Deep Interactive Thin Object Selection (Supplementary Material)

Jun Hao Liew[1] Scott Cohen[2] Brian Price[2] Long Mai[2] Jiashi Feng[1]
[1] National University of Singapore [2] Adobe Research
liewjunhao@u.nus.edu {scohen,bprice,malong}@adobe.com elefjia@nus.edu.sg

A. Additional Comparison with SOTAs

In addition to Fig. 3 in the main paper which compares the state-of-the-art interactive segmentation methods on HRSOD [15] dataset, we additionally provide evaluation results on COIFT [10] dataset (Fig. 1). We first notice that despite producing overall good quality segmentation (IoU $\sim 80\%$), existing state-of-the-art methods struggle when segmenting elongated thin parts ($\text{IoU}_{\text{thin}} \leq 55\%$). On the other hand, our TOS-Net outperforms all other methods by a significant margin ($\sim$ 10% IoU and $\geq$ 25% IoU_{thin}). Moreover, the relative performance gap between IoU and IoU_{thin} for TOS-Net is significantly smaller than other methods, demonstrating the effectiveness of both the three-stream design as well as the new dataset.

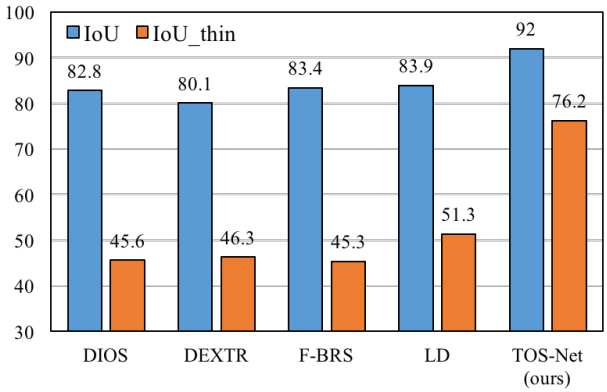


Figure 1: Comparison between our TOS-Net and existing state-of-the-art methods, including DIOS [14], DEXTR [9][1], f-BRS [13][2]and Latent Diversity [6][3]. We employ IoU and IoU_{thin} at 4th click as evaluation metric.

B. Extraction of Thin Parts

Intuitively, for a thin part, its innermost pixel should be close to the nearest boundaries. Inspired by this, we design an algorithm for extraction of thin parts for evaluation metric IoU_{thin}. The detailed steps are shown as follows:

• **Step 1:** Given an object mask $M \in \{0,1\}^{H \times W}$, we first compute an Euclidean distance transform as follows:

$$\phi(\mathbf{x}, \mathcal{B}) = \min_{\mathbf{x}' \in \mathcal{B}} D(\mathbf{x}, \mathbf{x}') \tag{1}$$

where $\mathcal{B}$ denotes the set of background pixels in M while $D(\mathbf{x}, \mathbf{x}')$ refers to the Euclidean distance between pixel locations $\mathbf{x}$ and $\mathbf{x}'$. Fig. 2(b) shows an example of distance map where each pixel value corresponds to the distance to its closest boundaries.

• **Step 2:** Given the distance map, we compute the local peaks and only retain those peaks whose distance value is smaller than a threshold τ. We denote these local peaks as "seeds" $\mathcal{S}$. In this work, we empirically set τ to:

