# TarViS: A Unified Approach for Target-based Video Segmentation Supplementary Material

Ali Athar[1] Alexander Hermans[1] Jonathon Luiten[1,2] Deva Ramanan[2] Bastian Leibe[1]

[1]RWTH Aachen University, Germany [2]Carnegie Mellon University, USA

{athar,hermans,luiten,leibe}@vision.rwth-aachen.de deva@cs.cmu.edu

## 1. Extended VOS/PET Ablations

Extended ablation results are given in Table 1 and discussed below. For these experiments we use a shorter/lighter training schedule compared to the results presented in the main text: the network is pre-trained on augmented image sequences generated from the COCO dataset for 360k iterations on 8 GPUs, followed by fine-tuning on actual video data from the DAVIS [13] and BURST [2] datasets.

Table 1. Extended ablation results for VOS and PET tasks on DAVIS [13] and BURST [2] benchmarks, respectively.

| Setting | VOS ($\mathcal{J}\&\mathcal{F}$) | PET ($\text{HOTA}_{\text{all}}$) |
|---|---|---|
| Without $Q_{\text{bg}}$ | 78.0 | 25.9 |
| $\lvert Q_{\text{obj}}\rvert = q_0 = 1$ | 80.6 | 28.2 |
| Final | 81.5 | 29.2 |

**Background Queries (row 1).** We stated in the main text that we model the non-object pixels in the input video using background queries for the VOS and PET task. We ablate this design decision by training TarViS without this sort of background modeling, *i.e.* for both VOS and PET tasks, the input set of queries contains only the object queries $Q_{\text{obj}}$. This reduces the $\mathcal{J}\&\mathcal{F}$ score for VOS from 81.5 to 78.0, and the $\text{HOTA}_{\text{all}}$ score for PET from 29.2 to 25.9. Thus, we conclude that background modeling has a noticeable, positive impact on prediction quality.

**Number of Object Queries (row 2).** We mentioned in the main text that we modify the approach adopted by HODOR [1] for VOS by using multiple ($q_0$) object queries to represent a single target object. We ablate this by training our model using $q_0 = 1$ (in the final setting we use $q_0 = 4$). We see that this causes the performance on DAVIS to reduce from 81.5 to 80.6, and that on BURST from 29.2 to 28.2. Note that $q_0 = 1$ for PET even for the final setting, but because PET inference over lengthy videos involves VOS-style mask-guidance, the choice of $q_0$ for VOS affects performance for PET as well.

Table 2. Extended results for PET on the BURST [2] validation and test sets. ('H' denotes 'HOTA' [10]).

| Method | BURST (val) | | | BURST (test) | | |
|---|---|---|---|---|---|---|
| | $\text{H}_{\text{all}}$ | $\text{H}_{\text{com}}$ | $\text{H}_{\text{unc}}$ | $\text{H}_{\text{all}}$ | $\text{H}_{\text{com}}$ | $\text{H}_{\text{unc}}$ |
| Box Tracker [6] | 12.7 | 31.7 | 7.9 | 10.1 | 24.4 | 7.3 |
| STCN+M2F [3, 4] | 24.4 | 44.0 | 19.5 | 24.9 | 39.5 | 22.0 |
| **TarViS (R-50)** | 30.9 | 43.2 | 27.8 | 32.1 | 41.5 | 30.2 |
| **TarViS (Swin-T)** | 36.0 | 47.7 | 33.0 | 36.4 | 45.0 | 34.7 |
| **TarViS (Swin-L)** | **37.5** | **51.7** | **34.0** | **36.1** | **47.1** | **33.8** |

## 2. Detailed BURST Metrics

Due to space constraints, we only presented the final $\text{HOTA}_{\text{all}}$ score for the BURST benchmark in the main paper. Table 2 gives a more detailed breakdown for those results.

## 3. Implementation Details

Several details related to the training and inference setup which were omitted from the main paper are given below.

**Hardware Setup and Training Schedule.** We train our models on 32 Nvidia A100 GPUs with a batch size of 32 with clips of 3 frames. The pretraining takes 2-3 days depending on the backbone whereas finetuning takes 10-16 hours. An AdamW optimizer is used with a learning rate of $10^{-4}$ at the start, followed by two step decays with a factor of 0.1 each.

**Inference.** Inference is performed on a single RTX 3090 and runs at 6-10 fps using a Swin-T backbone. The variation mainly arises because different datasets have different image resolutions. For most datasets, we use clips containing 12 frames with a 6 frame overlap between successive clips.

