Supplemental Material for
Neural Fields for Structured Lighting

Aarrushi Shandilya Benjamin Attal Christian Richardt [∞] James Tompkin [†] Matthew O'Toole
Carnegie Mellon University [∞] Meta Reality Labs Research [†] Brown University

A. Real Data

Our proposed framework uses real data captured with an Intel RealSense D435 camera and a built-in infrared dot projector. In this section, we detail our calibration routine to recover the intrinsic and extrinsic parameters of the infrared camera and projector, and recover the projector's structured light dot pattern. Our image formation model uses these calibration parameters to synthesize structured light images of the scene. While the RealSense has two infrared cameras and both cameras are used to calibrate the system and compute poses through COLMAP, only a single camera stream is used during training for all our structured light experiments (for simplicity). Note that the RealSense uses both cameras to generate dense depth maps, which serves as input images for depth-supervised NeRF methods.

A.1. Camera Calibration

The first step of the calibration procedure is to compute the intrinsic matrices $(\mathbf{K}_1, \mathbf{K}_2)$, distortion $(\mathbf{d}_1, \mathbf{d}_2)$ and rectification $(\mathbf{r}_1, \mathbf{r}_2)$ parameters, relative extrinsics $(\mathbf{R}, \mathbf{T})$ and projection matrices $(\mathbf{P}_1, \mathbf{P}_2)$ for the monochromatic stereo cameras. Streaming both the cameras (with projector *off*) at a resolution of 848 × 480 pixels, we capture images of a 7×8 planar checkerboard in a variety of poses (Figure A). Using standard OpenCV functions and calibration flow, we compute these parameters as outlined Figure B and in pseudocode Algorithm 1.

A.2. Projector Calibration

The next step is to calibrate the projector itself. The stream for both cameras is enabled (with projector *on*) at a resolution of 848 × 480 pixels. We capture the pattern projected onto a white plane as it is moved at different depths, and kept roughly parallel to the camera plane. We assume the scene contains no ambient illumination (*i.e*., the images are captured in a dark room).

Since the camera is imaging a planar surface, the structured light pattern formed on the sensor is related to all other frames through a homography transform. The key idea is to compute homography between any two images of the pattern using an image alignment technique such as Enhanced Correlation Coefficient (ECC) Maximization. This homography can then be used to warp one image onto another or transform the pixel coordinates of an image to corresponding coordinates in the other image.

The calibration flow is depicted in Figure C and the pseudocode in Algorithm 2. Assigning the first frame to be the *base* image, we estimate its homography with respect to all other captured frames from both cameras. Using this

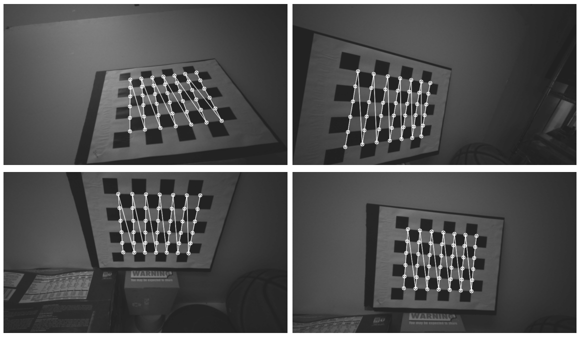


Figure A. **Sample checkerboard captures with detected corners.** 100+ checkerboard poses are used to span both cameras' field of view at different depths and orientations to accurately calibrate for the camera parameters.

