Time of the Flight of the Gaussians: Fast and Accurate Dynamic Time-of-Flight Radiance Fields

Runfeng Li[1] Mikhail Okunev[1] Zixuan Guo[1] Anh Ha Duong[1]
Christian Richardt[2] Matthew O'Toole[3] James Tompkin[1]

[1] Brown University [2] Meta Reality Labs [3] Carnegie Mellon University

Abstract

We present a method to reconstruct dynamic scenes from monocular continuous-wave time-of-flight cameras using raw sensor samples that is as accurate as past methods and is 100× faster. Quickly achieving high-fidelity dynamic 3D reconstruction from a single viewpoint is a significant challenge in computer vision. Recent 3D Gaussian splatting methods often depend on multi-view data to produce satisfactory results and are brittle in their optimizations otherwise. In time-of-flight radiance field reconstruction, the property of interest—depth—is not directly optimized, causing additional challenges. We describe how these problems have a large and underappreciated impact upon the optimization when using a fast primitive-based scene representation like 3D Gaussians. Then, we incorporate two heuristics into our optimization to improve the accuracy of scene geometry for under-constrained time-of-flight Gaussians. Experimental results show that our approach produces accurate reconstructions under constrained sensing conditions, including for fast motions like swinging baseball bats.

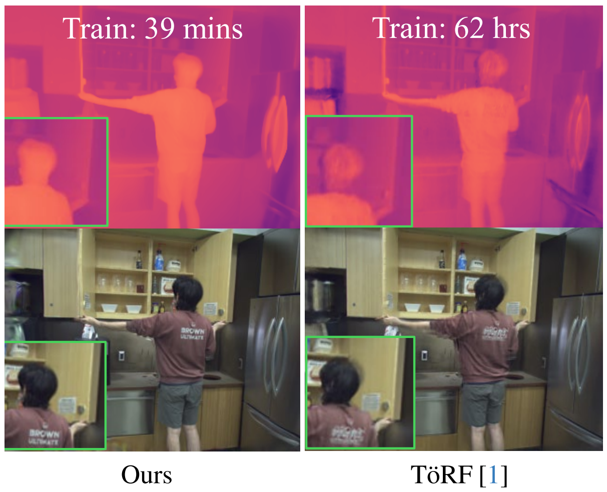

Figure 1. For time-of-flight radiance fields, when moving from easier-to-optimize NeRFs like TöRF to fast optimization via 4DGS, we must carefully consider ToF's ill-posed nature for effective indirect supervision of depth. Rendered novel views.

1. Introduction

Active illumination sensing, like continuous-wave time of flight (C-ToF), can help us to reconstruct dynamic scenes with just a single camera thanks to their depth estimates. C-ToF cameras derive depth with simple reconstruction models that assume that all surfaces are opaque and Lambertian, which is fast but can lead to depth errors. Recent optimization-based approaches attempt to resolve the scene with more sophisticated physics that model the emitted light and its reflection from an underlying transmissive 4D volume. While too slow to optimize for practical use, so-called neural time-of-flight radiance fields [1] are promising for scene reconstruction because, in principle, they are better at modeling superposition effects from multi-path light transport.

Our work considers two inter-related problems in this setting: how to make these methods faster, and how to make their optimizations more stable once we use faster—and brittler—reconstruction methods. By the end of this paper, we will have comparable accuracy to existing methods while increasing speed by 100×, so making single-camera 4D dynamic scene reconstruction more practical (Fig. 1).

First, we explain why such an optimization might be unstable. For a single camera, even with active illumination, the accurate reconstruction of depth is not sufficiently constrained by C-ToF sensor measurements under the transmissive image formulation model. As depth is only ever indirectly optimized, the scene can have highly inaccurate depth but still produce high-quality reconstructions of sensor measurements (Fig. 2). Thus, as the problem is under-constrained, its optimization is sensitive to the initialization and hyperparameters, and minor tweaks can cause large differences in the output. Past works have avoided this pernicious problem: TöRF [1] uses additional constraints to localize the depth by both moving the camera and by integrating RGB images. This reduces the problem, but is not suitable for fixed (static) cameras. F-TöRF [22] provides a more-challenging dataset with a static camera and fast-moving objects. But, without the additional constraints, its depth estimates can be worse than the simple derived depth even for opaque surfaces. Some works add priors on depth, e.g., using learned single-image depth [15, 34] or priors over scenes [3, 31]. While pragmatic, these approaches are tangential to our goal of attempting to use what sensing we have to accurately measure a scene.

Second, we explain how to speed up the optimization. One way to speed up NeRFs is to use a fast Gaussian splatting based approach [12, 18, 32]. When there are sufficient

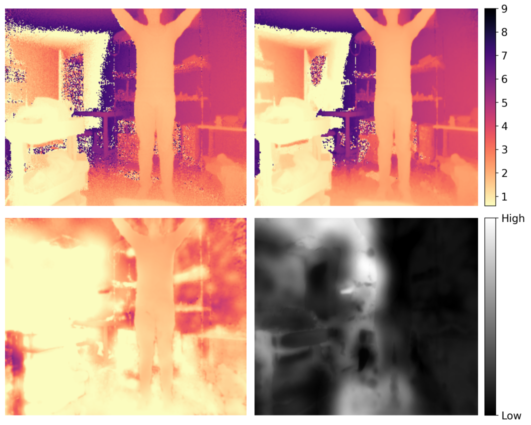

Figure 2. **Fitting ToF Images $\neq$ Fitting Depth.** *Top left:* Camera-derived depth from C-ToF. *Top right:* Rendering a GS scene reconstruction into C-ToF raw images samples, then deriving depth. As this is similar to the camera-derived depth to the left, the reconstruction objective was met. *Bottom left:* Rendered mean scene depth from Gaussians, which is highly inaccurate. *Bottom right:* Depth distortion error [9], which measures the Gaussian sparsity along each ray. Gaussians are not well localized.

cameras to constrain the optimization, GS methods can generally act as drop-in replacements for NeRFs. But, in our setting, we have only a single camera with a dynamic scene, and our desired property of depth is only indirectly optimized by the reconstruction of the sensor measurements. This makes GS methods brittle, and it is difficult to produce accurate depth results for 4D dynamic scenes with ToF imaging. While ToF NeRFs are slow, due to their MLP-based estimations, they are more robust to unfavourable initializations and hyperparameters within this under-constrained setting.

Thus, the goal is to close this gap by making fast GS-based methods more robust in our challenging setting. Existing approaches often add losses to implicitly enforce opaque surface assumptions, but these are counter to our goal of modeling more complex light transport. Instead, we contribute an analysis and corresponding insight into how to better optimize the indirect measurement of depth within time-of-flight radiance field reconstruction, using two simple heuristics. Then, we instantiate this insight into a dynamic scene reconstruction method for continuous-wave time-of-flight imaging from phasors or raw quads using 3D Gaussian splatting. This improves optimization and rendering time by $100\times$ over previous NeRF-based methods with comparable or better accuracy.

Assumptions. We assume that the emitter is co-located with the camera, which can cause shadowing errors on close objects. We assume that motion can be modeled as a piecewise linear function between timesteps; higher-order motion models may improve results for rotational motions. We use optical flow estimates within our optimization, which may have errors that propagate to the final results.

2. C-ToF Imaging Principles

Following Okunev et al. [22], C-ToF cameras illuminate the scene with a continuously-modulating amplitude of light, usually as a sinusoid $\sin(2\pi f t)$.[1] As light returns to the camera, it is correlated with a reference sinusoid to produce an image with pixel intensities $A\sin(\psi+\phi)+B$. Here, A represents the amount of light received at each pixel—the amplitude—and B represents the bias and depends upon the ambient illumination. Phase ψ captures the time that light is in flight. There is also a programmable temporal shift of the reference signal, ϕ. A, B, and ψ are three unknowns, and so we need at least three intensity measurements captured at different offsets ϕ. Many cameras use four offsets for robustness, where $\phi \in \{0, \frac{\pi}{2}, \pi, \frac{3\pi}{2}\}$, and the camera produces a quartet of raw frames $\mathbb{Q}_\phi = \{A\sin(\psi+\phi)+B\}$.

Given frames $\mathbb{Q}_\phi$, a typical C-ToF camera recovers the phase ψ and computes distance by multiplying time traveled with the speed of light c [8]:

$$d_{\text{ToF}} = \frac{\mathrm{c}}{4\pi f}\psi \quad \text{where} \quad \psi = \arctan\left(\frac{Q_0 - Q_\pi}{Q_{\frac{\pi}{2}} - Q_{\frac{3\pi}{2}}}\right), \quad (1)$$

$$\text{and where} \quad \arctan\left(\frac{Q_0 - Q_\pi}{Q_{\frac{\pi}{2}} - Q_{\frac{3\pi}{2}}}\right) =$$

$$\arctan\left(\frac{A\sin(\psi)+B-(-A\sin(\psi)+B)}{A\cos(\psi)+B-(-A\cos(\psi)+B)}\right) =$$

$$= \arctan\left(\tan(\psi)\right) = \psi.$$

We can represent the C-ToF signal as a complex phasor $a\cdot W(d) = a\cdot\exp\left(\mathrm{i}\frac{2\pi d f}{\mathrm{c}}\right) = (Q_0 - Q_\pi) + \mathrm{i}(Q_{\frac{\pi}{2}} - Q_{\frac{3\pi}{2}})$, where $a = 2A$ and d is a length of the total path that the light had to travel (thus, $d = 2\cdot d_{\text{ToF}}$) [7]. Then, the phase $\psi = \angle W(d)$ and $Q_{\text{a}} = A = \frac{1}{2}|a\cdot W(d)|$. a can also be further written as a product of the emitter signal s_{e} and a total reflectivity of the surface in a certain direction $\mathbf{r}_k$, $a = s_{\text{e}}\cdot\mathbf{r}_k$.