**Loss Supervision.** Table 3 shows the type of loss function applied for mask regression for different tasks. Generally, the supervision signal is a combination of DICE and

Table 3. Loss functions used for mask prediction for different targets. BCE: Binary cross-entropy, MCE: Multi-class cross-entropy, DICE: soft IoU loss

| Target Type | Task | Loss |
|---|---|---|
| Instance<br>Semantic Class | VIS | DICE + BCE<br>MCE |
| Instance<br>Semantic Class | VPS | DICE + BCE<br>MCE |
| Object | VOS/PET | DICE + BCE |

cross-entropy losses. For instances/objects we apply per-pixel binary cross-entropy whereas for semantic segmentation (where multiple classes compete for each pixel), we apply a multi-class cross-entropy loss. We apply a sparse loss similar to Cheng *et al*. [3], *i.e*., the loss is not applied to every pixel in the mask, but rather only to a subset of pixels which contain a certain fraction of hard negatives and other randomly sampled points. This type of supervision strategy was first proposed by Kirillov *et al*. [8].

**Pretraining.** We pretrain on synthetic video samples generated by applying random, on-the-fly augmentations from the following image-level datasets: COCO [9], ADE20k [17], Mapillary [12], Cityscapes [5]. Since these datasets provide panoptic annotations, we can train the data samples as any of the four target tasks (VPS, VIS, VOS, PET) *e.g*. to train for VOS/PET, we assume that the first-frame mask/point is available for a random subset of ground-truth objects and ignore the class labels. The task weights for pretraining are given in Table 4.

Table 4. Task weights during pretraining stage.

| Task | VPS | VIS | VOS | PET |
|---|---|---|---|---|
| Weight | 0.3 | 0.3 | 0.28 | 0.12 |

**Video Finetuning.** The finetuning is done on actual video datasets for all four tasks. The sampling weights for each dataset/task are given in Table 5. Note that data samples from DAVIS [13] and BURST [2] can be trained for both VOS and PET.

Table 5. Dataset weightage during video finetuning.

| Dataset | Task | Weight |
|---|---|---|
| KITTI-STEP [15] | VPS | 0.075 |
| CityscapesVPS [7] | VPS | 0.075 |
| VIPSeg [11] | VPS | 0.15 |
| YouTube-VIS [16] | VIS | 0.225 |
| OVIS [14] | VIS | 0.225 |
| DAVIS [13] | VOS/PET | 0.05 |
| BURST [2] | VOS/PET | 0.2 |

**Point Exemplar-guided Tracking Inference.** As mentioned in Sec. 3 of the main text, the PET task is tackled using the same workflow as for VOS *i.e*. the target objects are encoded as object queries using the Object Encoder. An additional detail about inference on arbitrary length video sequences which is not mentioned in the main text is as follows: the point $\rightarrow$ object query regression is only used for the first clip in which the object appears. For subsequent clips, we have access to the dense mask predictions for that object from our model. Hence, for subsequent clips, we regress the object query from the previous mask predictions (as we do for VOS).

## 4. Query Visualization

To gain some insight into the feature representation learned by TarViS for different targets, we provide visualizations of the target queries for various tasks and input video clips in Fig. 1,2,3. The setup is as follows: for each video clip, we run inference twice: (1) as VIS where the targets are all instances belonging to the 40 object classes from YouTube-VIS [16], and (2) as VOS by providing the first-frame mask for the objects. We deliberately used videos where the set of set of ground-truth objects would be the same for both tasks. The plot on the right visualizes the union of the target query set for both runs by projecting them from 256 dimensions down to 2 using PCA. The image tile on the left shows our model's predicted masks for the target objects (the prediction quality for these video is very good for both VIS and VOS, so we choose one set of results arbitrarily).

For ease of understanding, we use fixed colors for semantic and background queries (as indicated in the plot legend). For the object queries (VOS) and instance queries (VIS), the color of the query point is consistent with the color of the object mask in the image tile. Note that for VOS we used $q_o = 4$ object queries per target, hence there are 4 hollow diamond shaped points per object.

We stress that not all aspects of these plots are intuitively explainable. The main limitation here is the harsh dimensionality reduction from 256 dimensions to 2. Some speculative intuition based on the plots is as follows:

- The internal representation for a given object is generally consistent across tasks. As an example, consider the horse and person targets in Fig. 1: we note that the green query points (person) are close to each other for both VIS and VOS. Likewise the blue query points (horse) follow the same behavior.

- The network devotes a large portion of the feature space for instances/objects, and relatively less for the various semantic classes. As seen in all three plots, the semantic queries are tightly clustered together,

whereas the instance/object queries are spread out over a larger span of the feature space.

