EDTER: Edge Detection with Transformer

Mengyang Pu[1,3], Yaping Huang[1]*, Yuming Liu[2], Qingji Guan[1], Haibin Ling[3]
[1]Beijing Key Laboratory of Traffic Data Analysis and Mining, Beijing Jiaotong University, China
[2]Shenzhen Urban Transport Planning Center Co.,Ltd., China
[3]Department of Computer Science, Stony Brook University, USA
{mengyangpu, yphuang, qjguan}@bjtu.edu.cn; liuyuming@sutpc.com; hling@cs.stonybrook.edu

Abstract

Convolutional neural networks have made significant progresses in edge detection by progressively exploring the context and semantic features. However, local details are gradually suppressed with the enlarging of receptive fields. Recently, vision transformer has shown excellent capability in capturing long-range dependencies. Inspired by this, we propose a novel transformer-based edge detector, Edge Detection TransformER (EDTER)*, to extract clear and crisp object boundaries and meaningful edges by exploiting the full image context information and detailed local cues simultaneously. EDTER works in two stages. In Stage I, a global transformer encoder is used to capture long-range global context on coarse-grained image patches. Then in Stage II, a local transformer encoder works on fine-grained patches to excavate the short-range local cues. Each transformer encoder is followed by an elaborately designed Bi-directional Multi-Level Aggregation decoder to achieve high-resolution features. Finally, the global context and local cues are combined by a Feature Fusion Module and fed into a decision head for edge prediction. Extensive experiments on BSDS500, NYUDv2, and Multicue demonstrate the superiority of EDTER in comparison with state-of-the-arts. The source code is available at* https://github.com/MengyangPu/EDTER.

1. Introduction

Edge detection is one of the most fundamental problems in computer vision and has a wide variety of applications, such as image segmentation [8, 23, 39, 44, 45, 47], object detection [23], and video object segmentation [5, 57, 59]. Given an input image, edge detection aims to extract accurate object boundaries and visually salient edges. It is challenging due to many factors including complex backgrounds, inconsistent annotations, and so on.

*Corresponding author.

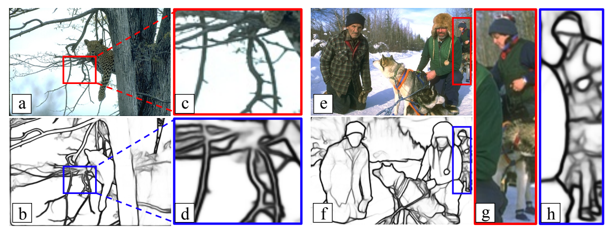


Figure 1. **Examples of edge detection.** Our method, EDTER, extracts clear boundaries and edges by exploiting both global and local cues. (a, e): Input images from BSDS500 [1]. (b,f): Detected edges by EDTER. (c,d,g,h): Zoomed-in patches.

Edge detection is closely related to the context and semantic image cues. It is thus crucial to obtain appropriate representation to capture both high and low level visual cues. Traditional methods [6,14,28,34,41,63] mostly obtain edges based on low-level local cues, *e.g.*, color and texture. Benefiting from the effectiveness of convolutional neural networks (CNNs) in learning semantic features, significant progress has been made for edge detection [3,4,29,48]. The CNN features progressively capture global and semantic-aware visual concepts with the enlargement of the receptive fields, while many essential fine details are inevitably and gradually lost at the mean time. To include more details, methods in [22,36,37,65,66] aggregate the features of deep and shallow layers. However, such shallow features reflect mainly local intensity variation without considering semantic context, resulting in noisy edges.

Inspired by the recent success of vision transformers [9, 16, 61, 72], especially their capability of modeling long-range contextual information, we propose to tailor transformers for edge detection. Two main challenges, however, need to be solved. Firstly, transformers are often applied to patches with a relatively large size due to computation concerns, while coarse-grained patches are unfavorable for learning accurate features for edges. It is crucial to perform self-attention on fine-grained patches without increasing the computational burden. Second, as shown in Fig. 1 (d), extracting precise edges from intersected and thin objects is challenging. So it is necessary to design an effective decoder for generating edge-aware high-resolution features.

To address the above issues, we develop a two-stage framework (Fig. 2), named Edge Detection TransformER (EDTER), to explore global context information and excavate fine-grained cues in local regions. In stage I, we split the image into coarse-grained patches and run a global transformer encoder on them to capture long-range global context. Then, we develop a novel Bi-directional Multi-Level Aggregation (BiMLA) decoder to generate high-resolution representations for edge detection. In stage II, we first divide the whole image into multiple sequences of fine-grained patches by sampling with a non-overlapping sliding window. Then a local transformer works on each sequence in turn to explore the short-range local cues. Afterward, all local cues are integrated and fed into a local BiMLA decoder to achieve the pixel-level feature maps. Finally, the information from both stages is fused by a Feature Fusion Module (FFM) and then is fed into a decision head to predict the final edge map. With the above efforts, EDTER can generate crisp and less noisy edge maps (Fig. 1).

Our contributions are summarized as follows: (1) We propose a novel transformer-based edge detector, Edge Detection TransformER (EDTER), to detect object contours and meaningful edges in natural images. To our best knowledge, it is the first transformer-based edge detection model. (2) EDTER is designed to effectively explore long-range global context (Stage I) and capture fine-grained local cues (Stage II). Moreover, we propose a novel Bi-directional Multi-Level Aggregation (BiMLA) decoder to boost the information flow in the transformer. (3) To effectively integrate the global and local information, we use a Feature Fusion Module (FFM) to fuse the cues extracted from Stage I and Stage II. (4) Extensive experiments demonstrate the superiority of EDTER over the state-of-the-art methods on three well-known edge detection benchmarks, including BSDS500, NYUDv2, and Multicue.

2. Related Work

As a fundamental task in computer vision, edge detection has been extensively studied over years. In the following, we highlight two lines of works most related to ours.

Edge Detection. Early edge detectors [6, 28, 63], such as Sobel [28] and Canny [6], focus on analysing the image gradients to extract the edges. These methods provide elementary low-level cues and are widely used in computer vision applications. Learning-based methods [14, 34, 41] tend to integrate different low-level features and train a classifier to obtain boundaries and edges. Although these approaches achieve impressive performance compared to early works, they are based on hand-crafted features, limiting the ability to detect semantic boundaries and meaningful edges.

Recently, convolutional neural networks (CNNs) have been successfully introduced the edge detection study [3, 4, 11, 12, 26, 29, 40, 46, 48, 52, 66]. DeepEdge [3] exploits object-aware cues extracted by multi-level CNN for contour detection. The method in [48] first partitions contour patches into sub-classes and then learns model parameters to fit each subclass. More recently, some approaches improve edge detection [22, 36, 37, 65, 66], segmentation [8, 54, 70], and object detection [35] by using hierarchical multi-scale features. Inspired by the seminal work of [65], most edge detectors [22, 36, 37, 66] generate object boundaries from hierarchical features by multi-level learning. Specifically, HED [65] learns rich hierarchical features by performing supervisions on side output layers, which boosts the performance of edge detection. RCF [36] combines hierarchical features from all convolutional layers into a holistic architecture. To achieve effective results, BDCN [22] uses layer-specific supervision inferred from a bi-directional cascade structure to guide the training of each layer. PiDiNet [53] integrates the traditional edge detection operators into a CNN model for enhanced performance.

Vision transformer. First introduced to handle natural language tasks [13, 30, 56], transformer is later extended to vision tasks owing to its capacity in modeling long-range dependencies including image classification [16], semantic segmentation [72], and object detection [7]. Recently, it is applied in conjunction with CNN in DETR [7] and the other variants [10, 27, 31, 58, 73]. More recently, vision transformer (ViT) [16] directly uses the transformer to the sequences of image patches and achieves the state-of-the-art. This architecture brings direct inspiration to other computer vision tasks [32, 38, 55, 68, 72]. For example, SETR [72] shows superior accuracy in semantic segmentation using a pure transformer on image patches. These works demonstrate the effectiveness of transformers in capturing long-range dependencies and global context.

