ScribbleBox: Interactive Annotation Framework for Video Object Segmentation

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

Abstract. Manually labeling video datasets for segmentation tasks is extremely time consuming. We introduce ScribbleBox, an interactive framework for annotating object instances with masks in videos with a significant boost in efficiency. In particular, we split annotation into two steps: annotating objects with tracked boxes, and labeling masks inside these tracks. We introduce automation and interaction in both steps. Box tracks are annotated efficiently by approximating the trajectory using a parametric curve with a small number of control points which the annotator can interactively correct. Our approach tolerates a modest amount of noise in box placements, thus typically requiring only a few clicks to annotate a track to a sufficient accuracy. Segmentation masks are corrected via scribbles which are propagated through time. We show significant performance gains in annotation efficiency over past work. We show that our ScribbleBox approach reaches 88.92% J&F on DAVIS2017 with an average of 9.14 clicks per box track, and only 4 frames requiring scribble annotation in a video of 65.3 frames on average.

1 Introduction

Video is one of the most common forms of visual media. It is used to entertain us (film, tv series), inform us (news), educate us (video lectures), connect us (video conferencing), and attract our interest via TV commercials and social media posts. Video is also a crucial modality for robotic applications such as self-driving cars, security applications, and patient monitoring in healthcare.

One of the fundamental tasks common to a variety of applications is the ability to segment and track individual objects across the duration of the video. However, the success of existing methods in this task is limited due to the fact that training data is hard to obtain. Labeling only a single object in a single frame can take up to a minute [7,8], thus annotating the full video is prohibitively time consuming. This fact also presents a major roadblock for video content editing where end-users may want to segment a particular object/person and replace the background with an alternative. Most film studios thus still mainly resort

⋆ authors contributed equally

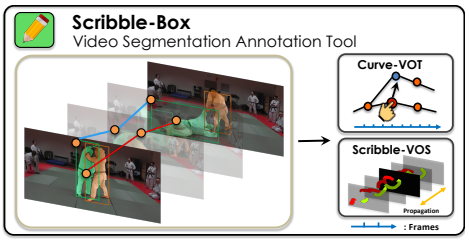


Fig. 1: **ScribbleBox**: Interactive framework for annotating object masks in videos. Our approach splits annotation into two steps: interactive box tracking where the user corrects keyframe boxes (Curve-VOT), and interactive segmentation inside tracked boxes with human-provided scribbles (Scribble-VOS).

to the use of "green-screen" during recording. Our interest here is in interactive annotation to assist human users in segmenting objects in videos.

Prior work on human assisted interactive annotation has addressed both the image [23,1,41,26,33,27] and video domains [9,4,24,18,6,30]. The prevailent image-based approaches employ grabCut-like techniques using scribbles as human feedback [33], track object boundaries via intelligent scissors [27], interactively edit predicted vertices of a polygon outlining the object [23,1], and providing human clicks on the erroneously predicted object boundary as a heatmap to a DeepLab-like architecture [26,41]. In video, methods either use online fine-tuning [29,22] of the model given a new user-annotated frame, or propose ways to propagate scribbles provided by the user in one frame to other frames. However, in current scribble propagation approaches a human-provided scribble typically only affects neighboring frames, and oftentimes the scribbles are ignored entirely by the neural network that processes them.

In this paper, we introduce ScribbleBox, a novel approach to interactive annotation of object instances in videos. In particular, we propose to split the task of segmenting objects into two simpler tasks: annotating and tracking a loose box around each object across frames, and segmenting the object in each tracked box. We make both steps interactive. For tracking, we represent object's motion using a sequence of linear motions. Our key contribution is to optimize for the control points defining these motions, and allow the user to interactively correct the erroneous control points. We perform local adjustments to allow for the boxes to slightly deviate from the piece-wise linearly interpolated track. To segment an object inside each tracked box, we exploit scribbles as a form of human input. We present a new scribble propagation network and a simulation scheme that encourages the network to accurately propagate human input through the video. In a user study, we show that our approach achieves about 4% higher J&F score compared to the current state-of-the-art interactive baseline IPN [30] given the same amount of annotation time.

Online demo and tool will be released at Project Page

2 Related Work

Literature on object tracking/segmentation is vast. Since automatic tracking and segmentation is not our contribution we limit our review to interactive methods.

Interactive Visual Object Tracking (VOT): In the traditional tracking annotation protocol, annotators are asked to determine keyframes and label the object's box in these frames. Other frames are then automatically annotated via either linear interpolation [15], or performing a shortest-path interpolation between manual annotations [37]. The annotator typically needs to go back and forth in determining the keyframes such that the final full track is accurate. To introduce intelligence in video annotation protocol and reduce cost, [38] exploits active learning to choose which frames to annotate. More recent and related to our method, PathTrack [25] presents an efficient framework to annotate tracking by tracing each object with a mouse cursor as the video plays. However, there is an ambiguity in determining the scale of the object which they try to automatically account for. In our work, the annotator does not need to watch the full video, and is only asked to inspect the automatically determined keyframes. While playing the in between segments in fast-forward mode is desired for verification, we experimentally show that no further inspection is typically needed, as our full method deals with a substantial amount of noise.

Interactive Video Object Segmentation (VOS): With the recently introduced datasets [6], there has been an increased interest in designing both automatic and interactive methods for video object segmentation. An interactive method expects user input, either clicks or scribbles, to refine the predicted output masks. For efficiency, human feedback is propagated to nearby frames.

