Computers, Materials & Continua

Tech Science Press

Doi:10.32604/cmc.2025.072399

ARTICLE

PIDINet-MC: Real-Time Multi-Class Edge Detection with PiDiNet

Mingming Huang[1], **Yunfan Ye**[1,*] **and Zhiping Cai**[2]

[1]School of Design, Hunan University, Changsha, 410082, China
[2]College of Computer, National University of Defense Technology, Changsha, 410082, China

*Corresponding Author: Yunfan Ye. Email: yeyunfan@hnu.edu.cn

Received: 26 August 2025; Accepted: 17 October 2025; Published: 09 December 2025

ABSTRACT: As a fundamental component in computer vision, edges can be categorized into four types based on discontinuities in reflectance, illumination, surface normal, or depth. While deep CNNs have significantly advanced generic edge detection, real-time multi-class semantic edge detection under resource constraints remains challenging. To address this, we propose a lightweight framework based on PiDiNet that enables fine-grained semantic edge detection. Our model simultaneously predicts background and four edge categories from full-resolution inputs, balancing accuracy and efficiency. Key contributions include: a multi-channel output structure expanding binary edge prediction to five classes, supported by a deep supervision mechanism; a dynamic class-balancing strategy combining adaptive weighting with physical priors to handle extreme class imbalance; and maintained architectural efficiency enabling real-time inference. Extensive evaluations on BSDS-RIND show our approach achieves accuracy competitive with state-of-the-art methods while operating in real time.

KEYWORDS: Multi-class edge detection; real-time; lightweight; deep supervision

1 Introduction

Edge detection is a basic and important task in computer vision [1]. The main goal is to find object boundaries and clear contours. It removes extra information and keeps the key structure of an image [2]. As one of the core parts of visual understanding, edge detection not only supports the perception of basic geometric cues, but also provides indispensable guidance for a wide range of downstream tasks such as object detection [3,4], semantic and instance segmentation [5–8], 3D reconstruction [9–11], medical image analysis [12,13], and image editing [10,14–16]. This problem is important because it works in two ways, giving simple low-level features and also connecting to high-level semantic reasoning.

Classical edge detection methods, including Sobel [17] and Canny [1] operators, rely on hand-crafted gradient filters to highlight rapid intensity variations in local neighborhoods [1,5,18,19]. These approaches, which were widely used in early vision systems, are computationally efficient and easy to implement. However, they rely only on low-level pixel gradients, which makes them very sensitive to noise, light changes, and textures. Consequently, they struggle to distinguish meaningful semantic boundaries from spurious edges [20]. This limitation prompted the transition toward learning-based techniques.

With the advent of deep learning [21–25], CNN-based approaches such as HED [24] and RCF [26] have substantially advanced the accuracy and robustness of edge detection by aggregating multi-scale deep features and employing deep supervision strategies [26]. Unlike their hand-crafted models, these models are capable of perceiving high-level object boundaries. But the improvements often need a lot of computing

power, which limits their use in real-world tasks like self-driving cars [27], augmented reality, and robotics. In particular, edge semantics have been shown to inform safe and efficient robot navigation [28]. As mobile and embedded devices grow fast, the conflict between accuracy and speed becomes clearer. Therefore, practical edge detectors need to provide both good accuracy and real-time performance.

To solve this trade-off, recent studies have started using lightweight designs that also use domain knowledge. A representative example is PiDiNet [29], which introduces the Pixel Difference Convolution (PDC) module. By fusing the prior knowledge of traditional gradient operators with the representational power of CNN [21,23,30], PiDiNet achieves accuracy comparable to much larger networks while maintaining high computational efficiency. By combining the knowledge from traditional gradient methods with the power of CNNs, PiDiNet reaches accuracy close to larger networks while staying efficient.

However, a critical limitation persists: most old and new methods see edge detection as a single-class task. In other words, they only answer "where is an edge?" but do not address the more informative question: "what caused this edge?" In reality, edges can come from many sources, like depth changes, surface shape changes, material differences, or lighting effects [31–33]. While visually similar, these edges convey entirely different semantic meanings [34,35]. For instance, depth and normal discontinuities are crucial for 3D reconstruction, whereas reflectance and illumination changes are more relevant for material recognition and scene relighting [36–39]. The lack of a method that combines high accuracy, multi-class understanding, and real-time speed is a major gap [40]. Using general multi-class segmentation methods is too slow, and training many separate binary detectors misses the connections between different types of edges.

While PiDiNet demonstrates the feasibility of real-time edge detection, it is fundamentally limited by its binary formulation. Real-world vision tasks, however, require not only the localization of edges but also the identification of their underlying physical causes. Existing multi-class edge detection methods prioritize accuracy at the cost of impractical runtime, creating a critical gap for resource-constrained applications. To bridge this gap, we transform PiDiNet into a real-time, multi-class edge detector. By introducing targeted architectural adaptations and a physics-informed training strategy, our approach achieves semantic richness without compromising efficiency.

We validate this design on BSDS-RIND [31], where the model runs at 250 FPS while delivering competitive accuracy across depth, normal, reflectance, and illumination edges. These results demonstrate that rich semantic edge perception is achievable in compact real-time networks and offer a practical path for deployment in resource-constrained systems.

2 Related Work

2.1 Traditional Edge Detection Methods

Edge detection aims to extract object boundaries and visually salient edges from natural images [41]. As a fundamental task in computer vision [42], edge detection has been extensively studied for many years [32,43]. Early edge detection methods relied on hand-crafted features and fixed operators. Operators such as Sobel and Prewitt detect edges by convolving the image to compute first-order gradients [44]. The Canny algorithm further introduced non-maximum suppression and dual-threshold hysteresis, establishing a classic and widely adopted standard pipeline [45]. Although these methods are computationally efficient, they are inherently low-level vision operations that are sensitive to noise and incapable of interpreting the semantic content of images [46]. As a result, they often produce false edges in textured regions or miss faint yet genuine edges [47].

2.2 Deep Learning-Based Edge Detection

The powerful feature learning capabilities of Convolutional Neural Networks (CNNs) have revolutionized the field of edge detection [21,48–52]. HED was the first end-to-end deep learning-based edge detection model [24]. It introduced a deep supervision mechanism that leverages multiple side-output layers to integrate features from different scales, significantly improving the continuity of semantic edges. Subsequent research has advanced in two main directions [53]. On one hand, methods like RCF [26] improved HED by using features from all convolutional layers to capture richer multi-scale information, while BDCN [54] introduced a bi-directional cascade structure that provides scale-specific supervision to different network layers, allowing more discriminative feature learning. More recent studies such as EDTER [55] have begun exploring the application of Vision Transformers in edge detection to capture long-range global context [26]. On the other hand, to meet the needs of real-time applications, researchers have worked on making models more lightweight [27]. For example, PiDiNet designed a pixel difference convolution (PDC) to mimic traditional edge operators, achieving accuracy close to large models with a compact design, while other architectures like DexiNed have also pursued this goal [29].

Despite their remarkable success, all these methods are designed to perform a single binary classification task (edge vs. non-edge) [56,57]. Their outputs provide no information about the physical attributes of edges, limiting their utility in downstream tasks that require fine-grained scene understanding [58].