Our work is inspired by the above pioneer studies [16, 38, 72], but is significantly different in two aspects. First, the proposed EDTER, to the best of my knowledge, is the first usage of the transformer for generic edge detection. Second, our key idea is to learn the features that contain the global image context and fine-grained local cues by a two-stage framework with an affordable computational cost. With the integration of the global context and local cues, EDTER is superior in edge detection.

3. Edge Detection with Transformer

3.1. Overview

The overall framework of the proposed EDTER is illustrated in Fig. 2. EDTER explores the full image context information and fine-grained cues in two stages. In Stage I, we first split the input image into a sequence of coarse-grained patches and use a global transformer encoder to learn the global context information. Then a Bi-directional

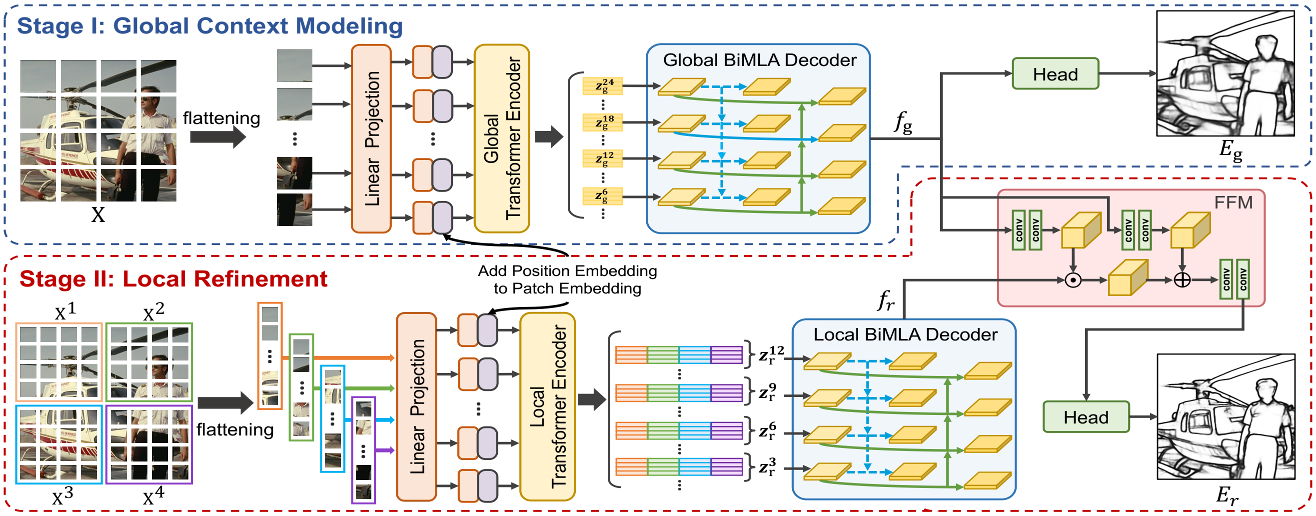


Figure 2. **Overall framework.** In Stage I, we first feed the image into a global transformer encoder to compute the global attentions. Then, a global BiMLA decoder (see Fig. 3) generates the high-resolution features that are used to predict the edge maps via a decision head. In Stage II, similar to Stage I, the partitioned patches are inputted into a local transformer encoder to generate the local attentions. The concatenated attentions are utilized to decode the high-resolution features. At last, a decision head predicts the edge maps with the features of Stage I and Stage II fused by FFM.

Multi-Level Aggregation (BiMLA) decoder is used to generate the high-resolution features. In Stage II, the whole image is divided into multiple sequences of fine-grained patches by sampling with a non-overlapping sliding window. Then we execute a local transformer encoder on each sequence in turn to capture short-range local cues. We integrate all local cues and input them into a local BiMLA decoder to achieve the pixel-level feature maps. Finally, the global and local features are integrated by a Feature Fusion Module (FFM) and then are fed into a decision head to predict the final edge maps.

3.2. Review Vision Transformer

The transformer encoders in our framework follow the vision transformer (ViT) in [16], as briefly described below.

Image Partition. The first step in ViT is to transform a 2D image, denoted by $X \in \mathbb{R}^{H \times W \times 3}$, into a 1D sequence of image patches [16, 72]. Concretely, we uniformly split X into a sequence of flattened image patches of size $P \times P$, resulting in $\frac{H}{P} \times \frac{W}{P}$ vision tokens. Then, the sequence is mapped into a latent embedding space by a learnable linear projection. The projected features are called patch embeddings. Further, to preserve positional information, the standard learnable 1D position embeddings are added to the patch embeddings. Finally, the combined embeddings (denoted as z^0) are fed into the transformer encoder.

Transformer Encoder. The standard transformer encoder [56] consists of L transformer blocks. Each block has a multi-head self-attention operation (MSA), a multi-layer perceptron (MLP), and two Layernorm steps (LN). Moreover, a residual connection layer is applied after each block. Generally, MSA performs M self-attentions in parallel and projects their concatenated outputs. In the m^{th} self-attention, given the output $z^{l-1} \in \mathbb{R}^{N \times C}$ of the $(l-1)^{th}$ transformer block, the queries $Q \in \mathbb{R}^{N \times U}$, keys $K \in \mathbb{R}^{N \times U}$, and values $V \in \mathbb{R}^{N \times U}$ are computed by

$$Q = \hat{z}^{l-1} W_Q, \quad K = \hat{z}^{l-1} W_K, \quad V = \hat{z}^{l-1} W_V, \tag{1}$$

where $\hat{z}^{l-1} = \mathrm{LN}(z^{l-1})$, $W_Q, W_K, W_V \in \mathbb{R}^{C \times U}$ are the parameter matrices, C is the dimension of embeddings, and U is the dimension of Q, K, and V. Then, we compute the output of the m^{th} self-attention based on the pairwise similarity between two elements of the sequence by

$$y_{\mathrm{sa}}^m = \mathrm{softmax}\Big(\frac{QK^\top}{\sqrt{d}}\Big)V. \tag{2}$$

where y_{sa}^m is the computed attention weight. Finally, MSA can be formulated as

$$y_{\mathrm{msa}} = \mathrm{MSA}(\hat{z}^{l-1}) = \left[y_{\mathrm{sa}}^1, y_{\mathrm{sa}}^2, \dots, y_{\mathrm{sa}}^M\right] W_{\mathrm{O}}, \tag{3}$$

where y_{msa} is the output of MSA, $W_{\mathrm{O}} \in \mathbb{R}^{M \cdot U \times C}$ represents the projection parameters, and $[\cdot]$ is the concatenation. In this work, we fix $M = 16$ following the setting in [16].

3.3. Stage I: Global Context Modeling

Generally, edges and boundaries in images are defined to be semantically meaningful. It is crucial to capture the abstract cues and the global context of the whole image. In the first stage, we explore the global contextual features on coarse-grained patches by a global transformer encoder $\mathcal{G}_{\mathrm{E}}$ and a global decoder $\mathcal{G}_{\mathrm{D}}$.

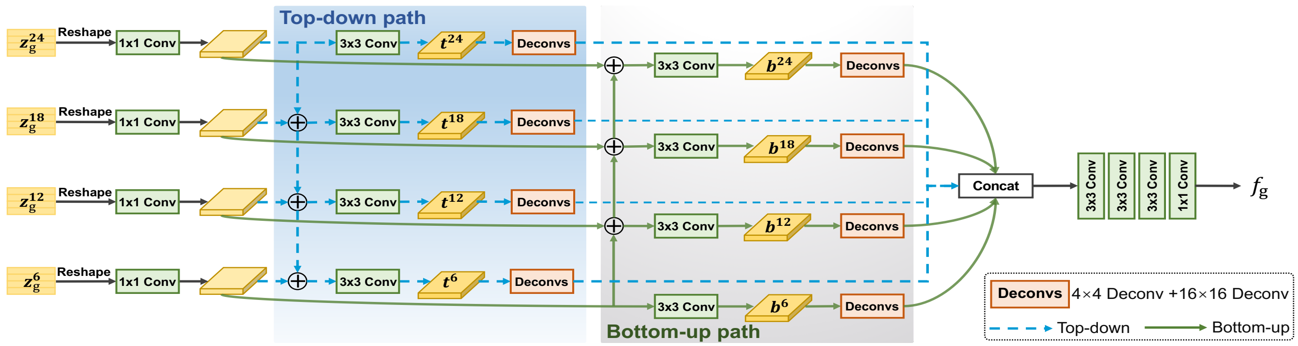


Figure 3. The detailed architecture of the BiMLA decoder consists of a top-down path and a bottom-up path.

