Supplementary Material for "SWEM: Towards Real-Time Video Object Segmentation with Sequential Weighted Expectation-Maximization"

Zhihui Lin[1*], Tianyu Yang[2], Maomao Li[2], Ziyu Wang[3], Chun Yuan[4†], Wenhao Jiang[3], and Wei Liu[3]
[1]Department of Computer Science and Technologies, Tsinghua University, Beijing, China
[2]Tencent AI Lab, Shenzhen, China [3]Tencent Data Platform, Shenzhen, China
[4]Tsinghua Shenzhen International Graduate School, Peng Cheng Lab, Shenzhen, China
{lin-zh14@mails, yuanc@sz}.tsinghua.edu.cn tianyu-yang@outlook.com
{limaomao07, cswhjiang}@gmail.com wangziyukobe@163.com wl2223@columbia.edu

1. Overview

We provide the supplementary results for our main paper, which are organized as follows. Section 2 introduces more training details not mentioned in the original paper. In Section 3, we provide more ablation analysis on the number of permutation-invariant features L (i.e., Eq.(12) in the main paper) and the temperature hyperparameter τ in the similarity calculation (i.e. Eq. 2 in this supplementary material). In Section 4, we provide the comparison between the proposed SWEM and STCN [1] on both efficiency and performance with a commonly used GPU, NVIDIA GTX 1080ti. We also report SWEM performance on YouTube-VOS 2019 validation set in Section 5. In Section 6, we analyze the effect of training with different datasets. Finally, we provide qualitative results in Section 7.

2. More Training Details

We set channel dimension as $C = 128$ for the **Key** and $C' = 512$ for the **Value** feature, which are the same as those in STM [9]. We first train SWEM on image datasets for 500k iterations, including COCO [7], MSRA10K [2], ECSSD [13], PASCOL-S [6] and PASCOL-VOC2012 [5], and then on video datasets for 200k iterations, including DAVIS [11], and the YouTube-VOS [14]. When using a single NVIDIA V100 GPU, the image training process takes about 2 days while the video training process needs about 1 day. Following STM and STCN [1], we sample video clips using an 11-times (3471 v.s 300) higher probability for YouTube-VOS than DAVIS 2017 training set. Our SWEM takes about 8GB GPU memory with batch size 4, which reveals its memory-efficient during training and testing.

*Work done during an internship at Tencent AI Lab
†Corresponding Author

Algorithm 1: Calculation of Permutation-Invariant Affinity Features

Input: $\mathcal{K}^{fg}$:$K \times HW$, $\mathcal{K}^{bg}$:$K \times HW$
Output: $\mathbf{S}$:$L \times HW$

function GetPermutInvariantFeatures($\mathcal{K}^{fg}$, $\mathcal{K}^{bg}$):
  // Get top-L affinities for each pixel.
  $\tilde{\mathcal{K}}^{fg/bg}$=torch.topk($\mathcal{K}^{fg/bg}$, k=$L$, dim=1)
  // Initialize $\mathbf{S}^{fg}$ and $\mathbf{S}^{bg}$ as zeros.
  $\mathbf{S}^{fg/bg}$=torch.zeros_like($\tilde{\mathcal{K}}^{fg/bg}$)
  $\mathbf{S}_0^{fg/bg}$=$\tilde{\mathcal{K}}_0^{fg/bg}$
  for l in range(1, L):
    $\mathbf{S}_l^{fg/bg}$=$\mathbf{S}_{l-1}^{fg/bg}$+$\tilde{\mathcal{K}}_l^{fg/bg}$
  $\mathbf{S}$=$\mathbf{S}^{fg}$/($\mathbf{S}^{fg}$+$\mathbf{S}^{bg}$)
  return $\mathbf{S}$

3. More ablation analysis

The number of permutation-invariant features L. Recall that we introduced a permutation-invariant operation to take advantage of affinities between pixels in the current frame and memory features in Section 4.3 of the main paper. The calculation of affinity features is detailed as Eq.(12) in the main paper, which is repeated as:

$$\mathbf{S}_{nl}^{(t)} = \frac{\sum_{j \in \text{topl}(\mathcal{K}_n^{fg,(t)})} \mathcal{K}_{nj}^{fg,(t)}}{\sum_{j \in \text{topl}(\mathcal{K}_n^{fg,(t)})} \mathcal{K}_{nj}^{fg,(t)} + \sum_{j \in \text{topl}(\mathcal{K}_n^{bg,(t)})} \mathcal{K}_{nj}^{bg,(t)}}, \tag{1}$$

where $l = 1, 2, ..., L$, $L \leq K$, and K is the number of foreground or background base features.

