Video Mask Transfiner for High-Quality Video Instance Segmentation (Supplemental material)

Lei Ke[1,2], Henghui Ding[1], Martin Danelljan[1], Yu-Wing Tai[3], Chi-Keung Tang[2], and Fisher Yu[1]

[1] Computer Vision Lab, ETH Zürich
[2] The Hong Kong University of Science and Technology
[3] Kuaishou Technology
http://vis.xyz/pub/vmt

We first provide more details on the HQ-YTVIS dataset creation and annotation process for the train, val and test splits in Section 1. Then, we conduct more experimental analysis on our VMT and the iterative training paradigm in Section 2. We further present more implementation and training/inference details of Video Mask Transfiner (VMT) in Section 3. Finally, we show extensive video results comparisons in our project page, which includes: 1) Annotation quality comparisons between YTVIS [9] and HQ-YTVIS; 2) Comparisons between VMT and existing state-of-the-art methods on four video instance segmentation benchmarks; 3) The results of the same methods trained on YTVIS vs. HQ-YTVIS; 4) Ablation results of the quadtree sequence grouping (QSG); 5) The failure cases of our VMT.

1 HQ-YTVIS Dataset Creation

To construct the HQ-YTVIS, we randomly re-split the original YTVIS training set (2238 videos) with coarse mask boundary annotations into train (1678 videos, 75%), val (280 videos, 12.5%) and test (280 videos, 12.5%) subsets following the splitting ratios in YTVIS. We observe that the proportions of 40 categories are nearly the same on the three splits and similar to original YTVIS training set. We keep at least 3 videos per category in the created validation and test splits, and fix this split for training, validation and test on HQ-YTVIS.

The details of automatically refining *train* annotations of HQ-YTVIS through iterative training VMT are described in Section 3.2 of the paper. We here provide more details on manually correcting the smaller validation and test sets of HQ-YTVIS using LabelMe [5] in Figure 1. The human annotators correct the YTVIS mask boundary errors by carefully relabeling it with dense polygon points along object boundary, which has over 130 points per object in average.

Also, we show more visual cases when degrading the masks of OVIS dataset (Figure 2), which is to equip VMT with initial boundary correction ability as discussed in Section 3.2 (paper), we simulate various shapes of inaccurate segmentation masks by by perturbing the ground truth of OVIS datasets. The GT

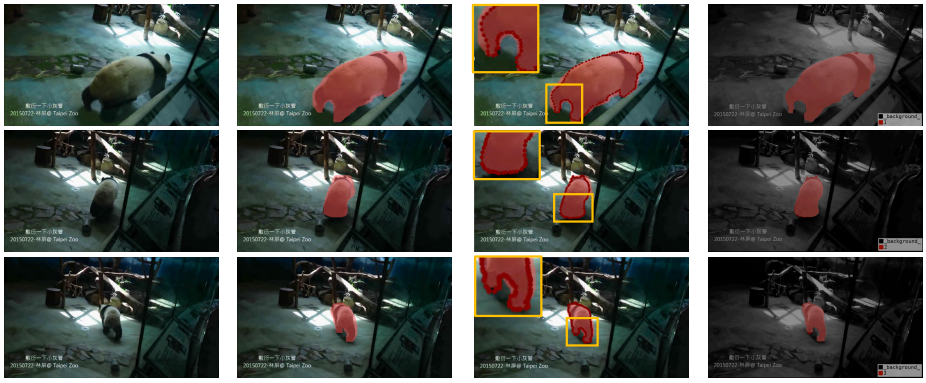

Fig. 1. Manual re-labeling of the coarse masks annotations of YTVIS to create the validation and test sets of HQ-YTVIS. Objects are relabeled with dense polygon points (over 130 per object in average) along the boundary to ensure high mask quality.

