2023

Learned Compressive Representations for Single-Photon 3D Imaging

Felipe Gutierrez-Barragan
University of Wisconsin-Madison

Fangzhou Mu
University of Wisconsin-Madison

Andrei Ardelean
Ecole Polytechnique Federale de Lausanne

Atul Ingle
Portland State University, atul.ingle@pdx.edu

Claudio Bruschini
Ecole Polytechnique Federale de Lausanne

See next page for additional authors

Citation Details

Gutierrez-Barragan, F., Mu, F., Ardelean, A., Ingle, A., Bruschini, C., Charbon, E., Li, Y., Gupta, M., & Velten, A. (2023, October 1). Learned Compressive Representations for Single-Photon 3D Imaging. 2023 IEEE/CVF International Conference on Computer Vision (ICCV). https://doi.org/10.1109/iccv51070.2023.00987

Learned Compressive Representations for Single-Photon 3D Imaging

Felipe Gutierrez-Barragan[1‡], Fangzhou Mu[1], Andrei Ardelean[3], Atul Ingle[2], Claudio Bruschini[3],
Edoardo Charbon[3], Yin Li[1], Mohit Gupta[1], Andreas Velten[1]
[1]University of Wisconsin-Madison, [2]Portland State University,
[3]Ecole Polytechnique Federale de Lausanne

Abstract

Single-photon 3D cameras can record the time-of-arrival of billions of photons per second with picosecond accuracy. One common approach to summarize the photon data stream is to build a per-pixel timestamp histogram, resulting in a 3D histogram tensor that encodes distances along the time axis. As the spatio-temporal resolution of the histogram tensor increases, the in-pixel memory requirements and output data rates can quickly become impractical. To overcome this limitation, we propose a family of linear compressive representations of histogram tensors that can be computed efficiently, in an online fashion, as a matrix operation. We design practical lightweight compressive representations that are amenable to an in-pixel implementation and consider the spatio-temporal information of each timestamp. Furthermore, we implement our proposed framework as the first layer of a neural network, which enables the joint end-to-end optimization of the compressive representations and a downstream SPAD data processing model. We find that a well-designed compressive representation can reduce in-sensor memory and data rates up to 2 orders of magnitude without significantly reducing 3D imaging quality. Finally, we analyze the power consumption implications through an on-chip implementation.

1. Introduction

3D cameras based on single-photon avalanche diode technology (SPAD) are becoming increasingly popular for a wide range of applications that require high-resolution and low-power depth sensing, ranging from autonomous vehicles [1] to consumer smartphones [2]. Kilo-to-megapixel resolution SPAD pixel arrays [27, 28] have the capability of capturing the time-of-arrival of billions of individual photons per frame with extremely high (picosecond) time resolution [31]. Unfortunately, this extreme sensitivity and high speed comes at a cost — the raw timestamp data causes a severe bottleneck between the image sensor and the image signal processor (ISP) that processes this data (Fig. 1(a)). This data bottleneck severely limits the wider use of high-resolution SPAD arrays in 3D sensing applications.

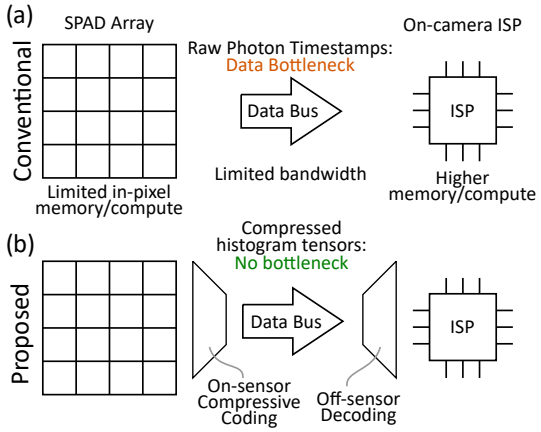

Figure 1. **Resolving SPAD data bottleneck with learned compression.** (a) Conventional SPAD-based 3D cameras stream raw photon timestamps or summary histograms off the image sensor which causes a data bottleneck between the image sensor and the on-camera image and signal processing (ISP) module. (b) Our method applies a lightweight, on-sensor compressive coding scheme to the photon timestamp data which is later decoded at the ISP, resolving the data bandwidth limitation.