Define $\mathcal{K}^{fg}$ and $\mathcal{K}^{bg}$ are affinity matrices between frame features and foreground-background base features. Then the permutation-invariant features with L channels can be

L	FPS	DAVIS 2016 val		DAVIS 2017 val	
		$\mathcal{J}\ \&\ \mathcal{F}\uparrow$	$\mathcal{J}_M\uparrow$	$\mathcal{J}\ \&\ \mathcal{F}\uparrow$	$\mathcal{J}_M\uparrow$
8	40.8	88.5	87.5	81.0	78.6
16	39.1	89.5	88.6	81.4	78.8
32	37.5	**89.6**	**88.8**	81.5	78.8
64	36.4	89.5	88.6	**81.9**	**79.3**
128	33.0	89.6	88.6	81.6	79.0

Table 1. Ablation study on the number of permutation-invariant features L.

calculated with Algorithm 1 in PyTorch [10] style. L is a hyperparameter to control the number of segmentation clues channels and the computation complexity. Table 1 shows the results when setting L as 8, 16, 32, 64 and 128 separately. The segmentation performance is robust to various L except $L = 8$, which brings a significant performance reduction. When increasing L from 64 to 128, the inference speed is dropped by 3.4 FPS while the performance is not improved.

The temperature hyperparameter τ. The similarity in this work is calculated by:

$$\mathcal{K}(\mathbf{a}, \mathbf{b}) = \exp\left(\frac{\mathbf{a}\mathbf{b}^\top/\tau}{\|\mathbf{a}\| \cdot \|\mathbf{b}\|}\right), \tag{2}$$

where the temperature hyperparameter τ controls the range of similarity measures. Table 2 reports the segmentation performance on the DAIVS 2016 and DAVIS 2017 validation sets with different τ. The segmentation performance is sensitive to τ. Too large or too small τ are both not appropriate, especially the large one. The overall $\mathcal{J}\&\mathcal{F}$ on the DAVIS 2017 validation set is dropped from 81.9 to 66.7 when changing τ from 0.05 to 1. As contrast, $\tau = 0.1$ and $\tau = 0.05$ 0.05 is able to achieve satisfying results.

4. Efficiency comparison with STCN

STCN [1] achieves state-of-the-art performance while maintaining an acceptable inference speed (26 FPS with an NVIDIA V100 GPU). However, Similar to STM [9], STCN still stores template features every T frames endlessly. We compare the efficiency between STCN and proposed SWEM with an NVIDIA GTX 1080ti GPU, which is a cheap and commonly used platform.

τ	DAVIS 2016 val		DAVIS 2017 val	
	$\mathcal{J}\ \&\ \mathcal{F}\uparrow$	$\mathcal{J}_M\uparrow$	$\mathcal{J}\ \&\ \mathcal{F}\uparrow$	$\mathcal{J}_M\uparrow$
0.01	88.5	87.6	79.6	77.1
0.02	89.4	88.5	80.5	78.1
0.05	**89.5**	**88.6**	**81.9**	**79.3**
0.1	89.5	88.5	81.7	79.0
0.2	87.5	87.2	72.8	69.9
1	82.1	82.6	66.7	63.9

Table 2. Ablation study on the temperature hyperparameter τ.

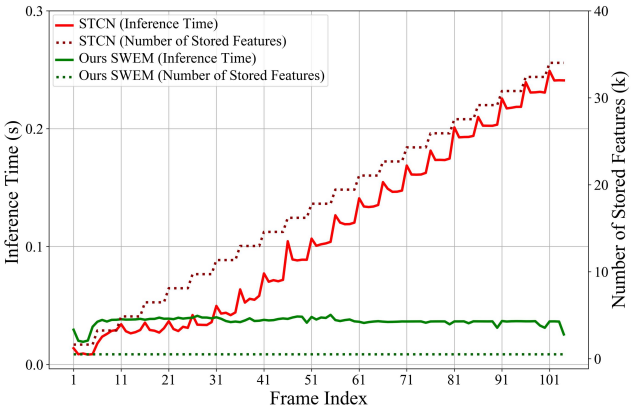

Figure 1. The comparison between STCN and proposed SWEM on per-frame inference time and number of stored features in a single-object segmentation scenario. SWEM has a fixed size of memory features and stable inference time. Methods are evaluated with an NVIDIA GTX 1080ti GPU.

