Supplementary Material

In this supplementary material, we provide further details on our approach (Section 6), experiment settings (Section 7) and further experimental results (Section 8) as well as visualizations (Section 9).

6. Our Approach

6.1. Placement in the Literature

There is a vast amount of related literature on segmentation, motion segmentation, moving object discovery and unsupervised feature learning. Table 8 summarizes the development in this field and where our approach fits in. We compare with previous work on supervised motion segmentation.

6.2. Architecture

We base our approach on the Mask2Former [14] architecture. Our two-stream fusion model is depicted in Figure 2. It features two identical branches with its own dedicated parameters Θ_{rgb}, Θ_{motion}, i.e. two sets of backbone, encoder and decoder. Learned attention masks M^{l-1}_{rgb}, M^{l-1}_{motion} let selected features z^{l}_{rgb}, z^{l}_{motion} from both streams at scale l interact with each other and two sets of queries q_{rgb} and q_{motion}. Finally, we fuse information from both streams into a single prediction with 1×1 convolution layers: We fuse output masks and class logits for all (100) queries. A twin-stream architecture is motivated out of convenience since we can combine pretrained branches and finetune them together. We believe that in the future a much lighter motion branch would suffice and further optimizations can be made since segmentation is mainly driven by the appearance branch. However, there may exist datasets where motion features mainly drive object detection and segmentation as can be seen in Section 9.4. We keep the default settings of [14] in terms of architecture hyperparameters. We use a ResNet-50 [26] backbone, pretrained on ImageNet [50]. Every motion stream has a 1×1 convolution layer as projection layer before the backbone.

Transformer Encoder. We use the multi-scale deformable attention Transformer [84] for encoding the backbone features. We use 6 layers applied to feature maps at resolution $1/8$, $1/16$ and $1/32$.

Transformer Decoder. We use the same transformer decoder as [14] with 9 layers in total and 100 queries. We also supervise intermediate predictions with the auxiliary loss.

7. Experimental Setup

7.1. Training Details

We follow a similar training setup as [14]. Our networks are optimized using AdamW [38] with a learning rate of 1.0×10^{-4} and a weight decay of 0.05 for all backbones. A learning rate multiplier of 0.1 is applied to the backbone. We employ gradient clipping when the 2-norm exceeds 0.1 for stability. For augmentation, we use DETR-style [12] random scaling, cropping and flipping. We follow the same losses $L = \lambda_{ce}L_{ce} + \lambda_{dice}L_{dice} + \lambda_{cls}L_{cls}$ with $\lambda_{ce} = 5.0$, $\lambda_{dice} = 5.0$. We set $\lambda_{cls} = 2.0$ for predictions

Paper	Publication	End-to-End	# Input Frames	Supervision	Modality	Task	Fusion
[5]	ECCV 2016	✗	2	✗	Optical Flow	BS	✗
[57]	ICCV 2017	✓	≥ 3	✓	RGB + Optica l Flow	BS	GRU
[6]	ECCV 2018	✓	2	✗	Optical Flow	BS	✗
[40]	ECCV 2018	✓	2	✓	RGB-D	BS, Odometry	✗
[7]	CVPR 2018	(✓)	2	✓	RGB + Optical Flow	BS	✗
[39]	CVPR 2019	✓	2	✗	RGB	BS	Attention
[68]	CVPR 2019	✓	> 2	✗	RGB + Optical Flow	IS / (VIS)	Convolution
[18]	ICCV 2019	✓	2	✓	RGB + Optical Flow	IS / (VIS)	Convolution
[42]	NeurIPS 2020	✓	2	✓	RGB + Optical Flow	Detection	Attention
[83]	AAAI 2020	✓	2	✓	RGB + Optical Flow	BS	Attention
[35]	NeurIPS 2021	✓	2	✗	Optical Flow	BS	✗
[74]	ICCV 2021	✓	2	✗	Optical Flow	BS	✗
[76]	CVPR 2021	✓	2	✓	Geom. Costs	BS / IS	✗
[44]	BMVC 2021	✓	3	✓	RGB + Geom. Costs	IS	Convolution
[3]	CVPR 2022	✓	3	✗	RGB	Object Discovery	GRU
[30]	NeurIPS 2022	✓	1	✗	RGB	BS	✗
[53]	NeurIPS 2022	✓	≤ 2	✗	RGB	VOS / Object Discovery	RNN
[20]	NeurIPS 2022	✓	≤ 2	✗	RGB	VOS / Object Discovery	RNN
[16]	BMVC 2022	✓	≤ 2	✗	RGB + Optical Flow	Binary Segmentation	Spectral Clustering
[4]	CVPR 2023	✓	3	✗	RGB	Object Discovery	Attention
Ours	2023	✓	≥ 2	✓	RGB + 3D scene flow	IS	Attention

