

DGS-LRM: Real-Time Deformable 3D Gaussian Reconstruction From Monocular Videos

Chieh Hubert Lin^{1,2} Zhaoyang Lv¹ Songyin Wu^{1,3} Zhen Xu¹ Thu Nguyen-Phuoc¹
Hung-Yu Tseng¹ Julian Straub¹ Numair Khan¹ Lei Xiao¹ Ming-Hsuan Yang^{1,2}
Yuheng Ren¹ Richard Newcombe¹ Zhao Dong¹ Zhengqin Li¹

¹Meta ²UC Merced ³UC Santa Barbara

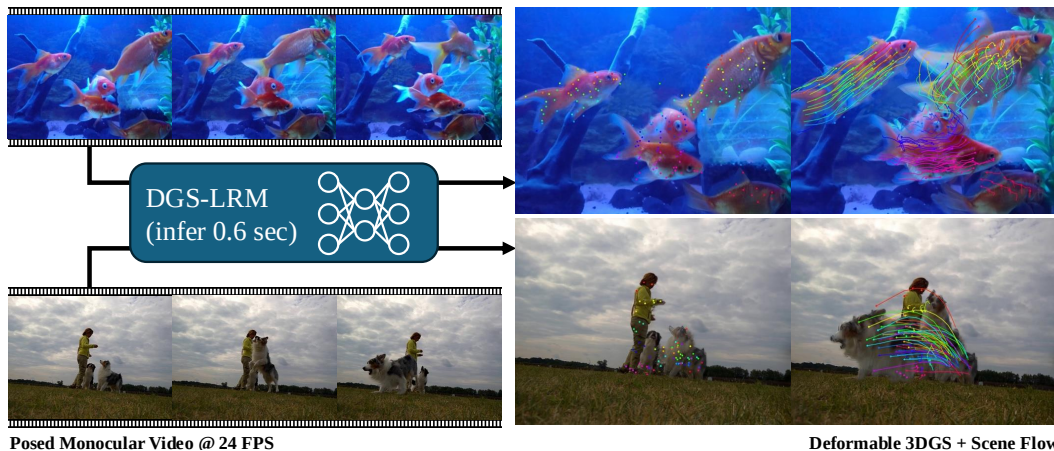


Figure 1: Our proposed Deformable Gaussian Splats Large Reconstruction Model (DGS-LRM) takes posed monocular videos as input and predicts deformable 3D Gaussians in a single feedforward pass. Our approach enables both novel view synthesis and 2D/3D tracking, as illustrated on the right.

Abstract

We introduce the Deformable Gaussian Splats Large Reconstruction Model (DGS-LRM), the first feed-forward method predicting deformable 3D Gaussian splats from a monocular posed video of any dynamic scene. Feed-forward scene reconstruction has gained significant attention for its ability to rapidly create digital replicas of real-world environments. However, most existing models are limited to static scenes and fail to reconstruct the motion of moving objects. Developing a feed-forward model for dynamic scene reconstruction poses significant challenges, including the scarcity of training data and the need for appropriate 3D representations and training paradigms. To address these challenges, we introduce several key technical contributions: an enhanced large-scale synthetic dataset with ground-truth multi-view videos and dense 3D scene flow supervision; a per-pixel deformable 3D Gaussian representation that is easy to learn, supports high-quality dynamic view synthesis, and enables long-range 3D tracking; and a large transformer network that achieves real-time, generalizable dynamic scene reconstruction. Qualitative and quantitative experiments demonstrate that DGS-LRM achieves dynamic scene reconstruction quality comparable to optimization-based methods, while significantly outperforming the previous predictive dynamic reconstruction method on real-world examples. Its predicted physically grounded 3D deformation is accurate and can be readily adapted for long-range 3D tracking tasks, achieving performance on par with state-of-the-art monocular video 3D tracking methods.

1 Introduction

Reconstructing a dynamic scene from a monocular video, recovering accurate geometry, appearance, and motion, remains a significant challenge in computer vision and graphics. This task has numerous applications, including visualization, augmented/virtual reality (AR/VR), and robotics. Recent advances in this domain [46, 83, 97] have been largely driven by the development of neural representations, such as neural radiance fields [55] and 3D Gaussian splats [38], as well as deep priors for specific scene attributes like depth [85, 42, 48, 95, 66] and flow [75, 76, 37]. These methods tackle dynamic scene reconstruction by optimizing particular scene representations using densely captured images, integrating various deep priors to provide robust regularizations. Although latest methods using variants of Gaussian splatting [38] representation achieve real-time rendering [97, 88, 23, 83], the optimization process is often time-consuming and computationally expensive, limiting their practical applicability.

Recent generalizable 3D feed-forward networks [85, 100, 30, 102, 94] directly predicts 3D representations from sparse-view image inputs, achieving speeds several orders of magnitude faster than previous optimization-based methods. Inspired by the success of foundation models in natural language processing [65] and 2D computer vision [19, 39], recent works explore training transformer-based [77] networks on large-scale 3D [14, 15] or video datasets to enable generalizable 3D reconstruction. However, prior efforts assume static scenes, leaving the challenge of handling dynamic scenes and accurately predicting motion still unsolved.

In this paper, we present DGS-LRM, the first feed-forward transformer designed to predict deformable 3D Gaussians from a posed monocular video. The predicted deformable 3D Gaussians can render novel view and scene flow of dynamic scenes. We introduce three key technical innovations to address this challenge.

First, we employ per-pixel deformable 3D Gaussians to represent predicted dynamic scenes, extending the recent success in feed-forward static 3D Gaussians prediction methods [102, 94, 10]. For each pixel in the target frame, our method predicts its corresponding 3D Gaussian splat and 3D scene flow from its current timestamp to all other timestamps in the input video, enabling high-quality novel view synthesis through warping 3D Gaussians across frames. Additionally, our representation is robust to occlusions and discontinuities. The predicted 3D scene flow can be chained together using sliding windows, achieving performance on par with the state-of-the-art 3D tracking method [90].

Second, motivated by the recent trend in training LRM with pure synthetic data [91, 35], we utilize multi-view synthetic data with ground-truth 3D scene flow as a primary supervision during training. This contrasts with prior scene-level feed-forward reconstruction models, which are typically trained on monocular video or RGB-D images. However, using pure photometric supervision on a monocular video has ambiguities in motion and geometry predictions for dynamic scenes. We create a customized large-scale dataset using Kubric [26], featuring multi-view renderings paired with per-pixel 3D scene flow. Our results show that training on such a large-scale multi-view synthetic dataset significantly enhances reconstruction quality and enables good generalization on real-world data as well.

Third, we employ temporal tokenization [4, 62] to compress input videos into compact, small pixel cubes. Unlike GS-LRM [102], which tokenizes each frame independently, our approach is computationally efficient and scales well for both training and inference. Additionally, we incorporate discretely sampled temporally distant reference frames as inputs to leverage frames with larger camera baselines, following [49], which effectively reduces geometric ambiguities.

Our best model achieves real-time inference on an A100 GPU while maintaining reconstruction quality on par with optimization-based deformable Gaussian splatting methods [97], which require hours of computation and several pre-trained models for initialization. Compared to the existing predictive dynamic object reconstruction method [67], DGS-LRM demonstrates significantly better generalization ability on real-world examples, improving PSNR by 3 points on [25]. Moreover, DGS-LRM accurately reconstructs object motion. Our predicted 3D scene flow delivers competitive quantitative performance [90] on standard point tracking benchmarks [26, 107]. More qualitative results from DAVIS [60] further demonstrate high-quality novel-view synthesis and accurate flow prediction, as shown in Figure 1.

2 Related Work

Dynamic reconstruction and view synthesis Early dynamic scene reconstruction methods [3, 33, 20, 57, 108] primarily focus on non-rigid mesh reconstruction, typically requiring RGBD images

as inputs. Video depth prediction methods predict consistent depth maps across video frames [40, 54, 105, 104] by integrating monocular depth priors with strong hand-crafted regularizations. Both of these approaches concentrate on geometry and are not equipped to support realistic novel view synthesis. Recent advancements in dynamic view synthesis have shown significant progress by incorporating novel neural representations, such as neural radiance fields [55] and 3D Gaussian splats [38], into dynamic scene reconstruction. Most of these methods require multi-view videos as inputs [52, 7, 9, 24, 44, 47, 72, 73, 79], which considerably limits their practical applicability. Several methods [51, 22, 23, 45, 58, 59, 78, 88, 89, 97, 96, 83, 29] target the more challenging task of monocular dynamic view synthesis. While these methods remarkably improve view synthesis quality, they heavily rely on geometry and motion priors from pre-trained models, alongside meticulously designed time-consuming optimization processes to achieve state-of-the-art reconstruction quality. On the contrary, DGS-LRM aims to learn priors from a single large transformer model, enabling efficient, generalizable dynamic scene reconstruction in a feed-forward manner. In particular, PGDVS achieves high-quality novel view synthesis on the DyCheck [25] benchmark while significantly reducing reconstruction time compared to many prior optimization-based methods. It leverages off-the-shelf depth and optical flow estimators for initialization. Uniquely, it renders novel views at different timestamps via image-space warping and aggregates results using dynamic object masks, which produce sharp appearances. However, the method still requires hours to build its representation, and in some challenging cases with complex motion, warping and masking errors can cause temporal flickering artifacts. In our work, we focus on an end-to-end feedforward representation that can deliver similar outputs while being orders of magnitude faster in inference.