Several earlier approaches employed graph cut techniques [39,32,20,3]. In [39], user's input spans three dimensions. Video is treated as a spatiotemporal graph and a preprocessing step is required to reduce the number of nodes for the min-cut problem. A much faster method is LIVEcut [32] where the user selects and corrects frames which are propagated forward frame by frame. In [2], an image with foreground and background scribble annotations is converted into a weighted graph, where the weights between nodes (pixels) are computed based on features like color or location. A pixel is classified based on its shortest geodesic distance to scribbles. JFS [28] propagates user annotations through the computed point trajectory and classifies the remaining pixels with a random walk algorithm. Modern methods [9,26,4,6,29,30] employ neural networks for interactive VOS. [9] formulates the segmentation task as a pixel-wise retrieval problem and supports different kinds of user input such as masks, clicks and scribbles. [6] proposes an Interactive Segmentation Track for the DAVIS2018 Challenge along with baselines. The first baseline is based on an online-learning VOS model OSVOS [5], while the second baseline trains a SVM classifier on pixels that are annotated with scribbles. Top performers in the challenge include SiamDLT [29] which performs similarity learning together with online fine-tuning on the user-provided scribbles and dense CRF as post-processing.

The winner of the DAVIS2018 Challenge, IPN [30] proposes an interactive framework by employing scribbles to correct masks and propagate corrections to the neighboring frames implicitly. Different from IPN, we incorporate tracking in the annotation process, and model scribble propagation explicitly: we directly predict scribbles for other frames based on the user's input, and employ a training

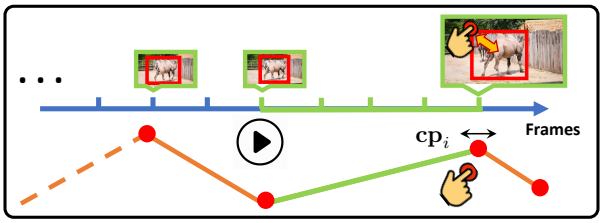


Fig. 2: **Curve-VOT:** Annotators inspect automatically determined keyframes and adjust control points of a parametric curve. They can fast-forward the video in between keyframes to ensure the object is adequately tracked.

strategy that encourages our network to utilize scribbles effectively. Note that our mask propagation module shares similarity with the recent mask propagation model Space-Time Memory Networks [31]. We employ Graph Convolutional Networks while they adopt a memory-network architecture.

3 Our Approach

The goal of our work is to efficiently label object tracks in videos with accurate masks. We split annotation into two steps: 1) box tracking, and 2) mask labeling given the tracked boxes. We argue that it is easier to segment the object inside cropped regions rather than the full image both for the human labeler as well as automatic methods as they can more explicitly exploit shape priors and correlations of motion across time.

We introduce automation and human-in-the-loop interaction in both stages, aiming to achieve the highest level of labeling efficiency. We represent the box track with a parametric curve (polygon or spline [12,23]) exploiting the fact that many motions are roughly linear locally. The curve is represented with a small set of control points which are obtained automatically. The annotator is then only shown the frames closest to the control points, and in case of errors a control point can be adjusted. We exploit this information to re-estimate the control points in the subsequent parts of the video. This process can be done iteratively. Importantly, the annotated tracked boxes obtained from this step can be quite loose, and mainly serve to roughly delineate where the object is in the frames. Our second step, the mask labeling step, is able to tolerate such noise and produces tracked masks with very few interactions with the annotator.

In the second annotation step, we first automatically label the object's masks. The annotator then inspects the video of the segmented object and provides feedback for the frames where errors occur by drawing scribbles in the erroneous areas. We automatically re-predict the frame and propagate this information to the subsequent frames. We explain both steps in more detail next.

3.1 Interactive Tracking Annotation

We here introduce *Curve-VOT*, our interactive approach for tracking annotation which tries to reduce the number of clicks required to annotate boxes around an

object of interest in a video. We approximate the trajectory of the box using a parametric curve which is estimated interactively. Specifically, we approximate a M-frame trajectory J as a polygonal curve P with a fixed number of control points, N. We obtain this curve automatically, and allow the annotator to edit the control points by moving them in space and time, and add or remove points.

Note that our approximation assumes that the motion of the box between two control points is linear. This assumption typically holds locally, but may be violated if the temporal gap between the two control points is large. We allow slight deviations by locally searching for a better box. We explicitly do not require this step to give us perfect boxes, but rather "good enough" region proposals. We defer the final accuracy to an automatic refinement step, and our interactive segmentation annotation module. Note that in practice $N \ll M$, resulting in a huge saving in annotation time. We illustrate our interface in Fig. 2.

We now describe how we obtain the curve for the object's trajectory. Our approach is optimization based: given the last annotated frame, we track the object using any of the existing trackers. In our work, we employ SiamMask [35] for its speed and accuracy. We then take the tracked box and fit a polygonal curve with N control points.

Let $\mathbf{cp}_i = [t_i, x_i, y_i, w_i, h_i]^T$ denote the i^{th} control point, where t represents continuous time, and $[x_i, y_i, w_i, h_i]$ denotes the center of the box and its width and height, respectively. Let $P = \{\mathbf{cp}_1, \mathbf{cp}_1, \cdots, \mathbf{cp}_N\}$ to be the sequence of all control points. Note that our curve is continuous and parametric, i.e., $P(t)$, with $t_i \leq t \leq t_{i+1}$, we define $P(t) = [(t - t_i)\mathbf{cp}_i + (t_{i+1} - t)\mathbf{cp}_{i+1}]/(t_{i+1} - t_i)$.