We can compute the amplitude of the returned light as:

$$Q_{\text{A}} = A = \tfrac{1}{2}\sqrt{(Q_0 - Q_\pi)^2 + (Q_{\frac{\pi}{2}} - Q_{\frac{3\pi}{2}})^2}. \quad (2)$$

C-ToF cameras can only measure depth unambiguously up to $d_{\text{u}} = \frac{\mathrm{c}}{2f}$ since all depths over this range will map back to $[0, d_{\text{u}}]$ due to sinusoidal periodicity. Real cameras usually have d_{u} in the range of 5–10 m. Second, as we must capture four frames, achieving 30 Hz depth output needs raw frame capture at 120 Hz, and deriving depth in this way assumes that the quartet $\mathbb{Q}_\phi$ was captured simultaneously. This means that any motion within the quartet will cause depth errors as the light arriving at a pixel will come from different world points. Third, this model assumes that light reflects back to the sensor from one surface only within a vacuum, when in truth it travels in complex paths.

[1]We can more accurately model light intensity with a non-negative signal $\frac{1}{2}\sin(2\pi f t) + \frac{1}{2}$; for clarity, we choose the simpler model.

3. C-ToF GS Image Formation

Next, we explain how we extend Gaussian splatting methods for dynamic scene time-of-flight radiance field reconstruction. We assume that the reader is familiar with both Kerbl et al. [12] and monocular 4D extensions for single-camera dynamic scenes, e.g., Yang et al. [32] or Liang et al. [18].

GS image formation model. Briefly, a scene is reconstructed by optimizing a large set of anisotropic Gaussians $\mathcal{G}_k \in \mathcal{G}$. Each is characterized by its center 3D position $\mathbf{x}_k \in \mathbb{R}^3$, covariance matrix $\mathbf{\Sigma}_k$ describing its 3D scale and rotation, opacity $o_k \in [0,1]$, and view-dependent color parameterized by 16 spherical harmonic coefficients $\mathbf{c}_k \in [0,1]^3$. Given a rasterizer to form a Gaussian as its 2D projection $\mathcal{G}_k^{\text{2D}}$ on the sensor, the color at a pixel in an output image is a weighted sum of contributing Gaussians:

$$\mathbf{c}(\mathbf{x}) = \mathbf{c}_{\text{bg}} T_N + \sum_{k=1}^{N} \mathbf{c}_k o_k \mathcal{G}_k^{2D}(\mathbf{x})\, T_k, \quad \text{where } T_k = \prod_{l=1}^{k-1} (1 - o_l \mathcal{G}_l^{2D}(\mathbf{x}))\,. \tag{3}$$

As we will discuss it later on, we explicitly include the background signal intensity beyond the scene far bounds as $\mathbf{c}_{\text{bg}}$ with the final transmittance at that bound as T_N.

C-ToF GS phasor formation model. Next, we adapt Eq. (3) to model ToF signals as phasors, following the image formation model from Attal et al. [1]. Phasor imaging [7] assumes that all $\mathbb{Q}_\phi$ are captured simultaneously. The phasor at pixel $\mathbf{x}$ is given by:

$$\mathbf{p}(\mathbf{x}) = \mathbf{p}_{\text{bg}} T_N + \sum_{i=1}^{N} \frac{s_{\text{e}} \mathbf{r}_k}{d_k^2} W(d_k)\, o_k\, \mathcal{G}_k^{2D}(\mathbf{x})\, T_k^2\,. \tag{4}$$

We note the differences from left to right. For the infrared ToF signal, rather than representing the scene radiance towards the camera $\mathbf{c}_k$ directly as in Eq. (3), we must describe the emitted and returned light. We model the returned light as the product of the source intensity s_{e} and surface reflectivity $\mathbf{r}_k$ modeled with 16 spherical harmonics coefficients, as one of the Gaussian properties. This light undergoes inverse-square falloff $1/d_k^2$, where d_k is the distance from each Gaussian center to the camera's optical center. Then, $W(d_k) = \exp(\mathrm{i}\psi_k)$ models the light path importance, where the phase shift between the emitted and reflected light is $\psi_k = \frac{4\pi d_k f}{\mathrm{c}}$, where f is the ToF camera modulation frequency, and where c is the speed of light. Finally, transmission $T_k = \prod_{l=1}^{k-1}(1 - o_l \mathcal{G}_l^{\text{2D}}(\mathbf{x}))$ is now squared to represent light traveling both to and from the scene through the radiance field.

C-ToF GS raw image formation model. Rather than a single phasor assumed to be captured at a single time, Okunev et al. [22] expand this model to consider that the $\mathbb{Q}_\phi$ set of raw images are captured over time. By replacing the phasor $W(d_k)$ with raw sample quads modulated using $\phi(d_k) : \{\sin(\psi_k), \cos(\psi_k), -\sin(\psi_k), -\cos(\psi_k)\}$, we transform the GS phasor formation model into:

$$\mathbf{q}(\mathbf{x}) = \mathbf{q}_{\text{bg}} T_N + \sum_{i=1}^{N} \frac{s_{\text{e}} \mathbf{r}_k}{d_k^2} \phi(d_k)\, o_k\, \mathcal{G}_k^{\text{2D}}(\mathbf{x})\, T_k^2\,. \tag{5}$$

Then, by finding correspondence across time between the raw quads, we can resolve depth even though the scene is moving by warping the quads to the same timestep. Within an optimization, Okunev et al. solve for this correspondence while also solving for the scene's geometry.

3.1. Where the Problems Lie

ToF samples are not depth. Unlike other problem settings that directly optimize the property of interest, e.g., color images for novel-view synthesis or depth maps for depth estimation [5, 12, 15], our challenge is that the optimized property—time-of-flight measurements, either as phasors (Eq. (4)) or as raw quads (Eq. (5))—are not the same as the property of interest (depth). Suppose we compute the depth of the scene as the mean Gaussian depth, as given by:

$$\mathbf{d}(\mathbf{x}) = \sum_{k=1}^{N} d_k\, o_k\, \mathcal{G}_k^{\text{2D}}(\mathbf{x})\, T_k. \tag{6}$$

Without additional constraints or guidance, fitting Eq. (4) volumetrically does not guarantee consistency with the depth rendered by Eq. (6). For example, when multiple density peaks exist along a ray but only a single surface exists, the rendered mean depth will diverge from the depth derived from the rendered phasor *even if* the phasor is accurately reconstructed, as shown in Fig. 2. This discrepancy arises because the sine and cosine functions in $\phi(d_k)$ oscillate between convex and concave regions, preventing equality in the finite form of Jensen's inequality. Only when all Gaussians are clustered at the same z-position along each ray does this depth gap reduce.

Previous approaches. Past works have integrated additional multi-view constraints from moving cameras or extra RGB input [1] to alleviate it. Of course, this works with a Gaussian renderer too, but such an approach avoids the problem rather than finds a better solution to it. Further, experimentally, previous NeRF-based ToF radiance field methods [1, 22] exhibit this gap to a lesser extent, possibly due to the MLP's inherent bias towards low-entropy (simple) solutions: a single high-density spike (a surface) has lower entropy along the ray than many locations with mid- or low-density values.

In contrast, 4DGS has no such inherent bias; Gaussians can be positioned arbitrarily along the ray to fit the target, which harms depth accuracy. Supposing that we start from randomly-initialized Gaussians, the flexibility in Gaussian reflectivities along each ray allows the optimizer to fit ToF data by forming high-entropy, multi-peak density distributions. While this approach still causes a rendering of the scene to match the ToF data, the scene diverges from our goal of producing accurate mean depth maps.

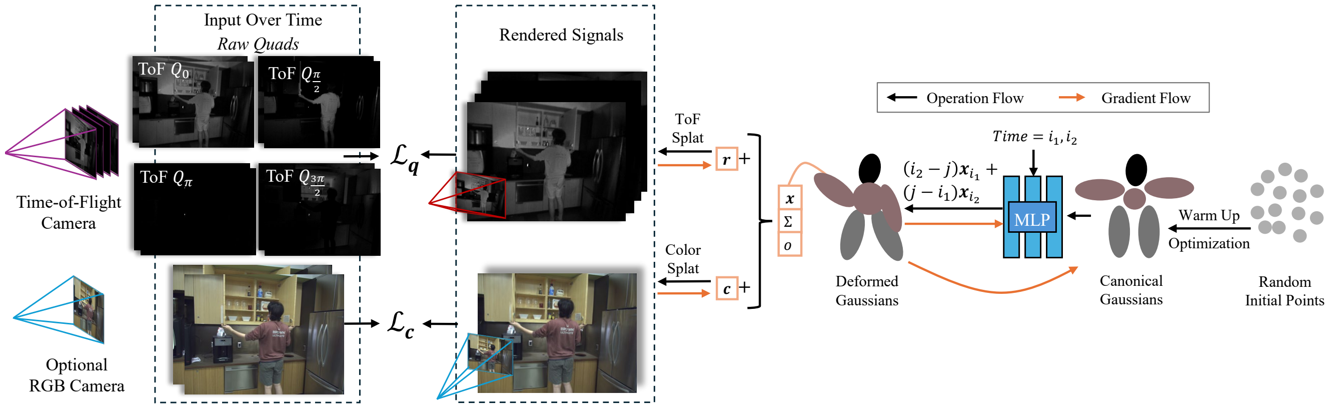

Figure 3. **Pipeline overview.** *Left:* We capture input raw quads (or phasors, not shown) from a time-of-flight camera with optional color camera. *Right to Left:* Starting from randomly initialized Gaussians, the warm-up stage estimates canonical scene geometry using ToF signals. Given time t and the canonical Gaussian positions, the MLP deforms the Gaussians using the offsets ($\delta\mathbf{x}$). These deformed Gaussians are then used to render the ToF and color images.