2.3 Multi-Class Edge Detection

According to David Marr's classic vision framework [59], the four edge types considered here correspond to fundamental discontinuities in surface attributes and geometry, forming essential building blocks of visual scenes: reflectance edges (RE), arising from changes in material appearance (e.g., albedo, texture, color) and marking material boundaries; illumination edges (IE), stemming from variations in lighting (e.g., cast shadows, light transitions, highlights) and revealing illumination discontinuities; normal edges (NE), due to changes in surface orientation and capturing local shape discontinuities; and depth edges (DE), caused by variations in distance to the camera (e.g., occlusion boundaries, silhouettes) and conveying 3D depth layering. Taken together, these classes span both photometric (reflectance, illumination) and geometric (normal, depth) dimensions, providing a compact, physically grounded taxonomy that underpins multi-class edge detection and supports a wide range of downstream visual understanding tasks. Moreover, reflectance edges have been shown to assist key tasks such as pavement crack detection in intelligent transportation systems [39]. Illumination edges are critical for shadow removal and path detection [38]. Effective representation of normal and depth edges drives surface normal estimation accuracy [37] and depth estimation improvement [60]. Together, the joint exploration of these four edge types enhances tasks like depth refinement and comprehensive scene understanding [40], providing a precise foundational cue for downstream computer vision tasks.

Building on this taxonomy, distinguishing edge categories has become an active research direction: not all edges are equal [7,31,32,58]. Beyond perception benchmarks, edge semantics can guide motion planning and safety checks in robotics [28]. For instance, in scene parsing, separating depth/normal edges from reflectance/illumination edges is highly valuable. However, the literature remains relatively scarce and often prioritizes accuracy at the expense of efficiency. Some methods infer edge types via post-processing or multi-model fusion [61], but such strategies typically incur high computational cost, making them unsuitable for real-time applications [27].

Overall, there is still a gap in existing research: a method that can run in real time while providing refined multi-class edge maps. Our work takes a step toward addressing this by building on the efficient design of PiDiNet and reframing the binary edge detection task as a multi-class pixel-wise classification

task. This approach allows us to distinguish different physical attributes of edges while maintaining practical inference speed.

3 Methodology

This chapter elaborates on our proposed real-time Multi-class edge detection framework. Section 3.1 provides an overview of the overall model architecture. Subsequently, Section 3.2 details the modifications made to the PiDiNet [29] backbone to adapt it for the Multi-class task. Finally, Section 3.3 offers an in-depth explanation of the dynamic class-balancing strategy designed to address the extreme class imbalance problem.

3.1 Overview

This work addresses a key question: how to perform pixel-wise multi-class edge detection while keeping very high inference speed. To do this, we build a single end-to-end deep learning framework. The main idea is to make focused changes to a proven, efficient lightweight backbone network and add a better training method. We use the recent PiDiNet as the base model because it combines the knowledge of traditional edge operators with the strength of modern CNN, giving a good balance between speed and accuracy.

The overall architecture of our method is illustrated in Fig. 1. The network takes a three-channel input image $\mathbf{I} \in \mathbb{R}^{3\times H\times W}$ and outputs a dense Multi-class edge prediction map. Its workflow can be clearly divided into four stages: feature extraction, multi-scale feature fusion, Multi-class prediction, and deep supervision [30].

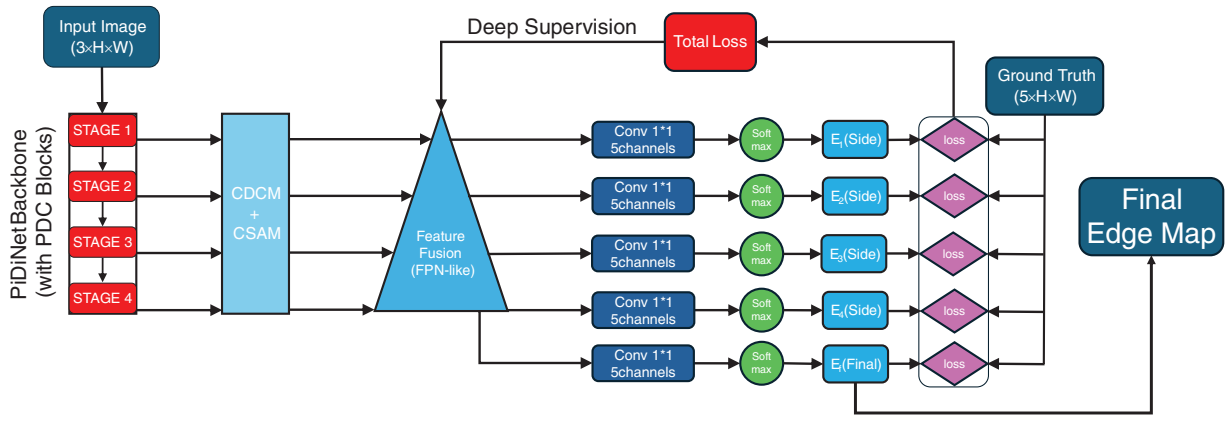

Figure 1: The architecture of the proposed Multi-class edge detection network. Based on the PiDiNet backbone, we expand the number of channels in all output layers (including four side outputs and one final output) from 1 to 5 (corresponding to the background and four edge categories). The network is trained with a deep supervision mechanism, where the total loss is a weighted sum of the losses from each output. The CSAM and CDCM modules are optional structures

First, the input image goes into the PiDiNet backbone network. This backbone has a series of Pixel Difference Convolution (PDC) modules. Unlike standard convolutional layers that only learn filters, the PDC modules actively calculate directional differences, like horizontal, vertical, and diagonal, between nearby pixels. This directly simulates the main steps of traditional edge detectors like Sobel and Canny [17,42]. This design helps the network create strong feature responses for edge areas in the early layers and reduces redundant information to process. This is the main reason why the network is so efficient.

Next, the network creates multi-scale feature maps with a structure similar to a Feature Pyramid Network (FPN). These feature maps hold information from fine local details to broad semantic context. This helps the network find edges at different scales more accurately.

Our core innovation lies in the Multi-class prediction stage. The original PiDiNet finally outputs a single-channel feature map, which is activated by a sigmoid function to obtain a binary edge probability map. To extend it to a multi-class setting, we restructured its output layer. Specifically, we changed the number of convolutional kernels in all four side output branches and the final fused output layer from 1 to C (in this work, the number of categories $C = 5$, corresponding to the background and four types of physical edges). Therefore, each output $\mathbf{E}*$ of the network is a C-channel feature map $\mathbf{E}* \in \mathbb{R}^{C\times H\times W}$, where each channel corresponds to the logits (unnormalized scores) of a specific class. Finally, these logits are converted into a categorical probability distribution $\hat{y}_i^{(c)}$ at each pixel via a softmax function applied along the channel dimension (as shown in Eq. (1)).

Lastly, we employ a deep supervision mechanism to optimize this complex network. We calculate the loss function for all five outputs (four side outputs $\mathbf{E}_1, \mathbf{E}_2, \mathbf{E}_3, \mathbf{E}_4$ and one final output $\mathbf{E}_f$) simultaneously. The predictions from the side outputs are upsampled to the input resolution via bilinear interpolation. The total loss is a weighted sum of their individual losses (Eq. (4)). This strategy significantly facilitates the training process by injecting gradients directly into the intermediate layers of the network, ensuring that features from shallow to deep layers are all optimized towards the common goal of Multi-class edge recognition. More importantly, this design does not alter the computational path of the original network during forward inference; thus, at test time, we still only use the final fused output $\mathbf{E}_f$, guaranteeing that the model's high efficiency remains completely unaffected.

To sum up, our framework skillfully leverages the efficient PiDiNet backbone, transforming it into a powerful Multi-class predictor through targeted output layer restructuring, and utilizes deep supervision to ensure training effectiveness. This enables our model to run at speeds exceeding 250 FPS while outputting rich, physically meaningful edge class information.

