Received 4 April 2023, accepted 17 May 2023, date of publication 23 May 2023, date of current version 1 June 2023.

Digital Object Identifier 10.1109/ACCESS.2023.3279276

RESEARCH ARTICLE

Pixel Difference Unmixing Feature Networks for Edge Detection

SHI-SHUI BAO[1], YOU-RUI HUANG[1,2], JIA-CHANG XU[1], AND GUANG-YU XU[1]
[1]School of Computer Science and Engineering, Anhui University of Science and Technology, Huainan 232001, China
[2]School of Electrical and Opto Electronic Engineering, West Anhui University, Lu'an 237012, China

Corresponding author: You-Rui Huang (hyr628@163.com)

This work was supported in part by the National Natural Science Foundation of China under Grant 61772033.

ABSTRACT The edge detection model based on deep learning significantly improves performance, but its generally high model complexity requires a large pretrained Convolutional Neural Networks (CNNs) backbone, and hence large memory and computing power. To solve this problem, we carefully choose proper components for edge detection, introduce a Multiscale Aware Fusion Module based on self-attention and a feature-unmixing loss function, and propose a lightweight network model, Pixel Difference Unmixing Feature Networks (PDUF). The backbone network of proposed model is designed to adopt skip long-short residual connection and does not use pre-trained weights, and requires straightforward hyper-parameter settings. Extensive experiments on the BSDS, NYUD, and Multi-cue datasets, we found that the proposed model has higher F-scores than current state-of-the-art lightweight models (those with fewer than 1 million parameters) on BSDS500 (ODS F-score of 0.818), NYUDv2 depth datasets (ODS F-score of 0.767) and Multi-Cue dataset (ODS F-score 0.871(0.002)), with similar performance compared with some large models (with about 35 million parameters).

INDEX TERMS Deep learning, edge detection, unmixing feature, lightweight.

I. INTRODUCTION

Edge detection is the basic work of image processing and machine vision. Edge information plays an important role in object recognition [13], [14], image segmentation [15], [16], and medical image processing [5]. The goal of semantic edge detection is to identify pixels belonging to predefined categories. Early edge detection algorithms interpreted the problem as one of low-level feature classification using local image information [1], [2], [3] and manually designed features [7], [8]. With the rapid development of machine learning, some deep learning models have surpassed the recorded results of human perception in terms of image edge detection performance [4], [5], [6]. Although significant progress has been made in edge detection performance, most such work aims to increase detection accuracy (based on metrics such as ODS, OIS, and AP) through a new network structure or loss function. In this case, the network structure is generally very deep, and the number of model parameters is huge, which brings large memory and computing overhead.

However, we believe that overly complex network structures are unnecessary for edge detection. These large models cannot be used in real-time on low memory or computing power devices. Furthermore, edge detection is relatively simple compared to semantic segmentation and object recognition. While feature extractors are required to recognize many different object patterns in semantic segmentation, edge detection only needs to recognize edge or non-edge pixels.

Therefore, we propose a novel architecture called Pixel Difference Unmixing Feature Networks that is both highly accurate and lightweight. The proposed method consists of a backbone network and a side structure, with the backbone network adopting dense residual connection. The parameters of the proposed model are only 4% of BDCN [41], while the ODS scores are almost equal. Figure 2 shows the relationship between detection accuracy and model complexity (size) of some methods based on deep learning.

In summary, the contribution in this paper can be summarized as follow:

The associate editor coordinating the review of this manuscript and approving it for publication was Sudhakar Radhakrishnan.

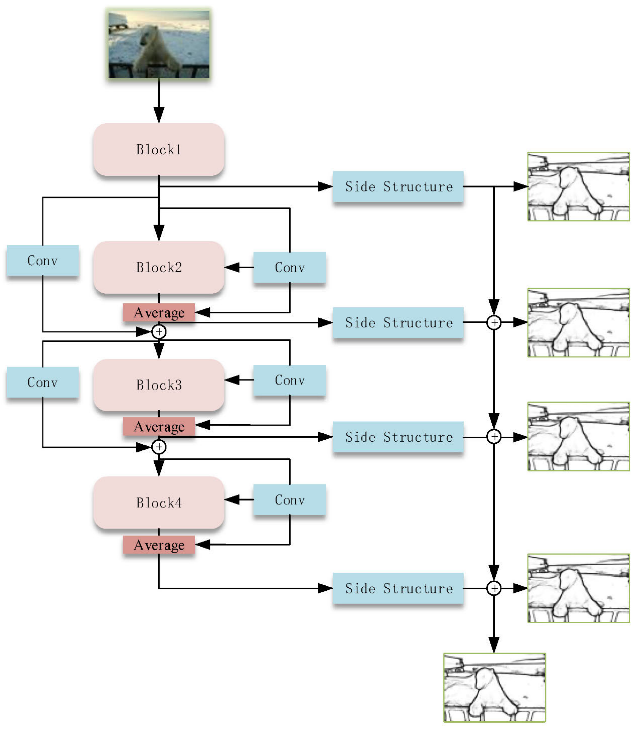

FIGURE 1. The overall flowchart of the proposed method.

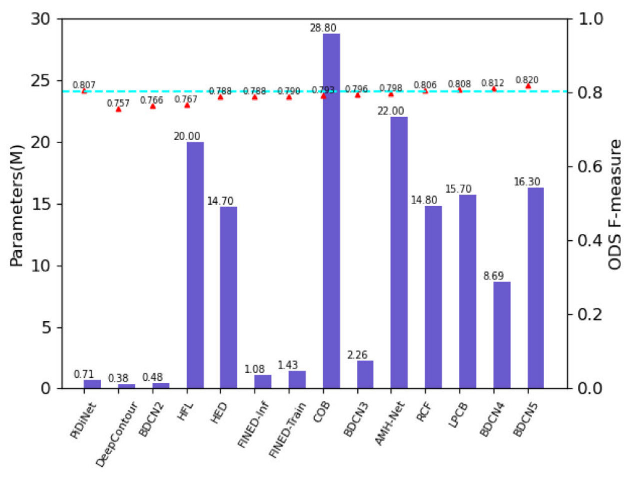

FIGURE 2. Complexity and accuracy performance of various edge detection models.

(1) A lightweight CNN architecture is proposed, which has just 0.66M parameters compared to 16.3M in state-of-the-art method BDCN [41].

(2) We propose a multiscale aware fusion module based on self-attention that can capture receptive fields of different sizes in feature maps.

(3) A novel feature unmixing loss function is designed, which fully exploits their respective advantages of different loss functions, according to the principle of the Nash equilibrium.

The remainder of the paper is organized as follows. Section II reviews the related work of edge detection. Then, Section III details the proposed method; experimental results are presented in Section IV. Finally, Section V provides a summary of the paper.

II. RELATED WORK

Since B. Julez completed his work in 1959 [20], there has much literature on edge detection, and object edge detection has long been of interest. According to the different ways of extracting edge features, these methods can be categorized as edge differential operators, traditional machine learning methods, or deep learning methods. Early edge differential operators mainly used the gray value gradient information between the target object edge and background pixel, such as the Canny [2], Sobel [13], or Laplace [30] operators, which use the first- and second-order information of the image to achieve edge detection. Canny [2] introduced non-maximum suppression methods to edge detection, and these have become representative of such methods. Kumawat et al. proposed a feature-based image registration (FBIR) method that combines fuzzy logic improved Canny algorithm to achieve accurate detection of image edges [46]. However, because this kind of method relies on the manual design of detection operators, the accuracy on a test set is low.