Current approaches in RGB scene reconstruction—even in more-constrained multi-view settings—use a depth distortion loss to help push Gaussians towards a surface [9]:

$$\mathbf{DD}(\mathbf{x}) = \sum_{k=1}^{N}\sum_{l=1}^{N} \omega_k \omega_l \|d_k - d_l\|^2, \tag{7}$$

where $\omega_k = o_k\, \mathcal{G}_k^{\text{2D}}(\mathbf{x})\, T_k$. This implicitly enforces an opaque surface assumption, and often the scenes reconstructed by these methods do conform to this assumption. However, this implicit assumption reduces our ability to model complex light transport. If we only wanted to reconstruct opaque surfaces, we would just derive depth from ToF in a closed form. The loss tends to produce oversmoothed depth reconstructions; see supplemental for ablations.

Heuristic 1: Occupancy bias. To address this problem, we note that, within the space of reconstruction parameters and mechanisms when trying to reconstruct scenes (reflectivity, position, opacity, densification), 'where things are' or 'occupancy' is the primary variance, and so reflectivity changes should be rarer than occupancy changes (position, opacity, densification). Given the choice, we would rather move points around, make new density, or remove density than change the proportion of light that is returned. To achieve this, we can decrease the learning rate within the optimization for scene reflectivity $10\times$, which increases the effect of position and opacity variation instead. While simple, this adjustment better encourages Gaussians to conforms to the true scene structure from the outset *without* forcing Gaussians to be close to each other as in Eq. (7).

Heuristic 2: Low-reflectivity bias. The initial reflectivity plays a crucial role in the optimization result, particularly for low-reflectivity regions of the scene. Assume initial reflectivity values are high (e.g., 0.5). Then, with an occupancy bias, due to the inverse-square falloff term, we encourage Gaussians for low-reflectivity regions to be incorrectly placed far away from the camera. Before the reflectivity of these far Gaussians can be corrected, and as Gaussians have spatial extent once splatted, other closer splat-overlapping Gaussians interfere to prevent the reflectivity correction. This leads to multiple opacity peaks.

Now let us assume a lower initial reflectivity ($0.01 - 0.1$). This enables Gaussians for low-reflectivity regions to be placed more correctly because the lower initialization provides a closer starting fit to the true scene. With an occupancy bias, then Gaussians for high-reflectivity regions will be placed closer to the camera during initial optimization stages also due to the inverse-square falloff term. As they occlude the scene, they are less likely to interfere with other Gaussians, and so have more iterations to optimize their reflectivities and positions to reach the true depth.

Empirically, combining occupancy and low-reflectivity biases is effective for both low and high reflectivity scene regions in our test sequences, whereas only using an occupancy bias (no low-reflectivity bias) harms low-reflectivity regions. While simple, both approaches are more effective than Eq. (7) without overly-constraining the final result.

4. Pipeline

As input, our system takes a video sequence of raw quartet ToF images and an optional color sequence (see Fig. 3). Our approach equivalently works with ToF phasors. We assume accurate camera poses, which align the scene scale with the ToF measurements. For our 4DGS approach [32], we use an MLP that takes time t and Gaussian positions $\mathbf{x}$ as input and outputs Gaussian 3D position offsets $\delta\mathbf{x}$. Thus, Gaussians are implicitly deformed from a canonical space.

Local linearity within each four raw frames is critical for fitting as the quartet is asynchronously captured. Let us consider an image in a quartet to be at an 'integer' timestep i, and its matching-amplitude raw quad in four ticks time to be at $i + 1$. Then we only deform Gaussians to integer time steps but ensure local linearity at intermediate fractional time steps j by linearly interpolating Gaussian positions from the two nearest integer timesteps i_1 and i_2:

$$\mathbf{x}_j = (i_2 - j)\mathbf{x}_{i_1} + (j - i_1)\mathbf{x}_{i_2}. \tag{8}$$

This is functionally equivalent to the phase-aware reprojection loss from Okunev et al. [22].

Optical flow weak supervision. To help correspond fast motions, we also weakly supervise the forward and backward 3D motion offsets of Gaussians against estimated optical flow from RAFT [27] with an L2 loss $\mathcal{L}_\text{f}$. Optical flow is projected from the estimated 3D position offsets, where the forward flow is computed as (similarly for backward flow):

$$\mathbf{f}(\mathbf{x}) = \Pi_i \left(\text{sg}(D_\mathbf{x}) + \sum_{k=1}^{N} \Delta\mathbf{x}_k o_k \mathcal{G}_k^{2D}(\mathbf{x}) T_k \right), \tag{9}$$

where $\Delta\mathbf{x}_k = \text{MLP}(\mathbf{x}, i+1) - \text{MLP}(\mathbf{x}, i)$ is the Gaussian scene flow, $D_\mathbf{x}$ is the back-projected 3D point from the rendered mean depth, sg stops gradient computation, and Π_i is the camera's perspective projection at time i.

Total loss. We combine three objectives:

$$\mathcal{L} = \alpha\mathcal{L}_\text{q} + \mathcal{L}_\text{c} + \beta\mathcal{L}_\text{f}, \tag{10}$$

where $\mathcal{L}_\text{q}$ is the raw quad reconstruction loss and $\mathcal{L}_\text{c}$ is the optional color reconstruction loss. Following Kerbl et al. [12], both use $\mathcal{L}_1$ and SSIM terms.

Initialization. We initialize Gaussians randomly within the camera frustum, bounded by near and far planes, and with a reflectivity of 0.1. Unlike other dynamic GS models, our Gaussian positions are in real-world units, potentially with large values. To ease optimization, we rescale input coordinates to the MLP (only) using the unambiguous depth range, ensuring most values lie within [-1, 1]. Further, in an initial warm-up stage for 2K iterations, we assume that the scene is static. This helps to place many of the Gaussians into useful positions.

Random background. For sensitive low-reflectivity scene areas, a low background contribution $\mathbf{q}_\text{bg}$ can lead to errors: Gaussians should be placed on a surface to contribute, but the background is already mostly sufficient to reproduce the ToF signal. This creates only low-contributing Gaussians. To fix this, we randomly change the background signal (uniformly between -1 and 1) at each training iteration.

MLP. The deformation MLP consists of 8 layers with 256 neurons each, using positional encoding frequencies of $L_x = 10$ and $L_t = 10$. MLP biases are initialized to zero, and weights are initialized with Xavier normal initialization in PyTorch, except for the final linear layer, which maps the final 256 features to the delta position. This layer's weights are initialized from a normal distribution with a standard deviation of 10^{-5} to ensure stable training by starting Gaussian deformations from small $\delta\mathbf{x}$ values.

Hyperparameters. In the loss, we set $\alpha = 5$ (1 for synthetic scenes) and $\beta = 0.0008$. We use the Adam optimizer with $\beta = (0.9, 0.999)$ and $\epsilon = 10^{-15}$. We initialize reflectivity as 0.1 and use a reflectivity learning rate of $\gamma_\text{r} = 0.00016$. The MLP learning rate starts at $\gamma_\text{MLP} = 0.0008$ and decays exponentially to 1.6×10^{-6} over 60K iterations.

Implementation Details. Our code is based on Kerbl et al. [12], with custom CUDA implementations for forward and backward raw ToF image rasterization. Details on ToF Gaussian gradient calculations are provided in the supplemental material. Gaussians are rendered twice using a differentiable rasterizer: once for the ToF view to compute the quad loss, and once for the RGB view to compute the color loss.

5. Experiments

TöRF Dataset. The dataset contains five real-world indoor dynamic monocular sequences (*Photocopier*, *Cupboard*, *DeskBox*, *PhoneBooth*, and *StudyBook*). All scenes have raw ToF images and RGB images with 12 bits of information. ToF signals were captured with a Texas Instruments OPT8241 sensor (320×240 at 30 fps) with 5 m unambiguous depth range. We use the calibrated camera poses optimized by retraining the TöRF model. The cameras are moving, which provides additional multi-view constraints, and scene motion is slow.

F-TöRF Dataset. The dataset contains five sequences from the dataset: *Pillow*, *Baseball*, *JumpingJacks*, *Target*, and *Fan*. Again, these scenes were captured with a Texas Instruments OPT8241 sensor (320×240 at 30 fps) with 5 m unambiguous depth range. The scenes are significantly more challenging: they contain no camera motion and they contain fast-moving objects such as falling pillows and swinging baseball bats.

F-TöRF Synthetic Dataset. We also use their set of seven didactic scenes. These are generated in Blender using physically-based path tracer PBRT [23] adapted for ToF emission, at a resolution of 320×240. The scenes show cubes undergoing axial, lateral, and rotational motions to induce varying disparities, with occlusion, both with and without texture, or with chairs to show thin features. The scenes are strictly monocular without camera motion. The fastest scenes (*3 Cubes Speed Test*, *3 Chairs Speed Test*, *Arcing Cube*, *Axial Speed Test*) have large disparity: maximally up to 30 pixels across raw frames, and often 9–18 pixels (*Sliding Cube*, *Occluded Cube*, *Orthogonal Speed Test*). Three challenge scenes test large axial motion (*Arcing Cube*, *Axial Speed Test*, *Orthogonal Speed Test*).

Metrics. For synthetic scenes, we have ground-truth depth and so can measure MSE. For real-world scenes, we only evaluate qualitatively. We report model training times for a single NVIDIA 3090 GPU.