3.2 Network Architecture Adaptation

The original PiDiNet was meticulously designed for binary edge detection, with the ultimate goal of classifying each pixel as either "edge" or "non-edge." However, directly applying it to multi-class edge detection tasks presents a fundamental mismatch, which primarily manifests in its output layer design and the guidance provided by its loss function. This subsection elaborates on two key structural modifications we implemented to address this issue: output layer restructuring and the adaptation of the deep supervision mechanism.

3.2.1 Output Layer Restructuring

The original PiDiNet produces a single-channel output feature map, whose physical meaning represents the "edge saliency" or "edge strength" at each pixel location. Through a sigmoid activation function, these scalar values are mapped to the interval $[0, 1]$ and are directly interpreted as the probability of the pixel being an edge. This design inherently aligns more closely with a regression or binary classification problem.

To transform it into a Multi-class classification problem, we must fundamentally restructure the output component of the network. Our core modification involves expanding the number of channels in all output layers from 1 to C (in this work, the number of categories $C = 5$, corresponding to the background, depth edge, normal edge, reflectance edge, and illumination edge). This modification was applied consistently in two places. First, in each side-output branch, we changed the final 1×1 convolutional layer to produce 5

channels instead of 1. Second, we made the exact same change to the final output layer at the end of the network, ensuring it also outputs a 5-channel feature map.

Consequently, every output of the network (including the four side outputs $\mathbf{E}_1, \mathbf{E}_2, \mathbf{E}_3, \mathbf{E}_4$ and the final fused output $\mathbf{E}_f$) becomes a tensor with shape $\mathbb{R}^{5\times H\times W}$. Now, at each spatial location (i, j), we no longer have a single scalar value, but rather a 5-dimensional vector $\mathbf{z}_{i,j} \in \mathbb{R}^5$. This vector represents the unnormalized scores (logits) for the pixel belonging to each of the categories.

To convert these logits into a probability distribution, we introduce a softmax function applied along the channel dimension (the C dimension). Finally, the probability that pixel (i, j) is predicted to belong to class c is given by Eq. (1):

$$\hat{y}_{i,j}^{(c)} = \frac{\exp(z_{i,j}^{(c)})}{\sum_{k=1}^{C} \exp(z_{i,j}^{(k)})} \tag{1}$$

where $z_{i,j}^{(c)}$ is the value of the output tensor at channel c and location (i, j). The softmax function ensures that the predicted probabilities for all categories sum to 1, thereby explicitly defining the network's learning objective as a Multi-class classification task.

The theoretical advantage of this modification lies in its ability to allow the network to learn a discriminative feature representation for each pixel. The five output channels compete with each other, with each channel specializing in activating a specific type of edge. This enables the model to express complex information such as "this pixel is not an edge (channel 0 activated), but rather a strong depth discontinuity boundary (channel 1 activated)"—a capability unattainable with a single-channel output.

The structural difference between the original binary-output PiDiNet and our modified multi-class version is illustrated in Fig. 2, where the replacement of the single-channel Sigmoid layer with a five-channel Softmax layer enables pixel-wise semantic edge classification.

3.2.2 Deep Supervision Mechanism

PiDiNet inherits the Deep Supervision strategy from HED [24], a mechanism proven to effectively improve gradient flow, accelerate convergence, and enhance multi-scale feature fusion capabilities. However, the original deep supervision was tailored for binary edge detection, where the side output losses computed the error between binary prediction maps and binary ground truth maps. We fully preserve the architectural advantages of this mechanism but transform its supervision signals to adapt to the Multi-class task. First of all, we no longer compute a binary loss for each side output. Instead, each side output $\mathbf{E}k$ now directly produces a 5-channel logits map and possesses its own independent Multi-class prediction capability. The loss $\mathcal{L}_{\text{CE}}^{(k)}$ for each side output is computed against the same Multi-class ground truth label using a weighted cross-entropy loss defined in Eq. (2).

$$\mathcal{L}_{\text{CE}} = -\frac{1}{N} \sum i = 1^{N} \sum_{c=1}^{C} w_c \cdot \mathbf{1}_{[y_i=c]} \cdot \log(\hat{y}_i^{(c)}) \tag{2}$$

where i indexes pixels after alignment to the input resolution, $N = H \times W$ is the number of pixels, $c \in \{1, \ldots, C\}$ indexes semantic categories (background, depth, normal, reflectance, illumination), $w_c \geq 0$ are class-balance weights to mitigate the strong foreground–background imbalance, y_i is the categorical ground-truth label at pixel i, and $\mathbf{1}_{[y_i=c]}$ is the indicator that equals 1 when pixel i belongs to class c and 0 otherwise. The term $\hat{y}_i^{(c,k)} \in [0, 1]$ denotes the predicted probability assigned by side output k to class c at pixel i; it is obtained by applying a channel-wise softmax to the corresponding 5-dimensional logit vector at that pixel

(cf. Eq. (1)). In this way, the inner summation accumulates the negative log-likelihood for the true class at each pixel, while the outer summation averages this quantity over the entire image.

Next, since side outputs reside at different depths of the network, the spatial dimensions (H, W) of their feature maps progressively decrease. Before computing the loss, we restore each side output's prediction map to the full resolution of the input image via bilinear upsampling to ensure spatial alignment with the ground truth label map. The overall training objective of the network is to minimize the weighted sum of all five output losses, as shown in Eq. (3):

$$\mathcal{L}_{\text{total}} = \sum k = 1^{4} \gamma_k \mathcal{L}_{\text{CE}}^{(k)} + \gamma_f \mathcal{L}_{\text{CE}}^{(f)} \tag{3}$$

where γ_k and γ_f are hyperparameters that balance the contribution of each loss term. In our configuration, we set $\gamma_1 = 0.2$, $\gamma_2 = 0.2$, $\gamma_3 = 0.3$, $\gamma_4 = 0.3$, and $\gamma_f = 1.0$. This configuration means we assign slightly higher weights to deeper side outputs ($\mathbf{E}_3$, $\mathbf{E}_4$) compared to shallower ones ($\mathbf{E}_1$, $\mathbf{E}_2$), and let the final fused output $\mathbf{E}_f$ dominate the loss to guide the network towards optimizing its final performance.

The core advantages of this mechanism are:

1. Gradient Optimization: Gradients are injected into the network simultaneously from multiple levels, effectively mitigating the vanishing gradient problem and guiding the network to learn features relevant to Multi-class edges from shallow to deep layers.
2. Multi-Scale Feature Learning: Shallow side outputs (e.g., $\mathbf{E}_1$) primarily receive detailed information from lower layers, compelling them to focus on distinguishing fine-grained, low-level texture edges (e.g., fabric textures). In contrast, deep side outputs (e.g., $\mathbf{E}_4$) incorporate more high-level semantic information, prompting them to learn to identify semantic edges formed by object boundaries. This explicit, scale-specific supervision makes the network's multi-scale feature fusion more effective.
3. Inference-Phase Efficiency: Crucially, deep supervision is only active during the training phase. During testing and deployment, we discard all side outputs and use only the final fused output $\mathbf{E}_f$ for prediction. This means we gain all the performance benefits brought by deep supervision without incurring any additional computational overhead or latency during inference, perfectly preserving PiDiNet's original high efficiency.

In summary, our network structure adaptation is more than just increasing the number of channels. It is a task-oriented adjustment that turns a binary edge detector into a model that can handle multiple edge classes while keeping the efficiency and strengths of the original architecture.

3.3 Dynamic Class Balancing Strategy

The Multi-class edge detection task faces a severe challenge: extreme class imbalance. This imbalance manifests at two levels: first, non-edge background pixels dominate the image (typically exceeding 95%); second, significant disparities exist among pixels of different edge categories themselves—for instance, 'depth discontinuity' edges might be far more frequent than 'reflection' edges. Training directly with a standard Cross-Entropy (CE) loss function causes the model's predictions to be heavily biased towards the dominant class (background), preventing effective learning of those rare yet crucial edge categories.