Traditional machine learning methods usually use the features of manual design for supervised learning [3], [17], [18], [19], [22], [23]. P.Dollár et al. proposed a fast random structure forest method [23], making full use of local edge templates for prediction. Martin et al. [22] proposed the Pb algorithm, using the low-level image brightness (BG), texture (CG), and color (TG) channel features, and a logical regression algorithm. Arbeláez et al. proposed the gPb algorithm [3], which gives each pixel a global probability by calculating global features on the basis of multiscale Pb. However, even if complex feature representation is designed, the edge information extracted from these methods is still relatively limited.

Deep learning is the popular method of edge detection, especially the convolutional neural network, because the neural network model has the ability of hierarchical feature learning. Early deep learning-based methods, such as N4-ield [24], DeepContour [25], and DeepEdge [26], used block-by-block or pixel-by-pixel strategies, which usually weakened the effectiveness of prediction. Xie et al. proposed the first end-to-end edge detection algorithm, HED [27], which includes encoding and decoding networks and uses backpropagation for end-to-end training. Liu et al. proposed the RCF model [28] on the basis of an HED network [27], which performs feature compression for each convolution layer with the same resolution, greatly improving the ability to express image features. Wang et al. proposed the CED [29] network model, added a residual enhancement module on the basis of the HED [27] network structure, and gradually restored the image resolution using sub-pixel convolution, which improved edge performance. LPCB [4] improved the loss function, and BDCN [41] trained the neural network using bidirectional multiscale features to improve edge detection accuracy. A new boundary tracking and texture suppression loss function was proposed in CATS [11], which effectively solved the problem of mixed pixels and texture regions near edges. In addition to the above methods, ContourGan [39], which is based on a GAN network, as well as the Transformer based EDTER [40] and Dexined [5], [18] models, have been recently proposed.

The abovementioned models either have low feature extraction efficiency or require a pretrained backbone network due to their large size, which cannot meet the real-time application of devices with low computing power. These shortcomings have promoted the development of lightweight models (usually with fewer than 1M parameters), which have achieved satisfactory results. Literature [10] proposed pixel difference convolution, whose normal version has only 710K parameters, but the F-measures of OIS and ODS reach 0.823 and 0.807, respectively. The FINED [33] and LDC [38] lightweight network models show improved edge detection performance.

III. PROPOSED METHOD

We describe the architecture of the proposed network model and the structure of the multiscale aware fusion module, as shown in Figures 3 and 4, respectively. The model consists of a feature extraction backbone network and four side output subnetworks. In addition, we introduce a feature unmixing loss function training network model.

A. BACKBONE NETWORK

The backbone network is designed to maintain the thinnest possible structure, to effectively extract target object features. Therefore, we divide it into four stages, each composed of four 3×3-pixel difference convolutions (the first layer of the first stage uses the initial convolutional layer). The last layer of 2 to 3 stages is the maximum pool layer (Maxpool 2×2). Each convolution layer is followed by batch normalization and a corrected linear unit ReLU function. With the increase in stage number, the dimension of the output feature map decreases, and the number of channels increases. We set the number of channels in stages 1 to 4 to reasonable sizes of 16, 32, 64 and 96, respectively, to effectively avoid large model sizes. To eliminate gradient disappearance of the backbone network with increasing numbers of convolution layers, we use a dense residual connection [31], [32], which can improve model training efficiency without increasing the number of parameters. Dense residuals use skip long-short residual connection methods. The skip short residual connection is based on the standard ResNet [31] residual connection, and no new parameters are added. We divide each of stages 1 to 4 into two blocks, each composed of two 3×3-pixel difference convolution layers, and each block (except the first block of stage 1) adopts a skip short residual connection, in which the output feature map of each block and the features transmitted from the skip short residual are averaged. A skip long residual connection is used between the input features of the second and third stages and the output features of the Max pooling layer. Since pixel-wise summation is adopted at the skip long residual connection, the two feature map dimensions must be consistent. Therefore, we add a 3×3-pixel difference convolution layer with stride 2 in the skip long residual connection path. The skip long-short residual connection effectively enhances the transmission of feature information and the use of hierarchical features in the training process.

B. SIDE STRUCTURE

We adopt the side output structure to learn and enrich multi-level features [27]. The model has four side output paths, corresponding to the stages of the backbone network. The input of a side path is from the average feature map of each stage, and the output generates an edge map and real edge labels for deep supervision training through the side loss function. To capture receptive fields of different sizes in feature maps, we propose a multiscale aware fusion module based on self-attention, which is embedded in each stage. Then sub-pixel convolution is used to restore the feature map to the original image size, and a 1×1 convolution layer reduces the feature map to a single channel and obtains the edge map through a sigmoid function. The final edge map for testing is obtained by fusing the four single-channel feature maps with concatenation, and then through 1×1 convolution and sigmoid functions.

Multiscale Aware Fusion Module: Figure 4 shows the structure of the MSAF module. First, we use the 1×1 conv + ReLU layer to receive and process the feature map of the backbone network output. To make full use of low-level edge map details and high-level edge global context information with robust texture suppression, we use four parallel arranged dilation convolutions to set the division sizes to 5, 7, 9, and 11. In PiDiNet [10] and FINED [33], the same fixed weight operation was used to fuse the feature map from different size dilation convolutions. Although effective, this averaging operation will cause side mixing problems because all pixels in the side image share the same fixed weight and are equally important in the fusion process. To avoid feature mixing and make full use of multi-level features of the dilation convolution output, inspired by CAT [11], the self-attention mechanism is used in the feature fusion process to give each output feature different weights. The number of MSAF receiving channels is the output channel corresponding to each stage. The number of output channels generated is uniformly set to 32 filters, which can reduce the computational complexity.

Let $\mathfrak{L} = [\mathfrak{L}_1, \mathfrak{L}_2, \cdots, \mathfrak{L}_s] \in \boldsymbol{R}^{H \times W \times S}$, $\mathfrak{L}_s \in \boldsymbol{R}^{H \times W \times C}$, where $\mathfrak{L}$ represents edge heatmaps of dilation convolution output. The self-attention module adopts two 3×3conv + ReLU and one 1×1conv + softmax, to derive a score map $\boldsymbol{M} \in [m_{ijs}] \boldsymbol{R}^{H \times W \times S}$. The weight map $\boldsymbol{W} = [\boldsymbol{w}_1, \boldsymbol{w}_2, \cdots, \boldsymbol{w}_s] \in \boldsymbol{R}^{H \times W \times S}$ is calculated by the score map, where

