DiffusionEdge: Diffusion Probabilistic Model for Crisp Edge Detection

Yunfan Ye[1,2*], **Kai Xu**[2*], **Yuhang Huang**[2†], **Renjiao Yi**[2], **Zhiping Cai**[2]

[1]Hunan University
[2]National University of Defense Technology

Abstract

Limited by the encoder-decoder architecture, learning-based edge detectors usually have difficulty predicting edge maps that satisfy both correctness and crispness. With the recent success of the diffusion probabilistic model (DPM), we found it is especially suitable for accurate and crisp edge detection since the denoising process is directly applied to the original image size. Therefore, we propose the first diffusion model for the task of general edge detection, which we call DiffusionEdge. To avoid expensive computational resources while retaining the final performance, we apply DPM in the latent space and enable the classic cross-entropy loss which is uncertainty-aware in pixel level to directly optimize the parameters in latent space in a distillation manner. We also adopt a decoupled architecture to speed up the denoising process and propose a corresponding adaptive Fourier filter to adjust the latent features of specific frequencies. With all the technical designs, DiffusionEdge can be stably trained with limited resources, predicting crisp and accurate edge maps with much fewer augmentation strategies. Extensive experiments on four edge detection benchmarks demonstrate the superiority of DiffusionEdge both in correctness and crispness. On the NYUDv2 dataset, compared to the second best, we increase the ODS, OIS (without post-processing) and AC by 30.2%, 28.1% and 65.1%, respectively. Code: https://github.com/GuHuangAI/DiffusionEdge.

Introduction

Edge detection is a longstanding vision task for detecting object boundaries and visually salient edges from images. As a fundamental problem, it benefits various downstream tasks ranging from 2D perception (Zitnick and Dollár 2014; Revaud et al. 2015; Cheng et al. 2020), generation (Nazeri et al. 2019; Xiong et al. 2019), and 3D curve reconstruction (Ye et al. 2023b).

There are three main challenges in general edge detection, *correctness* (identifying edge and non-edge pixels on noisy scenes), *crispness* (the width of edge lines, precisely localizing edges without confusing pixels) and *efficiency* (the inference speed). Traditional methods extract edges based on local features such as gradient (Kittler 1983; Canny 1986),

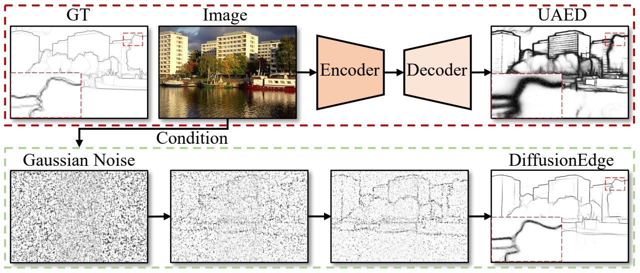

Figure 1: CNN-based methods, even the most recent and state-of-the-art one (UAED (Zhou et al. 2023)), generally have an encoder-decoder architecture with limitations of thick edges and more noise. We propose the diffusion-based edge detector which is superior in both correctness and crispness without any post-processing.

which can be crisp but not correct enough. Deep learning-based methods (Xie and Tu 2015; Liu et al. 2017; He et al. 2019; Poma, Riba, and Sappa 2020; Pu et al. 2022; Zhou et al. 2023) achieve significant progress by capturing local and global features with multi-layers, which is correct but not crisp enough. Recently, efforts have also been made to design lightweight architectures (Su et al. 2021) for efficiency, or loss functions (Deng et al. 2018; Huan et al. 2021) and refinement strategies (Ye et al. 2023a) for crisp edge detection. However, none of each single edge detector can directly predict edge maps that simultaneously satisfy both correctness and crispness, without a post-processing of morphological non-maximal suppression (NMS) scheme. We ask this question: Can we learn an edge detector that can directly generate both accurate and crisp edge maps without heavily relying on post-processing?

In this work, we try to answer the question through learning a diffusion model for edge detection. As demonstrated in Figure 1, DPMs have two main differences compared with methods based on the Convolutional Neural Network (CNN): (a) CNN-based models generally learn and infer the targets in a single round, while DPMs are trained to predict a denoised variant of the noisy input by several steps, which makes it easier for DPMs to learn the target distribution; (b) CNN-based edge detectors generally extract features from multi-layers and therefore are limited by the existence of downsampling (for high-level global features) and upsampling (for pixel-wise alignment) operators, which leads to

[*]These authors contributed equally.
[†]Corresponding author.

thick edge predictions in nature (Huan et al. 2021), while DPMs directly perform the denoising process on the level of original image size.

With the two characteristics, we found diffusion model is especially suitable for accurate and crisp edge detection. However, there are still several challenges for DiffusionEdge to be accurate and crisp enough with limited computational resources and inference time. We apply a decoupled diffusion architecture similar to DDM (Huang et al. 2023) to speed up the inference, and propose an adaptive Fourier filter before decoupling, which enables the network weights to adjust the components of the specific frequencies adaptively. Following (Rombach et al. 2022), we also train the diffusion model in latent space to reduce computations. However, most CNN-based edge detectors are trained by the annotator-robust cross entropy loss (Liu et al. 2017) in image pixel level, which provides uncertainty information when training edge datasets labeled by several annotators like BSDS (Arbelaez et al. 2010). To keep that free and valuable uncertainty prior, we apply an uncertainty distillation strategy by directly passing the optimized gradients from pixel level to latent space level based on the chain rule.

With the above efforts, extensive experiments on four edge detection benchmarks show that DiffusionEdge can directly generate accurate and crisp edge maps without any post-processing, and achieve superior qualitative and quantitative performance with much less augmentation strategies. On the NYUDv2 dataset (Silberman et al. 2012), compared to the second best, we increase the ODS, OIS (without post-processing) and AC by 30.2%, 28.1% and 65.1%, respectively. Our contributions include:

- A novel diffusion-based edge detector, named DiffusionEdge, which can predict accurate and crisp edge maps without post-processing. To our best knowledge, it is the first diffusion model toward edge detection.
- Several technical designs to ensure learning a satisfactory diffusion model in latent space, while keeping the uncertainty prior and adaptively filtering latent features in Fourier space.
- Superior performance on four edge detection benchmarks for both correctness and crispness.

