Color-Encoded Illumination for High-Speed Volumetric Scene Reconstruction

David Novikov Eilon Vaknin Narek Tumanyan Mark Sheinin
Weizmann Institute of Science, Israel
{david.novikov, eilon.vaknin, narek.tumanyan, mark.sheinin}@weizmann.ac.il
https://davidnovikov.github.io/color-encoded-illumination-website/

Abstract

The task of capturing and rendering 3D dynamic scenes from 2D images has become increasingly popular in recent years. However, most conventional cameras are bandwidth-limited to 30–60 FPS, restricting these methods to static or slowly evolving scenes. While overcoming bandwidth limitations is difficult for general scenes, recent years have seen a flurry of computational imaging methods that yield high-speed videos using conventional cameras for specific applications (e.g., motion capture and particle image velocimetry). However, most of these methods require modifications to a camera's optics or the addition of mechanically moving components, limiting them to a single-view high-speed capture. Consequently, these methods cannot be readily used to capture a 3D representation of rapid scene motion. In this paper, we propose a novel method to capture and reconstruct a volumetric representation of a high-speed scene using only unaugmented low-speed cameras. Instead of modifying the hardware or optics of each individual camera, we encode high-speed scene dynamics by illuminating the scene with a rapid, sequential color coded sequence. This results in simultaneous multi-view capture of the scene, where high-speed temporal information is encoded in the spatial intensity and color variations of the captured images. To construct a high-speed volumetric representation of the dynamic scene, we develop a novel dynamic Gaussian Splatting-based approach that decodes the temporal information from the images. We evaluate our approach on simulated scenes and real-world experiments using a multi-camera imaging setup, showing first-of-a-kind high-speed volumetric scene reconstructions.

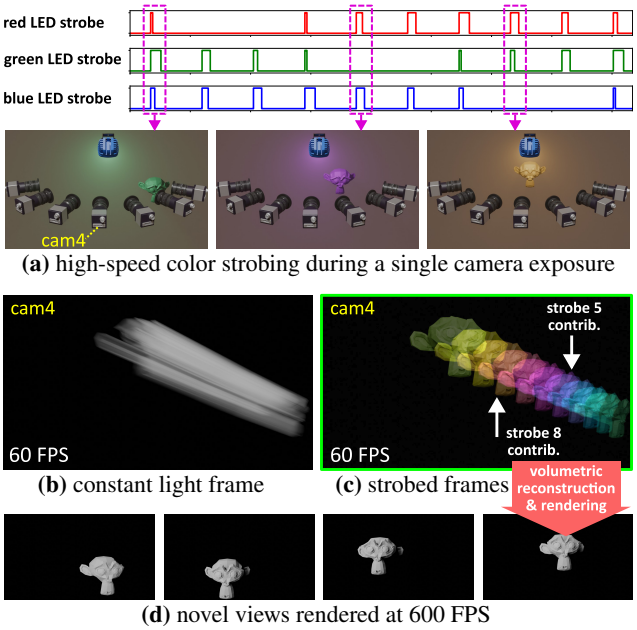


Figure 1. High-Speed volumetric scene encoding and reconstruction. **(a)** We strobe a scene having rapid motion with a sequence of 10 distinct colors during each captured frame. **(b)** With constant illumination, the object motion would result in a motion-blurred frame, where the exact object motion is obscured. **(c)** With our system, however, the resulting frames per camera contain a colorful mixture of the object's intermediate frames. **(d)** The individual strobed frames from all cameras are then used to recover a volumetric dynamic representation of the scene, enabling novel view synthesis at a high frame rate of 600 Hz.

1. Introduction

The capture speed of any camera is limited by its data bandwidth, namely the rate at which its readout electronics can transfer image data from the sensor to its memory. For general scenes, the inherent bandwidth limitation is hard to overcome without resorting to expensive, specialized high-speed cameras. Moreover, most high-speed cameras suffer from a poor 'temporal dynamic range', namely that the number of *total* frames that can be captured at high frame rates is low (*e.g.*, 256), severely limiting the capture duration and requiring precise triggering to capture the desired event [16]. However, some special yet important imaging scenarios like motion capture, particle imaging velocimetry (PIV), ballistics tracking, and other high-speed analytical techniques, have inherent characteristics that lend

themselves to computationally encoding high-speed videos within the limited bandwidth. Our method targets these scenarios in which the objects' high-speed motion, rather than their appearance, is of primary interest.

There are two main strategies for 'squeezing' higher frame rates from conventional, low-speed cameras. The first involves encoding full video frames into a subset of camera pixels, thereby increasing the frame rate by the full-frame-to-subset ratio [1, 28, 38, 39, 46]. The second strategy is multiplexing several high-speed frames into a single low-speed frame, then computationally decomposing the captured (multiplexed) frame into a high-speed video [3, 9]. Nevertheless, most prior works, irrespective of the frame encoding process, focus on recovering a single-view high-speed video of the scene. Moreover, since most prior approaches require specialized hardware to achieve the desired multiplexing (*e.g.*, spatial light modulators, novel sensors with per-pixel control, optical components such as diffraction gratings, diffusers, and more), scaling these approaches to multi-view scenes would require replicating and calibrating the complex imaging system many times over, as well as precise temporal synchronization.

In this work, we focus on recovering volumetric representations of high-speed scenes using low-speed cameras. We follow the second strategy by encoding multiple video frames into a single frame. Similarly to the work of Chan et al. [3], we use *color* to encode multiple interframes[1] during the camera's exposure period for each low-speed frame. However, unlike their method, which temporally augments the camera's point-spread function (PSF) using a spinning diffraction grating, we augment *the scene's* object color by strobing it with a pre-defined color sequence (see Fig. 1). The resulting low-speed frames contain a linear mixture of different timestamps with distinct colors. Like some prior works [13, 48], our method encodes the 'timestamp' information *directly within the scene* (via temporal color projection), making it invariant to the camera optics or type.

Our approach lends itself to volumetric high-speed capture since it can be applied to *multiple* unaugmented cameras simultaneously using a single scene-strobing illuminant. To recover the high-speed scene, we develop a dynamic Gaussian Splatting-based method that takes the color-strobed low-speed camera frames as input and fits a dynamic Gaussian splatting model to the high-speed 3D scene. Our optimization decodes the high-speed volumetric scene while leveraging the multi-view geometric data to constrain the result. To the best of our knowledge, our method is the first to tie together volumetric rendering with compressive high-speed imaging.

We evaluated our approach on simulated data and in real-world experiments using a hardware prototype, demonstrating its ability to capture rapidly moving objects. We present first-of-a-kind volumetric renderings of high-speed motion captured at 60 frames per second (FPS) and decoded to 600 FPS, revealing the dynamics of fast objects that cameras cannot capture at their maximum speed. We also use simulated data to conduct a performance analysis, demonstrating how different key factors in our approach affect its performance. Our method takes the first step toward unifying volumetric rendering with high-speed compressive video and will pave the way for future work in this domain.

