KnifeCut: Refining Thin Part Segmentation with Cutting Lines

Zheng Lin*
TMCC, College of Computer Science, Nankai University
Tianjin, China
frazer.linzheng@mail.nankai.edu.cn

Zheng-Peng Duan*
TMCC, College of Computer Science, Nankai University
Tianjin, China
adamduan0211@gmail.com

Zhao Zhang
SenseTime Research
Shanghai, China
zzhang@mail.nankai.edu.cn

Chun-Le Guo†
TMCC, College of Computer Science, Nankai University
Tianjin, China
guochunle@nankai.edu.cn

Ming-Ming Cheng
TMCC, College of Computer Science, Nankai University
Tianjin, China
cmm@nankai.edu.cn

ABSTRACT

Objects with thin structures remain challenging for current image segmentation techniques. Their outputs often do well in the main body but with thin parts unsatisfactory. In practical use, they inevitably need post-processing. However, repairing them is time-consuming and laborious, either in professional editing applications (*e.g.,* PhotoShop) or by current interactive image segmentation methods (*e.g.,* by click, scribble, and polygon). To refine the thin parts for unsatisfactory pre-segmentation, we propose an efficient interaction mode, where users only need to draw a line across the mislabeled thin part like cutting with a knife. This low-stress and intuitive action does not require the user to aim deliberately, and is friendly when using the mouse, touchpad, and mobile devices. Additionally, the line segment provides a contrasting prior because it passes through both the foreground and background regions and there must be thin part pixels on it. Based on the interaction idea, we propose KnifeCut, which offers the users two results, where one only focuses on the target thin part and the other provides the refinements for all thin parts that share similar features with the target one. To our best knowledge, KnifeCut is the first method to solve interactive thin structure refinement pertinently. Extensive experiments and visualized results further demonstrate its friendliness, convenience, and effectiveness. The project page is available on `http://mmcheng.net/knifecut/`.

CCS CONCEPTS

• **Human-centered computing** → **Human computer interaction (HCI)**; **Interaction design process and methods**; • **Computing methodologies** → **Artificial intelligence**.

KEYWORDS

interactive segmentation; thin object; user interaction

ACM Reference Format:
Zheng Lin, Zheng-Peng Duan, Zhao Zhang, Chun-Le Guo, and Ming-Ming Cheng. 2022. KnifeCut: Refining Thin Part Segmentation with Cutting Lines. In *Proceedings of the 30th ACM International Conference on Multimedia (MM '22), October 10–14, 2022, Lisboa, Portugal.* ACM, New York, NY, USA, 9 pages. https://doi.org/10.1145/3503161.3547803

1 INTRODUCTION

In the real world, many kinds of objects possess thin structures, such as railing, nets, and twigs. Current image segmentation methods often fail when processing these thin parts. In most cases, their output segmentation does well in the main body but achieves unsatisfactory results for the thin parts. Therefore, when requiring high-precision results, manual guided post-processing is inevitably introduced. People usually adopt professional image processing tools (*e.g.,* PhotoShop) or popular interactive segmentation technology [5, 24, 28, 38] to refine thin parts.

However, current interaction modes, *e.g.,* brush, click, and polygon, cannot handle thin parts efficiently. As shown in the top row of Fig. 1, marking the boundary by the polygon or painting the whole leg with the brush can be time-consuming and laborious. Click-based methods reduce the annotation complexity to some extent, but putting the positive points on the leg and negative points next to the leg is also a burden.

In this paper, to efficiently refine thin structures, we introduce a fast and low-pressure interaction mode, by which the user only needs to draw an intersecting line through the mislabeled pixels like cutting with a knife. As shown in Fig. 1, the action of cutting on the mislabeled leg is intuitive and quick. Fig. 2 shows more specific examples. The proposed cutting action can be done in an instant whether it is on a miss-segmented umbrella and racket, or over-segmented claws and net; while processing these cases is unimaginable for the previous interactions that need to carefully aim at the foreground and boundaries. Additionally, the cutting line provides a contrasting prior because there are both foreground and background contents on it and there must be thin part pixels on it.

Based on the proposed interaction idea, we further design a framework named KnifeCut. As shown in Fig. 1, KnifeCut aims to obtain local refinement containing the target ant's leg and the

*Both authors contributed equally to this research.
†Chun-Le Guo is the corresponding author.

MM '22, October 10–14, 2022, Lisboa, Portugal

ACM ISBN 978-1-4503-9203-7/22/10…$15.00
https://doi.org/10.1145/3503161.3547803

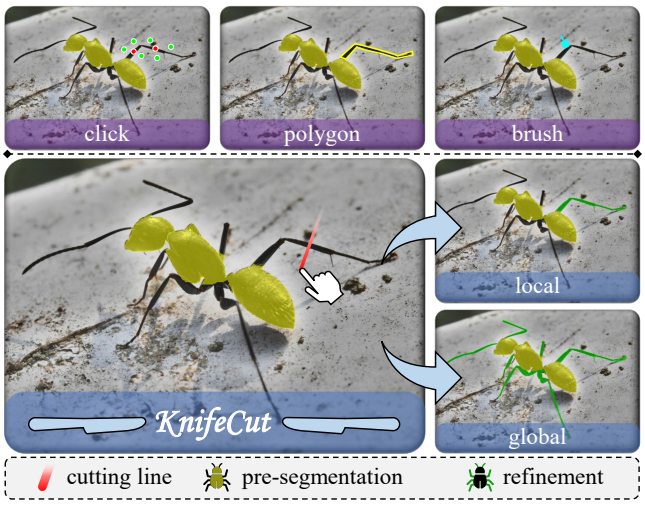

Figure 1: Illustration of KnifeCut and comparison with other refinement interactions. Top: Other refinement interactions, *e.g.*, based on click, polygon, and brush, are inefficient when dealing with thin parts. Bottom: With KnifeCut, the user can draw a line through the thin part for refinement like cutting off it by a knife. KnifeCut offers the user two results: one is the target thin part, and the other is all relevant ones.

global refinement containing all legs; meanwhile, maintains a well-segmented body without perceptible change. Specifically, our KnifeCut first predicts the region of thin part adjacent to the cutting line, and then utilizes its corresponding features to activate all relevant thin parts by similarity proxy. With cutting line priors and activated similarity maps, KnifeCut estimates local and global thin part scopes. It utilizes these estimated maps to further predict local and global thin part refinements in two branches, while avoiding the impact on the body. In addition, the user can select either of the two refinement results, where the local branch only focuses on the target thin part, and the global one refines all relevant thin parts.

The contributions can be summarized as follows:

- We propose an efficient interaction mode for thin part segmentation refinement, which just requires drawing one line through the poor-segmented regions like cutting.
- Based on the interactive idea, we propose KnifeCut, which provides both refinements from the local and global view according to the interaction.
- To our best knowledge, KnifeCut is the first method designed for interactive thin structure refinement. Extensive experiments further demonstrate its convenience and effectiveness.

