

# Robust Multi-Object 4D Generation for In-the-wild Videos

In this supplementary material, we provide additional insights and details across several sections. Section 1 delves deeper into the annotated MOSE-PTS dataset for point tracking, outlining the annotation procedure and presenting a difficulty analysis. Section 2 expands on the implementation and optimization process of our method. In Section 3, we include additional visual comparisons of point motion and analyze failure cases of our approach. Lastly, Section 4 offers further evaluation details for GenMOJO, with a particular focus on the design and execution of our user study experiment.

## 1. More Details on MOSE-PTS

### 1.1. Annotation Procedure

We annotate point tracks for 20 videos selected from MOSE [3], which is a complex and in-the-wild video dataset designed for video object segmentation with substantial real-world occlusions. Inspired by the TAP dataset [4], we aim for generality by allowing annotators to choose any object and any point they consider important, rather than specifying a closed-world set of points to annotate.

Given a video, annotation proceeds in two stages, depicted in Figure 1. First, annotators choose objects, especially moving ones, without regard to their difficulty in tracking. Next, they choose points on each selected object and track them. Finally, we review and mark low-quality annotations for correction, repeating this correction procedure as many times as needed. All annotators provided informed consent before completing tasks and were reimbursed for their time.

**Stage 1: Object Selection.** For object selection, we began with the MOSE [3] video mask annotations, initially identifying objects from these masks that predominantly captured moving elements within the scene. We then refined this selection by asking annotators to exclude overly simple objects, such as small, stationary, or occluded elements that contribute minimally to the scene’s complexity. To enhance the dataset’s challenge for point tracking, we included additional moving objects by human eyes, prioritizing those that experience partial occlusion during movement. This process ensured that the dataset maintained a

higher level of complexity, better suited for evaluating robust point-tracking algorithms.

**Stage 2: Point Annotation.** For each object selected in the initial stage, annotators identified a set of five key points on the first frame, including prominent features such as eyes for animals and key poses for humans. To streamline the annotation process, we used CoTracker2 [5] for point initialization across frames. Annotators then refined these points every alternate frame to ensure precise point tracking consistency with preceding frames, adjusting for minor shifts. In cases of occlusion, annotators marked frames where points were no longer visible, preserving accurate tracking continuity throughout the sequence.

### 1.2. Point Track Difficulty Analysis

To provide some insights into the difficulty of the annotated point tracks in MOSE-PTS, we run CoTrackerV2 [5] and CoTrackerV3 [6] on MOSE-PTS and report the Occlusion Accuracy (OA; accuracy of occlusion prediction), Average Delta ( $\delta$ , fraction of visible points tracked within 1, 2, 4, 8, and 16 pixels, averaged over the 5 threshold values), and Average Jaccard (AJ, a combination of tracking accuracy and occlusion prediction accuracy) [4]. For the evaluation on TAP-Vid subsets, we report the values reported in the original paper.

We find that in Table 1 that both versions of CoTracker see up to 10 points of performance drop on MOSE-PTS across all metrics. Even the most recent CoTrackerV3, which has been trained on extra point annotations extracted from real world data, see a noticeable drop in performance on MOSE-PTS. This shows that MOSE-PTS is even more challenging than the videos presented in TAP-Vid-Kinetics [4]. We visualized our point track annotations in the [attached HTML page](#).

## 2. Implementation Details

In this section, we provide some extra details on implementation and optimization, as well as the hyper-parameter choices for our experiments. We use the same set of hyper-parameters for all DAVIS and MOSE videos in our experiments.

