

Supplementary Material for Object Discovery from Motion-Guided Tokens

In this supplementary material, we provide additional experimental results, visualizations, and implementation details that were not included in the main paper due to space limitations. We begin by showing more visualizations of our models on TRI-PD [1] and KITTI [7], including an additional sample video in Section A. The full results of the architecture analysis (Table 1 in the main paper) are provided in Section B. In Section C, we show additional experimental evaluations including comparisons to a most recent approach, comparisons to the state-of-the-art on MOVi-C [8] dataset, per-frame evaluation on MOVi-E, and comparison to STEVE equipped with a perceiver decoder. Finally, we provide further implementation details in Section D.

A. Additional Visualizations

In this section, we provide more visualizations of the proposed MoToK framework on the realistic datasets – TRI-PD [1] and KITTI [7]. Full samples from the test set of TRI-PD can be found in the supplementary video, where we use the model trained with ground-truth motion cues to render the slot and token visualizations.

A.1. TRI-PD

We show more visualizations of the discovered object masks and corresponding tokens for our model trained with both ground-truth motion segmentation (upper) and estimated using [3] (bottom) in Figure A. Differently from the main paper, here we show the visualizations for all the slots but discard any slots that capture more than 20% of the pixels in a frame (usually background). When trained using ground-truth motion cues, our model captures accurate object masks and demonstrates strong temporal consistency; we still maintain good temporal consistency with estimated motion but the accuracy around the object boundaries drops.

We also find that token representations group pixels into mid-level regions based on texture, color, and location. These structured mid-level representations simplify the object discovery problem in our framework compared to grouping in the raw RGB space.

A.2. KITTI

We show additional full-resolution samples from the evaluation set of KITTI in Figure B. Our MoTok model accu-

Motion	Space	Decoder	FG. ARI
✗	VQ	Linear	14.6
✗	VQ	Linear-CNN	16.3
✗	VQ	Transformer	40.1
✗	VQ	Perceiver	52.4
✗	RGB	Linear	12.9
✗	RGB	Linear-CNN	11.7
✗	RGB	Transformer	38.4
✗	RGB	Perceiver	49.2
✗	Flow	Linear	8.5
✗	Flow	CNN	9.1
✗	Flow	Transformer	35.7
✗	Flow	Perceiver	36.3
✗	Depth	Linear	7.8
✗	Depth	Linear-CNN	7.6
✗	Depth	Linear-Transformer	13.2
✗	Depth	Perceiver	18.4
✗	Flow + Depth	Perceiver	38.0
✗	VQ (flow)	Perceiver	44.6
✓	VQ	Perceiver	63.8

Table A. Full model architecture analysis with different choices of slot decoders and reconstruction space on MOVi-E. Perceiver decoder + VQ reconstruction space yields the best performance, indicating that (1) the capacity of the decoder plays a key role in the object discovery; (2) learnable, vector-quantized reconstruction space outperforms fixed alternatives like depth of flow.

rately captures both pedestrians and vehicles in the real-world in a self-supervised way. Notably, our model captures the pedestrian in the shadow in the top right, and segments all the cars in the second row, illustrating the robustness of our approach.

However, compared with the results on TRI-PD (Figure A), we miss some objects and the masks are not as precise. For example, a clearly visible car in the top left is missed, and the van in the top right is under-segmented. These results demonstrate that object discovery in the real-world is still an open challenge, and further improvements are necessary both in the model architect and in learning objectives.