Table 8: Taxonomy of related segmentation literature. We distinguish Binary Segmentation (BS), Instance Segmentation (IS) and Object Segmentation (OS) as task acronyms.

matched with the groundtruth and 0.1 for the "no-object" class. Finally, we also use importance sampling like [14] with $K = 12544$, i.e. 122×122 points. We train our networks on two NVIDIA RTX A6000 GPU's.

Single-modality training. We train single-modality models on the FlyingThings3D [41] dataset for 30 epochs. Out of convenience, we finetune the pretrained COCO checkpoint from [14] and randomly initialize the classification head. We train with a batch size of 12 and reduce the learning rate by a factor of 0.1 every 8 epochs.

Multi-modal training. When training multi-modal models, we take the appearance branch [14], pretrained on COCO, and freeze it similar to the setup by [18, 44]. The idea is to *retain* semantic object knowledge from an off-the-shelf detector and extend it for motion segmentation. We randomly initialize the classification head. We take the respective motion branch pretrained in the previous single-modality training. Both branches are then finetuned to combine knowledge from both modalities.
Mix 1. We trained for 10 epochs with a batch size of 6 and reduce the learning rate by a factor of 0.1 after 8 epochs. We use neg. examples as augmentation with $p_{neg} = 0.3$.
Mix 2. We follow the same setup as for Mix 1.
Mix 3. Because of the larger size of the total dataset, we only train for 5 epochs and reduce the learning rate by a factor of 0.1 after 4 epochs. We use neg. examples as augmentation with $p_{neg} = 0.05$ as a trade-off between regularization and enough pos. examples for object discovery. We note how this trade-off needs to be carefully chosen depending on what is valued in a detector. We achieve a lower rate of false negatives in this way, but have more false positives. We believe fewer false negatives are more important from a safety perspective for applications like autonomous driving.

7.2. Evaluation

During inference, we use the standard Mask R-CNN [25] setting where an image with shorter side is resized to 800 and longer side up-to 1333. We report standard COCO [34] metrics, foreground/background precision [76], Precision (Pu), Recall (Ru) and F-score (Fu) [18] and the number of false positives and false negatives over the whole split [44]. Similar to [44], we average over different IoU's $[0.01,\ 0.1,\ 0.3,\ 0.5,\ 0.75,\ 0.9,\ 0.95]$. Since we can compute matchings between groundtruth and prediction for these metrics for different confidence thresholds, we also average over multiple confidence values $[0.3,\ 0.5,\ 0.7]$ like [44]. Since datasets come in different sizes, we normalize the number of false positives/negatives. Intuitively this means, we measure the number of false positives/negatives per frame on particular data. At this point, strong models for SotA have false positive/negative detections only every k-th frame.

We came up with a strong and simple image detector baseline: Map semantically likely object classes to moving objects and others to static. Table 9 shows our distinction into moving and static classes of the COCO dataset. In theory, everything can move once a force is applied to it. However, this is hard to observe when we measure performance with current datasets (since often relevant objects move all the time). We use this mapping for the baseline model [14].

