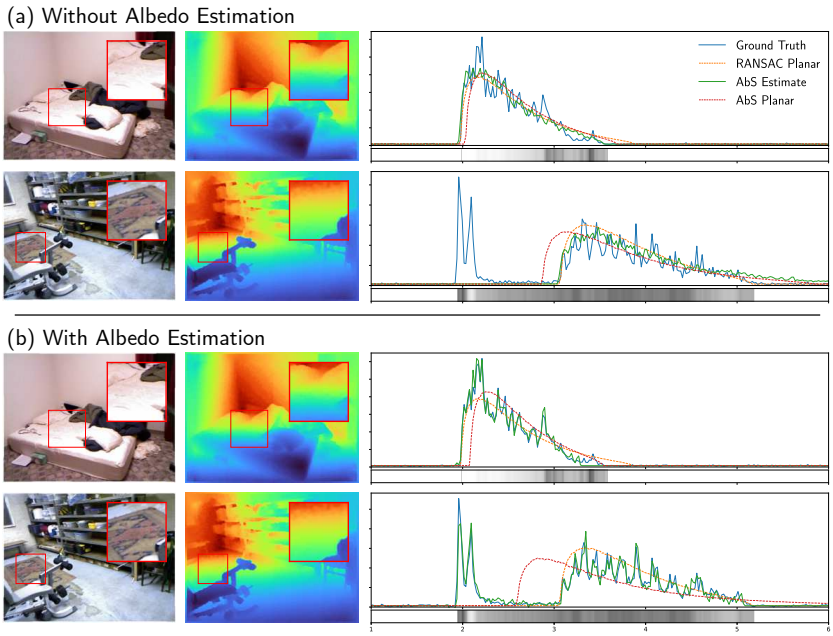
Supplementary Document for "3D Scene Inference from Transient Histograms"

Sacha Jungerman, Atul Ingle, Yin Li, Mohit Gupta
University of Wisconsin-Madison
{sjungerman, ingle, yin.li, mgupta37}@wisc.edu

Supplementary Note 1: Inference from Scenes with Non-Uniform Albedos



Suppl. Fig. 1. Robustness to Albedo Variations. We again show a few NYUv2 patches and their accompanying transients. We show the transients the AbS method has fitted to when not estimating scene albedo (a) and when jointly estimating scene albedo (b).

Robustness to albedo variations: We observe that, in most cases, both the theoretical and AbS approaches are robust to real-world albedo variations, and

slight geometry variations, despite making no explicit attempt at modeling non-uniform albedos. This is seen in both sample scenes in Suppl. Fig. 1(a): the dashed transient are nearly identical which indicates good parameter estimation despite challenging albedo variations.

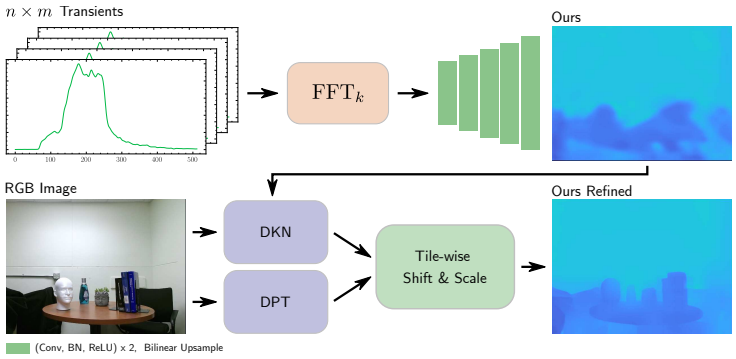
We attribute this robustness to albedo variations to the implicit albedo averaging that is performed through binning. Each transient bin will receive contributions from all scene patches at a specific distance from the sensor. The perceived albedo at that bin will thus effectively be the average albedo of all those scene patches. This is illustrated in Suppl. Fig. 1(a) as the horizontal greyscale color bar under each transient. While this per-bin albedo average could have sharp transitions for a carefully crafted, adversarial scene, we observe that when imaging real scenes, albedos are smooth over consecutive bins.

Estimating scene albedo: We can try to recover the scene's point-wise albedo by augmenting our AbS approach with a texture that stores the scene's albedo. Gradients with respect to this texture can then be computed, and the texture optimized. However, due to the albedo binning process described above, the best we can aim to recover is the average albedo per bin. The optimized albedo-texture cannot contain the scene's albedo as a single transient does not contain enough spatial information.