Figure 1 shows the per-frame inference time and memory capacity of STCN and SWEM in a single-object segmentation scenario. It clearly reflects the negative effects of growing memory features on the efficiency and storage during the long-term segmentation. Compared to STCN, our

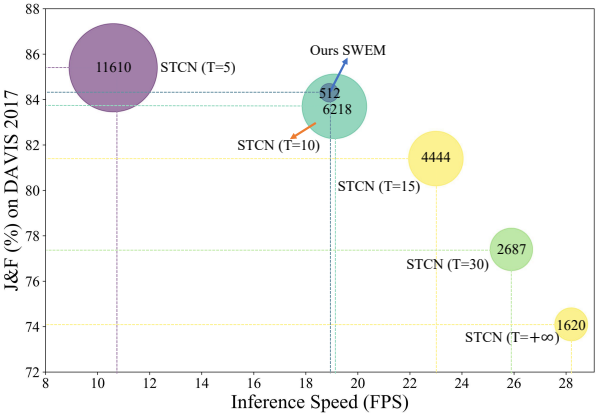

Figure 2. The comparison between STCN and proposed SWEM on inference speed and performance in a multi-object segmentation scenario. We set different memory interval (T=5, 10, 15, 30, $+\infty$) for STCN. The area of the circle represents the number of per-frame memory features involved in matching for each object. SWEM achieves a better trade-off between performance and efficiency than STCN.

[width=10em]Training DataMethod	DAVIS 2016 val $\mathcal{J}$ & $\mathcal{F}$			DAVIS 2017 val $\mathcal{J}$ & $\mathcal{F}$		
	STM [9]	STCN [1]	**Ours SWEM**	STM [9]	STCN [1]	**Ours SWEM**
DAVIS	74.5	84.6	**88.1**	48.6	71.2	**77.2**
DAVIS + YTVOS	88.2	**89.8**	89.5	80.0	81.1	**81.9**
Image + (DAVIS + YTVOS)	89.8	91.2	**91.3**	81.6	**84.5**	84.3

Table 3. Analysis of training on different datasets. We train STM [9], STCN [1] and proposed SWEM under three different conditions: 1) only training on video data DAVIS, 2) training on video datasets DAVIS and YouTube-VOS, and 3) pretraining on image data first and then training on DAVIS and YouTube-VOS.

SWEM stores far fewer memory features and has a stable inference time cost.

Figure 2 shows the efficiency and performance comparison between STCN and SWEM in a multi-object segmentation scenario. The area of the circle represents the number of per-frame per-object memory features. We also set different memory interval (T=5, 10, 15, 30, $+\infty$) for STCN. When $T = 10$, STCN gets a similar inference speed and performance with SWEM, but with much more memory features. When $T = 15$, the performance has a significant drop. If STCN only stores features of the first frame ($T = +\infty$), the overall performance is dropped from 85.4 to 74.1.

The above results demonstrate the superiority of SWEM in terms of efficiency and performance with common hardware.

5. Comparisons on YouTube-VOS 2019

To further evaluate the effectiveness of our method, we also carry out experiments on the YouTube-VOS 2019 dataset. Note that the YouTube-VOS 2019 validation set contains 507 videos, while the 2018 version has 474 videos. Specifically, our SWEM is trained on the YouTube-VOS 2019 training set and evaluated on the validation set through the official evaluation server. We compare SWEM with recent state-of-the-art methods in Table 4, including transformer-based methods SSTVOS [3], JOINT [8] and AOT [15]. SWEM achieves the 82.6% overall score while maintaining the real-time inference speed.

Method	$\mathcal{G}$	seen		unseen	
		$\mathcal{J}_M \uparrow$	$\mathcal{F}_M \uparrow$	$\mathcal{J}_M \uparrow$	$\mathcal{F}_M \uparrow$
STM [9]	79.2	79.6	83.6	73.0	80.6
CFBI [16]	81.0	80.6	85.1	75.2	83.0
LWL [4]	81.0	79.6	83.8	76.4	84.2
SSTVOS [3]	81.8	80.9	85.3	76.6	84.4
HMMN [12]	82.5	81.7	86.1	77.3	85.0
JOINT [8]	82.8	80.8	84.8	**79.0**	86.6
AOT [15]	**83.6**	**82.2**	**86.9**	78.3	**86.9**
SWEM*	82.6	82.0	86.1	77.2	85.2

Table 4. Comparison with state-of-the-art methods on the YouTube-VOS 2019 validation dataset. We report all of the mean Jaccard ($\mathcal{J}$), the boundary ($\mathcal{F}$) scores for seen and unseen categories, and the overall scores $\mathcal{G}$. Besides, we use '*' to indicate those methods with an inference speed $>$ 20 FPS. Note SSTVOS, JOINT and AOT are transformer-based methods.

