SuperEdge: Towards a Generalization Model for Self-Supervised Edge Detection

Author Name
Affiliation
email@example.com

Abstract

Edges are identified through pronounced pixel discontinuities, providing essential structural information, particularly in regions lacking texture. Existing edge detection approaches heavily rely on pixel-wise annotations, which are labour-intensive and subject to inconsistencies when acquired manually. This paper introduces a novel, self-supervised approach, SuperEdge, for edge detection using a multi-level, multi-homography technique, which enables the transfer of annotations from synthetic datasets to real-world scenarios. In particular, to fully leverage the generated edge annotations, we developed a streamlined yet efficient model to extract edges at pixel and object levels concurrently. Thanks to self-supervised training, our method eliminates the dependency on manually annotated edge labels, enhancing its generalizability across diverse datasets. Comparative evaluations reveal that SuperEdge advances edge detection, demonstrating improvements of 4.9% in ODS and 3.3% in OIS over the state-of-the-art STEdge method on the BIPEDv2 dataset.

1 introduction

Edge detection is a fundamental, long-standing, and extensively researched task in computer vision, focusing on identifying and extracting discontinuities in images. These discontinuities are sensitive to object features, such as shape, colour, material, and other attributes. It plays a vital role in complex vision tasks like semantic segmentation [Jin *et al.*, 2023], depth estimation [Kiran, 2022], and object detection [Li *et al.*, 2021]. Traditional edge detection methods, such as the Canny algorithm [Canny, 1986], rely on local gradient-based features [Oskoei and Hu, 2010]. In recent years, deep learning-based edge detectors have shown promising progress. For instance, networks like HED [Xie and Tu, 2015], BCDN [He *et al.*, 2019], DexiNed [Poma *et al.*, 2020], PiDiNet [Su *et al.*, 2021], and TEED [Soria *et al.*, 2023a] typically undergo training on manually annotated datasets and incorporate multi-layer supervision to enhance the models' generalization capabilities.

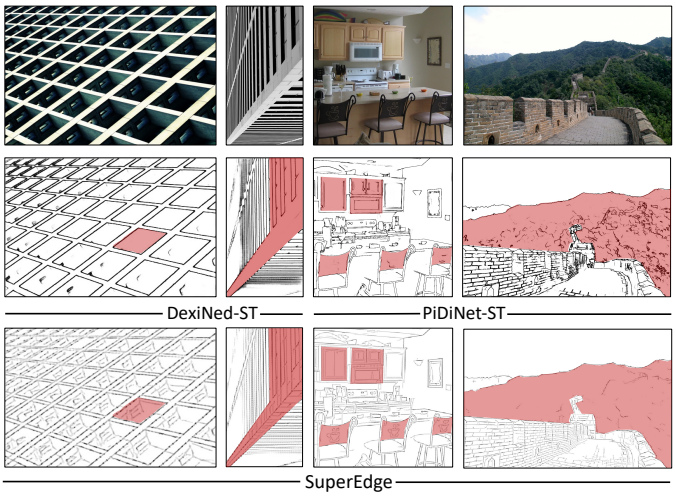

Figure 1: Comparison of edge detection: DexiNed-ST [Soria *et al.*, 2023b], PiDiNet-ST [Su *et al.*, 2021] and our proposed SuperEdge. We can clearly observe that the crucial details are properly preserved with our method (e.g., within the regions marked in red).

However, a notable drawback of existing edge detectors is their substantial reliance on manually annotated data, which is both costly and labour-intensive. Furthermore, edge annotation suffers from a critical issue of inconsistency, where different annotators often provide divergent interpretations of edge boundaries within the same image. This is in contrast to tasks such as object detection, where annotations tend to be more consistent. For instance, the classical edge detection dataset BSDS [Martin *et al.*, 2001] was annotated by five individuals, and the Multicue [Mély *et al.*, 2016] dataset involved six annotators, leading to frequent discrepancies. Consequently, the ground truths are typically obtained by taking the average of these annotations.

Given these observations, we propose an edge self-labelling methodology that facilitates the transition from virtual synthetic scenes to real-world scenes. This self-supervised generation method is based on Homography Adaption proposed by SuperPoint [DeTone *et al.*, 2018], which we adapt and apply to edge detection, marking its inaugural use in this domain. Similar to keypoints, this method of annotation generation exhibits high sensitivity to pixel-level edges within images. However, it falls short of capturing object-level edges. To address this limitation, we draw inspiration from STEdge [Ye *et al.*, 2023] and employ L0-

smoothing [Xu *et al.*, 2011] for image clustering based on colour attributes, in conjunction with the Canny algorithm, to extract object-level edges effectively.