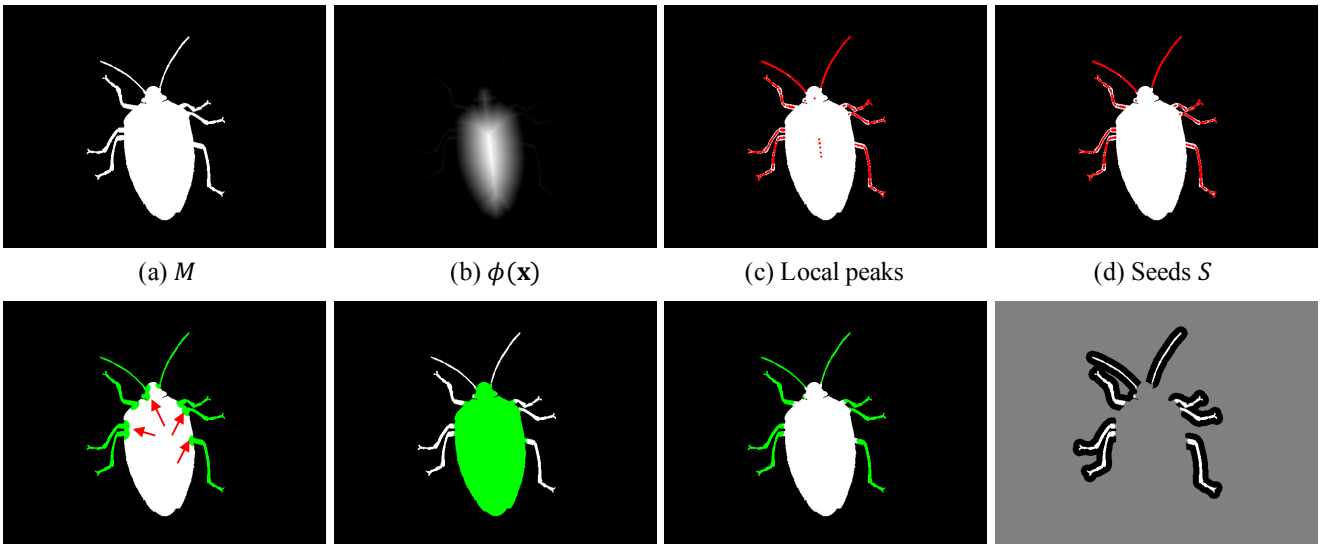
$$\tau = 10 \times \frac{\max(H_{box}, W_{box})}{300} \tag{2}$$

where H_{box} and W_{box} denote the height and width of the bounding box enclosing the object. An example of the extracted peaks before and after thresholding can be found in Fig. 2(c)(d).

• **Step 3:** We next aggregate all the pixels close to these seeds $\mathcal{S}$ to obtain the thin parts. To this end, we compute the shortest path between each pixel and the seeds $\mathcal{S}$ **within** the object mask M using `skfmm.distance`[4]. This is to avoid a pixel being included which is close in Euclidean distance to a seed point but the direct path would go through background. The resulting distance map $\psi(\mathbf{x})$ is then thresholded using the same threshold value τ in Eqn. (2) to obtain thin parts $T \in \{0,1\}^{H \times W}$ (Fig. 2(e)).

• **Step 4:** However, thin parts extracted from the previous step often contain some "non-thin" pixels. For example, some small parts of the insect body are included where the legs meet the body as indicated by the red arrows in Fig. 2(e). To remove these unwanted "non-thin"

[1] https://github.com/scaelles/DEXTR-PyTorch
[2] https://github.com/saic-vul/fbrs_interactive_segmentation
[3] https://github.com/intel-isl/Intseg
[4] https://pythonhosted.org//scikit-fmm/#skfmm.distance



(a) M (b) $\phi(\mathbf{x})$ (c) Local peaks (d) Seeds S

(e) Thin parts T (f) Non-thin parts N (g) Refined thin parts T^* (h) Evaluation mask

Figure 2: Extraction of thin parts for evaluation metric IoU_{thin}.

pixels, we first extract the "non-thin" region (Fig. 2(f)) via $N = M - T$ and aggregate the nearby pixels using the same `skfmm.distance` function in Step 3. The post-processed thin parts can then be obtained by simply thresholding the resulting distance map using τ and taking its complement. Fig. 2(g) shows an example of the refined thin parts, which we denote as T^*.

• **Step 5:** For evaluation, we also consider a small strip of background surrounding T^* to prevent trivial solution that predicts the entire mask to be foreground. Specifically, we extract an evaluation mask M_{eval} as follows:

$$M_{\text{eval}} = \{\mathbf{x} : \phi(\mathbf{x}, T^*) \leq \tau\} \setminus N^* \tag{3}$$

where $N^* = M - T^*$. As shown in Fig. 2(h), the gray pixels denote the "void" labels ($\{\mathbf{x} : M_{\text{eval}}(\mathbf{x}) = 0\}$) which will be excluded from evaluation. In this case, the metric IoU_{thin} can better evaluate the performance on thin parts.

C. Network Architecture

As described in the main paper, our TOS-Net consists of three separate streams: 1) context stream which accepts a *fixed-resolution* input image to extract global context for coarse prediction; 2) high-resolution edge stream that processes the *high resolution* input to delineate the object contours; and 3) fusion stream that fuses the information from the preceding two streams to produce the final segmentation output. The overall architecture is depicted in Fig. 3. The details of each stream will be illustrated below.

