Multi-scale pseudo labeling for unsupervised deep edge detection

Zhou, Changsheng; Yuan, Chao; Wang, Hongxin; Li, Lei; Oehmcke, Stefan; Liu, Junmin; Peng, Jigen

Published in:
Knowledge-Based Systems

DOI:
10.1016/j.knosys.2023.111057

Publication date:
2023

Document version
Publisher's PDF, also known as Version of record

Document license:
CC BY-NC-ND

Citation for published version (APA):
Multi-scale pseudo labeling for unsupervised deep edge detection

Changsheng Zhou a,∗, Chao Yuan a, Hongxian Wang a, Lei Li b, Stefan Oehmcke b, Junmin Liu c, Jigen Peng a

a School of Mathematics and Information Science, Guangzhou University, China
b Department of Computer Science, University of Copenhagen, Denmark
c School of Mathematics and Statistics, Xi’an Jiaotong University, China

A B S T R A C T

Deep learning currently rules edge detection. However, the impressive progress heavily relies on high-quality manually annotated labels which require a significant amount of labor and time. In this study, we propose a novel unsupervised learning framework for deep edge detection. It adopts a gradient-based method to generate scale-dependent pseudo edge maps, which match with the hierarchical structure of deep networks. It leverages both the representation learning capability of deep learning, and the simplicity of traditional methods. Experiments on three popular data sets show that the proposed method can suppress non-object edges and reduce the gap with its supervised counterpart due to the introduction of information of various scales and smoothing strategy.

1. Introduction

Edge detection or boundary detection is a fundamental task in computer vision [1,2]. It can capture salient object boundaries in digital images and contribute to high-level semantic tasks, such as object proposal [3,4], object detection [5], and semantic segmentation [6]. In spite of the distinction between edge and boundary [7], we in this paper indiscriminately use these two terminologies, both of which mean boundaries of objects of interest rather than texture-like or noise-induced edges.

Recently, influenced by the success of deep learning, edge detection has entered the data-driven era. The last decade has witnessed the rise of edge detectors [8–11] based on deep networks [12,13]. Detectors of this kind usually are trained in an end-to-end manner and can generally achieve higher accuracy than traditional methods (e.g., Canny detector [14], Structured Edge or SE [15]). Some of them even outperform human annotators [10,16]. A large part of these improvements stems from large-scale and high-quality data sets (e.g., BSDS500 [6], NYUDv2 [17], Multicue [18]) where the labels are from manual annotation. To fully exploit the potential of deep edge detectors, coarse edges extracted from the PASCAL VOC Context data set [19] are also used to augment edge labels [20–22]. Undoubtedly, manual annotation has long become the cornerstone of deep edge detectors.

Although more high-quality labels mean higher prediction performance [23,24], it is not an trivial work to acquire large-scale and high-quality annotations. On the one hand, edge annotation is time-consuming for human annotators, and it can take several minutes for one image of normal size [25]. This is one of the main reasons why the famous BSDS500 data set [6] has only 500 annotated images, much less than other well-known data sets (e.g., ImageNet [26] with about 1.2 millions images or COCO [27] with about 0.3 millions images). On the other hand, human annotators might disagree with one another whether a pixel is part of an edge or not. For example, images from Multicue data set [18] are labeled by six people, in which some salient edges shows position discrepancy. This inconsistency leads to training difficulty [10,21].

Actually, real-world objects exhibit structures of different hierarchies at different scales [28,29]. Taking "horses" in Fig. 1 as an example, classical Canny detector [14] clearly shows the scale-dependent property of edges in Fig. 2. At a fine scale (σ = 1.0), the edge map shows lots of details, such as grass, horses’ mane, barbed wire fence, etc. As the scale becomes coarser (σ = 2.0, 3.0, 4.0), some parts of horses, such as ears, heads, legs, gradually change from clearly visible to difficult to identify. By going further (σ = 5.0, 6.0), only rough outlines remain, and the corresponding objects cannot even be discerned. Therefore, scale-dependent information is necessary to detect meaningful edges. Although the hierarchical structure of deep networks contains information of various scales, there is currently a lack of effective unsupervised...
methods to extract. Our intuition is to combine the hierarchy of deep networks with traditional detectors at different scales.

It is theorized that edges are utilized in the early stages of human vision [30,31], which means edge detection should be a low-level task. Early pioneering methods [14,32] have shown convincing results by using low-level cues in an unsupervised way. Bio-inspired methods [33,34] also indicate that models constructed in a hierarchical and parallel way could not only perceive edges and contours in a straightforward and unsupervised manner, but also have the property of biological interpretability. This motivates the exploration of unsupervised learning approaches for deep edge detectors without expensive edge annotations.

Moreover, unsupervised learning approaches have gained more attention in recent years due to the powerful flexibility and the efficient utilization of unlabeled data [35–37]. Based on the well-designed pretext tasks, models of this kind learn useful features and generalize well to a wide range of downstream tasks [35,37]. However, few methods explore the potential of unsupervised learning in deep edge detection. One groundbreaking work proposes to learn motion edges simultaneously estimating edges and geometries (depths and normals) [38]. Early representative works result in popular detectors, or imitating the way how human vision works change as edges mainly in an unsupervised way, whether by computing image gradients [1,42], or imitating the way how human vision works [34,35,37]. However, supervised methods, such as classical machine learning based methods [15,36], generally have better performance, not to mention the deep learning based methods [10,11]. A comprehensive analysis is beyond the scope of this study, and we refer readers to the recent review articles [2,37] for more details.

2. Related works

Edge detection, as a fundamental problem in computer vision, has always been of interest to researchers since around 1970s. Lots of excellent methods have been proposed, whether in supervised or unsupervised scenarios. Early pioneering works treat the abrupt intensity change as edges mainly in an unsupervised way, whether by computing image gradients [1,42], or imitating the way how human vision works [34,35,37]. However, supervised methods, such as classical machine learning based methods [15,36], generally have better performance, not to mention the deep learning based methods [10,11]. A comprehensive analysis is beyond the scope of this study, and we refer readers to the recent review articles [2,37] for more details.

2.1. Traditional edge detection

Intuitively, edges are commonly treated as abrupt changes in image intensity, which then are detected by the first-order and/or second-order gradients. Early representative works result in popular detectors, such as Roberts [42], Laplacian of Gaussian (LoG) [1], etc. One well-known and widely-used method is the Canny detector [14]. Based on three general criteria of good detection, good localization, and single response, Canny describes the optimal detector as a sum of four exponential terms, which can be approximately implemented as the first derivative of Gaussian. The proposed non-maximum suppression also becomes a standard post-processing step afterwards. Without a doubt,
the mathematical rigor and simplicity of implementation makes it one of the most popular edge detector. Another excellent unsupervised edge detector is based on point-wise mutual information (PMI) [32]. PMI assumes that pixels on the same object should get higher statistical dependence than those on different objects. By combining the induced affinity with spectral clustering, PMI achieves amazing results, which makes it the best unsupervised edge detector at present. However, given an image of regular size, PMI needs to run several minutes for the state-of-the-art result. The time cost hinders its popularization, especially compared with Canny detector.