Related Work

Edge detection. Edge detection aims to extract object boundaries and visually salient edges from natural images. Traditional edge detectors as such Sobel (Kittler 1983) and Canny (Canny 1986) generate edges through local gradients, which often suffer from noisy pixels without global content. CNN-based methods start integrating features from multi-layers and improve the correctness of edge pixels by a large margin. HED (Xie and Tu 2015) proposed the first end-to-end edge detection architecture, and RCF (Liu et al. 2017) improved it by integrating more hierarchical features. BDCN (He et al. 2019) trains the edge detector with layer-specific supervisions in a bi-directional cascade architecture. PiDiNet (Su et al. 2021) introduced pixel difference convolution in the designed lightweight architectures for efficient edge detection. UAED (Zhou et al. 2023) measures the degree of ambiguity among different annotations from multiple annotations to focus more on hard samples. Also, EDTER (Pu et al. 2022) proposed to detect global context and local cues by vision transformers in two stages.

Those learning-based methods can achieve remarkable progress in correctness via integrating features from multi-layers and uncertainty information. However, the generated edge maps are too thick for downstream tasks and heavily rely on the post-processing. Although efforts for crisp edge detection have been made on loss functions (Deng et al. 2018; Huan et al. 2021) and the label refinement strategy (Ye et al. 2023a), we argue that the community still needs an edge detector that can directly satisfy both correctness and crispness without any post-processing.

Diffusion probabilistic model. Diffusion models (Sohl-Dickstein et al. 2015; Ho, Jain, and Abbeel 2020; Huang et al. 2023) are a class of generative models based on a Markov chain, which gradually recover the data sample via learning the denoising process. Diffusion models demonstrate remarkable performance in fields of computer vision (Nichol et al. 2021; Avrahami, Lischinski, and Fried 2022; Gu et al. 2022), nature language processing (Austin et al. 2021) and audio generation (Popov et al. 2021). Despite those great achievements in generative tasks, diffusion models also have great potential for perception tasks, such as image segmentation (Brempong et al. 2022; Wu et al. 2023) and object detection (Chen et al. 2022).

Inspired by the above pioneers (Xie and Tu 2015; Chen et al. 2022; Huang et al. 2023), our method has two main differences to directly generate accurate and crisp edge maps with acceptable inference time. First, we design to impose a learnable Fourier convolution module in the decoupled diffusion architecture, to adaptively filter latent features in Fourier space depending on the target distribution. Second, to keep the pixel-level uncertainty prior from edge datasets with multiple annotators, we distillate the gradients directly to latent space for improved results and stabilized training. The proposed DiffusionEdge, to the best of our knowledge, is the first usage of diffusion models for generic edge detection, and is superior in both correctness and crispness.

Method

The overall framework of the proposed DiffusionEdge is illustrated in Figure 2. Inspired by previous works (Rombach et al. 2022; Wu et al. 2023; Huang et al. 2023), we train the diffusion model with decoupled structure in latent space and take the input image as the extra condition. Based on the diffusion process introduced in preliminaries, we introduce the adaptive FFT-filter for frequency parsing. To keep pixel-level uncertainty from multiple annotators and reduce computational resources, we proposed to directly optimize the latent space with cross-entropy loss in a distillation manner.

Preliminaries

Current studies (Chen et al. 2022; Wu et al. 2023) have shown the great potential of DPMs in perception tasks, however, it suffers from prolonged sampling time. Inspired by (Huang et al. 2023), we adopt a decoupled diffusion model

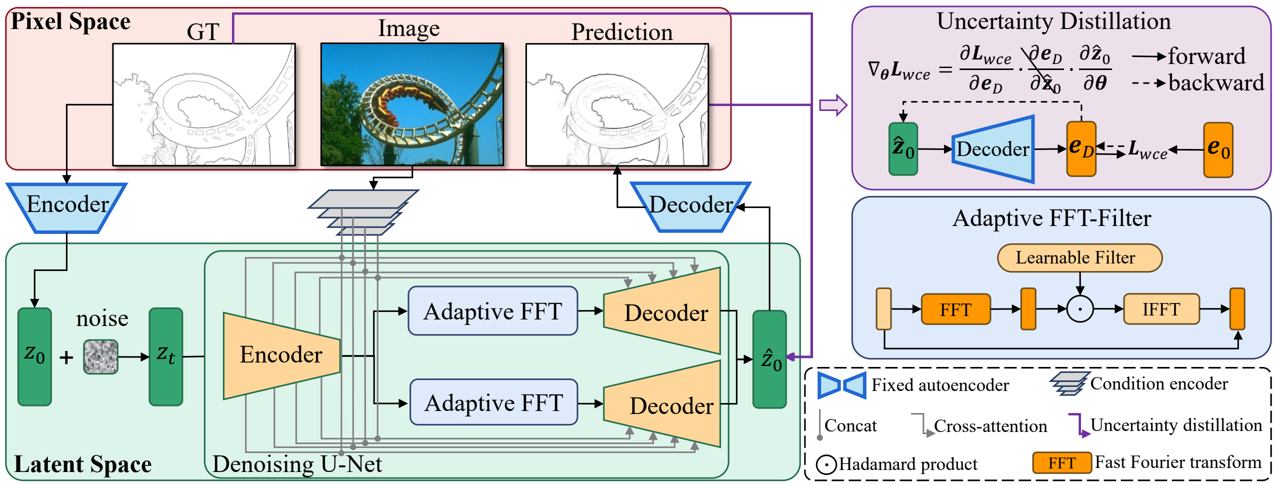

Figure 2: The overall framework of the proposed DiffusionEdge.