Context Stream. Our context stream employs ResNet-50-based [5] DeepLabv3+ [1] as its backbone. Specifically, it applies an Atrous Spatial Pyramid Pooling (ASPP) module for aggregating multi-scale context, followed by a decoder that incorporates low-level features from earlier layers for refinement. We append a 1×1 convolution layer with sigmoid activation at the end of the decoder to produce a fixed-resolution binary segmentation mask.

High-resolution Edge Stream. The high-resolution edge stream employs an FPN-style structure [7], containing multiple encoder and decoder blocks as shown in Fig. 4. At each encoder, the features are first passed to 2 consecutive {`conv-gn-relu`} blocks, followed by a max pooling layer before concatenated with the features from the context stream. However, we note that the features extracted from the context stream is incompatible with the encoder features when patch-based training scheme is adopted. To overcome this, we employ an `RoIAlign` [4] layer to extract the features from the corresponding patches in the context stream before the concatenation operation. During inference, we simply set the RoI to cover the full input image. Note that we do not apply max pooling for the last encoder block to avoid excessive loss in spatial details.

On the other hand, each decoder block first passes the features to 2 consecutive {`conv-gn-relu`} blocks, followed by a bilinear upsampling operation before being summed with features from the earlier encoder blocks. Two additional {`conv-gn-relu`} blocks are applied before passing to the next decoder. Similar to the context stream, we append a 1×1 convolution layer with sigmoid activation at the end of the last decoder to produce a boundary map whose resolution is the same as the input image.

Fusion Stream. The fusion stream takes the input im-

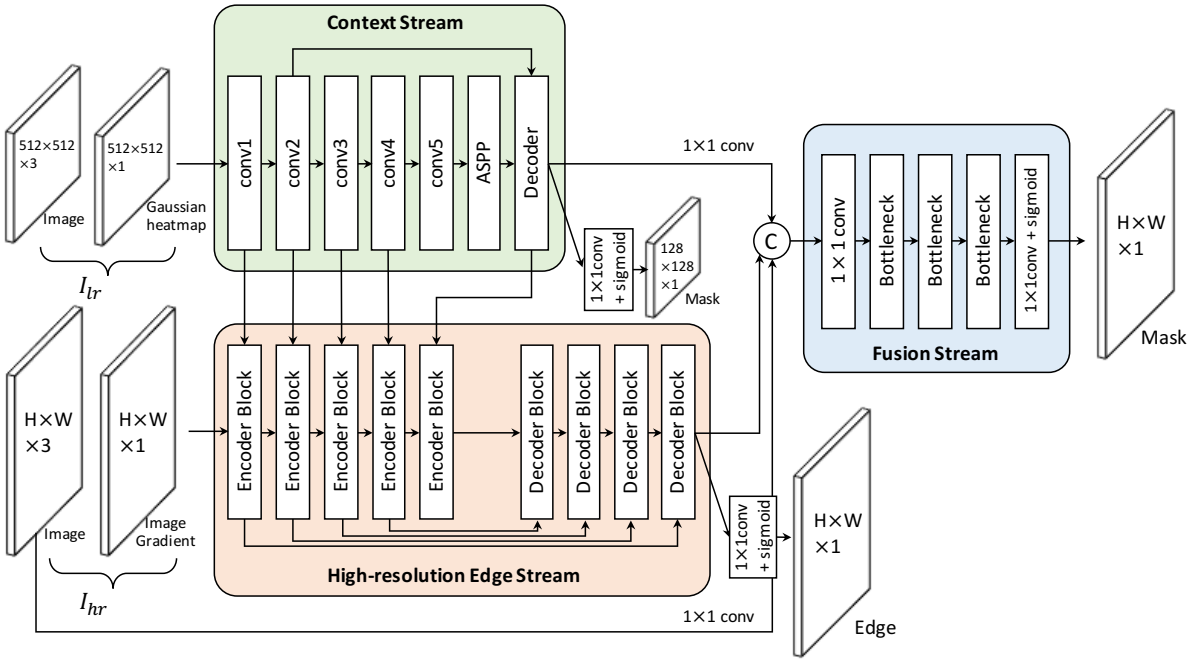
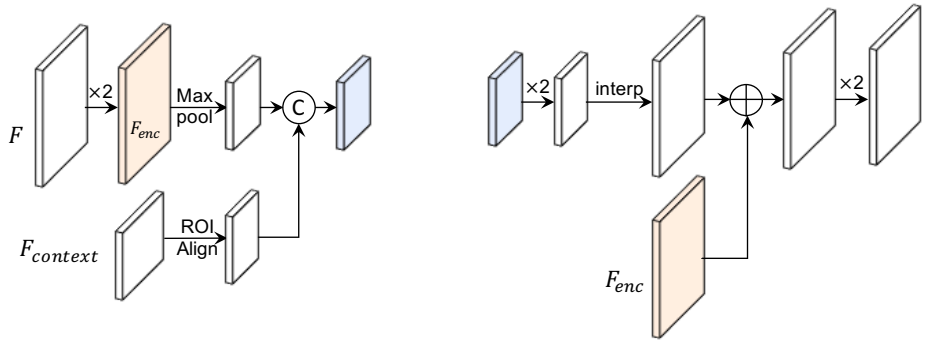