**Iterative Evolution of Feature Representation.** Fig. 4 shows a side-by-side visualization of how the query feature representation evolves inside the transformer decoder as it iteratively refined the queries using multiple attention layers. The plot on the left shows the queries at the 'zeroth' layer (*i.e.* prior to any interaction with the video features), and the plot on the right shows the final output queries from the last layer (these are identical to the plot in Fig. 1 except for the axes range). We note that the distance between the queries for the two objects increases after refinement, and that the semantic queries are also slightly more spaced out after refinement.

## 5. Qualitative Results

The following figures show qualitative results for the different tasks. Additional results are also present in the video attached to this supplementary archive. VIS on YouTube-VIS (Fig. 5,6,7) and OVIS (Fig. 8,9,10), VPS on KITTI-STEP (Fig. 11,12,13), VOS on DAVIS (Fig. 14,15,16), and PET on BURST (Fig. 17,18,19). One can see that TarViS is able to segment a broad range of objects depending on the target queries and overall is good at assigning consistent IDs. Fig. 20 shows an example of a failure case with several ID switches. Given that we run inference on short overlapping clips, once an ID switch has been made, we cannot recover the original ID. In the example, it seems that TarViS is not able to track the elephant while they are turning around, even though before and after the turn they are assigned consistent IDs. Given that we also train on similar short clips, it is not surprising that TarViS struggles here and we could potentially improve this by looking into other training schemes that span longer clips.

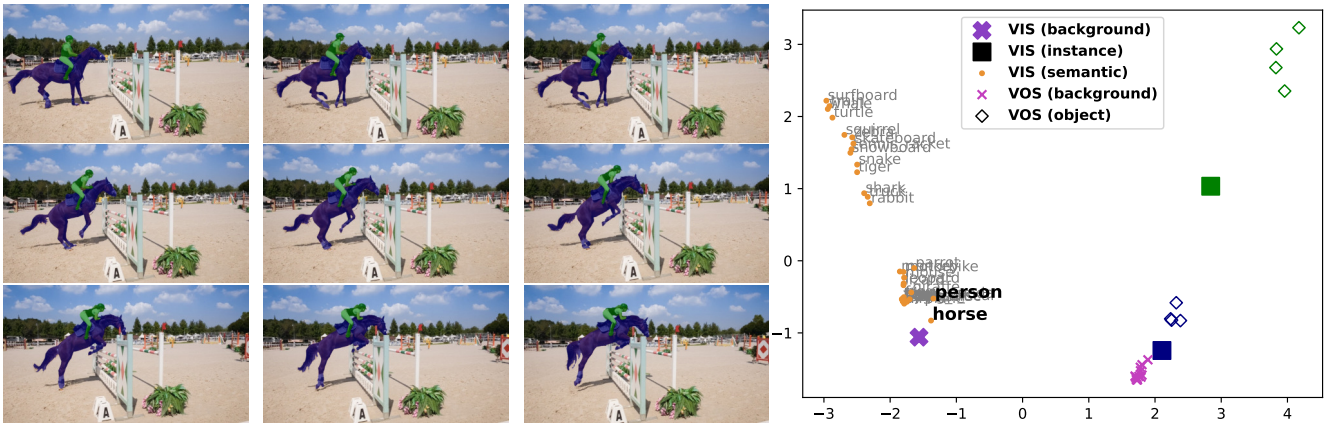



Figure 1. Target query visualization for the ‘horsejump-high’ sequence in DAVIS.

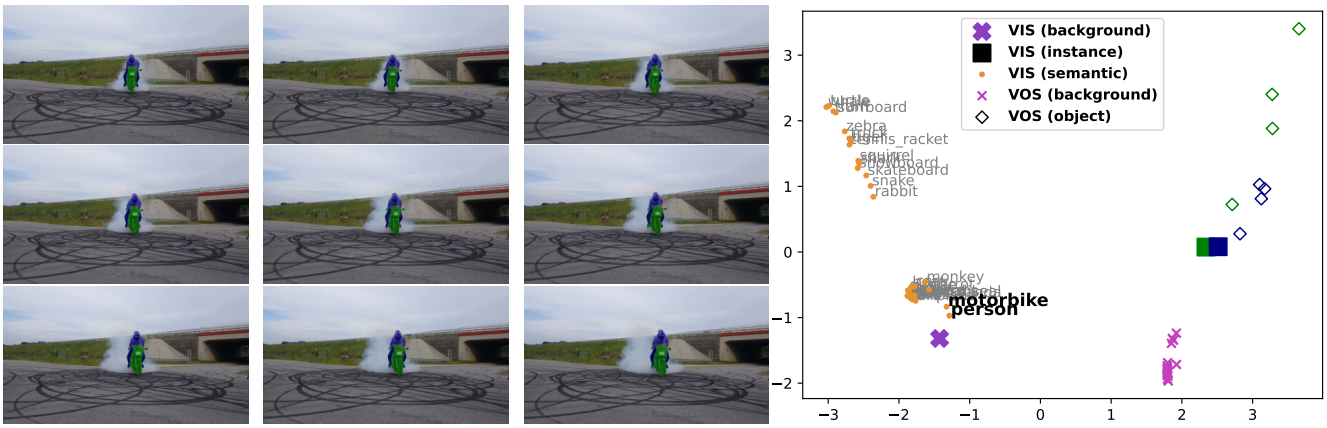



Figure 2. Target query visualization for the ‘mbike-trick’ sequence in DAVIS.

