NeLU3D: Neural Inverse Structured Light without Explicitly Modeling the Projector: Supplemental Document

Giancarlo Pereira[1], David Fouhey[1], Claudio T. Silva[1], and Daniele Panozzo[1]

New York University, New York NY, USA
{giancarlo.pereira,fouhey,csilva,panozzo}@nyu.edu

A Implementation Details

A.1 Volume Rendering

We follow the volume rendering implementation presented in NeuS [12]. For a camera pixel i, we shoot a ray from the camera's origin $\mathbf{o} \in \mathbb{R}^3$ in the direction of $\mathbf{v}_i \in \mathbb{R}^3$.

We evaluate the ray at N evenly spaced depth positions. At a certain depth d, then, we retrieve a 3D coordinate $\mathbf{x}(i,d) = \mathbf{o} + d\mathbf{v}_i$. We pass this 3D coordinate $\mathbf{x}$ to the MLP f, which outputs the SDF value. Following NeuS, we need to convert the SDF value into density and transmittance for volume rendering:

$$\Phi_s(f(\mathbf{x}(i,d))) = \frac{1}{1+\exp(-sf(\mathbf{x}(i,d)))} \tag{S1}$$

$$\sigma(i,d) = \max\left(\frac{-\frac{\partial}{\partial d}\Phi_s(f(\mathbf{x}(i,d)))}{\Phi_s(f(\mathbf{x}(i,d)))}, 0\right) \tag{S2}$$

$$T(i,d) = \exp\left(-\int_{d_{min}}^{d} \sigma(i,\delta)d\delta\right) \tag{S3}$$

where σ is the density function and T is the transmittance at point $\mathbf{x} = \mathbf{x}(i,d)$.

We also pass the 3D coordinate $\mathbf{x}$ to the frozen color MLP c. In the end, we alpha-composite the final rendered color at pixel i using the density σ along the ray:

$$\tilde{\mathbf{I}}(i) = \int_{d_{min}}^{d_{max}} c(\mathbf{x}(i,\delta))\sigma(i,\delta)d\delta \tag{S4}$$

A.2 Network Architecture & Training

The neural inverse structured light implementation is built on top of NerfAcc [3], an open-source neural inverse rendering framework, with a multiresolution hash encoding (MHE) from Instant-NGP [8]. We use the same settings from TurboSL's

open-source code [6][1]: a MHE fused with a multilayer perceptron (MLP) for the signed distance function (SDF) and residual/ambient contribution. For these two networks, we use 16 hash feature grids and 2 features for each level. The resolution of the coarsest grid is 16 and each level has a scale factor of 1.45. Then, the MLPs are built with 2 hidden layers of width 64 and ReLU activations. A sigmoid activation is applied to the outputs for the residual network to constrain it between 0 and 1 (and then later multiplied by the white capture to further constrain the residual contribution to the final image), whereas no activation function was used for the outputs of the SDF since it is unconstrained.

For the first 500 iterations, we run a warm-up of the learning rate: ramping it linearly from 0.01 to 1. After that, we decay the learning rate exponentially to reach 0.1 at the last iteration.

For occupancy grid and ray sampling, we also follow TurboSL [6]. We maintain an occupancy grid to prune empty regions for efficient sampling. Each dimension of the grid has a resolution of 128 and the density threshold for pruning is set to 10^{-3}.

We sample each ray in 1024 evenly spaced positions to render a single pixel; if a sample falls in an empty region of the occupancy grid, we skip it. In order to keep the total number of samples per iteration constant throughout training, we adjust the number of rays sampled at every iteration to compensate. We start the optimization loop with 256 rays, all sampled randomly. We set a hard limit to a maximum of 8192 rays sampled at each iteration. We shift and scale all scenes such that they fall inside a bounding sphere of radius 1.2 for optimization. However, we output all final meshes adjusted back to meters – all the outputted OBJ files from our demo code are in metric scale.

A.3 Masking

For *NeLU3D*, we calculate binary mask $\mathcal{M}$ as:

$$\mathcal{M} = \mathbf{I}_{\text{white}} > \text{threshold} \times \max\left(\mathbf{I}_{\text{white}}\right) \tag{S5}$$

Below are the values we use for all scenes in the paper and supplementary:

Unfortunately, for low-albedo objects, this approach to extracting a mask causes several points from the background to be considered as part of the mask $\mathcal{M}$. We **do not alter** any of these masks for the visualizations in this paper - for instance, one can see a lot of the noisy background reconstructed around the scissor in Fig. S39. We remove the background, however, for the error evaluation in Tab. 1 of main paper and in Sec. B.10 for the following scenes: chair, dodo, first aid kit, keyboard, mouse, power adapter, radio, remote, scissors, and teapot.

A.4 Lookup Table Calibration

We describe more details on how we calibrate and learn the lookup table as described in Pereira *et al.* [10]. The idea is to predict k color intensities given a 3D coordinate (in meters).

[1] https://github.com/parsa-spm/TurboSL

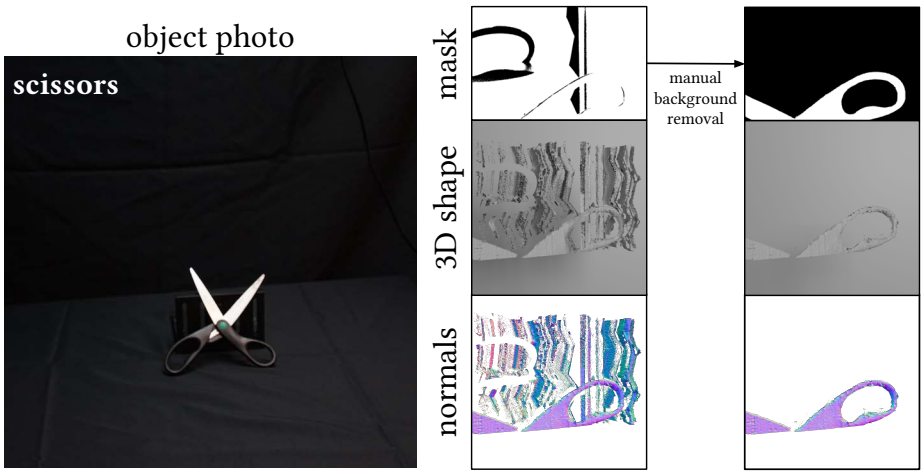
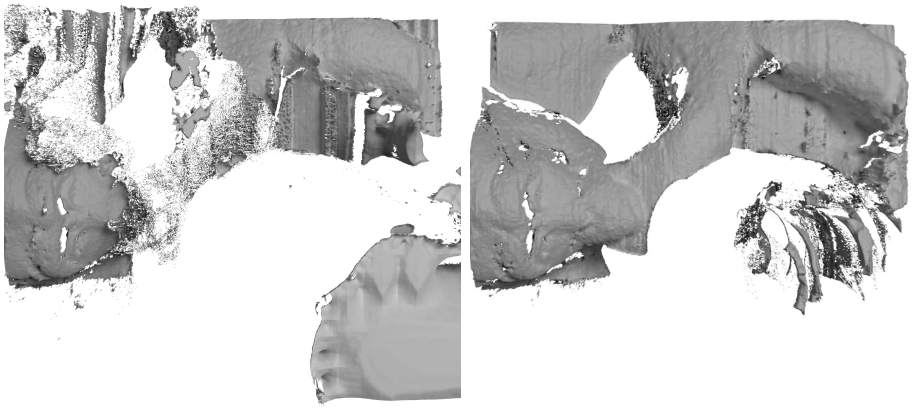


Fig. S1: Masking. For error evaluation, we manually remove the background of a dozen scenes (chair, dodo, first aid kit, keyboard, mouse, power adapter, radio, remote, scissors, and teapot). We do so to avoid comparing background points, which are outside of the volume of calibration for LookUp3D and *NeLU3D*.

There are various ways to bring the lookup table into a differentiable system. For instance, if one can hold the whole table in GPU memory, one can take a 3D position and retrieve the corresponding k intensities - for differentiability, one can then take its adjacent neighbors and use finite difference to analytically calculate gradients. The calibrated lookup tables, however, are not in a evenly spaced grid; therefore, a trivial implementation of nearest neighbor search grows intractable for the sizes of these calibrated tables (all with more than half a billion positions).

A simpler solution is to force the table into an even grid: with this option we can use PyTorch's *grid_sample* which is fast and handles gradients, but memory grows cubically with the grid size. At grid resolution 1024^3, we run out of memory in an RTX A6000 GPU, which has 49140MiB of VRAM. The MLP retains far greater detail at a much smaller memory requirement (Fig. 3). We are also interested in efficiency, especially allowing the whole system to fit and run in a desktop GPU and to output a single surface reconstruction fast.

Ultimately, we choose to train color MLP $c : \mathbb{R}^3 \rightarrow [0,1]^k$ (where k is the number of patterns). A good encoding choice is necessary to learn the color mapping well. We chose to encode the 3D position with a multiresolution hash encoding (MHE). For standardization, in all results of the main paper and this supplemental document, we use $L = 16$ hash table levels, $F = 2$ features for each level, and a total of $T = 2^{19}$ entries. The resolution of the coarsest grid is 16 and each level has a scale factor of 1.45. The MLP then is very compact: it has 2 hidden layers of width 64 and ReLU activations. We apply a sigmoid activation to the color outputs to constrain them between 0 and 1.



(a) Explicit Table (512^3): 23.05mm **(b)** MLP: 0.51mm

Fig. S2: Comparison Between Learned MLP and Explicit Lookup Table. We visualize reconstruction of monkey scene and report median depth error in millimeters for 3 random patterns. The explicit lookup table (Fig. S2a) was unable to accurately capture the geometry of the monkey statue.

The MHE has a few advantages over traditional frequency encoding, all described in InstantNGP [8]. The Fourier encodings require much larger neural networks to learn the fine details of the color mapping; these larger networks then are slower to train and slower to evaluate. For a mapping with 11 Gray code patterns, a positional encoding with 16 frequencies and an MLP of 16 hidden layers with width 128 (approximately a total of 263k parameters) trains at about 11 iterations per second in an RTX A6000 GPU. An MHE with $L = 16$ levels, $T = 2^{19}$ entries per levels and $F = 2$ feature dimensions per entry and an MLP 2 of hidden layers with width 64 (approximately a total of 13.6M parameters) trains at about 45 iterations per second, much faster due to the sparsity of weight updates as explained in [8]. During the 3D reconstruction, similarly, the frequency encoding and larger MLP slow down each iteration - it now runs at about 10 iterations per second, whereas the MHE combined with smaller MLP runs at 18 iterations per second.

To train the color MLP c, the only loss we optimize for is:

$$\mathcal{L} = \sum_{\mathbf{x} \in \mathcal{V}} \|\mathbf{I}(\mathbf{x}) - c(\mathbf{x})\| \tag{S6}$$

where $c(\mathbf{x}) \in \mathbb{R}^k$ is the learned output intensity for the k patterns, $\mathbf{I}$ is the intensity captured by the camera, and we sum over all 3D coordinates from the volume of calibration $\mathcal{V}$. The intensity $\mathbf{I}$ has ambient contribution removed and is normalized by the white flash. We use both $L1$ and $L2$ norms, with respective weights 0.1 and 1, to be robust to noisy pixels and outliers.

For all results in the main paper and supplemental, we trained the color network for $5,000$ iterations. At every iteration, we randomly sample $262,144$ points from the volume of calibration (each tends to have between 400 million and 10 billion points).

Below we also experiment with different parameters for the multiresolution hash encoding (number of levels L, number of feature dimensions per level F, and hash table size) T and plot the calibration results (PSNR in dB) in Fig. S3.

We also evaluate these same different parameters for the multiresolution hash encoding in the quality of reconstruction of different objects. We see no difference in the depth errors, but we do see significant differences in the training time for 10,000 iterations and variations in PSNR of the neurally rendered images when compared to the real SL captures. We plot these results in Fig. S4 for $k = 11$ Gray codes, $k = 6$ Hilbert patterns, and $k = 3$ random patterns for a handful of different reconstructed objects.

A.5 Hardware Details

The details below on projector calibration are not needed for our proposed method. We describe these details, however, for clarity and reproducibility purposes. In Fig. S5, we show the results of geometric and radiometric calibrations of the two digital projectors we use in this paper: the DLP projector (**top row** of Fig. S5) is the Texas Instruments DLP4710EVM-LC projector, with a native resolution of 1920×1080 and a DMD chip; the LCD projector (**bottom row** of Fig. S5) is the VOPLLS Mini Beaner N3, a 50USD projector equipped with an LCD screen and very cheap optics.

For the camera-projector calibration used for TurboSL [6], ZNCC [5], and for dense SL with 44 Gray codes, we use the method presented by Moreno and Taubin [7]. This method outputs the extrinsic and intrinsic parameters of the projector, which are needed for accurate triangulation. The **left column** of Fig. S5 shows the projector lens distortion in pixel units, with the contours showing level sets. The **middle column** of Fig. S5 shows the reprojection error, also in pixel units, represents difference between measured calibration points to and the projected points after calibration – this is a proxy for both intrinsic and extrinsic calibration. The smaller the reprojection error, the better fit the calibration has achieved.

For the camera-projector radiometric calibration, which again is needed for TurboSL [6] and ZNCC [5], we project grayscale images from 0 to 255 onto a white diffuse board (**left column** of Fig. S5) We then fit a polynomial response curve to accurately compensate for any black-box processing done by the projector and to accurately model the input to output intensity response.

B Supplemental Results

In this section, we present more results using *NeLU3D*, including using different seeds, which control the stochasticity of our method (Sec. B.1), more patterns

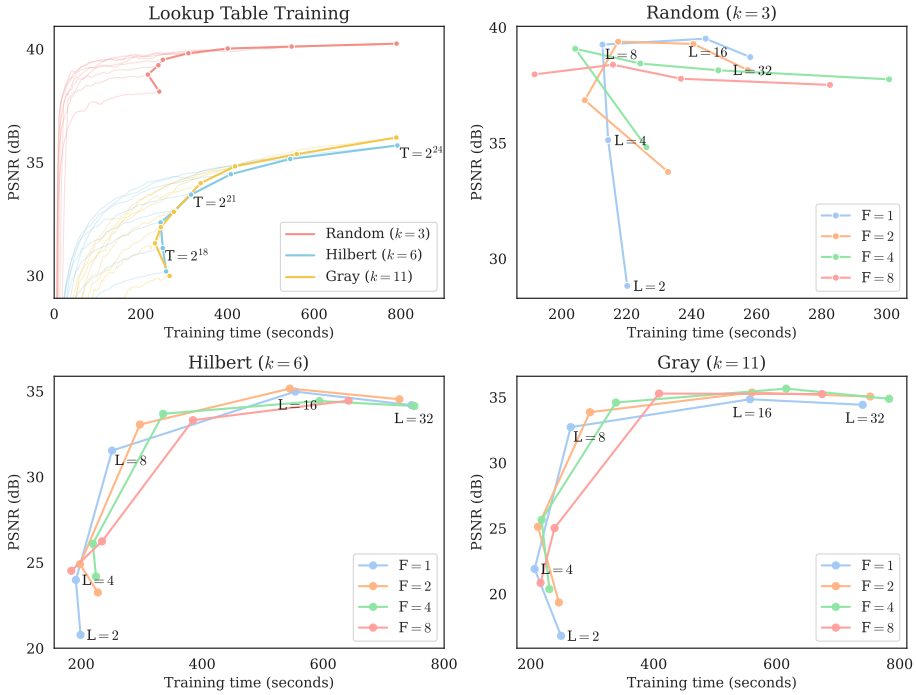


Fig. S3: Neural Lookup Table Calibration Quality for Different Encoding Parameter Choices. We evaluate the error for learning the lookup tables as a neural representation using multiresolution hash encoding (MHE) and a multilayer perceptron (MLP). The input of the MHE is a 3D coordinate $\mathbf{x}$, its output is a feature vector; the MHE output then goes to the MLP. MLP has 2 hidden layers, each with 64 neurons; its output is the k intensities of the pattern at the given 3D coordinate $\mathbf{x}$. **Upper left plot:** error over training time for varying hash table size T which determines the number of trainable encoding parameters. Increasing T improves reconstruction, at the cost of higher memory usage and slower training and inference. **All other plots:** error over training time for fixed values of feature dimensionality F as the number of hash table levels L is varied. To maintain a roughly equal trainable parameter count of encoding parameters, the hash table size T is set according to $L \times F \times T = 2^{28}$. Unfortunately, for $F = 8$, we could not reach $L = 32$ levels as it is not supported in our RTX A6000 GPU (CUDA throws a "cudaErrorInvalidConfiguration" error). Training finishes after 10,000 steps for all plots in all columns.

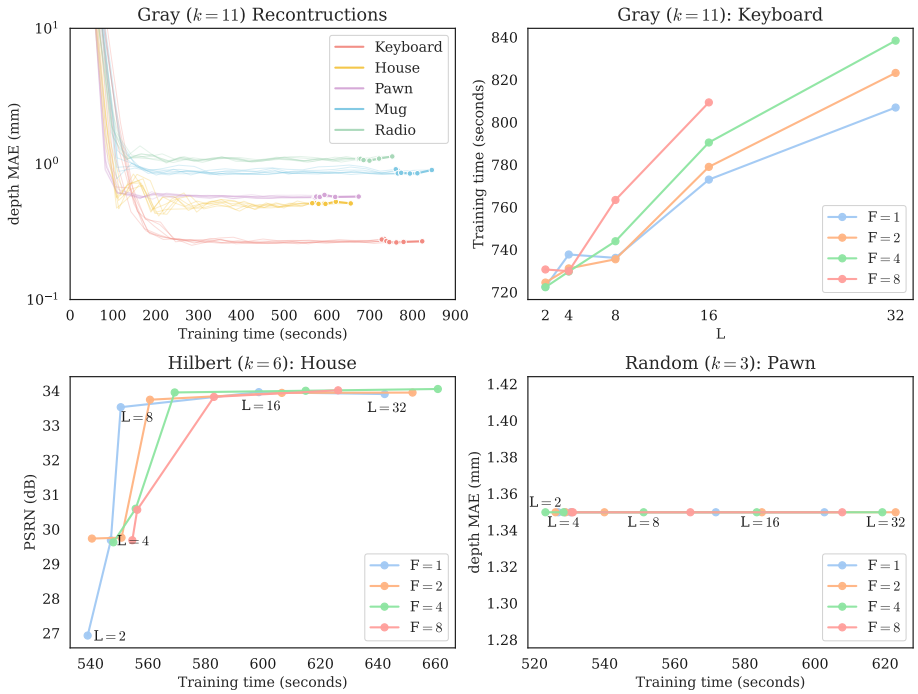


Fig. S4: Neural Reconstruction Quality for Different Encoding Parameter Choices During Calibration. We evaluate the quality for neural inverse structured light reconstructions using the learned color MLPs c. We alter the parameters of multiresolution hash encoding (MHE) (check Fig. S3).

