Supplementary Material
Tracking Anything with Decoupled Video Segmentation

Ho Kei Cheng[1†] Seoung Wug Oh[2] Brian Price[2] Alexander Schwing[1] Joon-Young Lee[2]
[1]University of Illinois Urbana-Champaign [2]Adobe Research
{hokeikc2,aschwing}@illinois.edu, {seoh,bprice,jolee}@adobe.com

This appendix is structured as follows:

- We first provide implementation details of our temporal propagation network (Section A).
- We then analyze the class-agnostic training data of the temporal propagation network (Section B).
- After that, we list additional details regarding our experimental settings and results (Section C).
- Next, we provide results on the small-vocabulary YouTube-VIS [35] dataset for reference (Section D).
- Lastly, we present qualitative results (Section E).

A. Implementation Details of Temporal Propagation

A.1. Overview

Recall that the temporal propagation model $\text{Prop}(\mathbf{H}, I)$ takes a set of segmented frames (memory) $\mathbf{H}$ and a query image I as input, and segments the query frame with the objects in the memory. For instance, $\text{Prop}\left(\{I_1, \mathbf{M}_1\}, I_2\right)$ propagates the segmentation $\mathbf{M}_1$ from the first frame I_1 to the second frame I_2. The memory $\mathbf{H}$ is a compact representation computed from all past segmented frames.

In our implementation, we adopt the design of the internal memory $\mathbf{H}$ from the recent Video Object Segmentation (VOS) approach XMem [6]. VOS algorithms are initialized by a first-frame segmentation (in our case, the first in-clip consensus output), and segment new incoming query frames. XMem is an online algorithm that maintains an internal feature memory representation $\mathbf{H}$. For each incoming frame I, it computes a query representation which is used to read from the feature memory. It then uses the memory readout $\mathbf{F}$ to segment the query frame. The segmentation result $(\text{Prop}(\mathbf{H}, I))$ is used to update the internal representation $\mathbf{H}$. With an internal memory management mechanism [6], this design has a bounded GPU memory cost with respect to the number of processed frames which is suitable for processing long video sequences.

We refer readers to [6] for details regarding XMem. We describe core details below for completeness. We make a few technical modifications to XMem to increase robustness in our generalized setting, which we also document below. We provide the full code at hkchengrex.github.io/Tracking-Anything-with-DEVA.

A.2. Network Architecture

The temporal propagation network consists of four network modules: a *key encoder*, a *value encoder*, a *mask decoder*, and a *Convolutional Gated Recurrent Unit (Conv-GRU)* [9].

The *key encoder*, implemented with a ResNet-50 [12], takes an image as input and produces multi-scale features at the first (stride 4), second (stride 8), and third (stride 16) stages. The fourth stage is discarded. The feature in the third stage is projected to a ‘key’, which is used for querying during memory reading. After segmentation, if we decide to add the segmented query frame into the memory $\mathbf{H}$, we will re-use this ‘key’ in the memory.

The *value encoder*, implemented with a ResNet-18 [12], takes an image and a corresponding object mask as inputs and produces a ‘value’ representation as part of the memory. We discard the fourth stage and only use the stride-16 output feature in the third stage. Objects are processed independently (done in mini-batches during inference).

The *mask decoder* takes the memory readout $\mathbf{F}$ and multi-scale skip connections from the key encoder as inputs and produces an object mask. It consists of three upsampling blocks. Each upsampling block uses the output from the previous layer as input, upsamples it bilinearly by a factor of two, and fuses the upsampled result with the skip connection at the corresponding scale with a residual block with two 3×3 convolutions. A 3×3 convolution is used as the last output layer to generate a single-channel (stride 4) logit and bilinearly upsamples it by four times to the original resolution. Similar to the value encoder, objects are processed independently, which can be done in mini-batches during

[†]This work was done during an internship at Adobe Research.

inference. Soft-aggregation [23] is used to combine logits for different objects as in [6].

The *Convolutional Gated Recurrent Unit (Conv-GRU)* [9] takes the last hidden state and the output of every upsampling block in the mask decoder as input and produces an updated hidden state. 3×3 convolutions are used as projections in the Conv-GRU.

Our Modifications. Firstly, in XMem [6], the 1024-channel third-stage feature from the key encoder is directly concatenated with the memory readout for mask decoding. For efficiency, we instead first project the 1024-channel feature to 512 channels with a 1×1 convolutional layer before concatenating it with the memory readout. Secondly, in each upsampling block of the mask decoder, XMem uses a 3×3 convolution to pre-process the skip-connected feature. We replace it with a 1×1 convolution. Moreover, XMem [6] and prior works [23, 8] take the image, the target mask, and the sum of all non-target masks (excluding background) as input for the value encoder. We discard the 'sum of all non-target masks' as we note that it becomes uninformative when there are many objects in the scene – typical in open-world scenarios. We notice a moderate speed-up (22.6→25.8 FPS in DAVIS-2017 [3]) from these modifications.

A.3. Feature Memory

Representation. The feature memory consists of three parts: a sensory memory, a working memory, and a long-term memory. The sensory memory is represented by the hidden state of the Conv-GRU and contains positional information for temporal consistency that is updated every frame. Both the working memory and the long-term memory are attention-based and contain key-value pairs. The working memory is updated every r frames and has a maximum capacity of T_{max} frames. During each update, the 'key' feature from the key encoder and the 'value' feature from the value encoder will be appended to the working memory after segmentation of the current frame. When the working memory reaches its capacity, the oldest $T_{\text{max}} - T_{\text{min}}$ frames will be consolidated into the long-term memory. Please refer to [6] for details.

