Dynamic Visual SLAM using a General 3D Prior

Supplementary Material

In this supplementary material, we present additional implementation details for both the training procedure and the SLAM system in Appendix A, and provide a runtime analysis in Appendix B. We include additional details of the moving object segmentation experiment and compare our method with two additional baselines for camera tracking in Appendix C. Finally, we showcase more qualitative results in Appendix D.

A. Additional Implementation Details

For training π^3_{mos}, each training sample consists of 2–24 images uniformly sampled from a video sequence. Within each sample, an object is labeled as dynamic if it changes its position in any of the included frames, in which case the entire object mask is set to 1. During training, all images are resized to 518×518 pixels, and the model is optimized for 20 epochs using AdamW [3] with a learning rate of $1 \cdot 10^{-4}$.

For the SLAM system, we set the number of patches per frame to $K = 128$, and the threshold applied to the motion probability for distinguishing static and dynamic regions to $s_d = 0.4$. The sliding window size for pose optimization is set to 10. The threshold t_σ, which is used to reject unreliable patches in scale estimation, is set to 0.75. Every 5 keyframes, we select one historical keyframe, resulting in a total of 5 historical keyframes that are used together with the current frame as input to π^3_{mos} for inference. Regarding the initialization of our SLAM system, we use the first 12 frames as input to π^3_{mos} to obtain their initial camera poses and depth maps. These poses are used as the initial estimates for all frames, and the predicted depth maps are used to initialize the patch depths. We then run our uncertainty-aware bundle adjustment to complete the initialization.

B. Runtime Analysis

In Fig. 1, we report the per-frame runtime of each module in our SLAM system on sequence `ANYmal2` of Wild-SLAM MoCap Dataset [8], which contains 1,200 images. These modules include the time of π^3_{mos} model inference, the scale estimation that aligns the predicted depth with the patches, the uncertainty-aware bundle adjustment, and the optical flow computation for patches. It can be observed that the runtime of both scale estimation and optical flow computation stays below 10 ms, while the bundle adjustment remains under 40 ms. The main bottleneck of the system lies in the model inference of π^3_{mos}, which takes approximately 350 ms to process six frames (5 historical keyframes and the current frame). At the beginning of the sequence, the runtime of bundle adjustment and model inference gradually increases because the keyframe buffer has not yet been filled, causing the number of frames involved in both model inference and bundle adjustment to grow over time.

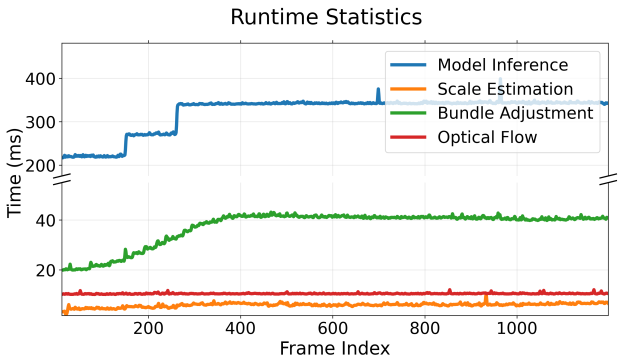

Figure 1. The runtime curves of each module in our SLAM system on the `ANYmal2` sequence of the Wild-SLAM MoCap Dataset [8].

Table 1. Ablation study for the number of selected historical frames for the inference of π^3_{mos}. We report the average ATE (cm) and FPS (Hz) for results on the Bonn RGB-D Dynamic [5] and the Wild-SLAM MoCap Dataset [8]. For the Wild-SLAM dataset, we report the average performance on all 10 sequences.

Number of Historical Keyframes	Bonn		Wild	
	ATE ↓	FPS ↑	ATE ↓	FPS ↑
2	2.75	**4.32**	0.43	**3.96**
3	2.38	<u>3.42</u>	0.39	<u>3.15</u>
4	2.26	2.93	**0.34**	2.73
5	**2.20**	2.61	<u>0.35</u>	2.42
6	<u>2.23</u>	2.43	<u>0.35</u>	2.20

To further analyze how the number of historical keyframes fed into π^3_{mos} affects system performance, we conduct an ablation study using different numbers of input frames. As shown in Tab. 1, reducing the number of historical frames improves runtime. Our SLAM system can run at 4 Hz if we only use 2 historical frames for the inference. However, increasing the number of historical keyframes beyond 5 does not yield further improvement in tracking performance, while adding unnecessary computational overhead. Therefore, we use 5 historical keyframes for all experiments reported in the main paper. This experiment was performed on an NVIDIA RTX A6000 GPU.