Feed-forward reconstruction Many feed-forward methods can predict neural 3D representations for novel view synthesis or geometry reconstruction. Early methods [11, 99, 81, 92] often struggle to match the reconstruction quality of optimization-based methods and, therefore, require fine-tuning [92, 11] to enhance their results. Recently, many works [85, 42, 13] have significantly increased network capacity by employing large transformer networks for better 3D reconstruction. Among these, large reconstruction models (LRMs) [30, 43, 93, 87, 80, 102] represent a family of approaches that achieve state-of-the-art novel view synthesis quality from sparse, posed, or unposed images. The latest LRM methods [102, 94, 98] predict pixel-aligned 3D Gaussians for realistic static scene-level view synthesis but fail to handle dynamic objects. Several concurrent works aim to reconstruct dynamic scene geometry [100, 48, 84] but not appearance and motion. The closest existing method [67] predicts time-dependent 3D Gaussians to reconstruct dynamic objects. However, we observe that it generalizes significantly worse on real-world examples, even when the dynamic objects are segmented out in the inputs. This may be due to the limited motion and appearance diversity in its training data and its 3D representation, which fails to model accurate scene flow. [49] achieves better generalization by training on self-curated internet videos, but it still does not reconstruct accurate scene flow as our model.

Flow and tracking The majority of tracking methods aim to find correspondences in 2D image space. Classical optical flow methods [2, 6, 5, 32, 18, 75] estimate dense 2D pixel motion between two consecutive frames and, therefore, are not suitable for long-range tracking. Feature tracking methods [1, 16, 53, 69] can track pixels over long ranges, but only handle sparse points. Several efforts [71, 68] have been made to combine the merits of the two to achieve dense long-range 2D tracking, either by concatenating consecutive 2D flows through test-time optimization [56, 82] or by relying on data-driven approaches [17, 28, 37]. Another line of research estimates correspondences in 3D space to mitigate issues caused by 3D-2D projection and to leverage better regularization. Most of these approaches require RGBD images or point clouds as inputs [27, 50, 63, 86, 34, 64, 74, 76] or test-time optimization [31, 46, 45] to jointly reconstruct geometry. SpatialTracker [90] is the closest state-of-the-art work that predicts dense 3D scene flow from a monocular video. Experiments show that DGS-LRM achieves comparable tracking accuracy while offering a more versatile framework that supports high-quality novel view synthesis.

3 Methodology

In Sec. 3.1, we introduce the network architecture and the convention of DGS-LRM. Sec. 3.2 outlines key training details. Sec. 3.3 describes synthetic training data.

3.1 DGS-LRM

Figure 2 illustrates the inputs, transformer network architecture, and the predicted 3D representation of DGS-LRM, which we will discuss in detail.

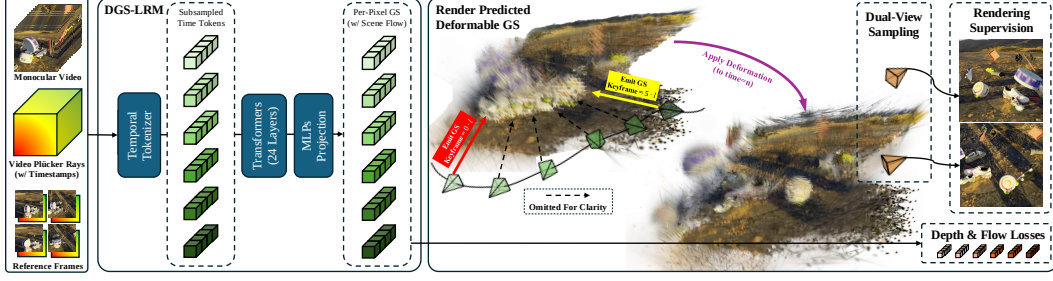


Figure 2: **DGS-LRM overview.** We first concatenate multi-view videos with a time-aware Plucker ray and tokenize them using the spatial-temporal tokenizer. Then, the transformers take the sampled time tokens as input and predict per-pixel deformable Gaussians with 3D scene flow. During training, we rendered multi-view synthetic videos using Kubric. We draw a dual-view ground-truth in each sample at the same timestamp, rendering views, depth, and scene flows.

Network inputs. Given a monocular video sequence $\mathcal{I} = \{I_0, \dots, I_N\}$ with per-frame camera parameters $\mathcal{C} = \{C_0, \dots, C_N\}$, DGS-LRM aims to predict a deformable Gaussian splatting (DGS) reconstruction of the visible environment. Following LRM [30], we encode camera calibrations as Plucker rays, where each $C_n \in \mathcal{C}$ pairs with an image $I_i \in \mathcal{I}$ with the same resolution ($H \times W$). Each pixel p of C_n is a seven-dimensional vector containing a Plucker ray and a timestamp. The timestamp is the temporal index of the frame within the input time window $\{0, \dots, N\}$, and normalized to $[0, 1]$.

Processing a large number of video frames using a standardized self-attention transformer requires prohibitively large GPU VRAM. However, limiting the window size N can reduce the camera baselines among input frames, making it more challenging for the model to reconstruct accurate geometry. To address this issue, we follow BTimer [49] and additionally introduce *optional* reference frames $\mathcal{R} = \{R_0, \dots, R_K\}$ (omitting the corresponding Plucker rays for clarity). These reference frames aim to sample more views with larger camera baselines, which provides additional clues for geometry reconstruction. Consequently, we sample $\mathcal{R}$ temporally distant from $\mathcal{I}$, and we do not predict per-pixel deformable 3D Gaussians for these frames.

Deformable Gaussian Splats. Given posed multiview images, DGS-LRM predicts a set of deformable Gaussian parameters by

$$\mathcal{G} = \text{DGS-LRM}(\mathcal{I}, \mathcal{C}, \mathcal{R}), \quad (1)$$

where $\mathcal{G} = \{G_0, G_l, G_{2-l}, \dots, G_N\}$. Here, l is the temporal downsampling rate, and G_{x-l} is a set of deformable 3D Gaussians that are pixel-aligned with a keyframe $x \cdot l \in \{0, \dots, N\}$. In G_{x-l} , each pixel p contains a single Gaussian splat g_p , parametrized by 1-channel depth d_p , 3-channel RGB colors, 4-channel quaternion rotation, 3-channel scale, 1-channel opacity, and a set of 3-channel deformation vectors $\mathbf{f}_p = \{f_0, f_1, \dots, f_N\}$ to warp Gaussian splat g_p into the target timestamp $n \in \{0, \dots, N\}$. Our deformation vectors only model translation. We observe that deforming rotation will not further improve reconstruction quality, possibly because our per-pixel Gaussian representation is dense enough. Deforming opacity, on the other hand, allows the model to fake dynamic appearances without predicting accurate motion. To render a novel view image at timestamp n , we warp every Gaussian splats g_p in $\mathcal{G}$ using the corresponding deformation vector $\mathbf{f}_p$ to form a new set of 3D Gaussian $\mathcal{W}_n$ for rendering

$$\mathcal{W}_n = \bigcup_p \text{warp}(g_p, \mathbf{f}_p). \quad (2)$$

A visualization of the warping process is shown in Figure 3.

Architecture. Our model consists of three components: an input tokenizer, a transformer, and output MLPs to project the results. The input images (both $\mathcal{I}$ and $\mathcal{R}$) and their corresponding Plucker rays are first channel-concatenated together. We use *temporal tokenization*, inspired by the temporal autoencoder from MovieGen [62]. Instead of patchifying each image patch with $s \times s$

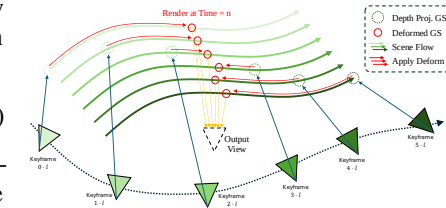


Figure 3: A visualization of pixel-aligned deformable 3D Gaussians are used for novel view synthesis at arbitrary timestamp n .

pixels, we consider a video as a volume and patchify a cube across spatial and temporal domains with $s \times s \times l$ pixels. This cube is turned into a token through a simple linear layer. We use $l = 4$ in practice, reducing the number of tokens by four times, which significantly accelerates both training and inference. In practice, the model is not trainable and expensive to infer without temporal tokenization. We use a standard transformer architecture [77] with 24 multi-head self-attention blocks. We additionally add weight normalization [70] to all parameters and observe that it can stabilize the training. Finally, we use two-layer MLPs to project the tokens into deformable GS parameters.

3.2 Training

Photometric losses Similar to prior works [102, 94], DGS-LRM can be trained end-to-end with photometric losses through differentiable rasterization. We sample a set of ground-truth images $\mathcal{I} = \{\bar{I}_0, \dots, \bar{I}_Q\}$ with corresponding camera poses $\bar{\mathcal{C}} = \bar{c}_1, \dots, \bar{c}_Q$, and define $n_q \in \{1, \dots, N\}$ as their timestamps. To produce a predicted rendering $\hat{I}_q$, we first deform the predicted 3D Gaussians to timestamp n_q as $\mathcal{W}_{n_q}$, then rasterize it to the output camera pose $\bar{c}_q$ with

$$\hat{I}_q = \text{rasterize}(\mathcal{W}_{n_q}, \bar{c}_q) . \quad (3)$$

We then use MSE loss and perceptual loss [103] to supervise the rendered image.

$$L_{\text{mse}} = \sum_q \text{MSE}(\bar{I}_q, \hat{I}_q); \quad L_{\text{lpips}} = \sum_q \text{LPIPS}(\bar{I}_q, \hat{I}_q) . \quad (4)$$

Output view selection and dual-view supervision. Static scene LRMs [30, 102] sample intermediate video frames from monocular videos as training data, which can lead to geometry and motion ambiguity when training with dynamic monocular videos. To remove this ambiguity, we use time-synchronized multi-view videos to train DGS-LRM.

