| Dataset | Videos (train/val) | Frames (train/val) | Avg. len. (sec) | Ann. fps | Avg. obj size (% of frame) |
|---|---|---|---|---|---|
| VISOR | 5.3k / 1.2k | 33k / 7.7k | 12.0 | 0.5 | 6.67 |
| VOST | 572 / 70 | 60k / 8k | 21.2 | 5.0 | 2.57 |

Table A1. **Dataset statistics for VISOR and VOST**. Avg. len. stands for average video length, and Ann. fps denotes annotation frames per second.

## Abstract

*This document provides additional material that is supplemental to our main submission. Section A describes the associated supplemental video. Section B includes additional implementation and dataset details, followed by Section C for additional experimental results and ablation studies. Finally, Section D details the societal impact of our work as standard practice in computer vision research.*

## A. Supplemental Video

We include an accompanying supplemental video - `video_demo.mp4` - as part of the supplemental materials. In this video we show qualitative segmentation results of our approach on the two benchmarks VOST [44] and VISOR [13]. For VOST, we show predictions on three videos showing our method's capability on small and multiple instances of objects compared to the previous best state-of-the-art method, AOT [54]. We additionally include a failure case. For VISOR, we present three videos demonstrating our method's tracking capabilities compared to the previous best method, STM [35], and include a failure case as well. Furthermore, we evaluate our model on generic videos sampled from YouTube, with the results presented in the final section of the video. The video is in MP4 format and is 4 minutes 17 seconds long. The codec used for creating the provided video is H.264 (x264). Please note that due to file size limitations for supplementary material, we provide the low-resolution version, while a higher-resolution demo video will be made publicly available upon acceptance.

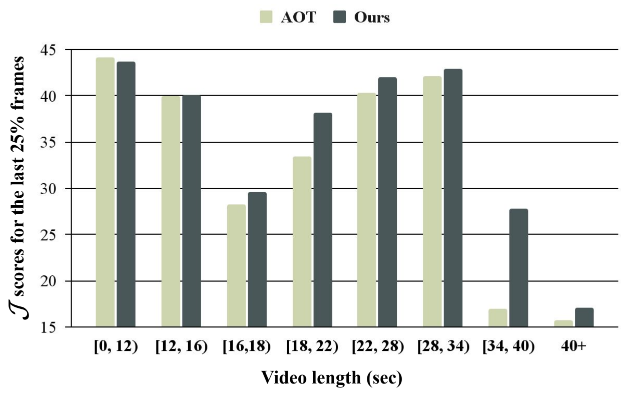
## B. Implementation and dataset details

**General details.** We follow a two-stage training protocol similar to prior works [29, 35, 54, 60], where we first pretrain our model using synthetic video sequences generated from static image datasets [10, 16, 18, 32, 43], and then finetune on target benchmarks. Consistent with prior works, we use ResNet-50 [19] as the backbone for image encoder and mask encoder with shared weights. In our Multi-Scale Matching Encoder, if not specified, we use feature maps from last two layers of the visual backbone at strides 32 and 16. During training, we use video length of 12, with clip length $L = 2$, and a memory bank size of $N = 7$. We use AdamW [33] optimizer with $2e^{-5}$ learning rate over 20 epochs with 624 and 480 image resolution for VOST and VISOR respectively. For the clip-based memory, we retain the ground truth for the initial reference frame and mask as the first element. We perform updates by storing the last frame and its predicted mask features for each clip in the memory in a FIFO fashion. During inference, we process the entire video clip-by-clip to predict the results. Please find more details in Suppl.

**Training and computation resources.** We use AdamW [33] optimizer with weight decay $1e^{-4}$, learning rate $1e^{-5}$ for the backbone and $2e^{-4}$ for the rest of the model. We train for 20 epochs and reduce learning rate by $1/10^{th}$ factor at the $8^{th}$ and $16^{th}$ epoch. Similar to AOT [54], we use random resizing and cropping for data augmentation, avoiding additional augmentations to ensure a fair comparison. We uniformly use batch size of 1 per GPU, and conduct experiments on VOST and VISOR datasets using 4 Nvidia A40 and 8 Nvidia T4 GPUs respectively. We use video lengths of 12 and 6 for VOST and VISOR respectively, necessitating the use of higher GPU memory in the case of VOST compared to VISOR. The input image resolution is set to 480 and 624 pixels on shorter side (keeping the aspect ratio same) on VISOR and VOST respectively.

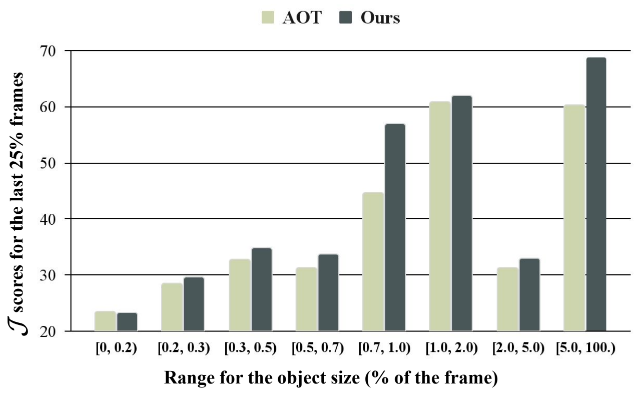
**Architecture.** Our model is built on ResNet-50 [19] backbone, which excludes dilation in the last convolutional block compared to STM and related approaches [35, 36]. The backbone is shared for encoding both the images and masks. Keeping in line with TubeDETR [51], we use fixed-sinusoidal 2D positional embedding for both image and mask features. For multi-scale query matching, we use a multi-head attention layer [27] with a hidden size of 256 and 4 heads for each scale. We perform LayerNorm [2] on queries and keys before multi-head attention, and on output after the multi-head attention. We use GELU [20] activation function on the residual connection for the output, and use a 10% attention dropout to reduce overfitting.

