STEm-Seg: Supplementary Material

Ali Athar*[1], Sabarinath Mahadevan*[1], Aljoša Ošep[2], Laura Leal-Taixé[2], and Bastian Leibe[1]

[1] RWTH Aachen University, Germany
{athar,mahadevan,leibe}@vision.rwth-aachen.de
[2] Technical University of Munich, Germany
{aljosa.osep,leal.taixe}@tum.de

Abstract. In this supplementary, we provide:
- A short explanation of the Lovàsz Hinge Loss, used for training our network (Sec. I.),
- implementation and training details for our method (Sec. II.),
- explanation and training details for several baselines, evaluated in our experimental section (Sec. III. and VI.),
- and finally, additional qualitative results on three different datasets (Sec. VII.).

I. Loss Function

As explained in Sec. 3.1 of the paper, we use a loss function that is a linear combination of three components:

$$L_{\text{total}} = L_{\text{emb}} + L_{\text{smooth}} + L_{\text{center}}, \tag{1}$$

here L_{smooth} is the variance smoothness loss, which ensures that the network outputs uniform variance values for every object instance. For example, if the network outputs the set of variances $\boldsymbol{\mathcal{V}}_j$ for the j^{th} instance in a video clip, then the variance smoothness loss for this set of variances is denoted by L^j_{smooth} and is computed as:

$$L^j_{\text{smooth}} = \frac{1}{|\boldsymbol{\mathcal{V}}_j|} \sum_{\boldsymbol{v} \in \boldsymbol{\mathcal{V}}_j} (\boldsymbol{v} - \bar{\boldsymbol{v}})^2 \,,$$

where $\bar{\boldsymbol{v}}$ is the *mean of the variances* in $\boldsymbol{\mathcal{V}}_j$. Likewise, the loss can be computed for all object instances in the video clip and averaged. No loss is applied to the variances output for background pixels.

L_{center} is a regression loss, which ensures that pixels belonging to a foreground object instance have probability values in the instance center heat map $\mathcal{H}$ that match the probability obtained by applying Eq. 2 (main text) to the embedding vector at that pixel location.

L_{emb} is the embedding loss, and is computed using the Lovàsz extension of the hinge loss for binary segmentation, as explained below.

The Lovàsz Hinge Loss: We use the Lovàsz Hinge Loss [1] to train the embeddings output by our network (L_{emb}). It is a convex surrogate of the Jaccard

index which directly optimizes the Intersection over Union (IoU) between the predicted and the ground truth object mask tubes, thereby alleviating class imbalance issues that arise from using, *e.g.*, the cross-entropy loss. In practice, we apply the Lovàsz Hinge loss for binary segmentation. For a given video object instance prediction, we use F to denote the set of scores for each pixel in the video[3], and denote by Δ_J the set of incorrect pixel predictions [4]. The loss $\mathcal{L}_{\text{lovasz}}$ can then be computed as follows:

$$\mathcal{L}_{\text{lovasz}}(F) = \bar{\Delta}_J(h(F)), \tag{2}$$

where $\bar{\Delta}_J$ is the Lovàsz extension of Δ_J, and h is the hinge loss associated with a binary prediction. Here we provide only a high-level description of the loss function. For a more detailed explanation of this loss we refer the reader to [1].

II. Implementation Details

Hardware: We train our network using a batch size of 2 on a workstation with 2 Nvidia RTX TITAN GPUs and 64GB RAM. The inference is performed on a workstation with a single Nvidia GTX 1080Ti GPU and 32GB RAM.

Training Schedule: For all tasks, the network is trained using an SGD optimizer with an initial learning rate of 10^{-3}. The learning rate is initially constant and then starts to decay exponentially after a certain number of iterations up to 10^{-5}. The exact number of iterations varies for each setting as follows:

- DAVIS'19 Unsupervised: 60k total iterations, decay begins after 20k iterations.
- YouTube-VIS: 150k total iterations, decay begins after 60k iterations.
- KITTI-MOTS: 100k total iterations, decay begins after 40k iterations.

Image Augmented Sequences: As mentioned in Sec. 4.1, we train our network on clips from actual video data in addition to sequences that have been synthesized from static images using random affine transformations and motion blur. Doing so allows us to utilize a large amount of publicly available image instance segmentation data (*e.g.*, COCO [8], PascalVOC [3], Mapillary Vistas [9]) for training purposes. We experimentally verified the performance benefit of incorporating such data in Sec. 4.3.