Figure 3: Overall architecture of TOS-Net. © denotes concatenation operation.



(a) Encoder block (b) Decoder block

Figure 4: Details of encoder and decoder blocks used in the high-resolution edge stream. '×2' denotes a stack of 2 consecutive {conv-gn-relu} blocks while 'interp' implies bilinear interpolation. © and ⊕ refer to concatenation and summation, respectively.

age, semantic features from the context stream and high-resolution boundary features from the edge stream as input for final prediction. Both the input image and boundary features are first passed to a 1×1 convolution layer prior to concatenation, followed by 3 bottleneck blocks [5]. Lastly, a 1×1 convolution layer with sigmoid activation is applied to output the final segmentation mask.

D. Comparison with Matting Algorithms

We also compare as a matting method as both our method and matting methods are designed to capture thin areas. The problems of course are slightly different in that matting methods receive a significant amount of user input in the form of a trimap (foreground/background/unknown) while our method only receives 4 input extreme points. On the other hand, matting methods can also capture a soft mask of thin areas where pixels are a mixture of foreground and background, while our method focuses on a binary segmentation. Nonetheless we think it is valuable to compare the performance on thin parts.

Specifically, we compare with IndexNet [8][5], a recently

[5] https://github.com/poppinace/indexnet_matting

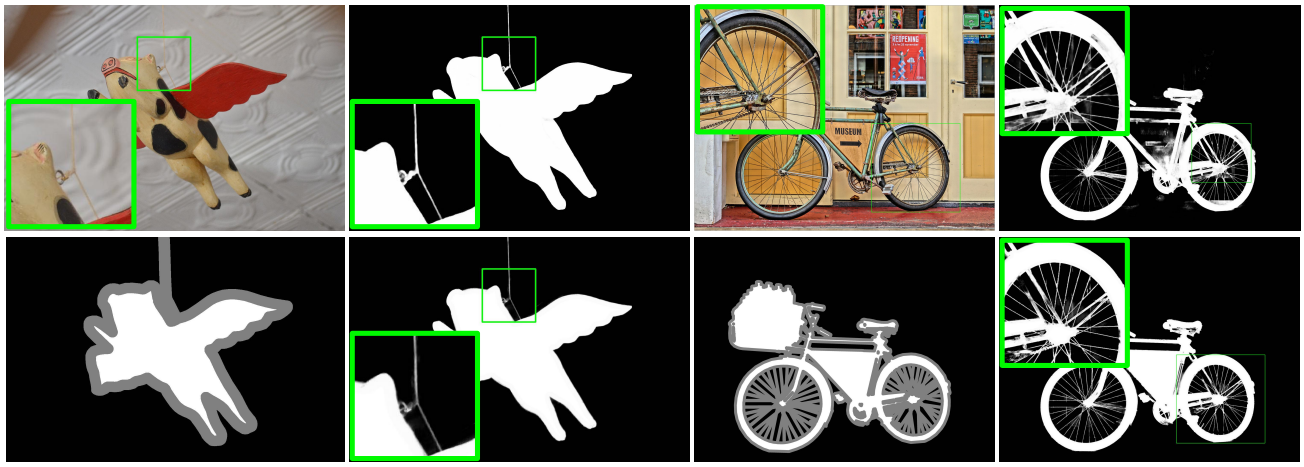


Figure 5: Comparison with image matting algorithm. **Top left:** Image taken from HRSOD [15] dataset. **Top Right:** Segmentation prediction by our TOS-Net. **Bottom left:** Trimap input for matting algorithm. **Bottom right:** Matting results by IndexNet [8]. Note that our method only requires 4 extreme clicks as input, while achieving comparable performance to that of matting algorithm which requires carefully drawn trimap as input.