C. Additional Experiments and Details

Moving Object Segmentation. For the moving-object segmentation experiment, we set the motion-probability threshold to $s_d = 0.4$, consistent with the value used in our

Table 2. Camera tracking performance on the Bonn RGB-D Dynamic Dataset [5] (ATE RMSE ↓ [cm]).

Method	Balloon	Balloon2	Crowd	Crowd2	Person	Person2	Moving	Moving2	Avg.
Monocular Methods									
VGGT-SLAM [4]	4.3	3.1	3.9	3.9	6.8	10.9	1.6	3.0	4.68
π^3-Long [2]	13.6	8.1	7.0	7.9	10.3	11.4	17.8	19.6	11.96
WildGS-SLAM [8]	2.8	2.4	1.5	2.3	**3.1**	**2.7**	1.6	2.2	2.36
ours	**2.6**	**2.2**	**1.3**	**2.2**	3.2	3.0	**1.2**	**1.9**	**2.20**

Table 3. Camera tracking performance on the Wild-SLAM MoCap Dataset [8] (ATE RMSE ↓ [cm]).

Method	ANYmal1	ANYmal2	Ball	Crowd	Person	Racket	Stones	Table1	Table2	Umbrella	Avg.
Monocular Methods											
VGGT-SLAM [4]	1.3	68.4	1.4	2.6	1.5	2.1	1.9	4.7	92.4	3.7	18.00
π^3-Long [2]	4.8	6.2	3.6	4.7	4.6	7.0	7.6	8.9	13.1	4.1	6.45
WildGS-SLAM [8]	**0.2**	**0.3**	**0.2**	**0.3**	0.8	**0.4**	**0.3**	0.6	1.3	**0.2**	0.46
ours	**0.2**	0.4	**0.2**	**0.3**	**0.2**	**0.4**	**0.3**	**0.2**	**1.1**	**0.2**	**0.35**

Table 4. Performance of online and offline setups for moving object segmentation on the DAVIS-16 and DAVIS-17 benchmarks.

Method	DAVIS-16		DAVIS-17	
	JM↑	JR↑	JM↑	JR↑
Easi3R$_{\text{DUSt3R}}$ [1]	53.1	60.4	49.0	56.4
Easi3R$_{\text{MonST3R}}$ [1]	57.7	71.6	56.5	68.6
ours (online)	68.1	78.3	70.6	81.3
ours (offline)	**71.9**	**82.3**	**73.5**	**84.8**

SLAM pipeline as described in Appendix A. We report results using two metrics: IoU mean (JM) and IoU recall (JR). The IoU of the prediction of a single frame is:

$$\hat{\mathbf{M}}_i = \mathbf{M}_i > s_d, \tag{1}$$

$$\text{IoU}(\hat{\mathbf{M}}_i, \bar{\mathbf{M}}_i) = 100 \times \frac{|\hat{\mathbf{M}}_i \cap \bar{\mathbf{M}}_i|}{|\hat{\mathbf{M}}_i \cup \bar{\mathbf{M}}_i|}, \tag{2}$$

where $\hat{\mathbf{M}}_i$ denotes the predicted binary mask for moving objects obtained by element-wise thresholding the motion probability map $\mathbf{M}_i$, and $\bar{\mathbf{M}}_i$ denotes the corresponding ground truth moving object mask. The metrics JM and JR are calculated as:

$$\text{JM} = \frac{1}{N} \sum_{i=1}^{N} \text{IoU}(\hat{\mathbf{M}}_i, \bar{\mathbf{M}}_i), \tag{3}$$

$$\text{JR} = \frac{1}{N} \sum_{i=1}^{N} \mathbf{1}\left(\text{IoU}(\hat{\mathbf{M}}_i, \bar{\mathbf{M}}_i) > 50\right), \tag{4}$$

where $\mathbf{1}$ is an indicator function that returns 1 if the condition inside is true and 0 otherwise, and N is the number of frames considered for the metrics calculation.

