Appendices

In this supplementary material, we discuss the broader impact (Section A), report experiment details including dataset, model, and training details (Section B), provide more details of Invariant Slot Attention, as mentioned in the main paper (Section C), perform extra ablation studies on input resolution and generalization performance of our model (Section D), and finally present extra visualizations (Section E). In the last section, we provide a visual comparison between models in our ablation studies as well as previous work. Moreover, we provide an analysis of our failure cases. We attached the code to the supplementary material, and we will share it publicly upon acceptance. We also prepared an HTML file containing the GIFs of our matched segmentation results.

A Broader Impact

We proposed an object-oriented approach that can be applied to videos of real-world environments. We trained our model on the Youtube-VIS 2019 dataset that includes videos of humans. The proposed work can be used to locate and segment multiple instances of a wide variety of objects including all kinds of animals and humans from videos. While progress in this area can be used to improve not only human life but also, for example, wildlife, we acknowledge that it could also inadvertently assist in the creation of computer vision applications that could potentially harm society.

B Experiment Details

B.1 Dataset Details

We eliminated the black borders for all videos in YTVIS19 dataset. Since we propose a self-supervised model, we merge the available splits on datasets for training. For all datasets except YTVIS, we use the available validation splits for evaluation. The annotations for the validation split are missing on the YTVIS, therefore, we use a subset of 300 videos from the train split for evaluation. We will share the indices of selected videos for future comparisons together with the code. For evaluation, we upsample the segmentation masks to match the resolution of the original input frames by using bilinear interpolation

B.2 Model Details

Feature Extractor (ϕ_{DINO}): We use the ViT-B/14 architecture with DINOv2 pretraining [5] as our default feature extractor. Our feature vector is the output of the last block without the CLS token. We add positional embeddings to the patches and then drop the tokens.

Spatial Binding($\psi_{\text{s-bind}}$): We project the feature tokens from ϕ_{DINO} to slot dimension $D_{\text{slot}} = 128$, with a 2-layer-MLP, followed by layer normalization. Then the slots and the projected tokens are passed to the Invariant Slot Attention (ISA) (as detailed in Section C) as input. After slot attention, slots are updated with a GRU cell. Following the sequential update, slots are passed to a residual MLP with a hidden size of $4 \times D_{\text{slot}}$. All projection layers (p, q, k, v, g) have the same size as slots, *i.e.* D_{slot}. We repeat the binding operation 3 times. We multiply the scale parameter $\mathbf{S}_s$ by $\delta = 5$ to prevent relative grid $\mathbf{G}_{\text{rel}}$ from containing large numbers.

We use the following initializations for the learnable parameters: $\mathbf{G}_{\text{abs}}$, a coordinate grid in the range $[-1, 1]$; slots $\mathbf{z}$, Xavier initialization [3]; slot scale $\mathbf{S}_s$ and slot position $\mathbf{S}_p$, a normal distribution.

Temporal Binding ($\psi_{\text{t-bind}}$): We use a transformer encoder with 3 layers and 8 heads [8] for temporal binding. The hidden dimension of encoder layers is set to $4 \times D_{\text{slot}}$. We initialize the temporal positional embedding with a normal distribution. We masked the slots of not available frames, *i.e.* frames with indices that are either less than 0 or exceed the frame number, in transformer layers.

Slot Merging (ψ_{merge}): We use the implementation of Agglomerative Clustering in the `sklearn` library [6] with complete linkage. For each cluster, we compute the mean slot and the sum of attention matrices, *i.e.* matrix $\mathbf{A}$ in (1), for the associated slots. We determine the scale $\mathbf{S}_s$ and position $\mathbf{S}_p$ parameters for the merged attention values to calculate the relative grid $\mathbf{G}_{\text{rel}}$ of the new slots. Then, $\mathbf{G}_{\text{rel}}$ is projected onto D_{slot} using a linear layer h and added to the broadcasted slots before decoding.