Our decoder consists of a pixel decoder [6] and a space-time transformer decoder [51] used after the multi-scale matching encoder. The pixel decoder [6] produces feature-pyramid at four scales using a combination of lateral and output `Conv` layers, with the hidden dimension / output channel size of 256. And the space-time transformer decoder [51] incorporates 6 layers of multi-head attention, comprising 8 heads with a hidden dimension of 256. For the output, we use `ReLU` as the activation function and maintain a 10% dropout rate during training.



| length | [0, 12) | [12, 16) | [16, 18) | [18, 22) | [22, 28) | [28, 34) | [34, 40) | [40+ |
|---|---|---|---|---|---|---|---|---|
| # video | 8 | 10 | 7 | 13 | 15 | 9 | 6 | 2 |

(a)



| Range | [0, 0.2) | [0.2, 0.3) | [0.3, 0.5) | [0.5, 0.7) | [0.7, 1.0) | [1.0, 1.2) | [2.0, 5.0) | [5.0+ |
|---|---|---|---|---|---|---|---|---|
| # video | 12 | 8 | 10 | 10 | 6 | 11 | 6 | 7 |

(b)

Figure A1. **Performance breakdown w.r.t. (a) video length and (b) small object size.** We show performance comparison of AOT [54] and our method on subsets of different video lengths and object sizes. (a) We observe that 1) performance decreases in long duration scenarios, demonstrating the complexity in long videos, and 2) our method generally outperforms AOT [54] on long-range subsets, especially upto 10% on videos longer than 34 secs. (b) We observe that 1) with smaller object size, the performance decreases confirming the complexity of the task, and 2) our method outperforms on all small object subsets compared to AOT [54] demonstrating the effectiveness of our method on small objects (SM). # video denotes the number of videos.

**Dataset.** The data statistics for VOST [44] and VISOR [13] are outlined in Tab. A1. Notably, the videos in VOST are twice as long in duration compared to VISOR. In particular, for VOST, the average length of videos is 21.2 seconds, annotated at 5 frames-per-second (fps). This amounts to approximately 106 annotated frames per video on average. In comparison, VISOR includes 6 annotated frames per video with a much lower 0.5 annotation frame rate (fps).

Both datasets focus on egocentric videos, thereby resulting in relatively smaller average object sizes in comparison to the conventional VOS datasets [38]. Specifically, VOST's average object size equates to 2.57% of the frame size, whereas VISOR's average object size corresponds to 6.67% of the frames. This indicates that designing a model tailored for objects of various sizes (including small objects) is essential in egocentric videos, which is tied to various applications in robotics and augmented/virtual reality.

## C. Additional experimental results

In this section, we provide additional quantitative and qualitative results, along with additional visualization and analysis.

### C.1. Additional quantitative results

**Comparing with recent methods of Cutie [7] and RMem [59]** We provide comparison with recent works of Cutie [7] and RMem [59]. In Table A4, we compare our performance with Cutie, by re-training it under the same settings using the official codebase. For completeness, in Table A3, we compare on DAVIS dataset. As shown in these tables, our model outperforms Cutie with considerable margin on VOST, the more challenging dataset. Additionally, our model achieves better performance on DAVIS and competitive results on VOST compared to RMem, as shown in Table A3.

**VOST subsets ablations.** In the main paper, we evaluate across various subsets of validation data to measure capability of the model across different aspects of the problem, which includes long videos (LNG) and small objects (SM) subsets. To further demonstrate our model's performance in these challenging scenarios, we provide a breakdown of results on the subsets at a granular level. The results based on video length subsets is shown in Fig. A1a, illustrating the $\mathcal{J}_{\text{tr}}$ scores on different video length ranges. Notably, while our model demonstrates strong performance in shorter videos (i.e., less than 16 sec), tracking objects becomes challenging in longer setting, with performance (in $\mathcal{J}_{\text{tr}}$-score) dropping from 43 to 27 in videos longer than 40 sec (or containing more than 200 frames). Despite this, our model outperforms AOT by $\sim$ 10% in longer video setting (*i.e.*, video length between 34 and 40 sec) - confirming the effectiveness of our clip-based memory and matching strategies in long-range scenarios.

Additionally, the results on small objects (SM) subset is shown in Fig. A1b, where we plot $\mathcal{J}_{\text{tr}}$ scores on different ob-

| Approach | $\mathcal{J}\&\mathcal{F}$ | $\mathcal{J}$ | $\mathcal{F}$ |
|---|---|---|---|
| SSTVOS [15] | 78.4 | 75.4 | 81.4 |
| AOT-S [54] | 79.2 | 76.4 | 82.0 |
| TBD [11] | 80.0 | 77.6 | 82.3 |
| HMMN [42] | 80.4 | **77.7** | 83.1 |
| TAM-VT(Ours) | **81.0** | 77.1 | **84.9** |

(a) Pre-trained on Static and fine-tuned on DAVIS'17 only.

| Approach | Object size | | |
|---|---|---|---|
| | $< 1\%$ | $< 0.5\%$ | $< 0.3\%$ |
| HMMN [42] | 65.1 | 52.3 | 17.4 |
| TAM-VT(Ours) | **69.5** | **63.0** | **42.6** |

(b) Performance breakdown on DAVIS'17 val set w.r.t. small object sizes.

