

# MotionCrafter: Dense Geometry and Motion Reconstruction with a 4D VAE

## Supplementary Material

In the **supplementary video**, we provide additional visual results. In this **supplementary document**, we present further details and analyses to complement the main paper.

### A. Data Processing

For each video sequence, we preprocess the corresponding point maps and scene flow into a unified [world]-coordinate system, referenced by the first camera pose. The processing pipeline consists of three steps: (1) camera-pose normalization, (2) transformation of point maps and scene flow into the world coordinate frame, and (3) global normalization of world-space geometry and motion. Below, we detail each component.

#### A.1. Camera Pose Normalization

Monocular reconstruction systems often produce camera poses that contain arbitrary global rotation and translation. To eliminate this ambiguity, following DUST3R [97], we align all poses to a canonical coordinate frame defined by the first camera. In particular, given a sequence of camera poses  $\{P_i\}_{i=1}^N$ , where each  $P_i \in \mathbb{R}^{4 \times 4}$ , we decompose the first pose as

$$R_0 = P_0[:3, :3], \quad t_0 = P_0[:3, 3]. \quad (8)$$

Each pose is then normalized by

$$\tilde{R}_i = R_0^\top R_i, \quad \tilde{t}_i = R_0^\top (t_i - t_0), \quad (9)$$

which preserves the relative motion within the sequence while removing global rotation and translation.

#### A.2. Point Map Transformation

Given a point map  $X_i^C \in \mathbb{R}^3$  expressed in the camera coordinate system of frame  $i$ , we transform it into the first-frame coordinate system using the normalized camera poses:

$$X_i = \tilde{R}_i X_i^C + \tilde{t}_i. \quad (10)$$

This transformation is applied to all valid pixels, while invalid pixels (as indicated by the validity mask) are set to zero. For these invalid points, we use pyramid padding [124] to fill them in. Note that, we do not supervise these invalid points; filling them in is solely to prevent the VAE from being affected by missing values during feature extraction.

#### A.3. Scene Flow Transformation

The original scene flow  $V_i^C$  is defined in the camera coordinates of frame  $i$ . To obtain the first-frame (world) space

scene flow, we first compute the deformed points:

$$X_{i \rightarrow i+1}^C = X_i^C + V_i^C, \quad (11)$$

and transform them using the camera pose of the next frame:

$$X_{i \rightarrow i+1} = \tilde{R}_{i+1} X_{i \rightarrow i+1}^C + \tilde{t}_{i+1}. \quad (12)$$

The world-space scene flow is computed as

$$V_i = X_{i \rightarrow i+1} - X_i. \quad (13)$$

If a deformability mask is available, we apply it to zero out scene flow in non-dynamic regions.

#### A.4. Global World-Coordinate Normalization

To ensure consistent training across scenes with different scales, the global normalization is applied in the [world]-space geometry.

**Centering.** We first compute the centroid of all valid points:

$$\mu = \frac{1}{|\mathcal{D}|} \sum_{d \in \mathcal{D}} X_d. \quad (14)$$

**Isotropic Rescaling.** Instead of scaling by the maximum radius, we compute the mean scale of valid points to the centroid:

$$S = \frac{1}{\mathcal{D}} \sum_{d \in \mathcal{D}} \|X_d - \mu\|_2. \quad (15)$$

Then we uniformly normalize the point map, camera pose, and scene flow with the affine transformation:

$$X_i \leftarrow \frac{X_i - \mu}{S}, \quad \tilde{t}_i \leftarrow \frac{\tilde{t}_i - \mu}{S}, \quad V_i \leftarrow \frac{V_i}{S}. \quad (16)$$

This isotropic scaling preserves the geometric structure while normalizing the absolute scale across datasets. The normalization parameters  $(\mu, S)$  are stored for optional recovery of the original metric scale.