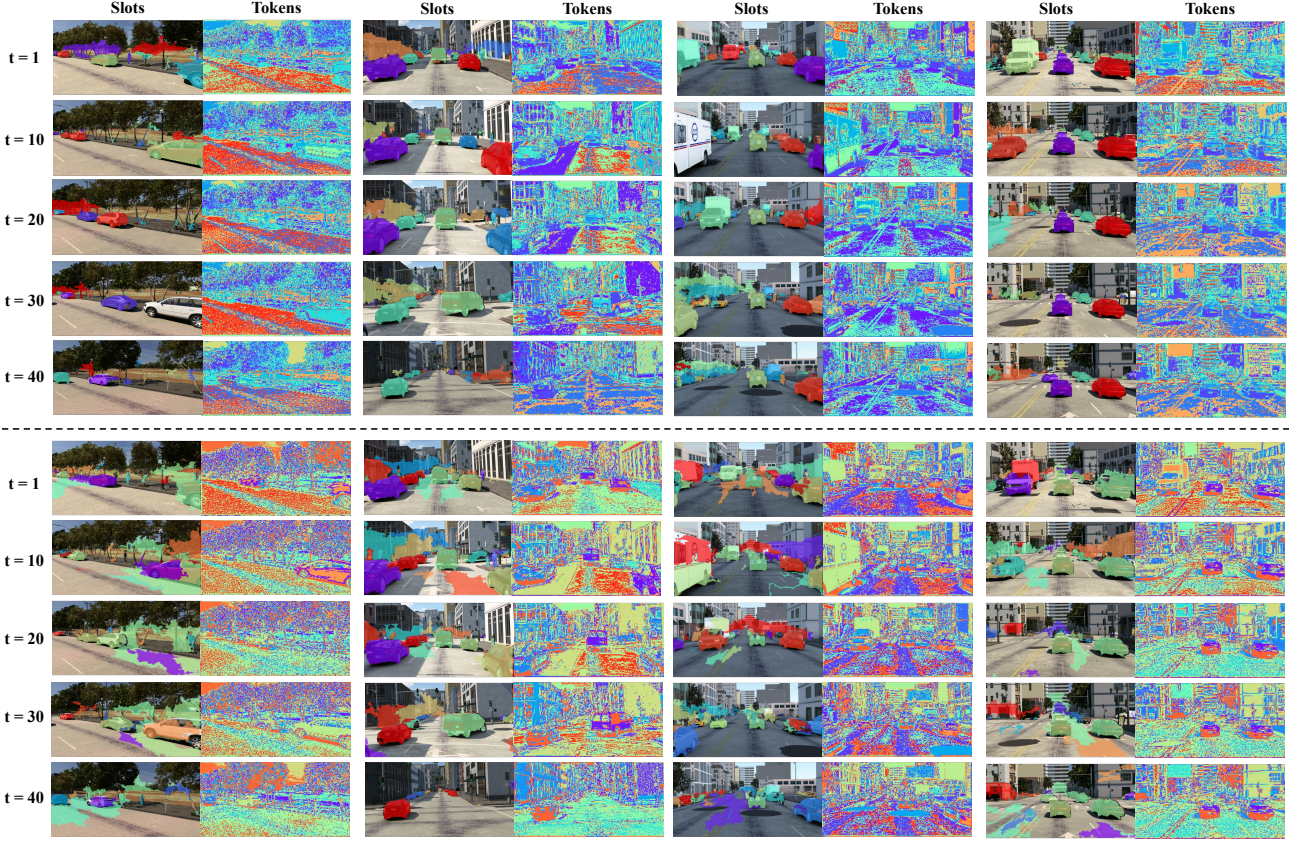


Figure A. Slot and token visualizations of the MoTok model on TRI-PD dataset. Upper: model trained with ground-truth motion cues; bottom: model trained with estimated motion cues. Our model captures accurate object masks and achieves strong temporal consistency. Token representations simplify the grouping task in the reconstruction space by grouping pixels into mid-level regions based on textures, color, and location.

B. Full Architecture Analysis

In Table 1 in the main paper, we separately study the influence of the slot decoder architecture and the reconstruction space. In Table B, we report the full results with the cross-selection of the slot decoders and reconstruction spaces.

Firstly, by comparing the results of different slot decoders, we find that the simple linear decoder generally fails to work. Adding an additional CNN decoder only brings limited performance gains, sometimes even hurting the performance. Between the two transformer variants, the more advanced perceiver decoder achieves top results, while being more computationally efficient, demonstrating the optimal model design of our MoTok framework.