| Approach | $\mathcal{J}\&\mathcal{F}$ | $\mathcal{J}$ | $\mathcal{F}$ |
|---|---|---|---|
| STM [13, 35] | 81.8 | 79.2 | 84.3 |
| CFBI [53] | 81.9 | 79.1 | 84.6 |
| CFBI+ [53] | 82.9 | 80.1 | 85.7 |
| STCN [9] | <u>85.4</u> | 82.2 | <u>88.6</u> |
| HMMN [42] | 84.7 | 81.9 | 87.5 |
| AOT [54] | 84.9 | 82.3 | 87.5 |
| AOT+RMem [59] | 85.2 | 82.5 | 87.9 |
| RDE [28] | 84.2 | 80.8 | 87.5 |
| XMem [8] | 86.2 | 82.9 | 89.5 |
| XMem+Boosting [58] | 87.7 | 84.1 | 91.2 |
| Cutie [7] | **88.8** | **85.4** | **92.3** |
| DeAOT [56] | 85.2 | 82.2 | 88.2 |
| TAM-VT(Ours) | <u>85.3</u> | <u>82.5</u> | 88.2 |

(c) Pre-trained on Static and jointly fine-tuned on YouTubeVOS and DAVIS'17.

Table A2. **Performance comparison on conventional datasets**. (a) DAVIS'17 val. set. (b) DAVIS'17 val. subsets w.r.t. object sizes. (c) DAVIS'17 val. set (jointly trained with YouTubeVOS dataset). Best results highlighted in **Bold** and <u>underline</u> respectively.

| Approach | Backbone | $\mathcal{J}\&\mathcal{F}$ | $\mathcal{J}$ | $\mathcal{F}$ |
|---|---|---|---|---|
| Xmem + Boosting [58] | R50 | 87.7 | 84.1 | 91.2 |
| Cutie [7] | R50 | **88.8** | **85.4** | **92.3** |
| AOT + Rmem [59] | R50 | 85.2 | 82.5 | 87.9 |
| Ours | R50 | 85.3 | 82.5 | 88.2 |

Table A3. **Comparison on DAVIS'17**.

| Approach | Backbone | $\mathcal{J}_{\text{tr}}$ | $\mathcal{J}$ |
|---|---|---|---|
| Cutie [7] | Static | 32.5 | 44.6 |
| AOT | Static | 35.1 | 47.1 |
| **Ours** | Static | **36.5** | **48.2** |
| AOT + RMem [59] | Static + DAVIS + YT | **39.8** | **50.5** |
| **Ours** | Static + DAVIS + YT | 38.2 | 49.5 |

Table A4. **Comparison on VOST with concurrent works**.

ject size ranges (as a percentage of the frame size). We observe that performance decreases as object size decreases, dropping from $\sim$ 65 to as low as 25 with objects smaller than $0.2\%$ of the frame size, further confirming the complexity of the task. However, we note that our model consistently outperforms the prior best model, AOT [54], across all subsets, highlighting the effectiveness of our multi-scale object tracking design.

**Performance Comparison on Conventional VOS Datasets** We also verify the effectiveness of our model on DAVIS'17 [38] benchmark. Following standard training protocol [42, 55], we pre-train on Static datasets [10, 16, 18, 32, 43]. First, we report fine-tuning performance on DAVIS'17 only in Table A2a. Our approach achieves best results on overall scores $\mathcal{J}\&\mathcal{F}$ and $1.7\%$ improvement on the $\mathcal{F}$ metric. Second, we draw comparisons with HMMN [42] on small objects, where the latter make use of efficient multi-scale memory matching. We divide DAVIS'17 into subsets based on object size (similar to the VOST dataset). Table A2b shows that our model is better equipped to capture small-sized objects on conventional benchmark, demonstrating the efficacy of our unified multi-scale encoder-decoder framework. Lastly, we provide the results of our model when trained on both YouTubeVOS and DAVIS'17 jointly and evaluated on the DAVIS'17 val set. Table A2c show that our model is close to or even outperform previous works, verifying the generalization of our approach. Note that in the challenging VOST our method outperforms XMem with a considerable margin.

**DAVIS'17 subsets analysis.** In the Table A2a, we demonstrate that our model achieves competitive performance on a conventional video object segmentation dataset compared to previous work. Table A6a shows additional results comparing our model with state-of-the-art method (XMem [8]) on different DAVIS'17 subsets with respect to object size to analyze performance on small objects. It clearly shows that our model is better equipped to capture small-sized objects on conventional benchmarks, demonstrating the efficacy of our unified multi-scale memory matching and decoding framework. It is worth noting that on subsets of extremely small objects ($< 0.3\%$ and $< 0.05\%$), our model consistently outperforms XMem by approximately $2\%$, showcasing the effectiveness of our design.

**Additional comparisons on VOST w/ concurrent works** We also provide comparison with concurrent works [7, 59], and the results are shown in Table A4. In this table, we compare our performance with Cutie [7] by re-training it under

| Approach | Backbone | Multiscale | Parameters (M) | fps |
|---|---|---|---|---|
| AOT | R50 | × | **15.23** | **24.7** |
| CFBI | R101 | ✓ | 66.05 | 5.2 |
| CFBI+ | R101 | ✓ | 74.06 | 10.1 |
| HMMN | R50 | ✓ | 42.76 | 11.3 |
| Ours | R50 | ✓ | <u>39.07</u> | <u>15.3</u> |

Table A5. **Inference cost and model parameter comparison**. All the models are evaluated using 480 as the shortest image size and batch size 1 on a single RTX 3090.

the same settings using the official codebase[2]. We observe our model to perform better than Cutie [7] with considerable margin. Additionally, our model achieves competitive results on VOST compared to RMem [59].