## B. Additional Ablations

### B.1. Ablation on the Multimodal Supervision

**Motivation.** While methods such as Geo4D [34] rely on multi-modality outputs (e.g., depth, point maps, and normals) together with a post-optimization fusion stage to obtain the final reconstruction, our main goal is to achieve *fully feed-forward* 4D geometry and motion reconstruction. Therefore, we intentionally avoid introducing any auxiliary outputs or post-refinement during inference. Interestingly,

Table 5. **Ablation study on Geometry VAE components.** Metrics are reported for **ScanNet**, **Sintel**, and **Monkaa** datasets: point accuracy ( $\text{Rel}^p \downarrow, \delta^p \uparrow$ ) and depth accuracy ( $\text{Rel}^d \downarrow, \delta^d \uparrow$ ).

Model	Training	Rescale	Depth Loss	ScanNet				Sintel				Monkaa			
				$\text{Rel}^p \downarrow$	$\delta^p \uparrow$	$\text{Rel}^d \downarrow$	$\delta^d \uparrow$	$\text{Rel}^p \downarrow$	$\delta^p \uparrow$	$\text{Rel}^d \downarrow$	$\delta^d \uparrow$	$\text{Rel}^p \downarrow$	$\delta^p \uparrow$	$\text{Rel}^d \downarrow$	$\delta^d \uparrow$
1	Original	Max	$\times$	14.96	86.95	1.98	99.90	39.96	62.63	12.62	92.65	23.78	67.33	4.30	98.91
2	From scratch	Mean	$\times$	7.01	99.29	1.88	99.92	10.40	93.89	42.44	92.38	8.02	96.91	4.10	98.47
3	From scratch	Max	$\times$	11.88	91.08	1.82	99.59	18.80	79.68	10.21	92.26	11.48	90.55	5.93	93.80
4	Finetune decoder	Max	$\times$	10.45	92.30	2.73	98.83	20.76	77.77	12.67	89.08	14.44	85.91	7.10	91.66
5	Finetune decoder	Max	$\checkmark$	4.46	98.20	<b>0.54</b>	99.97	12.04	88.28	<b>4.30</b>	<b>98.64</b>	8.50	93.36	<b>1.37</b>	99.64
6	Finetune all	Mean	$\times$	4.46	99.69	1.11	99.97	5.68	98.77	9.73	94.60	5.03	99.13	3.54	99.27
7	Finetune all	Mean	$\checkmark$	<b>3.03</b>	<b>99.88</b>	0.76	<b>99.99</b>	<b>4.39</b>	<b>99.14</b>	8.04	95.31	<b>3.74</b>	<b>99.47</b>	1.83	<b>99.77</b>

Table 6. **Ablation study on Unet components for Geometry Reconstruction.** We compare models trained with different strategies, rescaling methods, and decoder losses.

Model	Training Strategy	Normalization	Decoder Loss	GMU Kitchen		Monkaa		Sintel		DDAD	
				$\text{Rel}^p \downarrow$	$\delta^p \uparrow$	$\text{Rel}^p \downarrow$	$\delta^p \uparrow$	$\text{Rel}^p \downarrow$	$\delta^p \uparrow$	$\text{Rel}^p \downarrow$	$\delta^p \uparrow$
I	Finetuning VAE Decoder	Max	$\times$	17.49	80.65	33.66	56.42	40.47	49.42	41.66	39.45
II	Finetuning the whole VAE	Mean	$\times$	17.72	82.90	27.36	66.21	35.17	<b>58.08</b>	33.78	50.34
III	Finetuning the whole VAE	Mean	$\checkmark$	<b>14.47</b>	<b>90.54</b>	<b>25.33</b>	<b>71.77</b>	<b>33.39</b>	56.94	<b>22.39</b>	<b>70.42</b>

although multimodal outputs are not used at test time, we find that multimodal *supervision* during VAE training can still benefit the reconstruction quality of the 4D latent. In particular, we leverage depth as an additional supervision signal derived from the world-coordinate point maps.

