# Boosting Video Object Segmentation via Space-time Correspondence Learning
## *Supplemental Material*

Yurong Zhang[1*], Liulei Li[2*], Wenguan Wang[2†], Rong Xie[1], Li Song[1], Wenjun Zhang[1]
[1]School of Electronic Information and Electrical Engineering, Shanghai Jiao Tong University [2]ReLER, CCAI, Zhejiang University
https://github.com/wenguanwang/VOS_Correspondence

The appendix is **structured** as follows:

- §A provides the pseudo code of our correspondence matching based training strategy.
- §B offers analysis regarding the object-level matching process.
- §C presents the comparison of loss curve.
- §D gives visualization of correspondence matching.
- §E shows additional qualitative results comparing proposed methods to baselines and recent state-of-the-arts on DAVIS2017$_{test}$ [6] and YouTube-VOS2019$_{val}$ [7] dataset.
- §F supplements more implementation details of training.
- §G broadly discusses the limitation of our approach and outlines a few directions of future work.

## A. Pseudocode

The pseudo-code of the pixel-level and object-level correspondence learning is given in Algorithm S1 and S2 respectively.

## B. Object-level Matching Accuracy

Since matching accuracy is a critical factor for object-level correspondence learning, we measure the object-level matching process qualitatively. Two visual cases are shown in Fig. S1. The performance sufficiently demonstrates robustness of object-level matching, thus ensuring the efficacy of object-level representative learning.

## C. Loss Curve Analysis

We plot loss curves of $\mathcal{L}_{\text{SEG}}$, $\mathcal{L}_{\text{OCL}}$ and $\mathcal{L}_{\text{PCL}}$ (Eq. 12) of STCN+`Ours` and the original segmentation loss of STCN. Note that the fluctuation of the segmentation loss is caused by the gradually enlarged frame interval — the trend of our segmentation loss is similar to that of STCN, while our segmentation loss is lower. This confirms the efficacy of proposed correspondence-aware training strategy.

[1]The first two authors contribute equally to this work.
[2]Corresponding author.

Figure S1. Qualitative results of object-level matching performance on DAVIS2017 [6] and YouTube-VOS2019$_{val}$ [7].

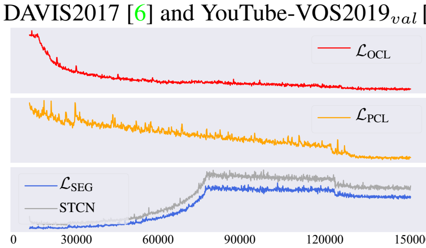


Figure S2. Loss curve comparison of STCN+`Ours` and STCN.

## D. Visualization of Correspondence Matching

Fig. S3 shows matching response between distant frames $I_t$ and $I_\tau$ in DAVIS2017 [6]. As seen, baseline model STCN [2] often suffers from mismatching when appearance-similar objects present. However, after our correspondence-aware training, the VOS model is able to build more precise and robust cross-frame correspondence. These results intuitively verify the effectiveness of our correspondence-aware training strategy.

## E. Additional Qualitative Results

We show additional VOS results on two datasets, namely YouTube-VOS2019$_{val}$ [7] and DAVIS2017$_{test}$ [6] in Fig. S4-S7. As can be seen, our space-time correspondence-aware training paradigm indeed boosts the segmentation performance of STCN [2] and XMem [1], even in challenging scenarios. We also provide visual comparisons with recent state-of-the-art methods, *i.e*., AOT[8], RDE[4], PCVOS [5], in Fig. S8-S9. As seen, our algorithm consistently yields more precise segmentation results compared with these powerful competitors. Notably, our approach is more favored in distinguishing between appearance-similar objects. We attribute this to the effect of our correspondence-aware training scheme.

## F. More Implementation Detail

During training, we use a batch size of 16 and an image crop size of $384 \times 384$. All backbones are initialized using corresponding weights pre-trained on ImageNet-1K[3], while remaining layers are randomly initialized. The initial learning rate is set to 5e-5 and scheduled according to a "step" policy.