Method	COIFT		HRSOD	
	IoU	IoU_{thin}	IoU	IoU_{thin}
IndexNet [8]	92.0	67.4	92.8	64.2
Ours	92.0	76.4	86.4	65.1

Table 1: Comparison with the state-of-the-art image matting method, IndexNet [8].

published state-of-the-art matting algorithm. Same as before, we evaluate on COIFT and HRSOD datasets using IoU and IoU_{thin} as metric. To generate a trimap input, we perform morphological dilation and erosion on the ground truth mask to create an "unknown" band. We use a larger kernel size ($k = 50$) for morpholocal operations on HRSOD dataset due to its average larger image size as compared to COIFT ($k = 25$). Examples of generated trimaps can be found in the bottom left of Fig. 5. We threshold the predicted alpha matte by 0.5 to obtain a binary mask for evaluation. The results are summarized in Table 1.

We first notice that IndexNet achieves very high IoU on both datasets ($\geq 92\%$) because the true labels of majority pixels are already known in the trimap input. In addition, it should also be noted that IndexNet receives significantly more information from the trimap as compared to four extreme points in our case, therefore leading to an overall better performance. On the other hand, in term of IoU_{thin} which focuses on segmentation accuracy on thin parts, we can see that our method attains a comparable or even better performance than IndexNet that requires carefully drawn trimap as inputs, suggesting the potential of our method for practical thin object segmentation application. We also provide some qualitative comparison in Fig. 5 where we can see our TOS-Net successfully segments the string (left subfigure) despite the strong appearance overlap between the string and its surrounding background.

E. Performance on General Objects

We also study the performance of TOS-Net on general object scenes, such as PASCAL VOC validation [2] (1,449 images), GrabCut [12] (50 images) and Berkeley [11] (100 images) datasets. The results are summarized in Table 2 and Fig. 6. We observe that our TOS-Net excels in extracting elongated thin details (*e.g.* potted plants in the first row of Fig. 6). However, in overall comparison, we notice that our TOS-Net trained on ThinObject-5K dataset performs poorly especially on PASCAL validation set when compared to DEXTR trained on PASCAL-10K dataset. When inspecting closer, we find this is mainly due to the inconsistent ground truth annotations between ThinObject-5K and PASCAL dataset. For example, as shown in the second row of Fig. 6, the 'table' class includes all the items on it. To validate this, we train our TOS-Net on both ThinObject-5K and PASCAL-1K [6] (which accounts for $\sim$13% of PASCAL-10K) and observe a significant boost in performance.

F. Additional Qualitative Results

In addition to Fig, 8 in our main paper, we also provide more qualitative results in Fig. 7, 8 and 9. Compared to

[6] We denote PASCAL train set augmented with additional labels from SBD [3] and the one without SBD labels as PASCAL-10K (10,582 images) and PASCAL-1K (1,464 images), respectively.

Methods	PASCAL	GrabCut	Berkeley
DEXTR [9]	91.5	94.4	-
TOS-Net	72.0	92.6	87.6
TOS-Net†	89.1	95.3	89.8

Table 2: General object segmentation on PASCAL [2], GrabCut [12] and Berkeley [11] datasets. † denotes training on both ThinObject-5K and PASCAL-1K datasets.

Input Image TOS-Net Ground truth

Figure 6: Qualitative results of our TOS-Net trained on ThinObject-5K dataset. The rightmost column shows the corresponding ground truth masks from PASCAL validation set where gray pixels denote "void" labels.

the baseline, our TOS-Net in general produces segmentation with accurately localized boundary, particularly along thin parts, Moreover, it performs well even on images with significant appearance overlap between thin parts and their surrounding backgrounds (*e.g.* necklace in Fig. 7, and butterfly antenna in Fig. 8 (2nd row)).

References

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

[2] Mark Everingham, Luc Van Gool, Christopher KI Williams, John Winn, and Andrew Zisserman. The pascal visual object classes (VOC) challenge. *IJCV*, 2010.

[3] Bharath Hariharan, Pablo Arbeláez, Lubomir Bourdev, Subhransu Maji, and Jitendra Malik. Semantic contours from inverse detectors. In *ICCV*, 2011.

[4] Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross Girshick. Mask R-CNN. In *ICCV*, 2017.