Memory Reading. The last hidden state of the Conv-GRU is used as the memory readout of the sensory memory. For the working and long-term memory, we compute a query from the query frame and perform space-time memory reading [23] to read from both types of memory. For spatial dimensions H, W, the memory readout for the sensory memory is $C_h \times H \times W$ and the memory readout for the working/long-term memory is $C_v \times H \times W$. In XMem, $C_h = 64$ and $C_v = 512$ and these two features are concatenated together as the final memory readout $\mathbf{F}$.

Our Modifications. For better temporal consistency, we expand the channel size C_h of the sensory memory to $C_h = C_v = 512$. For efficiency, we use 'addition' instead of the original 'concatenation' to fuse the memory readout from the sensory memory with the working/long-term memory. Besides, we supervise the sensory memory with an auxiliary loss – a 1×1 convolution is applied to the sensory memory to produce the weights and biases of a linear classifier on the stride 16 image feature (from the key encoder) for mask prediction. Cross-entropy loss with a weight of 0.1 is applied on this predicted mask and the network is trained end-to-end.

A.4. Inference Hyperparameters

Following [6], the sensory memory is updated every frame. A new memory frame is added to the working memory every r-th frame. We synchronize r with our in-clip consensus frequency, such that every in-clip consensus result is added to the working memory. Following the default hyperparameters in [6], we set $T_{\text{max}} = 10$, $T_{\text{min}} = 5$, the maximum number of long-term memory elements to be $10,000$, and use top-k filtering [7] with $k = 30$.

A.5. Training

XMem is first pretrained on static image segmentation datasets [29, 31, 38, 16, 5] by synthesizing small video clips of three frames with affine and thin-spline deformations. It is then trained on two video datasets: YouTubeVOS [34] and DAVIS [24] by sampling clips of length eight.

Our Modifications. We make three major modifications to the training process for better robustness:

1. We introduce the more challenging OVIS [25] data into training as we find models have already saturated and produced almost perfect segmentation results on DAVIS and YouTubeVOS during training.

2. We use a 'stable' data augmentation pipeline which leads to better temporal consistency. Current state-of-the-art data augmentation pipelines use aggressive augmentation, applying different rotations [6] or crops [36] to frames within the same sequence. This encourages an invariant appearance model but harms the learning of temporal information. We instead use the same rotation and crop augmentation for a video sequence. Figure S1 visualizes the difference.

3. We clip the norm of the gradient at 3.0 during training. We find that this leads to faster convergence and more stable training.

We use a batch size of 16, the same loss function (hard-mining cross-entropy loss with a warm-up and soft DICE loss) as XMem [6], and the AdamW [20] optimizer. During pre-training, we use a learning rate of $2e$-5 for 80,000

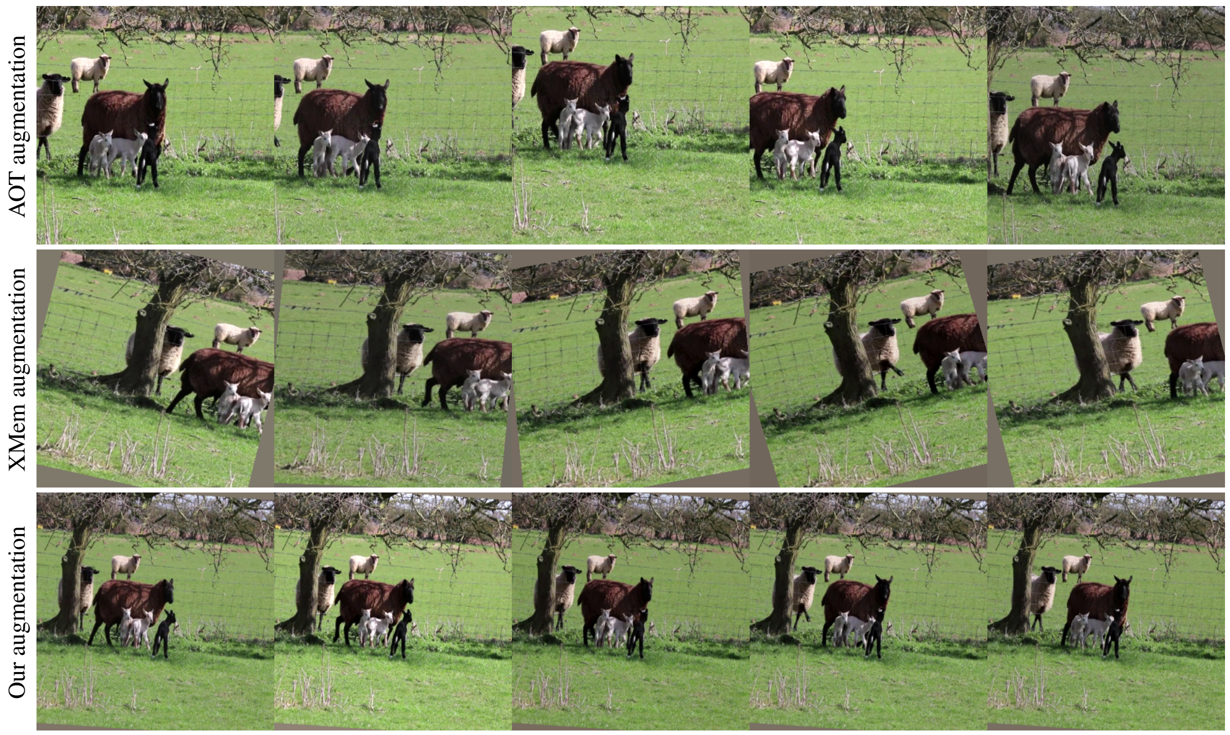

Figure S1. AOT [36] and XMem [6] use different crops and rotations within a sequence respectively. We fix both within a sequence to encourage the learning of positional information.

iterations. During main training, we use a learning rate of $1e$-5 for 150,000 iterations with a learning rate decay of 0.1 at the 120,000-th iteration and the 140,000-th iteration.