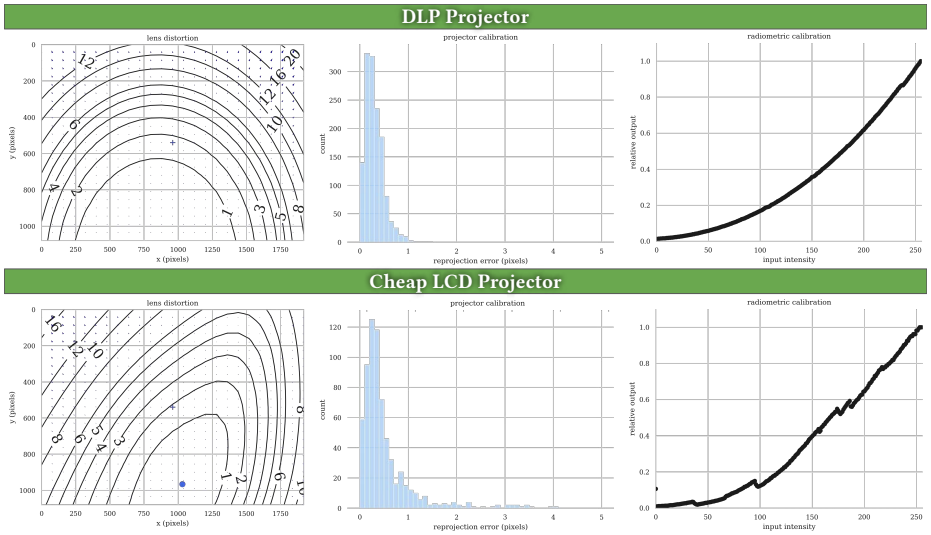


Fig. S5: Projector Calibration. We calibrate the DLP and LCD projectors with the same procedure. **Top row**: Texas Instruments DLP4710EVM-LC projector, a high-quality DLP projector equipped with a DMD chip; **bottom row**: VOPLLS Mini Beaner N3, a cheap LCD projector. **Left column**: lens distortion (measured in pixels) of the projector optics. **Middle column**: reprojection error (measured in pixels) – this takes into account both intrinsic and extrinsic parameters, highlighting the accuracy of the projector calibration. **Right column**: radiometric calibration of the projectors. The radiometric calibration is needed for TurboSL [6] and ZNCC [5] reconstructions, since they rely on the actual projected intensity value for each pixel.

(Sec. B.2), different patterns (Sec. B.3), different α values to balance between $\mathcal{L}_{\text{pattern}}$ and $\mathcal{L}_{\text{LookUp}}$ (Sec. B.4), and different exposure times and signal-to-noise-ratio (Sec. B.5). We additionally present the capture structured light images and 3D reconstructions from Fig. S6's object collection: check Sec. B.6 for more static results and Sec. B.7 for dynamic results. For more comparisons and error results with other Structured Light methods, check Sec. B.10.

Alongside an open-source implementation of method, we will release the unprocessed TIFF images captured with the Atlas camera and the fitted color MLPs so others can play with the data and explore our proposed system.

Fig. S6: Collection of Scanned Objects. We have scanned a collection of objects: from real-world, day-to-day items (keyboard, mug, alcohol wipes pack, cloth) to 3D printed objects (dodo, chair) to precisely milled ones (pawn, rook). This is to show the range of different geometries and materials that we can scan with *NeLU3D*.

B.1 Stochasticity of Optimization

The pixelwise decoders ZNCC [5] and LookUp3D [10] produce deterministic results. Our method (and TurboSL [6]) are stochastic by nature, since we initiate weights and biases of neural networks and optimize them to match captured structured light images. In this section, we run the same scene of a 3D printed house with the same settings, but with different seeds for random number generators: we use $0, 1, 42, 101999918, 2026$. We use the Atlas camera and the DLP

projector, with a fixed exposure of 1/4 seconds and the same three patterns (with the additional white flash) from the third column of Fig. 5.

Although we do not exploit the stochasticity in this submission (we set the seed to 42 for all results in main paper and supplementary), running the same scene with different seeds might create a probabilistic view of the quality of reconstruction. Similarly, Mirdehghan *et al.* in TurboSL [6] used a "front" and "back" rendering to output an error proxy wherever the reconstruction disagreed between the two renderings. "Back" rendering meant artificially placing the camera on the other side of scene and accumulating the weights in opposite order for image formation.

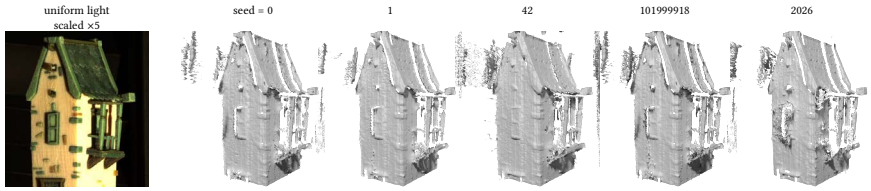


Fig. S7: Stochasticity of Optimization.

B.2 More Patterns

We use the DLP Projector and monochromatic Atlas Camera to study the effect of using more patterns while scanning objects (Fig. S8). We always capture the white flash, so we consider the number of different monochromatic (8-bit) patterns we project.

As expected, as we increase the number of patterns used, we get better surface quality, shown here as the error map and median error decreasing as we increase the number of patterns used. The errors are calculated against dense SL triangulation with 44 Gray codes (22 horizontal and 22 vertical). But, using just 3 patterns with the additional white flash already yields sub-mm accuracy and fine geometric details in both the black keyboard (**top row** of Fig. S8) and the monkey statue (**bottom row** of Fig. S8).

We would like to note that this is also the case with the pixelwise decoders (ZNCC [5] and LookUp3D [10]) and TurboSL [6]: the more patterns used, the more information is available for accurately reconstructing the geometry.

B.3 Different Patterns

We test *NeLU3D* with different pattern combinations (Fig. S9, Fig. S10, and Fig. S11); we would like to note that we are not claiming any new state-of-the-art pattern. Quite the contrary: we want to showcase that the system works with

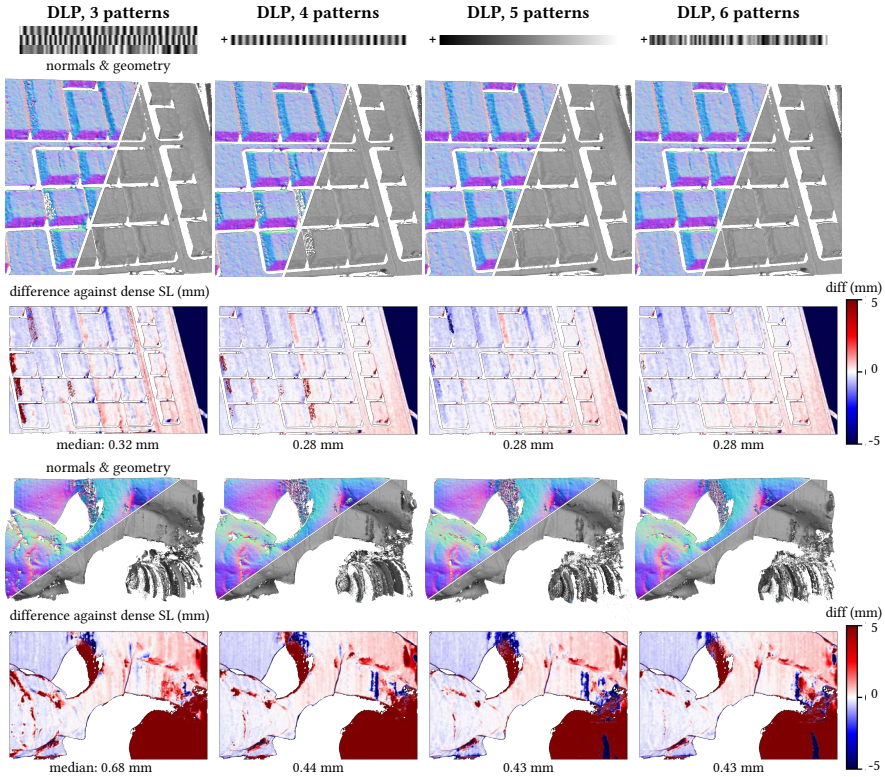


Fig. S8: Effect of More Patterns. We assess the impact on reconstruction quality of using three or more patterns to reconstruct two objects: a black keyboard (**top two rows**) and a monkey status (**bottom two rows**). The difference metric is calculated against SL triangulation with 44 gary codes (22 vertical and 22 horizontal) since we do not have ground truth of these objects. As expected, the more patterns we use to reconstruct the objects, the lower the error metric and the higher the final surface quality.

simple patterns which are not tailored to specific scenes or set-ups. Additionally, when performing reconstructions with *NeLU3D*, we do not need to perform any radiometric calibration or model any of the projector optics, which can cause blurring, defocusing, vignetting, *etc*. In Fig. S9, Fig. S10, and Fig. S11, we have a few different families of patterns: sinusoidals, triangular, ramp, Hilbert (a space-filling curve [2]), random, and micro phase shifting pattern [1].

We would also like to include details for reproducibility here. Both the projectors we use have a native resolution of 1920×1080 and 8-bit projection, so we design all these patterns as vertical stripes ranging from 0 to 255 in intensity. For the random patterns, we ran a random number generator (with seed 0) with *numpy* for a 20×3 array in the range $[0, 1]$. For each of the three random pattern sequences, we evenly distribute those 20 intensities horizontally and connected the values with cubic splines using *scipy*. We then convert them to 8-bit unsigned integers for projection.

B.4 Effect of Balancing $\mathcal{L}_{\text{pattern}}$ and $\mathcal{L}_{\text{LookUp}}$

Here we present two very different scenes: a black keyboard and the milled pawn. The keyboard has a very low albedo and, thus, the camera collects less light with the exact same parameters and settings, whereas the milled pawn is coated with a glossy white paint, making it an object with very high albedo; the only thing we alter is the α value that balances between $\mathcal{L}_{\text{pattern}}$ and $\mathcal{L}_{\text{LookUp}}$ according to Eq. 4. We do a sweep from $\alpha = 0$ (when $\mathcal{L}_{\text{pattern}}$ is the only loss for rendered image) to $\alpha = 1$ (when $\mathcal{L}_{\text{LookUp}}$ is the only loss for rendered image).

We use the Atlas camera and the DLP projector, with a fixed exposure of 1/4 seconds and the same three patterns (with the additional white flash) from the left column of Fig. 5. We see in Fig. S12 that the parameter doesn't affect the pawn reconstruction much. However, the parameter makes all the difference for the black keyboard: $\alpha = 0$ breaks the reconstruction, whereas an α close to 1 adds a lot of noise to the final surface reconstruction. For all results in the main paper and in the supplementary, we set $\alpha = 0.1$.

B.5 Effect of Exposure Time

Here we present the same exact scene of the silicone bunny, where we keep all of the reconstruction settings the same. We use the Chronos camera with an exposure of 1.0 millisecond and the analog projector with a fixed RGB pattern developed on film. We record 100 frames (*i.e.* 100 pattern acquisitions and *i.e.* 100 white flash acquisitions) to highlight the impact of exposure time in reconstruction results. By collecting 100 frames (effectively almost 0.2 seconds of light exposure), we can average them to achieve a much higher signal-to-noise ratio without overexposing the camera sensor. With a higher signal-to-noise ratio, the reconstructed surface is improved, as shown by the smoother surface. We perform the same experiment for the milled rook with the Triton10 camera with an exposure of 4.7 milliseconds. We use the same analog projector and the same RGB pattern developed on film.

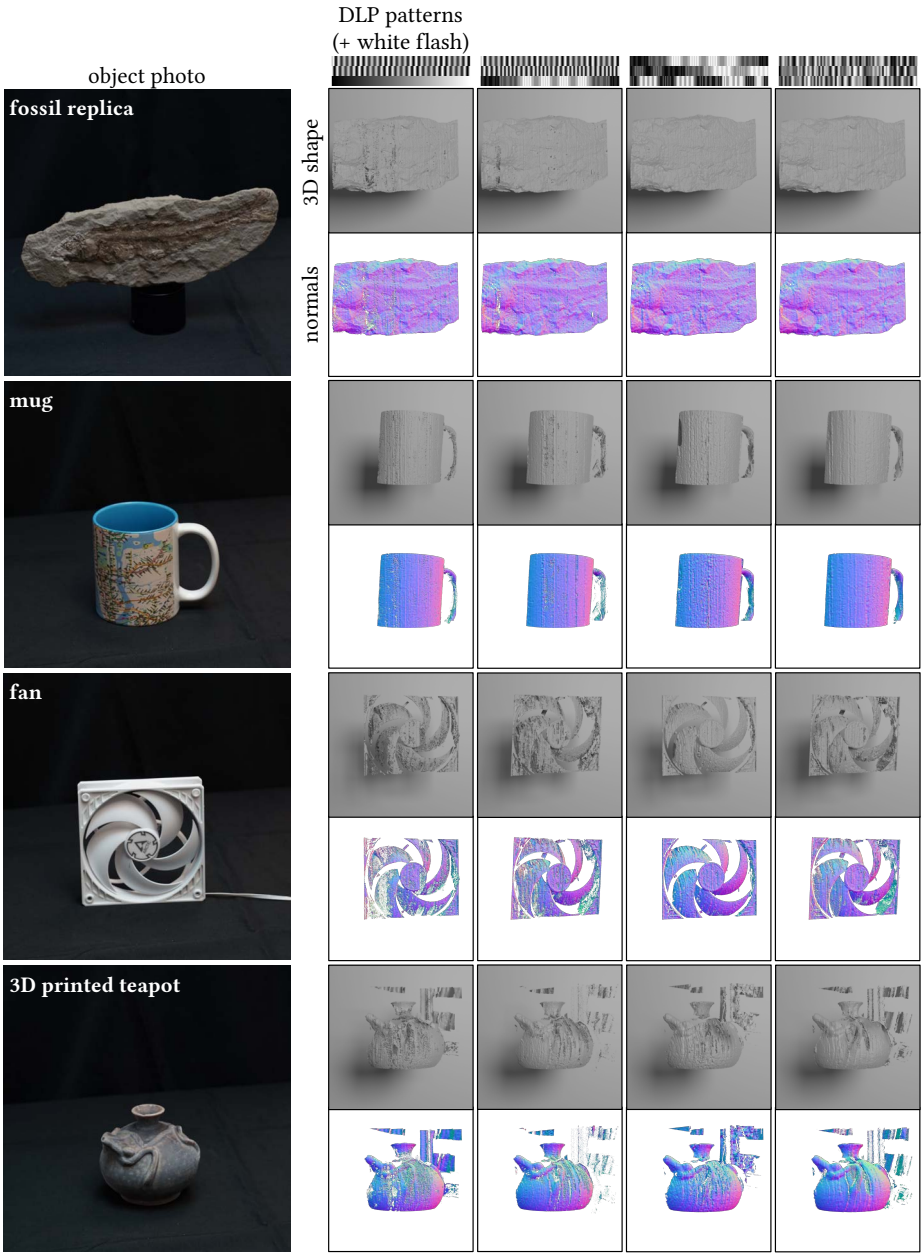


Fig. S9: Reconstructions with Different Patterns (part 1). We reconstruct and render the surface of four objects: a fossil replica, a mug, a fan, and a 3D printed teapot. **Column 1**: 2 phase-shifted sinusoidals and 1 ramp. **Column 2**: 1 sinusoidal, 1 triangular, and 1 space-filling Hilbert [2]. **Column 3**: 3 space-filling Hilbert [2]. **Column 4**: 3 random patterns.

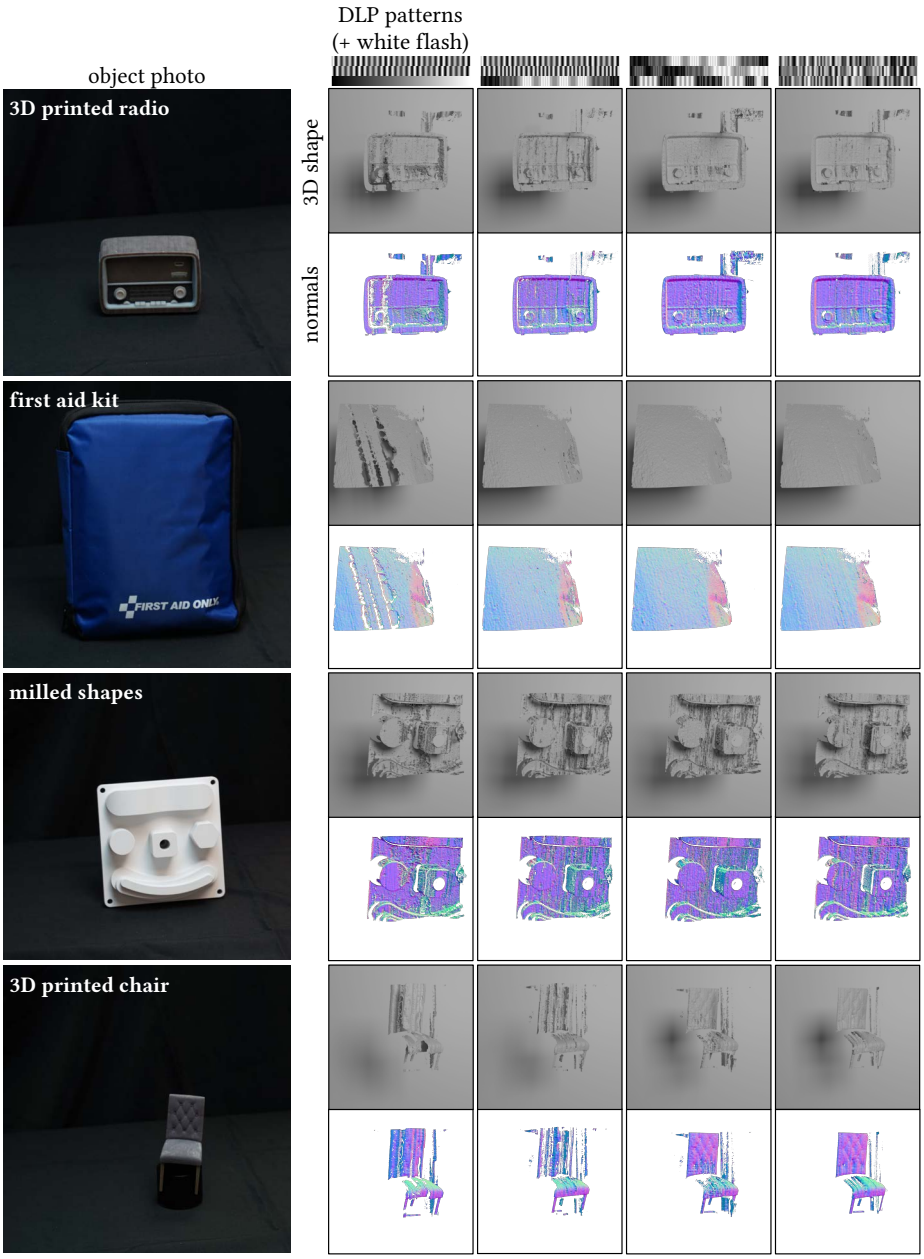


Fig. S10: Reconstructions with Different Patterns (part 2). We reconstruct and render the surface of four objects: a 3D printed radio, a first aid kit, a milled object with different shapes, and a 3D printed chair. **Column 1**: 2 phase-shifted sinusoidals and 1 ramp. **Column 2**: 1 sinusoidal, 1 triangular, and 1 space-filling Hilbert [2]. **Column 3**: 3 space-filling Hilbert [2]. **Column 4**: 3 random patterns.