7.3. Inference times and memory

We measure runtime and max. memory usage on the first 300 examples from DAVIS using *torchscript* on a NVIDIA GeForce RTX 2080 Ti (with Intel(R) Xeon(R) CPU E5-2630 v3 @ 2.40GHz). Measurements can be found in Table 10. We measure with a batch size of 1. Note how scaling depends on the individual complexities (of the attention mechanism). Since computation of pseudo-modalities is dependent on specific off-the-shelf expert models and input resolution, we omit a total runtime comparison. (Using an optical flow estimator like RAFT [54] runs for example at ≈ 500 ms for HD-video.) It can be seen that combining multiple modalities comes with a price both for memory usage and runtime. Fusing modalities at multiple locations in the architecture adds up to this footprint. While we found bottleneck tokens [43] to not have a strong impact on performance, it might be very helpful when scaling up to high-resolution inputs, longer video or large batch sizes.

moving	person, bicycle, car, motorcycle, airplane, bus, train, truck, boat, bird, cat, dog, horse, sheep, cow, elephant, bear, zebra, giraffe, frisbee, skis, snowboard, sports ball, kite, baseball bat, baseball glove, skateboard, surfboard, tennis racket
static	traffic light, hydrant, stop sign, parking meter, bench, backpack, umbrella, handbag, tie, suitcase, bottle, wine glass, cup, fork, knife, spoon, bowl, banana, apple, sandwich, orange, broccoli, carrot, hot dog, pizza, donut, cake, chair, couch, potted plant, bed, dining table, toilet, tv, laptop, mouse, remote, keyboard, cell phone, microwave, oven, toaster, sink, refrigator, book, clock, vase, scissors, teddy bear, hair drier, toothbrush

Table 9: Moving object classes of COCO [34].

Fusion mechanism	Runtime [ms]	Max. Memory [GB]
Single-modality	447.69	1.11
MBT [43] Decoder	594.72	1.63
Naive Decoder	631	2.20
Encoder	664.43	1.77
Encoder + Decoder	682.97	2.24

Table 10: Runtime and memory footprint of our multi-modal architecture. Memory footprint depends on the fusion mechanism.

7.4. Reconstruction on DAVIS

Accurate 3D motion estimates require scale-accurate and consistent depth, which standard single-image depth predictors [47] cannot provide. We use an end-to-end SLAM system [55] to create a map of the scene and recover the camera odometry. SLAM systems usually filter out dynamic contents and outliers which would corrupt the odometry estimates. In this manner, [55] estimates a confidence value c for the factor graph optimization both for x- and y-direction. We filter $||\,[c_x,\ c_y]\,||_2$ with a threshold $\tau = 0.2$ to reconstruct only the static scene. For each frame, we reproject a locally consistent window of 5 surrounding frames to create a consistent, *static* reference depth map Z_{ref}. Similar to [47], we align each monocular depth prediction to the reference frame by estimating shift and scale parameters. It can be seen in Figure 5 on the right side that noise of the scene flow estimates (especially in the translation part of the rigid body motions) can be reduced with this strategy. However, reconstruction of casual videos is still an open problem [37] and therefore a reconstruction is not possible on all video clips of DAVIS. Nonetheless, this acts as a *proof-of-concept* so that when such a reconstruction is possible, we can improve motion segmentation as well. Finally, our results in Figure 5 show that errors in motion estimates can be compensated very well with appearance data when the training data allows it.