One common approach to avoid transferring individual photon timestamps is to build a histogram in each pixel. This results in a 3D histogram tensor that is transferred off-sensor for processing. Although this may be practical at low spatio-temporal resolutions (e.g., 64x32 pixels with 16 time bins [15]), it requires higher in-sensor memory. Moreover, the data rates of this histogram tensor representation also scale rapidly with the spatio-temporal resolution and maximum depth range. For example, a megapixel SPAD-based 3D camera operating at 30 fps that outputs a histogram tensor with a thousand 8-bit bins per pixel would require an unmanageable data transfer rate of 240 Gbps.

To overcome the above limitations, we seek to design compressive representations of 3D histogram tensors. In order to reduce the data rates output by the SPAD camera, the compact representation needs to be built in-pixel or inside the focal plane array (FPA). This is illustrated in Fig. 1(b). Due to the limited in-pixel memory and compute, the compressive representation needs to be built in a streaming manner, with minimal computations per photon. Photon

‡Email: fgutierrez3@wisc.edu

histogram tensors are very different from conventional RGB images/video data. Therefore, traditional compression algorithms such as MPEG are not directly applicable.

We propose a family of compressive representations for 3D histogram tensors that can be computed in an online fashion with limited memory and compute. They are based on the *linear spatio-temporal projection* of each photon timestamp, which can be expressed as a simple matrix operation. Instead of constructing per-pixel timestamp histograms, a compressive encoding maps its spatio-temporal information into a *compressive histogram*. To exploit local spatio-temporal correlations, a single compressive histogram is built for a local 3D histogram block as illustrated in Fig. 2. Instead of building and storing the full 3D histogram tensor in-sensor, multiple compressive histograms are built and transferred off-sensor for processing, effectively reducing the required in-sensor memory and data rates. Recent works proposed a similar compression framework based on compressive histograms [13] or sketches [36]. In Sec. 4, we show that these prior works can be viewed as special cases of our proposed framework.

In this paper, we explore the design space of spatio-temporal compressive encodings and analyze the trade-offs between different design choices. Furthermore, we present a method to integrate our compression framework with data-driven SPAD data processing methods using convolutional neural networks (CNNs), which enables end-to-end optimization of the compressive encoding and a SPAD data processing CNN. We demonstrate the feasibility of compressive histograms through an on-chip implementation.

For our experimental evaluation, we integrate the compressive histograms framework with a state-of-the-art learning-based denoising model for SPAD-based 3D imaging [32]. Our results show that the jointly optimized compressive encoding and CNN can consistently reduce data rates up to 2 orders of magnitude in a wide range of signal and noise levels. Moreover, for a given compression level, it can increase 3D imaging accuracy over previous hand-designed compressive histograms that only exploit temporal information [13, 36], especially in low signal-to-background ratio (SBR) scenarios and at higher compression rates. Furthermore, we show that learned compressive histograms can perform comparably and sometimes even outperform a theoretical SPAD sensor design where the full 3D histogram tensor is stored in-sensor and only per-pixel depths are transferred off-sensor. Finally, we analyze the power consumption of a compressive histogram implemented on the UltraPhase SPAD processing chip [4].

2. Related Work

Compressive histograms, also called sketches, are an emerging framework for online in-sensor compression of SPAD timestamp data [13, 41, 36, 33]. A coarse histogram is one common compressive histogram approach [13] to reduce data rates and in-pixel memory [15, 8, 18]. Despite their practical hardware implementation, coarse histograms achieve sub-optimal depth accuracy compared to compressive histograms based on Fourier [36, 37, 13] and Gray [13] codes. One limitation of these approaches is that the compressive representation only exploits the temporal information of the incident timestamp, and disregards the spatial redundancy. In this work, we generalize the compressive histogram framework to utilize the *spatio-temporal information of each timestamp*. Moreover, instead of relying on hand-designed coded projections, in this paper we learn them as the first layer of a CNN.

Shared In-Pixel Histograms: One common design is to have multiple neighboring SPAD pixels have a single shared memory where all timestamps are aggregated into a coarse histogram (e.g., 4×4 [18, 15], 3×3 [20]). This approach throws away the local spatial information (i.e., pixel location) of the detected photon timestamps. A compressive histogram is well-suited for these shared memory designs because it can be shared among multiple SPAD pixels and preserves spatial information through the coded projection.