Baselines. We compare to ToF NeRFs using phasors (TöRF) and corresponded raw quads (F-TöRF). For a 4DGS comparison, we compare to DeformableGS [32] when it is given additional depth input derived from C-ToF imaging as in Eq. (1). We also include a 2D baseline that uses softmax splatting [21] and RAFT optical flow to warp raw quads to new timesteps.

5.1. Results

Computation Time. As expected, our approach is significantly faster than TöRF [1] (60 hours) or F-TöRF [22]

Table 1. **Depth error on the F-TöRF synthetic dataset**. Each number is a depth MSE×100. 'd' refers to a standard volume-integrated depth from Eq. (6). 'd_{ToF}' is a depth, derived from the reconstructed ToF signal using Eq. (1). Bold marks best result for both d and d_{ToF}

	Sliding Cube	*Occluded Cube*	*Axial ST*	*3 Cubes ST*	*3 Chairs ST*	*Arcing Cube*	*Ortho. ST*
C-ToF	0.096	0.130	2.068	3.131	0.787	36.768	41.931
2D Flow	0.018	0.088	0.662	0.916	**0.229**	1.678	19.805
F-TöRF d_{ToF}	0.023	0.114	0.251	0.501	0.324	**0.470**	27.488
Ours d_{ToF}	**0.005**	**0.062**	**0.123**	**0.281**	0.639	1.060	**14.933**
TöRF d	0.349	0.388	1.143	2.363	0.956	6.278	22.855
F-TöRF d	0.440	0.647	0.938	1.390	**0.855**	1.256	**7.527**
Ours d	**0.037**	**0.369**	**0.525**	**0.641**	1.023	**1.023**	22.772

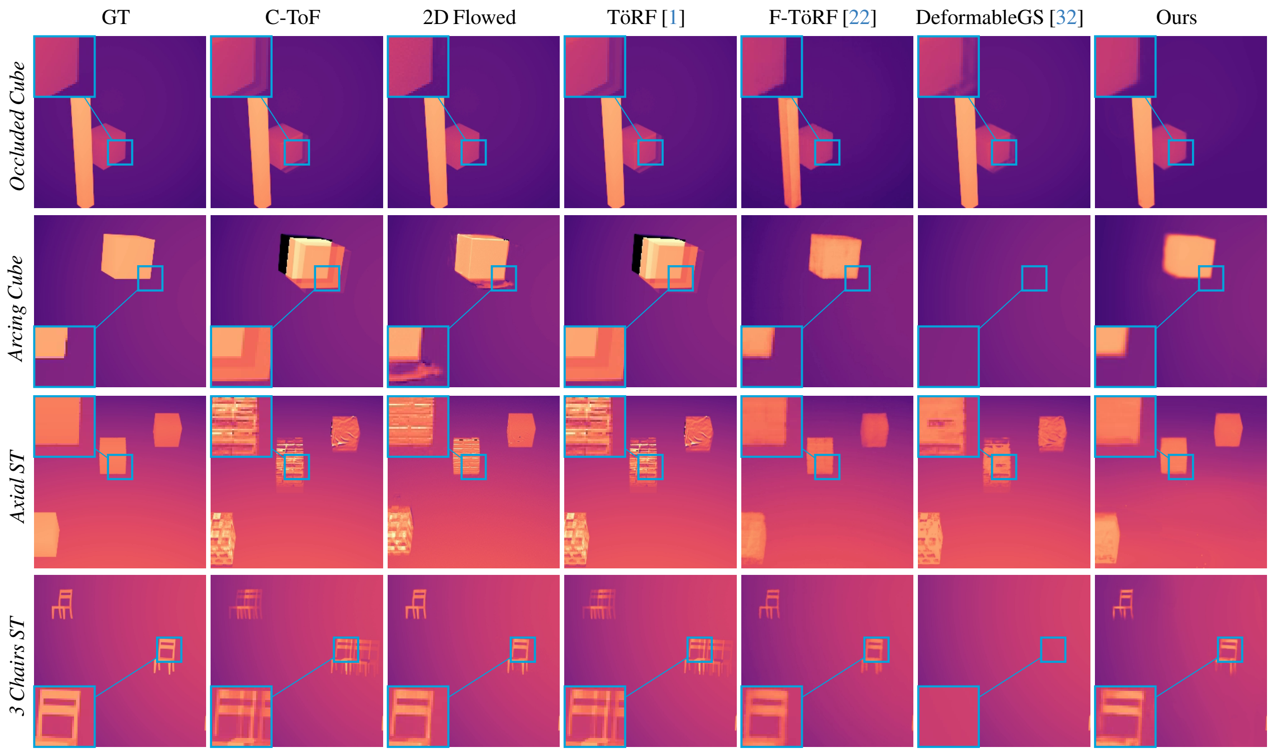

Figure 4. **Our approach is competitive or better in terms of accuracy against the state of the art on synthetic scenes, while being two orders of magnitude faster.** We use the synthetic dataset from F-TöRF to demonstrate that our model produces comparable quality reconstructions. All images show rendered volumetric depth d. SOTA NeRF model F-TöRF produces similar results to ours while being significantly slower. SOTA 4DGS model 'DeformableGS' uses only additional depth information without a physically-based ToF model, and fails to reconstruct the scene well. Baseline '2D Flowed' model is fast but cannot account for axial motion as well as our model, producing artifacts.

(72 hours) at 40–60 minutes for a corresponded 4D scene reconstruction. Rendering is real time at 100+ Hz.

Quantitative Results. It is well known that GS methods typically represent a tradeoff between quality and speed, with NeRF-based methods providing higher quality. Despite that, our method is competitive in terms of depth error (Tab. 1). For the reconstructed scene depth d, we show improved results on five out of seven synthetic scenes over all baselines and produce competitive results on the rest. Our predicted scene density is 'denser' than that of F-TöRF (*Arcing Cube*), as seen by the reduced gap between d and the depth derived by Eq. (1) d_{ToF} from the reconstructed sensor images. Thin structures (as in *3 Chairs Speed Test*) seem harder to reconstruct as they require a precise configuration of anisotropic Gaussians. Large-scale nonlinear motions (*Arcing Cube*) are also challenging to optimize under a canonical deformation model. *Orthogonal Speed Test* suffers from a local optima on one of the cubes, which biases the error metric upwards (see the supplemental videos).

Qualitative Results. On the easier TöRF dataset that has color and a moving camera (Fig. 6), we show that depth supervision is still critical for 4DGS: DeformableGS without the added depth fails to reconstruct the scene well. With naïve depth, DeformableGS improves substantially, but is

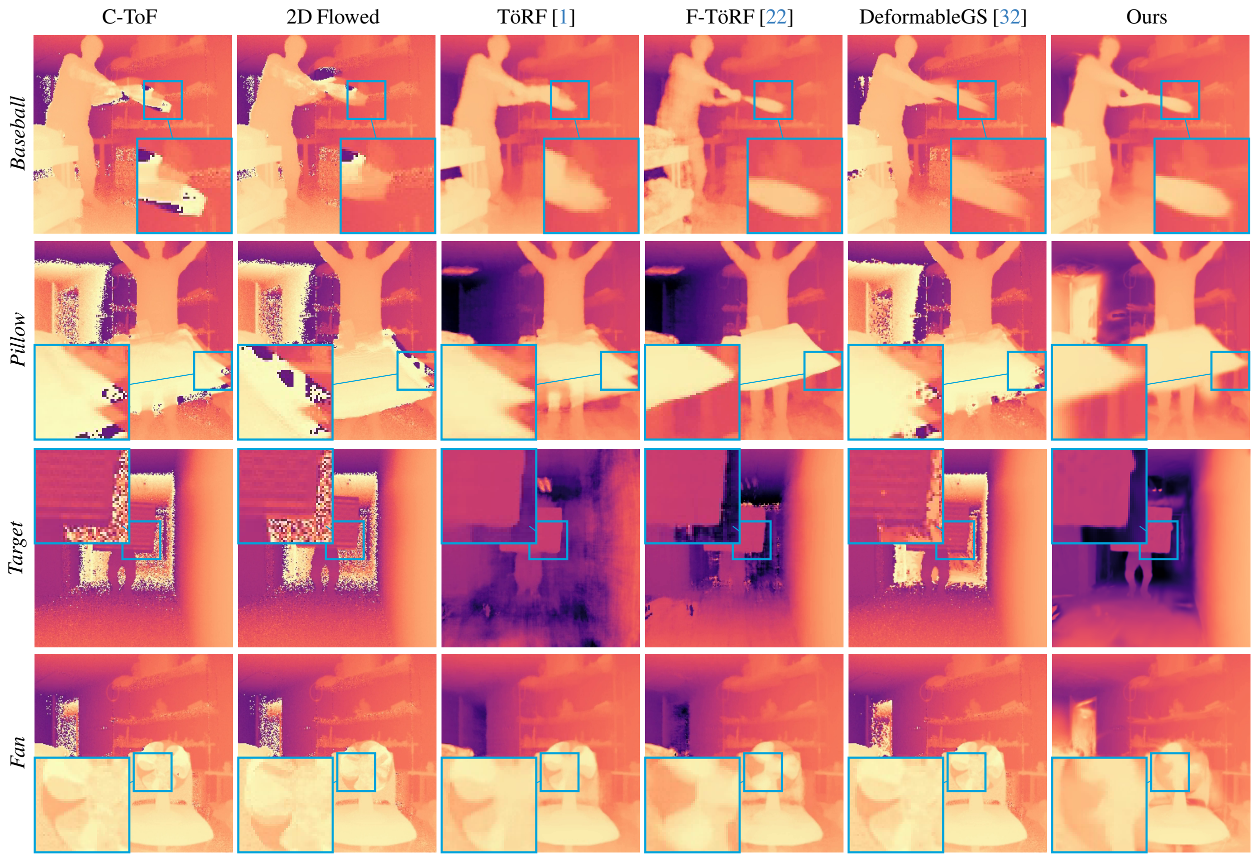

Figure 5. **F-TöRF real scenes; rendered scene depth** d**.** As it models temporal dynamics and constrains geometry appropriately during training, our model produces comparable quality reconstructions with F-TöRF while being consistently better than other baselines. Our method tends to reconstruct static geometry better (floor in *JumpingJacks*). *Fan* scene presents a significant challenge as the motion is nonlinear—our canonical field struggles to track this geometry over time. F-TöRF can approximate this scene better due to the lack of such a global constraint. Depth wrapping effects in the background occur in some scenes; these represent an ambiguity that is out of our scope for this work.

still inferior to our physically-based ToF supervision. Our method is comparable to TöRF, with some areas better for ours than TöRF (see plaque in *StudyBook*).