Furthermore, we propose an efficient and straightforward edge detector named SuperEdge. This model adopts an encoder-decoder architecture, distinguishing itself from prior approaches that simultaneously optimize predictions across all convolutional layers before computing the mean. Our model strategically splits its decoder output heads into two distinct streams: one dedicated to extracting pixel-level edges and the other to object-level edges. This design aligns seamlessly with the self-labelled data produced by our method and eliminates the complexity of concurrently optimizing multiple convolutional layers. When trained on self-supervised data from COCO2017 [Lin *et al.*, 2014], our strategy and model outperform the state-of-the-art method STEdge [Ye *et al.*, 2023] across various datasets. Furthermore, our approach demonstrates effective edge extraction capabilities in complex scenes (see Fig.1).

Succinctly, the main contributions are as follows:

- We introduce a self-supervised strategy for edge detection that effectively generates pixel-level and object-level edge annotations in real-world scenes, obviating the need for labour-intensive manual labelling.
- Based on the generated data, we designed a novel model named SuperEdge. Comprehensive evaluations underscore the robust cross-dataset generality of both the self-supervised strategy and the SuperEdge model.

2 Related Work

In this section, we briefly glance through several edge detectors and datasets. For a more detailed treatment of these topics in general, the recent compilation from [Yang *et al.*, 2022] also offers sufficiently comprehensive insights.

2.1 Edge Detectors

Edge detection methods can be divided into three categories: traditional edge detectors, learning-based detectors and deep learning-based edge detectors. (1) Traditional methods, including Sobel [Sobel, 1970] and Canny [Canny, 1986], extract edges through direct gradient analysis of images. (2) Learning-based methods amalgamate diverse low-level features, utilizing prior knowledge to train detectors for edge generation. (3) Deep learning-based edge detection predominantly employs Convolutional Neural Network (CNN) architectures, with prominent methods including HED [Xie and Tu, 2015], DexiNed [Poma *et al.*, 2020], PiDiNet [Su *et al.*, 2021]. These methods typically undergo training on manually annotated datasets designed for edge detection and frequently incorporate techniques such as multi-layer regularization and image augmentation to bolster the method's generalization capabilities.

2.2 Edge Datasets

Datasets play a pivotal role in deep learning-based edge detection methodologies. Current edge detection datasets include BSDS [Martin *et al.*, 2001], Multicue [Mély *et al.*, 2016] and NYUD [Silberman *et al.*, 2012], etc. For convention, introducing a new edge detector is normally accompanied by releasing a new dataset tailored for the method. For instance, RindNet [Pu *et al.*, 2021] not only classifies edges in images but also further annotates the BSDS dataset to create the enhanced BSDS-RIND dataset. Similarly, alongside DexiNed [Poma *et al.*, 2020], the BIPED dataset was introduced, followed by an extended version, BIPEDv2 [Soria *et al.*, 2023b], which provides further annotations for images in BIPED. The primary motivation for continuously introducing new datasets is to improve model generalization capabilities.

However, creating these datasets is an arduous and labour-intensive process, typically requiring the collective efforts of numerous annotators. Furthermore, the amount of manually labelled data is significantly constrained, where most datasets contain only a few hundred images for training purposes and even fewer for testing. While the quantity is insufficient for deep learning methods, it necessitates data augmentation techniques, such as rotation and scaling, to amplify the volume of training data. Addressing the challenges associated with manual data annotation, STEdge [Ye *et al.*, 2023] introduced a novel self-supervised edge detection strategy. It employs multi-layer regularization with the $L0$-smoothing algorithm and the Canny operator, facilitating self-supervised model training through iterative loops. This innovative solution enables the model to reduce its reliance on manually labelled datasets initially. Nevertheless, the heavy dependence on the Canny operator for iteration leads to noticeably noisy predictions in various scenarios due to its sensitivity to noise in the original image.

3 Method

Our proposed SuperEdge is based on the encoder-decoder architecture that can effectively extract edges from both pixel and object levels. To ensure effective learning and adaptation, the entire self-supervised training strategy unfolds in three steps: (1) initially, SuperEdge is trained on a synthetic edge dataset with known ground truth. (2) We leverage the model pre-trained on synthetic data to perform edge annotation on real-world scenes, utilizing methods such as homography adaptation, $L0$-smoothing, and the Canny algorithm. (3) SuperEdge is trained on real scenes using pseudo labels generated from the preceding stage. Fig. 2 provides an overview of our self-supervised pipeline.

3.1 Self-supervise Strategy

Drawing inspirations from SuperPoint [DeTone *et al.*, 2018] in keypoint detection and SOLD^2 [Pautrat *et al.*, 2021] in line detection, we extend their homography adaptation to edge detection to realize the self-annotation of edges in real-world scenes. In the **Synthetic Training** stage, we train on a virtual synthetic dataset containing known edge annotations. As shown in Fig. 2 (left), the synthetic dataset is generated by introducing different geometric shapes (cubes, polygons, stars, etc.) under the background of random Gaussian noise.