Neural Sensors and Pixel Processor Arrays: Pixel processor arrays (PPAs) are an emerging sensing technology that embeds processing electronics inside each pixel [9]. This new sensing paradigm begins processing the image at the focal plane array, which allows it to reduce the sensor data rates by transmitting only the relevant information, and consequently, it increases the sensor's throughput [9] which can enable computer vision at 3,000 fps [6, 7]. PPAs have also become building blocks of novel computational imaging systems optimized end-to-end for HDR imaging [25, 39], motion deblurring [30], video compressive sensing [25], and light field imaging [43]. A compressive histogram relies on a similar *in-pixel processing* paradigm. Similar to previous works, we jointly optimize the sensor parameters (i.e., the compressive histogram) and the processing algorithm (i.e., the CNN), but to our knowledge, this work is the first to use this approach for SPAD data compression.

3. Single-Photon 3D Histogram Tensors

SPAD-based 3D cameras consist of a SPAD sensor and a pulsed laser that illuminates the scene. The photon flux signal arriving at pixel, $\boldsymbol{p}$, can be written as:

$$\Phi_{\boldsymbol{p}}(t) = a_{\boldsymbol{p}} h(t - 2d_{\boldsymbol{p}}/c) + \Phi_{\boldsymbol{p}}^{\text{bkg}} = \Phi_{\boldsymbol{p}}^{\text{sig}}(t) + \Phi_{\boldsymbol{p}}^{\text{bkg}}, \quad (1)$$

where $a_{\boldsymbol{p}}$ is the amplitude of the returning signal accounting for laser power, reflectivity, and light fall-off; $h(t)$ is the system's impulse response function (IRF) which accounts for pulse waveform and sensor IRF; $d_{\boldsymbol{p}}$ is the distance to the point imaged by $\boldsymbol{p}$; c is the speed of light; and $\Phi_{\boldsymbol{p}}^{\text{bkg}}$ is the constant photon flux due to background illumination (e.g., sunlight). This model assumes direct-only reflections

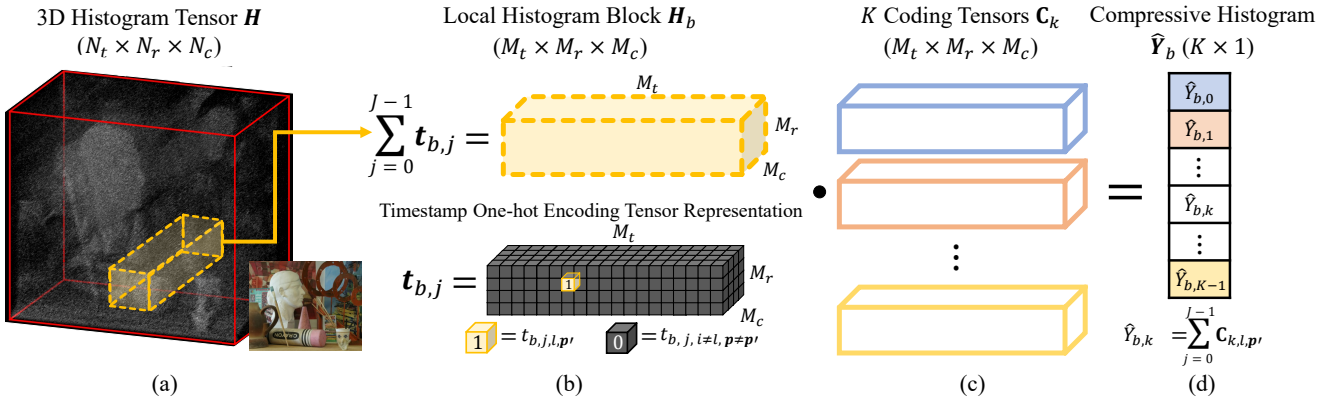

Figure 2. **Compressive Histogram Formation.** Histogram tensors, $\boldsymbol{H}$, are a 3D spatio-temporal grid whose elements store the number of photons that arrived within a short time interval. (a) In SPAD-based 3D imaging, the temporal axis of $\boldsymbol{H}$ encodes distances. (b) A histogram block, $\boldsymbol{H}_b$, can be expressed as the sum of J one-hot encoding tensors, where each tensor represents a photon timestamp. (c) A compact representation of $\boldsymbol{H}_b$ can be built by applying K linear projections (i.e., dot product) with pre-designed coding tensors. (d) The compressive histogram, will be a vector with K elements whose compression capacity is given by $\frac{M_t \cdot M_r \cdot M_c}{K}$.

which is a valid approximation, in particular, for scanning-based ToF 3D imaging systems [3].