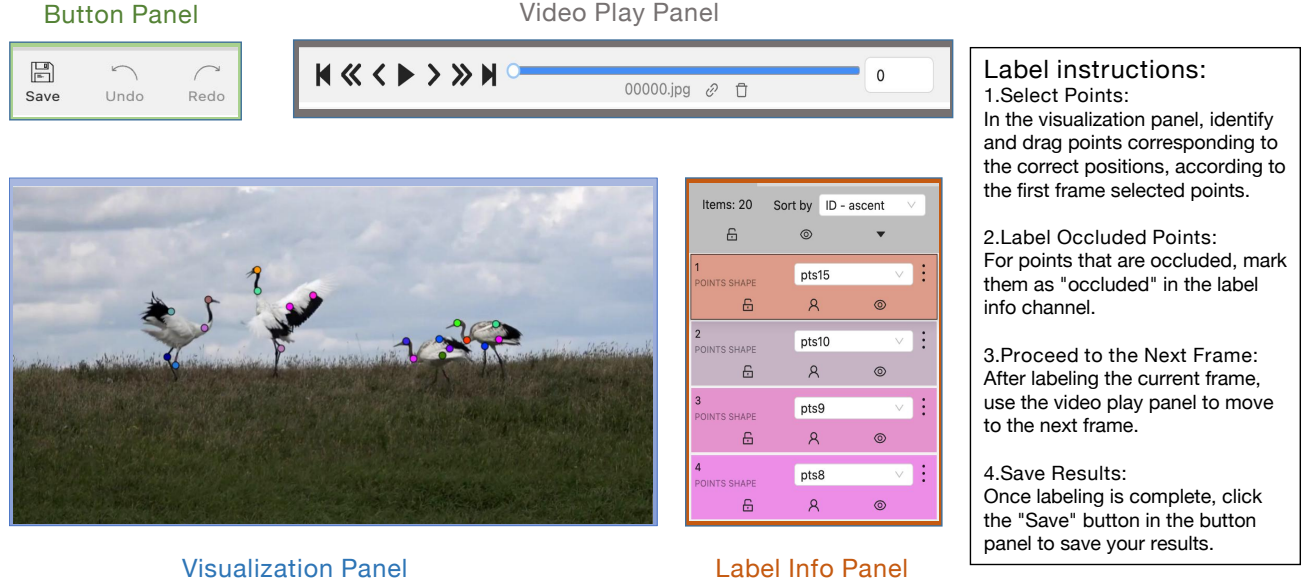


Figure 1. **The point track annotation interface** comprises four key components: (1) a visualization panel for reviewing and annotating points, (2) a button panel for interaction, (3) an information panel displaying relevant details, and (4) a video playback panel for navigating through frames. We also provide a simplified version of the annotation instructions to the right.

Table 1. **Point tracking performance of CoTrackerV2 and CoTrackerV3 on various datasets.** The performance of CoTracker degrades by up to 10 points in MOSE-PTS, suggesting that it is a much harder dataset compared to the subsets present in TAP-Vid [4].

Dataset	CoTrackerV2 [5]			CoTrackerV3 [6]		
	AJ	$\delta$	OA	AJ	$\delta$	OA
RGB-Stacking	67.4	78.9	85.2	71.7	83.6	91.1
DAVIS	61.8	76.1	88.3	63.8	76.3	90.2
Kinetics	49.6	64.3	83.3	55.8	68.5	88.3
MOSE-PTS	<b>40.4</b>	<b>56.1</b>	<b>75.5</b>	<b>46.7</b>	<b>63.6</b>	<b>77.9</b>

## 2.1. Static Gaussian Optimization

**Pruning and Densification.** Following [2, 7, 8], we prune Gaussians with an opacity smaller than 0.01 or a scale larger than 0.05. The densification is applied for Gaussians with an accumulated gradient larger than 0.5 and max scaling smaller than 0.05. Both pruning and densification are performed every 100 optimization steps if the total number of Gaussians is smaller than 20000. We impose this limit on the number of Gaussians due to memory constraints. Unlike traditional 3D Gaussian Splatting works, we do not reset the opacity values during optimization, which yields better results.

**Hyperparameters.** For static Gaussian optimization, we use the same set of hyperparameters from previous works [2, 7, 8] and use a learning rate that decays from  $1e^{-3}$  to  $2e^{-5}$  for the position, a static learning rate of 0.01 for the spherical harmonics, 0.05 for the opacity, and  $5e^{-3}$  for the scale and rotation. We optimize for a fixed number of 1000 steps for all objects in the video using a batch size of 16 when calculating the SDS loss.