With no added regularization, the albedo-texture map converges to what looks like salt and pepper noise. While this albedo map is of no immediate use, it can help the AbS method converge to better plane estimates. In Suppl. Fig. 1(b) we show sample results for which the scene's albedo was estimated. Notice that when jointly estimating plane parameters and albedo, the AbS transient (green) can more closely fit the observed one (blue). As seen in the last row, this can cause over-fitting issues and lead to estimated parameters that describe a plane that best fits the observed geometry instead of the planar part of the scene.

Supplementary Note 2: Deep Model and Training Details

Model Architecture:



Suppl. Fig. 2. Architecture of Model. Diagram showing our model architecture and the refinement steps. We use two existing methods [16,33] (seen in blue) to refine our depth predictions.

Suppl. Fig. 2 shows the full architecture of our model and the refinement steps. Transients are first encoded as their first k Fourier coefficients and then passed to our base network which consists of a few blocks, each containing two convolutions followed by bilinear upsampling by a factor of $2\times$. For the 20×15 tiling, we use $k = 16$ and five blocks. For the 4×3 tiling, which is $160\times$ smaller than the output dimension, we use six blocks with the first one upsampling by a factor of $5\times$, and we use $k = 4$.

The refinement step consists of two off-the-shelf networks: DKN [16] and DPT [33]. The former was finetuned to refine our depth estimate using the RGB image as an added input, while the latter is pretrained and only uses the RGB image. We then use the depth map produced by DKN as a guide to warp the DPT depth map. We first tile each depth map, then compute the scale and shift that will minimize the $\mathcal{L}_2$ loss between them. This scale and shift tensor, of shape $n \times m \times 2$, is then bilinearly interpolated to the full resolution of 640×480 and applied to DPT's output. This enables our approach to gain from the spatial details present in DPT's output but not suffer from its low accuracy. The tiling grid used by the tile-wise matching process can be adjusted to yield results with greater details or greater accuracy. For our experiments, it matched the transient tiling grid.

Loss Function: Traditionally, the loss function used for regression problems is the squared Euclidean distance, or the $\mathcal{L}_2$ norm $\mathcal{L}_2(\tilde{y}-y) = ||\tilde{y}-y||_2^2$ between the predictions $\tilde{y}$ and ground truth y. However, using this simple $\mathcal{L}_2$ loss function leads to blurry depth reconstructions [18,10,17]. To alleviate this, we use the reverse Hubert loss (a.k.a: BerHu loss), $\mathcal{B}$, as introduced by [30,45].

$$\mathcal{B}(x) = \begin{cases} |x| & |x| \leq c, \\ \frac{x^2+c^2}{2c} & |x| > c. \end{cases} \tag{S1}$$

This loss effectively acts like an $\mathcal{L}_1$ loss for small errors and an $\mathcal{L}_2$ for large ones. It has the advantage of penalizing large errors as much as an $\mathcal{L}_2$ loss would while still providing a large enough gradient for smaller errors. This yields a sharper final prediction and a lower final error. Further, the hyper-parameter c can be used to tune the BerHu loss and change where the $\mathcal{L}_1/\mathcal{L}_2$ switch occurs.

Implementation details: Unless otherwise noted, all experiments used a BerHu loss with tuning parameter $c = 0.1$ and ran on a single GeForce RTX 3090 or a GeForce RTX 2080 Super for up to 500 epochs. We used the Adam optimizer, with a learning rate of 0.01. Transient histograms are simulated from the ground-truth data using Eq. (1) where we used the green channel of the RGB images as a proxy for scene albedoes.

Supplementary Note 3: Definition of Metrics Used

The metrics used to evaluate our results follow that of previous work [7] and are defined as follows:

– Absolute Relative Error (AbsRel):

$$\frac{1}{n}\Sigma_n \frac{|\hat{y}_i - y_i|}{y_i}$$

– Root Mean Squared Error (RMSE):

$$\sqrt{\frac{1}{n}\Sigma_n \|\hat{y}_i - y_i\|_2}$$

– Average Log Error (Log10):