To address this challenge, we designed a hierarchical, dynamic class balancing strategy. This strategy incorporates not only adaptive, data-driven weight computation but also innovatively introduces a manual adjustment mechanism based on domain knowledge.

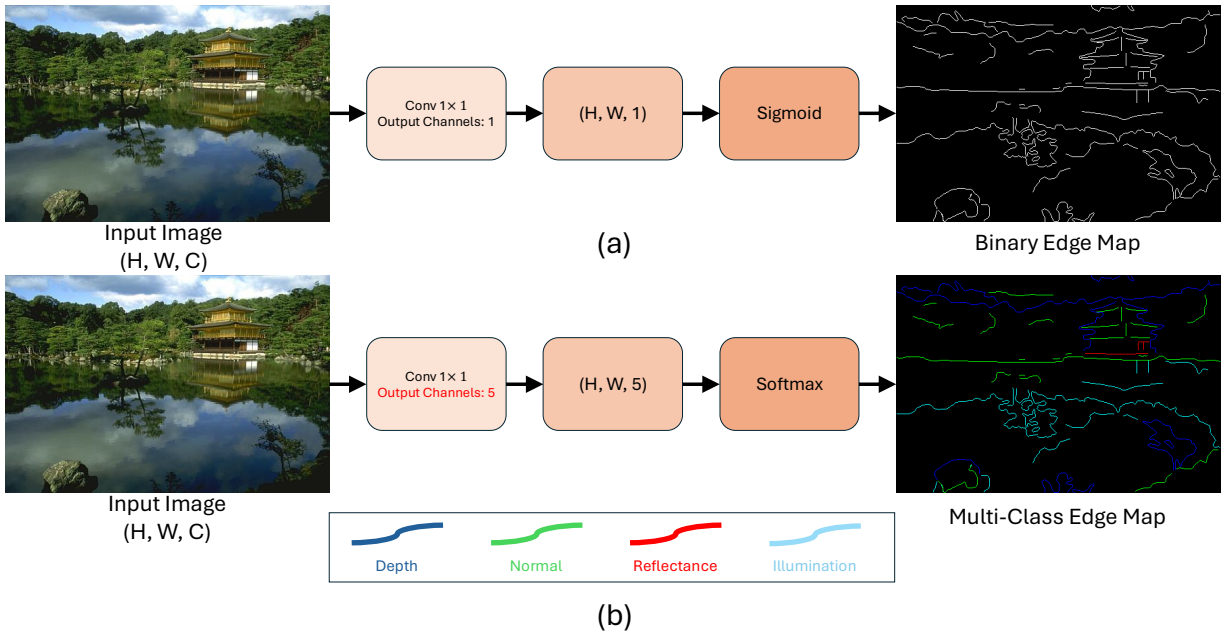

Figure 2: Schematic diagram of the network modification. The input is a 3-channel (C = 3) RGB image. (**a**) The original PiDiNet output layer, which produces a single-channel (C = 1) feature map for binary prediction using a Sigmoid function; (**b**) Our modified output layer, which produces a 5-channel (C = 5: Background + Depth/Normal/Reflectance/Illumination edges) feature map for multi-class prediction using a Softmax function. This modification is applied to all side output and final fusion layers

3.3.1 Weighted Cross-Entropy Loss Foundation

We adopt Weighted Cross-Entropy (WCE) as the foundational loss function. For a single pixel, the loss is calculated as:

$$\mathcal{L}_{\text{WCE}} = -\sum c = 1^{C} w_c \cdot \mathbf{1}_{[y_i=c]} \cdot \log(\hat{y}_i^{(c)}) \tag{4}$$

where y_i is the ground truth label, $\hat{y}_i$ is the predicted probability distribution, $\mathbb{1}[\cdot]$ is the indicator function, and w_c is the weight assigned to class c. By assigning larger weights w_c to rare classes, we force the model to pay more attention to these hard-to-classify samples during training.

3.3.2 Adaptive Weight Computation

Determining the weights $\mathbf{w} = [w_1, w_2, \ldots, w_C]$ is important. We compute these weights automatically based on the statistics of the training set. First, we count the total number of pixels N_c for each class c across the training set. Next, we calculate the frequency of each class as $f_c = N_c / \sum_{c'} N_{c'}$, and we apply the Median Frequency Balancing strategy to obtain the initial weights:

$$w_c^{\text{init}} = \frac{\text{median}(f_1, f_2, \ldots, f_C)}{f_c + \varepsilon} \tag{5}$$

where ε is a small value (default 1×10^{-6}) to prevent division by zero. The advantage of this method is its insensitivity to extreme frequency values, making it more stable than simple inverse frequency balancing. After all, Normalize the computed weights so that their mean is 1: $\mathbf{w} = \mathbf{w}/\text{mean}(\mathbf{w})$. This step ensures the loss function's scale does not change drastically, benefiting training stability.

3.3.3 Weight Adjustment Based on Physical Priors

Purely data-driven weights might not fully capture the importance differences of various edge categories in the specific task. For example, in our task, 'illumination edges' may contain critical information about shadows, highlights, etc., in a scene. Despite potentially having very few pixels, their physical significance is substantial.

Therefore, we introduce an adjustable scaling factor α_c for manual adjustment of the automatically computed weights:

$$w_c^{\text{final}} = \alpha_c \cdot w_c^{\text{init}} \tag{6}$$

According to our parameter settings, the scaling factors are specifically: $\alpha_{\text{depth}} = 0.5$ (Depth edge), $\alpha_{\text{normal}} = 0.4$ (Normal edge), $\alpha_{\text{reflect}} = 0.3$ (Reflectance edge), $\alpha_{\text{illum}} = 2.5$ (Illumination edge).

The core idea of this design is: we actively and purposefully guide the model's learning focus. By substantially increasing the focus on the rare illumination edge class while moderately reducing the weights of other edge classes, we further alleviate the imbalance between these categories, the background, and the illumination edges. This approach provides useful, domain knowledge-driven flexibility and controllability in addressing class imbalance, making it a practical improvement in our method.

3.3.4 Dice Loss Supplement and Total Loss Function

To further enhance the model's sensitivity to edge shapes and better handle the foreground-background imbalance, we introduce the Soft-Dice loss [62,63]. The Dice coefficient measures the overlap between predictions and ground truth, offering inherent robustness to class imbalance. The multi-class Dice loss is defined as:

$$\mathcal{L}_{\text{Dice}} = 1 - \frac{1}{C}\sum c = 1^{C} \frac{2\sum_i y_i^{(c)} \hat{y}_i^{(c)} + \varepsilon}{\sum_i y_i^{(c)} + \sum_i \hat{y}_i^{(c)} + \varepsilon} \tag{7}$$

Finally, the total loss $\mathcal{L}^{(k)}$ for each output (side output or final output) is a linear combination of the weighted cross-entropy loss and the Dice loss:

$$\mathcal{L}^{(k)} = \mathcal{L}_{\text{WCE}}^{(k)} + \lambda \cdot \mathcal{L}_{\text{Dice}}^{(k)} \tag{8}$$

where λ is the weight coefficient balancing the two loss terms, and ε is the smoothing term.

The overall training objective of the network is to minimize the weighted sum of losses from all five outputs, as shown in Eq. (9):

$$\mathcal{L}_{\text{total}} = \sum k = 1^{4} \gamma_k \mathcal{L}^{(k)} + \gamma_f \mathcal{L}^{(f)} \tag{9}$$

where γ_k and γ_f are hyperparameters balancing the contribution of each loss term.