**Depth supervision.** Given the reconstructed world-coordinate point map  $\hat{X}$  and the ground-truth point map  $X$ , we project both into the depth domain using the normalized camera pose  $\tilde{P}$ :

$$\hat{D} = \Pi(\hat{X}, \tilde{P}), \quad D = \Pi(X, \tilde{P}), \quad (17)$$

where  $\Pi(\cdot)$  denotes standard projection into the depth map. We apply two complementary losses:

- **Per-pixel L1 depth loss.** This loss encourages accurate depth prediction and is masked by the depth validity map:

$$\mathcal{L}_{\text{L1-D}} = \left\| (\hat{D} - D) \odot W \right\|_1, \quad (18)$$

where  $W$  is the binary valid-mask.

- **Multi-scale patch depth loss.** To improve geometric consistency across different spatial scales, we compute an L1 loss over patches defined by the scale factors  $\{4, 16, 64\}$ . For each scale, the depth maps are divided into non-overlapping patches; within each patch, the mean depth (computed with masked averaging) is subtracted to remove global bias:

$$\mathcal{L}_{\text{Patch-D}} = \sum_{s \in \{4, 16, 64\}} \left\| \left( \hat{D}^{(s)} - D^{(s)} \right) \odot W^{(s)} \right\|_1. \quad (19)$$

This term encourages consistent local geometric structure and suppresses depth-shift artifacts.

The total multimodal depth supervision is:

$$\mathcal{L}_{\text{depth}} = \lambda_{\text{L1-D}} \mathcal{L}_{\text{L1-D}} + \lambda_{\text{Patch-D}} \mathcal{L}_{\text{Patch-D}}. \quad (20)$$

**Results.** As shown in Tab. 5, introducing depth-based multimodal supervision significantly improves the reconstruction quality of the world-coordinate point maps (with 13.55% improvement in point map and 16.41% in depth map). Notably, this improvement is achieved *without* modifying the inference pipeline or introducing any extra modalities at test time. This ablation demonstrates that multimodal supervision is an effective strategy for enhancing the 4D latent representation learned by our VAE.

## B.2. Ablation on the Decoder Loss

**Motivation.** In the deterministic setting, the denoising process in conventional diffusion models can be viewed as collapsing from a multi-step procedure into a single step. Therefore, in addition to supervising the denoised latent representation, we introduce a *decoder loss* that directly supervises the VAE decoder’s output. Compared to latent regression, this supervision is more direct and provides stronger training signals to the UNet.

**Implementation.** During training, we do not update the VAE decoder’s weights. However, we still compute its gradients so that the loss can be back-propagated through the decoder to update the UNet parameters. To reduce memory consumption, we apply gradient checkpointing to the VAE decoder during this process.

**Results.** As shown in Tab. 6, incorporating the decoder loss consistently improves the UNet training. Across four unseen datasets, it yields an average improvement of 15.01%,

with particularly notable gains on the outdoor dataset DDAD, where the performance improves by 36.80%.

### B.3. Ablation on the training paradigm

**Motivation.** Following prior works [27, 111] in employing EDM [38] pre-conditioning, our framework supports both the *deterministic* and *denoising* diffusion paradigms, on top of the pretrained SVD model. Since 4D Reconstruction is a deterministic task, we use a deterministic paradigm by default, which has been widely explored and shown to be effective in previous dense prediction frameworks [80, 108, 111]. However, we also want to know exactly how different these two training paradigms are in our framework, especially for 4D latents that incorporate both geometry and motion information. Therefore, we conduct ablation experiments to verify this.

**Implementation.** As described in Section 3.3, we have already introduced the loss functions for two training paradigms. For a fair comparison, we do not use decoder loss in the deterministic paradigm. Specifically, we directly feed video latents into the U-Net to predict 4D latents. For the denoising paradigm, we first add noise to the 4D latents and channel-wise concatenate the video latents, then use the U-Net’s multi-step denoising to predict the 4D latents. All U-Net weights are initialized from the original SVD, with channel dimensions adjusted only at the first layer to accommodate different training paradigms.