We start with sampling two video sequences; one serves as the input $\mathcal{I}$, while we sample K frames from the other sequence as $\mathcal{R}$. For output view supervision, we empirically found that sampling two views at the same timestamp provides significantly better training convergence. We name this strategy dual-view sampling and show its effectiveness in Figure 7. With dual-view sampling, we sample two different training views for each timestamp from two synchronized video sequences, which retrieves $Q/2$ frames from each video for each training step. To ensure the inputs and outputs of DGS-LRM have sufficient covisibility, we allow one of the output sequences to overlap with either the input or the reference sequence.

Depth and scene flow supervisions. We apply depth supervision L_{depth} to improve surface geometry

$$L_{\text{depth}} = \sum_g \sum_p \|\bar{d}_p - d_p\|_1 . \quad (5)$$

Here, we directly supervise with pixel-aligned ground-truth depth values $\bar{d}_p$ for key frame $\{0, l, 2l, \dots, N\}$, which we observe to be more effective compared to supervising rendered depth maps at output views. Simply combining rendering loss and depth loss is still not enough for accurate motion reconstruction. We often observe discontinuities in the predicted deformation trajectories. Sometimes, the 3D Gaussian splats are even moved outside the camera frustum. Consequently, we introduce the flow loss L_{flow} to regularize the point deformation. Similar to the depth loss, for every key frame, we supervise our deformation vector $\mathbf{f}_p$ with ground-truth deformation to every timestamp. The flow loss is written as

$$L_{\text{flow}} = \sum_g \sum_p \|\bar{\mathbf{f}}_p - \mathbf{f}_p\|_1 . \quad (6)$$

How to obtain ground-truth deformation vectors $\bar{\mathbf{f}}_p$ will be detailed in Sec. 3.3.

Total loss. Finally, the whole network is end-to-end trained with the total objective

$$L_{\text{total}} = L_{\text{mse}} + \lambda_{\text{lpips}} \cdot L_{\text{lpips}} + \lambda_{\text{depth}} \cdot L_{\text{depth}} + \lambda_{\text{flow}} \cdot L_{\text{flow}} , \quad (7)$$

where the λ 's are weighting factors of the objectives. We use $\lambda_{\text{lpips}} = 0.5$, $\lambda_{\text{depth}} = 10$, and $\lambda_{\text{flow}} = 10$.

Scene normalization. We found that inconsistent scene scales in training and inference can cause instability and generalization issues. We use a normalization approach similar to MegaSaM [48] for both training and inference. We first use a monocular metric depth estimator [61] to identify the scale of the scene and then normalize the scene scale so that the disparity of the 20th depth percentile is equal to 2. Such a scene scale is applied to camera poses, ground-truth depths, and scene flows.

3.3 Training Data Creation

Due to limited real-world *posed* monocular videos that contain diverse *dynamics* with sufficient ground truths (3D scene flow, multi-view images) to address the motion ambiguity, we primarily train our DGS-LRM on a self-generated Kubric [26] dataset. We also explored combining it with various real-world videos for training but did not observe major improvements.

Customized Kubric. We follow the MOVi-E setting, using the Kubric engine [26] to create synthetic dynamic scenes. These scenes contain diverse objects being tossed around, simulated by a physics engine. To ensure that our video is closer to real-world monocular dynamic videos [25], we decrease the default maximum camera trajectory length from 8 meters to 0.5 meters to improve the model’s generalization ability. We generate the dataset with 4 synchronized cameras. We first sample one camera, then sample 3 other cameras relative to the first one, with distances ranging from 4 to 16 meters. All these cameras share the same look-at points so that they can have sufficient co-visibility. To further reduce the sim-to-real domain gap, we apply additional domain randomizations. For each scene, we add motion blur to one of the cameras and make sure the camera is never being sampled as the output supervision but only as the input. We also sample varying focal lengths for each camera, ranging from 25mm to 55mm.

Scene Flow Extraction. We extract ground-truth 3D scene flow following MegaSaM [48]. The Kubric engine supports rendering per-pixel object coordinates, which specify the location where the ray cast from the camera intersects the object surface. Moreover, Kubric also records the per-frame object trajectory and rotations during physics simulation. Given that Kubric only contains rigid objects, we can retrieve the 3D scene flow for every pixel at every timestamp by combining the 2 attributes together.

However, 3D scene flow has an $O(N \times M \times P \times 3)$ space complexity. In practice, using $N = 24$, $M = 6$, and $P = 512 \times 512$ results in 0.84 GB data for one camera. Loading a batch of such data in each iteration would exhaust the memory and data I/O. Fortunately, we observe that the scene flow is significantly sparse, where the majority of points in the scene are stationary. By storing 3D scene flow as sparse tensors, we reduce the memory and I/O cost by 80%.

3.4 Flow Chaining

Our DGS-LRM is trained to handle short video clips (1s). Theoretically, it can be trained to handle longer videos, but this is computationally expensive in practice. However, we can *chain* multiple sequences of scene flows together to achieve long-range tracking, which we will detail below.

Given two sequences with predicted scene flows, the *flow chaining* first set the end frame of the first video as the first frame of the second video. We deform the two independent deformable GS into the same timestamp. Then, we find a nearest neighbor for each scene flow to temporally chain two flows into one. The nearest neighbor is measured by two distance quantities: the distance between the deformed GS, and the direction similarity between the momentary scene flow. For the first video, the momentary scene flow is the relative scene flow between the last two timestamps; for the second video, it is the relative scene flow between the first two timestamps. The two distance quantities result in a six-value vector, and we measure the distance between these vectors with a simple L1 distance. We use Faiss [21] to efficiently compute such distance with GPU support. After computing all pair-wise distances between the scene flows from the first video and the scene flows from the second video, we index the nearest neighbor and temporally chain each pair of flows into one.

However, as some points in one video may be completely invisible in the second video (such as moving outside the frustum), some flows in one video may not have a valid match in the other video. We use a threshold to filter out such cases and concatenate these unmatched scene flows with zero values, meaning the points remain stationary after it loses track. These lost tracked scene flows are still evaluated when we compare them with the 3D tracking methods in Table 2. We also provide an idealized evaluation where these unmatched scene flows are excluded from the evaluation, marked as Flow Valid (FV) in Table 2.

4 Experiments

Hyperparameters. We train our method with 64 H100 GPUs with 80GB VRAM. For all variants of DGS-LRM, we use $N = 24$ input frames with temporal sampling rate $l = 4$, which results in 6 keyframes after temporal tokenization. We use $K = 4$ for reference views and set the number of output views per scene to $Q = 8$. For training efficiency, we first train the model at 256×256

Table 1: **Monocular Dynamic View Synthesis on DyCheck [25]**. DGS-LRM outperforms LRM-based L4GM [67], and is comparable to optimization-based novel-view synthesis methods with a substantially faster reconstruction time. DynMask applies a dynamic mask to evaluate the foreground only.

Method	Time (s)	DynMask	mPSNR ($\uparrow$)	mLPIPS ($\downarrow$)
D3DGS [97]	1-3 hours	$\times$	11.92	0.66
PGDV5 [106]	3 hours	$\times$	15.88	0.34
Ours	0.495 sec	$\times$	14.89	0.42
L4GM [67]	4.8 sec	$\checkmark$	5.84	0.67
Ours	0.495 sec	$\checkmark$	11.97	0.51

Table 2: **3D Tracking on PointOdyssey [107]**. DGS-LRM reconstructed 3D deformation field is comparable to state-of-the-art 3D tracking methods. FC is flow chaining that combines scene flows from multiple segments. FV is fully visible, which evaluates only the tracking points not occluded for more than 24 frames (DGS-LRM input length).

Method	Frames	PSNR	ATE-3D ($\downarrow$)	$\delta_{0.1}$ ($\uparrow$)	$\delta_{0.2}$ ($\uparrow$)
Chained RAFT3D [76]	120	N/A	0.70	0.12	0.25
Lifted CoTracker [37]	120	N/A	0.77	0.51	0.64
SpatialTracker [90]	120	N/A	0.22	0.59	0.76
Ours (FC)	120	27.77	0.21	0.57	0.68
Ours (Native)	24	27.77	0.11	0.72	0.84
Ours (FC + FV)	120	27.77	0.15	0.64	0.75

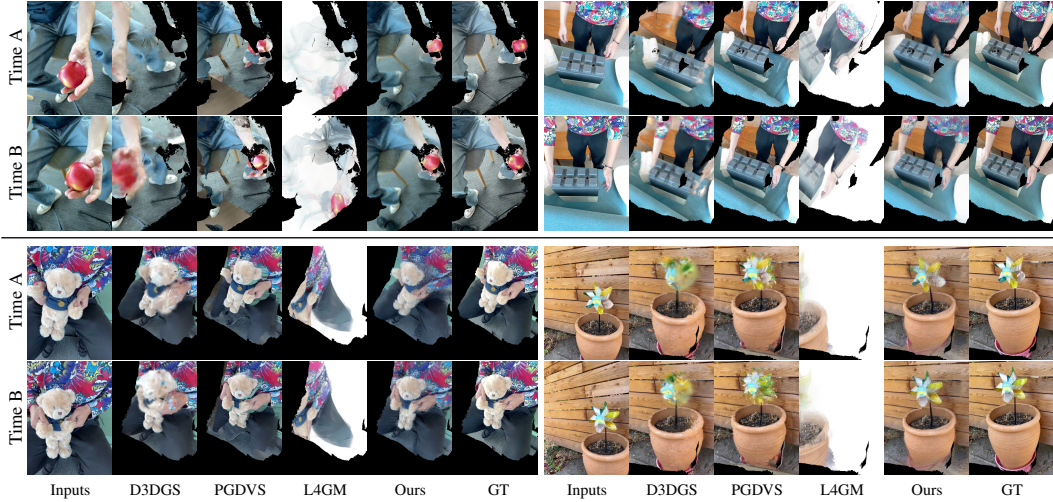


Figure 4: **DyCheck iPhone dataset**. Our DGS-LRM outperforms D3DGS [97] and does not have warping artifacts as in PGDV5 [106]. Both methods fail to recover the geometry and the repetitive motion of the windmill. We mask out the zero covisible regions with black pixels.

resolution and then fine-tune it at 512×512 resolution. We render the Kubric dataset according to these two setups and create 40,000 scenes (each with 4 synchronized cameras) for both resolutions. For the first stage of training, we use a batch size of 15 per GPU, train for 40k iterations with a learning rate of $4e-4$, and then decay to $1e-6$ with a cosine learning rate scheduler. For the second stage, we use a batch size of 8 per GPU, train for 20k iterations with a learning rate of $1e-4$, and then decay to $1e-6$ with a cosine learning rate scheduler. For both stages, we use a learning rate warm-up for 500 iterations, which linearly ramps up the learning rate from 0 to the initial learning rate. Similar to GS-LRM, we apply the common practice to save GPU VRAMs using xFormers [41], deferred backpropagation [101], gradient checkpointing [12], and BF16 mixed-precision training [36].