**Addtional comparison on generic VOS benchmarks - LVOS [21]** We experiment with LVOS dataset and report results in Table A6b. We observe that our method achieves SoTA results when compared to models that use the same backbone (=R50). MobileNet-V2 backbone leads to sizable improvements (see AOT-L). We focus on proposing a novel architecture for VOS with transformation, leaving the use of more effective backbones as future work.

**Inference cost and model parameter comparison** As shown in Table A5, our model size is slightly higher than AOT, but it's lighter than previous hierarchical/multiscale methods like CFBI+ and HMMN. Furthermore, our model can perform inference clip-by-clip, it allows us to outperform previous methods supporting multiscale processing (*i.e.*, CFBI, CFBI+ and HMMN) achieving the second best fps. Our model runs at 15 fps, which is sufficient for most real-time applications.

## C.2. Additional ablation studies

**Memory ablations.** In the main submission Section 3.4, we introduced the memory module implemented as a FIFO (first-in-first-out) queue, initially populated with the features of the frame and the ground-truth mask from the initial reference frame. During training, due to computational limitations, we set the memory bank size to 7 and clip-length of 2 for the memory module. However, during inference, when gradients are not computed, we have more GPU memory at our disposal, allowing us to expand the bank size and utilize smaller clip-lengths to include more frames in memory, potentially improving predictions [39]. Note that we apply linear interpolation on RTE to expand it accordingly for larger bank size. Hence, in this subsection, we delve into the impact of employing different bank sizes and clip lengths for the memory module during inference. The corresponding results are presented in Tab. A6c and Tab. A6d. In Tab. A6c, we observe a performance decrease as clip length increases, corroborating the observations made by a prior clip-based method PCVOS [36]. This suggests that larger clip lengths might enable tracking global features across lengthy videos, but it entails less frequent update to the memory possibly compromising the tracking. On the other hand, the results from testing with different bank sizes are shown in Tab. A6d. We observe a slight increase in performance from increasing the bank size from 7 to 9, but decrease in performance with the further increase. This suggests that enabling the model to include more frames in memory during inference results in increase in model's capacity to perform more accurate matching over longer-contexts, but reaches a peak in performance as the model was not trained on larger bank sizes to effectively use them.

[2]https://github.com/hkchengrex/Cutie

**Transformation-aware loss ablations.** As mentioned in the main submission Section 3.5, we compute weights for each frame in the proposed re-weighting objective in different ways. Specifically, we explored three different methods: connected components, center of mass, and masked area. For the masked area, we compute the changes in the foreground area of the binary masks. For the center of mass, we calculate the changes in the relative center of mass. The relative center of mass is calculated as the center of the foreground binary mask, considering the top-left corner of the mask as the origin. Lastly, for the connected components method, we employ `OpenCV` tools to identify the number of distinct groups or regions within the mask. The results of these approaches are presented in Tab. A7a. Notably, simply using the change in the masked area as the weights in the proposed objective yielded the most improvement, resulting in $\sim 2\%$ boost for both $\mathcal{J}^{\text{mean}}$ and $\mathcal{J}_{\text{tr}}^{\text{mean}}$. Based on this observation, we adopted the use of masked area as the default method for computing the weights in our experiments.

Additionally, we note that our transformation-aware reweighting strategy can be applied to individual or different combinations of the component segmentation losses (DICE [34] and Focal [31]) computed at the frame-level. We observe in Tab. A7b that applying re-weighting only to the Focal loss yields the best performance. Therefore, in our experiments, we use re-weighting on the focal loss alone as the default setting to achieve optimal performance.

Lastly, in our novel re-weighting loss we introduced a parameter, denoted as $\tau$, in the normalization process. This parameter allows us to control the smoothness of the weights assigned to each frame. In Fig. A2a, we empirically assess the effectiveness of different values of $\tau$. We observe that smaller values of $\tau$ result in higher performance. However, when $\tau$ falls below 1.0, the weights become overly sharp, causing a decrease in performance. Note that the larger the value of $\tau$ the smoother the weights become. Based on these results, we used $\tau = 1.0$ as our default setting for the re-weighting loss.

| Approach | Object size | | |
|---|---|---|---|
| | $< 0.5\%$ | $< 0.3\%$ | $< 0.05\%$ |
| XMem [8] | 63.9 | 39.7 | 33.8 |
| TAM-VT(Ours) | **64.1** | **43.1** | **35.9** |

(a) Performance on DAVIS'17 val set w.r.t. small object sizes.

| Approach | Backbone | $\mathcal{J}\&\mathcal{F}$ | $\mathcal{J}$ | $\mathcal{F}$ |
|---|---|---|---|---|
| AOT-L [54] | MobileNet-V2 | **60.9** | **55.1** | **66.8** |
| STCN [9] | R50 | 48.9 | 43.9 | 54.0 |
| RDE [28] | R50 | 53.7 | 48.3 | 59.2 |
| XMem [8] | R50 | 52.9 | 48.1 | 57.7 |
| AOT-L† [54] | R50 | 48.2 | 41.9 | 54.5 |
| Ours | R50 | **54.7** | **48.8** | **60.6** |

(b) Performance on LVOS [21] with approaches using R50 [19] as backbone. †reproduced using official code.

| Idx | Bank size | Clip length | $\mathcal{J}_{\text{tr}}$ | $\mathcal{J}$ |
|---|---|---|---|---|
| (1) | 9 | 1 | 35.5 | 47.4 |
| (2) | 9 | 2 | **37.7** | **49.3** |
| (3) | 9 | 3 | 32.7 | 44.2 |
| (4) | 9 | 4 | 32.8 | 44.7 |

(c) Performance on VOST w.r.t. clip lengths during inference.

| Idx | Bank size | Clip length | $\mathcal{J}_{\text{tr}}$ | $\mathcal{J}$ |
|---|---|---|---|---|
| (1) | 7 | 2 | 36.2 | 48.2 |
| (2) | 8 | 2 | 36.2 | 48.1 |
| (3) | 9 | 2 | **37.7** | **49.3** |
| (4) | 10 | 2 | 36.2 | 48.2 |
| (5) | 11 | 2 | 36.4 | 48.9 |
| (6) | 12 | 2 | 35.4 | 47.8 |

(d) Performance on VOST w.r.t. bank size during inference.