Decoder Mapper (ψ_{mapper}): The mapper ψ_{mapper} consists of 5 linear layers with ReLU activations and a hidden size of 1024. The final layer maps the activations to the dimension of ViT-B tokens with an extra alpha value *i.e.* 768 + 1.

B.3 Training Details

In all our experiments, unless otherwise specified, we employ DINOv2 [5] with the ViT-B/14 architecture. We set the number of consecutive frame range n to 2 and drop half of the tokens before the slot attention step. We train our models on 2×V100 GPUs using the Adam [4] optimizer with a batch size of 48. We clip the gradient norms at 1 to stabilize the training. We match mask indices of consecutive frames by applying Hungarian Matching on slot similarity to provide temporal consistency. To prevent immature slots in slot merging, we apply merging with a probability that is logarithmically increasing through epochs.

MOVi-E: We train our model from scratch for a total of 60 epochs, which is equivalent to approximately 300K iterations. We use a maximum learning rate of 4×10^{-4} and an exponential decay schedule with linear warmup steps constituting 5% of the overall training period. The model is trained using 18 slots and the input frames are adjusted to a size of 336×336, leading to 576 feature tokens for each frame. The slot merge coefficient in ψ_{merge} is configured to 0.12.

YTVIS19: Similar to MOVi-E, we train the model from scratch for 180 epochs, corresponding to approximately 300K iterations with a peak learning rate of 4×10^{-4} and decay it with an exponential schedule. Linear warmup steps are introduced for 5% of the training timeline. The model training involves 8 slots, and the input frames are resized to dimensions of 336×504, resulting in 864 feature tokens for each frame. The slot merge coefficient in ψ_{merge} is set to be 0.12.

DAVIS17: Due to the small size of DAVIS17, we fine-tune the model pretrained on the YTVIS19 dataset explained above. We finetune on DAVIS17 for 300 epochs, corresponding to approximately 40K iterations with a reduced learning rate of 1×10^{-4}. We use the same learning rate scheduling strategy as in YTVIS19. During the fine-tuning process, we achieve the best result without slot merging, likely due to the fewer number of objects, typically one object at the center, on DAVIS17 compared to YTVIS19.

C Invariant Slot Attention

In this section, we provide the details of invariant slot attention (ISA), initially proposed by Biza et al. [1]. We use ISA in our Spatial Binding module $\psi_{\text{s-bind}}$ with shared initialization as explained in the main paper. Given the shared initialization $\mathcal{Z}_\tau = \left\{\left(\mathbf{z}^j, \mathbf{S}_s^j, \mathbf{S}_p^j, \mathbf{G}_{\text{abs},\tau}\right)\right\}_{j=1}^{K}$, our goal is to update the K slots: $\{\mathbf{z}^j\}_{j=1}^{K}$. For clarification, in the following, we focus on the computation of single-step slot attention for time step τ:

$$\mathbf{A}^j := \underset{K}{\text{softmax}}\left(\mathbf{M}^j\right) \in \mathbb{R}^{N'}, \qquad \mathbf{M}^j := \frac{1}{\sqrt{D_{\text{slot}}}} p\left(k\left(\mathbf{f}_\tau\right) + g\left(\mathbf{G}_{\text{rel},\tau}^j\right)\right) q\left(\mathbf{z}^j\right)^T \in \mathbb{R}^{N'} \tag{1}$$

where p, g, k, and q are linear projections while the relative grid of each slot is defined as:

$$\mathbf{G}_{\text{rel},\tau}^j := \frac{\mathbf{G}_{\text{abs},\tau} - \mathbf{S}_p^j}{\mathbf{S}_s^j} \in \mathbb{R}^{N' \times 2} \tag{2}$$

