# DySurface: Consistent 4D Surface Reconstruction via Bridging Explicit Gaussians and Implicit Functions

**Minje Kim**[1] **Younghyun Noh**[2] **Jaesoon Kim**[1] **Tae-Kyun Kim**[1]

[1]KAIST [2]KT

{minjekim, kimtaekyun}@kaist.ac.kr younghyun.noh@kt.com winnercup@g.skku.edu

## Abstract

While novel view synthesis (NVS) for dynamic scenes has seen significant progress, reconstructing temporally consistent geometric surfaces remains a challenge. Neural Radiance Fields (NeRF) and 3D Gaussian Splatting (3DGS) offer powerful dynamic scene rendering capabilities; however, relying solely on photometric optimization often leads to geometric ambiguities. This results in discontinuous surfaces, severe artifacts, and broken surfaces over time. To address these limitations, we present DySurface, a novel framework that bridges the effectiveness of explicit Gaussians with the geometric fidelity of implicit Signed Distance Functions (SDFs) in dynamic scenes. Our approach tackles the structural discrepancy between the forward deformation of 3DGS ($canonical \rightarrow dynamic$) and the backward deformation required for volumetric SDF rendering ($dynamic \rightarrow canonical$). Specifically, we propose the VoxGS-DSDF branch that leverages deformed Gaussians to construct a dynamic sparse voxel grid, providing explicit geometric guidance to the implicit SDF field. This explicit anchoring effectively regularizes the volumetric rendering process, significantly improving surface reconstruction quality, with watertight boundaries and detailed representations. Quantitative and qualitative experiments demonstrate that DySurface significantly outperforms state-of-the-art baselines in geometric accuracy while maintaining competitive rendering performance. Project Page: `https://yunminjin2.github.io/projects/dysurface`.

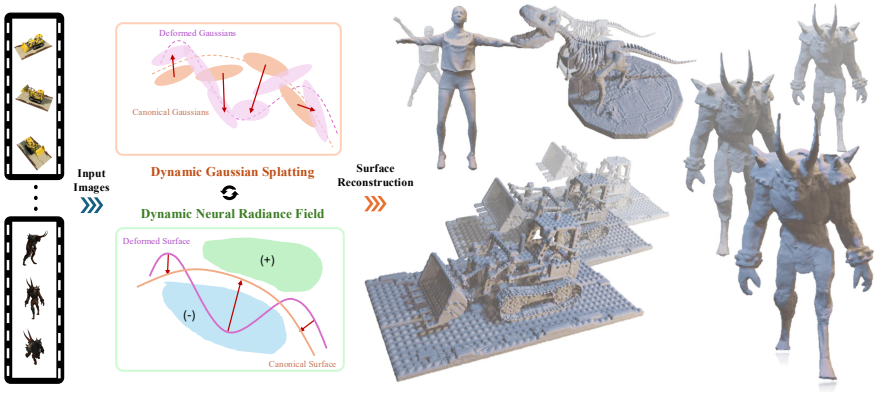



Figure 1: Given a video sequence, DySurface obtains temporally consistent, high-quality meshes by bridging between explicit Gaussian primitives and implicit continuous surface functions.

## 1 Introduction

3D object reconstruction remains an important task in computer vision. Neural Radiance Fields (NeRF) [31] and 3D Gaussian Splatting (3DGS) [21] have significantly advanced the field for static

40th Conference on Neural Information Processing Systems (NeurIPS 2026).

scenes, achieving high-fidelity rendering and accurate geometry. Recently, numerous works [10, 16, 27, 39, 40] have extended these representations to dynamic scenes, learning object deformations through the rendering process.

While prior methods effectively model temporal dynamics to achieve high photometric quality, they often fail to preserve fine-grained geometric details. NeRF-based approaches [36, 10, 34, 33, 17, 12] exhibit relatively low rendering quality and limited capability in recovering high-frequency geometry. In contrast, Gaussian Splatting-based methods [29, 45, 7, 19, 53] achieve photorealistic rendering and maintain real-time synthesis speeds by explicitly modeling geometric deformations. However, direct visualization of optimized Gaussians reveals significant artifacts, such as floaters or primitives detached from accurate surface geometry.

It is crucial to note that recovering explicit, deformable geometric surfaces is a fundamentally more demanding task than dynamic novel view synthesis (NVS). While dynamic NVS primarily requires photometric consistency across viewpoints, its surface reconstruction demands strict geometric and topological fidelity. Consequently, discrete representations optimized solely for view synthesis, such as dynamic 3DGS, fail to extract accurate meshes. These limitations necessitate a new paradigm that explicitly constructs discrete deformations with continuous surface regularizations.

Extracting meshes from pretrained deformable Gaussian Splatting models is non-trivial; the lack of explicit surface constraints often leads to geometric inconsistencies, resulting in noisy and blurred mesh reconstructions. DG-Mesh [26] tackles the task of reconstructing surfaces from dynamic scenes. However, since it extracts meshes directly from discrete Gaussian Splatting via ad-hoc algorithms (e.g., Poisson reconstruction [20]), it inherently lacks continuous surface constraints. This structural limitation results in blurry, fragmented, and inaccurate geometry under severe non-rigid deformations.

To address these challenges, we propose DySurface, a novel framework to extract highly accurate, temporally consistent deformable meshes from dynamic scenes. We tightly integrate explicit Gaussian primitives, implicit Signed Distance Functions (SDF), and explicit mesh representations into a unified pipeline. Specifically, our novel VoxGS-DSDF (Voxelized GS-driven Dynamic SDF) branch learns the continuous zero-level set of the object surface from the explicit geometric and appearance priors provided by the deformed Gaussians. Finally, we refine the extracted dynamic mesh by transferring the pretrained spatial transformation field from the GS branch, bridging the domain gap between the volumetric and explicit surface rendering. In summary, our main contributions are as follows:

- We propose DySurface, a tightly coupled framework that bridges the gap between discrete Gaussians, implicit continuous SDFs, and explicit meshes as mutually complementary representations.
- We introduce a novel VoxGS-DSDF branch, which explicitly anchors the dynamic implicit surface using sparse voxel latents derived from deformed Gaussian splatting, effectively resolving geometric ambiguities.
- Experiments demonstrate that DySurface achieves state-of-the-art geometric accuracies, reconstructing temporally consistent 4D meshes while maintaining rendering quality.

## 2 Related Work

### 2.1 Dynamic View Synthesis

Recent novel view synthesis (NVS) [2, 8, 14, 23, 37] has extended from static to dynamic scenes, where the primary challenge lies in accurately modeling object deformations. Among several works [36, 10, 34, 11, 4, 25, 9], D-NeRF [36] introduced a canonical neural radiance field with a continuous backward deformation network. To overcome the slow convergence of original NeRFs and handle more extreme dynamic changes, subsequent works such as TiNeuVox [10] and HyperNeRF [34] proposed utilizing explicit spatiotemporal feature grids and mapping deformations into higher-dimensional spaces, respectively. Despite achieving high photometric quality, NeRF-based dynamic models struggle to capture fine geometric details due to their reliance on backward mapping [17], learning non-smooth and discontinuous deformation fields. Furthermore, HexPlane [4] and K-Plane [11] project 4D spatial-temporal space to multiple 2D planes; thereby losing 3D features.

To overcome the limitations of implicit fields, recent literature [45, 29, 54, 22, 46, 3] has extended the explicit 3DGS [21] framework to the deformable domain. 4D-GS [45] achieves real-time dynamic

synthesis by seamlessly integrating temporal attributes into the 3DGS, modeling continuous motion via spatial-temporal HexPlanes. Similarly, GaGS [29] explicitly learns the non-rigid deformation of Gaussians, mapping with a sparse voxel grid, achieving high fidelity of rendering quality. While these GS-based methods establish the photometric rendering quality, they optimize strictly discrete, unconstrained primitives. Consequently, they are limited to view synthesis; without explicit surface regularizations, they fail to yield continuous, watertight geometric surfaces.

### 2.2 Surface Reconstruction

**Static Scenes.** Traditional volume rendering lacks continuous geometric constraints, making accurate mesh extraction non-trivial. To address this, numerous works [41, 42, 47, 24, 50, 32, 48, 51, 38] have incorporated SDF into the neural rendering pipeline. Specifically, NeuS [41] and VolSDF [47] successfully model solid surfaces by converting SDF values into density distributions for volume rendering. More recently, Neuralangelo [24] utilizes multi-resolution hash grids to recover high-frequency geometric details. However, these NeRF-based implicit models are constrained to static scenes; extending them to dynamic environments while preserving highly accurate geometric details remains a formidable challenge due to the complex, non-rigid topological variations over time. While 3DGS [21] offers a highly efficient representation, 3DGS optimizes discrete primitives driven purely by a photometric loss, thus lacking continuous surface constraints. Recent literature [15, 44, 1, 18, 5, 6, 49, 30] has attempted to extract surfaces from this discrete representation. SuGaR [15] introduces regularization terms to align Gaussians with the underlying geometry. Alternatively, GS2Mesh [44] avoids direct extraction by leveraging a stereo-matching network to generate depth maps from novel views, while 3DGS-to-PC [1] proposes probabilistically sampling dense point clouds from 3DGS. Despite these efforts, these methods rely on ad-hoc post-processing (e.g., Poisson reconstruction) or external 2D priors, thus struggle to maintain a globally consistent zero-level set, often yielding fragmented reconstructions that lack topological coherence, particularly in regions where the discrete Gaussian representation deviates from the actual surface geometry.

To harness both the explicit 3DGS and the implicit geometric fidelity of SDF, recent work [49, 55, 30] has explored hybrid architectures. Notably, GSDF [49] introduces a dual-branch framework that jointly optimizes an explicit 3DGS branch and an implicit SDF branch, employing mutual spatial guidance to enhance both view synthesis and surface reconstruction. While achieving state-of-the-art 3D geometry results, a major limitation of GSDF is its confinement to static scenes. The mutual guidance mechanism assumes a static coordinate space and lacks temporal deformation modeling. Therefore, it fails to reconstruct non-rigid, deformable changes of the object over time.

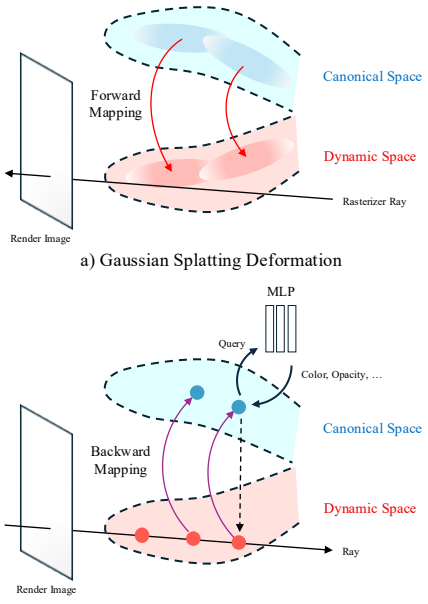



Figure 2: Illustration of forward mapping from Gaussian Splatting and backward mapping from neural radiance field.

**Dynamic Scenes.** While surface reconstruction for static scenes has been extensively studied, it remains under-explored in dynamic environments. For instance, DG-Mesh [26] attempts to recover the surfaces of deformable objects by extracting a preliminary mesh from 3D Gaussian Splatting via Poisson surface reconstruction, followed by a refinement stage. However, applying Poisson reconstruction to unconstrained, discrete Gaussians intrinsically fails to preserve geometric accuracy. D-2DGS [53] also extracts a deformable mesh from 2DGS [18]; frame-by-frame extraction does not guarantee geometric consistency. NeRF-based methods [36, 10, 34] extract surfaces by thresholding volume density; they still suffer from severe geometric ambiguity and lack of fidelity due to the absence of explicit surface regularization. In contrast, our approach achieves state-of-the-art mesh extraction accuracy by synergizing the explicit and efficient geometric deformation learned by 3D Gaussians with precise SDF fields optimized via volume rendering.

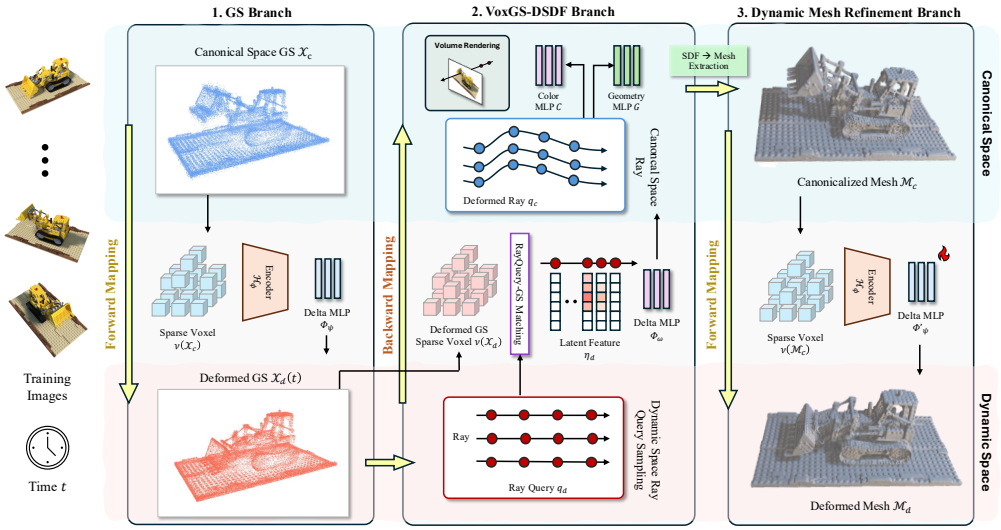



Figure 3: **Architecture of DySurface.** Our framework consists of three branches, learned in a stage-wise manner. (1) GS branch deforms canonical 3DGS into dynamic space using a transform field. (2) VoxGS-DSDF branch anchors the implicit SDF field to the deformed GS voxels through RayQuery-GS matching, resolving the forward-backward mapping gap while volume rendering. (3) The final branch extracts the canonical surface from the learned SDF and refines the transform field.