TCSPC-based SPAD cameras measure $\Phi_{\boldsymbol{p}}(t)$ by building a per-pixel timing histogram, where the ith histogram bin records the number of photons that arrived in a time interval of length Δ, which follows a Poisson process $\mathcal{P}$:

$$\Phi_{i,\boldsymbol{p}} = \mathcal{P}(\Phi^{\text{sig}}_{i,\boldsymbol{p}} + \Delta\Phi^{\text{bkg}}_{\boldsymbol{p}}). \tag{2}$$

The pulse repetition period, τ, determines the maximum timestamp value and the length of the histogram vector $\boldsymbol{\Phi}_{\boldsymbol{p}} = (\Phi_{i,\boldsymbol{p}})_{i=0}^{N_t-1}$, where $N_t = \tau/\Delta$. Therefore, one assumption built into $\boldsymbol{\Phi}_{\boldsymbol{p}}$, is that no signal photons had a timestamp larger than τ, which means that the maximum distance that $\boldsymbol{\Phi}_{\boldsymbol{p}}$ can encode is $d_{\text{max}} = \frac{c\tau}{2}$. Furthermore, we assume that pile-up distortions are minimized through various SPAD data acquisition techniques [11, 15, 16, 12].

This process generates a $N_t \times N_r \times N_c$ 3D histogram tensor, $\boldsymbol{H} = (\boldsymbol{\Phi}_{\boldsymbol{p}})_{\boldsymbol{p}=(0,0)}^{(N_r-1,N_c-1)}$. In challenging 3D imaging scenarios with high background illumination, building $\boldsymbol{H}$ off-sensor requires transferring thousands of photon timestamps per-pixel leading to data rates of hundreds of GB/s in a megapixel sensor. Moreover, building and storing a high-resolution $\boldsymbol{H}$ in-sensor would require significant memory (1GB for a megapixel SPAD camera with 1000 time bins per-pixel), and transferring it would continue to lead to impractical data rates of tens of GB/s on a SPAD-based 3D camera operating at 30fps. Overall, a practical SPAD-based 3D camera would build and store a *compact representation* of $\boldsymbol{H}$ in-sensor and then transfer it to a processing chip (e.g., FPGA, ISP, embedded computer) where it is processed.

4. Spatio-temporal Compressive Histograms

A natural approach to compress $\boldsymbol{H}$ that exploits its local correlations due to smooth depths and photon flux, is to build a compressive representation of a local 3D histogram block as illustrated in Fig. 2. To avoid storing or transferring the photon timestamp stream, the compressive representation is built as each timestamp arrives. In this section, we present an online compression framework for histogram blocks based on the coded projection of photon timestamps.

Let $\boldsymbol{H}_b$ be the bth histogram block of $\boldsymbol{H}$ with dimensions $M_t \times M_r \times M_c$, where $M_t \leq N_t$, $M_r \leq N_c$, and $M_c \leq N_c$. First, we observe that $\boldsymbol{H}_b$ can be expressed as the sum of J one-hot encoding tensors, each representing one photon detection within $\boldsymbol{H}_b$ (Fig. 2b). Specifically, let $\boldsymbol{t}_{b,j}$ be a $M_t \times M_r \times M_c$ one-hot encoding tensor representing the j^{th} photon timestamp detected in histogram block $\boldsymbol{H}_b$, whose elements are all 0 except for $t_{b,j,l,\boldsymbol{p}'} = 1$, where $l = \lfloor \frac{T_j \mod (\Delta M_t)}{\Delta} \rfloor$, T_j is the timestamp value, and $\boldsymbol{p}'$ is the pixel where the timestamps was detected. Using this representation we can write $\boldsymbol{H}_b$ as follows:

$$\boldsymbol{H}_b = \sum_{j=0}^{J-1} \boldsymbol{t}_{b,j}\ . \tag{3}$$

$\boldsymbol{H}_b$ can be compressed in an online fashion through the linear projection of each timestamp tensor; expressed as the inner product with K pre-designed *coding tensors*, $\mathbf{C}_k$, with dimensions $M_t \times M_r \times M_c$, as in Fig. 2. Mathematically,

$$\widehat{Y}_{b,k} = \mathbf{C}_k \cdot \boldsymbol{H}_b = \sum_{j=0}^{J-1} \mathbf{C}_k \cdot \boldsymbol{t}_{b,j} = \sum_{j=0}^{J-1} \mathbf{C}_{k,l,\boldsymbol{p}'}, \tag{4}$$

where $\cdot$ denotes element-wise multiplication, and l and $\boldsymbol{p}'$ are the indices where $\boldsymbol{t}_{b,j,l,\boldsymbol{p}'} = 1$. We define $\widehat{\boldsymbol{Y}}_b = (\widehat{Y}_{b,k})_{k=0}^{K-1}$ as the *compressive histogram* of $\boldsymbol{H}_b$. Compressive histograms, as defined in [13, 36], are a special case of Eq. 4 where $\mathbf{C}$ compresses individual histograms and disregards spatial information (i.e., M_t=N_t, M_r=1, M_c=1) Note that each timestamp in Eq. 4 is processed efficiently on-the-fly after each photon detection through a simple lookup

operation. Moreover, individual histogram blocks or timestamps are never explicitly stored or transferred off-sensor.

4.1. Practical Coding Tensor Design

Compressive histograms, when implemented as in Eq. 4, introduce an in-sensor memory overhead because, in addition to storing $\widehat{\boldsymbol{Y}}_b$, $\mathbf{C}$ needs to be stored in-sensor for efficient lookup operations. Therefore, a practical compressive single-photon camera implementation would rely on parameter-efficient coding tensors that minimize this overhead. Here, we present two strategies to design practical coding tensors that are evaluated in Sec. 6.

The memory overhead makes certain coding tensor designs impractical. Consider a set of coding tensors that operate on the full histogram tensor, i.e., $\boldsymbol{H}_b = \boldsymbol{H}$.In this case, the number of elements in $\mathbf{C}$ will exceed the number of elements of $\boldsymbol{H}$. Consequently, although the data rates are reduced in this scenario since $K < (N_t \cdot N_r \cdot N_c)$, the in-sensor memory required exceeds the size of the histogram tensor. To circumvent this issue, we propose two complementary strategies to design lightweight coding tensors: local block-based and separable.

Local Block-based Coding Tensors: As we reduce the size of the histogram block, $\boldsymbol{H}_b$, represented by a compressive histogram, the size of the coding tensors decreases. Therefore, compressing local histogram blocks is not only beneficial due to local spatio-temporal redundancies, but also because these local coding tensors have fewer parameters. One example of local block-based coding tensors are the ones used in temporal compressive histograms [13, 36] where $\boldsymbol{H}_b$ is a per-pixel histogram.

Separable Coding Tensors: One approach to designing lightweight coding tensors is to make them separable along the temporal and spatial dimensions. This approach is used in parameter-efficient CNN models that use separable depth-wise convolutional layers [17] to reduce model size. Formally, we can write a separable coding tensor as the outer product of two smaller tensors:

$$\mathbf{C}_k = \mathbf{C}_k^{\text{temporal}} \otimes \mathbf{C}_k^{\text{spatial}}, \tag{5}$$

where $\mathbf{C}_k^{\text{temporal}}$ is a $M_t \times 1 \times 1$ tensor, and $\mathbf{C}_k^{\text{spatial}}$ is a $1 \times M_r \times M_c$ tensor. This design is also motivated by the differences between the temporal and spatial correlations encountered in histogram blocks. In addition to local correlations present in both dimensions, the temporal dimension also exhibits long-range correlations due to the background illumination offset ($\Phi_{\boldsymbol{p}}^{\text{bkg}}$) in every histogram bin. Therefore, Eq. 5 may be able to represent this prior by encoding the temporal and spatial information independently.