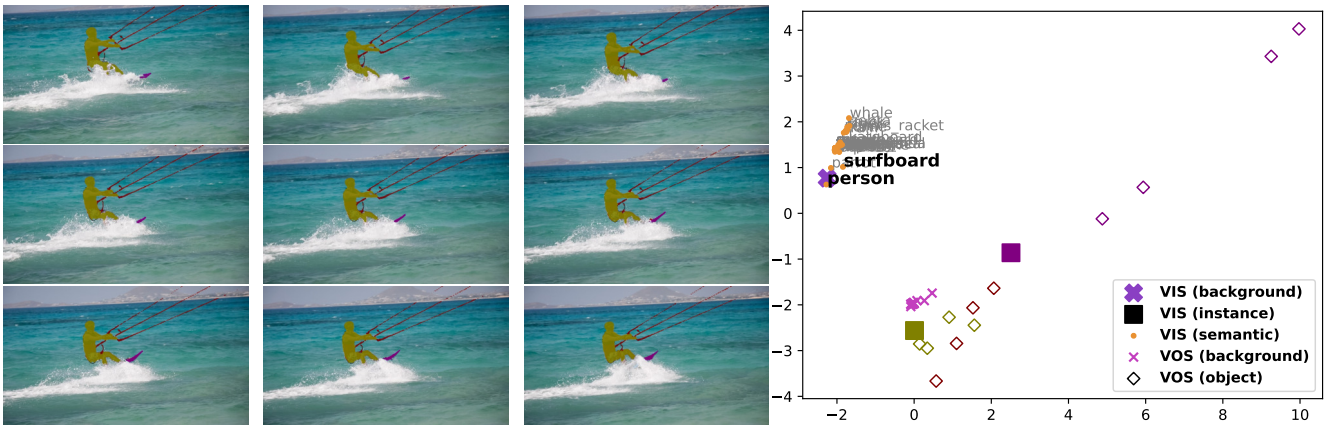
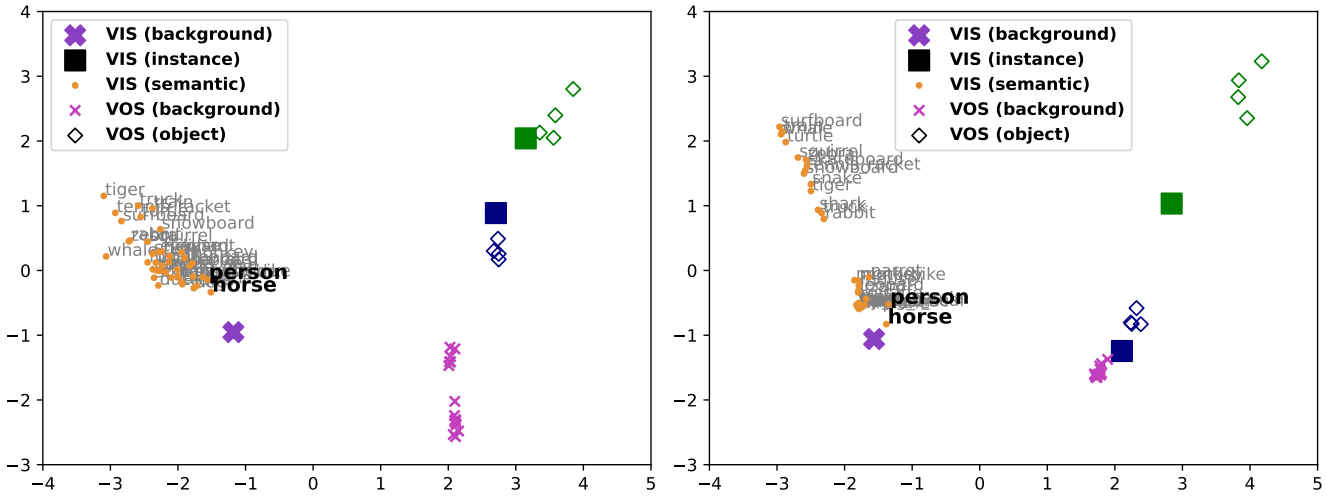



Figure 3. Target query visualization for the ‘kitesurf’ sequence in DAVIS.




