Ghosts in the Point Clouds: De-glaring LiDAR in the Transient Domain

Supplementary Material

1. Implementation Details

We captured scenes at a per-frame exposure of 1.16μs for 50 frames. The reason for capturing many frames with a low exposure is a constraint on the dynamic range of our sensor due to photon count overflow. The ADAPs sensor used in this study cannot store more than 12 bits of photon counts per time bin, per exposure. If more photons are detected in a single time bin, the measured histogram will be clipped to this maximum value. This poses an issue because during capture of scenes with severe pile-up we will have ambiguity with points that exceed this limit and we cannot accurately measure variance of time of flight as required by our pile-up correction method.

2. Glare Spread Function Details

Measurement To measure the GSF, we turned off the laser of our LiDAR and used an IR flashlight at the same wavelength (940nm). This allowed us to avoid the quantization limit of our sensor by distributing the energy over time, giving us a more accurate measure of the GSF. It additionally allowed us to remain well under the saturation limit of our sensor and avoid non-linear behavior. By using a baffle on the end of our flashlight, we could isolate the internal multipath bounces and avoid noise from external multipath bounces in the measurement environment. A small aperture on the end of the baffle was used when measuring the GSF to ensure that light from the flashlight was imaged onto at most one pixel in the array. This also ensured that the intensity was low enough to not completely oversaturate our sensor (causing pixels to "shutoff"). To capture any spatial variation of the GSF, we repeat this process to capture GSFs at 49 locations within the LiDAR FOV. The setup for capturing the GSFs can be found in Fig. 1.

Processing An analysis of our GSF characterization efforts is shown in Fig. 2. Because our LiDAR is 1D vertical scanning (with six rows of pixels switched on at a time), there is a correlation between the transmitted pulse and the received transient; this is why blooming artifacts appear as horizontal bands spanning six rows. Our method for measuring the GSF does not produce this pattern because the light is constant (not pulsed), and not scanned in sync with row activations in the receiver array. Thus, to mimic the effect of the scanning mechanism we only used a six-pixel wide horizontal band of the measured GSFs centered around the aggressor as depicted in Fig. 2. To normalize the GSF consider the unnormalized (measured) GSF $\mathbf{a}$ and let $N = \sum_i a_i$ be the total number of photon counts in $\mathbf{a}$ and α, the outscatter ratio, be defined as:

$$\alpha = 1 - \frac{a_0}{N} \tag{1}$$

where a_0 is the number of photon counts in the pixel that contains the GSF peak. The normalized GSF is given by:

$$\hat{\mathbf{a}} = \begin{cases} 0 & \text{at GSF peak pixel} \\ \frac{a_i}{\alpha N} & \text{elsewhere} \end{cases} \tag{2}$$

We further weight the GSF by a distance dependent factor

$$e^{w||p_0 - p_1||_2} \tag{3}$$

where p_0 is the coordinate location of the GSF peak pixel and p_1 is the pixel being weighted. We set $w = .01$.

3. DSP Pipeline

Our approach follows a typical digital signal processing pipeline that might be found in a commercial SP-LiDAR (though these are typically proprietary so details may vary). An overview of our pipeline can be found in Fig. 3. The details of peak finding and echo computation is given below.

Peak Finding We first locate peaks by taking the top n highest count bins after convolving the signal with a kernel. This kernel was measured from a scene containing a flat, low-reflectance surface to avoid nonlinearities in the measurement.

Echo Computation We take these n peaks and compute the first three moments around some fitting window: total measured counts, mean time of flight, and variance of the time of flight. With these we can start to consider pile-up.

4. Pileup Correction

Photon pile-up occurs when incident flux is too high, leading to photon arrivals within a SPADs deadtime that are not detected. This is a non-linear effect which poses a significant issue because our method assumes a linear relationship between the intensity of incident light and the number of measured photon counts.

4.1. The Problem of Pileup Distortion

Retroreflectors cause extreme pile-up because of the intensity of the return. The glare component, being significantly weaker, does not see the same pile-up.

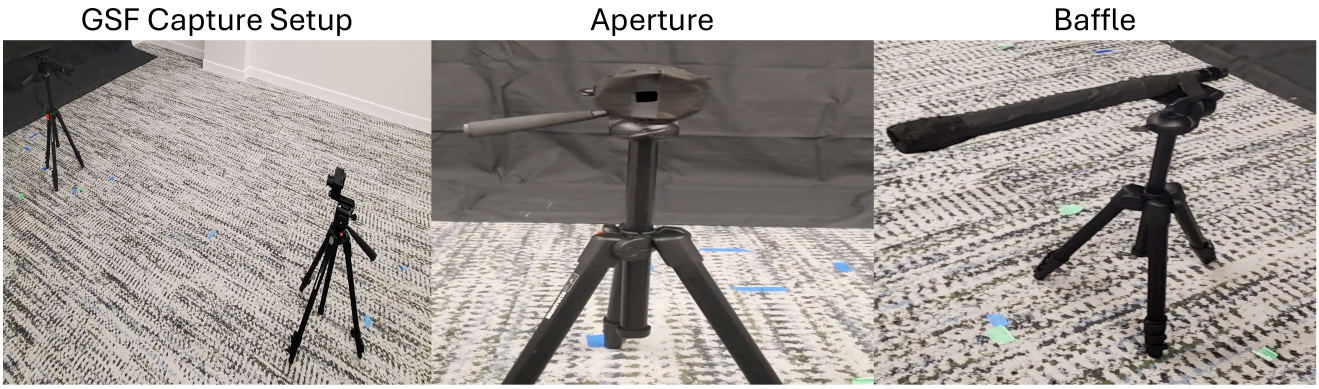

Figure 1. Our setup for capturing the GSFs. We include an image of the full setup as well as closeups of the baffle and aperture size. This was recreated but is roughly the same distance apart as during the GSF captures. The same baffle and aperture was used during capture.