The slot attention matrix $\mathbf{A}$ from (1) is used to compute the scale $\mathbf{S}_s$ and the positions $\mathbf{S}_p$ of slots following Biza et al. [1]:

$$\mathbf{S}_s^j := \sqrt{\frac{\text{sum}\left(\mathbf{A} \odot \left(\mathbf{G}_{\text{abs},\tau} - \mathbf{S}_p^j\right)^2\right)}{\text{sum}\left(\mathbf{A}^j\right)}} \in \mathbb{R}^2, \qquad \mathbf{S}_p^j := \frac{\text{sum}\left(\mathbf{A}^j \odot \mathbf{G}_{\text{abs},\tau}\right)}{\text{sum}\left(\mathbf{A}^j\right)} \in \mathbb{R}^2 \tag{3}$$

After this step, following the original slot attention, input features are aggregated to slots using the weighted mean with another linear projection v:

$$\mathbf{U} := \mathbf{W}^T p\left(v\left(\mathbf{f}_\tau\right) + g\left(\mathbf{G}_{\text{rel},\tau}^j\right)\right) \in \mathbb{R}^{K \times D_{\text{slot}}}, \qquad \mathbf{W}^j := \frac{\mathbf{A}^j}{\text{sum}\left(\mathbf{A}^j\right)} \in \mathbb{R}^{N'} \tag{4}$$

Then, $\mathbf{U}$ from (4) is used to update slots $\{\mathbf{z}^j\}_{j=1}^K$ with GRU followed by an additional MLP as residual connection as shown in (5). This operation is repeated 3 times.

$$\mathbf{z} := \mathbf{z} + \text{MLP}\left(\left(\text{norm}(\mathbf{z})\right), \qquad \mathbf{z} := \text{GRU}\left(\mathbf{z}, \mathbf{U}\right) \in \mathbb{R}^{K \times D_{\text{slot}}} \tag{5}$$

D Additional Ablation Studies

In this section, we provide additional experiments to show the effect of resolution on the segmentation quality. We also report the results by varying the evaluation split on the YTVIS to confirm the generalization capability of our model.

Effect of Resolution: We conducted experiments to investigate the effect of the input frame resolution, *i.e*. the number of input tokens, in Fig. 8. Specifically, we experimented with resolutions of 168×252, 224×336, and our default resolution 336×506, corresponding to 216, 384, and 864 input feature tokens, respectively. These experiments show that the input resolution is crucial for segmentation performance.

Varying The Evaluation Split: As stated before, we cannot use the validation split of YTVIS19 due to manual annotations that are not publicly available. For evaluation, we only choose a subset of 300 videos. Here, we perform an experiment to examine the effect of varying the set of evaluation videos on performance.

We repeat the experiment in Table 3 by choosing mutually exclusive subsets of 300 videos, 3 times. We report mean and standard deviation ($\mu \pm \sigma$) of experiments for each model in Table 6. These results are coherent with the reported result on the fixed subset, showing that segmentation performance peaks when all of our components are combined, *i.e*. model E. Similarly, removing our components $\psi_{\text{t-bind}}$ (model D) and $\psi_{\text{s-bind}}$ (model C) one at a time leads to a performance drop, as shown in Table 3.Removing both (model B), results in the worst performance, even falling behind its counterpart without slot merging (model A). Overall, the results of varying the evaluation set agree with the reported performance on the selected subset in the main paper. This shows that the performance of our model generalizes over different evaluation subsets.

Table 6: **Varying evaluation splits.** The effect of changing evaluation sets for models in the component ablation.

Model	mIoU ↑	FG-ARI ↑
A	37.72 ± 0.82	29.06 ± 2.24
B	37.20 ± 1.53	30.15 ± 1.90
C	42.76 ± 2.09	30.03 ± 1.67
D	42.98 ± 2.01	29.69 ± 2.02
E	43.28 ± 2.57	31.33 ± 1.51

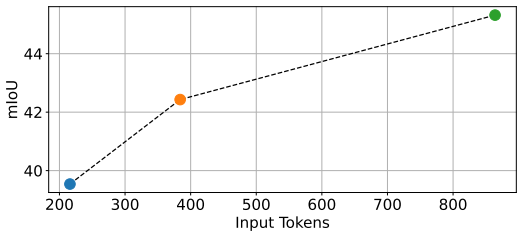


Figure 8: **Input resolution.** The effect of input resolution, *i.e*. the number of tokens, on performance (mIoU).

