Supplementary Information
Learned Feature Embeddings for Non-line-of-Sight Imaging and Recognition

WENZHENG CHEN*, University of Toronto, Vector Institute
FANGYIN WEI*, Princeton University
KIRIAKOS N. KUTULAKOS, University of Toronto
SZYMON RUSINKIEWICZ, Princeton University
FELIX HEIDE, Princeton University

ACM Reference Format:
Wenzheng Chen*, Fangyin Wei*, Kiriakos N. Kutulakos, Szymon Rusinkiewicz, and Felix Heide. 2020. Supplementary Information Learned Feature Embeddings for Non-line-of-Sight Imaging and Recognition. *ACM Trans. Graph.* 39, 6, Article 230 (December 2020), 14 pages. https://doi.org/10.1145/nnnnnnn.nnnnnnn

In this document we provide additional discussion and results in support of the primary text.

1 ADDITIONAL EXPERIMENTAL COMPARISONS

1.1 Filtered Backprojection Comparisons

The first proposed reconstruction method for NLOS reconstruction is filtered backprojection which propagates and aggregates time-resolved intensity measurements back into in the occluded volume, followed by a Laplacian or Laplacian of Gaussian filter [Velten et al. 2012], an approach extended in recent years [Arellano et al. 2017; Jarabo et al. 2017; Laurenzis and Velten 2014]. Although these methods have been evaluated in prior work [Lindell et al. 2019; Liu et al. 2019], for completeness, we provide results from different filtered backprojection variations in Fig. 1. Specifically, we compare the filtered backprojection using Laplacian filtering using the code from [Lindell et al. 2019] and filtered backprojection using Laplacian and LoG filtering [Laurenzis and Velten 2014] using the code from [Liu et al. 2019]. Although the Laplacian-filtered variants both implement the approach from [Velten et al. 2012] we did notice differences in both implementations, and, hence, present all methods for completeness.

Consistent with prior work, we find that existing backprojection methods succeed in recovering detail even for complex scenes, but this comes at the cost of compute time. Specifically, backprojection methods have a runtime complexity of $O(N^5)$ with N as the volume resolution. For our experiments, a single transient reconstruction

* indicates equal contribution.

Authors' addresses: Wenzheng Chen*, wenzheng@cs.toronto.edu, University of Toronto, Vector Institute; Fangyin Wei*, fwei@cs.princeton.edu, Princeton University; Kiriakos N. Kutulakos, kyros@cs.toronto.edu, University of Toronto; Szymon Rusinkiewicz, smr@princeton.edu, Princeton University; Felix Heide, fheide@cs.princeton.edu, Princeton University.

0730-0301/2020/12-ART230 $15.00
https://doi.org/10.1145/nnnnnnn.nnnnnnn

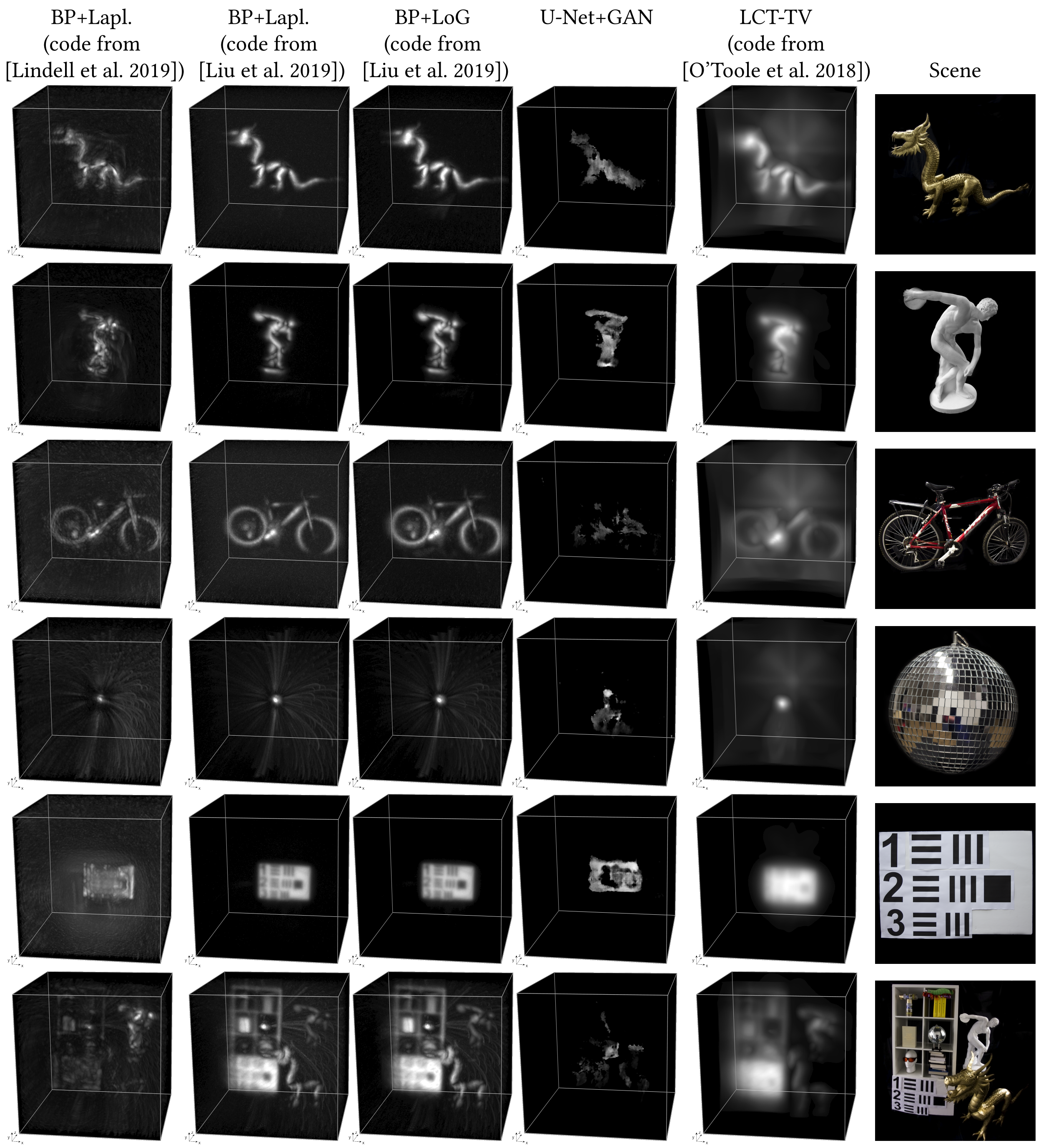
BP+Lapl. (code from [Lindell et al. 2019])
BP+Lapl. (code from [Liu et al. 2019])
BP+LoG (code from [Liu et al. 2019])
U-Net+GAN
LCT-TV (code from [O'Toole et al. 2018])
Scene
1
2
3