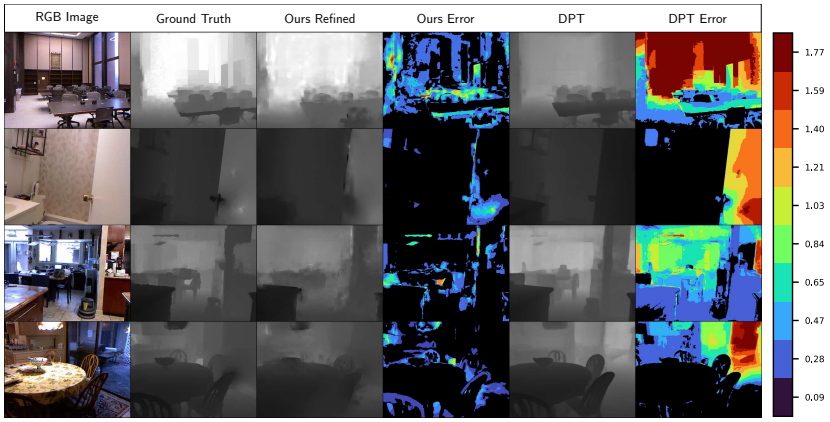
$$\frac{1}{n}\Sigma_n |log_{10}(\hat{y}_i) - log_{10}(y_i)|$$

– Threshold Accuracy ($\delta < thr$):

$$\max\left(\frac{\hat{y}_i}{y_i}, \frac{y_i}{\hat{y}_i}\right) = \delta < thr$$

where $\hat{y}_i$ is a pixel in the predicted depth map $\hat{y}$, y_i is a pixel in the ground-truth depth map y, and n is the number of valid pixels. Again, prior work has only used 1.25^i for $i = 1, 2, 3$ as the threshold for the threshold accuracy metric. These correspond to 25%, 56%, and 95% error respectively. We have introduced the tighter thresholds of 1.05^i for $i = 1, 2, 3$ to better characterize approaches. These correspond roughly to 5%, 10.25%, and 15.76% error.

Supplementary Note 4: Comparison with MDE

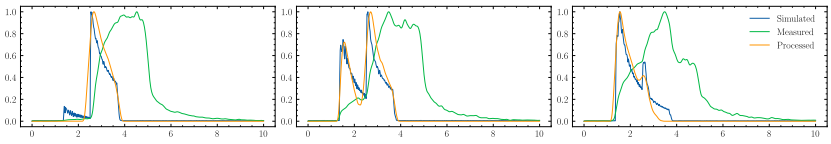


Suppl. Fig. 3. Results and $\mathcal{L}_1$ Error on NYUv2 Dataset. From left to right this figure shows the RGB image, ground truth depth, our refined for the 20×15 grid, the absolute error of our approach, a MDE result from DPT [33], and their absolute error.

In Suppl. Fig. 3, we show results from our method and a leading MDE method. While visually DPT results look marginally better, they often have large absolute depth errors. Even for small indoor scenes of less than 10 meters, DPT can produce absolute errors of the order of 2 meters. For these reasons our method is better suited for applications that require precise depth.

Supplementary Note 5: Additional Hardware Results

While our hardware setup, shown in Fig. 6(left), has a bin-width of 16 ps, our models were trained with transients having 512 bins over an unambiguous depth of 10 m. Instead of retraining our models, we preprocess the measured transients by re-binning them accordingly and subsequently applying a small amount of Gaussian smoothing ($\sigma = 2$ bins).

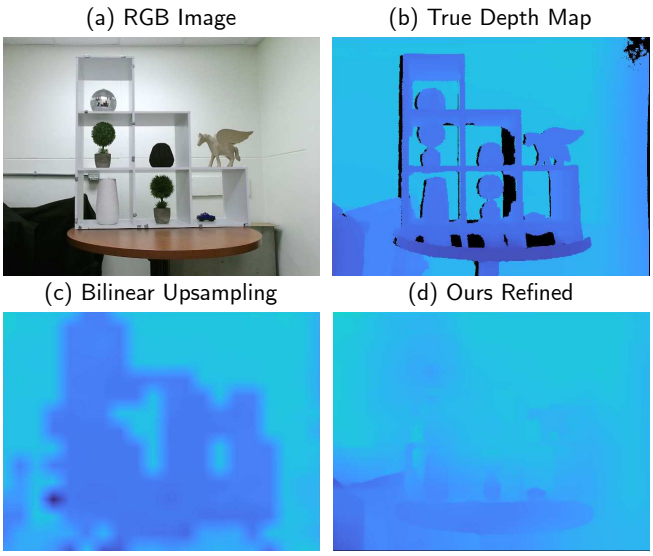


Suppl. Fig. 4. Transient Histogram Preprocessing. We show example measured, simulated and processed transients.