2. Related works

2.1. High-speed computational imaging

Many prior works have focused on achieving high-speed capture without using expensive, bulky, high-bandwidth cameras, instead coding the camera exposure at the aperture or pixel level [10, 27, 33, 34], with pre-defined [34] or random codes [31]. Many works also exploited the rapid inter-row rolling shutter rate to encode scene frames into individual rows [1, 8, 38, 46]. Later, works began to use emerging imaging technologies to capture high-speed video or enhance traditional camera footage. Event cameras, for example, were extensively investigated to improve low-speed conventional videos captured simultaneously [41, 42].

A common feature of nearly all prior works is that they use specialized hardware to either augment a conventional camera (*e.g.*, coded aperture, coded pixels, an added diffuser, or diffraction gratings), or to capture the scene itself. This limits such systems to single-camera capture, whereas simultaneous multi-camera capture would require replicating specialized, and sometimes costly, prototypes. Conversely, our method applies *no* camera augmentation. It can thus trivially accommodate an arbitrary number of cameras simultaneously capturing the scene. [2]

2.2. Visual coding using color

A subset of prior works used *color*, in particular, to encode spatio-temporal information. Xiong *et al.*, used the light exiting a diffraction grating to color particles at different depth planes for particle image velocimetry (PIV) [49]. Sheinin *et al.*, used the diffracted light entering a camera to encode the sparse positions of scene objects [38] and later more general scenes [3]. Tippur *et al.*, used color to encode the position of deformation of a tactile sensor to allow finer texture analysis of objects being grabbed [40]. Jaques *et al.*, used red, green, and blue illumination to upscale the temporal video resolution by a factor of three using each color channel as an interframe [13]. Recently, Verma *et al.*, combined flashing multi-spectral LEDs with a rolling shutter sensor to capture hyperspectral images [44]. Chan *et al.*, used a rotating diffraction grating to produce a time-varying point spread

[1] We borrow the term 'interframes' from the video compression field since it is elegantly more concise than 'intermediate frames'.

[2] Our method requires color cameras with global shutter sensors.

function [3]. Like them, we encode many interframes into a single exposure, but instead of modulating light in the sensor domain, we modify the appearance of scene objects, enabling multi-camera capture and, therefore, multi-view volumetric reconstruction. Lastly, Veeraraghavan *et al*. used color strobing for high-speed periodic scene sensing [43], whereas we target general, aperiodic scenes.

2.3. Dynamic 3D reconstruction

Recently, many methods have tackled dynamic 3D reconstruction from single- or multi-view video. Implicit representation-based methods [2, 18, 20, 29, 30, 32] use volumetric rendering, typically with NeRF, along with an optimizable neural deformation field to reconstruct the dynamic scene. After the invention of 3D Gaussian Splatting [14], the trend has shifted towards using explicit representations of dynamic scenes [26]. Some of the 3DGS-based methods optimize for canonical Gaussians along with a learned deformation field that determines the motion of each Gaussian [21, 47, 50]. Other works regularize the scene flow by imposing a low-rank assumption, representing each Gaussian trajectory by factorizing its motion into a learned motion basis, either local [11, 17] or global [15, 22, 45]. To incorporate temporal smoothness, some works represent the Gaussians' motion explicitly by spline or polynomial functions [19, 24]. Our work is the first to reconstruct high-speed 3D dynamic scenes from encoded low-speed images by building upon the Gaussian Flow framework [24].

3. Multi-color stroboscopic imaging

In this Section, we describe the image formation model of a single camera in our multi-camera system. For simplicity, here we assume all cameras share the same spectral sensitivity function, and discuss color calibration in Sec. 6. Consider a scene captured by a color camera and illuminated by a set of approximately co-located direct color LEDs.[3] Without loss of generality, suppose that the LED set contains the standard red, green, and blue (RGB) projective color primaries. Furthermore, assume that the scene objects we seek to capture in motion have an approximately uniform spectral reflectance and that the scene background is sufficiently dark such that it reflects a negligible amount of light with respect to the moving object.

Under these assumptions, the camera image $I^{\mathrm{rgb}}(\mathbf{x}, c)$, in graylevel units, captured for an exposure duration T^{exp} can be modeled as:

$$I^{\mathrm{rgb}}(\mathbf{x}, c) = \int_{t=0}^{T^{\mathrm{exp}}} \boldsymbol{c}_{\mathrm{RGB}}(t, c) I^{\mathrm{int}}(\mathbf{x}, t) dt, \qquad (1)$$

[3]The LEDs must be co-located or sufficiently diffused to avoid shadowing differences between colors.

where $\mathbf{x}$ denotes the image pixel location, c denotes the camera's color channel (*e.g*., RGB) and t denotes time. The integrand in Eq. (1) is a multiplication of two factors. The first factor, $\boldsymbol{c}_{\mathrm{RGB}}(t, c), c \in \{\mathrm{R}, \mathrm{G}, \mathrm{B}\}$, is the normalized instantaneous color captured by the camera at time t, namely:

$$\|\boldsymbol{c}_{\mathrm{RGB}}(t)\|_2 = 1, \qquad \text{where } \boldsymbol{c}_{\mathrm{RGB}}(t) \in \mathbb{R}^3. \qquad (2)$$

The second factor, $I^{\mathrm{int}}(\mathbf{x}, t)$, is an instantaneous image intensity having units of graylevel $\cdot\, s^{-1}$, where s is some scaling factor. Note that, under our formulation, the factor $\boldsymbol{c}_{\mathrm{RGB}}(t, c)$ encapsulates all color-related effects, including the LEDs' illuminant spectra, the object's spectral reflectance, and the camera's spectral response. The second factor, $I^{\mathrm{int}}(\mathbf{x}, t)$, encapsulates intensity-related effects, such as shading, surface reflectance variations (*i.e*., texture), scene radiance, and sensor quantum efficiency.

Let $\tilde{\boldsymbol{c}}_{\mathrm{RGB}}(t, c)$ denote the unnormalized instantaneous captured object color, namely $\tilde{\boldsymbol{c}}_{\mathrm{RGB}}(t, c) = s\boldsymbol{c}_{\mathrm{RGB}}(t, c)$. Then, $\tilde{\boldsymbol{c}}_{\mathrm{RGB}}(t, c)$ can be modeled as

$$\tilde{\boldsymbol{c}}_{\mathrm{RGB}}(t, c) = \alpha(t)\boldsymbol{c}_{\mathrm{R}}(c) + \beta(t)\boldsymbol{c}_{\mathrm{G}}(c) + \gamma(t)\boldsymbol{c}_{\mathrm{B}}(c) \qquad (3)$$

where $\boldsymbol{c}_{\mathrm{R}}$,$\boldsymbol{c}_{\mathrm{G}}$ and $\boldsymbol{c}_{\mathrm{B}}$ are the individual color primaries corresponding to the red, green and blue LEDs, respectively, and $\alpha(t), \beta(t)$ and $\gamma(t)$ are time dependent LED intensities having a range of $[0, 1]$. These primaries can be calibrated by sampling the RGB camera values of the same small object patch, for three different frames where the object is illuminated by a single LED (*e.g*., $\alpha = 1$, while $\beta = \gamma = 0$) keeping the exposure duration constant.