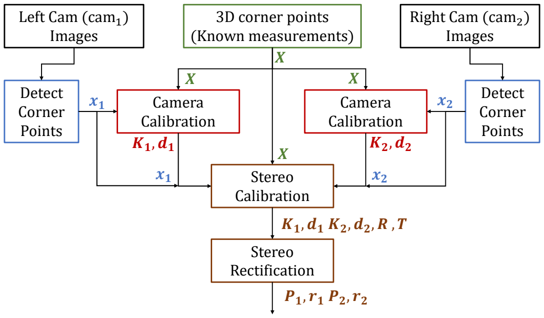
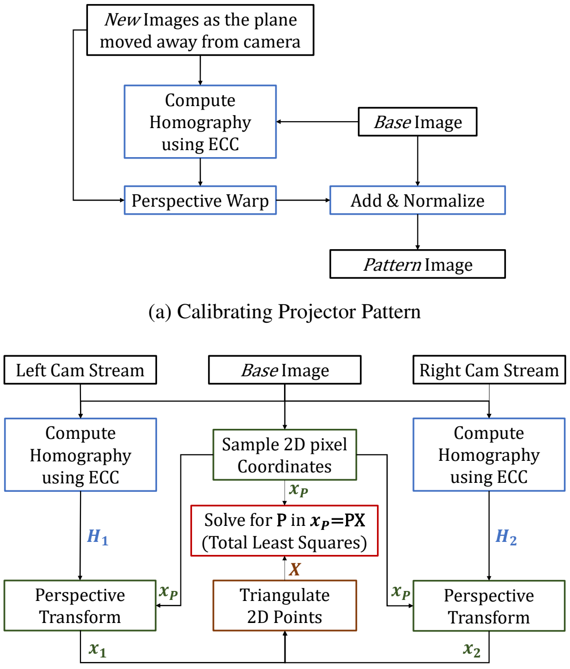


Figure B. **Camera calibration flow using OpenCV.** Corner point correspondences between the world frame and camera images are used to recover camera parameters via optimization function calls of Camera Calibration, Stereo Calibration and Stereo Rectification.

transform, each image is warped onto the *base* image to formulate the complete *pattern* image (Figure D) with some additional normalization. Similarly, the homographies are used to transform the 2D image coordinates sampled in the *pattern* image to corresponding coordinates in the left and right camera images. These 2D coordinates from both camera images are then triangulated to 3D world coordinates using the calibrated camera projection matrices (obtained in Section A.1). Repeating this for all images, we get a set of correspondences between the 2D projector *pattern* coordinates and 3D *world* coordinates, which allows us to solve for the projector's projection matrix using Singular Value Decomposition (SVD) for Total Least Squares. Note that this procedure does not account for effects like projector defocus. Additionally, we postprocess the pattern to remove any contribution from the fall-off in pattern intensities as we account for this factor explicitly in our image formation model.



New Images as the plane moved away from camera
Compute Homography using ECC
Base Image
Perspective Warp
Add & Normalize
Pattern Image
Left Cam Stream
Base Image
Right Cam Stream
Compute Homography using ECC
Sample 2D pixel Coordinates
Compute Homography using ECC
x_P
Solve for **P** in x_P=PX (Total Least Squares)
H_1
H_2
X
Perspective Transform
x_P
Triangulate 2D Points
x_P
Perspective Transform
x_1
x_2

(a) Calibrating Projector Pattern

(b) Calibrating Projection Matrix

Figure C. **Projector calibration flow.** (a) Warping each frame onto a *base* image using an estimated Homography formulates the *pattern* image. (b) Correspondences between the projector's *base* image pixels and the triangulated world coordinates are used to solve for the projector's projection matrix.

A.3. Pose Optimization

For each scene, we also use the stereo camera system to compute poses through COLMAP for frames where the projector is *off*. As the baseline between cameras is known, there is no scale ambiguity associated with the poses. Note that the stereo camera pair is used only for calibration, however all the demonstrated experiments use the captures from a single camera only.