E Additional Visualizations

We provide additional qualitative results in Fig. 9. Our model can recognize not only the most salient object in the middle but also small, subtle objects in the background. Furthermore, it can handle a varying number of objects in the scene with slot merging.

Feature Extractor: In Fig. 10, we provide visualizations of different feature extractors, corresponding to the quantitative evaluation in Table 5, including DINOv2 [5], DINO [2], and Supervised. Self-supervised models performs better than the supervised one, also qualitatively. In particular, DINOv2 stands out for its exceptional capability to learn object representations across a diverse range of categories. It also effectively identifies and segments intricate details that are missed by other feature extractors such as tree branches on the left and the bag on the table on the right.

Components: In Fig. 11, we visually compare the segmentation results of the models corresponding to the component ablation study in the main paper (Table 3). First of all, model A, which corresponds

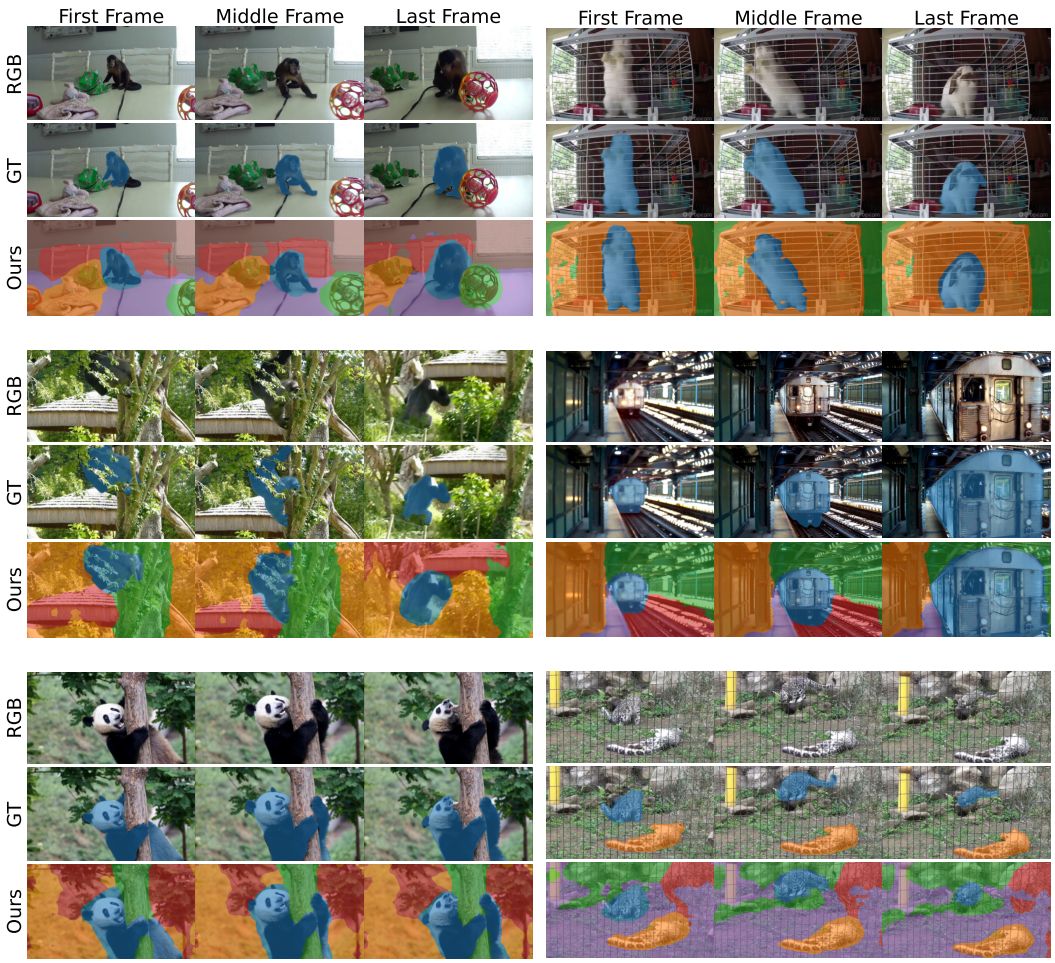


Figure 9: Qualitative results of multi-object video segmentation on YTVIS19

to the temporally consistent DINOSAUR [7], struggles to cluster the instances as a whole and also fails to track all objects, for instance, the human on the right, due to discrepancies in mask index across three frames. With the help of slot merging, model B effectively addresses the over-clustering issue, as observed in the mask of the calf on the left. Integrating our binding modules $\psi_{\text{t-bind}}$ and $\psi_{\text{s-bind}}$, resulting in model C