To get similar or even better result than human vision, studies in neuroscience also provide many promising clues about edge perceiving [30,31]. To model the classical and non-classical receptive field [54–56], bio-inspired methods of this kind use the difference of Gaussian or Gabor filter [57,58] to generate and regulate the response to edges. To get better performance, it is gradually realized that the importance of multi-level and multi-channel information. More complex strategies, such as neural dynamics [44] and spatial sparse color opponent processing [59], are developed. On the whole, bio-inspired methods are naturally interpretable compared with other methods, but in terms of performance, they have no obvious advantage.

Whether gradient-based methods or bio-inspired methods, one common property is that most of previous works are unsupervised, which means that they do not need manually annotated labels. Compared with learning-based methods (e.g., SketchTokens [52], gPb [6], SE [15]), these unsupervised methods cannot achieve satisfactory results. But they do not need the expensive manual labeling and lengthy learning process, which makes them simple and fast. These properties make them particularly suitable for resource-limited conditions.

2.2. Deep edge detection

Deep edge detection is a kind of method for edge detection using deep neural networks [12,13]. These methods not only effectively improve the accuracy of edge detection, but also greatly alleviate the difficulties of feature design and hyper-parameter selection in traditional methods. The leading methods of this kind [52,60,61] normally draw on the philosophy of traditional thoughts. They usually follows a multi-step pipeline, in which deep neural networks are mainly used to extract features. However, this field has been dominated soon by the end-to-end image-to-image deep detectors, due to the simpler process and the superior performance.

One of the representative methods is the Holistically-nested Edge Detection (HED) [10]. It uses the VGG-16 network to extract multi-level features, and each feature yields an side-output edge map. All of these side outputs are then fused as the final edge map. To train the network, HED employs the deep supervision strategy [62] to make output semantically meaningful. Additionally, to tackle the issue of gradient explosion in high layers, HED uses a consensus sampling approach. That is to say, only pixels labeled by at least three annotators are treated as ground truths, and experiments show that the simple approach can improve the performance. As a successful end-to-end deep edge detector, HED inspires a lot of follow-up works. The Relaxed Deep Supervision (RDS) [63] argues that it is the inconsistency between the diverse representations of hierarchical layers and the identical ground-truth labels that leads to false positives. RDS then constructs relaxed labels for each side-output branch by combining the manual labels with Canny or SE labels. However, it is difficult to determine layer-specific scales by manual intervention. To this end, Bi-Directional Cascade Network (BDCN) [64] uses the shallow-to-deep and deep-to-shallow structure together with scale enhancement module to fuse the multi-scale features. To capture more complex data structure, Rich Convolutional Features (RCF) [21] takes into consideration all convolutional features to fully exploit multi-scale and multi-level information. To improve the learning capacity of fine-level branches, Fine-scale Corrective Learning Net (FCL-Net) [65] develops a top-down attentional guiding and a pixel-level weighting module, and both of them help to detect fine-scale edges with high confidence.

Another kind of deep edge detectors use the encoder–decoder structure, which is popular for segmentation [66]. The Convolutional Encoder-Decoder Network (CEDN) [11] first compresses an image into a dense representation by the encoder and then decompresses the representation into an edge map as output. It has only one output branch, and mainly focuses on the object-level edge detection. A similar architecture DeepCrack [67] fuses those features from its encoder and decoder at the same scale, and it can detect for example pavement cracks well. Uncertainty-aware Edge Detector (UAED) [68] is also based on the encoder–decoder architecture, but it has two decoders. The first is to estimate a mean, and the second is to estimate a variance, and the final edge map is generated from a Gaussian distribution with the estimated mean and variance as the priori.

Additionally, other architectures are also explored for edge detection. For example, Contourgian [69] uses the generative adversarial networks [70] to learn adversarial similarity between ground truths and predictions. More recently, EDTER [16] and SAM [71] use the recently popular transformer [72] to build the network architecture. Although they achieve higher accuracy, these networks become more complex and require more resources for training.

Although making great progress, deep edge detection relies heavily on high-quality manual labels, and only a few of works attempt to learn detectors in an unsupervised manner. One pioneering work [38] explores the possibility of learning deep detectors on videos without human supervision. This method uses optical flow [73] and an initial edge map from image gradient to detect motion edges, and gradually evolves between motion estimation and edge detection. However, the inter-frame difference is an important reliance, and therefore, this method requires a large corpus of video data, rather than individual images. Later, the LEGO algorithm [39] introduces a 3D as-smooth-as-possible prior to jointly detect depth, normal and edge in an unsupervised manner. Especially, it use a L2 regularization term to prevent trivial predictions. However, this method are still based on continuous video frames and does not work on individual images. Another related research is the work about weakly supervised object boundaries [40]. To relax the requirement of manual labels, this method propose to detect object-centric edges with the help of bounding box annotations. Experiments show that this kind of annotations is helpful both in supervised and unsupervised situation. But, it should be noted that bounding boxes are still difficult to annotate manually.

In summary, deep edge detection is superior to traditional methods, especially in terms of performance. Although this advantage is based on the cost of manual annotation, the hierarchical representation ability are of incomparable advantages. This hierarchy provides natural information on scales, which inspires us to leverage this property together with traditional unsupervised methods.

2.3. Multi-scale edge detection

For the human visual system, real-world objects only have physical meaning within a specific scale range. For example, the horses on the grass in Fig. 1 are meaningful only on scales of a few meters to tens of meters. At the micron scale, one might talk more about bacteria, cells, etc., while at the kilometer scale, the overall landscape of the grassland should be more meaningful. Based on this fact, scale space theory was studied and established to address the multi-scale nature of image data [45,74–76]. One may find that the concept of multi-scale is inextricably linked to things like smoothness [77], time [78], information complexity [79], etc., but it always involves progressive levels of interpretation [80]. Multi-scale edge detection is built on the human interpretation, by increasing the attention on objects of interest and reducing the attention elsewhere. Early studies [1,28,81] have pointed out the need for multi-scale edge detection. Evidence reveals that edges detected at coarse scales are more robust against noise,
but can result in displacement error, and the opposite is true at fine scales [82,83].