In the **Real Scene Annotation** stage, the model pre-trained on synthetic data, denoted as a function $f(\cdot)$, undergoes a series of random homography transformations $\mathcal{H}$ on input

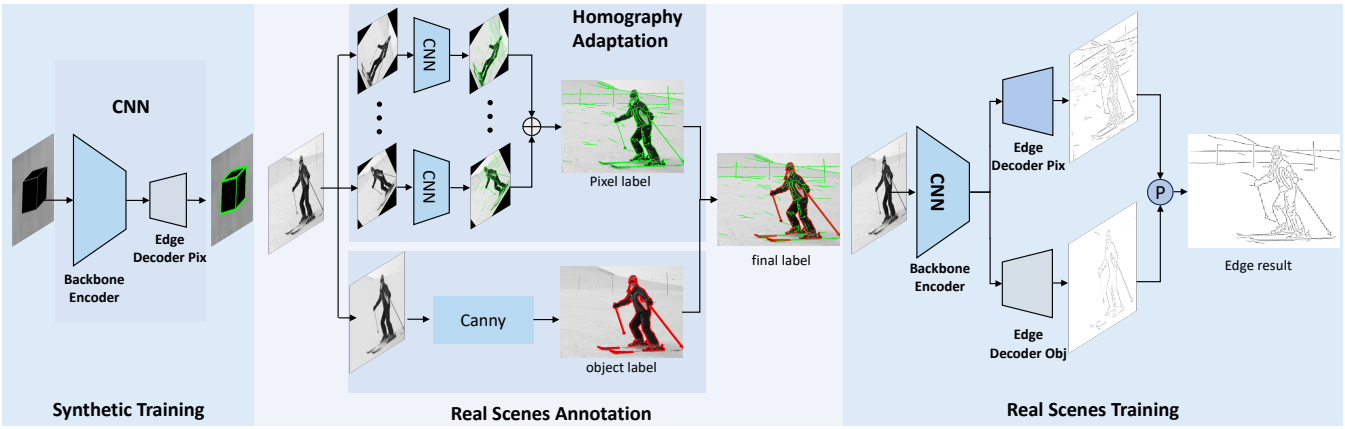

Figure 2: Training pipeline overview. **Left:** Our detector is first trained on an edge synthetic dataset of known ground truth. **Middle:** Object-level and pixel-level pseudo ground truth of edge is generated on real images through methods such as L0 smoothing, Canny, and homography adaptation. **Right:** Finally, the SuperEdge is trained on real images using the pseudo ground truth.

image I. We introduce a multitude of random homography transformations to the process. By aggregating the outcomes from these diverse transformations, we synthesize a more robust edge predictor denoted as $F(\cdot)$. This enhanced predictor excels within simulated environments and demonstrates commendable accuracy and stability when applied to real-world scenes. Moreover, it underpins the effective transition from synthetic to real-world datasets. Based on the settings above, we can get the following formula:

$$F\left(I;f\right)=\frac{1}{N_h}\sum_{i=1}^{N_h}\mathcal{H}_i^{-1}f\left(\mathcal{H}_i(I)\right) \quad . \tag{1}$$

where N_h represents the number of applied homography adaptations. $\mathcal{H}_i^{-1}$ is the inverse transformation corresponding to the i-th homography, ensuring that the predictions are aligned with the original image domain.

By utilizing the annotator function $F(\cdot)$, our framework can extract pixel-level edges from images, with intermediate results depicted in Fig. 2 (middle). This approach is adequate when extracting pixel-level edges is the sole objective. Yet, to enhance the granularity of edge processing, it is imperative to consider the object-level edges. In particular, we have engineered a process to annotate object-level edges within images, denoted as the function $G(\cdot)$. This process is structured into two sequential steps: (1) Application of Gaussian blur followed by $L0$-smoothing to reduce noise and enhance image smoothness, aiding in the demarcation of object boundaries. (2) Deployment of the Canny edge detection algorithm, succeeded by an image dilation process to highlight object-level edges. Combining these procedures results in the precise and enriched annotation of object-level edges. Then, the final generation of self-supervised annotation can be expressed as:

$$\mathbf{x}=F(I,f)+G(I) \quad . \tag{2}$$

where $\mathbf{x}$ represents the combined edge annotations.

Finally, in the **Real Scenes Training** stage, we trained the final model, SuperEdge, using the annotations generated from real-world scenes described in the second stage.

Table 1: SuperEdge overview, where B represents the batch size, H and W are the height and width of the input image, respectively.

