C4D: 4D Made from 3D through Dual Correspondences

Supplementary Material

1. More Visual Results

1.1. 4D Reconstruction

Given only a monocular video as input, C4D can reconstruct dynamic scenes along with camera parameters. Visual results are shown in Figure 1. To provide a comprehensive view of the 4D reconstruction, the static regions across all frames are retained, while the dynamic regions from uniformly sampled frames are also displayed.

1.2. Temporally Smooth Video Depth

In Figure 2, we compare the video depth estimation results of C4D with other DUSt3R-based methods, including DUSt3R [13], MASt3R [8], and MonST3R [15]. In addition to producing more accurate depth, C4D demonstrates superior temporal smoothness compared to other methods. To illustrate this, we highlight the y-t depth slices along the vertical red line in the red boxes, showing the depth variation over time. As observed, C4D achieves temporally smoother depth results, thanks to the *Point Trajectory Smoothness (PTS)* objective. In contrast, other methods exhibit zigzag artifacts in the y-t depth slices, indicating flickering artifacts in the video depth.

1.3. Motion Mask

One of the most critical aspects of reconstructing dynamic scenes is identifying dynamic regions, that is, predicting motion masks. In Figure 3, we provide a qualitative comparison of motion masks generated by our C4D and the concurrent work MonST3R on the Sintel dataset [1]. This dataset poses significant challenges due to its fast camera motion, large object motion, and motion blur.

In the first video, C4D demonstrates its ability to generate reliable motion masks on the Sintel dataset, even when a large portion of the frame content is dynamic. This success is attributed to our proposed correspondence-guided motion mask prediction strategy. In contrast, MonST3R struggles to recognize such dynamic regions under these challenging conditions. In the second video, C4D predicts more complete motion masks, whereas MonST3R only generates partial results. This improvement is due to C4D's consideration of multi-frame motion cues in our motion mask prediction strategy, which is crucial for practical scenarios.

2. More Experimental Results

2.1. Motion Segmentation Results

Unlike prompt-based video segmentation like SAM2 [10], motion segmentation aims to automatically segment the moving regions in the video. We evaluate our method on DAVIS 2016 [9] and compare it with some automatic motion segmentation methods in Tab. 1, where our approach outperforms both MonST3R [15] and the state-of-the-art supervised method, FlowP-SAM [14]. Note that the evaluation is conducted without Hungarian matching between predicted and ground-truth motion masks.

Method	Ours	MonST3R [15]	FlowP-SAM [14]
$IoU \uparrow$	47.89	31.57	42.23

Table 1. Motion segmentation results on DAVIS2016. Note that the evaluation is conducted without Hungarian matching between predicted and ground-truth motion masks.

2.2. Ablation on Different Tracker Variants

The tracker needs to predict additional mobility to infer the dynamic mask, which is a more difficult learning problem as it requires understanding spatial relationships. Table. 2 shows that a sole CNN or 3D-aware encoder struggles with multi-tasking, whereas combining both improves performance.

3. More Technical Details

3.1. Dynamic-aware Point Tracker (DynPT)

The ground truth used to supervise confidence is defined by an indicator that determines whether the predicted position lies within 12 pixels of the ground truth position. And since there are no currently available labels for mobility, we use the rich annotations provided by the Kubric [4] generator to generate ground-truth mobility labels. Specifically, the "positions" label, which describes the position of an object for each frame in world coordinates, is utilized.

As the movements of objects in Kubric are entirely rigid, we determine an object's mobility as follows: if the temporal difference in the "position" of an object exceeds a predefined threshold (e.g., 0.01), all the tracking points associated with that object are labeled as dynamic (i.e., mobility is labeled as True).

It is important to note that although an "is_dynamic" label is provided in Kubric, it only indicates whether the object is stationary on the floor (False) or being tossed (True) at the initial frame. However, some objects may collide and move in subsequent frames. In such cases, the "is_dynamic" label does not accurately represent the object's mobility, necessitating the use of our threshold-based approach.

