ScribbleBox: Interactive Annotation Framework for Video Object Segmentation

Supplementary Material

Bowen Chen[1⋆], Huan Ling[1,2,3,⋆], Xiaohui Zeng[1,2],
Jun Gao[1,2,3], Ziyue Xu[1], Sanja Fidler[1,2,3]

[1]University of Toronto [2]Vector Institute [3]NVIDIA
{chenbowen, linghuan, xiaohui, jungao, fidler}@cs.toronto.edu
{ziyue.xu}@mail.utoronto.ca

In the supplementary material, we provide a more detailed description of interactive segmentation training and the mask decoder's structure. We also show additional qualitative results, and provide an overview of our annotation tool when used in practice.

1 Interactive Segmentation Training Details

Multi-Stage Training For Mask Propagation Module: The *mask propagation module* is trained in two stages. We pretrain the module on synthetic video clips. Each clip includes a pair of reference and target frame, and is generated by applying random affine transformation and object composition following [4]. We expect this to make the network more robust to variations in object appearance. We then fine-tune it on video segmentation datasets. For each training video clip, we randomly sample 3 ordered frames and apply data augmentation including random flipping, 10% bounding box noise and affine transformation. The two-stage training takes about 1 day on synthetic video clips, 3 days on DAVIS2017 and 5 days on YoutubeVOS using 4 NVIDIA Tesla P100 GPUs.

Batch Hard Triplet Loss For Scribble Propagation Module: We utilize the batch hard triplet loss from [1] to supervise the embedding head in the scribble propagation network. First, we use $f_c \in \mathbb{R}^D$ to denote the feature vector at each location in the feature map F_c. Similarly, for each location in the image feature F_r of frame r, we have a feature vector f_r. For each $f_r \in F_r$, if it corresponds to f_c, we define it as positive/true sample with respect to f_c, denoted by f_r^+, otherwise it is negative/false sample, denoted as f_r^-. We denote the set of f_r^+ as $\{f_r^+\}$ and the set of f_r^- as $\{f_r^-\}$. Then the batch hard triplet loss can be written as:

$$L_{\text{BHTriplet}}(F_c, F_r) = \sum_{f_c \in F_c} l(f_c, \{f_r^+\}, \{f_r^-\}), \tag{1}$$

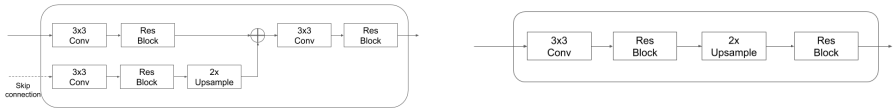
where

$$\begin{aligned} l(f_c, \{f_r^+\}, \{f_r^-\}) = &\min_{f_r^+ \in \{f_r^+\}} \|emb(f_c) - emb(f_r^+)\|_2^2 - \\ &\min_{f_r^- \in \{f_r^-\}} \|emb(f_c) - emb(f_r^-)\|_2^2 + \alpha \end{aligned} \tag{2}$$

⋆ authors contributed equally

Here, α is the minimal margin between positive and negative samples, and $emb(\cdot)$ is the embedding function. This loss is able to push two samples that do not correspond to each other away even if they have similar semantic information.

Implementation Details: We always crop the input images, masks (and scribbles) based on the input bounding box and resize them to 512×512 (or 256×256 when specifing). We use SGD optimizer with momentum of 0.9 and a cyclical learning rate policy [3] to speed up the training process. The minimum and maximum learning rate is set to 1e-5 and 1e-3, respectively. We use *triangular2* CLR policy and 4 cycles throughout training. Once mask propagation module is trained, we freeze the image encoder and train *interactive segmentation module* and *scribble propagation module*.



Refinement layer 1 and 2 Refinement layer 3

Fig. 1: Structure of three refinement layers in the decoder. Note that the skip connection in the refinement layer 3 is removed.