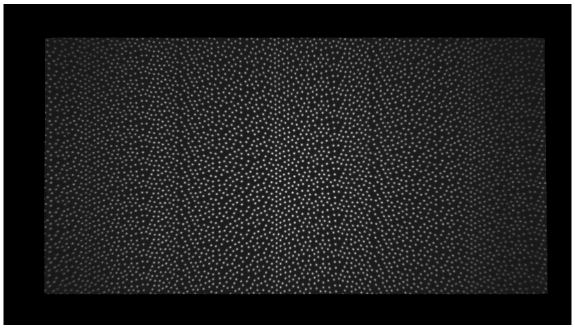


Figure D. **Calibrated projector pattern for Intel RealSense D435.**

During the stroboscopic streaming of the camera, there can be significant motion between the projector *on* and *off* images. Hence, the poses for *off* images obtained via COLMAP cannot be used directly for the *on* images. To calibrate for the different poses, we enable a one-shot pose-optimization flow as a pre-processing step. This allows us to leverage large number of views for accurate pose recovery irrespective of sparse-view training during our experiments. Representing the poses as twists, we initialize the *on* image poses same as the *off* image poses from COLMAP. Then, the *on* image poses across all captures of a scene are optimized using the total photometric loss of Equation 14 for a total of 160K iterations. For optimizing these poses, we use Adam optimizer with an initial learning rate of 0.001, which is decayed by a rate of 0.1 over the first 100K iterations. For the volumetric scene model, we use the same optimizer and learning rate as in NeRF, while the weight λ_2 is decayed from 1 to 0.001 over the first 100K iterations. Prediction and loss on normals is disabled during this optimization process, *i.e.*, $\lambda_2 = \lambda_3 = 0$ in Equation (14) to speed-up the calibration. Going forward, we use these per-scene calibrated poses for all our experiments without any further need of pose optimization during training or testing.

B. Synthetic Data

In Blender, we form a structured light system using a projector plugin (add on from https://github.com/Ocupe/Projectors). The scenes include objects from NeRF's Blender dataset and open source 3D models (from https://blendswap.com/) (license information in Table A) placed on a textureless plane. Keeping the relative camera-projector pose fixed, the structured light rig is perturbed to generate random views. For each view, we render 400×400 images with the projector illumination turned *on* and *off*. When the projector is *on*, it illuminates the scene with a high frequency dot pattern. Ground truth extrinsics and intrinsics are directly retrieved from Blender without any additional calibration.

Table A. **License information for open-source scenes used from** https://blendswap.com/.

Scene	License	Author	Link
Sofa	CC-0	Darilon	https://blendswap.com/blend/30053
Frog	CC-0	craggle	https://blendswap.com/blend/30092
Skateboard	CC-0	MattMump	https://blendswap.com/blend/4859
Domino	CC-0	alepx	https://blendswap.com/blend/6584

C. Quantitative Results

We provide experiment per-scene metrics for the sparse view reconstruction on real scenes (Tables B to E) and normal prediction on synthetic scenes (Table F).

Table B. **Quantitative analysis on *box* scene.**

	PSNR ▲			SSIM ▲			LPIPS ▼		
Method	2-v	4-v	8-v	2-v	4-v	8-v	2-v	4-v	8-v
NeRF	30.78	35.57	38.39	0.943	0.964	0.971	0.392	0.298	0.252
+ sparse depth	**38.50**	**43.41**	**45.51**	**0.979**	**0.991**	**0.993**	0.233	**0.166**	**0.162**
+ dense depth	36.31	42.47	44.89	0.972	0.989	0.992	0.252	0.194	0.177
Ours	34.90	43.40	44.82	0.967	**0.991**	**0.993**	0.244	0.179	0.171
+ dense depth	36.25	43.07	45.24	0.973	**0.991**	**0.993**	**0.230**	0.180	0.168

Table C. **Quantitative analysis on *sculpture* scene.**

	PSNR ▲			SSIM ▲			LPIPS ▼		
Method	2-v	4-v	8-v	2-v	4-v	8-v	2-v	4-v	8-v
NeRF	25.68	38.39	43.83	0.899	0.984	0.991	0.344	0.161	0.141
+ sparse depth	32.54	**43.15**	**44.42**	0.950	**0.992**	**0.993**	0.226	**0.125**	**0.117**
+ dense depth	33.17	41.30	43.30	0.958	0.987	0.991	0.202	0.146	0.142
Ours	27.17	40.96	43.15	0.911	0.988	0.991	0.277	0.145	0.141
+ dense depth	**33.22**	40.54	43.59	**0.959**	0.986	0.991	**0.198**	0.148	0.139

Table D. **Quantitative analysis on *woodshop* scene.**

	PSNR ▲			SSIM ▲			LPIPS ▼		
Method	2-v	4-v	8-v	2-v	4-v	8-v	2-v	4-v	8-v
NeRF	24.14	27.50	31.67	0.778	0.860	0.917	0.480	0.400	0.269
+ sparse depth	36.92	**38.45**	38.39	0.969	**0.984**	**0.986**	0.144	**0.115**	**0.101**
+ dense depth	37.19	37.64	38.17	0.970	0.979	0.983	0.163	0.149	0.140
Ours	37.60	37.65	38.37	0.973	0.977	0.982	0.147	0.146	0.138
+ dense depth	**37.78**	38.03	**38.58**	**0.975**	0.980	0.984	**0.143**	0.141	0.136