A.6. Video Object Segmentation Evaluation

We compare our temporal propagation model with state-of-the-art methods on three common video object segmentation benchmarks: DAVIS-2017 validation/test-dev [24], YouTubeVOS-2019 validation [34], and MOSE validation [10]. Table S1 tabulates our results. We resize all input such that the shorter side is 480px and bilinearly upsample the output back to the original resolution following XMem [6]. All frames in YouTubeVOS [34] are used by default. Our simple modifications bring noticeable improvements on all benchmarks. Though, we find that the overall framework is more important than these design choices (Section A.7.2).

A.7. Ablation Studies

A.7.1 On VOS Tasks

We assess the effects of our modifications on the training process on VOS tasks which purely evaluate temporal propagation performance. In addition to the standard DAVIS [24] dataset, we additionally convert the validation sets of OVIS [25] and UVO [32] to the VOS format. Following DAVIS [24], we discard any segments that do not appear in the first frame and provide the first-frame ground-truth segmentation as input. These datasets are more diverse and allow for a more complete evaluation of temporal propagation performance. Table S2 tabulates our findings. For a fair comparison, we also re-train the original XMem [6] with additional OVIS [25] data. A qualitative comparison of aggressive *vs*. stable data augmentation is illustrated in Figure S2.

A.7.2 On Large-Scale Video Panoptic Segmentation

Next, we assess whether these improvements in VOS tasks transfer to target tasks like video panoptic segmentation. We compare our final model with/without these modifications on a large-scale video panoptic segmentation dataset VIPSeg [22] with the Mask2Former-R50 [4] backbone. Table S3 (top) tabulates our findings. Note, our method still works well even without our modifications to the temporal propagation network. We find the overall framework to be more important than particular design choices within the temporal propagation model.

B. Training Data of Temporal Propagation

B.1. Sensitivity to Training Data

We train the temporal propagation on class-agnostic image segmentation and mask propagation data as described