Many multi-scale edge detection methods are inseparable from the concept of Gaussian scale space [45]. Although the undesirable displacement and vanishing phenomenon will appear when the scale becomes coarse, the Gaussian scale space is the only one that does not produce new maxima of the first derivative, and it can suppress erroneous responses due to noise, texture or unimportant objects [84]. Representative works usually detect edges explicitly at each scale and then combine edge maps. For example, edge focusing [85] adopts an coarse-to-fine strategy, starting from coarse-grained results, and then gradually using fine-grained results for iterative updates. The method achieves a balance between positioning accuracy and noise suppression. Although the effect is not ideal, the Canny detector actually provides a method of fine-to-coarse feature synthesis [14], that is, recursively comparing the coarse-grained results with the fine-grained synthesis results. Taking into account that the product is large only when all of the factors are large, the scale multiplication [86] computes the product of gradients at two different scales to enhance edge structures and dilutes noise. The Gaussian smoothing and edge tracking method [87] goes a step further and obtains a competitive edge detection scheme by simultaneously limiting the position offset and angle offset of edges between two adjacent scales.

Another strategy is first to collect the multi-scale representation for each pixel, and then discriminate whether it is an edge or not [88,89]. Similar ideas have been developed in deep edge detectors. Candidate edge pixels are first located by traditional methods, and then multi-scale deep features of these pixels are extracted to determine the final edges [8,9]. Perhaps a more intuitive voting method can be used, that is, counting the number of times a certain pixel is marked as an edge in multiple edge images [90]. Only pixels exceeding a certain threshold will be finally judged as edges. Surprisingly, a refined version of this idea yields results that are even comparable to manually annotated results [91]. Due to its simplicity and effectiveness, this idea is still used in deep edge detection [10,21].

In comparison, multi-scale strategies present different characteristics in deep edge detection. First, deep features are inherently multi-scale due to the design of the network structure, such as down-sampling layers or convolutional layers with large stride. For example, HED [10] uses this characteristic together with deep supervision [62] to provide hierarchical multi-scale representations for edge detection. Second, deep edge detectors (such as HED [10], CEDN [11], RCF [21]) directly use multiple spatially re-scaled images for training, where images are blurred and down-sampled to construct the so-called image pyramid. This kind of multi-scale training is time-consuming, but it adapts neural networks to images of different scales, which benefits edge detection. Third, the image pyramid is used again but for testing or predicting rather than training [21,23,64]. For an image, each re-scaled version yields an edge map which is then re-scaled to the original image size. The final edge map is the ensemble result by taking the average of edge maps. Besides, some studies conduct multi-scale operation on labels. For example, the RDS [63] constructs relaxed labels of different scales based on ground-truth labels and generated labels from traditional methods. This relaxed labels has a certain correspondence with the multi-scale representation of deep networks. Another method BDCN [64] decomposes ground-truth labels according to scales, based on which top-down and bottom-up paths are proposed to detect edges incrementally.

Different from traditional methods with deep mathematical derivation, multi-scale strategies in deep edge detection currently are often simpler. Despite its simplicity, these strategy does work accordingly. This may be attributed to the powerful representation learning capabilities of deep learning, which brings convenience in the design. However, this ability is still not discussed in unsupervised case, which is the main concentrate of this study. That is, we focuses on the impact of multi-scale unsupervised labels on deep edge detection, where “multi-scale” means richer information, “unsupervised” means lower labor costs.

3. Edge detection

At present, edge detection is usually described as an optimization problem with parameters. For deep edge detection, the deep supervision is an important component for performance improvement. In this section, we first describe the general formulation of edge detection under the framework of deep supervision, and then transition into our unsupervised point of view. Subsequently, we begin to introduce our proposed multi-scale pseudo labeling method, which includes stages of generating, fusing, smoothing, and training, etc.

3.1. General formulation

Deep supervision [62] is the most common strategy for deep edge detection [10,21,22,64] in the era of deep learning. It back-propagates error signals not only from the last output layer but also from intermediate hidden layers, which leverages features of different abstract levels.

Specifically, given X as an image, and \( \{Y^{(s)}\}_{s=1}^S \), as the corresponding ground-truth edges of S scales, the general form of edge detection with deep supervision is as follows,

\[
\min_{W} \frac{1}{B} \sum_{b=1}^{B} \sum_{s=1}^{S} \alpha_s \mathbb{E} [\hat{Y}^{(s)}(X; W) - Y^{(s)}(X)]^2,
\]

where \( \mathbb{E}[\cdot] \) denotes taking expectation over training data set \( D_{tr} \), \( \cdot \) is loss function, and \( \alpha_s \) hyper-parameters. The edge detector \( f \) parameterized with \( W \) predicts \( S \) edge maps \( \{Y^{(s)}\}_{s=1}^S \).

To solve the optimization problem above, a core point is to determine the scale-related ground truths \( \{Y^{(s)}\}_{s=1}^S \). A widely adopted approach is to use a single general supervision, \( Y^{(s)} = Y, s = 1, \ldots, S \), where \( Y \) is the annotated ground-truth label by human annotators [10, 21, 22]. This kind of simplification does avoid the difficulty of annotating in various scale spaces. However, this manual annotation is composed of edges of hybrid-scale information, i.e., a mixture of edge information at multiple scales [64]. This hybrid-scale annotation ignores the diversity nature of features of different layers in deep neural networks. Moreover, it takes lots of time and labor to annotate edges for human annotators, not to mention the fact that different people judge edges differently.

In order to avoid the drawback of manual annotation, we propose to use pseudo labels to replace ground truths in the Eq. (1). The pseudo labels has two properties. First, it should be generated in an unsupervised manner. Second, it should be disentangled at scale, i.e., associated with a specific scale parameter. Specifically, all ground truths \( \{Y^{(s)}\}_{s=1}^S \) are replaced by the following pseudo labels,

\[
Y^{(s)} = G(X; \sigma_s), s = 1, \ldots, S,
\]

where \( G(\cdot; \sigma) \) is a preassigned unsupervised edge detector with scale parameter \( \sigma \).

Although theoretically all unsupervised edge detectors are applicable, in practice it prefers to choose a simple, fast and accurate one to avoid excessive complexity. In this study, we employ the famous Canny detector [14] to generate pseudo edges, which are mainly determined by the following gradient magnitude,

\[
\|\nabla (\mathcal{N}_{\sigma_s} * X)\|,
\]

where \( \mathcal{N}_{\sigma_s} \) is the smoothing Gaussian filter with \( \sigma_s \) as standard derivation, i.e., scale parameter.

In practice, multiple different scale parameters are preferable rather than a single one, because multi-scale information can improve the fineness of edges by reducing the error of false detection, de-localization, etc. To acquire the final edge map, it is necessary to fuse edges from
different scales. By denoting a final pseudo edge map as $\hat{Y}^{(0)}$, the fusion step can be expressed as follows,

$$\hat{Y}^{(0)} = F(\hat{Y}^{(1)}, \ldots, \hat{Y}^{(S)})$$

where $F(\cdot)$ is a prescribed fusion function, such as taking an average.