6. Impact of Training Data

Table 3 presents the performance of various models on DAVIS 2016 and DAVIS 2017 validation sets with different training datasets. It can be seen that both pre-training on image data and using additional YouTube-VOS video data can boost performance. Note that our SWEM trained only on DAVIS data outperforms STM [9] and STCN [1] under the same setting with a large margin. Furthermore, STM [9] obtains 48.6% $\mathcal{J}\&\mathcal{F}$ on DAVIS 2017 validation set when it is trained only with DAVIS data. Authors claim it is due to the high risk of over-fitting on small datasets. However, STCN gets a much higher overall performance 71.2% $\mathcal{J}\&\mathcal{F}$ than STM. We argue the reason for this performance gap is that the features involved in matching in STM come from different feature spaces, whereas those features in STCN and SWEM are from the same feature spaces. Moreover, the direct usage of affinity features also accelerates the convergence of SWEM. SWEM gets similar performance with STCN when adopting additional YouTube-VOS video data or image data. It is worth noting that SWEM stores fixed-size (512) memory features which are much fewer than those in STCN. As an inference, a single frame with a size 480×864 produces 1,620 features.

7. Qualitative Results

We further provide the qualitative comparison between STM [9] and our SWEM on the validation set of the DAVIS 2017 and YouTube-VOS in Figure 3, which demonstrates the superiority of our SWEM.