Specifically, we first split the input image into a sequence of coarse-grained patches of size 16×16, and then generate the embeddings $z_{\rm g}^0$ that serve as input of the encoder. Next, the global transformer encoder $\mathcal{G}_{\rm E}$ works on the embeddings $z_{\rm g}^0$ to compute the global attentions,

$$z_{\rm g} = \{z_{\rm g}^1, z_{\rm g}^2, \dots, z_{\rm g}^{L_{\rm g}}\} = \mathcal{G}_{\rm E}(z_{\rm g}^0), \tag{4}$$

where $z_{\rm g}^1, z_{\rm g}^2, \dots, z_{\rm g}^{L_{\rm g}} \in \mathbb{R}^{\frac{HW}{256} \times C}$ represent the outputs of successive blocks in $\mathcal{G}_{\rm E}$, and $L_{\rm g}$ is the number of transformer blocks in $\mathcal{G}_{\rm E}$. In our experiments, we set $\mathcal{G}_{\rm E}$ to 24 following [16]. Next, the sequence of global context features $z_{\rm g}$ are upsampled to high-resolution features by the global decoder $\mathcal{G}_{\rm D}$ for incorporation.

BiMLA Decoder. It is crucial to generate edge-aware pixel-level representations for detecting precise and thin edges. Thus, we expect to design a practical decoder that can encourage the transformer encoder to compute the edge-aware attentions and upsample the attentions in a learnable manner. Inspired by the multi-level feature aggregation in vision tasks [22, 35–37, 65, 72], we propose a novel Bi-directional Multi-Level Aggregation (**BiMLA**) decoder, as illustrated in Fig. 3, to achieve the goal.

In BiMLA, a bi-directional feature aggregation strategy is designed that includes a top-down path and a bottom-up path to boost the information flow in the transformer encoder. More specifically, we first uniformly divide $L_{\rm g}$ transformer blocks into four groups, and take the embedding features $\{z_{\rm g}^6, z_{\rm g}^{12}, z_{\rm g}^{18}, z_{\rm g}^{24}\}$ from the last block of each group as inputs. Then we reshape them to 3D features with the size of $\frac{H}{16} \times \frac{W}{16} \times C$. For the top-down path, we attach the same design (one 1×1 convolutional layer and one 3×3 convolutional layer) to each reshaped feature and obtain four output features $t^6, t^{12}, t^{18}, t^{24}$, following the way of SETR-MLA [72]. Likewise, the bottom-up path starts from the lowest level (*i.e.*, $z_{\rm g}^6$) and gradually approaches the top level (*i.e.*, $z_{\rm g}^{24}$) by attaching one 3×3 convolutional layer on multi-level features, and finally produce another four output features $b^6, b^{12}, b^{18}, b^{24}$. Besides, unlike SETR-MLA [72] that upsamples the features via bilinear operation, our BiMLA passes each aggregated feature through a deconvolutional block, contains two deconvolutional layers with 4×4 kernels and 16×16 kernels, respectively. Each deconvolutional layer is followed by Batch Normalization (BN) and ReLU operations. The eight upsampled features from the bi-directional path are then concatenated into one tensor. Moreover, BiMLA uses an additional stack of convolutional layers to smooth the concatenated features. The stack consists of three 3×3 convolutional layers and one 1×1 convolutional layer with BN and ReLU. The process of BiMLA decoder is formulated as

$$f_{\rm g} = \mathcal{G}_{\rm D}(z_{\rm g}^6, z_{\rm g}^{12}, z_{\rm g}^{18}, z_{\rm g}^{24}), \tag{5}$$

where $f_{\rm g}$ is the pixel-level global features, and $\mathcal{G}_{\rm D}$ represents the global BiMLA decoder. After obtaining the coarse-grained global context features, we will capture the fine-grained local context features in the next stage.

3.4. Stage II: Local Refinement

It is essential to explore fine-grained context features for pixel-level predictions, especially for edge detection. The ideal edge width is one pixel, while 16×16 patches are not conducive to extracting thin edges. Taking pixels as tokens sounds an intuitive remedy, however, it is practically infeasible due to heavy computational cost. Our solution is to use a non-overlapping sliding window to sample the image and then calculate the attentions within the sampled regions. The number of patches in the window is fixed, so the computational complexity is linearly related to the image size.

Thus motivated, we propose to capture the short-range fine-grained context features in Stage II, as shown at the bottom of Fig. 2. In particular, we perform the non-overlapping sliding window with a size of $\frac{H}{2} \times \frac{W}{2}$ on image $X \in \mathbb{R}^{H \times W \times 3}$, and the input image X is decomposed into a sequence $\{X^1, X^2, X^3, X^4\}$. For each window, we split it into fine-grained patches of size 8×8 and compute the attentions by a shared local transformer encoder $\mathcal{R}_{\rm E}$. Then we concatenate the attentions of all windows to obtain $z_{\rm r} = \{z_{\rm r}^1, \dots, z_{\rm r}^{L_{\rm r}}\} \in \mathbb{R}^{\frac{HW}{64} \times C}$. To further economize the computing resource, we set $L_{\rm r} = 12$ that means the local transformer encoder consists of 12 transformer blocks.

Similar to global BiMLA, we evenly select $\{z_\mathrm{r}^3, z_\mathrm{r}^6, z_\mathrm{r}^9, z_\mathrm{r}^{12}\}$ from z_r and input them into the local BiMLA $\mathcal{R}_D$ to generate the local features with high-resolution,

$$f_\mathrm{r} = \mathcal{R}_\mathrm{D}(z_\mathrm{r}^3, z_\mathrm{r}^6, z_\mathrm{r}^9, z_\mathrm{r}^{12}), \tag{6}$$

where f_r indicates the local features. Different from global BiMLA, we replace the 3×3 convolutional layer with the 1×1 convolutional layer in local BiMLA, so as to avoid artificial edges caused by the padding operation.

Feature Fusion Module. Finally, we incorporate the context cues from both levels by a Feature Fusion Module (FFM) and predict the edge maps by a local decision head. FFM takes the global context as the prior knowledge and modulates the local context, which produces the fusion features containing global context and fine-grained local details. As shown in Fig. 2, FFM consists of a spatial feature transform block [60] and two 3×3 convolutional layers followed by BN and ReLU operations. The former is for modulating, and the latter is for smoothing. Then the fusion features are fed into the local decision head $\mathcal{R}_\mathrm{H}$ to predict the edge maps E_r,

$$E_\mathrm{r} = \mathcal{R}_\mathrm{H}\big(\mathrm{FFM}(f_\mathrm{g}, f_\mathrm{r})\big), \tag{7}$$

where $\mathcal{R}_\mathrm{H}$ is the local decision head that consists of a 1×1 convolutional layer and a sigmoid operation.

3.5. Network Training

To train the two-stage framework EDTER, we first optimize Stage I to generate global features that represent the whole image context information. Then, we fix the parameters of Stage I and train Stage II to generate edge maps.

Loss Function. We employ the loss function proposed in [65] for each edge map. Given an edge map E and the corresponding ground truth Y, the loss is calculated as

$$\begin{aligned}\ell\left(E,Y\right) = -\sum_{i,j}\big(&Y_{i,j}\alpha\log(E_{i,j})\\ &+(1-Y_{i,j})(1-\alpha)\log(1-E_{i,j})\big),\end{aligned} \tag{8}$$

where $E_{i,j}$ and $Y_{i,j}$ are the $(i,j)^{th}$ element of matrix E and Y, respectively. Moreover, $\alpha = |Y^-|/(|Y^-|+|Y^+|)$ indicates the percentage of negative pixel samples, where $|\cdot|$ denotes the number of pixels. In practice, the annotations of BSDS500 [1] are labeled by multiple annotators. Inconsistent annotations lead to problematic convergence behavior [65]. Following [36], we first normalize multiple labels to an edge probability map with ranges $[0,1]$, and then use a threshold η to select pixels. The pixel is marked as a positive sample if the probability value is higher than η; otherwise, it is indicated as a negative sample.