Table A6. (a) Note that these models are jointly trained with YouTubeVOS. (b) † denotes models reproduced using official code. (c) We observe a peak in distribution of performance w.r.t. clip lengths during inference. This suggests that while larger clip lengths (going from 1 to 2) allow for accommodating context from longer time spans, it leads to decrease in performance with further increase (2+) as information at granular time-scale is lost. (d) We observe a peak in performance w.r.t. bank sizes. This suggests that, on one hand, increase in bank size leads to increase in model's capacity to perform matching with more number of elements in memory (or longer-context) which translates to increase in performance. But, on the other hand, we observe decrease in performance upon further increase in bank size, suggesting that since the model was not trained on larger bank sizes, it fails to effectively utilize them. Bold number denotes the best performance.

| Method | Re-weighting | $\mathcal{J}_{\text{tr}}$ | $\mathcal{J}$ |
|---|---|---|---|
| Baseline | - | 34.6 | 46.1 |
| (1) | Connected components | 35.4 | 46.3 |
| (2) | Center of mass | 34.9 | 45.2 |
| (3) | Masked area | **36.5** | **48.2** |

(a)

| Method | Re-weighting | | $\mathcal{J}_{\text{tr}}$ | $\mathcal{J}$ |
|---|---|---|---|---|
| | Dice loss | Focal loss | | |
| Baseline | | | 34.6 | 46.1 |
| (1) | ✓ | | 33.2 | 45.4 |
| (2) | | ✓ | **36.5** | **48.2** |
| (3) | ✓ | ✓ | 33.4 | 46.0 |

(b)

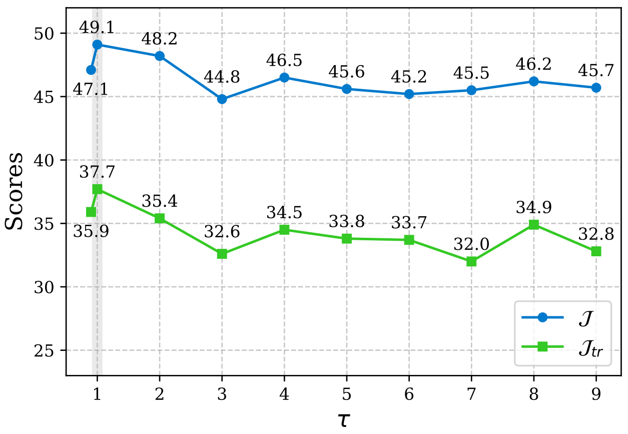
Table A7. **Performance w.r.t. (a) different re-weighting methods used in transformation-aware loss and (b) reweighting combinations of Dice and Focal segmentation loss**. (a) During training, we compute weights (or importance) of each frame while computing total loss using different methods: 1) *Connected components*: weights proportional to relative change in the number of connected components, 2) *Center of mass*: weights proportional to relative change in center of mass, and 3) *Masked area*: weights proportional to relative change in area of foreground objects. We observe best performance when re-weighting our loss with the *Masked Area* method. (b) We observe best performance when reweighting Focal loss (case 2) only, which we use as a default setting for all our experiments.

## C.3. Additional qualitative results

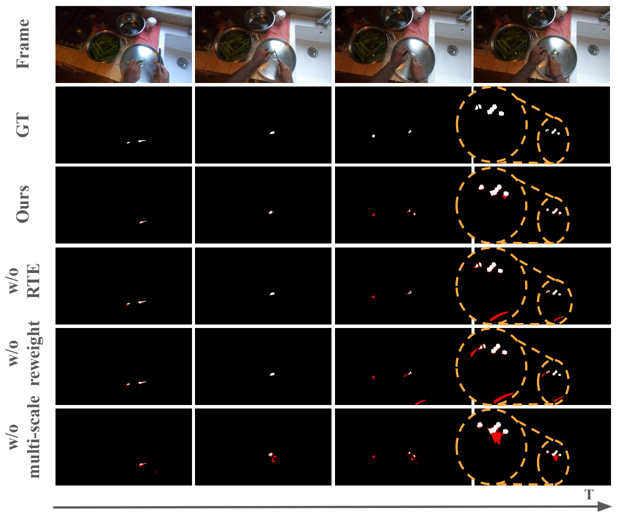
**VOST.** We conduct a qualitative ablation of our proposed approach in Fig. A2b. When comparing our model's performance (third row) and that without multi-scale (or single-scale) matching (last row), we observe that performing matching on a single-scale allows the model to locate the object of interest. However, it struggles to perform precise segmentation, resulting in false positives.

Additionally, when comparing our model (third row), ours without RTE (fourth row) and ours without re-weighting (fifth row), we notice that the baselines fail to track the object after transformations. This leads to false positives again, e.g., from confusing a wristband for a chili. Conversely, our best model adeptly captures the chili without confusion even after the chili is cut, thereby confirming the effectiveness of our design in video object segmentation, particularly in handling transformations.