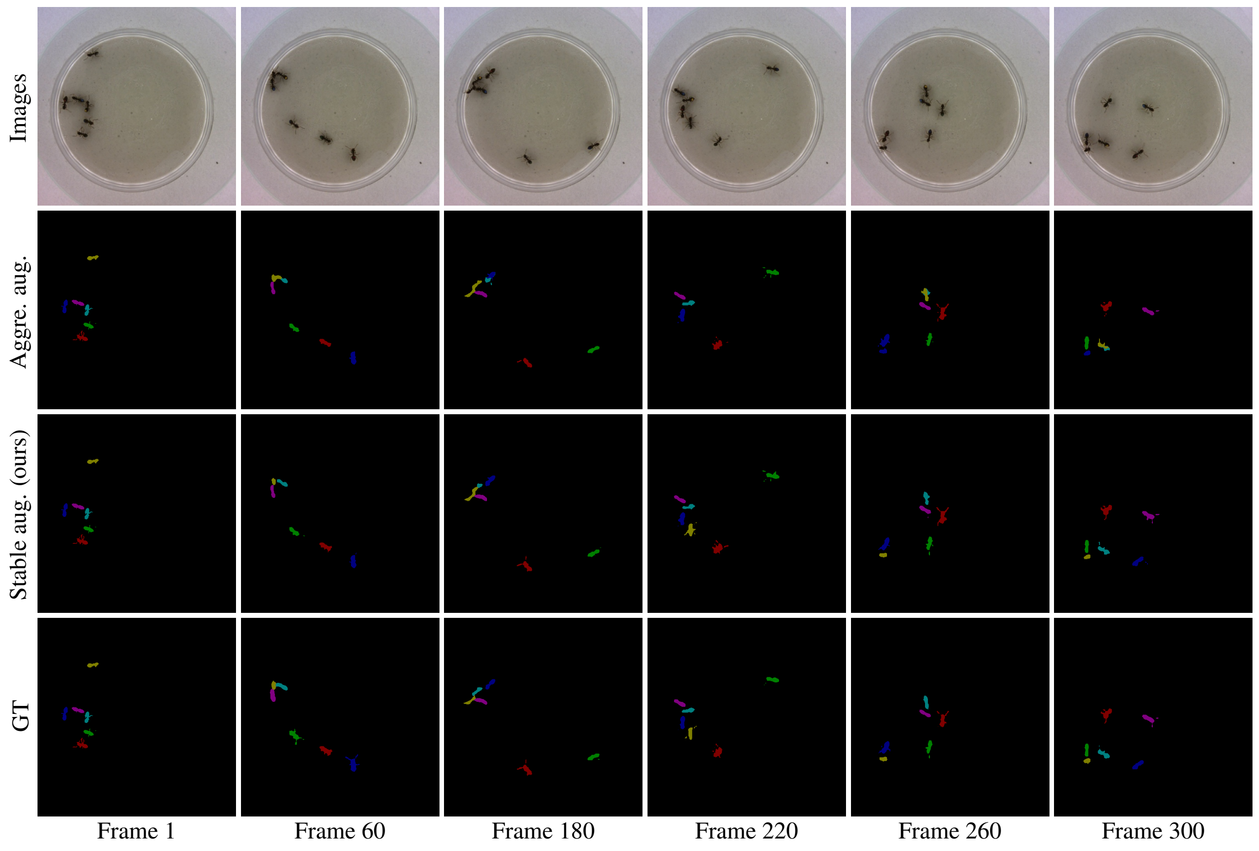

Figure S2. Comparison of methods tracking a group of ants with almost identical appearance. The variant with aggressive augmentation fails for the yellow, blue, and cyan ants toward the end while ours with stable data augmentation tracks all ants successfully. Ground-truth is annotated by us with an interactive image segmentation method, f-BRS [30]. Zoom in for details.

	MOSE			DAVIS-17 val			DAVIS-17 test-dev			YouTubeVOS-2019 val					
Method	$\mathcal{J}\&\mathcal{F}$	$\mathcal{J}$	$\mathcal{F}$	$\mathcal{J}\&\mathcal{F}$	$\mathcal{J}$	$\mathcal{F}$	$\mathcal{J}\&\mathcal{F}$	$\mathcal{J}$	$\mathcal{F}$	$\mathcal{G}$	$\mathcal{J}_s$	$\mathcal{F}_s$	$\mathcal{J}_u$	$\mathcal{F}_u$	FPS
STCN [8]	52.5	48.5	56.6	85.4	82.2	88.6	76.1	72.7	79.6	82.7	81.1	85.4	78.2	85.9	13.2
AOT-R50 [36]	58.4	54.3	62.6	84.9	82.3	87.5	79.6	75.9	83.3	85.3	83.9	88.8	79.9	88.5	6.4
XMem [6]	56.3	52.1	60.6	86.2	82.9	89.5	81.0	77.4	84.5	85.5	84.3	88.6	80.3	88.6	22.6
DEVA (ours), w/o OVIS	60.0	55.8	64.3	86.8	83.6	90.0	82.3	78.7	85.9	85.5	85.0	89.4	79.7	88.0	**25.3**
DEVA (ours), w/ OVIS	**66.5**	**62.3**	**70.8**	**87.6**	**84.2**	**91.0**	**83.2**	**79.6**	**86.8**	**86.2**	**85.4**	**89.9**	**80.5**	**89.1**	**25.3**

Table S1. Comparison of DEVA's temporal propagation module with state-of-the-art video object segmentation methods. FPS is measured on YouTubeVOS-2019 validation with a V100 GPU. All available frames in YouTubeVOS are used by default.

in Section A.5. We note that these datasets are cheap to access and amass as they do not require class-specific annotations. Here, we evaluate the importance of large-scale training of the temporal propagation model. We vary the amount of class-agnostic video-level training data under two settings: 1) with full image pretraining, and all three mask propagation datasets (DAVIS [24], YouTubeVOS [34] and OVIS [25]), and 2) without image pre-taining and using YouTubeVOS as the only training data. Table S2 (bottom) tabulates our findings on the VIPSeg [22] validation set. The performance of our model decays gracefully with fewer training data.

B.2. Class Overlaps with VIPSeg

While we train the temporal propagation network in a class-agnostic setting, the segmented objects in the training set might have object categories that overlap with the target task (e.g., with the classes in VIPSeg [22]). Here, we in-

Variant	DAVIS	OVIS	UVO	FPS
XMem [6]	86.1	69.0	82.7	22.6
XMem [6] train w/ OVIS	86.1	72.0	83.0	22.6
With all our modifications	**87.6**	**75.7**	**83.5**	**25.8**
w/o stable data aug.	87.5	73.6	83.2	**25.8**
w/o gradient clipping	85.2	71.3	82.7	**25.8**

Table S2. $\mathcal{J}\&\mathcal{F}$ performance comparisons of XMem [6] and our different modifications on VOS tasks.

vestigate the effect of this overlap of temporal propagation training data with target task data on the final performance.

For this, we train the temporal propagation network with only YouTubeVOS [34] data which has 65 object categories (other datasets that we use for training have no class annotation). We manually match these 65 categories with the classes in VIPSeg [22] to partition the classes of VIPSeg into three sets: 'overlapping', 'non-overlapping', or 'ambiguous'.[1] We then evaluate the final task performance on the overlapping and the non-overlapping sets separately, while ignoring the 'ambiguous' set. We perform the same evaluation on an end-to-end method, Video-K-Net [17], as a measure of 'baseline difficulty' for each set. Table S4 tabulates our findings. We observe no significant difference between the overlapping and non-overlapping set when accounting for the difficulty delta (Δ) observed in the baseline. This indicates that our class-agnostic training does not overfit to the object categories in the training set.

C. Detailed Experimental Settings and Results

C.1. Large-Scale Video Panoptic Segmentation

Following the standard practice [22], we use the 720p version of the VIPSeg [22] dataset. We evaluate using its validation set (343 videos) and compute VPQ/STQ using the official codebase. During temporal propagation, we downsample the videos such that the shortest side is 480px and bilinearly upsample the result back to the original resolution following [6].