$$[w_{ijs}] = e^{m_{ijs}} / \sum_{k=1}^{S} e^{m_{ijk}} \quad (1)$$

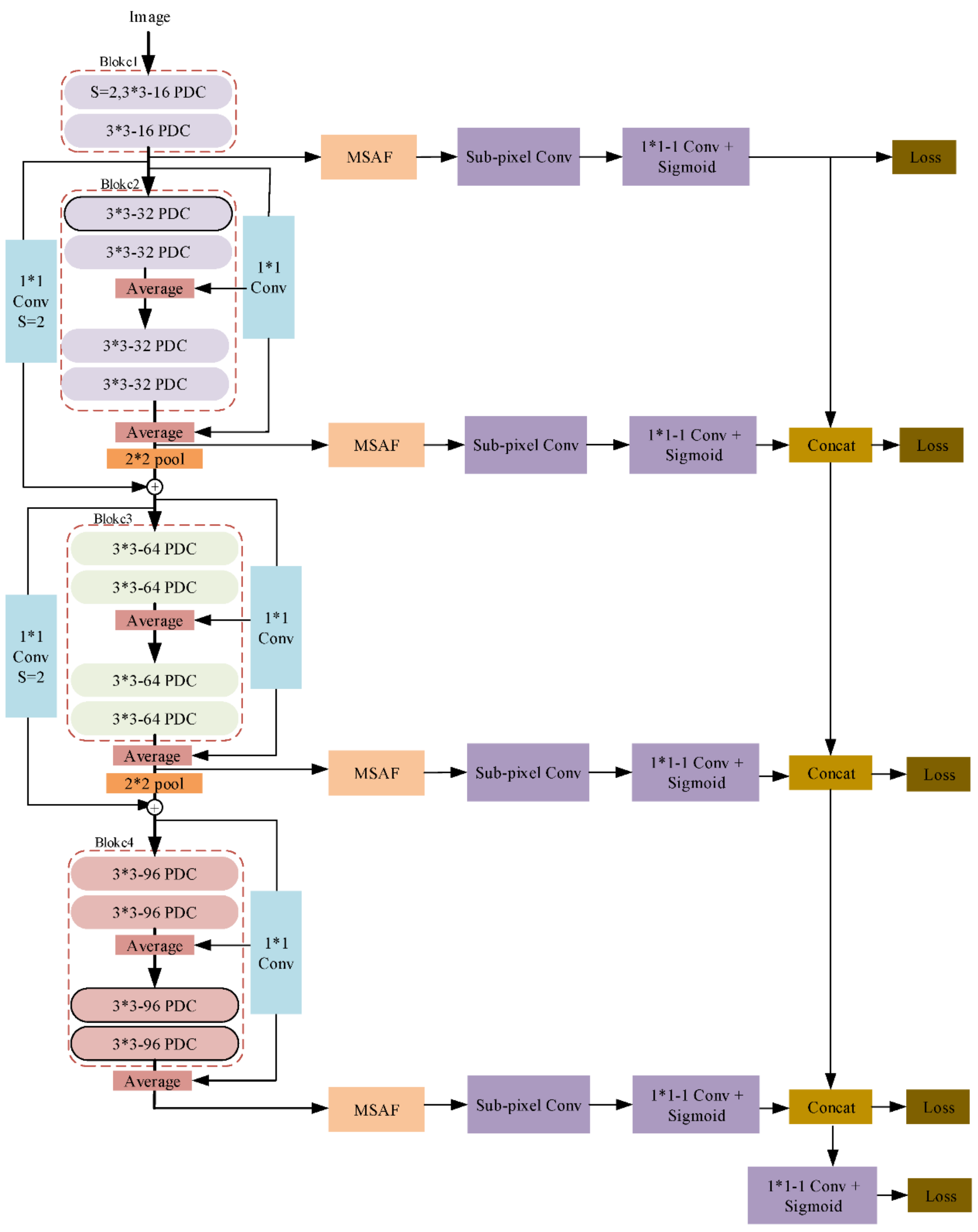

FIGURE 3. Overall architecture of PDUF network model.

With the weight map, $W \in R^{H\times W\times S}$, p_{final} can be calculated as

$$p_{final} = sigmoid\left(\sum_{s=1}^{S} W \otimes \mathfrak{L}_s\right) \tag{2}$$

where $\otimes$ denotes the Hadamard product.

C. PIXEL DIFFERENCE CONVOLUTION

In the task of edge detection, to easily extract rich gradient information, reduce the processing of irrelevant image features, and ensure that the deep CNN can learn meaningful semantic expressions, PiDiNet [10] proposes using the pixel difference convolution (PDC) method. In this paper, we adopt the pixel difference convolution. It is similar to vanilla convolution, replacing vanilla convolution kernels with pixel difference convolutions when convolving the original pixels on local image features. Vanilla convolution and PDC can be formulated as

$$y = \mathfrak{F}(x,\theta) = \sum_{i=1}^{k\times k} \omega_i * x_i \quad \text{(vanilla convolution)} \tag{3}$$

$$y = \mathfrak{F}(\nabla x,\theta) = \sum_{(x_i,x_i')\in p} \omega_i * (x_i - x_i') \quad \text{(PDC)} \tag{4}$$

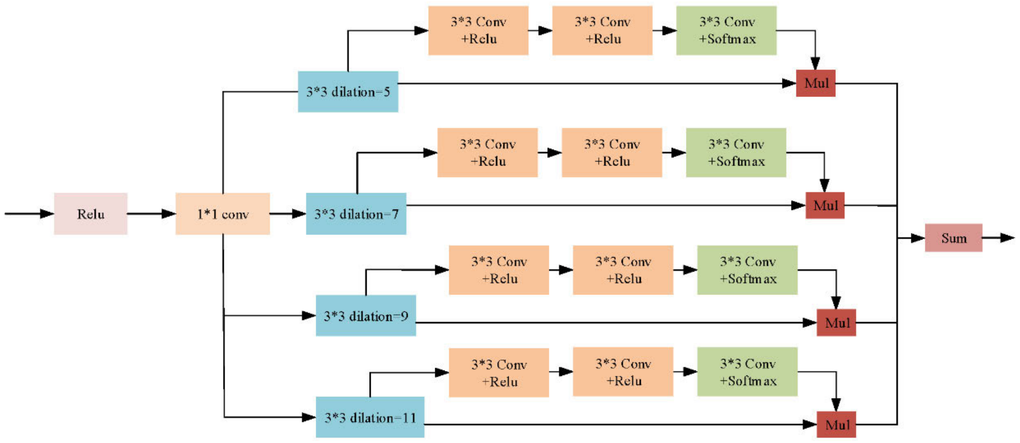

FIGURE 4. Structure of multiscale aware fusion model.

where x_i and x_i' are the input pixels, ω_i is the weight in the $k * k$ convolution kernel $p = \left\{ (x_1, x_1'), (x_2, x_2'), \cdots, (x_m, x_m') \right\}$ is the set of pixel pairs picked from the current local patch, and $m \leq k \times k$.

D. FEATURE UNMIXING LOSS FUNCTION

The loss function is an important part of an end-to-end CNN training model, as it directly affects the final prediction results. Therefore, when designing the loss function, we must pay attention to suppression of confusing pixels and texture regions near real edges, serious imbalance between edge and non-edge pixels, accuracy of detection results, and fusion of multiple loss functions. Based on these considerations, inspired by BMRN [7], CAT [11], HED [27], and LPCB [4], we propose a feature unmixing loss function that combines cross-entropy loss, Dice efficiency loss, the boundary tracing function, and the texture suppression function through adaptive weights, and improves the performance of edge detection.

To effectively solve the serious imbalance between edge pixels and non-edges in the input image, HED [27] uses a weight cross-entropy loss function,

$$\mathcal{L}_c (W, w) = -\gamma \sum_{j \in Y_+} logPr(y_j = 1|X; W, w) - (1-\gamma) \sum_{j \in Y_-} logPr(y_j = 0|X; W, w) \quad (5)$$

where Y_+ and Y_- represent edge and non-edge pixels, respectively; $\gamma = |Y_-| / |Y|$ and $1 - \gamma = |Y_+| / |Y|$; X represents the input image, and $Pr(y_j|X; W, w)$ is the classification probability obtained after pixel y_j passes the Softmax function.

