Supplementary Material for Discovering Objects that Can Move

Zhipeng Bao[*,†,1] Pavel Tokmakov[*,2] Allan Jabri[3]
Yu-Xiong Wang[4] Adrien Gaidon[2] Martial Hebert[1]
[1]CMU [2]Toyota Research Institute [3] UC Berkeley [4] UIUC

In this Supplementary, we provide additional experimental results, visualizations and implementation details that were not included in the main paper due to space limitations. We begin by analysing several aspects of our approach, including its memory constraints, effect of the network depth, self-supervised backbone pre-training, and influence of the number of slots in Section 1. A separate discussion of our post-processing approach for the motion segmentation outputs, together with a hyper-parameter ablation, is available in Section 2. We then report additional experimental comparisons in Section 3 including qualitative comparison with SCALOR, measurement with representative segmentation metrics, and comparison with state-of-the-art on CATER. Finally, we provide more statistics for the synthetic TRI-PD dataset in Section 4 and conclude by listing the remaining implementation details in Section 5.

1. Further ablations

1.1. Memory constrains

By adopting the learnable slot initialization and one-shot decoding strategies, our proposed method can greatly save the GPU memory. In Table 1 we compare the memory consumption for both the original SlotAttention architecture [9] and our optimized model on CATER [5] and TRI-PD [1] datasets for a single frame. Firstly, we observe that on CATER our approach does results in an about 25% reduction in the amount of memory required to train the model, though both methods are easy to fit on a single GPU due to the low resolution of CATER frames and a small number of slots. In contrast, on TRI-PD, where both the resolution and the number of slots are much larger, the memory constraints of SlotAttention become prohibitive whereas our proposed architecture can save 90% of the GPU memory, enabling experiments on this realistic dataset.

1.2. Deeper backbones

In the main paper, we used a ResNet18 backbone for all the experiments. We now further evaluate the proposed approach with ResNet34 and ResNet50 backbones on the TRI-PD dataset in Table 2. In addition, we explore the effect of self-supervised ImageNet [4] pre-training of the backbone with the recent SimSam [2] approach. We evaluate these variants with both ground truth motion segmentation and the outputs of [3] with RAFT flow, reporting Fg. ARI for both static and moving instance. We also visualize two generated samples for randomly initialized ResNet50 and pre-trained ResNet50, together with our ResNet18 baseline, for RAFT + [3] setting in Figure 1.

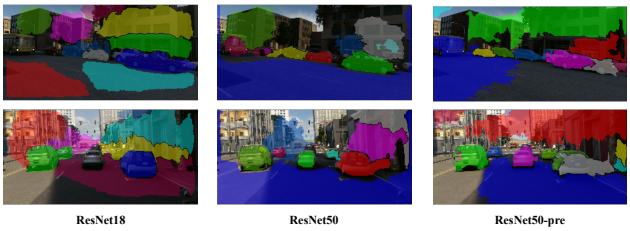

Figure 1. Visualizations of two samples on TRI-PD dataset with different backbones. Deeper backbone results in a higher confidence for the foreground objects, and self-supervised pre-training of the backbone helps to better capture the object masks.

Firstly, we observe that increasing the network depth indeed results in consistent performance improvements of our approach both with GT and estimated motion segmentation, but the improvements are somewhat larger in the former setting. This shows that the noisy estimated motion segmentation limits the performance of our method and improvements in motion segmentation algorithms would directly result in a better scalability. Secondly, self-supervised pre-training of the backbone results in further improvements for both variants, demonstrating that recent advances in self-supervised representation learning can be easily combined with our object discovery approach.

1.3. Stronger SlotAttention baselines

To further validate the effectiveness of both the proposed architecture and motion supervision, we now report two additional baseline for SlotAttention [9]. Firstly, we apply the same learning signal in the form of motion masks to [9] on

[*]Equal contribution
[†]Work done during an internship at TRI

Model	Dataset	Resolution	#Slots	GPU Memory
SlotAttention [9]	CATER	128×128	10	652 MB
Ours	CATER	128×128	10	483 MB
SlotAttention [9]	TRI-PD	548×1123	45	23,796 MB
Ours	TRI-PD	548×1123	45	2,297 MB

Table 1. GPU memory consumption for SlotAttention and our proposed method measured with Megabytes (MB). By applying learnable slot initialization and one-shot decoding, our model can save 90% of the GPU memory on the realistic TRI-PD datset with a large resolution and dozens of objects.