**Results.** As shown in Tab. 7, the deterministic paradigm reduces  $\text{Rel}^p$  by about 12.4% and improves  $\delta^p$  by approximately 12.7% compared to the diffusion paradigms averaged across datasets. This result strongly demonstrates the effectiveness of the deterministic paradigm in dense prediction tasks. Also, this shows that prior knowledge of SVD can be inherited by the model without relying on a denoising mechanism.

Table 7. Ablation on different training paradigm.

Training Type	Monkaa		Sintel		DDAD	
	$\text{Rel}^p \downarrow$	$\delta^p \uparrow$	$\text{Rel}^p \downarrow$	$\delta^p \uparrow$	$\text{Rel}^p \downarrow$	$\delta^p \uparrow$
Diffusion	30.11	65.49	35.95	53.82	24.58	67.54
Deterministic	<b>25.88</b>	<b>74.01</b>	<b>32.46</b>	<b>63.14</b>	<b>21.27</b>	<b>72.82</b>

## C. Implementation Details

### C.1. Hyperparameter

We list the loss weights used for training the 4D VAE.

- Point Map Reconstruction Loss:  $\lambda_{\text{point}} = 1.0$
- Per-pixel L1 Depth loss:  $\lambda_{\text{L1-D}} = 1.0$
- Multi-Scale Depth Supervision:  $\lambda_{\text{Patch-D}} = 1.0$
- Normal Consistency Loss:  $\lambda_{\text{normal}} = 0.2$

Table 8. An overview of the training datasets. To balance the training, we sample the subset of some datasets.

Dataset	Domain	#Frames	#Videos
DynamicReplica [36]	Indoor/Outdoor	145K	1126
GTA-SfM [92]	Outdoor	19K	234
Kubric [19]	Indoor	137K	5736
MatrixCity [46]	Outdoor-Driving	452K	3029
MVS-Synth [28]	Outdoor-Driving	12K	120
Spring [62]	Outdoor	5K	49
Point Odyssey [129]	Indoor	18K	120
Synthia [75]	Outdoor-Driving	178K	1276
TartanAir [98]	Outdoor	306K	2245
VirtualKitti2 [7]	Driving	43K	320
BlinkVision [47]	Indoor/Outdoor	11K	72
OmniWorld [130]	Indoor/Outdoor	35K	350
Scannet++ [118]	Indoor-Real	310K	2078
Total	-	1.67M	16.8K

- Scene Flow Reconstruction Loss:  $\lambda_{\text{scene flow}} = 1.0$
- Scene Flow Regulation Loss:  $\lambda_{\text{reg}} = 0.01$

The pretrained video diffusion UNet is optimized with the following latent regression loss and decoder loss (optional):

- Latent Regression Loss:  $\lambda_{\text{latent}} = 1.0$
- Point Map Decoder Loss:  $\lambda_G = 1.0$
- Scene Flow Decoder Loss:  $\lambda_M = 1.0$

### C.2. Used Training Set

We list the used training datasets in Tab. 8, and provide some visual samples in Fig. 7.

### C.3. Model Information

Our system adopts the VAE and video UNet backbone from the stable video diffusion (SVD) [1] model. The pipeline consists of three major components: (1) a video VAE encoder for encoding per-frame latent representations, (2) a 4D VAE decoder for reconstructing geometry and motion fields from latent space, and (3) a 3D spatiotemporal UNet for latent denoising. We report the parameter counts of each component below:

- **Video UNet:** 1524.62M parameters. This large spatiotemporal UNet is responsible for denoising the latent representations over both space and time, enabling the modeling of dynamic geometry and motion.
- **Video VAE Encoder:** 34.16M parameters. This module processes each input frame independently and encodes it into a latent space with a spatial downsampling factor of  $8\times$ .
- **4D VAE Decoder:** 99.00M parameters. This decoder reconstructs 4D point maps and scene flow from the latent representation.
- **Total Parameters:** 1657.79M parameters.

