Flowed Time of Flight Radiance Fields

Mikhail Okunev*[1], Marc Mapeke*[1], Benjamin Attal[2], Christian Richardt[3], Matthew O'Toole[2], and James Tompkin[1]

[1] Brown University
[2] Carnegie Mellon University
[3] Codec Avatars Lab, Meta

Abstract. Flowed time of flight radiance fields (F-TöRF) is a method to correct for motion artifacts in continuous-wave time of flight imaging (C-ToF). As C-ToF cameras must capture multiple exposures over time to derive depth, any moving object will exhibit depth errors. We formulate an optimization problem to reconstruct the raw frames captured by the camera via an underlying 4D volumetric scene and a physically-based differentiable C-ToF simulator. With weak optical flow supervision, we can infer a 3D volume with scene flow that explains the raw captures, even though any particular time instant does not provide sufficient constraints upon the depth or motion. On synthetic sequences, we find that our approach reduces depth errors on dynamic objects by up to $20\times$ compared to C-ToF, particularly for axial motions and large disparities (≥ 25 pixels) between raw frames. On real-world sequences, we see qualitatively similar gains with artifacts resolved on falling pillows and swinging baseball bats.

1 Introduction

Estimating depth is a core measurement task in imaging, providing opportunities in computational photography, media editing, AR/VR, and as a precursor to 3D scene reconstruction. Modern multi-camera systems can estimate depth via passive stereo, but depth is still difficult to estimate in textureless regions. Instead, many cameras like smartphones include active illumination time-of-flight (ToF) sensors to improve depth estimation [17]. One scheme is continuous-wave or correlation-based ToF (C-ToF): infrared light is emitted into the scene over time with sinusoidal amplitude at a particular frequency. Then, when the light is reflected back to the sensor, the camera captures three or more raw frames over time to estimate the phase and amplitude of the returned sinusoid per pixel. With the phase offset, we can compute depth given the emission frequency and the speed of light [7, 8]. C-ToF's active approach is significantly more accurate than passive stereo depth estimation, especially in textureless regions.

However, C-ToF's imaging model assumes that scenes and the camera are static: as we must estimate the phase offsets from multiple frames, depth is inaccurate if anything moves, causing large errors such as ghosting or blurring under fast motion. This affects objects with motions relative to the camera, and

All experiments were performed by university authors; CR's role was as an advisor.

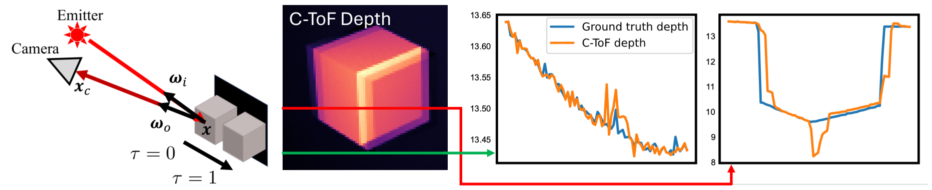


Fig. 1: Deriving depth from C-ToF samples over time causes motion artifacts. The cube's motion (*left*) invalidates C-ToF algorithm assumptions. Depth is derived both from light reflected from the cube and background, causing ghosts to appear (*center*). *Right:* C-ToF depth along the red and green lines, showing discrepancies.

also any motions of the camera itself relative to the scene—static or otherwise—such as handheld motions of a smartphone. Further, depth is estimated per pixel without correspondence over time; no 4D reconstruction is possible.

Existing approaches have attempted to mitigate motion errors. We might ignore moving regions by detecting pixels whose raw frames have inconsistencies [13], but this reduces the completeness of our estimate. Under small motion assumptions, we can use the temporal derivative to detect and correct for motion artifacts [31], or we could simplify scene motion assumptions [10, 12]. We might try to estimate 2D optical flow between the raw frames captured at different time instances, and then reproject all raw frames to a consistent time instance [22]. While optical flow can measure lateral motion, it cannot measure axial motion. Further, optical flow typically assumes brightness constancy, and any motion with an axial component will break this assumption. To counteract this, we might try to learn a correspondence function from data using a C-ToF simulator [30]. But, this can be brittle as it leaves a domain gap to real C-ToF data. Fundamentally, the 3D scene under motion must be known to correctly synthesize a raw frame at an unseen time so as to remove motion errors (Figure 1).

Our approach attempts to better principally model the image formation of raw C-ToF frames via a neural time-of-flight radiance field optimization [2, 24, 37]. Given raw C-ToF frames, we explain their intensities across time by rendering an underlying 4D scene (geometry and motion) using simulations of a C-ToF camera from a physically-based radiative transport model, without directly computing depth from C-ToF measurements via phase offsets. This allows us to avoid prematurely making decisions about where scene elements must be while motion is still unexplained. For motion, we induce a velocity field that explains how the 3D geometry moves [1]; this is possible by relaxing an initial 4D scene reconstruction that assumes synchronous raw frame capture to an asynchronous one, and by using weak supervision from a pre-trained 2D optical flow network that the projected 3D velocity must match. Once optimized, the representation provides depth estimates for moving objects with reduced ghosting.

We show our method's efficacy on seven synthetic sequences and on six real ones. Our approach is more effective than a neural time-of-flight radiance field method that does not account for motion [2], and is more effective for axial motions than a 2D optical flow interpolation method [26]. More broadly, as motion

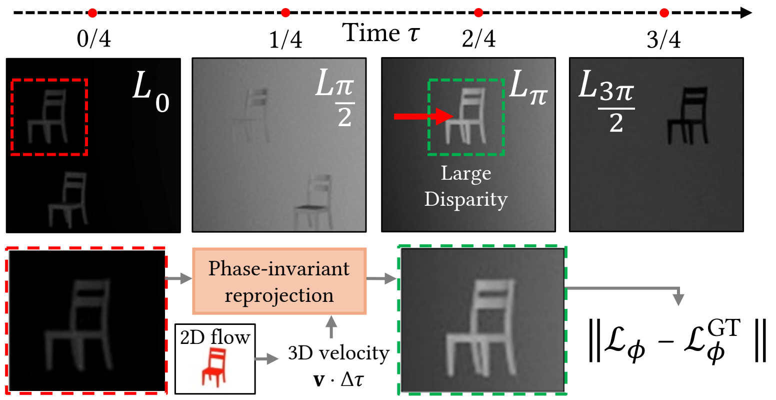

Time τ
0/4
1/4
2/4
3/4
L_0
$L_{\frac{\pi}{2}}$
L_π
$L_{\frac{3\pi}{2}}$
Large Disparity
Phase-invariant reprojection
2D flow
3D velocity $\mathbf{v} \cdot \Delta\tau$
$\|\mathcal{L}_\phi - \mathcal{L}_\phi^{\mathrm{GT}}\|$

Fig. 2: Method overview. The four raw frames captured sequentially over time τ (*top*) show large brightness differences due to the emission phase. The large disparity on moving objects causes depth artifacts. Recovering a 4D scene representation with help from 2D optical flow lets us correctly reproject one timestep onto another in a phase-aware way, such that we can supervise the scene at fractional time moments.

artifacts occur across many multi-shot imaging processes, we show a principled method to account for and correct them through 4D scene reconstruction.

Assumptions and Limitations. We assume a co-located emitter, that each raw frame is captured at separate time steps (no multi-tap), and that motion is locally linear; higher-order motion models exist and may improve quality. We use optical flow to weakly supervise our reconstruction, so flow estimation errors may propagate to our results. Our implementation builds upon standard NeRF [24] so the method is slow; many acceleration techniques exist and could be used.

2 Image Formation Model

2.1 C-ToF Camera Principles

First, we introduce C-ToF principles to motivate why motion artifacts exist. Many variations exist; we focus on the principles for our real-world setup (Appx. B).

To measure depth, a C-ToF camera system emits a periodic time-varying infrared light signal into a scene, typically modelled as a sinusoid $\sin(2\pi ft)$.[4] In an idealized model, the camera receives the reflected light and, through correlation with a reference sinusoidal signal, produces an image with pixel intensities of the form $A\sin(\psi + \phi) + B$. Here, the amplitude A reflects the amount of light

[4] Light intensity is more accurately modelled with a non-negative signal $\frac{1}{2}\sin(2\pi ft) + \frac{1}{2}$, but we opt to use the simpler model for clarity.

received at each pixel, the bias B depends on ambient illumination, and the phase ψ captures the time for which light is in flight. The offset ϕ represents a programmable temporal shift of the reference signal. To solve for the three unknowns A, B, and ψ, at least three intensity measurements are required, using different offsets ϕ. In practice, four offsets are typically used for robustness, where $\phi \in \{0, \frac{\pi}{2}, \pi, \frac{3\pi}{2}\}$, and the camera produces a quartet of raw frames $\mathbb{L}_\phi = \{A\sin(\psi+\phi)+B\}$.