Deng et al. [4] found through experiments that the edge of cross-entropy loss function detection was relatively thick, so they proposed a Dice coefficient loss function,

$$\mathcal{L}_d (P, G) = Dist (P, G) = \frac{\sum_i^N p_i^2 + \sum_i^N g_i^2}{2 \sum_i^N p_i g_i} \quad (6)$$

where p_i and g_i are the values of the i-th pixel of the predicted P and G real label graphs, respectively. Since the Dice coefficient loss function can be used to measure the approximation of two datasets, the loss function is used to calculate the approximation of P and G and minimize their distance through training data. The Dice coefficient loss function does not need to consider the imbalance between edge and non-edge pixels and can obtain crisp edges through training. It is considered to be a picture-level similarity measurement function.

Boundary tracing and texture suppression functions have been proposed to solve the problem of confusing pixels and texture regions near edges CAT [11]. The boundary tracking function weakens the interaction between the edge and confused neighbors by expanding the response difference of crisp edge description. It is expressed as

$$\mathcal{L}_b (\hat{Y}, Y) = -\sum_{p \in E} log \left(\sum_{i \in L_p} \hat{y}_i / \left(\sum_{i \in R_p^e \setminus L_p} \hat{y}_i + \sum_{i \in L_p} \hat{y}_i \right) \right) \quad (7)$$

where E is the set of all edge points of edge label Y, R_p^e is a small image patch containing edge fragments (e.g., a 5×5 rectangle patch), and the small image block is centered on the edge point p. The set of edge points in image R_p^e is represented by L_p.

The boundary tracking function solves the problem of the mixing of edge and confusion pixels. The texture suppression function effectively suppresses the remaining texture regions to make them smoother,

$$\mathcal{L}_t (\hat{Y}, Y) = -\sum_{p \in Y \setminus \hat{E}} log \left(1 - \sum_{i \in R_p^t} \hat{y}_i / \left| R_p^t \right| \right) \quad (8)$$

where R_p^t represents an image patch that centered on non-edge p (e.g., a 3×3 rectangle), and $\hat{E}$ represents a small image patch that contains all edges and confusion pixels in the edge tracking function. Based on the analysis of the above-mentioned loss functions, to fully exploit their respective

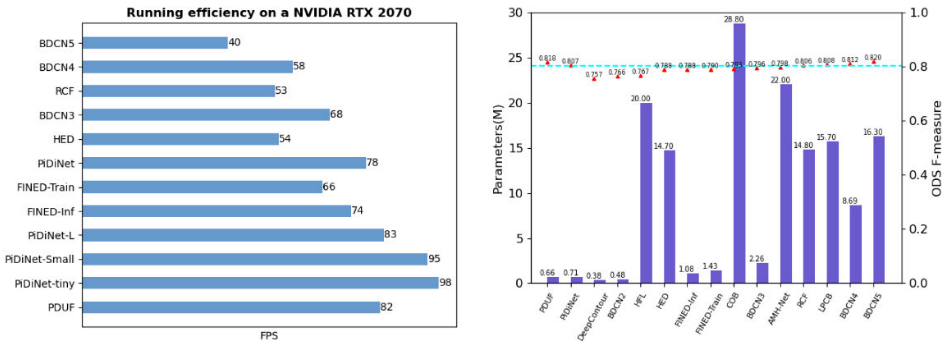

FIGURE 5. Comparison with other methods in terms of network complexity, running efficiency, and detection performance on the BSDS500 dataset. Running speeds of FINED [33] are cited from the original paper; the rest are evaluated by our implementations.

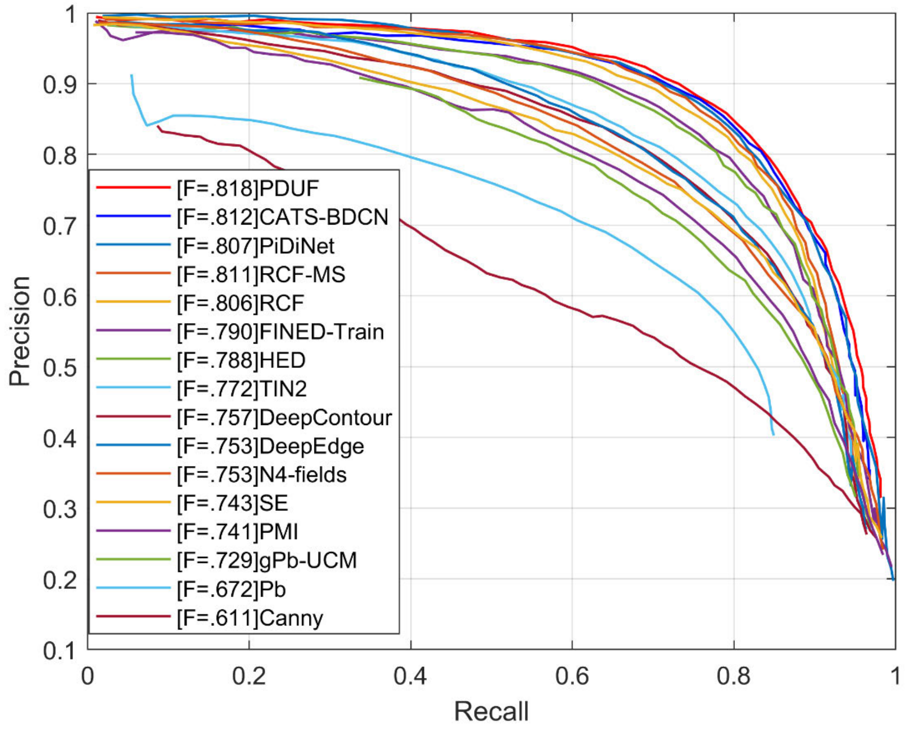

FIGURE 6. P–R curves of different algorithms on BSDS500; our method achieves state-of-the-art results (ODS F = 0.818).

advantages, according to the principle of the Nash equilibrium, $\boldsymbol{\alpha}\mathcal{L}_c + \boldsymbol{\beta}\mathcal{L}_d + \boldsymbol{\lambda}\mathcal{L}_b + \boldsymbol{\theta}\mathcal{L}_t$ is taken as one side of a game, and the term $\boldsymbol{log}\left(\mathbf{1} + \frac{\mathbf{1}}{\boldsymbol{\alpha}+\boldsymbol{\beta}+\boldsymbol{\lambda}+\boldsymbol{\theta}}\right)$, opposite the original cost function, is constructed as the other side. Through end-to-end training of the network model, we obtain the weight coefficients $\boldsymbol{\alpha}$, $\boldsymbol{\beta}$, $\boldsymbol{\lambda}$, and the $\boldsymbol{\theta}$ values reach Nash equilibrium. Then we construct an adaptive weight loss feature unmixing loss function,

$$\mathcal{L}(\boldsymbol{P}, \boldsymbol{Y}) = \boldsymbol{\alpha}\mathcal{L}_c + \boldsymbol{\beta}\mathcal{L}_d + \boldsymbol{\lambda}\mathcal{L}_b + \boldsymbol{\theta}\mathcal{L}_t + \boldsymbol{log}\left(\mathbf{1} + \frac{\mathbf{1}}{\boldsymbol{\alpha}+\boldsymbol{\beta}+\boldsymbol{\lambda}+\boldsymbol{\theta}}\right) \tag{9}$$

where $\mathcal{L}_c$, $\mathcal{L}_d$, $\mathcal{L}_b$, and $\mathcal{L}_t$ correspond to the loss functions defined in equations (5)–(8), respectively. In the training process, larger values of $\boldsymbol{\alpha}$, $\boldsymbol{\beta}$, λ, and $\boldsymbol{\theta}$ increase the contribution of $\boldsymbol{\alpha}\mathcal{L}_c + \boldsymbol{\beta}\mathcal{L}_d + \boldsymbol{\lambda}\mathcal{L}_b + \boldsymbol{\theta}\mathcal{L}_t$. The last term, $\boldsymbol{log}\left(\mathbf{1} + \frac{\mathbf{1}}{\alpha+\beta+\lambda+\theta}\right)$, is the regularization term of parameters $\boldsymbol{\alpha}$, $\boldsymbol{\beta}$, λ, and $\boldsymbol{\theta}$.