Table E. **Quantitative analysis on *doll* scene.**

	PSNR ▲			SSIM ▲			LPIPS ▼		
Method	2-v	4-v	8-v	2-v	4-v	8-v	2-v	4-v	8-v
NeRF	29.51	32.14	39.37	0.925	0.938	0.980	0.381	0.369	0.261
+ sparse depth	39.01	39.88	41.21	0.977	0.983	0.986	0.254	0.225	0.211
+ dense depth	39.77	41.54	41.87	0.983	0.987	0.988	0.212	0.202	0.199
Ours	**39.90**	41.85	41.74	**0.984**	**0.988**	0.988	**0.204**	0.197	0.200
+ dense depth	39.55	**41.91**	**41.99**	0.983	**0.988**	**0.989**	0.214	**0.195**	**0.195**

Table F. **Scene-wise quantitative analysis on synthetic data.**

	Method	PSNR ▲	SSIM ▲	LPIPS ▼	Depth MSE ▼	Normals MAE °▼
lego	NeRF	**43.62**	**0.984**	**0.407**	0.915	23.90
	Flood Light	41.47	0.979	0.418	0.574	7.95
	Structured Light	42.62	0.982	0.411	**0.008**	**2.61**
skate	NeRF	**45.49**	**0.985**	**0.323**	0.409	24.62
	Flood Light	44.32	0.983	0.327	0.632	7.30
	Structured Light	44.68	0.984	0.325	**0.004**	**2.71**
sofa	NeRF	**46.22**	**0.986**	**0.389**	0.846	23.33
	Flood Light	45.17	0.985	0.395	1.096	7.28
	Structured Light	45.59	**0.986**	**0.389**	**0.033**	**2.40**
dom	NeRF	**44.58**	**0.984**	**0.353**	0.291	25.51
	Flood Light	43.09	0.981	0.358	0.841	12.50
	Structured Light	43.65	0.983	0.355	**0.008**	**3.61**

D. Qualitative Results

D.1. Point Cloud Visualization

In this section, we demonstrate point cloud visualizations for some of the real scenes decomposed in Section 5.5. Given a camera view, the point cloud location for each pixel is computed using $\mathbf{o} + z\mathbf{d}$, where $\mathbf{o}$ is the camera origin, $\mathbf{d}$ is the unit ray direction pointing away from the camera, and z is the depth computed by weighted accumulation of the learnt volume density. Figures E and F show the point cloud reconstructions for our method using 50 views.