We evaluate our approach in both online and offline modes. In the online setup, we follow the same configuration as in our SLAM system by maintaining a sliding window of size 6 and report moving-object segmentation results only on the last frame of each window in a rolling manner. For the offline setting, we split the entire sequence into batches of at most 20 frames and evaluate the moving object segmentation on all frames within each batch. We demonstrate the results in Tab. 4. Since baselines like Easi3R operate in an incremental online manner, we only report our online results in the main paper to ensure a fair comparison. As shown, our approach can further leverage the additional contextual information available in longer input sequences to achieve better segmentation performance.

Additional Baselines for Camera Tracking. We further compare our method against other feed-forward reconstruction model–based approaches, including VGGT-SLAM [4] and π^3-Long [2]. Since π^3-Long extends VGGT-Long [2] by replacing VGGT with π^3 as the backbone and achieves consistently better performance, we only include π^3-Long in our comparison. These baselines are not reported in the main paper because they do not estimate camera poses in a per-frame manner. Instead, they divide the sequence into chunks or submaps and perform registration across submaps, which is not suitable for online and incremental robotic applications. It is also worth noting that both methods rely entirely on the feed-forward model's point cloud predictions and do not use camera intrinsics.

We report their results on both the Bonn and Wild-SLAM datasets in Tabs. 2 and 3. Due to their heavy reliance on the model's geometric priors and the absence of explicit handling of moving objects, these methods produce large tracking errors on both datasets and clearly underperform compared to ours. In addition, we provide the complete results of our method on all 10 sequences of the Wild-SLAM dataset, along with a comparison to WildGS-SLAM [8].

D. Additional Qualitative Results

In this section, we present additional qualitative results of our SLAM system. Figure 2 shows more examples of our dense reconstructions, and Fig. 3 illustrates the moving-object segmentation results produced by π^3_{mos} on image sequences, highlighting the temporal consistency across frames.