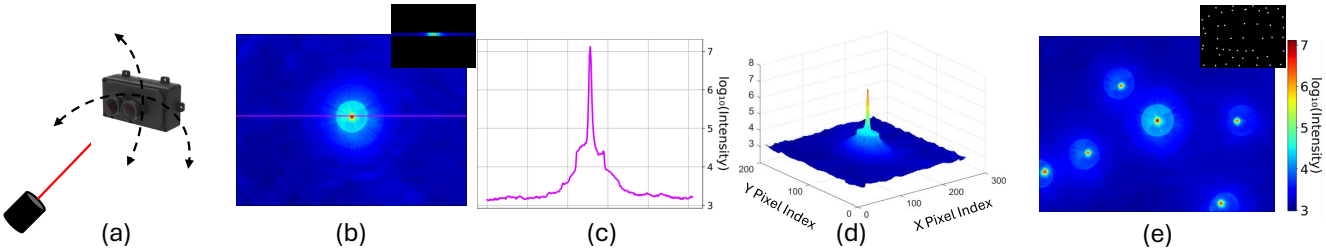

Figure 2. (a) Animation depicting our GSF acquisition setup. We used an IR flashlight with a small aperture and baffle to generate bright points in the LiDAR field of view (FOV). To measure different locations we kept the flashlight at a fixed location and pivoted the LiDAR. (b) A single measured GSF positioned at the center of the LiDAR FOV. The cropped inset depicts a horizontal band of the GSF. In practice this band was used as a GSF to model the behavior of the scanning system of the LiDAR. (c) Horizontal cross section of the GSF depicted in (b). Intensity is taken after background subtraction. (d) Log-scale surface plot of GSF. (e) Depiction of the spatial variation in GSF shape. We recorded GSFs at 49 locations in the LiDAR FOV (all locations shown in inset).

Pile-up correction is a topic of current research and there are methods to correct for it. Coates correction [2] is one common method to address pile-up. Another more recent method can be found in [5]. The issue with both of these methods is they only correct pile-up in the scenarios where some photons are detected in later bins, they assume there is a bias that shifts the peak, but some parts of the signal in bins later than the peak still exist. In the scenario where pile-up is so extreme that no photons appear in later bins, these methods do not work.

One way to avoid extreme pile-up and further mitigate the overflow saturation as previously mentioned is to use a neutral density filter. However, we found to effectively mitigate pile-up effects with this approach we required an extremely strong filter with a strength of OD 4.0 to attenuate the signal. This is not practical in a real deployment scenario as the signal is too weak in even moderately noisy conditions to rise above the noise floor. Thus, we provide a method to mitigate pile-up that utilizes a pile-up corruption forward model to populate lookup tables which can then be used to correct the range-walk errors and intensity underestimation in extreme pile-up conditions.

4.2. Correction Method

To build these lookup tables lets first discuss the forward model for pileup corruption.

Forward Model Consider some normalized transmitted signal $\vec{\lambda}^s \in \mathbb{R}^T$ for T timing bins such that $\sum_i^T \vec{\lambda}_i^s = 1$ with some background noise $\vec{\lambda}^b \in \mathbb{R}^T$ where $\lambda_i^b = \frac{1}{T}$, i.e. $\sum_i^T \lambda_i^b = 1$. Thus the incident signal is just a linear combination of the transmitted signal with the background:

$$\vec{\lambda} = \alpha\vec{\lambda}^s + \beta\vec{\lambda}^b = \alpha\vec{\lambda}^s + \frac{\beta}{T} \tag{4}$$

where α is the number of signal photons detected and β the number of background photons detected ***per-pulse*** when summed over all time bins.

Because counts follow a poisson distribution under the assumption there is no pileup we can say that the probability of detecting ≥ 1 photon in bin i is $p_i = 1 - e^{-\lambda_i}$ where $\lambda_i = \alpha\vec{\lambda}_i^s + \frac{\beta}{T}$. Thus, if we consider the effect of some dead-time D (given in number of bins) we can compute the