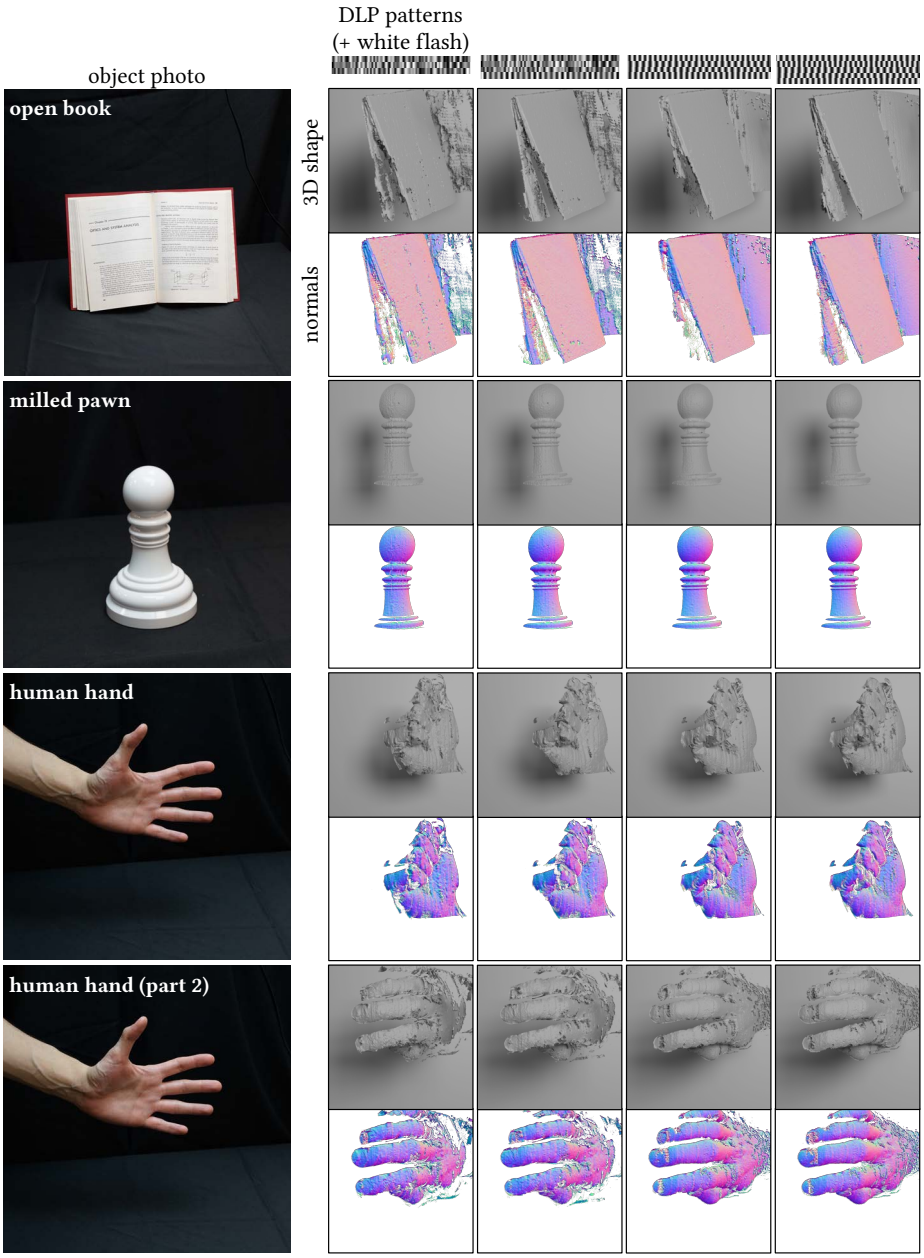


Fig. S11: Reconstructions with Different Patterns (part 3). We reconstruct and render the surface of four scenes: an open book, the milled pawn, a human hand making a fist, and the back of a human hand. **Column 1**: 3 random. **Column 2**: 1 sinusoidal, 3 random. **Column 3**: 4 micro phase shift [1]. **Column 4**: 5 micro phase shift [1].

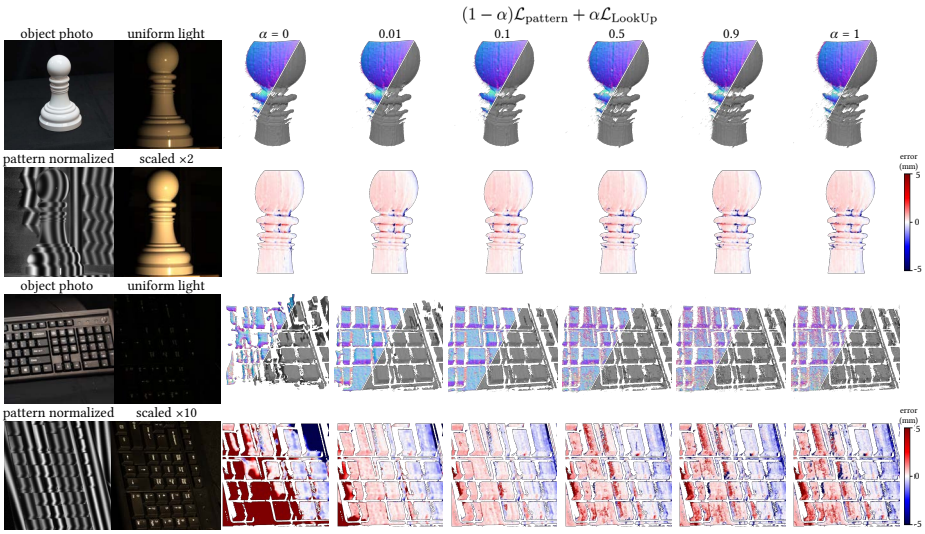


Fig. S12: Effect of parameter α. We highlight here the impact of α for balancing between $\mathcal{L}_{\text{LookUp}}$ and $\mathcal{L}_{\text{pattern}}$, following Eq. 4. **Top row:** milled pawn, with a high albedo. **Bottom row:** black keyboard, with much lower albedo. For darker scenes, the $\mathcal{L}_{\text{pattern}}$ is not strong enough to guide the optimization to the correct surface, whereas $\mathcal{L}_{\text{LookUp}}$ by itself always increases the noise in the reconstructed surface. We experimentally see that a value of α in 10^{-1} order of magnitude works well.

We would like to note that using an analog projector is not a necessity for performing slow-motion capture. As shown in [9], one can use a digital projector equipped with a DMD chip and a high-speed camera to perform structured light triangulation. The authors of the paper outline ways to calibrate the temporal dithering between a high-speed camera and a DLP projector. But, for all our slow-motion results in the main paper and in the supplementary, we use the high-speed cameras with an analog projector.

B.6 Static Results

We include in this section additional static results capture with our scanning setups. We display the camera captures $\mathbf{I}_{\text{white}}$ and $\mathbf{I}_{\text{pattern}}$ for all the figures below, in addition to the residual/ambient term fitted during optimization and the rendered shape (for the rendering, we use Blender to display the normals of the surface and the surface visualized with a glossy BSDF). With the trichromatic cameras, we also display the normalized image $\mathbf{I}_{\text{pattern}}/\mathbf{I}_{\text{white}}$ to highlight the RGB pattern projected onto the scene. Figs. S32-S52 were captured with our Atlas camera and the DLP projector (the caption for each figure indicates which patterns were used for the reconstruction). Figs. S53-S56 were captured with our Atlas camera and the cheap LCD projector. Figs. S57-S61 are of static objects

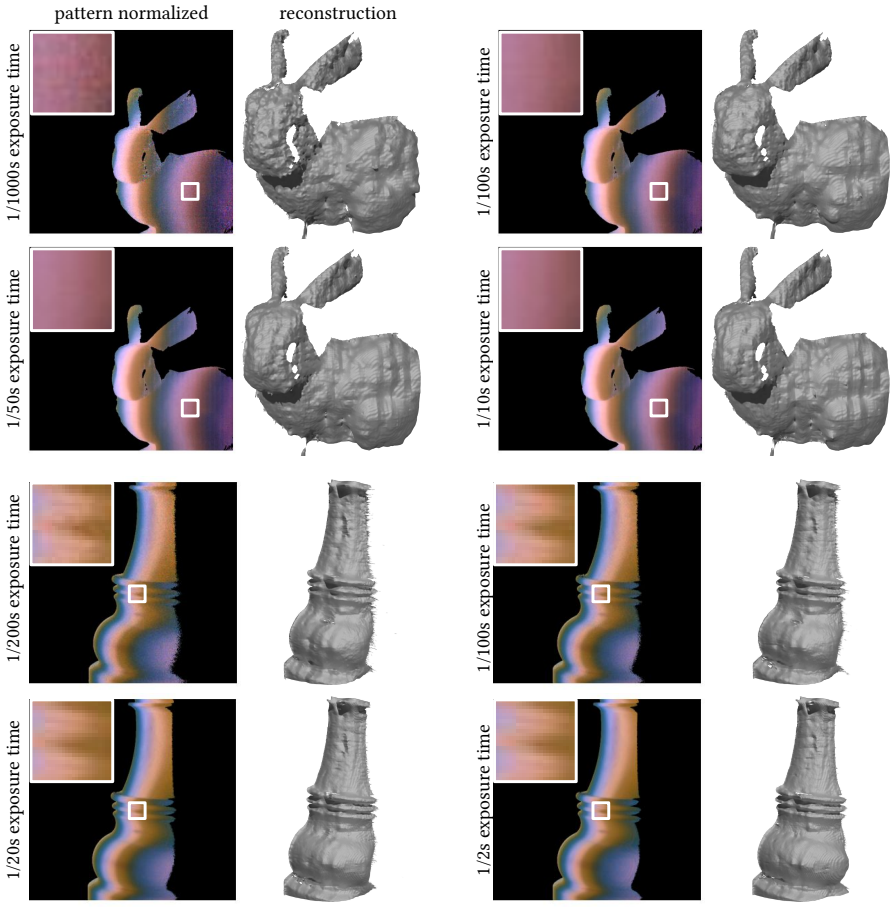


Fig. S13: Effect of Exposure Time. We capture the same bunny scene repeatedly with the Chronos camera and the same milled rook scene repeatedly with the Triton camera. For all reconstructions, we use the analog projector equipped with a helical RGB pattern. For Chronos, each pattern and white flash capture are exposed for 1/1000s; for Triton, 1/200s. As expected, with a higher signal-to-noise ratio, the reconstructed surface quality improves.

captured with our Triton camera at 4.7ms exposure and an RGB pattern on film projected from an analog projector.

B.7 Dynamic Results

All our dynamic results in the main paper and here are done with the open-source analog projector presented in Pereira *et al.* [10]. For the following results, we used two high-speed cameras, Chronos HD camera and Lucid Vision's Triton10 5MP, both equipped with a Bayer filter, to capture dynamic scenes. We reconstruct them frame by frame.

Fig. S62 is a dynamic scene captured with the Chronos camera at 4.9ms exposure and the analog projector. Fig. S63 is a dynamic scene of the bunny being held by its ears and spun around captured with the Triton camera at 4.7ms and the analog projector.

B.8 Monocular Depth Estimation

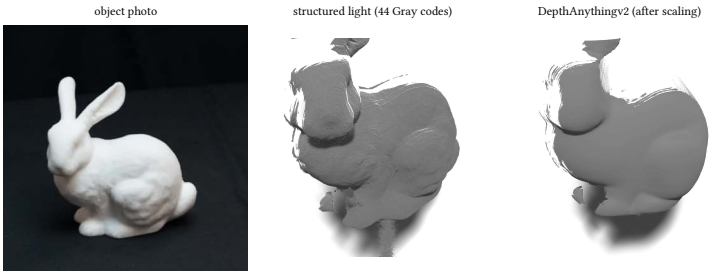


Fig. S14: Monocular Depth Estimation. DepthAnythingv2 [13, 14] predicts a lot of the features correctly, but the overall surface of the bunny is too smooth and flat.

Monocular depth estimation (see, for example, MiDAS [11] and DepthAnything [13, 14]), has surged in popularity in the last couple of years. It is great at tasks involving relative depth; it is, however, still imperfect for metrology. This is because metrology from a single view is an ill-posed problem: a single 2D image cannot discern between scale and depth.

In Fig. S14, we display the silicone bunny reconstructed with traditional SL using 44 Gray codes (22 horizontal and 22 vertical) and the DepthAnythingv2 [14] output after being rescaled to match the SL reconstruction. Although much of the bunny shape is accurately captured by DepthAnythingv2, the output is much flatter than the actual surface of the silicone bunny and a lot of the geometric details are lost. At the moment, we believe such depth prediction tasks are not at the required level for accurate surface reconstruction, but we are

excited about future developments of such learning-based strategies and might incorporate them into the supervision of our reconstructions for future work.

B.9 Failures

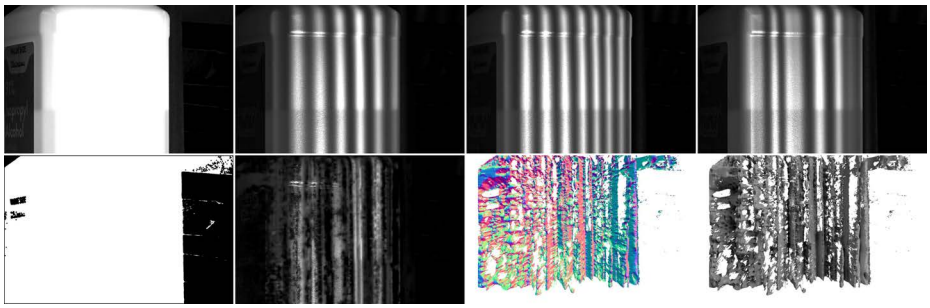


Fig. S15: Failure: translucent object. The IPA bottle is translucent and has liquid inside – the subsurface scattering and refraction completely destroy the reconstruction. **Top row:** captured images from camera under uniform lighting and three patterns (sinusoidal, triangular, and random) from DLP projector (scaled $\times 5$). **Bottom row:** binary mask, fitted residual/ambient term (scaled $\times 10$), normals, and rendered 3D shape.

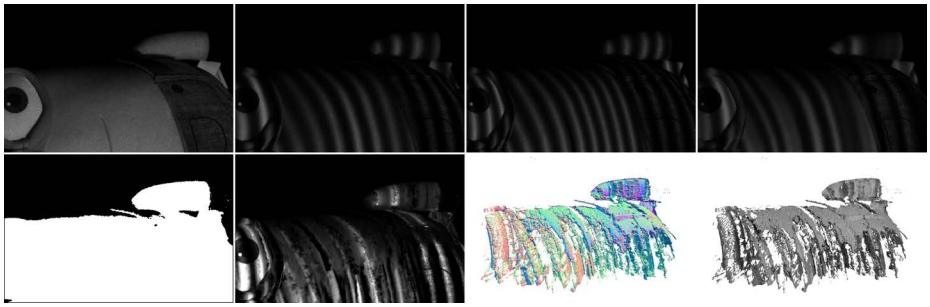


Fig. S16: Failure: fabric toy outside calibrated volume. Top row: captured images from camera under uniform lighting and three patterns (sinusoidal, triangular, and random) from DLP projector. **Bottom row:** binary mask, fitted residual/ambient term (scaled $\times 10$), normals, and rendered 3D shape.

In this section, we add some failed results from *NeLU3D*. These failures also serve to highlight the limitations of the system, namely that

1. transparent and translucent objects and mirror surfaces cannot be scanned (see Fig. S15)

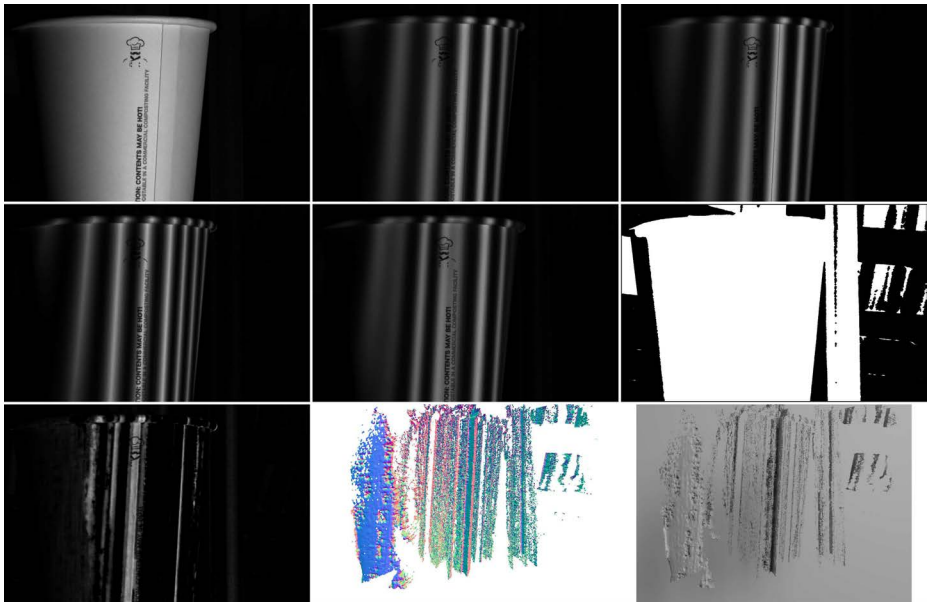


Fig. S17: Failure: paper cup outside calibrated volume. Top row: captured under uniform lighting and two phase-shifted sinusoidal patterns from DLP projector. **Middle row:** two additional pattern captures (one triangular, one random) and the binary mask for reconstruction. **Bottom row:** fitted residual/ambient term (scaled $\times 10$), normals, and rendered 3D shape.

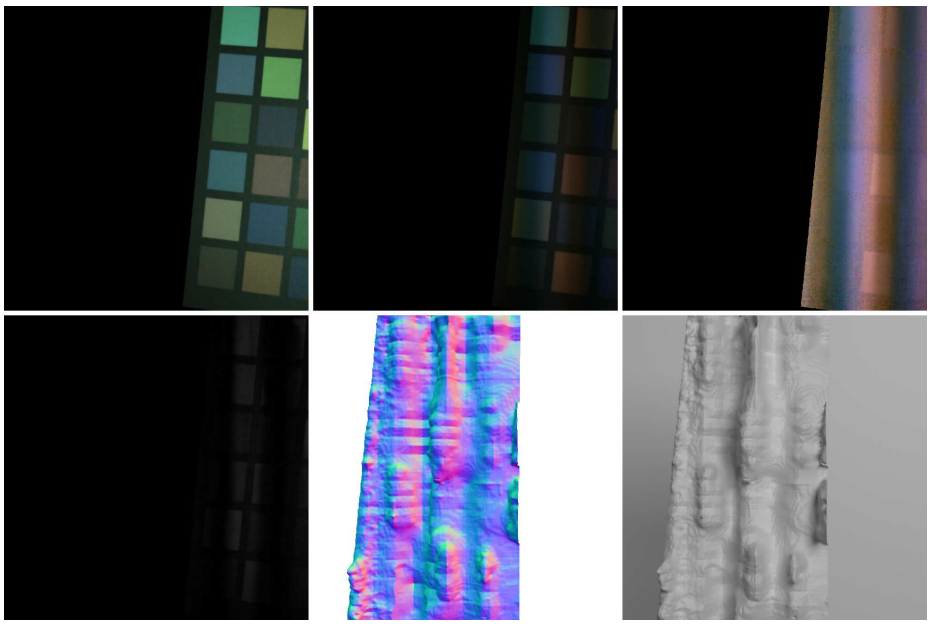


Fig. S18: Failure: color mixing in Chronos, a trichromatic camera, and analog projector. Top row: captured images from camera under uniform lighting and RGB pattern at 4.9 ms exposure (both scaled $\times 10$), with the normalization $\mathbf{I}_{\text{pattern}}/\mathbf{I}_{\text{white}}$. **Bottom row:** fitted residual/ambient term (scaled $\times 10$), normals, and rendered 3D shape.