4.1 Novel Dynamic View Synthesis

We evaluate DGS-LRM on DyCheck [25] and DAVIS [8]. DGS-LRM requires the input to be a temporally continuous video with a non-stationary camera, and cannot process teleporting cameras with discrete poses. Due to the lack of real-world benchmarks with multiple non-stationary and synchronized cameras, our quantitative evaluation is limited to DyCheck. We use the iPhone subset of DyCheck, which includes two synchronized novel-view cameras for reconstruction metrics evaluation. The iPhone subset contains 7 long monocular videos, 200-400 frames each. In addition, DyCheck also labels the covisibility between training and novel-view cameras and evaluates the masked version of reconstruction metrics. For fair comparison, all methods use the same structure-from-motion depth for initialization or scene normalization. DAVIS provides a large and diverse set of in-the-wild monocular videos to test the method’s generalization. We use MegaSaM [48] to create the camera poses for DAVIS. We only evaluate qualitative results, as DAVIS does not have novel-view cameras.

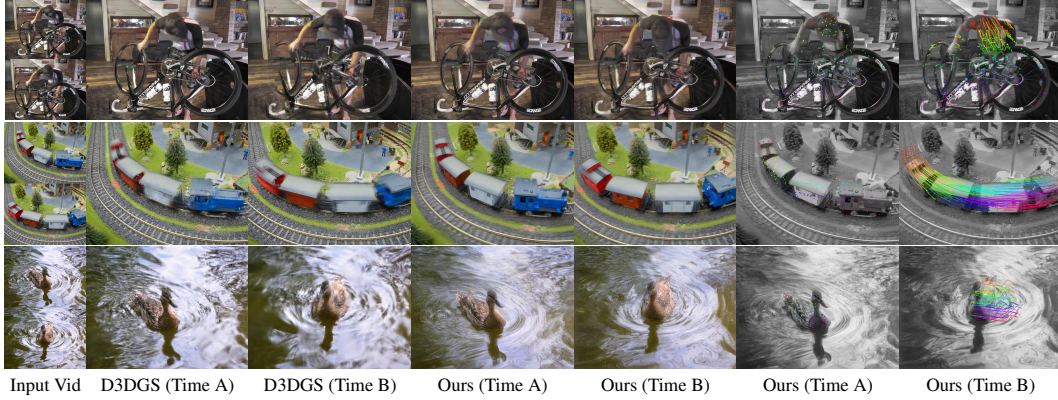


Figure 5: **DAVIS dataset.** DGS-LRM outperforms D3DGS in synthesizing thin details and can predict consistent dynamic motion trajectories (visualized in color).

In Table 1 and Figure 4, we show that our DGS-LRM outperforms the baseline predictive method, L4GM [67], while performing comparably to the state-of-the-art optimization-based reconstruction methods, which all take hours to complete. L4GM is designed for 4D object reconstruction without reconstructing explicit deformation. As L4GM only reconstructs dynamic foreground objects, we label the foreground mask following [83], then reconstruct and evaluate the masked region only. In Figure 4, both D3DGS [97] and L4GM often resolve a wrong scene scale or semi-transparent geometries, PGDVS [106] often creates broken geometry due to warping, while our method can handle them well. Neither D3DGS nor PGDVS can correctly resolve the repetitive and fine-grained deformations in the paper windmill scene. In addition, all optimization baseline methods are optimized with the entire sequence, while our DGS-LRM only uses a 24-frame continuous clip and 4 reference frames to reconstruct the scene. The 4 reference frames are selected $[-96, -48, 48, 96]$ frames apart from the main input sequence, and we increase the number of frames in the opposite direction when meeting the start or end of the sequence. In addition, the covisibility labeling of DyCheck is dedicated to methods optimizing with the full video sequence, which is disadvantageous to DGS-LRM with a small input window.

In Figure 5, we show that DGS-LRM can perform well even on in-the-wild videos. Compared with D3DGS [97], we can correctly reconstruct the thin geometries of the bike wheels and challenging scenes with water deformation. In flow visualizations, we illustrate that DGS-LRM effectively tracks the complex deformations in hand motion and wheel turning while maintaining consistent flow in the train’s rigid body movements. For visual clarity, we randomly downsample the per-pixel dense flows into sparse flows and then mask out the background flows using the object mask provided by DAVIS.

4.2 3D Tracking

An accurate deformable 3D reconstruction should have a 3D deformation field aligned to the physically grounded 3D tracking trajectories. In Table 2 and Figure 6, we evaluate the quality of the reconstructed scene flow on the PointOdyssey benchmark [107]. PointOdyssey includes 13 videos (ranging from 1,000 frames to 4,000 frames) of synthetic scenes with humanoid and animal meshes articulated with transferred real-world motions. We compare DGS-LRM with the state-of-the-art SpatialTracker [90] and two baselines proposed in it. DGS-LRM with flow chaining achieves comparable performance with SpatialTracker and performs better than other baselines. In Figure 6, we found that SpatialTracker struggles on texture-less surfaces, where the tracking points drift over time in the regions with similar textures. In contrast, as DGS-LRM prediction involves a total reconstruction of the object geometry with accurate depth prediction, it can better track points in completely texture-less areas. For instance, the point registration on the ear root of the rabbit and the knee of the humanoid. As discussed in Sec 3.4, the chaining process involves failures when some points are completely invisible in a video segment. Therefore, we provide two additional variants: the native tracking performance without chaining (marked as Native) and an oracle case where we omit the flows with a significant discontinuity (threshold by L1 distance 0.1 meters) during the chaining process, which is marked as FC + FV. While not directly comparable to SpatialTracker, these two variants show the actual quality of the predicted scene flows without the impairment caused by flow chaining.

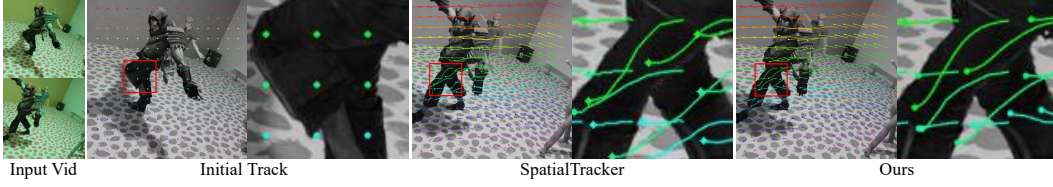


Figure 6: **Qualitative comparisons to SpatialTracker[90] on the PointOdyssey dataset.** DGS-LRM shows better performance and consistency in texture-less areas. SpatialTracker predicts certain tracks inconsistent with the object’s moving direction. Such as several tracking points drift and collide in the humanoid’s knee.

Table 3: **Ablation Study.** We show that each component contributes to the final performance. Note that the study is conducted with low-resolution models trained at 256×256 resolution. (TT: temporal tokenization. DV: dual-view sampling. SF: scene flow loss. RF: reference frames.)

Method	DyCheck		Kubric-MV (Test)	
	mPSNR ($\uparrow$)	mLPIPS ($\downarrow$)	mPSNR ($\uparrow$)	mLPIPS ($\downarrow$)
w/o TT	OOM	OOM	OOM	OOM
w/o DV	14.72	0.412	25.77	0.171
w/o SF	14.29	0.423	25.06	0.189
w/o RF	13.91	0.438	24.69	0.186
Full	14.67	0.412	26.05	0.161

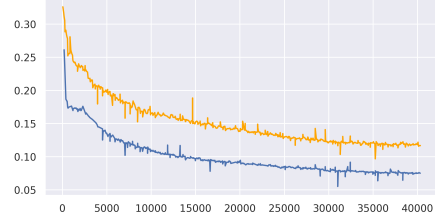


Figure 7: Dual view supervision (blue curve) provides better training convergence in Perceptual Loss (LPIPS) compared to its counterpart (orange curve).

4.3 Ablation Study

In Table 3, we show that each proposed components contribute to the final performance. The ablation is conducted at the 256×256 resolution training stage, and we compare the rendering quality on the DyCheck benchmark. The temporal tokenization enables training at scale by significantly reducing the memory consumption. In Figure 7, we show that the dual-view sampling improves the training convergence on Kubric and leads to a better performance on our additionally rendered Kubric holdout test set (with new objects and HDRIs unseen during training). The scene flow objective and reference frames both significantly boost the performance by a large margin. In practice, we found the scene flow loss significantly improves the rigidity of the deformation, and the reference frames help solve the scene scale and depth by triangulating with distant views.