Curve Fitting: To fit the curve to the observed boxes (obtained by the tracker), we initialize the control points by uniformly selecting key frames and placing points in the center of each frame, and run optimization. We uniformly sample K points along both the parametric curve P and the observed object track J, and optimize the following cost:

$$L_{\text{match}}(\{\mathbf{cp}_i\}) = \sum_{k=1}^{K} \|[t_k^P, x_k^P, y_k^P, w_k^P, h_k^P]^T - [t_k^J, x_k^J, y_k^J, w_k^J, h_k^J]^T\|_1 \quad (1)$$

Note that each $[t_k^P, x_k^P, y_k^P, w_k^P, h_k^P]^T$ is a linear function of its neighboring control points, allowing us to compute the gradients with respect to $\{\mathbf{cp}_i\}$. We use gradient descent to optimize our objective. In our experiments, we choose $N = 10$ and we set K to be 300 to ensure number of sampled points is greater than number of frames. We run optimization for 100 steps.

Interactive Annotation: We pop the frames closest to the estimated control points, and allow the user to make adjustments if necessary. In particular, each control point can be adjusted both in space and time to accurately place its corresponding box around the object. The user is also asked to fast-forward the video between the control points, making sure that the object is loosely tracked in each interval. In case the interpolation loses the object, the annotator is able to add an additional control point. We then take the last annotated frame and re-run the tracker in the subsequent video. In our case, the last annotated box is used to compute correlation using SiamMask in the following frames. We re-fit

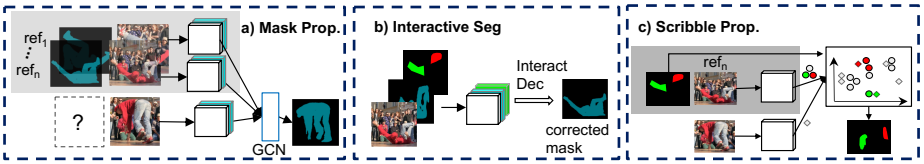


Fig. 3: Modules in our model: **a)** Given user annotated reference frames, our mask propagation module predicts object masks in subsequent frames. **b)** Interactive segmentation module takes user scribbles as input (red and green scribbles indicate false positive and false negative regions, respectively), and returns the corrected mask for the chosen frame. **c)** We propagate scribbles to correct masks in nearby frames.

the remaining control points to the subsequent track. The interactive process is repeated until the annotator deems the object to be well tracked.

Box Refinement: After interactive annotation, we get a curve which models piece-wise linear motions. To refine the linearly interpolated box in the frames in between every two control points, we crop the images based on the size of the box following SiamRPN [19] and re-run the SiamMask tracker by constraining it to these crops. This in practice produces significantly more accurate boxes that we exploit further in our interactive segmentation module.

3.2 Interactive Segmentation Annotation

Given the annotated tracked box from Section 3.1, we now aim to interactively segment the object across the video with minimal human labour. We make use of the SiamMask's mask prediction in the first frame and predict masks for the rest of the video by our *mask propagation module*. If errors occur, the user can edit the inaccurate masks by drawing positive and negative scribbles. Here, positive scribbles indicate missing regions, while negative scribbles indicate false positive regions. To best utilize human's feedback, our method propagates both the corrected mask as well as the input scribbles to the subsequent frames. We refer to our interactive segmentation module as *Scribble-VOS*.

Network Design Our interactive segmentation approach consists of three different modules: a *mask propagation module* to propogate segmentation masks, an *interactive segmentation module* for single image editing via scribbles, and a *scribble propagation module* that propagates user's feedback across video. Overview is in Fig 3. We explain each module in detail next.

Image Encoder: We crop the image inside the box and encode it with ResNet50 [14] pre-trained on ImageNet. We extract image features from the Conv4 layer and denote it as $F_t \in \mathbb{R}^{W \times H \times D}$, where t is the time step in the video, and W and H are the width and height of the feature map. Note that unlike most previous works that concatenate images with additional information including masks/scribbles as input to the encoder, our encoder only takes images

as input, which allows us to share the encoder among different modules and use an encoder pre-trained on any (large) image dataset.

Mask Decoder: There are two decoders in our framework. *Interactive decoder* is used to produce a refined mask of single frame based on the human interaction. We also have a *propagation decoder* which is shared by both the mask and scribble propagation modules. We describe the structure of the decoder next, and specify the input feature for the two decoders later in the section.

Our decoder consists of three refinement heads [42] that upsample image features (1/16 resolution wrt input) by 8x to produce the object mask. Sizes of the output channels of the refinement heads are 224, 224, 128. We also remove the skip connection for the last refinement head as it yields better results.

Interactive Segmentation Module: To correct errors produced by the model, our interactive module takes user's scribbles for the chosen frame and outputs a refined mask. Specifically, we first convert user's scribbles into two binary maps, one for positive and the other for negative scribbles. The input to the *interactive decoder* concatenates 2-channel scribble input, image features, and the mask of the frame that we want to correct, as shown in Fig. 3b.

To force the model to behave consistently with the user's scribbles, we utilize a *scribble consistency loss*. And, to prevent the predicted mask from being modified in regions where the initial mask was accurate, we restrict the network to only affect a local neighbourhood surrounding the scribbles. In particular, we mask out new predictions that are more than 10 pixels away from the scribble areas.