Overview	
Operation	Output Shape
Input Image	[*B*, *H*, *W*]
Encoder	[*B*, *128*, *H/8*, *W/8*]
vgg block1	[*B*, *64*, *H/2*, *W/2*]
vgg block2	[*B*, *64*, *H/4*, *W/4*]
vgg block3	[*B*, *128*, *H/8*, *W/8*]
vgg block4	[*B*, *128*, *H/8*, *W/8*]
Pixel-edge Decoder	[*B*, *65*, *H/8*, *W/8*]
conv2d, bn, relu	[*B*, *256*, *H/8*, *W/8*]
conv2d, bn, softmax	[*B*, *65*, *H/8*, *W/8*]
reshape	[*B*, *H*, *W*]
Object-edge Decoder	[*B*, *65*, *H/8*, *W/8*]
conv2d, bn, relu	[*B*, *256*, *H/8*, *W/8*]
conv2d, bn	[*B*, *195*, *H/8*, *W/8*]
self-attention, softmax	[*B*, *65*, *H/8*, *W/8*]
reshape	[*B*, *H*, *W*]

3.2 Network and Loss Design

As presented in Fig. 2 (right) and Table 1, our model uses grayscale images as input and performs feature extraction through a shared backbone encoder. We split the output into two branches to correspond to the annotation generated by the Real Scene Annotation stage. One branch is used to predict pixel-level edges, while the other branch focuses on detecting object-level edges. Thus, our loss function $\mathcal{L}_{sum}$ is:

$$\mathcal{L}_{sum}=\mathcal{L}_{pix}+\mathcal{L}_{obj} \quad . \tag{3}$$

where $\mathcal{L}_{pix}$ and $\mathcal{L}_{pix}$ denote the loss in pixel- and object-level branches, respectively. We empirically set their weights as one since both contribute equally to the final output.

Regarding the pixel-level branch, we adopt a similar approach to the SuperPoint [DeTone *et al.*, 2018] keypoint extraction method. The pixel decoder outputs a coarse feature map, denoted as $\mathbf{P}$, with dimensions $\frac{W}{8}\times\frac{H}{8}\times 65$. In this map, each 65-dimensional vector represents an 8×8 image patch, complemented by an additional dimension to indicate the absence of an edge ('no edge' dustbin). Pixel-level edge

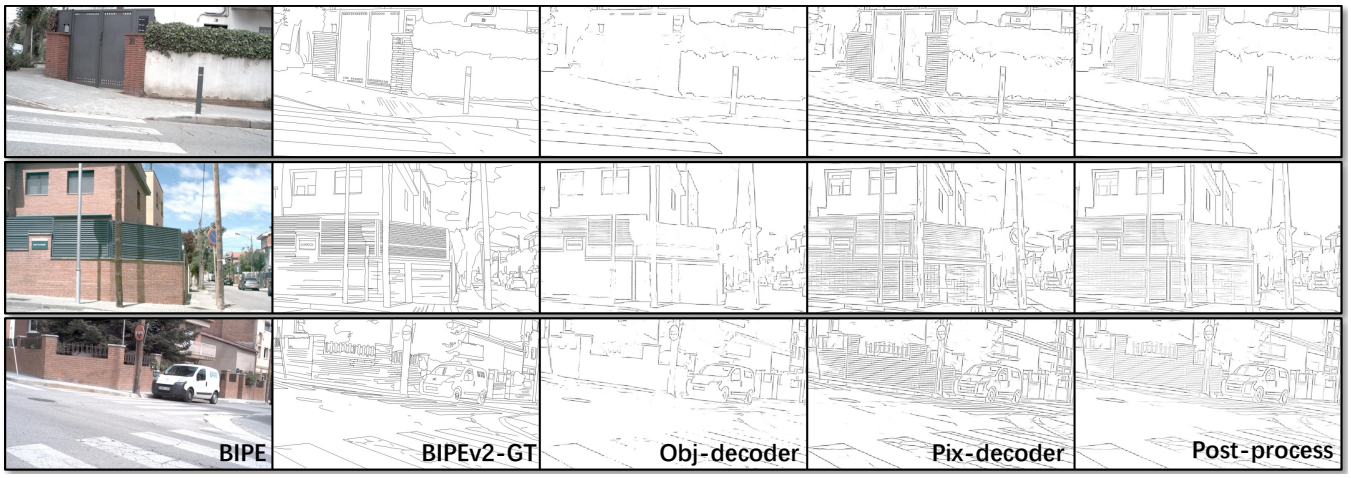

Figure 3: Results of each detection head of SuperEdge on BIPED.

involves a Cross-Entropy Loss between the feature map $\mathbf{P}$ and the corresponding ground truth $\mathbf{y}$:

$$\mathcal{L}_{pix} = \frac{64}{H \times W} \sum_{h=1,w=1}^{\frac{H}{8},\frac{W}{8}} l_p(x_{hw}; y_{hw}) \quad . \tag{4}$$

where H and W represent the height and width of the input image, respectively. $l_p(x_{hw}; y)$ is computed by

$$l_p(x_{hw}; y) = -\log\left(\frac{\exp\left(\mathbf{P}_{hwy}\right)}{\sum_{k=1}^{65}\exp\left(\mathbf{P}_{hwk}\right)}\right) \quad . \tag{5}$$

where we apply a softmax operation along the channel dimension of $\mathbf{P}$ to construct the final edge map of dimensions $H \times W$. This operation is followed by the exclusion of the 65th dimension, which is designated for the 'no edge' classification. It is important to note that the loss function in Eq. 4, originally developed for keypoint extraction in the SuperPoint model, has been adeptly repurposed for pixel-level edge detection in our approach. Although the SuperPoint model was initially intended to segment the image into 64 distinct subimages for keypoint classification, the flexibility of the loss function allows for effective pixel-level edge detection, even in scenarios with multiple keypoints within a single patch.