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

$$d_{\text{ToF}} = \frac{c}{4\pi f}\psi \quad \text{where} \quad \psi = \arctan\left(\frac{L_0 - L_\pi}{L_{\frac{\pi}{2}} - L_{\frac{3\pi}{2}}}\right), \tag{1}$$

since

$$\arctan\left(\frac{L_0 - L_\pi}{L_{\frac{\pi}{2}} - L_{\frac{3\pi}{2}}}\right) = \arctan\left(\frac{A\sin(\psi)+B-(-A\sin(\psi)+B)}{A\cos(\psi)+B-(-A\cos(\psi)+B)}\right) \tag{2}$$

$$= \arctan\left(\tan(\psi)\right) = \psi. \tag{3}$$

The amplitude of the returned light can be computed as:

$$L_{\text{a}} = A = \tfrac{1}{2}\sqrt{(L_0 - L_\pi)^2 + (L_{\frac{\pi}{2}} - L_{\frac{3\pi}{2}})^2}. \tag{4}$$

The camera treats this quartet as representing one tick of time, e.g., a 30 Hz depth output requires capturing raw frames at 120 Hz. But, moving objects cause errors as the reflected light arriving at a pixel across the quartet $\mathbb{L}_\phi$ does not correspond to a single scene point, causing incorrect phase—and so depth—estimates (Figure 1). 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.

2.2 Volume-based C-ToF Image Formation

To fix these issues, we will infer an underlying 4D scene representation that can explain both geometry and motion. We consider the volume rendering model that represents a scene as a density field $\sigma(\mathbf{x})$ defined for each world point $\mathbf{x}$, and a radiance field $L_{\text{s}}(\mathbf{x}, \boldsymbol{\omega})$ representing the reflected radiance, defined also for each direction $\boldsymbol{\omega}$. The total light arriving into a camera point $\mathbf{x}_{\text{c}}$ from direction $\boldsymbol{\omega}_{\text{o}}$ can be defined by the ray integral [6]:

$$L(\mathbf{x}_{\text{c}}, \boldsymbol{\omega}_{\text{o}}) = \int_{t_{\text{n}}}^{t_{\text{f}}} T(\mathbf{x}_{\text{c}}, \mathbf{x}_t)\,\sigma(\mathbf{x}_t)\,L_{\text{s}}(\mathbf{x}_t, \boldsymbol{\omega}_{\text{o}})\,dt, \tag{5}$$

$$T(\mathbf{x}_{\text{c}}, \mathbf{x}_t) = \exp\left(-\int_{t_{\text{n}}}^{t} \sigma(\mathbf{x}_{\text{c}} - \boldsymbol{\omega}_{\text{o}} s)\,ds\right). \tag{6}$$

$T(\mathbf{x}_{\text{c}}, \mathbf{x}_t)$ describes the transmittance for light propagating from position $\mathbf{x}_{\text{c}}$ to $\mathbf{x}_t = \mathbf{x}_{\text{c}} - \boldsymbol{\omega}_{\text{o}} t$, for $t \in [t_{\text{n}}, t_{\text{f}}]$ within the near and far bounds. We can produce the depth along a ray similarly by replacing the reflected radiance L_s with the distance from the camera to the point, i.e., $\|\mathbf{x}_\mathrm{c} - \mathbf{x}_t\| = t$:

$$d(\mathbf{x}_\mathrm{c}, \boldsymbol{\omega}_\mathrm{o}) = \int_{t_\mathrm{n}}^{t_\mathrm{f}} T(\mathbf{x}_\mathrm{c}, \mathbf{x}_t)\, \sigma(\mathbf{x}_t)\, t\, dt. \tag{7}$$

Some approaches have used C-ToF-derived depth estimates (Equation 1) to directly supervise scene reconstruction [4, 25]. However, any errors in the supervising depth due to simplified image formation assumptions will persist in the reconstruction. Attal et al. [2] showed that Equation 5 can be more-principally adapted to model emitted light I from a time-of-flight system, assuming that the emitter is a point light co-located with the infrared sensor [2, 3]:

$$L_\mathrm{ToF}(\mathbf{x}_\mathrm{c}, \boldsymbol{\omega}_\mathrm{o}) = \int_{t_\mathrm{n}}^{t_\mathrm{f}} \frac{T(\mathbf{x}_\mathrm{c}, \mathbf{x}_t)^2}{t^2} \sigma(\mathbf{x}_t) I_\mathrm{s}(\mathbf{x}_t, \boldsymbol{\omega}_\mathrm{o}) W(2t)\, dt, \tag{8}$$

where I_s is the scattered intensity of the emitted light reflected back along the same ray ($\boldsymbol{\omega}_\mathrm{i} = \boldsymbol{\omega}_\mathrm{o}$), the transmittance term is squared due to light traveling twice the distance ($2t$) between camera origin $\mathbf{x}_\mathrm{c}$ and scene point $\mathbf{x}_t$, and the $1/t^2$ term is the point light's inverse square intensity falloff.

The path-length importance function $W(2t)$ weights the contribution of a light path of total length p. Pediredla et al. [27] show that this can represent C-ToF images using the phasor $W(p) = \exp\big(\mathrm{i}\frac{2\pi p f}{\mathrm{c}}\big)$. As the function $W(p)$ is complex-valued, the radiance $L_\mathrm{ToF}(\mathbf{x}, \boldsymbol{\omega}_\mathrm{o})$ will also be a complex-valued phasor image [7]. In practice, the supervising phasor images are created from the four real-valued raw frames under linear combination: $L_\mathrm{ToF} = (L_0 - L_\pi) - \mathrm{i}(L_{\frac{\pi}{2}} - L_{\frac{3\pi}{2}})$, where the phasor magnitude corresponds to the amplitude.

This model allows emitted light to integrate to produce a phasor to supervise the density field, rather than supervising depth directly. But, phasor creation assumes that raw frames are acquired simultaneously when, in fact, they are not.

2.3 Raw-Frame C-ToF Image Formation under Synchronous Capture

Let us expand the model to predict raw frames assuming, for now, unrealistic synchronous capture. This model is fine for static scenes or scenes with very slow motion. Under integration, this lets us recreate the intensity of raw frames via depth derived from the density field σ and an amplitude field L_a:

$$L_\phi(\mathbf{x}_\mathrm{c}, \boldsymbol{\omega}_\mathrm{o}) = \int_{t_\mathrm{n}}^{t_\mathrm{f}} \frac{T(\mathbf{x}_\mathrm{c}, \mathbf{x}_t)^2}{t^2} \sigma(\mathbf{x}_t) \phi(t) L_\mathrm{a}(\mathbf{x}_t, \boldsymbol{\omega}_\mathrm{o})\, dt, \tag{9}$$

where ϕ models the modulation of the light via one of the raw sample quads: $\{\sin(t), \cos(t), -\sin(t), -\cos(t)\}$. The amplitude L_a represents the remaining unknown spatially-varying brightness of the scene, including view-dependent effects. The model can also be extended to learn the bias B, but we omit it for simplicity. This formulation initially may seem underconstrained because both density σ and amplitude L_a are free to vary. However, both must be consistent between all four phase offsets and through the transmission and fall-off terms, such that the correct raw intensities are produced.

Neural Field, Rendering, and Reconstruction Loss. Both density σ and amplitude L_a are represented by a neural field using a simple MLP: $F_{\boldsymbol{\theta}} : (\mathbf{x}, \boldsymbol{\omega}) \rightarrow (\sigma(\mathbf{x}), L_\mathrm{a}(\mathbf{x}, \boldsymbol{\omega}))$. For volume rendering, we follow Mildenhall et al. [24]. Penalizing a reconstruction loss $\mathcal{L}_\phi = \|L_\phi - L_\phi^{GT}\|^2$ between the set of raw frames $\mathbb{L}_\phi$ and their renderings is sufficient to optimize σ to be meaningful. Given the density field, Equation 7 will render scene depth.

2.4 Asynchronous Capture via Dynamic Field Reprojection

For a dynamic scene, we can no longer expect each pixel across the quartet to image the same world point. In this case, we must form additional constraints over time such that quartet pixels can provide meaningful depth constraints. First, we extend Equation 9 to time by parameterizing our MLP by time τ:

$$L_\phi(\mathbf{x}_\mathrm{c}, \boldsymbol{\omega}_\mathrm{o}, \tau) = \int_{t_\mathrm{n}}^{t_\mathrm{f}} \frac{T(\mathbf{x}_\mathrm{c}, \mathbf{x}_t, \tau)^2}{t^2} \sigma(\mathbf{x}_t, \tau) \phi(t) L_\mathrm{a}(\mathbf{x}_t, \boldsymbol{\omega}_\mathrm{o}, \tau)\, dt, \tag{10}$$

Then, an ideal depth constraint would allow us to sample the three missing raw frames from the quartet at the time instance of the remaining raw frame. One might naïvely hope that an optimized MLP will produce a smooth function over time for this sampling task, but fast motion can induce large disparities that make this fail, and correspondence must be induced.

First, we will describe a correspondence model, then discuss its supervision. We introduce motion as a pair of instantaneous velocity fields of 3D vectors $\mathbf{v}_\mathrm{f}$, $\mathbf{v}_\mathrm{b}$ for forward and backward time [1, 19]. Since both fields are used in a symmetrical way, we will denote them $\mathbf{v}$ where appropriate to not complicate the notation. The full model becomes $F_{\boldsymbol{\theta}} : (\mathbf{x}, \boldsymbol{\omega}, \tau) \rightarrow (\sigma(\mathbf{x}, \tau), L_\mathrm{a}(\mathbf{x}, \boldsymbol{\omega}, \tau), \mathbf{v}(\mathbf{x}, \tau))$.