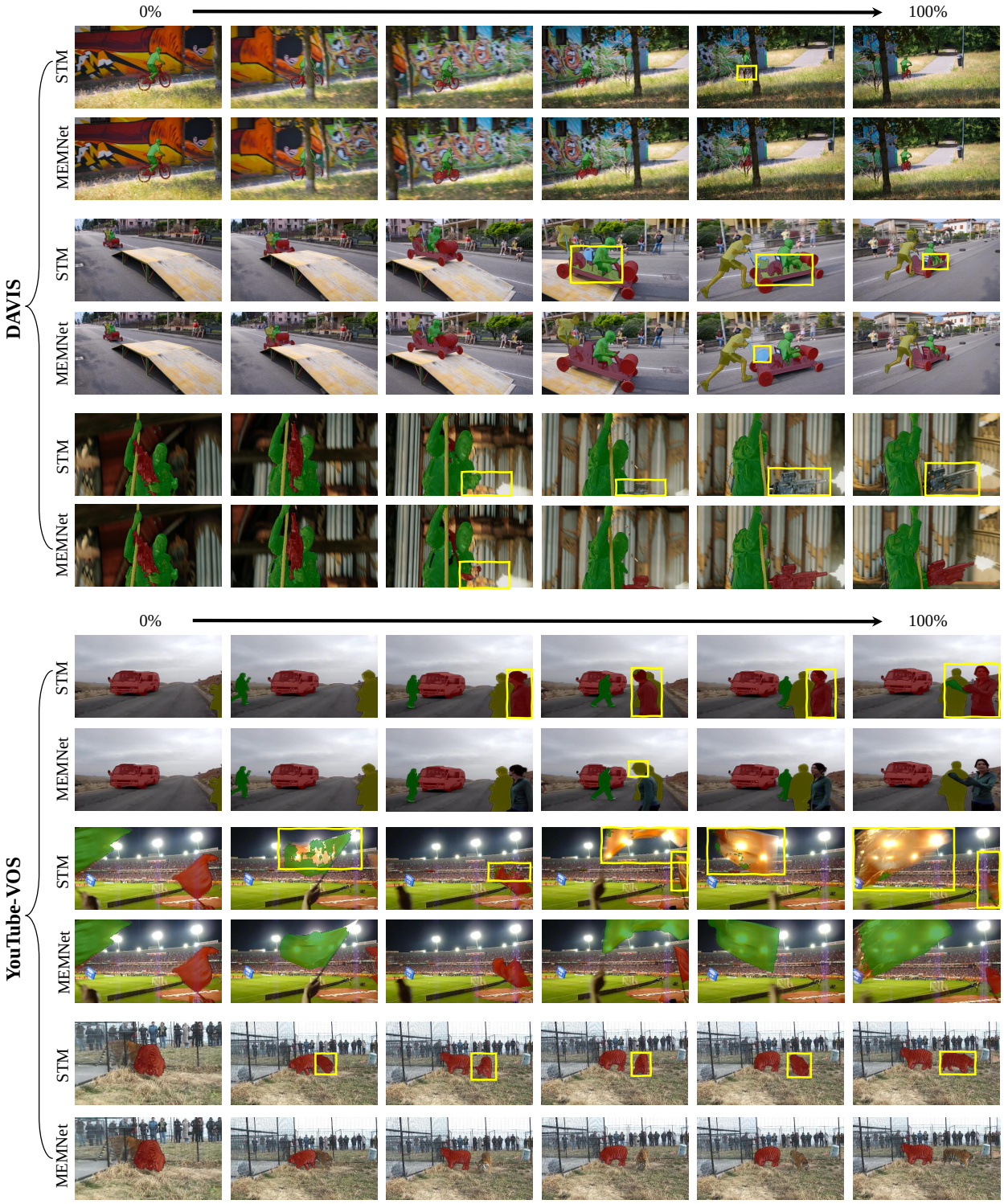

Figure 3. The qualitative comparison between STM [9] and our SWEM on the DAVIS 2017 and YouTube-VOS. The obvious failure segmentation is indicated by yellow bounding boxes. Our SWEM is robust with rapid motion and similar distractors.

References

[1] 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. 1, 2, 3

[2] Ming-Ming Cheng, Niloy J. Mitra, Xiaolei Huang, Philip H. S. Torr, and Shi-Min Hu. Global contrast based salient region detection. *IEEE TPAMI*, 37(3):569–582, 2015. 1

[3] Brendan Duke, Abdalla Ahmed, Christian Wolf, Parham Aarabi, and Graham W Taylor. Sstvos: Sparse spatiotemporal transformers for video object segmentation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 5912–5921, 2021. 3

[4] Bhat G. et al. Learning what to learn for video object segmentation. In *European Conference on Computer Vision (ECCV)*, 2020. 3

[5] Mark Everingham, Luc Van Gool, Christopher KI Williams, John Winn, and Andrew Zisserman. The pascal visual object classes (voc) challenge. *International journal of computer vision*, 88(2):303–338, 2010. 1

[6] Yin Li, Xiaodi Hou, Christof Koch, James M Rehg, and Alan L Yuille. The secrets of salient object segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 280–287, 2014. 1

[7] 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 *European conference on computer vision*, pages 740–755. Springer, 2014. 1

[8] Yunyao Mao, Ning Wang, Wengang Zhou, and Houqiang Li. Joint inductive and transductive learning for video object segmentation. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 9670–9679, 2021. 3

[9] Seoung Wug Oh, Joon-Young Lee, Ning Xu, and Seon Joo Kim. Video object segmentation using space-time memory networks. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, October 2019. 1, 2, 3, 4

[10] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. Pytorch: An imperative style, high-performance deep learning library. *Advances in neural information processing systems*, 32:8026–8037, 2019. 2

[11] Jordi Pont-Tuset, Federico Perazzi, Sergi Caelles, Pablo Arbeláez, Alex Sorkine-Hornung, and Luc Van Gool. The 2017 davis challenge on video object segmentation. *arXiv preprint arXiv:1704.00675*, 2017. 1

[12] Hongje Seong, Seoung Wug Oh, Joon-Young Lee, Seongwon Lee, Suhyeon Lee, and Euntai Kim. Hierarchical memory matching network for video object segmentation. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 12889–12898, 2021. 3

[13] Jianping Shi, Qiong Yan, Li Xu, and Jiaya Jia. Hierarchical image saliency detection on extended cssd. *IEEE transactions on pattern analysis and machine intelligence*, 38(4):717–729, 2015. 1

[14] Ning Xu, Linjie Yang, Yuchen Fan, Jianchao Yang, Dingcheng Yue, Yuchen Liang, Brian Price, Scott Cohen, and Thomas Huang. Youtube-vos: Sequence-to-sequence video object segmentation. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 585–601, 2018. 1

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

[16] Yang Y. Yang Z., Wei Y. Collaborative video object segmentation by foreground-background integration. In *European Conference on Computer Vision (ECCV)*, 2020. 3