2 Mask Decoder's Structure

There are two decoders in our framework. *Interactive decoder* is used to produce a refined mask of a single frame based on the human-in-the-loop interaction. We also have a *propagation decoder* which is shared by both the mask and scribble propagation networks.

In particular, we combine three refinement modules [4] as our decoder. Different from the original structure, we remove the first refinement module and add an additional refinement module without skip connection before the last convolution layer. Three refinement modules produce feature maps with 224, 224, 128 channels, respectively, and the last convolution layer produces the final mask. The size of the output mask is half the size of the input image.

The detailed structures of our refinement modules are shown in Figure 1.

3 Qualitative Results

We illustrate an example Curve-VOT result in Fig. 2. In Figure 3 we provide a qualitative comparison for our interactive segmentation network trained with and without the scribble consistency loss. The user input scribble is typically ignored without this loss.

We show additional annotation examples on the EPIC-Kitchen dataset [2] using our annotation tool in Figure 4.

4 Annotation Tool

We demonstrate the step-by-step usage of our annotation tool in Fig 5. Please refer to demo video from 00:47s.

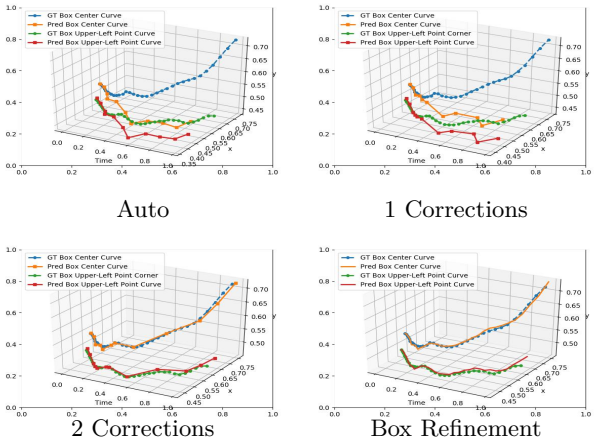


Fig. 2: This plot shows an example of how a box track gets refined with user's corrections. Results are reported on KITTI. Blue & green are ground-truth trajectories of the center and top-left box coordinate. Orange & red is the track from Curve-VOT. Only 2 corrections are required to produce an accurate track.

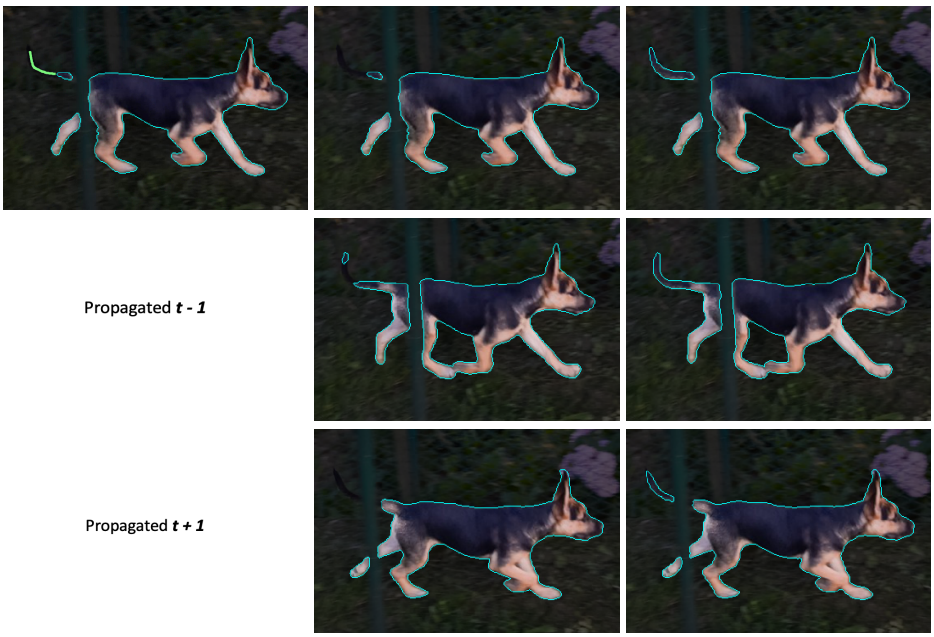


Fig. 3: Qualitative examples demonstrate the effectiveness of our scribble consistency loss. The second and third rows show neighbour frames propagated from t with and without scribble consistency loss respectively.