Secondly, the learnable VQ space yields the best object discovery performance among the four reconstruction spaces. The underlying reason is that, as shown in Figure A, the VQ space provides a structured reconstruction space, which simplifies the grouping problem. Finally, we observe that, given a powerful enough decoder, RGB space is the most effective of all the fixed reconstruction spaces we studied,

and using a more structured space, such as flow and depth, can instead decrease performance.

C. Further Experimental Evaluations

C.1. Comparison to Concurrent Work

We additionally compare our method with a concurrent approach [14], which conducts object discovery in the ImageNet [4] pre-trained feature space of DINO [2]. For a fair comparison, we use a ViT [5] backbone for these experiments, but train it from scratch. In addition, following [14] we report per frame FG. ARI. Moreover, on KITTI, we divide the whole image into 4 patches and use 9 slots for each patch. The results in Table B demonstrate that MoTok outperforms their approach on both MOVi-E and KITTI dataset even *without* ImageNet pre-training, demonstrating the effectiveness of our proposed motion-guided tokenization.

C.2. Evaluation on MOVi-C

We additionally evaluate our method on MOVi-C dataset [8]. MOVi-C features realistic foreground and back-

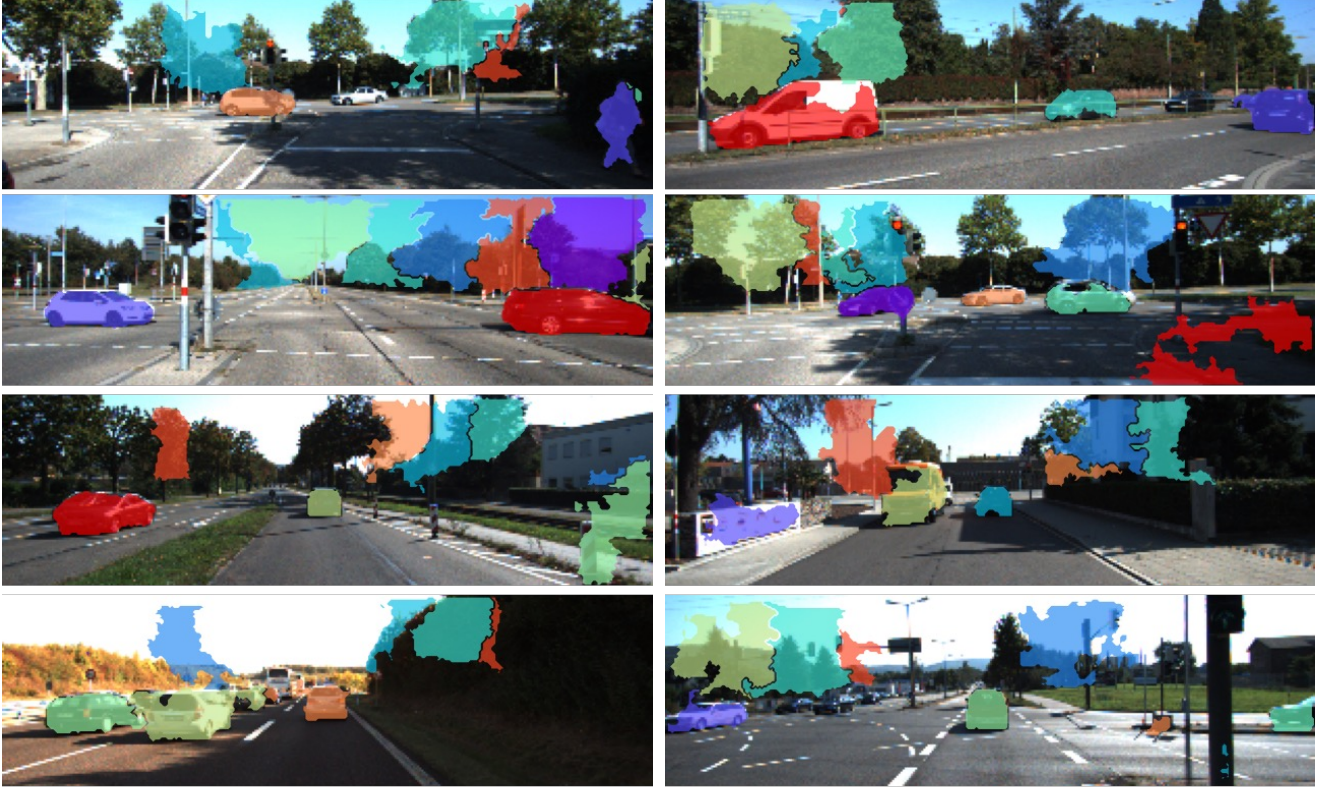


Figure B. Additional visualizations of our MoToK on the evaluation set of KITTI. We visualize top-10 slot masks and discard the masks containing more than 20% of the whole pixels (usually background). Our model successfully discovers both pedestrians (top left) and vehicles (second row). However, the results are not as accurate as those of the synthetic TRI-PD, indicating the challenges of object discovery in the real world.

Model	MOVi-E	KITTI
DINOSAUR [14]	65.1	70.3
MoTok	67.8	70.9

Table B. Per-frame FG. ARI comparison between DINOSAUR [14] and our MoTok on MOVi-E and KITTI datasets. We still outperform their approach on both datasets even *without* ImageNet pre-training, demonstrating the effectiveness of our proposed motion-guided tokenization.

ground appearance. The maximum number of objects is 10, and there is no camera motion in this dataset. We compare our method with Bao *et al.* [1], SAVI [12], SAVI++ [6], and STEVE [16], and *MoTok (no motion)* is our model without the motion cues.