Fig. 1. **Additional Reconstructions from Pulsed Single-Photon Measurements** For completeness, we show reconstructions from different filtered back-projection (BP) implementations. Specifically, we list BP + Laplacian filtering using the code from [Lindell et al. 2019], BP + Laplacian filtering and LoG filters using the code from [Liu et al. 2019]. As additional methods using priors, we show results from a UNet trained with generative adversarial training in the fourth columns and iterative LCT with total variation regularization using the code and parameters from [O'Toole et al. 2018].

took around 1 hour for the 256×256×512 transient measurement while the recent LCT, FK or Phasor method take several seconds, and the proposed method takes runs at 45 milliseconds for a single reconstruction.

1.2 Encoder-decoder and Generative Adversarial Baselines

In Fig. 12 of the main paper, we show that encoder-decoder networks, based on the popular U-Net architecture [Ronneberger et al. 2015], fail to generalize to real data when only trained in simulation. In this section, we provide additional detail on the U-Net architecture we evaluate, and we validate that generative adversarial training represents no alternative to the proposed method.

Transient U-Net Architecture Details. To map transient images to 2D renderings, we use a 3D U-Net embedded and the differentiable rendering architecture described in the main paper. Specifically, given a raw transient image of size (512, 256, 256, 3), the encoder network applies four convolutional downsampling blocks, resulting in a feature of size (32, 32, 32, 64). The 3D U-Net has four downsampling and four upsampling steps. At each downsampling step a $3 \times 3 \times 3$ convolution with stride 2 and output channel number doubled is applied and followed by an instance normalization layer and LeakyReLU. Each upsampling step consists of a $3 \times 3 \times 3$ up-convolution with stride 2 that halves the number of input feature channels, an instance normalization layer, a ReLU, and a concatenation with a feature map from its corresponding downsampling stage. We apply dropout during the early-stage feature extraction to enhance the networks's robustness. The image and depth rendering network shares exactly the same structure as described in the main document.

Adversarial Loss. In addition to the vanilla U-Net model, we also evaluate whether adding adversarial losses [Goodfellow et al. 2014] can help the generalization to real data. To this end, we use a U-Net model with the same architecture as described above, and add a discriminator network to adversarially train the reconstruction model. The discriminator network similar to [Radford et al. 2015] which is composed of 5 downsampling convolutional layers with kernel size 4 and stride 2 and one final fully connected layer to distinguish between GT and reconstructed images. All convolutional layers are interlaced with IN and LeakyReLU (with slope 0.2). We use the same GAN loss as described in [Radford et al. 2015] plus previous RGB-D reconstruction loss as described in Sec. 5.2 in the main paper. In Fig. 1, we show experimental reconstructions from U-Net architecture trained with additional GAN loss. While this model recovers rough shape, it fails to recover fine detail and complex structures such as the bike or the complex room scene. This experiment and the U-Net results in Fig. 12 of the main paper both validate that existing encoder-decoder networks and generative adversarial networks do not generalize to experimental data, confirming the effectiveness of the proposed network architecture.

1.3 Total Variation Regularization

Iterative LCT with total variation regularization [O'Toole et al. 2018] allows to incorporate traditional sparse gradient priors into the reconstruction. The authors rely on an ADMM optimization algorithm to combine LCT-based inverse filtering with sparsity priors. While this process method takes around 5 seconds to optimize a 64×64×512 transient cube *per iteration*, this runtime increases to 60 seconds for our measurement resolution of 256×256×512. The overall optimization requires hundreds of iterations translating to hours of runtime, whereas the proposed method runs at interactive framerates and allows to incorporate more effective learned priors instead of traditional sparsity priors.

Even with author-provided parameters and using the authors' code, the total variation-regularized reconstructions fail to recover detail. Reconstruction noise, present in the Phasor reconstruction method [Liu et al. 2019] is suppressed at the cost of losing fine structures. The proposed method learns more effective priors that preserve fine detail while suppressing substantial reconstruction noise.