Correspondingly, predicted edges $\{\hat{Y}^{(s)}\}_{s=1}^{S}$ should also be fused to include all edges of interest. Different from the case of generating pseudo labels, we add a fusion head $f_0$ with parameter $W_0$ to be learned to get the final predicted edges, which is more flexible than a fixed fusion function. This fusion step is expressed as follows,

$$\hat{Y}^{(0)} = f_0(\hat{Y}^{(1)}, \ldots, \hat{Y}^{(S)}; W_0)$$

where $f_0$ is usually designed as a $1 \times 1$ convolutional operation in practice [10,21,22,64].

To train the deep edge detector, the proposed Multi-Scale Pseudo Labeling (MSPL) method is expressed as a constrained optimization problem as follows,

$$\min_{W} E \left[ \sum_{s=0}^{S} s \cdot f(\hat{Y}^{(s)}, \hat{Y}^{(s)}) \right]$$

s.t. $\hat{Y}^{(s)} = f(X; W), s = 0, \ldots, S$.

$$\hat{Y}^{(s)} = G(X; \sigma_s), s = 1, \ldots, S$$

$$\hat{Y}^{(0)} = f(\hat{Y}^{(1)}, \ldots, \hat{Y}^{(S)})$$

Notably, the detector $f(\cdot; W)$ absorbs the fusion head $f_0(\cdot; W_0)$ into itself as stated above. Next, we will give the details of the pseudo-label generator $G(\cdot; \sigma)$ and the fusion function $F(\cdot)$.

### 3.2. Multi-scale pseudo labeling method

To train the deep edge detector, the proposed MSPL method generates pseudo labels with different scale parameters. To this end, the famous Canny detector [14] is used as the pseudo label generator $G(\cdot; \sigma)$. The Canny detector is a gradient-based unsupervised edge detector. Generally, the detector first smoothes an image by a Gaussian filter, and then determines the magnitude and direction of the edges by calculating the gradient. Finally, post-processing techniques such as non-maximum suppression are used to determine the position of edges pixels in the image. For the case of multiple scales, one could easily vary the scale parameter $\sigma_s$, i.e., the standard deviation of the Gaussian filter, to get scale-related edges. These edges are exactly the desired pseudo labels, which are used to guide the learning of those predictions from the side branches in the edge detector $f(\cdot; W)$.

The efficient combination of pseudo edges with different scales is another concern, which helps the learning of the fusion head in the edge detector and results in more complete edge maps with minimal redundancy. Classical combination strategies include fine-to-coarse feature synthesis [14], coarse-to-fine edge focusing [85], etc. However, deep edge detectors seem to prefer taking average [10,11,16,64,68], whether dealing with multiple manual labels or conducting multi-scale testing. This might be affected by the powerful ability of feature extraction, i.e., as long as the extracted feature is good enough, other steps can be simplified. Inspired by these works, the fusion function $F$ in Eq. (5) is designed to take average over scales, i.e., the hybrid-scale pseudo edge is a simple average by edges with different scales,

$$\hat{Y}^{(0)} = \frac{1}{S} \sum_{s=1}^{S} \hat{Y}^{(s)}$$

Compared to classical combination method, this operation treats all edges of different scales equally and does not need the order relation.

Although it may be not the case in theory, the operation works well in practice in spite of the simplicity.

Another issue about pseudo labels is false detection. Although Canny can obtain pseudo labels, its quality is still not satisfactory, which may result in false detection. To avoid this impact, the proposed MSPL employs a distance-aware label smoothing strategy, which is a simple but efficient technique inspired by label smoothing [49,50]. Instead of training with one-hot labels, the label smoothing method re-labels the ground truth $Y$ by the following mixing strategy,

$$Y' = (1 - \epsilon)Y + \epsilon \frac{1}{K}$$

where $\epsilon \in [0,1]$ is a smoothing hyper-parameter, and $K$ is the number of categories for a classification problem. Because there is no distance involved between different categories, the uniform perturbation is reasonable for classification. In edge detection, however, the distance dependence among edges indicates that it is not a proper way by simply replacing $1/K$ with the proportion of edge pixels in an edge map.

To this end, we propose the distance-aware label smoothing strategy. The intuition behind this approach is that pixels closer to ground-truth edges are more likely to be edges than those distant. Therefore, predictions closer to ground-truth edges should be encouraged (i.e., triggering small loss) while those far from ground truths should be suppressed (i.e., triggering big loss). Fig. 3 illustrates this strategy in one-dimensional case. Row (a) is the standard one-hot label, which encourages models to trust the label completely, Row (b) is the result of label smoothing, which can provide a certain level of regularization, Row (c) is the result of proposed distance-aware label smoothing, which decreases the probability of being an edge as going further from the ground truth.

Mathematically, this strategy takes the radial basis function [92] for smoothing,

$$M(p, p^*) = \begin{cases} e^{-\gamma \|p - p^*\|^2}, & p \in N(p^*) \\ 0, & \text{otherwise} \end{cases}$$

where $p$ and $p^*$ are the positions of the predicted edges and the ground-truth edges, respectively, $\gamma$ is a shape parameter, and the neighborhood $N(p^*)$ indicates the operating range. In experiment, it is implemented as an approximated discrete form of the neighborhood, trimmed as a $3 \times 3$ 2D window, which is shown in Fig. 4.

Specifically, let $Y$ and $F^{(L,S)}$ denote the original edge map and smoothed edge map, respectively. For a pixel $p$, we consider its neighbors as follows.
follows, three important steps, pseudo label generating, edge predicting, and (MSPL) method is shown in Alg. 1. Basically, the procedure consists of hyper-parameters take the recommended values as default \[21,22\].

Values smaller than \(\lambda\) are ignored by setting the loss value to 0. All these are hyper-parameters, and \(\alpha\) of edge and non-edge pixels, respectively, which means \(\lambda\) and \(\alpha\), \(\beta\) are the percentage of edge and non-edge pixels, where \(\alpha\) and \(\beta\) are the percentage of edge and non-edge pixels, respectively, which means \(\alpha + \beta = 1\). The hyper-parameter \(\alpha\) is to balance the impact of edge and non-edge pixels. For those pseudo labels with small values, it is hard to determine whether they are edges or not. Hence, all weak edges with pseudo labels with small values are ignored by setting the loss value to 0. All these hyper-parameters take the recommended values as default \[21,22\].

Then the proposed distance-aware label smoothing is expressed as follows,

\[
\mathcal{Y}_p^{(LS)} = \begin{cases} 
0, & \forall q \in N_8(p), \hat{Y}_q = 0 \\
\max\left\{ \frac{1}{\eta}, \mathcal{Y}_p^{(LS)} \right\}, & \exists q \in N_8(p), \hat{Y}_q = 1 \\
\frac{1}{\eta}, & \forall q \in N_8(p), \hat{Y}_q = 1 \\
\hat{Y}_p = 1 & \end{cases}
\]

That is to say, for a given pixel, its smoothed value is totally determined by the original values in the 3 \(\times\) 3 window.