**Inference Timing.** All timings are measured on a single GPU with 40 GB of memory. For a video clip of 25 frames at resolution  $320 \times 640$ , the average processing time per frame is as follows: 52.0 ms for VAE encoding, 13.4 ms for latent denoising, and 73.5 ms for VAE decoding, resulting in a total of 138.9 ms per frame. These measurements reflect the end-to-end processing required for a full forward pass of our geometry–motion reconstruction pipeline.

#### C.4. Evaluation Metrics

We provide detailed definitions of the evaluation metrics used for geometry and motion reconstruction.

**Geometry Alignment.** Since monocular reconstruction is defined up to scale ambiguity, the predicted world-space point map  $\hat{\mathbf{X}}_i$  is aligned to the ground truth  $\mathbf{X}_i$  using a per-sequence scale  $s$  and shift  $\mathbf{t}$ :

$$\tilde{\mathbf{X}}_i = s\hat{\mathbf{X}}_i + \mathbf{t}, \quad (21)$$

where  $s$  and  $\mathbf{t}$  are optimized by minimizing:

$$\min_{s, \mathbf{t}} \sum_i \left\| s\hat{\mathbf{X}}_i + \mathbf{t} - \mathbf{X}_i \right\|_2^2. \quad (22)$$

**Relative Point Error (Rel<sup>p</sup>).** We measure the relative geometry error as:

$$\text{Rel}^p = \frac{1}{N} \sum_i \frac{\left\| \tilde{\mathbf{X}}_i - \mathbf{X}_i \right\|_2}{\left\| \mathbf{X}_i \right\|_2}. \quad (23)$$

**Inlier Ratio ( $\delta^p$ ).** We compute the percentage of points whose relative error is below a threshold  $\tau$  (0.25 in our experiments):

$$\delta^p = \frac{1}{N} \sum_i \mathbf{1} \left( \frac{\left\| \tilde{\mathbf{X}}_i - \mathbf{X}_i \right\|_2}{\left\| \mathbf{X}_i \right\|_2} < \tau \right). \quad (24)$$

**Scene Flow Alignment.** The predicted scene flow  $\hat{\mathbf{V}}_i$  is scaled using the same geometry scale  $s$ :

$$\tilde{\mathbf{V}}_i = s\hat{\mathbf{V}}_i. \quad (25)$$

**End-Point Error (EPE).** We compute the average endpoint error between predicted and ground-truth scene flow:

$$\text{EPE} = \frac{1}{N} \sum_i \left\| \tilde{\mathbf{V}}_i - \mathbf{V}_i \right\|_2. \quad (26)$$

**Average Percent of Points within Delta (APD).** APD measures the percentage of scene flow vectors whose error is below a threshold  $\gamma$ :

$$\text{APD}_\gamma = \frac{1}{N} \sum_i \mathbf{1} \left( \left\| \tilde{\mathbf{V}}_i - \mathbf{V}_i \right\|_2 < \gamma \right). \quad (27)$$

#### D. More Visualization Results

We select some in-the-wild videos from the Davis [67] dataset as samples for zero-shot testing, and the results are shown in Fig. 8. A more intuitive visualization is provided in the attached video demo. We also provide more qualitative comparisons with other methods, as shown in Figs. 9 to 11. The comparisons are for two different tasks: 1) joint geometry and motion estimation, and 2) geometry reconstruction only.