(a) First layer queries.

(b) Last layer queries.

Figure 4. Evolution of the different queries from the first layer to the last layer of the transformer decoder. Queries correspond to the 'horsejump-high' video from DAVIS as shown in Figure 1

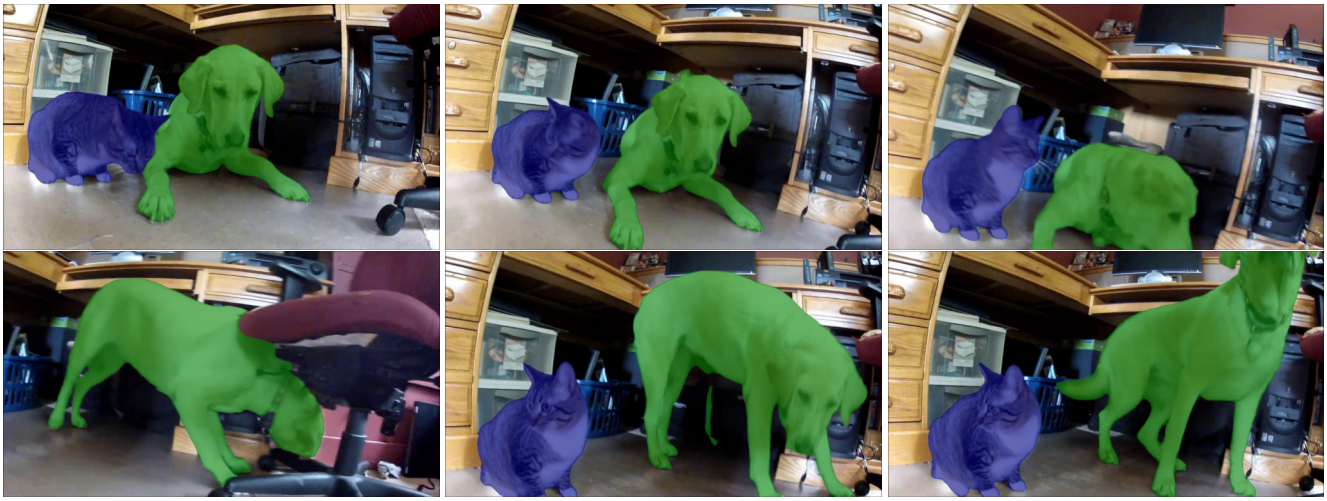


Figure 5. VIS on a YTVIS sequence showing a cat and a dog.

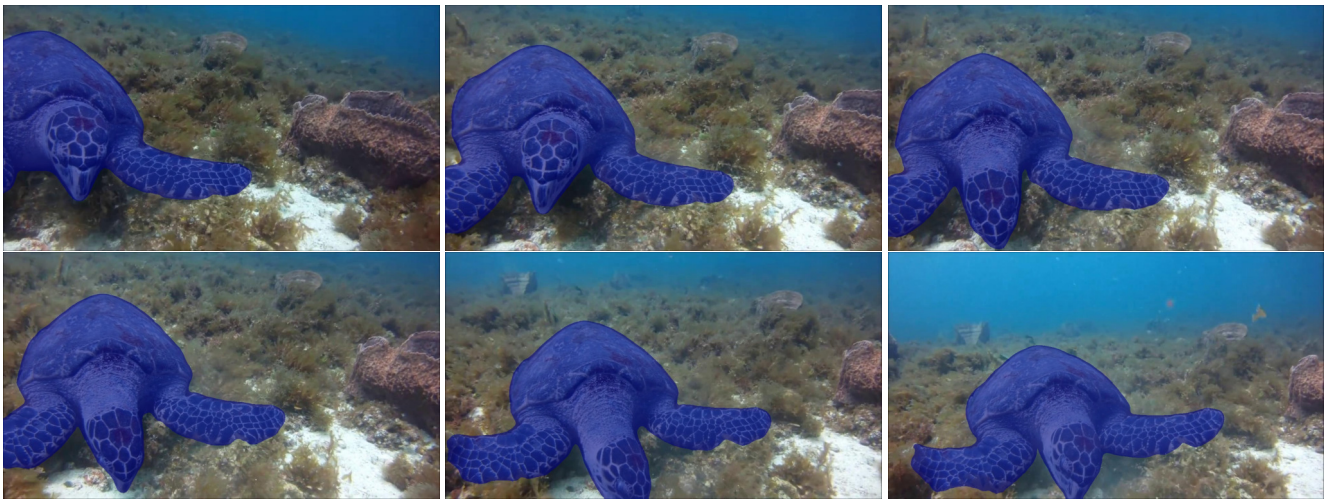


Figure 6. VIS on a YTVIS sequence showing a turtle.