We train DynPT on the training sets of the panning

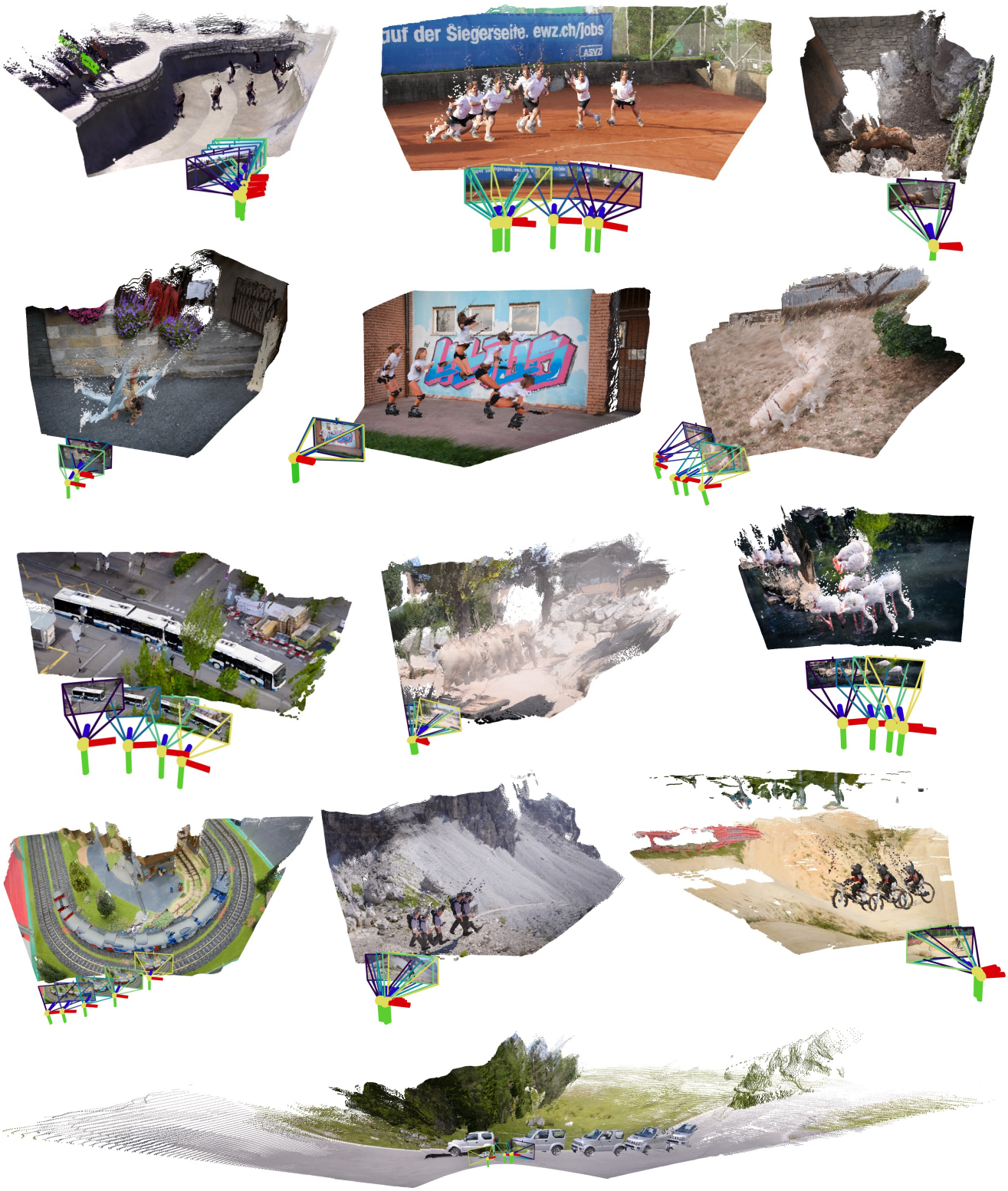

Figure 1. **Visual results of 4D reconstruction on DAVIS dataset [9].** C4D can reconstruct the dynamic scene and recover camera parameters from monocular video input.