**Running Time.** On the A6000 GPU, the static 3D-lifting process takes around 5.63 minutes for the 1000 steps, comparable to what was reported in previous works.

## 2.2. Dynamic Gaussian Optimization

**Deformation Network.** We follow previous works [2, 7] and use a Hexplane [1] backbone representation with a 2-layer MLP head on top to predict the required outputs. We set the resolution of the Hexplanes to  $[64, 64, 64, 0.8T]$  for  $(x, y, z, t)$  dimensions, where  $T$  is the number of frames in the input video sequence. This is the same setting used in the baselines, and we keep this setting to ensure fair comparisons.

**Hyperparameters.** The learning rate of the Hexplane is set to  $6.4e^{-4}$ , and the learning rate of the MLP prediction heads is set to  $6.4e^{-3}$  [2, 7]. We optimize for a  $35 \cdot T$  steps with a batch size of 8, where  $T$  is the number of frames in the video. Like previous works, we sample 4 novel views

Table 2. **Additional Video to 4D Scene Generation Ablations.** We augment DreamScene4D with DepthCrafter (+DC) and disallow color changes (-SH).

Method	MOSE			DAVIS		
	CLIP $\uparrow$	PSNR $\uparrow$	LPIPS $\downarrow$	CLIP $\uparrow$	PSNR $\uparrow$	LPIPS $\downarrow$
DreamScene4D	85.16	22.98	0.169	84.13	21.73	0.163
DreamScene4D (+DC)	85.22	23.01	0.169	84.15	22.07	0.163
GenMOJO (-SH)	85.37	24.25	0.168	84.16	23.17	0.164
GenMOJO (Full)	<b>85.41</b>	<b>25.56</b>	<b>0.168</b>	<b>84.17</b>	<b>23.51</b>	0.163

per frame in the batch to calculate the SDS loss. We use the AdamW optimizer to optimize the deformation parameters.

**Running Time.** The running time is heavily dependent on the length of the video, as the number of optimization steps is dependent on the total number of frames, as well as the number of objects. For reference, on the A6000 GPU, a video with 32 frames with 3 objects takes around 74 minutes. DreamScene4D [2] is roughly 10% slower as the optimization process needs to be done independently, but GenMOJO requires 15% more VRAM on a GPU due to the joint splitting procedure requiring Gaussians of all objects to be kept in memory.

### 3. Additional Results

#### 3.1. Point Motion Visualizations

We visualize some comparisons of the point tracking between Shape of Motion [9], GenMOJO, and CoTrackerV3 [6] in Figure 2. We can see the GenMOJO produces more accurate point tracks compared to other optimization-based methods like Shape of Motion and DreamScene4D. CoTrackerV3 produces very accurate point tracks when it succeeds, but also produces tremendous errors when it fails, as the error can be unbounded.

#### 3.2. Additional Ablations

We conduct additional evaluations by replacing DepthAnything in DreamScene4D with DepthCrafter. Furthermore, we also ablate the effects of enabling and disabling SH coefficient changes on novel view synthesis. As shown in Table 2, it leads to a minor improvement, as DepthCrafter sometimes provides better depth priors. However, without joint object optimization, DreamScene4D cannot fully recover from erroneous depth estimates.

We also include a visualization of the Gaussian trajectories with and without allowing the colors of the Gaussians to change over time in Figure 3.

#### 3.3. Failure Cases

We visualize some failure cases corresponding to the limitations section in the main text in Figure 4. We envision that as better-performing foundational models for novel-view

synthesis and video depth estimators are released, GenMOJO will also become more robust.

## 4. Evaluation Details

### 4.1. Novel View Synthesis

We first recenter the Gaussians based on the median  $(x, y, z)$ , then render from the following combination of (elevation, azimuth) angles:  $(0, 30)$ ,  $(0, -30)$ ,  $(30, 0)$ ,  $(-30, 0)$ , where  $(0, 0)$  corresponds to the front view. These novel view renders are then compared with the reference view at each timestep to obtain the CLIP and LPIPS scores. We then average the scores to obtain the final score.