The 4D scene can be supervised via reprojection across time. We denote a field $f(\mathbf{x}, \tau, ...)$ being reprojected from $\tau = i$ to $\tau = j$ by the velocity field $\mathbf{v}$ as $f^{i\rightarrow j}$. Assuming $i < j$, the reprojection is defined as

$$\begin{gathered} f(\mathbf{x}_j, \tau = j, ...)^{i\rightarrow j} = f(\mathbf{x}_j^{j\rightarrow i}, \tau = i, ...) \\ \text{where } \mathbf{x}_j^{j\rightarrow i} = \mathbf{x}_j + \mathbf{v}_\mathrm{b}(\mathbf{x}_j, \tau = j) \cdot (j - i). \end{gathered} \tag{11}$$

Reprojection is defined symmetrically for $i > j$, using the forward flow. We use the nomenclature that L_0 is captured at an *integer* time moment, e.g., $L_{\phi=0}^{\tau=1}$ at $\tau = 1$; and that $L_{\frac{\pi}{2}}, L_\pi, L_{\frac{3\pi}{2}}$ are captured at *fractional* time moments, e.g., $\tau = 1.25, 1.5, 1.75$. For instance, to render a raw frame for $\phi = \frac{\pi}{2}$ at timestep $\tau = 1.25$, using the density and amplitude from timestep $\tau = 1$:

$$\begin{aligned} L_{\frac{\pi}{2}}^{1\rightarrow 1.25}(\mathbf{x}_\mathrm{c}, \boldsymbol{\omega}_\mathrm{o}, \tau = 1.25) = \int_{t_\mathrm{n}}^{t_\mathrm{f}} & \frac{T^{1\rightarrow 1.25}(\mathbf{x}_\mathrm{c}, \mathbf{x}_t)^2}{t^2} \cdot \\ & \sigma^{1\rightarrow 1.25}(\mathbf{x}_t) \cos(t) L_\mathrm{a}^{1\rightarrow 1.25}(\mathbf{x}_t, \boldsymbol{\omega}_\mathrm{o})\, dt, \end{aligned} \tag{12}$$

where $\boldsymbol{\omega}_\mathrm{o}$ is constant across both time moments.

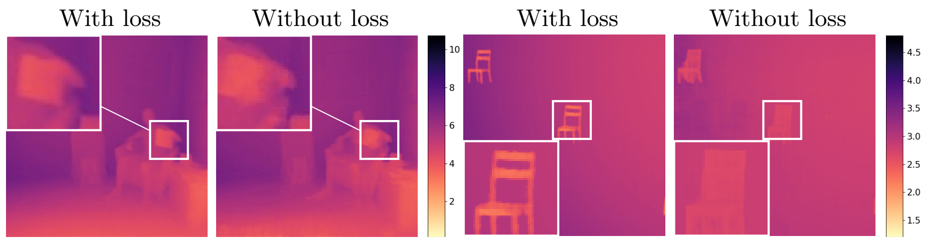


Fig. 3: Phase-invariant reprojection loss ablation. *Left:* In *StudyBook*, the model fails to reconstruct the outline of the dynamic object in the absence of the phase-invariant reprojection loss. *Right:* The model without the phase-invariant reprojection loss captures the outline of the object, but fails to reconstruct fine details, while also generating depth artifacts in the background.

Phase-invariant Reprojection. To supervise velocity, one might immediately turn to 2D optical flow, but there are challenges here. Due to the phase of the sinusoidal emission, raw frames will only have similar intensities at integer increments, e.g., $L_{\phi=0}^{\tau=1}$ and $L_{\phi=0}^{\tau=2}$, with significantly different intensities across fractional increments, e.g., between $L_{\phi=0}^{\tau=1}$ and $L_{\phi=\frac{\pi}{2}}^{\tau=1.25}$. This breaks typical brightness constancy assumptions from 2D motion estimators like optical flow, making it difficult to supervise motion between fine-grained fractional moments in time—even though this is needed to reconstruct fast motion. However, recovering a 4D representation via physically-based image formation *naturally aids this*: as raw frames only depend upon the scene density and amplitude, reprojection across time will automatically account for any intensity variation that would have occurred by the illumination phase variation. This makes it possible to reproject *any* fractional time moment to any other with correct intensity accommodation.

3 Optimization Losses

Phase-invariant Reprojection. We exploit this property of our model to supervise the 4D scene with sufficient depth and motion constraints. For each fractional time moment j, we have two nearby integer moments i_1 and i_2. If depth and motion are correct, then reprojecting the scene to time j and rendering it will reproduce the raw frame from the camera. These two constraints supervise density and velocity: $\mathcal{L}_{\phi}^{i_1\to j}$ and $\mathcal{L}_{\phi}^{i_2\to j}$, where $\mathcal{L}_{\phi}^{i\to j} = \left\| L_{\phi}^{i\to j}(\mathbf{x}_{\mathrm{c}}, \boldsymbol{\omega}_{\mathrm{o}}, \tau=j) - L_{\phi}^{\mathrm{GT}}(\mathbf{x}_{\mathrm{c}}, \boldsymbol{\omega}_{\mathrm{o}}, \tau=j) \right\|^2$ for each fractional raw frame of the quartet $\mathbb{L}_{\phi}$.

Penalizing this phase-invariant reprojection loss across fractional timesteps—as would normally be difficult as raw frames exhibit large intensity variations—allows us to resolve finer details under motions. To show this, we compare to only penalizing a reprojection loss across integer timespans (Figure 3). Our approach improves depth edges on dynamic objects: the detail of the two back slats is recovered with the loss, where only an incorrect back plane is recovered otherwise.

Optical Flow Supervision. As fast motions induce large disparities, we must still supervise the velocity field. We can only do this across integer timespans as weaker supervision. We use RAFT [33] to compute optical flow $\mathbf{u}_\mathrm{F}$ between L_0^i and L_0^{i+1}. Then, this 2D optical flow prediction $\mathbf{u}_\mathrm{F}$ should match the (2D) perspective projection of the velocity field $\mathbf{u}$:

$$\mathbf{u}(\mathbf{x}_\mathrm{c}, \boldsymbol{\omega}_\mathrm{o}, \tau = i) = \mathit{\Pi}_i \left(\int_{t_\mathrm{n}}^{t_\mathrm{f}} T(\mathbf{x}_\mathrm{c}, \mathbf{x}_t) \sigma(\mathbf{x}_t) \mathbf{v}(\mathbf{x}_t, \tau = i) \, dt \right), \tag{13}$$

where $\mathit{\Pi}_i$ is the perspective projection for the camera at time i.

The motion reconstruction loss is defined as $\mathcal{L}_\mathbf{u} = \sum_i \|\mathbf{u}_i - \mathbf{u}_{\mathrm{F}i}\|^2$, where i is an integer timestep. As optical flow prediction produces both forwards and backwards flow, we penalize this loss using both directions. This helps to correct for occlusion and disocclusion artifacts that otherwise could appear in the density field. As optical flow estimates may be in error, we only use this supervision with a small weight $\lambda_\mathbf{u}$ and allow the model to correct the velocity through the reprojected raw frame constraints.

One might think to supervise scene flow directly. While scene flow estimators that use depth input do exist [34], this would require resolving depth *first*—this is exactly the process we wish to avoid as resolving depth induces motion artifacts.

Two-stage Optimization. Simultaneously estimating density and velocity is underconstrained for monocular capture, but careful staging within the optimization can lead to reasonable solutions for depth and motion. As such, we initially optimize the scene via Equation 10 by assuming that the quartet of raw frames were captured synchronously. This produces blurry and ghosted density in dynamic regions and is similar to previous work that does not account for fractional time samples [2]. Then, we use Equation 12 for the remainder of training.

Stage 1 of the optimization penalizes raw ToF images from Equation 10:

$$\mathcal{L} = \mathcal{L}_0 + \mathcal{L}_{\frac{\pi}{2}} + \mathcal{L}_\pi + \mathcal{L}_{\frac{3\pi}{2}}, \tag{14}$$

where $\mathcal{L}_\phi = \sum_i \left\| L_\phi^{\tau=i} - L_{\phi,\mathrm{GT}}^{\tau=i} \right\|^2$ and i is an integer timestep.

For stage 2, we penalize:

$$\mathcal{L} = \sum_{i,j} \lambda_\phi \mathcal{L}_\phi^{i \to j} + \lambda_\mathbf{u} \mathcal{L}_\mathbf{u}, \tag{15}$$

where i is an integer timestep, and j are fractional timesteps such that $|i - j| < 1$. As L_0^i are integer timestep aligned, we can skip their reprojection.

4 Experiments

Datasets. For quantitative evaluation, we generate a set of seven didactic scenes in Blender at a resolution of 320×240. These scenes are so simple that any method has 'nowhere to hide' in its depth recovery; motions are fast and artifacts are prevalent. The scenes show cubes undergoing axial, lateral, and rotational motions of different speeds to induce varying disparities, undergoing occlusion, both with and without texture, and with chair objects to assess thin features. The scenes are strictly monocular without camera motion. For the fastest scenes (*3 Cubes Speed Test*, *3 Chairs Speed Test*, *Arcing Cube*, *Axial Speed Test*), the disparity ranges up to 127 pixels across integer timesteps, and 9–18 pixels for slower scenes (*Sliding Cube*, *Occluded Cube*, *Orthogonal Speed Test*). Three scenes test large axial motion (*Arcing Cube*, *Axial Speed Test*, *Orthogonal Speed Test*).