Training Stage I. For training Stage I, we first incorporate the global decision head on the global feature maps to generate the edge maps E_g by

$$E_\mathrm{g} = \mathcal{G}_\mathrm{H}(f_\mathrm{g}), \tag{9}$$

where $\mathcal{G}_\mathrm{H}$ indicates the global decision head that consists of a 1×1 convolutional layer and a sigmoid layer. Moreover, we obtain multiple side outputs $S_\mathrm{g}^1, S_\mathrm{g}^2, \ldots, S_\mathrm{g}^8$ by performing the same design (a 4×4 deconvolutional layer and a 16×16 deconvolutional layer) to the intermediate features $t^6, t^{12}, t^{18}, t^{24}$ and $b^6, b^{12}, b^{18}, b^{24}$ extracted by the global BiMLA decoder, which progressively enforce the encoder to emphasize edge-aware attentions.

Stage I is optimized by minimizing the losses between each edge map and the ground truth. The loss function of Stage I is formulated as

$$\mathcal{L}_\mathrm{g} = \mathcal{L}_\mathrm{g}^E + \lambda\mathcal{L}_\mathrm{g}^\mathrm{side} = \ell\left(E_\mathrm{g},Y\right) + \lambda\sum_{k=1}^{8}\ell\left(S_\mathrm{g}^k,Y\right), \tag{10}$$

where $\mathcal{L}_\mathrm{g}^E$ is the loss for E_g, $\mathcal{L}_\mathrm{g}^\mathrm{side}$ denotes side loss, and λ is the weight for balancing $\mathcal{L}_\mathrm{g}^E$ and $\mathcal{L}_\mathrm{g}^\mathrm{side}$. In our experiments, we set λ to 0.4.

Training Stage II. After training Stage I, we fix the parameters of Stage I and move on to Stage II. Similar to the training of Stage I, we perform the same operation (a 4×4 deconvolutional layer and an 8×8 deconvolutional layer) on the intermediate features extracted from the local BiMLA decoder to generate the side outputs $S_\mathrm{r}^1, S_\mathrm{r}^2, \ldots, S_\mathrm{r}^8$. Finally, the loss function of Stage II is defined as

$$\mathcal{L}_\mathrm{r} = \mathcal{L}_\mathrm{r}^E + \lambda\mathcal{L}_\mathrm{r}^\mathrm{side} = \ell\left(E_\mathrm{r},Y\right) + \lambda\sum_{k=1}^{8}\ell\left(S_\mathrm{r}^k,Y\right), \tag{11}$$

where $\mathcal{L}_\mathrm{r}^E$ and $\mathcal{L}_\mathrm{r}^\mathrm{side}$ are the losses for E_r and side outputs, respectively. We again set $\lambda = 0.4$.

4. Experiments

4.1. Datasets

We conduct the experiments on three popular benchmarks: BSDS500 [1], NYUDv2 [49] and Multicue [42].

BSDS500 [1] contains 500 RGB natural images, 200 for training, 100 for validation, and 200 for testing. Each image is manually annotated by five different subjects on average. Our model is trained on the training and validation sets and evaluated on the testing set. Similar to [22, 36, 65], we augment the dataset by rotating each image at 16 different angles and flipping the image at each angle. Moreover, most previous works [22, 36, 37, 62] use PASCAL VOC Context Dataset [17] as the additional training data, which provides full-scene segmentation annotations with

Figure 4. Qualitative comparison of different decoders in EDTER on BSDS500. From left to right are the input images, the results of EDTER using SETR-MLA and BiMLA decoder, respectively.

more than 400 classes, and consists of 10,103 images for training. The outside boundaries extracted from the segmentation annotations are beneficial to infer semantic and context cues in Stage I. Therefore, we first pre-train Stage I on PASCAL VOC Context Dataset [17] and then fine-tune it on BSDS500 [1]. The PASCAL VOC Context Dataset [17] is only used for training Stage I.

NYUDv2 [49] contains 1,449 pairs of aligned RGB and depth images, and it is split into 381 training, 414 validation, and 654 testing images. Following [36, 65], we combine the training and validation sets as the training data, and then augment them by rotating the images and annotations to 4 different angles, randomly flipping, and scaling.

Multicue [42] is composed of 100 challenging natural scenes captured by a binocular stereo camera. Each scene contains a left-view and a right-view short sequences. The last frame of left-view sequences from each scene is labeled with edges and boundaries. Following [22, 36, 65], we randomly select 80 images for training and the remaining 20 images for testing. We repeat the process three times and average the scores of three independent trials as the final results. The data augmentation follows [36, 65].

4.2. Implementation Details

We implement our EDTER using PyTorch [43]. We initialize the transformer blocks of our model using the pre-trained weights by ViT [16]. We set the threshold η as 0.3 to select positive samples for BSDS500 and Multicue Edge, and 0.4 for Multicue Boundary. Each image has only one annotation in NYUDv2, thus no η is needed. We use SGD optimizer with momentum=0.9 and weight decay=2e-4, and adopt a polynomial learning rate decay schedule [71] on all datasets. The initial learning rate is set as 1e-6 for BSDS500, NYUDv2 and Multicue Boundary, and 1e-7 for Multicue Edge. During training, we set the same iteration numbers for both stages. Specially, we train 80k iterations for BSDS500 and Multicue boundary, 40k for NYUDv2, and 4k for Multicue Edge. Each image is randomly cropped to 320×320 in training. Compared with BSDS500, the annotations of NYUDv2 is unitary, and the scale of Multicue is small, which quickly overfits of the model trained on them. Therefore, we set the batch size to 8 for BSDS500, and 4 for NYUDv2 and Multicue. All the

Table 1. Ablation study of the effectiveness of the proposed two-stage strategy, BiMLA decoder and FFM in EDTER on BSDS500. All results are computed with a single scale input.

Stage	Decoder		FFM	ODS	OIS	AP
	SETR-MLA	BiMLA				
I	√	×		0.790	0.806	0.836
	×	√		**0.817**	**0.835**	**0.867**
II	√	×	√	0.799	0.816	0.848
	×	√	√	**0.824**	**0.841**	**0.880**
	×	√	×	0.820	0.835	0.867

experiments are conducted on a V100 GPU. The training of EDTER takes about 26.4 hours (15.1 for Stage I and 11.3 for Stage II). The inference runs at 2.2 *fps* on a V100. During Training, the GPU consumption of Stage I and Stage II are about 15GB and 14GB for 320×320 images respectively. Besides, EDTER brings 332.0G FLOPs in Stage I and 470.25G FLOPs in Stage II.

During evaluation, we record three metrics for all datasets: fixed contour threshold (ODS), per-image best threshold (OIS), and average precision (AP). Moreover, a non-maximum suppression [6] is performed on the predicted edge maps before evaluation. Following previous works [36, 65], the localization tolerance controls the maximum allowed distance in matches between edge results and the ground truth, which is set to 0.0075 for BSDS500 and Multicue, and 0.011 for NYUDv2.

4.3. Ablation Study

Effectiveness of key components in EDTER. We first conduct experiments to verify the impact of key EDTER components: BiMLA and FFM. The quantitative results are summarized in Table 1. First, the performance of ODS, OIS, AP is largely improved (about 2.5%, 3%, 3%) by BiMLA compared with SETR-MLA [72] in both stages. The performance of Stage II significantly surpasses Stage I under either decoder. It illustrates that the two-stage strategy fuses more critical information for edge detection. Besides, we present the predicted edge maps by the SETR-MLA and BiMLA decoders shown in Fig. 4. With the BiMLA decoder, EDTER can accurately detect edges in some local areas (red bounding boxes) and produce less noisy edges. To verify the effectiveness of FFM, we remove the FFM and directly concatenate the feature maps from two stages to construct a variant of EDTER. Without using FFM (row 5), the scores drop by 0.4%, 0.6%, and 1.3% in ODS, OIS, and AP, respectively.