Backbone	Motion seg.	Fg. ARI Stat.	Fg. ARI Mov.	Fg. ARI All
ResNet18	GT moving	48.4	66.7	59.6
ResNet18	RAFT flow + [3]	45.6	56.7	50.9
ResNet34	GT moving	50.1	69.0	61.3
ResNet34	RAFT flow + [3]	46.2	57.1	51.7
ResNet50	GT moving	51.3	69.7	62.0
ResNet50	RAFT flow + [3]	47.2	57.2	52.0
ResNet50-pre	GT moving	**53.6**	**71.2**	**64.1**
ResNet50-pre	RAFT flow + [3]	48.5	68.6	53.1

Table 2. Analysis of the effect of the backbone depth and self-supervised pre-training on the model's performance on the validation set of TRI-PD. Both deeper backbones and better weight initialization result in performance improvements with either GT or estimated motion segmentation, but the improvements are somewhat higher in the former setting.

Model	Motion masks	Fg. ARI
SlotAttention	✗	64.4
SlotAttention	✓	83.1
Ours	✓	**92.7**

Table 3. Comparison with SlotAttention using motion masks supervision on CATER. Independent motion signal can also improve the perofrmance of this baseline, but it remains below that of our model, indicating the effectiveness of our model design.

CATER and report the results in Table 3. We observe that the independent motion prior indeed also improves the performance of the SlotAttention, but it remains 9.6 Fg. ARI points below our method. This result indicates that our model architecture not only dramatically reduces memory consumption, enabling the experiments on the realistic TRI-PD and KITTI datasets, but also improves the object discovery capabilities of the approach.

Next, we explore whether the supervised pre-training of the motion segmentation approach of Dave et al. [3] on the FlyingThings3D dataset [10] provides an unfair advantage to our method. To this end, we pre-train SlotAttention on FlyingThings3D, which is a more direct form of utilizing these labels, and report the results in Table 4. The results indicate that while pre-training on object segmentation labels in [10] does result in a small improvement for the SlotAttention, its the performance remains low and the model fails to discover the objects in realistic images. This is due to the large domain gap between the toy FlyingThings3D and the photo-realistic TRI-PD datasets. This toy data, however, is

Model	Pre-training	Fg. ARI
Slot Attention	None	10.2
Slot Attention	FlyingThings3D [10]	19.1
Ours	None	**50.9**

Table 4. Comparison with SlotAttantion pre-trained on FlyingThings3D [10] on the validation set of TRI-PD. Direct pre-training only results in minor improvements for SlotAttention, whereas using these labels to train a motion segmentation approach which later bootstraps object discovery in our framework is a much more effective strategy.

sufficient to learn to segment moving objects in the optical flow field - a low level task with an appearance agnostic input. The resulting model can then be used to bootstrap object discovery in real world environments.

1.4. Influence of the number of slots

In the main paper, we use a fixed number of slots, which is slightly larger than the maximum number of objects for each dataset. Here we ablate the effect of the number of slots on our method's performance and run-time on CATER and TRI-PD in Figure 2. Firstly, we observe that, unsurprisingly, using fewer slots than the maximum object count in a dataset results in a decrease in performance. However, increasing the slot number has a minimal effect on Fg. ARI and run-time. The latter is due to our efficient 1-shot decoding strategy, described in Section 3.2 of the paper. These results demonstrate the flexibility of our method, which does not require the ground truth object count for training.

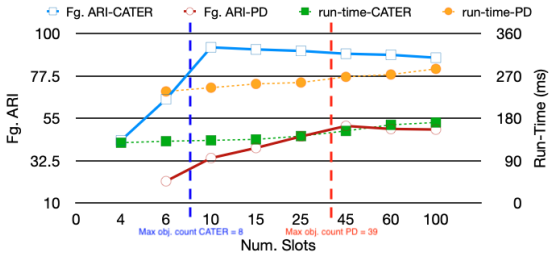

Figure 2. Influence of the number of slots on our method's performance and run-time on CATER and TRI-PD.