## 3 Preliminary

**Deformation Formulations: Forward vs. Backward Mapping.** In modeling dynamic scenes, 3D Gaussian Splatting (3DGS) [21] and Neural Radiance Fields (NeRF) [31] adopt opposite deformation directions. As illustrated in Fig. 2, dynamic 3DGS employs a forward mapping framework. It explicitly transforms the geometric attributes (e.g., positions, rotations) of discrete Gaussians from a canonical space to the dynamic observation space ($T_{canonical \to dynamic}$). Conversely, NeRF relies on a backward mapping approach, since it synthesizes images by casting rays into the dynamic observation space. It warps the sampled 3D points along the rays back to a canonical space ($T_{dynamic \to canonical}$) to query continuous volume density and radiance.

This inherent distinction arises from their respective rendering processes: NeRF queries a continuous static field via backward mapping, whereas 3DGS relies on the explicit forward rasterization of moving discrete primitives. Consequently, directly applying SDF-based surface regularization to dynamic Gaussian Splatting is a non-trivial challenge, as it requires resolving the structural conflict between forward-mapped dynamics and backward-mapped implicit surfaces.

## 4 Method

In this section, we introduce DySurface, a method for reconstructing a detailed deformable mesh from a dynamic scene. Our method is constructed with three stages: i) the Gaussian Splatting (GS) branch (Sec. 4.1), ii) the VoxGS-DSDF branch (Sec. 4.2), and iii) the Dynamic-Mesh Refinement branch (Sec. 4.3). The overall framework is illustrated in Fig. 3

### 4.1 Gaussian Splatting Branch

Following GaGS [29], our Gaussian Splatting (GS) branch explicitly models the temporal deformation of canonical 3D Gaussian primitives $\mathcal{X}_c$, parameterized by mean position $\mu_c \in \mathbb{R}^3$, scale $s_c \in \mathbb{R}^3$, and rotation $r_c \in \mathbb{R}^4$, along with time-invariant opacity $\alpha$ and SH coefficients.

To model non-rigid dynamics, we define a forward transformation field $\Delta\overrightarrow{q}_{fwd}$. We first discretize the canonical gaussians $\mathcal{X}_c$ into a sparse voxel grid $\nu(\mathcal{X}_c)$ to extract a global feature volume via a 3D sparse convolutional encoder $\mathcal{H}_\phi$. For any coordinate $x$, its queried global latent $\mathbf{f}_{global}(x)$ is concatenated with a point-wise local feature $\mathbf{f}_{local}(x)$. An MLP decoder $\Phi_\psi$ then predicts the

time-dependent residuals, formulating the transformation field $\mathcal{T}_\theta$ (parameterized by $\theta = \{\phi, \psi\}$) as:

$$\mathcal{T}_\theta(x, t) = \Phi_\psi(\text{Concat}(\mathbf{f}_{global}(x), \mathbf{f}_{local}(x)), t) \tag{1}$$

Applying these residuals yields the explicitly deformed attributes for the $k$-th Gaussian at time $t$. We denote the per-component output of the transformation field as $\mathcal{T}_\theta(\mu_{c,k}, t) = (\Delta\mu_k, \Delta s_k, \Delta r_k)$:

$$\mu_{d,k}(t) = \mu_{c,k} + \Delta\mu_k, \quad s_{d,k}(t) = s_{c,k} + \Delta s_k, \quad r_{d,k}(t) = \frac{r_{c,k} + \Delta r_k}{\|r_{c,k} + \Delta r_k\|} \tag{2}$$

The resulting deformed primitives, denoted as $\mathcal{X}_d$, synthesize the novel view at time $t$ and serve as the geometric prior for the VoxGS-DSDF branch.

### 4.2 Voxelized GS-driven Dynamic SDF Branch

The Voxelized GS-driven Dynamic SDF (VoxGS-DSDF) branch learns a high-fidelity continuous SDF in the canonical space. To overcome the lack of geometric guidance in implicit methods, we leverage the deformed Gaussians $\mathcal{X}_d(t) = \{x_d^i(t)\}_{i=1}^{N_G}$ from the GS branch to construct a dynamic sparse voxel grid, anchoring the continuous volumetric field to explicit geometric priors, where $x_d^i(t)$ denotes the $i$-th deformed Gaussian primitive.

As detailed in Sec. 3, the rendering begins by sampling points $q_d$ along rays in the dynamic space. To extract spatial context for each query point $q_d$, we perform a K-Nearest Neighbors (KNN) search to identify the explicitly deformed Gaussian sparse voxels $\nu(\mathcal{X}_d(t))$. Let $\mathcal{N}_K(q_d; \nu(\mathcal{X}_d(t)))$ denote the initial set of the $K$-nearest voxel neighbors of $q_d$. To filter out irrelevant features, we apply a spatial distance threshold $\tau$, retaining only the valid subset of neighbors $\mathcal{N}'(q_d; \nu(\mathcal{X}_d(t))) = \{k \in \mathcal{N}_K(q_d; \nu(\mathcal{X}_d(t))) \mid \|v_k - q_d\|_2^2 < \tau\}$, where $v_k$ is the center position of the $k$-th neighbor in $\nu(\mathcal{X}_d(t))$. From these $K' = |\mathcal{N}'(q_d; \nu(\mathcal{X}_d(t)))|$ valid neighbors, we extract geometric features $\mathbf{f}_{geo,k}$ from the deformed mean positions and appearance features $\mathbf{f}_{app,k}$ from the embedded color attributes. We then concatenate these features with the positional encoding $\gamma(q_d)$ to construct the spatial conditioning vector $\eta_d$.

$$\eta_d = \text{Concat}\left(\gamma(q_d), \frac{1}{K'} \sum_{k \in \mathcal{N}'(q_d; \nu(\mathcal{X}_d(t)))} \mathbf{f}_{geo,k}, \frac{1}{K'} \sum_{k \in \mathcal{N}'(q_d; \nu(\mathcal{X}_d(t)))} \mathbf{f}_{app,k}\right) \tag{3}$$

Conditioned on a temporal embedding $t$, an MLP $\Phi_\omega$ takes latent $\eta_d$ to predict the backward deformation $\Delta\overleftarrow{q}_{bwd}$, mapping the dynamic point back to the canonical space: $q_c = q_d + \Phi_\omega(\eta_d, t)$.

The mapped canonical coordinate $q_c$ is fed into a geometry network $G$ to predict the SDF value $s_i = G(q_{c,i})$ and geometric features, which are then passed to a color network $C$ for view-dependent radiance $c_i$. The SDF values are converted into discrete alpha values $\alpha_i$ using a learnable logistic distribution $\mathcal{G}(x) = (1 + e^{\beta x})^{-1}$ with parameter $\beta$ controlling surface sharpness:

$$\alpha_i = \max\left(\frac{\mathcal{G}(s_i) - \mathcal{G}(s_{i+1})}{\mathcal{G}(s_i)}, 0\right) \tag{4}$$

Finally, the predicted pixel color $\hat{C}(\mathbf{r})$ is synthesized via standard volumetric integration, accumulating radiance and opacity of the number of $N$ sampled points along the ray:

$$\hat{C}(\mathbf{r}) = \sum_{i=1}^{N} T_i \alpha_i c_i, \quad \text{where} \quad T_i = \prod_{j=1}^{i-1}(1 - \alpha_j) \tag{5}$$

where $T_i$ is the accumulated transmittance, ensuring the rendering respects the implicit zero-level set.

### 4.3 Dynamic Mesh Refinement Branch

Upon convergence of the VoxGS-DSDF branch, we extract a high-fidelity canonical mesh $\mathcal{M}_c$ through the mesh extraction algorithm [28] to the zero-level set of the SDF. As implicit SDF field and explicit GS primitives are anchored and optimized within the same canonical space, the extracted mesh vertices $v_c \in \mathcal{M}_c$ are inherently aligned with the canonical Gaussians.

To articulate a static canonical mesh $v_c$ across the temporal domain, we can seamlessly reuse the forward transformation field already learned in the GS branch. As explicit discrete primitives, 3D Gaussians excel at tracking fast and complex non-rigid dynamics during the initial optimization. We transfer this pre-trained explicit motion prior to the mesh surface. The dynamic vertices $v_d(t)$ at any given time step $t$ are obtained by querying the spatial deformation component of the transformation field $\mathcal{T}_{\theta'}$ initialized from $\mathcal{T}_\theta$, thus is defined as $v_d(t) = v_c + [\mathcal{T}_{\theta'}(v_c, t)]_\mu$.

### 4.4 Learning Objective

**Gaussian Splatting Branch.** In the first stage, we optimize the canonical 3DGS and the forward transformation field $\mathcal{T}_\theta$. The overall loss is:

$$\mathcal{L}_{GS} = \mathcal{L}_{rgb} + \lambda_m \mathcal{L}_{mask} + \lambda_{reg}^{gs} \mathcal{L}_{reg}^{gs} \tag{6}$$

where $\mathcal{L}_{rgb}$ combines $\mathcal{L}_1$ and SSIM, $\mathcal{L}_{mask}$ is the opacity $\mathcal{L}_1$ error, and $\mathcal{L}_{reg}^{gs} = \|\Delta\mu\|_1$ encourages minimal spatial deformations.

**VoxGS-DSDF Branch.** We train the SDF network $(C, G)$ and the backward mapping field $\Phi_\omega$. Alongside the rendering losses ($\mathcal{L}_{rgb}^{sdf}$ and $\mathcal{L}_{mask}^{sdf}$), we propose constraints to bridge explicit and implicit representations.

*Cycle Consistency Loss.* To ensure bijection between the forward ($\Delta\overrightarrow{q}_{fwd}$ and backward ($\Delta\overleftarrow{q}_{bwd}$) deformations, we minimize their sum:

$$\mathcal{L}_{cycle} = \|\Delta\overleftarrow{q}_{fwd} + \Delta\overrightarrow{q}_{bwd}\|_1 \tag{7}$$

*SDF-GS Anchoring Loss.* To guide the continuous zero-level set with discrete primitives, we penalize their divergence using a BCE loss between absolute SDF values and the K-nearest canonical Gaussian centers. Let $d_{NN}$ denote the mean distance from $q_c$ to centers in $\mathcal{N}_K$.

$$\mathcal{L}_{SDF-GS} = \text{BCE}(|s(q_c)| > \epsilon, d_{NN}(q_c; \mathcal{X}_c) > \epsilon) \tag{8}$$

We distill geometric details using depth and normal maps from the GS branch ($\mathcal{L}_{distill}$), and apply Eikonal ($\mathcal{L}_{eik} = (\|\nabla s(q_c)\|_2 - 1)^2$) and temporal smoothing ($\mathcal{L}_{smooth}$) regularizations to the total SDF loss $\mathcal{L}_{SDF}$. The temporal smoothing term $\mathcal{L}_{smooth}$ is described in the supplementary.

$$\mathcal{L}_{SDF} = \mathcal{L}_{rgb}^{sdf} + \lambda_c \mathcal{L}_{cycle} + \lambda_{SG} \mathcal{L}_{SDF-GS} + \lambda_{dist} \mathcal{L}_{distill} + \lambda_e \mathcal{L}_{eik} + \lambda_s \mathcal{L}_{smooth} \tag{9}$$

**Dynamic Mesh Refinement.** In the final fine-tuning stage, we optimize the refined forward mapping field network $\mathcal{T'}_\theta$. The refinement loss is defined as:

$$\mathcal{L}_{Mesh} = \mathcal{L}_{rgb}^{mesh} + \lambda_m^{mesh} \mathcal{L}_{mask}^{mesh} + \lambda_{lap} \mathcal{L}_{lap} + \lambda_{reg}^{mesh} \mathcal{L}_{reg}^{mesh}, \tag{10}$$

where $\mathcal{L}_{reg}^{mesh} = \|v_d - v_c\|_1$. For stable geometry, Laplacian smoothing ($\mathcal{L}_{lap}$) is applied to vertices with the lowest 25% curvature errors. The $\mathcal{L}_{reg}^{mesh}$ regularizes the minimal deformation of the object.

## 5 Experiments

### 5.1 Datasets

The D-NeRF [36] dataset provides synthetic multi-view renderings of dynamic objects; however, it only supplies photometric images as ground truth and lacks evaluation of 3D geometry metrics. Since the original dataset does not officially release the 3D sequence meshes, we curated a novel 3D benchmark by sourcing the publicly accessible original assets. Specifically, we perform experiments on three scenes: *Lego*, *Hell Warrior*, and *T-Rex*. For each scene, we extracted the ground-truth 3D meshes at three distinct temporal snapshots: $t \in \{0.0, 0.5, 1.0\}$. This allows us to quantitatively measure the geometric fidelity and accuracy of the reconstructed surfaces over time using pure 3D metrics. Qualitative results on the remaining scenes can be found in the supplementary material. Tab. 1 uses three snapshots, while we conduct a dense 60-frame evaluation in the supplementary.