We use a C-ToF simulator based on the physically-based path tracer PBRT [28] with multi-bounce and scattering effects. Some render noise exists.

For real-world evaluation, we capture five sequences with fast motion: *Pillow*, *Baseball*, *JumpingJacks*, *Target*, and *Fan* with a Texas Instruments OPT8241 sensor (320×240 at 30 fps) with 5 m unambiguous depth range. We also use the existing *StudyBook* sequence from Attal et al. [2]—most of their published sequences contain only small and slow motions, but this one induces artifacts. This sequence uses a complementary color camera (see supplement). For *StudyBook* only, the camera moves. We use camera poses provided by the sequence authors.

Methods. **C-ToF Depth** denotes depth produced by the camera itself through integration of the four captured images (Section 2.1). **2D Flowed Raw Frames** attempts to align raw frames through optical flow warping before subsequent integration. We compute RAFT flow between matching phase offsets, then warp using Softmax Splatting [26]. For example, we estimate flow between $L_1^{1.25}$ and $L_1^{2.25}$, and warp to produce $L_1^{1.25\to 2}$. Repeating this across all phase offsets lets us integrate $\{L_0^2, L_1^{1.25\to 2}, L_2^{1.50\to 2}, L_3^{1.75\to 2}\}$ to produce an estimate at timestep $\tau = 2$. **TöRF** is the previous phasor-based work of Attal et al. [2] that (incorrectly) assumes synchronous capture. This is similar to Stage 1 of our approach, but trained to convergence. TöRF uses separate networks for static and dynamic parts and blends them together. For fair comparison, we disable the static network; in principle, it could be added to our method too.

For our method, we render two depth outputs: d or 'Ours (Depth)', and d_{ToF} or 'Ours (ToF)'. d is produced by volume rendering density σ via Eq. 7. As σ is only indirectly optimized via the reconstruction of $\mathbb{L}_\phi$, d can be a little soft (especially for static camera sequences). Thus, we also derive d_{ToF} from the volume rendered $\mathbb{L}_\phi$ via Eq. 1, which maintains sharpness as individual raw frames have high PSNR (≈30–40 db).

Metrics. For synthetic scenes, we compute MSE against ground-truth rendered depth, and MSE for dynamic regions using ground-truth rendered optical flow.

Please see supplemental material for pre-processing and optimization details.

4.1 Results

Quantitative Results. In scenes with strong axial motion (*Arcing Cube*, *Axial Speed Test*), our approach of reconstructing a 4D scene fares better than the 2D Flowed baseline (Table 3). This is expected as the approach can better resolve motion in depth thanks to the scene flow estimation. With axial-only motion (*Orthogonal Speed Test*), no method fares particularly well still. In sequences

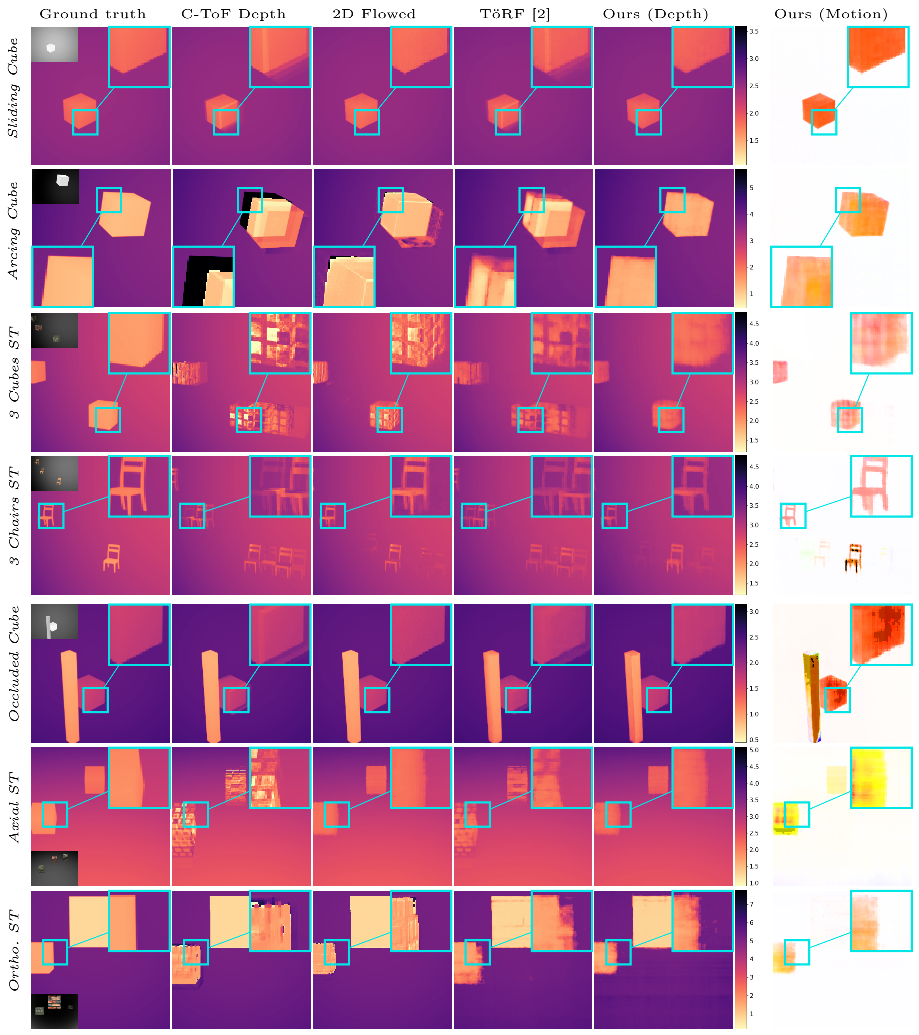


Fig. 4: Our approach reduces ghosting on dynamic objects by recovering scene flow. Synthetic scenes showing different motion characteristics across methods. C-ToF causes ghosting on moving objects. Our approach is able to reconstruct the apparent motion better than a volume integration method that does not account for asynchronous raw frame capture (TöRF [2]), and similarly to a 2D flow method [26]. For large axial motions, our approach tends to improve over the 2D flow baseline. Inset on ground truth: Corresponding RGB frame.

Table 1: 4D reconstruction from raw frames reduces depth error. Synthetic scenes; showing MSE×100. The camera is fixed, so error will mostly exist on dynamic objects. We report all-region error and dynamic-region error (with % pixels across all frames). 'Ours d_{ToF}' refers to rendering raw frames via our 4D reconstruction and then deriving depth via Eq. 1, while 'Ours' refers to rendering density σ as depth via Eq. 7.

Scene		Depth from Eq. 1			Depth from Eq. 7	
		C-ToF	2D flow	Our d_{ToF}	TöRF	Our d
Arcing Cube	Dyn. 6.70%	303.350	15.570	**6.824**	76.179	13.310
Large axial+lateral+rot. motion	All	36.768	1.678	**0.470**	6.278	1.256
Axial Speed Test	Dyn. 7.51%	20.147	5.912	**1.999**	8.193	3.430
Varying axial motions	All	2.068	0.662	**0.251**	1.143	0.938
Orthogonal Speed Test	Dyn. 9.71%	322.139	101.369	131.710	172.835	**33.021**
Varying axial-only motion	All	41.931	19.805	27.488	22.855	**7.527**
Sliding Cube	Dyn. 4.17%	1.575	**0.119**	0.195	2.247	1.080
Lateral+minor axial motion	All	0.096	**0.018**	0.023	0.349	0.440
3 Cubes Speed Test	Dyn. 7.65%	18.440	6.320	**3.323**	12.681	6.268
Textured; varying lateral motions	All	3.131	0.916	**0.501**	2.363	1.390
3 Chairs Speed Test	Dyn. 4.68%	10.686	**2.562**	3.647	9.979	5.540
Thin lines, varying lateral motions	All	0.787	**0.229**	0.324	0.956	0.855
Occluded Cube	Dyn. 3.63%	1.179	**0.247**	0.519	0.866	0.809
Lateral+minor axial motion	All	0.130	**0.088**	0.114	0.388	0.647
Avg. reduction vs. C-ToF	Dyn.	-	7.3×	10.8×	1.8×	6.6×
As error factor, higher is better	All	-	5.8×	14.6×	1.8×	5.8×

with lateral motion, both our method and the 2D Flowed baseline are effective at reducing motion error. Volume rendering depth d is less sharp, increasing error over d_{ToF}. Finally, *Occluded Cube* is more challenging for our method.