In the right table, we evaluate static GS-LRM on DyCheck dataset, quantitatively showing the significance of modeling the deformation. Additionally, despite our DGS-LRM only emits 3D Gaussians at down-sampled keyframes, we show that the performance gap between the keyframes and non-keyframes are negligible.

Method	PSNR	LPIPS
GS-LRM (static)	13.02	0.444
DGS-LRM (keyframes)	15.04	0.419
DGS-LRM (non-keyframes)	14.84	0.421
DGS-LRM (all)	14.89	0.420

5 Conclusion and Limitations

We introduce DGS-LRM, the first feed-forward network capable of predicting deformable 3D Gaussians from a posed monocular video in real-time. The predicted deformable Gaussians enable novel-view rendering, geometry reconstruction, and 3D scene flow estimation in world space. We train DGS-LRM on a large-scale multi-view rendered synthetic dataset and show that it generalizes well to real-world videos of varying complexity. Unlike prior monocular deformable 3D reconstruction methods, which require lengthy optimization to reconstruct and fuse priors from multiple individual networks, we demonstrate the potential of predicting deformable 3D Gaussians end-to-end and learning dynamic scene priors within a single network.

Limitations. DGS-LRM has a few limitations that can be explored in future works. As the model is trained with temporally continuous video, it cannot handle discrete image frames. Our predicted scene flow cannot handle extremely large motion in the scene, which may stem from the motion distribution of the physically simulated synthetic dataset. Although our flow chaining enables continuous 3D tracking, it still causes a noticeable jump in appearances for novel view synthesis. The input video baseline and distribution significantly influence the quality of novel view rendering quality. As the view deviates from the input trajectory, artifacts gradually intensify in the rendered images as well.

References

- [1] Herbert Bay, Tinne Tuytelaars, and Luc Van Gool. Surf: Speeded up robust features. In *ECCV*, pages 404–417, 2006. [3](#)
- [2] Michael J Black and Padmanabhan Anandan. A framework for the robust estimation of optical flow. In *ICCV*, pages 231–236, 1993. [3](#)
- [3] Aljaz Bozic, Michael Zollhofer, Christian Theobalt, and Matthias Nießner. Deepdeform: Learning non-rigid rgb-d reconstruction with semi-supervised data. In *CVPR*, pages 7002–7012, 2020. [2](#)
- [4] Tim Brooks, Bill Peebles, Connor Holmes, Will DePue, Yufei Guo, Li Jing, David Schnurr, Joe Taylor, Troy Luhman, Eric Luhman, Clarence Ng, Ricky Wang, and Aditya Ramesh. Video generation models as world simulators, 2024. [2](#)
- [5] Thomas Brox, Andrés Bruhn, Nils Papenberg, and Joachim Weickert. High accuracy optical flow estimation based on a theory for warping. In *ECCV*, pages 25–36, 2004. [3](#)
- [6] Thomas Brox, Christoph Bregler, and Jitendra Malik. Large displacement optical flow. In *CVPR*, pages 41–48, 2009. [3](#)
- [7] Michael Broxton, John Flynn, Ryan Overbeck, Daniel Erickson, Peter Hedman, Matthew Duvall, Jason Dourgarian, Jay Busch, Matt Whalen, and Paul Debevec. Immersive light field video with a layered mesh representation. *ACM Transactions on Graphics*, 39(4):86–1, 2020. [3](#)
- [8] Sergi Caelles, Alberto Montes, Kevis-Kokitsi Maninis, Yuhua Chen, Luc Van Gool, Federico Perazzi, and Jordi Pont-Tuset. The 2018 davis challenge on video object segmentation. *arXiv:1803.00557*, 2018. [7](#)
- [9] Ang Cao and Justin Johnson. Hexplane: A fast representation for dynamic scenes. In *CVPR*, pages 130–141, 2023. [3](#)
- [10] David Charatan, Sizhe Lester Li, Andrea Tagliasacchi, and Vincent Sitzmann. pixelsplat: 3d gaussian splats from image pairs for scalable generalizable 3d reconstruction. In *CVPR*, pages 19457–19467, 2024. [2](#)
- [11] Anpei Chen, Zexiang Xu, Fuqiang Zhao, Xiaoshuai Zhang, Fanbo Xiang, Jingyi Yu, and Hao Su. Mvsnerf: Fast generalizable radiance field reconstruction from multi-view stereo. In *ICCV*, pages 14124–14133, 2021. [3](#)
- [12] Tianqi Chen, Bing Xu, Chiyuan Zhang, and Carlos Guestrin. Training deep nets with sublinear memory cost. *CoRR*, abs/1604.06174, 2016. [7](#)
- [13] Yuedong Chen, Haofei Xu, Chuanxia Zheng, Bohan Zhuang, Marc Pollefeys, Andreas Geiger, Tat-Jen Cham, and Jianfei Cai. Mvsplat: Efficient 3d gaussian splatting from sparse multi-view images. In *ECCV*, pages 370–386, 2024. [3](#)
- [14] Matt Deitke, Dustin Schwenk, Jordi Salvador, Luca Weihs, Oscar Michel, Eli VanderBilt, Ludwig Schmidt, Kiana Ehsani, Aniruddha Kembhavi, and Ali Farhadi. Objaverse: A universe of annotated 3d objects. In *CVPR*, pages 13142–13153, 2023. [2](#)
- [15] Matt Deitke, Ruoshi Liu, Matthew Wallingford, Huong Ngo, Oscar Michel, Aditya Kusupati, Alan Fan, Christian Laforte, Vikram Voleti, Samir Yitzhak Gadre, et al. Objaverse-xl: A universe of 10m+ 3d objects. In *NeurIPS*, 2024. [2](#)
- [16] Daniel DeTone, Tomasz Malisiewicz, and Andrew Rabinovich. Superpoint: Self-supervised interest point detection and description. In *CVPR*, pages 224–236, 2018. [3](#)
- [17] Carl Doersch, Yi Yang, Mel Vecerik, Dilara Gokay, Ankush Gupta, Yusuf Aytar, Joao Carreira, and Andrew Zisserman. Tapir: Tracking any point with per-frame initialization and temporal refinement. In *ICCV*, pages 10061–10072, 2023. [3](#)
- [18] Alexey Dosovitskiy, Philipp Fischer, Eddy Ilg, Philip Hausser, Caner Hazirbas, Vladimir Golkov, Patrick Van Der Smagt, Daniel Cremers, and Thomas Brox. FlowNet: Learning optical flow with convolutional networks. In *ICCV*, pages 2758–2766, 2015. [3](#)
- [19] 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. *arXiv preprint arXiv:2010.11929*, 2020. [2](#)

- [20] Mingsong Dou, Sameh Khamis, Yury Degtyarev, Philip Davidson, Sean Ryan Fanello, Adarsh Kowdle, Sergio Orts Escolano, Christoph Rhemann, David Kim, Jonathan Taylor, et al. Fusion4d: Real-time performance capture of challenging scenes. *ACM Transactions on Graphics*, 35(4):1–13, 2016. [2](#)
- [21] Matthijs Douze, Alexandr Guzhva, Chengqi Deng, Jeff Johnson, Gergely Szilvassy, Pierre-Emmanuel Mazaré, Maria Lomeli, Lucas Hosseini, and Hervé Jégou. The faiss library. *arXiv preprint arXiv:2401.08281*, 2024. [6](#)
- [22] Yilun Du, Yanan Zhang, Hong-Xing Yu, Joshua B Tenenbaum, and Jiajun Wu. Neural radiance flow for 4d view synthesis and video processing. In *ICCV*, pages 14304–14314, 2021. [3](#)
- [23] Yuanxing Duan, Fangyin Wei, Qiyu Dai, Yuhang He, Wenzheng Chen, and Baoquan Chen. 4d-rotor gaussian splatting: towards efficient novel view synthesis for dynamic scenes. In *ACM SIGGRAPH*, pages 1–11, 2024. [2](#), [3](#)
- [24] Sara Fridovich-Keil, Giacomo Meanti, Frederik Rahbæk Warburg, Benjamin Recht, and Angjoo Kanazawa. K-planes: Explicit radiance fields in space, time, and appearance. In *CVPR*, pages 12479–12488, 2023. [3](#)
- [25] Hang Gao, Ruilong Li, Shubham Tulsiani, Bryan Russell, and Angjoo Kanazawa. Dynamic novel-view synthesis: A reality check. In *NeurIPS*, 2022. [2](#), [3](#), [6](#), [7](#)
- [26] Klaus Greff, Francois Belletti, Lucas Beyer, Carl Doersch, Yilun Du, Daniel Duckworth, David J Fleet, Dan Gnanapragasam, Florian Golemo, Charles Herrmann, Thomas Kipf, Abhijit Kundu, Dmitry Lagun, Issam Laradji, Hsueh-Ti (Derek) Liu, Henning Meyer, Yishu Miao, Derek Nowrouzezahrai, Cengiz Oztireli, Etienne Pot, Noha Radwan, Daniel Rebain, Sara Sabour, Mehdi S. M. Sajjadi, Matan Sela, Vincent Sitzmann, Austin Stone, Deqing Sun, Suhani Vora, Ziyu Wang, Tianhao Wu, Kwang Moo Yi, Fangcheng Zhong, and Andrea Tagliasacchi. Kubric: a scalable dataset generator. In *CVPR*, 2022. [2](#), [6](#)
- [27] Xiuye Gu, Yijie Wang, Chongruo Wu, Yong Jae Lee, and Panqu Wang. Hplflownet: Hierarchical permutohedral lattice flownet for scene flow estimation on large-scale point clouds. In *CVPR*, pages 3254–3263, 2019. [3](#)
- [28] Adam W Harley, Zhaoyuan Fang, and Katerina Fragkiadaki. Particle video revisited: Tracking through occlusions using point trajectories. In *ECCV*, pages 59–75, 2022. [3](#)
- [29] Jixuan He, Chieh Hubert Lin, Lu Qi, and Ming-Hsuan Yang. Restage4d: Reanimating deformable 3d reconstruction from a single video. In *NeurIPS*, 2025. [3](#)
- [30] Yicong Hong, Kai Zhang, Jiuxiang Gu, Sai Bi, Yang Zhou, Difan Liu, Feng Liu, Kalyan Sunkavalli, Trung Bui, and Hao Tan. LRM: Large reconstruction model for single image to 3d. In *ICML*, 2024. [2](#), [3](#), [4](#), [5](#)
- [31] Junhwa Hur and Stefan Roth. Self-supervised monocular scene flow estimation. In *CVPR*, pages 7396–7405, 2020. [3](#)
- [32] Eddy Ilg, Nikolaus Mayer, Tonmoy Saikia, Margret Keuper, Alexey Dosovitskiy, and Thomas Brox. FlowNet 2.0: Evolution of optical flow estimation with deep networks. In *CVPR*, pages 2462–2470, 2017. [3](#)
- [33] Matthias Innmann, Michael Zollhöfer, Matthias Nießner, Christian Theobalt, and Marc Stamminger. Volumedeform: Real-time volumetric non-rigid reconstruction. In *ECCV*, pages 362–379, 2016. [2](#)
- [34] Mariano Jaimez, Mohamed Souiai, Javier Gonzalez-Jimenez, and Daniel Cremers. A primal-dual framework for real-time dense rgb-d scene flow. In *ICRA*, pages 98–104, 2015. [3](#)
- [35] Hanwen Jiang, Zexiang Xu, Desai Xie, Ziwen Chen, Haian Jin, Fujun Luan, Zhixin Shu, Kai Zhang, Sai Bi, Xin Sun, Jiuxiang Gu, Qixing Huang, Georgios Pavlakos, and Hao Tan. Megasynt: Scaling up 3d scene reconstruction with synthesized data. 2024. [2](#)
- [36] Dhiraj D. Kalamkar, Dheevatsa Mudigere, Naveen Mellempudi, Dipankar Das, Kunal Banerjee, Sasikanth Avancha, Dharma Teja Vooturi, Nataraj Jammalamadaka, Jianyu Huang, Hector Yuen, Jiyan Yang, Jongsoo Park, Alexander Heinecke, Evangelos Georganas, Sudarshan Srinivasan, Abhisek Kundu, Misha Smelyanskiy, Bharat Kaul, and Pradeep Dubey. A study of BFLOAT16 for deep learning training. *CoRR*, abs/1905.12322, 2019. [7](#)
- [37] Nikita Karaev, Ignacio Rocco, Benjamin Graham, Natalia Neverova, Andrea Vedaldi, and Christian Rupprecht. Cotracker: It is better to track together. In *ECCV*, pages 18–35, 2024. [2](#), [3](#), [7](#)