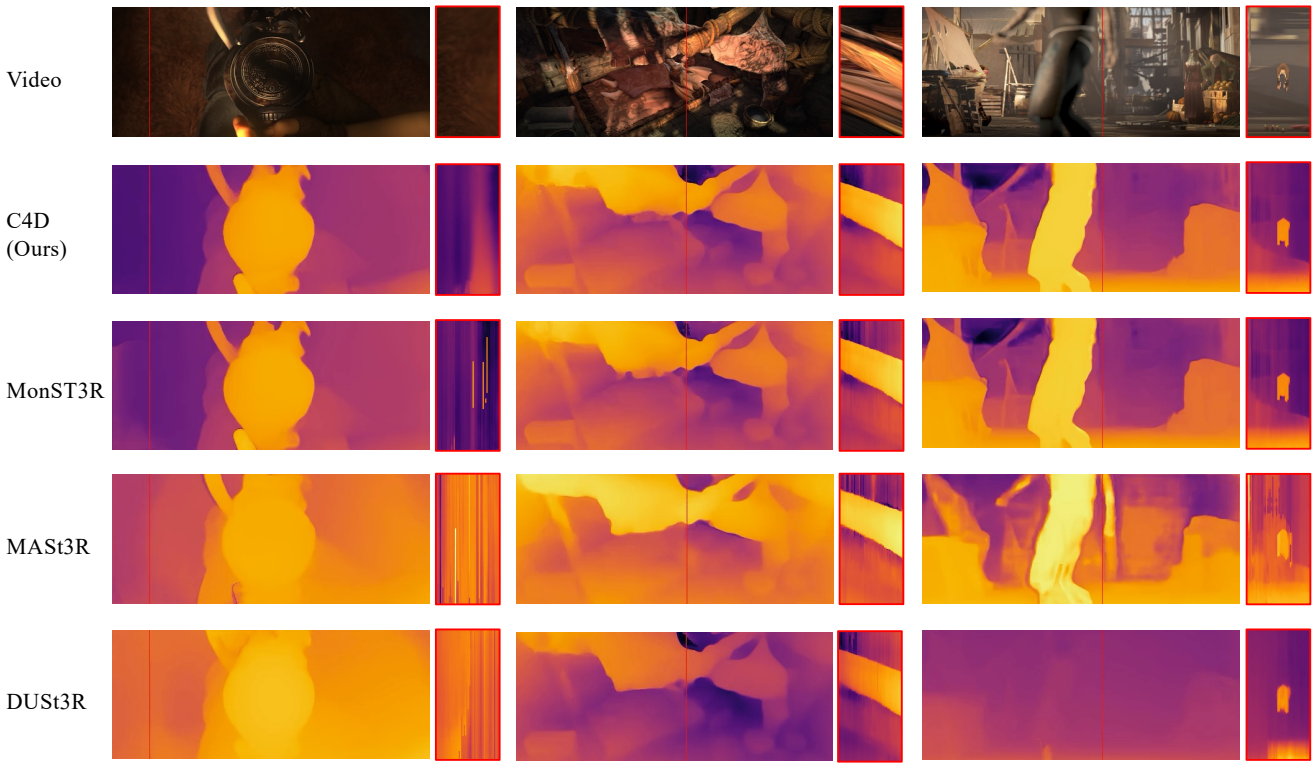

Figure 2. **Qualitative comparison of video depth on Sintel [1].** We compare C4D with MonST3R, MASt3R, and DUSt3R. To better visualize the temporal depth quality, we highlight y-t depth slices along the vertical red line within red boxes. For optimal viewing, please zoom in.

Method	MOVi-E	Pan. MOVi-E	MOVi-F	TAP-Vid DAVIS			TAP-Vid Kinetics		
	D-ACC ↑	D-ACC ↑	D-ACC ↑	AJ ↑	$< \delta^x_{\text{avg}}$ ↑	OA ↑	AJ ↑	$< \delta^x_{\text{avg}}$ ↑	OA ↑
DynPT	**87.9**	**94.1**	**91.5**	**61.6**	**75.4**	**87.4**	**47.8**	**62.6**	**82.3**
CE only	82.6	90.4	86.8	60.6	74.3	86.8	46.2	62.1	81.7
3E only	85.4	92.2	90.4	42.4	56.6	73.4	38.9	54.6	70.4

Table 2. Ablation study of different design choices for DynPT. "CE" denotes the use of a CNN encoder, while "3E" refers to the 3D-aware encoder.

MOVi-E and MOVi-F datasets. These datasets are chosen for their non-trivial camera movements and motion blur, which closely resemble real-world scenarios. For evaluation, in addition to the the panning MOVi-E and MOVi-F datasets, we also evaluate on the MOVi-E dataset to assess the generalization ability of DynPT.