The challenging F-TöRF dataset (Figs. 4 and 5) reveals that our method is comparable with F-TöRF, and only these two methods can correct for fast motion artifacts. DeformableGS often completely fails to reconstruct a fast-moving object (*Arcing Cube*, *3 Chairs Speed Test*), and 2D Flowed cannot handle axial motion—something that our method can do. While faster, our method also often produces better static region reconstructions compared to F-TöRF (floor in *JumpingJacks*, *Target*, cart in *Baseball*). But, similar to the quantitative analysis, our method somewhat struggles with thin structures in *3 Chairs Speed Test* and can produce depth that is less sharp than F-TöRF. Finally, no approach can accurately reconstruct spinning fan blades in *Fan*.

6. Discussion

Limitations. *Depth wrapping.* Several of our reconstructions (*JumpingJacks*, *Target*, *Pillow*) contain depth wrapping in the background, which is a typical C-ToF artifact. Although some baselines happen to avoid this issue on the surface, we argue that this problem is fundamentally ill-posed for a static camera, and predictions heavily depend on the optimization biases since a unique solution is unattainable.

Nonlinear motion. Our model can struggle when the motion is strongly nonlinear (as in *Fan* and to a smaller extent in *Arcing Cube*). This can be in principle resolved with a more flexible model that allows nonlinear trajectories, although it will likely make the problem even more ill-posed and require stronger regularizations.

Sharp reconstruction. Our volume reconstruction can be visually less sharp that NeRF-based methods around the edges, despite having smaller depth error. This is likely a consequence of spatial smoothness that Gaussians exhibit. More complex surface representations might help [6, 9].

Conclusion. We present a Gaussian splatting approach for fast reconstruction and rendering of dynamic monocular sequences with asynchronous ToF exposures. Adapting 4DGS to this setting required us to consider how the underlying optimization affects the indirect optimization of depth, and devise two heuristics that better condition the optimization. These successfully avoid overfitting across a set of synthetic and real-world scenes. In sum, our method is 100× faster than

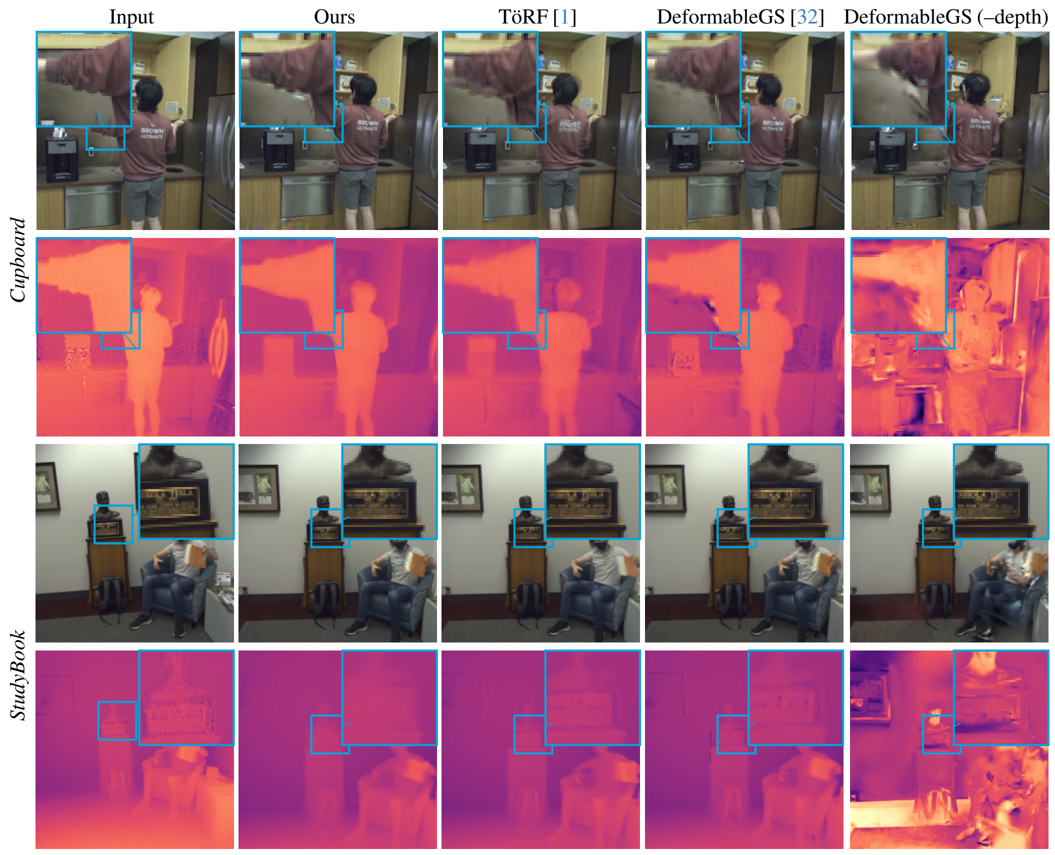

Figure 6. **Our approach is competitive with other baselines on TöRF dataset.** Comparison of our method with TöRF and DeformableGS (with and without C-ToF depth prior). Images are novel views of rendered volumetric depth d and color, rendered along a spiral path around the training camera. Our approach demonstrates same or better depth reconstruction for both static and dynamic objects, particularly in dynamic regions with limited multi-view signals. Our method mitigates overfitting to noisy raw depth measurements in the *StudyBook* scene.

baselines, and produces comparable or better reconstructions of fast-moving objects even with a static monocular camera.

7. Related Works

Accurately reconstructing geometry is important for many applications, such as in measurement or physics simulation [30]. When the capturing setup is equipped with a depth sensor, high-quality reconstruction becomes possible even with a small camera baseline. In TöRF [1], a NeRF model takes advantage of a C-ToF camera and uses phasors to optimize the geometry of a dynamic scene. In F-TöRF [22] the model works directly with raw C-ToF camera outputs and recovers motion in addition to geometry, while avoiding typical C-ToF camera motion artifacts. Another work [25] uses a structured light camera in a static setting not only to recover the geometry, but also normals and direct and indirect illumination. PlatoNeRF [13] recovers scene geometry from a single view using two-bounce signals captured by a single-photon lidar. However, each takes many hours to optimize a single scene. Our work adapts such techniques to fast Gaussian splatting.

Dynamic Gaussian Splatting. Since 3DGS [12] was introduced, it has been explored and adapted widely for reconstruction due to its efficiency. Extending 3DGS, a plethora of works on dynamic Gaussian splatting emerged recently, differing in their motion representations, constraints and optimization approaches. In Dynamic 3D Gaussians [20], the model assumes a freeform rotation and translation of Gaussians between consecutive time steps and iteratively optimizes the scene with rigidity constraints, but it heavily relies on multi-view input. Some works [18, 29, 32] map each time moment to a canonical space to model the dynamic content. Another approach to model the motion is a global trajectory reconstruction using Fourier, polynomial or a learned basis approximation [11, 14, 19, 28]. In Dynamic Gaussian Marbles [26], isotropic Gaussians are equipped with trajectories are learned through a divide-and-conquer optimization and by using a pretrained single-image depth prior. A local approach is taken in Spacetime Gaussians [16]: the temporal opacity of Gaussians is modeled through radial basis functions making them appearing and disappearing when needed. Finally, 4D Gaussian splatting [33] avoids using an explicit motion model at all by using 4D Gaussians, where the motion is a byproduct of time-conditioned projection to 3D space.

Gaussian splatting for other input domains. Gaussian splatting models are not restricted to only reconstruct from RGB color inputs. The method can be adapted in a non-RGB setting as well. Some works [2, 4, 10, 17, 24] focus on extending 3DGS framework to HDR, X-ray, sonar, or thermal images. However, without a sufficient variation in camera poses, high quality reconstruction remains a challenge. To address that, some methods [5] integrate an additional depth input to improve the quality. However, these methods all rely on the quality of the monocular depth supervision. Within an under-constrained and more brittle optimization than NeRFs, we contribute a way to effectively use Gaussian splatting with time-of-flight imaging.