Note that in [30], both scribbles and masks are encoded along with the raw image, which requires an additional encoding for each interaction. Here, we only run the encoding once and re-use the image feature for each interaction.

Mask Propagation Module: Given the user-annotated object masks, our mask propagation module aims at re-predicting the masks for the subsequent frames. We rely on a Graph Convolutional Network (GCN) [17] to perform this propagation, and additionally employ a decoder which produces the segmentation mask based on the GCN features. Let R denotes the set of user-annotated frames, which serve as reference frames for propagation, and let c denotes the current frame that we wish to predict the mask for. We build a graph $G = (V, E)$, where V denotes nodes and E denotes edges to encode the structure between frames in R and the current frame c. In particular, we make every location in the $W \times H$ feature map of every frame a node in the graph, i.e., there are $(|R| + 1) \times W \times H$ nodes in total. Each node in frame c is connected to all the nodes in frames $\in R$ via an edge. An example of the graph is illustrated in Fig. 3a.

As input to our GCN, we concatenate the image feature with corresponding masks for each frame in R. We perform a similar concatenation operation for c but use a uniform "mask" instead. The input feature f_u for vertex u in GCN is computed as $f_u = \text{concat}\{F(x, y), M(x, y)\} \in \mathbb{R}^{(D+1)}$, where F and M are the image feature and the corresponding mask at time step t_u, respectively, and D is dimension. Here, (x, y) is the coordinate of the node u in the feature map.

We assign a weight $w_e(u, v) = \frac{\exp(f_u \cdot f_v)}{\sum_{v' \in \mathcal{N}(u)} \exp(f_u \cdot f_{v'})}$ to each edge (u, v) in the graph. The following propagation step is performed for node u:

$$f'_u = W_0 f_u + \sum_{v \in \mathcal{N}(u)} w_e(u,v) W_1 f_v, \tag{2}$$

where $\mathcal{N}(u)$ denotes neighbours of u in the graph, and W_0, W_1 are the weight matrices of the GCN layers. We take the feature $f'_u \in \mathbb{R}^K$ for all nodes u in current frame c as the input feature $F_c^g \in \mathbb{R}^{W \times H \times D'}$ to the *propagation decoder* which predicts the refined mask. We use D' to denote the output dimension of the GCN feature, and it is also the dimension of the input feature for the decoder. The superscript g stands for GCN. The feature, F_c^g, is used by the Scribble Propagation module described next.

Scribble Propagation Module: Most existing works employ user's corrections by only propagating the annotated masks to other frames. We found that propagating the scribbles explicitly produces notably better results. By feeding scribble information explicitly to the network, we are providing useful information about the regions it should pay attention to. Qualitatively, without providing this information (see Fig. 9), we notice that the propagation module typically ignores the corrections and repeats its mistakes in the subsequent frames.

As shown in Fig. 3c, inspired by [9], we formulate scribble propagation problem as a pixel-wise retrieval problem. For each pixel in current frame c, we find the pixel in the reference frame with the most similar embedding and assign the label of that pixel to the pixel in c. In the scribble propagation stage, we choose the reference frame as the annotated frame that is closest to the current time step. Formally, we first project the encoded pixel features into an embedding space and adopt the label of the nearest neighbour in this space. Pixels in the reference frame can be classified into three classes: background (regions with no scribbles), negative, and positive scribbles, respectively. We first project image features $F \in \mathbb{R}^{W \times H \times D}$ of the reference and current frames into a $d < D$ dimensional space using a shared embedding head. We use $d = 128$ and a 3×3 convolutional layer for the projection. We denote the transferred label for frame c as $S_c \in \mathbb{R}^{W \times H \times 2}$. To get the final mask, we concatenate S_c with F_c^g, i.e., the feature obtained from the GCN, reduce the dimension of the concatenation to D', and then feed it to the *propagation decoder*. Here, we employ a 3×3 convolution layer to perform the dimension reduction. Thus, the input feature to the *propagation decoder* is

$$F_c^{gs} = M(\text{concat}\{F_c^g, S_c\}) \in \mathbb{R}^{W \times H \times D'}, \tag{3}$$

where M is the conv layer for dimensionality reduction. Superscript gs denotes merging GCN's output and the predicted scribbles. The *propagation decoder* is shared for mask and scribble propagation. The decoder takes the encoded feature (either the feature F_c^g from the mask propagation module or the feature F_c^{gs} from the scribble propagation stage), and outputs an object mask. This scribble network propagation is designed for propagating local errors within a short range and thus is different in purpose from the mask propagation module.

Network Training: All networks are trained end-to-end using binary cross entropy loss for mask prediction. To better supervise our model, we further use

Correction Rounds	0	4	6	9
IPN-Box & Line-Scrib.	76.93	78.89	79.73	80.88
GCN Mask Prop. & Line-Scrib.	79.33	86.70	87.48	88.48
GCN Mask Prop. & Area-Scrib.	79.33	87.62	89.07	89.75
+ Scribble Prop.	-	88.92	90.90	91.16
+ GT First Frame	85.14	89.61	90.28	90.91

Table 1: **Ablation study** (DAVIS'17). Round 0 means mask prop. with predicted first frame mask.