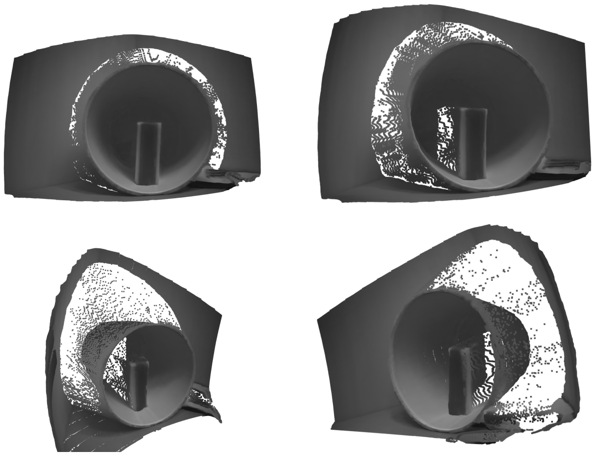


Figure E. **Point cloud visualization for the *candle* scene.**

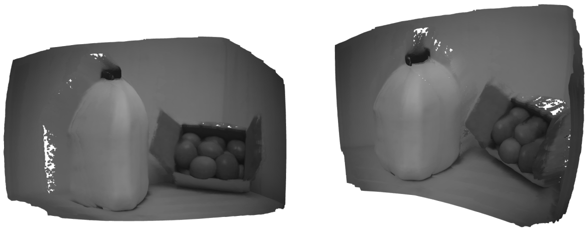


Figure F. **Point cloud visualization for the *milk* scene.**

Algorithm 1 Camera Calibration using standard OpenCV functions

```
$\mathbf{x}_1 \leftarrow$ findChessboardCorners ($cam_1$ images)
$\mathbf{x}_2 \leftarrow$ findChessboardCorners ($cam_2$ images)
$\mathbf{X} \leftarrow$ 3D world coordinates for checkerboard corners
$\mathbf{K}_1, \mathbf{d}_1 \leftarrow$ calibrateCamera$(\mathbf{X}, \mathbf{x}_1)$
$\mathbf{K}_2, \mathbf{d}_2 \leftarrow$ calibrateCamera$(\mathbf{X}, \mathbf{x}_2)$
$\mathbf{K}_1, \mathbf{d}_1, \mathbf{K}_2, \mathbf{d}_2, \mathbf{R}, \mathbf{T} \leftarrow$ stereoCalibrate$(\mathbf{X}, \mathbf{x}_1, \mathbf{x}_2, \mathbf{K}_1, \mathbf{d}_1, \mathbf{K}_2, \mathbf{d}_2)$
$\mathbf{P}_1, \mathbf{P}_2, \mathbf{r}_1, \mathbf{r}_2 \leftarrow$ stereoRectify$(\mathbf{K}_1, \mathbf{d}_1, \mathbf{K}_2, \mathbf{d}_2, \mathbf{R}, \mathbf{T})$
```

Algorithm 2 Projector Calibration using standard OpenCV functions

```
$stream_1, stream_2 \leftarrow$ load stream from both cameras as grayscale images
$stream_1, mask_1, stream_2, mask_2 \leftarrow$ undistort, rectify images using $\mathbf{d}_1, \mathbf{d}_2, \mathbf{r}_1, \mathbf{r}_2$
$base \leftarrow$ first frame from $stream_1$
$pattern, mask \leftarrow 0$
$\mathbf{x}_p \leftarrow 0$ (2D *pattern* coordinates)
$\mathbf{X} \leftarrow 0$ (3D *world* coordinates)
$\mathbf{H}_1 \leftarrow \mathbf{I}_{2\times 3}$
$\mathbf{H}_2 \leftarrow \begin{bmatrix} 1 & -0.005 & 0 \\ 0 & 1 & -48.9 \end{bmatrix}$

for all $\{img_1, img_2\} \in \{stream_1, stream_2\}$ do
    $\mathbf{H}_1 \leftarrow$ findTransformECC$(base, img_1, \mathbf{H}_1, mask_1)$
    $rimg_1 \leftarrow$ warpPerspective$(img_1, \mathbf{H}_1^{-1})$
    $rmask_1 \leftarrow$ warpPerspective$(mask_1, \mathbf{H}_1^{-1})$
    $pattern \leftarrow pattern + rimg_1 * rmask_1$
    $mask \leftarrow mask + rmask_1$

    $\mathbf{H}_2 \leftarrow$ findTransformECC$(base, img_2, \mathbf{H}_2, mask_2)$
    $rimg_2 \leftarrow$ warpPerspective$(img_2, \mathbf{H}_2^{-1})$
    $rmask_2 \leftarrow$ warpPerspective$(mask_2, \mathbf{H}_2^{-1})$
    $pattern \leftarrow pattern + rimg_2 * rmask_2$
    $mask \leftarrow mask + rmask_2$

    $\{u, v\} \leftarrow$ sample image coordinates on the *pattern* image
    $\{u_1, v_1\} \leftarrow$ perspectiveTransform$(\{u, v\}, \mathbf{H}_1)$
    $\{u_2, v_2\} \leftarrow$ perspectiveTransform$(\{u, v\}, \mathbf{H}_2)$
    $\{X, Y, Z\} \leftarrow$ triangulatePoints$(\{u_1, v_1\}, \mathbf{P}_1, \{u_2, v_2\}, \mathbf{P}_2)$
    $\mathbf{x}_p \leftarrow$ append $\{u, v\}$ points
    $\mathbf{X} \leftarrow$ append $\{X, Y, Z\}$ points
end for
$pattern = 0.5 \times (pattern/mask)$
Projector's $\mathbf{P} \leftarrow$ Total Least Squares using the point correspondences $\mathbf{x}_p$ and $\mathbf{X}$
```