To further validate the robustness and generalization of our proposed framework, we conduct experiments on the DG-Mesh [26] and Nerfies [33] datasets. The DG-Mesh dataset features diverse

Table 1: **Quantitative comparisons on the D-NeRF [36] dataset.** We evaluate both 3D geometric accuracy and photometric rendering quality. The best, second-best, and third-best results are highlighted in red, orange, and yellow, respectively.

| Scene | Methods | vIoU (↑) | CD (↓) | PSNR (↑) | SSIM (↑) | LPIPS (↓) |
|---|---|---|---|---|---|---|
| *"Lego"* | TiNeuVox [10] | 0.1080 | 0.0174 | 25.17 | 0.9217 | 0.0689 |
| | NeuS2 [42] | 0.2578 | 0.0552 | 22.88 | 0.8945 | 0.0716 |
| | 4DGS [45] | 0.2154 | 0.0180 | 25.40 | 0.9434 | 0.0377 |
| | GaGS [29] | 0.2309 | 0.0155 | 25.44 | 0.9474 | 0.0329 |
| | DG-Mesh [26] | 0.2655 | 0.0236 | 21.29 | 0.8382 | 0.1590 |
| | D-2DGS [53] | 0.2138 | 0.0135 | 23.29 | 0.8872 | 0.1121 |
| | Ours | 0.2988 | 0.0133 | 25.40 | 0.9439 | 0.0245 |
| *"T-Rex"* | TiNeuVox | 0.4232 | 0.0071 | 32.77 | 0.9783 | 0.0307 |
| | NeuS2 | - | - | - | - | - |
| | 4DGS | 0.2702 | 0.0710 | 33.39 | 0.9869 | 0.0130 |
| | GaGS | 0.2530 | 0.0069 | 39.02 | 0.9952 | 0.0052 |
| | DG-Mesh | 0.5736 | 0.0044 | 28.95 | 0.9590 | 0.0651 |
| | D-2DGS | 0.4997 | 0.0054 | 28.68 | 0.9668 | 0.0434 |
| | Ours | 0.5783 | 0.0050 | 36.75 | 0.9932 | 0.0069 |
| *"HellWarrior"* | TiNeuVox | 0.2468 | 0.0252 | 28.20 | 0.9661 | 0.0631 |
| | NeuS2 | 0.2548 | 0.0326 | 18.02 | 0.8977 | 0.1064 |
| | 4DGS | 0.2296 | 0.0221 | 28.12 | 0.9730 | 0.0276 |
| | GaGS | 0.2427 | 0.0215 | 32.27 | 0.9835 | 0.0164 |
| | DG-Mesh | 0.3068 | 0.0198 | 25.46 | 0.9590 | 0.0841 |
| | D-2DGS | 0.3079 | 0.0132 | 25.50 | 0.9583 | 0.0488 |
| | Ours | 0.3024 | 0.0124 | 31.14 | 0.9835 | 0.0169 |
| Average | TiNeuVox | 0.2593 | 0.0165 | 28.71 | 0.9382 | 0.0673 |
| | NeuS2 | - | - | - | - | - |
| | 4DGS | 0.2384 | 0.0370 | 28.97 | 0.9677 | 0.0261 |
| | GaGS | 0.2422 | 0.0146 | 32.24 | 0.9753 | 0.0181 |
| | DG-Mesh | 0.3814 | 0.0137 | 25.23 | 0.9186 | 0.1027 |
| | D-2DGS | 0.3426 | 0.0111 | 25.82 | 0.9374 | 0.0681 |
| | Ours | 0.3932 | 0.0102 | 31.09 | 0.9735 | 0.0161 |

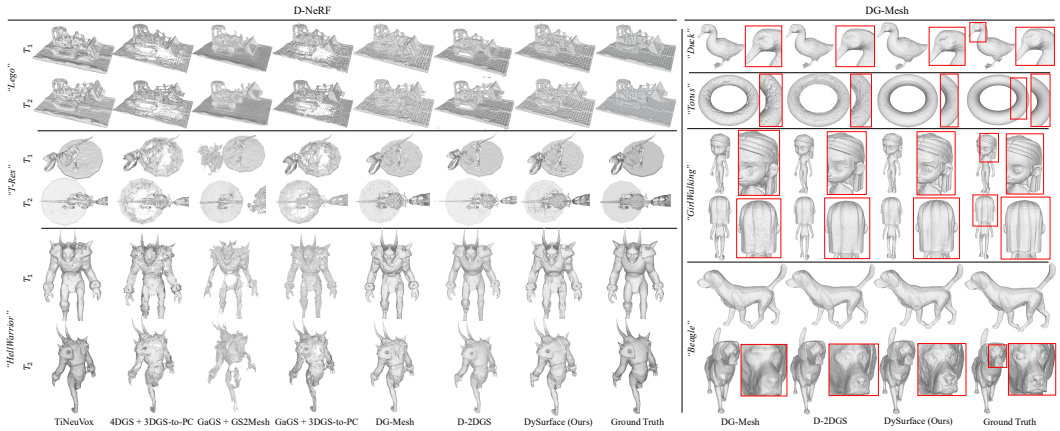



Figure 4: **Qualitative results of D-NeRF [36] and DG-Mesh [26] dataset.** We visualize the results at two different time steps from distinct viewpoints. Please zoom to check details.

deforming objects rendered from 360-degree viewpoints and provides ground-truth meshes. This allows us to evaluate the 3D geometric metrics using the same criteria applied to our D-NeRF benchmark. Finally, the Nerfies dataset consists of in-the-wild video captures of complex, non-rigidly deforming subjects recorded by a moving camera. Since high-fidelity 3D geometric ground truth is not available for such real-world captures, our evaluation on this dataset focuses primarily on qualitative mesh and quantitative NVS metrics, following the prior work [26]. There are additional real-world datasets; we experiment with DyCheck [13] in the supplementary material.

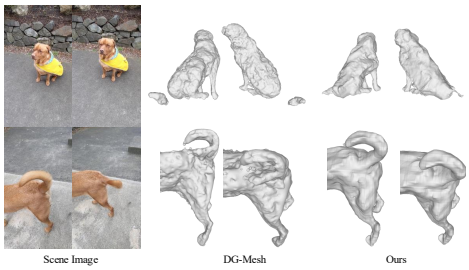



Figure 5: **Qualitative results on Nerfies [33].**

Table 2: **Quantitative results on Nerfies [33] dataset.** Our method maintains better rendering quality than DG-Mesh.

| Methods | DG-Mesh [26] | Ours |
|---|---|---|
| PSNR(↑) | 27.23 | **27.91** |
| SSIM (↑) | 0.9126 | **0.9201** |
| LPIPS (↓) | 0.0927 | **0.0815** |

### 5.2 Experimental Setup

**Baselines.** We evaluate our method against state-of-the-art methods [10, 29, 45, 26] across dynamic scene representations and surface extraction. Since 3DGS-based methods [29, 45] rely on discrete primitives without continuous surface modeling, we establish explicit baselines by applying recent mesh extraction techniques, specifically GS2Mesh [44] and 3DGS-to-PC [1] to their outputs. As 3DGS-to-PC, built upon SuGaR [15], consistently yields better accuracy than GS2Mesh, we report its results in the main paper and defer GS2Mesh comparisons to the supplementary. Furthermore, we directly compare our approach against DG-Mesh [26], D-2DGS [53], a dynamic mesh reconstruction framework, and TiNeuVox [10], a NeRF-based dynamic volume rendering method. NeuS2 [42] is evaluated under an adapted monocular setting using consecutive temporal frames as pseudo-multiview observations.

**Evaluation Metrics.** To measure 3D geometric accuracy, we report Volumetric Intersection over Union (vIoU) and Chamfer Distance (CD) to assess the mesh topology. All 3D reconstruction metrics are reported with the average score across representative temporal states ($t \in \{0.0, 0.5, 1.0\}$). For the photometric quality of NVS, we employ PSNR, Structural Similarity Index (SSIM) [43], and Learned Perceptual Image Patch Similarity (LPIPS) [52].

We report temporal consistency performance with 3D End-point error (EPE-3D), Flow Angular Error, Trajectory Normal Angular Error (TNAE), and Trajectory Position Error (TPE) metrics. We sample surface points on the canonical mesh and track their predicted and ground-truth trajectories. Note that the per-frame reconstruction baselines [53, 29, 45] reconstruct their mesh independently per time step. As they do not preserve consistent vertex correspondences across frames, vertex-level trajectory metrics are not directly defined. We therefore compare with DG-Mesh, which maintains a fixed-topology deformable mesh in Tab. 4 and Fig. 6.

**Implementation Details.** We implement our framework using PyTorch [35] and optimize it on a single NVIDIA RTX 4090 GPU (24GB VRAM). Each stage optimization takes approximately 2 hours for Stage 1, 2 hours for Stage 2, and 1 hour for Stage 3 mesh refinement.

### 5.3 Results

**Quantitative Results.** Tab. 1 presents the quantitative comparisons on the D-NeRF [36], Tab. 3 on the DG-Mesh [26] dataset, and Tab. 2 on the Nerfies [33] dataset. Our method achieves state-of-the-art performance on most geometry metrics, maintaining high rendering quality. Notably, our method outperforms the mesh reconstruction baseline, DG-Mesh [26] and D-2DGS [53], not only in geometric fidelity but also by a large margin in photometric quality. Note that NeuS2 [42] fails to converge in the *T-Rex* scene.

Table 3: **Quantitative results on DG-Mesh [26] dataset.**

| | vIoU(↑) | CD(↓) | PSNR(↑) | LPIPS(↓) |
|---|---|---|---|---|
| D-2DGS [53] | 0.6147 | <u>0.0084</u> | 24.84 | 0.0989 |
| DG-Mesh [26] | **0.7602** | 0.0090 | <u>30.45</u> | <u>0.0554</u> |
| Ours | <u>0.7571</u> | **0.0078** | **30.92** | **0.0401** |

Our method achieves competitive photometric rendering performance against purely NVS-optimized methods such as GaGS [29] and 4DGS [45]. Crucially, while these NVS-centric approaches sacrifice the 3D geometry to minimize 2D projection errors, our framework simultaneously maintains high geometric accuracy. This proves that our architecture successfully enforces strict, continuous surface constraints without degrading rendering details.

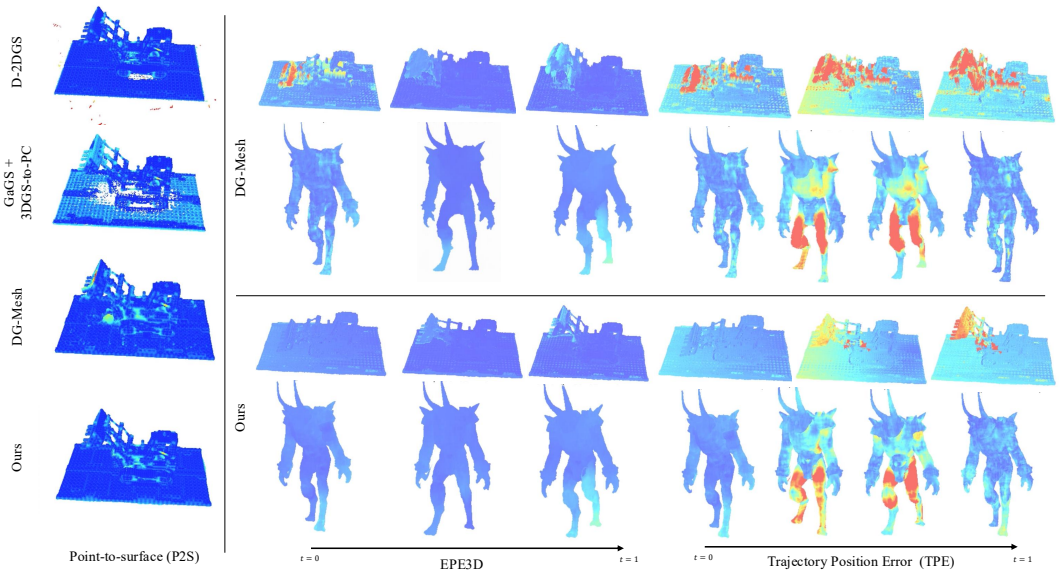



Figure 6: **Qualitative visualization of temporal consistency with a heatmap.** We show the heatmap rendered image of P2S, EPE3D, and TPE metrics.

Tab. 4 shows the evaluation results of temporal consistency. As shown in Tab. 4, our proposed method improves all four metrics, with particularly clear gains in TNAE and trajectory position error, supporting improved normal and trajectory stability. Our transform field learned in both explicit representations (GS and SDF Mesh) enables effective capture of surface flow with plausible geometries. These quantitative results demonstrate that our framework achieves accurate tracking stability with higher normal consistency.

Table 4: **Quantitative Comparison with Temporal and Trajectory Flow Metrics.**

| Method | EPE3D ↓ | Flow Angular ↓ | TNAE ↓ | Trajectory Position Error ↓ |
|---|---|---|---|---|
| DGMesh | 0.007823 | 20.42 | 15.08 | 0.00637 |
| Ours | **0.007237** | **20.13** | **11.30** | **0.00508** |

**Qualitative Results.** Fig. 4 provides qualitative comparisons against baseline methods [53, 29, 26] on the D-NeRF [36] and DG-Mesh [26] datasets. Standard GS-based baselines exhibit severe geometric artifacts, floating primitives, and fragmented surfaces under complex motions. While DG-Mesh [26] relies on direct mesh extraction from discrete Gaussians, it causes degraded geometric details and rendering quality. D-2DGS [53] also misses high-frequency details, as its meshes are built upon an estimated depth model. In contrast, our method reconstructs smooth and temporally consistent deformation while preserving fine-grained geometric details.