4 Experiments

4.1 Experimental Setup

4.1.1 Dataset and Evaluation Metrics

We conducted comprehensive experiments on the BSDS-RIND benchmark [31], the most challenging dataset for multi-class edge detection, which provides 300 training and 200 test images with pixel-level

annotations for four physical edge categories: depth discontinuities, surface normal variations, material reflectance changes, and illumination boundaries. Owing to its diverse indoor and outdoor scenes with complex lighting and material properties, it offers a rigorous testbed for robustness evaluation. For quantitative assessment, we reported ODS, OIS, and AP by converting probability maps into binary edge maps with optimal thresholds, while efficiency was evaluated in terms of parameters, FPS, and memory footprint on an NVIDIA RTX 4090 at 512 × 512 resolution. In addition, extensive qualitative comparisons were conducted to examine edge continuity, localization accuracy, and class discrimination. To ensure fair comparability, we adhere to the standard BSDS-RIND protocol widely used in prior work. The benchmark defines exactly four physical edge categories (depth, normal, reflectance, illumination) with pixel-level labels across diverse scenes, providing a rigorous testbed. We also average metrics over multiple runs (mean ± std) to mitigate small-data variance.

4.1.2 Implementation Details

Our model is implemented using the PyTorch framework. Training employs the Adam optimizer with an initial learning rate of 1×10^{-3} and a StepLR scheduler that reduces the learning rate by a factor of 0.5 every 30 epochs. To accommodate high-resolution inputs while maintaining training stability, we use a DataLoader batch size of 1 but implement gradient accumulation with an iteration size of 24. This strategy effectively provides an equivalent batch size of 24 for parameter updates, enabling stable optimization while respecting GPU memory constraints. All input images and ground truth labels are resized to 512 × 512 resolution during both training and evaluation. The only data augmentation applied is random horizontal flipping with a probability of 0.5.

We fully enable PiDiNet's optional modules to maximize the model's expressive power: namely, the Spatial Attention Module (CSAM) and the Compact Dilated Convolution Module (CDCM). The network backbone is constructed using the `carv4` configuration to build PDC blocks for the `base` architecture.

Class balancing weights are automatically calculated on the training set using the median frequency method, and are further adjusted based on physical priors. The scaling factors are specifically set as follows: depth edges $\alpha_{\text{depth}} = 0.5$, normal edges $\alpha_{\text{normal}} = 0.4$, reflectance edges $\alpha_{\text{reflect}} = 0.3$, and illumination edges $\alpha_{\text{illum}} = 2.5$. This configuration aims to significantly enhance the model's focus on the rare yet important illumination edges.

Regarding the loss function, we utilize a hybrid of weighted cross-entropy and Dice loss. The weight coefficient λ for the Dice loss is set to 0.4, and the smoothing term ε is set to 1.0. The deep supervision weights are set to $\boldsymbol{\gamma} = [0.2, 0.2, 0.3, 0.3, 1.0]$.

The model is trained for 70 epochs. Training is performed on a dataset containing BSDS-RIND, and performance is monitored on a separate validation set to save the best checkpoint. All experiments are conducted on an NVIDIA GeForce RTX 4090 GPU.

4.2 Ablation Studies

As shown in Tables 1 and 2, we performed systematic ablation studies to validate the necessity and contribution of each proposed module. Starting from the original PiDiNet with binary output as the baseline, we incrementally introduced the multi-class output restructuring, dynamic class balancing strategy, and Dice loss function, and analyzed the impact of each component on performance. Furthermore, we examined different weight adjustment strategies, including no weighting, inverse frequency balancing, median frequency balancing, and our proposed method incorporating physical priors, to evaluate their effects on the detection performance of each class. We also investigated the effectiveness of deep supervision

in the multi-class setting, particularly the influence of different supervision weight configurations. All ablation studies were conducted with identical hyperparameter settings and on the same data split to ensure comparability.

Table 1: Impact of various ablation strategies

Method	Depth ODS ↑	Depth OIS ↑	Normal ODS ↑	Normal OIS ↑	Reflectance ODS ↑	Reflectance OIS ↑	Illumination ODS ↑	Illumination OIS ↑	FPS↑
Baseline (Multi-class PiDiNet)	0.530	0.554	0.318	0.344	0.320	0.365	0.187	0.230	253
+ CSAM	0.536	0.566	0.324	0.354	0.328	0.374	0.196	0.238	250
+ CDCM	0.540	0.565	0.330	0.352	0.324	0.368	0.190	0.236	249
+ Class Balancing	0.549	0.576	0.335	0.355	0.313	0.355	0.203	0.232	249
+ Dice Loss(=0.2)	**0.555**	**0.582**	**0.343**	**0.369**	**0.318**	**0.357**	**0.188**	**0.217**	**251**

Note: Bold numbers indicate the best performance among the compared settings for each edge category.

Table 2: Impact of various ablation strategies

Parameter (Dice loss = 0.4) ($\alpha_d, \alpha_n, \alpha_r, \alpha_i$)	Depth ODS ↑	Depth OIS ↑	Normal ODS ↑	Normal OIS ↑	Reflectance ODS ↑	Reflectance OIS ↑	Illumination ODS ↑	Illumination OIS ↑
0.8, 0.5, 0.5, 2.0	0.558	0.586	0.346	0.363	0.349	0.386	0.224	0.260
0.6, 0.5, 0.3, 2.5	0.568	0.599	0.348	0.365	0.368	0.394	0.235	0.286
0.5, 0.4, 0.1, 2.5	0.582	0.605	0.384	0.413	0.260	0.276	0.239	0.288
0.3, 0.2, 0.1, 3.0	0.579	0.599	0.354	0.365	0.209	0.220	0.229	0.281
0.5, 0.4, 0.3, 2.5	**0.598**	**0.617**	**0.386**	**0.416**	**0.387**	**0.415**	**0.245**	**0.285**

Note: Bold numbers indicate the best performance among the compared parameter settings for each edge category.

4.3 Comparative Experiments

As shown in Tables 3 and 4, we compared our proposed method with several state-of-the-art edge detection approaches, including deep learning-based models such as HED [24], RCF [26], and BDCN [54], efficient architectures such as DexiNed [29], and the multi-class edge detection framework RINDNet [31], as well as DOOBNet [64] and DeepLabv3+ [65], among others. Through both quantitative and qualitative comparisons, we observed that our method, while falling somewhat short of the latest methods in terms of accuracy, achieves results that are not far behind and provides a distinct advantage in speed and efficiency.

Table 3: Accuracy—efficiency trade-off across different methods

Method	Depth ODS ↑	Depth OIS ↑	Normal ODS ↑	Normal OIS ↑	Reflectance ODS ↑	Reflectance OIS ↑	Illumination ODS ↑	Illumination OIS ↑	FPS↑
HED	0.644	0.679	0.457	0.505	0.412	0.466	0.256	0.290	84
RCF	0.648	0.679	0.444	0.503	0.429	0.448	0.257	0.283	63
BDCN	0.628	0.661	0.427	0.484	0.358	0.458	0.151	0.219	56
DexiNed	0.637	0.673	0.444	0.486	0.402	0.454	0.157	0.199	38
RINDNet	0.678	0.702	0.474	0.513	0.468	0.52	0.264	0.306	32
DOOBNet	0.658	0.689	0.442	0.490	0.431	0.489	0.143	0.210	48
DeepLabv3+	0.535	0.579	0.366	0.398	0.297	0.338	0.103	0.150	107
Ours	0.598	0.617	0.386	0.416	0.387	0.415	0.245	0.285	**251**

Note: Bold numbers highlight the best trade-off between accuracy and real-time efficiency (FPS).