(DDM) to speed up the sampling process. The decoupled forward diffusion process is governed by the combination of the explicit transition probability and the standard Wiener process:

$$q(\mathbf{e}_t|\mathbf{e}_0) = \mathcal{N}(\mathbf{e}_0 + \int_0^t \mathbf{f}_t \mathrm{d}t, t\mathbf{I}), \tag{1}$$

where $\mathbf{e}_0$ and $\mathbf{e}_t$ are the initial and noisy edges, and $\mathbf{f}_t$ is the explicit transition function representing the opposite direction of the gradient of the edge. Following (Huang et al. 2023), we use the constant function as default $\mathbf{f}_t$. The corresponding reversed process is represented by:

$$\begin{aligned} q(\mathbf{e}_{t-\Delta t}|\mathbf{e}_t, \mathbf{e}_0) = \mathcal{N}(\mathbf{e}_t + \int_t^{t-\Delta t} \mathbf{f}_t \mathrm{d}t \\ - \frac{\Delta t}{\sqrt{t}} \boldsymbol{n}, \frac{\Delta t(t-\Delta t)}{t} \mathbf{I}), \end{aligned} \tag{2}$$

where $\boldsymbol{n} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$. To train the decoupled diffusion model, we need to supervise the data and noise components simultaneously, therefore, the training objective is parameterized by:

$$\min_{\boldsymbol{\theta}} \mathbb{E}_{q(\mathbf{e}_0)} \mathbb{E}_{q(\boldsymbol{n})} [\|\mathbf{f}_{\boldsymbol{\theta}} - \mathbf{f}\|^2 + \|\boldsymbol{n}_{\boldsymbol{\theta}} - \boldsymbol{n}\|^2], \tag{3}$$

where $\boldsymbol{\theta}$ is the parameter of the denoising network. Since diffusion models take up too much computational cost in original image space, we follow (Rombach et al. 2022) to transfer the training process into latent space with $4\times$ downsampling spatial size.

As shown in Fig. 2, we first train an autoencoder that consists of an encoder for compressing the edge ground truth to latent code and a decoder for recovering it from the latent code, respectively. Then, in the stage of training denoising U-Net, we fix the weights of the autoencoder and train the denoising process in latent space. The process can be represented as:

$$\begin{aligned} \mathbf{f}_{\boldsymbol{\theta}}, \boldsymbol{n}_{\boldsymbol{\theta}} &= \mathbf{Net}_{\boldsymbol{\theta}}(\mathbf{z}_t, t), \\ \mathbf{z}_t &= \mathbf{z}_0 + \int_0^t \mathbf{f}_t \mathrm{d}t + \sqrt{t}\boldsymbol{n}, \end{aligned} \tag{4}$$

where $\mathbf{Net}_{\boldsymbol{\theta}}$ denotes the denoising U-Net, $\mathbf{z}_0 = \mathcal{E}(\mathbf{e}_0)$ is the latent code compressed by the encoder of autoencoder, t is the time step.

We also incorporate several technical designs for edge detection, making it available to obtain accurate and crisp predictions within acceptable inference time.

Adaptive FFT-filter

The denoising U-Net aims to decouple the noisy input $\mathbf{e}_t$ into the denoised data $\mathbf{e}_0$ and the noise component $\boldsymbol{n}$. The vanilla convolution layers are adopted as the decoupling operator, to separate the denoised edge maps and noise component from the noisy variable. However, the convolution operators focus more on feature aggregation, and no not adjust the components of specific frequencies. Therefore, we introduce a decoupling operator that can filter out different components adaptively. As shown in the left-top of Figure 2, we integrate the adaptive Fast Fourier Transform filter (Adaptive FFT-filter) into the denoising Unet to filter out edge maps and noise components in the frequency domain. Specifically, given the encoder feature $\mathbf{F} \in \mathbb{R}^{H\times W\times C}$, we first perform 2D FFT along the spatial dimensions, and represent the transformed feature as $\mathbf{F_c} = \mathscr{F}[\mathbf{F}], \mathbf{F_c} \in \mathbb{C}^{H\times W\times C}$. Then, to learn an adaptive spectrum filter, we construct a learnable weight map $\mathbf{W} \in \mathbb{C}^{H\times W\times C}$ and multiply $\mathbf{W}$ to $\mathbf{F_c}$. The spectrum filter benefits the training since it can globally adjust the specific frequencies and the learned weights are adaptive for different frequencies of target distributions. With the useless components filtered out adaptively, we project the feature from the frequency domain back to the spatial domain by Inverse Fast Fourier Transform (IFFT). Finally, we adopt a residual connection from $\mathbf{F}$ to avoid filtering useful information out. We can describe the above process by the following equation:

$$\mathbf{F}_o = \mathbf{F} + \mathscr{F}^{-1}[\mathbf{W} \circ \mathbf{F_c}], \tag{5}$$

where $\mathbf{F}_o$ is the output feature, $\circ$ represents the hadamard product.

Uncertainty Distillation

Since the numbers of edge and non-edge pixels are highly imbalanced (the majority of pixels are non-edges), HED (Xie and Tu 2015) propose to apply weighted binary cross-entropy (WCE) loss for optimization, which is further improved by RCF (Liu et al. 2017) with uncertainty prior from multiple annotators. With E_i to be the ground truth edge probability of ith pixel, for the ith pixel in the jth edge map with value p_i^j, the uncertainty-aware WCE loss is calculated as:

$$l_i^j = \begin{cases} \alpha \cdot \log\left(1 - p_i^j\right), & if\ E_i = 0, \\ 0, & if\ 0 < E_i < \eta, \\ \beta \cdot \log E_i^j, & otherwise, \end{cases} \tag{6}$$

in which

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

where η is the threshold to decide uncertain edge pixels in ground truths, and such ambiguous samples will be ignored during subsequent optimization. E^+ and E^- denote the number of edge and non-edge pixels in the ground truth edge maps. λ is the weight for balancing E^+ and E^-. The final loss for each edge map is $\mathcal{L}_{wce} = \sum_i^j l_i^j$.

Ignoring ambiguous pixels during optimization can avoid confusing the network and stabilizing the training process with improved performance. However, it is almost impossible to apply the WCE loss to the latent space with the misalignment in both numerical range and spatial size. In particular, the threshold η (generally ranges from 0 to 1) of WCE loss is defined on image space, but the latent code follows the normal distribution and has a various range. Moreover, the pixel-level uncertainty is hard to be aligned with the encoded and down-sampled latent features of different sizes. Therefore, applying the cross-entropy loss directly to latent code inevitably leads to incorrect uncertainty.

On the other hand, one may choose to decode the latent code back to the image level and thus use the uncertainty-aware cross-entropy to directly supervise the predicted edge maps. Unfortunately, this implementation lets the backward gradient go through the redundant autoencoder, making it hard to feed back effective gradients. Besides, the additional gradient computation in the autoencoder leads to a huge GPU memory cost. As shown in Figure 3, we conduct two experiments to show the negative impact of feeding back the gradient through the autoencoder. We name the setting with gradient through autoencoder Baseline-A. As a comparison, we remove the WCE loss but just use Eq. 3 to supervise the latent code, which is named Baseline-B. The performance of Baseline-B is not satisfactory, and Baseline-A even performs worse with $1.5\times$ more GPU memory.