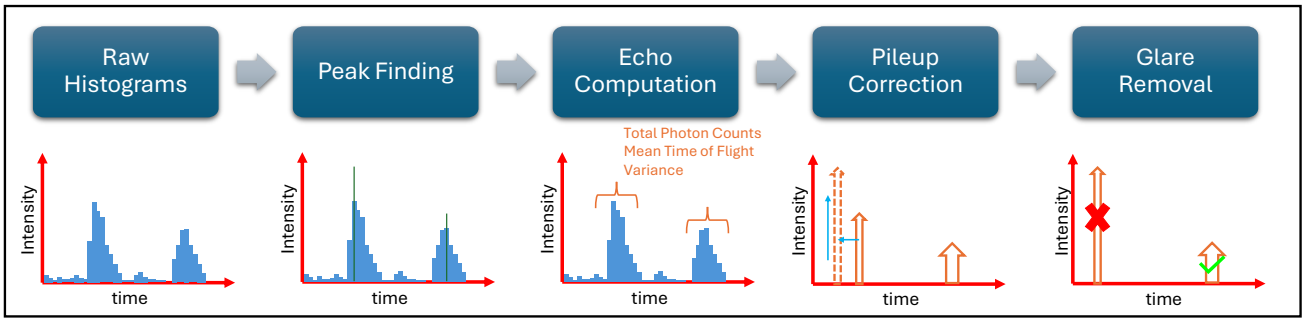

Figure 3. Our DSP pipeline is typical of one likely found in commercial SP-LiDAR devices. Starting with raw histograms, peak finding is done to determine echos. We then use the total photon counts, mean time of flight and time of flight variance of a fitting window around the peak to perform pileup correction in the echo space. Finally, we perform glare removal as described throughout the paper.

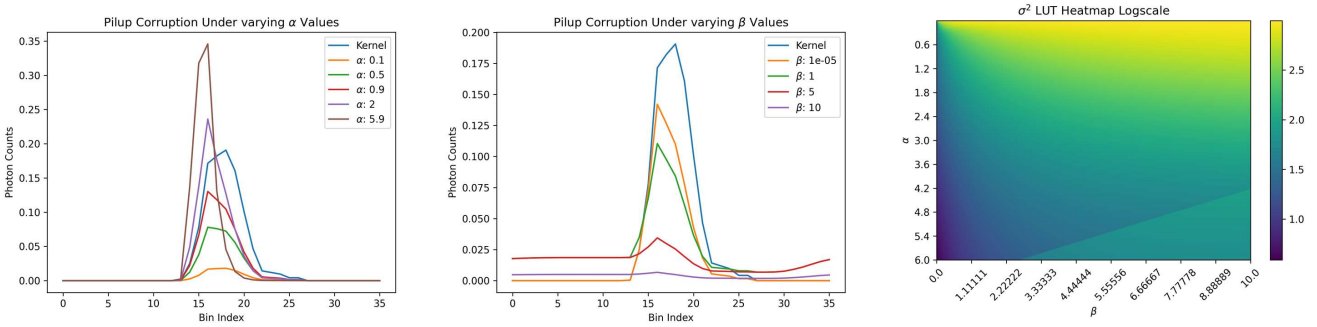

Figure 4. This demonstrates the forward model as alpha and beta are varied and gives a heatmap for the variance lookup table.

$\mathbb{E}$[detections in bin i] per pulse by:

$$q_i(\alpha,\beta,\lambda) = p_i \prod_{j=(i-D-1) \bmod T}^{(i-1) \bmod T}(1-p_j) = (1-e^{-\lambda_i}) \prod_{j=(i-D-1) \bmod T}^{(i-1) \bmod T} e^{-\lambda_j} \quad (5)$$

To improve the numerical stability we compute the logarithm of Eq. (5) as:

$$\ln[q_i(\alpha,\beta,\lambda)] = \ln(1-e^{-\lambda_i}) - \sum_{j=(i-D-1) \bmod T}^{(i-1) \bmod T} \lambda_j \quad (6)$$

Generate Lookup Tables We then create lookup tables for the 0th, 1st and 2nd moments of a pileup corrupted signal which we will refer to as LUT_γ LUT_μ, LUT_{σ^2} respectively. To do this first we start with a pileup free signal, λ_k, which we measured using a dark background.

We then compute $\hat{q}_i = q_i(\alpha,\beta,\lambda_{k,i})$, the pileup corrupted waveform, for evenly spaced $\alpha \in [0,a]$ and $\beta \in [0,b]$ and compute the LUT entries as $LUT_\gamma[\alpha,\beta] = \sum_j \hat{q}_j$ $LUT_\mu[\alpha,\beta] = \mu_\lambda - \mu_q$ where μ_q and μ_λ are the means computed over some fitting window j around the peak of the pileup corrupted waveform and the representative waveform (kernel) respectively. $LUT_{\sigma^2}[\alpha,\beta] = \sigma^2_{\hat{q}}$ where the variance is computed over some fitting window around the peak of the pileup corrupted waveform. See Fig. 4 for the effects of varying α and β as well as a heatmap of the LUT_{σ^2}. See Fig. 5 for a more detailed view of the LUT_{σ^2} and LUT_γ under varying α and β.

Pileup Correction Once we have these lookup tables we can correct for pileup in a measured signal λ_m. First, consider a typical signal processing pipeline for SP-LiDARs where we would first perform some kind of peak detection and get n peaks/pixel (referred to as "echos" in the literature). For each of these echos we can compute the statistics (total energy, variance and mean) of some fitting window around the peak (typically the same size as what is used in the LUT creation) to get some $\gamma_m, \sigma^2_m, \mu_m$.