Table 4: Model parameters

Method	Parameters↓
HED	14.7 M
RCF	14.8 M
BDCN	16.3 M
DexiNed	4.6 M
RINDNet	16.5 M
DOOBNet	3.2 M
DeepLabv3+	5.8 M
Ours	**0.8 M**

Note: Bold numbers indicate the most compact (smallest) model in terms of parameter count.

4.4 Analysis and Conclusion

As shown in Fig. 3, a comparison of multi-class edge detection results between PIDINet-MC (the model proposed in this paper) and the existing multi-class edge detection framework RINDNet on the same input images is presented. Different colors represent four types of physical edges: blue for depth discontinuities, green for surface normal variations, red for material reflectance differences, and cyan for illumination changes. The figure also shows the final fused detection results of both models. The visualization demonstrates that PIDINet-MC can effectively distinguish the four types of physical edges while maintaining real-time inference (over 250 FPS). Although its edge localization accuracy and class discrimination are slightly lower than RINDNet (the current high-precision multi-class edge detection model), it can clearly capture the core contours of each edge type. In particular, for rare classes such as illumination edges (cyan), the proposed dynamic class balancing strategy (combined with physics-informed weight adjustments) effectively reduces missed detections. This validates PIDINet-MC's advantage in balancing accuracy and efficiency, enabling real-time acquisition of multi-class edge semantic information in resource-constrained scenarios.

Secondly, the ablation studies demonstrate that each proposed component contributes to overall performance improvements. Introducing multi-class output restructuring provides the foundation for semantic edge prediction, while class balancing and the Dice loss further enhance detection quality, particularly for rare categories such as illumination edges. Adjusting class weights based on physical priors shows noticeable benefits, indicating the importance of incorporating domain knowledge into the training process.

Hyperparameter exploration highlights the sensitivity of the model to weight settings, where moderate adjustments yield more balanced results across all categories. Deep supervision also facilitates stable training and enhances feature learning across multiple scales, without compromising inference efficiency.

Most importantly, comparative experiments reveal that, while our model does not reach the same accuracy levels as heavier frameworks like RINDNet or BDCN, it achieves competitive results with a significant advantage in speed and computational efficiency. Operating at over 250 FPS with minimal parameters, the model is well suited for real-time applications where both efficiency and semantic understanding are critical.

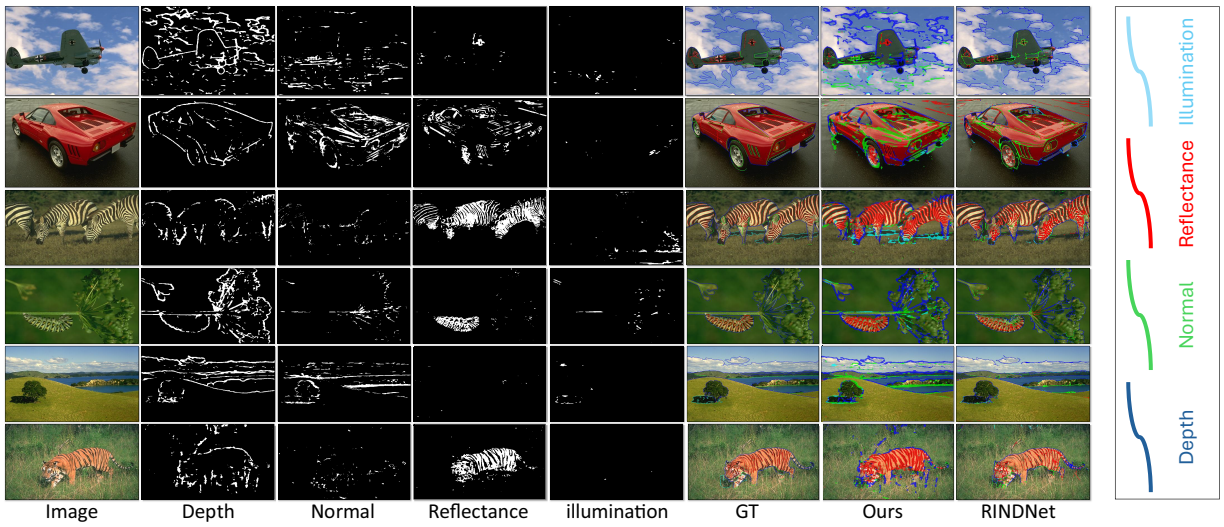

Figure 3: Visualization and comparison of multi-class edge detection results between PIDINet-MC and RINDNet on the BSDS-RIND dataset

Overall, the results suggest that the proposed framework offers a practical balance between accuracy and efficiency, and it provides a solid foundation for further exploration in lightweight, real-time multi-class edge detection.

5 Conclusion

This paper addresses the challenging problem of efficient and accurate multi-class edge detection by proposing a real-time framework based on the lightweight PiDiNet architecture. We restructure the network's output layer from binary to multi-class semantic output, incorporate a deep supervision mechanism to enhance multi-scale feature learning, and design a dynamic class-balancing strategy that combines data-driven weight computation with physically-informed manual adjustments to handle class imbalance. Experimental results on the BSDS-RIND dataset demonstrate that our model achieves real-time inference at 250 FPS while maintaining reasonable accuracy.

However, the performance remains limited in several important aspects. Most notably, detection of illumination edges falls significantly behind more computationally intensive models like RINDNet (ODS: 0.245 vs. 0.264), revealing a fundamental limitation of our lightweight architecture in capturing subtle lighting variations. The dynamic class-balancing strategy, while helpful, relies on heuristic adjustments that may not generalize well across different domains. Furthermore, our approach is constrained by the four-class taxonomy of BSDS-RIND, which cannot represent the full spectrum of edge phenomena encountered in practical applications.

These limitations define clear directions for future work. We plan to develop more automated weight adjustment strategies to reduce manual intervention, extend the framework to recognize a broader range of edge categories, and explore advanced data augmentation techniques specifically designed to improve performance on challenging cases like illumination edges. Adapting the model for more complex real-time scene understanding tasks presents another important avenue for future research.

Acknowledgement: The authors would like to express their sincere gratitude to the BSDS-RIND dataset developers for providing the benchmark used in this study.

Funding Statement: This work is supported by the National Natural Science Foundation of China 62402171.

Author Contributions: The authors confirm contribution to the paper as follows: Study conception and design: Mingming Huang, Yunfan Ye; Data collection and experimental implementation: Mingming Huang; Analysis and interpretation of results: Mingming Huang, Yunfan Ye, Zhiping Cai; Draft manuscript preparation: Mingming Huang, Yunfan Ye; Critical revision and final approval: Zhiping Cai. All authors reviewed the results and approved the final version of the manuscript.

Availability of Data and Materials: The dataset (BSDS-RIND) used in this study is publicly available and cited in the manuscript. All experimental codes and model configurations will be made available upon reasonable request to the corresponding author.

Ethics Approval: Not applicable. This study did not involve human participants or animal subjects. All image data were obtained from publicly available benchmark datasets with appropriate citations.

Conflicts of Interest: The authors declare no conflicts of interest to report regarding the present study.