- [38] Bernhard Kerbl, Georgios Kopanas, Thomas Leimkühler, and George Drettakis. 3d gaussian splatting for real-time radiance field rendering. *ACM Transactions on Graphics*, 42(4):139–1, 2023. 2, 3
- [39] Alexander Kirillov, Eric Mintun, Nikhila Ravi, Hanzi Mao, Chloe Rolland, Laura Gustafson, Tete Xiao, Spencer Whitehead, Alexander C Berg, Wan-Yen Lo, et al. Segment anything. In *ICCV*, pages 4015–4026, 2023. 2
- [40] Johannes Kopf, Xuejian Rong, and Jia-Bin Huang. Robust consistent video depth estimation. In *CVPR*, pages 1611–1621, 2021. 3
- [41] Benjamin Lefaudeux, Francisco Massa, Diana Liskovich, Wenhan Xiong, Vittorio Caggiano, Sean Naren, Min Xu, Jieru Hu, Marta Tintore, Susan Zhang, Patrick Labatut, Daniel Haziza, Luca Wehrstedt, Jeremy Reizenstein, and Grigory Sizov. xformers: A modular and hackable transformer modelling library. <https://github.com/facebookresearch/xformers>, 2022. 7
- [42] Vincent Leroy, Yohann Cabon, and Jérôme Revaud. Grounding image matching in 3d with mast3r. In *ECCV*, pages 71–91, 2024. 2, 3
- [43] Jiahao Li, Hao Tan, Kai Zhang, Zexiang Xu, Fujun Luan, Yinghao Xu, Yicong Hong, Kalyan Sunkavalli, Greg Shakhnarovich, and Sai Bi. Instant3d: Fast text-to-3d with sparse-view generation and large reconstruction model. *arXiv preprint arXiv:2311.06214*, 2023. 3
- [44] Tianye Li, Mira Slavcheva, Michael Zollhoefer, Simon Green, Christoph Lassner, Changil Kim, Tanner Schmidt, Steven Lovegrove, Michael Goesele, Richard Newcombe, et al. Neural 3d video synthesis from multi-view video. In *CVPR*, pages 5521–5531, 2022. 3
- [45] Zhengqi Li, Simon Niklaus, Noah Snavely, and Oliver Wang. Neural scene flow fields for space-time view synthesis of dynamic scenes. In *CVPR*, pages 6498–6508, 2021. 3
- [46] Zhengqi Li, Qianqian Wang, Forrester Cole, Richard Tucker, and Noah Snavely. Dynibar: Neural dynamic image-based rendering. In *CVPR*, pages 4273–4284, 2023. 2, 3
- [47] Zhan Li, Zhang Chen, Zhong Li, and Yi Xu. Spacetime gaussian feature splatting for real-time dynamic view synthesis. In *CVPR*, pages 8508–8520, 2024. 3
- [48] Zhengqi Li, Richard Tucker, Forrester Cole, Qianqian Wang, Linyi Jin, Vickie Ye, Angjoo Kanazawa, Aleksander Holynski, and Noah Snavely. Megasam: Accurate, fast, and robust structure and motion from casual dynamic videos. *arXiv preprint arXiv:2412.04463*, 2024. 2, 3, 5, 6, 7
- [49] Hanxue Liang, Jiawei Ren, Ashkan Mirzaei, Antonio Torralba, Ziwei Liu, Igor Gilitschenski, Sanja Fidler, Cengiz Oztireli, Huan Ling, Zan Gojcic, et al. Feed-forward bullet-time reconstruction of dynamic scenes from monocular videos. *arXiv preprint arXiv:2412.03526*, 2024. 2, 3, 4
- [50] Xingyu Liu, Charles R Qi, and Leonidas J Guibas. FlowNet3D: Learning scene flow in 3d point clouds. In *CVPR*, pages 529–537, 2019. 3
- [51] Yu-Lun Liu, Chen Gao, Andreas Meuleman, Hung-Yu Tseng, Ayush Saraf, Changil Kim, Yung-Yu Chuang, Johannes Kopf, and Jia-Bin Huang. Robust dynamic radiance fields. In *CVPR*, pages 13–23, 2023. 3
- [52] Stephen Lombardi, Tomas Simon, Jason Saragih, Gabriel Schwartz, Andreas Lehrmann, and Yaser Sheikh. Neural volumes: Learning dynamic renderable volumes from images. *arXiv preprint arXiv:1906.07751*, 2019. 3
- [53] David G Lowe. Distinctive image features from scale-invariant keypoints. *IJCV*, 60:91–110, 2004. 3
- [54] Xuan Luo, Jia-Bin Huang, Richard Szeliski, Kevin Matzen, and Johannes Kopf. Consistent video depth estimation. *ACM Transactions on Graphics*, 39(4):71–1, 2020. 3
- [55] Ben Mildenhall, Pratul P Srinivasan, Matthew Tancik, Jonathan T Barron, Ravi Ramamoorthi, and Ren Ng. Nerf: Representing scenes as neural radiance fields for view synthesis. *Communications of the ACM*, 65(1):99–106, 2021. 2, 3
- [56] Michal Neoral, Jonáš Šerých, and Jiří Matas. Mft: Long-term tracking of every pixel. In *WACV*, pages 6837–6847, 2024. 3
- [57] Richard A Newcombe, Dieter Fox, and Steven M Seitz. Dynamicfusion: Reconstruction and tracking of non-rigid scenes in real-time. In *CVPR*, pages 343–352, 2015. 2