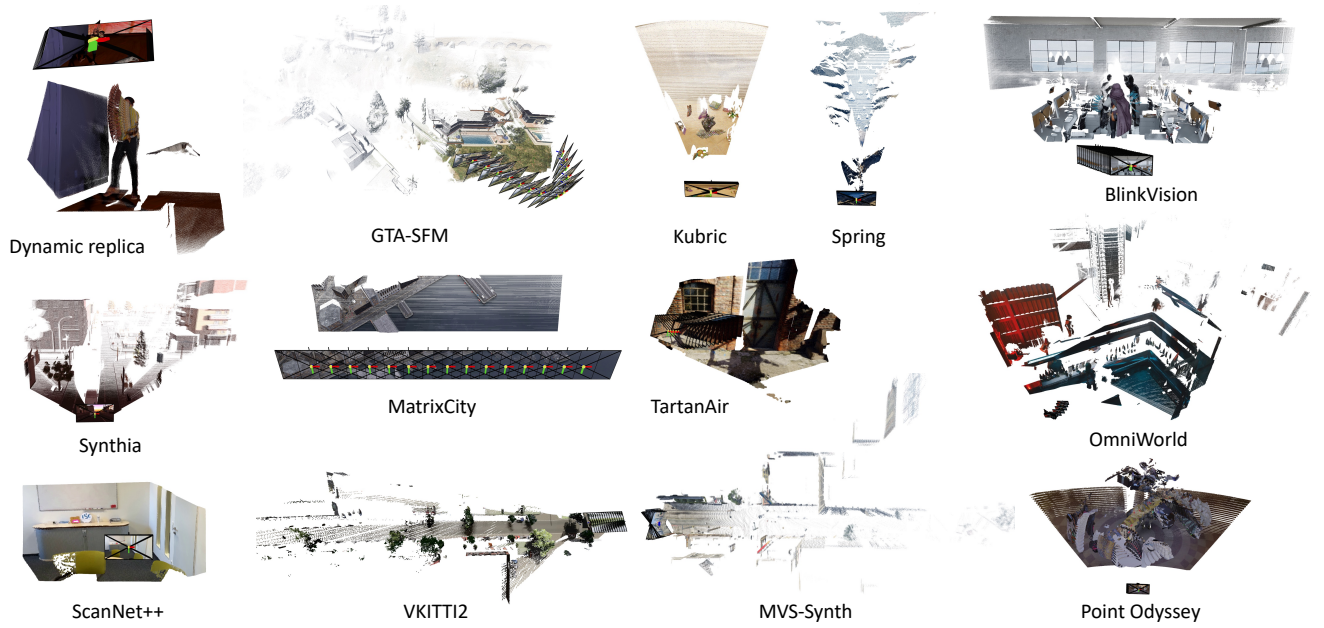


Figure 7. **The examples of our training set.** We randomly sample video frames from these datasets. In geometric training, we set a random stride to sample the video at different intervals. In motion training, we always keep the stride at 1 to continuously sample frames.



Figure 8. **Zero-shot results on Davis [67] dataset.** Despite the very limited number of samples used for training scene flow estimation, our method generalizes well across different scene types. Thanks to our end-to-end model design and unified definitions of geometry and motion in the world coordinate system, all results are directly output by the model without any post-optimization. See the video visualization for a more intuitive understanding of the dynamics.