Strobing the scene with a sequence of colors. Let t_n denote the strobe start time of the n-th strobe, and T^{strobe} denote the strobe duration. During each strobe, the values of $\{\alpha(t), \beta(t), \gamma(t)\}$ are set to some constants denoted by $\{\alpha_n, \beta_n, \gamma_n\}$ to yield a sequence of N distinct colors, while outside the strobing intervals, all values are set to zero (*i.e*., the LEDs are off). Then, Eq. (1) becomes

$$I^{\mathrm{rgb}}(\mathbf{x}, c) = \sum_{n=1}^{N} \int_{t=t_n}^{t_n + T^{\mathrm{strobe}}} \boldsymbol{c}_{\mathrm{RGB}}(t, c) I^{\mathrm{int}}(\mathbf{x}, t) dt = \sum_{n=1}^{N} \boldsymbol{c}^n_{\mathrm{RGB}}(c) \int_{t=t_n}^{t_n + T^{\mathrm{strobe}}} I^{\mathrm{int}}(\mathbf{x}, t) dt \approx \sum_{n=1}^{N} \boldsymbol{c}^n_{\mathrm{RGB}}(c) I^{\mathrm{int}}(\mathbf{x}, t_n), \qquad (4)$$

where $\boldsymbol{c}^n_{\mathrm{RGB}}(c)$ and $I^{\mathrm{int}}(\mathbf{x}, t_n)$ are the object color the high-speed intensity frame at strobe n. In Eq. (4) we assume that the strobe duration T^{strobe} is short relative to object motion, having the object approximately static during each strobe. Later, in Sec. 7, we describe how this assumption allows for setting different intensities for $\alpha(t)$, $\beta(t)$, and $\gamma(t)$ using only 'binary intensity' LEDs. In the last step in Eq. (4), we absorbed the constant T^{strobe} in $I^{\mathrm{int}}(\mathbf{x}, t_n)$.

4. Volumetric scene reconstruction

In this section, we describe our novel method for modeling high-speed 3D dynamic scenes by extending an existing dynamic 3DGS method with a modified training process. For the reader's benefit, we start with a brief description of the dynamic 3DGS method we rely on, followed by a description of the extensions we made to its training process.

Scene representation using Gaussian splatting In Gaussian splatting, the 3D scene is represented by a set of Gaussians G. Given an arbitrary camera position $\phi \in \mathbb{R}^6$, the set of Gaussians can be used to render an image from view ϕ using the rendering function $\mathcal{R}$:

$$\mathcal{R}(G, \phi) = I^{\text{rgb}}(c). \tag{5}$$

For convenience, we drop the pixel $\mathbf{x}$ notation from $I^{\text{rgb}}(\mathbf{x}, c)$ from this point on. Each individual Gaussian $g \subset G$ is defined by a set of parameters (*e.g.*, positions, scales, rotations, colors). Given M input views ϕ_m of the scene $\{I^{\text{rgb}}_m(c, \phi_m)\}_{m=1}^{M}$ indexed by $m=1, 2, ..M$, the Gaussians in G are optimized via differentiable gradient-based optimization, such that their 'splatting' on the input camera positions ϕ_m matches the input images from these views. The 3DGS representation, however, was designed for static scenes. Next, we describe *Gaussian-Flow*, an extension which was designed to handle scene motion [24].

Gaussian-Flow In Gaussian-Flow, the Gaussian parameters evolve with time t according to $g(t)=g(0)+d(t)$, where $g(0)$ is the initial state and $d(t)$ is some chosen deformation function. Lin *et al*., modeled the deformation functions as a sum of a polynomial and a Fourier series, referring to their model as the Dual-Domain Deformation Model (DDDM) [24]. Since the Gaussians are time-dependent, the optimization now also requires the timestamps of the input frames, $\{t_k\}_{k=1}^{K}$. Let $G(0)$ and $D(t)$ denote the initial state of all the Gaussians and their time-dependent deformations, respectively. Then, the Gaussian-Flow objective is:

$$\underset{G(0),D(t)}{\text{argmin}} \sum_{k=1}^{K} \sum_{m=1}^{M} \left| \mathcal{R}(G(0)+D(t_k), \phi_m) - I^{\text{rgb}}(c, \phi_m, t_k) \right|. \tag{6}$$

Gaussian-Flow with strobing We employ the Gaussian-Flow framework to reconstruct the high-speed scene representation at timestamps $\{t_k+t_n\}|_{n=1}^{N}$ for each low-speed timestamp t_k. Our optimization does not account for multiple low-speed timestamps and uses only the M frames captured at timestamp t_k. Thus, without loss of generality, we set $t_k=0$ in the formulation below and drop the frame time from $I^{\text{rgb}}(c, \phi_m, t_k) \rightarrow I^{\text{rgb}}(c, \phi_m)$. The same procedure, described next, is applied to each frame t_k independently.

To model the 3D dynamic scene captured by our system, we make two main modifications to Gaussian-Flow. First, since the objects we seek to capture are assumed to have uniform spectral reflectance, we change the rendering function $\mathcal{R}(G, \phi)$ to output single-channel intensity images by assigning only a single color channel to each Gaussian.

Secondly, we modify the objective in Eq. (6). As formalized by Eq. (4), each color channel c of an image $I^{\text{rgb}}(c, \phi_m)$ captured from a camera view ϕ_m by our system can be represented as a linear combination of the interframes $\{I^{\text{int}}(\phi_m, t_n)\}_{n=1}^{N}$. If $(G(0), D(t))$ model the 3D dynamic scene accurately then for the rendered image from view ϕ_m at timestep t_n, denoted $R_n(\phi_m) \equiv \mathcal{R}(G(0) + D(t_n), \phi_m)$, it follows that $R_n(\phi_m) \approx I^{\text{int}}(\phi_m, t_n)$. Therefore, by replacing in Eq. (4) the interframes with the rendered images $\{R_n(\phi_m)\}_{n=1}^{N}$ we construct an estimator of the captured image

$$\hat{I}^{\text{rgb}}(c, \phi_m) \approx \sum_{n=1}^{N} \mathbf{c}^{n}_{\text{RGB}}(c) \cdot R_n(\phi_m) \tag{7}$$