- [58] Keunhong Park, Utkarsh Sinha, Jonathan T Barron, Sofien Bouaziz, Dan B Goldman, Steven M Seitz, and Ricardo Martin-Brualla. Nerfies: Deformable neural radiance fields. In *ICCV*, pages 5865–5874, 2021. 3
- [59] Keunhong Park, Utkarsh Sinha, Peter Hedman, Jonathan T Barron, Sofien Bouaziz, Dan B Goldman, Ricardo Martin-Brualla, and Steven M Seitz. Hypernerf: A higher-dimensional representation for topologically varying neural radiance fields. *arXiv preprint arXiv:2106.13228*, 2021. 3
- [60] 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 *CVPR*, pages 724–732, 2016. 2
- [61] Luigi Piccinelli, Yung-Hsu Yang, Christos Sakaridis, Mattia Segu, Siyuan Li, Luc Van Gool, and Fisher Yu. Unidepth: Universal monocular metric depth estimation. In *CVPR*, pages 10106–10116, 2024. 5
- [62] Adam Polyak, Amit Zohar, Andrew Brown, Andros Tjandra, Animesh Sinha, Ann Lee, Apoorv Vyas, Bowen Shi, Chih-Yao Ma, Ching-Yao Chuang, David Yan, Dhruv Choudhary, DingKang Wang, Geet Sethi, Guan Pang, Haoyu Ma, Ishan Misra, Ji Hou, Jialiang Wang, Kiran Jagadeesh, Kunpeng Li, Luxin Zhang, Mannat Singh, Mary Williamson, Matt Le, Matthew Yu, Mitesh Kumar Singh, Peizhao Zhang, Peter Vajda, Quentin Duval, Rohit Girdhar, Roshan Sumbaly, Sai Saketh Rambhatla, Sam Tsai, Samaneh Azadi, Samyak Datta, Sanyuan Chen, Sean Bell, Sharadh Ramaswamy, Shelly Sheynin, Siddharth Bhattacharya, Simran Motwani, Tao Xu, Tianhe Li, Tingbo Hou, Wei-Ning Hsu, Xi Yin, Xiaoliang Dai, Yaniv Taigman, Yaqiao Luo, Yen-Cheng Liu, Yi-Chiao Wu, Yue Zhao, Yuval Kirstain, Zecheng He, Zijian He, Albert Pumarola, Ali Thabet, Arsiom Sanakoyeu, Arun Mallya, Baishan Guo, Boris Araya, Breena Kerr, Carleigh Wood, Ce Liu, Cen Peng, Dimitry Vengertsev, Edgar Schonfeld, Elliot Blanchard, Felix Juefei-Xu, Fraylie Nord, Jeff Liang, John Hoffman, Jonas Kohler, Kaolin Fire, Karthik Sivakumar, Lawrence Chen, Licheng Yu, Luya Gao, Markos Georgopoulos, Rashel Moritz, Sara K. Sampson, Shikai Li, Simone Parmeggiani, Steve Fine, Tara Fowler, Vladan Petrovic, and Yuming Du. Movie gen: A cast of media foundation models, 2025. 2, 4
- [63] Gilles Puy, Alexandre Boulch, and Renaud Marlet. Flot: Scene flow on point clouds guided by optimal transport. In *ECCV*, pages 527–544, 2020. 3
- [64] Julian Quiroga, Thomas Brox, Frédéric Devernay, and James Crowley. Dense semi-rigid scene flow estimation from rgbd images. In *ECCV*, pages 567–582, 2014. 3
- [65] Alec Radford, Karthik Narasimhan, Tim Salimans, and Ilya Sutskever. Improving language understanding by generative pre-training, 2018. 2
- [66] René Ranftl, Alexey Bochkovskiy, and Vladlen Koltun. Vision transformers for dense prediction. In *ICCV*, pages 12179–12188, 2021. 2
- [67] Jiawei Ren, Cheng Xie, Ashkan Mirzaei, Karsten Kreis, Ziwei Liu, Antonio Torralba, Sanja Fidler, Seung Wook Kim, Huan Ling, et al. L4gm: Large 4d gaussian reconstruction model. In *NeurIPS*, pages 56828–56858, 2024. 2, 3, 7, 8
- [68] Michael Rubinstein, Ce Liu, and William T Freeman. Towards longer long-range motion trajectories. In *BMVC*, 2012. 3
- [69] Ethan Rublee, Vincent Rabaud, Kurt Konolige, and Gary Bradski. Orb: An efficient alternative to sift or surf. In *ICCV*, pages 2564–2571, 2011. 3
- [70] Tim Salimans and Durk P Kingma. Weight normalization: A simple reparameterization to accelerate training of deep neural networks. In *NeurIPS*, 2016. 5
- [71] Peter Sand and Seth Teller. Particle video: Long-range motion estimation using point trajectories. *IJCV*, 80:72–91, 2008. 3
- [72] Liangchen Song, Anpei Chen, Zhong Li, Zhang Chen, Lele Chen, Junsong Yuan, Yi Xu, and Andreas Geiger. Nerfplayer: A streamable dynamic scene representation with decomposed neural radiance fields. *IEEE Transactions on Visualization and Computer Graphics*, 29(5):2732–2742, 2023. 3
- [73] Timo Stich, Christian Linz, Georgia Albuquerque, and Marcus Magnor. View and time interpolation in image space. In *Computer Graphics Forum*, pages 1781–1787, 2008. 3
- [74] Deqing Sun, Erik B Sudderth, and Hanspeter Pfister. Layered rgbd scene flow estimation. In *CVPR*, pages 548–556, 2015. 3
- [75] Zachary Teed and Jia Deng. Raft: Recurrent all-pairs field transforms for optical flow. In *ECCV*, pages 402–419, 2020. 2, 3

- [76] Zachary Teed and Jia Deng. Raft-3d: Scene flow using rigid-motion embeddings. In *CVPR*, pages 8375–8384, 2021. [2](#), [3](#), [7](#)
- [77] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *NeurIPS*, 2017. [2](#), [5](#)
- [78] Chaoyang Wang, Ben Eckart, Simon Lucey, and Orazio Gallo. Neural trajectory fields for dynamic novel view synthesis. *arXiv preprint arXiv:2105.05994*, 2021. [3](#)
- [79] Liao Wang, Jiakai Zhang, Xinhang Liu, Fuqiang Zhao, Yanshun Zhang, Yingliang Zhang, Minye Wu, Jingyi Yu, and Lan Xu. Fourier plenotrees for dynamic radiance field rendering in real-time. In *CVPR*, pages 13524–13534, 2022. [3](#)
- [80] Peng Wang, Hao Tan, Sai Bi, Yinghao Xu, Fujun Luan, Kalyan Sunkavalli, Wenping Wang, Zexiang Xu, and Kai Zhang. Pf-lrm: Pose-free large reconstruction model for joint pose and shape prediction. *arXiv preprint arXiv:2311.12024*, 2023. [3](#)
- [81] Qianqian Wang, Zhicheng Wang, Kyle Genova, Pratul P Srinivasan, Howard Zhou, Jonathan T Barron, Ricardo Martin-Brualla, Noah Snavely, and Thomas Funkhouser. Ibrnet: Learning multi-view image-based rendering. In *CVPR*, pages 4690–4699, 2021. [3](#)
- [82] Qianqian Wang, Yen-Yu Chang, Ruojin Cai, Zhengqi Li, Bharath Hariharan, Aleksander Holynski, and Noah Snavely. Tracking everything everywhere all at once. In *ICCV*, pages 19795–19806, 2023. [3](#)
- [83] 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. [2](#), [3](#), [8](#)
- [84] Qianqian Wang, Yifei Zhang, Aleksander Holynski, Alexei A. Efros, and Angjoo Kanazawa. Continuous 3d perception model with persistent state, 2025. [3](#)
- [85] Shuzhe Wang, Vincent Leroy, Yohann Cabon, Boris Chidlovskii, and Jerome Revaud. Dust3r: Geometric 3d vision made easy. In *CVPR*, pages 20697–20709, 2024. [2](#), [3](#)
- [86] Zirui Wang, Shuda Li, Henry Howard-Jenkins, Victor Prisacariu, and Min Chen. Flownet3d++: Geometric losses for deep scene flow estimation. In *CVPR*, pages 91–98, 2020. [3](#)
- [87] Xinyue Wei, Kai Zhang, Sai Bi, Hao Tan, Fujun Luan, Valentin Deschaintre, Kalyan Sunkavalli, Hao Su, and Zexiang Xu. Meshlrm: Large reconstruction model for high-quality mesh. *arXiv preprint arXiv:2404.12385*, 2024. [3](#)
- [88] Guanjuan Wu, Taoran Yi, Jiemin Fang, Lingxi Xie, Xiaopeng Zhang, Wei Wei, Wenyu Liu, Qi Tian, and Xinggang Wang. 4d gaussian splatting for real-time dynamic scene rendering. In *CVPR*, pages 20310–20320, 2024. [2](#), [3](#)
- [89] Wenqi Xian, Jia-Bin Huang, Johannes Kopf, and Changil Kim. Space-time neural irradiance fields for free-viewpoint video. In *CVPR*, pages 9421–9431, 2021. [3](#)
- [90] Yuxi Xiao, Qianqian Wang, Shangzhan Zhang, Nan Xue, Sida Peng, Yujun Shen, and Xiaowei Zhou. Spatialtracker: Tracking any 2d pixels in 3d space. In *CVPR*, pages 20406–20417, 2024. [2](#), [3](#), [7](#), [8](#), [9](#)
- [91] Desai Xie, Sai Bi, Zhixin Shu, Kai Zhang, Zexiang Xu, Yi Zhou, Sören Pirk, Arie Kaufman, Xin Sun, and Hao Tan. Lrm-zero: Training large reconstruction models with synthesized data. *arXiv preprint arXiv:2406.09371*, 2024. [2](#)
- [92] Qiangeng Xu, Zexiang Xu, Julien Philip, Sai Bi, Zhixin Shu, Kalyan Sunkavalli, and Ulrich Neumann. Point-nerf: Point-based neural radiance fields. In *CVPR*, pages 5438–5448, 2022. [3](#)
- [93] Yinghao Xu, Hao Tan, Fujun Luan, Sai Bi, Peng Wang, Jiahao Li, Zifan Shi, Kalyan Sunkavalli, Gordon Wetzstein, Zexiang Xu, et al. Dmv3d: Denoising multi-view diffusion using 3d large reconstruction model. *arXiv preprint arXiv:2311.09217*, 2023. [3](#)
- [94] Yinghao Xu, Zifan Shi, Wang Yifan, Hansheng Chen, Ceyuan Yang, Sida Peng, Yujun Shen, and Gordon Wetzstein. Grm: Large gaussian reconstruction model for efficient 3d reconstruction and generation. In *ECCV*, pages 1–20, 2024. [2](#), [3](#), [5](#)
- [95] Lihe Yang, Bingyi Kang, Zilong Huang, Xiaogang Xu, Jiashi Feng, and Hengshuang Zhao. Depth anything: Unleashing the power of large-scale unlabeled data. In *CVPR*, pages 10371–10381, 2024. [2](#)
- [96] Zeyu Yang, Hongye Yang, Zijie Pan, and Li Zhang. Real-time photorealistic dynamic scene representation and rendering with 4d gaussian splatting. *arXiv preprint arXiv:2310.10642*, 2023. [3](#)