2 RELATED WORK

2.1 Interactive Common Object Segmentation

Most interactive segmentation methods view user interaction, such as points and scribbles, as constraints of foreground and background. In the early years, traditional methods combine the constraints with low-level features of the image to predict the background/foreground probability. Rother *et al.* [37] used Gaussian mixture models to construct the GraphCut [5] model and solved it by the min-cut/max-flow algorithm [4]. Li *et al.* [21] made instant feedback possible by combining graph cut with pre-computed over-segmentation. Grady *et al.* [11] assigned each pixel to the user-defined label most likely to be reached, which is improved by introducing in a restart simulation in [18]. However, considering only low-level features and ignoring high-level information make these methods ineffective in complex environments.

Although there are still some works [17, 41, 42] to further improve traditional methods, deep-learning-based methods fully exploit the high-level features to utilize the constraints. Xu *et al.* [43] proposed the first deep-learning-based method and several random point sampling strategies which become standard protocol in this field. Some works focus on the design of network architecture. Zhao *et al.* [47] used the interactive scribble to train models for both region and boundary domains and made convex inferences to get the final results. Hu *et al.* [13] proposed a two-stream fusion network to decouple images and interaction. As for guidance maps, Majumder *et al.* [31] made use of user clicks to generate content-aware guidance maps. Lin *et al.* [27] emphasized the critical role of the first click and took it as special guidance. In order to force every interacted pixel to be segmented correctly, Jang *et al.* [15] proposed BRS, which is improved by f-BRS [38]. Kontogianni *et al.* [19] made continuous adaptation to the model parameters by learning from user correction. Lin *et al.* [29] further used more information in the annotating process. Since user interaction has ambiguity inevitably, some research [22, 25] has been carried out to tackle this problem. To fully exploit user interaction and better combine local and global features, Liew *et al.* [23] developed RIS-Net to capture the information around each click pair for further refinement. Chen *et al.* [8] introduced in non-local method to propagate information from clicked points to target regions properly. Lin *et al.* [26] introduced the focus view to exploit each click's surrounding information.

While most research is conducted with annotation for background and foreground, multiple interaction modes have also been explored in this task. Mortensen *et al.* [35] proposed intelligent scissors to extract objects using simple gesture motions with a mouse. Castrejon *et al.* [6] took an image crop as input and produced vertices of the polygon outlining the object in a recurrent manner, which is improved by Polygon-RNN++ [1]. With the same input method, Ling *et al.* [30] used a graph convolutional network to predict all vertices simultaneously. Jain *et al.* [14] and Le *et al.* [20] adopted boundary clicks as interaction. There are also some works combining different interaction modes to achieve better effects, such as the bounding box and clicks [3, 45]. The extreme points have been proved effective for common objects [32], objects with thin structures [24], and the full image [2].

2.2 Interactive Thin Object Segmentation

Interactive segmentation for the thin objects has already been studied for a long time. Vicente *et al.* [39] imposed additional connectivity prior to graph cut [5] to tackle this task. Jegelka *et al.* [16] introduced a discount for homogeneous boundaries in segmentation for images with thin objects. Mansilla *et al.* [33, 34] proposed COIFT which incorporates the connectivity constraint in OIFT to improve the segmentation of thin objects. Dong *et al.* [10] designed

a new sub-Markov random walk algorithm with a label prior to solve this problem. Liew *et al.* [24] proposed TOS-Net using extreme points as interaction, along with a large-scale dataset for this task named ThinObject-5K. However, this method is incapable of handling the refinement jobs for thin objects in practical use. In order to solve the problem and based on our interactive idea, we propose our KnifeCut, which provides the refinement for thin parts on the pre-segmentation mask with little impact on the body part.

3 PROPOSED INTERACTION

3.1 Interaction Introduction

For a better explanation for the following parts, we define some symbols here. Let $\mathbf{G}$ represent the ground truth of image $\mathbf{I}$. By using the same strategy in [24], we can extract the ground truth of thin parts $\mathbf{G}_{\text{thin}}$ from $\mathbf{G}$. Thus, the ground truth of the non-thin part $\mathbf{G}_{\text{non-thin}}$ can be obtained by $\mathbf{G}_{\text{non-thin}} = \mathbf{G} - \mathbf{G}_{\text{thin}}$.

We denote the pre-segmentation obtained by segmentation methods as $\mathbf{P}'$. Compared with $\mathbf{G}$, $\mathbf{P}'$ usually achieves overall good performance with high IoU scores. However, these methods often fail to process the object with thin parts, which is hard to be reflected by IoU score because thin pixels are too few. As Fig. 2 shows, $\mathbf{P}'$ mainly encounters the following two situations regarding thin structures: 1) **Miss segmentation**: the details, even the whole region, of the thin part are missing, *e.g.*, the beach umbrella and tennis racket in Fig. 2 (a-b). 2) **Over segmentation**: the thin parts stay too dense to be separated, with background in the gaps included, *e.g.*, the claws and the net in Fig. 2 (c-d).

In order to overcome this problem, we provide an efficient refinement tool designed specifically for thin parts. Compared with $\mathbf{G}_{\text{non-thin}}$, the $\mathbf{P}'$ has already done well; thus we only focus on the refinement of the thin region with little change to the body segmentation. Considering the above-mentioned complicated situation and inspired by human behavior, we propose an efficient interaction mode to serve for refinement. Just as people tend to cut off thin objects with a knife, the users only need to draw an intersecting line through the mislabeled region of the thin part. Fig. 2 also shows the practical examples handled by our method. Whether it is miss-segmented or over-segmented, the interaction can be done without consideration and effort.

Besides its convenience and efficiency, the cutting line also contains information that other interaction methods cannot provide. Because there are both background and foreground contents on it, the cutting line contains a contrasting prior. Further experiments in Sec. 5.2 will demonstrate the superiority of our method over the click-based one when processing thin parts.

3.2 Interaction Simulation

As we all know, even when facing the same image, different users tend to make different interactions. To make our KnifeCut better adapt to this, we need various interactions for training. Since collecting them from real users is too expensive and impractical, we thus design a simulation algorithm to mimic the user and generate the pairs automatically. In order to avoid overfitting during training, the simulation algorithm has certain randomness. However, during testing, we need to make sure of the fairness of each test and cater to users' operating habits. Thus, we make a little change to the

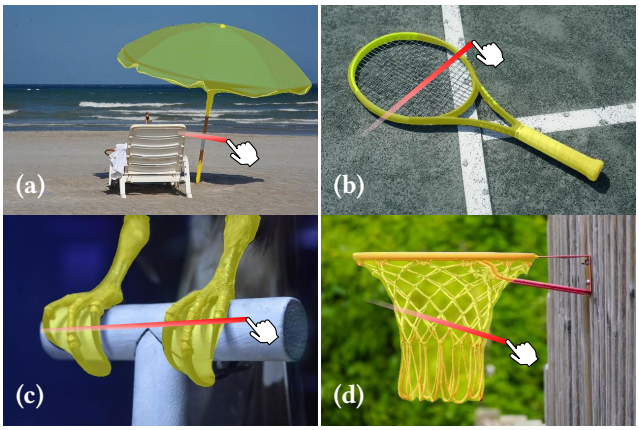

Figure 2: Various thin situations handled by the proposed KnifeCut. (a) The beach umbrella has missed a part along the handle. (b) The net of the tennis racket has not been segmented out. (c) The claws are not segmented finely. (d) The net is segmented including the background in the gaps.

simulation algorithm under the assumption that the user cuts the largest error region approximately perpendicular to the skeleton. The details of the simulation algorithm are as follows.