IV. EXPERIMENT

We introduce the datasets and hyperparameter settings. We used ablation experiments to verify the effectiveness and versatility of the proposed method, so as to show the influence of each sub-module and loss function on overall network performance. Through further experimental comparison with

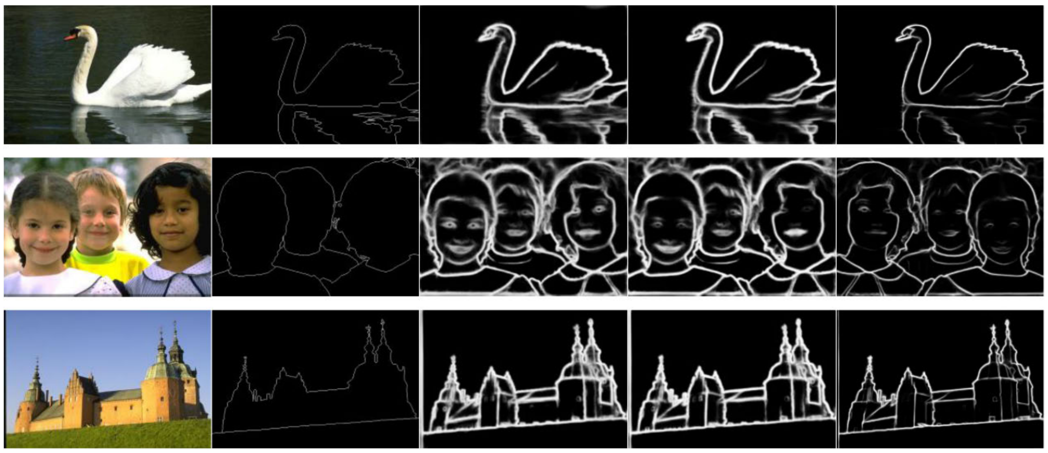

FIGURE 7. Qualitative comparison of different detection algorithms on BSDS500. Left to right: original image, ground truth, HED detection result, CED detection result, our method's detection result.

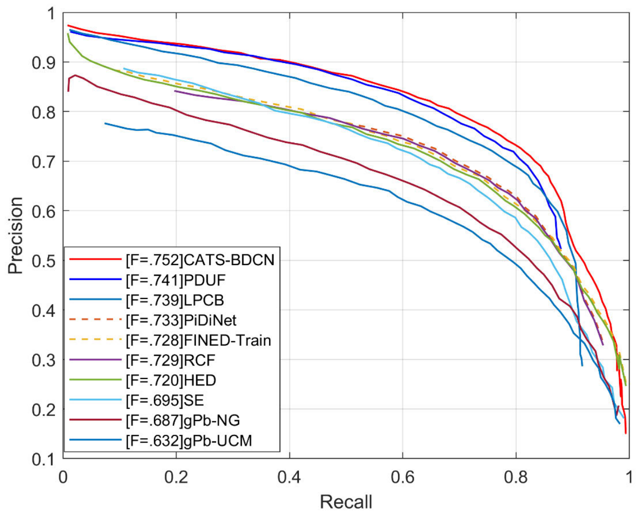

FIGURE 8. Our method achieves best results (F = 0.782) on the NYUDv2 dataset for P–R curves of different algorithms.

state-of-the-art algorithms, the proposed method was verified to better extract crisp edges of objects.

A. DATASETS

We used the popular Berkeley segmented dataset BSDS500 [3], deep dataset NYUDv2 [24], and Multi-cue dataset [18] to train and evaluate our model.

BSDS500 contains 500 image labels, which we divided into training, verification, and test sets, which, respectively, contained 200, 100, and 200 image-label pairs. To avoid insufficient training and poor robustness of the model due to a lack of pictures in the training set, we augmented it to realize more than 10,000 image-label pairs using the methods proposed in HED [27] and RCF [28].

NYUDv2 includes 1449 RGB-D images, which were divided into 381 image-tag pairs as training sets, 414 as verification sets, and 654 as test sets. We augmented the dataset by rotating each image at four different angles (0°, 90°, 180°, and 270°), and flipping it at each angle.

Multi-cue strictly distinguishes the definitions of boundary and edge and is composed of the Multi-cue Boundary and Multi-cue Edge sub-datasets. This dataset regards semantically meaningful pixels as object boundary points and treats pixels with abrupt perceptional changes as low-level edge points. The dataset contains 100 natural scenes, each with two frame sequences, obtained from the left and right view, and the last frame of the left view sequence is labeled as the edge and boundary. According to the HED [27] and RCF [28] methods, we randomly split 100 labeled scene images, using 80 for training and 20 for testing.

B. EXPERIMENT DETAILS

We adopted the popular open-source PyTorch 1.8.0 framework for deep learning [36] to implement the proposed neural network model. The parameters of the backbone network were randomly initialized and trained using the Adam optimizer. Other hyperparameters were set as follows: the minimum batch size was 10; the momentum was 0.9; the total number of iterations on the training set was 20; and the global learning rate was initially set to 0.005, and decayed in a multistep way (at epochs 5, 10, and 15, with decaying rate 0.1). The initial values of trainable parameters $\boldsymbol{\alpha}$, $\boldsymbol{\beta}$, $\boldsymbol{\lambda}$, and $\boldsymbol{\theta}$ were all set to 1.0. All experiments were performed on an NVIDIA 2070 video card with 8 GB of RAM.

According to previous work, the final edge was obtained by performing standard non-maximum suppression on the predicted edge. To compare the performance of different algorithms, we adopted the widely used evaluation criteria of Average Precision (AP) and F-measure at both the Optimal Dataset Scale (ODS) and Optimal Image Scale (OIS) [34], [35]. To make the edge prediction value match the ground truth, the maximum tolerance distance was set to 0.0075 on BSDS500 and Multi-cue, and 0.011 [29] on NYUDv2.