For the object-level branch, considering that the task necessitates the acquisition of more semantic information, we introduce a single-layer self-attention mechanism to capture the global semantics and structure of the entire image. Given the substantial imbalance in the distribution of edge/non-edge pixels within natural images, we employ the robust weighted Cross-Entropy Loss ($\mathcal{L}_{obj}$) to guide the training process:

$$\mathcal{L}_{\text{obj}} = \begin{cases} \alpha \cdot l_p(x_{hw}; y_{hw}), & \text{if } y_{hw} = 65 \\ \beta \cdot l_p(x_{hw}; y_{hw}), & \text{if } y_{hw} < 65 \end{cases} \quad . \tag{6}$$

in which

$$\begin{aligned} \alpha &= \lambda \cdot \frac{|Y^+|}{|Y^+| + |Y^-|} \\ \beta &= \frac{|Y^-|}{|Y^+| + |Y^-|} \end{aligned} \quad . \tag{7}$$

where Y^+ and Y^- represent the counts of edge pixels and non-edge pixels in the pseudo ground truth, respectively. λ is a hyper-parameter to balance positive and negative samples.

3.3 Post-processing Module

Fig. 3 demonstrates that our pixel-level edge detector can effectively identify various edge types. However, it is susceptible to noise and other destabilizing factors, leading to potential inaccuracies in edge representation. In contrast, object-level edge detection provides noise-reduced results but may lack detailed edge information. An intuitive understanding is that pixel- and object-levels complement each other. Thus, we have developed a streamlined post-processing algorithm to address these limitations, employing the object-level edge predictions as a framework. A Breadth-First Search (BFS) expansion method further enhances it with pixel-level edge details. The specific algorithm is as follows:

Algorithm 1 Post-processing algorithm

Input: Set of object-level edges $\mathcal{U}$, pixel-level edges $\mathcal{E}$.
Output: Fused edges X.

```
for p ∈ U do
    if visited(p) then
        continue;
    end if.
    Search and mark on E starting from p:
        Y = BFS( p, E).
        Add Y to X.
end for.
```

The effectiveness of this approach is visually represented in Fig. 3 (right). The final edge detection result, denoted as O_{fusion}, is calculated using the following equation:

$$O_{fusion} = norm(\frac{\mathcal{P}(O_{pix}, O_{obj}) + O_{obj}}{2}) \quad . \tag{8}$$

where O_{pix} and O_{obj} are the edge prediction results of two detection heads, respectively. $\mathcal{P}$ is the post-processing operation. $norm$ is the image normalization operation.

3.4 Implementation

For network optimization in SuperEdge, we employed the Adam optimizer set to a learning rate of 0.001. The training was structured into sessions of 100 epochs each, using a batch size of 16. The training and testing process used

an NVIDIA GeForce RTX 3090 GPU with the PyTorch library. Initially, the model was trained on the 120×160 Synthetic dataset, completed in approximately 1 hour. After that, labelling and training on the 240×320 COCO-train2017 dataset [Lin *et al.*, 2014] were addressed, which took around 50 hours. Finally, secondary labelling and training on the 480×640 COCO-val2017 dataset were conducted, which took about 10 hours. The source code for SuperEdge is available at https://github.com/lktidaohuoxing/SuperEdge.

4 Experiments

4.1 Dataset

The datasets utilized in our experimental analysis encompass COCO [Lin *et al.*, 2014], BSDS [Martin *et al.*, 2001], BSDS-RIND [Pu *et al.*, 2021], BIPED [Poma *et al.*, 2020] and BIPEDv2 [Soria *et al.*, 2023b]:

- BSDS was created explicitly for image segmentation and edge detection tasks, comprising 200 training set samples, 100 validation set samples, and 200 test set samples. Each image in BSDS was annotated by a minimum of 5 annotators, making it a valuable resource for image segmentation and boundary detection research.
- BSDS-RIND is a further extension of the BSDS dataset. It categorizes edges within images into four distinct types: reflectance edges, illumination edges, regular edges, and depth edges. This additional annotation enriches the content and diversity of the dataset.
- BIPED consists of 250 annotated images of outdoor scenes, divided into a training set comprising 200 images and a testing set containing 50 images. BIPEDv2 represents the final iteration of the BIPED dataset, incorporating further annotations and refinements.
- NYUD comprises 1449 RGBD images, encompassing 464 indoor scenarios specifically tailored for segmentation purposes. Following [Gupta *et al.*, 2013], this dataset is subdivided into training, validation, and testing subsets, with 654 images for testing, while the remaining images are allocated for training and validation.