Qualitative Results. Our method resolves the motion artifacts in most scenes, producing sharper reconstructions on dynamic objects like a falling pillow and a swinging baseball bat (Figure 4 and Figure 5). The scenes with strong axial motion (*Arcing Cube*, *Axial Speed Test*, *Orthogonal Speed Test*) show clearer resolution of motion artifacts than baselines, with more correct depth and without strong ghosting. Concerning textured objects, both synthetic *3 Cubes Speed Test* and real *Target* show false texture entering the depth reconstruction in baselines: even though both scenes show planar objects with lateral motion, as the returned intensity of the signal varies under motion from the texture, the induced depth is incorrect. In our reconstruction, this artifact is removed. *Fan* is a challenging scene since the fast rotational movement of the fan breaks our linear motion assumption. Still, our method produces sharp outlines of fan blades, but generates an incorrect depth prediction within the swept volume of the fan's blades. Another challenging scene is *3 Chairs Speed Test*, where our model and 2D Flowed generates mild

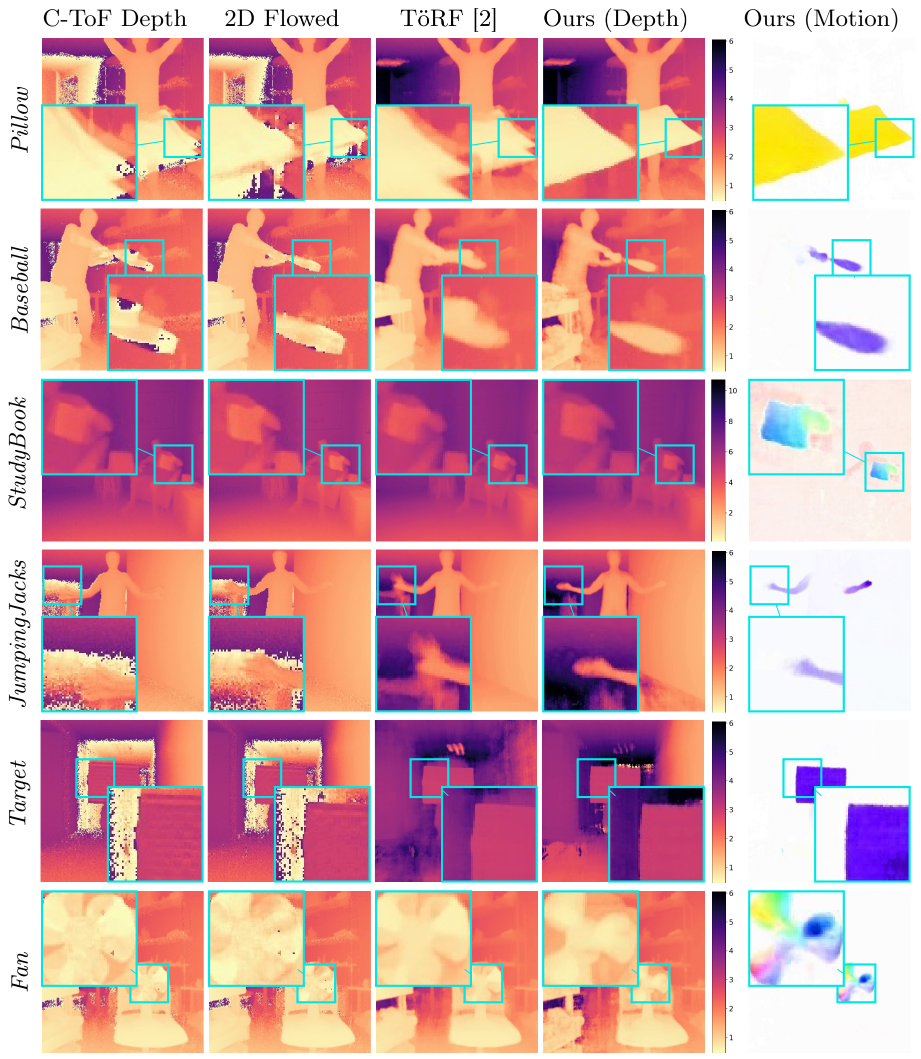


Fig. 5: Our approach reduces ghosting on dynamic objects by recovering scene flow in real scenes. Real scenes showing different motion characteristics across methods. Our method resolves motion artifacts better than baselines.

ghosting artifacts for the fastest chair ($\geq$25 pixel disparity per raw frame), but reproduces the other two chairs correctly. Finally, both neural reconstruction methods in the test sometimes struggle to accurately place density in the scene, e.g., the floor of *Target*.

Please see our supplemental video for most of our results, including raw quad reconstructions, ablations, and superresolved time by interpolating scene flow.

5 Discussion

Co-located Light. The current model assumes that the light emitter is co-located with the camera, when in fact the light source is to the side of the camera. This induces slight inaccuracies to path length. More importantly, this can introduce shadows around regions with depth discontinuities. In practice, our scene depths are far enough that these effects can be ignored.

Network Capacity. In synthetic scenes with low disparity (*Sliding Cube* and *Occluded Cube*), we note that TöRF demonstrates lower full frame depth error than our method (Table 3). We suppose this is because our MLP must additionally estimate a bi-directional velocity field with the same model capacity.

Static/Dynamic Separation. Some dynamic scene representations model static and dynamic scene parts separately [2, 19]. This is effective under large baselines, but challenging when baselines are small, requiring tricky regularization [36] or additional semantic information [20]. As C-ToF may be noisy, this problem is exacerbated. As such, we leave this for future investigation.

Flow at Every Integer Timespan. It is possible to supervise the 3D velocity field across all pairs of matching phase offsets across integer timespans (e.g., $L_0^i \rightarrow L_0^{i+1}$, $L_1^{i+0.25} \rightarrow L_1^{i+1.25}$). However, this can quickly become computationally prohibitive: GPU memory is a bottleneck in our system thanks to the $4\times$ temporal resolution, requiring expensive gradient paths back to the scene from every pixel; it would require supervising our model at $4\times$ the number of time moments (integer and fractional) as opposed to only integer time moments.

Estimating Scattering Functions and Normals. When illuminating a scene with a co-located point source, it is possible to estimate surface normal through shading cues [3, 32]. In principle, this can also be achieved with a C-ToF camera, by modelling the scattering function f_p and normal $\mathbf{n}$ explicitly to account for shading variations. However, even though much progress has been made in recovering surface normals [35], estimating the normal typically requires large variation in the illumination or viewpoint, which is not present in our case.

6 Related Work

Active Illumination and Neural Fields. Neural fields and volume rendering can accurately simulate physically-based models for active illumination, improving flexibility for the 3D reconstruction of complex scenes and objects. For example, simply co-locating a point light source with a camera enables more accurate recovery of surface normals and reflectance [3]. By structuring the incident illumination, Li et al. [18] differentiably render multi-view structured light pattern images to reconstruct 3D objects with a neural signed distance field, and Shandilya et al. [32] demonstrate that neural fields can explicitly model raw structured light images to recover scene geometry and additional scene properties such as surface normals, and direct and indirect lighting components.

Time-of-flight cameras use the co-located illumination setup to recover scene depth by measuring the time required for light to travel from the source, to the scene, and back to the sensor. Neural fields based off of time-of-flight imaging has been explored for C-ToF cameras [2], LiDARs [11], and SPAD sensors that form transient measurements [23]. PlatoNeRF [16] directly models optical paths of two-bounce signals captured from single-photon LiDAR [9], allowing reconstruction of both visible and occluded geometry. In contrast to these works, we focus on developing an approach to account for fast dynamics, specifically for the case of C-ToF cameras that use multiple exposures to recover depth.

Motion Correction in Time-of-Flight Imaging. Several methods, such as Lindner and Kolb [22], compensate for the motion artifacts based on 2D optical flow only. However, the raw ToF measurements do not fulfill brightness constancy requirements expected for optical flow, so normalization tricks are typically applied. To tenably correct for errors, one approach is to constrain the motion artifact corrections to certain types of motion. For example, the method by Hussman et al. [12] is restricted to small (less than 1 meter) linear motion along conveyor belts. Other methods, such as Hoegg et al.'s [10], restrict the motion artifacts to only blurred areas. Schmidt's method [31] detects and corrects motion artifacts in ToF measurements based solely on temporal relations and temporal derivatives in the raw ToF measurements. Their method is simple to implement, and avoids expensive spatial operations. However, spatial information (such as from the raw ToF measurements) captures additional cues that we can exploit in ToF corrections, which we use in this work.

7 Conclusion

C-ToF cameras form depth measurements through multiple exposures, making them susceptible to fast scene dynamics and producing depth errors as a result. To overcome this, we present an approach to reconstruct a 4D neural time-of-flight radiance field from raw frames. By explicitly modeling these individual exposures, and through the use of a phase-invariant reprojection loss, we demonstrate the ability to reduce depth errors on dynamic objects even with large disparities, as demonstrated on both synthetic and real-world sequences. Besides C-ToF cameras, there are several other imaging techniques that rely on multiple exposures, including structured light systems [29], multi-exposure high-dynamic range (HDR) imaging [14], and low-light imaging [21]. Our framework for dealing with dynamics can potentially be extended to these other imaging regimes.

Acknowledgements MO, JT thank NSF CAREER 2144956 and Cognex. MM acknowledges support from a Jack Kent Cooke Foundation scholarship, and BA from a Meta Research PhD Fellowship. MOT thanks NSF CAREER 2238485.