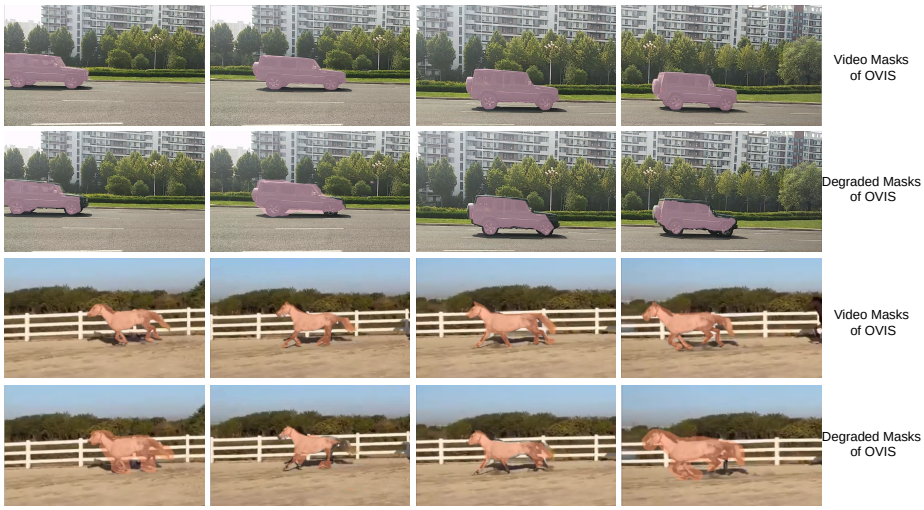

Fig. 2. Mask degradation on the better annotated OVIS dataset to simulate the coarse mask errors. The GT masks and perturbed masks of OVIS serves as training data for VMT to obtain initial boundary correction ability of video objects at the 1st iteration.

contour is first sub-sampled and then perturbed with random dilations and erosions. We also show more video cases comparisons on annotation quality between YTVIS and HQ-YTVIS in the attached video file.

2 More Experimental Analysis

Tube-Boundary AP on OVIS We report the tube mask AP evaluated by the online server of OVIS [4] in Table 8 of the paper. To further compare the

Table 1. Tube-Boundary AP^B comparisons on our created OVIS validation split.

Method	Backbone	AP^B	AP^B_{50}	AP^B_{75}	AR^B_1	AR^B_{10}
CMTrack RCNN [4]	R50	8.8	21.0	6.8	7.1	10.9
SeqFormer [7]	R50	9.1	21.2	6.9	7.0	11.7
VMT (Ours)	R50	**11.4**	**22.8**	**8.4**	**7.6**	**13.5**
CMTrack RCNN [4]	R101	9.8	21.2	8.0	7.6	11.5
SeqFormer [7]	R101	10.3	21.6	7.8	7.9	12.3
VMT (Ours)	R101	**12.9**	**22.9**	**9.7**	**8.3**	**14.6**

Table 2. Effect of iterative training using VMT on YTVIS. Model after each training iteration is evaluated on manually annotated HQ-YTVIS *val* and our created OVIS *val* splits using **GT** object classes, identities and corresponding coarse masks input. IoU^B denotes frame-wise boundary IoU, while IoU^M denotes frame-wise mask IoU.

Iterations	HQ-YTVIS						OVIS					
	AP^B	AP^B_{50}	IoU^B	AP^M	AP^M_{50}	IoU^M	AP^B	AP^B_{50}	IoU^B	AP^M	AP^M_{50}	IoU^M
1	60.3	80.1	63.1	83.2	93.2	87.3	65.1	78.2	65.2	82.9	91.9	86.5
2	75.5	91.5	78.5	92.5	97.9	94.1	79.6	93.0	80.1	92.3	96.3	93.6
3	**87.1**	94.9	87.8	**95.6**	97.4	98.6	**89.2**	**95.7**	**88.7**	96.6	97.9	99.2
4	86.9	**95.1**	**88.4**	95.5	**98.2**	**98.9**	88.8	95.6	88.1	**96.8**	**98.7**	**99.5**

Table 3. Effect of iterative training using SeqFormer [7] by replacing our VMT in Table 2 to self-correct YTVIS. The improvement scales brought by the alternative method SeqFormer are much smaller.