We compute the average measured background β_m as the mean of the last 557 bins (note that with the limited depths in the scene these bins only contained background counts). With β_m and σ^2_m we can first find the corresponding row in our lookup table $LUT_{\sigma^2}[:,\beta_m]$ and find the closest fitting α

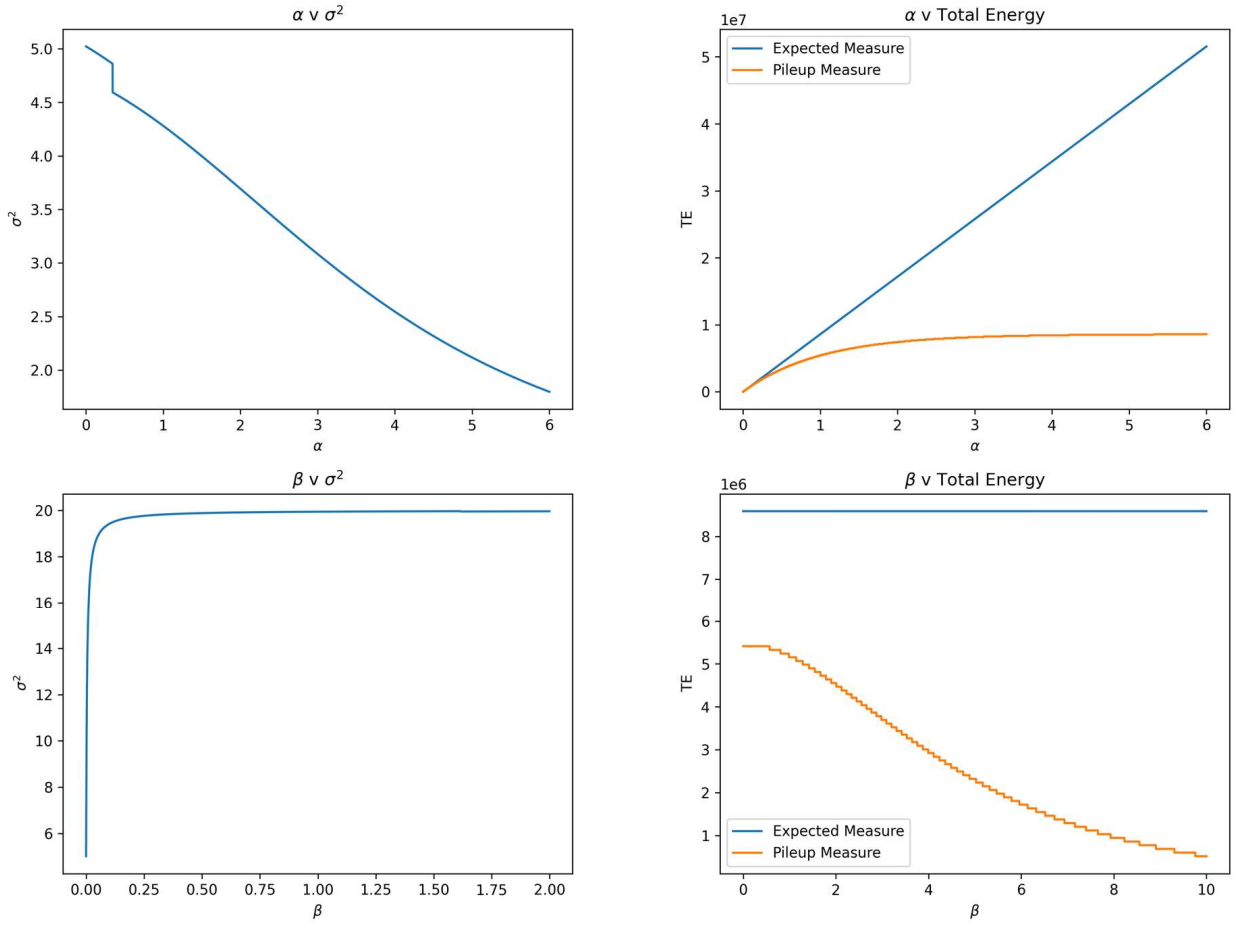

Figure 5. The top row shows the change of total energy and variance as alpha is varied. Two things to note here are that as alpha goes up, variance decreases. The same happens with total energy, we expect linear behavior as in the blue curve, but due to pileup and dead-time effects we get nonlinearities as depicted in the orange curve. Our pileup correction method recovers this linear behavior. For the bottom row variance increases as background counts begin to dominate the signal and energy decreases due to dead-time effects. Photon counts in the we are computing the energy on are lost due to background photons causing a deadtime to occur before the signal reaches the sensor.

(which we will call α_m) using interpolation or some search method like binary search. With β_m, α_m we can then correct the mean shift (Fig. 6) by

$$\mu'_m = \mu_m + LUT_\mu[\alpha_m, \beta_m] = \mu_m + \mu_\lambda - \mu_q \quad (7)$$

and the total energy by

$$TE'_m = \alpha \cdot (\#\text{pulses}). \quad (8)$$

Doing this for each of the n echos gives us n pileup corrected echos and fits into traditional DSP pipelines. We will also use $\gamma' = LUT_\gamma[\alpha_m, \beta_m]$ for computing expected glare. See Fig. 7 for a comparison of our method with and without pileup correction.

5. Our Method

Once we have echos that are pile-up corrected we can move to the glare removal part of our pipeline.