Video Panoptic Segmentation (VPS) requires the prediction of class labels. We obtain these labels from the image segmentation model and use online majority voting to determine the output label. Formally, we keep a list of class labels $\mathbf{Cl}_i$ for each object r_i. When an existing (propagated) segment r_i matches with a segment from the in-clip consensus c_j, i.e., $a_{ij} = 1$, we take the class label from the consensus c_j and append it to the list $\mathbf{Cl}_i$. At the output of every frame, we determine the class label associated with segment r_i by performing majority voting in $\mathbf{Cl}_i$. Note, in accordance with VPS evaluation [15], an object can only have one class label throughout the video. This means a change in class label necessitates a change in object id, which we also implemented. Thus, a change in class label might lead to lower association accuracy. An alternative algorithm would be to use the final major voting result to retroactively apply the class label in all frames, which would however not be strictly online/semi-online.

Running time Analysis Under our default semi-online setting, we use a clip size of 3 and perform merging every 5 frames (i.e., invoking the image model on 60% of all frames). We report time on VIPSeg [22], averaged across all frames, on an A6000 GPU. The mask propagation module takes 83ms per frame (VIPSeg has more objects per video than the standard VOS timing benchmark DAVIS-2017). For every merge, pre-processing (spatial alignment and finding pairwise IoU) takes 211ms, and solving the integer program takes 15ms. For the image model (R50 backbone), both Video-K-Net [17] and Mask2Former [4] take around 200ms per frame. Overall, our method runs at 4.0fps. Meanwhile, state-of-the-art Video-K-Net runs at 4.9fps. Ours is 18% slower but has a 52% higher $\overline{\text{VPQ}}$.

C.2. Open-World Video Segmentation

We evaluate on the validation (993 videos) and test (1421 videos) sets of BURST [1]. As in Section C.1, we downsample the videos during temporal propagation such that the shortest side is 480px and bilinearly upsample the result back to the original resolution following [6]. For efficiency, we process only every three frames. Since the ground-truth is sparse (annotated every 24 or 30 frames), we can still perform a complete evaluation.

For the Mask2Former [4] image model, we follow BURST [1] and use the best-performing Swin-L checkpoint trained on COCO [18] provided by the authors. For the EntitySeg [26] image model, we also use the best available Swin-L model checkpoint trained on COCO [18]. For overlapping predictions, we use the post-processing for panoptic segmentation in Mask2Former [4] to resolve them.

We assess Open World Tracking Accuracy (OWTA) using official tools. OWTA is the geometric mean of Detection Recall (DetRe) and Association Accuracy (AssA). Please refer to [1] for details. For completeness, we additionally report DetRe and AssA of baselines and our method in Table S5.

C.3. Referring Video Segmentation

To evaluate on Ref-DAVIS [14] and Ref-YouTubeVOS [28], we use ReferFormer Swin-L [33] as the image model. The network is first pretrained on Ref-COCO [37], Ref-COCO+ [37], and G-Ref [21] datasets and finetuned on Ref-YouTubeVOS [28] following [33].

[1]The overlapping set includes flag, parasol_or_umbrella, car, bus, truck, bicycle, motorcycle, ship_or_boat, airplane, person, cat, dog, horse, cattle, skateboard, ball, box, bottle_or_cup, table_or_desk, mirror, and train (21 in total). The ambiguous set includes other_animal, bag_or_package, toy, and textiles (4 in total). The remaining (99) classes in VIPSeg are in the non-overlapping set.

Varying Temporal Propagation Model	VPQ^1	VPQ^2	VPQ^4	VPQ^6	VPQ^8	VPQ^{10}	VPQ^∞	$\overline{\text{VPQ}}$	STQ
With standard XMem [6]	41.9	41.3	40.6	39.9	39.5	39.0	35.4	37.9	41.3
With our modified XMem [6]	**42.1**	**41.5**	**40.8**	**40.1**	**39.7**	**39.3**	**36.1**	**38.3**	**41.5**
Varying Training Data of Temporal Propagation	VPQ^1	VPQ^2	VPQ^4	VPQ^6	VPQ^8	VPQ^{10}	VPQ^∞	$\overline{\text{VPQ}}$	STQ
Image pretraining + 100% video training	**42.1**	**41.5**	**40.8**	**40.1**	**39.7**	**39.3**	**36.1**	**38.3**	**41.5**
Image pretraining + 50% video training	42.0	41.4	40.7	40.1	39.7	39.4	36.0	38.3	41.3
Image pretraining + 10% video training	40.7	40.1	39.3	38.5	38.1	37.7	34.6	36.8	40.1
Training on 100% YouTube-VOS [34] only	41.4	40.9	40.2	39.5	39.1	38.7	35.6	37.8	41.0
Training on 50% YouTube-VOS [34] only	40.5	39.4	38.0	36.6	35.6	34.4	31.3	34.4	37.8

Table S3. Performance comparisons of our method with different temporal propagation model settings on the VIPSeg [22] validation set. For a fair comparison, all are semi-online with a Mask2Former-R50 [4] image model input.

Method	$\overline{\text{VPQ}}_{\text{overlap}}$	$\overline{\text{VPQ}}_{\text{no-overlap}}$	$\Delta_{\text{overlap}\rightarrow\text{non-overlap}}$
Video-K-Net	25.0	25.7	+0.7
Ours	38.1	38.6	+0.5

Table S4. Performance comparison on different classes of VIPSeg that overlap or do not overlap with the training data of temporal propagation. As a baseline, we use Video-K-Net-R50 [17]. For ours, we use Mask2Former-R50 with a temporal propagation model that is only trained on YouTubeVOS [34] and evaluated in a semi-online setting.