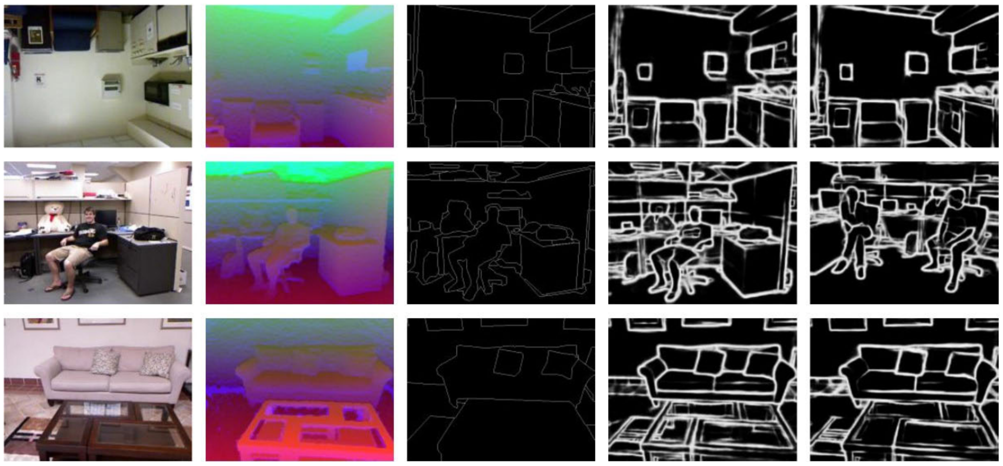

FIGURE 9. Comparison of our method with state-of-the-art PiDiNet on NYUDv2. Left to right: RGB image, HHA features, ground truth, PiDiNet experimental results, PDUF experimental results.

TABLE 1. Performance comparison between pixel different convolutions (PDC) and vanilla convolutions (VC) on BSDS500 test set.

Method	ODS	OIS	AP
VC	0.812	0.828	0.855
PDC	**0.818**	**0.836**	**0.862**

TABLE 2. Quantitative study of impact of different dilation convolutions on overall network performance. CDCM and DDR are dilation convolution modules proposed in PiDiNet and FINED methods, respectively.

Method	ODS	OIS	AP
PDUF+DDR	0.811	0.823	0.853
PDUF+CDCM	0.806	0.812	0.848
PDUF+MSAF	**0.818**	**0.836**	**0.862**

C. ABLATION EXPERIMENTS

We analyzed the effect of PDC and the proposed algorithm sub-module and loss function on the performance of the overall network model through ablation experiments, using the BSDS500 training and verification sets to train the network model, and the test set to evaluate model performance. The results are summarized in Table 1, which shows that the pixel different convolution operations in the backbone network can achieve better output results. Table 2 shows the impact of different dilation convolution modules on the algorithm performance in the proposed network model. It can be found that the proposed multiscale Aware Fusion module performs better in edge detection than the previous dilation convolution module [10], [33]. In addition, for the different combinations of the proposed feature unmixing loss function, a comparative experiment was carried out on the proposed PDUF model, with results as shown in Table 3, from which it can be seen that the proposed feature unmixing loss function has better output results. Other parameters were set the same during ablation experiments.

TABLE 3. Effect of proposed loss function on performance of our network models. L_c: Cross loss function; L_d: Dice coefficient loss function; $L_t + L_b$: Tracking loss function.

L_c	L_d	L_t+L_b	ODS/OIS
√	×	×	0.802/0.816
×	√	×	0.806/0.821
×	×	√	0.812/0.828
√	√	√	**0.818/0.836**

D. COMPARISON WITH STATE-OF-THE-ART

We statistically compared the performance of the proposed method and state-of-the-art algorithms on BSDS500, NYUDv2, and Multi-cue.

1) PERFORMANCE COMPARISON ON BSDS500

We compared the proposed method with some previous non-deep learning algorithms and recent deep learning-based edge detection algorithms, including N4-field [24], DeepContour [25], DeepEdge [26], HED [27], RCF [28], BDCN [41], CED [29], BMRN [7], FINED [33], PiDiNet [10], CATS [11], and traditional edge detection algorithms Canny [2], gPb [3], SE [23], and PMI [42]. Table 4 shows the experimental results, with ODS, OIS, and AP as the evaluation criteria. The proposed method achieves better results (ODS = 0.818, OIS = 0.836, AP = 0.862) than the state-of-the-art lightweight PiDiNet [10] model and exceeds the human standard on BSDS500 (ODS = 0.803). Figure 5 compares the model complexity, operational efficiency, and detection performance of the proposed method and state-of-the-art methods, from which it can be found that the proposed method has better comprehensive performance than methods using the same level parameter. Figure 6 shows the precision–recall curve of the experimental results. From the qualitative experimental results in Figure 7, it can be seen

TABLE 4. Performance of ours and other methods on BSDS500. † indicates cited GPU speeds; ‡ indicates speeds with our implementations on NVIDIA RTX 2070 GPU.

Method	ODS	OIS	AP	FPS
Canny	0.611	0.676	0.520	28
gPb	0.729	0.755	0.745	1/240
SE	0.743	0.764	0.800	2.5
PMI	0.741	0.769	0.799	-
N^4-fields	0.753	0.772	0.802	1/5†
DeepEdge	0.753	0.772	0.787	1/1000†
DeepContour	0.757	0.776	0.790	1/30†
HED	0.788	0.808	0.840	54‡
RCF	0.806	0.823	-	53‡
CED	0.803	0.820	0.871	30†
LPCB	0.808	0.816	-	30†
BDCN	0.806	0.826	0.847	40‡
TIN1	0.749	0.772	-	30†
TIN2	0.772	0.795	-	30†
FINED-Inf	0.788	0.804		74‡
FINED-Train	0.790	0.808		66‡
PiDiNet	0.807	0.823	-	78‡
CATS-BDCN	0.812	0.828	0.851	39‡
PDUF	**0.818**	**0.836**	**0.862**	**82‡**

that the proposed algorithm extracts finer and crisper contours than PiDiNet [10] and CATS [11].

2) PERFORMANCE COMPARISON ON NYUDV2

NYUDv2 is composed of two sub-datasets, RGB and HHA, on which all experiments were carried out. The final edge map of RGB-HHA is obtained from the prediction results of the average RGB and HHA training models. Comparing the proposed method with several non-deep learning models (e.g., gPb-UCM [3], gPb-NG [43], SE [23]) and deep learning models (e.g., HED [27], RCF [28], AMH-Net-ResNet [44], LPCB [4], BDCN [41], FINED [33], PiDiNet [10], CATS [11]), all experiments were based on single-scale input. The quantitative experimental results are shown in Table 5, from which we can find that the proposed model has higher F-scores than current state-of-the-art lightweight models, with similar performance compared with some large models. Figure 8 shows the precision–recall curves of different algorithms. Figure 9 qualitatively shows the experimental results of the proposed method and the lightweight models FINED [33] and PiDiNet [10]. Our prediction results show crisper edges than the lightweight models.

3) PERFORMANCE COMPARISON ON MULTI-CUE

The Multi-cue dataset is composed of the Multi-cue Boundary and Multi-cue Edge sub-datasets, on which the proposed

TABLE 5. Quantitative comparison between proposed method and other methods on NYUDv2. † indicates cited GPU speeds; ‡ indicates speeds with our implementations based on an NVIDIA RTX 2070 GPU.