References

1. Attal, B., Huang, J.B., Richardt, C., Zollhoefer, M., Kopf, J., O'Toole, M., Kim, C.: HyperReel: High-fidelity 6-DoF video with ray-conditioned sampling. In: CVPR (2023)
2. Attal, B., Laidlaw, E., Gokaslan, A., Kim, C., Richardt, C., Tompkin, J., O'Toole, M.: TöRF: Time-of-flight radiance fields for dynamic scene view synthesis. In: NeurIPS (2021)
3. Bi, S., Xu, Z., Srinivasan, P., Mildenhall, B., Sunkavalli, K., Hašan, M., Hold-Geoffroy, Y., Kriegman, D., Ramamoorthi, R.: Neural reflectance fields for appearance acquisition (2020), arXiv:2008.03824
4. Deng, K., Liu, A., Zhu, J.Y., Ramanan, D.: Depth-supervised NeRF: Fewer views and faster training for free. In: CVPR (2022)
5. Georgiev, M., Bregović, R., Gotchev, A.: Fixed-pattern noise modeling and removal in time-of-flight sensing. IEEE Transactions on Instrumentation and Measurement **65**(4), 808–820 (2015)
6. Glassner, A.S.: Principles of Digital Image Synthesis. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA (1994)
7. Gupta, M., Nayar, S.K., Hullin, M.B., Martin, J.: Phasor imaging: A generalization of correlation-based time-of-flight imaging. ACM Trans. Graph. **34**(5), 156:1–18 (2015). `https://doi.org/10.1145/2735702`
8. Hansard, M., Lee, S., Choi, O., Horaud, R.P.: Time-of-Flight Cameras: Principles, Methods and Applications. Springer (2012). `https://doi.org/10.1007/978-1-4471-4658-2`
9. Henley, C., Hollmann, J., Raskar, R.: Bounce-flash lidar. IEEE Transactions on Computational Imaging **8**, 411–424 (2022). `https://doi.org/10.1109/TCI.2022.3174802`
10. Hoegg, T., Lefloch, D., Kolb, A.: Real-time motion artifact compensation for pmd-tof images. Proceedings of the Workshop on Imaging New Modalities, German Conference of Pattern Recognition (GCPR) **8200**, 273–288 (05 2013)
11. Huang, S., Gojcic, Z., Wang, Z., Williams, F., Kasten, Y., Fidler, S., Schindler, K., Litany, O.: Neural LiDAR fields for novel view synthesis. In: ICCV (2023)
12. Hussmann, S., Hermanski, A., Edeler, T.: Real-time motion artifact suppression in TOF camera systems. IEEE Transactions on Instrumentation and Measurement **60**(5), 1682–1690 (2011). `https://doi.org/10.1109/TIM.2010.2102390`
13. Jimenez, D., Pizarro, D., Mazo, M.: Single frame correction of motion artifacts in PMD-based time of flight cameras. Image and Vision Computing **32**(12), 1127–1143 (2014). `https://doi.org/10.1016/j.imavis.2014.08.014`
14. Kalantari, N.K., Ramamoorthi, R.: Deep HDR video from sequences with alternating exposures. Comput. Graph. Forum **32**(2), 193–205 (2019). `https://doi.org/10.1111/cgf.13630`
15. Kingma, D.P., Ba, J.: Adam: A method for stochastic optimization. In: ICLR (2015)
16. Klinghoffer, T., Xiang, X., Somasundaram, S., Fan, Y., Richardt, C., Raskar, R., Ranjan, R.: PlatoNeRF: 3D reconstruction in Plato's cave via single-view two-bounce lidar. In: CVPR (2024)
17. Kolb, A., Barth, E., Koch, R., Larsen, R.: Time-of-flight cameras in computer graphics. Comput. Graph. Forum **29**(1), 141–159 (2010). `https://doi.org/10.1111/j.1467-8659.2009.01583.x`
18. Li, C., Hashimoto, T., Matsumoto, E., Kato, H.: Multi-view neural surface reconstruction with structured light. In: BMVC (2022), `https://bmvc2022.mpi-inf.mpg.de/0550.pdf`

19. Li, Z., Niklaus, S., Snavely, N., Wang, O.: Neural scene flow fields for space-time view synthesis of dynamic scenes. In: CVPR (2021)
20. Liang, Y., Laidlaw, E., Meyerowitz, A., Sridhar, S., Tompkin, J.: Semantic attention flow fields for monocular dynamic scene decomposition. In: ICCV (2023)
21. Liba, O., Murthy, K., Tsai, Y.T., Brooks, T., Xue, T., Karnad, N., He, Q., Barron, J.T., Sharlet, D., Geiss, R., et al.: Handheld mobile photography in very low light. ACM Trans. Graph. **38**(6), 164–1 (2019)
22. Lindner, M., Kolb, A.: Compensation of motion artifacts for time-of-flight cameras. In: Workshop on Dynamic 3D Imaging. LNCS, vol. 5742, pp. 16–27 (2009). https://doi.org/10.1007/978-3-642-03778-8_2
23. Malik, A., Mirdehghan, P., Nousias, S., Kutulakos, K., Lindell, D.: Transient neural radiance fields for lidar view synthesis and 3D reconstruction. In: NeurIPS (2024)
24. Mildenhall, B., Srinivasan, P.P., Tancik, M., Barron, J.T., Ramamoorthi, R., Ng, R.: NeRF: Representing scenes as neural radiance fields for view synthesis. Communications of the ACM **65**(1), 99–106 (2021)
25. Neff, T., Stadlbauer, P., Parger, M., Kurz, A., Mueller, J.H., Chaitanya, C.R.A., Kaplanyan, A., Steinberger, M.: DONeRF: Towards real-time rendering of compact neural radiance fields using depth oracle networks. Computer Graphics Forum **40**(4), 45–59 (2021)
26. Niklaus, S., Liu, F.: Softmax splatting for video frame interpolation. In: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition. pp. 5437–5446 (2020)
27. Pediredla, A., Veeraraghavan, A., Gkioulekas, I.: Ellipsoidal path connections for time-gated rendering. ACM Trans. Graph. **38**(4), 38:1–12 (2019). https://doi.org/10.1145/3306346.3323016
28. Pharr, M., Jakob, W., Humphreys, G.: Physically Based Rendering: From Theory to Implementation. Elsevier, 3rd edn. (2016), https://www.pbr-book.org/
29. Salvi, J., Pagès, J., Batlle, J.: Pattern codification strategies in structured light systems. Pattern Recognition **37**(4), 827–849 (2004)
30. Schelling, M., Hermosilla, P., Ropinski, T.: Weakly-supervised optical flow estimation for time-of-flight. In: WACV (2023)
31. Schmidt, M.: Analysis, Modeling and Dynamic Optimization of 3D Time-of-Flight Imaging Systems. Ph.D. thesis, University of Heidelberg (2011)
32. Shandilya, A., Attal, B., Richardt, C., Tompkin, J., O'Toole, M.: Neural fields for structured lighting. In: ICCV. pp. 3512–3522 (2023). https://doi.org/10.1109/ICCV51070.2023.00325
33. Teed, Z., Deng, J.: RAFT: Recurrent all pairs field transforms for optical flow. In: ECCV (2020)
34. Teed, Z., Deng, J.: RAFT-3D: Scene flow using rigid-motion embeddings. In: CVPR (2021)
35. Verbin, D., Hedman, P., Mildenhall, B., Zickler, T., Barron, J.T., Srinivasan, P.P.: Ref-NeRF: Structured view-dependent appearance for neural radiance fields. In: CVPR (2022)
36. Wu, T., Zhong, F., Tagliasacchi, A., Cole, F., Oztireli, C.: D^2NeRF: Self-supervised decoupling of dynamic and static objects from a monocular video. In: NeurIPS (2022)
37. Xie, Y., Takikawa, T., Saito, S., Litany, O., Yan, S., Khan, N., Tombari, F., Tompkin, J., Sitzmann, V., Sridhar, S.: Neural fields in visual computing and beyond. Computer Graphics Forum (2022). https://doi.org/10.1111/cgf.14505

A Additional Details

Fixed Pattern Noise. One source of noise associated with sensors (including those used in C-ToF cameras) is fixed-pattern noise [5]. This noise originates from spatial non-uniformities with the physical sensor pixels themselves, resulting in differences in their response to light. The raw images captured by our C-ToF camera also have fixed noise pattern, which we model as a per-pixel offset in the intensity. When computing C-ToF derived depth, these fixed patterns in the raw frames are typically cancelled out within the computation of the phase offset (Equation 1). Given that our method uses the raw frames directly, it is important to run a pre-processing step that subtracts the fixed-pattern noise from the raw frames. We calculated the fixed-pattern noise by (i) capturing four raw images of a static scene, and (ii) averaging the result. Figure 7 demonstrates the raw images before and after fixed pattern noise calibration.

Data Normalization. To eliminate outliers in the real data, usually resulting from oversaturated pixels in the sensor, we use a two-step normalization process. Through empirical observation, this improves the quality of the 4D volume reconstruction. Initially, we normalize the raw ToF captures by normalizing the amplitudes of all the quartets to be between $[0, 1]$ using the maximum amplitude

Table 2: Mathematical symbol legend.