3.3. Overall framework

Fig. 5 shows the framework of the proposed method, which is divided into two main parts. The first part is a deep edge detector similar to RCF \[21\], which contains 3 convolutional stages with a 2 \(\times\) 2 pooling layer inserted between. Each stage connects to a convolutional side branch, which would generate an edge map \(\mathcal{Y}^{(S)}\) as its side output. At last, all of side outputs are fused to generate the final edge map \(\hat{Y}\). More details can be found in the original paper \[21\]. The other part is used to generate pseudo labels \(\mathcal{Y}^{(O)}\), each of which is controlled by a scale-related hyper-parameter \(\sigma_s\). These labels together with the hybrid-scale label \(\mathcal{Y}^{(H)}\) would guide the learning procedure of the edge detector. The popular supervised strategy is shown at the bottom of the figure to illustrate the difference from our method.

We use the annotator-robust loss function \[21\] to train the deep edge detector. Specifically, let \(\hat{Y}^{(O)}\) and \(\hat{Y}^{(S)}\) denote the predicted value and pseudo value of a particular pixel located at position \(i\) at scale \(s\), respectively, and then the loss function has the following piece-wise form,

\[
l^{(s)}_{i} = \begin{cases} 
\lambda \log(1 - \hat{Y}^{(S)}), & \hat{Y}^{(O)} = 0, \\ 
0, & 0 < \hat{Y}^{(O)} < \xi, \\ 
\beta \log(\hat{Y}^{(S)}), & \text{otherwise}
\end{cases}
\]

where \(\lambda\) and \(\xi\) are hyper-parameters, and \(\lambda\) and \(\beta\) are the percentage of edge and non-edge pixels, where \(\alpha\) and \(\beta\) are the percentage of edge and non-edge pixels, respectively, which means \(\alpha + \beta = 1\). The hyper-parameter \(\lambda\) is to balance the impact of edge and non-edge pixels. For those pseudo labels with small values, it is hard to determine whether they are edges or not. Hence, all weak edges with pseudo values smaller than \(\xi\) are ignored by setting the loss value to 0. All these hyper-parameters take the recommended values as default \[21,22\].

The learning procedure of the proposed Multi-Scale Pseudo Labeling (MSPL) method is shown in Alg. 1. Basically, the procedure consists of three important steps, pseudo label generating, edge predicting, and network updating. After that, the learned network \(f(\cdot, W)\) would then be used to detect edges of new samples.

### 4. Experiments

We conduct experiments on three commonly used data sets to compare the performance of the proposed model with other models. In order to explore the impact of different settings of the model, we carry out the ablation study. It should be noted that our purpose is to provide a novel unsupervised deep edge detection method, rather than to strive for the state-of-the-art performance. Therefore, we take for reference without changes the training strategies of previous works \[21,22\] in our experiments. One might obtain better results than those shown in our experiments by comprehensively searching hyper-parameters.

#### 4.1. Experiment settings

To initialize the part of convolutional stages of deep edge detectors, the weights pre-trained on ImageNet data set \[26\] are used, while the part of side branches is initialized randomly by a zero-mean Gaussian distribution with standard deviation 0.01. The stochastic gradient descent with a global initial learning rate \(1e^{-6}\) batches 10 images for each iteration. We run 200 epochs for training on each data set, and use a validation set to determine the best model, which can usually be achieved within 10–40 epochs. All codes are implemented on PyTorch and run on an NVIDIA GeForce RTX 3080 Laptop GPU and 11-th-Gen Intel Core CPU.

To quantitatively evaluate models, we use the widely-adopted measure F-scores. The score is mathematically defined as the harmonic mean of the precision \(P\) and recall \(R\),

\[
F = \frac{2 \cdot P \cdot R}{P + R}
\]

with

\[
P = \frac{TP}{TP + FP}, \quad R = \frac{TP}{TP + FN}
\]

where \(TP\), \(FP\), and \(FN\) denote the number of true-positive, false-positive, and false-negative edge pixels, respectively. An edge pixel is positive when its predictive value exceeds a preset threshold, otherwise negative. There are usually two different choices to set the threshold:

- Optimal Data set Scale (ODS), which means all images in a data set share a single best threshold;
- Optimal Image Scale (OIS), which means each image in a data set takes an individual best threshold.

---

Algorithm 1 MSPL for Deep Edge Detection

**Require:** Data set \(D\), training epochs \(E\), batch size \(B\), learning rate \(\eta\), initialization parameters \(W_0\)

1. Initialize the network \(f(\cdot, W)\) with \(W = W_0\)
2. for \(e = 1, \ldots, E\) do
3. Batch \(D\) randomly, \(\{X^{(i)}|i=1\}^{D/B}\)
4. for \(i = 1, \ldots, |D|/B\) do
5. Shuffle samples in \(X^{(i)}\)
6. Initialize loss \(L \leftarrow 0\)
7. for \(b = 1, \ldots, B\) do
8. Select input \(X_b \in X^{(i)}\)
9. Generate pseudo labels, \(\{\hat{Y}^{(i)}|S\}\)
10. Predict edges, \(\{\hat{Y}^{(i)}|S\}\)
11. Compute loss, \(L \leftarrow L + \frac{1}{S} \sum_{s=0}^{S} l^{(s)}_{i}\)
12. end for
13. Average loss, \(L \leftarrow \frac{1}{B} L\)
14. Update parameters, \(W \leftarrow W - \eta \nabla_{W} L\)
15. end for
16. end for

---
It can be easily seen that the OIS F-score will not be less than ODS F-score.

Also, the precision–recall curve shows the trade-off between precision and recall as the decision threshold varies. We also plot the curve on each data set to comprehensively evaluate the performance of different methods.

To show the effectiveness of the proposed MSPL, we train two deep edge detectors (RCF [21], PiDiNet [22]) under the proposed MSPL strategy on three popular data sets (BSDS500 [6], NYUDv2 [17], Multicue [18]).

In order to investigate the impact of scale-related labels on model performance, three different types of labels are used for training in the experiments:

- Single-scale labels ("Sg") generated by Canny detector with a preset scale hyper-parameter. In other words, all output branches are supervised by the single-scale pseudo labels. On validation set, experiments show that models with \( \sigma = 3 \) work best among all candidates \( \{ \sigma_1 = 2, \sigma_2 = 2.5, \sigma_3 = 3, \sigma_4 = 3.5, \sigma_5 = 4 \} \).
- Hybrid-scale labels ("H") fused from Canny detector with all hyper-parameters above. This means that all output branches are supervised by the hybrid-scale pseudo labels. This scheme is the same as the supervised ones except for the difference in the annotation.
- Multi-scale labels ("M") with all hyper-parameters above together with the hybrid ones. In this case, each output branch has its own scale-related labels, and one can expect that the correspondence of the label scales and the hierarchical down-sampling structure of the neural networks could make gains.