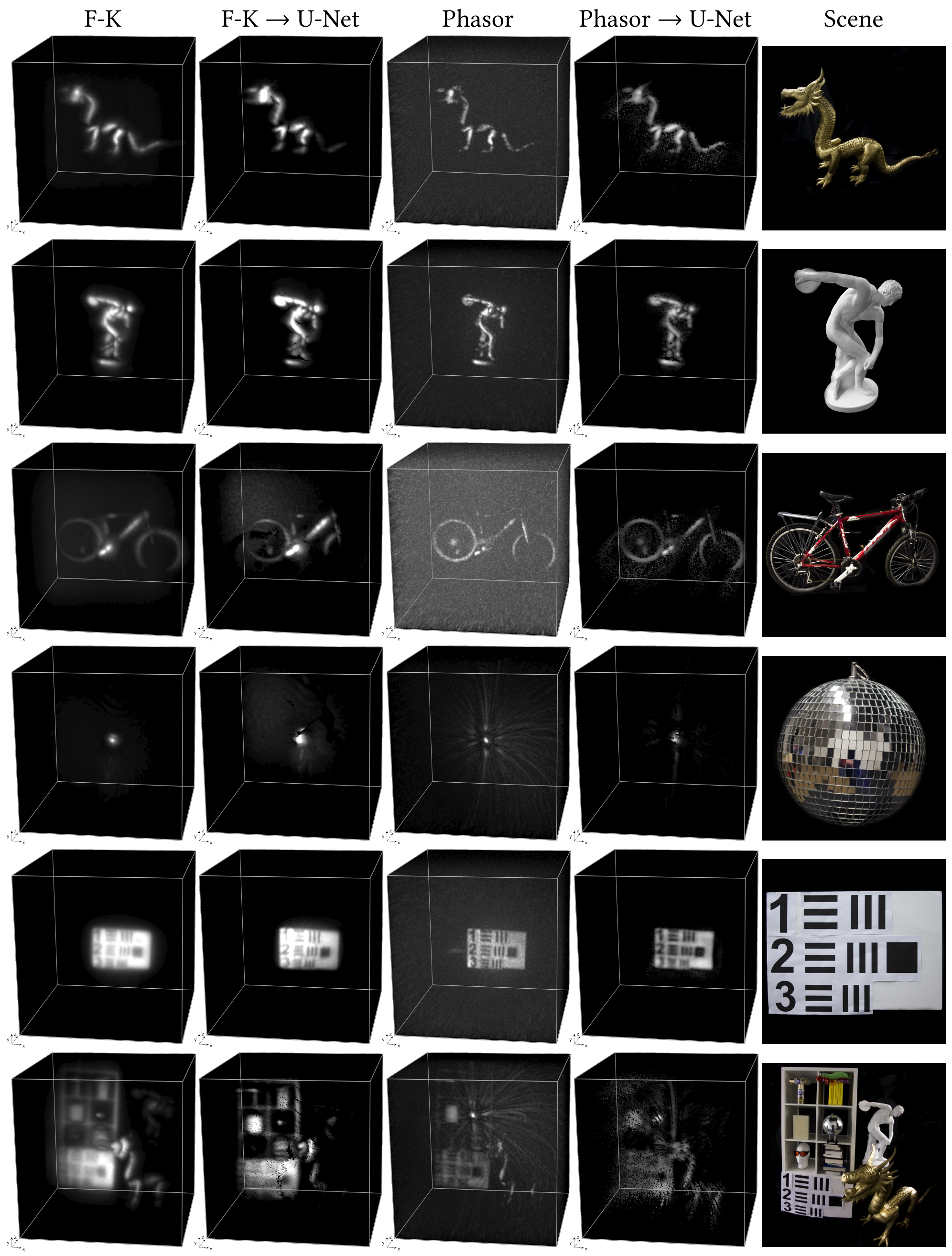

Fig. 2. **Additional Learned Denoising Comparisons from Pulsed Single-Photon Measurements** We train a 2D U-Net denoiser on top of F-K and Phasor predictions and train it on synthetic bike dataset. We find that a 2D denoiser helps to reduce reconstruction noise and produce a cleaner background, especially for Phasor estimates. However, such post-processing approaches struggle to distinguish scene content from unwanted low-frequency recovery noise. As such, learned denoisers also tend to eliminate fine scene geometry from the estimates making them no alternative to the proposed method demonstrated on the same data as Fig. 12 of the main manuscript.

1.4 Denoising Baselines

As additional comparisons, Fig. 2 shows the reconstruction of F-K [Lindell et al. 2019] and Phasor [Liu et al. 2019] combined with a learned 2D U-Net denoiser. Specifically, we first use the F-K and Phasor method to predict volumetric albedo on our training dataset and compute maximum-intensity projections along the z-axis. Next, we train a vanilla 2D U-Net on this dataset to improve the reconstruction quality with L_2 loss supervision using the simulated ground truth images. The architecture of this standard 2D U-Net [Ronneberger et al. 2015] is described in the following for completeness. The network first applies one convolutional layer with kernel size 5, stride 2, input channel number 1 and output channel number 8. Then the downsampling stage contains seven downsampling blocks where each block is composed of a 2D convolutional layer, a batch normalization layer and a leaky ReLU (slope 0.2). Each of the seven blocks downsamples its input feature by a factor of two spatially (stride two for the convolutional layer) and doubles the input channel number (except for the last two blocks whose output channel number is the same as the input channel number). The convolutional layers in the first six downsampling blocks have kernel size 5, and are padded by two more zeros on each boundary to guarantee the output feature is exactly halved spatially. The convolutional layer in the last downsampling block has kernel size 3, and is padded by one more zero on each boundary to guarantee the output feature is exactly halved spatially. The upsampling stage, containing six upsampling blocks, is a mirror of the downsampling stage. Each upsampling block accepts the output of its preceding upsampling block and the output of a previous downsampling block with the same spatial size. In each upsampling block, it first upsamples the feature from its preceding upsampling block by a factor of two and applies one convolutional layer (with kernel size 5, stride 1) combined with one batch normalization and Leaky ReLU to this upsampled feature. Then it concatenates this upsampled feature with the feature from previous downsampling block. After the upsampling stage, three branches are created on top of the outputs of the last three upsampling blocks, respectively. All branches are passed through one convolutional layers with kernel size 5, stride 1, and then upsampled to the same spatial resolution as the output image. Then these three branches are added together to obtain the final denoised prediction.

The resulting denoiser is then tested on experimental data. In Fig. 2, for consistent visualization, we convert the denoised 2D image back into the 3D volume with depth extracted from maximum intensity projection of the F-K and Phasor method. We find that such a learned denoiser does improve reconstruction quality, especially for the Phasor reconstructions. Specifically, the post-filtering approach helps to suppress background noise. However, the learned denoiser is not semantic-aware in contrast to the proposed method and struggles to unmix signal from low-frequency artefacts and noise. This results in the suppression of object details along with noise. For example the low-intensity statue in the far back of the complex scene is removed by the post-filtering denoiser, and object geometry of the bike scene are missing after denoising the F-K and Phasor estimates.

1.5 Additional Non-Confocal Reconstructions

While the LCT and F-K methods require confocal data [O'Toole et al. 2018], and, as such, rely on a conversion process for non-confocal data, the Phasor propagation methods [Liu et al. 2020, 2019] directly support non-confocal data. Our feature propagation framework provides a flexible approach to these input setup configurations without the need for conversion. It naturally generalizes from confocal data to non-confocal setups by switching or fully learning the feature propagation unit $\mathcal{F}_{t\to s}$. As such, the method also supports unknown setup configurations. We have implemented propagation methods relying on confocal data, non-confocal setups, and fully learned feature propagation networks as described in the main draft.