Iterations	HQ-YTVIS						OVIS					
	AP^B	AP^B_{50}	IoU^B	AP^M	AP^M_{50}	IoU^M	AP^B	AP^B_{50}	IoU^B	AP^M	AP^M_{50}	IoU^M
1	58.2	76.1	59.3	81.4	83.1	79.5	61.4	73.3	60.6	80.9	85.6	82.2
2	58.9	77.2	61.7	81.8	90.2	82.7	61.9	74.0	61.5	81.6	86.0	84.0
3	**59.7**	**78.1**	63.5	82.6	90.8	82.4	**62.9**	**75.1**	**62.8**	82.6	86.9	**84.8**
4	59.6	77.9	**63.8**	**82.8**	**91.2**	**82.9**	62.6	74.9	62.5	**82.8**	**87.1**	84.8

predicted masks boundary quality of video tracklets, we create our own *train* (480 videos), *val* (127 videos) splits from the OVIS train annotations by randomly splitting because the annotations for OVIS val set is not available. Then, we retrain CMaskTrack R-CNN [4], SeqFormer [7] and our VMT on the new *train* split from scratch, and evaluate them on our *val* split. The results of Tube-Boundary AP^B in Table 1 show that VMT achieves consistent improvements on the new validation split with the performance gain about 2.5 AP^B.

Effect of Iterative Training We further analyze the effect of iterative training. Different from the evaluation in Table 5 of the paper, we adopt the ground truth (GT) object identities and class labels, and use coarse object masks as the initial correction input. This is to directly evaluate of the refinement mask quality on *val* splits by using the same way of correcting the training labels of YTVIS. Thus, the reported AP here are solely dependent on the refined object masks quality. In Table 2, we select the trained VMT models after different iterations, and evaluate their mask quality on HQ-YTVIS and OVIS respectively. VMT achieves consistent and large mask quality improvements after three training iterations on both two benchmarks. We further check the small amount of failure cases with mask IoU lower than 50 after the 4th iteration, and find that most of them

Table 4. Ablation on instance guidance layer (IGL). IGL-1/2/3: IGL guidance after first/second/third layer of NAL.

IGL-1	IGL-2	IGL-3	AP^B	AP^B_{50}	AR^B_1	AP^M	AP^M_{50}	AR^M_1
			31.0	64.8	29.2	48.3	70.3	43.5
✓			31.8	65.9	30.5	48.8	71.1	44.9
	✓		32.1	66.3	31.0	49.1	71.5	45.3
✓	✓		32.6	66.4	31.5	49.7	71.8	45.6
✓	✓	✓	**33.7**	**67.2**	**31.8**	**50.5**	**72.4**	**46.2**

Table 5. Ablation on the RGB feature point encoding, and instance-specific dynamic pixel decoder on HQ-YTVIS *val* set.

RGB Encoding	Dynamic MLP	AP^B	AP^B_{50}	AR^B_1	AP^M	AP^M_{50}	AR^M_1
		31.2	65.1	30.2	49.3	71.6	45.1
✓		32.0	65.8	30.6	49.6	71.9	45.4
✓	✓	**33.7**	**67.2**	**31.8**	**50.5**	**72.4**	**46.2**

take the coarse masks input with gross annotation errors, such as missing large parts of the object. For evaluation on HQ-YTVIS, we use the coarse object masks provided by YTVIS as the correction target. For evaluation on OVIS [4], we take the coarse masks produced by mask degradation (Figure 2).