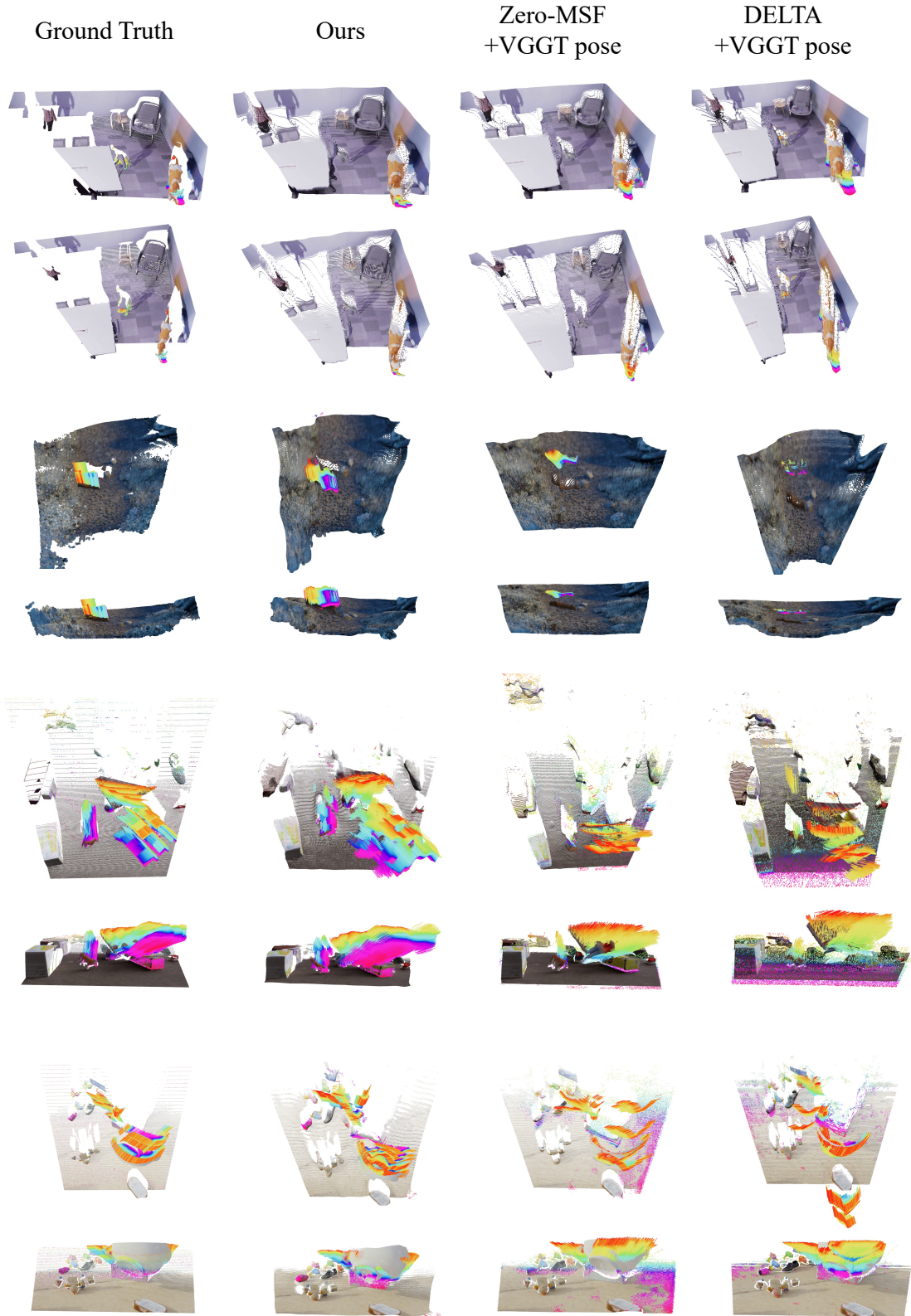


Figure 9. **Qualitative comparison with the state-of-the-art methods Zero-MSF [54] and DELTA [65].** In the first case, our method demonstrates scene flow estimation accuracy comparable to Zero-MSF, even without training on the dynamic replica dataset like it. In the other cases, our method significantly outperforms existing methods in both geometric structure and motion pattern estimation.