### 4.2. User Study

For the user study, we consider the videos in MOSE-PTS. For each method, we render from the reference view, as well as 2 novel views corresponding to the input video. We use Amazon Turk to outsource evaluations and ask the workers to compare the 2 sets of 3 videos with the input video. Each set of videos is reviewed by 30 workers with a HIT rate of over 95% for a total of 900 answers collected. The whole user preference study takes about 89s per question and 22.25 working hours in total.

To filter out poor quality responses, we intentionally included “dummy” questions where one set of renders was replaced by renders from non-converged optimization where the visual quality is noticeably worse, and used these “dummy” questions to filter out workers who submitted responses that did not pass these questions. We additionally filtered out workers who submitted the same answer for all the videos. For every worker whose answers were rejected, we assigned new ones until the desired number of answers had been collected.

We include the full instructions shown to the workers below:

*Please read the instructions and check the videos carefully.*

*Please take a look at the reference video on the top. There are 2 sets of rendered videos (A and B) for this reference video: one rendered from the original viewpoint, and two rendered from other viewpoints. Please choose the set of videos that looks more realistic and better represents the original video to you. The options (A) and (B) correspond to the two sets of videos, as denoted in the captions under the videos.*

*To judge the quality of the videos, consider the following points, listed in order of importance:*

1. *Do the objects in the rendered videos correspond to the reference video?*





Figure 2. **Point Tracking Comparisons.** In complex MOSE videos, Shape of Motion [9] and CoTracker [6] point tracks can drift off the object of interest, while GenMOJO’s point tracks are anchored to the object thanks to the object priors.

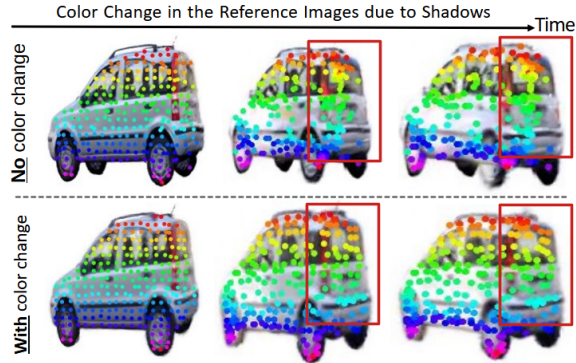


Figure 3. 4D Gaussians in red boxes do not maintain on the rear of the car under **temporal lighting changes** with fixed Gaussian colors (top row).

*Please ignore the background in the original video.*

A GUI sample of a survey question is also provided in Figure 5 for reference.

### 4.3. Multi-Object DAVIS Split

We list the multi-object DAVIS video names that were used to perform evaluations:

boxing-fisheye, car-shadow, crossing,  
dog-geoses, dogs-jump, gold-fish,  
horsejump-high, india, libby, judo,  
loading, pigs, schoolgirls,  
scooter-black, soapbox

2. Do the objects clip or penetrate each other in the rendered videos?
3. Does the video look geometrically correct (e.g. not overly flat) when observed from other viewpoints?
4. Are there any visual artifacts (e.g. floaters, weird textures) in the rendered videos?

*Please start from the first criteria to select the better set of renderings. If they are equal, please use the next criteria. If they are equal for all 4 points, please select Equally Preferred.*