Step 1: Among all thin parts in an image, users tend to firstly refine the worst pre-segmented one, thus we need to evaluate their current qualities. For simplicity, we assume that each component on $\mathbf{G}_{\text{thin}}$ can be viewed as one thin part, which is denoted as $\mathbf{T}$. Following [24], we first set a threshold τ, which is defined as

$$\tau = 10 \times \frac{\max(H_{\text{box}}, W_{\text{box}})}{300}, \tag{1}$$

where the H_{box} and W_{box} denote the height and width of the bounding box enclosing the object. We use $d(\mathbf{p}_1, \mathbf{p}_2)$ to represent the Euclidean distance between points $\mathbf{p}_1$ and $\mathbf{p}_2$. Given a ground truth $\mathbf{G} \in \{0,1\}^{H \times W}$, where H and W denote height and width, we use the function $\phi(\mathbf{p}, \mathbf{G})$ to calculate the distance transform as follows,

$$\phi(\mathbf{p}, \mathbf{G}) = \min_{\forall \{\mathbf{q} | \mathbf{G}_\mathbf{q} = 1\}} d(\mathbf{p}, \mathbf{q}), \tag{2}$$

where $\mathbf{G}_\mathbf{q}$ refers to the value of $\mathbf{G}$ at pixel location $\mathbf{q}$. So the expanded region $\mathbf{E}$, which is also used as the evaluation mask, for each thin part can be formulated by

$$\mathbf{E}^i = \{\mathbf{p} : (\phi(\mathbf{p}, \mathbf{T}^i) \le \tau) \wedge ((\mathbf{G}_{\text{non-thin}})_\mathbf{p} = 0)\}, \tag{3}$$

where $\mathbf{T}^i$ and $\mathbf{E}^i$ mean the i-th thin part and the corresponding expanded region. As shown in Fig. 3 (2), the expanded region $\mathbf{E}$ is indicated by a white borderline.

Step 2: We calculate the number $\mathbf{N}^i$ of wrong pixels for $\mathbf{T}^i$:

$$\mathbf{N}^i = \sum (|\mathbf{P}' - \mathbf{G}| \times \mathbf{E}^i), \tag{4}$$

where $\times$ means the element-wise multiplication. For *training*, we will randomly select the thin part for interaction, where the random probability is proportional to $\mathbf{N}^i$. However, during *testing*, the thin part with the most error pixels will be selected for later steps. Fig. 3 (2-3) show an example of this process.

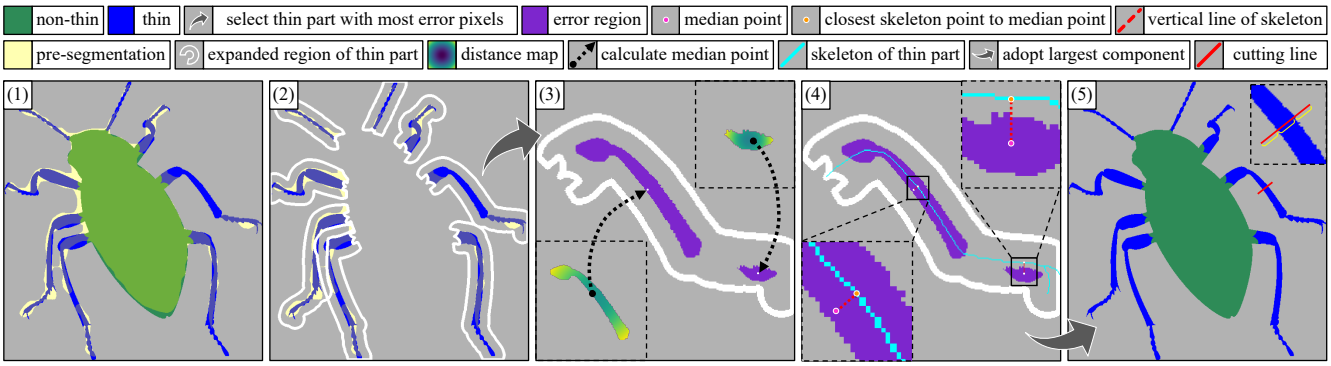

Figure 3: Visualized steps of our simulation algorithm during testing. (1) The pre-segmentation, non-thin region, and thin region are presented. (2) The expanded region of the thin parts is indicated by a white borderline. After counting the error pixels in each expanded region, the thin part with most error pixels will be selected. (3) Calculate the distance map where each pixel value corresponds to the distance sum to other points, then find the median point in the error region. (4) First extract the skeleton of the thin part, and then use the closest skeleton point to the median point as the anchor point. Based on the anchor point, make a vertical line of the skeleton. (5) Adopt the largest component of the error region to make the cutting line. Both ends will be extended by the same length on the thin part. The final effect is shown in the figure.

Step 3: For the selected thin part, we want the line to pass through the error region. Thus, we first calculate the error region, which is indicated in purple in Fig. 3. For *training*, we will randomly sample a point $\mathbf{m}$ in the error area. But during *testing*, we will select the largest component of the error area, which is denoted as $\mathbf{C}$. The point $\mathbf{m}$ will be located at the median point of $\mathbf{C}$, which has the minimum distance sum to other points. This can be formulated as

$$\mathbf{m} = \underset{\forall\{\mathbf{p}|\mathbf{C}_\mathbf{p}=1\}}{\arg\min} \sum_{\mathbf{C}_\mathbf{q}=1} d(\mathbf{p},\mathbf{q}). \tag{5}$$

Fig. 3 (3) shows two examples of distance maps where each pixel value corresponds to the distance sum to other points.

Step 4: In order to make the line vertical to the thin part, we first extract the skeleton of the selected thin part and select an anchor point $\mathbf{a}$ on it. The skeleton, which is denoted as $\mathbf{K}$, is extracted by the algorithm proposed in [46]. Whether in *training* or *testing*, The anchor point $\mathbf{a}$ will be selected from the skeleton $\mathbf{K}$ with the shortest distance from the point $\mathbf{m}$, which can be formulated as

$$\mathbf{a} = \underset{\forall\{\mathbf{p}|\mathbf{K}_\mathbf{p}=1\}}{\arg\min}\ d(\mathbf{p},\mathbf{m}). \tag{6}$$

After determining the anchor point, we then need to select the angle of the line. During *training*, to avoid the overfitting during training, the angle θ is set randomly within $[0, 180°)$, by which the line is rotated clockwise from the positive x-axis. For *testing*, we make a vertical line of the skeleton based on the anchor point $\mathbf{a}$, which tends to pass through the point $\mathbf{m}$. Fig. 3 (4) shows two examples of the vertical line.

Step 5: With the anchor point $\mathbf{a}$ determined and the angle vertical to the skeleton, we can draw the cutting line. We first locate both ends of the line on the boundary of the thin part. To cater to users' operating habits, we give both ends of this line a certain extension l. The initial l is the same length as the line on the thin part. For *training*, the extension l is scaled randomly for two ends respectively. However, during *testing*, both ends will be extended by initial l with equal length. Fig. 3 (5) shows the final visualized result of the simulated line.