References

1. Canny J. A computational approach to edge detection. IEEE Trans Pattern Anal Mach Intell. 2009;6(6):679–98. doi:10.1109/tpami.1986.4767851.
2. Torre V, Poggio TA. On edge detection. IEEE Trans Pattern Anal Mach Intell. 1986;2(2):147–63. doi:10.1109/tpami.1986.4767769.
3. Li J, Wang Z, Pan Z, Liu Q, Guo D. Looking at boundary: siamese densely cooperative fusion for salient object detection. IEEE Trans Neural Netw Learn Syst. 2021;34(7):3580–93. doi:10.1109/tnnls.2021.3113657.
4. Zhu G, Li J, Guo Y. Supplement and suppression: both boundary and nonboundary are helpful for salient object detection. IEEE Trans Neural Netw Learn Syst. 2021;34(9):6615–27. doi:10.1109/tnnls.2021.3127959.
5. Ben Chaabane S, Bushnag A. Color edge detection using multidirectional sobel filter and fuzzy fusion. Comput Mater Contin. 2023;74(2):2839–52. doi:10.32604/cmc.2023.032760.
6. Li M, Chen D, Liu S, Liu F. Semisupervised boundary detection for aluminum grains combined with transfer learning and region growing. IEEE Trans Neural Netw Learn Syst. 2021;34(9):6158–72. doi:10.1109/tnnls.2021.3133760.
7. Rother C, Kolmogorov V, Blake A. GrabCut interactive foreground extraction using iterated graph cuts. ACM Trans Graph. 2004;23(3):309–14.
8. Li C, Xia W, Yan Y, Luo B, Tang J. Segmenting objects in day and night: edge-conditioned CNN for thermal image semantic segmentation. IEEE Trans Neural Netw Learn Syst. 2020;32(7):3069–82. doi:10.1109/tnnls.2020.3009373.
9. Ye Y, Yi R, Gao Z, Zhu C, Cai Z, Xu K. Nef: neural edge fields for 3D parametric curve reconstruction from multi-view images. In: Proceedings of the 2023 IEEE/CVF Conference on Computer Vision and Pattern Recognition; 2023 Jun 17–24; Vancouver, BC, Canada. p. 8486–95.
10. Nazeri K, Ng E, Joseph T, Qureshi FZ, Ebrahimi M. Edgeconnect: generative image inpainting with adversarial edge learning. arXiv:1901.00212. 2019.
11. Xiong W, Yu J, Lin Z, Yang J, Lu X, Barnes C, et al. Foreground-aware image inpainting. In: Proceedings of the 2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition; 2019 Jun 15–20; Long Beach, CA, USA. p. 5840–8.
12. Alkhaldi NA, Halawani HT. Intelligent machine learning enabled retinal blood vessel segmentation and classification. Comput Mater Contin. 2023;74(1):399–414. doi:10.32604/cmc.2023.030872.
13. Pourreza R, Zhuge Y, Ning H, Miller R. Brain tumor segmentation in MRI scans using deeply-supervised neural networks. In: International MICCAI Brainlesion Workshop. Cham, Switzerland: Springer; 2017. p. 320–31 doi: 10.1007/978-3-319-75238-9_28.
14. Zhao B, Li X. Edge-aware network for flow-based video frame interpolation. IEEE Trans Neural Netw Learn Syst. 2022;35(1):1401–8. doi:10.1109/tnnls.2022.3178281.

15. Fang F, Li J, Yuan Y, Zeng T, Zhang G. Multilevel edge features guided network for image denoising. IEEE Trans Neural Netw Learn Syst. 2020;32(9):3956–70. doi:10.1109/tnnls.2020.3016321.
16. Elder JH, Goldberg RM. Image editing in the contour domain. In: Proceedings of 1998 IEEE Computer Society Conference on Computer Vision and Pattern Recognition; 1998 Jun 23–25; Santa Barnara, CA, USA. p. 374–81.
17. Sobel I, Feldman G. A 3 × 3 isotropic gradient operator for image processing. Stanford, CA, USA: Stanford University, Stanford Artificial Intelligence Laboratory (SAIL); 1990.
18. Arbelaez P, Maire M, Fowlkes C, Malik J. Contour detection and hierarchical image segmentation. IEEE Trans Pattern Anal Mach Intell. 2010;33(5):898–916. doi:10.1109/tpami.2010.161.
19. Dollár P, Zitnick CL. Fast edge detection using structured forests. IEEE Trans Pattern Anal Mach Intell. 2014;37(8):1558–70. doi:10.1109/tpami.2014.2377715.
20. Jung H, Park H, Jung HS, Lee K. Enhancing building facade image segmentation via object-wise processing and cascade U-Net. Comput Mater Contin. 2024;81(2):2261–79. doi:10.32604/cmc.2024.057118.
21. Bertasius G, Shi J, Torresani L. Deepedge: a multi-scale bifurcated deep network for top-down contour detection. In: Proceedings of the 2015 IEEE Conference on Computer Vision and Pattern Recognition; 2015 Jun 7–12; Boston, MA, USA. p. 4380–9.
22. Kokkinos I. Pushing the boundaries of boundary detection using deep learning. arXiv:1511.07386. 2015.
23. Shen W, Wang X, Wang Y, Bai X, Zhang Z. Deepcontour: a deep convolutional feature learned by positive-sharing loss for contour detection. In: Proceedings of the 2015 IEEE Conference on Computer Vision and Pattern Recognition; 2015 Jun 7–12; Boston, MA, USA. p. 3982–91.
24. Xie S, Tu Z. Holistically-nested edge detection. In: Proceedings of the 2015 IEEE International Conference on Computer Vision; 2015 Dec 7–13; Santiago, Chile. p. 1395–403.
25. Wang Y, Zhao X, Huang K. Deep crisp boundaries. In: Proceedings of the 2017 IEEE Conference on Computer Vision and Pattern Recognition; 2017 Jul 21–26; Honolulu, HI, USA. p. 3892–900.
26. Liu Y, Cheng MM, Hu X, Wang K, Bai X. Richer convolutional features for edge detection. In: Proceedings of the 2017 IEEE Conference on Computer Vision and Pattern Recognition; 2017 Jul 21–26; Honolulu, HI, USA. p. 3000–9.
27. Shi P, Tang Y, Li Y, Dong X, Sun Y, Yang A. Unsupervised monocular depth estimation with edge enhancement for dynamic scenes. Comput Mater Contin. 2025;84(2):3321–43. doi:10.32604/cmc.2025.065297.
28. Bakirci M, Demiray A. Autonomous navigation of service robots in complex industrial environments with SAR-based vision and advanced detection for industry 4.0. In: 2025 International Russian Smart Industry Conference (SmartIndustryCon); 2025 Mar 24–28; Sochi, Russia. p. 71–6.
29. Su Z, Liu W, Yu Z, Hu D, Liao Q, Tian Q, et al. Pixel difference networks for efficient edge detection. In: Proceedings of the 2021 IEEE/CVF International Conference on Computer Vision; 2021 Oct 11–17; Montreal, QC, Canada. p. 5117–27.
30. Liu Y, Lew MS. Learning relaxed deep supervision for better edge detection. In: Proceedings of the 2016 IEEE Conference on Computer Vision and Pattern Recognition; 2016 Jun 27–30; Las Vegas, NV, USA. p. 231–40.
31. Pu M, Huang Y, Guan Q, Ling H. Rindnet: Edge detection for discontinuity in reflectance, illumination, normal and depth. In: Proceedings of the 2021 IEEE/CVF International Conference on Computer Vision; 2021 Oct 10–17; Montreal, QC, Canada. p. 6879–88.
32. Gong XY, Su H, Xu D, Zhang ZT, Shen F, Yang HB. An overview of contour detection approaches. Int J Autom Comput. 2018;15(6):656–72. doi:10.1007/s11633-018-1117-z.
33. Xuan W, Huang S, Liu J, Du B. FCL-Net: towards accurate edge detection via Fine-scale Corrective Learning. Neural Netw. 2022;145(5):248–59. doi:10.1016/j.neunet.2021.10.022.
34. Hu Y, Chen Y, Li X, Feng J. Dynamic feature fusion for semantic edge detection. arXiv:1902.09104. 2019.
35. Yu Z, Feng C, Liu MY, Ramalingam S. Casenet: deep category-aware semantic edge detection. In: Proceedings of the 2017 IEEE Conference on Computer Vision and Pattern Recognition; 2017 Jul 21–26; Honolulu, HI, USA. p. 5964–73.