The results are shown in Table C. Our model outperforms all the baselines with the help of motion cues and vector quantization. We also have the following observations. Firstly, SAVI and SAVI++, which reconstruct in the flow and depth space, achieve better performance compared with their results on MOVi-E in this scenario (all objects are moving, no camera motion). These results indicate that, while flow

Model	FG. ARI
Bao <i>et al.</i>	59.5
SAVi	35.1
SAVi++	37.4
STEVE	49.7
MoTok (no motion)	51.0
MoTok	69.3

Table C. Comparison to the state-of-the-art approaches to object discovery on the validation sets of MOVi-C using Fg. ARI. Our approach outperforms all the recent methods with the help of motion cues and vector quantization.

and depth space can help to resolve object/background ambiguity in a simplified environment, they are not sufficient as the setting becomes more realistic. We also see that, in this dataset, motion is indeed a very strong cue as our model achieves a larger improvement from motion cues compared to MOVi-E.

C.3. Per-Frame Evaluation on MOVi-E

To compare to the reported results of [10], we evaluate our method on MOVi-E using per-frame FG. ARI in Table D.

Model	per-frame FG. ARI
Bao <i>et al.</i>	55.3
SAVi	39.2
SAVi++	41.3
STEVE	54.1
Karazija <i>et al.</i> [10]	63.1
MoTok (no motion)	56.6
MoTok	66.7

Table D. Per-frame FG. ARI evaluation on the validation sets of MOVi-E. Our approach outperforms all the recent state-of-the-art approaches with the help of motion cues and vector quantization.

Model	FG. ARI
STEVE	51.2
STEVE-p	45.8
MoTok (no motion)	52.4

Table E. Comparison to STEVE with a perceiver decoder backbone. Equipping STEVE with a perceiver decoder instead decreases the original performance. Our MoTok (no motion) still achieves better performance compared to the best variant of STEVE.

Our MoTok still outperforms all the recent state-of-the-art approaches with the help of motion cues and vector quantization. Comparing with the results in Table 1 in the main paper, we find that the gap between the per-frame evaluation and video-level evaluation for SAVi and SAVi++ is much larger than for the other models, showing a worse temporal consistency of these methods. These results indicate that using a more powerful decoder is also the key to achieving strong temporal consistency in object discovery.