To address this problem, we propose the uncertainty distillation loss that can directly optimize the gradient on the latent space. The results of Baseline-A illustrate that feeding back the gradient through the redundant autoencoder leads to a huge GPU memory cost and hurts the performance, which introduces an inspiration of eliminating the gradient

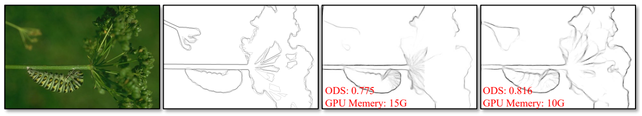

Figure 3: Examples of two baselines with accuracy and memory cost.

of autoencoder based on Baseline-B. Specifically, assuming the reconstructed latent code is $\hat{\boldsymbol{z}}_0$, the decoder of the autoencoder is $\mathcal{D}$, and the decoded edge is $\mathbf{e}_{\mathcal{D}}$, we consider the gradient of WCE loss $\mathcal{L}_{wce}$ by the Chain Rule:

$$\nabla_{\boldsymbol{\theta}} \mathcal{L}_{wce} = \frac{\partial \mathcal{L}_{wce}}{\partial \mathbf{e}_{\mathcal{D}}} \frac{\partial \mathbf{e}_{\mathcal{D}}}{\partial \hat{\boldsymbol{z}}_0} \frac{\hat{\boldsymbol{z}}_0}{\partial \boldsymbol{\theta}}. \tag{8}$$

To remove the negative influence of autoencoder, we skip the gradient through the autoencoder $\partial \mathbf{e}_{\mathcal{D}} / \partial \hat{\boldsymbol{z}}_0$ and modify the gradient $\nabla_{\boldsymbol{\theta}} \mathcal{L}_{wce}$ by:

$$\nabla_{\boldsymbol{\theta}} \mathcal{L}_{wce} = \frac{\partial \mathcal{L}_{wce}}{\partial \mathbf{e}_{\mathcal{D}}} \frac{\hat{\boldsymbol{z}}_0}{\partial \boldsymbol{\theta}}. \tag{9}$$

This implementation reduces the computational cost greatly and allows the WCE loss to be applied to latent code directly. In this way, with the time-variant loss weight $\sigma_t = (1-t)^2$, our final training objective is represented by:

$$\mathcal{L} = \|\mathbf{f}_{\boldsymbol{\theta}} - \mathbf{f}\|^2 + \|\boldsymbol{n}_{\boldsymbol{\theta}} - \boldsymbol{n}\|^2 + \sigma_t \mathcal{L}_{wce}(\mathbf{e}_{\mathcal{D}}, \mathbf{e}_0). \tag{10}$$

Experiments

Datasets

We conduct experiments on four popular edge detection datasets: BSDS (Arbelaez et al. 2010), NYUDv2 (Silberman et al. 2012), Multicue (Mély et al. 2016) and BIPED (Poma, Riba, and Sappa 2020).

BSDS consists of 200, 100, and 200 images in the training set, validation set, and test set, respectively. Each image has 4 to 9 annotators and the final edge ground truth is computed by taking their average.

NYUDv2 is built for indoor scene parsing and is also applied for edge detection evaluation. It contains 1449 densely annotated RGB-D images, and is divided into 381 training, 414 validation and 654 testing images.

Multicue consists of images from 100 challenging natural scenes. Each image is annotated by several people as well. We randomly split the 100 images into training and evaluation sets, consisting of 80 and 20 images respectively. We repeat the process on Multicue-edge three times and average the scores as the final results.

BIPED contains 250 annotated images of outdoor scenes and is split into a training set of 200 images and a testing set of 50 images. All images are carefully annotated at single-pixel width by experts in the computer vision field.

Previous methods generally augment the dataset with various strategies. For example, images in BSDS are augmented with flipping (2×), scaling (3×), and rotation (16×), leading to a training set that is 96× larger than the original version. Others are concluded in Table 1. However, our method trains all datasets with only randomly cropped patches of

320×320. In BSDS, we apply random flipping and scaling. In NYUDv2, Multicue and BIPED datasets, only random flipping is adopted.

Datasets	Augmentation strategies
BSDS	F (2×), S (3×), R (16×)=96×
NYUD	F (2×), S (3×), R (4×)=24×
Multicue	F (2×), C (3×), R (16×)=96×
BIPED	F (2×), C (3×), R (16×), G(3×)= 288×

Table 1: Augmentation strategies adopted on four edge detection benchmarks for previous methods. F: flipping, S: scaling, R: rotation, C: cropping, G: gamma correction.

Implementation Details

We implement our DiffusionEdge using PyTorch (Paszke et al. 2019). To train the autoencoder, we collect the edge labels from the training set of all the datasets. For training the denoising U-Net, we set the smallest time step to 0.0001. We train the models using AdamW optimizer with an attenuated learning rate (from $5e^{-5}$ to $5e^{-6}$) for 25k iterations, and each training takes up about 15 GPU hours. We employ the exponential moving average (EMA) to prevent unstable model performances during the training process. The balancing weight λ and the threshold η to identify uncertain edge pixels are set to 1.1 and 0.3, respectively, for all experiments. We train all datasets with randomly cropped patches of size 320×320 with batch size 16. We conduct inferences with slide 240×240 and take the average value under overlap areas. All the training is conducted on a single RTX 3090 GPU. When inferencing each single image on BSDS dataset, with the sampling Equation 2, it takes about 3.5GB GPU memory, 1.2 seconds for one-step sampling and 3.2 seconds for five steps on a 3080Ti GPU.

Evaluation Metrics

To evaluate the precision, recall, and F-score for general edge detection, the predicted edge map should be binarized by an optimal threshold. Following prior works, we compute the F-scores of Optimal Dataset Scale (ODS) and Optimal Image Scale (OIS). ODS employs a fixed threshold throughout the dataset, while OIS chooses an optimal threshold for each image. F-scores are computed by $F = \frac{2 \cdot P \cdot R}{P+R}$, where P denotes precision and R denotes recall. For ODS and OIS, the maximum allowed distances between corresponding pixels from predicted edges and ground truths are set to 0.011 for NYUD and 0.0075 for other datasets.