2. anything outside of the volume of calibration will not converge to the correct surface (see Fig. S16 and Fig. S17)
3. with the trichromatic cameras (Chronos and Triton10), objects with many colors can break reconstruction, since there is a lot of cross-talk between the different color channels in the camera sensor and the light of the projector does not come from monochromatic R, G, B lasers (see Fig. S18)

B.10 More Comparisons to Previous Methods

In this section, we add further comparisons of *NeLU3D* against previous methods: ZNCC [4, 5], LookUp3D, and TurboSL [6]. We analyze depth and disparity error metrics for a few objects and a few different patterns: black remote (Tab. S1), 3D printed house (Tab. S2), milled pawn (Tab. S3), black keyboard (Tab. S4), 3D printed radio (Tab. S5), and color calibration board (Tab. S6). In addition to these tables reporting error metrics, we also display visualizations of the rendered point clouds and meshes and the respective error maps. Once again, we report the depth error (the colormap goes from -5mm to 5mm) and disparity error (the colormap goes from -10px to 10px) for several objects and different pattern combinations: black keyboard (Fig. S19), 3D printed radio (Fig. S20), monkey statue (Fig. S21), mug (Fig. S22), milled rook (Fig. S23), paper roll (Fig. S24), bunny (Fig. S25), wipes (Fig. S26), and 3D printed chair (Fig. S27). The errors we report here are done against dense SL reconstruction using 44 Gray codes (11 horizontal patterns and their inverse, 11 vertical patterns and their inverse) and ZNCC. Additionally, as described in Sec. A.3, we manually remove the background for keyboard (Tab. S4 and Fig. S19), 3D printed radio (Tab. S5 and Fig. S20), 3D printed chair (Fig. S27), black remote (Tab. S1), and 3D printed house (Tab. S2) so that the evaluation does not get skewed by background points. We also highlight the strengths of TurboSL [6] in Fig. S28. TurboSL is a neural inverse structured light method capable of extracting accurate surface and normals with as few as three monochromatic patterns. In their original paper, Mirdehghan *et al.* showcase excellent results with A La Carte patterns [5] and Micro Phase Shifting patterns [1]. We validate in Fig. S28 with the 3D printed teapot the reconstruction quality of TurboSL when paired with these patterns.

We also compare, side-by-side, the reconstructions using LookUp3D and *NeLU3D* for dynamic scenes using an analog projector. In Fig. S29, we highlight a dynamic scene captured with the Chronos camera at 4.9ms exposure and the analog projector. In Fig. S30, we highlight a dynamic scene captured with the same Chronos camera but now at 1.0ms exposure. LookUp3D, since it is a per-pixel method, outputs results covered by noisy outliers, which usually require ad-hoc procedures and are hard to remove (for example, in their original implementation, Pereira *et al.* [10] use a residual threshold to remove points). The inductive bias of the signed distance function (SDF) effectively rejects outliers and the final reconstructions have less noise in *NeLU3D* when compared to LookUp3D. In TurboSL, Mirdehghan *et al.* report the same effect of using the SDF.

Table S1: Error Analysis: remote. We compare ZNCC [4,5], LookUp3D, TurboSL [6], and our proposed method *NeLU3D*. 3 Misc. is one sinusoidal pattern, one triangular pattern, and one random pattern. 4 Misc. we add another sinusoidal pattern, but phaseshifted with the first one. 5 Misc. we add a ramp pattern. We report mean absolute error (MAE), median absolute error (MedAE), and root mean squared error (RMSE) for both depth (in millimeters) and disparity (in pixels).

		ZNCC		LookUp3D		TurboSL		*NeLU3D*	
pattern	metric	depth	disp	depth	disp	depth	disp	depth	disp
3 Hilbert	MAE	91.42	385.88	17.68	53.33	31.38	105.60	0.35	1.15
	MedAE	90.92	381.97	2.41	8.33	26.40	96.35	0.17	0.59
	RMSE	93.28	395.75	39.78	107.33	35.71	119.45	3.25	7.43
3 Random	MAE	> 100	580.60	17.47	52.39	26.65	87.38	0.28	0.97
	MedAE	98.10	501.47	0.56	1.95	28.79	95.72	0.13	0.44
	RMSE	> 100	710.72	40.14	109.11	27.82	91.03	1.36	5.21
3 Misc.	MAE	> 100	558.82	18.79	55.13	53.55	167.21	0.28	0.97
	MedAE	> 100	507.10	0.44	1.51	56.60	173.65	0.13	0.44
	RMSE	> 100	683.39	43.54	118.48	55.53	172.42	2.38	9.90
4 Misc.	MAE	> 100	584.16	16.43	49.08	39.46	126.53	0.26	0.90
	MedAE	> 100	523.10	0.45	1.54	41.14	132.92	0.11	0.39
	RMSE	> 100	714.73	38.78	105.30	44.38	144.03	2.53	8.31
5 Misc.	MAE	75.54	249.67	14.92	47.38	17.30	59.19	0.29	0.96
	MedAE	57.99	193.82	0.87	3.02	18.74	63.88	0.12	0.42
	RMSE	> 100	326.55	31.34	84.62	22.95	84.14	3.52	10.30
11 Gray	MAE	0.13	0.45	34.48	107.21	0.24	0.84	0.25	0.81
	MedAE	0.12	0.40	23.14	80.96	0.18	0.61	0.09	0.32
	RMSE	0.52	1.47	50.55	142.63	0.67	2.35	3.79	8.54

Table S2: Error Analysis: house. We compare ZNCC [4, 5], LookUp3D, TurboSL [6], and our proposed method *NeLU3D*. 3 Misc. is one sinusoidal pattern, one triangular pattern, and one random pattern. 4 Misc. we add another sinusoidal pattern, but phaseshifted with the first one. 5 Misc. we add a ramp pattern. We report mean absolute error (MAE), median absolute error (MedAE), and root mean squared error (RMSE) for both depth (in millimeters) and disparity (in pixels).

		ZNCC		LookUp3D		TurboSL		*NeLU3D*	
pattern	metric	depth	disp	depth	disp	depth	disp	depth	disp
3 Hilbert	MAE	> 100	529.29	2.49	7.55	17.36	58.05	0.89	2.94
	MedAE	> 100	495.81	0.60	2.03	17.49	59.10	0.31	1.07
	RMSE	> 100	539.01	13.97	31.53	20.19	66.09	3.10	9.77
3 Random	MAE	> 100	522.51	11.24	31.41	47.33	147.50	0.80	2.66
	MedAE	> 100	459.42	0.44	1.51	43.24	136.88	0.31	1.04
	RMSE	> 100	628.40	36.31	96.40	50.67	157.40	3.11	9.88
3 Misc.	MAE	> 100	498.61	15.26	43.05	47.73	148.96	1.25	4.17
	MedAE	> 100	497.78	0.41	1.39	56.18	176.92	0.34	1.16
	RMSE	> 100	599.22	40.73	110.21	50.08	156.07	4.15	13.35
4 Misc.	MAE	> 100	545.55	15.61	45.32	41.26	129.89	1.00	3.34
	MedAE	> 100	544.44	0.41	1.42	41.06	131.16	0.31	1.05
	RMSE	> 100	642.24	38.46	106.85	44.94	140.98	3.82	12.17
5 Misc.	MAE	79.07	270.64	7.14	22.57	26.49	85.47	1.01	3.39
	MedAE	53.82	196.10	0.42	1.45	23.57	76.82	0.32	1.09
	RMSE	> 100	364.93	18.28	50.62	30.67	98.31	3.69	11.84
11 Gray	MAE	0.28	0.95	42.92	126.17	1.36	4.69	0.42	1.43
	MedAE	0.20	0.68	22.52	74.79	0.31	1.07	0.27	0.93
	RMSE	0.99	2.79	62.89	177.23	4.31	14.77	1.32	3.65

Table S3: Error Analysis: pawn. We compare ZNCC [4,5], LookUp3D, TurboSL [6], and our proposed method *NeLU3D*. 3 Misc. is one sinusoidal pattern, one triangular pattern, and one random pattern. 4 Misc. we add another sinusoidal pattern, but phaseshifted with the first one. 5 Misc. we add a ramp pattern. We report mean absolute error (MAE), median absolute error (MedAE), and root mean squared error (RMSE) for both depth (in millimeters) and disparity (in pixels).

		ZNCC		LookUp3D		TurboSL		*NeLU3D*	
pattern	metric	depth	disp	depth	disp	depth	disp	depth	disp
3 Hilbert	MAE	> 100	822.13	1.03	2.78	77.44	246.34	0.53	1.47
	MedAE	> 100	808.88	0.57	1.60	73.40	235.29	0.46	1.29
	RMSE	> 100	827.61	5.78	9.43	82.14	252.43	1.73	3.17
3 Random	MAE	> 100	513.42	1.37	3.79	31.73	96.51	0.60	1.68
	MedAE	> 100	485.98	0.46	1.29	27.59	79.21	0.50	1.41
	RMSE	> 100	588.13	7.29	16.18	38.56	122.12	1.39	4.71
3 Misc.	MAE	> 100	496.96	2.43	6.54	22.42	65.05	0.60	1.67
	MedAE	> 100	497.06	0.47	1.31	20.93	60.42	0.50	1.41
	RMSE	> 100	572.65	11.71	28.42	23.94	70.90	1.16	3.76
4 Misc.	MAE	> 100	481.08	2.08	5.77	25.34	76.10	0.57	1.59
	MedAE	> 100	503.10	0.46	1.30	23.61	68.03	0.52	1.44
	RMSE	> 100	558.49	10.19	25.69	33.10	104.26	1.01	3.34
5 Misc.	MAE	88.81	259.78	2.51	6.85	33.49	100.31	0.54	1.49
	MedAE	78.23	198.12	0.47	1.32	36.55	109.78	0.50	1.40
	RMSE	> 100	353.54	9.36	22.32	37.75	116.26	0.80	2.80
11 Gray	MAE	0.15	0.42	31.32	88.14	1.02	3.07	0.58	1.62
	MedAE	0.07	0.20	27.16	77.92	0.15	0.41	0.51	1.42
	RMSE	0.54	1.47	40.25	110.58	7.36	26.45	1.03	3.37

Table S4: Error Analysis: keyboard. We compare ZNCC [4, 5], LookUp3D, TurboSL [6], and our proposed method *NeLU3D*. 3 Misc. is one sinusoidal pattern, one triangular pattern, and one random pattern. 4 Misc. we add another sinusoidal pattern, but phaseshifted with the first one. 5 Misc. we add a ramp pattern. We report mean absolute error (MAE), median absolute error (MedAE), and root mean squared error (RMSE) for both depth (in millimeters) and disparity (in pixels).

		ZNCC		LookUp3D		TurboSL		*NeLU3D*	
pattern	metric	depth	disp	depth	disp	depth	disp	depth	disp
3 Hilbert	MAE	> 100	710.80	21.67	59.46	48.48	160.71	0.90	2.68
	MedAE	> 100	675.57	8.27	24.65	42.72	132.58	0.30	0.89
	RMSE	> 100	762.22	41.48	98.50	58.63	201.93	2.61	7.26
3 Random	MAE	> 100	519.88	20.97	57.66	35.85	111.44	0.67	2.00
	MedAE	> 100	469.29	2.66	7.72	33.63	102.93	0.29	0.86
	RMSE	> 100	619.93	41.53	99.23	45.14	141.25	2.95	9.98
3 Misc.	MAE	> 100	546.67	23.13	62.75	34.91	101.41	0.62	1.87
	MedAE	> 100	519.69	1.66	4.87	25.56	77.11	0.27	0.81
	RMSE	> 100	644.38	44.89	107.64	46.04	132.54	2.63	8.65
4 Misc.	MAE	> 100	552.59	24.03	64.81	31.96	93.21	0.44	1.26
	MedAE	> 100	535.90	1.91	5.59	25.97	76.37	0.23	0.68
	RMSE	> 100	653.06	45.27	108.06	42.34	122.87	4.83	10.70
5 Misc.	MAE	98.63	303.88	28.33	79.88	23.72	70.38	0.33	0.99
	MedAE	72.59	236.86	25.64	79.19	19.17	56.12	0.23	0.69
	RMSE	> 100	402.17	44.92	111.92	32.19	95.59	1.57	5.70
11 Gray	MAE	0.39	1.14	33.09	95.07	0.97	2.90	0.28	0.84
	MedAE	0.34	1.02	26.20	78.74	0.37	1.09	0.18	0.53
	RMSE	0.83	2.37	46.57	120.40	3.72	11.49	1.30	4.73

Table S5: Error Analysis: radio. We compare ZNCC [4,5], LookUp3D, TurboSL [6], and our proposed method *NeLU3D*. 3 Misc. is one sinusoidal pattern, one triangular pattern, and one random pattern. 4 Misc. we add another sinusoidal pattern, but phaseshifted with the first one. 5 Misc. we add a ramp pattern. We report mean absolute error (MAE), median absolute error (MedAE), and root mean squared error (RMSE) for both depth (in millimeters) and disparity (in pixels).

		ZNCC		LookUp3D		TurboSL		*NeLU3D*	
pattern	metric	depth	disp	depth	disp	depth	disp	depth	disp
3 Hilbert	MAE	> 100	865.63	3.10	7.85	37.89	111.66	1.02	2.57
	MedAE	> 100	883.02	0.84	2.25	35.23	100.20	0.61	1.64
	RMSE	> 100	892.81	14.43	27.29	47.11	140.70	5.89	11.91
3 Random	MAE	> 100	505.93	3.87	9.97	18.10	51.07	0.97	2.43
	MedAE	> 100	488.18	0.63	1.71	8.52	23.33	0.58	1.57
	RMSE	> 100	590.10	17.34	38.02	27.79	78.59	6.01	12.36
3 Misc.	MAE	> 100	540.88	5.03	12.64	13.24	36.44	1.00	2.52
	MedAE	> 100	522.21	0.61	1.63	6.10	16.49	0.61	1.63
	RMSE	> 100	630.22	18.64	40.55	20.30	55.25	5.54	11.07
4 Misc.	MAE	> 100	526.95	6.06	15.65	14.12	38.84	0.96	2.40
	MedAE	> 100	519.52	0.58	1.57	9.82	26.64	0.60	1.63
	RMSE	> 100	610.60	20.90	48.72	20.66	56.26	6.15	13.35
5 Misc.	MAE	> 100	315.47	7.55	19.93	11.93	32.81	0.91	2.29
	MedAE	82.17	248.01	0.65	1.75	5.97	16.11	0.62	1.66
	RMSE	> 100	420.96	18.91	44.02	18.61	50.38	5.56	11.43
11 Gray	MAE	0.23	0.58	36.96	100.30	1.01	2.60	1.11	2.90
	MedAE	0.14	0.37	34.29	93.43	0.17	0.46	0.59	1.58
	RMSE	2.97	5.74	45.12	119.84	7.99	18.50	10.20	26.55

Table S6: Error Analysis: color calibration board. We compare ZNCC [4, 5], LookUp3D, TurboSL [6], and our proposed method *NeLU3D*. 3 Misc. is one sinusoidal pattern, one triangular pattern, and one random pattern. 4 Misc. we add another sinusoidal pattern, but phaseshifted with the first one. 5 Misc. we add a ramp pattern. We report mean absolute error (MAE), median absolute error (MedAE), and root mean squared error (RMSE) for both depth (in millimeters) and disparity (in pixels).

		ZNCC		LookUp3D		TurboSL		*NeLU3D*	
pattern	metric	depth	disp	depth	disp	depth	disp	depth	disp
3 Hilbert	MAE	> 100	843.72	9.29	24.83	54.12	147.58	0.35	0.86
	MedAE	> 100	842.86	1.15	2.85	57.28	152.80	0.25	0.62
	RMSE	> 100	868.74	25.07	66.91	61.80	168.33	2.17	1.93
3 Random	MAE	> 100	505.48	9.01	24.76	53.34	143.82	0.36	0.87
	MedAE	> 100	483.88	0.58	1.44	54.21	146.57	0.25	0.62
	RMSE	> 100	588.93	26.47	72.12	58.80	158.19	2.13	1.38
3 Misc.	MAE	> 100	530.26	11.87	32.84	47.59	127.95	1.93	5.00
	MedAE	> 100	517.36	0.47	1.17	46.83	121.85	0.25	0.62
	RMSE	> 100	615.90	31.26	86.65	57.14	154.02	12.61	33.25
4 Misc.	MAE	> 100	521.99	11.95	32.89	54.03	146.66	0.25	0.61
	MedAE	> 100	515.64	0.45	1.11	57.15	155.14	0.21	0.51
	RMSE	> 100	605.91	30.99	85.38	61.86	168.68	2.08	0.82
5 Misc.	MAE	> 100	314.76	13.93	36.21	55.04	149.38	0.26	0.62
	MedAE	86.87	247.93	0.65	1.60	57.92	157.14	0.21	0.51
	RMSE	> 100	418.63	28.01	72.91	62.01	168.91	2.15	1.70
11 Gray	MAE	1.13	2.75	46.02	123.44	4.16	10.51	0.26	0.62
	MedAE	1.09	2.71	42.25	104.80	1.02	2.52	0.21	0.53
	RMSE	1.35	3.54	56.70	156.10	16.69	50.67	2.10	1.08

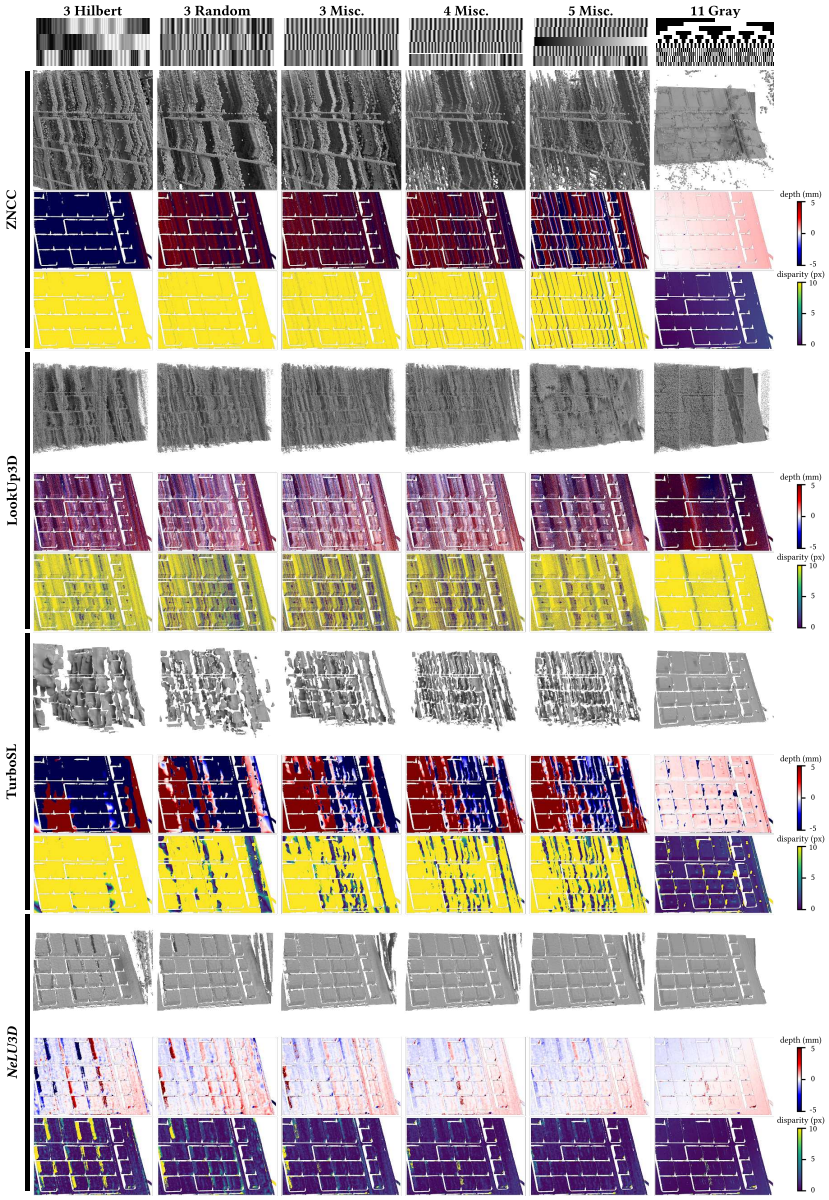


Fig. S19: Quantitative Comparison: keyboard. We compare ZNCC [4, 5], LookUp3D, TurboSL [6], and *NeLU3D* for different patterns. We report error for depth (in millimeters) and disparity (in pixels) measured against dense SL.