Furthermore, Fig. 5 demonstrates the qualitative results of our method on the real-world Nerfies [33] dataset. Due to real-world data noises, including casual camera trajectories, complex backgrounds, and highly textured objects, DG-Mesh [26] produces noisy surfaces, failing to capture stable geometric details as it extracts surfaces from noisy Gaussian primitives. In contrast, our method captures high-frequency details of the mesh surface with a stable deformation.

### 5.4 Ablation Study

We perform an ablation study to analyze the individual contributions of the core components of the DySurface. Quantitative and qualitative evaluations are shown in Tab. 5 and Fig. 7, respectively.

**Effect of VoxGS-DSDF.** To validate the efficacy of our VoxGS-DSDF branch, we ablate the deformed Gaussian voxelization and ray-query matching mechanisms. In this baseline configuration, the spatial condition latent $\eta_d$ relies on the standard positional encoding of the dynamic query point $q_d$, operating without any explicit guidance. As shown in both quantitative and qualitative results, omitting this module results in a clear degradation of geometric accuracy and photometric fidelity. This confirms

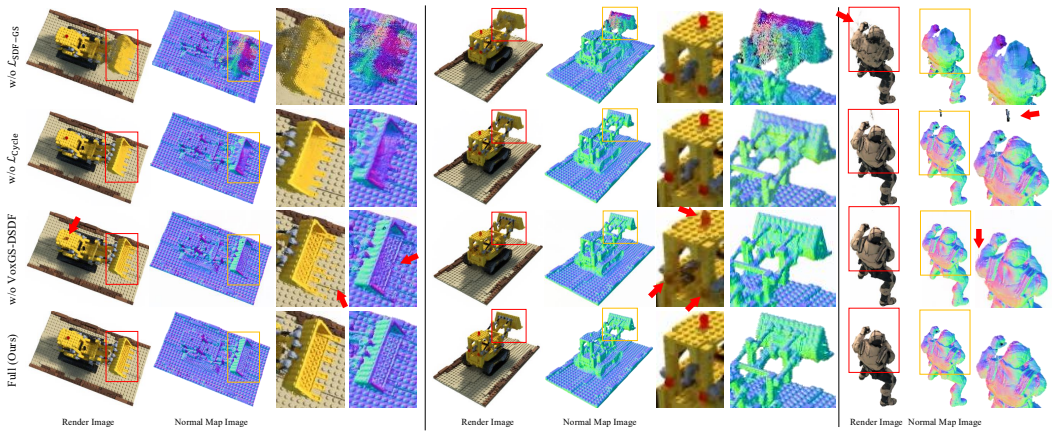



Figure 7: **Qualitative results of the ablation study.** We show the rendered image and normal map image results from the VoxGS-DSDF branch. Zoom in to check the details.

that embedding explicitly deformed Gaussian attributes as structural priors into the implicit function effectively improves the SDF optimization.

**Effect of SDF-GS Anchoring Loss.** The removal of the explicit anchoring loss ($w/o\ \mathcal{L}_{SDF-GS}$) induces a drastic performance drop in both geometric accuracy and photometric fidelity. In the absence of this structural constraint, the implicit SDF field operates independently of the explicit voxelized Gaussians, leading to ill-posed geometric optimization. Consequently, the reconstructed surfaces exhibit severe fragmentation and floating artifacts. This highlights the necessity of our explicit-to-implicit anchoring formulation for establishing robust, watertight boundaries.

Table 5: **Ablation study of SDF-GS anchoring and cycle consistency loss.** The results are reported with the average of each scene.

| | vIoU (↑) | CD (↓) | PSNR (↑) | SSIM (↑) | LPIPS (↓) |
|---|---|---|---|---|---|
| w/o $\mathcal{L}_{SDF-GS}$ | 0.3388 | 0.0248 | 24.15 | 0.9308 | 0.0420 |
| w/o $\mathcal{L}_{Cycle}$ | 0.3519 | 0.0197 | 27.50 | 0.9558 | 0.0298 |
| w/o VoxGS-DSDF | 0.3726 | 0.0120 | 27.22 | 0.9602 | 0.0297 |
| Full (Ours) | **0.3932** | **0.0102** | **31.09** | **0.9735** | **0.0161** |

**Effect of Cycle Consistency Loss.** Disabling the cycle consistency loss ($w/o\ \mathcal{L}_{Cycle}$) yields a significant degradation in both geometric precision and photometric rendering fidelity. Notably, using the cycle consistency loss increases the vIoU from 0.3519 to 0.3932, within a corresponding increase in image-level evaluation metrics. Although the base geometry remains constrained by the VoxGS-DSDF branch, removing bidirectional regularization reduces the model's capacity to track complex deformations. As illustrated in Fig. 7, high-frequency details are significantly enhanced with the $\mathcal{L}_{Cycle}$, demonstrating that $\mathcal{L}_{Cycle}$ is an effective regularizer, ensuring spatio-temporal coherence for 3D structures and enhancing the visual quality of synthesized views.

## 6 Conclusion

In this paper, we introduced DySurface, a novel hybrid framework designed to achieve high-fidelity and temporally consistent 3D surface reconstruction in dynamic scenes. By bridging the structural gap between the explicit 3D Gaussian Splatting and the geometric fidelity of implicit Signed Distance Functions (SDF), our method robustly extracts highly detailed deformable surfaces. Our core VoxGS-DSDF architecture explicitly anchors the continuous SDF field using dynamically deformed sparse voxels, robustly resolving geometric ambiguities during deformation. Comprehensive evaluations across synthetic and real-world datasets support that DySurface achieves significant performance in both geometric accuracy and photometric quality, successfully producing watertight meshes rather than fragmented proxies. We believe our approach provides a step forward for reliable 4D content creation. Our future work explores accelerating the optimization process and extending the framework to accommodate severe topological changes, such as tearing and merging.

## 7 Acknowledgement

This work was in part supported by NST grant (CRC 21015, MSIT), IITP grant (RS-2023-00228996, RS-2024-00459749, RS-2025-25443318, RS-2025-25441313, MSIT) and InnoCORE program (N10260110, MSIT).

## References

[1] Lewis A G Stuart, Andrew Morton, Ian Stavness, and Michael P Pound. 3dgs-to-pc: 3d gaussian splatting to dense point clouds. In *ICCV*, 2025.

[2] Chris Buehler, Michael Bosse, Leonard McMillan, Steven Gortler, and Michael Cohen. Unstructured lumigraph rendering. In *SIGGRAPH*, 2001.

[3] Weiwei Cai, Weicai Ye, Peng Ye, Tong He, and Tao Chen. Dynasurfgs: Dynamic surface reconstruction with planar-based gaussian splatting. *arXiv preprint arXiv:2408.13972*, 2024.

[4] Ang Cao and Justin Johnson. Hexplane: A fast representation for dynamic scenes. In *CVPR*, 2023.

[5] Danpeng Chen, Hai Li, Weicai Ye, Yifan Wang, Weijian Xie, Shangjin Zhai, Nan Wang, Haomin Liu, Hujun Bao, and Guofeng Zhang. Pgsr: Planar-based gaussian splatting for efficient and high-fidelity surface reconstruction. *TVCG*, 2024.

[6] Hanlin Chen, Chen Li, and Gim Hee Lee. Neusg: Neural implicit surface reconstruction with 3d gaussian splatting guidance. *arXiv preprint arXiv:2312.00846*, 2023.

[7] Jie Chen, Zhangchi Hu, Peixi Wu, Huyue Zhu, Hebei Li, and Xiaoyan Sun. Dash: 4d hash encoding with self-supervised decomposition for real-time dynamic scene rendering. In *ICCV*, 2025.

[8] Shenchang Eric Chen and Lance Williams. View interpolation for image synthesis. In *SIGGRAPH*, 1993.

[9] Yilun Du, Yinan Zhang, Hong-Xing Yu, Joshua B. Tenenbaum, and Jiajun Wu. Neural radiance flow for 4d view synthesis and video processing. In *ICCV*, 2021.

[10] Jiemin Fang, Taoran Yi, Xinggang Wang, Lingxi Xie, Xiaopeng Zhang, Wenyu Liu, Matthias Nießner, and Qi Tian. Fast dynamic radiance fields with time-aware neural voxels. In *SIGGRAPH Asia*, 2022.

[11] 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*, 2023.

[12] Wanshui Gan, Hongbin Xu, Yi Huang, Shifeng Chen, and Naoto Yokoya. V4d: Voxel for 4d novel view synthesis. *TVCG*, 2023.

[13] Hang Gao, Ruilong Li, Shubham Tulsiani, Bryan Russell, and Angjoo Kanazawa. Monocular dynamic view synthesis: A reality check. In *NIPS*, 2022.

[14] Ned Greene. Environment mapping and other applications of world projections. *IEEE Computer Graphics and Applications*, 1986.

[15] Antoine Guédon and Vincent Lepetit. Sugar: Surface-aligned gaussian splatting for efficient 3d mesh reconstruction and high-quality mesh rendering. In *CVPR*, 2024.

[16] Xiang Guo, Guanying Chen, Yuchao Dai, Xiaoqing Ye, Jiadai Sun, Xiao Tan, and Errui Ding. Neural deformable voxel grid for fast optimization of dynamic view synthesis. In *ACCV*, 2022.

[17] Xiang Guo, Jiadai Sun, Yuchao Dai, Guanying Chen, Xiaoqing Ye, Xiao Tan, Errui Ding, Yumeng Zhang, and Jingdong Wang. Forward flow for novel view synthesis of dynamic scenes. In *ICCV*, 2023.

[18] Binbin Huang, Zehao Yu, Anpei Chen, Andreas Geiger, and Shenghua Gao. 2d gaussian splatting for geometrically accurate radiance fields. In *SIGGRAPH*, 2024.

[19] Xu Jiawei, Fan Zexin, Yang Jian, and Xie Jin. Grid4D: 4D decomposed hash encoding for high-fidelity dynamic gaussian splatting. In *NeurIPS*, 2024.

[20] Michael Kazhdan, Matthew Bolitho, and Hugues Hoppe. Poisson surface reconstruction. In *SGP*, 2006.

[21] Bernhard Kerbl, Georgios Kopanas, Thomas Leimkühler, and George Drettakis. 3d gaussian splatting for real-time radiance field rendering. *TOG*, 2023.

[22] Junoh Lee, Chang-Yeon Won, Hyunjun Jung, Inhwan Bae, and Hae-Gon Jeon. Fully explicit dynamic gaussian splatting. In *NeurIPS*, 2024.

[23] Marc Levoy and Pat Hanrahan. Light field rendering. In *SIGGRAPH*, 1996.

[24] Zhaoshuo Li, Thomas Müller, Alex Evans, Russell H Taylor, Mathias Unberath, Ming-Yu Liu, and Chen-Hsuan Lin. Neuralangelo: High-fidelity neural surface reconstruction. In *CVPR*, 2023.

[25] Zhengqi Li, Simon Niklaus, Noah Snavely, and Oliver Wang. Neural scene flow fields for space-time view synthesis of dynamic scenes. In *CVPR*, 2021.

[26] Isabella Liu, Hao Su, and Xiaolong Wang. Dynamic gaussians mesh: Consistent mesh reconstruction from monocular videos. In *ICLR*, 2025.

[27] 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*, 2023.

[28] William E. Lorensen and Harvey E. Cline. Marching cubes: A high resolution 3d surface construction algorithm. In *SIGGRAPH*, 1987.

[29] Zhiwen Lu, Xiang Guo, Le Hui, Tianrui Chen, Min Yang, Xiao Tang, and Yuchao Dai. 3d geometry-aware deformable gaussian splatting for dynamic view synthesis. *CVPR*, 2024.

[30] Xiaoyang Lyu, Yang-Tian Sun, Yi-Hua Huang, Xiuzhe Wu, Ziyi Yang, Yilun Chen, Jiangmiao Pang, and Xiaojuan Qi. 3dgsr: Implicit surface reconstruction with 3d gaussian splatting. *TOG*, 2024.

[31] 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. In *ECCV*, 2020.

[32] Jeong Joon Park, Peter Florence, Julian Straub, Richard Newcombe, and Steven Lovegrove. Deepsdf: Learning continuous signed distance functions for shape representation. In *CVPR*, 2019.

[33] 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*, 2021.

[34] 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. *ACM Trans. Graph.*, 2021.

[35] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. Pytorch: An imperative style, high-performance deep learning library. In *NIPS*, 2019.

[36] Albert Pumarola, Enric Corona, Gerard Pons-Moll, and Francesc Moreno-Noguer. D-nerf: Neural radiance fields for dynamic scenes. In *CVPR*, 2021.

[37] Ayush Tewari, O Fried, J Thies, V Sitzmann, S Lombardi, Z Xu, T Simon, M Nießner, E Tretschk, L Liu, et al. Advances in neural rendering. In *SIGGRAPH*, 2021.

[38] Briac Toussaint, Diego Thomas, and Jean-Sébastien Franco. Probesdf: Light field probes for neural surface reconstruction. In *CVPR*, 2025.

[39] Edgar Tretschk, Ayush Tewari, Vladislav Golyanik, Michael Zollhöfer, Christoph Lassner, and Christian Theobalt. Non-rigid neural radiance fields: Reconstruction and novel view synthesis of a dynamic scene from monocular video. In *ICCV*, 2021.

[40] Haithem Turki, Jason Y Zhang, Francesco Ferroni, and Deva Ramanan. Suds: Scalable urban dynamic scenes. In *CVPR*, 2023.

[41] Peng Wang, Lingjie Liu, Yuan Liu, Christian Theobalt, Taku Komura, and Wenping Wang. Neus: Learning neural implicit surfaces by volume rendering for multi-view reconstruction. In *NeurIPS*, 2021.