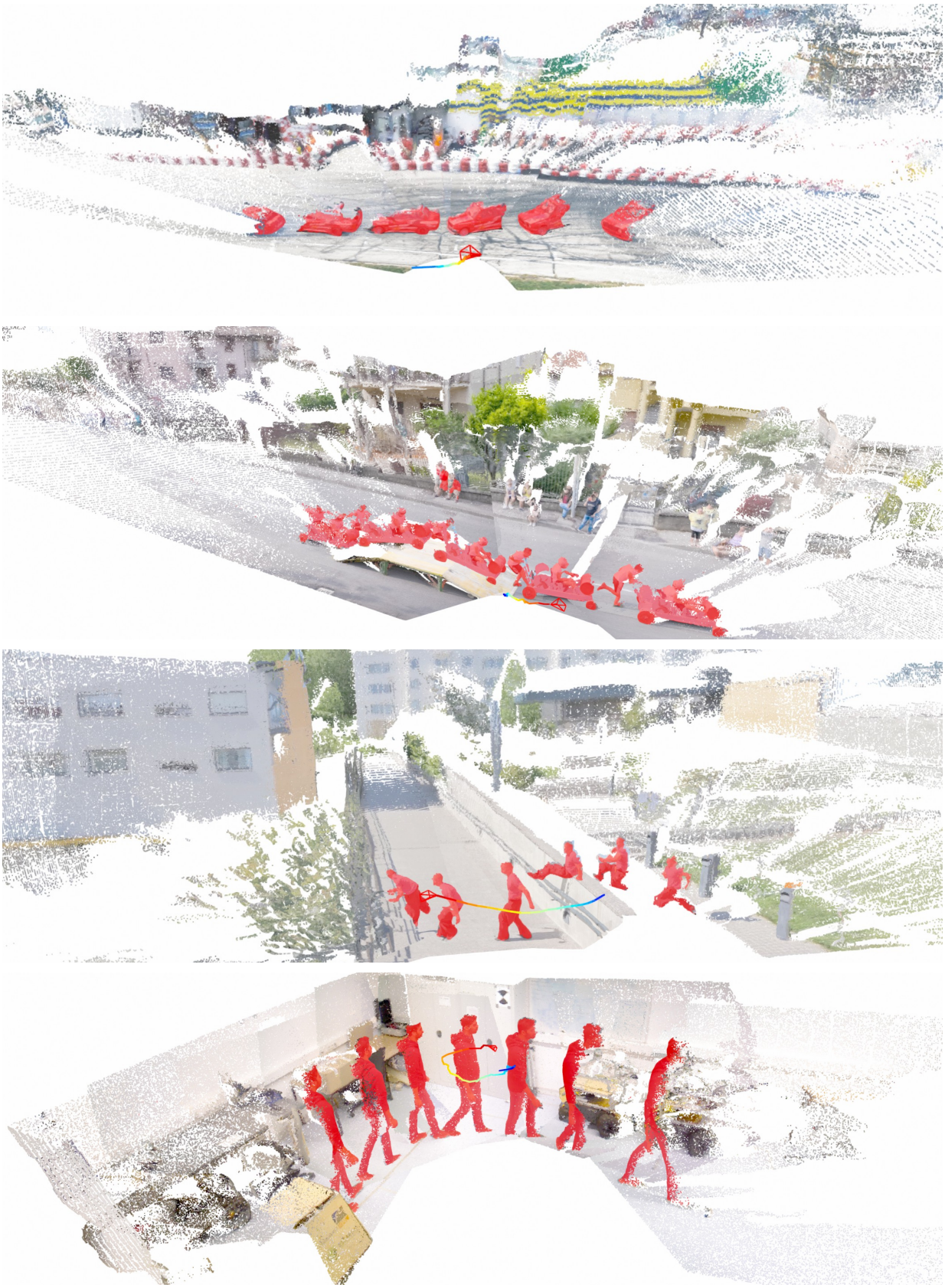

Figure 2. Dense reconstruction results of our SLAM system. For each sequence, we visualize the camera trajectory and global point cloud map obtained by accumulating the depth predictions from keyframes. Points belonging to moving objects are highlighted in red. These results demonstrate that our system incrementally produces depth estimates that remain globally consistent across the entire sequence.