4.2. Scales for pseudo labels

High-quality pseudo labels lead to high-performance models. As stated above, the quality of pseudo labels is affected by the scale parameter heavily. Therefore, the first task before training the deep edge detector is to determine the scale parameters, the standard deviation of Gaussian distribution, \( \sigma \). Besides, as our generator for pseudo labels, the Canny detector has two other main parameters that can be adjusted. They are thresholds to determine the type of predicted edges. One is the lower threshold \( T_l \) and edges with smaller predicted values than this threshold are ignored, while the other is the upper threshold \( T_u \) and edges with greater predicted values than this threshold are marked as strong. All of the other predicted edges are marked as weak.

We conduct experiments on the BSDS500 validation data set [6] by the brute-force search. The lower and upper thresholds are selected within their respective given sets, i.e., \( T_l \in \{0.15, 0.20, 0.25, 0.30, 0.35, 0.40, 0.45\} \) and \( T_u \in \{0.50, 0.55, 0.60, 0.65, 0.70, 0.75, 0.80, 0.85, 0.90\} \). For each pair of \( T_l \) and \( T_u \), the Canny detector predicts edges for given scale parameters, i.e., \( \{ \sigma_1 = 2, \sigma_2 = 2.5, \sigma_3 = 3, \sigma_4 = 3.5, \sigma_5 = 4 \} \). Finally, the trend of ODS F-score w.r.t scale parameter is plotted to determine the optimal scale parameter.

Fig. 6 shows the trend of ODS F-score w.r.t the scale parameter for each given lower threshold. Generally, all subplots show that the ODS F-score first rises up and then falls down as the scale increases. This indicates that false positives caused by noise or texture are suppressed at first as the scale increases, but then edges of interest are also suppressed as the scale continues to increase and only the most salient edges are preserved. This phenomenon becomes more severe as the upper threshold gets greater. Moreover, in any subplot, the ODS F-score increases with the increase of the upper threshold when the scale parameter is less than 3.0, and the situation is roughly the opposite when the scale parameter is greater than 3.0. In other words, the ODS F-score always achieves its maximum at around 3.0. Fig. 7 is the same as Fig. 6 but from another perspective, i.e., given upper threshold, where the ODS F-score exhibits similar patterns and the lower threshold also acts like the upper threshold.

Furthermore, we compute the maximum of the ODS F-score over the scale parameter for each pair of \( T_l \) and \( T_u \), and shows the result in Fig. 8. Given a pair of \( (T_l, T_u) \), the colored point represents the corresponding maximum of the ODS F-score, where a larger score means a bigger point and a larger scale means a darker-colored point. It can be seen that the color of most larger points is neither too dark nor too light. These points correspond exactly to the scale parameter equal to 3.0.

Considering that deep edge detectors used in experiments have five side branches, in subsequent experiments we will take 3.0 as the center and 0.5 as the step size to expand the scale parameter range to both sides. That is to say, all experiments below use these scale parameters, i.e., \( \{ \sigma_1 = 2, \sigma_2 = 2.5, \sigma_3 = 3, \sigma_4 = 3.5, \sigma_5 = 4 \} \).

4.3. Results on the BSDS500 data set

The BSDS500 data set [6] is one of the most popular data sets for edge detection, samples of which are shown in Fig. 9. It is composed...
Fig. 6. ODS F-score w.r.t scale parameter (std. of Gaussian). Each subplot is corresponding to a low threshold $T_l$, and each curve in a subplot is corresponding to a high threshold $T_u$. Best viewed in electronic version.

Fig. 7. ODS F-score w.r.t scale parameter (std. of Gaussian). Each subplot is corresponding to a high threshold $T_u$, and each curve in a subplot is corresponding to a lower threshold $T_l$. Best viewed in electronic version.

Fig. 8. The maximum of ODS F-score for each pair of $T_l$ and $T_u$. The size of each dot corresponds to the size of the ODS F-score, and the depth of the color corresponds to the size of the scale parameter (the height on the Z axis) when reaching the maximum value.

Fig. 8. The maximum of ODS F-score for each pair of $T_l$ and $T_u$. The size of each dot corresponds to the size of the ODS F-score, and the depth of the color corresponds to the size of the scale parameter (the height on the Z axis) when reaching the maximum value.

are all size of $481 \times 321$ or $321 \times 481$. Each image is annotated by various human subjects, whose average agreement is 0.803 at F-score. The multiple annotated labels are then averaged as the final ground-truth labels. In this study, it should be noted that these labels are only used for evaluation rather than network training or hyperparameter searching. The common techniques for data augmentation based on previous works [10, 21] are also used in our experiments, such as scaling, rotating, and flipping. Samples from the PASCAL VOC Context data set [19] are also added into the training set.

Evaluation results are shown in Table 1 and Fig. 10. For comparison, the unsupervised methods (Canny [14], PMI [32]), weakly supervised method (WSBD [40]), and supervised methods (SE [15], HED [10], RCF [21], BDCN [64], PiDiNet [22]) are taken into consideration. As Fig. 10 shows, undoubtedly, supervised methods perform the best, and achieve competitive and even better results than human beings (the green dot), which is 0.803 at F-score. Notably, all unsupervised methods perform worse than the supervised ones. This suggests that label quality remains critical to performance, but the unsupervised performance gradually gets closer to the supervised ones by the introduction of deep features and other strategies.

The statistic comparison is shown in Table 1, where the average results and their fluctuations based on three runs are reported. When compared to the primitive Canny detector, the proposed MSPL of all
specifications performs better, and even the worst case of Row “Sg” has some slight improvement which is brought by deep features. When taking results of Row “Sg” as baseline, results of Row “H” and Row “M” are better due to information of various scales. Moreover, results become better with the introduction of our smoothing strategy. The best result achieves 0.72 at ODS F-score and 0.74 at OIS F-score. It indicates that the proposed distance-aware label smoothing is helpful to improve performance.

However, whatever the RCF-based or PiDiNet-based detectors, the MSPL strategy does have a noticeable gap when compared with its supervised counterpart. One possible reason is that the labels generated by the pseudo-labeling approach cannot accurately distinguish between non-object edges and object edges. Besides, the MSPL performs slightly worse than the unsupervised PMI method, which uses both features of color and variance and performs test in three scales. By getting rid of these strategies, the PMI could only get 0.70 ODS F-score and 0.74 OIS F-score. To be fair, we report the results of the simpler PMI in the following experiments.