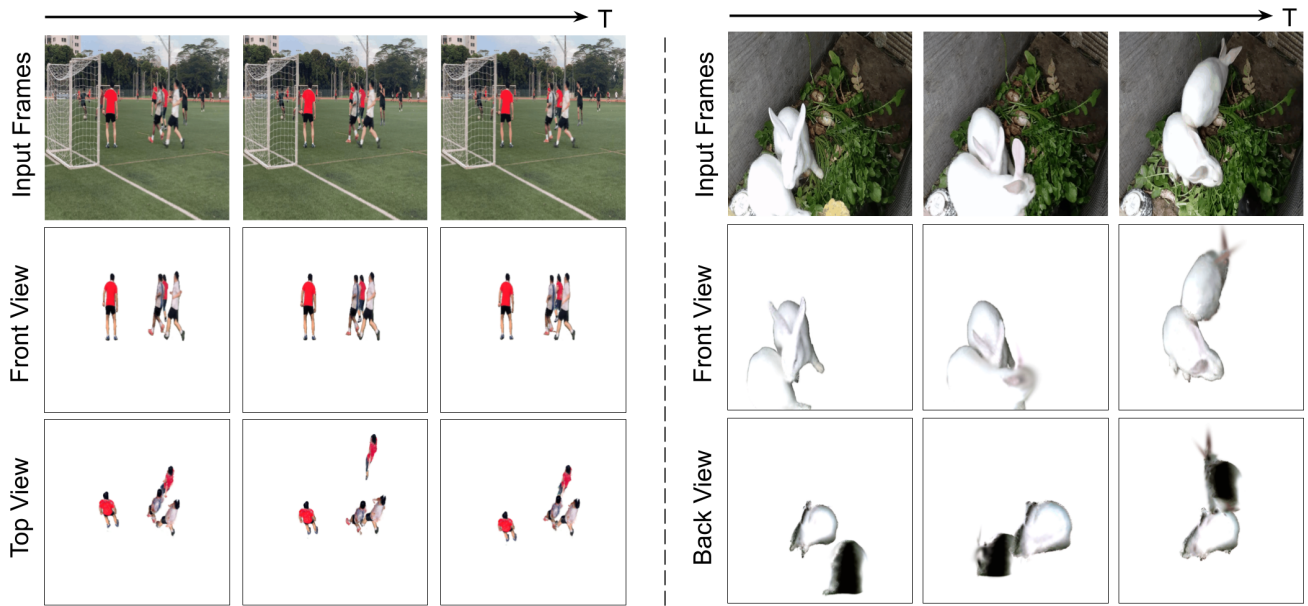


Figure 4. **Failure cases.** Top: Erroneous depth predictions can cause entities to jitter along the depth dimension. Bottom: View synthesis diffusion model failures result in degenerate textures in unseen viewpoints.