Method		ODS	OIS	AP	FPS
gPb-UCM		0.632	0.661	-	1/365
gPb-NG		0.687	0.716	0.629	1/375
SE		0.695	0.708	0.679	5
HED	RGB	0.720	0.734	0.734	20†
	HHA	0.682	0.695	0.702	20†
	RGB-HHA	0.746	0.761	0.786	10†
RCF	RGB	0.729	0.742	-	20†
	HHA	0.705	0.715	-	20†
	RGB-HHA	0.757	0.771	-	10†
AMH-Net-ResNet50	RGB	0.744	0.758	0.765	-
	HHA	0.716	0.729	0.734	-
	RGB-HHA	**0.771**	**0.786**	0.802	-
LPCB	RGB	0.739	0.754	-	30†
	HHA	0.707	0.719	-	30†
	RGB-HHA	0.762	0.778	-	15†
BDCN	RGB	0.748	0.763	0.770	25‡
	HHA	0.707	0.719	0.731	25‡
	RGB-HHA	0.765	0.781	0.813	13‡
CATS-BDCN	RGB	0. 752	0. 765	0.776	26‡
	HHA	0. 712	0. 724	0.738	26‡
	RGB-HHA	0. 770	0. 783	0.824	13‡
FINED-Inf	RGB	0.721	0.736	-	34‡
	HHA	0.703	0.712	-	34‡
	RGB-HHA	0.748	0.762	-	17‡-
FINED-Train	RGB	0.728	0.742	-	32‡
	HHA	0.712	0.724	-	32‡
	RGB-HHA	0.752	0.769	-	16‡
PiDiNet	RGB	0. 733	0. 747	-	35‡
	HHA	0. 715	0. 728	-	35‡
	RGB-HHA	0. 756	0. 773	-	18‡
PDUF	RGB	0. 741	0. 756	0.772	40‡
	HHA	0. 718	0. 732	0.741	40‡
	RGB-HHA	0. 767	0. 782	0.812	**20‡**

method, HED [27], RCF [28], BDCN [41], FINED [33], and PiDiNet [10] were independently tested. Table 6 compares the experimental results. Our algorithm achieves higher performance than the lightweight models FINED [33] and PiDiNet [10]. In the task of edge detection, our method performance is 2.96% and 1.87% higher than that of FINED [33] and PiDiNet [10] in terms of F-measure ODS, respectively.

TABLE 6. Quantitative comparison between proposed and recent methods on Multi-Cue. ‡ indicates speeds with our implementations based on NVIDIA RTX 2070 GPU.

Cat	Method	ODS	OIS	AP	FPS
Boundary	Human	0.760(0.017)	-	-	-
	Multicue	0.720(0.014)	-	-	-
	HED	0.814(0.011)	0.822(0.008)	0.869(0.015)	12‡
	RCF	0.817(0.004)	0.825(0.005)	-	10‡
	BDCN	0.836(0.001)	0.846(0.003)	**0.893(0.001)**	7‡
	FINED	0.813(0.002)	0.822(0.003)	-	12‡
	PiDiNet	0.818(0.003)	0.830(0.005)	-	11‡
	PDUF	**0.828(0.002)**	**0.846(0.004)**	0.885(0.002)	16‡
Edge	Human	0.750(0.024)	-	-	-
	Multicue	0.830(0.002)	-	-	11‡
	HED	0.851(0.014)	0.864(0.011)	-	9‡
	RCF	0.857(0.004)	0.862(0.004)	-	7‡
	BDCN	**0.891(0.001)**	**0.898(0.002)**	**0.935(0.002)**	6‡
	FINED	0.846(0.004)	0.851(0.003)	-	11‡
	PiDiNet	0.855(0.007)	0.860(0.005)	-	10‡
	PDUF	0.871(0.002)	0.875(0.003)	0.932(0.001)	**18‡**

V. CONCLUSION

We conducted experiments on the BSDS500, NYUD, and Multi-cue datasets, analyzed edge detection models with different weight levels, and proposed a robust, straightforward, and practical lightweight network model that is memory-friendly and has a high inference speed. We introduced a multiscale aware fusion module, which obtains rich multi-scale edge information at different stages. To fully exploit the advantages of different loss functions, we proposed a feature unmixing loss function, which fuses multiple loss functions to effectively solve the problem of prediction edge blurring caused by feature pixel mixing and the imbalance of edge and non-edge pixels. The proposed model has fewer than 0.66 million parameters and can achieve state-of-the-art results comparable to those of current lightweight edge detection models.

REFERENCES

[1] J. M. Prewitt, "Object enhancement and extraction," *Picture Process. Psychopictorics*, vol. 10, no. 1, pp. 15–19,1970.

[2] J. Canny, "A computational approach to edge detection," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. PAMI-8, no. 6, pp. 679–698, Nov. 1986, doi: 10.1109/TPAMI.1986.4767851.

[3] 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, May 2011, doi: 10.1109/TPAMI.2010.161.

[4] R. Deng, C. Shen, S. Liu, H. Wang, and X. Liu, "Learning to predict crisp boundaries," in *Computer Vision ECCV 2018*, vol. 11210, V. Ferrari, M. Hebert, C. Sminchisescu, Y. Weiss, Eds. Cham, Switzerland: Springer, 2018, pp. 570–586, doi: 10.1007/978-3-030-01231-1_35.

[5] X. Soria, E. Riba, and A. Sappa, "Dense extreme inception network: Towards a robust CNN model for edge detection," in *Proc. IEEE Winter Conf. Appl. Comput. Vis. (WACV)*, Mar. 2020, pp. 1912–1921, doi: 10.1109/WACV45572.2020.9093290.

[6] X. Du, Y. Nie, F. Wang, T. Lei, S. Wang, and X. Zhang, "AL-Net: Asymmetric lightweight network for medical image segmentation," *Frontiers Signal Process.*, vol. 2, p. 25, May 2022, doi: 10.3389/frsip.2022.842925.

[7] S. Bao, Y. Huang, and G. Xu, "Bidirectional multiscale refinement network for crisp edge detection," *IEEE Access*, vol. 10, pp. 26282–26293, 2022, doi: 10.1109/ACCESS.2022.3146339.

[8] S. Hallman and C. C. Fowlkes, "Oriented edge forests for boundary detection," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2015, pp. 1732–1740, doi: 10.1109/CVPR.2015.7298782.

[9] J. K. Wibisono and H.-M. Hang, "Traditional method inspired deep neural network for edge detection," in *Proc. IEEE Int. Conf. Image Process. (ICIP)*, Oct. 2020, pp. 678–682, doi: 10.1109/ICIP40778.2020.9190982.

[10] Z. Su, W. Liu, Z. Yu, D. Hu, Q. Liao, Q. Tian, M. Pietikäinen, and L. Liu, "Pixel difference networks for efficient edge detection," in *Proc. IEEE/CVF Int. Conf. Comput. Vis. (ICCV)*, Oct. 2021, pp. 5097–5107, doi: 10.1109/ICCV48922.2021.00507.

[11] L. Huan, N. Xue, X. Zheng, W. He, J. Gong, and G. Xia, "Unmixing convolutional features for crisp edge detection," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 44, no. 10, pp. 6602–6609, Oct. 2022, doi: 10.1109/TPAMI.2021.3084197.

[12] L. Liu, W. Ouyang, X. Wang, P. Fieguth, J. Chen, X. Liu, and M. Pietikäinen, "Deep learning for generic object detection: A survey," *Int. J. Comput. Vis.*, vol. 128, no. 2, pp. 261–318, Feb. 2020, doi: 10.1007/s11263-019-01247-4.

[13] J. Kittler, "On the accuracy of the Sobel edge detector," *Image Vis. Comput.*, vol. 1, no. 1, pp. 37–42, 1983, doi: 10.1016/0262-8856(83)90006-9.

[14] 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, Jan. 2008, doi: 10.1109/TPAMI.2007.1144.

[15] G. Bertasius, J. Shi, and L. Torresani, "Semantic segmentation with boundary neural fields," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2016, pp. 3602–3610, doi: 10.1109/CVPR.2016.392.

[16] 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 *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jun. 2014, pp. 891–898, doi: 10.1109/CVPR.2014.119.