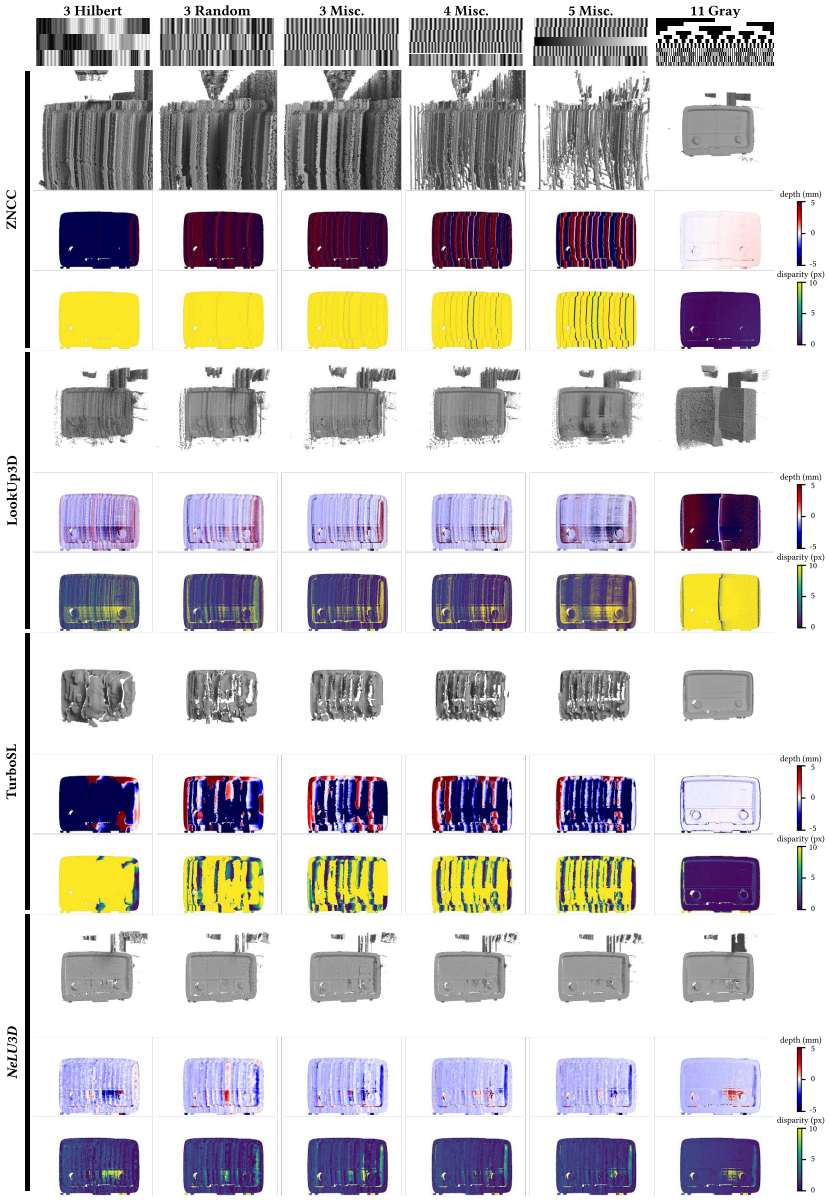


Fig. S20: Quantitative Comparison: radio. We compare ZNCC [4, 5], LookUp3D, TurboSL [6], and *NeLU3D* for different patterns. We report error for depth (in millimeters) and disparity (in pixels) measured against dense SL.

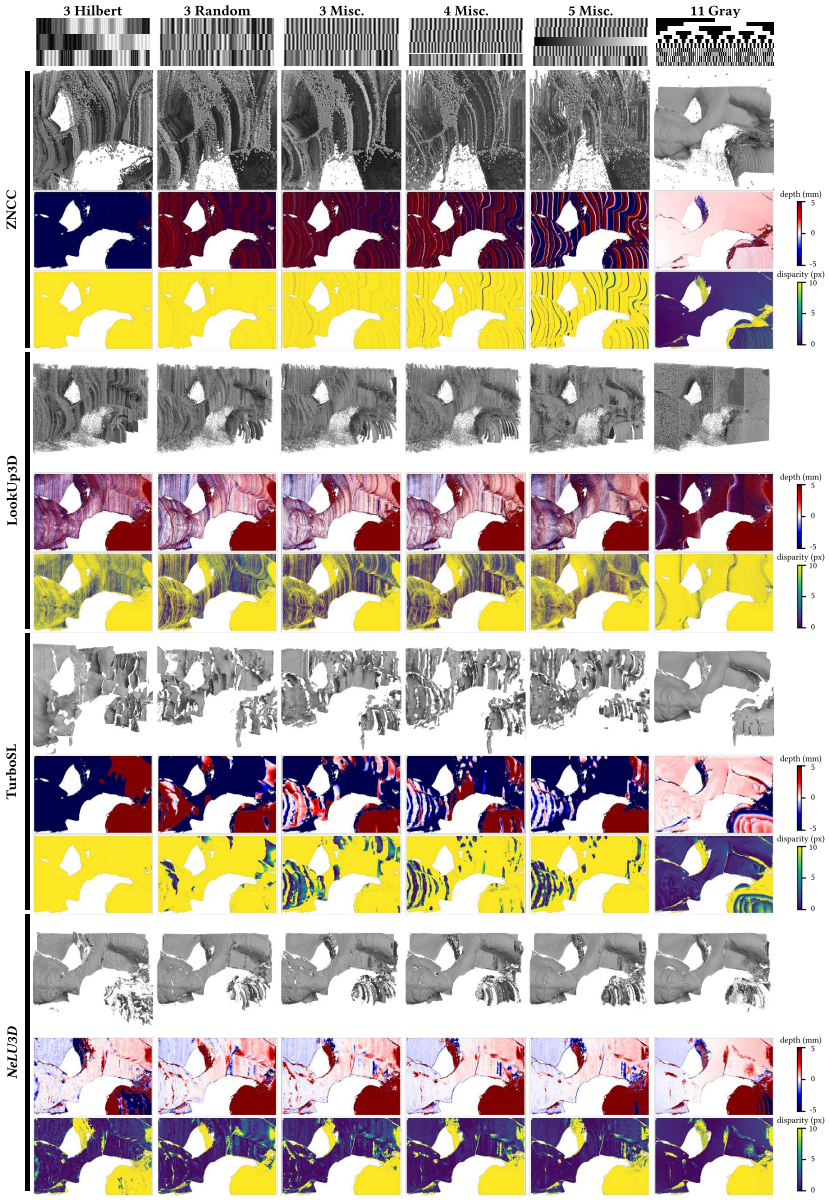


Fig. S21: Quantitative Comparison: monkey. We compare ZNCC [4, 5], LookUp3D, TurboSL [6], and *NeLU3D* for different patterns. We report error for depth (in millimeters) and disparity (in pixels) measured against dense SL.

Fig. S22: Quantitative Comparison: mug. We compare ZNCC [4, 5], LookUp3D, TurboSL [6], and *NeLU3D* for different patterns. We report error for depth (in millimeters) and disparity (in pixels) measured against dense SL.

Fig. S23: Quantitative Comparison: rook. We compare ZNCC [4, 5], LookUp3D, TurboSL [6], and *NeLU3D* for different patterns. We report error for depth (in millimeters) and disparity (in pixels) measured against dense SL.

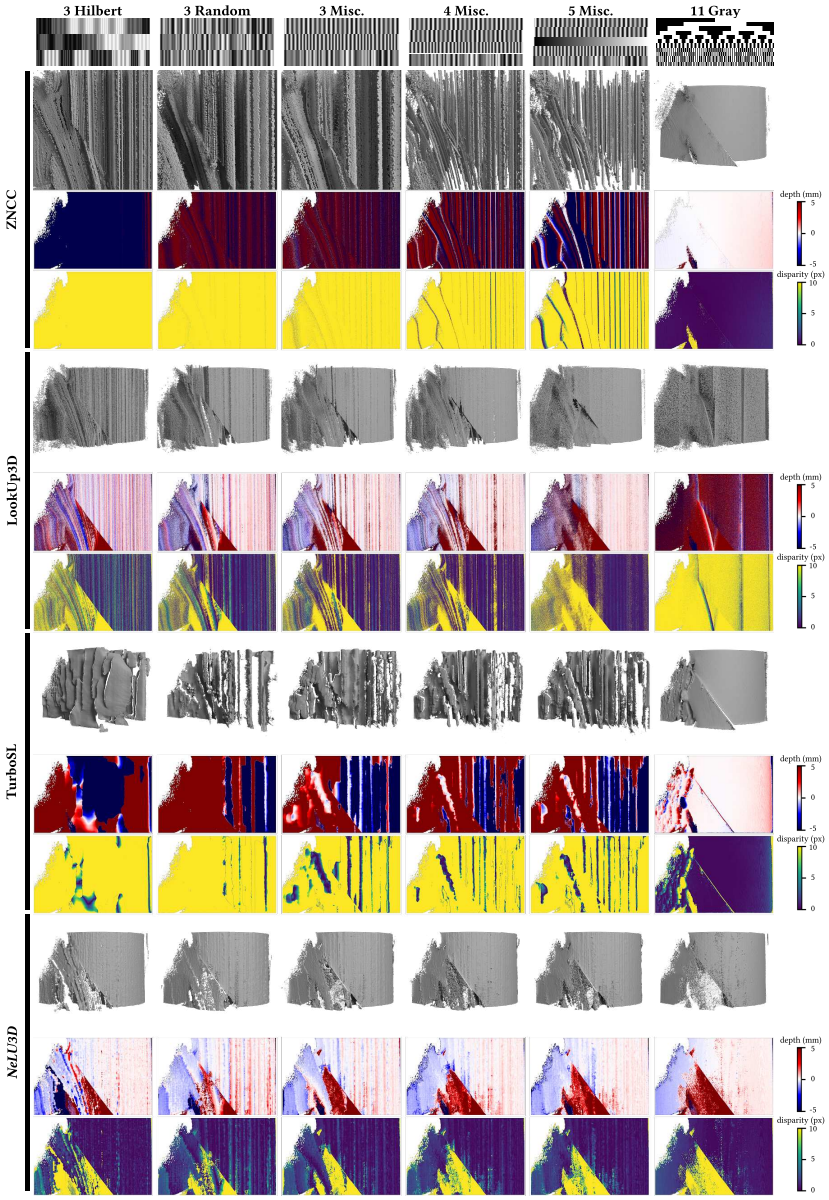


Fig. S24: Quantitative Comparison: paper roll. We compare ZNCC [4, 5], LookUp3D, TurboSL [6], and *NeLU3D* for different patterns. We report error for depth (in millimeters) and disparity (in pixels) measured against dense SL.

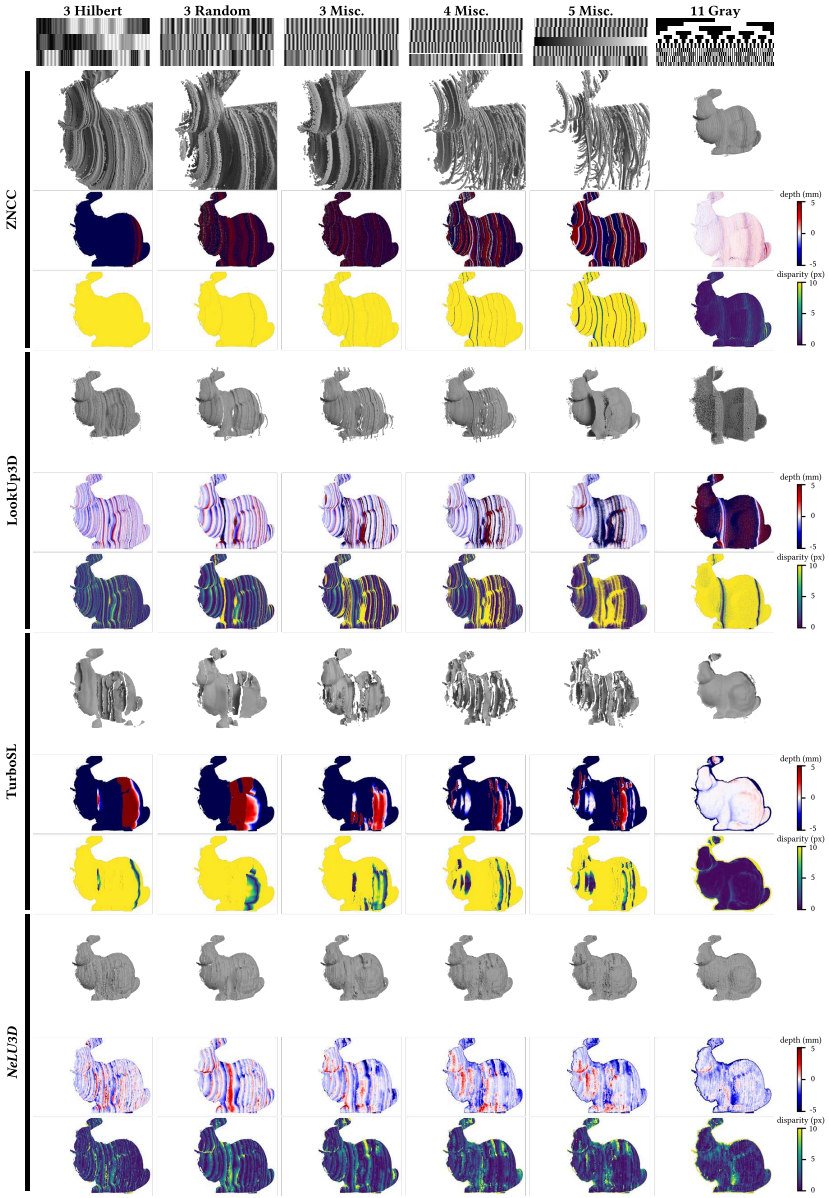


Fig. S25: Quantitative Comparison: bunny. We compare ZNCC [4,5], LookUp3D, TurboSL [6], and *NeLU3D* for different patterns. We report error for depth (in millimeters) and disparity (in pixels) measured against dense SL.

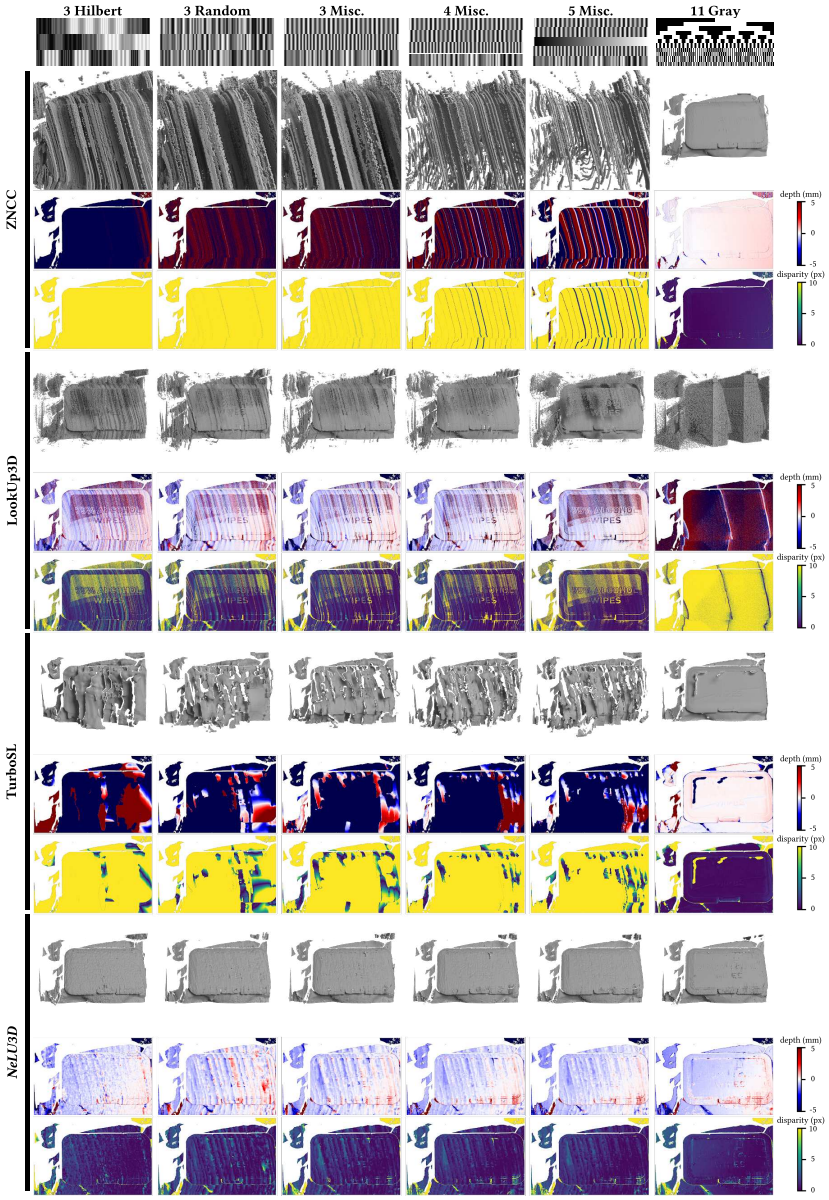


Fig. S26: Quantitative Comparison: wipes. We compare ZNCC [4,5], LookUp3D, TurboSL [6], and *NeLU3D* for different patterns. We report error for depth (in millimeters) and disparity (in pixels) measured against dense SL.