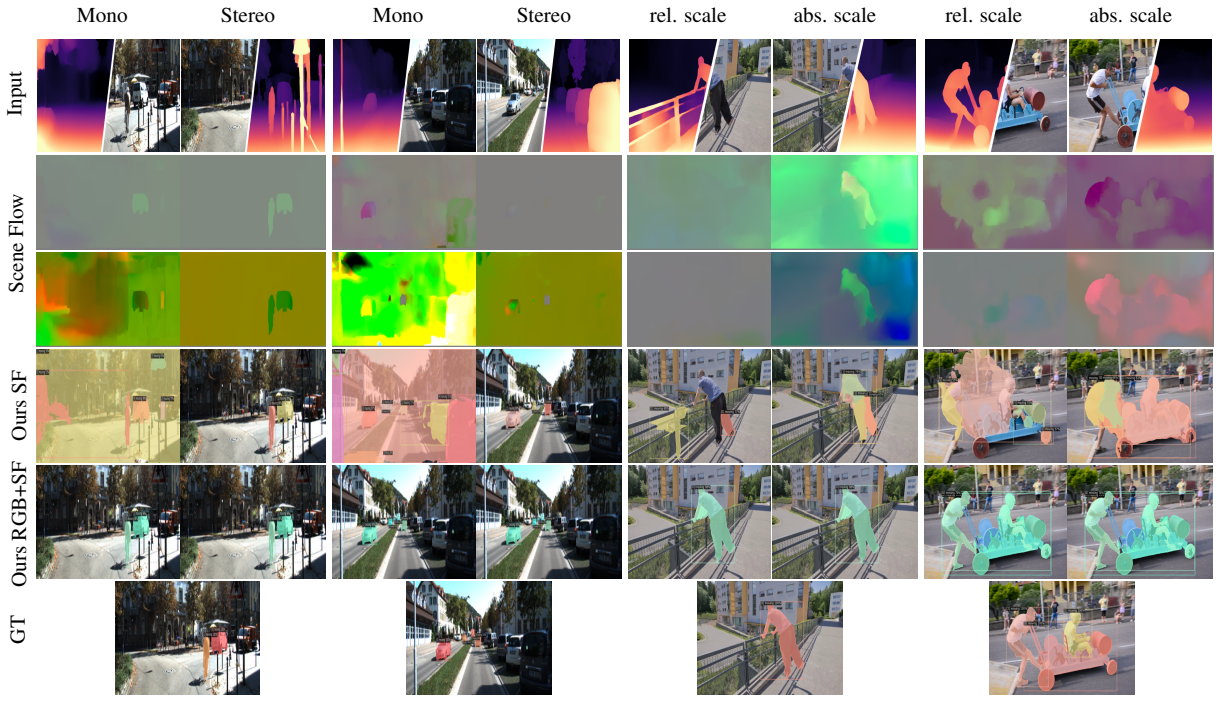

Figure 5: *Quality matters* - Failures in motion estimation can ruin a segmentation. With 3D motion, this quality also depends on the additional depth prediction. On in-the-wild data, the depth map often lacks an absolute scale, which creates even more noise in the motion field. However, this noise can be compensated by combining with appearance data. We compare the quality of monocular and stereo depth on Kitti. On Davis we can correctly adjust the scales of single-image depth and improve the downstream motion segmentation when a reconstruction [55] is possible. Note how when using a two-stream model trained on Mix 3, the differences in the motion map stop to matter as the appearance stream mainly drives the segmentation.

8. Additional Results

8.1. Ablation Fusion Strategies

We ablate different fusion mechanisms for image and optical flow input data and measure the effect of using different training data. Results can be seen in Table 11. During our initial experiments we did not find consistent performance gains from a single fusion strategy across different i) datasets ii) input modalities. For this reason, we choose to focus on late fusion in the decoder (D) or fusion at all locations (E+D) for all dataset mixes. While bottleneck tokens reduce both time and memory complexity, the performance lacks behind a naive fusion strategy. (Early) Fusion with deformable attention in the encoder can be very effective. We hypothesize that fusion in our architecture with the attention mechanism offers high degrees of freedom. This affects training dynamics considerably. We believe that similar results could be achieved with all strategies when training for long enough. Differences in this ablation experiment could be solely observable due to the training time cutoff.