Unlike in video panoptic segmentation or open-world video segmentation, we do not need to use integer programming to associate segments from the image model in different frames. This is because each segment corresponds to a known language expression. Thus, we process each object independently and use argmax to fuse the final segmentations. As mentioned in the main paper, we employ an offline setting as in prior works [28, 33, 11].

In the offline setting, we first perform in-clip consensus by selecting 10 uniformly spaced frames in the video and using the frame with the highest confidence given by the image model as a 'key frame' for aligning the other frames. Soft probability maps are used in the consensus to preserve confidence levels in the prediction. We then forward- and backward-propagate from the key frame without incorporating additional image segmentations.

Formally, for a given object, we denote its soft probability map and confidence score given by the image model as $\mathbf{Pr}_t \in [0,1]^{H\times W}$ and $c_t \in [0,1]$ respectively. We denote the frame index of the ten chosen frames as $\mathbf{T}_c = \{t_1, t_2, ..., t_{10}\}$.

We aim to compute a soft probability consensus $\mathbf{Cs}_{t_k}$ at a keyframe index t_k by a weighted summation of the soft probability maps of the chosen frames $\{\mathbf{Pr}_{t_1}, \mathbf{Pr}_{t_2}, ..., \mathbf{Pr}_{t_{10}}\}$:

$$\mathbf{Cs}_{t_k} = \sum_{i\in\mathbf{T}_c} w_i \mathbf{Pr}_i, \tag{S1}$$

where w_i is a weighting coefficient, with $\sum_i w_i = 1$.

We use the frame with the highest confidence predicted by the image model as the keyframe:

$$t_k = \text{argmax}_{i\in\mathbf{T}_c} c_i. \tag{S2}$$

We compute the weighting coefficients using a softmax of the confidences such that we weigh confident predictions more:

$$w_i = \frac{e^{c_i}}{\sum_{i\in\mathbf{T}_c} e^{c_i}}. \tag{S3}$$

After the consensus, $\mathbf{Cs}_{t_k}$ is used to initialize forward and backward propagation from frame t_k without incorporating additional image segmentations. The propagation is implemented as standard semi-supervised video object segmentation inference with the keyframe as initial guidance. During propagation, the internal memory $\mathbf{H}$ is updated every 5 frames using its own prediction as in [6].

C.4. Unsupervised Video Object Segmentation

For single-object unsupervised video object segmentation (DAVIS-2016 [24]), we use DIS [27] as the image segmentation model. Since it does not provide segmentation confidence, we approximate it with the normalized area of the predicted mask to ignore null detections, i.e., $c_i = \frac{1}{HW}\|\mathbf{Pr}_i\|_1$.

For multi-object unsupervised video object segmentation (DAVIS-2017 [3]), we follow our semi-online protocol in open-world video segmentation. The exception being DAVIS-2017 [3] allows a maximum of 20 objects in the prediction. We overcome this limitation online by only accepting the first 20 objects and discarding the rest. When there are more than 20 objects in the frame, we prioritize the ones with larger areas as they are less likely to be noisy.

		Validation								
		All			Common			Uncommon		
Method		DetRe	AssA	OWTA	DetRe	AssA	OWTA	DetRe	AssA	OWTA
Mask2Former	w/ Box tracker [1]	66.9	55.8	60.9	78.7	57.1	60.9	20.1	30.5	24.0
Mask2Former	w/ STCN tracker [1]	67.0	62.6	64.6	78.8	64.1	71.0	20.0	33.3	25.0
OWTB [19]		70.9	45.2	56.2	76.8	47.0	59.8	46.5	34.3	38.5
Mask2Former	w/ ours online	72.1	67.5	69.5	80.2	69.9	74.6	39.8	46.4	42.3
Mask2Former	w/ ours semi-online	71.8	**68.5**	**69.9**	**80.3**	**70.7**	**75.2**	37.9	46.8	41.5
EntitySeg	w/ ours online	72.3	66.0	68.8	77.7	68.4	72.7	**50.3**	50.2	49.6
EntitySeg	w/ ours semi-online	**72.4**	67.1	69.5	78.1	69.3	73.3	50.0	**52.2**	**50.5**
		Test								
		All			Common			Uncommon		
Method		DetRe	AssA	OWTA	DetRe	AssA	OWTA	DetRe	AssA	OWTA
Mask2Former	w/ Box tracker [1]	61.5	51.1	55.9	71.4	52.5	61.0	21.1	30.0	24.6
Mask2Former	w/ STCN tracker [1]	61.6	54.1	57.5	71.5	55.7	62.9	21.0	28.6	23.9
OWTB [19]		70.9	45.2	56.2	76.8	47.0	59.8	46.5	34.3	38.5
Mask2Former	w/ ours online	72.2	68.6	70.1	**79.8**	70.8	75.0	40.7	49.2	44.1
Mask2Former	w/ ours semi-online	71.9	**69.6**	**70.5**	79.7	**71.7**	**75.4**	39.5	50.7	44.1
EntitySeg	w/ ours online	**72.5**	67.3	69.5	77.3	69.2	72.9	**52.3**	55.0	53.0
EntitySeg	w/ ours semi-online	72.4	67.7	69.8	77.4	69.5	73.1	51.9	**55.9**	**53.3**

Table S5. Extended results comparing baselines and our methods in the validation/test sets of BURST [1]. Baseline performances are transcribed from [1].