For completeness, we provide additional results on non-confocal experimental measurements provided by [Liu et al. 2020, 2019]. As in the main draft, our model is trained on the synthetic bike dataset only and we find that it well generalized to real captures for both non-confocal and confocal setups. As shown in Fig. 3, we successfully reconstruct the details of the shelf, achieving results comparable with the fast Phasor propagation method [Liu

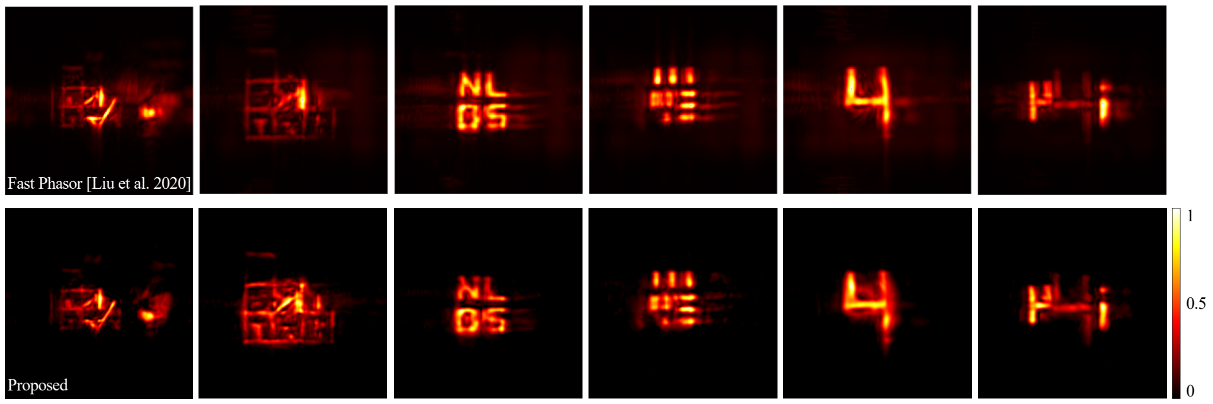

Fig. 3. **Additional Non-Confocal Reconstruction Comparisons with Measurements from [Liu et al. 2020, 2019].** Compared to the fast Phasor propagation method [Liu et al. 2020], the proposed model, trained only on synthetic bike dataset, recovers NLOS images with comparable details and substantially reduced reconstruction artefacts in the background. In particular, the proposed model is able to reconstruct details of the shelf scenes with higher-order multi-path contributions beyond the third-bounce transport simulated in the training data for our model. We use the same color scheme for visualization as [Liu et al. 2020].

Scene Complexity	128×128 quads	16×16 quads	4×4 quads
Jarabo et al. [Jarabo and Arellano 2018; Jarabo et al. 2014]	21.64s	19.91s	18.94s
Pediredla et al. [Pediredla et al. 2019]	26.2s	25.8s	25.5s
Iseringhausen and Hullin [Iseringhausen and Hullin 2020]	1032.2ms	19.02ms	5.44ms
Proposed	26.90ms	24.06ms	23.19ms

Table 1. **Transient Rendering Time Comparison** While the method from Iseringhausen and Hullin [Iseringhausen and Hullin 2020] requires a second of rendering time for scenes that have more than a few hundreds of triangle primitives, the proposed rasterization-based rendering method renders both simple and complex scenes at real-time rates, which is 30× faster than [Iseringhausen and Hullin 2020]. Our simulation dataset contains scenes shown in Fig. 6 in the main paper with 52081 vertices and 200018 faces, where only the proposed method achieves real-time rates.

et al. 2020]. For reference, we run the the MATLAB code provided by [Liu et al. 2020] on a general-purpose CPU with a runtime of 3 seconds on our desktop machine.

Note that our model is trained entirely on synthetic data with three-bounce light propagation only, while we test our model on the complex shelf examples with higher-order indirect path contributions. Note that our target reconstruction volumes do not cover the postitions of the rear walls in the scenes from [Liu et al. 2020, 2019]. Overall, we demonstrate that the proposed method does not suffer from overfitting with our three-bounce training dataset, which we attribute to the transient measurements being dominated by third bounce returns.

2 ADDITIONAL TRANSIENT RENDERING COMPARISONS

Compared to multi-path ray tracing-based transient renderers, for the mesh shown in Fig. 5 of the main paper, the method from Jarabo et al. [2018; 2014] takes 23 seconds to render a transient image with the same settings as ours, while the method of Pediredla et al. [2019] takes about 26 seconds for our setting.

Fig. 4. Architecture details of the feature extraction network and feature abstraction network. In the table, "conv-k(a)-s(b)-p(c)-LReLU" represents a convolution layer with an $a \times a$ kernel window, using stride b, padded c values on the boundary by replication, followed by a Leaky ReLU ($\alpha = 0.2$) activation function.

Feature Extraction Network

Layer Name	Type	Input Name	Channels	Output Size
Input1	Transient Image		3	(512, 256,256,3)
down1	conv-k3-s2-p1-LReLU	Input1	3	(256,128,128,3)
down2	conv-k3-s2-p1-LReLU	Input1	3	(256,128,128,3)
res1-1	conv-k3-s1-p1-LReLU	down2	3	(256,128,128,3)
res1-2	conv-k3-s1-p1	res1-1	3	(256,128,128,3)
res2-1	conv-k3-s1-p1-LReLU	res1-2	3	(256,128,128,3)
res2-2	conv-k3-s1-p1	res2-1	3	(256,128,128,3)
concat1	concat	(down1, res2-2)	6	(256,128,128,6)

Feature Abstraction Network

Layer Name	Type	Input Name	Channels	Output Size
Input2	Propagated Feature		6	(256,128,128,6)
down2	conv-k3-s1-p1	Input2	6	(256,128,128,6)

While these methods offer not alternative to the proposed transient renderer, they do simulate higher-order bounces beyond the third bounce. The very recent work from Iseringhausen and Hullin [Iseringhausen and Hullin 2020] and Tsai et al. [Tsai et al. 2019] explore efficient methods for three-bounce transient rendering. The method from Iseringhausen and Hullin achieves real-time runtimes for simple scenes with less than a few hundred primitives, but requires seconds of rendering time for more complex scenes. We list rendering runtime comparisons in Table. 1. Specifically, their method takes 1032.2 ms to 128×128 quads into a 128×128 ×192 transient measurement. For the same setting, the proposed method takes only 26.90 ms to render the same mesh into a 256×256×600 transient measurement. While [Iseringhausen and Hullin 2020] is sensitive to the quads numbers, our rasterization-based OpenGL rendering framework supports models with large number of vertices and faces, as shown in Fig. 6 in the main paper, with 52081 vertices and 200018 faces.

The method from Tsai et al. [Tsai et al. 2019] does not present an alternative to the proposed approach. While their approach gradually renders path contributions intertwined with the optimization, rendering 64×64×1200 transient measurement for scenes of comparable complexity to the proposed dataset alone takes tens of minutes, while intertwined with the optimization, it requires more than two hours on a 24-core workstation machine.