4.4. Results on the NYUDv2 data set

The NYUDv2 data set [17] consists of 1,449 RGB-D images of 26 indoor-scene classes. Previous works usually follow a common split rule [93] for edge detection, i.e., 381 samples for training, 414 samples for validating, and 654 samples for testing. Images in this data set are all size of $560 \times 425$, and each image has only one ground-truth annotation. This data set is also augmented by scaling, rotating, and flipping. Notably, the depth channel in this data set is encoded by HHA [94] (Horizontal disparity, Height above ground, and Angle with gravity). Then, the predicted edge maps are first generated separately from the RGB channels and HHA channels, and the final edge map is then by averaging those two edges. Besides, the localization tolerance between predicted edges and ground truths increases from 0.0075 to 0.011 when evaluating [10,21].

The evaluation results on NYUDv2 data set are shown in Table 2 and Fig. 11. The classical methods (Canny [14] and SE [15]) and deep-learning-based methods (HED [10], RCF [21], BDCN [64], and PiDiNet [22]) are presented in the figure and table for comparison. Fig. 11 shows that the supervised deep edge detectors still perform best. It should be noted that because the indoor-scene images usually contains lots of objects and have cluttered backgrounds, results on this data set are generally worse than those on the BSDS500 data set.

Table 2 shows that all results with various scales, e.g., Row “H” and Row “M”, are not only better than those of the Canny detector but also better than those of the single-scale case. Except the single-scale case, our method gets competitive or even better result than PMI which is the best unsupervised edge detector at present. This once again indicates the importance of multi-scale information. Similar conclusions are also reached in the case of HHA, but with slightly worse performance. An explanation is that the HHA inputs are less descriptive for edges of

---

**Table 1**

<table>
<thead>
<tr>
<th>Method</th>
<th>Type</th>
<th>ODS (↑)</th>
<th>OIS (↑)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Canny [14]</td>
<td>US</td>
<td>.61</td>
<td>.65</td>
</tr>
<tr>
<td>PMI [32]</td>
<td>US</td>
<td>.72</td>
<td>.75</td>
</tr>
<tr>
<td>WSBD [40]</td>
<td>WS</td>
<td>.73</td>
<td>.76</td>
</tr>
<tr>
<td>SE [15]</td>
<td>S</td>
<td>.74</td>
<td>.76</td>
</tr>
<tr>
<td>HED [10]</td>
<td>S</td>
<td>.78</td>
<td>.80</td>
</tr>
<tr>
<td>RCF [21]</td>
<td>S</td>
<td>.81</td>
<td>.83</td>
</tr>
<tr>
<td>BDCN [64]</td>
<td>S</td>
<td>.83</td>
<td>.84</td>
</tr>
<tr>
<td>PiDiNet [22]</td>
<td>S</td>
<td>.81</td>
<td>.82</td>
</tr>
</tbody>
</table>

---

**Fig. 10.** Precision-Recall curves on the BSDS500 data set.

---

**Fig. 9.** Image-label pairs from different data sets. Row (a): Samples from BSDS500 data set. Row (b): Samples from NYUDv2 data set in RGB format. Row (c): The same as Row (b) but in HHA format. Row (d): Boundary annotation from Multicue data set. Row (e): Edge annotation from Multicue data set.
objects than the RGB inputs, as shown in Row (b) and Row (c) in Fig. 9, which brings difficulties in distinguishing edge and non-edge pixels for detectors.

4.5. Results on the Multicue data set

Recently, the Multicue data set [18] has been proposed both for object-level boundary detection and edge-level boundary detection. This data set contains 100 short (10-frame) binocular video clips of natural scenes. For each video, only the last left-view frame is annotated with two kinds of labels,

- Boundary labels, which care more about the contour of salient objects.

- Edge labels, which prefer discontinuities in brightness or color.

This results in 100 image-boundary or image-edges pairs, which would be randomly split into a training set with 80 images and a test set with 20 images. Images in this data set are relatively large, with a spatial size of 1280 × 720. Please refer to Fig. 9 for demonstration.

Evaluation results are in Table 3 and Fig. 12. As Table 3 shows, whether on the boundary detection or the edge detection, similar conclusions to the above still holds. Since other methods do not disclose the detailed results, only the result of our MSPL method is shown in Fig. 12. It is obvious that the figure is divided into upper (edge results) and lower (boundary results) parts, with an obvious gap in the middle. This gap is mainly from the level of label details. This phenomenon indicates that the proposed MSPL is still affected by details and is susceptible to cluttered edges, and in other words, it is unable to accurately grasp the concept of object in the sense of semantics.

4.6. Qualitative results

The comparison of the outputs under different configurations is shown in Fig. 13. As indicated by the red dotted boxes, the single-scale case “Sg” suffers from texture edges, but with the introduction of information of various scales or smoothing strategies, cluttered edges are gradually eliminated. However, the elimination is not as good as expected, especially about the changes in light and shadow on the iceberg at the front of the bear, which requires further study.

In order to get an visual intuition, Fig. 14 shows the outputs from different branches, with the related pseudo labels below. From left to right, it shows that trivial details slowly disappear while boundaries of objects become more and more clear, which just corresponds to the change in scale from fine to coarse. This phenomenon shows that the proposed MSPL has the ability to perceive the change of scale. At the fourth and fifth columns, our method paints edges with thicker strokes which mostly are the boundaries of objects. The fused edges at the sixth column suppress non-object edges better than those pseudo labels.

Although the proposed MSPL can detect edges of interest, it has some shortcomings. As shown in Fig. 15, for scenes with clutter but obvious local contrast, such as books at the second row or the trees at the fifth row, our method does not perform well. The effect of local contrast is more clear at the fourth row, where the bird is divided into two parts due to the influence of discontiuity. In other words, the impact of local contrast is greater than deep features or smoothing strategy. It could be expected that an regularization about “objectness” might be avoid this kind of error, which would be our further research.

---

Table 2

Results on the NYUDv2 data set. The mean ± std. are based on three runs.

<table>
<thead>
<tr>
<th>Method</th>
<th>Type</th>
<th>ODS (1)</th>
<th>OIS (1)</th>
<th>ODS (1)</th>
<th>OIS (1)</th>
<th>ODS (1)</th>
<th>OIS (1)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Canny [14]</td>
<td>US</td>
<td>.59</td>
<td>.60</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>PMI [32]</td>
<td>US</td>
<td>.63</td>
<td>.65</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>SE [15]</td>
<td>S</td>
<td>.69</td>
<td>.70</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>RGB</td>
<td>HHA</td>
<td>RGB + HHA</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>HED [10]</td>
<td>S</td>
<td>.72</td>
<td>.73</td>
<td>.68</td>
<td>.69</td>
<td>.74</td>
<td>.76</td>
</tr>
<tr>
<td>RCF [21]</td>
<td>S</td>
<td>.74</td>
<td>.75</td>
<td>.70</td>
<td>.71</td>
<td>.76</td>
<td>.78</td>
</tr>
<tr>
<td>BDCN [64]</td>
<td>S</td>
<td>.75</td>
<td>.76</td>
<td>.71</td>
<td>.72</td>
<td>.76</td>
<td>.78</td>
</tr>
<tr>
<td>PiDiNet [22]</td>
<td>S</td>
<td>.73</td>
<td>.75</td>
<td>.71</td>
<td>.73</td>
<td>.75</td>
<td>.77</td>
</tr>
</tbody>
</table>

* Abbreviations are the same as Table 1.