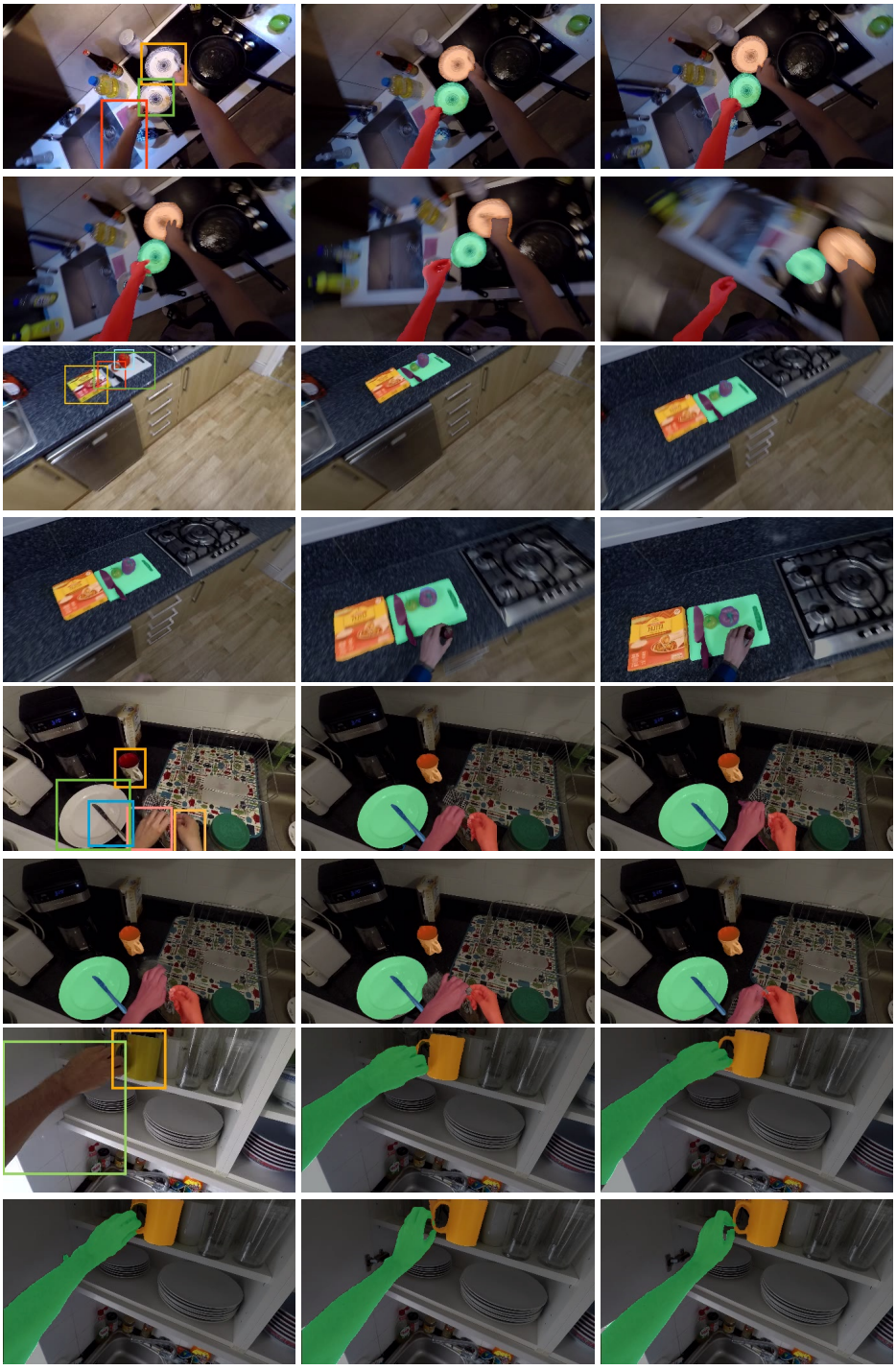
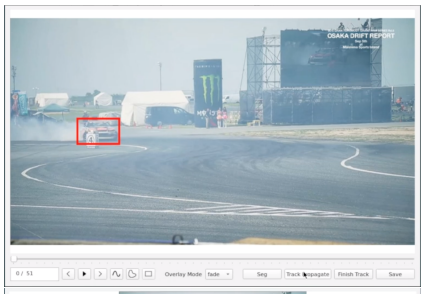
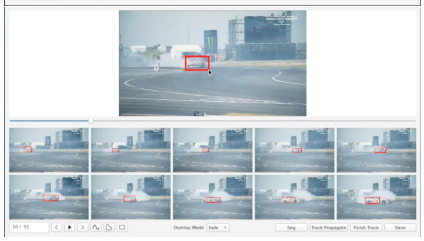