4 PROPOSED NETWORK

4.1 Feature Extractor

With the popularity of the deep-learning-based method, classic networks are usually adopted to extract the feature of the image. Since most networks extract the feature through a high-to-low resolution fashion, the information of thin parts may be lost during the down-sampling process, resulting in poor segmentation performance for thin structures. In order to maintain the high-resolution feature through the whole architecture, we take HRNet [40] as our feature extractor. The output feature with a resolution of $\frac{1}{4}$ will be used as the feature in KnifeCut, which is denoted as $\mathbf{F}$.

In addition, ResNet [12] is also adopted as our feature extractor to keep on bar with other interactive segmentation methods. However, since it extracts the feature through a high-to-low process, the high-resolution feature output by the first few layers contains limited high-level representations. Thus, in order to maintain the feature with the same resolution as HRNet, we construct the similar architecture of DeepLab v3+ [7]. The feature output by ResNet will be fed into ASPP and the decoder module; meanwhile its resolution will be restored to $\frac{1}{4}$. The effects of the feature, extracted by HRNet and ResNet, will both be presented in Sec. 5.3.

4.2 Thin Similarity Module

Based on the observation that the thin parts of an object usually share similar features, like the ant's legs, we design a simple yet effective module to utilize the similarity information. We first concatenate the line mask $\mathbf{L}$ and feature $\mathbf{F}$ together and feed them into seven 3×3 convolution layers along with batch normalization layers and ReLU activation functions. Multiplying the output by $\mathbf{L}$, we can obtain the prediction of the thin part adjacent to the line, which is denoted as $\mathbf{P}_{\text{line}}$. Using $\mathbf{F}$ and prediction $\mathbf{P}_{\text{line}}$, we can obtain the

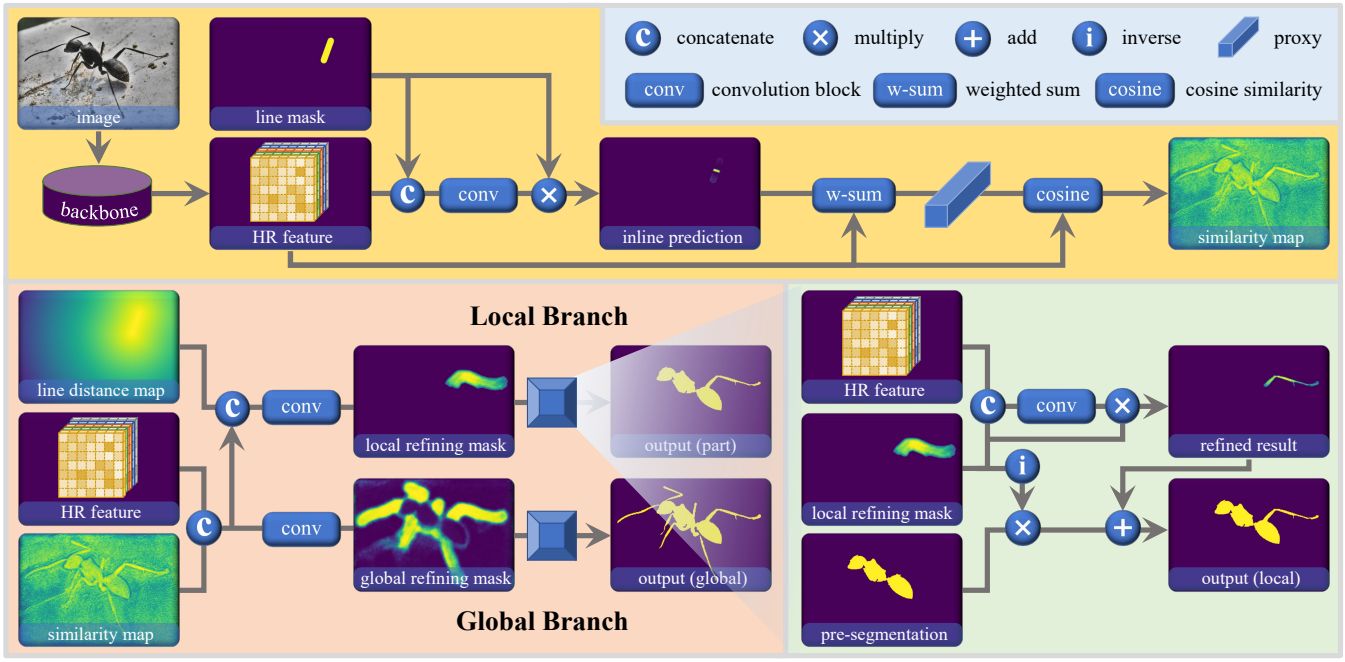

Figure 4: The network architecture of KnifeCut. The top part is the thin similarity module, which activates all relevant thin parts by similarity proxy on the cutting line. The bottom part is the segmentation refinement module with two branches. It first estimates the local and global refining masks, and then gets the refined results with the help of the two masks. Combining the refined results and pre-segmentation mask, the network outputs the local and global results. Consult Sec. 4 for more details.

proxy vector $\boldsymbol{s}$ for the thin parts, which is defined as

$$\boldsymbol{s} = \frac{\tilde{\sum}(\mathbf{F} \times \mathbf{P}_{\text{line}})}{\sum \mathbf{P}_{\text{line}}}, \tag{7}$$

where $\tilde{\sum}$ means the channel-wise sum operation. Thus, the dimension of $\boldsymbol{s}$ shares the same number as the layers of $\mathbf{F}$. Then we can obtain the similarity map $\mathbf{S}$ by cosine similarity:

$$\mathbf{S}_{\mathbf{p}} = \frac{\boldsymbol{s} \cdot \mathbf{F}_{\mathbf{p}}}{\|\boldsymbol{s}\| \|\mathbf{F}_{\mathbf{p}}\|}, \tag{8}$$

where $\|\boldsymbol{s}\|$ and $\|\mathbf{F}_{\mathbf{p}}\|$ mean the norm of $\boldsymbol{s}$ and $\mathbf{F}_{\mathbf{p}}$. The $\mathbf{F}_{\mathbf{p}}$ refers to the feature vector of $\mathbf{F}$ at pixel location $\mathbf{p}$, which has the same dimension as $\boldsymbol{s}$. The visualized $\mathbf{S}$ can be seen in Fig. 4 and Fig. 5.

4.3 Segmentation Refinement Module

Considering different intentions of the users, we equip the segmentation refinement module with two branches. One grasps the interaction provided by users, providing the refinement only for the target thin part, while the other utilizes the similarity shared by thin parts, refining all the thin parts at one time. We first obtain the distance map $\mathbf{D}$ by $\mathbf{D}_{\mathbf{p}} = \phi(\mathbf{p}, \mathbf{L})$. To utilize the information of the interaction, the local branch takes the feature $\mathbf{F}$, the similarity map $\mathbf{S}$ and the distance map $\mathbf{D}$ as input; while the input of the global branch excludes $\mathbf{D}$.