Thus, the training objective minimizes the discrepancy between the estimated and input images

$$\underset{G(0),D(t)}{\text{argmin}} \underbrace{\sum_{m=1}^{M} \sum_{c \in \{\text{R, G, B}\}} \left| \hat{I}^{\text{rgb}}(c, \phi_m) - I^{\text{rgb}}(c, \phi_m) \right|}_{\mathcal{L}_1}. \tag{8}$$

Remark 1 *The proposed scheme can be used with various dynamic 3DGS methods. We used Gaussian-Flow because of its simple deformation functions, which require training only a few parameters. Since the movements in our scenes are rather simple, Gaussian-Flow provides a sufficiently expressive model while keeping the training process efficient.*

Remark 2 *The framework above can be applied to the single-camera case as well, yielding high-speed videos from a single encoded frame (see project webpage for a result).*

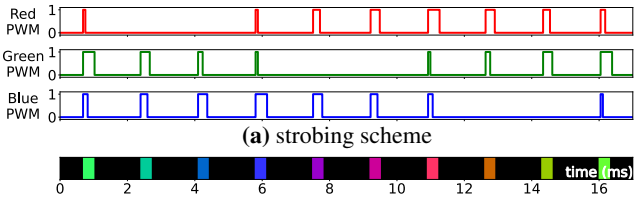
Regularization by depth Similar to prior 3DGS methods [5], we additionally incorporate a depth regularization term to promote a geometrically meaningful 3D scene reconstruction. Specifically, we penalize the total variation of the inverse depth. Let $Z(G, \phi_m, t_n)$ denote the rendered depth from view ϕ_m at timestep t_n. The total variation of the inverse depth, $\mathcal{L}_{\text{TV-depth}}$ is

$$\frac{1}{2NM} \sum_{m=1}^{M} \sum_{n=1}^{N} |\nabla_x \frac{1}{Z(G, \phi_m, t_n)}| + |\nabla_y \frac{1}{Z(G, \phi_m, t_n)}|,$$

where ∇_x, ∇_y are the gradients computed with respect to the x and y axis respectively. Our final loss is therefore

$$\mathcal{L} = \mathcal{L}_1 + \lambda_{\text{depth}} \mathcal{L}_{\text{TV-depth}}, \tag{9}$$

where $\mathcal{L}_1$ in defined in Eq. (8).



(a) strobing scheme

(b) resulting color

Figure 2. Generating colors by pulse modulation. We illuminate the scene with three direct-color LEDs. **(a)** Each LED is driven by an Arduino using on/off digital pulses whose widths set the per-strobe color. **(b)** Because each strobe is short compared to scene motion, the object appears static during a strobe, and the integrated LED pulses produce the desired color.

5. Strobing with color

Given some desired number of interframes N, we must select the LED intensities to yield the resulting colors $\boldsymbol{c}^n_{\mathrm{RGB}}$. The values of $\boldsymbol{c}^n_{\mathrm{RGB}}$ will affect the reconstruction quality. We tested various methods for selecting the 'best' set of colors for $\boldsymbol{c}^n_{\mathrm{RGB}}$, given N. Experimentally, we converged on the color selection in which we sample $\{\alpha_n, \beta_n, \gamma_n\}$ uniformly from a circle in the α, β, γ space. For the individual LED intensities, this amounts to sampling three sine waves, each separated by a 120-degree phase shift and normalized to the $[0, 1]$ range. For a 'well-behaved' object reflectance, the resulting circle in α, β, γ space will approximately map to some ellipse in the camera's RGB space [35].[4]

6. Data processing and calibrations

Camera color calibration In Sec. 3, we assume all cameras share the same spectral sensitivity response. To enforce this assumption, we calibrate the color response of each camera by imaging a 24-tile Spyder color checker (SCK300) [6]. Then, we select an arbitrary camera and map all other cameras to the selected camera's color space using a per-camera transformation matrix.

Background subtraction and LED color dictionary calibration For each experiment, we capture the static scene before or after the high-speed motion to extract the background images for all frames. We then subtract the background images from the raw frames before applying our strobed Gaussian-Flow model. The color dictionary $\mathbf{c}^n_{\mathrm{RGB}}(c)$ is determined directly from the foreground images.

Camera view calibration The Gaussian-Flow method requires the camera views as input. We calibrate the cameras using COLMAP by capturing a calibration object without strobing [37]. To initialize the Gaussian-Flow point cloud for dynamic scenes, we run COLMAP on the strobed foreground images. In practice, this initialization suffices.

[4] 'Well-behaved' in our context refers to an off-white object color having non-negligible reflectance in most of the visible spectrum.

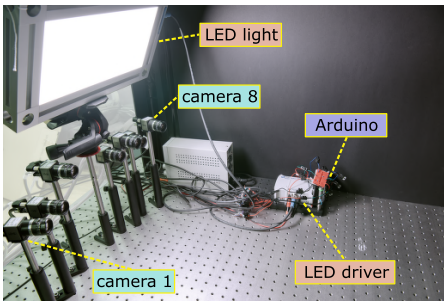


Figure 3. Experimental prototype. Our prototype uses three LED channels (R/G/B) driven by an LED driver and controlled via Arduino PWM. Each channel is strobed independently at high speed.

7. Imaging prototype

Our experimental prototype is shown in Fig. 3. We captured data using 8 global-shutter IDS UI-3240CP cameras, each with a resolution of 1280x1024 [12]. For each camera, we set the frame rate to 60 FPS (its maximum) and applied a hardware trigger from an Arduino Due [7] to synchronize the cameras (discussed further below). The scene is illuminated using a MOBL-300x150-RGBW light [23]. The light has three-color LED channels (Red, Green, and Blue) that can be individually strobed at high speed. We built a custom circuit to strobe the light using the Arduino's digital outputs by generating a PWM signal for each LED color from the output pins. A fourth Arduino PWM output signal is used to trigger the cameras. Our strobing can turn the LEDs on and off, but cannot alter their intensities. We circumvent this limitation and generate different colors by varying the strobe length of each channel, as described below.

As illustrated in Fig. 2, we generate different colors by using various combinations of digital pulse durations. Each combination, integrated over the short pulse duration, yields a distinct color in the captured frame. This type of strobing assumes that the object motion is relatively slow with respect to the strobe duration, which was about 0.083ms on average. Namely, during each strobe, the scene is approximately static, accumulating the duration of each LED to produce the desired color. Our results (*e.g.*, Fig. 4) show that this assumption holds in our experiments, as no noticeable motion blur artifacts are visible in any experiment. Each intensity interval lasted approximately 16.7μs, allowing for six discrete intensity values (from zero to five), yielding $6^3 = 216$ possible color combinations using the three LEDs. This yields 175 unique, usable colors, excluding scalar multiples of each other and the color no color.