Training data	Fusion mechanism	Kitti			Davis		
		AP_{50} ↑	FP ↓	FN ↓	AP_{50} ↑	FP ↓	FN ↓
Mix 1	Encoder	**46.18**	0.17	**0.37**	**36.1**	**0.24**	0.15
	Decoder	37.16	0.12	0.42	23.98	0.39	0.16
	Decoder MBT [43]	34.23	**0.08**	0.45	21.5	0.41	0.16
	Encoder + Decoder	39.65	0.15	0.40	35.88	0.29	**0.15**
Mix 2	Decoder	**64.27**	0.32	**0.26**	**65.82**	**0.16**	0.11
	Encoder + Decoder	56.41	**0.13**	0.35	65.81	0.25	**0.09**
Mix 3	Decoder	**70.82**	0.32	**0.16**	54.01	0.12	**0.01**
	Encoder + Decoder	60.88	**0.29**	0.24	**61.12**	**0.11**	0.12

Table 11: Use of different fusion mechanisms on image and optical flow data.

9. More Visualizations

In this section we add more visualizations to better explain our model behavior. We give further examples of the attention in both streams, failure cases, differences between training data mixes and generalization on the Moving Camouflaged Animal (MoCA) dataset [33].

9.1. Multi-modal Attention

Figure 6 shows the learned attention masks from both streams in our model. We further show the gradients of our output w.r.t the input data. It can be seen how different streams focus on different parts of the image to come to an output.

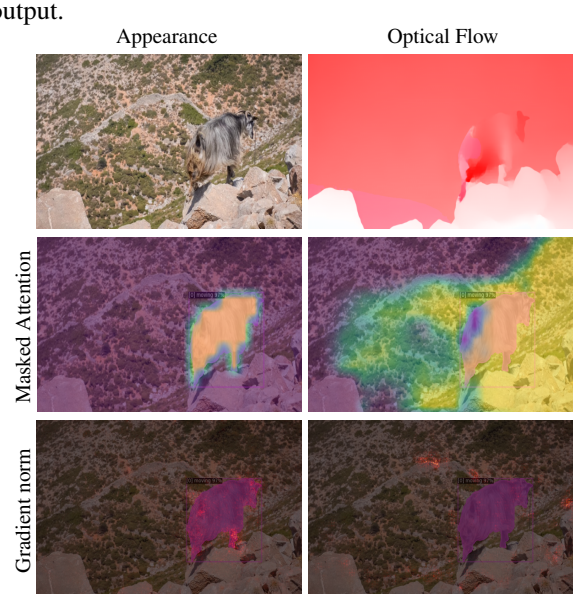

Figure 6: *What does our network see?* **a)** Multi-modal attention in our two-stream motion segmentation architecture: Masked-attention in the individual decoders. **b)** Gradient norm $||\frac{\partial S}{\partial I}||$ of the segmentation masks S w.r.t to input data I. While the appearance stream is focused on recognizing objects, motion is more global to distinguish between moving foreground and background.