Figure 7. VIS on a YTVIS sequence showing a man and a lizard.

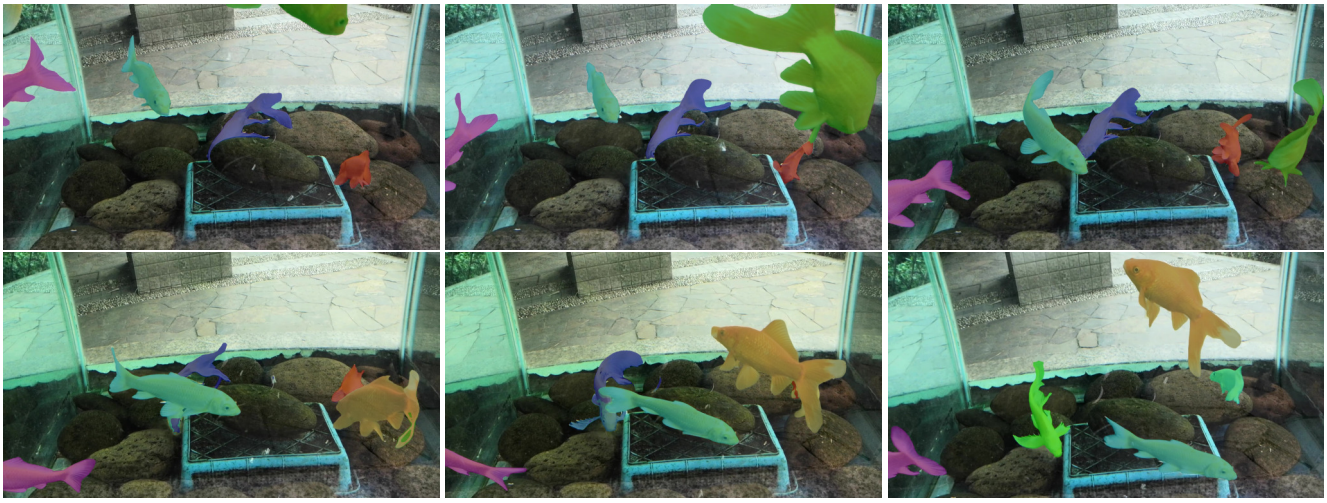


Figure 8. VIS on an OVIS sequence showing an aquarium with fish.

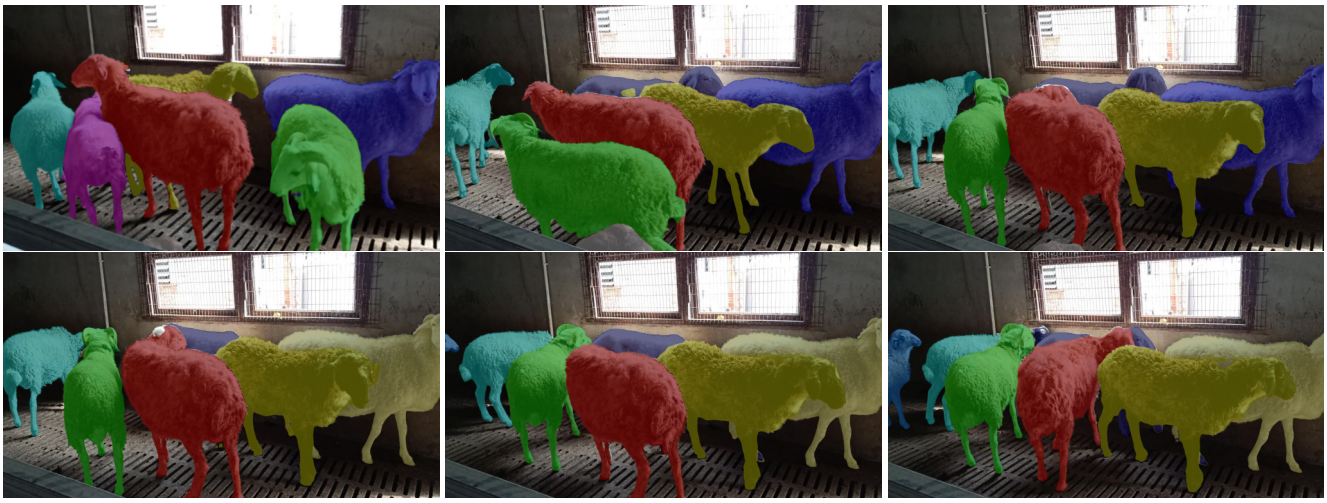


Figure 9. VIS on an OVIS sequence showing several sheep.

