such as AMH [28] and CED [27]. Note that AMH and CED use complex networks with more weights than our simple RCF. Our RCF network only adds some 1×1 *conv* layers to HED, so the time consumption is on par with HED. We can see that RCF achieves a good trade-off between effectiveness and efficiency.

4.2 NYUD Dataset

NYUD [30] dataset is composed of 1449 densely labeled pairs of aligned RGB and depth images captured from indoor scenes. Recently, many works have conducted edge evaluation on it, such as [10]. Gupta *et al.* [37] split NYUD dataset into 381 training, 414 validation, and 654 test images. We follow their settings and train RCF using the training and validation sets as in HED [16].

We utilize depth information by using HHA [38], in which depth information is encoded into three channels: horizontal disparity, height above ground, and angle with gravity. Thus HHA features can be represented as a color image by normalization. Then, two models for RGB images and HHA feature images are trained separately. In the training process, λ is set to 1.2 for both RGB and HHA. Since NYUD only has one ground truth for each image, η is useless here. Other network settings are the same as used for BSDS500. At the test phase, the final edge predictions are defined by averaging the outputs of RGB model and HHA model. Since there is already an average operation, the multiscale test is not evaluated here. When evaluating, we increase localization tolerance, which controls the maximum allowed distance in matches between predicted edges and ground truth, from 0.0075 to 0.011, because images in NYUD dataset are larger than images in BSDS500 dataset.

We compare our single-scale version of RCF with some well-established competitors. OEF [35] only uses RGB images, while other methods employ both depth and RGB information. The precision-recall curves are shown in Fig. 3b. RCF achieves competitive performance on NYUD dataset, and it is significantly better than HED. Fig. 5 shows the statistical comparison. We can see that RCF outperforms HED not only on separate HHA or RGB data, but also on the merged RGB-HHA data. For HHA and RGB data, ODS F-measure of RCF is 2.2% and 2.6% higher than HED, respectively. For merging RGB-HHA data, RCF is 2.4% higher than HED. Furthermore, HHA edges perform worse than RGB, but averaging HHA and RGB edges achieves much higher results. It suggests that combining different types of information is very useful for edge detection, and this may explain why OEF performs much

Method	ODS	OIS	FPS
OEF [35]	0.651	0.667	1/2
gPb+NG [37]	0.687	0.716	1/375
SE [10]	0.695	0.708	5
SE+NG+ [38]	0.706	0.734	1/15
HED-HHA [16]	0.681	0.695	**20**†
HED-RGB [16]	0.717	0.732	**20**†
HED-RGB-HHA [16]	0.741	0.757	10†
RCF-HHA	0.703	0.717	**20**†
RCF-RGB	0.743	0.757	**20**†
RCF-RGB-HHA	0.765	0.780	10†
RCF-ResNet50-RGB-HHA	**0.781**	**0.793**	7†

Fig. 5: The comparison with some competitors on the NYUD dataset [30]. †means GPU time.

worse than other methods. RCF with ResNet50 [17] improves a 1.6% ODS F-measure when compared with RCF with VGG16 [11].

4.3 Multicue Dataset

Multicue dataset is proposed by Mély *et al.* [40] to study psychophysics theory for boundary detection. It is composed of short binocular video sequences of 100 challenging natural scenes captured by a stereo camera. Each scene contains a left and a right view short (10-frame) color sequences. The last frame of the left images for each scene is labeled for two annotations: object boundaries and low-level edges. Unlike people who usually use boundary and edge interchangeably, they strictly defined boundary and edge according to visual perception at different stages. Thus, boundaries are referred to the boundary pixels of meaningful objects, and edges are abrupt pixels at which the luminance, color, or stereo changes sharply. In this subsection, we use boundary and edge as defined by Mély *et al.* [40] while considering boundary and edge having the same meaning in previous sections.