## G. Discussion

**Limitation.** Currently, we only demonstrate the critical role of space-time correspondence learning in training matching-based VOS solutions. It is unclear whether our algorithm can contribute to other VOS algorithms. We believe it is highly necessary to deeply embed space-time correspondence learning into both network architecture design and training scheme of VOS models, as correspondence matching addresses the dense-tracking nature of VOS. Moreover, our current correspondence-aware VOS training algorithm can evolve with the advance of the field of unsupervised correspondence matching. In our practice, we find that our model sometimes struggles in handling fast-moving objects.

**Broader Impact.** This work can benefit the wide application scenarios of VOS, such as video editing, intelligent conferencing, and augmented reality.

**Future Work.** The aforementioned limitations demonstrate the directions of our future work. Moreover, it is also interesting to explore the extra use of massive unlabeled video data within our framework, since our correspondence-matching learning operates without annotations.

**Algorithm S1** Pseudo-code for pixel-level consistency in a PyTorch-like style.

```
# k_t1, k_t2: representation of two successive frames
# k_r: representation of the remote frame
# R: radius of the sampling grid

def grid_sample(key, R):
    h, w = H // R, W // R
    x_idx = arange(0, W, R).view(1, 1, w)
    y_idx = arange(0, H, R).view(1, h, 1)

    # random offsets
    x_idx = x_idx + randint(0, R, (B, 1, 1))
    y_idx = y_idx + randint(0, R, (B, 1, 1))

    # B × w × h
    xy_idx = x_idx + y_idx * W
    # B × wh × C
    xy_idx = xy_idx.view(B,-1,1).expand(-1,-1,C)

    # BHW × C → B × HW × C
    key = key.reshape(B, HW, C)
    # Bhw × C
    key = (gather(key,dim=1,index=xy_idx).flatten(0,1)

    return key

def pixel_level_consistency(k_t1, k_t2, k_r, R=8):
    # BHW × C → Bhw × C
    k_r = grid_sample(k_r, radius=R)

    #======== estimate the affinity (Eq.5) ========#
    # BHW × Bhw
    A_t1_r = softmax(k_t1 * k_r.transpose(), dim=1)
    A_t2_r = softmax(k_t2 * k_r.transpose(), dim=1)

    #======== generate pseudo label (Eq.6) ========#
    # BHW, 1
    pseudo = argmax(A_t1_r, dim=1)

    #==== pixel-level consistency loss (Eq.7) =====#
    l_pc = nll_loss(A_t2_r, pseudo)


    return l_pc
```

**Algorithm S2** Pseudo-code for object-level coherence in a PyTorch-like style.

```
# res4_p, res4_q: features of two distant frames
# box_p, box_q: bounding boxes of objects
# D: dimension of object-level representation
# N: number of objects drawn from 𝒫
# PROJ: project head to map object representations

def object_level_coherence(res4_p, res4_q, box_p,
   box_q, N):
    #===== object-level representation (Eq.8) =====#
    # B × C × H × W → B × N_p × D × h × w
    o_p = PROJ(roi_align(res4_p, box_p))
    # B × N_p × D × h × w → B × N_p × D
    o_p = avg_pool(o_p, kernel_size=(h, w)).squeeze()
    # B × C × H × W → B × N_q × D × h × w
    o_q = PROJ(roi_align(res4_q, box_q))
    # B × N_q × D × h × w → B × N_q × D
    o_q = avg_pool(o_q, kernel_size=(h, w)).squeeze()

    # randomly drawn subset
    idx_p = randint(N)
    # B × N_p × D → B × N × D
    o_p = index_select(o_p, dim=1, index=idx_p)

    #========= bipartite matching (Eq.9) ==========#
    # B × N × N_q
    A_p_q = hungarian_matcher(o_p, o_q)
    #======== counterpart alignment (Eq.10) ========#
    # B × N
    indices = argmax(A_p_q, dim=-1)

    # indices after flatten the batch
    indices = [indice + b_idx * indices.size(0) for
         b_idx, indice in enumerate(indices)]
    indices = stack(indices, dim=0).flatten(0, 1)

    # cross batch affinity for more negative samples
    affinity = softmax(o_p.flatten(0,1) * o_q.
         flatten(0,1).transpose(), dim=1)

    #==== object-level coherence loss (Eq.11) =====#
    l_oc = nll_loss(affinity, indices)

    return l_oc
```