- [97] Ziyi Yang, Xinyu Gao, Wen Zhou, Shaohui Jiao, Yuqing Zhang, and Xiaogang Jin. Deformable 3d gaussians for high-fidelity monocular dynamic scene reconstruction. In *CVPR*, pages 20331–20341, 2024. [2](#), [3](#), [7](#), [8](#)
- [98] Junqi You, Chieh Hubert Lin, Weijie Lyu, Zhengbo Zhang, and Ming-Hsuan Yang. Instainpaint: Instant 3d-scene inpainting with masked large reconstruction model. In *NeurIPS*, 2025. [3](#)
- [99] Alex Yu, Vickie Ye, Matthew Tancik, and Angjoo Kanazawa. pixelnerf: Neural radiance fields from one or few images. In *CVPR*, pages 4578–4587, 2021. [3](#)
- [100] 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. [2](#), [3](#)
- [101] Kai Zhang, Nick Kolkin, Sai Bi, Fujun Luan, Zexiang Xu, Eli Shechtman, and Noah Snavely. Arf: Artistic radiance fields. In *ECCV*, page 717–733, Berlin, Heidelberg, 2022. [7](#)
- [102] Kai Zhang, Sai Bi, Hao Tan, Yuanbo Xiangli, Nanxuan Zhao, Kalyan Sunkavalli, and Zexiang Xu. Gs-lrm: Large reconstruction model for 3d gaussian splatting. In *ECCV*, pages 1–19, 2024. [2](#), [3](#), [5](#)
- [103] Richard Zhang, Phillip Isola, Alexei A Efros, Eli Shechtman, and Oliver Wang. The unreasonable effectiveness of deep features as a perceptual metric. In *CVPR*, 2018. [5](#)
- [104] Zhoutong Zhang, Forrester Cole, Richard Tucker, William T Freeman, and Tali Dekel. Consistent depth of moving objects in video. *ACM Transactions on Graphics*, 40(4):1–12, 2021. [3](#)
- [105] Zhoutong Zhang, Forrester Cole, Zhengqi Li, Michael Rubinstein, Noah Snavely, and William T Freeman. Structure and motion from casual videos. In *ECCV*, pages 20–37, 2022. [3](#)
- [106] Xiaoming Zhao, R Alex Colburn, Fangchang Ma, Miguel Ángel Bautista, Joshua M. Susskind, and Alex Schwing. Pseudo-generalized dynamic view synthesis from a video. In *ICLR*, 2024. [7](#), [8](#)
- [107] Yang Zheng, Adam W. Harley, Bokui Shen, Gordon Wetzstein, and Leonidas J. Guibas. Pointodyssey: A large-scale synthetic dataset for long-term point tracking. In *ICCV*, 2023. [2](#), [7](#), [8](#)
- [108] Michael Zollhöfer, Matthias Nießner, Shahram Izadi, Christoph Rehmann, Christopher Zach, Matthew Fisher, Chenglei Wu, Andrew Fitzgibbon, Charles Loop, Christian Theobalt, et al. Real-time non-rigid reconstruction using an rgb-d camera. *ACM Transactions on Graphics*, 33(4):1–12, 2014. [2](#)

NeurIPS Paper Checklist

1. Claims

Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

Answer: [\[Yes\]](#)

Justification: We propose a feed-forward framework for deformable 3DGS reconstruction. We benchmark our model on the commonly used benchmarks to show its effectiveness and generalization. We also evaluate the scene flow quality on a benchmark to show the quality and continuity of our deformation.

Guidelines:

- The answer NA means that the abstract and introduction do not include the claims made in the paper.
- The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A No or NA answer to this question will not be perceived well by the reviewers.
- The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
- It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. Limitations

Question: Does the paper discuss the limitations of the work performed by the authors?

Answer: [\[Yes\]](#)

Justification: We include such a discussion in the last section.

Guidelines:

- The answer NA means that the paper has no limitation while the answer No means that the paper has limitations, but those are not discussed in the paper.
- The authors are encouraged to create a separate "Limitations" section in their paper.
- The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
- The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
- The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
- The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
- If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
- While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. Theory assumptions and proofs

Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

Answer: [NA]

Justification: This paper does not propose new theories or mathematical proofs.

Guidelines:

- The answer NA means that the paper does not include theoretical results.
- All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
- All assumptions should be clearly stated or referenced in the statement of any theorems.
- The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
- Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
- Theorems and Lemmas that the proof relies upon should be properly referenced.

4. Experimental result reproducibility

Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

Answer: [Yes]

Justification: We reported all the architectural design, training data creation, and hyperparameters with our best effort. Other unspecified hyperparameters follow the originally released codes or papers of the cited works.

Guidelines:

- The answer NA means that the paper does not include experiments.
- If the paper includes experiments, a No answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
- If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
- Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general, releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
- While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
 - (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
 - (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
 - (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
 - (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. Open access to data and code

Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

Answer: [No]

Justification: The code release requires additional review. We will put our best effort into releasing the codes and the pretrained model.

Guidelines:

- The answer NA means that paper does not include experiments requiring code.
- Please see the NeurIPS code and data submission guidelines (<https://nips.cc/public/guides/CodeSubmissionPolicy>) for more details.
- While we encourage the release of code and data, we understand that this might not be possible, so “No” is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
- The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (<https://nips.cc/public/guides/CodeSubmissionPolicy>) for more details.
- The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
- The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
- At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
- Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. Experimental setting/details

Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer, etc.) necessary to understand the results?

Answer: [Yes]

Justification: Yes, we put our best effort into including all the implementation and experiment setting details. We followed the previous proposed evaluation protocol as closely as possible.

Guidelines:

- The answer NA means that the paper does not include experiments.
- The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
- The full details can be provided either with the code, in appendix, or as supplemental material.

7. Experiment statistical significance

Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

Answer: [No]

Justification: Our framework involves large-scale and resource-intensive training. It is infeasible to report a meaningful error margin with multiple individually trained models.

Guidelines:

- The answer NA means that the paper does not include experiments.
- The authors should answer "Yes" if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.

- The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
- The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
- The assumptions made should be given (e.g., Normally distributed errors).
- It should be clear whether the error bar is the standard deviation or the standard error of the mean.
- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g. negative error rates).
- If error bars are reported in tables or plots, The authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. Experiments compute resources

Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

Answer: [Yes]

Justification: Yes, we report the GPUs for training the model and execution time. However, we do not keep track of the past experiments, thus challenging to report meaningful total resource consumption throughout the project.

Guidelines:

- The answer NA means that the paper does not include experiments.
- The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
- The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
- The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. Code of ethics

Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics <https://neurips.cc/public/EthicsGuidelines>?

Answer: [Yes]

Justification: We put our best effort into complying with NeurIPS Code of Ethics and all privacy concerns.

Guidelines:

- The answer NA means that the authors have not reviewed the NeurIPS Code of Ethics.
- If the authors answer No, they should explain the special circumstances that require a deviation from the Code of Ethics.
- The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. Broader impacts

Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

Answer: [NA]

Justification: Our work improves the inference speed of deformable 3D reconstruction, which aims to recreate a real-world environment. Our method does not allow altering content or spreading fake information. We comply with privacy considerations, our training data does not include human or unlicensed data.

Guidelines:

- The answer NA means that there is no societal impact of the work performed.
- If the authors answer NA or No, they should explain why their work has no societal impact or why the paper does not address societal impact.
- Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
- The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. Safeguards

Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pretrained language models, image generators, or scraped datasets)?

Answer: [NA]

Justification: Our dataset is purely synthetic without unlicensed content. Our model is a reconstruction model that preserves the real-world information without any alteration.

Guidelines:

- The answer NA means that the paper poses no such risks.
- Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
- Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
- We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. Licenses for existing assets

Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

Answer: [Yes]

Justification: We utilize pure synthetic data during training. The Google Scanned Objects (GSO) assets are licensed under CC-BY 4.0. The PolyHaven HDRI is licensed under CC0.

Guidelines:

- The answer NA means that the paper does not use existing assets.
- The authors should cite the original paper that produced the code package or dataset.
- The authors should state which version of the asset is used and, if possible, include a URL.

- The name of the license (e.g., CC-BY 4.0) should be included for each asset.
- For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
- If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, paperswithcode.com/datasets has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
- For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
- If this information is not available online, the authors are encouraged to reach out to the asset’s creators.

13. **New assets**

Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

Answer: [NA]

Justification: We only provide alterations to an existing simulation engine. We do not include additional data or propose new datasets.

Guidelines:

- The answer NA means that the paper does not release new assets.
- Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
- The paper should discuss whether and how consent was obtained from people whose asset is used.
- At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and research with human subjects**

Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

Answer: [NA]

Justification: We did not use human-related data for model training.

Guidelines:

- The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
- Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
- According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

Answer: [NA]

Justification: Our research does not involve crowdsourcing nor research with human subjects.

Guidelines:

- The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.

- Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
- We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
- For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. **Declaration of LLM usage**

Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does not impact the core methodology, scientific rigorousness, or originality of the research, declaration is not required.

Answer: [NA]

Justification: We did not use LLM in developing the research. The only potential use of LLM is grammatical error correction after writing is completed.

Guidelines:

- The answer NA means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
- Please refer to our LLM policy (<https://neurips.cc/Conferences/2025/LLM>) for what should or should not be described.