One assumption made in our memory overhead analysis is that a compressive SPAD-based 3D camera only needs to store a single $\mathbf{C}$ that is shared across the full sensor, which can be implemented in two general ways. One approach could distribute $\mathbf{C}$ across the local memory of all pixels and then allow communication across pixels as in PPAs [6]. A second approach could store $\mathbf{C}$ in a global memory that can be accessed by all pixels which could be enabled in 3D-stacked SPAD cameras [44]. Finally, some of the coding tensor designs explored in this paper have as few as 640 parameters. In this case, even if $\mathbf{C}$ is stored for every 4×4 group of pixels, the in-sensor memory is still reduced by $20\times$ compared to storing a 1024 bin per-pixel histogram.

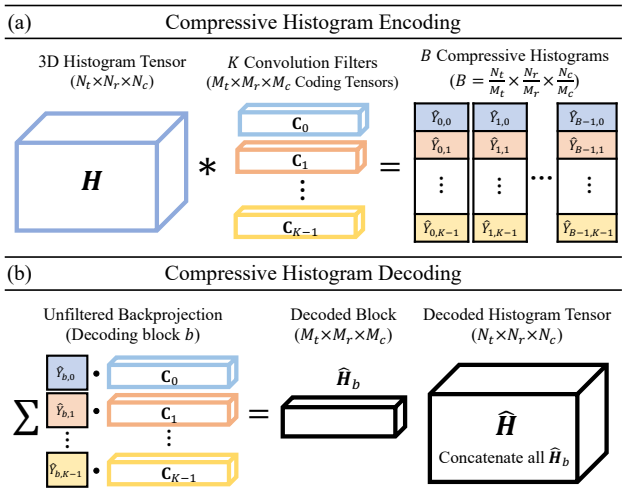

Figure 3. **Convolutional Compressive Histogram Layer.** (a) Building a compressive histogram for each block in the histogram tensor, can be viewed as applying K strided convolutional filters whose weights are the coding tensors. The compressed histogram tensor will be K $\frac{N_t}{M_t} \times \frac{N_r}{M_r} \times \frac{N_c}{M_c}$ tensors. (b) To lift the compressed histogram tensor back to its original domain an unfiltered backprojection operation can be applied on each compressive histogram which decodes a single block. The decoded histogram tensor can then be assembled by concatenating all the decoded blocks.

4.2. Convolutional Compressive Histogram Layer

A compressive histogram is built for each histogram block. Therefore, multiple compressive histograms are used to encode the complete histogram tensor. In this way, the coding tensors can be viewed as a set of 3D convolutional filters, which can be implemented as the first layer of a 3D CNN. For simplicity, we assume that histogram blocks do not overlap, and therefore, the stride of the convolutional filters will equal their dimensions. Fig. 3(a) illustrates this convolutional compressive histogram encoding.

Unfortunately, the compressed histogram tensor representation is not directly compatible with 3D CNNs that have been designed for SPAD-based 3D imaging (e.g., [32, 21]). To this end, each compressive histogram is lifted back to the original 3D domain through an unfiltered backprojection:

$$\widehat{\boldsymbol{H}}_b = \sum_{k=0}^{K-1} \mathbf{C}_k \widehat{Y}_{b,k} \, . \tag{6}$$

Here $\widehat{\boldsymbol{H}}_b$ is the decoded compressive histogram for block b, which is the weighted linear combination of the coding tensors. The decoded histogram blocks are then concatenated and given as input to the processing 3D CNN. Fig. 3(b) illustrates this decoding step. The decoding step in Eq. 6 will occur off-sensor, after the compressive histograms have been moved to the camera compute module which has access to larger memory and computational resources than sensor module. One benefit of using the unfiltered backprojection as the upsampling operator is that if all coding tensors are mutually orthogonal, in the limit when K approaches the size of $\boldsymbol{H}_b$ (i.e., no compression), then $\widehat{\boldsymbol{H}}_b \approx \boldsymbol{H}_b$. This suggests that at compresion rates close to unity, an appropriately trained compressive histogram layer should be approximately equal to an identity transformation applied to $\boldsymbol{H}_b$.