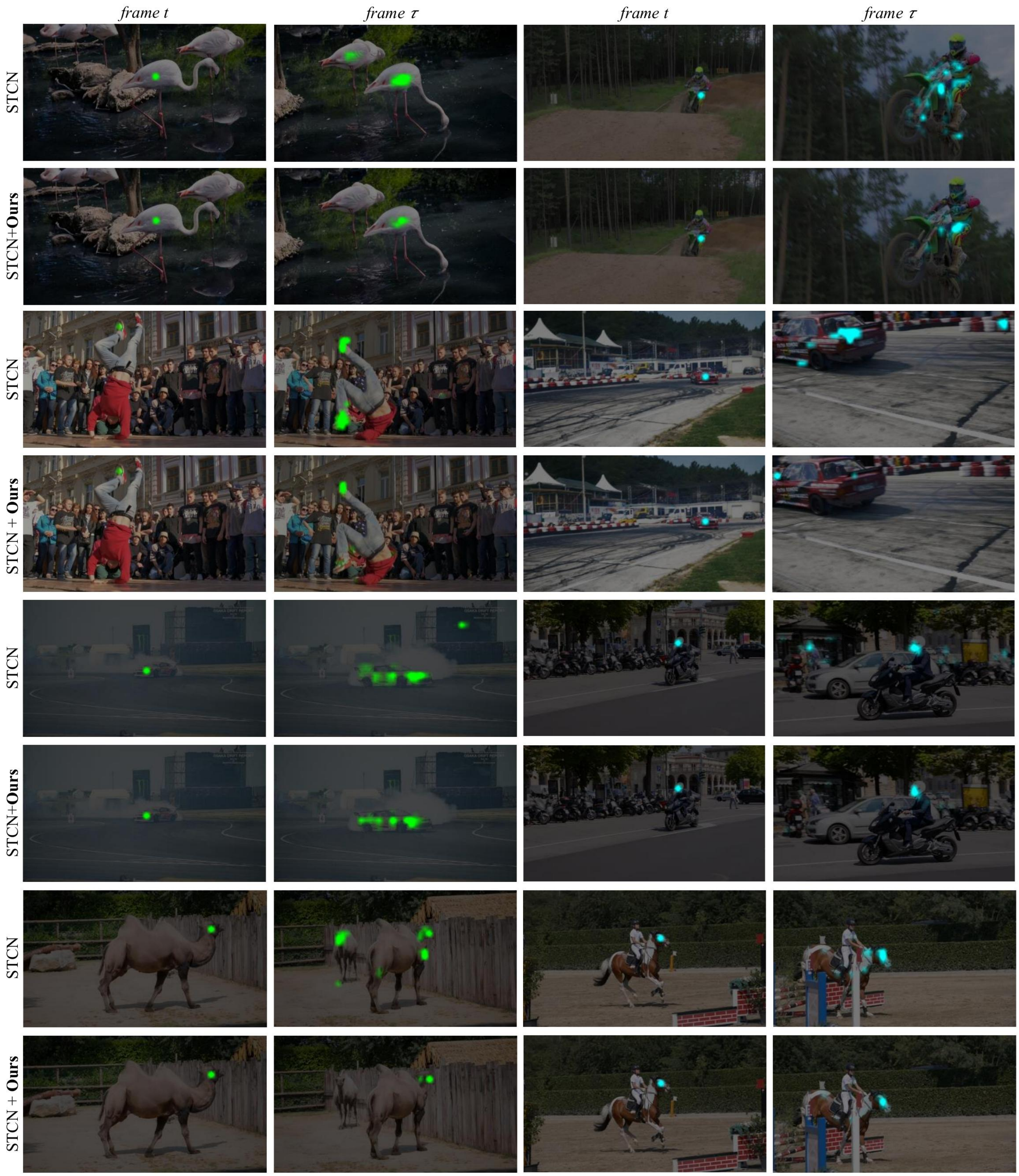


Figure S3. **Correspondence matching results** for STCN+`Ours` and STCN [2] on DAVIS2017 [6] dataset, where the query pixel and the matching response in another distant frame are highlighted.


Figure S4. **More qualitative comparisons** between STCN+`Ours` and STCN [2] on YouTube-VOS2019$_{val}$ [7] and DAVIS2017$_{test}$ [6].