These augmentations were applied using the `imgaug` [5] library, which provides a built-in function that simulates image blur caused by camera motion. The affine transformations we apply consist of rotations in the range $[-10°, 10°]$, translations of up to 10% of the image dimension along each axis, and scale

[3] this is practically just the logit value for the probability computed in Eq. 2 (main text).

[4] Obtained by thresholding the probabilities as in Eq. 3 of the main text and comparing against the ground truth mask.

[5] https://github.com/aleju/imgaug

variations in the range $[0.8, 1.2]$. We also apply small random offsets to the hue and saturation values of each image. All random transformations are independent of one another, *i.e.*, we do not try to simulate consistent motion by sequentially applying the same transformation multiple times.

Video Data Augmentation: For training clips sampled from actual video data, random horizontal flipping is the only augmentation used. This is applied randomly to entire clips and not to individual frames within clips.

III. Baselines for DAVIS'19 Unsupervised

In Sec. 4.4 we compared our method to two simple proposal-based baselines: optical flow tracker (`OF-Tracker`) and Re-ID tracker (`RI-Tracker`), in addition to other published methods on DAVIS'19 unsupervised benchmark. For both, we generate per-frame mask proposals $M \in \{m_1, ..., m_n\}$ for all the objects in a video using a ResNet-101 based Mask R-CNN [4]. To ensure a fair comparison with our architecture, we train the Mask R-CNN jointly on YouTube-VIS [14], DAVIS'19 [2], and augmented images from COCO [8], as well as Pascal VOC [3] dataset for 120k iterations. We use SGD with a momentum of 0.9 and an initial learning rate of 10^{-3} with exponential decay. The mask proposals that are generated by such a re-trained Mask R-CNN are then linked over time using optical flow and re-id for `OF-Tracker` and `RI-Tracker`, respectively.

`OF-Tracker`: We use PWC-Net [11] to generate optical flow for each subsequent pair of frames in the DAVIS'19 validation set. The optical flow is then used to warp m_{i+1} onto m_i for each frame pair $\{i, i+1\}$ to generate a set of warped masks per-frame $W \in \{w_1, ..., w_{n-1}\}$ for a video sequence. A simple linear assignment based on object overlap between the warped frame w_i and the proposal m_i is then used to associate the objects in the adjacent video frames. The associated object IDs are further propagated forward throughout the video sequence.

`RI-Tracker`: For the `RI-Tracker`, we train a re-id network with a ResNet-50 [5] backbone on the DAVIS'19 [2] training set. The network is trained using a batch hard triplet loss [6] on randomly selected triplets from a random video sequence for 25k iterations. This network is then used to generate re-id vectors for all the object proposals in M, which are further associated over time using linear assignment based on the Euclidean distance between embedding vectors.

IV. Extended Ablations for Embedding Mixing Function

In Sec. 4.3, we ablated the impact of using different mixing functions $\phi(\cdot)$ that modify the embedding representation as discussed in Sec. 3.2. In Tab. 1(a) of the main text, we reported the results of this ablation on the DAVIS'19 Unsupervised validation set. Here, we provide extended results of applying different $\phi(\cdot)$ on the YouTube-VIS [14] and KITTI-MOTS [13] datasets in Tab. I. The results for DAVIS'19 have also been repeated for reference.

Mixing Function	E	DAVIS	YT-VIS	KITTI MOTS	
		$\mathcal{J}\&\mathcal{F}$	$\mathcal{AP}$	sMOTSA (*car*)	sMOTSA (*pedestrian*)
ϕ_{xy}	2	61.6	30.5	64.2	41.1
ϕ_{xyt}	3	62.6	31.8	72.5	**48.9**
ϕ_{xyf}	3	62.8	32.6	71.8	42.2
ϕ_{xytf}	4	64.2	32.4	71.9	43.6
ϕ_{xyff}	4	**64.4**	**35.0**	73.2	47.3
ϕ_{xyfff}	5	62.4	34.0	**73.4**	41.5

Table I: Ablation studies on the Impact of different embedding mixing functions on DAVIS '19, YouTube-VIS (YT-VIS) and KITTI MOTS.