4.2 Metric

As with previous work on edge detection, we use the following three indicators to evaluate the effect of detectors:

- Optimal Dataset Scale (ODS) computed using a global threshold for the entire dataset.
- Optimal Image Scale (OIS) computed by using a different threshold on every image.
- Average Precision (AP) is the integral of the precision/recall (P/R) curve.

It should be noted that in some cases, the P/R curve is shorter and does not cover the entire range. In such a case, AP values are lower, and the assessment reliability of AP may be lower compared to ODS and OIS. Additionally, in our experiments, we followed the convention of setting the maximum allowable distance between the predicted edge and the corresponding pixel of the ground truth value to 0.0075 for both ODS and OIS evaluations.

4.3 Ablation Study

As mentioned in Section 3, we introduce a network based on an encoder-decoder architecture, incorporating two decoders responsible for pixel-level edges and object-level edge detection. To verify the effectiveness of each module in this design, we conducted a series of ablation experiments. These experiments were designed to independently evaluate the contribution of each decoder head in the edge detection task, as well as the impact of the post-processing algorithm on the results.

The ablation experiments were performed on the BIPED dataset, evaluating the results obtained from (1) two detection heads (individually), (2) their combination (through simple addition), and (3) using a post-processing algorithm. The quantitative results are detailed in Table 2. Additionally, visualization results are depicted in Fig. 3.

As indicated in Table 2, the AP values of the two decoders are relatively low. After combination through a simple addition, the overall AP value shows a significant improvement (around 24.5%). This observation suggests the effective complementarity between the two detection decoders for different types of edges. Furthermore, introducing the post-processing algorithm leads to noticeable improvements in ODS (3.0%) and OIS (2.4%), even when the AP values remain comparable. The main reason is that the post-processing algorithm effectively eliminates edge noise at the pixel level. In other words, each decoder head and post-processing algorithm is essential in the edge detection task.

Table 2: Ablation study on the BIPED dataset.

Method	ODS	OIS	AP
object-decoder	0.707	0.707	0.537
pixel-decoder	0.784	0.789	0.680
pixel&object	0.787	0.799	**0.758**
pixel&object + post-process	**0.811**	**0.818**	0.755

Table 3: Comparative experiments on BIPED dataset. * denotes training with pseudo labels without human annotation.

Method	Train	ODS	OIS	AP
HED	BSDS	0.711	0.725	0.714
BDCN	BSDS	0.714	0.725	0.687
DexiNed	BSDS	0.699	0.716	0.664
PiDiNet	BSDS	0.776	0.785	0.740
DexiNed-ST	COCO-val2017*	0.760	0.783	**0.798**
PiDiNet-ST	COCO-val2017*	0.786	0.804	0.259
SuperEdge	COCO-val2017*	**0.811**	**0.818**	0.755

4.4 Comparison

We compare SuperEdge with existing self-supervised method STEdge [Ye *et al.*, 2023] across multiple models, including PiDiNet [Su *et al.*, 2021] and DexiNed [Poma *et al.*, 2020], which are two lightweight edge detection methods. We also compared it against classical methods such as HED [Xie and Tu, 2015] and BDCN [He *et al.*, 2019]. In particular, we trained four methods, namely HED, BDCN and PiDiNet,

Table 4: Comparative experiments on different datasets, all models were trained on the COCO-val2017.

	BIPEDv2			BSDS-RIND			NYUD			BSDS		
	ODS	OIS	AP	ODS	OIS	AP	ODS	OIS	AP	ODS	OIS	AP
DexiNed-ST	0.780	0.804	0.668	0.662	0.685	0.544	0.536	0.553	0.385	0.648	0.659	0.527
PiDiNet-ST	0.786	0.802	0.340	0.670	0.695	0.223	0.544	0.556	0.114	0.656	0.672	0.186
SuperEdge	**0.825**	**0.829**	**0.752**	**0.722**	**0.733**	**0.640**	**0.581**	**0.591**	**0.438**	**0.672**	**0.686**	**0.571**