Band Size While in theory (according to provided documentation) a 6 row band would be optimal, we found that in practice a larger band size worked better so we extended this to 17 rows.

EigenCWD Details To address the spatial variance of the GSFs, one could naively interpolate the remaining GSFs, this however would require a massive 49152x49152 array which would be computationally expensive. To get around this we took inspiration from [7]. Instead of interpolating

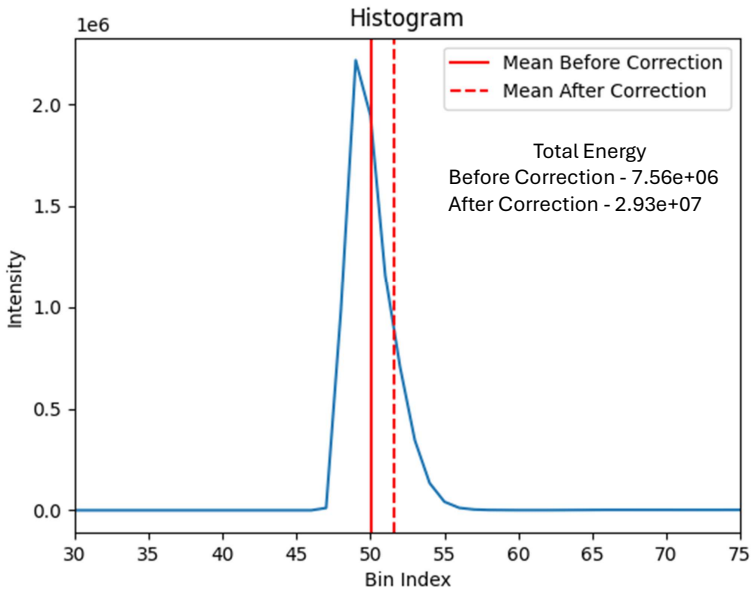

Figure 6. This shows the histogram under pileup corruption from a real capture with a stop sign. This is still with a relatively strong ND filter, with no ND filter the pileup may be too high for this method to work. You can see though that there is a mean shift and energy correction to this pileup corrupted waveform.

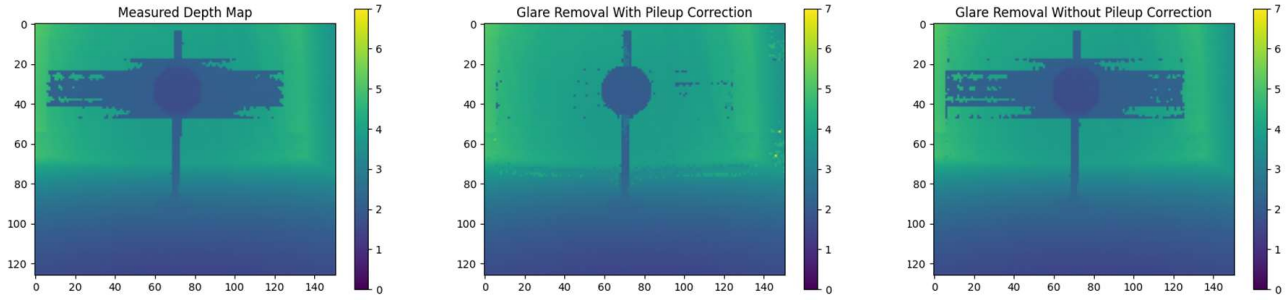

Figure 7. Here is a demonstrated use case of this method. The middle image depicts glare correction without pileup correction and the right image with pileup correction. As you can see points around the stop sign are not corrected due to the distortion caused by pileup.

the array, we store the weights (computed by the inverse square of this distance to an unmeasured point) for each of the 49 GSFs giving us a much more compact 49152x49 size array. We then computed the expected glare at some pixel u, v coming from another pixel at u', v' using a normalized weighted sum of the measured GSF, $\widehat{GSF}$, as

$$\mathbb{E}[G_{(u,v)}] = \widehat{GSF}_{(u',v')} \cdot \gamma'_{(u',v')} \tag{9}$$

where γ' is from Sec. 3

Because this weighted average requires shifting the GSFs, the borders may be unmeasured, to account for this, rather than zero padding, we expanded the borders using the same distance weighted exponential as in Eq. (3) with the base value determined by the pixels at the edge of the pad region.

Background Determination We computed the background the same as in Sec. 3 but we further set a minimum background of 53 photons to account for higher background counts in earlier bins due to effects such as afterpulsing.

Depth Determination After computing the negative log liklihood, one may notice that echos with measured counts well under the expected glare are treated similarly to points

with measured counts well above the expected glare. To solve for this we zero out points that are below the expected glare. Because we also zero out points that are below 5σ of the expected background, in the event that all peaks are zero'd out, we instead retain peaks that are below the expected glare and weight them by the sigmoid of $\frac{\text{counts}_{measured} - \mathbb{E}[Glare]}{T}$ with $T = 90$. The echo with the highest confidence is then selected as the depth.

6. Retroreflector Analysis

Early on in this work we examined how different retroreflector characteristics effected glare. We looked at retroreflector sizes, distances and colors.