Effectiveness of stages and patch size. We run ablation experiments to verify the effectiveness of the two-stage strategy. The comparative results are presented in Table 2. Compared with Stage I (row 1), we add the second stage and set the patch size to 8×8 (row 2), which obtains the performance gain by 0.7%, 0.6%, 1.3% in ODS, IS, and AP, respectively. Moreover, as visualized in Fig. 5, the predicted edges of Stage II are more clear and crisp in some local de-

Figure 5. Qualitative comparison of different stages in EDTER on BSDS500. From left to right are the input images, the results of EDTER-Stage I and EDTER-Stage II, respectively.

Table 2. Quantitative results of EDTER with training on different patch sizes and stage numbers on BSDS500. "-" means Stage III is not extended. All results are computed with a single scale input.

Patch Size			ODS	OIS	AP
Stage I	Stage II	Stage III			
16×16	-	-	0.817	0.835	0.867
16×16	8×8	-	0.824	0.841	0.880
16×16	4×4	-	0.825	0.843	0.882
16×16	8×8	4×4	0.826	0.843	0.883

tails. It shows that the effectiveness of the two-stage strategy for edge detection. Then, to analyze the impact of the patch size, we create a variant that uses the patch size of 4×4 in Stage II (row 3). Concretely, in Stage II, the image is first decomposed into a sequence $\{X^1, X^2, \ldots, X^{16}\}$ by the sliding window with a size of $\frac{H}{4} \times \frac{W}{4}$, and then we split each window into a sequence of 4×4 patches and generate local cues. Compared with row 2 using 8×8 patches in Stage II, the performance is slightly improved. Moreover, we report experiments with EDTER model variants that use more stages to capture local context, obtained by adding Stage III and setting the patch size to 4×4. In Table 2 (row 4), the scores are marginally increased by fusing context cues of three stages. Since the edge extracted from the networks inevitably occupies multiple pixels, 4×4 patches seldom bring significant gains. Considering the trade-off between computational efficiency and performance, we employ the setting of *16×16 in Stage I and 8×8 in Stage II* to perform all subsequent experiments.

4.4. Comparison with State-of-the-arts

On BSDS500. We compare our model with *traditional detectors* including Canny [6], Felz-Hutt [18], gPb-owt-ucm [2], SCG [64], Sketch Tokens [34], PMI [25], SE [15], OEF [21] and MES [50], and *deep-learning-based detectors* including DeepEdge [3], CSCNN [24], DeepContour [48], HFL [4], HED [65], Deep Boundary [29], CEDN [67], RDS [37], COB [40], DCD [33], AMH-Net [66], RCF [36], CED [62], LPCB [12], BDCN [22], DexiNed [52], DSCD [11] and PiDiNet [53]. The best results of all the methods are taken from their publications.

Quantitative results are shown in Table 3, and Fig. 6 shows Precision-Recall curves of all methods. By training on the trainval set of BSDS500, our method achieves the F-measure ODS of 0.824 with single-scale testing and obtains 0.840 with multi-scale inputs, which already outperforms most edge detectors. With the extra training data and multi-scale testing (following the settings of RCF, CED, BDCN, *etc.*), our method achieves 84.8% (ODS), 86.5% (OIS) 90.3% (AP), which is superior to all the state-of-the-art edge detectors. Some qualitative results are shown in Fig. 7. We observe that the proposed EDTER shows a clear advantage in prediction quality, both crisp and accurate.

Table 3. Results on BSDS500 [1] testing set. The best two results are highlighted in **red** and **blue**, respectively, and same for other tables. MS is the multi-scale testing, and VOC means training with extra PASCAL VOC data.

	Method	Pub.'Year	ODS	OIS	AP
Traditional Method	Canny [6]	PAMI'86	0.600	0.640	0.580
	Felz-Hutt [18]	IJCV'04	0.610	0.640	0.560
	gPb-owt-ucm [2]	PAMI'10	0.726	0.757	0.696
	SCG [64]	NeurIPS'12	0.739	0.758	0.773
	Sketch Tokens [34]	CVPR'13	0.727	0.746	0.780
	PMI [25]	ECCV'14	0.741	0.769	0.799
	SE [15]	PAMI'14	0.746	0.767	0.803
	OEF [21]	CVPR'15	0.746	0.770	0.820
	MES [50]	ICCV'15	0.756	0.776	0.756
CNN-based Method	DeepEdge [3]	CVPR'15	0.753	0.772	0.807
	CSCNN [24]	ArXiv'15	0.756	0.775	0.798
	MSC [51]	PAMI'15	0.756	0.776	0.787
	DeepContour [48]	CVPR'15	0.757	0.776	0.800
	HFL [4]	ICCV'15	0.767	0.788	0.795
	HED [65]	ICCV'15	0.788	0.808	0.840
	Deep Boundary [29]	ICLR'15	0.813	0.831	0.866
	CEDN [67]	CVPR'16	0.788	0.804	-
	RDS [37]	CVPR'16	0.792	0.810	0.818
	COB [40]	ECCV'16	0.793	0.820	0.859
	DCD [33]	ICME'17	0.799	0.817	0.849
	AMH-Net [66]	NeurIPS'17	0.798	0.829	0.869
	RCF [36]	CVPR'17	0.811	0.830	-
	CED [62]	CVPR'17	0.815	0.833	0.889
	LPCB [12]	ECCV'18	0.815	0.834	-
	BDCN [22]	CVPR'19	0.828	0.844	0.890
	DexiNed [52]	WACV'20	0.729	0.745	0.583
	DSCD [11]	ACMMM'20	0.822	**0.859**	-
	PiDiNet [53]	ICCV'21	0.807	0.823	-
Ours	EDTER	-	0.824	0.841	0.880
	EDTER-VOC	-	0.832	0.847	0.886
	EDTER-MS	-	**0.840**	0.858	**0.896**
	EDTER-MS-VOC	-	**0.848**	**0.865**	**0.903**

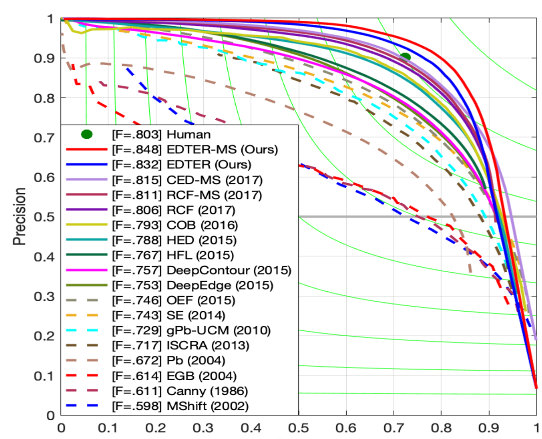
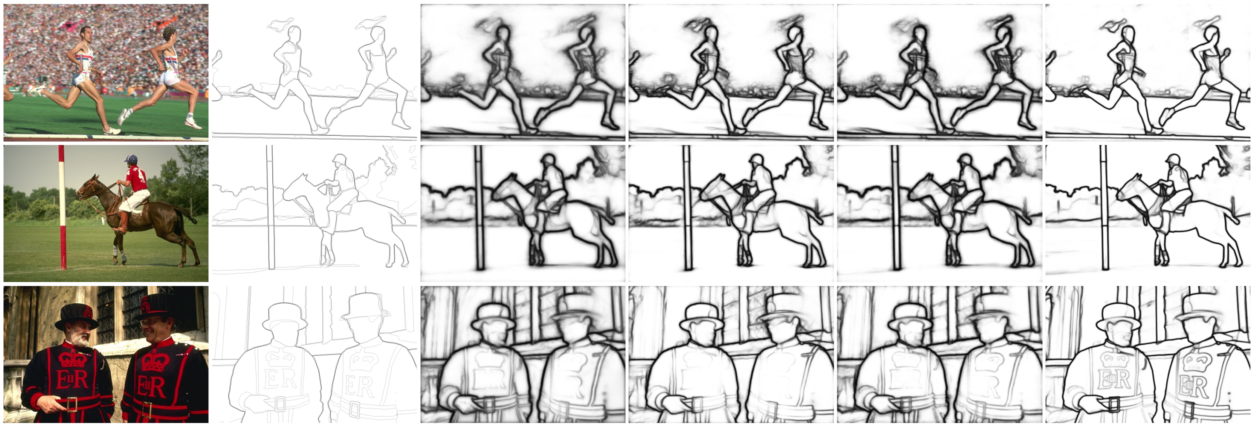


Figure 6. The precision-recall curves on BSDS500.