36. Ramamonjisoa M, Du Y, Lepetit V. Predicting sharp and accurate occlusion boundaries in monocular depth estimation using displacement fields. In: Proceedings of the 2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition; 2020 Jun 13–19; Seattle, WA, USA. p. 14648–57.
37. Wang X, Fouhey D, Gupta A. Designing deep networks for surface normal estimation. In: Proceedings of the 2015 IEEE Conference on Computer Vision and Pattern Recognition; 2015 Jun 7–12; Boston, MA, USA. p. 539–47.
38. Wu Q, Zhang W, Kumar BV. Strong shadow removal via patch-based shadow edge detection. In: 2012 IEEE International Conference on Robotics and Automation; 2012 May 14–18; St. Paul, MN, USA. p. 2177–82.
39. Yang F, Zhang L, Yu S, Prokhorov D, Mei X, Ling H. Feature pyramid and hierarchical boosting network for pavement crack detection. IEEE Trans Intell Transp Syst. 2019;21(4):1525–35. doi:10.1109/tits.2019.2910595.
40. Kim K, Torii A, Okutomi M. Joint estimation of depth, reflectance and illumination for depth refinement. In: Proceedings of the 2015 IEEE International Conference on Computer Vision Workshops; 2015 Dec 7–13; Santiago, Chile. p. 1–9.
41. Ming Y, Li H, He X. Contour completion without region segmentation. IEEE Trans Image Process. 2016;25(8):3597–611. doi:10.1109/tip.2016.2564646.
42. Liufu X, Tan C, Lin X, Qi Y, Li J, Hu JF. SAUGE: taming SAM for uncertainty-aligned multi-granularity edge detection. In: Proceedings of the 39th AAAI Conference on Artificial Intelligence. Vol. 39; 2025 Feb 25–Mar 4; Philadelphia, PA, USA. p. 5766–74.
43. Yang W, Chen XD, Wu W, Qin H, Yan K, Mao X, et al. Boosting deep unsupervised edge detection via segment anything model. IEEE Trans Ind Inform. 2024;20(6):8961–71. doi:10.1109/tii.2024.3376726.
44. Ahmed AS. Comparative study among Sobel, Prewitt and Canny edge detection operators used in image processing. J Theor Appl Inf Technol. 2018;96(19):6517–25.
45. Lu Z, Wang Fl, Chang Yq. Improved Canny algorithm for edge detection. J Northeast Univ Nat Sci. 2007;28(12):1681.
46. Zhou C, Huang Y, Pu M, Guan Q, Deng R, Ling H. Muge: multiple granularity edge detection. In: Proceedings of the 2024 IEEE/CVF Conference on Computer Vision and Pattern Recognition; 2024 Jun 16–22; Seattle, WA, USA. p. 25952–62.
47. Akinlar C, Topal C. EDLines: a real-time line segment detector with a false detection control. Pattern Recognit Lett. 2011;32(13):1633–42. doi:10.1016/j.patrec.2011.06.001.
48. Deng R, Liu S. Deep structural contour detection. In: Proceedings of the 28th ACM International Conference on Multimedia. New York, NY, USA: ACM; 2020. p. 304–12.
49. Deng R, Shen C, Liu S, Wang H, Liu X. Learning to predict crisp boundaries. In: Computer vision—ECCV 2018. Cham, Switzerland: Springer; 2018. p. 562–78. doi:10.1007/978-3-030-01231-1_35.
50. Maninis KK, Pont-Tuset J, Arbeláez P, Van Gool L. Convolutional oriented boundaries: from image segmentation to high-level tasks. IEEE Trans Pattern Anal Mach Intell. 2017;40(4):819–33. doi:10.1109/tpami.2017.2700300.
51. Poma XS, Riba E, Sappa A. Dense extreme inception network: towards a robust cnn model for edge detection. In: Proceedings of the 2020 IEEE/CVF Winter Conference on Applications of Computer Vision; 2020 Mar 1–5; Snowmass Village, CO, USA. p. 1923–32.
52. Xu D, Ouyang W, Alameda-Pineda X, Ricci E, Wang X, Sebe N. Learning deep structured multi-scale features using attention-gated crfs for contour prediction. In: Advances in Neural Information Processing Systems. Cambridge, MA, USA: MIT Press; 2017.
53. Elharrouss O, Hmamouche Y, Idrissi AK, El Khamlichi B, El Fallah-Seghrouchni A. Refined edge detection with cascaded and high-resolution convolutional network. Pattern Recognit. 2023;138(1):109361. doi:10.1016/j.patcog.2023.109361.
54. He J, Zhang S, Yang M, Shan Y, Huang T. Bi-directional cascade network for perceptual edge detection. In: Proceedings of the 2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition; 2019 Jun 15–20; Long Beach, CA, USA. p. 3828–37.
55. Pu M, Huang Y, Liu Y, Guan Q, Ling H. Edter: edge detection with transformer. In: Proceedings of the 2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition; 2022 Jun 18–24; New Orleans, LA, USA. p. 1402–12.

56. Zhou C, Huang Y, Pu M, Guan Q, Huang L, Ling H. The treasure beneath multiple annotations: an uncertainty-aware edge detector. In: Proceedings of the 2023 IEEE/CVF Conference on Computer Vision and Pattern Recognition; 2023 Jun 17–24; Vancouver, BC, Canada. p. 15507–17.
57. Cetinkaya B, Kalkan S, Akbas E. Ranked: addressing imbalance and uncertainty in edge detection using ranking-based losses. In: Proceedings of the 2024 IEEE/CVF Conference on Computer Vision and Pattern Recognition; 2024 Jun 16–22; Seattle, WA, USA. p. 3239–49.
58. Bansal A, Kowdle A, Parikh D, Gallagher A, Zitnick L. Which edges matter?. In: Proceedings of the 2013 IEEE International Conference on Computer Vision Workshops; 2013 Dec 2–8; Sydney, NSW, Australia. p. 578–85.
59. Marr D. Vision: a computational investigation into the human representation and processing of visual information. Cambridge, MA, USA: MIT Press; 2010.
60. Hornauer J, Belagiannis V. Gradient-based uncertainty for monocular depth estimation. In: European Conference on Computer Vision. Cham, Switzerland: Springer; 2022. p. 613–30.
61. Wang Z, Li K, Wang X, Lee A. An image edge detection algorithm based on Multi-Feature fusion. Comput Mater Contin. 2022;73(3):4995–5009. doi:10.32604/cmc.2022.029650.
62. Deng R, Liu S, Wang J, Wang H, Zhao H, Zhang X. Learning to decode contextual information for efficient contour detection. In: Proceedings of the 29th ACM International Conference on Multimedia; 2021 Oct 20–24; Chengdu, China. p. 4435–43.
63. Deng R, Chen ZM, Chen H, Hu J. Learning to refine object boundaries. Neurocomputing. 2023;557(1):126742. doi:10.1016/j.neucom.2023.126742.
64. Wang G, Wang X, Li FW, Liang X. Doobnet: deep object occlusion boundary detection from an image. In: Asian Conference on Computer Vision. Cham, Switzerland: Springer; 2018. p. 686–702.
65. Peng H, Xue C, Shao Y, Chen K, Xiong J, Xie Z, et al. Semantic segmentation of litchi branches using DeepLabV3+ model. IEEE Access. 2020;8:164546–55. doi:10.1109/access.2020.3021739.