[17] P. Dollar, Z. Tu, and S. Belongie, "Supervised learning of edges and object boundaries," in *Proc. IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit. (CVPR)*, vol. 2, Jun. 2006, pp. 1964–1971, doi: 10.1109/CVPR.2006.298.

[18] X. Soria, A. Sappa, P. Humanante, and A. Akbarinia, ''Dense extreme inception network for edge detection,'' 2021, *arXiv:2112.02250*.

[19] J. J. Lim, C. L. Zitnick, and P. Dollar, ''Sketch tokens: A learned mid-level representation for contour and object detection,'' in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jun. 2013, pp. 3158–3165, doi: 10.1109/CVPR.2013.406.

[20] B. Julesz, ''A method of coding television signals based on edge detection,'' *Bell Syst. Tech. J.*, vol. 38, no. 4, pp. 1001–1020, Jul. 1959, doi: 10.1002/j.1538-7305.1959.tb01586.x.

[21] G. S. Robinson, ''Color edge detection,'' *Opt. Eng.*, vol. 16, no. 5, Oct. 1977, Art. no. 165479, doi: 10.1117/12.7972120.

[22] 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, May 2004, doi: 10.1109/TPAMI.2004.1273918.

[23] P. Dollar and C. L. Zitnick, ''Fast edge detection using structured forests,'' *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 37, no. 8, pp. 1558–1570, Aug. 2015, doi: 10.1109/TPAMI.2014.2377715.

[24] Y. Ganin and V. Lempitsky, ''N4-fields: Neural network nearest neighbor fields for image transforms,'' in *Proc. Asian Conf. Comput. Vis.*, Cham, Switzerland: Springer, 2014, pp. 536–551.

[25] 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 *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2015, pp. 3982–3991, doi: 10.1109/CVPR.2015.7299024.

[26] G. Bertasius, J. Shi, and L. Torresani, ''DeepEdge: A multi-scale bifurcated deep network for top-down contour detection,'' in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2015, pp. 4380–4389, doi: 10.1109/CVPR.2015.7299067.

[27] S. Xie and Z. Tu, ''Holistically-nested edge detection,'' in *Proc. IEEE Int. Conf. Comput. Vis. (ICCV)*, Dec. 2015, pp. 1395–1403, doi: 10.1109/ICCV.2015.164.

[28] Y. Liu, M. Cheng, X. Hu, K. Wang, and X. Bai, ''Richer convolutional features for edge detection,'' in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jul. 2017, pp. 5872–5881, doi: 10.1109/CVPR.2017.622.

[29] Y. Wang, X. Zhao, Y. Li, and K. Huang, ''Deep crisp boundaries: From boundaries to higher-level tasks,'' *IEEE Trans. Image Process.*, vol. 28, no. 3, pp. 1285–1298, Mar. 2019, doi: 10.1109/TIP.2018.2874279.

[30] L. J. van Vliet, I. T. Young, and G. L. Beckers, ''A nonlinear Laplace operator as edge detector in noisy images,'' *Comput. Vis., Graph., Image Process.*, vol. 45, no. 2, pp. 167–195, Feb. 1989.

[31] K. He, X. Zhang, S. Ren, and J. Sun, ''Deep residual learning for image recognition,'' in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2016, pp. 770–778, doi: 10.1109/CVPR.2016.90.

[32] Y. Zhang, Y. Tian, Y. Kong, B. Zhong, and Y. Fu, ''Residual dense network for image super-resolution,'' in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, Jun. 2018, pp. 2472–2481.

[33] J. K. Wibisono and H. Hang, ''Fined: Fast inference network for edge detection,'' in *Proc. IEEE Int. Conf. Multimedia Expo. (ICME)*, Jul. 2021, pp. 1–6, doi: 10.1109/ICME51207.2021.9428230.

[34] 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, Jul. 2016, doi: 10.1109/TPAMI.2015.2481406.

[35] J. Pont-Tuset and F. Marques, ''Measures and meta-measures for the supervised evaluation of image segmentation,'' in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jun. 2013, pp. 2131–2138, doi: 10.1109/CVPR.2013.277.

[36] A. Paszke, ''Pytorch: An imperative style, high-performance deep learning library,'' in *Proc. NeurIPS*, 2019, pp. 8024–8035.

[37] F. Chollet, ''Xception: Deep learning with depthwise separable convolutions,'' in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jul. 2017, pp. 1800–1807, doi: 10.1109/CVPR.2017.195.

[38] X. Soria, G. Pomboza-Junez, and A. D. Sappa, ''LDC: Lightweight dense CNN for edge detection,'' *IEEE Access*, vol. 10, pp. 68281–68290, 2022, doi: 10.1109/ACCESS.2022.3186344.

[39] H. Yang, Y. Li, X. Yan, and F. Cao, ''ContourGAN: Image contour detection with generative adversarial network,'' *Knowl.-Based Syst.*, vol. 164, pp. 21–28, Jan. 2019.

[40] M. Pu, Y. Huang, Y. Liu, Q. Guan, and H. Ling, ''EDTER: Edge detection with transformer,'' in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2022, pp. 1392–1402, doi: 10.1109/CVPR52688.2022.00146.

[41] J. He, S. Zhang, M. Yang, Y. Shan, and T. Huang, ''BDCN: Bi-directional cascade network for perceptual edge detection,'' *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 44, no. 1, pp. 100–113, Jan. 2022, doi: 10.1109/TPAMI.2020.3007074.

[42] P. Isola, D. Zoran, D. Krishnan, and E. H. Adelson, ''Crisp boundary detection using pointwise mutual information,'' in *Computer Vision ECCV 2014*, vol. 8691, D. Fleet, T. Pajdla, B. Schiele, and T. Tuytelaars, Eds. Cham, Switzerland: Springer, 2014, pp. 799–814, doi: 10.1007/978-3-319-10578-9_52.

[43] S. Gupta, P. Arbelaez, and J. Malik, ''Perceptual organization and recognition of indoor scenes from RGB-D images,'' in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jun. 2013, pp. 564–571, doi: 10.1109/CVPR.2013.79.

[44] 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 *Proc. NIPS*, 2017, pp. 3964–3973.

[45] K. Maninis, J. Pont-Tuset, P. Arbeláez, and L. Van Gool, ''Convolutional oriented boundaries: From image segmentation to high-level tasks,'' *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 40, no. 4, pp. 819–833, Apr. 2018, doi: 10.1109/TPAMI.2017.2700300.

[46] A. Kumawat and S. Panda, ''A robust edge detection algorithm based on feature-based image registration (FBIR) using improved Canny with fuzzy logic (ICWFL),'' *Vis. Comput.*, vol. 38, no. 11, pp. 3681–3702, Nov. 2022, doi: 10.1007/s00371-021-02196-1.

SHI-SHUI BAO received the master's degree from the Anhui University of Science and Technology, where he is currently pursuing the Ph.D. degree. His research interests include image processing, computer vision, and machine learning.

YOU-RUI HUANG was born in 1971. He received the Ph.D. degree. He is currently a Professor with the Anhui University of Science and Technology, Huainan, Anhui, China. His research interests include intelligent information process and wireless sensor networks.

JIA-CHANG XU was born in 1979. He received the master's and Ph.D. degrees in computer science from the Anhui University of Science and Technology. His research interests include computational intelligence and optimal control.

GUANG-YU XU was born in 1975. He received the master's degree from the Anhui University of Science and Technology, China, and the Ph.D. degree in computer science from the Hefei University of Technology. His research interests include digital image processing and artificial intelligence.