With a similar network structure, the concatenated inputs are fed into the convolution block. Thus we can obtain the refining mask $\mathbf{M}$ for local and global branches respectively. Then we concatenate the $\mathbf{M}$ with the $\mathbf{F}$ and feed them into the convolution block, thus obtaining the prediction results $\mathbf{R}$ of the two branches. Since we aim to refine the results of thin parts based on the prediction $\mathbf{P}'$ obtained by other methods, we retain the results of $\mathbf{P}'$ for the body part and only change the parts for thin structures. The final prediction result $\mathbf{P}$ for two branches can be formulated by

$$\mathbf{P}_{\text{branch}} = \mathbf{M}_{\text{branch}} \times \mathbf{R}_{\text{branch}} + (1 - \mathbf{M}_{\text{branch}}) \times \mathbf{P}', \tag{9}$$

where the `branch` can be `local` or `global`.

4.4 Loss Function

As for the binary segmentation tasks, binary cross entropy (BCE) is usually adopted with weight map $\mathbf{W}$ as the loss function, which can be formulated as:

$$\ell(\mathbf{P}, \mathbf{G}, \mathbf{W}) = -\frac{1}{N} \sum \mathbf{W} \times (\mathbf{G} \times \log(\mathbf{P}) + (1 - \mathbf{G}) \times \log(1 - \mathbf{P})), \tag{10}$$

where $\mathbf{P}$ means the probability of prediction mask, and $\mathbf{G}$ means the label (0 or 1) of the ground truth. N is the total number of pixels. The weight map is used to force the network to focus on the segmentation quality of thin parts. In the thin similarity module, we adopt $\ell(\mathbf{P}_{\text{line}}, \mathbf{G}_{\text{thin}}, \mathbf{L})$ to supervise the inline prediction. As for the refined results in the segmentation refinement module, the weight map $\tilde{\mathbf{W}}$ can be formulated as

$$\tilde{\mathbf{W}}_{\text{branch}} = 1 - \mathbf{G}_{\text{non-thin}} + \mathbf{M}_{\text{branch}}, \tag{11}$$

where the `branch` refers to `local` or `global`. For the loss function to supervise the final output of the two branchs, we adopt the conventional BCE loss. The ground truth differs from each branch,

which can be formulated as

$$\tilde{\mathbf{G}}_{\text{branch}} = \begin{cases} \mathbf{P}' \times (1 - \mathbf{E}^*) + \mathbf{G} \times \mathbf{E}^*, & \text{branch = local} \\ \mathbf{P}' \times (1 - \mathbf{E}) + \mathbf{G} \times \mathbf{E}, & \text{branch = global} \end{cases}, \tag{12}$$

where $\mathbf{E}^*$ means the expanded region for the selected thin part and $\mathbf{E}$ means all the expanded regions. Therefore, the loss for both branches can be formulated as

$$\mathcal{L}_{\text{branch}} = \ell(\mathbf{R}_{\text{branch}}, \mathbf{G}, \tilde{\mathbf{W}}_{\text{branch}}) + \ell(\mathbf{P}_{\text{branch}}, \tilde{\mathbf{G}}_{\text{branch}}, \mathbf{A}), \tag{13}$$

where $\mathbf{A}$ refers to the all-one matrix. The total loss function $\mathcal{L}_{\text{total}}$ can be formulated as

$$\mathcal{L}_{\text{total}} = \alpha \mathcal{L}_{\text{line}} + \beta \mathcal{L}_{\text{local}} + \gamma \mathcal{L}_{\text{global}}, \tag{14}$$

where the $\mathcal{L}_{\text{local}}$ and $\mathcal{L}_{\text{global}}$ are calculated according to Equ. (13) by assigning `branch` with `local` or `global`. In our experiments, we impose a hard constraint on the segmentation of the cutting line. Thus, the α, β, and γ are set as 10,1, and 1.

5 EXPERIMENTS

5.1 Settings

Datasets. We adopt the following datasets for our experiments:

- **ThinObject-5K [24]:** The dataset contains 4748, 500, and 500 for training, validation, and testing respectively. All the images in the dataset are synthetic by compositing the foreground objects on the background images. In this paper, we train our model on the training set and evaluate it on the test set.
- **HRSOD [44]:** The dataset is initially proposed for salient object detection tasks. As previous works [24] did, we use the same 287 images for evaluation, which include 305 objects with thin parts.
- **COIFT [33, 34]:** To follow previous works [24], the dataset contains 280 images, which combines 3 datasets of birds and insects in [33, 34]. Note that the images in this dataset are with much lower resolution than the other two datasets.

Evaluation Metric. As our method is proposed specifically for thin objects or parts, and the number of pixels belonging to the thin region is often so small that the improvements cannot be reflected obviously on the Intersection over Union (IoU) metric. Following [24], we adopt the thin IoU score IoU_{thin} on the expanded thin part region $\mathbf{E}$ (detailed in Sec. 3.2) to better reflect the segmentation performance. We first obtain our predictions for thin parts by $\mathbf{P}_{\text{thin}} = \mathbf{P} \times \mathbf{E}$. Thus the IoU_{thin} can be formulated as

$$\text{IoU}_{\text{thin}} = \frac{\sum(\mathbf{P}_{\text{thin}} \cap \mathbf{G}_{\text{thin}})}{\sum(\mathbf{P}_{\text{thin}} \cup \mathbf{G}_{\text{thin}})}. \tag{15}$$

Moreover, the boundary measure $\mathcal{F}$ [36] will also be adopted to evaluate the quality of the segmented edge. We compute the contour-based precision and recall P_c and R_c between the contour points of $c(\mathbf{P}_{\text{thin}})$ and $c(\mathbf{G}_{\text{thin}})$ via morphology operators. The $\mathcal{F}_{\text{thin}}$ is defined as

$$\mathcal{F}_{\text{thin}} = \frac{2 P_c R_c}{P_c + R_c}. \tag{16}$$

For the two metrics, the higher they are, the better performance is.

Implementation Details. The experiments based on ResNet [12] and HRNet [40] are conducted with ResNet-50 and HRNet-18, pre-trained on ImageNet [9]. We train our KnifeCut on ThinObject-5K train set. The training process lasts for 30 epochs and the batch size is set to 4. We adopt the exponential learning rate decay strategy with the initial learning rate of 7×10^{-3} and gamma of 0.9 for each epoch. Stochastic Gradient Descent (SGD) is used as the parameter optimizer with a momentum of 0.9 and weight decay of 5×10^{-4}. During training, we use random flipping to augment the data. Additionally, in order to help the similarity module recognize the difference between thin parts, we also adopt a random pasting strategy, which will be detailed in Sec. 5.2. While for testing, the simulation algorithm is introduced in Sec. 3.2. Note that post-processing is included in our evaluation, binarizing the refining mask by threshold.

Speed Analysis. We test the inference time of our KnifeCut on a single NVIDIA Titan XP GPU. When adopting HRNet as the feature extractor, it takes 0.057 seconds to process one 1024×1024 image; while it is 0.281 seconds for ResNet-based Deeplab v3+. Both speeds are fast enough to meet the need for real-time inference.