**VISOR.** To illustrate our prediction results on VISOR, we provide qualitative results in Fig. A3. In this example, the target video contains 6 objects of interest with different appearance and size. Our model is able to, not only track all objects without confusion, but also capture the objects with tiny size. This demonstrates the effectiveness of our model



(a)



(b)

Figure A2. **(a) Performance w.r.t. different $\tau$ used in the transformation-aware reweighting**. We observe peak in performance ($\mathcal{J}_{\text{tr}}$) at $\tau = 1.0$, which we use as a default setting for all our experiments. **(b) Qualitative comparison on VOST for different model ablations.** Best viewed in color; red indicates incorrect predictions.

in challenging scenarios.

## C.4. Additional Visualization and Analysis

**Multi-scale matching** Following the experimental results in Section 4.3, we qualitatively explore the effectiveness of this multi-scale design on VOST in Fig. A4. We select a video featuring small objects, and plot attention heat maps produced by our multi-scale matching encoder. We observe that our model is able to finely match small objects in feature maps at multiple-scales. In Fig. A4, for the top-side results, we select a video with multiple instances. Traditional approaches, which only consider matching on a single scale (i.e., scale 1), may struggle when dealing with objects with similar visual appearances at coarse feature maps. This limitation is evident in the matching attention results at scale 1, where the model might confuse different onions and fail to distinguish between them. In contrast, our multi-scale design, depicted in the scale 2 matching attention results, enables the model to capture subtle visual differences between two onions, focusing more accurately on the correct onion.

For the bottom-side results, we select a video with small instances. In the last attention map results, the model struggles to match the object on the coarse features (scale 1), confusing the object of interest with the human hand and other objects. However, in finer features (scale 2) matching, it successfully attends to the object of interest (the herbs being cut). These findings illustrate that our multi-scale matching not only allows the model to capture smaller objects but also potentially prevents the model from confusing objects with similar visual appearance.

**Transformation-aware re-weighting** In Section 3.5 in the main submission, we propose the transformation-aware re-weighting to enable the model to place greater emphasis on objects during transformations. In this subsection, we provide examples of training data along with the calculated weights $w^t$ for re-weighting to illustrate how this mechanism operates in practical scenarios. These results are showcased in Fig. A5a. The figure depicts a video sequence where a person starts cutting an eggplant in the second frame, leading to a significant change in the foreground mask's area. Our designed re-weighting strategy calculates weights for each frame based on these observed area changes. The weights, displayed in the first row, highlight the highest peak occurring in the second frame, aligning to the frame with the highest degree in object transformation. Consequently, applying these computed weights to re-weight the loss enables us to focus on frames where objects undergo complex transformations.

**Multiplicative Relative Time Encoding** As outlined in Section 3.2 in the main submission, our multi-scale matching encoder integrates relative-time-encoding (RTE) to discern the significance of each frame within the memory. In this paragraph, we present a qualitative demonstration of the learned embeddings in RTE, depicted in Fig. A5b. The results showcase the outcomes obtained using element-wise multiplication in Eq. 3. The findings in both figures illustrate how our RTE highlights the importance of frames during the matching module. Notably, the results reveal that the matching of the current frame heavily relies on its neigh-