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

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

[7] Tsung-Yi Lin, Piotr Dollar, Ross Girshick, Kaiming He, Bharath Hariharan, and Serge Belongie. Feature pyramid networks for object detection. In *CVPR*, 2017.

[8] Hao Lu, Yutong Dai, Chunhua Shen, and Songcen Xu. Indices matter: Learning to index for deep image matting. In *ICCV*, 2019.

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

[10] Lucy AC Mansilla and Paulo AV Miranda. Oriented image foresting transform segmentation: Connectivity constraints with adjustable width. In *SIBGRAPI Conference on Graphics, Patterns and Images (SIBGRAPI)*, 2016.

[11] Kevin McGuinness and Noel E O'Connor. Toward automated evaluation of interactive segmentation. *Computer Vision and Image Understanding*, 2011.

[12] Carsten Rother, Vladimir Kolmogorov, and Andrew Blake. Grabcut: Interactive foreground extraction using iterated graph cuts. In *ACM ToG*, 2004.

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

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

[15] Yi Zeng, Pingping Zhang, Jianming Zhang, Zhe Lin, and Huchuan Lu. Towards high-resolution salient object detection. In *ICCV*, 2019.

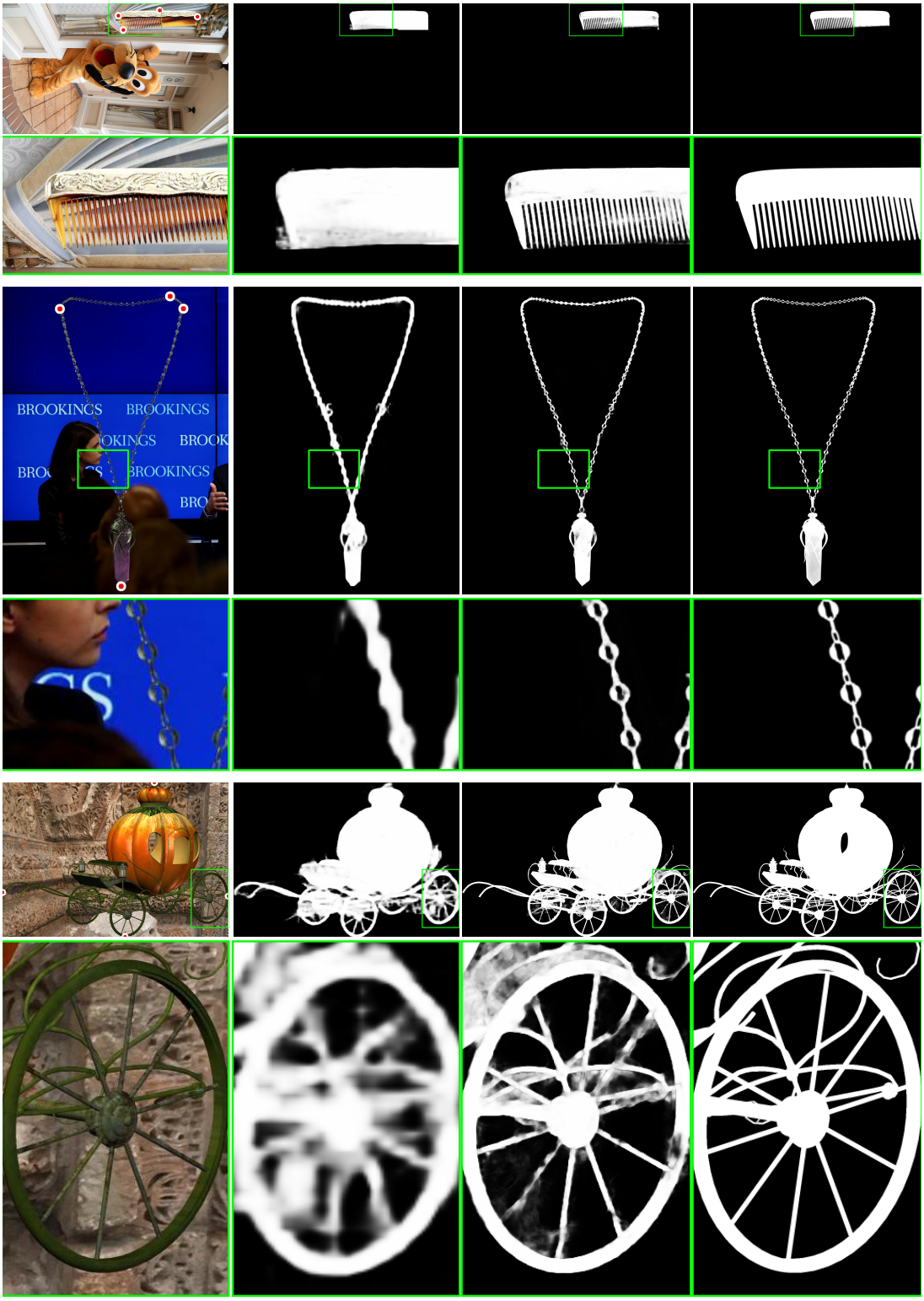


Figure 7: Qualitative comparison between DEXTR [9] (M2) and our TOS-Net (M8) on ThinObject-5K dataset. Note that both models are trained on ThinObject-5K for fair comparison. The red clicks denote the input extreme points.

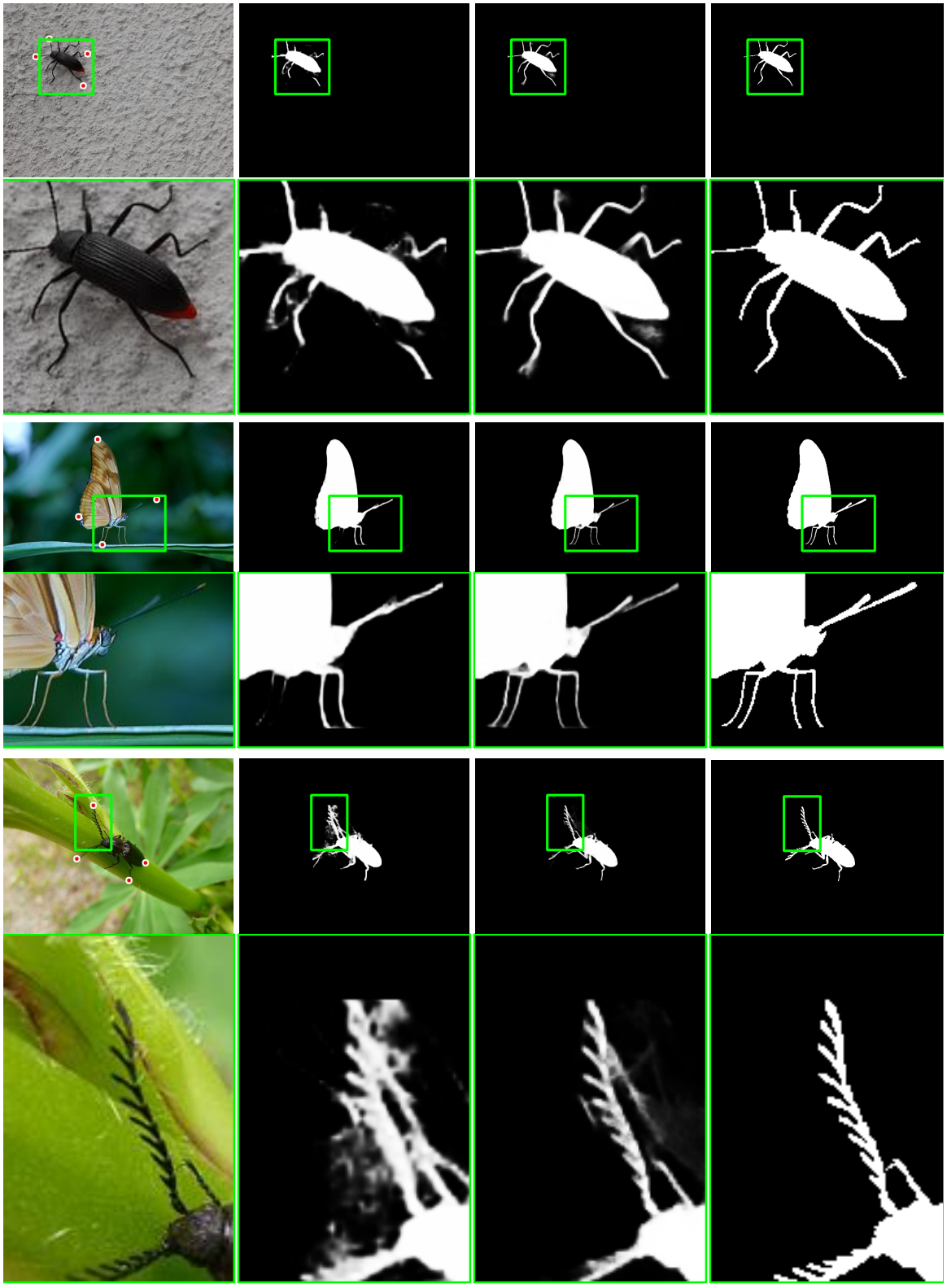


Figure 8: Qualitative results on COIFT [10] dataset.