[42] Yiming Wang, Qin Han, Marc Habermann, Kostas Daniilidis, Christian Theobalt, and Lingjie Liu. Neus2: Fast learning of neural implicit surfaces for multi-view reconstruction. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, 2023.

[43] Zhou Wang, Alan C Bovik, Hamid R Sheikh, and Eero P Simoncelli. Image quality assessment: from error visibility to structural similarity. In *IEEE TIP*, 2004.

[44] Yaniv Wolf, Amit Bracha, and Ron Kimmel. GS2Mesh: Surface reconstruction from Gaussian splatting via novel stereo views. In *ECCV*, 2024.

[45] Guanjun 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*, 2024.

[46] Zeyu Yang, Hongye Yang, Zijie Pan, and Li Zhang. Real-time photorealistic dynamic scene representation and rendering with 4d gaussian splatting. In *ICLR*, 2024.

[47] Lior Yariv, Jiatao Gu, Yoni Kasten, and Yaron Lipman. Volume rendering of neural implicit surfaces. In *NeurIPS*, 2021.

[48] Lior Yariv, Yoni Kasten, Dror Moran, Meirav Galun, Matan Atzmon, Basri Ronen, and Yaron Lipman. Multiview neural surface reconstruction by disentangling geometry and appearance. In *NeurIPS*, 2020.

[49] Mulin Yu, Tao Lu, Linning Xu, Lihan Jiang, Yuanbo Xiangli, and Bo Dai. Gsdf: 3dgs meets sdf for improved rendering and reconstruction. *NeurIPS*, 2024.

[50] Jingyang Zhang, Yao Yao, Shiwei Li, Tian Fang, David McKinnon, Yanghai Tsin, and Long Quan. Critical regularizations for neural surface reconstruction in the wild. In *CVPR*, 2022.

[51] Jingyang Zhang, Yao Yao, and Long Quan. Learning signed distance field for multi-view surface reconstruction. In *ICCV*, 2021.

[52] 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.

[53] Shuai Zhang, Guanjun Wu, Zhoufeng Xie, Xinggang Wang, Bin Feng, and Wenyu Liu. Dynamic 2d gaussians: Geometrically accurate radiance fields for dynamic objects. In *ACM MM*, 2025.

[54] Ziyue Zhu, Shenlong Wang, Jin Xie, Jiang-jiang Liu, Jingdong Wang, and Jian Yang. Voxelsplat: Dynamic gaussian splatting as an effective loss for occupancy and flow prediction. In *CVPR*, 2025.

[55] Zuo-Liang Zhu, Jian Yang, and Beibei Wang. Gaussian splatting with discretized sdf for relightable assets. In *ICCV*, 2025.

## 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: Please refer to abstract and introduction section.

   Guidelines:

   - The answer [N/A] 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 [N/A] 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: Please refer to supplementary material.

   Guidelines:

   - The answer [N/A] 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: [Yes]

Justification: Please refer to the methods section.

Guidelines:

- The answer [N/A] 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: Please refer to the experiment section.

   Guidelines:

   - The answer [N/A] 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: [Yes]

Justification: We will publicly open code on GitHub.

Guidelines:

- The answer [N/A] means that paper does not include experiments requiring code.
- Please see the NeurIPS code and data submission guidelines (`https://neurips.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://neurips.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) necessary to understand the results?

   Answer: [Yes]

   Justification: Please refer to the experiment section, and additional experiments will be presented on supplemental material.

   Guidelines:

   - The answer [N/A] 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: [Yes]

   Justification: Please refer to the additional experiments in the supplemental material.

   Guidelines:

   - The answer [N/A] 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: Please refer to the experiment section.

   Guidelines:

   - The answer [N/A] 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 have reviewed the NeurIPS Code of Ethics.

   Guidelines:

   - The answer [N/A] 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: [N/A]

    Justification: Please refer to the conclusion section.

    Guidelines:

    - The answer [N/A] means that there is no societal impact of the work performed.
    - If the authors answer [N/A] 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., pre-trained language models, image generators, or scraped datasets)?

    Answer: [N/A]

    Justification: This paper does not contain such risks.

    Guidelines:

    - The answer [N/A] 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 have referenced all used assets.

    Guidelines:

    - The answer [N/A] 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: [N/A]

    Justification: This paper does not introduce new assets.

    Guidelines:

    - The answer [N/A] 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: [N/A]

    Justification: This paper does not include crowdsourcing experiments.

    Guidelines:

    - The answer [N/A] 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: [N/A]

    Justification:

    Guidelines:

    - The answer [N/A] 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 rigor, or originality of the research, declaration is not required.

Answer: [N/A]

Justification:

Guidelines:

- The answer [N/A] 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 in the NeurIPS handbook for what should or should not be described.

# A Supplementary Material

Here, we present additional experimental results with DySurface that are not included in the main paper. We provide detailed descriptions of DySurface and several additional experimental results.

# B Additional Implementation Details

## B.1 Network Architecture

### B.1.1 Geometry Representation (SDF).

To represent the continuous implicit surface, we utilize a multi-resolution hash grid architecture. The spatial coordinates are encoded using a 16-level progressive band hash grid with a base resolution of 32, 2 feature dimensions per level, and a maximum hash map size of $2^{21}$. The Geometry MLP consists of 1 hidden layer with 128 neurons and employs Softplus activation along with weight normalization. To ensure stable early training, we apply geometric initialization to approximate an initial sphere with a radius of 0.8. The network outputs a 128-dimensional geometric feature alongside the SDF value. The color network predicts RGB values using the 128-dimensional geometric feature, time embeddings, and viewing directions encoded via Spherical Harmonics (degree of 4). The color network is composed of 2 hidden layers with 128 neurons each, utilizing ReLU activations for intermediate layers and a final Sigmoid activation to bound the RGB output.

### B.1.2 Sparse Voxel & Dynamic Deformation.

We anchor the Gaussian features into a sparse volumetric representation. The canonical space is bounded by a sparse voxel grid configured with a maximum resolution of $256^3$ and a voxel size of 0.002. Each valid voxel stores a 6-dimensional latent feature vector. We apply an occupancy threshold of 0.001 to aggressively prune empty grid cells and maintain computational efficiency. Instead of heuristic functional bases, the non-rigid dynamics are modeled through a dedicated Delta network. The continuous deformation from the canonical to the dynamic space is parameterized by mapping the sparse voxel features through this Delta network, ensuring structurally coherent motion tracking.

Table 6: Loss weight configurations across the three progressive training stages of DySurface. Symbols correspond to those defined in the main paper. RGB rendering losses ($\mathcal{L}_{rgb}$, $\mathcal{L}_{rgb}^{sdf}$, $\mathcal{L}_{rgb}^{mesh}$) use a weight of 1.0 by default in all stages.

| Stage | Symbol | Loss Term | Value |
|---|---|---|---|
| GS Branch | $\lambda_m$ | $\mathcal{L}_{mask}$ | 0.3 |
| | $\lambda_{reg}^{gs}$ | $\mathcal{L}_{reg}^{gs}$ | 1.0 |
| VoxGS-DSDF Branch | $\lambda_c$ | $\mathcal{L}_{cycle}$ | 1.0 |
| | $\lambda_{SG}$ | $\mathcal{L}_{SDF\text{-}GS}$ | 1.0 |
| | $\lambda_{dist}$ | $\mathcal{L}_{distill}$ | 1.0 |
| | $\lambda_e$ | $\mathcal{L}_{eik}$ | 0.1 |
| | $\lambda_s$ | $\mathcal{L}_{smooth}$ | 1.0 |
| Mesh Refinement | $\lambda_m^{mesh}$ | $\mathcal{L}_{mask}^{mesh}$ | 0.5 |
| | $\lambda_{lap}$ | $\mathcal{L}_{lap}$ | 1.0 |
| | $\lambda_{reg}^{mesh}$ | $\mathcal{L}_{reg}^{mesh}$ | 0.1 |

For volume rendering, we dynamically sample up to 1024 points per ray, which includes 256 fine samples to capture detailed topological structures accurately. Its dynamic field motion is learned using the same network structure as the Gaussian Splatting branch.

## B.2 Hyperparameters and Training Details

Our framework is trained progressively in three stages using the AdamW optimizer. Each prior stage is frozen during the next stage optimization. The Gaussian Splatting branch is trained for 30,000 steps, followed by 30,000 steps for the VoxGS-DSDF branch, and a final 1,000 steps for the Mesh Refinement branch. The base learning rate is set to 0.01 for the texture network, while the geometry and deformation modules use 0.001. A sequential learning rate scheduler is applied, multiplying the learning rate by 0.1 at 20,000 steps, with additional decay at later milestones.

Due to space limitations in the main paper, we provide the complete loss weight configurations across all three training stages in Table 6. The reported values are fixed across all evaluated scenes, and the

majority of weights are simply set to 1.0; per-scene configurations, when adjusted in specific scenes, will be released in our codebase.

In Eq. (8), $\epsilon$ serves as a spatial distance threshold that defines the boundary proximity for anchoring the implicit zero-level set to the explicit canonical Gaussian. We set $\epsilon$ relative to the overall spatial scale of the scene. For the D-NeRF benchmark, we set $\epsilon = 1 \times 10^{-4}$, which corresponds to approximately 0.008 of the object's bounding radius. Setting $\epsilon$ too large relaxes the geometric anchoring constraint, allowing the implicit SDF field to drift away from explicit Gaussians and causing loss of sharp surface details. Conversely, setting $\epsilon$ too small overly constrains the continuous SDF field to discrete Gaussian centers, which can introduce local surface noise. Our relative scaling choice balances smooth zero-level set continuity with sharp explicit boundary alignment.

### B.3 Details of Temporal Smoothness Term

To encourage temporally smooth deformation in the VoxGS-DSDF branch, we regularize the predicted backward deformation offsets of ray-sampled query points. Let $\Delta\mathbf{q}_i(t)$ denote the predicted deformation offset of the $i$-th query point at time $t$. Given its neighboring timestamps $t_{\text{prev}}$ and $t_{\text{next}}$, we randomly sample intermediate timestamps as

$$\tau_{\text{prev}} = t + \alpha(t_{\text{prev}} - t), \qquad \alpha \sim \mathcal{U}(0,1), \tag{11}$$

and

$$\tau_{\text{next}} = t + \beta(t_{\text{next}} - t), \qquad \beta \sim \mathcal{U}(0,1). \tag{12}$$

The temporal smoothness loss is defined as

$$\mathcal{L}_{\text{smooth}} = \mathcal{L}^{\text{prev}}_{\text{smooth}} + \mathcal{L}^{\text{next}}_{\text{smooth}}, \tag{13}$$

where

$$\mathcal{L}^{\text{prev}}_{\text{smooth}} = \frac{1}{N}\sum_{i=1}^{N} \left\| \Delta\mathbf{q}_i(t) - \overline{\Delta\mathbf{q}_i(\tau_{\text{prev}})} \right\|_2^2, \tag{14}$$

and

$$\mathcal{L}^{\text{next}}_{\text{smooth}} = \frac{1}{N}\sum_{i=1}^{N} \left\| \Delta\mathbf{q}_i(t) - \overline{\Delta\mathbf{q}_i(\tau_{\text{next}})} \right\|_2^2. \tag{15}$$

Here, $N$ denotes the number of ray-sampled query points, and the overline denotes a gradient-detach operation. The deformation offsets predicted at the intermediate timestamps are treated as detached temporal targets, encouraging the backward deformation field to vary smoothly over time.

## C Additional Experiments

### C.1 Evaluation Across Time Steps

Unlike prior works such as GaGS [S8], DG-Mesh [S7], 4DGS [S6], and FDNeRF [S5], which only quantitatively measure novel view synthesis photometric quality, we are the first to explicitly measure and report the 3D geometry accuracy on the D-NeRF [S2] dataset.

We present additional quantitative results to provide a fully transparent and granular breakdown of our framework's performance. Tab. 7 details the geometric evaluation scores (vIoU and Chamfer Distance) grouped by distinct continuous time steps ($t \in \{0.0, 0.5, 1.0\}$). Within each time step, we provide the metric for individual scenes alongside their cross-scene averages to demonstrate temporal stability. Furthermore, the overall photometric metrics (PSNR, SSIM, and LPIPS) are reported at the bottom of the table. This discrete time step breakdown confirms that our method maintains robust reconstruction fidelity and structural coherence not only near the canonical space but also at distant temporal states.

### C.2 Results on Remaining Scenes

In Fig. 9, Fig. 12, and Fig. 13, we provide visual results on the remaining scenes of the D-NeRF [S2] dataset. The qualitative results demonstrate that DySurface consistently captures continuous deformation, high-fidelity rendering results, and preserves sharp structural details.