8. Experimental evaluation

8.1. Real-world experiments

We demonstrate our approach by capturing rapidly moving objects, such as NERF gun darts, flying chess pieces, large

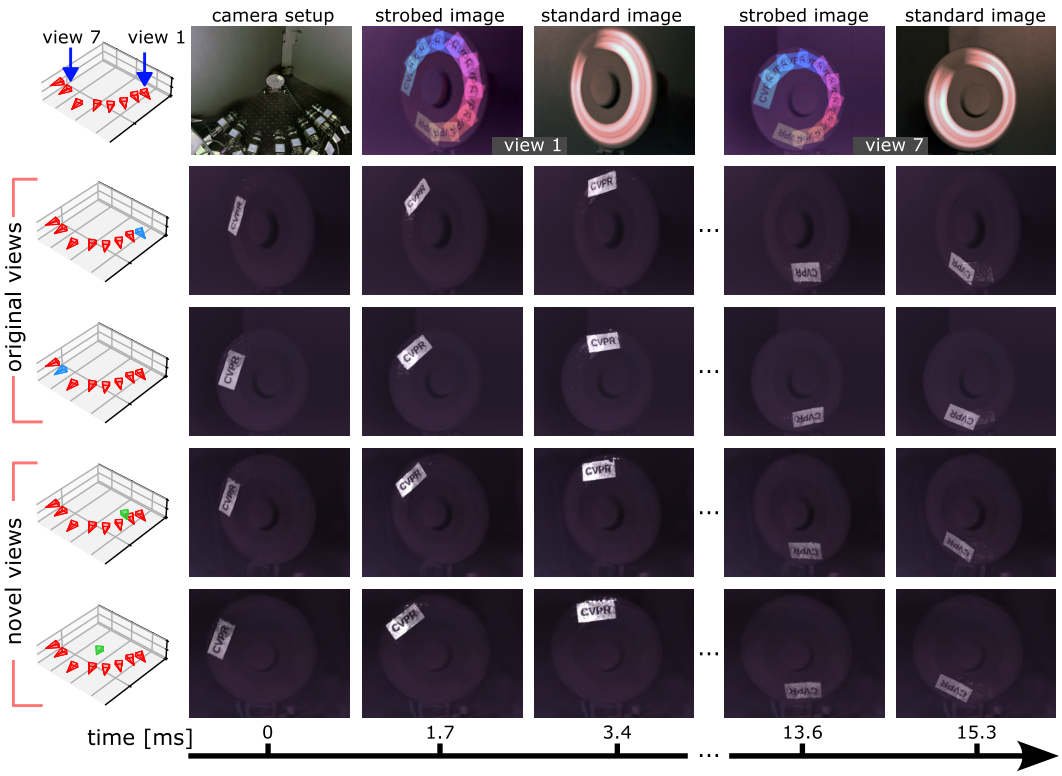


Figure 4. Experimental results on spinning disk. We capture a rapidly spinning disk using eight cameras. **(Top row)** shows the experiment setup and the raw input frames captured from two of the cameras' views. On the right of each strobed input image is a same-exposure image without strobing, showing the resulting images without color encoding. **(Middle rows)** Several high-speed frames from the original views. Note that our method correctly decodes the high-speed frames from the color-encoded frames, yielding high-speed multi-view videos. **(Bottom rows)** Several high-speed frames of the spinning disk from *novel views* rendering using our modified Gaussian-Flow.

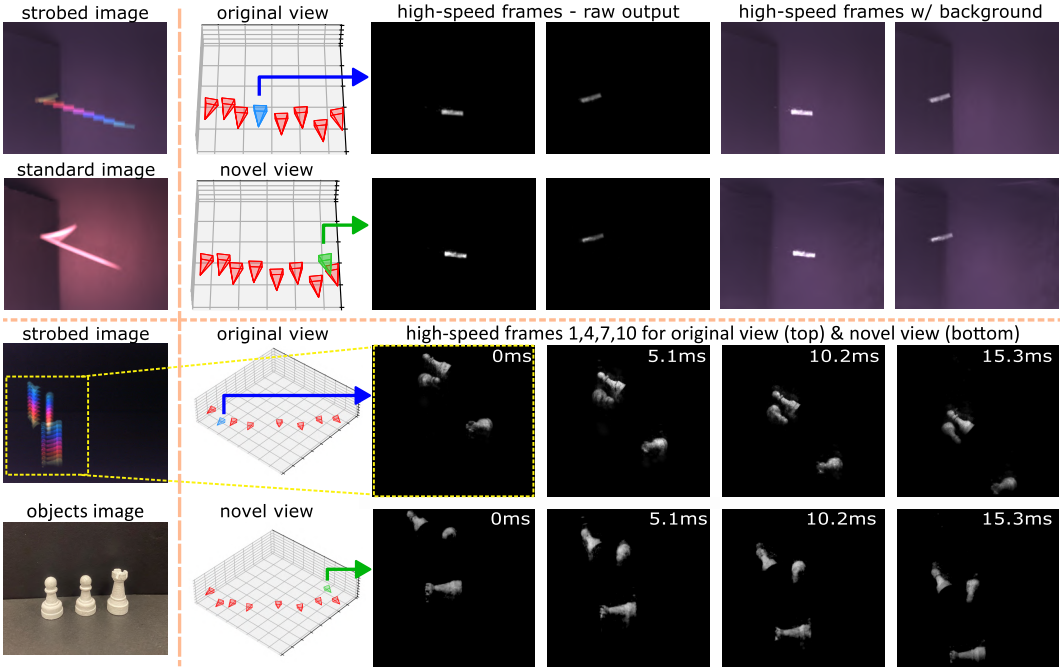


Figure 5. Motion capture experiments. **(Top)** Nerf dart experiment showing the strobed and non-strobed frames as well as recovered high-speed interframes. **(Bottom)** Flying chess pieces experiment; The rows show the recovered interframes for an existing and novel view.

particles in motion, and a rotating chopper. Our scenes were selected to be visually similar to the most relevant prior works [3, 38] to allow for a qualitative visual comparison. For each experiment, we capture two videos: one with our strobing system and one with 'standard' constant illumination, with all LEDs on. In our experiments, the light is set to strobe $N = 10$ colors, which upscales the camera's temporal resolution from 60 to 600 FPS. As shown in the top row of Fig. 4, in our experiments, the object motion is too fast for the camera to capture at its maximum speed, resulting in severe motion blur. Our method, however, can recover the high-speed object motion. The project webpage contains additional experiments and volumetric renderings of the captured scenes.

In all experiments, the background pixels are segmented as described in Sec. 6, and the Gaussian-Flow model is applied only on the foreground images. However, for better visualization, we add the background to some of the reconstructed frames. The background is computed by fitting a static Gaussian splatting model [14] on the background images using the known camera locations. Then, we render the same view from both models (a dynamic foreground and a static background) and combine the frames. Since our background is assumed to be much darker than the dynamic objects, this process yields a reasonable visualization of the full background and foreground scene.