As done in Mély *et al.* [40] and HED [16], we randomly split these human-labeled images into 80 training and 20 test samples, and average the scores of three independent trials as final results. When training on Multicue, λ is set to 1.1, and η is set to 0.4 for boundary task and 0.3 for edge task. For boundary detection task, we use learning rate 1e-6 and run SGD for 2k iterations. For edge detection task, we use learning rate 1e-7 and run SGD for 4k iterations. Since the image resolution of Multicue is very high, we randomly crop 500×500 patches from original images at each iteration.

We use VGG16 [11] as the backbone net. The evaluation results are summarized in Fig. 9. Our proposed RCF achieves substantially higher results than HED. For boundary task, RCF-MS is 1.1% ODS F-measure higher and 1.4% OIS F-measure higher than HED. For edge task, RCF-MS is 0.9% ODS F-measure higher than HED. Note that the fluctuation of RCF is also smaller than HED, which suggests RCF is more robust over different kinds of images. Some qualitative results are shown in Fig. 6.

4.4 Network Discussion

To further explore the effectiveness of our network architecture, we implement some mixed networks using VGG16 [11] by connecting our richer feature side outputs to some convolution stages while connecting side outputs of HED to the other stages. With training only on BSDS500 [9] dataset and testing on the single scale, evaluation results of these mixed networks are shown in Fig. 10. The last two lines of this table correspond to HED and RCF, respectively. We can observe that all of these mixed networks perform better than HED and worse than RCF that is fully connected to RCF side outputs. It clearly demonstrates the importance of our strategy of richer convolutional features.

In order to investigate whether including additional nonlinearity helps, we connecting ReLU layer after $1 \times 1 - 21$ or $1 \times 1 - 1$ *conv* layers in each stage. However, the network performs worse. Especially, when we attempt to add nonlinear layers to $1 \times 1 - 1$ *conv* layers, the network can not converge properly.

5 Experiments on Image Segmentation

The predicted edges of natural images are often used in another low-level vision technique, image segmentation, which aims to cluster similar pixels to form perceptual regions. To transform a predicted edge map into a segmentation, Arbeláez [9] introduced the Ultrametric Contour Map (UCM) that can generate different image partitions when thresholding this hierarchical contour map at various contour probability values. MCG [6] develops a fast normalized cuts algorithm to accelerate [9] and makes effective use of multiscale information to generate an accurate hierarchical segmentation tree. Note that MCG needs edge orientations as input. These orientations are usually computed using simple morphological operations. COB [46] simultaneously predicts the magnitudes and orientations of edges using HED-based CNNs, and then applies MCG framework to convert these predictions to UCM. Since much more accurate edge orientations are used, COB achieves the state-of-the-art segmentation results.

In order to demonstrate the versatility of the proposed method, here we evaluate the edges of RCF in the context of image segmentation. Specifically, we apply the COB framework but replacing the HED edges with our RCF edges to perform image segmentation. We evaluate the resulting segmenter on the BSDS500 [9] and NYUD [30] datasets. Note that COB uses ResNet50 as its backbone net, so we also test RCF with ResNet for fair comparison. Besides the boundary measure (F_b) [8] used in Sec. 4, we also use the evaluation metric of precision-recall for objects and parts (F_{op}) [39] to evaluate the region similarity between the segmentation and the corresponding ground truth.

BSDS500 Dataset

On the challenging BSDS500 dataset [9], we compare RCF with some well-known generic image segmenters, including NCut [41], MShift [42], EGB [43], gPb-UCM [9], ISCRA [44], MCG [6], LEP [45], and COB [46]. The evaluation results are shown in Fig. 7. RCF achieves the new state of the art on this dataset, both in terms of boundary and region quality. COB [46] gets the second place. We show the numeric comparison in Fig. 11. For the boundary measure, both the ODS and OIS F-measure of RCF are 1.3% higher than COB. For the region measure, the ODS and OIS F-measure of RCF are 2.4% and 3.0% higher than COB, respectively. Using the ResNet as the backbone net, RCF can further improve performance. Since COB uses the edges produced by HED [16], these results demonstrate the effectiveness of RCF architecture. From Fig. 7, we can also see that although the boundary measure of RCF segmentation have reached human performance, the region measure is still far away from the human performance. It indicates that better perception regions should be the main pursuit of classical image segmentation in the future.