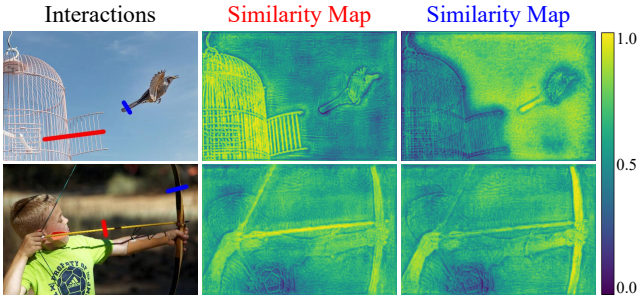

Figure 5: Illustration for different similarity maps according to the cutting line. The left and right similarity maps correspond to the red and blue lines. The active regions of similarity map are highly-related to the interacted thin part.

5.2 Ablations

We perform ablation experiments to demonstrate the effectiveness of each module adopted by the local and global refinement. The results are shown in Tab. 1. We use the $\mathbf{G}_{\text{non-thin}}$ as our pre-segmentation mask, and choose IoU_{thin} and $\mathcal{F}_{\text{thin}}$ as the evaluation metrics. The modules will be gradually removed from the KnifeCut. Additionally, we replace the cutting line with a click located on the thin part to demonstrate the superiority of our interaction mode. Both experiments based on HRNet and ResNet will be carried out on ThinObject-5K, HRSOD, and COIFT three datasets.

Similarity Module. We first remove the similarity module for both the local and global segmentation modules. The results are shown in Tab. 1. As for the local branch, the performance of KnifeCut worsens simultaneously on three datasets with approximately 3% reduction of IoU_{thin} and $\mathcal{F}_{\text{thin}}$ for ResNet. However, for HRNet, the improvement brought by the similarity map is limited but still a little. This is probably because HRNet maintains the high-resolution features, while the ResNet loses some thin part information during the down-sampling process, so the activation of the thin part makes a bigger impact. Thus the results demonstrate that the similarity map assists the network, especially for those with low-resolution feature, to recognize the thin part better.

Table 1: The core ablation study of the KnifeCut. We use the metric IoU_{thin} and $\mathcal{F}_{\text{thin}}$ to evaluate the segmentation of the thin parts. Symbol ↑ means that the performance is better when the metric is larger. 'A/B' means adopting ResNet and HRNet as the feature extractor. $\text{G}_{\text{non-thin}}$ is adopted as the pre-segmentation. 'SM' and 'DM' mean the similarity module and distance map respectively. Symbol → means the operation of replacement.

#	Candidate	ThinObject5K		HRSOD		COIFT	
		$\text{IoU}_{\text{thin}}\uparrow$	$\mathcal{F}_{\text{thin}}\uparrow$	$\text{IoU}_{\text{thin}}\uparrow$	$\mathcal{F}_{\text{thin}}\uparrow$	$\text{IoU}_{\text{thin}}\uparrow$	$\mathcal{F}_{\text{thin}}\uparrow$
	Pre-segmentation ($\text{G}_{\text{non-thin}}$)	0.000	0.000	0.000	0.000	0.000	0.000
local	KnifeCut	0.637/0.661	0.776/0.785	0.400/0.413	0.674/0.659	0.448/0.493	0.667/0.713
	KnifeCut (w/o SM)	0.604/0.655	0.738/0.778	0.371/0.403	0.629/0.659	0.416/0.482	0.640/0.702
	KnifeCut (w/o SM & DM)	0.554/0.618	0.692/0.749	0.164/0.206	0.283/0.339	0.279/0.403	0.447/0.602
	KnifeCut (Line → Click)	0.584/0.615	0.692/0.716	0.361/0.379	0.575/0.578	0.386/0.439	0.524/0.578
global	KnifeCut	0.798/0.821	0.926/0.932	0.497/0.504	0.809/0.792	0.671/0.688	0.903/0.922
	KnifeCut (w/o SM)	0.780/0.803	0.910/0.915	0.285/0.297	0.497/0.488	0.600/0.628	0.838/0.850
	KnifeCut (Line → Click)	0.793/0.822	0.916/0.931	0.489/0.498	0.788/0.779	0.667/0.690	0.894/0.918

The situation is much worse for global segmentation refinement. On the one hand, the worse performance proves the significant role of similarity map in global refinement. All similar thin parts activated on the image, the network can accurately estimate the scope of thin parts and focus on the refinement of them, which is in line with our expectations. On the other hand, the various degrees of worsening for datasets can be attributed to the distribution of different datasets. HRSOD is composed of real images with complex environment, thus increasing the segmentation difficulty; while ThinObject-5K `test` has the same data distribution as our training set, and only contains synthetic images with apparent boundaries. Therefore, the improvement for HRSOD better demonstrates the necessity of similarity map when putting into practical use.

Another advantage of the similarity map which cannot be reflected by statistics is to distinguish the thin parts belonging to different objects. Activating all thin parts regardless of their features will confuse the network about the location to be refined and be likely to segment them all, which will violate the users' intention in most cases. However, limited by the training sets, it is rare for different thin objects to appear at the same time. To make the network better deal with such situations, we adopt the random pasting strategy in the training process. Two suitable images will be randomly selected to paste together and fed into the network, while only one thin object will be segmented. The results are shown in Fig. 5. We can see that when cutting the cage, only the cage will be activated, and so is the bird tail. The second image also performs well, with bow and arrow activated respectively according to the cutting line. Note that since the similarity map is independent of the pre-segmentation map, we do not show it in Fig. 5.

Line Distance Map. For the local segmentation refinement module, we further remove the distance map. As shown in Tab. 1, the performance has become worse to varying degrees for the three datasets. The reason why the fall differs from the datasets is the same with the similarity module. Thus it also demonstrates the dominant role of distance map for local refinement. Under its guidance, the network is aware of the local scope of the thin part and focuses on the refinement in the marked region.

Cutting Line. As a significant contribution of our KnifeCut, the ablation study for the interaction mode is also carried out. Since we view clicks as the degradation of lines, we replace the cutting line with one click in KnifeCut. Following the common practice [43] of click-based interactive methods, we place the click on the center of the largest thin error region. The results are shown in Tab. 1. We can see that the performance becomes worse to some extent for both IoU_{thin} and $\mathcal{F}_{\text{thin}}$. Compared to clicks, the cutting line is easy to pass through multiple thin parts rather than only one. It also offers the contrasting prior as it passes through the background and foreground. Additionally, drawing a cutting line is convenient and intuitive for users, and is friendly for most devices, such as a mouse, touchpads, and mobile devices; while, for clicks, carefully aiming and clicking on each thin part is time-consuming and laborious.

5.3 Comparisons

Performance Evaluation. As shown in Tab. 2, we compare our KnifeCut with other methods on ThinObject-5K, HRSOD, and COIFT datasets. Because our method is designed specifically for thin refinement, We only use evaluation metrics with regard to thin parts. These adopted metrics have been introduced in Sec. 5.1. As for interactive segmentation methods for common objects, we choose f-BRS [38] and FCA-Net [27], as they have well-maintained codes. To keep fair with these methods, we adopt their results with two clicks as the pre-segmentation for our KnifeCut. Since the cutting line can be viewed as two clicks (two endpoints), we compare our results to theirs with 4 clicks. It is worth noting that we fine-tune these models with ThinObject-5K training set. We can see that the improvement for thin parts brought by our method is considerable for both local and global refinement. This not only demonstrates the feasibility of our method to serve as the refinement tool, but also shows the suitability and effectiveness of the cutting line to deal with thin parts segmentation.