and D, respectively, leads to a marked improvement in both segmentation and tracking quality. On the other hand, both models C and D have shortcomings in detecting the human in certain frames of the right video. Finally, combining them, our full model, *i.e.* model E, excels at segmenting and tracking not only labeled objects but also other objects, such as the car visible in the first frame of the second video.

Comparison: We provide a visual comparison between OCLR [9] and our model in Fig. 12 after matching the ground-truth and predictions. OCLR fails to detect static objects (top-left, middle-left, bottom-left) and deformable objects (middle-right) due to failures of optical flow in these cases. Moreover, it considers moving regions as different objects such as the water waves (top-right). On the other hand, SOLV can accurately detect all objects as a whole.

Failure Cases: Although our model can detect in-the-wild objects in different scales, segmentation boundaries are not perfectly aligned with the object due to patch-wise segmentation, as has been pointed out in the Discussion (Section 5). In addition to these observations, we identify three types of commonly occurring failure cases: (i) The over-clustering issue that remains unresolved in some cases even with slot merging. (ii) The tendency to cluster nearby instances of the same class into a single slot. (iii) Failure to detect small objects, particularly when they are situated near large objects. We provide visual examples of these cases in Fig. 13.

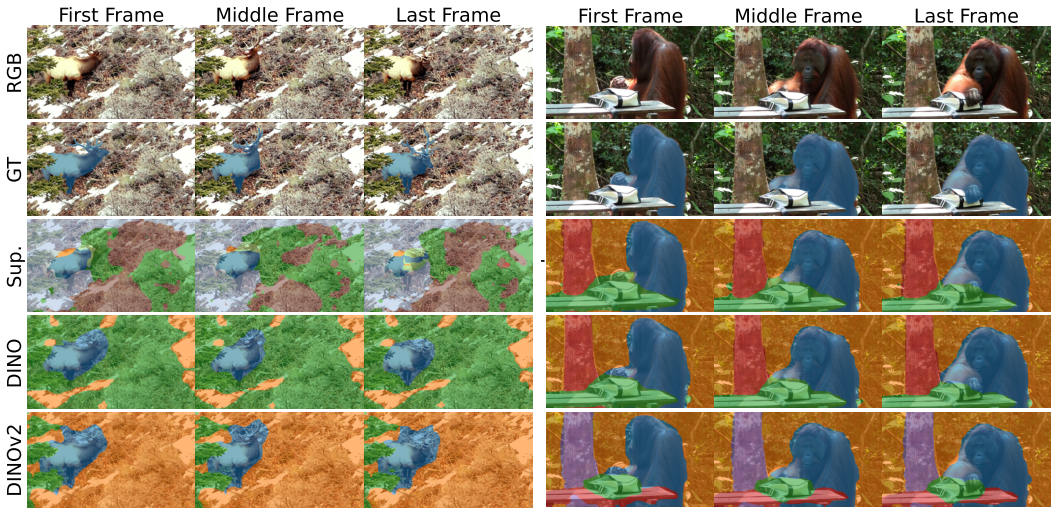


Figure 10: Qualitative comparison of different pretraining methods for visual encoder.