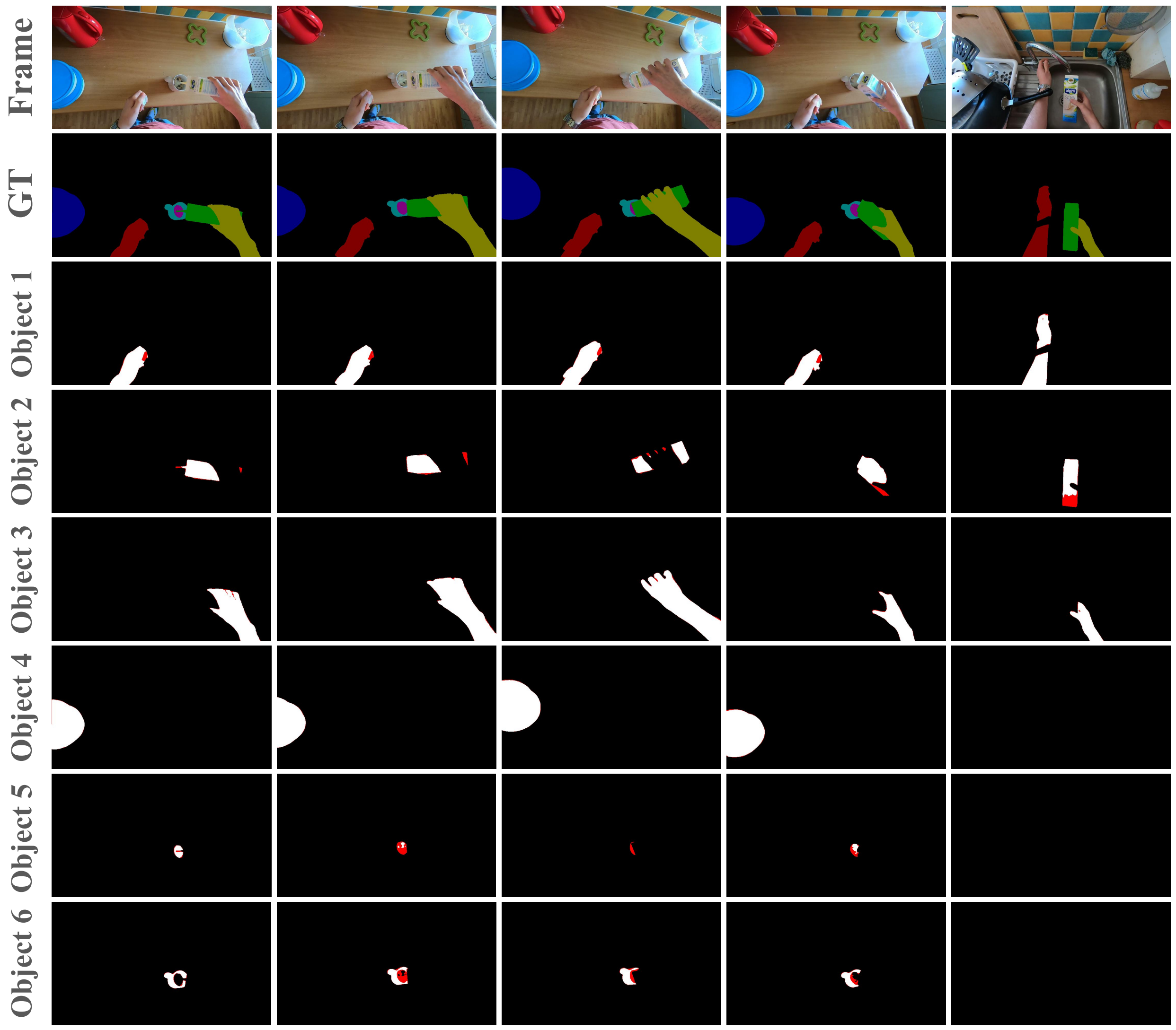


Figure A3. **Qualitative results on VISOR.** Best viewed in color; red indicates incorrect predictions.

boring frames compared to those further away in the sequence. Furthermore, regardless of the number of frames in the memory, the first couple of frames in the memory always hold significance, as it represents the ground truth mask and its neighbouring frame.

## C.5. Limitation and failure cases

While our method excels in video object segmentation with object shape and appearance transformations, yet it still faces various challenges as follows: (1) *Object moves out of frame:* In our video demonstration ("video_demo.mp4"), there is a scenario where the object of interest temporarily moves out of the field of view. Our model, like prior methods such as AOT, struggles to successfully track the object upon its return. When the object is out of view, methods relying on matching the current frame with previous frames encounter difficulties in retrieving it due to the lack of the object presence in the frame history. (2) *Object occlusion with complex transformation*: We illustrate this type of failure in Fig. A6. In this figure, a worker is spreading cement, and our model, along with other methods, can track both the shovel and cement initially. However, once the shovel obstructs the view of the cement in the subsequent frames while the cement undergoes complex transformation, the models lose track of the cement and can only track the shovel. Occlusion events like this, pose challenges for current video object segmentation approaches when objects are undergoing such significant deformations. These cases present significant challenges for existing video object segmentation methodologies. We leave these directions for future research.

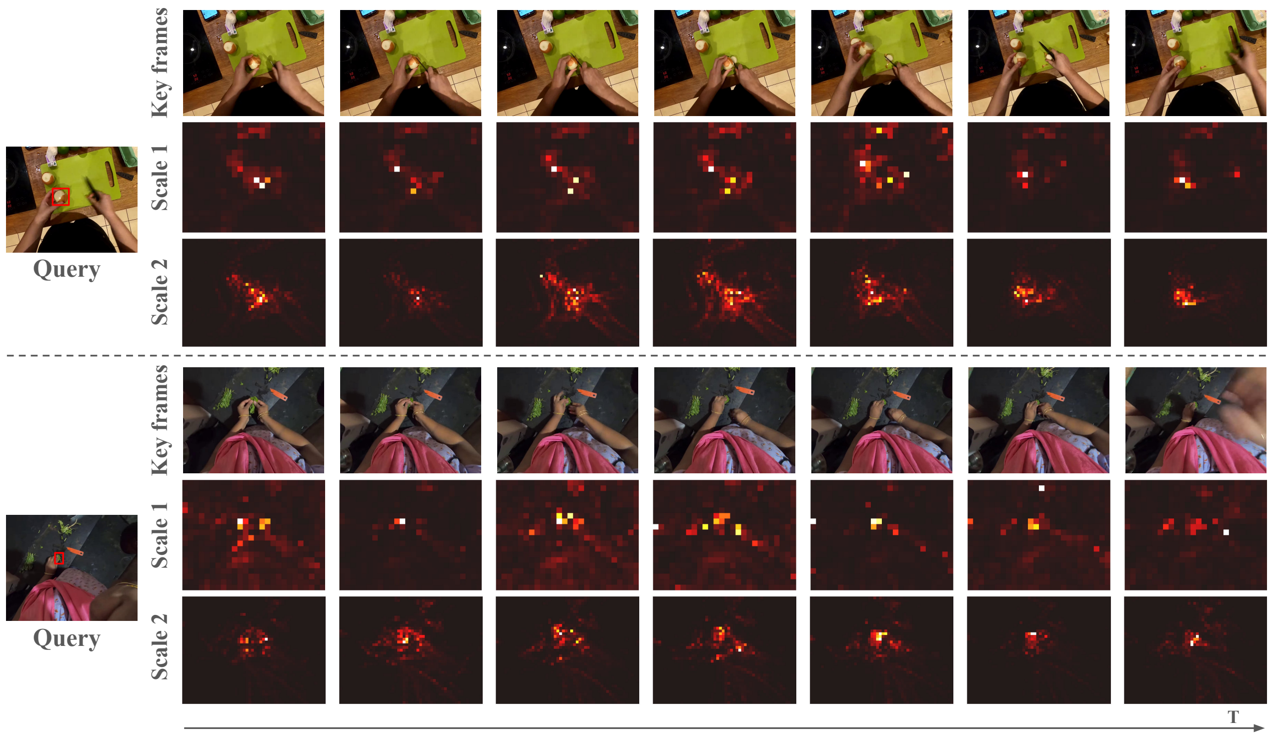
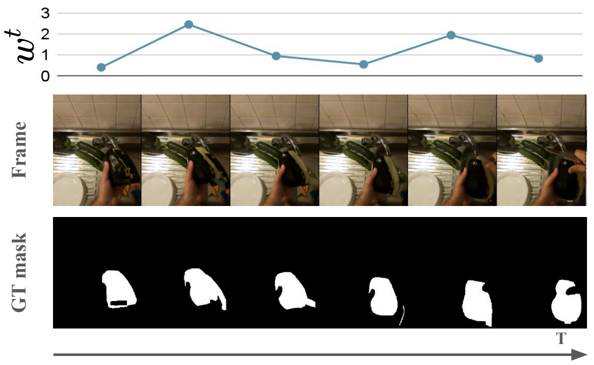