During inference, DynPT processes videos by querying a sparse set of points in a sliding window manner to maintain computational efficiency as in [6, 7]. The query points are sampled based on grids: the image is divided into grids of 20×20 pixels, and one point with the maximum image gradient is sampled from each grid to capture the most distinguishable descriptor. Additionally, one random point is sampled from each grid to ensure diversity and prevent bias towards only high-gradient areas. This combination of gradient-based sampling and random sampling ensures a balanced selection of points, enabling robust and diverse feature extraction across the image.

3.2. Point Trajectory Smoothness (PTS) Objective

The primary goal of this objective is to ensure temporal smoothness in the per-frame pointmaps. Directly performing dense tracking for every pixel at every frame is computationally expensive. To address this, we propose an efficient strategy for generating dense, smoothed pointmaps. First, we track a sparse set of points and smooth their 3D trajectories using adaptive weighting (Sec 3.2.1). Next, we propagate the displacements resulting from the

Figure 3. **Qualitative comparison of motion mask on Sintel dataset [1].** We present the motion masks generated by C4D and MonST3R. Video frames and ground-truth motion masks are also included for reference. The white regions indicate dynamic areas.

smoothing process to their local neighbors through linear blending (Sec 3.2.2), ultimately producing dense, smoothed pointmaps.

This smoothing process is applied in a non-overlapping sliding window manner. For each local window, smoothing is performed on an extended window that includes additional frames on both ends. However, only the smoothed results within the original window are retained. This approach ensures both computational efficiency and temporal consistency.

3.2.1. Trajectory Smoothing with Adaptive Weighting

To enhance the smoothness of 3D trajectories while mitigating the influence of outliers, we employ a 1D convolution-based smoothing process with adaptive weights. This method ensures that trajectories are refined effectively without over-smoothing salient features. The core steps of the process are described below.

Trajectory Representation. The input trajectories are represented as a tensor $\mathbf{T} \in \mathbb{R}^{T\times N\times C}$, where T is the number of time frames, N is the number of tracked points, and C is the dimensionality of the coordinates (e.g., $C = 3$ for 3D trajectories).

Smoothing Kernel. A uniform smoothing kernel of size k is defined as:

$$\mathbf{K} = \frac{1}{k}\mathbf{1}_k, \tag{1}$$

where $\mathbf{1}_k$ is a vector of ones with length k, and we set it to 5. The kernel is normalized to ensure consistent averaging across the kernel window size.

Outlier-Aware Weighting. To reduce the influence of outliers, we compute a weight matrix $\mathbf{W} \in \mathbb{R}^{T \times N}$ based on the differences between consecutive trajectory points:

$$\Delta \mathbf{T}_t = \|\mathbf{T}_t - \mathbf{T}_{t-1}\|_2, \tag{2}$$

where $\Delta \mathbf{T}_t$ is the norm of the difference between consecutive points. The weights are then defined as:

$$\mathbf{W}_{t,n} = \exp\left(-\lambda \cdot \Delta \mathbf{T}_{t,n}\right), \tag{3}$$

where λ is a smoothing factor controlling the decay of weights for larger deviations and we set it to 1. To ensure temporal alignment, the weights are padded appropriately:

$$\mathbf{W}_t = \begin{cases} \mathbf{W}_1, & t = 1, \\ \mathbf{W}_{t-1}, & \text{otherwise.} \end{cases} \tag{4}$$

Weighted Convolution. To smooth the trajectories, we apply a weighted 1D convolution to each trajectory point:

$$\tilde{\mathbf{T}} = \frac{\text{Conv1D}(\mathbf{T} \odot \mathbf{W}, \mathbf{K})}{\text{Conv1D}(\mathbf{W}, \mathbf{K})}, \tag{5}$$

where $\odot$ denotes element-wise multiplication. The convolution is applied independently for each trajectory and coordinate dimension.

The output $\tilde{\mathbf{T}}$ is a smoothed trajectory tensor with the same shape as the input, ensuring that both global and local trajectory consistency are preserved.

3.2.2. Linear Blend Displacement (LBD) for Point Transformation

After obtaining the smoothed 3D trajectories of sparse tracking points, we leverage the observation that the displacements caused by the smoothing process are approximately consistent within local regions. Inspired by linear blend skinning, we treat the smoothed tracking points as control points. To transform all other 3D points based on the displacements of these control points, we employ a Linear Blend Displacement (LBD) approach. This method calculates proximity-weighted displacements for each point by considering its k nearest control points, ensuring smooth and locally influenced transformations. The detailed steps are described below.