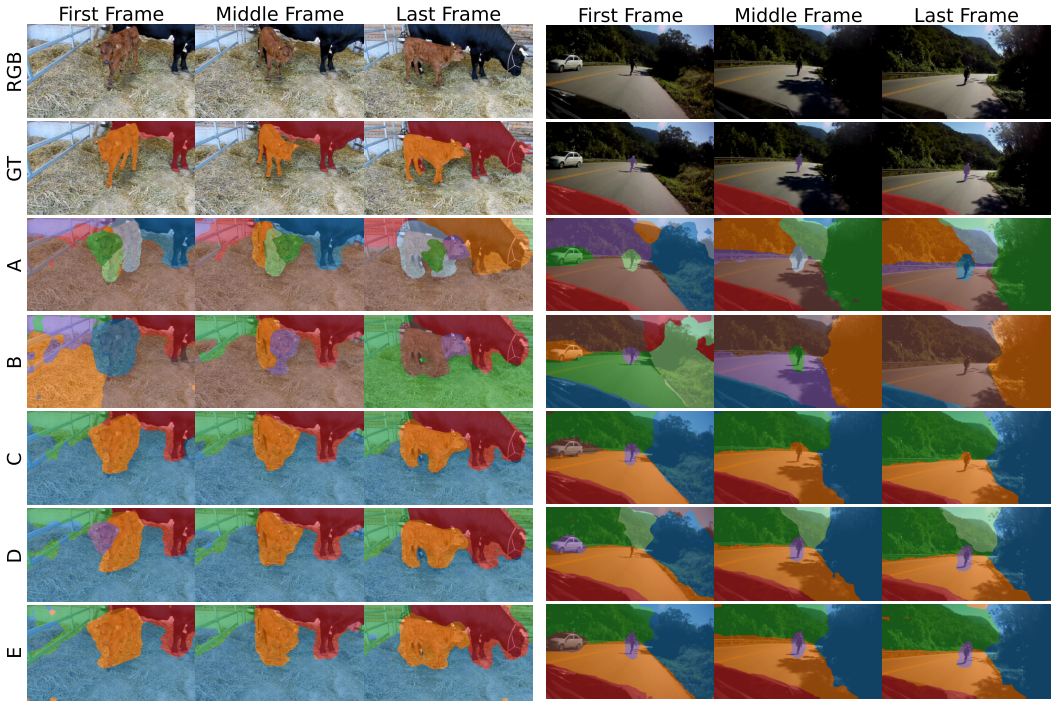


Figure 11: Qualitative results of different models in the component ablation study (Table 3).