NYUD Dataset

On the RGB-D dataset of NYUD [30], we compare not only with some RGB based methods, *e.g.* gPb-UCM [9] and MCG [6], but also with some RGB-D based methods, *e.g.* gPb+NG [37], SE+NG+ [38], and COB [46]. The precision-recall curves of boundary and region measures are displayed in Fig. 8. The numeric comparison is summarized in Fig. 12. Our RCF with VGG16 achieves higher F-measure score than COB on the region measure, while performs slightly worse than original COB on the boundary measure. With ResNet50 as the backbone net, RCF achieves similar performance with COB on the boundary measure but 1.6% higher on the region

Fig. 6: Some examples of RCF. **Top two rows**: BSDS500 [9]. **Bottom two rows**: NYUD [30]. From **Left** to **Right**: origin image, ground truth, RCF edge map, RCF UCM map.

measure. Moreover, both COB and RCF outperform traditional methods by a large margin, which demonstrates the importance of accurate edges in the classic image segmentation.

6 Conclusion

In this paper, we introduce richer convolutional features (RCF), a novel CNN architecture which makes good usage of feature hierarchies in CNNs, for edge detection. RCF encapsulates both semantic and fine detail features by leveraging all convolutional features. RCF is both accurate and efficient, making it promising to be applied in other vision tasks. We also achieve competitive results when applying RCF edges for classical image segmentation. RCF architecture can be seen as a development direction of fully convolutional networks, like FCN [12] and HED [16]. It would be interesting to explore the effectiveness of our network architecture in other hot topics [21], [22], [23], [24]. Source code is available at https://mmcheng.net/rcfedge/.

Acknowledgments

This research was supported by NSFC (NO. 61620106008, 61572264), the national youth talent support program, Tianjin Natural Science Foundation for Distinguished Young Scholars (NO. 17JCJQJC43700), Huawei Innovation Research Program.

References

[1] Y. Liu, M.-M. Cheng, X. Hu, K. Wang, and X. Bai, "Richer convolutional features for edge detection," in *IEEE Conf. Comput. Vis. Pattern Recog.*, 2017, pp. 5872–5881.

[2] S. Ullman and R. Basri, "Recognition by linear combinations of models," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 13, no. 10, pp. 992–1006, 1991.

[3] V. Ferrari, L. Fevrier, F. Jurie, and C. Schmid, "Groups of adjacent contour segments for object detection," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 30, no. 1, pp. 36–51, 2008.

[4] C. L. Zitnick and P. Dollár, "Edge boxes: Locating object proposals from edges," in *Eur. Conf. Comput. Vis.*, 2014, pp. 391–405.

[5] Z. Zhang, Y. Liu, X. Chen, Y. Zhu, M.-M. Cheng, V. Saligrama, and P. H. Torr, "Sequential optimization for efficient high-quality object proposal generation," *IEEE Trans. Pattern Anal. Mach. Intell.*, 2017.

[6] P. Arbeláez, J. Pont-Tuset, J. T. Barron, F. Marques, and J. Malik, "Multiscale combinatorial grouping," in *IEEE Conf. Comput. Vis. Pattern Recog.*, 2014, pp. 328–335.

[7] M.-M. Cheng, Y. Liu, Q. Hou, J. Bian, P. Torr, S.-M. Hu, and Z. Tu, "HFS: Hierarchical feature selection for efficient image segmentation," in *Eur. Conf. Comput. Vis.*, 2016, pp. 867–882.

[8] D. R. Martin, C. C. Fowlkes, and J. Malik, "Learning to detect natural image boundaries using local brightness, color, and texture cues," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 26, no. 5, pp. 530–549, 2004.