For both DAVIS'19 and YouTube-VIS, the results are consistent: for the same number of total embedding dimensions (E), having a free dimension is more beneficial than having a temporal coordinate dimension. For KITTI-MOTS, however, the trend differs. In particular, we obtain similar performance with ϕ_{xyt} (72.5 and 48.9 sMOTSA on the *car* and *pedestrian* class, respectively) and ϕ_{xyff} (73.2 and 47.3 sMOTSA). In Tab. 4 (main text), we reported the results for ϕ_{xyt} since the mean sMOTSA score for the two categories (60.70) is slightly better than that of ϕ_{xyff} (60.25). We attribute this difference in part to the fact that the temporal coordinate is a more useful feature for instance separation in KITTI-MOTS than in DAVIS'19 due to the fact that object instances undergo faster motion and often enter/exit the scene mid-way through a video clip. Furthermore, the performance trends for the *car* and *pedestrian* classes seem to follow different patterns, *e.g.*, while ϕ_{xyfff} yields the highest sMOTSA for the *car* class (73.4), it is significantly lower for the *pedestrian* class.

V. UnOVOST Training on KITTI-MOTS

In the main paper (Sec. 4.4), we reported the performance of UnOVOST [7], the highest-scoring workshop submission for the DAVIS'19 Unsupervised Challenge [2], for the task of Multi-object Tracking and Segmentation (MOTS) using the KITTI-MOTS dataset [13]. We obtained the implementation from the authors [7] and re-trained and tuned the model as follows:

- We initialized a Mask R-CNN [4] network with a ResNet-101 [12] backbone with weights from an off-the-shelf model trained for instance segmentation on the COCO dataset [8]. We then altered the output layers to predict two categories, *i.e.*. *car* and *pedestrian*, and trained the network for 60k iterations on Mapillary Vistas [9] and KITTI-MOTS datasets. The training data and the backbone is thus identical to the one used for our STEm-Seg network.
- We trained a ReID network on image instance crops from KITTI-MOTS using a triplet loss [10] and *batch-hard sampling* [6].

The two most important hyper-parameters in UnOVOST are the IoU thresholds used for pruning object detections and for associating object detections based

on optical flow, respectively. We performed a grid search for these two parameters on the KITTI-MOTS validation set in order to optimize the sMOTSA score. Our observation was that the UnOVOST framework is fairly insensitive to these parameters; however, the final scores on KITTI-MOTS are consistently low (see Tab. 4 in the paper). Qualitative analysis of the results showed that the ReID network frequently makes spurious associations. We postulate that this is because object instances in KITTI-MOTS frequently have similar appearances. This differs from the object instances in DAVIS whose appearances usually differ since they span a large variety of object classes.

VI. Adaptation of TrackR-CNN to YouTube-VIS

As discussed in Sec. 4.4, we adapted the publicly available implementation [6] of TrackR-CNN [13] to the task of Video Instance Segmentation and evaluated it on the Youtube-VIS dataset [14]. To this end, we initialized the parameters of the network, which overlap with Mask R-CNN [4] with weights from a model trained for instance segmentation on COCO [8] and Mapillary Vistas [9].

In the original implementation, a class-specific re-identification embedding head was used. This was feasible for KITTI-MOTS, where there are only two object classes. In YouTube-VIS, however, there are 40 object classes, and several occur infrequently in the dataset. Furthermore, video sequences are significantly shorter, and there are usually only 1–2 objects of the same class present in a video clip. For that reason, we adapted the TrackR-CNN architecture and kept a single ReID head that is shared among all object classes. We trained the network under this setting using a batch size of 8 images for 400k iterations and evaluated multiple intermediate checkpoints. Despite these efforts, the highest $\mathcal{AP}$ score obtained was less than 10%.

A major performance bottleneck we identified is a low-resolution 14x14 RoI-Align [4] layer used in TrackR-CNN that limit the memory usage to a reasonable level. This suffices for KITTI-MOTS, which contains small pedestrian instances and cars with simple shapes, but results in very coarse segmentation masks on the YouTube-VIS dataset which contains a diverse set of objects that cover a large area of the image. The $\mathcal{AP}$ measure heavily penalizes such coarse segmentation as it is computed by taking the average over a set of IoU thresholds ranging from 0.5 to 0.95.

VII. Additional Qualitative Results

In this section, we provide additional qualitative results on the validation split of all three datasets, DAVIS'19 [2] in Fig. 1, YouTube-VIS [14] in Fig. 2 and KITTI-MOTS [13] in Fig. 3. As can be seen, our method can reliably segment and track a large variety of different objects in diverse scenarios and is fairly robust to occlusions and scale changes.