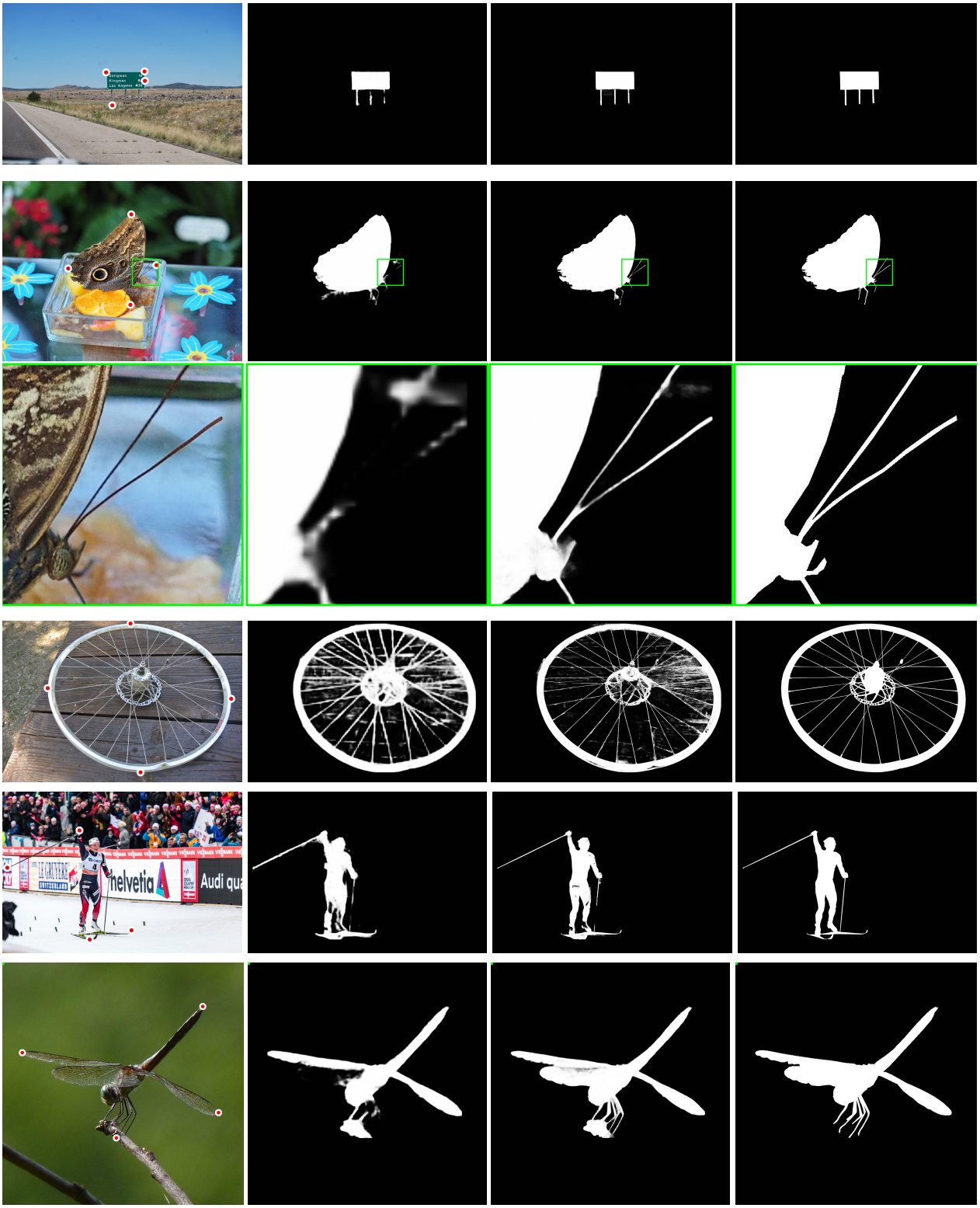


Figure 9: Qualitative results on HRSOD [15] dataset.