[9] P. Arbeláez, M. Maire, C. Fowlkes, and J. Malik, "Contour detection and hierarchical image segmentation," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 33, no. 5, pp. 898–916, 2011.

[10] P. Dollár and C. L. Zitnick, "Fast edge detection using structured forests," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 37, no. 8, pp. 1558–1570, 2015.

[11] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," in *Int. Conf. Learn. Represent.*, 2015.

[12] J. Long, E. Shelhamer, and T. Darrell, "Fully convolutional networks for semantic segmentation," in *IEEE Conf. Comput. Vis. Pattern Recog.*, 2015, pp. 3431–3440.

[13] G. Bertasius, J. Shi, and L. Torresani, "DeepEdge: A multi-scale bifurcated deep network for top-down contour detection," in *IEEE Conf.*

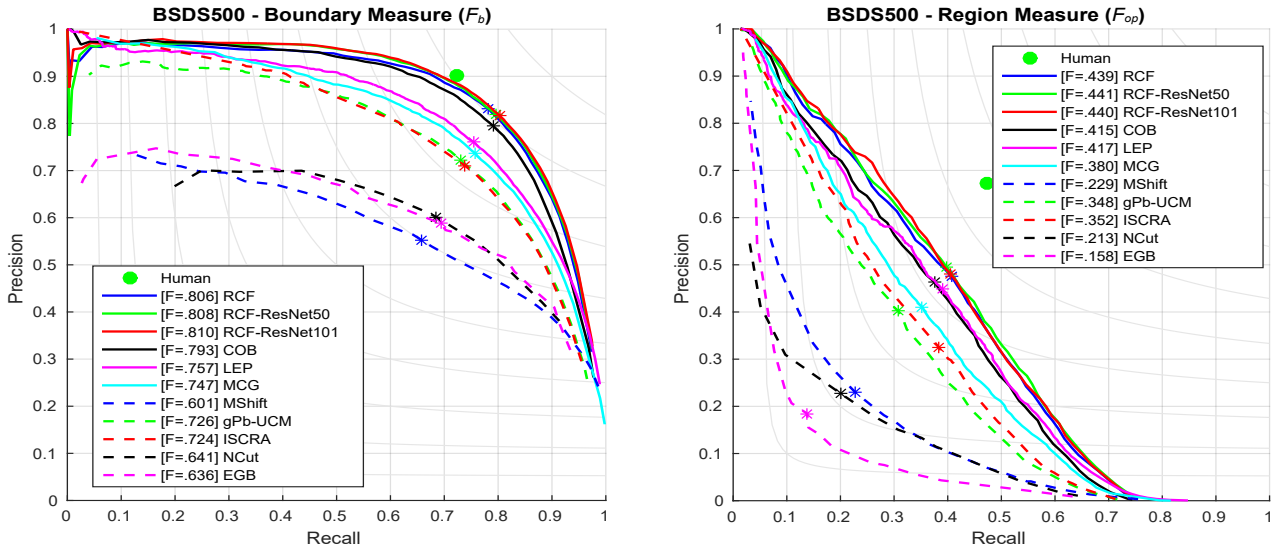


Fig. 7: The precision-recall curves for the evaluation of boundary measure (F_b [8]) and region measure (F_{op} [39]) of classical image segmentation on the BSDS500 test set [9].