Problem Formulation. Given a set of query points $\mathbf{X} \in \mathbb{R}^{P_1 \times 3}$, control points $\mathbf{C} \in \mathbb{R}^{P_2 \times 3}$, and control displacements $\mathbf{D} \in \mathbb{R}^{P_2 \times 3}$, the goal is to compute the transformed points $\tilde{\mathbf{X}} \in \mathbb{R}^{P_1 \times 3}$ using a weighted combination of the control displacements. Here, P_1 is the number of query points, and P_2 is the number of control points.

Nearest Neighbor Search. For each query point, we identify its k nearest control points using the L_2 distance. This yields:

$$\mathbf{d}_{j,k} = \|\mathbf{X}_j - \mathbf{C}_{\mathbf{I}_{j,k}}\|^2, \tag{6}$$

$$\mathbf{I}_{j,k} = \text{Indices of the } k \text{ nearest control points}, \tag{7}$$

where $\mathbf{d}_{j,k}$ is the squared distance between the j-th query point and the k-th nearest control point. We set k to 4 in our experiments.

Weight Computation. We compute proximity-based weights using inverse distance weighting:

$$w_{j,k} = \frac{1}{\mathbf{D}_{j,k}}, \tag{8}$$

The weights are normalized across the k nearest neighbors:

$$\hat{w}_{j,k} = \frac{w_{j,k}}{\sum_{k'} w_{j,k'}}. \tag{9}$$

Displacement Aggregation. Using the computed weights, the displacement for each query point is aggregated as linear blend of control displacements:

$$\mathbf{\Delta x}_j = \sum_k \hat{w}_{j,k} \mathbf{D}_{\mathbf{I}_{j,k}}, \tag{10}$$

where $\mathbf{d}_{j,k}$ is the displacement of the k-th nearest control point.

Point Transformation. Finally, the transformed query points are computed by adding the aggregated displacements:

$$\tilde{\mathbf{X}}_j = \mathbf{X}_j + \mathbf{\Delta x}_j. \tag{11}$$

3.3. Implementation Details

Training Details. We train DynPT for 50,000 steps with a total batch size of 32, starting from scratch except for the 3D-aware encoder, which is initialized from DUSt3R's pretrained encoder and kept frozen during training. The learning rate is set to 5e-4, and we use the AdamW optimizer with a OneCycle learning rate scheduler [11].

Inference Details. To ensure fast computation during motion mask calculation, we sample static points only from the latest sliding window of DynPT, as this window already includes the majority of points in the frame. The default tracking window size for DynPT is set to 16, with a stride of 4 frames. For the Point Trajectory Smoothness (PTS) objective, the default window size is 20 frames, extended by adding 5 additional frames on each end to ensure continuity and smoothness. And for longer videos, the window sizes for DynPT and PTS can be further extended to reduce computational costs.

Optimization Details. The correspondence-aided optimization is performed in two stages. In the first stage, we optimize using the global alignment (GA), camera movement alignment (CMA), and camera trajectory smoothness (CTS) objectives, with respective weights $w_{\text{GA}} = 1$, $w_{\text{CMA}} = 0.01$, and $w_{\text{CTS}} = 0.01$. During this stage, the optimization targets the depth maps $\hat{\mathbf{D}}$, camera poses $\hat{\mathbf{P}}$ and

camera intrinsics $\hat{\mathbf{K}}$. After completing the first stage, we fix the camera pose $\hat{\mathbf{P}}$ and intrinsics $\hat{\mathbf{K}}$ and proceed to optimize depth maps $\hat{\mathbf{D}}$ only in the second stage. We apply only the point trajectory smoothness (PTS) objective with weight $w_{\text{PTS}} = 1$ to further refine the depth maps $\hat{\mathbf{D}}$ in the second stage. Both stages are optimized for 300 iterations using the Adam optimizer with a learning rate of 0.01.

Datasets and Evaluation. Following [5, 15], we sample the first 90 frames with a temporal stride of 3 from the TUM-Dynamics [12] and ScanNet [2] datasets for computational efficiency. For dynamic accuracy evaluation, we use the validation sets of the MOVi-E, Panning MOVi-E, and MOVi-F datasets, comprising 250, 248, and 147 sequences, respectively. Each sequence contains 256 randomly sampled tracks spanning 24 frames. The resolution is fixed at 256×256, consistent with the TAPVid benchmark [3]. The evaluation metric used is accuracy, which assesses both dynamic (positive) and static (negative) states, defined as:

$$\text{D-ACC} = \frac{\text{TP} + \text{TN}}{\text{TP} + \text{TN} + \text{FP} + \text{FN}}, \tag{12}$$

where TP denotes true positives, TN denotes true negatives, FP denotes false positives, and FN denotes false negatives.