Model	IPN	Ours-256	Ours-512
Interaction	18	10.5	12
Curve Fitting	-	12	12
Mask Prop.	15	29	54
w/Cache	-	8	26
Scribble Prop.	-	15	28

Table 2: **Running Time**. Numbers are reported in ms per frame.

Scribble Consistency Loss for training the *interactive segmentation module*, and Batch Hard Triplet Loss [9] for training the *scribble propagation module*. We first describe how we synthesize scribbles and then specify the scribble consistency loss in the following paragraphs. We defer the Batch Hard Triplet Loss and implementation details to the appendix.

Scribble Correspondence: A big challenge of training the scribble propagation module is the lack of ground-truth pixel-wise correspondences between the scribbles in frame r and scribbles in frame c. To solve this problem, we create the ground-truth correspondences by synthesizing new frames: we augment the reference frame r to obtain the current frame c. The same augmentation is applied on the scribble map which gives us ground-truth correspondences between scribbles. Specifically, we use thin-plate spline transformation with 4 control points for the augmentation.

Scribble Consistency Loss: Typically, a user, who draws the scribble, would expect the network to behave consistently with the annotation, i.e., expects to see positive predictions around positive scribbles and vice versa for the negative. To encourage this behavior, we employ a scribble consistency loss:

$$L_{\text{sc}} = \sum_{i,j} -S_{\text{p}}(i,j) \log M_{\text{pred}}(i,j) - S_{\text{n}}(i,j) \log(1 - M_{\text{pred}}(i,j)), \tag{4}$$

where S_{p} and S_{n} are the positive and negative scribble maps, (i,j) is the coordinate on the mask. Here, M_{pred} denotes the predicted object mask. Without this loss, the model in many cases ignores the user's input.

4 Experimental Results

We perform extensive evaluation of ScribbleBox for video object annotation. To test generalization capabilities, we evaluate our approach both in the same domain, as well as in the out-of-domain datasets. For the In-Domain experiment, following IPN [30], we show results on DAVIS2017 [16] validation set, which has 65.3 frames per object on average. For the Out-of-Domain experiment, we showcase our results on MOTS-KITTI [36] which is a recent multi-object tracking and segmentation subset of KITTI [13]. We evaluate on a subset of 514 objects from the training and validation set where video length is between 20 and 150 frames. The average video length is 48.8 frames per object. We also perform a user study with real annotators labeling videos with our annotation tool.

Fig. 4: Qualitative results on DAVIS2017 val set using our full annotation framework. 5 rounds of scribble correction + 8.85 box corrections were used.

Baseline: We use IPN [30] as our baseline, which ranks first in DAVIS'18 Interactive Challenge. Note that the official benchmark is not applicable in our case since our method is a two-stage method and the DAVIS-Agent does not support bounding box correction. Thus, we evaluate against the baseline by reporting box correction and scribble drawing effort v.s J&F via simulation (Table 1, Fig. 6) and annotation time v.s J&F through a user study (Fig. 7).

Scribbles: DAVIS Interactive Challenge [6] defines scribbles as thin skeletons inside erroneous areas, which we refer to as Thin-Line-Scribble. We instead propose to use more informative scribbles, where we ask the user to roughly trace the erroneous area with a cursor. We name our scribbles as Area-Scribble. While this may require slightly more effort, we show significant performance gains in our ablation study for both types of scribbles.

To simulate the Area-Scribble, given a previously predicted object mask and the ground-truth mask, we sample positive and negative scribbles from the false negative and false positive areas, respectively. To simulate realistic "trace" scribbles that a human would provide, we perform a binary erosion to the erroneous area and take the resulting regions as our simulated scribbles.

Details: We use the official SiamMask [40] tracker provided by the authors which was trained on COCO [21], ImageNetVID [34] and YouTubeVOS [43]. We perform inference for each object in the video. In Scribble-VOS, we propagate the corrected mask using mask propagation network to the end of the video. In addition, we propagate scribbles in two directions (forward and backward) for at most n frames (we use $n = 5$ in experiments).

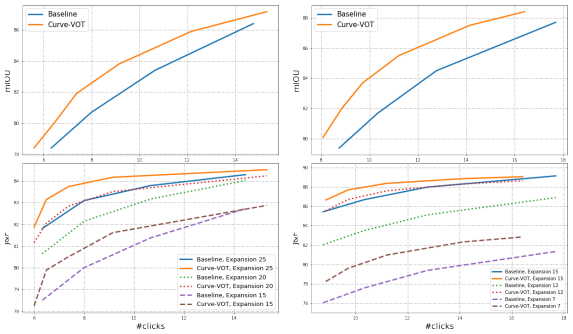
4.1 In-Domain Annotation

Tracking Annotation: We compare Curve-VOT with a brute-force baseline which also uses SiamMask [40]. Instead of manipulating trajectories via curves, the user watches the video and corrects a box when it significantly deviates from the object. We use this correction as a new reference and run the tracker again starting from this frame. We simulate the user by correcting frames with IOU

lower than a threshold, where we evaluate different thresholds from 0.4 to 0.8, representing different effort levels by the users. We treat each user's correction as two clicks, i.e., representing two corners of the box.

We first compare our approach with the baseline in terms of box IOU. As shown in the top-left of Fig. 5, our Curve-VOT requires fewer clicks to achieve the same IOU. In the bottom-left of Fig. 5, we provide a comparison in terms of segmentation J&F. We take the corrected bounding boxes and expand them to crop image. We report results with different expansion sizes. Since tracking the box only acts as a region proposal for VOS, we evaluate tolerance for tracking noise. As shown in the bottom-left of Fig. 5, using a 25 pixels expansion, J&F converges at around 9 clicks. This plot also emphasizes the importance of Scribble-VOS since additional box clicks only increase the accuracy marginally, which is a wasted effort. We instead turn to Scribble Annotation to further improve results.