Iterative Training using VMT vs. SeqFormer During iterative training, we further conduct an ablation experiment by replacing the prediction model from our VMT to SeqFormer [7]. Compared to Table 2, we adopt SeqFormer [7] to correct coarse masks of YTVIS in the iterative training. However, we observe that the improvement scales after each iteration is minor in Table 3, where the boundary quality AP^B after the 4th iteration are still coarse (around 60.0 using **GT** object classes, identities and corresponding coarse masks). The alternative method SeqFormer cannot achieve the refinement effects similar to VMT, which reveals the design advantages of our 3D incoherent regions detection and temporal refinement transformer. Note that for fair comparison, we keep the same iterative setting, and also pretrain the mask head of SeqFormer on the better annotated OVIS at the beginning.

Effect of the Instance Guidance Layer In the temporal refinement transformer, we additionally integrate our Instance Guidance Layer (IGL) after the Node Attention Layer (NAL). In Table 4, we study different injection positions and find that a level-wise manner after all three NAL achieves the best performance. The boost of 2.7 AP^B and 2.2 AP^M shows the benefit of guiding fined-grained local features using global instance-aware context.

Ablation on the RGB Encoding and Dynamic Pixel Decoder We study the effect of RGB encoding and dynamic MLP decoder in Table 5. The RGB encoding for enriching low-level details improves 0.8 AP^B while the instance-specific MLP weights further promotes AP^M from 49.3 to 50.5.

3 More Implementation Details

Implementation and Training/Inference Details We implement Video Mask Transfiner based on Detectron2 [8] and DETR [11]. We use 300 instance queries per video. We employ the backbone features from the last three stages $\{C3, C4, C5\}$ with stride $\{8, 16, 32\}$ for the base detector [7] outputting coarse object masks and detecting 3D incoherent quadtree root nodes. Then, for achieving lower-level feature details, we break down these quadtree root nodes at $\{C3\}$

to their quadrants recursively on backbone features $\{C2, C1\}$. To further enrich the object edge details, we obtain RGB encoding using a three 3×3 Conv with channels numbers $\{32, 64, 64\}$ to operate on the image. For the node encoder, we replace the 2D positional encoding in [2] with 3D encoding to incorporate the temporal dimension [6].

We set the loss function to combine Dice loss and L1 loss when performing refinement between the predicted labels for 3D incoherent nodes and their GT labels. For the experiment of Table 2 in the paper, the boundary regions are pixels within three-pixel Euclidean distance to the detected object mask contours on three corresponding feature levels.

During training, for Quadtree Sequence Grouping (QSG), we randomly permute the order of detected 3D incoherent points for each video tracklet in a video clip of length 5, and select 3,000 from them to form the new spatio-temporal sequence for refinement. This keeps the the balance between information amount contributed by each frame. Also, the same input sequence length is desired for batch efficiency. Note that during training, the detected incoherent sequence from multiple frames contains its GT labels, thus not requiring converting to final spatio-temporal masks.

During inference, we set video clip length to whole video length on YTVIS [9] and our HQ-YTVIS. In cases of OVIS [4] and BDD100K MOTS [10], we set the video clip length to 40. When video length is larger than the clip length, we split the input video into multiple clips for inference. When splitting a long video, we keep a 10-frame overlap between neighboring clips. This implicitly helps maintain instance order. We also tested a greedy matching across clips using the Tube Mask IoU, which only slightly improves the performance on OVIS (16.8→17.0 AP on OVIS with R50). Thus, we remove greedy matching to simplify VMT. Then the predicted objects masks in each video clip could be simply concatenated **w/o** further object associations and we only keep instances with scores larger than 0.001. In each video clip, to obtain the final video mask predictions, we convert the grouped sequential prediction by QSG in a video clip by splitting them to each frame.

YTVIS & HQ-YTVIS: We adopt one video clip per mini-batch. During training, the input frames are resized randomly to a shorter edge from [288, 512] pixels with a step of 32 pixels, and flip ratio is 0.5. When selecting frames to form the video clip during training, we randomly sample 5 video frames in sequence order without repetition from the same video. During inference, the testing images are resized to a shorter edge size of 360 pixels w/o data augmentation. The max number of predicted instances per frame is 10.