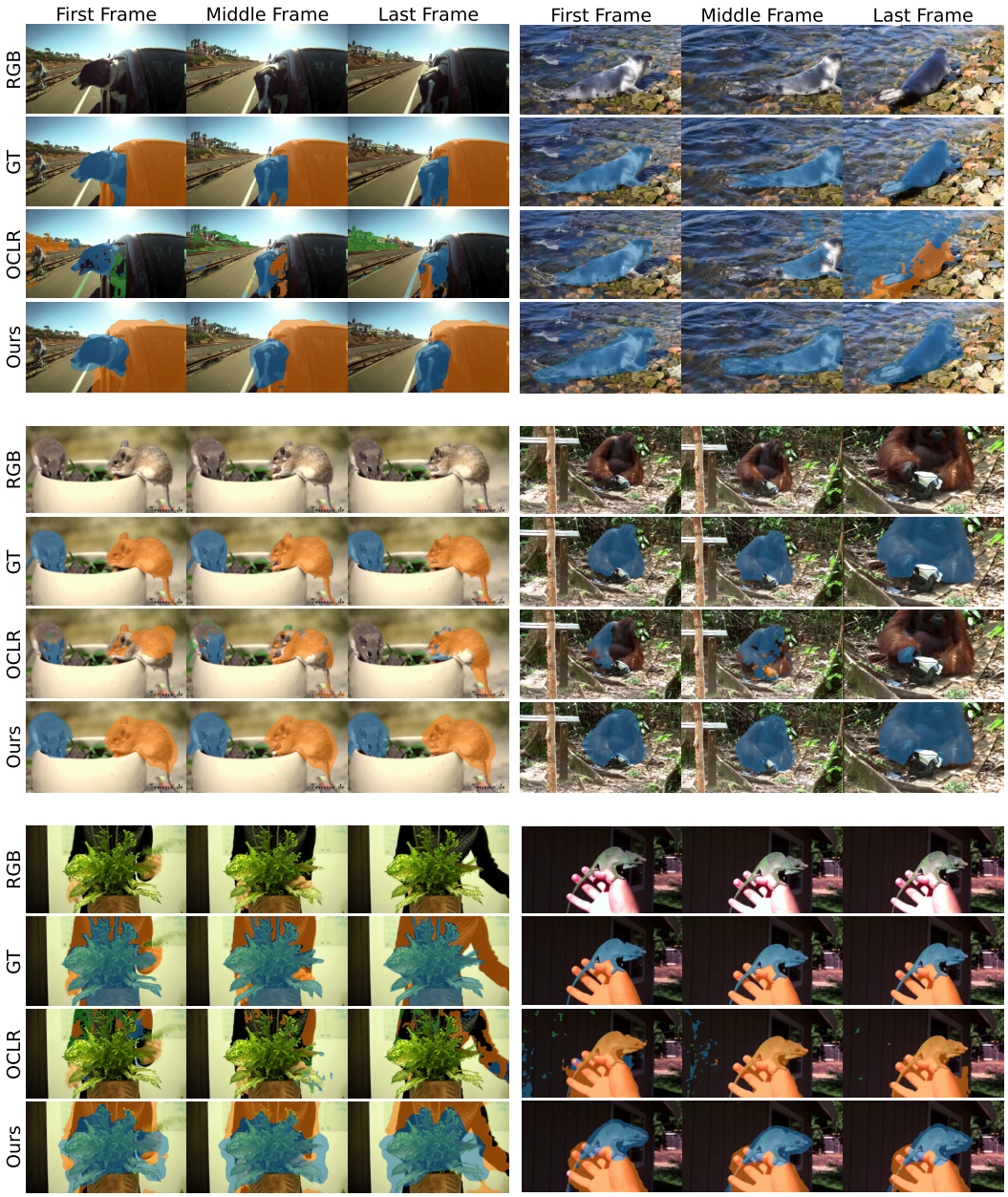
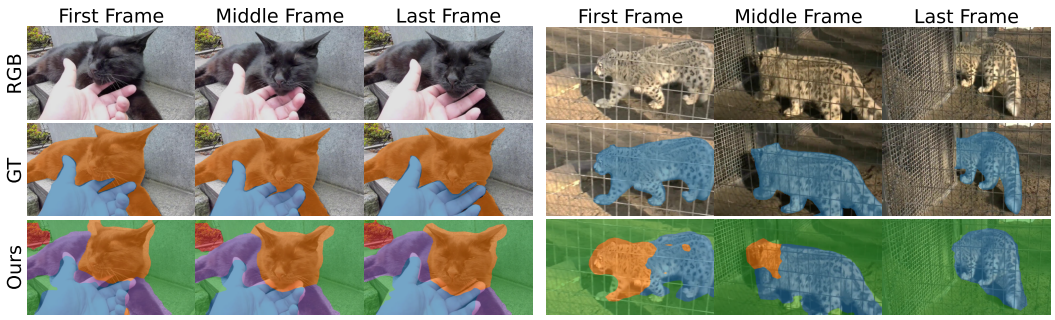
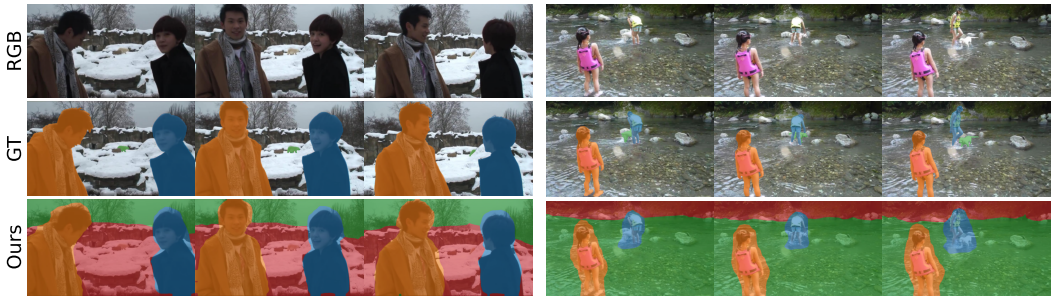