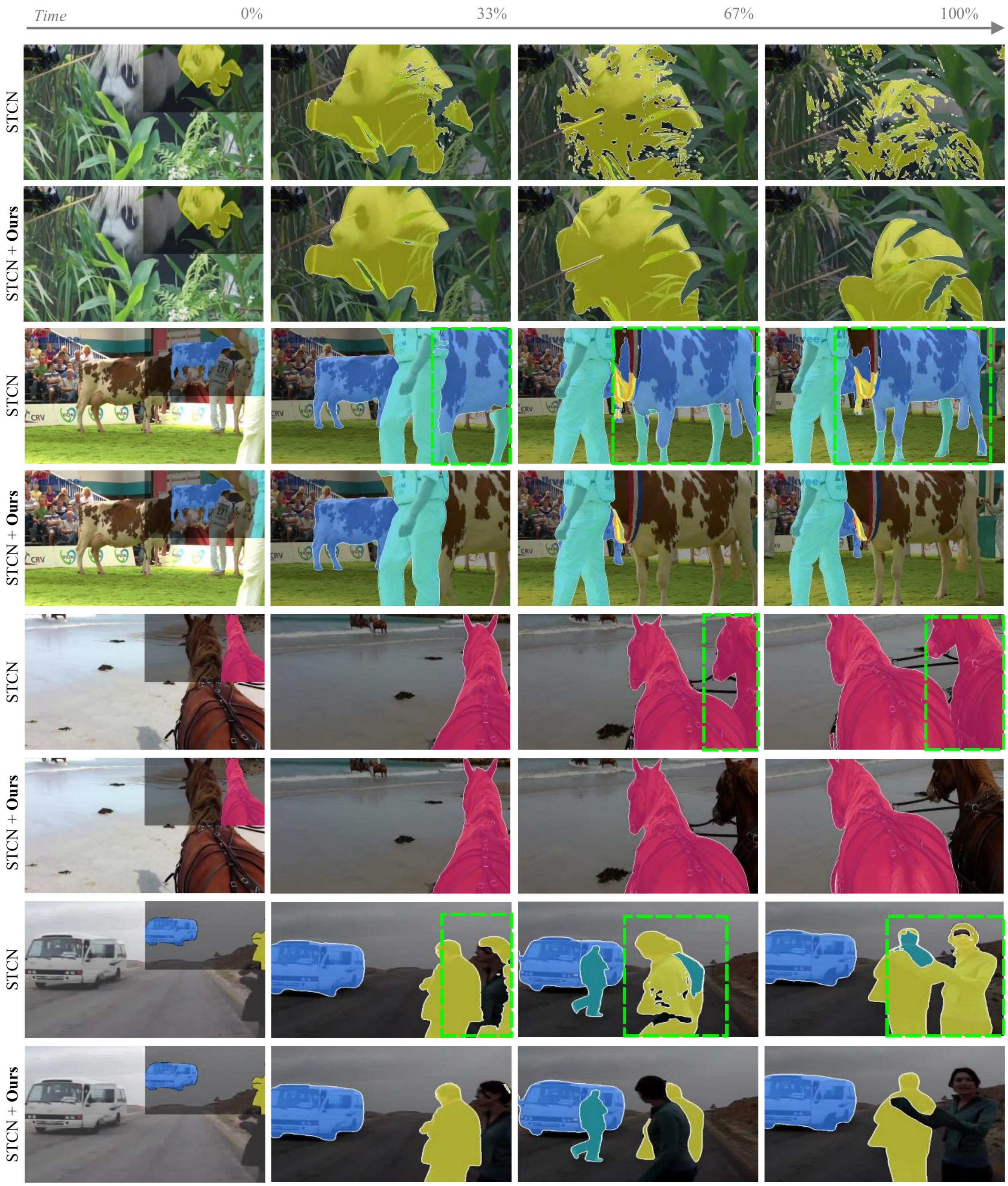


Figure S5. **More qualitative comparisons** between STCN+**`Ours`** and STCN[2] on YouTube-VOS2019$_{val}$ [7].