2. Motion-segmentation post-processing

We now describe the motion segmentation post-processing steps applied in our work. Firstly, we filter out extremely small (fewer than 100 pixels, or smallest dimension of the enclosing bounding box less than 10) and extremely large (occupying more than 60% of the image) segments since those typically correspond to random noise or capture background regions. We then additionally remove the segments that are within 15 pixels from the image boundary, as well as segments containing more than one connected component, which also typically correspond to background and noisy regions respectively.

Method	T_{con}	T_{mag}	#Seg	mIoU (↑)
CUT [7]	-	-	1.53	2.5
TSAM [3]	0.4	0.1	0.38	2.9
TSAM [3]	0.4	0.05	0.51	3.1
TSAM [3]	0.25	0.1	0.57	3.2
TSAM [3]	0.25	0.05	0.63	3.4
TSAM [3]	0.1	0.1	0.71	3.4
TSAM [3]	0.1	0.05	0.92	3.3

Table 5. Fg. ARI measurements and averaged number of segments on TRI-PD dataset with different threshold for the post-processed motion segments. We set T_{con} as 0.25 and T_{mag} as 0.05 in the main paper based on the ARI scores.

These rules are applied to the outputs of both the heuristic-based (CUT) [7] and the learning-based (TSAM) [3] motion segmentation algorithms, with the only difference being that, since [7] outputs spatio-temporal segments, we average the frame-level statistics over time. For the method of Dave et al. [3] we directly apply the rules at every frame.

In addition, unlike the heuristic-based approach, the method of Dave et al. [3] also predicts a confidence score for each segment and applies an internal pre-processing step to the optical flow, zeroing out flow vectors with a low magnitude, since those are unreliable. We integrate both of these components into our post-processing algorithm by filtering out segments with confidence score lower than T_{conf}, and average normalized flow magnitude lower than T_{mag} (we first normalize flow magnitude to be $\in [0, 1]$ for each frame).

We select these two thresholds on the validation set of TRI-PD using the FG.ARI score between the post-processed motion segments and ground-truth segments corresponding to the moving objects in Table 5. In addition, we report the average number of segments per frame after post-processing under $\#Seg$. Firstly, we evaluate the motion segments produced by [7] for reference and observe that while this approach outputs more segments, its accuracy is quite low, as indicated by the mIoU score. In contrast, the learning-based approach of Dave et al. [3] produces fewer segments, but they are a lot more accurate across a variety of thresholds. We also visualize 2 sample frames in Figure 3 for a qualitative comparison. For our main paper, we set T_{con} to 0.25 and T_{mag} to 0.05 to balance segmentation precision and recall.

3. Additional experimental comparisons

3.1. Qualitative comparison to SCALOR

Here we qualitatively compare our approach to the top-preforming SCALOR [6] baseline on the validation set of TRI-PD in Figure 4. Since SCALOR does not provide scores for the generated segments, we sampled 10 masks uniformly to generate the visualizations. The results indicate that SCALOR also did not work well with complicated backgrounds and could not discover the objects.

	Slot Attention	MONet	SCALOR	S-IODINE	MCG	Ours
Fg. ARI	10.2	11.0	18.6	9.8	25.1	**50.9**
F-measure	11.0	9.4	14.1	10.2	25.8	**47.1**
mIoU	9.2	7.7	12.9	13.6	24.5	**38.0**

Table 6. Evaluation on TRI-PD with with Fg. ARI, F-measure and mIoU. Metrics that do not penalize background over-segmentation are more informative but our approach shows top results overall.

3.2. Evaluation with segmentation metrics

We use Fg. ARI as the standard metric for object discovery in the main paper. The key reason is that it ignores the (unimportant) differences in how methods segment the background. For a more comprehensive understanding of the method, we also evaluate with F-measure [11] and mIoU, which are more standard segmentation metrics, on TRI-PD dataset and report the results in Table 6. Firstly, we observe that our method still outperforms prior work on both metrics. Secondly we notice that F-measure has the same property as Fg. ARI (ignoring the background segments) and provides similar conclusions. In contrast, mIoU penalizes background over-segmentation and thus is less informative in this setting.