Figure A4. **Visualization of the attention maps** in the multi-scale matching modules. Scale 1 has a resolution of $\frac{1}{32}$ of the input frame, and scale 2 is at $\frac{1}{16}$ resolution. Best viewed in color; lighter colors indicate higher attention scores. The red box in the first frame denotes the object of interest in the query frame.

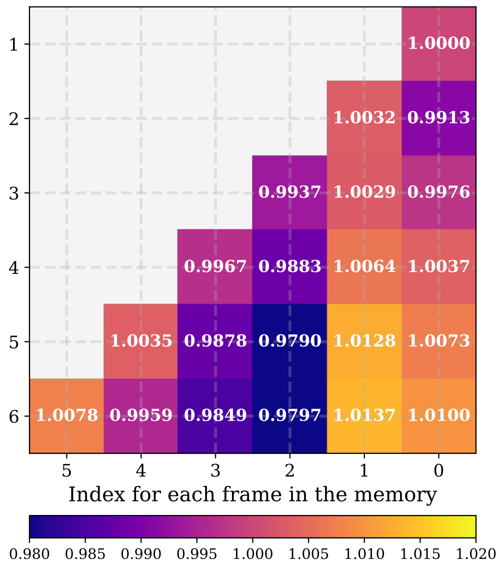
## D. Societal impact

Semi-automatic video object segmentation has multiple positive societal impacts as it can be used for a variety of useful applications, *e.g*. robot manipulation and augmented/virtual reality. The ability to track and segment objects in a class agnostic manner can be used in such application areas to enable better human interaction with the environment and improve the user experience. The ability to track under complex transformations is crucial when deploying in the wild in these applications.

However, as with many artificial intelligence algorithms, segmentation and tracking can have negative societal impacts, *e.g*., through application to target tracking in military systems. To some extent, movements are emerging to limit such applications, *e.g*. pledges on the part of researchers to ban the use of artificial intelligence in weaponry systems. We have participated in signing that pledge and are supporters of its enforcement through international laws.



(a)



(b)

Figure A5. **(a) Visualization of Transformation-aware reweighting. (b) Visualization of RTE.** Value in each grid represents the learned importance score for each frame w.r.t. different number of frames in the memory ($n$). Lighter colors indicate higher scores.

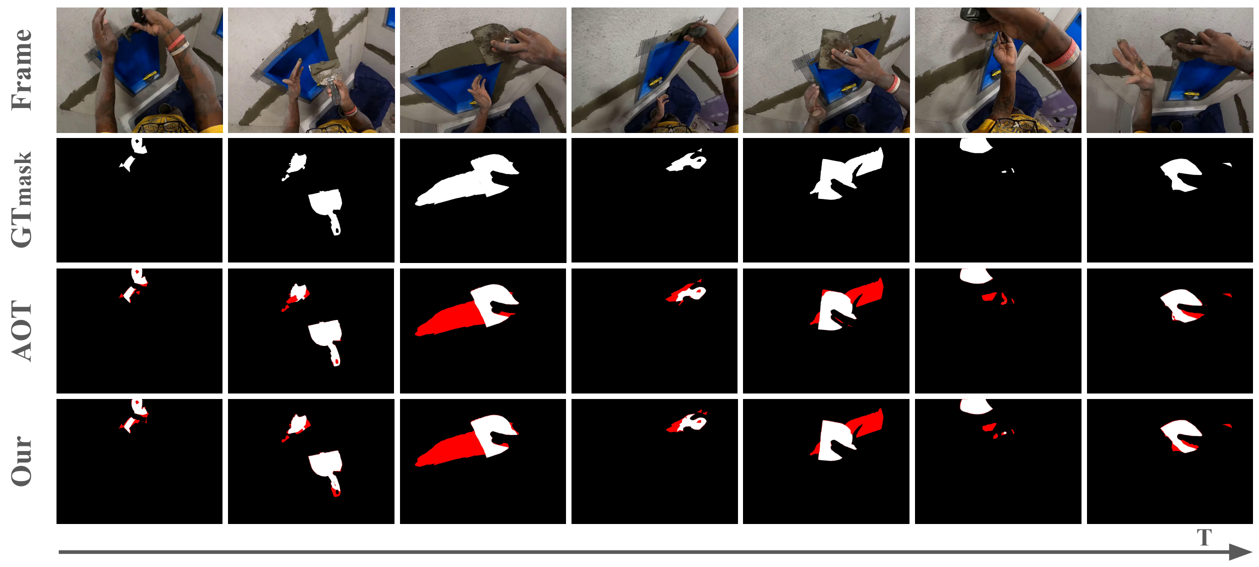


Figure A6. **Failure case.** Best viewed in color; red indicates incorrect predictions. Note that incorrect predictions cover situations where predictions are missing or when objects are wrongly predicted with another object's index, particularly occurring when multiple objects are present in the video. For this figure, both AOT and our model have missed predictions (shown in red color).