3 LEARNED NLOS SCENE REPRESENTATIONS

In this section, we provide additional detail on the proposed network architecture and design choices.

3.1 Spatio-Temporal Feature Extraction

For an RGB transient image with 512 time bins and a spatial resolution of 256×256, an input a tensor of size $(512, 256, 256, 3)$ is fed into the feature extraction network, which immediately applies a convolutional downsampling block. The downsampling block is composed of two branches. The first branch contains one convolutional layer, and the second branch includes another convolutional layer followed by one ResNet block [He et al. 2016] to refine the downsampled features. Each ResNet block contains two convolutional layers, interlaced with one LeakyReLU layer. All convolutional layers have kernel size 3, stride 1, and three output channels (limited by our training hardware memory), except for the first convolutional layer of both branches. These first layers have stride 2 spatially and temporally to immediately compress features in the spatio-temporal domain. The outputs of the two branches are concatenated along the channels, resulting in a final extracted feature of size $(256, 128, 128, 6)$, i.e. $4\times$ smaller in size than the input raw data. Please refer to Fig. 4 for a full architecture description.

3.2 Feature Abstraction

To further abstract and complete hidden scene information, we process the propagated feature the previous stage with an additional volumetric embedding block. In particular, after feature propagation, the volumetric representation is passed through a 3D convolutional layer with kernel size 3, stride 1, and output channel number as 6 without bias parameter. Instead of opting for a larger ResNet block, we initialize the weights such that its output is identical to its input when the training starts. This feature abstraction block is used to further finetune the encoded feature representation before using it for reconstruction or recognition. The output of this block is the final learned volumetric feature representation. Please refer to Fig. 4 for a full architecture description.

3.3 Image and Depth Rendering Network

For completeness, we also describe in detail the architecture for the image and depth rendering network. Please refer to Fig. 6 and Fig. 8 for a full architecture description of image rendering network and depth rendering network, respectively.

4 NETWORK TRAINING DETAILS

As we reason over full transient images of resolution $(512, 256, 256, 3)$, translating to input tensors of size $\approx$ 400 Mb without compression, training NLOS networks is challenging compared to learned methods that reason on 2D images with one to two orders of magnitude reduced memory requirements on the input representation. Although we tailor our network to the NLOS tasks, learning from the large corpus of simulated training data still takes on the order of weeks with modern GPU hardware. During training, we set the batch size to be one and use Adam as the optimization method with $\beta_1 = 0.5$, $\beta_2 = 0.999$ and learning rate 0.0002. We train for 50 epochs on the motorbike data set containing ≈ 6000 training examples. This takes 7 days to reach full convergence on a single NVIDIA TITAN X GPU. We envision dedicated training solutions, such as the NVIDIA DGX training machines, or multi-GPU training to expedite training time for experimentation in the future.

Aiding the stability of the training of all networks, we also found that initializing the first downsampling convolutional layer in the feature extraction network as an average pooling layer over the temporal domain aids the convergence of the network. Intuitively, this allows the network to extract features from downsampled temporal histograms before tailoring the features to the full-resolution details. To further improve stability, we have also experimented with progressive growing [Karras et al. 2017] but did not find substantial benefits for our network architecture.

5 ADDITIONAL RGB, DEPTH AND MULTI-VIEW NLOS IMAGING RESULTS

In this section, we provide additional results for RGB, depth and multi-view predictions. We first show the RGB prediction on car and bike class in Fig. 10 and Fig. 11. The proposed model generalizes well to unseen examples from the same class. Next, by adding a depth rendering network and multi-view supervision, the proposed method allows for joint image and depth reconstruction from multiple viewpoints in an end-to-end fashion. As

Fig. 6. Architecture details of our image rendering network. In the table, "conv-k(a)-s(b)-p(c)-LReLU" represents a convolution layer with an $a \times a$ kernel window, using stride b, padded c values on the boundary by replication, followed by a Leaky ReLU ($\alpha = 0.2$) activation function.

Image Rendering Network				
Layer Name	Type	Input Name	Channels	Output Size
Input3 (down2)	Learned Volumetric Feature		6	(256,128,128,6)
project-index	argmax	down2	6	(256, 128,128,6)
project	maxproject	down2, project-index	6	(128,128,6)
concat2	concat	(project, project-index)	12	(128,128, 12)
slice	slice	concat2	4	(128, 128,4)
up1	upsample	slice	3	(256,256, 3)
same1	conv-k1-s1-p1-LReLU	up1	3	(256,256,3)
same2	conv-k3-s1-p1-LReLU	Input3	3	(128,128,3)
res3-1	conv-k3-s1-p1-LReLU	same2	3	(128,128,3)
res3-2	conv-k3-s1-p1	res3-1	3	(128,128,3)
res4-1	conv-k3-s1-p1-LReLU	res3-2	3	(128,128,3)
res4-2	conv-k3-s1-p1	res4-1	3	(128,128,3)
up2	upsample	res4-2	3	(256,256, 3)
concat3	concat	(up1, up2)	6	(256,256, 6)
same3	conv-k3-s1-p1-LReLU	concat3	6	(256,256, 6)
res5-1	conv-k3-s1-p1-LReLU	same3	6	(256,256, 6)
res5-2	conv-k3-s1-p1	res5-1	6	(256,256, 6)
res6-1	conv-k3-s1-p1-LReLU	res5-2	6	(256,256, 6)
res6-2	conv-k3-s1-p1	res6-1	6	(256,256, 6)
same4	conv-k3-s1-p1-LReLU	res6-2	3	(256,256, 3)
output	add	same1, same4	3	(256,256,3)

illustrated in Fig.12, the model is able to faithfully reconstruct RGB images and depth maps from multiple views for a variety of objects.

6 ADDITIONAL OBJECT DETECTION RESULTS

Detector	Avg. IoU	Airplane	Bench	Cabinet	Chair	Display	Lamp	Speaker	Firearm	Sofa	Table	Phone	Watercraft	Car	Bike
End-to-end	0.73	0.65	0.67	0.80	0.75	0.72	0.69	0.73	0.67	0.73	0.66	0.81	0.65	0.74	0.74
Sequential-Ours	0.69	0.62	0.63	0.79	0.75	0.72	0.66	0.74	0.64	0.70	0.66	0.79	0.61	0.73	0.68
Sequential-F-K	0.67	0.60	0.63	0.78	0.70	0.70	0.62	0.73	0.65	0.68	0.64	0.75	0.63	0.71	0.69

Table 2. Full 2D Detection IoU Comparison. We report average IoU across all 14 classes and IoU for each individual class.