For interactive thin object segmentation, we also compare our method with TOS-Net [24]. Also in order to control the same number of clicks, we retrain a model using a bounding box (two outside clicks at the symmetrical corner locations) as the interaction mode,

Table 2: Comparison of IoU_{thin} and $\mathcal{F}_{thin}$ metrics with other methods in three evaluation datasets. Symbol † and § means the output of the local branch and global one. 'A/B' means the results adopting ResNet and HRNet as the feature extractor.

Method	Interaction	ThinObject5K		HRSOD		COIFT	
		$IoU_{thin}\uparrow$	$\mathcal{F}_{thin}\uparrow$	$IoU_{thin}\uparrow$	$\mathcal{F}_{thin}\uparrow$	$IoU_{thin}\uparrow$	$\mathcal{F}_{thin}\uparrow$
FCA-Net [27]	4 guidance clicks	0.645	0.776	0.372	0.618	0.498	0.726
KnifeCut†	2 guidance clicks + 1 line	0.758/0.770	0.887/0.888	0.488/0.501	0.814/0.800	0.601/0.623	0.869/0.888
KnifeCut§	2 guidance clicks + 1 line	0.811/0.826	0.927/0.927	0.519/0.532	0.850/0.843	0.678/0.694	0.932/0.939
f-BRS [38]	4 guidance clicks	0.784	0.872	0.455	0.713	0.631	0.855
KnifeCut†	2 guidance clicks + 1 line	0.812/0.823	0.914/0.916	0.502/0.512	0.823/0.805	0.666/0.679	0.914/0.921
KnifeCut§	2 guidance clicks + 1 line	0.828/0.840	0.929/0.931	0.523/0.535	0.850/0.847	0.692/0.704	0.939/0.943
TOS-Net [24]	4 extreme clicks	0.865	0.938	0.651	0.916	0.764	0.962
KnifeCut†	1 bounding box + 1 line	0.874/0.875	0.944/0.945	0.666/0.668	0.916/0.917	0.772/0.775	0.965/0.966
KnifeCut§	1 bounding box + 1 line	0.873/0.877	0.946/0.946	0.664/0.670	0.917/0.917	0.786/0.793	0.968/0.969

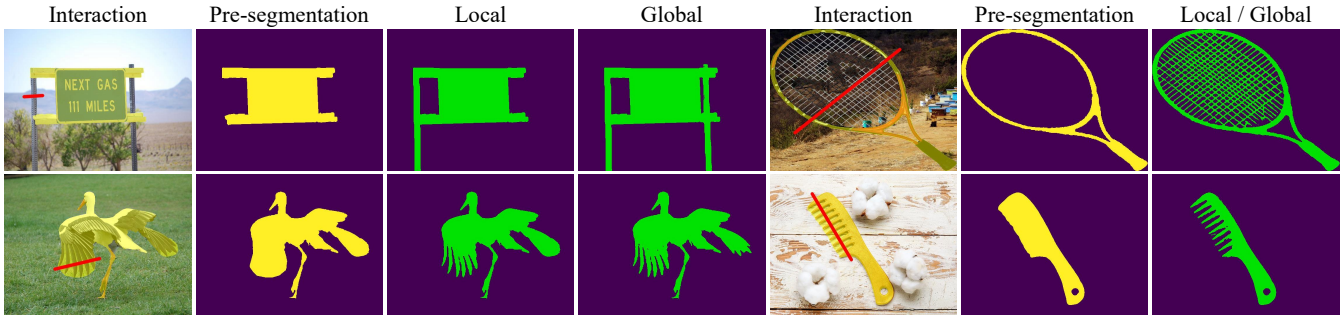

Figure 6: The quality results of the KnifeCut. The pre-segmentation and the cutting line are shown above. Both the local and global refinement results are provided. 'Local/Global' means the results are almost the same and the local one is presented.

which has the same network structure as TOS-Net. The results obtained by the model will be used as pre-segmentation, so that the total interaction can still be controlled as 4. We can see that after refined by KnifeCut, the segmentation for thin parts has been improved, surpassing that of TOS-Net.

Qualitative Results. Fig. 6 shows some situations suitable for KnifeCut to deal with. When thin parts are miss-segmented, KnifeCut can refine the pre-segmentation and thus generate excellent prediction with only one cutting line through the missed thin part. For example, the left pole of the sign is lost, and the user is required to draw a line through it. As the local result, the left pole will be completed on the mask; while the right one will also be included at one time in the global result. As for more complicated cases, the racket lacks the segmentation of the net, which is unimaginable for existing refinement tools. Fortunately, it will be solved by KnifeCut in the same way even in this hard situation without any increase in time and consideration. In another case, the thin parts often stay too close, resulting in over-segmentation. We take the bird's wings as instance to present the situation. Cutting the over-segmented parts, and then the target thin part will be refined in the local branch, while both wings for the global result. The comb is the more complicated cases, yet KnifeCut works well. Note that since for the net and comb, the results for local and global are almost the same, we only display the local one.

6 CONCLUSIONS

In this paper, to segment vexing thin objects, we propose a novel interaction mode, which only needs users to draw a line cutting through the mislabeled thin parts. Compared with current interactive modes, it effectively reduces the burden on the users, and is easy to control with a mouse, touchpad, and mobile device. To fully explore and demonstrate the superiority of the mode, we propose a post-processing model, KnifeCut, to further refine the pre-segmentation obtained by other segmentation methods, especially for the miss-segmented or over-segmented thin parts. Experiments on three datasets have demonstrated the superiority of the effectiveness of KnifeCut as the annotation tool.

ACKNOWLEDGMENTS

This work is funded by the National Key Research and Development Program of China (NO. 2018AAA0100400), NSFC (NO. 61922046), The Fundamental Research Funds for the Central Universities, Nankai University, China Postdoctoral Science Foundation (NO.2021M701780).

REFERENCES

[1] David Acuna, Huan Ling, Amlan Kar, and Sanja Fidler. 2018. Efficient interactive annotation of segmentation datasets with polygon-rnn++. In *CVPR*.

[2] Eirikur Agustsson, Jasper RR Uijlings, and Vittorio Ferrari. 2019. Interactive full image segmentation by considering all regions jointly. In *CVPR*.

[3] Rodrigo Benenson, Stefan Popov, and Vittorio Ferrari. 2019. Large-scale interactive object segmentation with human annotators. In *CVPR*.

[4] Yuri Boykov and Vladimir Kolmogorov. 2004. An experimental comparison of min-cut/max-flow algorithms for energy minimization in vision. *IEEE TPAMI* (2004).

[5] Yuri Y Boykov and M-P Jolly. 2001. Interactive graph cuts for optimal boundary & region segmentation of objects in ND images. In *ICCV*.

[6] Lluis Castrejon, Kaustav Kundu, Raquel Urtasun, and Sanja Fidler. 2017. Annotating object instances with a polygon-rnn. In *CVPR*.

[7] Liang-Chieh Chen, Yukun Zhu, George Papandreou, Florian Schroff, and Hartwig Adam. 2018. Encoder-decoder with atrous separable convolution for semantic image segmentation. In *ECCV*.