(a) Input (b) Ground truth (c) RCF [36] (d) CED [62] (e) BDCN [22] (f) EDTER (Ours)

Figure 7. Qualitative comparisons on three challenging samples in the testing set of BSDS500.

Table 4. Quantitative comparisons on NYUDv2 [49]. All results are computed with a single scale input.

	Method	Pub.'Year	ODS	OIS	AP
Traditional	gPb-ucm [1]	PAMI'11	0.632	0.661	0.562
	Silberman *et al.* [49]	ECCV'12	0.658	0.661	-
	gPb+NG [19]	CVPR'13	0.687	0.716	0.629
	SE [15]	PAMI'14	0.695	0.708	0.679
	SE+NG+ [20]	ECCV'14	0.706	0.734	0.738
	OEF [21]	CVPR'15	0.651	0.667	-
	SemiContour [69]	CVPR'16	0.680	0.700	0.690
CNN-based	HED [65]	ICCV'15	0.720	0.734	0.734
	RCF [36]	CVPR'17	0.729	0.742	-
	AMH-Net [66]	NeurIPS'17	0.744	0.758	0.765
	LPCB [12]	ECCV'18	0.739	0.754	-
	BDCN [22]	CVPR'19	**0.748**	**0.763**	**0.770**
	PiDiNet [53]	ICCV'21	0.733	0.747	-
	EDTER (Ours)	-	**0.774**	**0.789**	**0.797**

Table 5. Comparisons on Multicue [42]. All results are computed with a single scale input.

	Method	Pub.'Year	ODS	OIS	AP
Edge	Human [42]	VR'16	.750 (0.024)	-	-
	Multicue [42]	VR'16	.830 (0.002)	-	-
	HED [65]	ICCV'15	.851 (0.014)	.864 (0.011)	-
	RCF [36]	CVPR'17	.857 (0.004)	.862 (0.004)	-
	BDCN [22]	CVPR'19	**.891 (0.001)**	**.898 (0.002)**	**.935(0.002)**
	DSCD [11]	ACMMM'20	.871 (0.007)	.876 (0.002)	-
	PiDiNet [53]	ICCV'21	.855 (0.007)	.860 (0.005)	-
	EDTER (Ours)	-	**.894 (0.005)**	**.900 (0.003)**	**.944 (0.002)**
Boundary	Human [42]	VR'16	.760 (0.017)	-	-
	Multicue [42]	VR'16	.720 (0.014)	-	-
	HED [65]	ICCV'15	.814 (0.011)	.822 (0.008)	.869 (0.015)
	RCF [36]	CVPR'17	.817 (0.004)	.825 (0.005)	-
	BDCN [22]	CVPR'19	**.836 (0.001)**	**.846 (0.003)**	**.893 (0.001)**
	DSCD [11]	ACMMM'20	.828 (0.003)	.835 (0.004)	-
	PiDiNet [53]	ICCV'21	.818 (0.003)	.830 (0.005)	-
	EDTER (Ours)	-	**.861 (0.003)**	**.870 (0.004)**	**.919 (0.003)**

On NYUDv2. In NYUDv2, we conduct experiments on RGB images and compare against the state-of-the-art methods including gPb-ucm [1], Silberman *et al.* [49], gPb+NG [19], SE [15], SE+NG+ [20], OEF [21], SemiContour [69], HED [65], RCF [36], AMH-Net [66], LPCB [12], BDCN [22], and PiDiNet [53]. All results are based on single-scale input. Table 4 shows the quantitative results of our method and other competitors. Our method achieves the best scores of 77.4%, 78.9%, and 79.7% of ODS, OIS, and AP, respectively. Compared to the second best, we increase the scores by 2.6%, 2.6%, 2.7% in three metrics, respectively. More results including HHA, RGB-HHA inputs, and visualizations are reported in the supplementary material.

On Multicue. Multicue consists of two kinds of annotations, *i.e.*, Multicue Edge and Multicue Boundary. For each type of annotations, we compare against the state-of-the-art methods including HED [65], RCF [36], BDCN [22], DSCD [11], and PiDiNet [53]. We report the comparisons in Table 5, and the results show consistent performance. Our method produces competitive results on Multicue Edge. For Multicue Boundary, EDTER achieves the F-measure ODS of 86.1%, higher than all other competitors. The visualization results are provided in the supplementary material.

5. Conclusion and Limitation

In this paper, we propose a novel two-stage edge detection framework, namely EDTER. By introducing the vision transformer, EDTER captures both coarse-grained global context and fine-grained local context in two stages. Moreover, it employs a novel Bi-directional Multi-Level Aggregation (BiMLA) decoder to explore high-resolution representations. Besides, a Feature Fusion Module (FFM) incorporates global and local contexts to predict the edge results. Experimental results illustrate that EDTER yields competitive results in comparison with state-of-the-arts.

Limitation. The width of the edges extracted by EDTER occupies multiple pixels, which still has a gap with the ideal edge width. Without any post-processing, generating clear and thin edges is still a future direction to explore.

Acknowledgements. This work is supported by Beijing Natural Science Foundation (M22022, L211015), Fundamental Research Funds for the Central Universities (2019JBZ104), and National Natural Science Foundation of China (61906013, 62106017).

References

[1] Pablo Arbelaez, Michael Maire, Charless Fowlkes, and Jitendra Malik. Contour detection and hierarchical image segmentation. *IEEE Trans. Pattern Anal. Mach. Intell.*, 33(5):898–916, 2010. 1, 5, 6, 7, 8

[2] Pablo Arbelaez, Michael Maire, Charless Fowlkes, and Jitendra Malik. Contour detection and hierarchical image segmentation. *IEEE Trans. Pattern Anal. Mach. Intell.*, 33(5):898–916, 2010. 7

[3] Gedas Bertasius, Jianbo Shi, and Lorenzo Torresani. Deepedge: A multi-scale bifurcated deep network for top-down contour detection. In *IEEE Conf. Comput. Vis. Pattern Recog.*, pages 4380–4389, 2015. 1, 2, 7

[4] Gedas Bertasius, Jianbo Shi, and Lorenzo Torresani. High-for-low and low-for-high: Efficient boundary detection from deep object features and its applications to high-level vision. In *Int. Conf. Comput. Vis.*, pages 504–512, 2015. 1, 2, 7

[5] Sergi Caelles, Kevis-Kokitsi Maninis, Jordi Pont-Tuset, Laura Leal-Taixé, Daniel Cremers, and Luc Van Gool. One-shot video object segmentation. In *IEEE Conf. Comput. Vis. Pattern Recog.*, pages 221–230, 2017. 1

[6] John F. Canny. A computational approach to edge detection. *IEEE Trans. Pattern Anal. Mach. Intell.*, 8(6):679–698, 1986. 1, 2, 6, 7

[7] Nicolas Carion, Francisco Massa, Gabriel Synnaeve, Nicolas Usunier, Alexander Kirillov, and Sergey Zagoruyko. End-to-end object detection with transformers. In *Eur. Conf. Comput. Vis.*, pages 213–229. Springer, 2020. 2

[8] Liang-Chieh Chen, George Papandreou, Iasonas Kokkinos, Kevin Murphy, and Alan L Yuille. Deeplab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected crfs. *IEEE Trans. Pattern Anal. Mach. Intell.*, 40(4):834–848, 2017. 1, 2