To comprehensively evaluate the crispness of edge maps, following previous works (Huan et al. 2021; Ye et al. 2023a), we also report the Standard evaluation protocol (SEval), Crispness-emphasized evaluation protocol (CEval), and the Average Crispness (AC). SEval is calculated after applying a standard post-processing scheme containing an NMS step and a mathematical morphology operation to obtain thinner edge maps. CEval is calculated without any post-processing so that thick edge maps generally get lower precision with more false positive samples. The AC for each edge map is calculated as the ratio of the sum of pixel values after NMS, to the sum of pixel values before NMS, which ranges from 0 to 1. Larger AC means crisper edge maps.

Ablation Study

The effect of key components. We first conduct experiments to verify the impact of the Adaptive FFT-filter (AF) and Uncertainty Distillation (UD) strategy. The quantitative results are summarized in Table 2. We can observe that each single AF or UD can promote the performance, while UD is more critical since it plays an important role of optimizing the latent space with valuable uncertainty information. Considering that the AC varies very slightly, the combination of AF and UD achieves the best performance.

AF	UD	ODS	OIS	AC
×	×	0.816	0.829	0.521
×	√	0.831	0.845	0.528
√	×	0.825	0.837	0.461
√	√	0.834	0.848	0.476

Table 2: Ablation study of the effectiveness of the proposed Adaptive FFT-filter (AF) and Uncertainty Distillation (UD) in DiffusionEdge on BSDS dataset. All results are computed with a single scale input, and the same for others.

The effect of backbones and diffusion steps. We study the impact of different backbones for the image (condition) encoder with ResNet101 (He et al. 2016), Effecientnet-b7 (Tan and Le 2019) and Swin-B (Liu et al. 2021). Also, the number of iterating steps could be another key parameter in diffusion models. All the results are reported in Table 3. We can observe that the crispness varies slightly in all settings, revealing the superiority of DiffusionEdge for crisp edge detection. Swin performs better than other backbones, and we find the number of sampling steps (ranging from 1 to 50) brings litter difference (<0.4% in ODS and OIS) to the final results. Moreover, only one sample step can already achieve state-of-the-art performance. Since more steps mean more inference time, considering all the correctness, crispness and efficiency, we adopt step 5 as the standard setting for all experiments.

Backbone	ODS	OIS	AC
ResNet	0.823	0.837	0.514
EffecientNet	0.829	0.840	0.508
Swin	0.834	0.848	0.476
Steps (with Swin)	ODS	OIS	AC
Step 1	0.833	0.844	0.453
Step 3	0.835	0.847	0.476
Step 5	0.834	0.848	0.476
Step 10	0.834	0.848	0.476
Step 20	0.833	0.847	0.475
Step 50	0.833	0.846	0.478

Table 3: The ablations about different backbones and the number of iterating steps for DiffusionEdge.

Comparison with State-of-the-arts

On BSDS. We compare our model with *traditional detectors* including Canny (Canny 1986), SE (Dollár and Zit-

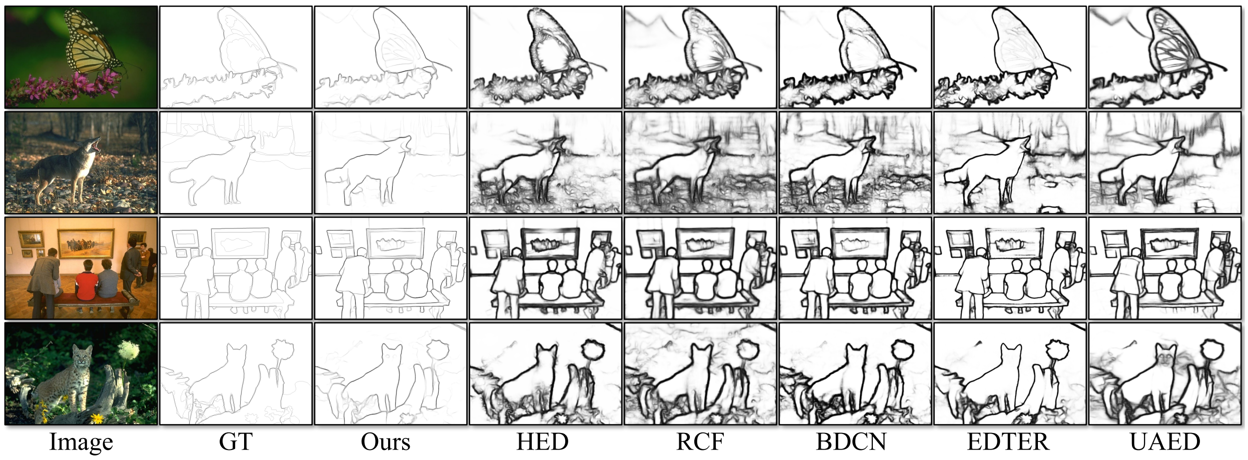

Figure 4: Qualitative comparisons on BSDS dataset with previous state-of-the-arts. Edge maps generated by our DiffusionEdge are both accurate and crisp with less noise. Zoom-in is highly recommended to observe the details.

nick 2014) and OEF (Hallman and Fowlkes 2015), *CNN-based detectors* including N^4-Fields (Ganin and Lempitsky 2014), DeepContour (Shen et al. 2015), HFL (Bertasius, Shi, and Torresani 2015), CEDN (Yang et al. 2016), Deep Boundary (Kokkinos 2015), COB (Maninis et al. 2017), CED (Wang et al. 2018), AMH-Net (Xu et al. 2017), DCD (Liao et al. 2017), LPCB (Deng et al. 2018), HED (Xie and Tu 2015), RCF (Liu et al. 2017), BDCN (He et al. 2019), PiDiNet (Su et al. 2021), UAED (Zhou et al. 2023) and *the transformer-based detector* EDTER (Pu et al. 2022). The best results of all methods are taken from their publications.