[6] https://github.com/VisualComputingInstitute/TrackR-CNN

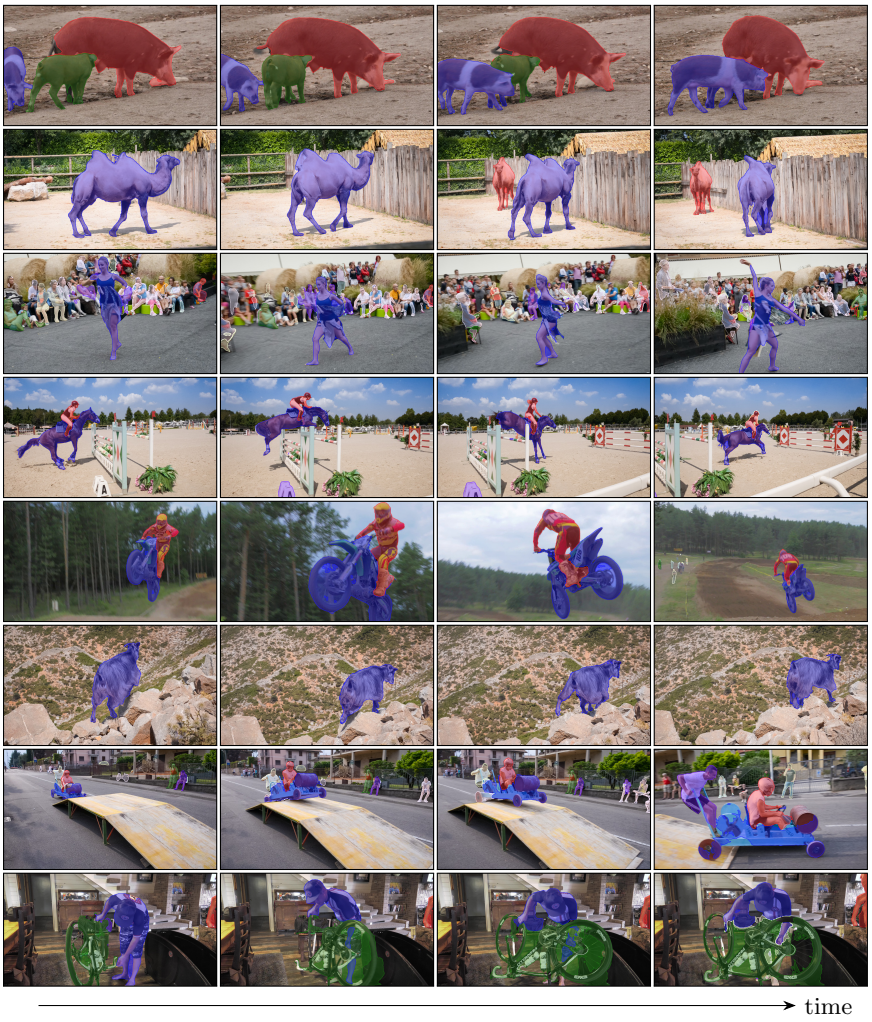

Fig. 1: Additional qualitative results on DAVIS'19. STEm-Seg generates consistently good results under varied scenarios. E.g., in the `motocross-jump` sequence (*fifth row*) it demonstrates robustness to a large change in scale. In the `bike-packing` sequence (*bottom row*), it is robust to sudden pose changes.