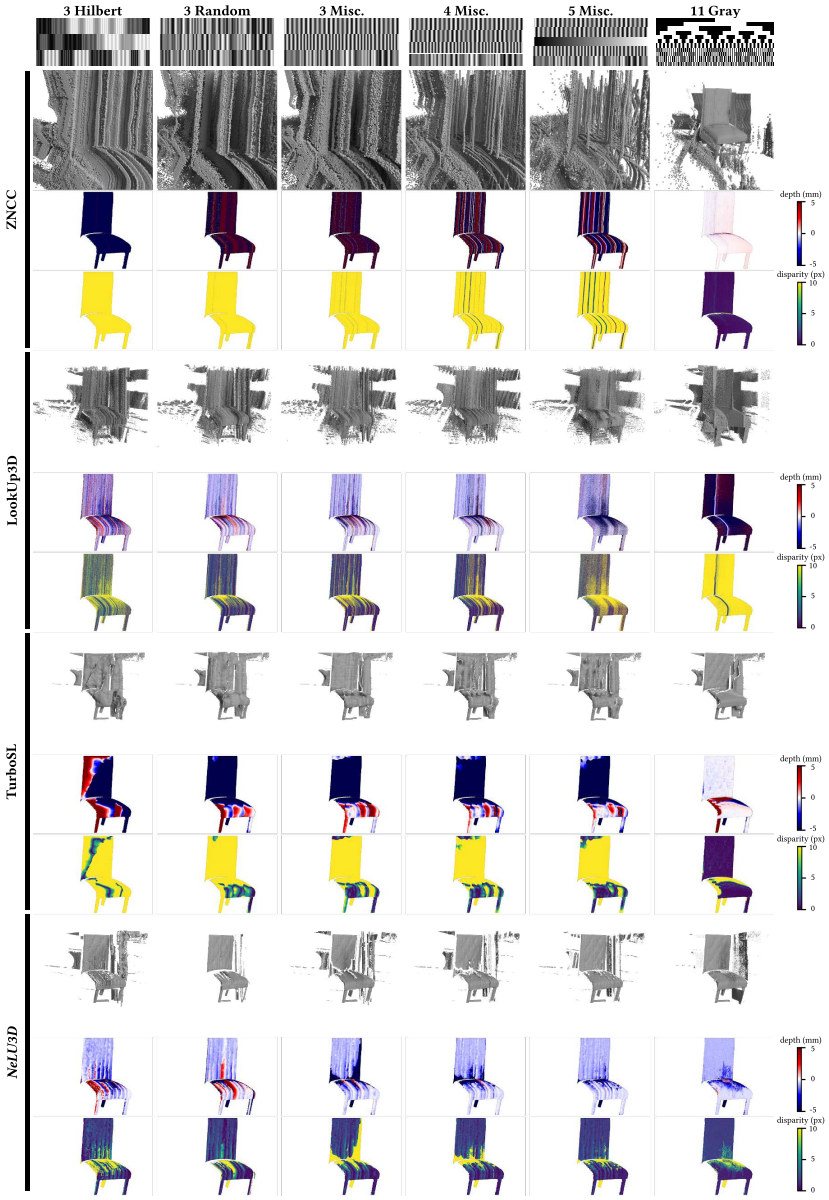


Fig. S27: Quantitative Comparison: chair. We compare ZNCC [4, 5], LookUp3D, TurboSL [6], and *NeLU3D* for different patterns. We report error for depth (in millimeters) and disparity (in pixels) measured against dense SL.

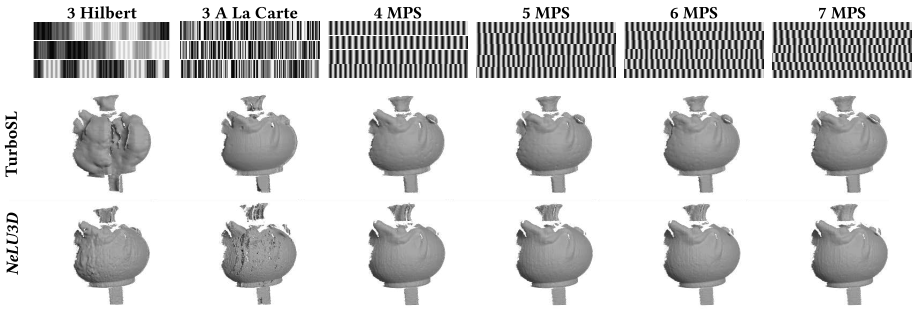


Fig. S28: TurboSL Performance with Optimal Patterns. We would like to highlight the strengths of TurboSL, on which we base our proposed method. TurboSL

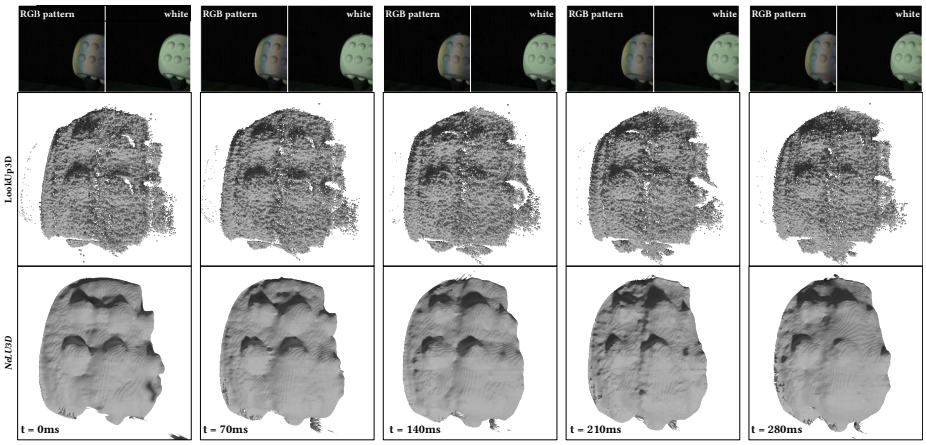


Fig. S29: Dynamic Comparison A side-by-side comparison of LookUp3D and *NeLU3D* on a dynamic scene: squeezing a silicone cube. We captured this scene with the Chronos trichromatic camera at 200fps. We also use the analog projector as shown in LookUp3D with a helical RGB pattern developed on film.

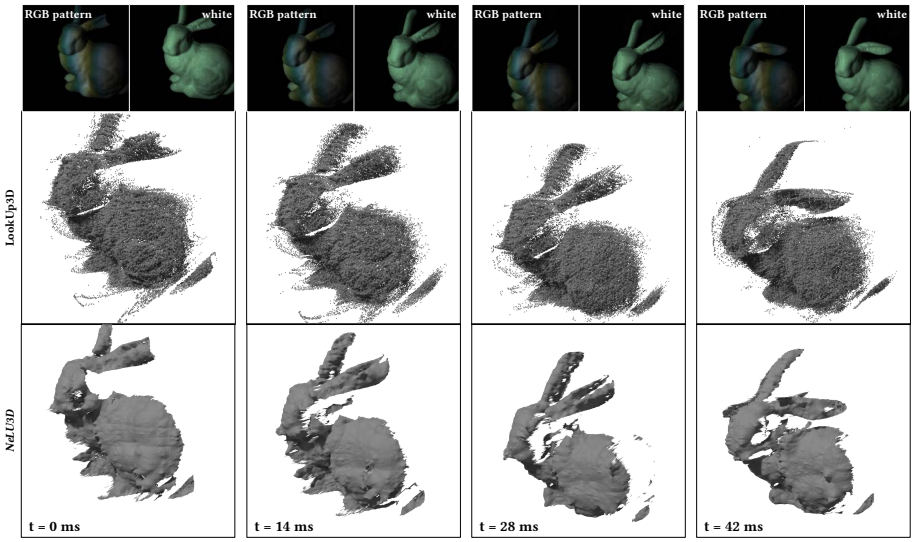


Fig. S30: Dynamic Comparison A side-by-side comparison of LookUp3D and *NeLU3D* on a dynamic scene: bunny falling. We captured this scene with the Chronos trichromatic camera at 1000fps. We also use the analog projector as shown in LookUp3D with a helical RGB pattern developed on film.

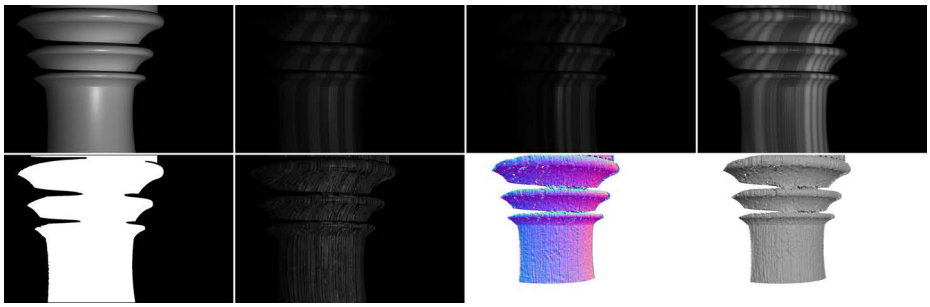


Fig. S31: Additional results: milled rook. Top row: captured images from camera under uniform lighting and three Hilbert patterns from DLP projector. **Bottom row:** binary mask, fitted residual/ambient term (scaled $\times 10$), normals, and rendered 3D shape.

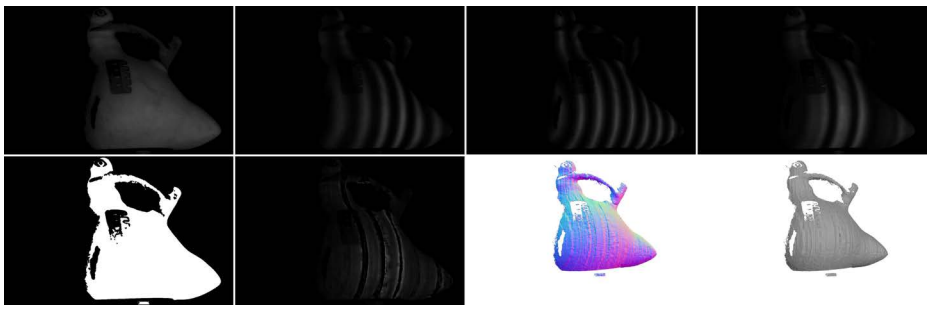


Fig. S32: Additional results: 3D printed bird. Top row: captured images from camera under uniform lighting and three patterns (sinusoidal, triangular, and random) from DLP projector. **Bottom row:** binary mask, fitted residual/ambient term (scaled $\times 10$), normals, and rendered 3D shape.

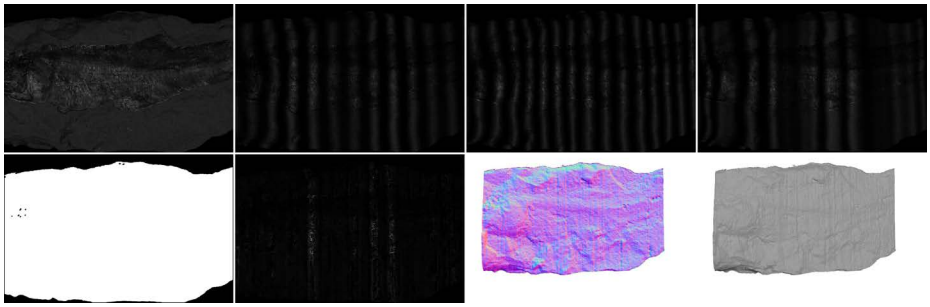


Fig. S33: Additional results: fossil replica. Top row: captured images from camera under uniform lighting and three patterns (sinusoidal, triangular, and random) from DLP projector. **Bottom row:** binary mask, fitted residual/ambient term (scaled $\times 10$), normals, and rendered 3D shape.

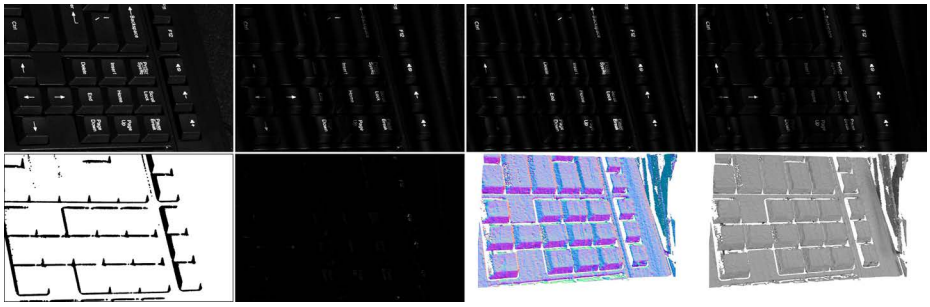


Fig. S34: Additional results: keyboard. Top row: captured images from camera under uniform lighting and three patterns (sinusoidal, triangular, and random) from DLP projector (scaled $\times 10$). **Bottom row:** binary mask, fitted residual/ambient term (scaled $\times 10$), normals, and rendered 3D shape.

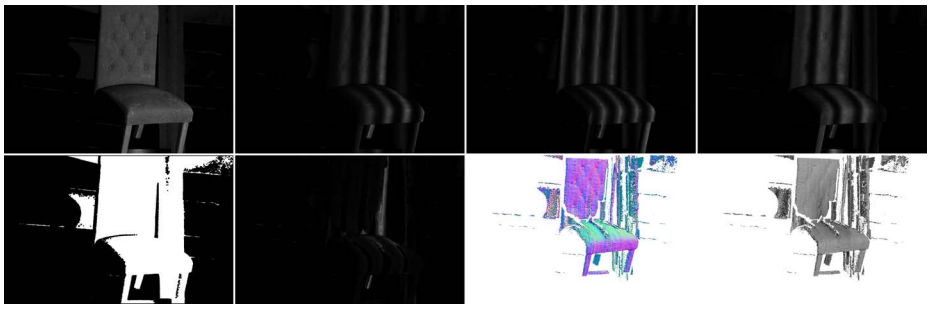


Fig. S35: Additional results: 3D printed chair. Top row: captured images from camera under uniform lighting and three patterns (sinusoidal, triangular, and random) from DLP projector (scaled $\times 5$). **Bottom row:** binary mask, fitted residual/ambient term (scaled $\times 10$), normals, and rendered 3D shape.

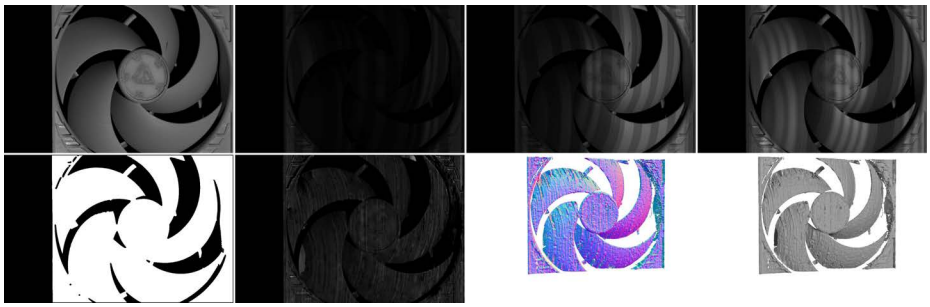


Fig. S36: Additional results: fan. Top row: captured images from camera under uniform lighting and three Hilbert patterns [2] from DLP projector. **Bottom row:** binary mask, fitted residual/ambient term (scaled $\times 10$), normals, and rendered 3D shape.

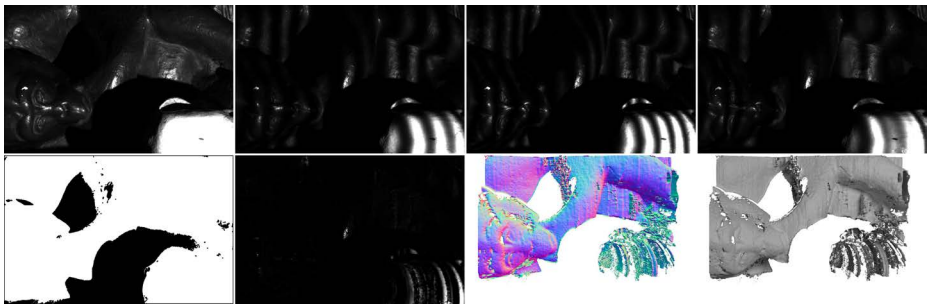


Fig. S37: Additional results: monkey statue. Top row: captured images from camera under uniform lighting and three patterns (sinusoidal, triangular, and random) from DLP projector (scaled $\times 5$). **Bottom row:** binary mask, fitted residual/ambient term (scaled $\times 10$), normals, and rendered 3D shape. Note that the white skull was outside the calibrated volume and therefore was incorrectly reconstructed.

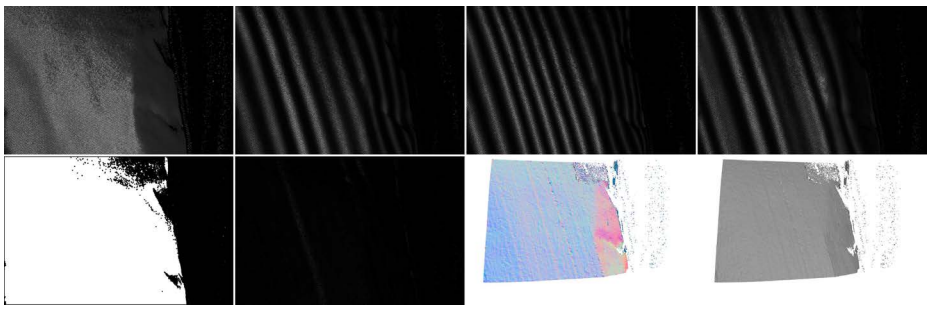


Fig. S38: Additional results: first aid kit. Top row: captured images from camera under uniform lighting and three patterns (sinusoidal, triangular, and random) from DLP projector (scaled $\times 10$). **Bottom row:** binary mask, fitted residual/ambient term (scaled $\times 10$), normals, and rendered 3D shape.

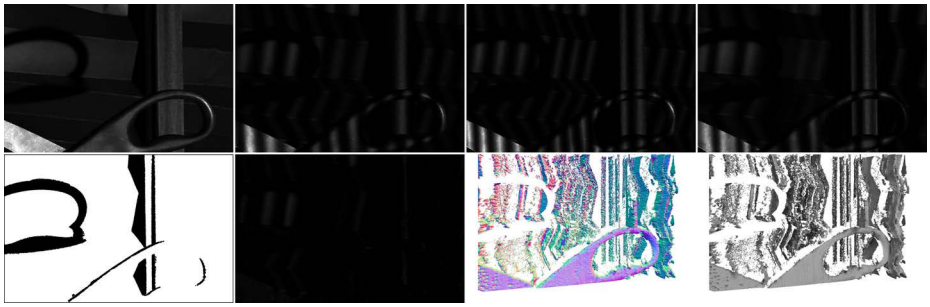


Fig. S39: Additional results: scissors. Top row: captured images from camera under uniform lighting and three patterns (sinusoidal, triangular, and random) from DLP projector (scaled $\times 10$). **Bottom row:** binary mask, fitted residual/ambient term (scaled $\times 10$), normals, and rendered 3D shape.

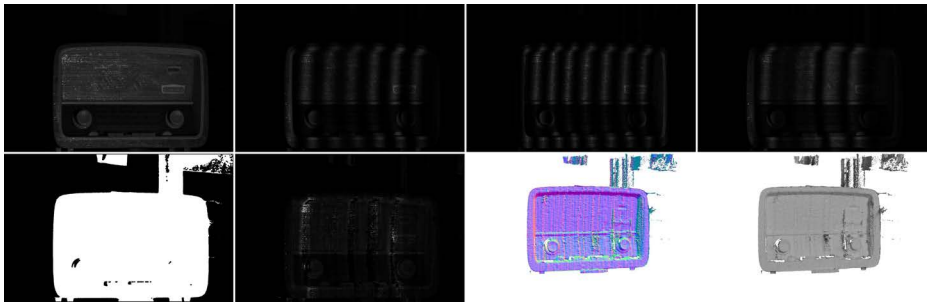


Fig. S40: Additional results: 3D printed radio. Top row: captured images from camera under uniform lighting and three patterns (sinusoidal, triangular, and random) from DLP projector. **Bottom row:** binary mask, fitted residual/ambient term (scaled $\times 10$), normals, and rendered 3D shape.