Mask Annotation: We first conduct an ablation study on individual components in Scribble-VOS in Table 1. Following IPN [30], we use the ground truth box in the first frame. To remove the effect of box annotation, we modify the IPN baseline to work on boxes. We name this baseline as IPN-box, where we use the officially released code from [30] and take box-cropped image as input. Same as our model, we first pre-train IPN-box on synthetic video clips, then fine-tune it on DAVIS2017. We add GCN-Mask Propagation module and multi-reference frames for interactions to the IPN-box baseline. It gives 2.4%, 7.81%, 7.75%, 7.6% improvements, respectively. To ablate our human interaction, we replace Thin-Line-Scribble with Area-Scribble. Our Area-Scribble yields 0.92%, 1.59% and 1.33% improvements, respectively. Results of ScribbleBox without scribble propagation also indicate that our *scribble propagation module* plays an important role in improving J&F (1.3%, 1.83%, and 1.41%, respectively). We further analyze impact on quality of the first frame's mask. As shown in the last line of Table 1, ground truth first frame mask gives 5.81% improvement without interactions but the improvement becomes marginal after more rounds of interaction. These demonstrate the effectiveness of our interaction model.



(DAVIS2017) (KITTI)

Fig. 5: **Interactive-VOT**: (top) Box IoU vs number of simulated clicks (we report IoU averaged across frames), (bottom) Segmentation J&F. Note that in this experiment segmentation is automatic using annotated boxes, i.e., there are no clicks to improve masks, only to improve boxes.

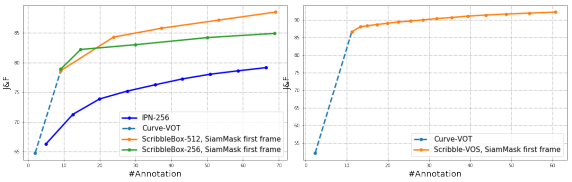


Fig. 6: **Simulated** full annotation workflow on (left) DAVIS2017, (right) KITTI. Dashed blue lines denote Curve-VOT, continuing lines Scribble-VOS annotation.

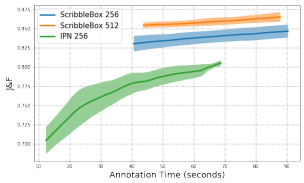


Fig. 7: Real annotator **user study** on DAVIS'17. Filled area denotes variance.

Fig. 8: Qualitative results on DAVIS2017 val. **Auto**: SiamMask, **Box**: Results after box correct. (Curve-VOT), **Scribble**: Results after Scribble-VOS.

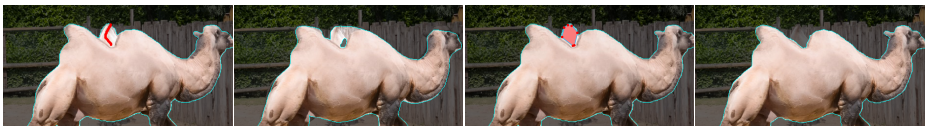


Fig. 9: Qualitative example demonstrating **effectiveness of our scribble propagation**. a) & c): user draws a scribble in frame t. Note that IPN uses skeleton-like scribbles while we use trace scribbles. b) & d): results after scribble propagation at frame t+1.

Full Framework Comparison: We conduct a full framework analysis in a simulated environment. We compare with previous work in terms of the number of human interactions. For a baseline, we run the officially released DAVIS Interactive agent locally and evaluate upon IPN's checkpoint which achieved the first place in DAVIS2018 Interactive Challenge. Each correction round returned by the DAVIS-Agent consists of multiple scribbles. We count each scribble as one annotation. Our results are conducted on the annotated tracked boxes after 9.14 box clicks (J&F 78.64%, i.e., second point on the dashed line in Fig. 6). Fig. 6 reports a comparison of our framework with IPN [30]. Note that our model with only Curve-VOT interaction already performs better than baseline with mask annotation. We also ablate the impact of different image resolutions.

Qualitative Results: We show qualitative results on DAVIS2017 val in Fig. 4. All results shown are annotated via Curve-VOT (9.14 clicks on average) following 5 rounds of scribble correction. Qualitative examples are in Fig. 8.

Results indicate that Curve-VOT corrects large errors and Scribble-VOS further refines them. Fig. 9 shows an example indicating the importance of our scribble propagation and differences between Thin-Line-Scribble and our Area-Scibble.

4.2 Out-of-Domain Annotation

We now run inference of our model on an unseen dataset (MOTS-KITTI). As shown in Fig. 5(right), Curve-VOT outperforms the baseline by a large margin. Note that objects in KITTI are typically smaller than DAVIS, and so we adopt a smaller expansion size. Qualitative interactive tracking results are shown in Fig. 10 and Fig. 11. The car with blue mask in Fig. 11 also shows our intuition about why we need two-stage annotation. Curve-VOT first ensures that the intended annotated object is not lost. This usually happens when multiple small and similar objects are adjacent.

4.3 User Study

We put our framework to practice and show annotation results using human annotators with a simple tool that runs our models in the backend. All human experiments were done on the same desktop with a Nvidia-Titan Xp GPU.