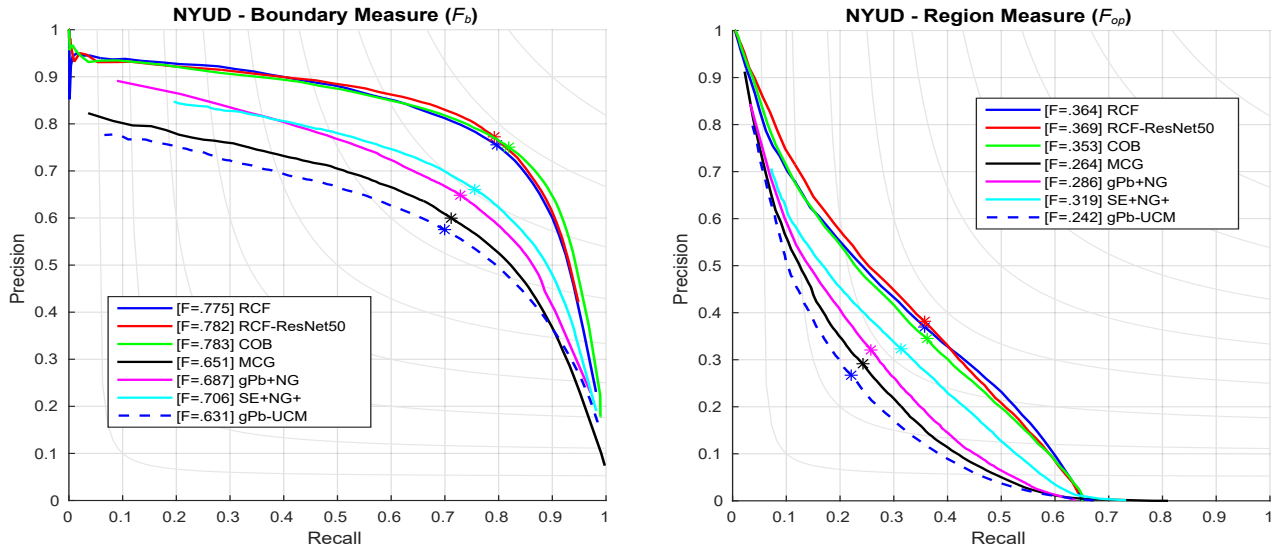


Fig. 8: The precision-recall curves for the evaluation of boundary measure (F_b [8]) and region measure (F_{op} [39]) of classical image segmentation on the NYUD test set [30].

Method	ODS	OIS
Human-Boundary [40]	0.760 (0.017)	–
Multicue-Boundary [40]	0.720 (0.014)	–
HED-Boundary [16]	0.814 (0.011)	0.822 (0.008)
RCF-Boundary	0.817 (**0.004**)	0.825 (**0.005**)
RCF-MS-Boundary	**0.825** (0.008)	**0.836** (0.007)
Human-Edge [40]	0.750 (0.024)	–
Multicue-Edge [40]	0.830 (**0.002**)	–
HED-Edge [16]	0.851 (0.014)	**0.864** (0.011)
RCF-Edge	0.857 (0.004)	0.862 (**0.004**)
RCF-MS-Edge	**0.860** (0.005)	**0.864** (**0.004**)

Fig. 9: The comparisons on the Multicue dataset [40]. The numbers in the parentheses mean standard deviations.

RCF Stage	HED Stage	ODS	OIS
1, 2	3, 4, 5	0.792	0.810
2, 4	1, 3, 5	0.795	0.812
4, 5	1, 2, 3	0.790	0.810
1, 3, 5	2, 4	0.794	0.810
3, 4, 5	1, 2	0.796	0.812
–	1, 2, 3, 4, 5	0.788	0.808
1, 2, 3, 4, 5	–	**0.798**	**0.815**

Fig. 10: Results of some thought networks.

Comput. Vis. Pattern Recog., 2015, pp. 4380–4389.

[14] Y. Ganin and V. Lempitsky, "N^4-Fields: Neural network nearest neighbor fields for image transforms," in *ACCV*, 2014, pp. 536–551.

[15] W. Shen, X. Wang, Y. Wang, X. Bai, and Z. Zhang, "DeepContour: A deep convolutional feature learned by positive-sharing loss for contour detection," in *IEEE Conf. Comput. Vis. Pattern Recog.*, 2015, pp. 3982–3991.

[16] S. Xie and Z. Tu, "Holistically-nested edge detection," *Int. J. Comput. Vis.*, vol. 125, no. 1-3, pp. 3–18, 2017.