Table 7: **Comprehensive Detailed Quantitative Evaluation by Time Step.** We report the detailed metrics grouped by each specific time step ($t \in \{0.0, 0.5, 1.0\}$). Methods in parentheses indicate the mesh extraction method. HW stands for HellWarrior.

| **Metric** | TiNeuVox | 4DGS + (3DGS-to-PC) | GaGS + (GS2Mesh) | GaGS + (3DGS-to-PC) | DG-Mesh | D-2DGS | **Ours** |
|---|---|---|---|---|---|---|---|
| | | | $t = 0.0$ | | | | |
| vIoU (Lego) ↑ | 0.1300 | 0.2252 | 0.3286 | 0.2446 | 0.2614 | 0.2196 | 0.3012 |
| vIoU (T-Rex) ↑ | 0.5545 | 0.2764 | 0.0783 | 0.2878 | 0.5693 | 0.4949 | 0.5532 |
| vIoU (HW) ↑ | 0.1991 | 0.2454 | 0.1990 | 0.2347 | 0.3037 | 0.3047 | 0.2906 |
| **vIoU (Avg)** ↑ | 0.2945 | 0.2490 | 0.2020 | 0.2557 | <u>0.3781</u> | 0.3397 | **0.3817** |
| CD (Lego) ↓ | 0.0180 | 0.0188 | 0.0170 | 0.0157 | 0.0242 | 0.0128 | 0.0132 |
| CD (T-Rex) ↓ | 0.0073 | 0.0700 | 0.0211 | 0.0068 | 0.0046 | 0.0056 | 0.0051 |
| CD (HW) ↓ | 0.0272 | 0.0232 | 0.0434 | 0.0230 | 0.0201 | 0.0135 | 0.0129 |
| **CD (Avg)** ↓ | 0.0175 | 0.0373 | 0.0272 | 0.0150 | 0.0163 | <u>0.0106</u> | **0.0104** |
| | | | $t = 0.5$ | | | | |
| vIoU (Lego) ↑ | 0.1080 | 0.2210 | 0.2772 | 0.2289 | 0.2701 | 0.2200 | 0.3015 |
| vIoU (T-Rex) ↑ | 0.1634 | 0.2542 | 0.0517 | 0.2198 | 0.5779 | 0.5029 | 0.6285 |
| vIoU (HW) ↑ | 0.3122 | 0.2168 | 0.1566 | 0.2586 | 0.3164 | 0.3162 | 0.3101 |
| **vIoU (Avg)** ↑ | 0.1945 | 0.2307 | 0.1618 | 0.2358 | <u>0.3881</u> | 0.3464 | **0.4134** |
| CD (Lego) ↓ | 0.0174 | 0.0187 | 0.0171 | 0.0164 | 0.0242 | 0.0139 | 0.0139 |
| CD (T-Rex) ↓ | 0.0068 | 0.0680 | 0.0275 | 0.0071 | 0.0042 | 0.0053 | 0.0048 |
| CD (HW) ↓ | 0.0226 | 0.0216 | 0.0472 | 0.0185 | 0.0189 | 0.0121 | 0.0115 |
| **CD (Avg)** ↓ | 0.0156 | 0.0361 | 0.0306 | 0.0140 | 0.0158 | <u>0.0104</u> | **0.0101** |
| | | | $t = 1.0$ | | | | |
| vIoU (Lego) ↑ | 0.0860 | 0.2001 | 0.2699 | 0.2191 | 0.2649 | 0.2019 | 0.2938 |
| vIoU (T-Rex) ↑ | 0.5518 | 0.2799 | 0.0620 | 0.2513 | 0.5735 | 0.5014 | 0.5532 |
| vIoU (HW) ↑ | 0.2292 | 0.2266 | 0.1934 | 0.2347 | 0.3002 | 0.3029 | 0.2841 |
| **vIoU (Avg)** ↑ | 0.2890 | 0.2355 | 0.1751 | 0.2350 | **0.3795** | 0.3354 | <u>0.3770</u> |
| CD (Lego) ↓ | 0.0168 | 0.0166 | 0.0160 | 0.0145 | 0.0225 | 0.0139 | 0.0129 |
| CD (T-Rex) ↓ | 0.0072 | 0.0750 | 0.0492 | 0.0069 | 0.0045 | 0.0055 | 0.0052 |
| CD (HW) ↓ | 0.0260 | 0.0215 | 0.0403 | 0.0230 | 0.0206 | 0.0139 | 0.0128 |
| **CD (Avg)** ↓ | 0.0167 | 0.0377 | 0.0352 | 0.0148 | 0.0158 | <u>0.0111</u> | **0.0103** |
| | | | ***Overall Metrics*** | | | | |
| **PSNR (Avg)** ↑ | 28.71 | 28.97 | **32.24** | | 25.23 | 25.82 | <u>31.09</u> |
| **SSIM (Avg)** ↑ | 0.9554 | 0.9678 | **0.9754** | | 0.9187 | 0.9374 | <u>0.9735</u> |
| **LPIPS (Avg)** ↓ | 0.0542 | 0.0261 | <u>0.0182</u> | | 0.1027 | 0.0681 | **0.0161** |

We report the quantitative photometric metrics on the remaining dynamic scenes in Tab. 8 and Fig. 8. We compare against representative baselines, including 3D-GS [S1], D-NeRF [S2], TiNeuVox [S3], NDVG [16], FDNeRF [S5], 4D-GS [S6], DG-Mesh [S7], D-2DGS [53], and GaGS [S8]. As our framework is fundamentally designed to extract geometrically accurate and topologically coherent surfaces, it enforces spatial regularization of the continuous SDF field. Consequently, while purely NVS-dedicated methods may yield marginally higher photometric performance by freely scattering unconstrained Gaussians to overfit the training views, they fail to recover meaningful geometry. In contrast, DySurface maintains highly competitive photometric fidelity while providing state-of-the-art geometric reconstruction, achieving an optimal balance between visual rendering quality and structural 3D geometry. Fig. 8 also shows that our method achieves a competitive rendering score against NVS-optimized methods, compared to the prior surface reconstruction methods [53, 26].

### C.3 Results on Real-world Dataset

We additionally test our method on real-world datasets to validate practical applicability. In addition to our quantitative evaluations on the real-world Nerfies [32] dataset (Table 2 and Figure 5), we conducted additional experiments on the real-world DyCheck [S11] benchmark across 7 different scenes *(haru-sit, backpack, teddybear, pillow, creeper, mochi-high-five, sriracha-tree)*. As presented

Table 8: **Quantitative comparison on the D-NeRF [S2] dataset.** We compare our method against state-of-the-art baselines. The best results are highlighted in **bold**.

| | Hell Warrior | | | Mutant | | | Hook | | | Bouncing Balls | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Method | PSNR↑ | SSIM↑ | LPIPS↓ | PSNR↑ | SSIM↑ | LPIPS↓ | PSNR↑ | SSIM↑ | LPIPS↓ | PSNR↑ | SSIM↑ | LPIPS↓ |
| 3D-GS | 15.39 | 0.8776 | 0.1300 | 21.75 | 0.9359 | 0.0575 | 18.69 | 0.8733 | 0.1144 | 22.56 | 0.9485 | 0.0647 |
| D-NeRF | 25.02 | 0.9506 | 0.0691 | 31.29 | 0.9739 | 0.0268 | 29.26 | 0.9650 | 0.1174 | 38.93 | 0.9900 | 0.1031 |
| TiNeuVox | 28.20 | 0.9661 | 0.0631 | 33.90 | 0.9771 | 0.0301 | 31.79 | 0.9718 | 0.0436 | 40.85 | 0.9913 | 0.0401 |
| NDVG | 26.49 | 0.9600 | 0.0670 | 34.41 | 0.9801 | 0.0270 | 30.00 | 0.9626 | 0.0463 | 37.52 | 0.9874 | 0.0751 |
| FDNeRF | 27.71 | 0.9665 | 0.0508 | 34.97 | 0.9810 | 0.0312 | 32.29 | 0.9756 | 0.0388 | 40.02 | 0.9912 | 0.0395 |
| 4D-GS | 28.12 | 0.9730 | 0.0276 | 38.34 | 0.9936 | 0.0062 | 33.16 | 0.9810 | 0.0168 | 40.74 | 0.9941 | 0.0105 |
| GaGS | **32.27** | **0.9835** | **0.0164** | **41.43** | **0.9969** | **0.0029** | **36.96** | **0.9916** | **0.0076** | **43.59** | **0.9960** | **0.0061** |
| DG-Mesh | 25.46 | 0.9590 | 0.0842 | 30.40 | 0.9680 | 0.0550 | 27.88 | 0.9540 | 0.0740 | 29.14 | 0.9690 | 0.0990 |
| D-2DGS | 25.50 | 0.9583 | 0.1121 | 28.12 | 0.9600 | 0.0423 | 27.80 | 0.9620 | 0.0419 | 27.78 | 0.9690 | 0.0730 |
| Ours | 31.14 | **0.9835** | 0.0169 | 32.25 | 0.9833 | 0.0099 | 32.00 | 0.9822 | 0.0148 | 35.99 | 0.9856 | 0.0136 |
| | Lego | | | T-Rex | | | Stand Up | | | Jumping Jacks | | |
| Method | PSNR↑ | SSIM↑ | LPIPS↓ | PSNR↑ | SSIM↑ | LPIPS↓ | PSNR↑ | SSIM↑ | LPIPS↓ | PSNR↑ | SSIM↑ | LPIPS↓ |
| 3D-GS | 23.10 | 0.9329 | 0.0567 | 25.75 | 0.9567 | 0.0474 | 19.38 | 0.9200 | 0.0909 | 20.72 | 0.9227 | 0.0980 |
| D-NeRF | 21.64 | 0.8394 | 0.1654 | 31.75 | 0.9767 | 0.0396 | 32.80 | 0.9818 | 0.0215 | 32.80 | 0.9810 | 0.0373 |
| TiNeuVox | 25.17 | 0.9217 | 0.0689 | 32.77 | 0.9783 | 0.0307 | 36.20 | 0.9859 | 0.0199 | 34.74 | 0.9823 | 0.0328 |
| NDVG | 25.04 | 0.9395 | 0.0534 | 32.62 | 0.9781 | 0.0330 | 33.22 | 0.9793 | 0.0302 | 31.25 | 0.9737 | 0.0398 |
| FDNeRF | 25.27 | 0.9390 | 0.0460 | 30.71 | 0.9731 | 0.0368 | 36.91 | 0.9878 | 0.0188 | 33.55 | 0.9812 | 0.0329 |
| 4D-GS | 25.40 | 0.9434 | 0.0377 | 33.39 | 0.9869 | 0.0130 | 38.26 | 0.9923 | 0.0071 | 35.67 | 0.9882 | 0.0159 |
| GaGS | **25.44** | **0.9474** | 0.0329 | **39.03** | **0.9952** | **0.0052** | **42.21** | **0.9966** | **0.0028** | **37.96** | **0.9928** | **0.0088** |
| DG-Mesh | 21.29 | 0.8380 | 0.1590 | 28.95 | 0.9590 | 0.0650 | 30.21 | 0.9740 | 0.0510 | 31.77 | 0.9770 | 0.0450 |
| D-2DGS | 23.29 | 0.8872 | 0.1124 | 28.67 | 0.9670 | 0.0434 | 29.51 | 0.9768 | 0.0280 | 29.29 | 0.9741 | 0.0322 |
| Ours | 25.40 | 0.9439 | **0.0245** | 36.75 | 0.9932 | 0.0069 | 36.08 | 0.9903 | 0.0089 | 32.35 | 0.9822 | 0.0171 |

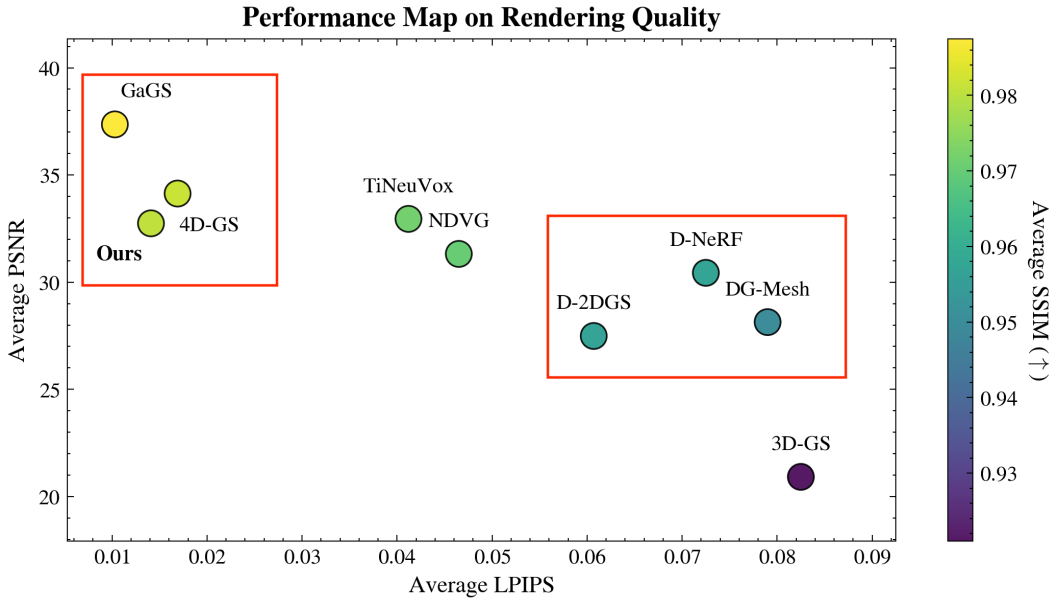



Figure 8: **Average performance map on the D-NeRF dataset.** This scatter plot visualizes the quantitative results averaged over all eight scenes. The x-axis and y-axis represent the average LPIPS and PSNR, respectively, while the marker color indicates the average SSIM score. Our method is highlighted in bold.