Methods	SEval		CEval		AC
	ODS	OIS	ODS	OIS	
Canny	0.611	0.676	-	-	-
SE	0.743	0.764	-	-	-
OEF	0.746	0.770	-	-	-
N^4-Fields	0.753	0.769	-	-	-
DeepContour	0.757	0.776	-	-	-
HFL	0.767	0.788	-	-	-
CEDN	0.788	0.804	-	-	-
DeepBoundary	0.789	0.811	-	-	-
COB	0.793	0.820	-	-	-
CED	0.794	0.811	0.642	0.656	0.207
AMH-Net	0.798	0.829	-	-	-
DCD	0.799	0.817	-	-	-
LPCB	0.800	0.816	0.693	0.700	-
HED	0.788	0.808	0.588	0.608	0.215
RCF	0.798	0.815	0.585	0.604	0.189
BDCN	0.806	0.826	0.636	0.650	0.233
PiDiNet	0.789	0.803	0.578	0.587	0.202
EDTER	0.824	0.841	0.698	0.706	0.288
UAED	0.829	0.847	0.722	0.731	0.227
Ours	**0.834**	**0.848**	**0.749**	**0.754**	**0.476**

Table 4: Quantitative results on the BSDS dataset. For fair comparison, we only list the single-scale results generated by models trained with only BSDS data. Note that other methods are trained with augmented dataset (96×), while we train DiffusionEdge with only random flipping and scaling.

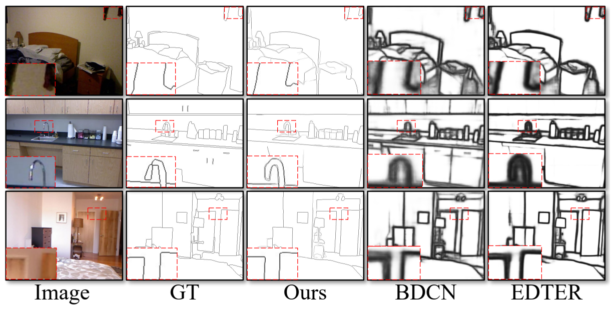

Figure 5: Qualitative comparisons on NYUDv2 dataset with two state-of-the-art CNN-based and transformer-based methods. Edge maps generated by DiffusionEdge are much crisper and cleaner with competitive performance.

By observing the quantitative and qualitative results in Table 4 and Figure 4, several conclusions can be drawn: (a) The proposed method achieves the best results in all settings, especially the AC, which means edge maps generated by DiffusionEdge are much more crisper than other methods; (b) Generally, the performance drop between SEval and CEval is smaller with crisper edge maps (larger AC), it is reasonable that thick edge maps contain many ambiguous false positive edges around true positive ones, evaluating without any post-processing lead to very low precision and thus low F-scores of ODS and OIS; (c) Thanks to the adaptive FFT-filter and uncertainty distillation strategy, our qualitative results perform even better with much less noise and more semantically meaningful contours, especially in challenging scenarios with complicated background and texture.

On NYUDv2. We conduct experiments on RGB images and compare DiffusionEdge with state-of-the-art methods including AMH-Net, LPCB, HED, RCF, BDCN, PiDiNet and EDTER. Quantitative and qualitative results are shown in Table 5 and Figure 5, respectively. Our method achieves comparable performance under SEval. However, edge maps generated by other methods are extremely thick with all ACs smaller than 0.2, leading to a significant performance drop under CEval. Such thick edge maps may come from training with the possibly existing label offsets for CNN-based methods (Ye et al. 2023a). However, DiffusionEdge can directly learn to recover the single-width label and maintain the crispness with slight performance change without post-processing. Consequently, compared to the second best (EDTER), we increase the ODS, OIS of CEval and AC by a large margin of 30.2%, 28.1% and 65.1%, respectively.

Methods	SEval		CEval		AC
	ODS	OIS	ODS	OIS	
AMH-Net	0.744	0.758	-	-	-
LPCB	0.739	0.754	-	-	-
HED	0.722	0.737	0.387	0.404	-
RCF	0.745	0.759	0.398	0.413	-
BDCN	0.748	0.762	0.426	0.450	0.162
PiDiNet	0.733	0.747	0.399	0.424	0.173
EDTER	**0.774**	**0.789**	0.430	0.457	0.195
Ours	0.761	0.766	**0.732**	**0.738**	**0.846**

Table 5: Quantitative comparisons on NYUDv2. All results are computed with a single scale input. Note that other methods are trained with augmented dataset (24×), while we train DiffusionEdge with only random flipping.

On Multicue and BIPED. We further compare DiffusionEdge with HED, RCF, BDCN, DexiNed, PiDiNet, EDTER and UAED, on the datasets of Multicue-edge and BIPED, via the standard evaluation procedure. As shown in Table 6, our method is superior in both correctness and crispness. It is worth noting that our method achieves a high AC of 0.849 on the BIPED dataset, which means the edges are almost all single-width with no ambiguity, as demonstrated in Figure 6. Such a success reveals the great potential to directly adopt the predicted results of DiffusionEdge without any post-processing for downstream tasks.

Methods	Multicue dataset			BIPED dataset		
	ODS	OIS	AC	ODS	OIS	AC
Human	0.750	-	-	-	-	-
Multicue	0.830	-	-	-	-	-
HED	0.851	0.864	-	0.829	0.847	-
RCF	0.851	0.862	-	0.843	0.859	-
BDCN	0.891	0.898	-	0.839	0.854	-
DexiNed	0.872	0.881	0.274	0.859	0.867	0.295
PiDiNet	0.874	0.878	0.204	0.868	0.876	0.232
EDTER	0.894	0.900	0.196	0.893	0.898	0.26
UAED	0.895	0.902	0.211	-	-	-
Ours	**0.904**	**0.909**	**0.462**	**0.899**	**0.901**	**0.849**

Table 6: Quantitative comparisons on Multicue and BIPED. All results are computed with a single scale input.