Fig. 4 demonstrates the strength of our method to render novel views at high speeds. Here, eight cameras capture a white sticker on a fast-rotating disk. The figure shows the reconstruction of a *single low-speed frame per camera* into a volumetric high-speed representation of the motion. Rows two and three show the time evolution on two original input views, while rows four and five show novel views. Our method reconstructs the 3D motion with good visual quality (few visible artifacts). We repeated this experiment with a yellow sticker, yielding similar results and showing that the method can handle non-white albedo (see webpage).

In Fig. 5, we show experiments representative of motion capture [3, 38]. In Fig. 5(Top), we shoot a Nerf dart at a board (located on the left of the frame) and capture its motions as it impacts and bounces back. The result shows the dart bouncing off the left wall of the frame. Here we show rendered frames from the dynamic Gaussian-Flow model with and without the added background, for reference. Again, we can render novel and existing views during the high-speed dart motion. In Fig. 5(Bottom), we toss several chess pieces and capture their motion. Here, we demonstrate the capacity to reconstruct the motion of several larger objects that can overlap.

8.2. Simulation-based performance evaluation

We analyzed the performance and limitations of our method using simulated scenes. Our aim here is to gather insights

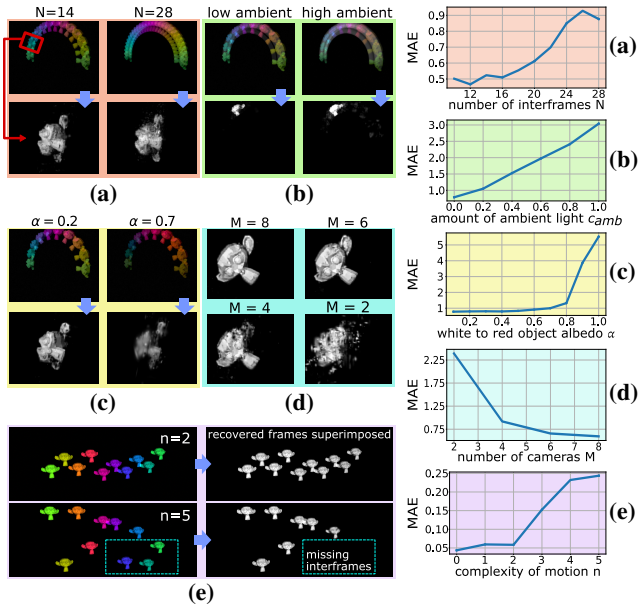


Figure 6. Simulation performance analysis. The left columns show example recovered frames; the right column shows MAE plots. **(a)** Effect of interframe number N. **(b)** Effect of ambient-light level (x-axis: ambient light as a percentage of total signal). **(c)** Effect of non-white albedo (x-axis: blend ratio between pure-white and pure-red reflectance). **(d)** Reconstruction quality versus the number of cameras. **(e)** Effect of the motion complexity.

and trends about the method's performance, not to characterize the performance of any one particular camera. All plots display the Mean Absolute Error (MAE) across all interframes, using a novel camera view as the ground truth.

Increasing the number of interframes: As shown in Fig. 6(a), increasing the number of interframes N decreases the cosine distance between the resulting $\alpha_n, \beta_n, \gamma_n$ vectors in $\alpha\beta\gamma$ space and the resulting camera RGB space, making the recovery more susceptible to noise. In addition, as the number of interframes in a single frame increases, their overlap also increases, making decoupling the interframes challenging. The plot shows performance degradation near N=28, leading to artifacts.

Effect of ambient light: Our method is sensitive to ambient light, reducing the $\boldsymbol{c}^n_{\mathrm{RGB}}$ color contrast and potentially causing saturation during long exposures. In Fig. 6(b) we simulate this effect by adding ambient signal to the input, $I^c + c_{\mathrm{amb}} I^{\mathrm{DC}}$, where I^{DC} is the scene under constant white light and c_{amb} controls its strength. We assume ambient light affects only the foreground, since background pixels can be removed via subtraction. Results show a roughly linear increase in reconstruction error with ambient-light level.

Effect of object albedo: Fig. 6(c) shows how object reflectance affects reconstruction quality. We simulate an albedo that transitions from pure white to pure red using a blending parameter α_{albedo}. The method performs well

for broad-spectrum (off-white) albedos but degrades as the object becomes less reflective in one or more spectral bands.

Number of cameras: In Fig. 6(d), we analyze the reconstruction quality as a function of the number of cameras M. The plot indicates that our method can synthesize unseen views reliably with as few as six cameras.

Sensitivity to motion complexity: We evaluated recovery performance as interframe motion becomes less smooth. To this end, we simulated a scene in which motion starts as a simple line and progressively becomes more erratic by randomly sampling the interframe y-coordinate as $y \sim \mathcal{U}(-n, n)$. Fig. 6(e) shows that for high-variance motion, the method may fail by "dropping" several interframes. We believe this is partially due to vanishing Gaussian gradients under large displacements [4].

9. Discussion and limitations

Specialized scenes and beyond. Our current method assumes uniform albedo objects with spatially varying shading. Future extensions could relax this constraint by explicitly modeling the object surface's spatially varying spectral response.[5] For example, as the object moves during the strobing, the same surface patch may appear at different image locations under multiple known strobes; aggregating these observations across interframes could enable joint recovery of motion and surface appearance. Moreover, combining white-light strobes with cameras operating at different exposure timings could produce strobed frames that enable recovery of scenes with spatially varying albedo. Adding data-driven priors that regularize object shape, texture distributions, and physically plausible dynamics could also help extend the method beyond specialized scenes.

Encoding with color: strengths and weaknesses. As in some prior works, we encode high-speed temporal information using color [3, 13, 38]. However, in contrast to earlier works, our method requires no mechanically moving parts [3], specialized optics [38], and involves multi-view optimization-based volumetric scene reconstruction capable of temporal up-scaling by factors of more than three, across multiple views [13]. While unmixing the strobed input frames is an ill-posed problem, the multi-view solver implicitly leverages epipolar geometry constraints, thereby yielding more robust solutions.

Nevertheless, encoding with color has disadvantages. Firstly, the uniform albedo assumption limits applicability to specialized scenes. The light-squared-distance fall-off may degrade SNR for distant objects (common to all active methods), especially when strobing with color. Nevertheless, integrating multiple frames into a single exposure reduces read noise per frame [36]; in our method, the light source is decoupled from the cameras and can be placed close to the scene.[6] Currently, our experiments require a black background, but our simulations show promise for handling non-dark backgrounds too (see project webpage). Lastly, unlike the standard Gaussian-Flow objective, which uses the object's color via the spherical harmonics loss term, our objective operates on *monochrome* interframes, making it less robust to recovering colorful objects. Nevertheless, this limitation diminishes when objects are predominantly single-colored, as is our premise.