Table 3

Results on the Multicue data set. The mean ± std. are based on three runs.

<table>
<thead>
<tr>
<th>Method</th>
<th>Type</th>
<th>Boundary</th>
<th>Edge</th>
</tr>
</thead>
<tbody>
<tr>
<td>ODS (1)</td>
<td>OIS (1)</td>
<td>ODS (1)</td>
<td>OIS (1)</td>
</tr>
<tr>
<td>Canny [14]</td>
<td>US</td>
<td>.61</td>
<td>.65</td>
</tr>
<tr>
<td>PMI [32]</td>
<td>US</td>
<td>.69</td>
<td>.71</td>
</tr>
<tr>
<td>HED [10]</td>
<td>S</td>
<td>.81</td>
<td>.82</td>
</tr>
<tr>
<td>RCF [21]</td>
<td>S</td>
<td>.83</td>
<td>.84</td>
</tr>
<tr>
<td>BDCN [64]</td>
<td>S</td>
<td>.84</td>
<td>.85</td>
</tr>
<tr>
<td>PiDiNet [22]</td>
<td>S</td>
<td>.81</td>
<td>.83</td>
</tr>
</tbody>
</table>

* Abbreviations mean the same as Table 1.

---

Fig. 11. Precision-Recall curves on the NYUDv2 data set.
4.7. Ablation study over data sets

In order to verify the effectiveness of different configurations, it is necessary to analyze the improvements they each bring. To this end, we take the Canny detector with $\sigma = 3.0$ as the baseline and calculate the successive improvements. To make it clear, we describe these configurations as follows.

- The deep edge detector trained by the single-scale Canny labels. By examining the difference between this detector and the baseline, we can get the benefits brought by deep features, which is hence denoted as Deep Neural Networks (DNN).
- The deep edge detector trained by the hybrid-scale Canny labels. The difference between this detector and DNN shows the benefits brought by the Hybrid-Scale (HS) strategy.
- The deep edge detector trained by the multi-scale Canny labels. The difference between this detector and HS shows the benefits brought by the Multi-Scale (MS) strategy.
- The deep edge detector trained by the distance-aware smoothing labels. The difference between this detector and that without smoothing strategy shows the benefits brought by the Label Smoothing (LS) strategy.

To make the results more compact, the performance is averaged over all data sets. Formally, let $F^{(D)}_{\text{Canny}}$, $F^{(D)}_{\text{DNN}}$, $F^{(D)}_{\text{HS}}$, and $F^{(D)}_{\text{MS}}$ denote F-scores on a data set $D$, respectively, and then the above differences are expressed as follows,

$$
\Delta F_{\text{DNN}} = \operatorname{Avg}(F^{(D)}_{\text{DNN}} - F^{(D)}_{\text{Canny}})
$$

$$
\Delta F_{\text{HS}} = \operatorname{Avg}(F^{(D)}_{\text{HS}} - F^{(D)}_{\text{DNN}})
$$

$$
\Delta F_{\text{MS}} = \operatorname{Avg}(F^{(D)}_{\text{MS}} - F^{(D)}_{\text{HS}})
$$

$$
\Delta F_{\text{LS}} = \operatorname{Avg}(F^{(D)}_{\text{LS}} - F^{(D)}_{\text{¬LS}})
$$

where Avg means taking average over three data sets $D \in \{\text{BSDS500, NYUDv2, Multicue}\}$, and ¬LS means the counterparts without label smoothing.

Results are shown in Fig. 16, which tells that all configurations contribute to performance improvement. One-third of the improvement comes from deep features (DNN), and the other two-thirds benefit from information of various scales and the smoothing strategy. The absolute improvement is about 0.078 (relatively 11.9%) from Canny detector, and it still has an absolute improvement of 0.053 (relatively 7.79%) even taking DNN as the baseline. This suggests that the MSPL method does reduce the gap between unsupervised deep edge detectors and their supervised counterparts. Besides, it seems that MS configuration make a lower contribution, especially with the PiDiNet detector. However, it should be noted that this improvement is based on HS, which has already provided information at various scales. The improvement of MS means the additional gain by scale decomposition.

5. Conclusions

In this study, we propose a novel framework, referred to as multi-scale pseudo labeling method, for unsupervised deep edge detection. This method generates pseudo edge maps with different scales, each of which guides the learning process of one side branch in deep edge detectors. Experiments on three popular data sets show that the method reduce the gap with the supervised counterparts, and can be easily embedded into existed deep edge detectors. Visual results also show that the proposed method has the ability to suppress non-object edges. It is a promising research for reducing the labor of manual annotation. However, the proposed method still lacks in understanding the concept of objects, which makes it difficult to extract “semantic edges”. Hence, it is meaningful to study the introduction of additional constraints about “objectness”. Moreover, this kind of method can also be used on related research topics, such as object proposals and salient object detection. The source code will be available on our GitHub website https://github.com/wasaCheney.
Fig. 14. The outputs from different side branches and the corresponding pseudo labels from Canny detector. From left to right are outputs from shallow to deep layers, followed by the fused ones.

CRediT authorship contribution statement

Changsheng Zhou: Conceptualization, Formal analysis, Funding acquisition, Writing – original draft, Writing – review & editing. Chao Yuan: Writing – original draft, Writing – review & editing. Hongxin Wang: Formal analysis, Methodology. Lei Li: Conceptualization, Writing – original draft. Stefan Oehmcke: Conceptualization, Formal analysis, Writing – original draft. Junmin Liu: Conceptualization, Formal analysis, Writing – original draft. Jigen Peng: Funding acquisition, Project administration, Resources, Supervision.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Data availability

Data will be made available on request.

Acknowledgments

This work was supported by National Natural Science Foundation of China (Grant Nos. 12201145, 12031003, 62276208), China Postdoctoral Science Foundation, China (Grant No. 2022M710862), Natural Science Foundation of Guangdong Province, China (Grant No. 2022A1515110336), International Development Plan for Young Scientists of Guangdong Province, China (Grant No. 06400802), and Postdoctoral Startup Foundation of Guangdong Province, China (Grant No. 62104330). This work was conducted when Changsheng Zhou was an visiting scholar at UCPH with Christian Igel. The authors would like to thank Ankit Kariryaa, Mathias Perslev, Yijie Zhang, Lin Zhang and Haiyang Li for their insightful comments.
References

C. Zhou et al.

Knowledge-Based Systems 280 (2023) 111057