[9] Xin Chen, Bin Yan, Jiawen Zhu, Dong Wang, Xiaoyun Yang, and Huchuan Lu. Transformer tracking. In *IEEE Conf. Comput. Vis. Pattern Recog.*, pages 8126–8135, 2021. 1

[10] Zhigang Dai, Bolun Cai, Yugeng Lin, and Junying Chen. Up-detr: Unsupervised pre-training for object detection with transformers. In *IEEE Conf. Comput. Vis. Pattern Recog.*, pages 1601–1610, 2021. 2

[11] Ruoxi Deng and Shengjun Liu. Deep structural contour detection. In *ACM Int. Conf. Multimedia*, pages 304–312, 2020. 2, 7, 8

[12] Ruoxi Deng, Chunhua Shen, Shengjun Liu, Huibing Wang, and Xinru Liu. Learning to predict crisp boundaries. In *Eur. Conf. Comput. Vis.*, pages 562–578, 2018. 2, 7, 8

[13] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. *NAACL*, 2019. 2

[14] Piotr Dollár, Zhuowen Tu, and Serge J. Belongie. Supervised learning of edges and object boundaries. In *IEEE Conf. Comput. Vis. Pattern Recog.*, volume 2, pages 1964–1971, 2006. 1, 2

[15] Piotr Dollár and C Lawrence Zitnick. Fast edge detection using structured forests. *IEEE Trans. Pattern Anal. Mach. Intell.*, 37(8):1558–1570, 2014. 7, 8

[16] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, et al. An image is worth 16x16 words: Transformers for image recognition at scale. *Int. Conf. Learn. Represent.*, 2020. 1, 2, 3, 4, 6

[17] Mark Everingham, Luc Van Gool, Christopher KI Williams, John Winn, and Andrew Zisserman. The pascal visual object classes (voc) challenge. *Int. J. Comput. Vis.*, 88(2):303–338, 2010. 5, 6

[18] Pedro F Felzenszwalb and Daniel P Huttenlocher. Efficient graph-based image segmentation. *Int. J. Comput. Vis.*, 59(2):167–181, 2004. 7

[19] Saurabh Gupta, Pablo Arbelaez, and Jitendra Malik. Perceptual organization and recognition of indoor scenes from rgb-d images. In *IEEE Conf. Comput. Vis. Pattern Recog.*, pages 564–571, 2013. 8

[20] Saurabh Gupta, Ross Girshick, Pablo Arbeláez, and Jitendra Malik. Learning rich features from rgb-d images for object detection and segmentation. In *Eur. Conf. Comput. Vis.*, pages 345–360. Springer, 2014. 8

[21] Sam Hallman and Charless C Fowlkes. Oriented edge forests for boundary detection. In *IEEE Conf. Comput. Vis. Pattern Recog.*, pages 1732–1740, 2015. 7, 8

[22] Jianzhong He, Shiliang Zhang, Ming Yang, Yanhu Shan, and Tiejun Huang. Bi-directional cascade network for perceptual edge detection. In *IEEE Conf. Comput. Vis. Pattern Recog.*, pages 3828–3837, 2019. 1, 2, 4, 5, 6, 7, 8

[23] Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross B. Girshick. Mask r-cnn. *Int. Conf. Comput. Vis.*, pages 2980–2988, 2017. 1

[24] Jyh-Jing Hwang and Tyng-Luh Liu. Pixel-wise deep learning for contour detection. *arXiv preprint arXiv:1504.01989*, 2015. 7

[25] Phillip Isola, Daniel Zoran, Dilip Krishnan, and Edward H Adelson. Crisp boundary detection using pointwise mutual information. In *Eur. Conf. Comput. Vis.*, pages 799–814. Springer, 2014. 7

[26] André Peter Kelm, Vijesh Soorya Rao, and Udo Zölzer. Object contour and edge detection with refinecontournet. In *International Conference on Computer Analysis of Images and Patterns*, pages 246–258. Springer, 2019. 2

[27] Bumsoo Kim, Junhyun Lee, Jaewoo Kang, Eun-Sol Kim, and Hyunwoo J. Kim. Hotr: End-to-end human-object interaction detection with transformers. In *IEEE Conf. Comput. Vis. Pattern Recog.*, pages 74–83, 2021. 2

[28] Josef Kittler. On the accuracy of the sobel edge detector. *Image Vis. Comput.*, 1(1):37–42, 1983. 1, 2

[29] Iasonas Kokkinos. Pushing the boundaries of boundary detection using deep learning. *Int. Conf. Learn. Represent.*, 2016. 1, 2, 7

[30] Zhenzhong Lan, Mingda Chen, Sebastian Goodman, Kevin Gimpel, Piyush Sharma, and Radu Soricut. Albert: A lite bert for self-supervised learning of language representations. *Int. Conf. Learn. Represent.*, 2020. 2

[31] Ke Li, Shijie Wang, Xiang Zhang, Yifan Xu, Weijian Xu, and Zhuowen Tu. Pose recognition with cascade transformers. In

IEEE Conf. Comput. Vis. Pattern Recog., pages 1944–1953, 2021. 2

[32] Yulin Li, Jianfeng He, Tianzhu Zhang, Xiang Liu, Yongdong Zhang, and Feng Wu. Diverse part discovery: Occluded person re-identification with part-aware transformer. In *IEEE Conf. Comput. Vis. Pattern Recog.*, pages 2898–2907, 2021. 2

[33] Yuan Liao, Songping Fu, Xiaoqing Lu, Chengcui Zhang, and Zhi Tang. Deep-learning-based object-level contour detection with ccg and crf optimization. In *Int. Conf. Multimedia and Expo*, pages 859–864. IEEE, 2017. 7

[34] Joseph J. Lim, C. Lawrence Zitnick, and Piotr Dollár. Sketch tokens: A learned mid-level representation for contour and object detection. In *IEEE Conf. Comput. Vis. Pattern Recog.*, pages 3158–3165, 2013. 1, 2, 7

[35] Tsung-Yi Lin, Piotr Dollár, Ross B. Girshick, Kaiming He, Bharath Hariharan, and Serge J. Belongie. Feature pyramid networks for object detection. In *IEEE Conf. Comput. Vis. Pattern Recog.*, pages 936–944, 2017. 2, 4

[36] Yun Liu, Ming-Ming Cheng, Xiaowei Hu, Kai Wang, and Xiang Bai. Richer convolutional features for edge detection. In *IEEE Conf. Comput. Vis. Pattern Recog.*, pages 3000–3009, 2017. 1, 2, 4, 5, 6, 7, 8

[37] Yu Liu and Michael S Lew. Learning relaxed deep supervision for better edge detection. In *IEEE Conf. Comput. Vis. Pattern Recog.*, pages 231–240, 2016. 1, 2, 4, 5, 7

[38] Ze Liu, Yutong Lin, Yue Cao, Han Hu, Yixuan Wei, Zheng Zhang, Stephen Lin, and Baining Guo. Swin transformer: Hierarchical vision transformer using shifted windows. In *Int. Conf. Comput. Vis.*, 2021. 2

[39] Jonathan Long, Evan Shelhamer, and Trevor Darrell. Fully convolutional networks for semantic segmentation. In *IEEE Conf. Comput. Vis. Pattern Recog.*, pages 3431–3440, 2015. 1

[40] Kevis-Kokitsi Maninis, Jordi Pont-Tuset, Pablo Arbeláez, and Luc Van Gool. Convolutional oriented boundaries. In *Eur. Conf. Comput. Vis.*, pages 580–596. Springer, 2016. 2, 7

[41] David R. Martin, Charless C. Fowlkes, and Jitendra Malik. Learning to detect natural image boundaries using local brightness, color, and texture cues. *IEEE Trans. Pattern Anal. Mach. Intell.*, 26(5):530–549, 2004. 1, 2

[42] David A Mély, Junkyung Kim, Mason McGill, Yuliang Guo, and Thomas Serre. A systematic comparison between visual cues for boundary detection. *Vis. Res.*, 120:93–107, 2016. 5, 6, 8

[43] Adam Paszke, Sam Gross, Soumith Chintala, Gregory Chanan, Edward Yang, Zachary DeVito, Zeming Lin, Alban Desmaison, Luca Antiga, and Adam Lerer. Automatic differentiation in pytorch. In *Adv. Neural Inform. Process. Syst.*, 2017. 6