3.3. Comparison with SOTA on CATER

For completeness, we now compare our approach (with estimated motion segmentation) to state-of-the-art on the toy CATER dataset, and report the results in Table 7. For these experiments, we use the original, shallow backbones for prior works, in contrast to ResNet18 used in KITTI evaluation, since we observed that they achieve higher performance on CATER. We find that: (1) the baselines' performances are lower compared to CLEVR used in these works due to higher scenes complexity (*e.g.*, more occlusions); (2) The conclusions from the main paper hold, with our method showing top results; (3) heuristic-based MCG outperforms most of the recent object-centric learning approaches even on this toy dataset, highlighting the importance of using strong baselines.

	SlotAttention	MONet	S-IODINE	SCALOR	MCG	Ours
Fg.ARI	67.3	88.6	73.5	74.6	84.0	**90.4**

Table 7. Comparison with prior art on CATER. Our method shows top results, and MCG outperforms most learning-based methods.

4. Parallel Domain dataset details

In this section we describe the details of our synthetic TRI-PD dataset, which was collected through a state-of-the-art synthetic data generation service [1]. The whole dataset contains 200 photo-realistic scenes with driving scenarios in city environments captured at 20 FPS. Each video is 10 seconds long, with a fixed shape at 1936×1216, and comes with 7 different independent camera views. A comprehensive set of ground truth labels is provided for every

Figure 3. Visualizations of the motion segmentation post-processing with different methods and thresholds. For TSAM, with a loosing constrain, we can find more segments, but also more noisy parts. We set T_{con} as 0.25 and T_{mag} as 0.05 to balance the quality and quantity of the segments.

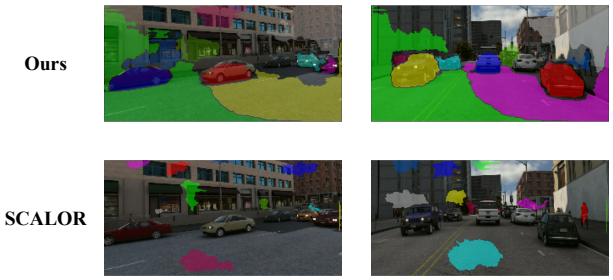

Figure 4. Visual comparison of our method and SCALOR. Despite relatively high performance, this approach also fails to discover most of the objects.

Category	Num./f	RoM	RoS	SObj	MObj	LObj
Pedestrian	0.12	0.76	0.24	0.82	0.16	0.02
Bus	0.13	0.73	0.27	0.21	0.46	0.33
Car	5.44	0.37	0.63	0.49	0.38	0.13
Bicyclist	0.15	0.73	0.27	0.79	0.19	0.02
Caravan/RV	0.05	0.75	0.25	0.20	0.56	0.24
OtherRider	0.09	0.78	0.22	0.72	0.25	0.03
ConstructionVehicle	0.01	0.78	0.22	0.72	0.25	0.03
Bicycle	0.15	0.73	0.27	0.79	0.19	0.02

Table 8. Statistics of Parallel Domain (TRI-PD) dataset. The averaged number of object per frame (Num./f), Ratio of Moving objects (RoM), Ratio of Static objects (RoS), Small Object ratio (SObj), Medium Object ratio (MObj), and Large Object ratio (LObj) are reported.

video, which include: camera pose, calibration, depth, instance segmentation, semantic segmentation, 2D bounding box, 3D bounding box, depth, forward 2D motion vectors, backward 2D motion vectors, forward 3D motion vectors, and backward 3D motion vectors. Figure 5 shows several samples of the data and corresponding annotations. We filter out scenarios with low visibility (e.g. foggy and dark scenes) which are not useful in the context of object discovery, resulting in 154 scenes which we use for training. In addition, we render another 17 scenes separately for evaluation. We use 6 camera views for training, and 3 for evaluation, resulting in 924 training and 51 test videos. The dataset is available at our project web page: https://github.com/zpbao/Discovery_Obj_Move.