Figure 12: Qualitative results of multi-object video segmentation on YTVIS19 after Hungarian Matching is applied. Segmentation results of OCLR [9] are provided in third row.



(a) Failure cases due to over-clustering.

(b) Failure cases due to grouping nearby instances of the same class into one cluster.



(c) Failure cases due to missing relatively small objects.

Figure 13: Failure cases of our model with potential reasons, grouped into three.

References

[1] Ondrej Biza, Sjoerd van Steenkiste, Mehdi S. M. Sajjadi, Gamaleldin F. Elsayed, Aravindh Mahendran, and Thomas Kipf. Invariant slot attention: Object discovery with slot-centric reference frames. In *arXiv.org*, 2023. 2

[2] Mathilde Caron, Hugo Touvron, Ishan Misra, Hervé Jégou, Julien Mairal, Piotr Bojanowski, and Armand Joulin. Emerging properties in self-supervised vision transformers. In *Proc. of the IEEE International Conf. on Computer Vision (ICCV)*, 2021. 3

[3] Xavier Glorot and Yoshua Bengio. Understanding the difficulty of training deep feedforward neural networks. In *Journal of Machine Learning Research (JMLR)*, 2010. 1

[4] Diederick P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *Proc. of the International Conf. on Learning Representations (ICLR)*, 2015. 2

[5] Maxime Oquab, Timothée Darcet, Theo Moutakanni, Huy V. Vo, Marc Szafraniec, Vasil Khalidov, Pierre Fernandez, Daniel Haziza, Francisco Massa, Alaaeldin El-Nouby, Russell Howes, Po-Yao Huang, Hu Xu, Vasu Sharma, Shang-Wen Li, Wojciech Galuba, Mike Rabbat, Mido Assran, Nicolas Ballas, Gabriel Synnaeve, Ishan Misra, Herve Jegou, Julien Mairal, Patrick Labatut, Armand Joulin, and Piotr Bojanowski. Dinov2: Learning robust visual features without supervision. In *arXiv.org*, 2023. 1, 2, 3

[6] Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, Jake Vanderplas, Alexandre Passos, and David Cournapeau. Scikit-learn: Machine learning in Python. In *Journal of Machine Learning Research (JMLR)*, 2011. 1

[7] Maximilian Seitzer, Max Horn, Andrii Zadaianchuk, Dominik Zietlow, Tianjun Xiao, Carl-Johann Simon-Gabriel, Tong He, Zheng Zhang, Bernhard Schölkopf, Thomas Brox, et al. Bridging the gap to real-world object-centric learning. In *Proc. of the International Conf. on Learning Representations (ICLR)*, 2023. 4

[8] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2017. 1

[9] Junyu Xie, Weidi Xie, and Andrew Zisserman. Segmenting moving objects via an object-centric layered representation. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2022. 4, 6