[44] Pedro OO Pinheiro, Ronan Collobert, and Piotr Dollár. Learning to segment object candidates. In *Adv. Neural Inform. Process. Syst.*, pages 1990–1998, 2015. 1

[45] Pedro O Pinheiro, Tsung-Yi Lin, Ronan Collobert, and Piotr Dollár. Learning to refine object segments. In *Eur. Conf. Comput. Vis.*, pages 75–91, 2016. 1

[46] Mengyang Pu, Yaping Huang, Qingji Guan, and Haibin Ling. Rindnet: Edge detection for discontinuity in reflectance, illumination, normal and depth. In *Int. Conf. Comput. Vis.*, pages 6879–6888, October 2021. 2

[47] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. In *International Conference on Medical Image Computing and Computer-Assisted Intervention (MICCAI)*, pages 234–241, 2015. 1

[48] Wei Shen, Xinggang Wang, Yan Wang, Xiang Bai, and Zhijiang Zhang. Deepcontour: A deep convolutional feature learned by positive-sharing loss for contour detection. In *IEEE Conf. Comput. Vis. Pattern Recog.*, pages 3982–3991, 2015. 1, 2, 7

[49] Nathan Silberman, Derek Hoiem, Pushmeet Kohli, and Rob Fergus. Indoor segmentation and support inference from rgbd images. In *Eur. Conf. Comput. Vis.*, pages 746–760, 2012. 5, 6, 8

[50] Amos Sironi, Vincent Lepetit, and Pascal Fua. Projection onto the manifold of elongated structures for accurate extraction. In *Int. Conf. Comput. Vis.*, pages 316–324, 2015. 7

[51] Amos Sironi, Engin Türetken, Vincent Lepetit, and Pascal Fua. Multiscale centerline detection. *IEEE Trans. Pattern Anal. Mach. Intell.*, 38(7):1327–1341, 2015. 7

[52] Xavier Soria, Edgar Riba, and Ángel D. Sappa. Dense extreme inception network: Towards a robust cnn model for edge detection. In *IEEE Winter Conf. Appl. Comput. Vis.*, pages 1923–1932, 2020. 2, 7

[53] Zhuo Su, Wenzhe Liu, Zitong Yu, Dewen Hu, Qing Liao, Qi Tian, Matti Pietikainen, and Li Liu. Pixel difference networks for efficient edge detection. In *Int. Conf. Comput. Vis.*, pages 5117–5127, October 2021. 2, 7, 8

[54] Andrew Tao, Karan Sapra, and Bryan Catanzaro. Hierarchical multi-scale attention for semantic segmentation. *arXiv preprint arXiv:2005.10821*, 2020. 2

[55] Ashish Vaswani, Prajit Ramachandran, Aravind Srinivas, Niki Parmar, Blake Hechtman, and Jonathon Shlens. Scaling local self-attention for parameter efficient visual backbones. In *IEEE Conf. Comput. Vis. Pattern Recog.*, pages 12894–12904, 2021. 2

[56] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Adv. Neural Inform. Process. Syst.*, pages 5998–6008, 2017. 2, 3

[57] Paul Voigtlaender, Yuning Chai, Florian Schroff, Hartwig Adam, Bastian Leibe, and Liang-Chieh Chen. Feelvos: Fast end-to-end embedding learning for video object segmentation. In *IEEE Conf. Comput. Vis. Pattern Recog.*, pages 9481–9490, 2019. 1

[58] Ning Wang, Wengang Zhou, Jie Wang, and Houqiang Li. Transformer meets tracker: Exploiting temporal context for robust visual tracking. In *IEEE Conf. Comput. Vis. Pattern Recog.*, pages 1571–1580, 2021. 2

[59] Qiang Wang, Li Zhang, Luca Bertinetto, Weiming Hu, and Philip HS Torr. Fast online object tracking and segmentation: A unifying approach. In *IEEE Conf. Comput. Vis. Pattern Recog.*, pages 1328–1338, 2019. 1

[60] Xintao Wang, Ke Yu, Chao Dong, and Chen Change Loy. Recovering realistic texture in image super-resolution by deep spatial feature transform. In *IEEE Conf. Comput. Vis. Pattern Recog.*, pages 606–615, 2018. 5

[61] Yuqing Wang, Zhaoliang Xu, Xinlong Wang, Chunhua Shen, Baoshan Cheng, Hao Shen, and Huaxia Xia. End-to-end video instance segmentation with transformers. In *IEEE Conf. Comput. Vis. Pattern Recog.*, pages 8741–8750, 2021. 1

[62] Yupei Wang, Xin Zhao, and Kaiqi Huang. Deep crisp boundaries. In *IEEE Conf. Comput. Vis. Pattern Recog.*, pages 3892–3900, 2017. 5, 7, 8

[63] Holger Winnemöller, Jan Eric Kyprianidis, and Sven C. Olsen. Xdog: An extended difference-of-gaussians compendium including advanced image stylization. *Comput. Graph.*, 36(6):740–753, 2012. 1, 2

[64] Ren Xiaofeng and Liefeng Bo. Discriminatively trained sparse code gradients for contour detection. *Adv. Neural Inform. Process. Syst.*, 25, 2012. 7

[65] Saining Xie and Zhuowen Tu. Holistically-nested edge detection. In *Int. Conf. Comput. Vis.*, pages 1395–1403, 2015. 1, 2, 4, 5, 6, 7, 8

[66] Dan Xu, Wanli Ouyang, Xavier Alameda-Pineda, Elisa Ricci, Xiaogang Wang, and Nicu Sebe. Learning deep structured multi-scale features using attention-gated crfs for contour prediction. In *Adv. Neural Inform. Process. Syst.*, pages 3961–3970, 2017. 1, 2, 7, 8

[67] Jimei Yang, Brian Price, Scott Cohen, Honglak Lee, and Ming-Hsuan Yang. Object contour detection with a fully convolutional encoder-decoder network. In *IEEE Conf. Comput. Vis. Pattern Recog.*, pages 193–202, 2016. 7

[68] Pengchuan Zhang, Xiyang Dai, Jianwei Yang, Bin Xiao, Lu Yuan, Lei Zhang, and Jianfeng Gao. Multi-scale vision longformer: A new vision transformer for high-resolution image encoding. *arXiv preprint arXiv:2103.15358*, 2021. 2

[69] Zizhao Zhang, Fuyong Xing, Xiaoshuang Shi, and Lin Yang. Semicontour: A semi-supervised learning approach for contour detection. In *IEEE Conf. Comput. Vis. Pattern Recog.*, pages 251–259, 2016. 8

[70] Hengshuang Zhao, Jianping Shi, Xiaojuan Qi, Xiaogang Wang, and Jiaya Jia. Pyramid scene parsing network. In *IEEE Conf. Comput. Vis. Pattern Recog.*, pages 6230–6239, 2017. 2

[71] Hengshuang Zhao, Jianping Shi, Xiaojuan Qi, Xiaogang Wang, and Jiaya Jia. Pyramid scene parsing network. In *IEEE Conf. Comput. Vis. Pattern Recog.*, pages 6230–6239, 2017. 6

[72] Sixiao Zheng, Jiachen Lu, Hengshuang Zhao, Xiatian Zhu, Zekun Luo, Yabiao Wang, Yanwei Fu, Jianfeng Feng, Tao Xiang, Philip HS Torr, et al. Rethinking semantic segmentation from a sequence-to-sequence perspective with transformers. In *IEEE Conf. Comput. Vis. Pattern Recog.*, pages 6881–6890, 2021. 1, 2, 3, 4, 6

[73] Cheng Zou, Bohan Wang, Yue Hu, Junqi Liu, Qian Wu, Yu Zhao, Boxun Li, Chenguang Zhang, Chi Zhang, Yichen Wei, and Jian Sun. End-to-end human object interaction detection with hoi transformer. In *IEEE Conf. Comput. Vis. Pattern Recog.*, pages 11825–11834, 2021. 2