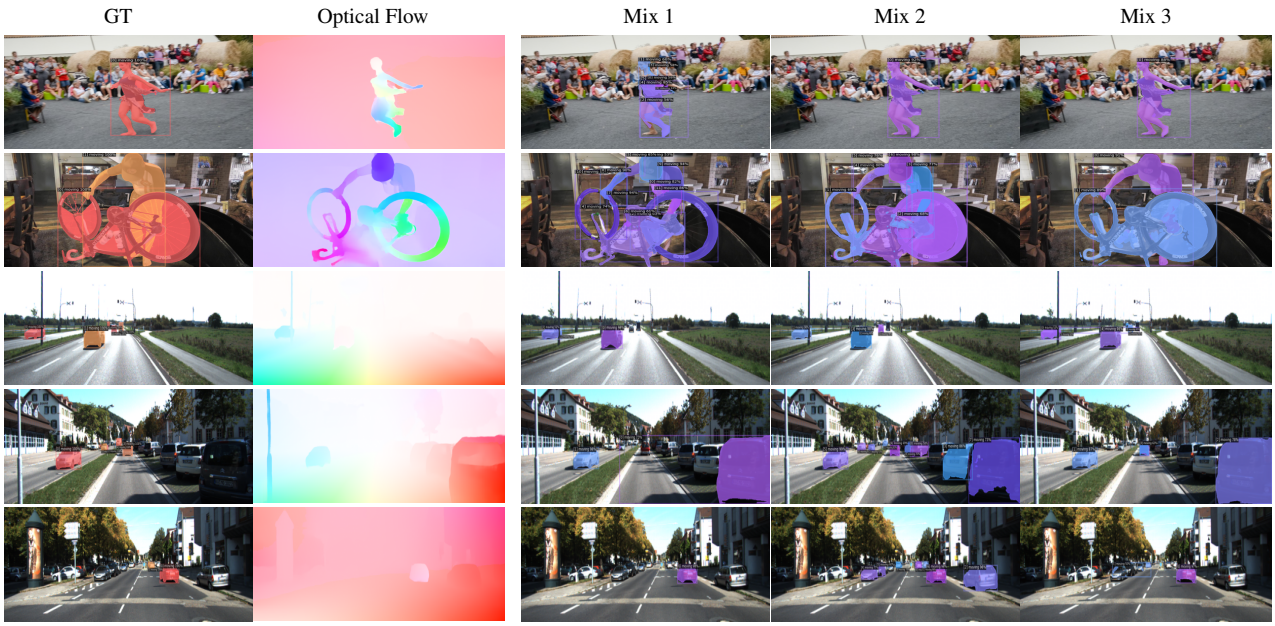

Figure 7: *Effect of diverse training data*: Models have failure cases depending on their training data. We showcase predictions by our fusion model with optical flow on different training mixes. Datasets should have: i) many different semantic classes ii) diverse motion scenarios iii) non-rigid motions iv) group of objects moving in union v) objects that *could* move but *don't*. Leveraging more data will eliminate these failure cases. This shows, that we dont actually need many changes in the model architecture for motion segmentation. Using high quality motion data and a good training set resolves most issues. Finally, even ill-posed ambiguities by 2D motion representations can be compensated with image context, e.g. a car on a driving lane is more likely to drive than a parked car to the side.

9.2. Why Diverse Training Data Is Necessary

Figure 7 shows an output progression depending on the training data mix. We observed that many common failure cases are very causal w.r.t the input training data: Models simply cannot learn non-rigid motion grouping, when not enough non-rigid motion patterns exist in the data. Simultaneously, we want driving data with common degenerate motion cases for autonomous driving. Including many diverse such cases in the training data can *logically* resolve these issues. However, overfitting can be an issue: When not balancing the training dataset cautiously, performance degrades on some dataset (here Davis), while being very good on another (here Kitti). This observation is similar to experiments on other tasks [47] or multi-task training: When combining multiple datasets, the balancing/sampling strategy is yet another optimization problem. These issues are also partially visualized in Figure 8 and 9.

We want to highlight that training on mix 2 seems to strike a very good balance when evaluating on Kitti and Davis. Note how *no real* driving data is used, yet we achieve strong mAP on Kitti even when using ill-posed optical flow. Since Mix 3 includes much more driving data, we overfit on this type of data and performance on Davis degrades. Adding other datasets like YTVOS [71] would resolve this problem. We leave the data balancing problem for future work.