On Crispness. To further verify the superiority of DiffusionEdge for crisp edge detection, we compare the AC of our method and other strategies proposed for generating crisp edge maps. Here we apply the Dice loss (Deng et al. 2018) ("-D" in table), the tracing loss (Huan et al. 2021) ("-T" in table) and the Guided Label Refinement (Ye et al. 2023a) ("-R" in table) based on PiDiNet (Su et al. 2021). As shown

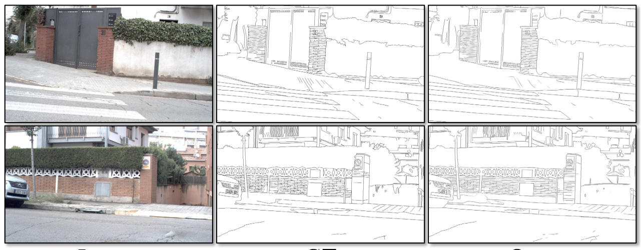

Figure 6: Qualitative examples on BIPED dataset.

in Table 7, our DiffusionEdge achieves the best crispness in all cases compared with other methods. Although much efforts have been made for improving the crispness of CNN-based networks (PiDiNet here as an example), the crispness is still limited by the encoder-decoder architecture in nature. However, the diffusion-based edge detection scheme recovers edge maps directly on the original size and the predictions can be almost as crisp as the ground truths.

Methods	AC		
	BSDS	Multicue	BIPED
PiDiNet-D	0.306	0.208	0.34
PiDiNet-T	0.333	0.217	0.296
PiDiNet-R	0.424	0.424	0.512
Ours	**0.476**	**0.462**	**0.849**

Table 7: Comparisons of the average crispness (AC) on BSDS, Multicue and BIPED dataset with the backbone of PiDiNet. "-D", "-T" and "-R" means training with dice loss, tracing loss and training with refined labels, respectively.

Conclusions and Limitations

In this paper, we introduce the first diffusion-based network for crisp edge detection. With several technical designs including the adaptive FFT-filter and uncertainty distillation strategy, our DiffusionEdge is able to directly generate accurate and crisp edge maps without any post-processing. Extensive experiments demonstrate the superiority of DiffusionEdge both quantitatively and qualitatively. The crispness is even satisfactory enough and shows the potential for benefiting subsequent tasks in an end-to-end manner.

Limitations. The correctness and crispness of edge maps extracted by DiffusionEdge can be simultaneously qualified for downstream tasks. However, another one of the three challenges, the efficiency, remains an open problem. Improving the diffusion model for faster inference speed is still a promising future direction to explore.

Acknowledgments

This work is supported in part by the NSFC (62172155, 62072465, 62325221, 62132021, 62002375, 62002376), the National Key Research and Development Program of China (2018AAA0102200), the Natural Science Foundation of Hunan Province of China(2021RC3071, 2022RC1104, 2021JJ40696) and the NUDT Research Grants (ZK22-52).

References

Arbelaez, P.; Maire, M.; Fowlkes, C.; and Malik, J. 2010. Contour detection and hierarchical image segmentation. *IEEE transactions on pattern analysis and machine intelligence*, 33(5): 898–916.

Austin, J.; Johnson, D. D.; Ho, J.; Tarlow, D.; and Van Den Berg, R. 2021. Structured denoising diffusion models in discrete state-spaces. *Advances in Neural Information Processing Systems*, 34: 17981–17993.

Avrahami, O.; Lischinski, D.; and Fried, O. 2022. Blended diffusion for text-driven editing of natural images. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 18208–18218.

Bertasius, G.; Shi, J.; and Torresani, L. 2015. High-for-low and low-for-high: Efficient boundary detection from deep object features and its applications to high-level vision. In *Proceedings of the IEEE international conference on computer vision*, 504–512.

Brempong, E. A.; Kornblith, S.; Chen, T.; Parmar, N.; Minderer, M.; and Norouzi, M. 2022. Denoising pretraining for semantic segmentation. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 4175–4186.

Canny, J. 1986. A computational approach to edge detection. *IEEE Transactions on pattern analysis and machine intelligence*, (6): 679–698.

Chen, S.; Sun, P.; Song, Y.; and Luo, P. 2022. Diffusiondet: Diffusion model for object detection. *arXiv preprint arXiv:2211.09788*.

Cheng, T.; Wang, X.; Huang, L.; and Liu, W. 2020. Boundary-preserving mask r-cnn. In *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part XIV 16*, 660–676. Springer.

Deng, R.; Shen, C.; Liu, S.; Wang, H.; and Liu, X. 2018. Learning to predict crisp boundaries. In *Proceedings of the European conference on computer vision (ECCV)*, 562–578.

Dollár, P.; and Zitnick, C. L. 2014. Fast edge detection using structured forests. *IEEE transactions on pattern analysis and machine intelligence*, 37(8): 1558–1570.

Ganin, Y.; and Lempitsky, V. 2014. -fields: neural network nearest neighbor fields for image transforms. In *Asian conference on computer vision*, 536–551. Springer.

Gu, S.; Chen, D.; Bao, J.; Wen, F.; Zhang, B.; Chen, D.; Yuan, L.; and Guo, B. 2022. Vector quantized diffusion model for text-to-image synthesis. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 10696–10706.

Hallman, S.; and Fowlkes, C. C. 2015. Oriented edge forests for boundary detection. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, 1732–1740.

He, J.; Zhang, S.; Yang, M.; Shan, Y.; and Huang, T. 2019. Bi-directional cascade network for perceptual edge detection. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 3828–3837.

He, K.; Zhang, X.; Ren, S.; and Sun, J. 2016. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, 770–778.

Ho, J.; Jain, A.; and Abbeel, P. 2020. Denoising diffusion probabilistic models. *Advances in neural information processing systems*, 33: 6840–6851.

Huan, L.; Xue, N.; Zheng, X.; He, W.; Gong, J.; and Xia, G.-S. 2021. Unmixing convolutional features for crisp edge detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 44(10): 6602–6609.

Huang, Y.; Qin, Z.; Liu, X.; and Xu, K. 2023. Decoupled Diffusion Models with Explicit Transition Probability. *arXiv preprint arXiv:2306.13720*.

Kittler, J. 1983. On the accuracy of the Sobel edge detector. *Image and Vision Computing*, 1(1): 37–42.

Kokkinos, I. 2015. Pushing the boundaries of boundary detection using deep learning. *arXiv preprint arXiv:1511.07386*.

Liao, Y.; Fu, S.; Lu, X.; Zhang, C.; and Tang, Z. 2017. Deep-learning-based object-level contour detection with CCG and CRF optimization. In *2017 IEEE International Conference on Multimedia and Expo (ICME)*, 859–864. IEEE.

Liu, Y.; Cheng, M.-M.; Hu, X.; Wang, K.; and Bai, X. 2017. Richer convolutional features for edge detection. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, 3000–3009.