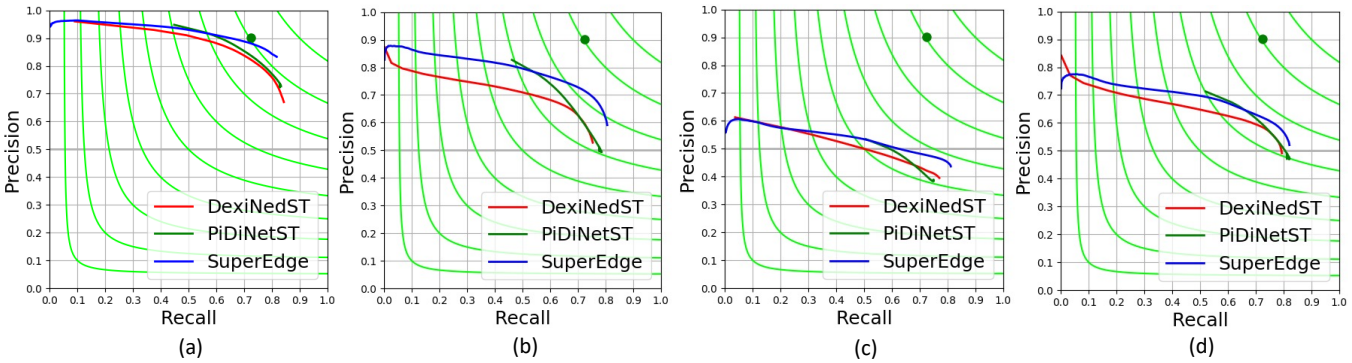

Figure 4: P/R curves on different datasets. (a) BIPEDv2. (b) BSDS-RIND. (c) NYUD. (d) BSDS.

based on the BSDS dataset. Subsequently, we retrained DexiNed and PiDiNet using the self-supervised strategy implemented in STEdge on the COCO-val2017 dataset, resulting in DexiNed-ST and PiDiNet-ST. Finally, this study applied the self-supervised data generation method proposed in Section 3 to train SuperEdge on the COCO-val2017 dataset.

The quantitative results are presented in Table 3. We observe that the models trained using the STEdge self-supervised strategy exhibited superior generalization capabilities on unseen data. Specifically, they achieved higher ODS (0.786) and OIS (0.804) scores on the untouched BIPED dataset. Furthermore, our model and training approach demonstrated significant advantages compared to other self-supervised comparative methods. Notably, there is a 3.2% improvement in ODS and a 1.7% improvement in OIS. These results validate the effectiveness of the self-supervised method and SuperEdge model proposed in this study for complex edge detection tasks.

Next, we compare SuperEdge, DexiNed-ST, and PiDiNet-ST with multiple datasets, including BIPEDv2, BSDS-RIND, BSDS, and NYUD. As shown in the quantification results Table 4 and the corresponding P/R curve Fig. 4, SuperEdge consistently outperforms other methods across all datasets and metrics, around 4.9% (ODS) on the BIPEv2 dataset, 7.7% (ODS) on the BSDS-RIND dataset. It once again verifies the superiority of the self-supervised method and model design employed in this study for edge detection. In addition, the visual results of the BIPEDv2 and BSDS-RIND are shown in Fig. 5. It shows that the performance improvement of SuperEdge mainly lies in its robustness to noise (such as lawn, asphalt road, ocean, etc.) and its ability to effectively extract various edges in the image (such as illumination changes, depth changes, object contours, etc.). For instance, as shown in the results of BIPEv2 in Fig. 5, our method effectively identifies the edges on the asphalt road while simultaneously eliminating noise. In contrast, other methods produce results that are heavily contaminated with noise. Moreover, in the third image, SuperEdge also outperforms existing methods when processing object outlines and can extract more detailed edges of the sailboat. In addition, in the fourth picture, when facing double glass edges, our method can effectively distinguish different types of edges, whereas other methods tend to confuse them.

5 Conclusion

Summary. In this paper, we present a novel self-supervised edge detection strategy along with its corresponding model design. In particular, we introduce Homography Adaptation into edge detection and combine it with Canny and L0-smooth to enable edge annotation. Furthermore, we propose a dual decoder model called SuperEdge for training in real-world scenarios. SuperEdge achieves marked improvements on different datasets compared with current state-of-the-art methods.

Limitations. Currently, while our method demonstrates promising performance in terms of generalization, the training effect of self-supervision still cannot exceed that of supervised training on specific datasets. For instance, dexiNed trained on the BIPEv2 training set can achieve an ODS of 0.893 on the BIPEv2 test set. Furthermore, the existing search-based post-processing algorithms are computed using CPU resources, thereby under-utilizing the potential of GPU resources and resulting in decreased inference speed.

Future Works. In future research, we aim to integrate keypoint detection with edge detection, enabling simultaneous output from a single model. Furthermore, we will explore a unified approach combining keypoints and edges to reveal the geometric representation and description of any edge in an image. Our ultimate objective is to extract, describe, and match various elements within an image pair.

Figure 5: Visualization results on BSDS-RIND (the first five rows) and BIPEDv2 (the last five rows).

References

[Canny, 1986] John Canny. A computational approach to edge detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, (6):679–698, 1986.

[DeTone *et al.*, 2018] Daniel DeTone, Tomasz Malisiewicz, and Andrew Rabinovich. Superpoint: Self-supervised interest point detection and description. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, pages 224–236, 2018.