D. Results on YouTube-VIS

Here we present additional results on the small-vocabulary YouTube-VIS [35] dataset, but unsurprisingly recent end-to-end specialized approaches perform better because a sufficient amount of data is available in this case. For this task, we use our online video panoptic segmentation setting. Besides the difference in the scale of vocabularies, our method assumes that no two objects occupy the same pixel, and produces a non-overlapping mask. Although this assumption is usually true, it harms the Average Precision (AP) evaluation of our method in VIS, with other methods typically outputting many ($\geq$100) potentially overlapping proposals for higher recall. We provide our result in Table S6.

Method	mAP	AP@75
MaskProp [2]	40.0	42.9
Video-K-Net [17]	40.5	44.5
MinVIS [13]	**47.4**	**52.1**
Mask2Former [4] w/ Ours	40.8	44.3

Table S6. Performance comparisons on YouTube-VIS 2019 validation. All models use a ResNet-50 backbone. Note MinVIS is optimized for small-vocabulary YouTube-VIS and underperforms by 8.4 $\overline{\text{VPQ}}$ compared with our method in large-vocabulary VIPSeg (Tab. 6 of the main paper, Query assoc. *vs.* Ours).

E. Qualitative Results

E.1. Visualization

For all results (see our project page), we associate each object id with a unique color. When a segment changes color, its object id has changed. This change might happen often (e.g., flicker) if the method is not stable. We additionally show an 'overlay' which is a composite of the colored segmentation with the input image.

E.2. Large-Scale Video Panoptic Segmentation

We compare our method with state-of-the-art Video-K-Net [17]. We use the semi-online setting Mask2Former [4] as the image model. Videos are taken from VIPSeg [22] validation set.

E.3. Open-World Video Segmentation

We compare our method with the best open-world segmentation baseline (Mask2Former + STCN tracker). We use the semi-online setting EntitySeg [26] as the image model. Videos are collected from BURST [1] and the Internet.

E.4. Referring Video Segmentation in the Wild

We compare our method with state-of-the-art referring video segmentation ReferFormer [33]. We are interested in the open-world setting beyond standard Ref-DAVIS [14] and Ref-YouTubeVOS [28]. For this, we use a recent open-world referring image segmentation model X-Decoder [39] as our image model. The agility to switch image backbones and use the latest advancements in image segmentation is one of the main advantages of our decoupled formulation. We employ an offline setting following our referring video segmentation evaluation protocol (Section C.3). Note, ReferFormer [33] is also offline. Our model can segment rare objects like 'wheel of fortune' accurately.

References

[1] Ali Athar, Jonathon Luiten, Paul Voigtlaender, Tarasha Khurana, Achal Dave, Bastian Leibe, and Deva Ramanan. Burst: A benchmark for unifying object recognition, segmentation and tracking in video. In *WACV*, 2023. 5, 7

[2] Gedas Bertasius and Lorenzo Torresani. Classifying, segmenting, and tracking object instances in video with mask propagation. In *CVPR*, 2020. 7

[3] Sergi Caelles, Jordi Pont-Tuset, Federico Perazzi, Alberto Montes, Kevis-Kokitsi Maninis, and Luc Van Gool. The 2019 davis challenge on vos: Unsupervised multi-object segmentation. In *arXiv preprint arXiv:1905.00737*, 2019. 2, 6

[4] Bowen Cheng, Ishan Misra, Alexander G Schwing, Alexander Kirillov, and Rohit Girdhar. Masked-attention mask transformer for universal image segmentation. In *CVPR*, 2022. 3, 5, 6, 7

[5] Ho Kei Cheng, Jihoon Chung, Yu-Wing Tai, and Chi-Keung Tang. Cascadepsp: Toward class-agnostic and very high-resolution segmentation via global and local refinement. In *CVPR*, 2020. 2

[6] Ho Kei Cheng and Alexander G Schwing. Xmem: Long-term video object segmentation with an atkinson-shiffrin memory model. In *ECCV*, 2022. 1, 2, 3, 4, 5, 6

[7] Ho Kei Cheng, Yu-Wing Tai, and Chi-Keung Tang. Modular interactive video object segmentation: Interaction-to-mask, propagation and difference-aware fusion. In *CVPR*, 2021. 2

[8] Ho Kei Cheng, Yu-Wing Tai, and Chi-Keung Tang. Rethinking space-time networks with improved memory coverage for efficient video object segmentation. In *NeurIPS*, 2021. 2, 4

[9] Junyoung Chung, Caglar Gulcehre, KyungHyun Cho, and Yoshua Bengio. Empirical evaluation of gated recurrent neural networks on sequence modeling. *NIPS Workshop*, 2014. 1, 2

[10] Henghui Ding, Chang Liu, Shuting He, Xudong Jiang, Philip HS Torr, and Song Bai. MOSE: A new dataset for video object segmentation in complex scenes. In *ICCV*, 2023. 3

[11] Henghui Ding, Chang Liu, Suchen Wang, and Xudong Jiang. Vlt: Vision-language transformer and query generation for referring segmentation. In *TPAMI*, 2022. 6

[12] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *CVPR*, 2016. 1

[13] De-An Huang, Zhiding Yu, and Anima Anandkumar. Minvis: A minimal video instance segmentation framework without video-based training. In *NeurIPS*, 2022. 7

[14] Anna Khoreva, Anna Rohrbach, and Bernt Schiele. Video object segmentation with language referring expressions. In *ACCV*, 2019. 5, 7