[17] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *IEEE Conf. Comput. Vis. Pattern Recog.*, 2016, pp. 770–778.

[18] Q. Hou, M.-M. Cheng, J. Liu, and P. H. Torr, "Webseg: Learning semantic segmentation from web searches," *arXiv preprint arXiv:1803.09859*, 2018.

[19] X.-C. Liu, M.-M. Cheng, Y.-K. Lai, and P. L. Rosin, "Depth-aware

Methods	Boundaries (F_b)		Regions (F_{op})	
	ODS	OIS	ODS	OIS
NCut [41]	0.641	0.674	0.213	0.270
MShift [42]	0.601	0.644	0.229	0.292
EGB [43]	0.636	0.674	0.158	0.240
gPb-UCM [9]	0.726	0.760	0.348	0.385
ISCRA [44]	0.724	0.752	0.352	0.418
MCG [6]	0.747	0.779	0.380	0.433
LEP [45]	0.757	0.793	0.417	0.468
COB [46]	0.793	0.820	0.415	0.466
RCF	0.806	0.833	0.439	0.496
RCF-ResNet50	0.808	0.833	**0.441**	0.500
RCF-ResNet101	**0.810**	**0.836**	0.440	**0.501**

Fig. 11: Evaluation results of boundaries (F_b [8]) and regions (F_{op} [39]) on the BSDS500 test set [9].

Methods	Boundaries (F_b)		Regions (F_{op})	
	ODS	OIS	ODS	OIS
gPb-UCM [9]	0.631	0.661	0.242	0.283
MCG [6]	0.651	0.681	0.264	0.300
gPb+NG [37]	0.687	0.716	0.286	0.324
SE+NG+ [38]	0.706	0.734	0.319	0.359
COB [46]	**0.783**	**0.804**	0.353	0.396
RCF	0.775	0.798	0.364	**0.409**
RCF-ResNet50	0.782	0.803	**0.369**	0.406

Fig. 12: Evaluation results of boundaries (F_b [8]) and regions (F_{op} [39]) on the NYUD test set [30].

neural style transfer," in *Proceedings of the Symposium on Non-Photorealistic Animation and Rendering*, 2017.

[20] X. Song, X. Zhao, H. Hu, and L. Fang, "EdgeStereo: A context integrated residual pyramid network for stereo matching," *arXiv preprint arXiv:1803.05196*, 2018.

[21] K. Zhao, W. Shen, S. Gao, D. Li, and M.-M. Cheng, "Hi-Fi: Hierarchical feature integration for skeleton detection," in *IJCAI*, 2018.

[22] C. Liu, W. Ke, J. Jiao, and Q. Ye, "RSRN: Rich side-output residual network for medial axis detection," in *ICCV Workshop*, 2017, pp. 1739–1743.

[23] Q. Zeng, Y. Yuan, C. Fu, and Y. Zhao, "People detection in crowded scenes using hierarchical features," in *IST*, 2017, pp. 1–5.

[24] Y. Xu, Y. Bao, J. Chen, W. Zuo, and H. Li, "Surface fatigue crack identification in steel box girder of bridges by a deep fusion convolutional neural network based on consumer-grade camera images," *Structural Health Monitoring*, 2018.

[25] J. Canny, "A computational approach to edge detection," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 8, no. 6, pp. 679–698, 1986.

[26] Y. Liu and M. S. Lew, "Learning relaxed deep supervision for better edge detection," in *IEEE Conf. Comput. Vis. Pattern Recog.*, 2016, pp. 231–240.

[27] Y. Wang, X. Zhao, Y. Li, and K. Huang, "Deep crisp boundaries: From boundaries to higher-level tasks," *arXiv preprint arXiv:1801.02439*, 2018.

[28] D. Xu, W. Ouyang, X. Alameda-Pineda, E. Ricci, X. Wang, and N. Sebe, "Learning deep structured multi-scale features using attention-gated CRFs for contour prediction," in *Adv. Neural Inform. Process. Syst.*, 2017, pp. 3961–3970.