References

[1] Benjamin Attal, Eliot Laidlaw, Aaron Gokaslan, Changil Kim, Christian Richardt, James Tompkin, and Matthew O'Toole. TöRF: Time-of-flight radiance fields for dynamic scene view synthesis. In *NeurIPS*, 2021. 1, 3, 5, 6, 7, 8, 14, 15, 16

[2] Yuanhao Cai, Yixun Liang, Jiahao Wang, Angtian Wang, Yulun Zhang, Xiaokang Yang, Zongwei Zhou, and Alan Yuille. Radiative Gaussian splatting for efficient X-ray novel view synthesis. In *ECCV*, 2024. 9

[3] David Charatan, Sizhe Lester Li, Andrea Tagliasacchi, and Vincent Sitzmann. pixelSplat: 3D Gaussian splats from image pairs for scalable generalizable 3D reconstruction. In *CVPR*, 2024. 1

[4] Qian Chen, Shihao Shu, and Xiangzhi Bai. Thermal3d-gs: Physics-induced 3d gaussians for thermal infrared novel-view synthesis. In *European Conference on Computer Vision*, pages 253–269. Springer, 2024. 9

[5] Jaeyoung Chung, Jeongtaek Oh, and Kyoung Mu Lee. Depth-regularized optimization for 3D Gaussian splatting in few-shot images. In *CVPR*, pages 811–820, 2024. 3, 9

[6] Antoine Guédon and Vincent Lepetit. SuGaR: Surface-aligned gaussian splatting for efficient 3D mesh reconstruction and high-quality mesh rendering. In *CVPR*, 2024. 7

[7] Mohit Gupta, Shree K Nayar, Matthias B Hullin, and Jaime Martin. Phasor imaging: A generalization of correlation-based time-of-flight imaging. *ACM Transactions on Graphics (ToG)*, 34(5):1–18, 2015. 2, 3

[8] Miles Hansard, Seungkyu Lee, Ouk Choi, and Radu Horaud. *Time-of-Flight Cameras: Principles, Methods and Applications*. Springer Publishing Company, Incorporated, 2012. 2

[9] Binbin Huang, Zehao Yu, Anpei Chen, Andreas Geiger, and Shenghua Gao. 2D Gaussian splatting for geometrically accurate radiance fields. In *SIGGRAPH*, 2024. 2, 4, 7, 11

[10] Xin Jin, Pengyi Jiao, Zheng-Peng Duan, Xingchao Yang, Chun-Le Guo, Bo Ren, and Chongyi Li. Lighting every darkness with 3DGS: Fast training and real-time rendering for HDR view synthesis. arXiv:2406.06216, 2024. 9

[11] Kai Katsumata, Duc Minh Vo, and Hideki Nakayama. An efficient 3D Gaussian representation for monocular/multi-view dynamic scenes. arXiv:2311.12897, 2023. 8

[12] Bernhard Kerbl, Georgios Kopanas, Thomas Leimkühler, and George Drettakis. 3D Gaussian splatting for real-time radiance field rendering. *ACM Transactions on Graphics*, 42 (4), 2023. 1, 3, 5, 8, 11

[13] Tzofi Klinghoffer, Xiaoyu Xiang, Siddharth Somasundaram, Yuchen Fan, Christian Richardt, Ramesh Raskar, and Rakesh Ranjan. PlatoNeRF: 3D reconstruction in Plato's cave via single-view two-bounce lidar. In *CVPR*, 2024. 8

[14] Agelos Kratimenos, Jiahui Lei, and Kostas Daniilidis. DynMF: Neural motion factorization for real-time dynamic view synthesis with 3D Gaussian splatting. In *ECCV*, 2024. 8

[15] Jiahe Li, Jiawei Zhang, Xiao Bai, Jin Zheng, Xin Ning, Jun Zhou, and Lin Gu. DNGaussian: Optimizing sparse-view 3D Gaussian radiance fields with global-local depth normalization. In *CVPR*, 2024. 1, 3

[16] Zhan Li, Zhang Chen, Zhong Li, and Yi Xu. Spacetime Gaussian feature splatting for real-time dynamic view synthesis. In *CVPR*, 2024. 8

[17] Zhihao Li, Yufei Wang, Alex Kot, and Bihan Wen. From chaos to clarity: 3DGS in the dark. arXiv:2406.08300, 2024. 9

[18] Yiqing Liang, Numair Khan, Zhengqin Li, Thu Nguyen-Phuoc, Douglas Lanman, James Tompkin, and Lei Xiao. GauFRe: Gaussian deformation fields for real-time dynamic novel view synthesis. arXiv:2312.11458, 2023. 1, 3, 8

[19] Youtian Lin, Zuozhuo Dai, Siyu Zhu, and Yao Yao. Gaussian-Flow: 4D reconstruction with dynamic 3D Gaussian particle. In *CVPR*, 2023. 8

[20] Jonathon Luiten, Georgios Kopanas, Bastian Leibe, and Deva Ramanan. Dynamic 3D Gaussians: Tracking by persistent dynamic view synthesis. In *3DV*, 2024. 8

[21] Simon Niklaus and Feng Liu. Softmax splatting for video frame interpolation. In *CVPR*, 2020. 5

[22] Mikhail Okunev, Marc Mapeke, Benjamin Attal, Christian Richardt, Matthew O'Toole, and James Tompkin. Flowed time of flight radiance fields. In *ECCV*, 2024. 1, 2, 3, 5, 6, 7, 8, 14, 15

[23] Matt Pharr, Wenzel Jakob, and Greg Humphreys. *Physically Based Rendering: From Theory to Implementation*. Elsevier, 3rd edition, 2016. 5

[24] Ziyuan Qu, Omkar Vengurlekar, Mohamad Qadri, Kevin Zhang, Michael Kaess, Christopher Metzler, Suren Jayasuriya, and Adithya Pediredla. Z-splat: Z-axis Gaussian splatting for camera-sonar fusion. arXiv:2404.04687, 2024. 9

[25] Aarrushi Shandilya, Benjamin Attal, Christian Richardt, James Tompkin, and Matthew O'Toole. Neural fields for structured lighting. In *ICCV*, 2023. 8

[26] Colton Stearns, Adam W Harley, Mikaela Uy, Florian Dubost, Federico Tombari, Gordon Wetzstein, and Leonidas Guibas. Dynamic Gaussian marbles for novel view synthesis of casual monocular videos. In *SIGGRAPH Asia Conference Papers*, 2024. 8

[27] Zachary Teed and Jia Deng. RAFT: Recurrent all-pairs field transforms for optical flow. In *ECCV*, 2020. 5

[28] Qianqian Wang, Vickie Ye, Hang Gao, Jake Austin, Zhengqi Li, and Angjoo Kanazawa. Shape of motion: 4D reconstruction from a single video. arXiv:2407.13764, 2024. 8

[29] 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. 8

[30] Tianyi Xie, Zeshun Zong, Yuxing Qiu, Xuan Li, Yutao Feng, Yin Yang, and Chenfanfu Jiang. PhysGaussian: Physics-integrated 3D Gaussians for generative dynamics. In *CVPR*, 2024. 8

[31] Yinghao Xu, Zifan Shi, Wang Yifan, Hansheng Chen, Ceyuan Yang, Sida Peng, Yujun Shen, and Gordon Wetzstein. GRM: Large Gaussian reconstruction model for efficient 3D reconstruction and generation, 2024. arXiv:2403.14621. 1

[32] Ziyi Yang, Xinyu Gao, Wen Zhou, Shaohui Jiao, Yuqing Zhang, and Xiaogang Jin. Deformable 3D Gaussians for high-fidelity monocular dynamic scene reconstruction. In *CVPR*, 2024. 1, 3, 4, 5, 6, 7, 8, 14, 15, 16

[33] Zeyu Yang, Hongye Yang, Zijie Pan, and Li Zhang. Real-time photorealistic dynamic scene representation and rendering with 4D Gaussian splatting. In *ICLR*, 2024. 8

[34] Zehao Zhu, Zhiwen Fan, Yifan Jiang, and Zhangyang Wang. FSGS: Real-time few-shot view synthesis using Gaussian splatting. In *ECCV*, 2024. 1

Time of the Flight of the Gaussians: Fast and Accurate Dynamic Time-of-Flight Radiance Fields

Supplementary Material

8. Supplemental Video + Full Figures

We have included a supplemental video showing all dataset sequences plus two ablation sequences. Further, we include full sequence qualitative figures for all three datasets at the back of this supplemental material; this reproduces some results from the main paper but groups and includes all results for completeness.

9. Ablations

We show quantitative (Tab. 2) ablations on F-TöRF synthetic scenes and also qualitative ablation results on F-TöRF real-world and synthetic scenes (Fig. 7). "No bias" refers to no consideration of either of our heuristics. "DD" means adding the depth distortion loss [9], and "H1" and "H2" means our control of the optimization using our occupancy bias (H1) and low initial reflectivity bias (H2). For result variants, 'd' refers to the mean depth from Eq. (6), and 'd_{ToF}' refers to the depth derived from the reconstructed quads using Eq. (1).

10. ToF Gradient Computation Details

We briefly explain the opacity gradient computation for our ToF image formation model, which is less trivial compared to other gradients computed via simple chain rules. Let L denote the loss function, and $\alpha_k = o_k \mathcal{G}_k^{2D}(\mathbf{x})$ the opacity term of the k-th Gaussian. Using the chain rule:

$$\frac{\partial L}{\partial \alpha_k} = \frac{\partial L}{\partial \mathbf{c}(\mathbf{x})} \frac{\partial \mathbf{c}(\mathbf{x})}{\partial \alpha_k}, \tag{S11}$$

the term $\frac{\partial \mathbf{c}(\mathbf{x})}{\partial \alpha_k}$ is computed recursively in the original 3D Gaussian Splatting (3DGS) method [12]:

$$\frac{\partial \mathbf{c}(\mathbf{x})}{\partial \alpha_k} = T_k(\mathbf{c}_k - \text{acc}_k), \tag{S12}$$

$$\text{acc}_k = \begin{cases} \alpha_{k+1}\mathbf{c}_{k+1} + (1-\alpha_{k+1})\text{acc}_{k+1}, & k < N, \\ 0, & k = N, \end{cases}$$

where T_k is the accumulated transmittance, $\mathbf{c}_k$ is the Gaussian's color, and acc_k aggregates contributions of later Gaussians.