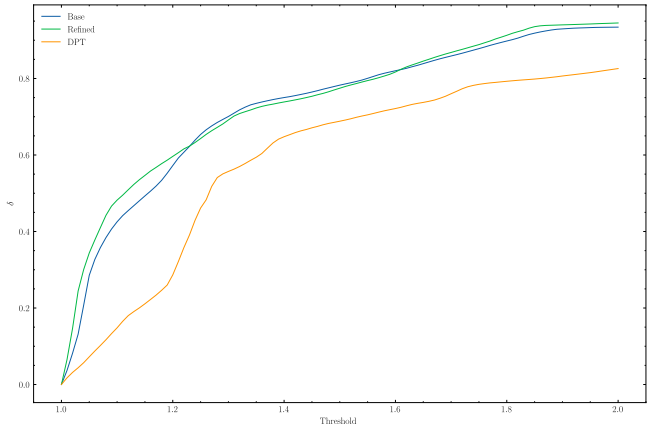
An assumption which was made while training the network on simulated data is that the laser pulse could be modeled as a Dirac delta pulse (with respect to the bin-width), however, the hardware setup uses a laser with a pulse width of 6 ns which corresponds to about 375 bins. Not only is the pulse width much larger than in simulation, but it is also not the same shape: all measured transients are effectively convolved with an unknown kernel.

To estimate this kernel, we first simulated transients from the ground truth depth, as measured by the Kinect, using a laser pulse of larger width. We then trained a shallow fully connected neural network to map the measured transients to the simulated ones using only 20% of the acquired data as a training set. This small model was trained with an Adam optimizer (with default parameters) and a Kullback-Leibler divergence loss. Examples of the measured transients, simulated ones, and neural network processed ones are shown in Suppl. Fig. 4.

Results using the processed transients can be seen in Fig. 6(right) and Suppl. Fig. 5. Finally Suppl. Fig. 6 shows the accuracy of our method (with and without refinement) as well as DPT's [33] on the scene shown in Suppl. Fig. 5. Our results, on this scene, achieve better threshold accuracy overall, especially for lower thresholds. With more data this preprocessing scheme would not be needed, as our model could be finetuned to adapt to the characteristics of the hardware setup in use.

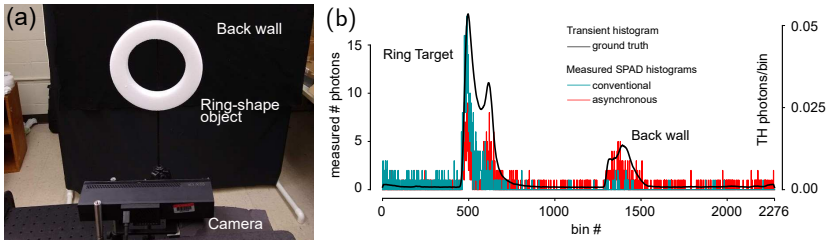


Suppl. Fig. 5. Results with experimental setup. (a) We image a table-top scene with a wide range of albedos and textured objects. (b) The true depth map captured using a Kinect v2. (c) Since the capture is a low-resolution 20×15 grid, simple peak-finding-based depth map provides no depth details. (d) Using the RGB image as a guide, our method generates a high-resolution depth map.



Suppl. Fig. 6. Threshold Accuracy on Real Data. We show the continuous threshold accuracy for our method without refinement ("Base"), with refinement ("Refined"), and DPT [33] for all thresholds in the range $[1, 2]$. Results are reported on a single scene in Suppl. Fig. 5.

Supplementary Note 6: Dealing with Pile-up Distortions



Suppl. Fig. 7. Measuring a transient histogram using a SPAD. (a) The test scene consists of a ring-shaped target with a background wall. (b) Asynchronous acquisition using a SPAD reliably captures later photons by avoiding pile-up distortion seen in conventional acquisition.

If the total photon flux incident on the SPAD is high enough, photon detections in earlier histogram bins prevent later bins from capturing photons. This causes pile-up distortions that follow a characteristic exponentially decaying shape. This pile-up can be due to ambient light or due to laser photons reflecting from a high albedo target. Suppl. Fig. 7 shows a histogram with and without pileup for the ring-shaped target with a wall in the background. Observe that the pile-up distorted histogram has fewer total photon counts in later photon bins. We use an *asynchronous* acquisition scheme [11] that counteracts pile-up distortion by operating the SPAD in a free-running mode. The SPAD pixel enters a dead-time immediately after each photon detection event and is reset to capture the next photon after the dead-time period ends. By capturing raw timestamps for each photon detection and laser pulse, photon timestamps can be re-synchronized to the periodic laser source and a histogram free from pile-up distortion can be captured.