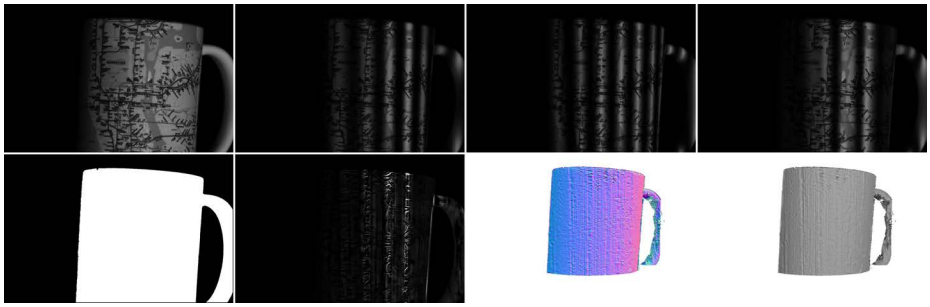


Fig. S41: Additional results: mug. Top row: captured images from camera under uniform lighting and three patterns (sinusoidal, triangular, and random) from DLP projector. **Bottom row:** binary mask, fitted residual/ambient term (scaled $\times 10$), normals, and rendered 3D shape.

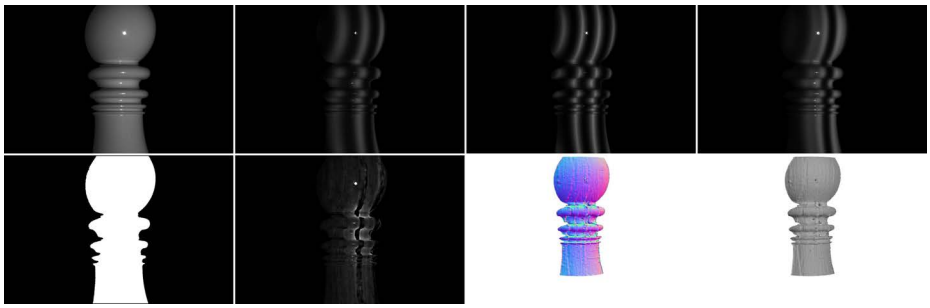


Fig. S42: Additional results: milled pawn. Top row: captured images from camera under uniform lighting and three patterns (sinusoidal, triangular, and random) from DLP projector. **Bottom row:** binary mask, fitted residual/ambient term (scaled $\times 10$), normals, and rendered 3D shape.

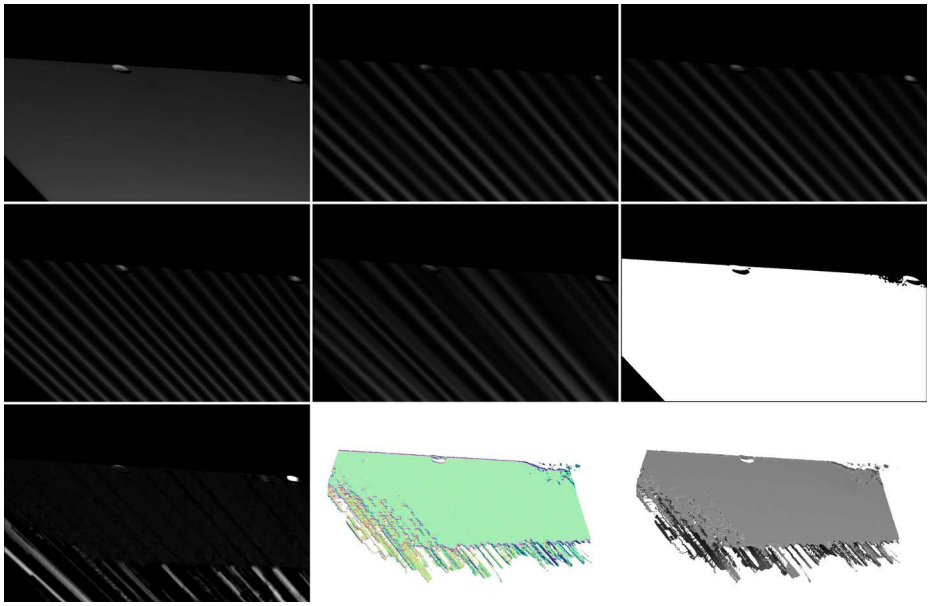


Fig. S43: Additional results: flat, white plane. Top row: captured images from camera under uniform lighting and three patterns (sinusoidal, triangular, and random) from DLP projector. **Bottom row:** binary mask, fitted residual/ambient term (scaled $\times 10$), normals, and rendered 3D shape. Note that the bottom part of the plane was outside the calibrated volume and therefore cannot be correctly reconstructed.

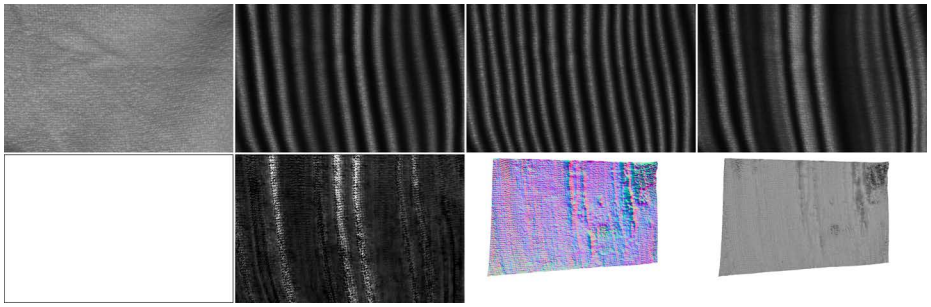


Fig. S44: Additional results: white cloth. Top row: captured images from camera under uniform lighting and three patterns (sinusoidal, triangular, and random) from DLP projector. **Bottom row:** binary mask, fitted residual/ambient term (scaled $\times 10$), normals, and rendered 3D shape.

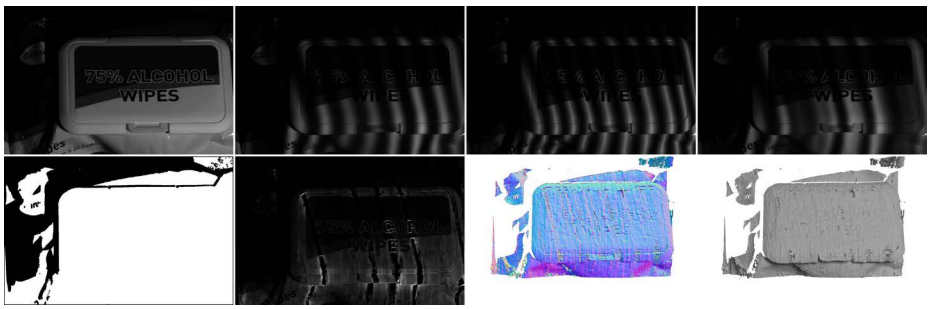


Fig. S45: Additional results: pack of alcohol wipes. Top row: captured images from camera under uniform lighting and three patterns (sinusoidal, triangular, and random) from DLP projector. **Bottom row:** binary mask, fitted residual/ambient term (scaled $\times 10$), normals, and rendered 3D shape.

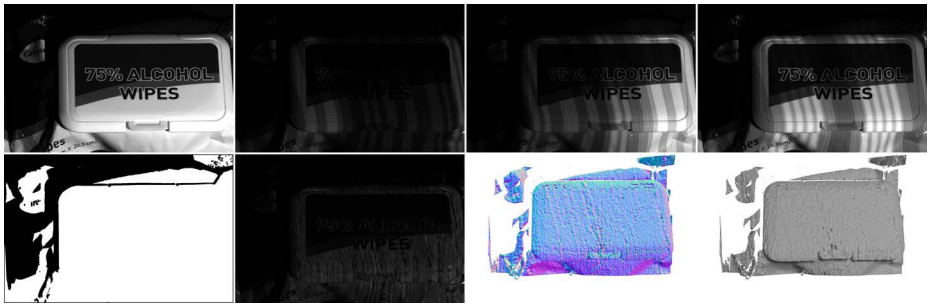


Fig. S46: Additional results: pack of alcohol wipes. Top row: captured images from camera under uniform lighting and three Hilbert patterns [2] from DLP projector (scaled $\times 2$). **Bottom row:** binary mask, fitted residual/ambient term (scaled $\times 10$), normals, and rendered 3D shape.

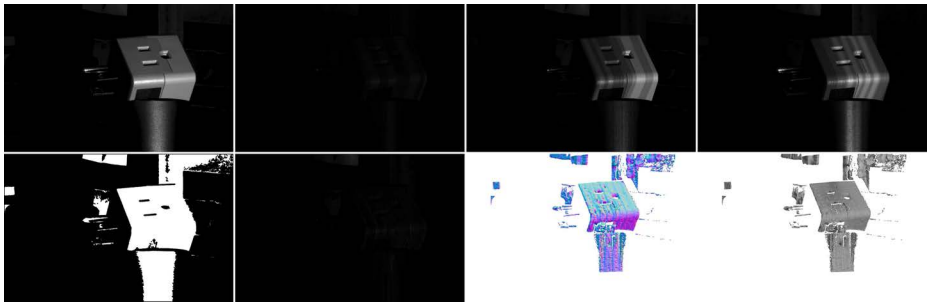


Fig. S47: Additional results: power adapter. Top row: captured images from camera under uniform lighting and three Hilbert patterns [2] from DLP projector (scaled $\times 5$). **Bottom row:** binary mask, fitted residual/ambient term (scaled $\times 10$), normals, and rendered 3D shape.

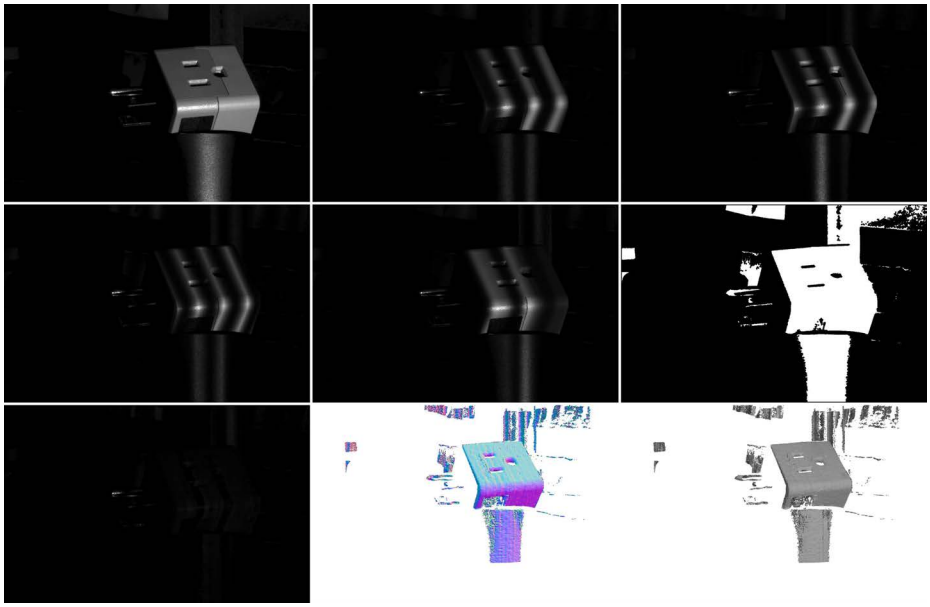


Fig. S48: Additional results: power adapter. Top row: captured images from camera under uniform lighting and three patterns (sinusoidal, triangular, and random) from DLP projector (scaled $\times 5$). **Bottom row:** binary mask, fitted residual/ambient term (scaled $\times 10$), normals, and rendered 3D shape.

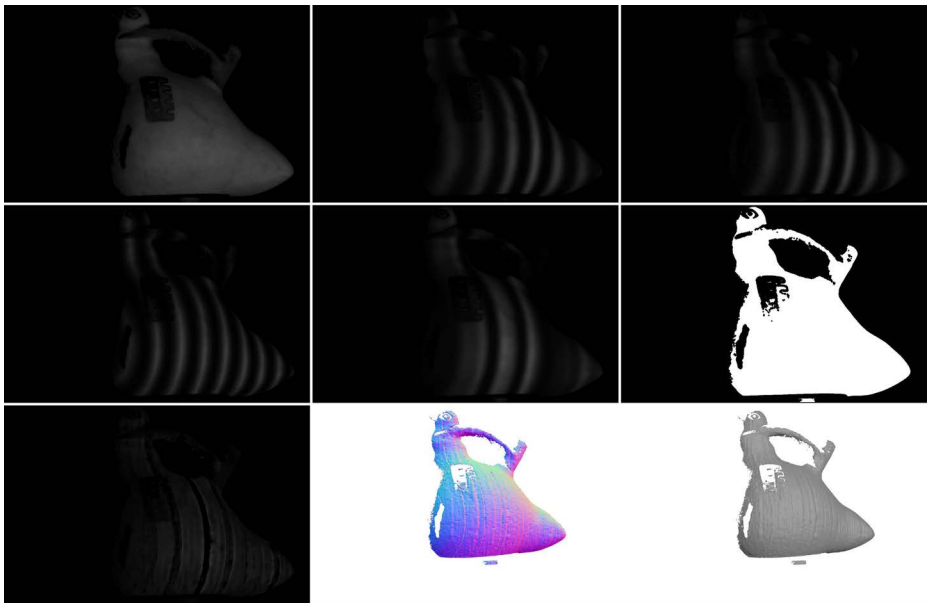


Fig. S49: Additional results: 3D printed bird four patterns. Top row: captured under uniform lighting and two phase-shifted sinusoidal patterns from DLP projector. **Middle row:** two additional pattern captures (one triangular, one random) and the binary mask for reconstruction. **Bottom row:** fitted residual/ambient term (scaled $\times 10$), normals, and rendered 3D shape.

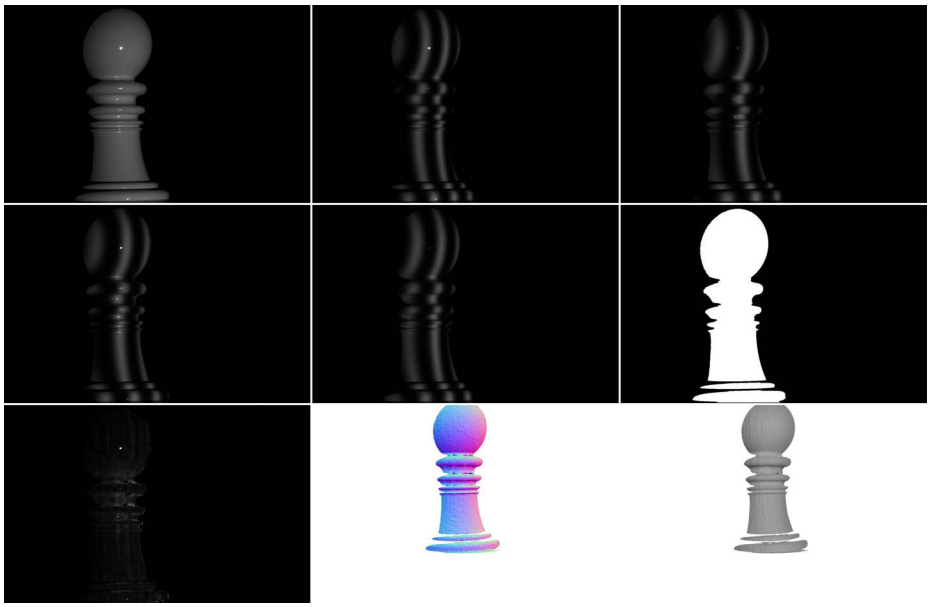


Fig. S50: Additional results: milled pawn with four patterns. Top row: captured under uniform lighting and two micro phase shift (frequency 64) patterns [1] from DLP projector. **Middle row:** two additional micro phase shift captures and the binary mask for reconstruction. **Bottom row:** fitted residual/ambient term (scaled $\times 10$), normals, and rendered 3D shape.

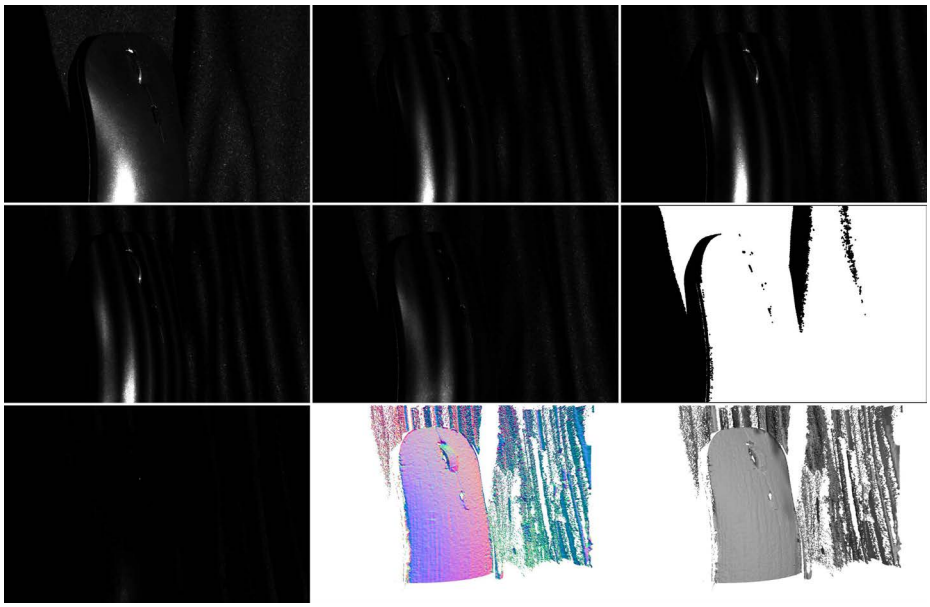


Fig. S51: Additional results: mouse with four patterns. Top row: captured under uniform lighting and two phase-shifted sinusoidal patterns from DLP projector (scaled $\times 10$). **Middle row:** two additional pattern captures (one triangular, one random, both scaled $\times 10$) and the binary mask for reconstruction. **Bottom row:** fitted residual/ambient term (scaled $\times 10$), normals, and rendered 3D shape.

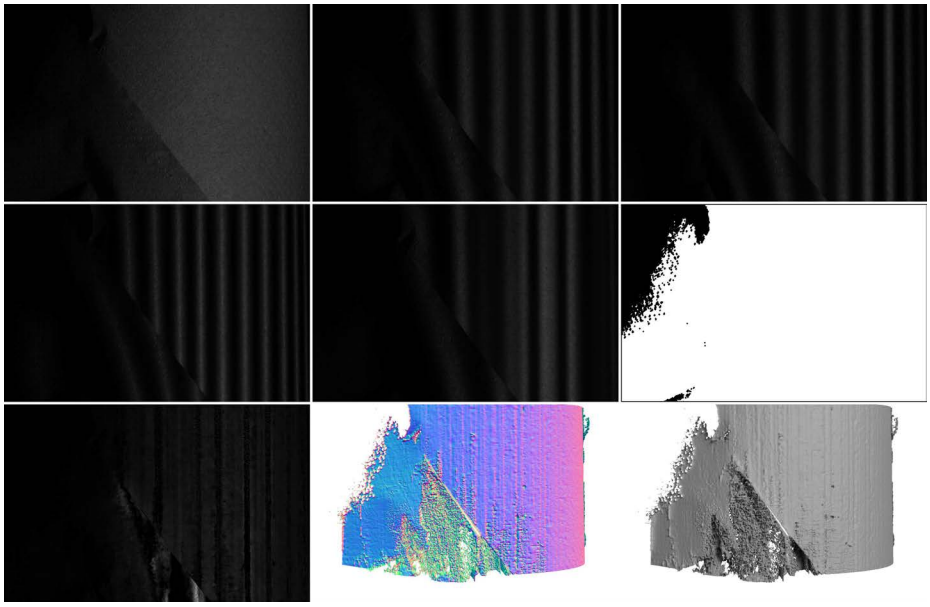


Fig. S52: Additional results: paper towel roll with four patterns. Top row: captured under uniform lighting and two phase-shifted sinusoidal patterns from DLP projector. **Middle row:** two additional pattern captures (one triangular, one random) and the binary mask for reconstruction. **Bottom row:** fitted residual/ambient term (scaled $\times 10$), normals, and rendered 3D shape. Note that the paper towel sheet on the left dropped outside the calibrated volume and therefore was incorrectly reconstructed.