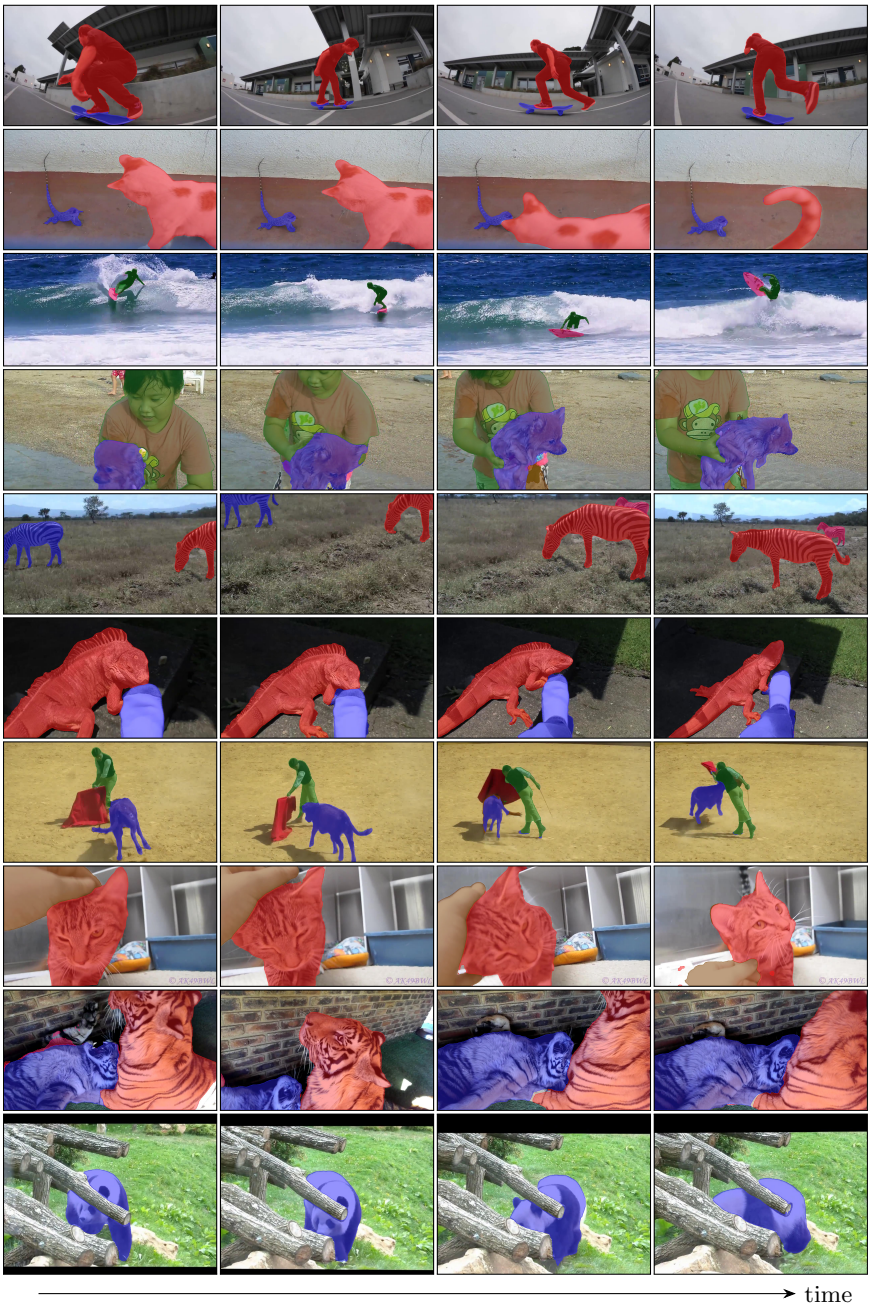

Fig. 2: Additional qualitative results on YouTube-VIS (YT-VIS) [14]. Most of the semantically challenging animal categories are successfully segmented by STEm-Seg. It also captures some fine object details such as the skateboard (*top row*) and the surfboard (*third row*) well.