In-Domain: We conduct a user study on the DAVIS2017 validation set. We randomly select one object per video which adds up to 1999 frames including blank frames. We employed four in-house annotators. Each was asked to annotate the same videos using both the baseline (IPN) and our method. For our method, we evaluate two models trained and tested at different resolutions: 256 (same as IPN) and 512, and we refer to them as ScibbleBox-256, ScibbleBox-512, respectively. For fairness, annotators first annotate with our model so that they gain familiarity with the data before using the baseline method. We ask annotators to annotate until there is no visible improvement. We show the mean curve for J&F v.s annotation time in seconds in Fig. 7. We include the data processing and model running times in the cumulative annotation time. We use filled area to denote variance between annotators. Note that the starting point of our model is to the right of our baseline because we only calculate J&F after Curve-VOT corrections and their corresponding J&F have already outperformed baseline's converging performance by a large margin. We also calculate mean J&F in the overlapping time interval, which gives IPN: 79.25%, ScribbleBox-256: 83.62% and ScribbleBox-512: 85.66%.

We further report model running times in Table 2. We report times with feature cache as "Mask Prop. w/Cache". Although our model with 512-resolution is slower, it outperforms both 256-resolution model and baseline given the same annotation time budget.

Annotating new datasets: We annotate a subset of the EPIC-Kitchen [10,11] dataset using our tool. As shown in Fig. 12, without any finetuning on out of domain data, our method generates high-quality masks with only a few annotations. We plan to annotate and make available a large portion of the dataset.

Fig. 10: Qualitative results on MOTS-KITTI [36] using our full annotation framework. 5 rounds of scribble correction + 11.2 box corrections were used.

Fig. 11: Qualitative results on MOTS-KITTI [36]. **Auto**: SiamMask, **Box**: Results after box correct. (Curve-VOT), **Scribble**: Results after Scribble-VOS.

Fig. 12: Example annotations using our annotation tool on the EPIC-Kitchen dataset. Each object in a 100-frame video requiring on average 69.87s of annotation time (including inference time). The first column indicates target objects.

5 Conclusion

We introduced a novel video annotation framework that splits annotation into two steps: annotating box tracks, and labeling masks inside these tracks. Box tracks are annotated efficiently by approximating the trajectory using a parametric curve with a small number of control points which the annotator can interactively correct. Segmentation masks are corrected via scribbles which are efficiently propagated through time. We showed significant performance gains in annotation efficiency over past work in two major benchmarks.

Acknowledgments. This work was supported by NSERC. SF acknowledges the Canada CIFAR AI Chair award at the Vector Institute.

References

1. D. Acuna, H. Ling, A. Kar, and S. Fidler. Efficient interactive annotation of segmentation datasets with polygon-rnn++. In *CVPR*, 2018.
2. X. Bai and G. Sapiro. A geodesic framework for fast interactive image and video segmentation and matting. In *IEEE 11th International Conference on Computer Vision, ICCV 2007, Rio de Janeiro, Brazil, October 14-20, 2007*, pages 1–8. IEEE Computer Society, 2007.
3. X. Bai, J. Wang, D. Simons, and G. Sapiro. Video snapcut: robust video object cutout using localized classifiers. *ACM Trans. Graph.*, 28(3):70, 2009.
4. A. Benard and M. Gygli. Interactive video object segmentation in the wild. *ArXiv*, abs/1801.00269, 2018.
5. S. Caelles, K. Maninis, J. Pont-Tuset, L. Leal-Taixé, D. Cremers, and L. V. Gool. One-shot video object segmentation. In *2017 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017, Honolulu, HI, USA, July 21-26, 2017*, pages 5320–5329. IEEE Computer Society, 2017.
6. S. Caelles, A. Montes, K.-K. Maninis, Y. Chen, L. Van Gool, F. Perazzi, and J. Pont-Tuset. The 2018 davis challenge on video object segmentation. *arXiv:1803.00557*, 2018.
7. L. Castrejon, K. Kundu, R. Urtasun, and S. Fidler. Annotating object instances with a polygon-rnn. In *CVPR*, 2017.
8. L.-C. Chen, S. Fidler, A. Yuille, and R. Urtasun. Beat the mturkers: Automatic image labeling from weak 3d supervision. In *CVPR*, 2014.
9. Y. Chen, J. Pont-Tuset, A. Montes, and L. Van Gool. Blazingly fast video object segmentation with pixel-wise metric learning. In *Computer Vision and Pattern Recognition (CVPR)*, 2018.
10. 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.
11. D. Damen, H. Doughty, G. M. Farinella, S. Fidler, A. Furnari, E. Kazakos, D. Moltisanti, J. Munro, T. Perrett, W. Price, and M. Wray. The epic-kitchens dataset: Collection, challenges and baselines. *IEEE Trans. Pattern Anal. Mach. Intell.*, 2020.
12. J. Gao, C. Tang, V. Ganapathi-Subramanian, J. Huang, H. Su, and L. J. Guibas. Deepspline: Data-driven reconstruction of parametric curves and surfaces. *arXiv preprint arXiv:1901.03781*, 2019.
13. A. Geiger, P. Lenz, and R. Urtasun. Are we ready for autonomous driving? the kitti vision benchmark suite. In *2012 IEEE Conference on Computer Vision and Pattern Recognition*, pages 3354–3361, June 2012.
14. K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.
15. Jenny Yuen, B. Russell, Ce Liu, and A. Torralba. Labelme video: Building a video database with human annotations. In *2009 IEEE 12th International Conference on Computer Vision*, pages 1451–1458, Sep. 2009.
16. A. Khoreva, A. Rohrbach, and B. Schiele. Video object segmentation with language referring expressions. In *ACCV*, 2018.
17. T. N. Kipf and M. Welling. Semi-supervised classification with graph convolutional networks. In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*, 2017.
18. E. Levinkov, J. Tompkin, N. Bonneel, S. Kirchhoff, B. Andres, and H. Pfister. Interactive multicut video segmentation. In *PG 2016*, 2016.