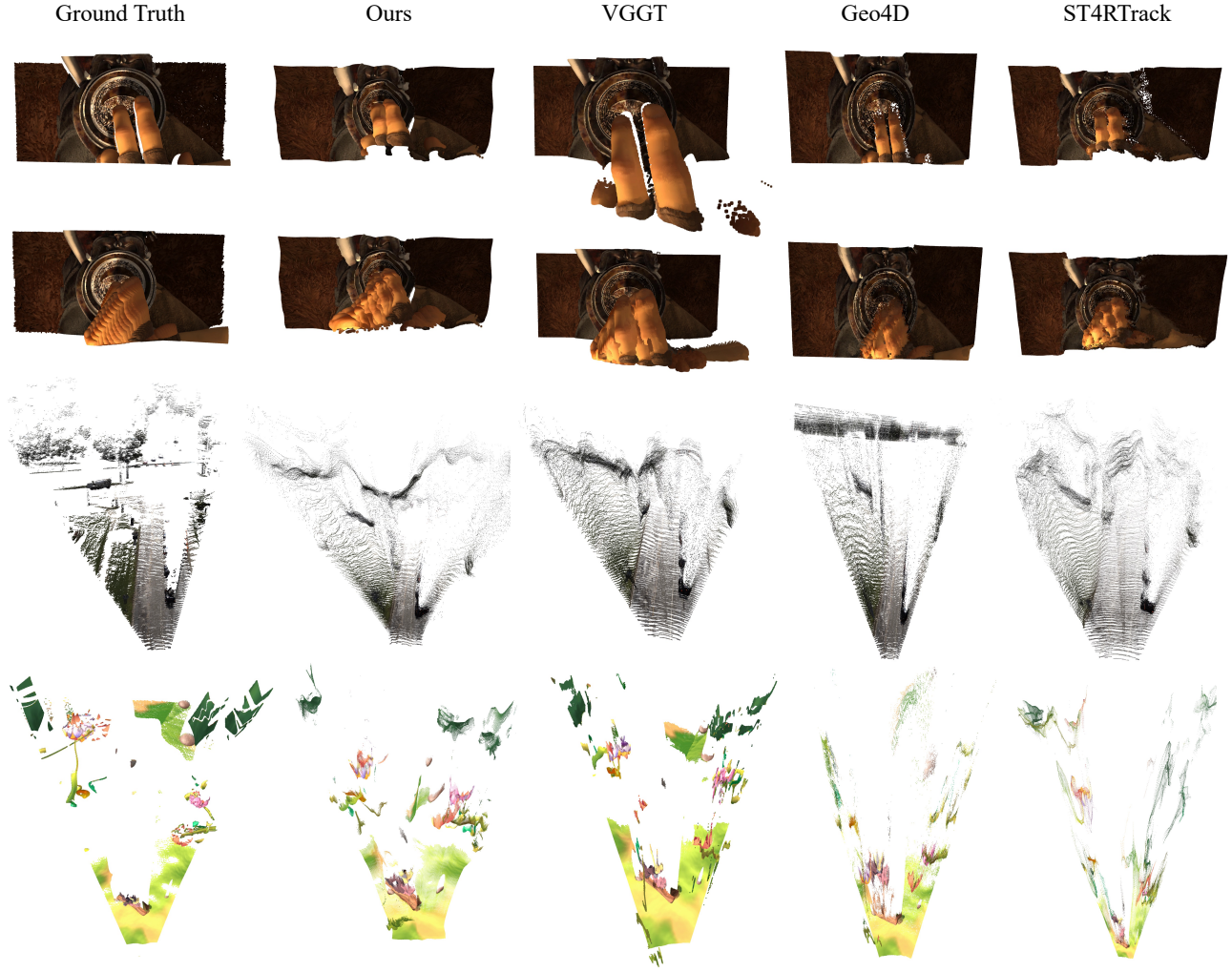


Figure 10. **Qualitative geometric comparison with VGGT [91], Geo4D [34], and ST4RTrack [14].** For moving objects, such as the finger in the first case, our method estimates more accurate scale and motion changes. For outdoor scenes, our method estimates a more accurate scene structure. Notably, our method, like VGGT, can directly output point clouds in world coordinates without requiring post-optimization steps such as Geo4D. Furthermore, our method has a much smaller training scale than VGGT, yet exhibits good robustness in dynamic scenes. We attribute this to pre-training knowledge of video diffusion and our proposed training strategy.



Figure 11. **Qualitative geometric comparison with ST4RTrack [14] on zero-shot generalization.** Compared with ST4RTrack, our results show better multi-view consistency, smoother Geometry, and fewer stray spots.