C.4. Equipping STEVE with the Perceiver Decoder

Both STEVE and our model use a discrete vector space in the model design. However, we propose a more powerful perceiver decoder. To reduce the impact of the decoder, we build a baseline *STEVE-p*, for which we replace the transformer decoder with the perceiver but keep the other components unchanged and report the results on MOVi-E in Table E.

Intriguingly, we find that equipping STEVE with a perceiver decoder decreases the model’s performance. We analyzed their model architecture in detail and hypothesize that this is due to the fact that they use learnable DVAE token representations as the query for the transformer. Their DVAE token representations are not optimized by the DVAE reconstruction but *only* optimized through the transformer decoder. Therefore, the self-attention + cross-attention mechanism in the transformer decoder is crucial to learn a rich feature representation, whereas the perceiver decoder only conducts a cross-attention operation with the query.

In comparison, MoTok optimizes the token representation directly through VQ-VAE reconstruction, using a learned positional embedding as a lightweight query for the transformer/perceiver decoder. Therefore, the perceiver decoder

leads to a better performance in the MoTok framework. Besides, our MoTok (no motion) also achieves better performance compared to the best variant of STEVE.

D. Implementation Details

D.1. MoTok Architecture Details

We adapt the encoder and the slot attention module from [1]. Concretely, we first use a ResNet18 as the backbone and reduce the downsampling ratio from 16 to 4 by using stride 1 for all the convolutional blocks except for the first one (but keep the stride 2 for the first convolution *layer*). We further drop the last fully-connected layers of the ResNet to obtain a feature map. We also change the hidden dimensions of the residue block to [64,64,128,128] accordingly. We use a single-layer ConvGRU with a hidden dimension of 128 and the final dimension of 64.

For slot decoders, we use two convolutional layers for the CNN decoder with kernel sizes 5 and 3. The hidden dimensions are set to 64. For the transformer decoder and the perceiver decoder, we use both single-layer decoders, with a hidden dimension of 64 as well. We use a learnable positional embedding as the query. For the implementation of transformer and perceiver decoders, we refer to the public implementations of the transformer¹ and perceiver².

For the decoder required for the flow and depth reconstruction space, we use a shallow CNN-based decoder containing 6 transposed convolutional layers. The kernel sizes are 5 except for the last layer, which is 3. The strides for the first and fourth layers are 2, otherwise are one. For the VQ space, we adopt a public VQ-VAE implementation³.

We will release our code and models for reproducibility.

D.2. Baseline Details

SAVi / SAVi++ [6,12] We use the same encoder backbone and slot attention architecture as in MoTok. We also add the corrector module designed by [12] to the slot attention union. For the decoder architecture, we also use the same CNN decoder as ours except for the stride design. We change the strides for the first four layers to 2 and the last two layers to 1, leading to an overall up-sample ratio of 16, which is the same as the original design of SAVi and SAVi++. We did not provide the data augmentation and the first-frame bounding box supervision to the two models.

STEVE [16] We implement the STEVE model by referring to their paper and the public released code of [15], which is a foundation work of STEVE. We use the same encoder backbone and slot attention architecture as MoTok and use

¹<https://github.com/lucidrains/vit-pytorch>

²<https://github.com/esceptico/perceiver-io>

³<https://github.com/ritheshkumar95/pytorch-vqvae>

the same DVAE and transformer architecture of the released code.

Bao *et al.* [1] We use the public released code⁴ for the implementation. We use the same encoder backbone and slot attention architecture as MoTok.

Estimated Annotation Generation We generated estimation annotations on TRI-PD and KITTI datasets. For flow estimation, we use self-supervised SMURF flow [17] for the two datasets. We use a self-supervised GUDA [9] approach to generate the depth annotations for both TRI-PD and KITTI. For the motion segmentation, we use [3], following the process in [1].