Size The size of the retroreflector directly contributes to the extent of the glare. Each pixel that contains a retroreflector element not only contributes to the glare of pixels outside the retroreflector but also other adjacent pixels on the retroreflector. This is part of the reason pile-up was so problematic. In theory, every scene element would contribute some glare, in practice, elements that do not cause strong returns have a low enough probability of causing glare that it is usually not detected. However, even retroreflective elements that spanned a single pixel we noticed caused some glare in adjacent pixels demonstrating the high intensity of these returns.

Distances Distance is less interesting, as it simply follows the inverse square law of intensity falloff. But again, distance is important when considering the need to mitigate pile-up.

Color We tried multiple colors of retroreflectors (yellow, red, orange, white) and multiple colors were demonstrated in our results. We found color had negligible effects and thus we did not perform an ablation study for this.

7. Photographic De-glare Method

7.1. Pileup-free Measurement Model

When pileup is minimal, measurements collected by a LiDAR with a camera-like receiver are well modeled by Eq. 2 of the main text. If the GSF is known, the uncorrupted datacube x can be recovered from measurements y by solving a linear inverse problem. Furthermore, because the GSF is effectively time-independent, we can solve for each time slice of the datacube independently. This allows us to reduce the large problem of inferring a three-dimensional data volume from three-dimensional measurements, to a set of smaller, two-dimensional image recovery problems.

We write down a discrete version of Eq. 2 in the main text:

$$\mathbf{y}_t = \mathbf{A}\mathbf{x}_t + \eta \tag{10}$$

Here $\mathbf{y}_t$, $\mathbf{x}_t \in \mathbb{R}^{MN}$ are $M \times N$ pixel images that correspond to time slices of the measured and uncorrupted datacubes, respectively, at time t. $\eta \in \mathbb{R}^{MN}$ is a random noise vector, which we assume follows a noise distribution that is independent of t. The GSF is represented by the matrix $A \in \mathbb{R}^{MN \times MN}$.

7.2. Fast Glare Removal with Sharpening Operator

We use a single-step glare removal method originally proposed by Talvala et al. [4] for glare removal in high dynamic range photography. We use the method to remove glare from individual time slices of a LiDAR datacube, which we treat as still images. The single-step glare removal method is similar in spirit to sharpening operators used in image processing [3]—that is, an approximate inverse that can be applied to an image in minimal computation time. Although iterative optimization methods in theory provide accurate solutions, they are arguably too computationally intensive to provide real-time de-glaring in such scenarios. While the quality of de-glared images produced using single-step de-glare operators may suffer relative to those produced using iterative optimization, our ultimate goal is not to produce clear images, but rather to suppress glare signals that might otherwise be interpreted as real objects in the scene. We will show that, when pileup distortion is not severe, our approximate de-glare operators are effective at accomplishing this task.

To construct the de-glare operator, we first deconstruct the GSF into two components: an identity matrix that accounts for unscattered light, and a second operator $\mathbf{B}$ that accounts for light scattered away from the intended pixel:

$$\mathbf{A} = (1 - \alpha)\mathbb{I} + \alpha\mathbf{B}. \tag{11}$$

Here α quantifies the fraction of light scattered away from the uncorrupted image.[1], with $0 \leq \alpha \leq 1$. All entries of $\mathbf{B}$ are positive, and all columns of B sum to one–i.e. $\sum_i B_{ij} = 1$ for all columns j–which ensures that A conserves the energy of the uncorrupted image.

We construct the following glare removal operator $\mathbf{S}$ from components of the GSF, and apply it to measurements $\mathbf{y}_t$ to obtain the approximate solution $\hat{\mathbf{x}}_t^{(S)}$:

$$\hat{\mathbf{x}}_t^{(S)} = \mathbf{S}\mathbf{y}_t = \left[(1 + \alpha)\mathbb{I} - \alpha\mathbf{B}\right]\mathbf{y}_t. \tag{12}$$

The operator $\mathbf{S}$ subtracts the unwanted glare component while simultaneously scaling the input image to account for light lost due to outscatter. Like $\mathbf{A}$, the columns of $\mathbf{S}$ sum to

[1]For notational simplicity, we've assumed the fraction of outscattered light is the same for all image pixels. To account for variable outscatter fractions, we could define a vector $\tilde{\alpha}$ of same length as $\mathbf{x}_t$, and replace Eq. 11 with $\mathbf{A} = \text{diag}[\mathbf{1} - \tilde{\alpha}] + \mathbf{B}\,\text{diag}[\tilde{\alpha}]$

1, and $\mathbf{S}$ is energy-conserving. The errors in $\hat{\mathbf{x}}_t^{(S)}$ scale with α^2, and so we expect $\hat{\mathbf{x}}_t^{(S)}$ to be a reasonable approximation when α is small. Importantly, $\hat{\mathbf{x}}_t^{(S)}$ can be evaluated with a single matrix-vector product, which requires significantly less computation time that an iterative solver.