[15] Dahun Kim, Sanghyun Woo, Joon-Young Lee, and In So Kweon. Video panoptic segmentation. In *CVPR*, 2020. 5

[16] Xiang Li, Tianhan Wei, Yau Pun Chen, Yu-Wing Tai, and Chi-Keung Tang. Fss-1000: A 1000-class dataset for few-shot segmentation. In *CVPR*, 2020. 2

[17] Xiangtai Li, Wenwei Zhang, Jiangmiao Pang, Kai Chen, Guangliang Cheng, Yunhai Tong, and Chen Change Loy. Video k-net: A simple, strong, and unified baseline for video segmentation. In *CVPR*, 2022. 5, 6, 7

[18] Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C Lawrence Zitnick. Microsoft coco: Common objects in context. In *ECCV*, 2014. 5

[19] Yang Liu, Idil Esen Zulfikar, Jonathon Luiten, Achal Dave, Deva Ramanan, Bastian Leibe, Aljoša Ošep, and Laura Leal-Taixé. Opening up open world tracking. In *CVPR*, 2022. 7

[20] Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. In *ICLR*, 2019. 2

[21] Junhua Mao, Jonathan Huang, Alexander Toshev, Oana Camburu, Alan L Yuille, and Kevin Murphy. Generation and comprehension of unambiguous object descriptions. In *CVPR*, 2016. 5

[22] Jiaxu Miao, Xiaohan Wang, Yu Wu, Wei Li, Xu Zhang, Yunchao Wei, and Yi Yang. Large-scale video panoptic segmentation in the wild: A benchmark. In *CVPR*, 2022. 3, 4, 5, 6, 7

[23] Seoung Wug Oh, Joon-Young Lee, Ning Xu, and Seon Joo Kim. Video object segmentation using space-time memory networks. In *ICCV*, 2019. 2

[24] Federico Perazzi, Jordi Pont-Tuset, Brian McWilliams, Luc Van Gool, Markus Gross, and Alexander Sorkine-Hornung. A benchmark dataset and evaluation methodology for video object segmentation. In *CVPR*, 2016. 2, 3, 4, 6

[25] Jiyang Qi, Yan Gao, Yao Hu, Xinggang Wang, Xiaoyu Liu, Xiang Bai, Serge Belongie, Alan Yuille, Philip HS Torr, and Song Bai. Occluded video instance segmentation. *IJCV*, 2022. 2, 3, 4

[26] Lu Qi, Jason Kuen, Yi Wang, Jiuxiang Gu, Hengshuang Zhao, Zhe Lin, Philip Torr, and Jiaya Jia. Open-world entity segmentation. In *arXiv preprint arXiv:2107.14228*, 2021. 5, 7

[27] Xuebin Qin, Hang Dai, Xiaobin Hu, Deng-Ping Fan, Ling Shao, and Luc Van Gool. Highly accurate dichotomous image segmentation. In *ECCV*, 2022. 6

[28] Seonguk Seo, Joon-Young Lee, and Bohyung Han. Urvos: Unified referring video object segmentation network with a large-scale benchmark. In *ECCV*, 2020. 5, 6, 8

[29] Jianping Shi, Qiong Yan, Li Xu, and Jiaya Jia. Hierarchical image saliency detection on extended cssd. In *TPAMI*, 2015. 2

[30] Konstantin Sofiiuk, Ilia Petrov, Olga Barinova, and Anton Konushin. f-brs: Rethinking backpropagating refinement for interactive segmentation. In *CVPR*, 2020. 4

[31] Lijun Wang, Huchuan Lu, Yifan Wang, Mengyang Feng, Dong Wang, Baocai Yin, and Xiang Ruan. Learning to detect salient objects with image-level supervision. In *CVPR*, 2017. 2

[32] Weiyao Wang, Matt Feiszli, Heng Wang, and Du Tran. Unidentified video objects: A benchmark for dense, open-world segmentation. In *CVPR*, 2021. 3

[33] Jiannan Wu, Yi Jiang, Peize Sun, Zehuan Yuan, and Ping Luo. Language as queries for referring video object segmentation. In *CVPR*, 2022. 5, 6, 7, 8

[34] Ning Xu, Linjie Yang, Yuchen Fan, Dingcheng Yue, Yuchen Liang, Jianchao Yang, and Thomas Huang. Youtube-vos: A large-scale video object segmentation benchmark. In *ECCV*, 2018. 2, 3, 4, 5, 6

[35] Linjie Yang, Yuchen Fan, and Ning Xu. Video instance segmentation. In *ICCV*, 2019. 1, 7

[36] Zongxin Yang, Yunchao Wei, and Yi Yang. Associating objects with transformers for video object segmentation. In *NeurIPS*, 2021. 2, 3, 4

[37] Licheng Yu, Patrick Poirson, Shan Yang, Alexander C Berg, and Tamara L Berg. Modeling context in referring expressions. In *ECCV*, 2016. 5

[38] Yi Zeng, Pingping Zhang, Jianming Zhang, Zhe Lin, and Huchuan Lu. Towards high-resolution salient object detection. In *ICCV*, 2019. 2

[39] Xueyan Zou, Zi-Yi Dou, Jianwei Yang, Zhe Gan, Linjie Li, Chunyuan Li, Xiyang Dai, Harkirat Behl, Jianfeng Wang, Lu Yuan, et al. Generalized decoding for pixel, image, and language. In *CVPR*, 2023. 8