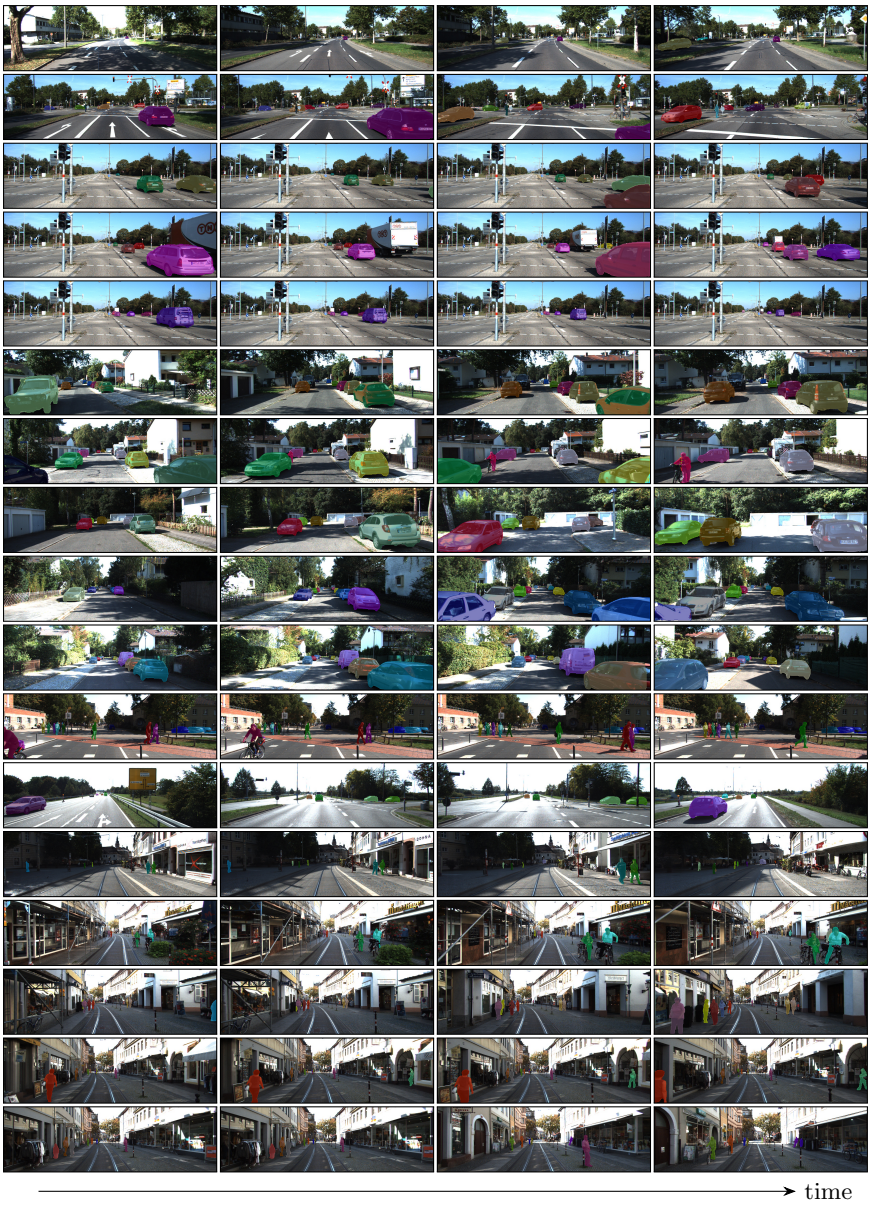

Fig. 3: Additional qualitative results on KITTI-MOTS. Our method successfully tracks and segments cars and pedestrians in automotive scenarios, even when observed from a large distance (*sixth row from the bottom*) and bridges occlusions (*fifth row*).

References

1. Berman, M., Blaschko, M.B.: Optimization of the jaccard index for image segmentation with the lovász hinge. CVPR (2018)
2. Caelles, S., Pont-Tuset, J., Perazzi, F., Montes, A., Maninis, K., Gool, L.V.: The 2019 DAVIS challenge on VOS: unsupervised multi-object segmentation. arXiv arXiv:1905.00737 (2019)
3. Everingham, M., Van Gool, L., Williams, C., Winn, J., Zisserman, A.: The pascal visual object classes (VOC) challenge. IJCV **88**(2), 303–338 (2010)
4. He, K., Gkioxari, G., Dollár, P., Girshick, R.: Mask R-CNN. In: ICCV (2017)
5. He, K., Zhang, X., Ren, S., Sun, J.: Deep residual learning for image recognition. In: CVPR (2016)
6. Hermans, A., Beyer, L., Leibe, B.: In defense of the triplet loss for person re-identification. arXiv preprint arXiv:1703.07737 (2017)
7. I. E. Zulfikar, J. Luiten, B.L.: UnOVOST: Unsupervised Offline Video Object Segmentation and Tracking for the 2019 Unsupervised DAVIS Challenge. The 2019 DAVIS Challenge on Video Object Segmentation - CVPR Workshops (2019)
8. Lin, T.Y., Maire, M., Belongie, S., Hays, J., Perona, P., Ramanan, D., Dollár, P., Zitnick, C.L.: Microsoft COCO: Common objects in context. In: ECCV (2014)
9. Neuhold, G., Ollmann, T., Bulo, S.R., Kontschieder, P.: The Mapillary Vistas dataset for semantic understanding of street scenes. In: ICCV (2017)
10. Schroff, F., Kalenichenko, D., Philbin, J.: Facenet: A unified embedding for face recognition and clustering. In: CVPR (2015)
11. Sun, D., Yang, X., Liu, M.Y., Kautz, J.: PWC-Net: CNNs for optical flow using pyramid, warping, and cost volume. In: CVPR (2018)
12. Szegedy, C., Ioffe, S., Vanhoucke, V., Alemi, A.A.: Inception-v4, inception-resnet and the impact of residual connections on learning. In: American Ass. of Art. Intelligence (2017)
13. Voigtlaender, P., Krause, M., Osep, A., Luiten, J., Sekar, B., Geiger, A., Leibe, B.: MOTS: Multi-object tracking and segmentation. In: CVPR (2019)
14. Yang, L., Fan, Y., Xu, N.: Video instance segmentation. In: ICCV (2019)