References

[1] Daniel J. Butler, Jonas Wulff, Garrett B. Stanley, and Michael J. Black. A naturalistic open source movie for optical flow evaluation. In *ECCV*, pages 611–625, 2012. 1, 3, 4

[2] Angela Dai, Angel X. Chang, Manolis Savva, Maciej Halber, Thomas Funkhouser, and Matthias Nießner. ScanNet: Richly-annotated 3D reconstructions of indoor scenes. In *CVPR*, pages 5828–5839, 2017. 6

[3] Carl Doersch, Ankush Gupta, Larisa Markeeva, Adria Recasens, Lucas Smaira, Yusuf Aytar, Joao Carreira, Andrew Zisserman, and Yi Yang. Tap-vid: A benchmark for tracking any point in a video. *Advances in Neural Information Processing Systems*, 35:13610–13626, 2022. 6

[4] Klaus Greff, Francois Belletti, Lucas Beyer, Carl Doersch, Yilun Du, Daniel Duckworth, David J Fleet, Dan Gnanapragasam, Florian Golemo, Charles Herrmann, et al. Kubric: A scalable dataset generator. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 3749–3761, 2022. 1

[5] Wenbo Hu, Xiangjun Gao, Xiaoyu Li, Sijie Zhao, Xiaodong Cun, Yong Zhang, Long Quan, and Ying Shan. DepthCrafter: Generating consistent long depth sequences for open-world videos. *arXiv preprint arXiv:2409.02095*, 2024. 6

[6] Nikita Karaev, Ignacio Rocco, Benjamin Graham, Natalia Neverova, Andrea Vedaldi, and Christian Rupprecht. Cotracker: It is better to track together. *arXiv preprint arXiv:2307.07635*, 2023. 3

[7] Nikita Karaev, Iurii Makarov, Jianyuan Wang, Natalia Neverova, Andrea Vedaldi, and Christian Rupprecht. Cotracker3: Simpler and better point tracking by pseudo-labelling real videos. *arXiv preprint arXiv:2410.11831*, 2024. 3

[8] Vincent Leroy, Yohann Cabon, and Jérôme Revaud. Grounding image matching in 3d with mast3r. *arXiv preprint arXiv:2406.09756*, 2024. 1

[9] 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 *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 724–732, 2016. 1, 2

[10] Nikhila Ravi, Valentin Gabeur, Yuan-Ting Hu, Ronghang Hu, Chaitanya Ryali, Tengyu Ma, Haitham Khedr, Roman Rädle, Chloe Rolland, Laura Gustafson, et al. Sam 2: Segment anything in images and videos. *arXiv preprint arXiv:2408.00714*, 2024. 1

[11] Leslie N Smith and Nicholay Topin. Super-convergence: Very fast training of neural networks using large learning rates. In *Artificial intelligence and machine learning for multi-domain operations applications*, pages 369–386. SPIE, 2019. 5

[12] Jürgen Sturm, Nikolas Engelhard, Felix Endres, Wolfram Burgard, and Daniel Cremers. A benchmark for the evaluation of RGB-D SLAM systems. pages 573–580, 2012. 6

[13] Shuzhe Wang, Vincent Leroy, Yohann Cabon, Boris Chidlovskii, and Jerome Revaud. Dust3r: Geometric 3d vision made easy. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 20697–20709, 2024. 1

[14] Junyu Xie, Charig Yang, Weidi Xie, and Andrew Zisserman. Moving object segmentation: All you need is sam (and flow). In *Proceedings of the Asian Conference on Computer Vision*, pages 162–178, 2024. 1

[15] Junyi Zhang, Charles Herrmann, Junhwa Hur, Varun Jampani, Trevor Darrell, Forrester Cole, Deqing Sun, and Ming-Hsuan Yang. Monst3r: A simple approach for estimating geometry in the presence of motion. *arXiv preprint arXiv:2410.03825*, 2024. 1, 6