Interframe temporal resolution. The temporal upsampling we achieve depends on our capacity to decompose the mixtures of frame colors. The more interframes we have, the less distinct the resulting colors become in the camera's RGB space, increasing the susceptibility to noise and reconstruction artifacts. This is exacerbated by non-white object albedos, which can reduce the separation in the RGB space. However, since strobing LEDs at high speeds (*i.e.*, hundreds of KHz) is much easier than increasing a camera's frame rate, our method could be used to upscale the speed of faster cameras too, using the same strobe number (*e.g.*, 60 FPS to 600 FPS, 600 FPS to 6000 FPS).

Cameras and light synchronization. All cameras and LEDs are driven by one Arduino. To continuously strobe N interframes during T^{exp}, the strobe spacings yield a margin of $T^{\text{marg}} \equiv T^{\text{exp}}/(2N)$ at both exposure boundaries (see Fig. 2). Thus, each camera's trigger can drift up to T^{marg} without affecting the captured frame. For us, $T^{\text{marg}} = 0.83$ms, which is $\sim 27\times$ larger than the typical 30μs hardware trigger jitter reported for our cameras [25]. This margin eliminates frame-strobe mis-synchronization risk.

10. Conclusion

We presented a novel framework for volumetric high-speed scene reconstruction using conventional low-speed cameras. By encoding temporal information directly into the scene via high-frequency color strobes, our method multiplexes several high-speed interframes into each captured image without requiring specialized sensors or complex optical components. We also developed a novel approach that uses a dynamic Gaussian-flow representation. Our approach enables the recovery of high-speed 3D motion from multi-view observations, effectively bridging compressive video techniques with modern volumetric rendering.

Acknowledgments: We thank Tali Dekel for helpful discussions. This work was supported by the Shimon and Golde Picker – Weizmann Annual Research Grant.

[5] Spectral response here accounts for both the object's spectral reflectance and the camera's spectral sensitivity functions.

[6] In fact, multiple light sources can be used at once as long as they are synchronized temporally.

References

[1] Nick Antipa, Patrick Oare, Emrah Bostan, Ren Ng, and Laura Waller. Video from stills: Lensless imaging with rolling shutter. In *2019 IEEE International Conference on Computational Photography (ICCP)*, pages 1–8. IEEE, 2019. 2

[2] Ang Cao and Justin Johnson. Hexplane: A fast representation for dynamic scenes. *CVPR*, 2023. 3

[3] Dorian Chan, Mark Sheinin, and Matthew O'Toole. Spincam: High-speed imaging via a rotating point-spread function. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 10789–10799, 2023. 2, 3, 7, 8

[4] David Charatan, Sizhe Li, Andrea Tagliasacchi, and Vincent Sitzmann. pixelsplat: 3d gaussian splats from image pairs for scalable generalizable 3d reconstruction. In *CVPR*, 2024. 8

[5] Jaeyoung Chung, Jeongtaek Oh, and Kyoung Mu Lee. Depth-regularized optimization for 3d gaussian splatting in few-shot images. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 811–820, 2024. 4

[6] Datacolor. Spyder checkr photo (model sck300). https://www.datacolor.com/spyder/products/spyder-checkr-photo/, 2025. Accessed: 2025-11-12. 5

[7] docs.arduino.cc. Arduino due. Accessed: March 7, 2025. 5

[8] Jinwei Gu, Yasunobu Hitomi, Tomoo Mitsunaga, and Shree Nayar. Coded rolling shutter photography: Flexible space-time sampling. In *2010 IEEE International Conference on Computational Photography (ICCP)*, pages 1–8. IEEE, 2010. 2

[9] Yasunobu Hitomi, Jinwei Gu, Mohit Gupta, Tomoo Mitsunaga, and Shree K Nayar. Video from a single coded exposure photograph using a learned over-complete dictionary. In *2011 International Conference on Computer Vision*, pages 287–294. IEEE, 2011. 2

[10] Jason Holloway, Aswin C Sankaranarayanan, Ashok Veeraraghavan, and Salil Tambe. Flutter shutter video camera for compressive sensing of videos. In *2012 IEEE International Conference on Computational Photography (ICCP)*, pages 1–9. IEEE, 2012. 2

[11] Yi-Hua Huang, Yang-Tian Sun, Ziyi Yang, Xiaoyang Lyu, Yan-Pei Cao, and Xiaojuan Qi. Sc-gs: Sparse-controlled gaussian splatting for editable dynamic scenes. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 4220–4230, 2024. 3

[12] IDS Imaging Development Systems. Ids imaging development systems. https://en.ids-imaging.com/. Accessed: March 8, 2025. 5

[13] Christian Jaques, Emmanuel Pignat, Sylvain Calinon, and Michael Liebling. Temporal super-resolution microscopy using a hue-encoded shutter. *Biomed. Opt. Express*, 10(9): 4727–4741, 2019. 2, 8

[14] Bernhard Kerbl, Georgios Kopanas, Thomas Leimkühler, and George Drettakis. 3d gaussian splatting for real-time radiance field rendering. *ACM Trans. Graph.*, 42(4):139–1, 2023. 3, 7

[15] Agelos Kratimenos, Jiahui Lei, and Kostas Daniilidis. Dynmf: Neural motion factorization for real-time dynamic view synthesis with 3d gaussian splatting. In *European Conference on Computer Vision*, pages 252–269. Springer, 2024. 3

[16] Kyros Kutulakos. The ultimate video camera. https://www.youtube.com/watch?v=7i8E6uQQtog, 2024. Online; accessed 2025. 1

[17] Jiahui Lei, Yijia Weng, Adam W. Harley, Leonidas J. Guibas, and Kostas Daniilidis. Mosca: Dynamic gaussian fusion from casual videos via 4d motion scaffolds. *2025 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 6165–6177, 2024. 3

[18] Zhengqi Li, Simon Niklaus, Noah Snavely, and Oliver Wang. Neural scene flow fields for space-time view synthesis of dynamic scenes. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2021. 3

[19] Zhan Li, Zhang Chen, Zhong Li, and Yi Xu. Spacetime gaussian feature splatting for real-time dynamic view synthesis. *2024 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 8508–8520, 2023. 3

[20] Zhengqi Li, Qianqian Wang, Forrester Cole, Richard Tucker, and Noah Snavely. Dynibar: Neural dynamic image-based rendering. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2023. 3

[21] 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. In *Proc. IEEE/CVF Winter Conference on Applications of Computer Vision (WACV)*, 2025. 3

[22] Yiming Liang, Tianhan Xu, and Yuta Kikuchi. HiMoR: Monocular deformable gaussian reconstruction with hierarchical motion representation. In *CVPR*, 2025. 3