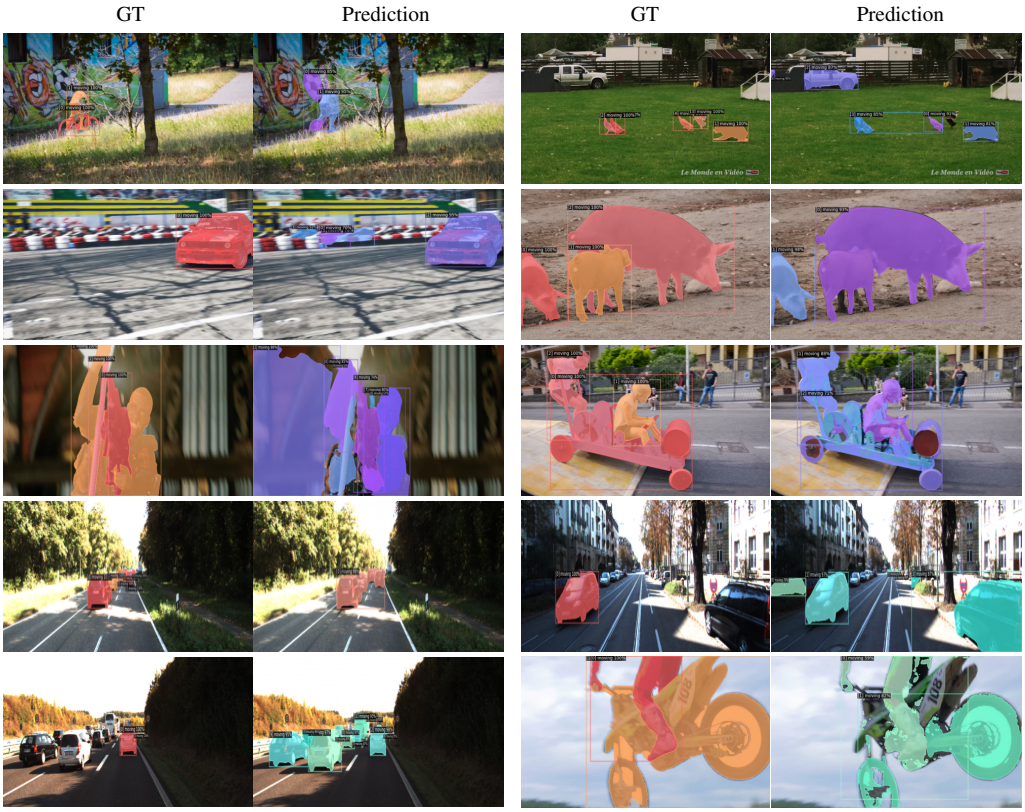

Figure 8: *Failure cases*: i) Moving objects are missed even when the motion map defines them clearly. ii) Multiple objects are grouped together iii) Objects bleed into the background iv) False Positives due to noise in motion map v) Misalignment appearance and motion stream. Semantic classes that can move often overrule the motion stream. vi) Undersegmentation vii) Small objects

9.3. Failure Cases

We show multiple failure cases explicitly in Figure 8. Other failures can be partially observed in Figure 7 and 9. We further encourage readers to view the additional videos, which contain much more information on multiple datasets and compare our trained models on multiple modalities with [44].

Figure 9: *Benefit of a twin two-stream architecture*: There exist scenes, where motion can drive the discovery and segmentation of objects, e.g. camouflaged animals [33]. While a purely appearance based detector [13] can fail, optical flow acts as a strong cue for detection, but cannot complete partial objects or leads to undersegmentations. Combining both motion and appearance cues allows to do part-based completion to semantically meaningful animals. Our model seems to generalize better than related approaches [44], likely due to the bigger architecture, attention mechanism, better appearance stream and more diverse training data. We additionally show some failure cases, where the model streams are not aligned correctly.

9.4. Why A Twin 2-stream Architecture Can be A Good Idea

In this paper, we chose an equal number of weights Θ_{rgb} and Θ_{motion}. In light of the fact that usually video drives segmentation performance, this seems to be an overkill. In Figure 9 we show a counter example where clearly the motion stream drives performance.