[8] Xi Chen, Zhiyan Zhao, Feiwu Yu, Yilei Zhang, and Manni Duan. 2021. Conditional Diffusion for Interactive Segmentation. In *ICCV*.

[9] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. 2009. Imagenet: A large-scale hierarchical image database. In *CVPR*.

[10] Xingping Dong, Jianbing Shen, Ling Shao, and Luc Van Gool. 2015. Sub-Markov random walk for image segmentation. *IEEE TIP* (2015).

[11] Leo Grady. 2006. Random walks for image segmentation. *IEEE TPAMI* (2006).

[12] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2016. Deep residual learning for image recognition. In *CVPR*.

[13] Yang Hu, Andrea Soltoggio, Russell Lock, and Steve Carter. 2019. A fully convolutional two-stream fusion network for interactive image segmentation. *NN* (2019).

[14] Suyog Dutt Jain and Kristen Grauman. 2019. Click carving: Interactive object segmentation in images and videos with point clicks. *IJCV* (2019).

[15] Won-Dong Jang and Chang-Su Kim. 2019. Interactive Image Segmentation via Backpropagating Refinement Scheme. In *CVPR*.

[16] Stefanie Jegelka and Jeff Bilmes. 2010. *Cooperative cuts for image segmentation.* Technical Report. University of Washington.

[17] Meng Jian and Cheolkon Jung. 2016. Interactive image segmentation using adaptive constraint propagation. *IEEE TIP* (2016).

[18] Tae Hoon Kim, Kyoung Mu Lee, and Sang Uk Lee. 2008. Generative image segmentation using random walks with restart. In *ECCV*.

[19] Theodora Kontogianni, Michael Gygli, Jasper Uijlings, and Vittorio Ferrari. 2020. Continuous adaptation for interactive object segmentation by learning from corrections. In *ECCV*.

[20] Hoang Le, Long Mai, Brian Price, Scott Cohen, Hailin Jin, and Feng Liu. 2018. Interactive boundary prediction for object selection. In *ECCV*.

[21] Yin Li, Jian Sun, Chi-Keung Tang, and Heung-Yeung Shum. 2004. Lazy snapping. *ACM TOG* (2004).

[22] Zhuwen Li, Qifeng Chen, and Vladlen Koltun. 2018. Interactive image segmentation with latent diversity. In *CVPR*.

[23] JunHao Liew, Yunchao Wei, Wei Xiong, Sim-Heng Ong, and Jiashi Feng. 2017. Regional interactive image segmentation networks. In *ICCV*.

[24] Jun Hao Liew, Scott Cohen, Brian Price, Long Mai, and Jiashi Feng. 2021. Deep Interactive Thin Object Selection. In *WACV*.

[25] Jun Hao Liew, Scott Cohen, Brian Price, Long Mai, Sim-Heng Ong, and Jiashi Feng. 2019. Multiseg: Semantically meaningful, scale-diverse segmentations from minimal user input. In *ICCV*.

[26] Zheng Lin, Zheng-Peng Duan, Zhao Zhang, Chun-Le Guo, and Ming-Ming Cheng. 2022. FocusCut: Diving Into a Focus View in Interactive Segmentation. In *CVPR*.

[27] Zheng Lin, Zhao Zhang, Lin-Zhuo Chen, Ming-Ming Cheng, and Shao-Ping Lu. 2020. Interactive image segmentation with first click attention. In *CVPR*.

[28] Zheng Lin, Zhao Zhang, Ling-Hao Han, and Shao-Ping Lu. 2022. Multi-Mode Interactive Image Segmentation. In *ACM MM*.

[29] Zheng Lin, Zhao Zhang, Zi-Yue Zhu, Deng-Ping Fan, and Xia-Lei Liu. 2022. Sequential Interactive Image Segmentation. *CVMJ* (2022).

[30] Huan Ling, Jun Gao, Amlan Kar, Wenzheng Chen, and Sanja Fidler. 2019. Fast interactive object annotation with curve-gcn. In *CVPR*.

[31] Soumajit Majumder and Angela Yao. 2019. Content-Aware Multi-Level Guidance for Interactive Instance Segmentation. In *CVPR*.

[32] Kevis-Kokitsi Maninis, Sergi Caelles, Jordi Pont-Tuset, and Luc Van Gool. 2018. Deep extreme cut: From extreme points to object segmentation. In *CVPR*.

[33] Lucy AC Mansilla and Paulo AV Miranda. 2016. Oriented image foresting transform segmentation: Connectivity constraints with adjustable width. In *SIGGRAPH*.

[34] Lucy AC Mansilla, Paulo AV Miranda, and Fábio AM Cappabianco. 2016. Oriented image foresting transform segmentation with connectivity constraints. In *ICIP*.

[35] Eric N Mortensen and William A Barrett. 1995. Intelligent scissors for image composition. In *SIGGRAPH*.

[36] Federico Perazzi, Jordi Pont-Tuset, Brian McWilliams, Luc Van Gool, Markus Gross, and Alexander Sorkine-Hornung. 2016. A benchmark dataset and evaluation methodology for video object segmentation. In *CVPR*.

[37] Carsten Rother, Vladimir Kolmogorov, and Andrew Blake. 2004. Grabcut: Interactive foreground extraction using iterated graph cuts. *ACM TOG* (2004).

[38] Konstantin Sofiiuk, Ilia Petrov, Olga Barinova, and Anton Konushin. 2020. f-BRS: Rethinking backpropagating refinement for interactive segmentation. In *CVPR*.

[39] Sara Vicente, Vladimir Kolmogorov, and Carsten Rother. 2008. Graph cut based image segmentation with connectivity priors. In *CVPR*.

[40] Jingdong Wang, Ke Sun, Tianheng Cheng, Borui Jiang, Chaorui Deng, Yang Zhao, Dong Liu, Yadong Mu, Mingkui Tan, Xinggang Wang, et al. 2020. Deep high-resolution representation learning for visual recognition. *IEEE TPAMI* (2020).

[41] Tao Wang, Zexuan Ji, Quansen Sun, Qiang Chen, Qi Ge, and Jian Yang. 2018. Diffusive likelihood for interactive image segmentation. *PR* (2018).

[42] Tao Wang, Jian Yang, Zexuan Ji, and Quansen Sun. 2018. Probabilistic diffusion for interactive image segmentation. *IEEE TIP* (2018).

[43] Ning Xu, Brian Price, Scott Cohen, Jimei Yang, and Thomas S Huang. 2016. Deep interactive object selection. In *CVPR*.

[44] Yi Zeng, Pingping Zhang, Jianming Zhang, Zhe Lin, and Huchuan Lu. 2019. Towards High-Resolution Salient Object Detection. In *ICCV*.

[45] Shiyin Zhang, Jun Hao Liew, Yunchao Wei, Shikui Wei, and Yao Zhao. 2020. Interactive object segmentation with inside-outside guidance. In *CVPR*.

[46] Tongjie Y Zhang and Ching Y. Suen. 1984. A fast parallel algorithm for thinning digital patterns. *Commun. ACM* (1984).

[47] Yibiao Zhao, Song-Chun Zhu, and Siwei Luo. 2010. Co3 for ultra-fast and accurate interactive segmentation. In *ACM MM*.