Liu, Z.; Lin, Y.; Cao, Y.; Hu, H.; Wei, Y.; Zhang, Z.; Lin, S.; and Guo, B. 2021. Swin transformer: Hierarchical vision transformer using shifted windows. In *Proceedings of the IEEE/CVF international conference on computer vision*, 10012–10022.

Maninis, K.-K.; Pont-Tuset, J.; Arbeláez, P.; and Van Gool, L. 2017. Convolutional oriented boundaries: From image segmentation to high-level tasks. *IEEE transactions on pattern analysis and machine intelligence*, 40(4): 819–833.

Mély, D. A.; Kim, J.; McGill, M.; Guo, Y.; and Serre, T. 2016. A systematic comparison between visual cues for boundary detection. *Vision research*, 120: 93–107.

Nazeri, K.; Ng, E.; Joseph, T.; Qureshi, F. Z.; and Ebrahimi, M. 2019. Edgeconnect: Generative image inpainting with adversarial edge learning. *arXiv preprint arXiv:1901.00212*.

Nichol, A.; Dhariwal, P.; Ramesh, A.; Shyam, P.; Mishkin, P.; McGrew, B.; Sutskever, I.; and Chen, M. 2021. Glide: Towards photorealistic image generation and editing with text-guided diffusion models. *arXiv preprint arXiv:2112.10741*.

Paszke, A.; Gross, S.; Massa, F.; Lerer, A.; Bradbury, J.; Chanan, G.; Killeen, T.; Lin, Z.; Gimelshein, N.; Antiga, L.; et al. 2019. Pytorch: An imperative style, high-performance deep learning library. *Advances in neural information processing systems*, 32.

Poma, X. S.; Riba, E.; and Sappa, A. 2020. Dense extreme inception network: Towards a robust cnn model for edge detection. In *Proceedings of the IEEE/CVF winter conference on applications of computer vision*, 1923–1932.

Popov, V.; Vovk, I.; Gogoryan, V.; Sadekova, T.; and Kudinov, M. 2021. Grad-tts: A diffusion probabilistic model for text-to-speech. In *International Conference on Machine Learning*, 8599–8608. PMLR.

Pu, M.; Huang, Y.; Liu, Y.; Guan, Q.; and Ling, H. 2022. Edter: Edge detection with transformer. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 1402–1412.

Revaud, J.; Weinzaepfel, P.; Harchaoui, Z.; and Schmid, C. 2015. Epicflow: Edge-preserving interpolation of correspondences for optical flow. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, 1164–1172.

Rombach, R.; Blattmann, A.; Lorenz, D.; Esser, P.; and Ommer, B. 2022. High-resolution image synthesis with latent diffusion models. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 10684–10695.

Shen, W.; Wang, X.; Wang, Y.; Bai, X.; and Zhang, Z. 2015. Deepcontour: A deep convolutional feature learned by positive-sharing loss for contour detection. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, 3982–3991.

Silberman, N.; Hoiem, D.; Kohli, P.; and Fergus, R. 2012. Indoor segmentation and support inference from rgbd images. In *Computer Vision–ECCV 2012: 12th European Conference on Computer Vision, Florence, Italy, October 7-13, 2012, Proceedings, Part V 12*, 746–760. Springer.

Sohl-Dickstein, J.; Weiss, E.; Maheswaranathan, N.; and Ganguli, S. 2015. Deep unsupervised learning using nonequilibrium thermodynamics. In *International conference on machine learning*, 2256–2265. PMLR.

Su, Z.; Liu, W.; Yu, Z.; Hu, D.; Liao, Q.; Tian, Q.; Pietikäinen, M.; and Liu, L. 2021. Pixel difference networks for efficient edge detection. In *Proceedings of the IEEE/CVF international conference on computer vision*, 5117–5127.

Tan, M.; and Le, Q. 2019. Efficientnet: Rethinking model scaling for convolutional neural networks. In *International conference on machine learning*, 6105–6114. PMLR.

Wang, Y.; Zhao, X.; Li, Y.; and Huang, K. 2018. Deep crisp boundaries: From boundaries to higher-level tasks. *IEEE Transactions on Image Processing*, 28(3): 1285–1298.

Wu, J.; FU, R.; Fang, H.; Zhang, Y.; Yang, Y.; Xiong, H.; Liu, H.; and Xu, Y. 2023. MedSegDiff: Medical Image Segmentation with Diffusion Probabilistic Model. In *Medical Imaging with Deep Learning*.

Xie, S.; and Tu, Z. 2015. Holistically-nested edge detection. In *Proceedings of the IEEE international conference on computer vision*, 1395–1403.

Xiong, W.; Yu, J.; Lin, Z.; Yang, J.; Lu, X.; Barnes, C.; and Luo, J. 2019. Foreground-aware image inpainting. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 5840–5848.

Xu, D.; Ouyang, W.; Alameda-Pineda, X.; Ricci, E.; Wang, X.; and Sebe, N. 2017. Learning deep structured multi-scale features using attention-gated crfs for contour prediction. *Advances in neural information processing systems*, 30.

Yang, J.; Price, B.; Cohen, S.; Lee, H.; and Yang, M.-H. 2016. Object contour detection with a fully convolutional encoder-decoder network. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, 193–202.

Ye, Y.; Yi, R.; Gao, Z.; Cai, Z.; and Xu, K. 2023a. Delving into Crispness: Guided Label Refinement for Crisp Edge Detection. *IEEE Transactions on Image Processing*.

Ye, Y.; Yi, R.; Gao, Z.; Zhu, C.; Cai, Z.; and Xu, K. 2023b. NEF: Neural Edge Fields for 3D Parametric Curve Reconstruction from Multi-view Images. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 8486–8495.

Zhou, C.; Huang, Y.; Pu, M.; Guan, Q.; Huang, L.; and Ling, H. 2023. The Treasure Beneath Multiple Annotations: An Uncertainty-aware Edge Detector. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 15507–15517.

Zitnick, C. L.; and Dollár, P. 2014. Edge boxes: Locating object proposals from edges. In *Computer Vision–ECCV 2014: 13th European Conference, Zurich, Switzerland, September 6-12, 2014, Proceedings, Part V 13*, 391–405. Springer.