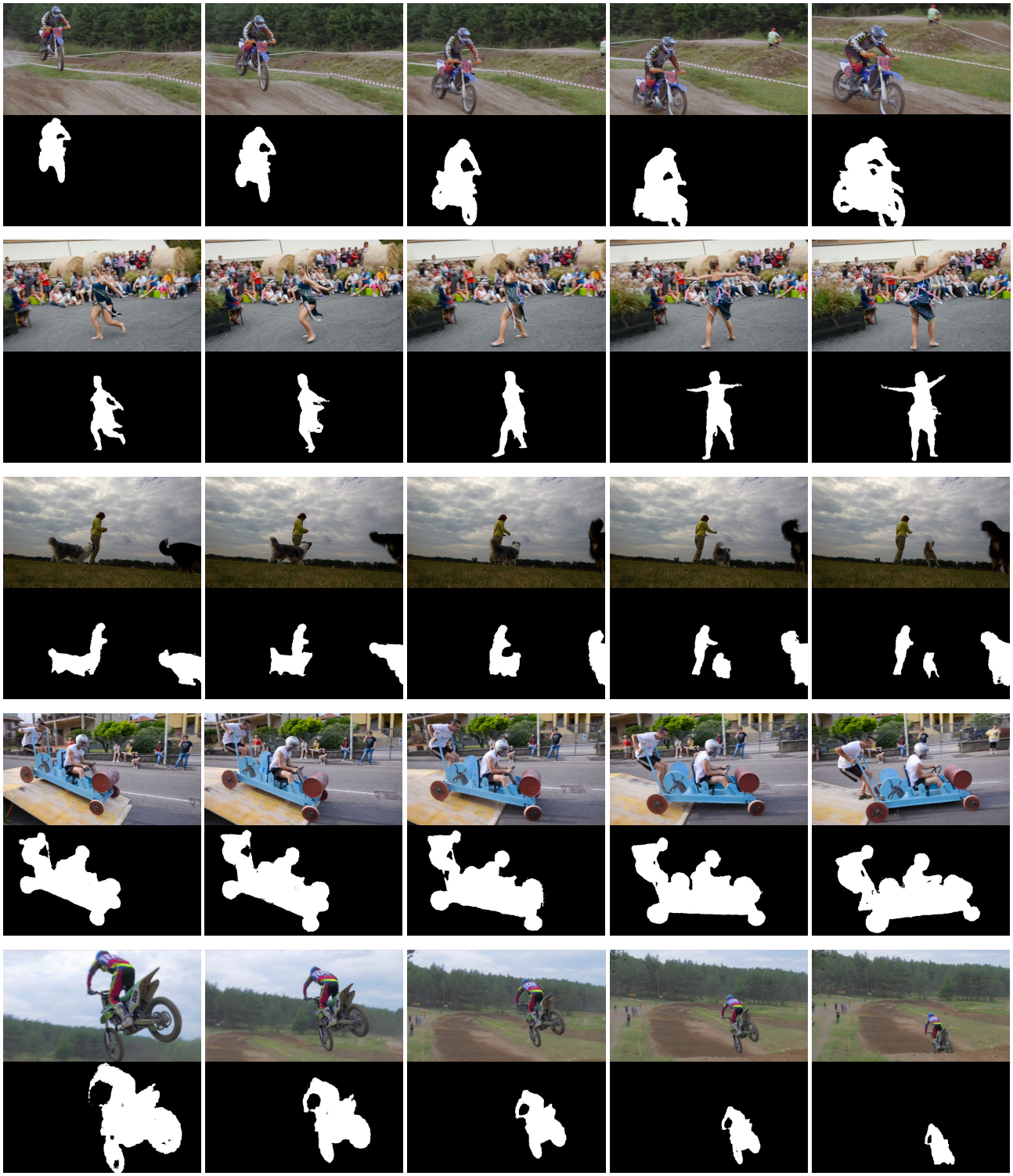

Figure 3. Additional qualitative results of moving object segmentation on the DAVIS [6, 7] datasets. For each sequence, the first row shows the input images, and the second row presents the masks of moving objects predicted by our model π^3_{mos}. As shown, the model accurately segments dynamic objects and maintains strong temporal consistency across frames.

References

[1] Xingyu Chen, Yue Chen, Yuliang Xiu, Andreas Geiger, and Anpei Chen. Easi3R: Estimating Disentangled Motion from DUSt3R Without Training. In *Proc. of the IEEE/CVF Intl. Conf. on Computer Vision (ICCV)*, 2025. 2

[2] Kai Deng, Zexin Ti, Jiawei Xu, Jian Yang, and Jin Xie. VGGT-Long: Chunk it, Loop it, Align it–Pushing VGGT's Limits on Kilometer-scale Long RGB Sequences. *arXiv preprint*, arXiv:2507.16443, 2025. 2

[3] Ilya Loshchilov and Frank Hutter. Decoupled Weight Decay Regularization. In *Proc. of the Int. Conf. on Learning Representations (ICLR)*, 2019. 1

[4] Dominic Maggio, Hyungtae Lim, and Luca Carlone. VGGT-SLAM: Dense RGB SLAM Optimized on the SL (4) Manifold. In *Proc. of the Conf. on Neural Information Processing Systems (NeurIPS)*, 2025. 2

[5] Emanuele Palazzolo, Jens Behley, Philipp Lottes, Philippe Giguere, and Cyrill Stachniss. ReFusion: 3D Reconstruction in Dynamic Environments for RGB-D Cameras Exploiting Residuals. In *Proc. of the IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems (IROS)*, 2019. 1, 2

[6] Federico Perazzi, Jordi Pont-Tuset, Brian McWilliams, Luc Van Gool, Markus Gross, and Alexander Sorkine-Hornung. A Benchmark Dataset and Evaluation Methodology for Video Object Segmentation. In *Proc. of the IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2016. 4

[7] Jordi Pont-Tuset, Federico Perazzi, Sergi Caelles, Pablo Arbeláez, Alexander Sorkine-Hornung, and Luc Van Gool. The 2017 DAVIS Challenge on Video Object Segmentation. *arXiv preprint*, arXiv:1704.00675, 2017. 4

[8] Jianhao Zheng, Zihan Zhu, Valentin Bieri, Marc Pollefeys, Songyou Peng, and Armeni Iro. WildGS-SLAM: Monocular Gaussian Splatting SLAM in Dynamic Environments. In *Proc. of the IEEE/CVF Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2025. 1, 2