[29] Z. Yu, C. Feng, M.-Y. Liu, and S. Ramalingam, "CASENet: Deep category-aware semantic edge detection," in *IEEE Conf. Comput. Vis. Pattern Recog.*, 2017, pp. 21–26.

[30] N. Silberman, D. Hoiem, P. Kohli, and R. Fergus, "Indoor segmentation and support inference from RGBD images," in *Eur. Conf. Comput. Vis.*, 2012, pp. 746–760.

[31] Y. Jia, E. Shelhamer, J. Donahue, S. Karayev, J. Long, R. Girshick, S. Guadarrama, and T. Darrell, "Caffe: Convolutional architecture for fast feature embedding," in *ACM Int. Conf. Multimedia*, 2014, pp. 675–678.

[32] J. Yang, B. Price, S. Cohen, H. Lee, and M.-H. Yang, "Object contour detection with a fully convolutional encoder-decoder network," in *IEEE Conf. Comput. Vis. Pattern Recog.*, 2016, pp. 193–202.

[33] I. Kokkinos, "Pushing the boundaries of boundary detection using deep learning," in *Int. Conf. Learn. Represent.*, 2015.

[34] R. Mottaghi, X. Chen, X. Liu, N.-G. Cho, S.-W. Lee, S. Fidler, R. Urtasun, and A. Yuille, "The role of context for object detection and semantic segmentation in the wild," in *IEEE Conf. Comput. Vis. Pattern Recog.*, 2014, pp. 891–898.

[35] S. Hallman and C. C. Fowlkes, "Oriented edge forests for boundary detection," in *IEEE Conf. Comput. Vis. Pattern Recog.*, 2015, pp. 1732–1740.

[36] G. Bertasius, J. Shi, and L. Torresani, "High-for-low and low-for-high: Efficient boundary detection from deep object features and its applications to high-level vision," in *Int. Conf. Comput. Vis.*, 2015, pp. 504–512.

[37] S. Gupta, P. Arbelaez, and J. Malik, "Perceptual organization and recognition of indoor scenes from RGB-D images," in *IEEE Conf. Comput. Vis. Pattern Recog.*, 2013, pp. 564–571.

[38] S. Gupta, R. Girshick, P. Arbeláez, and J. Malik, "Learning rich features from RGB-D images for object detection and segmentation," in *Eur. Conf. Comput. Vis.*, 2014, pp. 345–360.

[39] J. Pont-Tuset and F. Marques, "Supervised evaluation of image segmentation and object proposal techniques," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 38, no. 7, pp. 1465–1478, 2016.

[40] D. A. Mély, J. Kim, M. McGill, Y. Guo, and T. Serre, "A systematic comparison between visual cues for boundary detection," *Vision Research*, vol. 120, pp. 93–107, 2016.

[41] J. Shi and J. Malik, "Normalized cuts and image segmentation," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 22, no. 8, pp. 888–905, 2000.

[42] D. Comaniciu and P. Meer, "Mean shift: A robust approach toward feature space analysis," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 24, no. 5, pp. 603–619, 2002.

[43] P. F. Felzenszwalb and D. P. Huttenlocher, "Efficient graph-based image segmentation," *Int. J. Comput. Vis.*, vol. 59, no. 2, pp. 167–181, 2004.

[44] Z. Ren and G. Shakhnarovich, "Image segmentation by cascaded region agglomeration," in *IEEE Conf. Comput. Vis. Pattern Recog.*, 2013, pp. 2011–2018.

[45] Q. Zhao, "Segmenting natural images with the least effort as humans." in *Brit. Mach. Vis. Conf.*, 2015, pp. 110.1–110.12.

[46] K.-K. Maninis, J. Pont-Tuset, P. Arbeláez, and L. Van Gool, "Convolutional oriented boundaries," in *Eur. Conf. Comput. Vis.*, 2016, pp. 580–596.