In our model, this computation extends to ToF-specific signals like phasor or quad pixels ($\mathbf{p}(\mathbf{x}), \mathbf{q}(\mathbf{x})$) as described in Eq. (5) of the main paper. We take $\mathbf{q}(\mathbf{x})$) as an example, the corresponding recursion becomes to:

$$\frac{\partial \mathbf{q}(\mathbf{x})}{\partial \alpha_k} = T_k^2(\mathbf{q}_k + 2(\alpha_k - 1)\text{acc}_k^q), \tag{S13}$$

$$\text{acc}_k^q = \begin{cases} \alpha_{k+1}\mathbf{q}_{k+1} + (1-\alpha_{k+1})^2\text{acc}_{k+1}^q, & k < N, \\ 0, & k = N. \end{cases}$$

Here, $\mathbf{q}_k$ represents the quad contribution of the k-th Gaussian, and acc_k^q accumulates later quad contributions.

Table 2. **Ablation depth error on the F-TöRF synthetic dataset**. Each number is a depth MSE$\times$100. Bold marks the smallest result for each of d and d_{ToF}. On average, our approach is reliably more accurate without any catastrophic failures, though scene specific variations exist. *Sliding Cube* is too simple to induce large differences. *Ortho. ST* is the only sequence with large low reflectivity areas, and we see the advantage of our low initial reflectivity bias (H2) in the scene's reconstructed d. Using both heuristics H1, H2 plus the depth distortion (DD) loss theoretically should reduce the gap between d and d_{ToF}, but in practice, because of the spatial extent of the Gaussians, this usually led to oversmooth depth, worse thin structures, and is sometimes unstable because of the existence of large Gaussians representing large homogeneous textureless areas (Fig. 7).

		Mean	*Sliding Cube*	*Occ. Cube*	*Axial ST*	*3 Cubes ST*	*3 Chairs ST*	*Arcing Cube*	*Ortho. ST*
d_{ToF}	No bias	144.134	0.013	13.227	0.286	1.152	0.885	5.884	987.492
	+DD	146.189	0.119	11.422	5.413	7.289	1.224	13.415	984.440
	+H1	3.641	0.017	**0.024**	0.206	0.916	**0.605**	**1.113**	22.610
	+H1 +H2 (ours)	**3.360**	**0.008**	0.073	**0.154**	**0.294**	0.650	1.526	**20.818**
	+H1 +H2 +DD	255.717	0.012	148.578	0.422	1.010	0.895	1537.952	101.148
d	No bias	365.686	0.049	15.604	55.834	24.892	1.482	4.456	2457.487
	+DD	274.316	0.172	11.675	5.922	7.805	1.421	13.067	1880.148
	+H1	44.520	0.058	**0.310**	**0.726**	1.610	**1.011**	**1.025**	306.898
	+H1 +H2 (ours)	**5.824**	0.033	0.348	0.730	**1.059**	1.072	1.479	**36.048**
	+H1 +H2 +DD	310.564	**0.026**	147.856	0.787	1.362	1.106	1537.840	484.997

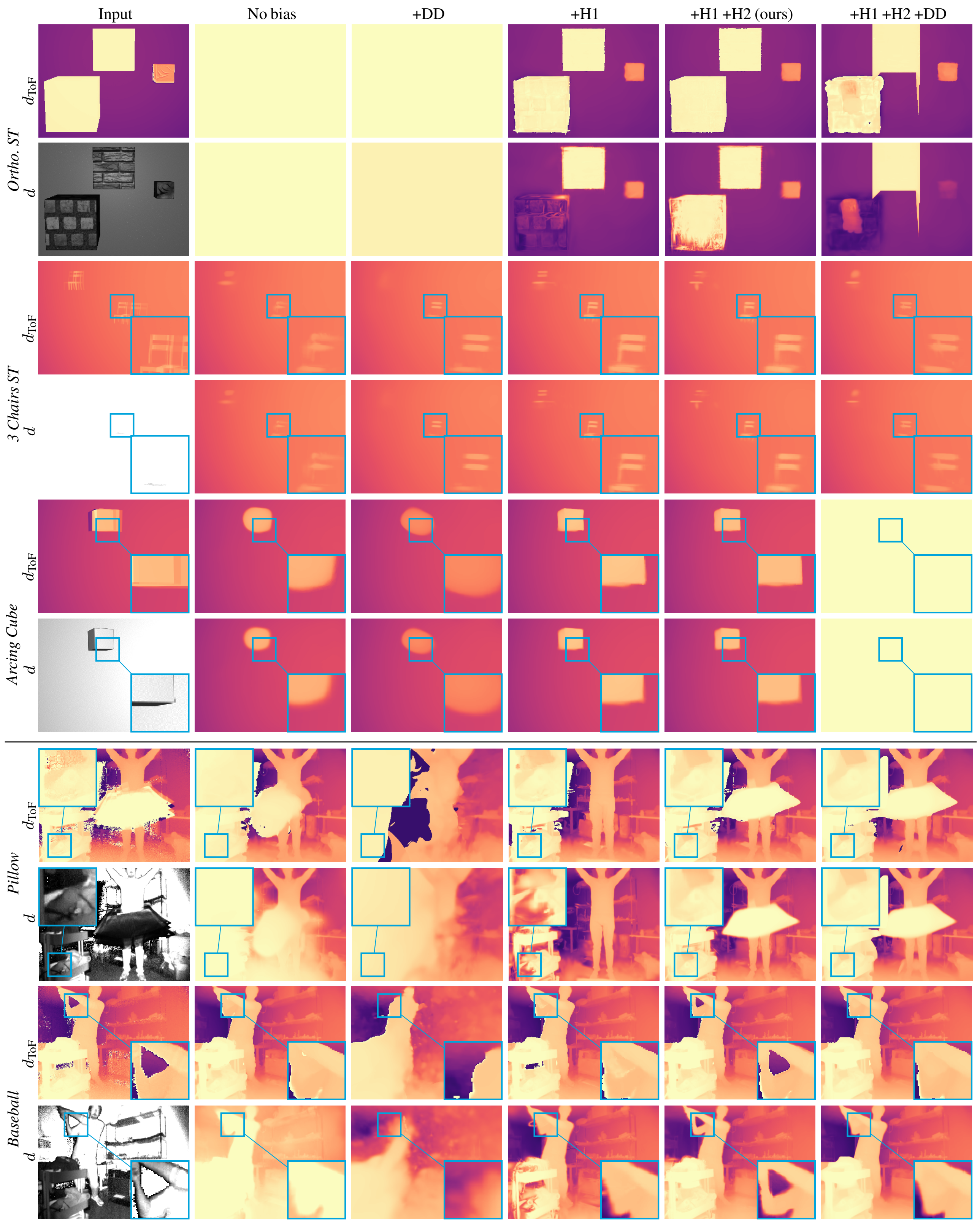

Figure 7. **Ablations**. In the first column, the even rows show the reflectivity map, which is computed as input amplitude multiplied with the square of input depth from ToF (light falloff), and can be understood as the expected of Gaussian reflectivity at the corresponding surface. For visualization, the map is clipped to the range [0, 1], where overexposed areas only indicate high target reflectivity. No bias led to arbitrary number of density peaks even for opaque surfaces, thus inaccurate depth from Eq. (6). Adding DD overly stack large Gaussians that led to overly smoothed mean depth. Occupancy bias improves the placement of Gaussians but still struggles in low-reflectivity areas (e.g., bottom left of the cart, and the pillow). Initializing Gaussians with low reflectivity mitigates this issue. Reintroducing the DD loss again after applying the two heuristics still led to oversmooth depth due to Gaussians' spatial extent.