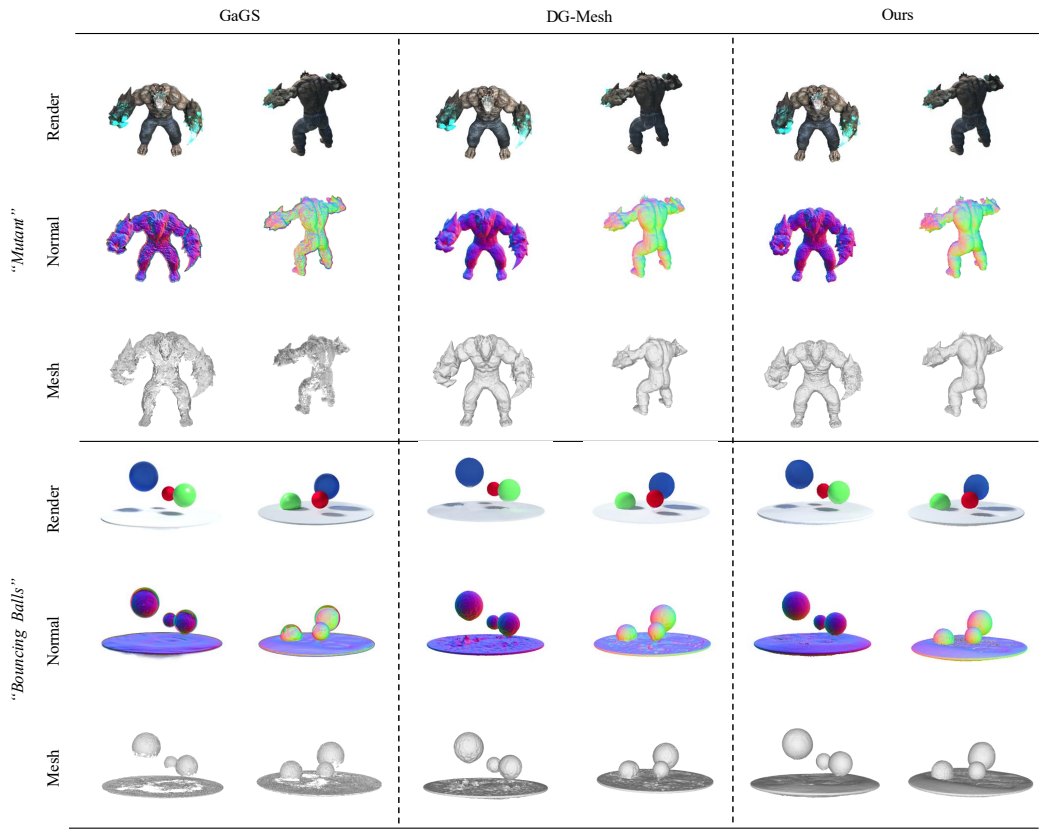



Figure 9: **Additional qualitative comparison results on diverse D-NeRF scenes.** We show the rendered image, normal map image, and mesh surface image results of GaGS [S8], DG-Mesh [S7], and DySurface (Ours). Please zoom to check the details.

Table 9: **Real-world dataset experiment on DyCheck [S11].**

| Method | PSNR ↑ | SSIM ↑ | LPIPS ↓ |
|---|---|---|---|
| D-2DGS | 21.91 | 0.8769 | 0.1546 |
| DG-Mesh | 22.06 | 0.8705 | 0.1858 |
| Ours | **24.45** | **0.8871** | **0.1309** |

in Tab. 9, DySurface outperforms baseline methods in photometric quality under unconstrained camera trajectories.

### C.4 Results on Comprehensive Frame Test

The D-NeRF [36] benchmark is a synthetic dataset generated from dynamic 3D assets. However, the official release provides only rendered image sequences and does not include ground-truth dynamic meshes for geometric evaluation. To establish an accurate geometry benchmark consistent with the motion trajectories used in D-NeRF, we curated the original 3D source assets for *Lego*, *T-Rex*, and *HellWarrior*, and matched their corresponding keyframe animation tracks. Using Blender, we exported the ground-truth deformed meshes over 60 consecutive timestamps covering the full motion interval, $t \in [0, 1]$.

We further conduct a dense-frame geometry evaluation on all available frames of *Lego*, *T-Rex*, and *Hell Warrior*. We report volumetric IoU (vIoU) and Chamfer Distance (CD) for 3D geometric accuracy, together with Depth Mean Absolute Error (Depth MAE) and depth accuracy within 5% of the object scale (D-Acc@5%) under the test view cameras. We additionally provide qualitative depth

Table 10: **Dense geometry evaluation over 60 consecutive timestamps.** vIoU and CD evaluate 3D geometric accuracy, while Depth MAE and D-Acc@5% evaluate rendered depth accuracy.

| Scene | Method | vIoU ↑ | CD ↓ | DepthMAE ↓ | D-Acc@5% ↑ |
|---|---|---|---|---|---|
| Lego | DG-Mesh | 0.2173 | 0.0731 | 0.0470 | 93.09 |
| | D-2DGS | 0.1643 | **0.0439** | 0.0393 | **95.74** |
| | Ours | **0.2263** | **0.0439** | **0.0384** | 95.69 |
| T-Rex | DG-Mesh | 0.5549 | **0.0538** | **0.0364** | 94.34 |
| | D-2DGS | 0.4679 | 0.0565 | 0.0387 | 94.31 |
| | Ours | **0.5608** | 0.0580 | 0.0399 | **94.45** |
| HellWarrior | DG-Mesh | 0.2971 | 0.0298 | **0.0225** | **97.92** |
| | D-2DGS | **0.2983** | 0.0272 | 0.0255 | 96.95 |
| | Ours | 0.2959 | **0.0240** | 0.0250 | 97.59 |
| Average | DG-Mesh | 0.3564 | 0.0522 | 0.0350 | 95.11 |
| | D-2DGS | 0.3102 | 0.0425 | 0.0345 | 95.66 |
| | Ours | **0.3610** | **0.0420** | **0.0344** | **95.91** |

and normal renderings to visualize surface accuracy throughout the sequences. As shown in Tab. 10, our method achieves the best average performance across all four metrics, demonstrating accurate geometry reconstruction over dense temporal sequences. Fig. 10 shows the qualitative result of depth and normal renderings.

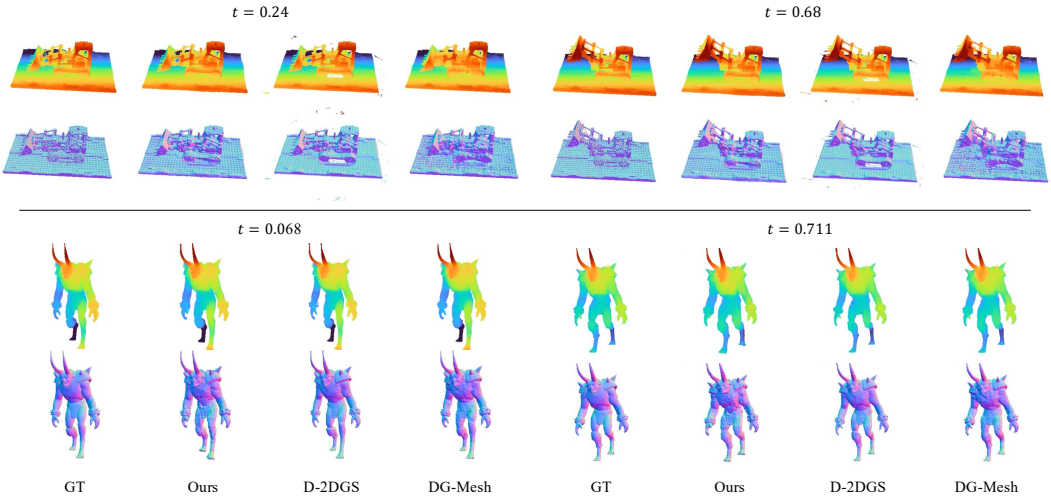



Figure 10: **Qualitative results of the dense frame test.** We visualize the depth and normal-rendered image at randomly selected times, including ground truth.

### C.5 Effect of Gaussian Representation: 3DGS vs. 2DGS

To investigate the effect of the Gaussian representation, we replace the 3DGS primitives in our framework with 2DGS [17], while keeping the remaining pipeline unchanged. As shown in Table 11, the 3DGS-based configuration achieves better average performance in both geometric reconstruction and rendering quality. In particular, it substantially improves Chamfer Distance (CD), while maintaining comparable or better photometric quality across the evaluated scenes.

Although 2DGS provides an explicit surface-oriented representation, its primitives are constrained to planar surfels. Under continuous non-rigid deformation, we observed optimization instability and occasional primitive flipping, which can degrade geometric reconstruction. In contrast, 3DGS provides full 3D covariance and greater flexibility for modeling complex deformations. Since surface

Table 11: **Impact of Gaussian representation on the D-NeRF benchmark.** We compare our default 3DGS-based model with a variant using 2DGS primitives.

| Scene | Method | vIoU ↑ | CD ↓ | PSNR ↑ | SSIM ↑ | LPIPS ↓ |
|---|---|---|---|---|---|---|
| Lego | Ours-2DGS | 0.2943 | 0.0315 | 25.09 | 0.9374 | 0.0327 |
| | Ours | **0.2988** | **0.0133** | **25.40** | **0.9439** | **0.0245** |
| T-Rex | Ours-2DGS | 0.3861 | 0.0286 | **36.92** | **0.9933** | 0.0072 |
| | Ours | **0.5783** | **0.0050** | 36.75 | 0.9932 | **0.0069** |
| HellWarrior | Ours-2DGS | **0.3057** | 0.0193 | 31.09 | 0.9821 | 0.0195 |
| | Ours | 0.3024 | **0.0124** | **31.14** | **0.9835** | **0.0169** |
| Average | Ours-2DGS | 0.3287 | 0.0265 | 31.03 | 0.9709 | 0.0198 |
| | Ours | **0.3932** | **0.0102** | **31.10** | **0.9735** | **0.0161** |

regularization is already enforced by our implicit SDF branch, 3DGS provides a more suitable explicit representation for dynamic tracking in our framework.

## D Runtime specification

The Gaussian Splatting branch is trained for 2 hours to capture the initial dynamic topology. Subsequently, the implicit VoxGS-DSDF branch is optimized for another 2 hours to build the continuous SDF field. Finally, the joint mesh refining stage requires approximately 1 hour to converge and output the final temporally consistent meshes.

## E Limitation and Future Work

While DySurface effectively extracts high-fidelity meshes from dynamic scenes, there exist limitations regarding extreme topological transformations. The Stage 2 implicit SDF presents a clean surface; minor artifacts arise during Stage 3 mesh refinement. Fitting fine-grained photometric losses during non-rigid dynamics can induce minor vertex noise and normal shifts. Increasing Laplacian regularization ($\mathcal{L}_{lap}$) in Stage 3 suppresses these artifacts; however, this may reduce fine-grained geometric details.

In our framework, we define a specific time step $t$ as the canonical space; however, this single canonical space might not contain sufficient context of the object's surface that undergoes severe fragmentation or topology changes. Extending our approach to effectively model topology-breaking dynamics, such as fracturing or fluid splashing, remains an important direction for future research.

Additionally, while our method excels on 360-degree captures with clear foreground separation, unconstrained in-the-wild videos with severe single-view occlusions receive insufficient photometric supervision, causing localized surface over-smoothing.

## F Discussion of Related Dynamic Surface Reconstruction Methods

**Comparison with DGNS.** DGNS [S9] also combines deformable 3D Gaussian Splatting with a neural implicit surface representation. However, its formulation does not explicitly address the structural discrepancy between the forward deformation of Gaussian primitives and the backward mapping required for volumetric implicit rendering. In contrast, DySurface explicitly couples these two deformation directions through KNN-based RayQuery matching over a dynamically voxelized Gaussian representation, together with the SDF-GS anchoring and cycle-consistency objectives. This design directly associates the dynamic Gaussian geometry with the implicit SDF queries and encourages bidirectional alignment between the explicit and implicit representations. In the qualitative results reported by DGNS, the reconstructed surfaces tend to appear smoother with limited high-frequency geometric details, whereas DySurface preserves finer surface structures.

**Comparison with DynaSurfGS.** DynaSurfGS [S10] reconstructs dynamic surfaces using planar Gaussian surfels based on a 2DGS [17] representation. Such surface-oriented primitives provide an explicit geometric representation, but their planar parameterization can be restrictive under complex non-rigid deformation. DySurface instead adopts deformable 3D Gaussian primitives, which provide full 3D covariance for modeling dynamic motion, while delegating continuous surface regularization to a multi-resolution hash-grid SDF. The resulting implicit surface is subsequently converted into a canonical mesh, to which the learned deformation field is transferred for temporal articulation. Qualitatively, the meshes reported by DynaSurfGS appear relatively smooth and contain fewer fine-scale surface details, while DySurface better preserves high-frequency geometric structures.

## G Statistical Significance

To ensure the reliability and robustness of our quantitative evaluation, we report the statistical significance of the results. For each experimental setting, we independently perform the evaluation 10 times. All values reported in the quantitative tables represent the average (mean) across these 10 runs, accompanied by their corresponding standard deviations ($\pm$).