19. B. Li, J. Yan, W. Wu, Z. Zhu, and X. Hu. High performance visual tracking with siamese region proposal network. In *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 8971–8980, June 2018.
20. Y. Li, J. Sun, and H. Shum. Video object cut and paste. *ACM Trans. Graph.*, 24(3):595–600, 2005.
21. T.-Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. L. Zitnick. Microsoft coco: Common objects in context. *Lecture Notes in Computer Science*, page 740–755, 2014.
22. Z. Lin, J. Xie, C. Zhou, J. Hu, and W. Zheng. Interactive video object segmentation via spatio-temporal context aggregation and online learning. *The 2019 DAVIS Challenge on Video Object Segmentation - CVPR Workshops*, 2019.
23. H. Ling, J. Gao, A. Kar, W. Chen, and S. Fidler. Fast interactive object annotation with curve-gcn. In *CVPR*, June 2019.
24. S. Mahadevan, P. Voigtlaender, and B. Leibe. Iteratively trained interactive segmentation. *arXiv preprint arXiv:1805.04398*, 2018.
25. S. Manen, M. Gygli, D. Dai, and L. Van Gool. Pathtrack: Fast trajectory annotation with path supervision. *arXiv:1703.02437*, 2017.
26. K.-K. Maninis, S. Caelles, J. Pont-Tuset, and L. Van Gool. Deep extreme cut: From extreme points to object segmentation. In *CVPR*, 2018.
27. E. N. Mortensen and W. A. Barrett. Intelligent scissors for image composition. In *SIGGRAPH*, pages 191–198, 1995.
28. N. S. Nagaraja, F. R. Schmidt, and T. Brox. Video segmentation with just a few strokes. In *2015 IEEE International Conference on Computer Vision, ICCV 2015, Santiago, Chile, December 7-13, 2015*, pages 3235–3243. IEEE Computer Society, 2015.
29. M. Najafi, V. Kulharia, T. Ajanthan, and P. H. S. Torr. Similarity learning for dense label transfer. *The 2018 DAVIS Challenge on Video Object Segmentation - CVPR Workshops*, 2018.
30. S. W. Oh, J.-Y. Lee, N. Xu, and S. J. Kim. Fast user-guided video object segmentation by interaction-and-propagation networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 5247–5256, 2019.
31. S. W. Oh, J.-Y. Lee, N. Xu, and S. J. Kim. Video object segmentation using space-time memory networks. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 9226–9235, 2019.
32. B. L. Price, B. S. Morse, and S. Cohen. Livecut: Learning-based interactive video segmentation by evaluation of multiple propagated cues. In *IEEE 12th International Conference on Computer Vision, ICCV 2009, Kyoto, Japan, September 27 - October 4, 2009*, pages 779–786. IEEE Computer Society, 2009.
33. C. Rother, V. Kolmogorov, and A. Blake. Grabcut: Interactive foreground extraction using iterated graph cuts. In *SIGGRAPH*, 2004.
34. O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, A. C. Berg, and L. Fei-Fei. ImageNet Large Scale Visual Recognition Challenge. *International Journal of Computer Vision (IJCV)*, 115(3):211–252, 2015.
35. R. Tao, E. Gavves, and A. W. M. Smeulders. Siamese instance search for tracking. *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Jun 2016.
36. P. Voigtlaender, M. Krause, A. Osep, J. Luiten, B. B. G. Sekar, A. Geiger, and B. Leibe. Mots: Multi-object tracking and segmentation, 2019.
37. C. Vondrick, D. Patterson, and D. Ramanan. Efficiently scaling up crowdsourced video annotation. *Int. J. Comput. Vision*, 101(1):184–204, Jan. 2013.

38. C. Vondrick and D. Ramanan. Video annotation and tracking with active learning. In *Proceedings of the 24th International Conference on Neural Information Processing Systems*, NIPS'11, pages 28–36, USA, 2011. Curran Associates Inc.
39. J. Wang, P. Bhat, A. Colburn, M. Agrawala, and M. F. Cohen. Interactive video cutout. *ACM Trans. Graph.*, 24(3):585–594, 2005.
40. Q. Wang, L. Zhang, L. Bertinetto, W. Hu, and P. H. Torr. Fast online object tracking and segmentation: A unifying approach. In *CVPR*, 2019.
41. Z. Wang, H. Ling, D. Acuna, A. Kar, and S. Fidler. Object instance annotation with deep extreme level set evolution. In *CVPR*, 2019.
42. 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.
43. N. Xu, L. Yang, Y. Fan, J. Yang, D. Yue, Y. Liang, B. Price, S. Cohen, and T. Huang. Youtube-vos: Sequence-to-sequence video object segmentation. *Lecture Notes in Computer Science*, page 603–619, 2018.