Symbol	Description
$\mathbf{x}$	A point $\in \mathbb{R}^3$.
$\boldsymbol{\omega}$	A direction; unit vector $\in \mathbb{S}^2$.
$\mathbf{x}_t$	A point t units along a direction $\boldsymbol{\omega}$, $\mathbf{x}_t = \mathbf{x} + \boldsymbol{\omega} t$.
$\boldsymbol{\omega}_\mathrm{i}$	A direction incoming to a point.
$\boldsymbol{\omega}_\mathrm{o}$	A direction outgoing from a point.
ψ	C-ToF phase, linearly related to distance.
ϕ	Field offset $\in \{0, \frac{\pi}{2}, \pi, \frac{3\pi}{2}\}$.
τ	Timestep associated with each frame.
L_a	Amplitude of returned/reflected light.
L_ϕ	Raw quad using field offset ϕ.
$L_{\phi=0}^{\tau=1}$	Raw quad using field offset $\phi = 0$ at timestep $\tau = 1$.
$\mathbb{L}_\phi$	Quartet of raw quad images.
$\mathbf{v}(\mathbf{x}, \tau)$	Velocity vector $\in \mathbb{R}^3$ at point $\mathbf{x}$ and time τ.
$\mathbf{u}(\mathbf{x}, \boldsymbol{\omega}, \tau)$	Projected integrated scene flow $\in \mathbb{R}^2$ for a ray from point $\mathbf{x}$ in direction $\boldsymbol{\omega}$ at time τ.
$D(\tau = t)$	Integrated depth map at time t
$L(\mathbf{x}, \boldsymbol{\omega})$ or $L(\mathbf{x}, \boldsymbol{\omega}, \tau)$	Radiance measured by a camera at point $\mathbf{x}$ in direction $\boldsymbol{\omega}$.
$\sigma(\mathbf{x})$ or $\sigma(\mathbf{x}, \tau)$	Density function at a point.
$T(\mathbf{x}, \mathbf{x}_t)$ or $T(\mathbf{x}, \mathbf{x}_t, \tau)$	Transmittance function, i.e., accumulated density.
$W(p)$	Importance function for light path of length p.

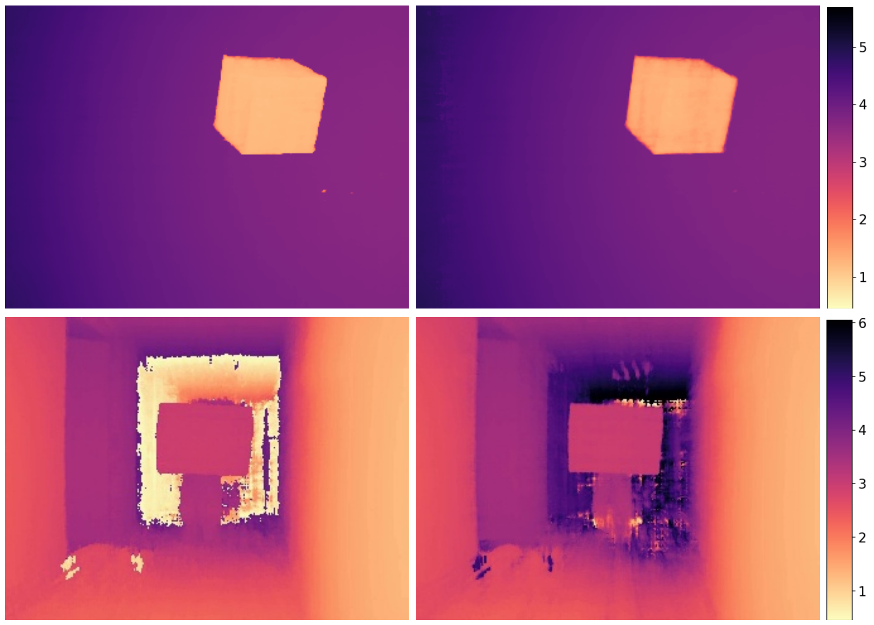


Fig. 6: Different Depth Output Examples from *Arcing Cube* and *Target* Left: Depth d_{ToF} derived via rendered raw frames. In real sequence *Target*, the scene's range exceeds C-ToF range and we see phase wrapping. Rendering raw frames from our 4D scene reconstruction cannot automatically unwrap phase because depth in d_{ToF} is still derived via Eq. 1. **Right:** Depth d is produced by volume rendering density σ as depth via Eq. 7. d can be fuzzy because σ is only indirectly optimized. Please see our supplemental video for additional results.

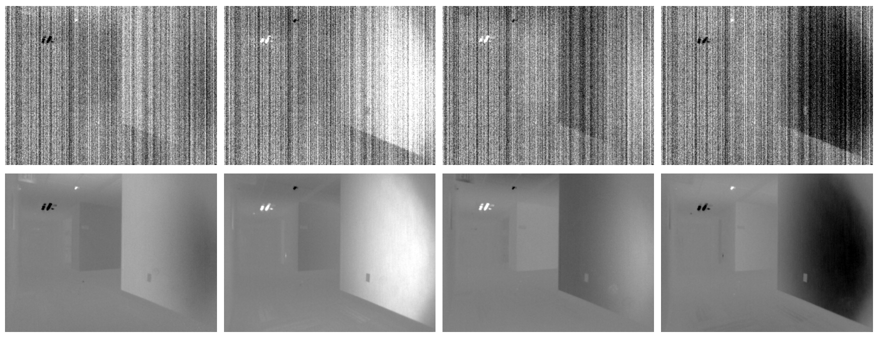


Fig. 7: Removing fixed-pattern noise. Top: Raw images of a hallway captured with a C-ToF camera. These images are averaged to compute the fixed-pattern noise associated with this sensor. **Bottom:** Same raw images of the scene, after subtracting the fixed-pattern noise.

Table 3: Ablations. Synthetic scenes; showing MSE×100. Variations:
- No OF Loss: No optical flow loss $\mathcal{L}_{\mathbf{u}}$.
- Single Stage: Optimizing the velocity field from the beginning, rather than after a set of iterations with zero velocity.
- No Color: Without the input color channel as an additional loss.
- No Repro.: Without using the phase-aware reprojection loss.
- Ours: Full model.

Scene		No OF Loss	Single Stage	No Color	No Repro.	Ours
Arcing Cube	Dyn.	12.186	**10.679**	10.906	62.834	13.310
	All	1.330	**1.132**	1.155	92.070	1.256
Axial Speed Test	Dyn.	5.008	**3.386**	3.773	4.589	3.430
	All	1.067	**0.812**	0.917	4.101	0.938
Orthogonal Speed Test	Dyn.	28.958	**24.008**	107.507	82.382	33.021
	All	8.673	9.082	24.735	39.838	**7.527**
Sliding Cube	Dyn.	1.427	1.319	1.134	**0.965**	1.080
	All	**0.431**	0.503	0.441	2.370	0.440
3 Cubes Speed Test	Dyn.	10.485	6.336	10.556	7.256	**6.268**
	All	2.057	1.463	2.402	5.484	**1.390**
3 Chairs Speed Test	Dyn.	**5.337**	6.105	6.743	11.099	5.540
	All	0.868	0.990	1.066	2.946	**0.855**
Occluded Cube	Dyn.	1.303	1.166	1.818	4.281	**0.809**
	All	0.853	0.691	0.871	1.894	**0.647**

observed across all frames within a scene. For real data, then, we truncate values exceeding 0.1 and perform the same amplitude normalization again.

DC Offset. As our method directly uses raw frames, we must correctly account for average amount of light emitted—the so-called DC offset—around which the sinusoidal emission varies. We estimate the intensity of the emitter's light source as a constant additive value present in the captured signal: light intensity has the non-negative signal $\frac{1}{2}\sin(2\pi f t) + C$, where C represents the DC offset (cf. idealized model in main paper). In synthetic scenes, we render the raw frame quartets with a fixed DC offset $C = 0.5$. In real scenes, the value of the DC offset is unknown, so we optimize a predicted DC offset during training after initializing it to zero.

Reprojection Loss Cost. We supervise the density field at integer time moments only (aligned with the capture of L_0); that is, we reproject the density field at integer time moments to recreate the appearance of $L_{\frac{\pi}{2}}$, L_{π}, and $L_{\frac{3\pi}{2}}$. In principle, we can penalize reprojection losses to optimize the scene density at any and all fractional time moments; however, in practice, this is too computationally and memory expensive. Similarly, we could supervise the motion using 2D flow

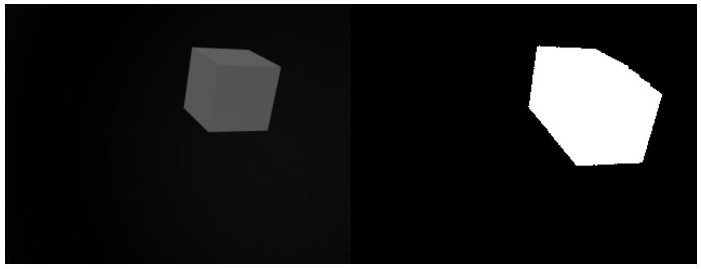


Fig. 8: Dynamic Mask Example from *Arcing Cube*. Left: Frame at integer time moment. **Right:** Dynamic mask spanning an integer unit of time. The dynamic mask captures the object's geometry during motion across the quartet. These masks are used to evaluate depth MSE metrics on dynamic regions.

computed between all pairs of matching quads; in practice, we use only L_0 to reduce expense.