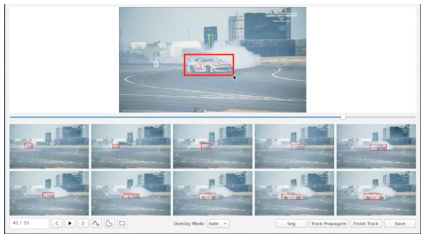
Fig. 4: Example annotations using our annotation tool in practice on the EPIC-Kitchen dataset. Each object in a 100-frame video requiring on average 67.1s of annotation time (including inference time). The first column in each two rows indicates target objects.



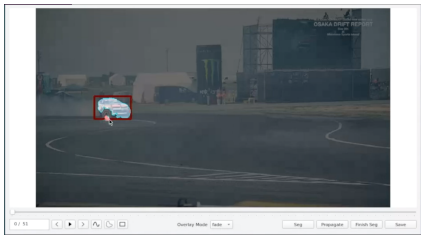
Step1: User draws a box at the first frame.



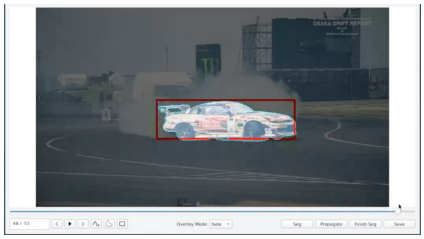
Step2: Auto tracking and pop out key frames (nearest frames to control points).



Step3: User spots a mistake in one of the keyframes and corrects it by drawing a new box. We then re-run the tracker and re-fit the curve.



Step4: User corrects the mask with scribbles, which are propagated to other frames.



Step5: Done!

Fig. 5: Step-by-step overview of Scribble-Box annotation tool. See video for seeing the tool being used in practice.

References

1. Y. Chen, J. Pont-Tuset, A. Montes, and L. V. Gool. Blazingly fast video object segmentation with pixel-wise metric learning. *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, Jun 2018.
2. D. Damen, H. Doughty, G. M. Farinella, S. Fidler, A. Furnari, E. Kazakos, D. Moltisanti, J. Munro, T. Perrett, W. Price, and M. Wray. Scaling egocentric vision: The epic-kitchens dataset. In *European Conference on Computer Vision (ECCV)*, 2018.
3. L. N. Smith. Cyclical learning rates for training neural networks. *2017 IEEE Winter Conference on Applications of Computer Vision (WACV)*, Mar 2017.
4. S. Wug Oh, J.-Y. Lee, K. Sunkavalli, and S. Joo Kim. Fast video object segmentation by reference-guided mask propagation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 7376–7385, 2018.