The evaluation of $\hat{\mathbf{x}}_t^{(S)}$ can be sped up further by assuming that the GSF is *shift-invariant*, which is often approximately true–particularly at the center of the lidar field of view. In this case, the matrix $\mathbf{B}$ becomes the convolutional kernel $\mathbf{b}$, and the de-glare operator can be written as

$$\hat{\mathbf{x}}_t^{(S)} = (1+\alpha)\mathbf{y}_t - \alpha\left(\mathbf{b} * \mathbf{y}_t\right) \tag{13}$$

7.3. Experimental Details

For evaluating the PDG method we needed to use an ND filter, however, because the lens on our ADAPs evaluation kit is not replaceable we needed to mount the filter over the lens. It is worth noting that we mounted the ND filter in front of the transmitter, not the receiver, as mounting it in front of the receiver would alter the GSF that was measured without use of the ND filter. The filter caused significant crosstalk between the transmitter and receiver, so we used a crosstalk barrier. The ADAPs kit comes with a crosstalk barrier which we extended with a small piece of black foil.

This foil was within the FOV of the sensor thus when applying any method we first zeroed out the borders so this did not impact the glare removal process. Additionally, we cropped out these regions for the figures. We used the same cropped region for the scenes with multiple retroreflectors.

7.4. De-glare Operator Computation

To generate the convolutional version of the de-glare operator described in Eq. 13, we use only one GSF, captured at the center of the lidar FOV (see Fig. 2b,c,d). The unnormalized GSF $\mathbf{a}$ can be decomposed as

$$\mathbf{a} = N\left[(1-\alpha)\hat{e}_0 + \alpha\mathbf{b}\right], \tag{14}$$

where $N = \sum_i a_i$ is the total number of photon counts in the measured GSF, $\hat{e}_0$ is a one-hot vector that equals 1 at the pixel that contains the GSF peak and 0 elsewhere, $\mathbf{b}$ is the convolutional kernel described in Eq. 13, and α is the outscatter ratio from Eq. 13.

The outscatter ratio α can be computed as follows:

$$\alpha = 1 - \frac{a_0}{N}, \tag{15}$$

where a_0 is the number of photon counts in the pixel that contains the GSF peak. The convolutional kernel $\mathbf{b}$ can be computed by taking the vector $\mathbf{a}/N$ and setting the entry of the GSF peak pixel to zero, i.e.

$$b_i = \begin{cases} 0 & \text{at GSF peak pixel} \\ a_i/N & \text{elsewhere} \end{cases} \tag{16}$$

7.5. Bias Introduced by De-glare Operator

The de-glare operator $\mathbf{S}$ introduced in Sec. 7.2 of is not a true inverse, and so will introduce bias in the recovered frames. The relationship between the true, uncorrupted image $\mathbf{x}_t$ and its estimate $\hat{\mathbf{x}}_t^{(S)}$ can be obtained by plugging the expression

$$\hat{\mathbf{y}}_t = \mathbf{A}\mathbf{x}_t = (1-\alpha)\mathbf{x}_t + \alpha\mathbf{B}\mathbf{x}_t. \tag{17}$$

into Eq. 12. When we do so, we obtain the expression

$$\hat{\mathbf{x}}_t^{(S)} = \mathbf{x}_t - \alpha^2\left(\mathbb{I} - \mathbf{B}\right)^2\mathbf{x}_t, \tag{18}$$

which consists of the true image and an additive bias term. Inspection of Eq. 18 shows that the bias scales as a function of α^2 and $(\mathbb{I}-\mathbf{B})^2$. Thus, the bias will remain small when the outscatter fraction α is small. The bias is also strictly negative (assuming $\mathbf{x}_t$ is strictly positive), and takes the form of $(\mathbb{I}-\mathbf{B})^2\mathbf{x}_t$.

When the convolutional form of the de-glare operator (Eq. 13) is used, additional bias will be introduced. The severity of this bias will be low if the GSF is well-approximated as shift-invariant, and should increase with the degree to which that assumption is violated.

8. Baseline Implmentation Details

Because we only have one baseline to compare to it is worth going into greater detail about the implementation and how we reconstructed their method. Following [1], we implement a learning-based method that predicts per-pixel glare likelihoods. Let $H \in \mathbb{R}^{H\times W\times T}$ denote the input histogram volume (we use T=672 time bins). We extract a depth-index map $\hat{t} \in \{0,\ldots,T-1\}^{H\times W}$ and an intensity map $I \in \mathbb{R}^{H\times W}$ via

$$\hat{t}_{ij} = \arg\max_k \; H_{ij[k]}, \tag{19}$$

$$I_{ij} = H_{ij[\hat{t}_{ij}]}. \tag{20}$$

GSF fitting. Given a transient histogram volume $H \in \mathbb{R}^{H\times W\times T}$ containing real-world glare artifacts, we first project it to a per-pixel depth map $D \in \mathbb{R}^{H\times W}$ and intensity map $I \in \mathbb{R}^{H\times W}$. In contrast to [1], which operates directly on linear intensities I, we work in the $\log_2$-intensity domain, since the intensity contrast between retroreflectors and background spans several orders of magnitude, making a direct exponential fit in linear space unstable. For each row containing an "aggressor" region (retroreflector pixels), we collect tuples $(\Delta x, \log r, w)$, where Δx is the lateral distance from the aggressor edge, w is the aggressor width for that row, and

$$\log r = \log_2\left(\frac{I_{\text{pixel}}}{I_{\text{peak}}}\right) \tag{21}$$

is the log-intensity ratio relative to the aggressor peak. Aggressor pixels themselves are excluded via a log-intensity threshold.