OVIS: For input video clips during training, we follow the training schedule and frame size settings in YTVIS & HQ-YTVIS, and limit the frame sampling range to 5 frames before and after the one sampled target frame. This is due to the large variances in the video length of OVIS dataset. We set the max number of instance to 25 since OVIS has more dense objects. Note all compared methods are trained with the same data sampling and image size settings.

BDD100K MOTS: On BDD100K MOTS, we build VMT based on the MOTS method PCAN [3]. As the base tracker, PCAN is used to produce coarse video masks for each instance. We follow the training settings of [3] and set image size 1296×720 for both training and test. Since PCAN is an online method, when building the video clip at inference, we adopt an online tube accumulation strategy, where only the previous frames before the current frame is stored. We limit the maximum tube length to 30 by removing the old frames. Since PCAN has no leanable video instance queries, incoherent regions are detected per frame follow [2] and the IGL is removed from the Quadtree Sequence Encoder when refining the input sequence produced QSG.

4 More Qualitative Comparisons

Please refer to the video file on our project page for extensive video visualization comparisons on four evaluation benchmarks YTVIS [9], the new HQ-YTVIS, OVIS [4] and BDD100K MOTS [10]. Our Video Mask Transfiner consistently produces masks with substantially higher precision and temporal consistency than existing state-of-the-art VIS methods [1,3,6,7]. Also, we show comparisons between YTVIS and HQ-YTVIS, visual ablation results of QSG and typical failure cases of VMT.

References

1. Hwang, S., Heo, M., Oh, S.W., Kim, S.J.: Video instance segmentation using inter-frame communication transformers. In: NeurIPS (2021) 6
2. Ke, L., Danelljan, M., Li, X., Tai, Y.W., Tang, C.K., Yu, F.: Mask transfiner for high-quality instance segmentation. In: CVPR (2022) 5, 6
3. Ke, L., Li, X., Danelljan, M., Tai, Y.W., Tang, C.K., Yu, F.: Prototypical cross-attention networks for multiple object tracking and segmentation. In: NeurIPS (2021) 6
4. Qi, J., Gao, Y., Hu, Y., Wang, X., Liu, X., Bai, X., Belongie, S., Yuille, A., Torr, P., Bai, S.: Occluded video instance segmentation. arXiv preprint arXiv:2102.01558 (2021) 2, 3, 4, 5, 6
5. Russell, B.C., Torralba, A., Murphy, K.P., Freeman, W.T.: Labelme: a database and web-based tool for image annotation. International journal of computer vision **77**(1), 157–173 (2008) 1
6. Wang, Y., Xu, Z., Wang, X., Shen, C., Cheng, B., Shen, H., Xia, H.: End-to-end video instance segmentation with transformers. In: CVPR (2021) 5, 6
7. Wu, J., Jiang, Y., Zhang, W., Bai, X., Bai, S.: Seqformer: a frustratingly simple model for video instance segmentation. arXiv preprint arXiv:2112.08275 (2021) 3, 4, 6
8. Wu, Y., Kirillov, A., Massa, F., Lo, W.Y., Girshick, R.: Detectron2. https://github.com/facebookresearch/detectron2 (2019) 4
9. Yang, L., Fan, Y., Xu, N.: Video instance segmentation. In: ICCV (2019) 1, 5, 6
10. Yu, F., Chen, H., Wang, X., Xian, W., Chen, Y., Liu, F., Madhavan, V., Darrell, T.: Bdd100k: A diverse driving dataset for heterogeneous multitask learning. In: CVPR (2020) 5, 6

11. Zhu, X., Su, W., Lu, L., Li, B., Wang, X., Dai, J.: Deformable detr: Deformable transformers for end-to-end object detection. arXiv preprint arXiv:2010.04159 (2020) 4