Temporal Superresolution. The explicit motion modeling allows us to interpolate density to an arbitrary time moment and increase the temporal resolution of the original video. For example, to generate a depth map at the novel time moment j, we blend depth maps reprojected from the two nearby integer time moments i and $i+1$:

$$D(\tau = j) = (1-\delta) \cdot D^{i \to j}(\tau = j) + \delta \cdot D^{(i+1) \to j}(\tau = j), \tag{16}$$

where $\delta = j - i$ and $j \in [i, i+1]$. Please see the supplemental videos for example temporally-superresolved videos.

Dynamic Masks. To compute the dynamic masks used to evaluate the Depth MSE on synthetic scenes, we generate ground-truth motion vectors for each quartet, $\{L_0, L_\pi, L_{\frac{\pi}{2}}, L_{\frac{3\pi}{2}}\}$. Then, we mask pixels with no motion, and then union the masks to produce an integer-timestep aligned mask. This allows the dynamic masks to capture both the dynamic object's true location and the regions where motion artifacts are expected (assuming synchronous capture). Figure 8 shows an example of this.

Velocity Regularizations. Following NSFF [19], we apply regularizations $\mathcal{L}_{\text{reg}}$ to the flow to encourage flow smoothness and symmetry. Cycle consistency minimizes the summation of forward and backward scene flow for corresponding points across time :

$$\begin{aligned} \mathcal{L}_{\text{cyc}} = \sum_{i \in \mathbb{N}} \sum_{\mathbf{x}} & \left\| \mathbf{v}_{\text{f}}(\mathbf{x}, \tau = i) + \mathbf{v}_{\text{b}}(\mathbf{x}^{i \to (i+1)}, \tau = (i+1)) \right\|_1 \\ & + \left\| \mathbf{v}_{\text{b}}(\mathbf{x}, \tau = i) + \mathbf{v}_{\text{f}}(\mathbf{x}^{i \to (i-1)}, \tau = (i-1)) \right\|_1 . \end{aligned} \tag{17}$$

Temporal smoothness minimizes the summation of forward and backward scene flow for each point in the volume:

$$\mathcal{L}_{\text{temp}} = \sum_{i \in \mathbb{N}} \sum_{\mathbf{x}} \left\| \mathbf{v}_{\text{f}}(\mathbf{x}, \tau = i) + \mathbf{v}_{\text{b}}(\mathbf{x}, \tau = i) \right\|_2^2 . \tag{18}$$

We use L1 regularization of the velocity field,

$$\mathcal{L}_{\min} = \sum_{i \in \mathbb{N}} \sum_{\mathbf{x}} \left\| \mathbf{v}(\mathbf{x}, \tau = i) \right\|_1 , \tag{19}$$

to encourage minimal motion, i.e. a static reconstruction wherever feasible. The final velocity regularization loss is

$$\mathcal{L}_{\text{reg}} = \lambda_{\text{cyc}} \mathcal{L}_{\text{cyc}} + \lambda_{\text{temp}} \mathcal{L}_{\text{temp}} + \lambda_{\min} \mathcal{L}_{\min}, \tag{20}$$

with hyperparameters set to $\lambda_{\text{cyc}} = 0.0001$, $\lambda_{\text{temp}} = 0.001$, $\lambda_{\min} = 0.001$.

Integrating RGB Cameras. Following Equation 5, the presented formulation can be extended to integrate color cameras, too [2]. For instance, the *StudyBook* data sequence from Attal et al. includes this extra information. As the color sensor is offset slightly from the C-ToF sensor, the density σ is subtly supervised by small-baseline multi-view constraints under the assumption that σ can be shared between the infrared ToF and color channels. Adding a moving color camera can overcome phase wrapping and lead to denoised and superresolved density fields—this is notable in the significantly higher quality of density reconstruction in the *StudyBook* sequence (supplemental video). We label the color reconstruction loss $\mathcal{L}_{\text{RGB}}$ and its equivalent reprojection loss $\mathcal{L}_{\text{RGB}}^{i \to j}$.

In the presence of an RGB signal, stage 1 of the optimization penalizes

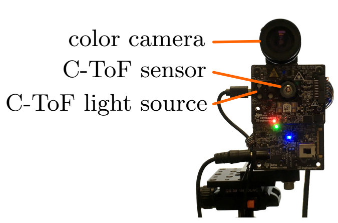
$$\mathcal{L} = \lambda_{\phi} \mathcal{L}_{\phi} + \lambda_{\text{RGB}} \mathcal{L}_{\text{RGB}}, \tag{21}$$

where $\mathcal{L}_{\phi}$ is a loss from Equation 14. For stage 2, we penalize

$$\mathcal{L} = \sum_{i,j} \lambda_{\phi} \mathcal{L}_{\phi}^{i \to j} + \lambda_{\mathbf{u}} \mathcal{L}_{\mathbf{u}} + \lambda_{\text{RGB}} \mathcal{L}_{\text{RGB}} + \sum_{i,k} \lambda_{\text{RGB}} \mathcal{L}_{\text{RGB}}^{i \to k}, \tag{22}$$

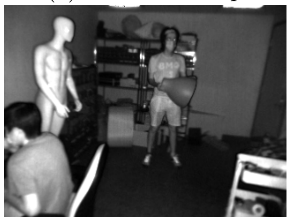
where i and k are integer timesteps such that $|i - k| = 1$ and j is a fractional timestep, such that $|i - j| < 1$. The hyperparameters are set to $\lambda_{\phi} = 10.0$, $\lambda_{\mathbf{u}} = 0.01$, and $\lambda_{\text{RGB}} = 1.0$.

Model and Optimization. We parameterise $F_{\boldsymbol{\theta}}(\mathbf{x}, \boldsymbol{\omega}, \tau)$ with an 8-layer MLP with 256 neurons each. Each input parameter is transformed with a 10-band positional encoding. The MLP has separate heads for density, amplitude, and velocity. Stage 1 of the optimization occurs for 25–100K iterations, with 200K iterations in stage 2. Loss hyperparameters are set to $\lambda_{\mathbf{u}} = 0.01$, where the velocity network output is initialized to be near zero and $\lambda_{\mathbf{u}}$ is decayed to zero during training. We use the Adam optimizer [15] with $\beta_1 = 0.9$, $\beta_2 = 0.999$ and $\epsilon = 10^{-7}$. We set the learning rate to $\eta = 2 \cdot 10^{-4}$, and use a batch size of 1024 simulated through gradient accumulation, since it empirically leads to better convergence. The training is performed on a single RTX 3090 GPU with 24 GB of RAM and typically takes three days.

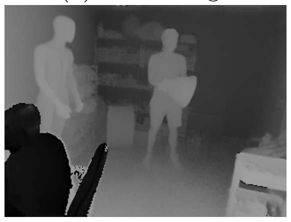


(a) Photo of setup

(b) Color image



(c) C-ToF amplitude



(d) C-ToF phase

Fig. 9: (a) Photo of the proposed hardware setup, consisting of a single ToF and a color camera. **(b)** Color image from color camera. **(c)** Amplitude image; represents the average amount of infrared light reflected by the scene. **(d)** Phase image; values are approximately proportional to range.

B Experimental C-ToF Setup

Our hardware setup is the same as in Attal et al. [2]. Here, we reproduce text from that paper as the details are the same.

The hardware setup shown in Figure 9(a) consists of a standard machine vision camera and a time-of-flight camera. Our USB 3.0 industrial color camera (UI-3070CP-C-HQ Rev. 2) from iDS has a sensor resolution of 2056×1542 pixels, operates at 30 frames per second, and uses a 6 mm lens with an $f/1.2$ aperture. Our high-performance time-of-flight camera (OPT8241-CDK-EVM) from Texas Instruments has a sensor resolution of 320×240 pixels, and also operates at 30 frames per second (software synchronized with the color camera). Camera exposure was 10 ms. The illumination source wavelength of the time-of-flight camera is infrared (850 nm) and invisible to the color camera. The modulation frequency of the time-of-flight camera is $\omega = 30\,\text{MHz}$, resulting in an unambiguous range of 5 m. Both cameras are mounted onto an optical plate, and have a baseline of approximately 41 mm.

We use OpenCV to calibrate the intrinsics, extrinsics and distortion coefficients of the stereo camera system. We undistort all captured images, and resize the color image to 640×480 to improve optimization performance. In addition, the phase associated with the C-ToF measurements may be offset by an unknown constant; we recover this common zero-phase offset by comparing the measured phase values to the recovered position of the calibration target. For simplicity, we assume that the modulation frequency associated with the C-ToF camera is an approximately sinusoidal signal, and ignore any nonlinearities between the recovered phase measurements and the true depth.

Along with the downsampled 640×480 color images, the C-ToF measurements consist of the four 320×240 images, each representing the scene response to a different predefined phase offset ϕ. For visualization in Figure 9, ToF amplitude and phase images are computed from the quartet of raw frames.