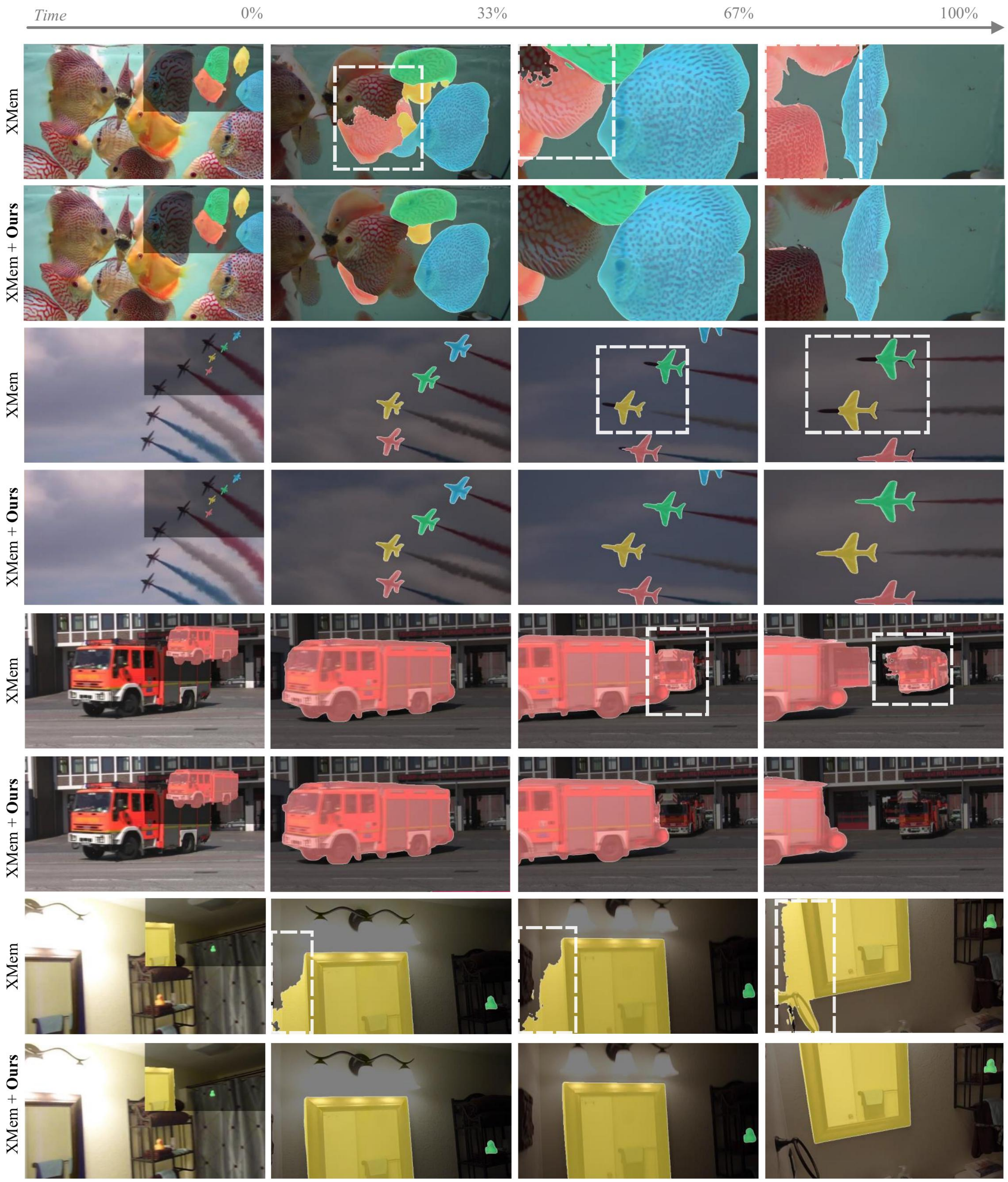


Figure S6. **More qualitative comparisons** between XMem+**Ours** and XMem[1] on YouTube-VOS2019$_{val}$ [7].

