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.

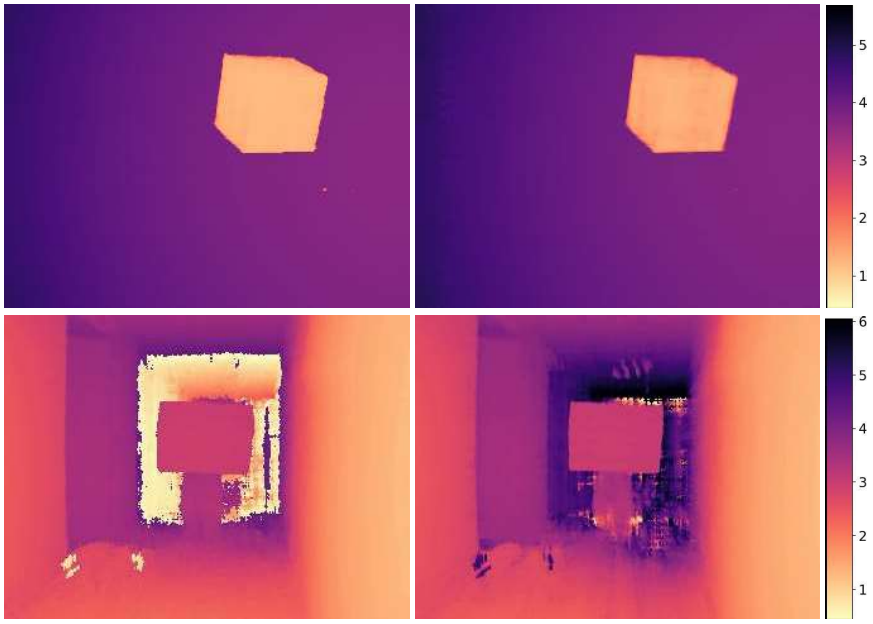


Fig. 6: Different Depth Output Examples from *Arcing Cube* and *Target* **Left:** Depth d_{ToF} derived via rendered raw frames. In real sequence *Target*, the scene's range exceeds C-ToF range and we see phase wrapping. Rendering raw frames from our 4D scene reconstruction cannot automatically unwrap 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.

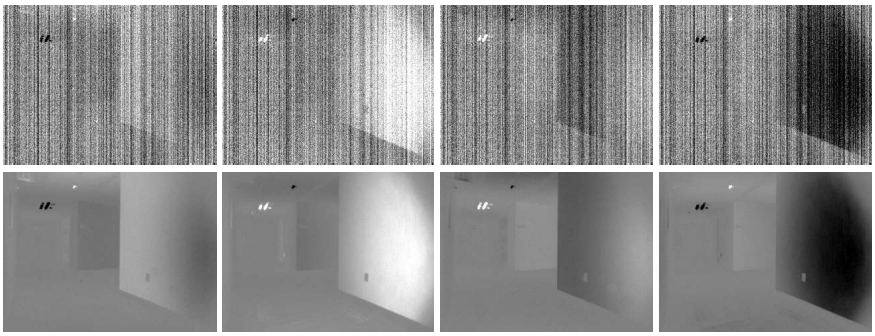


Fig. 7: Removing fixed-pattern noise. Top: Raw images of a hallway captured with a C-ToF camera. These images are averaged to compute the fixed-pattern noise associated with this sensor. **Bottom:** Same raw images of the scene, after subtracting the fixed-pattern noise.

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 $\frac{1}{2}\sin(2\pi ft) + C$, 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

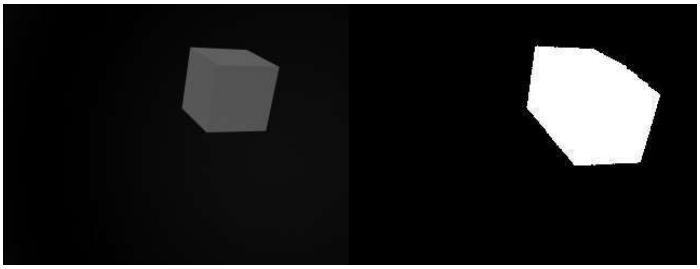


Fig. 8: Dynamic Mask Example from *Arcing Cube*. Left: Frame at integer time moment. **Right:** Dynamic mask spanning an integer unit of time. The dynamic mask captures the object's geometry during motion across the quartet. These masks are used to evaluate depth MSE metrics on dynamic regions.

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,

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

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

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

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

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

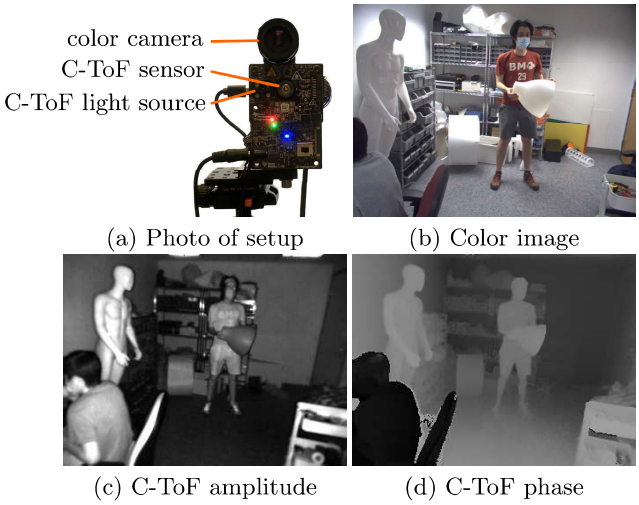
$$\mathcal{L} = \lambda_{\phi} \mathcal{L}_{\phi} + \lambda_{\text{RGB}} \mathcal{L}_{\text{RGB}}, \tag{21}$$

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 $\omega = 30$ 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.