Table 12: **Quantitative results of statistical significance.** All reported values represent the mean and standard deviation ($\pm$) computed over 10 independent evaluation runs.

| Methods | vIoU (↑) | CD (↓) | PSNR (↑) | SSIM (↑) | LPIPS (↓) |
|---|---|---|---|---|---|
| GaGS | $0.2422_{\pm 0.0031}$ | $0.0146_{\pm 0.0006}$ | $\mathbf{32.24}_{\pm 0.04}$ | $\mathbf{0.9753}_{\pm 0.0007}$ | $0.0181_{\pm 0.0002}$ |
| DG-Mesh | $0.3814_{\pm 0.0009}$ | $0.0137_{\pm 0.0002}$ | $25.23_{\pm 0.04}$ | $0.9186_{\pm 0.0008}$ | $0.1027_{\pm 0.0004}$ |
| D-2DGS | $0.3426_{\pm 0.0021}$ | $0.0111_{\pm 0.0004}$ | $25.82_{\pm 0.05}$ | $0.9374_{\pm 0.0006}$ | $0.0681_{\pm 0.0003}$ |
| Ours | $\mathbf{0.3932}_{\pm 0.0005}$ | $\mathbf{0.0102}_{\pm 0.0002}$ | $31.09_{\pm 0.04}$ | $0.9735_{\pm 0.0016}$ | $\mathbf{0.0161}_{\pm 0.0004}$ |

## H Application Example

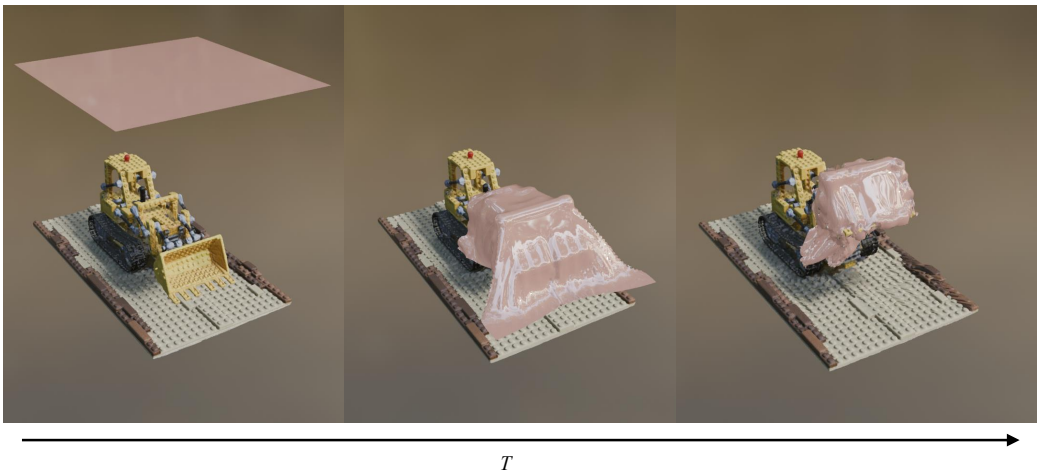



Figure 11: **Illustration of a DySurface application.** We demonstrate an example downstream application enabled by our reconstructed dynamic surfaces. The reconstructed excavator mesh sequence serves as an animated kinematic collision boundary, while a separately initialized cloth is simulated as a deformable body interacting with it.

Mesh remains the predominant representation supported by physics simulators and rendering engines. We illustrate a downstream application of DySurface for physics-based simulation in Fig. 11. Since DySurface reconstructs temporally consistent dynamic meshes from image observations, the recovered geometry can be directly integrated into standard simulation engines without manual retopology. Specifically, we import the reconstructed mesh sequence into the simulator as an animated kinematic collision boundary, where vertex positions are driven frame-by-frame by our reconstruction. We then

drop a secondary deformable object (a cloth in Fig. 11) with assigned material parameters (density, stiffness, damping), and run a forward simulation under gravity. The cloth deforms and responds to collisions with the moving reconstructed surface, demonstrating that DySurface is not only effective for high-fidelity reconstruction but also practical for geometry-aware downstream tasks requiring simulation-ready dynamic surfaces.

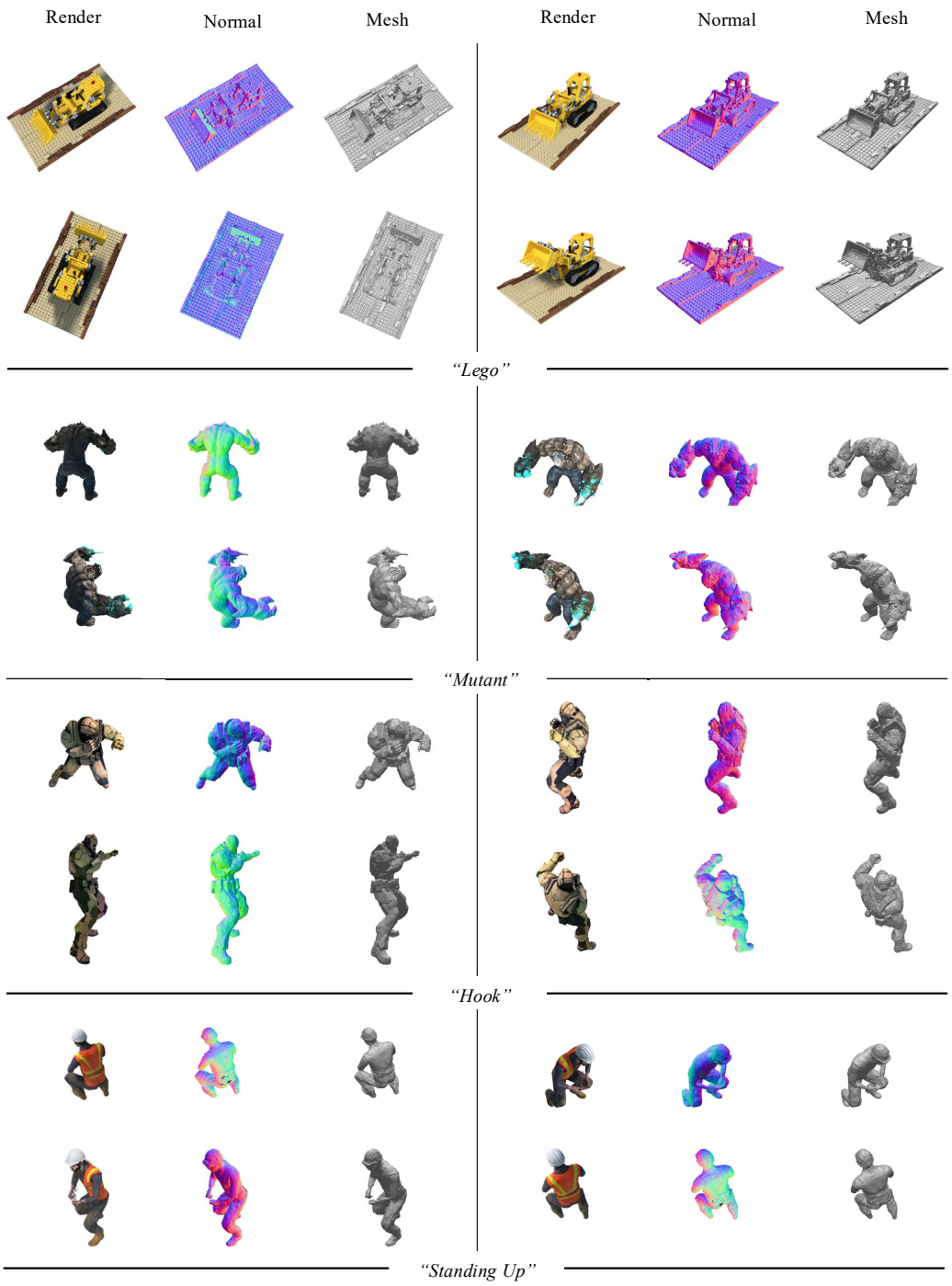



Figure 12: **Additional qualitative results of diverse D-NeRF scenes.** We show the rendered image, normal maps, and mesh surfaces results of DySurface. Please zoom to check the details.

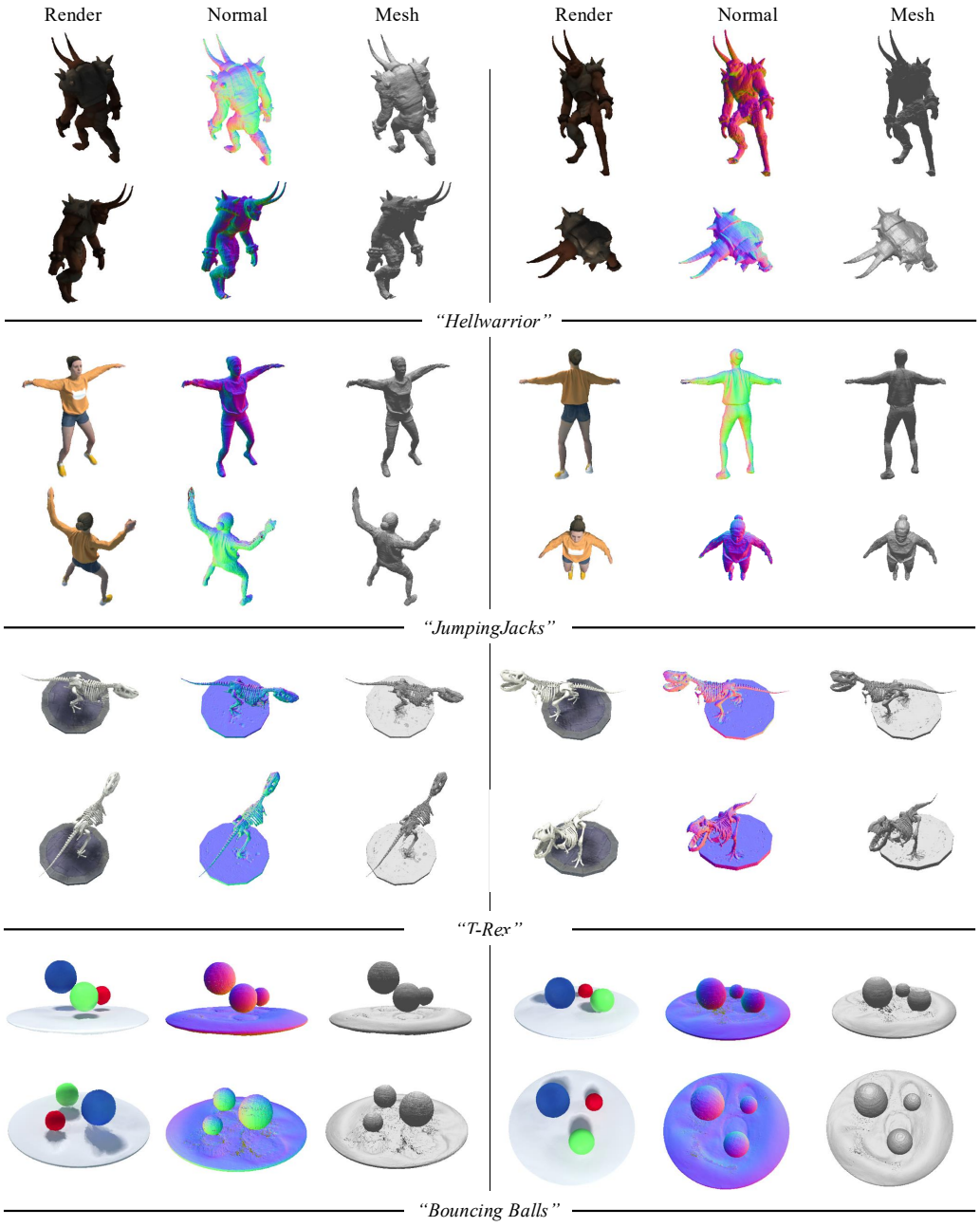



Figure 13: **Additional qualitative results of diverse D-NeRF scenes.** We show the rendered image, normal maps, and mesh surfaces results of DySurface. Please zoom to check the details.

## References


[S1] Bernhard Kerbl, Georgios Kopanas, Thomas Leimkühler, and George Drettakis. 3D Gaussian splatting for real-time radiance field rendering. In SIGGRAPH, 2023.

[S2] Albert Pumarola, Enric Corona, Gerard Pons-Moll, and Francesc Moreno-Noguer. D-NeRF: Neural radiance fields for dynamic scenes. In CVPR, 2021.

[S3] Jiemin Fang, Taoran Yi, Xinggang Wang, Lingxi Xie, Xiaopeng Zhang, Wenyu Liu, Matthias Nießner, and Qi Tian. Fast dynamic radiance fields with time-aware neural voxels. In SIGGRAPH Asia, 2022.

[S4] Xiang Guo, Guanying Chen, Yuchao Dai, Xiaoqing Ye, Jiadai Sun, Xiao Tan, and Errui Ding. Neural deformable voxel grid for fast optimization of dynamic view synthesis. In ACCV, 2022.

[S5] Jingbo Zhang, Xiaoyu Li, Ziyu Wan, Can Wang, and Jing Liao. FDNeRF: Few-shot dynamic neural radiance fields for face reconstruction and expression editing. In SIGGRAPH Asia, 2022.

[S6] Guanjun 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, 2024.

[S7] Isabella Liu, Hao Su, and Xiaolong Wang. Dynamic Gaussians Mesh: Consistent Mesh Reconstruction from Monocular Videos. In ICLR, 2025.

[S8] Zhiwen Lu, Xiang Guo, Le Hui, Tianrui Chen, Min Yang, Xiao Tang, Feng Zhu, and Yuchao Dai. 3D geometry-aware deformable Gaussian splatting for dynamic view synthesis. In CVPR, 2024.

[S9] Xuesong Li, Jinguang Tong, Jie Hong, Vivien Rolland, and Lars Petersson. DGNS: Deformable Gaussian Splatting and Dynamic Neural Surface for Monocular Dynamic 3D Reconstruction. In ACM MM, 2025.

[S10] Weiwei Cai, Weicai Ye, Peng Ye, Tong He, and Tao Chen. DynaSurfGS: Dynamic Surface Reconstruction with Planar-based Gaussian Splatting. arXiv preprint arXiv:2408.13972, 2024.

[S11] Hang Gao, Ruilong Li, Shubham Tulsiani, Bryan Russell, and Angjoo Kanazawa. Monocular dynamic view synthesis: A reality check. In NeurIPS, 2022.