[23] Smart Vision Lights. Mobl (rgbw) series backlight datasheet. Technical report, Smart Vision Lights, 2022. Revision 11/11/2022. 5

[24] Youtian Lin, Zuozhuo Dai, Siyu Zhu, and Yao Yao. Gaussian-flow: 4d reconstruction with dynamic 3d gaussian particle. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 21136–21145, 2024. 3, 4

[25] Cheng Liu, Shuai Xiong, Yongchao Geng, Song Cheng, Fang Hu, Bo Shao, Fang Li, and Jie Zhang. An embedded high-precision gnss-visual-inertial multi-sensor fusion suite. *NAVIGATION: Journal of the Institute of Navigation*, 70: navi.607, 2023. 8

[26] Jonathon Luiten, Georgios Kopanas, Bastian Leibe, and Deva Ramanan. Dynamic 3d gaussians: Tracking by persistent dynamic view synthesis. In *2024 International Conference on 3D Vision (3DV)*, pages 800–809. IEEE, 2024. 3

[27] Julien NP Martel, Lorenz K Mueller, Stephen J Carey, Piotr Dudek, and Gordon Wetzstein. Neural sensors: Learning pixel exposures for hdr imaging and video compressive sensing with programmable sensors. *IEEE transactions on pattern analysis and machine intelligence*, 42(7):1642–1653, 2020. 2

[28] Kristina Monakhova, Vi Tran, Grace Kuo, and Laura Waller. Untrained networks for compressive lensless photography. *Optics Express*, 29(13):20913–20929, 2021. 2

[29] Keunhong Park, Utkarsh Sinha, Jonathan T. Barron, Sofien Bouaziz, Dan B Goldman, Steven M. Seitz, and Ricardo Martin-Brualla. Nerfies: Deformable neural radiance fields. *ICCV*, 2021. 3

[30] Keunhong Park, Utkarsh Sinha, Peter Hedman, Jonathan T. Barron, Sofien Bouaziz, Dan B Goldman, Ricardo Martin-Brualla, and Steven M. Seitz. Hypernerf: A higher-dimensional representation for topologically varying neural radiance fields. *ACM Trans. Graph.*, 40(6), 2021. 3

[31] Travis Portz, Li Zhang, and Hongrui Jiang. Random coded sampling for high-speed hdr video. In *IEEE International Conference on Computational Photography (ICCP)*, pages 1–8. IEEE, 2013. 2

[32] Albert Pumarola, Enric Corona, Gerard Pons-Moll, and Francesc Moreno-Noguer. D-nerf: Neural radiance fields for dynamic scenes. *2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 10313–10322, 2020. 3

[33] Ramesh Raskar, Amit Agrawal, and Jack Tumblin. Coded exposure photography: motion deblurring using fluttered shutter. In *Acm Siggraph 2006 Papers*, pages 795–804. 2006. 2

[34] Dikpal Reddy, Ashok Veeraraghavan, and Rama Chellappa. P2c2: Programmable pixel compressive camera for high speed imaging. In *CVPR 2011*, pages 329–336. IEEE, 2011. 2

[35] Erik Reinhard, Erum Arif Khan, Ahmet Oguz Akyuz, and Garrett Johnson. *Color imaging: fundamentals and applications*. CRC Press, 2008. 5

[36] Yoav Y Schechner, Shree K Nayar, and Peter N Belhumeur. Multiplexing for optimal lighting. *IEEE Transactions on pattern analysis and machine intelligence*, 29(8):1339–1354, 2007. 8

[37] Johannes Lutz Schönberger and Jan-Michael Frahm. Structure-from-motion revisited. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016. 5

[38] M. Sheinin, M. O'Toole, and S. G. Narasimhan. Deconvolving diffraction for fast imaging of sparse scenes. In *Proc. ICCP*. IEEE, 2021. 2, 7, 8

[39] Kevin Tandi, Xiang Dai, Chinmay Talegaonkar, Gal Mishne, and Nick Antipa. Rngcam: High-speed video from rolling & global shutter measurements. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 8830–8840, 2025. 2

[40] Megha H. Tippur and Edward H. Adelson. Rainbowsight: A family of generalizable, curved, camera-based tactile sensors for shape reconstruction, 2024. 2

[41] Stepan Tulyakov, Daniel Gehrig, Stamatios Georgoulis, Julius Erbach, Mathias Gehrig, Yuanyou Li, and Davide Scaramuzza. Time lens: Event-based video frame interpolation. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 16155–16164, 2021. 2

[42] Stepan Tulyakov, Alfredo Bochicchio, Daniel Gehrig, Stamatios Georgoulis, Yuanyou Li, and Davide Scaramuzza. Time lens++: Event-based frame interpolation with parametric non-linear flow and multi-scale fusion. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 17755–17764, 2022. 2

[43] Ashok Veeraraghavan, Dikpal Reddy, and Ramesh Raskar. Coded strobing photography: Compressive sensing of high speed periodic videos. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 33:671 – 686, 2011. 3

[44] Dhruv Verma, Ian Ruffolo, David B. Lindell, Kiriakos N. Kutulakos, and Alex Mariakakis. Chromaflash: Snapshot hyperspectral imaging using rolling shutter cameras. *Proc. ACM Interact. Mob. Wearable Ubiquitous Technol.*, 8(3), 2024. 2

[45] Qianqian Wang, Vickie Ye, Hang Gao, Weijia Zeng, Jake Austin, Zhengqi Li, and Angjoo Kanazawa. Shape of motion: 4d reconstruction from a single video. In *International Conference on Computer Vision (ICCV)*, 2025. 3

[46] Gil Weinberg and Ori Katz. 100,000 frames-per-second compressive imaging with a conventional rolling-shutter camera by random point-spread-function engineering. *Optics Express*, 28(21):30616–30625, 2020. 2

[47] 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 *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 20310–20320, 2024. 3

[48] Jinhui Xiong, Ramzi Idoughi, Andres A Aguirre-Pablo, Abdulrahman B Aljedaani, Xiong Dun, Qiang Fu, Sigurdur T Thoroddsen, and Wolfgang Heidrich. Rainbow particle imaging velocimetry for dense 3d fluid velocity imaging. *ACM Transactions on Graphics (TOG)*, 36(4):1–14, 2017. 2

[49] Jinhui Xiong, Andres Aguirre-Pablo, Ramzi Idoughi, S Thoroddsen, and W Heidrich. Rainbowpiv with improved depth resolution – design and comparative study with tomopiv. *Measurement Science and Technology*, 32, 2020. 2

[50] Ziyi Yang, Xinyu Gao, Wen Zhou, Shaohui Jiao, Yuqing Zhang, and Xiaogang Jin. Deformable 3d gaussians for high-fidelity monocular dynamic scene reconstruction. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 20331–20341, 2024. 3