In this section, we provide additional qualitative and quantitative evaluation of the proposed end-to-end detection model. The data set for detection is created by merging the multi-class data set described in the main paper (containing 13 classes) and 500 additional motorbike examples randomly sampled from the motorbike data set. Therefore, the data set in total includes the following 14 classes: airplane, bench, cabinet, chair, display,

Fig. 8. Architecture details of depth rendering network. In the table, "conv-k(a)-s(b)-p(c)-LReLU" represents a convolution layer with an $a \times a$ kernel window, using stride b, padded c values on the boundary by replication, followed by a Leaky ReLU ($\alpha = 0.2$) activation function.

Depth Rendering Network

Layer Name	Type	Input Name	Channels	Output Size
Input3 (down2)	Learned Volumetric Feature		6	(256,128,128,6)
project-index	argmax	down2	6	(256, 128,128,6)
project	maxproject	down2, project-index	6	(128,128,6)
concat2	concat	(project, project-index)	12	(128,128, 12)
slice	slice	concat2	4	(128, 128,4)
up1	upsample	slice	1	(256,256, 1)
same1	conv-k1-s1-p1-LReLU	up1	3	(256,256,3)
same2	conv-k3-s1-p1-LReLU	Input3	3	(128,128,3)
res3-1	conv-k3-s1-p1-LReLU	same2	3	(128,128,3)
res3-2	conv-k3-s1-p1	res3-1	3	(128,128,3)
res4-1	conv-k3-s1-p1-LReLU	res3-2	3	(128,128,3)
res4-2	conv-k3-s1-p1	res4-1	3	(128,128,3)
up2	upsample	res4-2	3	(256,256, 3)
concat3	concat	(up1, up2)	6	(256,256, 6)
same3	conv-k3-s1-p1-LReLU	concat3	6	(256,256, 6)
res5-1	conv-k3-s1-p1-LReLU	same3	6	(256,256, 6)
res5-2	conv-k3-s1-p1	res5-1	6	(256,256, 6)
res6-1	conv-k3-s1-p1-LReLU	res5-2	6	(256,256, 6)
res6-2	conv-k3-s1-p1	res6-1	6	(256,256, 6)
same4	conv-k3-s1-p1-LReLU	res6-2	1	(256,256, 1)
output	add	same1, same4	1	(256,256,1)

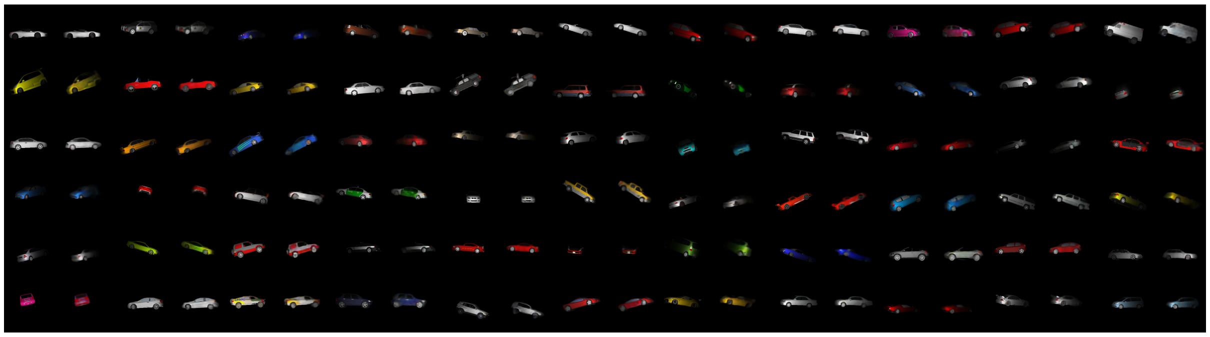

Fig. 10. **Recovery with Intra-Class Variation for the Car Class.** We visualize test outputs from the proposed method trained on car for unseen test samples of the car class with high variety of shape and color across the data set.

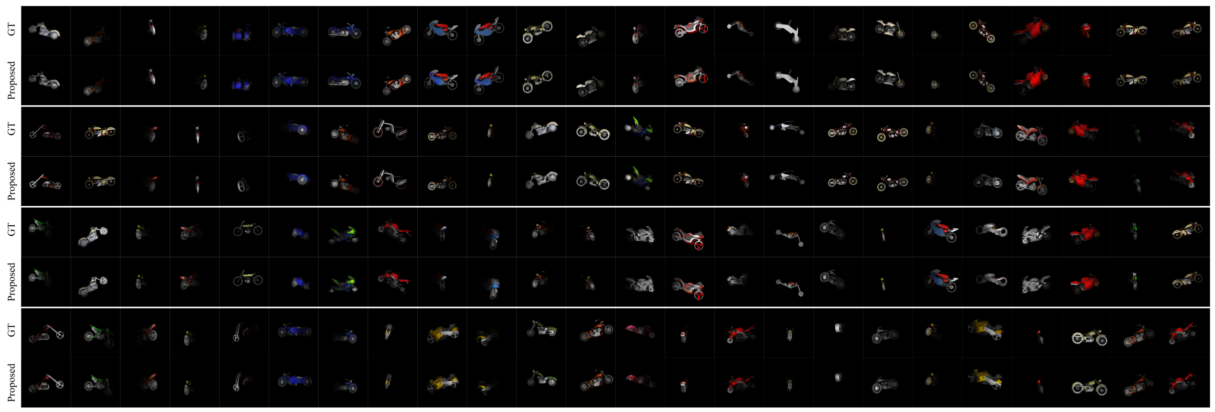

Fig. 11. **Recovery with Intra-Class Variation for the Bike Class.** We visualize test outputs from the proposed method trained on motorbike for unseen test samples of the motorbike class with high variety of shape and color across the data set. The proposed model is able to reconstruct motorbikes with thin structures and various poses. Please zoom into electronic version for details.

lamp, speaker, firearm, sofa, table, phone, watercraft, car and bike. In Tab. 2, we show a comparison to two baseline methods. While the detection head of the end-to-end method accepts an intermediate learned feature (a collapsed 2D feature combined with the index map used for 3D-to-2D projection) as input, the two baselines accept concatenated image and depth map rendered from the proposed reconstruction method (Sequential-Ours) or from F-K (Sequential-Ours). Detectors of the three compared methods all share the similar structure –the network consists of four convolutional layers with stride two and kernel size three, followed by an average pooling layer that extracts a one-dimensional feature of length 512 and a fully-connected layer that predicts five values. The only difference among the three detectors is in the input channel number of the first convolutional layer in the detector to accommodate various input sizes and our feature representation instead of the intermediary image representation of the sequential methods. Each method is evaluated on the average IoU (Intersection over Union) across all 14 classes and IoU for each individual class. We can see that while all methods struggle on classes whose rendered images are too small (e.g., firearm), too big (e.g., cabinet), or too dark (e.g., airplane), the proposed end-to-end detection method outperforms the baselines on almost all classes.