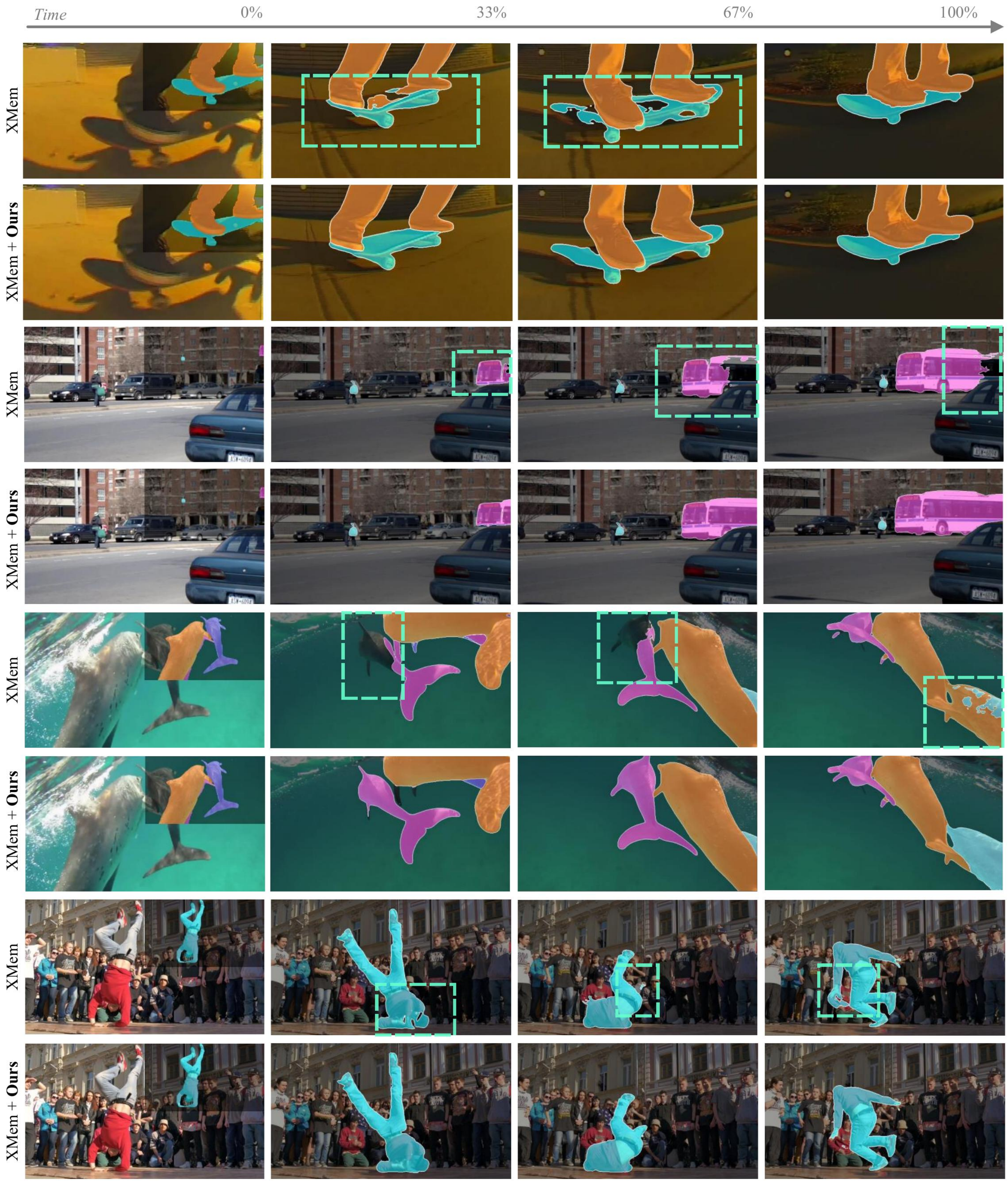


Figure S7. **More qualitative comparisons** between XMem+`Ours` and XMem [1] on YouTube-VOS2019$_{val}$ [7] and DAVIS2017$_{val}$ [6].


Figure S8. **More qualitative comparisons** between XMem+**`Ours`** and AOT [8], RDE [4], PCVOS [5] on YouTube-VOS2019$_{val}$ [7].