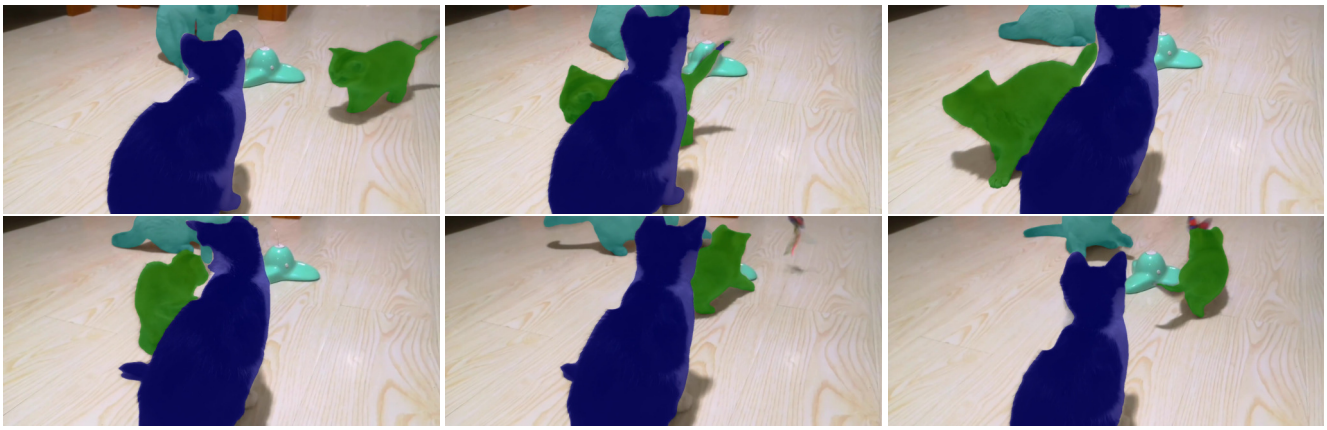


Figure 10. VIS on an OVIS sequence showing three cats.

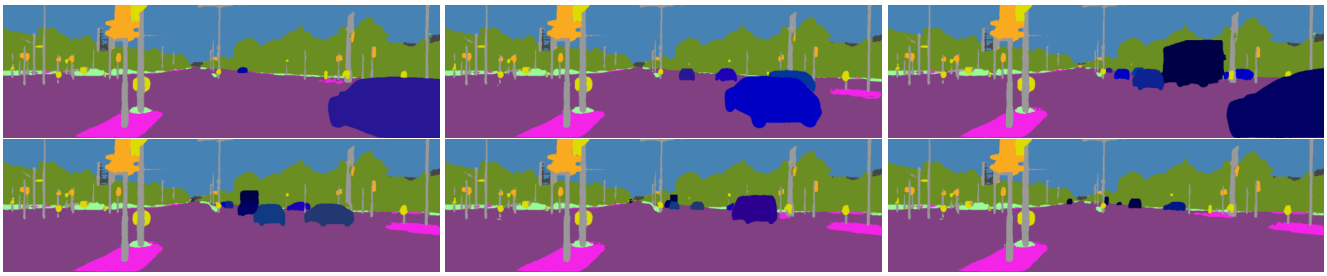


Figure 11. VPS on a KITTI STEP sequence showing a busy intersection.

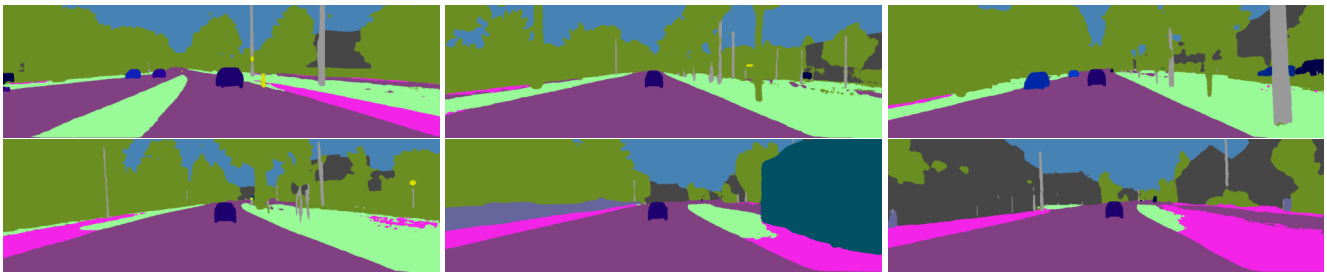


Figure 12. VPS on a KITTI STEP sequence showing how a car is followed for a while.