[Gupta *et al.*, 2013] Saurabh Gupta, Pablo Arbelaez, and Jitendra Malik. Perceptual organization and recognition of indoor scenes from rgb-d images. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 564–571, 2013.

[He *et al.*, 2019] Jianzhong He, Shiliang Zhang, Ming Yang, Yanhu Shan, and Tiejun Huang. Bi-directional cascade network for perceptual edge detection. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 3828–3837, 2019.

[Jin *et al.*, 2023] Jianhui Jin, Wujie Zhou, Rongwang Yang, Lv Ye, and Lu Yu. Edge detection guide network for semantic segmentation of remote-sensing images. *IEEE Geoscience and Remote Sensing Letters*, 34:1–5, 2023.

[Kiran, 2022] Ezgi Erbek Kiran. An investigation on edge detection of structures and depth estimation of isparta angle region (southwest turkey) using aeromagnetic data. *Mühendislik Bilimleri ve Tasarım Dergisi*, 10(1):142–151, 2022.

[Li *et al.*, 2021] Junxia Li, Ziyang Wang, Zefeng Pan, Qingshan Liu, and Dongyan Guo. Looking at boundary: Siamese densely cooperative fusion for salient object detection. *IEEE Transactions on Neural Networks and Learning Systems*, 10:3580–3593, 2021.

[Lin *et al.*, 2014] Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C Lawrence Zitnick. Microsoft coco: Common objects in context. In *Proceedings of the European Conference on Computer Vision*, pages 740–755. Springer, 2014.

[Martin *et al.*, 2001] D. Martin, C. Fowlkes, D. Tal, and J. Malik. A database of human segmented natural images and its application to evaluating segmentation algorithms and measuring ecological statistics. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, volume 2, pages 416–423, 2001.

[Mély *et al.*, 2016] David A Mély, Junkyung Kim, Mason McGill, Yuliang Guo, and Thomas Serre. A systematic comparison between visual cues for boundary detection. *Vision Research*, 120:93–107, 2016.

[Oskoei and Hu, 2010] Mohammadreza Asghari Oskoei and Huosheng Hu. A survey on edge detection methods. *University of Essex, UK*, 33, 2010.

[Pautrat *et al.*, 2021] Rémi Pautrat, Juan-Ting Lin, Viktor Larsson, Martin R Oswald, and Marc Pollefeys. Sold2: Self-supervised occlusion-aware line description and detection. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 11368–11378, 2021.

[Poma *et al.*, 2020] Xavier Soria Poma, Edgar Riba, and Angel Sappa. Dense extreme inception network: Towards a robust cnn model for edge detection. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, pages 1923–1932, 2020.

[Pu *et al.*, 2021] Mengyang Pu, Yaping Huang, Qingji Guan, and Haibin Ling. Rindnet: Edge detection for discontinuity in reflectance, illumination, normal and depth. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 6879–6888, 2021.

[Silberman *et al.*, 2012] Nathan Silberman, Derek Hoiem, Pushmeet Kohli, and Rob Fergus. Indoor segmentation and support inference from rgbd images. In *Proceedings of the European Conference on Computer Vision*, pages 746–760, 2012.

[Sobel, 1970] Irwin Edward Sobel. *Camera models and machine perception*. Stanford University, 1970.

[Soria *et al.*, 2023a] Xavier Soria, Yachuan Li, Mohammad Rouhani, and Angel D Sappa. Tiny and efficient model for the edge detection generalization. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 1364–1373, 2023.

[Soria *et al.*, 2023b] Xavier Soria, Angel Sappa, Patricio Humanante, and Arash Akbarinia. Dense extreme inception network for edge detection. *Pattern Recognition*, 139:109461, 2023.

[Su *et al.*, 2021] Zhuo Su, Wenzhe Liu, Zitong Yu, Dewen Hu, Qing Liao, Qi Tian, Matti Pietikäinen, and Li Liu. Pixel difference networks for efficient edge detection. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 5117–5127, 2021.

[Xie and Tu, 2015] Saining Xie and Zhuowen Tu. Holistically-nested edge detection. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 1395–1403, 2015.

[Xu *et al.*, 2011] Li Xu, Cewu Lu, Yi Xu, and Jiaya Jia. Image smoothing via l-0 gradient minimization. In *Proceedings of the SIGGRAPH Asia Conference*, pages 1–12, 2011.

[Yang *et al.*, 2022] Daipeng Yang, Bo Peng, Zaid Al-Huda, Asad Malik, and Donghai Zhai. An overview of edge and object contour detection. *Neurocomputing*, 488:470–493, 2022.

[Ye *et al.*, 2023] Yunfan Ye, Renjiao Yi, Zhiping Cai, and Kai Xu. Stedge: Self-training edge detection with multilayer teaching and regularization. *IEEE Transactions on Neural Networks and Learning Systems*, 10:1–11, 2023.