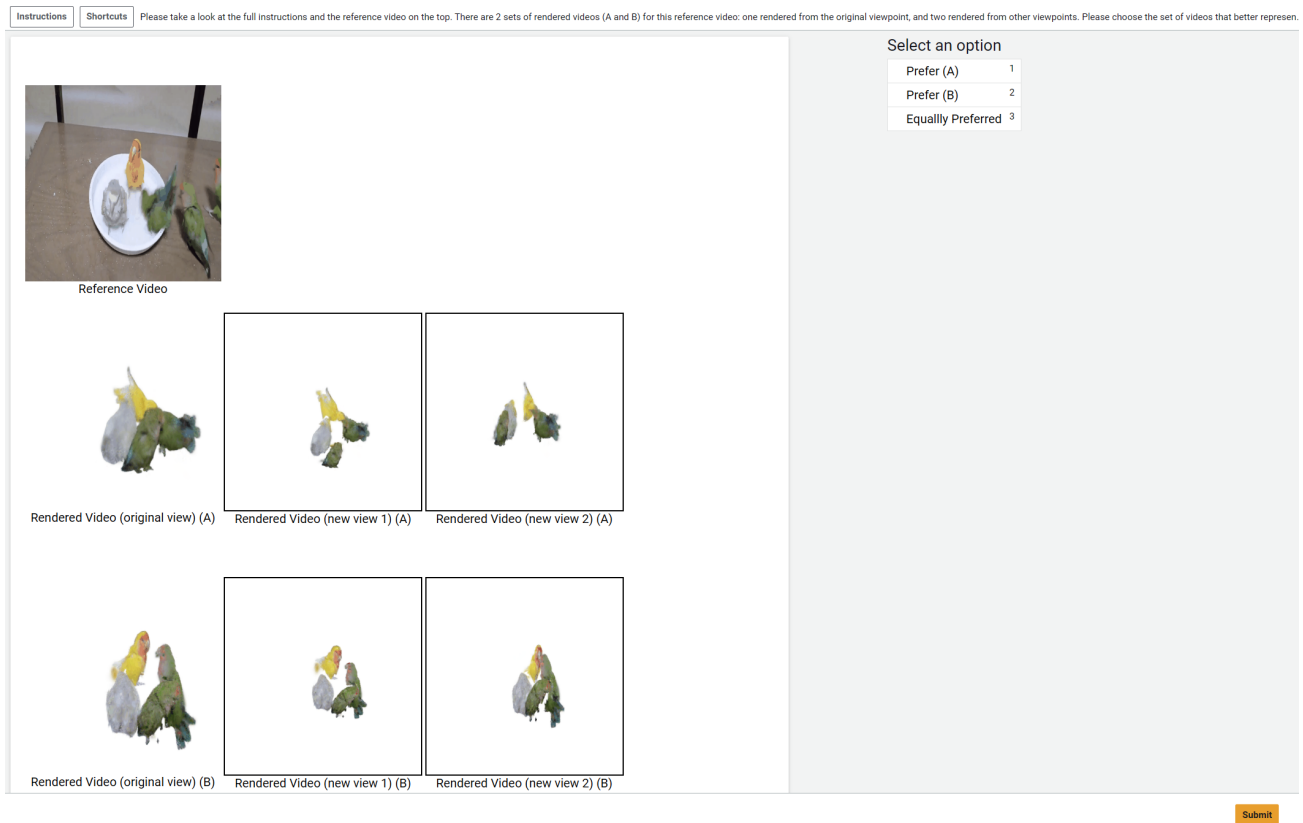


Figure 5. **User survey interface.** A GUI sample of what an Amazon Turk worker sees for the user study.

## References

- [1] Ang Cao and Justin Johnson. Hexplane: A fast representation for dynamic scenes. In *CVPR*, 2023. [2](#)
- [2] Wen-Hsuan Chu, Lei Ke, and Katerina Fragkiadaki. Dream-scene4d: Dynamic multi-object scene generation from monocular videos. In *NeurIPS*, 2024. [2](#), [3](#)
- [3] Henghui Ding, Chang Liu, Shuting He, Xudong Jiang, Philip HS Torr, and Song Bai. Mose: A new dataset for video object segmentation in complex scenes. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 20224–20234, 2023. [1](#)
- [4] 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. In *NeurIPS*, 2022. [1](#), [2](#)
- [5] Nikita Karaev, Ignacio Rocco, Benjamin Graham, Natalia Neverova, Andrea Vedaldi, and Christian Rupprecht. Co-tracker: It is better to track together. *arXiv preprint arXiv:2307.07635*, 2023. [1](#), [2](#)
- [6] Nikita Karaev, Iurii Makarov, Jianyuan Wang, Natalia Neverova, Andrea Vedaldi, and Christian Rupprecht. Co-tracker3: Simpler and better point tracking by pseudo-labelling real videos. *arXiv preprint arXiv:2410.11831*, 2024. [1](#), [2](#), [3](#), [4](#)
- [7] Jiawei Ren, Liang Pan, Jiaxiang Tang, Chi Zhang, Ang Cao, Gang Zeng, and Ziwei Liu. Dreamgaussian4d: Generative 4d gaussian splatting. *arXiv preprint arXiv:2312.17142*, 2023. [2](#)
- [8] Jiaxiang Tang, Jiawei Ren, Hang Zhou, Ziwei Liu, and Gang Zeng. Dreamgaussian: Generative gaussian splatting for efficient 3d content creation. *ICLR*, 2024. [2](#)
- [9] Qianqian Wang, Vickie Ye, Hang Gao, Jake Austin, Zhengqi Li, and Angjoo Kanazawa. Shape of motion: 4d reconstruction from a single video. *arXiv preprint arXiv:2407.13764*, 2024. [3](#), [4](#)