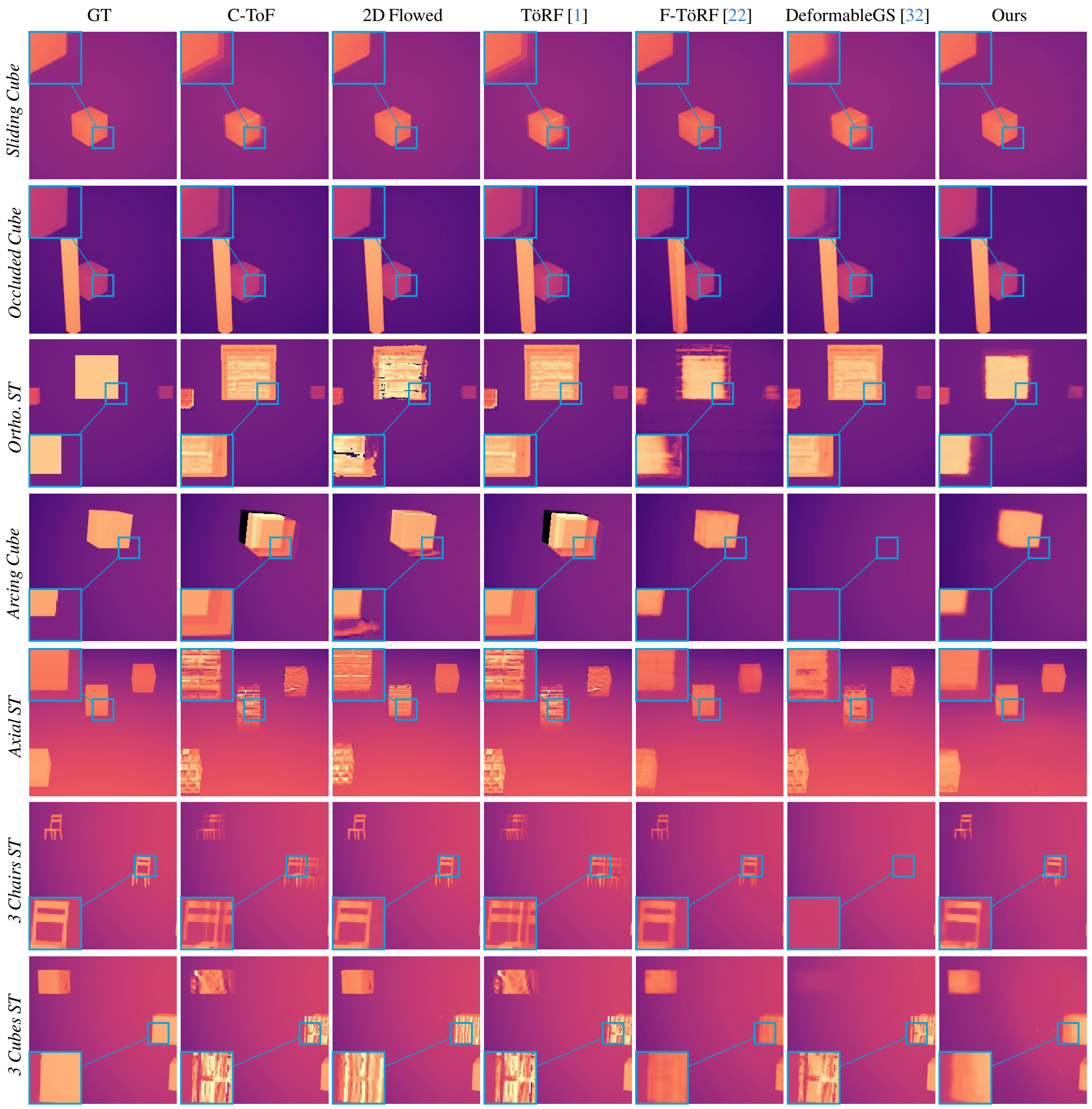

Figure 8. **Results on all F-TöRF synthetic scenes**. DeformableGS is given C-ToF-derived depth as an additional input.

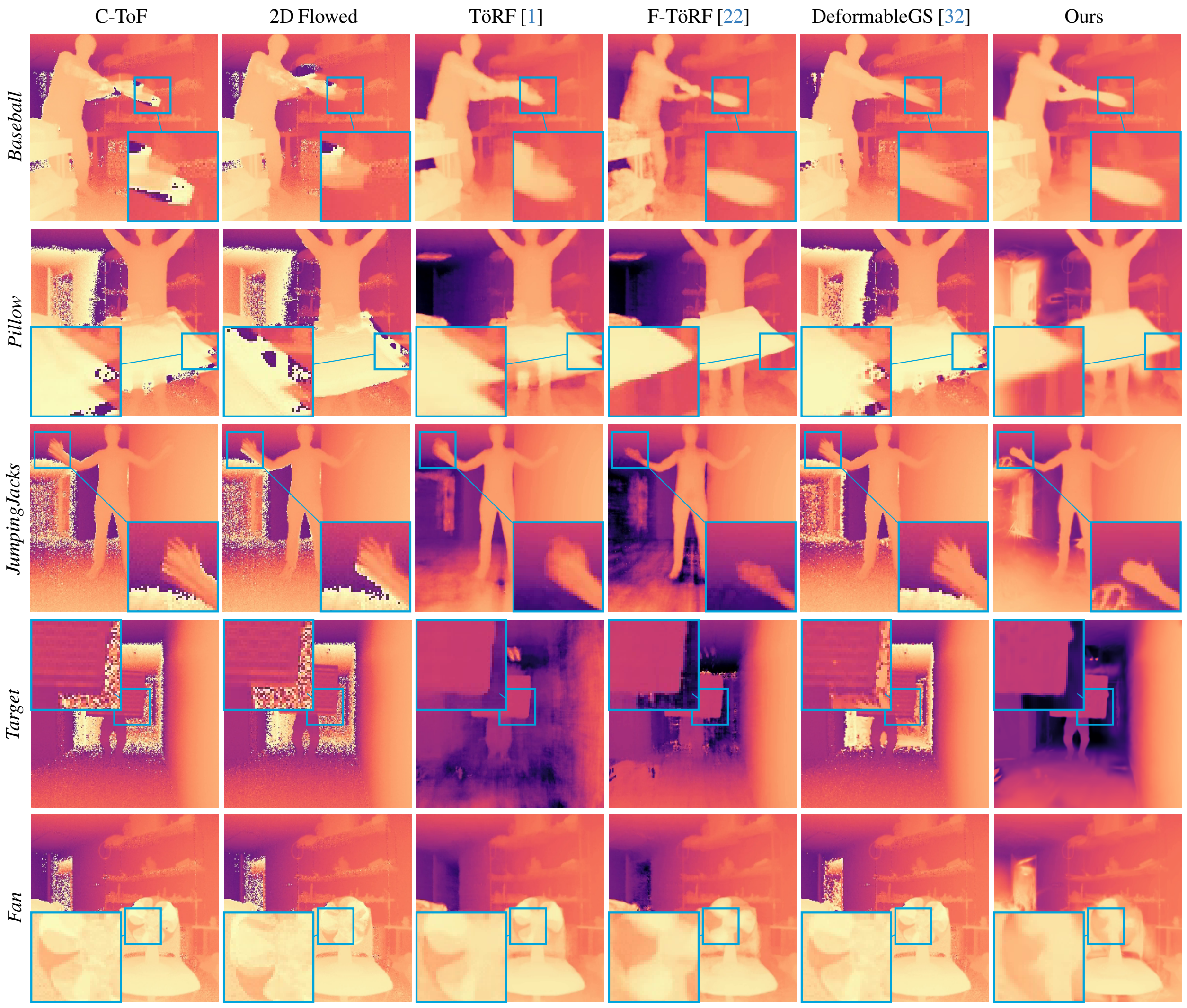

Figure 9. **Results on all F-TöRF real-world scenes**. DeformableGS is given C-ToF-derived depth as an additional input.

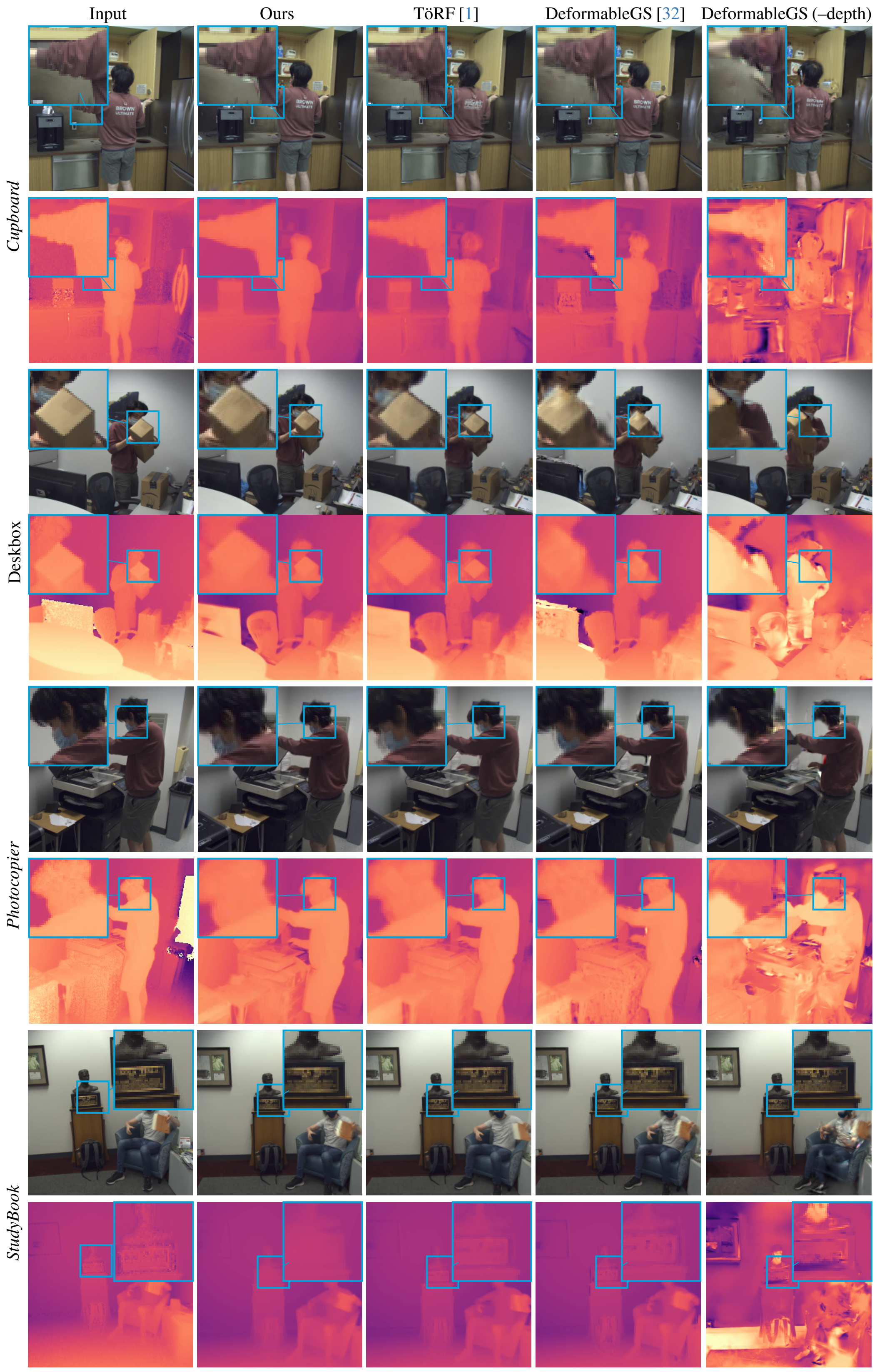

Figure 10. **More results on TöRF real-world scenes**. DeformableGS is given C-ToF-derived depth as an additional input.