We define the objects belonging to the following categories as the foreground objects: Pedestrian, Bus, Car, Bicyclist, Caravan/RV, OtherMoveable, Motorcycle, Motorcyclist, OtherRider, Train, Truck, ConstructionVehicle, and Bicycle. We filter out the objects with over 50% occlusion and fewer than 150 visible pixels. To find the independently moving objects in a pair of consecutive frames F^t, F^{t+1}, we first propagate all the object centers from F^t to F^{t+1} with the ground truth camera motion. Then we calculate the distances between these propagated object centers and the ground truth centers in F^{t+1}. If the distance is larger than 0.05, we label that object as moving independently from the camera.

We also report some statistics for each foreground category, including averaged number of object per frame (Num./f), Ratio of Moving objects (RoM), Ratio of Static objects (RoS), Small Object ratio (SObj), Medium Object ratio (MObj), and Large Object ratio (LObj) in Table 8. Notice that for some foreground categories, there is no object found in our subset. We define objects that cover fewer than 2000 pixels in the original resolution as small objects, larger than 2000 pixels but fewer than 30000 pixels as medium objects, and larger than 30000 pixels as large objects. We only count the objects for which at least 50% of the object mask is visible. Although for most categories more than half of the objects are in motion, in practice only a tiny fraction of these objects are captured by our motion segmentation algorithm (see low FG.ARI values in Table 5).

5. Implementation details

To increase the output resolution of the feature map of the encoder, we modify the standard PyTorch implementation of a ResNet [1]. In particular, we reduce the downsampling ratio