We also visualize additional predictions of the proposed model on our test dataset in Fig. 13. The visualization shows that the data set includes examples with various color, shape, and pose. And the predictions (green boxes) validate that our model predicts for a wide range of bounding box size and aspect ratio. The proposed model is able to learn meaningful features that facilitate downstream tasks such as detection. We hope that in the future, once more rich experimental NLOS datasets become available, researchers will adopt our detection and classification networks as building blocks for complex NLOS scene understanding tasks.

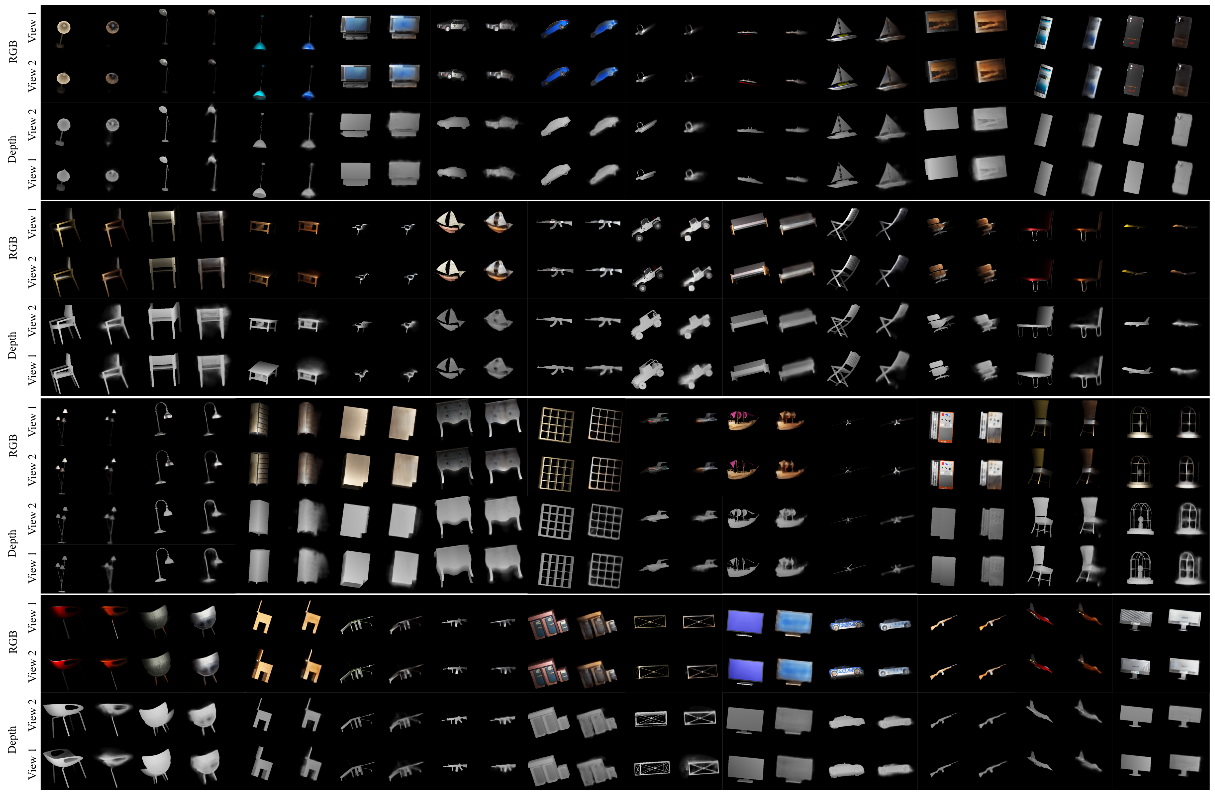

Fig. 12. **Multi-View NLOS RBG-D Recovery.** We show unseen test examples from the NLOS RGB-D and multi-view model trained on the multi-class data set. The proposed model is able to render both intensity and depth map for all thirteen classes from different views. It learns to robustly recover NLOS RGB and depth across all classes with varying topologies and levels of detail.

REFERENCES

Victor Arellano, Diego Gutierrez, and Adrian Jarabo. 2017. Fast back-projection for non-line of sight reconstruction. *Optics Express* 25, 10 (2017), 11574–11583.

Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. 2014. Generative adversarial nets. In *Advances in neural information processing systems*. 2672–2680.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2016. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*. 770–778.

Julian Iseringhausen and Matthias B Hullin. 2020. Non-line-of-sight reconstruction using efficient transient rendering. *ACM Transactions on Graphics (TOG)* 39, 1 (2020), 1–14.

Adrian Jarabo and Victor Arellano. 2018. Bidirectional rendering of vector light transport. In *Computer Graphics Forum*, Vol. 37. Wiley Online Library, 96–105.

Adrian Jarabo, Julio Marco, Adolfo Munoz, Raul Buisan, Wojciech Jarosz, and Diego Gutierrez. 2014. A Framework for Transient Rendering. *ACM Transactions on Graphics (Proceedings of SIGGRAPH Asia)* 33, 6 (nov 2014). https://doi.org/10.1145/2661229.2661251

Adrian Jarabo, Belen Masia, Julio Marco, and Diego Gutierrez. 2017. Recent advances in transient imaging: A computer graphics and vision perspective. *Visual Informatics* 1, 1 (2017), 65–79.

Tero Karras, Timo Aila, Samuli Laine, and Jaakko Lehtinen. 2017. Progressive growing of gans for improved quality, stability, and variation. *arXiv preprint arXiv:1710.10196* (2017).