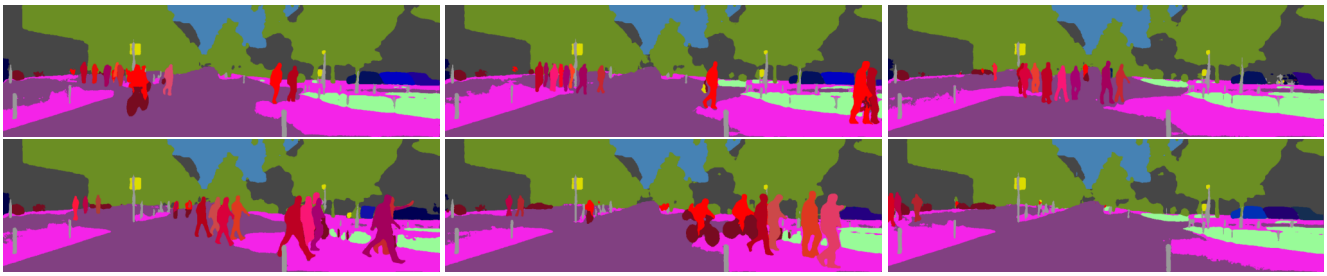


Figure 13. VPS on a KITTI STEP sequence showing a busy pedestrian crossing.

Figure 14. VOS on a DAVIS sequence of a dancer.

Figure 15. VOS on a DAVIS sequence showing several goldfish.

Figure 16. VOS on DAVIS sequence an action movie scene.

Figure 17. PET on a BURST sequence showing three men and a gun.

Figure 18. PET on a BURST sequence showing two bears fighting, note there is no ID switch.


Figure 19. PET on a BURST sequence showing several cars on a street.

Figure 20. VIS on an OVIS sequence of several elephants and their trainers. This sequence shows that TarVis sometimes has issues with ID switches, especially when the appearance of objects changes, e.g. here the elephants are not tracked consistently while turning around..

## References


[1] Ali Athar, Jonathon Luiten, Alexander Hermans, Deva Ramanan, and Bastian Leibe. Hodor: High-level object descriptors for object re-segmentation in video learned from static images. In *CVPR*, 2022. 1

[2] 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. 1, 2

[3] Bowen Cheng, Anwesa Choudhuri, Ishan Misra, Alexander Kirillov, Rohit Girdhar, and Alexander G Schwing. Mask2former for video instance segmentation. In *CVPR*, 2022. 1, 2

[4] 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

[5] Marius Cordts, Mohamed Omran, Sebastian Ramos, Timo Rehfeld, Markus Enzweiler, Rodrigo Benenson, Uwe Franke, Stefan Roth, and Bernt Schiele. The cityscapes dataset for semantic urban scene understanding. In *CVPR*, 2016. 2

[6] Arne Hoffhues Jonathon Luiten. Trackeval. https://github.com/JonathonLuiten/TrackEval, 2020. 1

[7] Dahun Kim, Sanghyun Woo, Joon-Young Lee, and In So Kweon. Video panoptic segmentation. In *CVPR*, 2020. 2

[8] Alexander Kirillov, Yuxin Wu, Kaiming He, and Ross Girshick. Pointrend: Image segmentation as rendering. In *CVPR*, 2020. 2

[9] 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. 2

[10] Jonathon Luiten, Aljosa Osep, Patrick Dendorfer, Philip Torr, Andreas Geiger, Laura Leal-Taixé, and Bastian Leibe. Hota: A higher order metric for evaluating multi-object tracking. *IJCV*, 2020. 1

[11] 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. 2

[12] Gerhard Neuhold, Tobias Ollmann, Samuel Rota Bulo, and Peter Kontschieder. The mapillary vistas dataset for semantic understanding of street scenes. In *ICCV*, 2017. 2

[13] Jordi Pont-Tuset, Federico Perazzi, Sergi Caelles, Pablo Arbeláez, Alex Sorkine-Hornung, and Luc Van Gool. The 2017 davis challenge on video object segmentation. In *Arxiv*, 2017. 1, 2

[14] 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: A benchmark. In *IJCV*, 2022. 2

[15] Mark Weber, Jun Xie, Maxwell Collins, Yukun Zhu, Paul Voigtlaender, Hartwig Adam, Bradley Green, Andreas Geiger, Bastian Leibe, Daniel Cremers, Aljoša Ošep, Laura Leal-Taixé, and Liang-Chieh Chen. STEP: Segmenting and tracking every pixel. In *NeurIPS*, 2021. 2

[16] Linjie Yang, Yuchen Fan, and Ning Xu. Video instance segmentation. In *ICCV*, 2019. 2

[17] Bolei Zhou, Hang Zhao, Xavier Puig, Sanja Fidler, Adela Barriuso, and Antonio Torralba. Scene parsing through ade20k dataset. In *CVPR*, 2017. 2