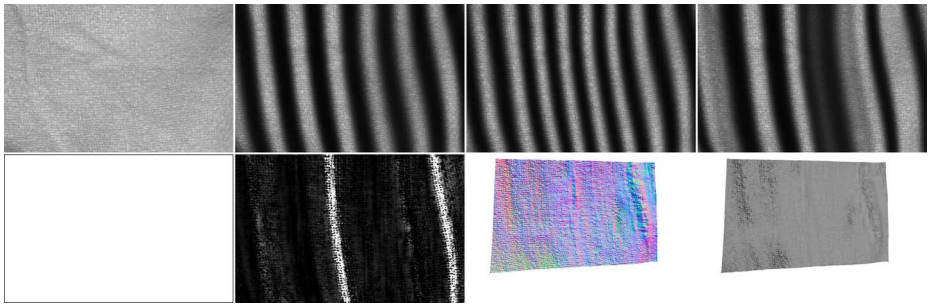


Fig. S53: Additional results: white cloth with cheap projector. Top row: captured images from camera under uniform lighting and three patterns (sinusoidal, triangular, and random) from LCD projector. **Bottom row:** binary mask, fitted residual/ambient term (scaled $\times 10$), normals, and rendered 3D shape.

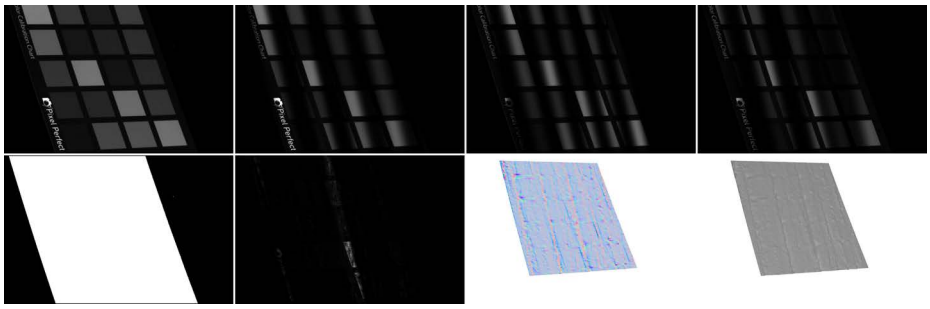


Fig. S54: Additional results: color calibration board with cheap projector. Top row: captured images from camera under uniform lighting and three patterns (sinusoidal, triangular, and random) from LCD projector (scaled ×2). **Bottom row:** binary mask, fitted residual/ambient term (scaled ×10), normals, and rendered 3D shape.

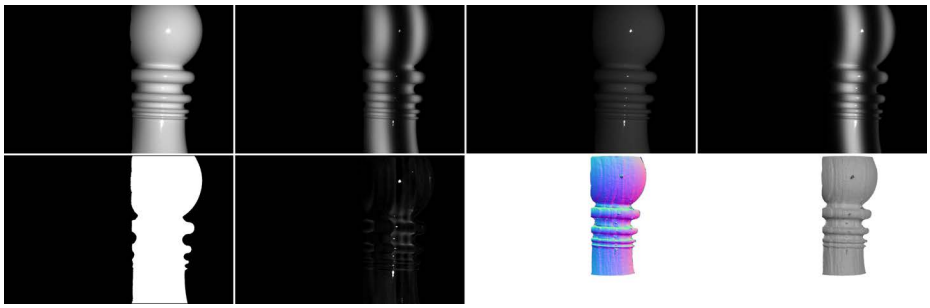


Fig. S55: Additional results: milled pawn with cheap projector. Top row: captured images from camera under uniform lighting and three patterns (2 phase-shifted sinusoidals and 1 ramp) from LCD projector. **Bottom row:** binary mask, fitted residual/ambient term (scaled ×10), normals, and rendered 3D shape.

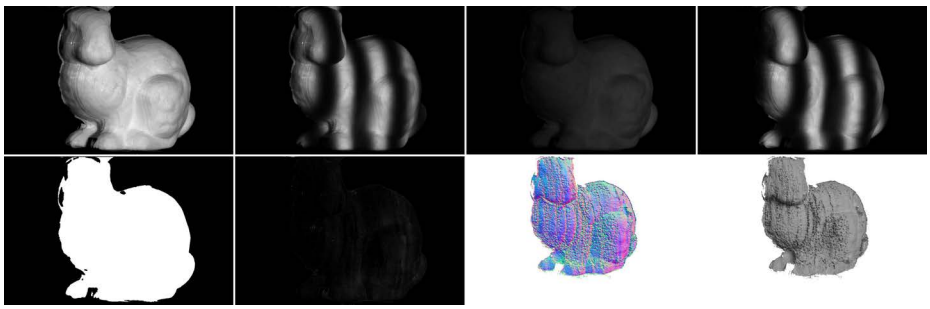


Fig. S56: Additional results: silicone bunny with cheap projector. Top row: captured images from camera under uniform lighting and three patterns (2 phase-shifted sinusoidals and 1 ramp) from LCD projector. **Bottom row:** binary mask, fitted residual/ambient term (scaled ×10), normals, and rendered 3D shape.

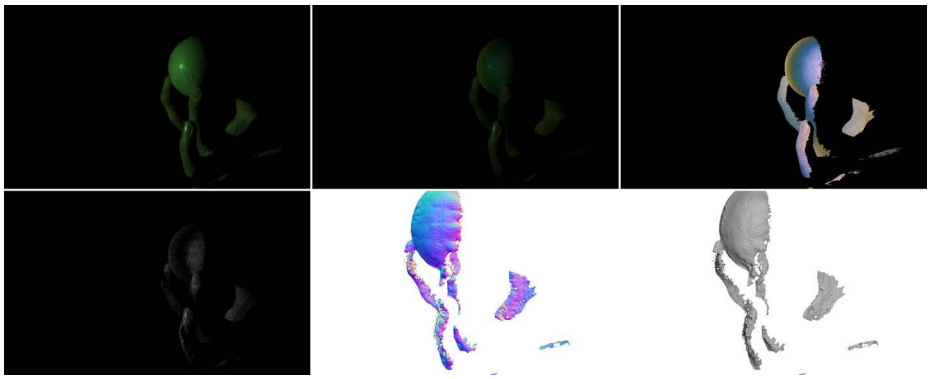


Fig. S57: Additional results: bouncy ball with Triton10 camera and analog projector. Top row: captured images from camera under uniform lighting and RGB pattern at 4.7ms exposure (both scaled $\times 2$), with the normalization $\mathbf{I}_{\text{pattern}}/\mathbf{I}_{\text{white}}$. **Bottom row:** fitted residual/ambient term (scaled $\times 10$), normals, and rendered 3D shape.

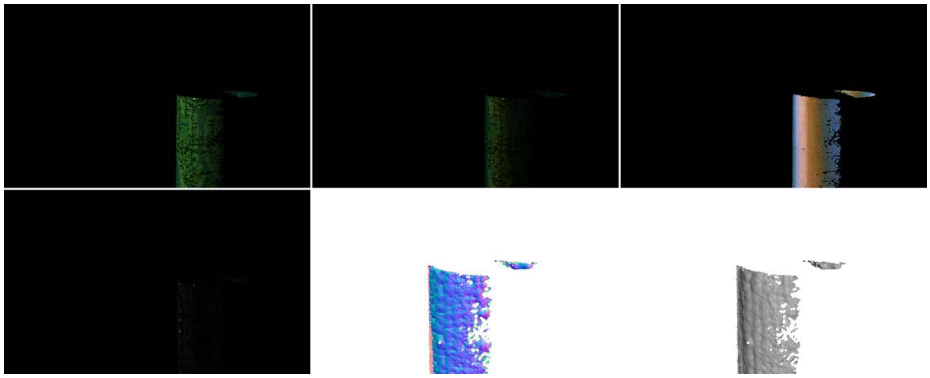


Fig. S58: Additional results: mug with Triton10 camera and analog projector. Top row: captured images from camera under uniform lighting and RGB pattern at 4.7ms exposure (both scaled $\times 2$), with the normalization $\mathbf{I}_{\text{pattern}}/\mathbf{I}_{\text{white}}$. **Bottom row:** fitted residual/ambient term (scaled $\times 10$), normals, and rendered 3D shape.

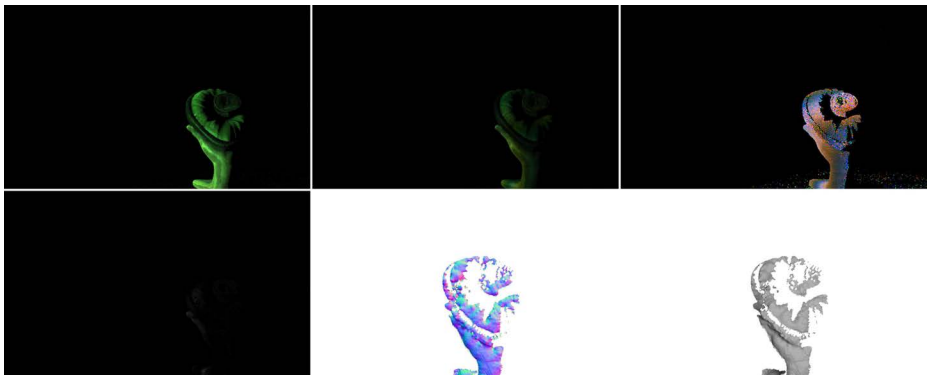


Fig. S59: Additional results: 3D printed avocado-hand with Triton10 camera and analog projector. Top row: captured images from camera under uniform lighting and RGB pattern at 4.7ms exposure (both scaled $\times 10$), with the normalization $\mathbf{I}_{\text{pattern}}/\mathbf{I}_{\text{white}}$. **Bottom row:** fitted residual/ambient term (scaled $\times 10$), normals, and rendered 3D shape.

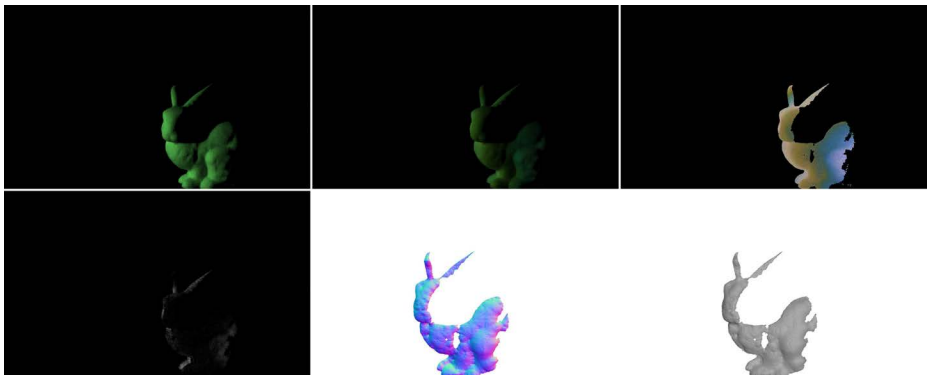


Fig. S60: Additional results: silicone bunny with Triton10 camera and analog projector. Top row: captured images from camera under uniform lighting and RGB pattern at 4.7ms exposure (both scaled $\times 2$), with the normalization $\mathbf{I}_{\text{pattern}}/\mathbf{I}_{\text{white}}$. **Bottom row:** fitted residual/ambient term (scaled $\times 10$), normals, and rendered 3D shape.

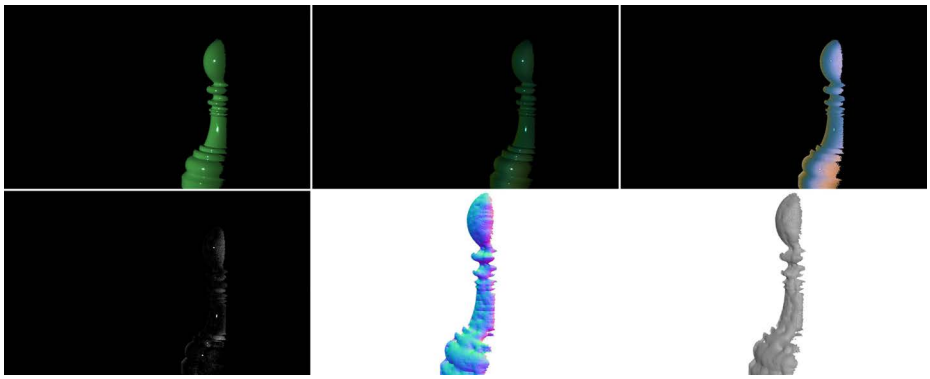


Fig. S61: Additional results: milled pawn with Triton10 camera and analog projector. Top row: captured images from camera under uniform lighting and RGB pattern at 4.7ms exposure (both scaled ×2), with the normalization $\mathbf{I}_{\text{pattern}}/\mathbf{I}_{\text{white}}$. **Bottom row:** fitted residual/ambient term (scaled ×10), normals, and rendered 3D shape.

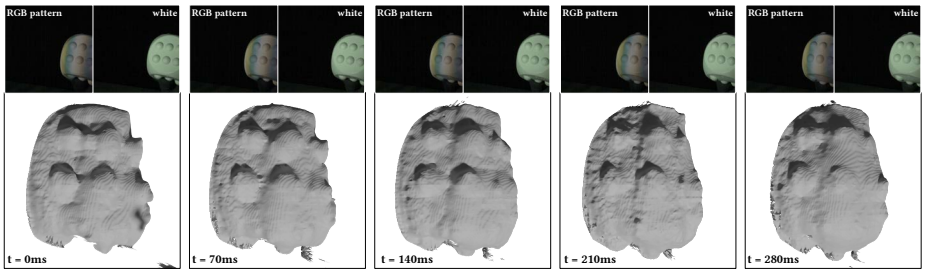


Fig. S62: Additional results: dynamic silicone cube with Chronos camera and analog projector. Top row: captured images from camera under RGB pattern and white flash (gamma mapped for better visualization). **Bottom row:** five rendered frames of the reconstruction.

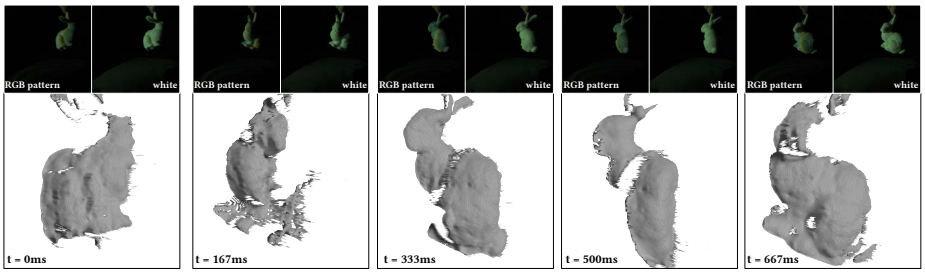


Fig. S63: Additional results: dynamic bunny with Triton10 camera and analog projector. Top row: captured images from camera under RGB pattern and white flash (gamma mapped for better visualization). **Bottom row:** five rendered frames of the reconstruction.

References

1. Gupta, M., Nayar, S.K.: Micro Phase Shifting. In: 2012 IEEE Conference on Computer Vision and Pattern Recognition. pp. 813–820. IEEE, Providence, RI (Jun 2012). https://doi.org/10.1109/CVPR.2012.6247753, http://ieeexplore.ieee.org/document/6247753/
2. Horn, E., Kiryati, N.: Toward optimal structured light patterns1based on "toward optimal structured light patterns" by e. horn and n. kiryati which appeared in proceedings of 3-d digital imaging and modelling, ottawa, may 97; (pp. 28—35). © 1999 ieee.1. Image and Vision Computing **17**(2), 87–97 (1999). https://doi.org/https://doi.org/10.1016/S0262-8856(98)00113-9, https://www.sciencedirect.com/science/article/pii/S0262885698001139
3. Li, R., Gao, H., Tancik, M., Kanazawa, A.: Nerfacc: Efficient sampling accelerates nerfs. arXiv preprint arXiv:2305.04966 (2023)
4. Martin, J., Crowley, J.L.: Comparison of correlation techniques. In: Intelligent Autonomous Systems. pp. 86–93 (1995)
5. Mirdehghan, P., Chen, W., Kutulakos, K.N.: Optimal structured light à la carte. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR) (June 2018)
6. Mirdehghan, P., Wu, M., Chen, W., Lindell, D.B., Kutulakos, K.N.: Turbosl: Dense accurate and fast 3d by neural inverse structured light. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR). pp. 25067–25076 (June 2024)
7. Moreno, D., Taubin, G.: Simple, Accurate, and Robust Projector-Camera Calibration. In: 2012 Second International Conference on 3D Imaging, Modeling, Processing, Visualization & Transmission. pp. 464–471. IEEE, Zurich, Switzerland (Oct 2012). https://doi.org/10.1109/3DIMPVT.2012.77, http://ieeexplore.ieee.org/document/6375029/
8. Müller, T., Evans, A., Schied, C., Keller, A.: Instant neural graphics primitives with a multiresolution hash encoding. ACM Trans. Graph. **41**(4) (Jul 2022). https://doi.org/10.1145/3528223.3530127, https://doi.org/10.1145/3528223.3530127
9. Narasimhan, S.G., Koppal, S.J., Yamazaki, S.: Temporal dithering of illumination for fast active vision. In: Forsyth, D., Torr, P., Zisserman, A. (eds.) Computer Vision – ECCV 2008. pp. 830–844. Springer Berlin Heidelberg, Berlin, Heidelberg (2008)
10. Pereira, G., Gao, Y., Piadyk, Y., Fouhey, D., Silva, C.T., Panozzo, D.: Lookup3d: Data-driven 3d scanning. In: Proceedings of the SIGGRAPH Asia 2025 Conference Papers. SA Conference Papers '25, Association for Computing Machinery, New York, NY, USA (2025). https://doi.org/10.1145/3757377.3763986, https://doi.org/10.1145/3757377.3763986
11. Ranftl, R., Lasinger, K., Hafner, D., Schindler, K., Koltun, V.: Towards robust monocular depth estimation: Mixing datasets for zero-shot cross-dataset transfer. IEEE Transactions on Pattern Analysis and Machine Intelligence **44**(3) (2022)
12. Wang, P., Liu, L., Liu, Y., Theobalt, C., Komura, T., Wang, W.: Neus: Learning neural implicit surfaces by volume rendering for multi-view reconstruction. NeurIPS (2021)
13. Yang, L., Kang, B., Huang, Z., Xu, X., Feng, J., Zhao, H.: Depth anything: Unleashing the power of large-scale unlabeled data. In: CVPR (2024)

14. Yang, L., Kang, B., Huang, Z., Zhao, Z., Xu, X., Feng, J., Zhao, H.: Depth anything v2. arXiv:2406.09414 (2024)