D.3. Training Details

All the models are trained for 500 epochs using Adam [11]. Following [13], we use a learning rate warm-up for 3000 iterations to achieve the initial learning rate of 0.0005. For the exponential learning rate decay schedule, we set the decay rate as 0.5 and the decay step as 50,000. We set λ_M to 1 and λ_T to 0.05. The gradients are further clipped to a global norm value of 1, 0.05, and 0.05 for MOVi-E, TRI-PD, and KITTI respectively. We use batch size 64 for the training of MOVi-E, and 8 for TRI-PD and KITTI. We train our model with 4 NVIDIA-A100 GPUs in parallel and it will take one day, three days, and half a day to train the model on MOVi-E, TRI-PD, and KITTI accordingly. We use 128 tokens for TRI-PD and KITTI, and 64 for MOVi. See our code implementation for more details.

References

- [1] Zhipeng Bao, Pavel Tokmakov, Allan Jabri, Yu-Xiong Wang, Adrien Gaidon, and Martial Hebert. Discovering objects that can move. In *CVPR*, 2022. 1, 3, 4, 5
- [2] Mathilde Caron, Hugo Touvron, Ishan Misra, Hervé Jégou, Julien Mairal, Piotr Bojanowski, and Armand Joulin. Emerging properties in self-supervised vision transformers. In *ICCV*, 2021. 2
- [3] Achal Dave, Pavel Tokmakov, and Deva Ramanan. Towards segmenting anything that moves. In *ICCV Workshops*, 2019. 1, 5
- [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. 2
- [5] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, et al. An image is worth 16x16 words: Transformers for image recognition at scale. In *ICLR*, 2021. 2
- [6] Gamaleldin F Elsayed, Aravindh Mahendran, Sjoerd van Steenkiste, Klaus Greff, Michael C Mozer, and Thomas Kipf. Savi++: Towards end-to-end object-centric learning from real-world videos. In *NeurIPS*, 2022. 3, 4
- [7] Andreas Geiger, Philip Lenz, and Raquel Urtasun. Are we ready for autonomous driving? the kitti vision benchmark suite. In *CVPR*, 2012. 1
- [8] Klaus Greff, Francois Belletti, Lucas Beyer, Carl Doersch, Yilun Du, Daniel Duckworth, David J Fleet, Dan Gnanaprasam, Florian Golemo, Charles Herrmann, et al. Kubric: A scalable dataset generator. In *CVPR*, 2022. 1, 2
- [9] Rares Guizilini, Vitor abd Ambrus, Sudeep Pillai, Allan Raventos, and Adrien Gaidon. 3d packing for self-supervised monocular depth estimation. In *CVPR*, 2020. 5
- [10] Laurynas Karazija, Subhabrata Choudhury, Iro Laina, Christian Rupprecht, and Andrea Vedaldi. Unsupervised multi-object segmentation by predicting probable motion patterns. In *NeurIPS*, 2022. 3, 4
- [11] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *ICLR*, 2015. 5
- [12] Thomas Kipf, Gamaleldin F Elsayed, Aravindh Mahendran, Austin Stone, Sara Sabour, Georg Heigold, Rico Jonschkowski, Alexey Dosovitskiy, and Klaus Greff. Conditional object-centric learning from video. In *ICLR*, 2022. 3, 4
- [13] 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. 5
- [14] Maximilian Seitzer, Max Horn, Andrii Zadaianchuk, Dominik Zietlow, Tianjun Xiao, Carl-Johann Simon-Gabriel, Tong He, Zheng Zhang, Bernhard Schölkopf, Thomas Brox, et al. Bridging the gap to real-world object-centric learning. In *ICLR*, 2023. 2, 3
- [15] Gautam Singh, Fei Deng, and Sungjin Ahn. Illiterate dall-e learns to compose. In *ICLR*, 2021. 4
- [16] Gautam Singh, Yi-Fu Wu, and Sungjin Ahn. Simple unsupervised object-centric learning for complex and naturalistic videos. *arXiv preprint arXiv:2205.14065*, 2022. 3, 4
- [17] Austin Stone, Daniel Maurer, Alper Ayyaci, Anelia Angelova, and Rico Jonschkowski. SMURF: Self-teaching multi-frame unsupervised RAFT with full-image warping. In *CVPR*, 2021. 5

⁴https://github.com/zpbao/Discovery_Obj_Move