We normalize the distance by the aggressor width, $x = \Delta x / w$, aggregate all samples across rows, and fit a low-order polynomial $p(x)$ in the natural-log domain of the intensity ratio by solving

$$\min_{p} \sum_{i} \big(p(x_i) - \ln r_i\big)^2, \tag{22}$$

where $\ln r_i = \log r_i \cdot \ln 2$. This yields a smooth decay function $\hat{r}(x) = \exp\big(p(x)\big)$ defined over $x \in [0, 1]$.

Synthetic data generation. During synthesis, the glare intensity along each row is regenerated by evaluating the learned decay $\hat{r}(x)$ (with x denoting the normalized lateral distance) and scaling the saturated retro intensity accordingly, while keeping the depth constant inside the glare band:

$$I_{\text{glare}}(x) = I_{\text{retro}} \cdot \hat{r}(x), \tag{23}$$

$$D_{\text{glare}}(x) = D_{\text{retro}}. \tag{24}$$

This procedure produces synthetic depth and intensity maps, $D_{\text{synthetic}} \in \mathbb{R}^{H \times W}$ and $I_{\text{synthetic}} \in \mathbb{R}^{H \times W}$, that contain the synthetic retroreflector and its glare artifacts.

Network and training. We adopt a lightweight SqueezeSegV2 [6] backbone that takes the two-channel tensor $[D_{\text{synthetic}} \,\|\, I_{\text{synthetic}}] \in \mathbb{R}^{H \times W \times 2}$ as input and outputs a binary glare likelihood map. The network is trained with class-balanced BCE-with-logits loss using Adam and a cosine-annealed learning rate schedule. We employ only horizontal/vertical flips as data augmentation. The model is trained for 50 epochs with a batch size of 4 on a dataset of 300 synthetic samples.

Inference and histogram-domain suppression. Since [1] outputs a glare-probability map $P \in [0, 1]^{H \times W}$, we extend this baseline to the transient-histogram domain to enable a fair comparison with our method, which requires restoring the original depth and intensity maps from glare-corrupted data. Specifically, for any pixel whose confidence exceeds a fixed threshold ($P_{ij} \geq \tau$, with $\tau = 0.5$), we suppress the corresponding histogram bins within a full-width-at-half-maximum (FWHM) window centered at the estimated time bin $\hat{t}_{ij}$. Throughout our experiments, we set $\text{FWHM} = 10$ time bins, matching the temporal FWHM of the emitted laser pulse in our LiDAR system and assuming it remains unchanged after reflection. Formally,

$$H'_{ij[k]} = \begin{cases} 0, & \text{if } \left|k - \hat{t}_{ij}\right| \leq \frac{\text{FWHM}}{2} \text{ and } P_{ij} \geq \tau, \\ H_{ij[k]}, & \text{otherwise.} \end{cases} \tag{25}$$

The resulting histogram H' is thus "glare-cleaned," in the sense that glare contributions are nulled while the rest of the temporal response remains intact.

Comparison to Our Method One major downside to this method is, as discussed in the main paper, that it operates in the depth map space which may be too late. Signals masked by glare may have already been removed during the post-processing required to transform histograms to depth maps. Another downside is that this method requires retroreflective elements to be geometries that can be easily modeled parametrically (e.g. octagons, rectangles, circles). Objects with more complex geometry such as the cones and drums we tested underperformed. Finally, this method assumes a single retroreflector in the scene and does not adapt well to scenes with multiple retroreflectors, unlike our approach which is scene independent.

References

[1] Sungjin Cheong and Jusung Ha. Lidar blooming artifacts estimation method induced by retro-reflectance with synthetic data modeling and deep learning. In *2024 IEEE International Conference on Consumer Electronics-Asia (ICCE-Asia)*, pages 1–4, 2024. 7, 8

[2] P B Coates. The correction for photon 'pile-up' in the measurement of radiative lifetimes. *Journal of Physics E: Scientific Instruments*, 1(8):878, 1968. 2

[3] Richard Szeliski. *Computer vision: algorithms and applications*, chapter 3. Springer Nature, 2 edition, 2022. 6

[4] Eino-Ville Talvala, Andrew Adams, Mark Horowitz, and Marc Levoy. Veiling glare in high dynamic range imaging. *ACM Trans. Graph.*, 26(3):37–es, 2007. 6

[5] Alessandro Tontini, Sonia Mazzucchi, Roberto Passerone, and Leonardo Gasparini. A post-processing histogram compensation method for spad-based d-tof lidar systems for high photon flux measurements. *IEEE Access*, 12:135390–135397, 2024. 2

[6] Bichen Wu, Xuanyu Zhou, Sicheng Zhao, Xiangyu Yue, and Kurt Keutzer. Squeezesegv2: Improved model structure and unsupervised domain adaptation for road-object segmentation from a lidar point cloud. In *2019 International Conference on Robotics and Automation (ICRA)*, pages 4376–4382, 2019. 8

[7] Joel Yeo, N. Duane Loh, Ramon Paniagua-Dominguez, and Arseniy I. Kuznetsov. Eigencwd: a spatially varying deconvolution algorithm for single metalens imaging. *Optics Express*, 33(13):28481, 2025. 4