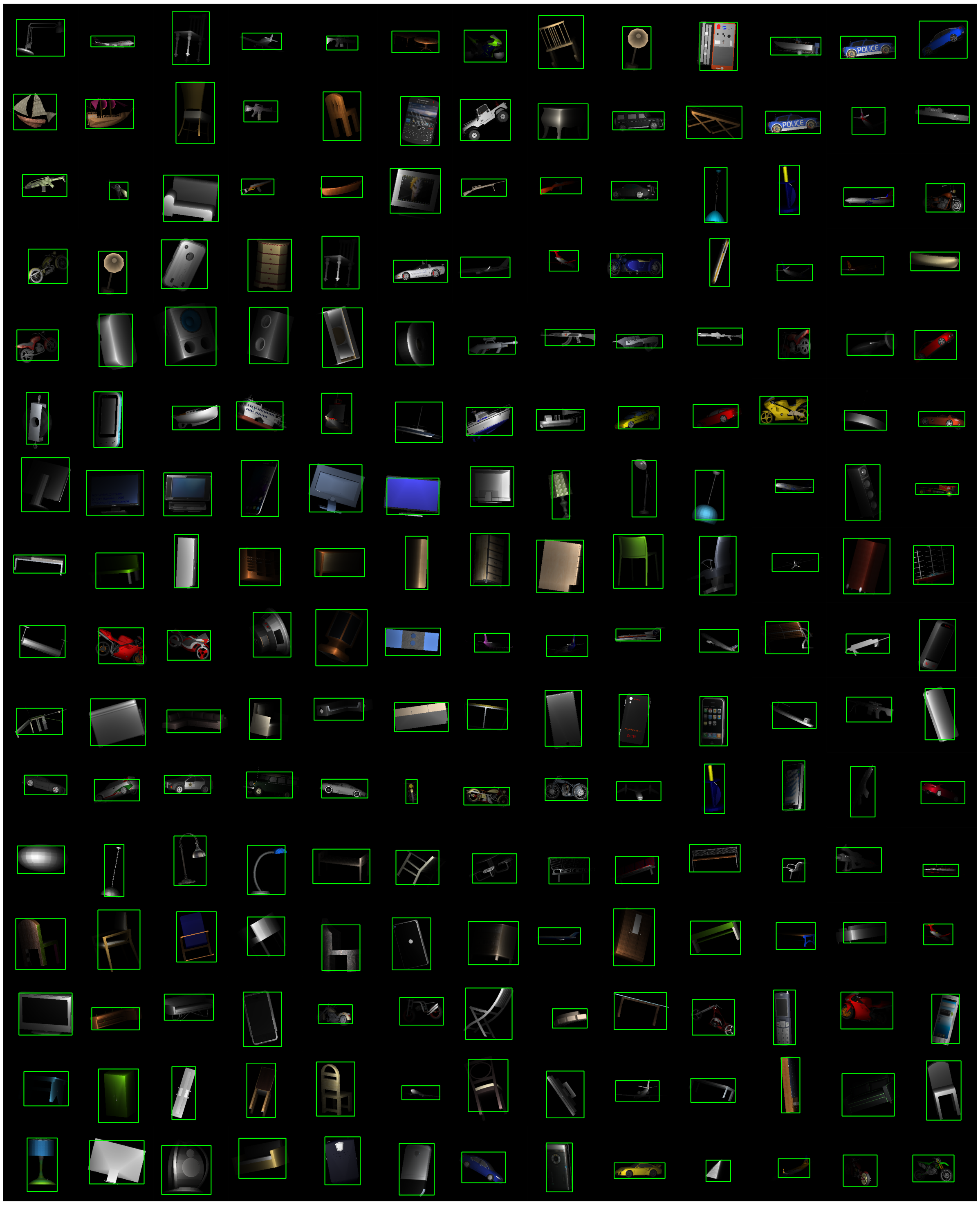

Fig. 13. Additional qualitative results of the end-to-end detection model on our multi-class data set. The model predicts accurate bounding boxes for a wide range of object sizes and classes.

Martin Laurenzis and Andreas Velten. 2014. Feature selection and back-projection algorithms for nonline-of-sight laser–gated viewing. *Journal of Electronic Imaging* 23, 6 (2014), 063003.

David B. Lindell, Gordon Wetzstein, and Matthew O'Toole. 2019. Wave-based non-line-of-sight imaging using fast f-k migration. *ACM Trans. Graph. (SIGGRAPH)* 38, 4 (2019), 116.

Xiaochun Liu, Sebastian Bauer, and Andreas Velten. 2020. Phasor field diffraction based reconstruction for fast non-line-of-sight imaging systems. *Nature Communications* 11 (2020). https://doi.org/10.1038/s41467-020-15157-4

Xiaochun Liu, Ibón Guillén, Marco La Manna, Ji Hyun Nam, Syed Azer Reza, Toan Huu Le, Adrian Jarabo, Diego Gutierrez, and Andreas Velten. 2019. Non-line-of-sight imaging using phasor-field virtual wave optics. *Nature* (2019), 1–4.

Matthew O'Toole, David B. Lindell, and Gordon Wetzstein. 2018. Confocal Non-line-of-sight imaging based on the light cone transform. *Nature* (2018), 338–341. Issue 555.

Adithya Pediredla, Ashok Veeraraghavan, and Ioannis Gkioulekas. 2019. Ellipsoidal Path Connections for Time-gated Rendering. *ACM Trans. Graph. (SIGGRAPH)* (2019).

Alec Radford, Luke Metz, and Soumith Chintala. 2015. Unsupervised representation learning with deep convolutional generative adversarial networks. *arXiv preprint arXiv:1511.06434* (2015).

Olaf Ronneberger, Philipp Fischer, and Thomas Brox. 2015. U-net: Convolutional networks for biomedical image segmentation. In *International Conference on Medical image computing and computer-assisted intervention*. Springer, 234–241.

Chia-Yin Tsai, Aswin C Sankaranarayanan, and Ioannis Gkioulekas. 2019. Beyond Volumetric Albedo–A Surface Optimization Framework for Non-Line-Of-Sight Imaging. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 1545–1555.

A. Velten, T. Willwacher, O. Gupta, A. Veeraraghavan, M.G. Bawendi, and R. Raskar. 2012. Recovering three-dimensional shape around a corner using ultrafast time-of-flight imaging. *Nature Communications* 3 (2012), 745.