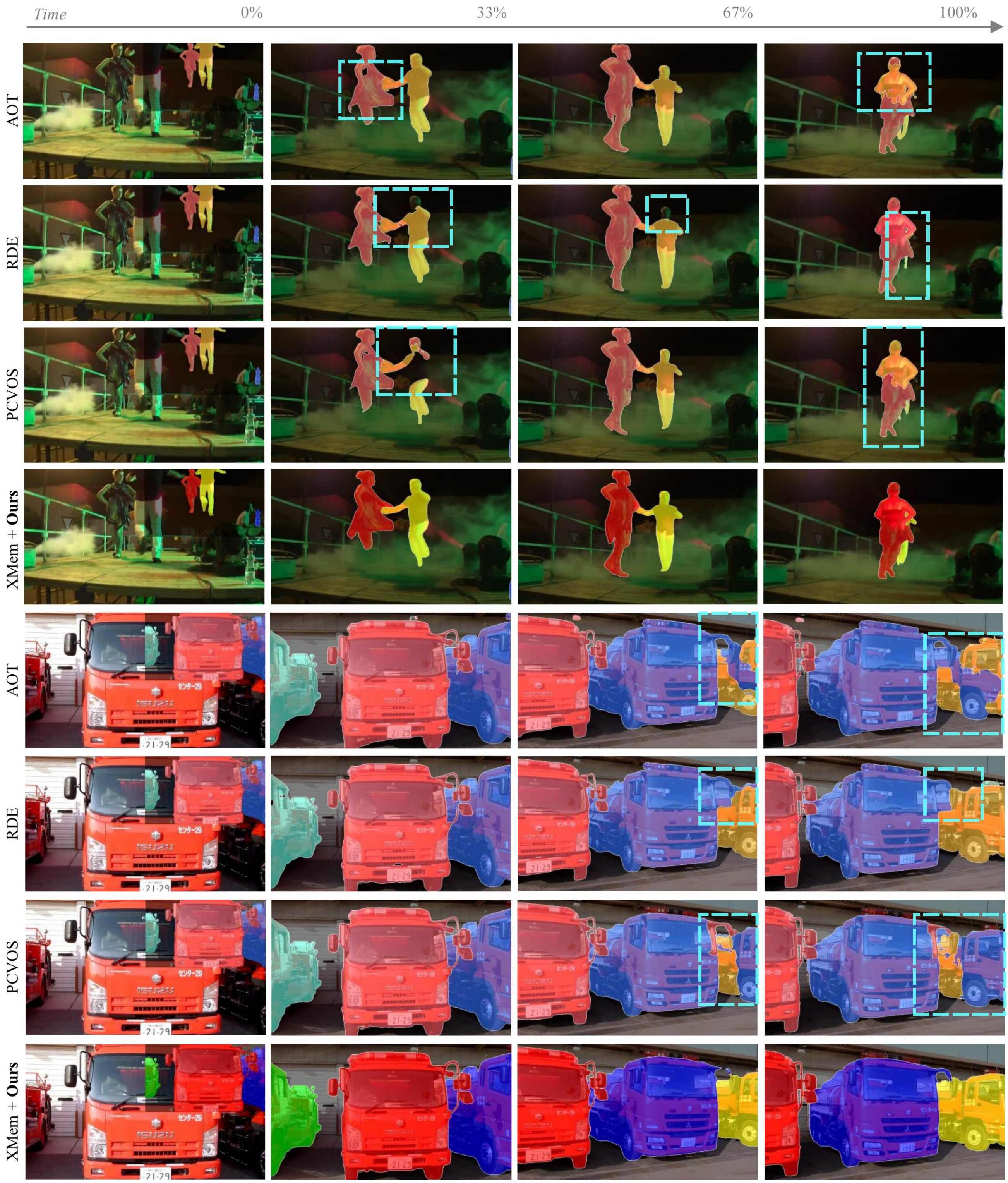


Figure S9. **More qualitative comparisons** between XMem+**Ours** and AOT[8], RDE[4], PCVOS[5] on YouTube-VOS2019$_{val}$ [7].

## References


[1] Ho Kei Cheng and Alexander G. Schwing. Xmem: Long-term video object segmentation with an atkinson-shiffrin memory model. In *ECCV*, 2022. 1, 7, 8

[2] 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, 4, 5, 6

[3] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *CVPR*, 2009. 2

[4] Mingxing Li, Li Hu, Zhiwei Xiong, Bang Zhang, Pan Pan, and Dong Liu. Recurrent dynamic embedding for video object segmentation. In *CVPR*, 2022. 1, 9, 10

[5] Kwanyong Park, Sanghyun Woo, Seoung Wug Oh, In So Kweon, and Joon-Young Lee. Per-clip video object segmentation. In *CVPR*, 2022. 1, 9, 10

[6] 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, 4, 5, 8

[7] 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 *ECCV*, 2018. 1, 5, 6, 7, 8, 9, 10

[8] Zongxin Yang, Yunchao Wei, and Yi Yang. Associating objects with transformers for video object segmentation. In *NeurIPS*, 2021. 1, 9, 10