[1] https://github.com/pytorch/vision/blob/main/torchvision/models/resnet.py

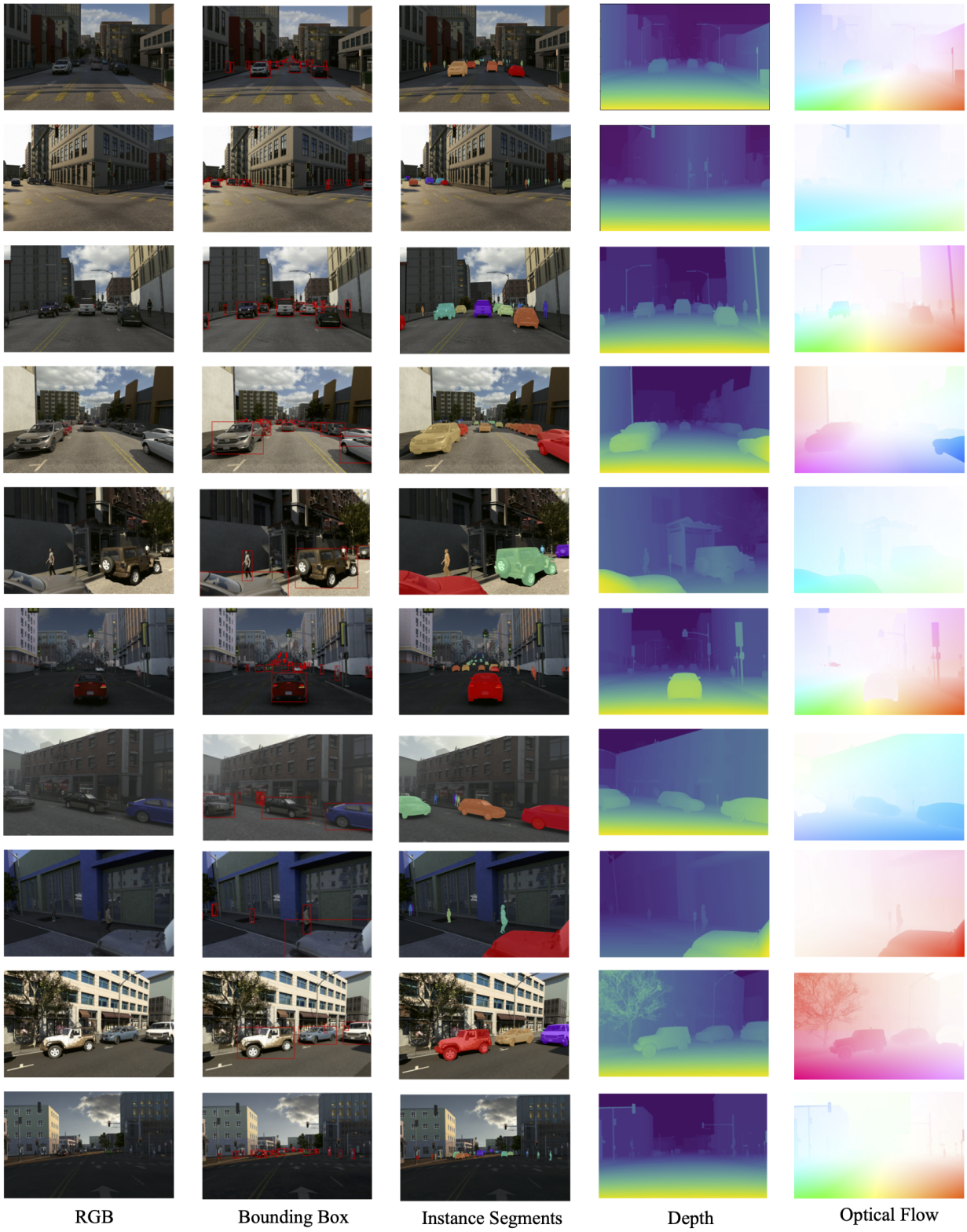

Figure 5. Several samples from the synthetic TRI-PD dataset, together with corresponding annotations.

from 16 to 4 by using stride 1 for for all the convolutional blocks except for the first one. We further drop the last fully-connected layers of the ResNet to obtain a feature map. For the decoder, we adopt the 4-layer shallow decoder following [9].

All the models are trained for 500 epochs using Adam [8] with a batch size 20 and learning rate 0.001. Following [9], we use a learning rate warm up for 2000 iterations. For the exponential learning rate decay schedule, we set the decay rate as 0.5 and the decay step as 500000. We set λ_M to 0.5 and λ_T to 0.01. D_{slot} and the output dimension for convGRU are set to 64.

To convert the attention maps W to segmentation masks, we first apply a SoftMax along the slot dimension to obtain a distribution over slots for each pixel. We then take the argmax of this distribution to assign each pixel to one of the slots and treat the resulting assignment as the masks.

References

[1] Parallel domain. https://paralleldomain.com/, November 2021. 1, 3

[2] Xinlei Chen and Kaiming He. Exploring simple siamese representation learning. In *CVPR*, 2021. 1

[3] Achal Dave, Pavel Tokmakov, and Deva Ramanan. Towards segmenting anything that moves. In *ICCV Workshops*, 2019. 1, 2, 3

[4] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. ImageNet: A large-scale hierarchical image database. In *CVPR*, 2009. 1

[5] Rohit Girdhar and Deva Ramanan. CATER: A diagnostic dataset for compositional actions and temporal reasoning. In *ICLR*, 2020. 1

[6] Jindong Jiang, Sepehr Janghorbani, Gerard De Melo, and Sungjin Ahn. SCALOR: Generative world models with scalable object representations. In *ICLR*, 2020. 3

[7] Margret Keuper, Bjoern Andres, and Thomas Brox. Motion trajectory segmentation via minimum cost multicuts. In *ICCV*, 2015. 3

[8] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *ICLR*, 2015. 6

[9] Francesco Locatello, Dirk Weissenborn, Thomas Unterthiner, Aravindh Mahendran, Georg Heigold, Jakob Uszkoreit, Alexey Dosovitskiy, and Thomas Kipf. Object-centric learning with slot attention. In *NeurIPS*, 2020. 1, 2, 6

[10] Nikolaus Mayer, Eddy Ilg, Philip Hausser, Philipp Fischer, Daniel Cremers, Alexey Dosovitskiy, and Thomas Brox. A large dataset to train convolutional networks for disparity, optical flow, and scene flow estimation. In *CVPR*, 2016. 2

[11] Peter Ochs, Jitendra Malik, and Thomas Brox. Segmentation of moving objects by long term video analysis. *PAMI*, 2013. 3