To summarize, a compressive histogram layer comprises an encoding/compression step followed by a decoding step, which uses the coding tensors as the convolutional filters. This layer can be appended to the beginning of any CNN that has been designed to process 3D histogram tensors. Finally, the coding tensors can be jointly optimized with the downstream CNN in an end-to-end manner.

5. Datasets and Implementation

In this section, we describe the datasets used for model training and testing, and also provide implementation details for the compressive histogram layer and the 3D CNN used for the experiments.

5.1. Datasets

For training, we generate a synthetic SPAD measurement dataset containing different scenes at a wide range of illumination settings. We use a similar synthetic data generation pipeline used in previous learning-based SPAD-based 3D imaging works [21, 32, 40]. Using Eq. 2, SPAD measurements can be simulated given an RGB-D image, the pulse waveform ($h(t)$), and the average number of detected signal and background photons per pixel. Please refer to the supplement for a detailed overview of the simulation pipeline.

Simulated Training Dataset: We use the RGB-D images from the NYU v2 dataset [38]. The simulated histograms have $N_t = 1024$ bins and a $\Delta = 80$ps bin size (12.3m depth range). The pulse waveform used has a full-width half maximum (FWHM) of 400ps obtained from [21]. For each scene, we randomly set the average number of signal and background photons detected per pixel to [2, 5, or 10] and [2, 10, 50], respectively. With appropriate normalization, the models generalize to other photon levels despite being trained on this photon-starved dataset. A total of 16,628 histogram tensors with dimensions $1024 \times 64 \times 64$ are simulated and split into a training and a validation set with 13,851 and 2,777 examples, respectively.

Simulated Test Dataset: For testing we use 8 RGB-D images from the Middlebury stereo dataset [35]. The simulated histograms have $N_t = 1024$ bins and a $\Delta = 100$ps bin size (15.3m depth range). The pulse waveform used is a Gaussian pulse with an FWHM of 318ps ($\sigma = 135$ps). A total of 128 test histogram tensors are generated by simulating each scene with the following average number of detected signal/background photons: 2/2, 2/5, 2/50, 5/2, 5/10, 5/50, 10/2, 10/10, 10/50, 10/200, 10/500, 10/1000, 50/50, 50/200, 50/500, and 50/1000.

Real-world Experimental Data: To evaluate the generalization of the proposed models, we downloaded raw histogram tensor captured in [21] with a line-scanning SPAD-based 3D camera prototype. Please refer to the supplementary document for details on this dataset.

5.2. Training and Implementation

To simplify training, the input to all models is a 3D histogram tensor, even though compressive histograms can work directly on streams of photon timestamps (Eq. 4).

Compressive Histogram Layer: The encoder is implemented as a 3D convolution with a stride equal to the filter size, whose learned filters are the coding tensors, $\mathbf{C}_k$. We constraint $\mathbf{C}_k$ to be zero-mean along the time dimension. The unfiltered backprojection decoder is implemented as a 3D transposed convolution with a stride equal to its filter size.To help the CNN model generalize to different photon count levels we apply zero-normalization along the channel dimension (i.e., K) to the inputs ($\widehat{\boldsymbol{Y}}_b$) and the weights ($\mathbf{C}$) of the transposed convolution, also known as layer-norm [5]. This normalization is also commonly used in depth decoding algorithms [14, 13, 26].

Depth Estimation 3D CNN: To estimate depths from the decoded histogram tensor we use the 3D deep boosting CNN model proposed by [32] for single-photon 3D imaging, without the non-local block. Similar to [21, 32] we use the pixel-wise KL-divergence between the output histogram tensor and a normalized true histogram tensor as our objective, and estimate depths using a softargmax.

Training: At each training iteration we randomly sample patches of size $1024 \times 32 \times 32$. We train all models using the ADAM optimizer [19] with $\beta_1 = 0.9$, $\beta_2 = 0.999$, batch size of 4, and a learning rate of 0.001 that decays by 0.9 after every epoch. We train all models for 30 epochs with periodical checkpoints, and for a given model we choose the checkpoint that achieves the lowest root mean squared error (RMSE) on the validation set.

6. Experiments and Results

In this section, we present the performance at various compression levels for different coding tensor designs jointly optimized with the depth estimation CNN described in Sec. 5. A coding tensor design is determined by the

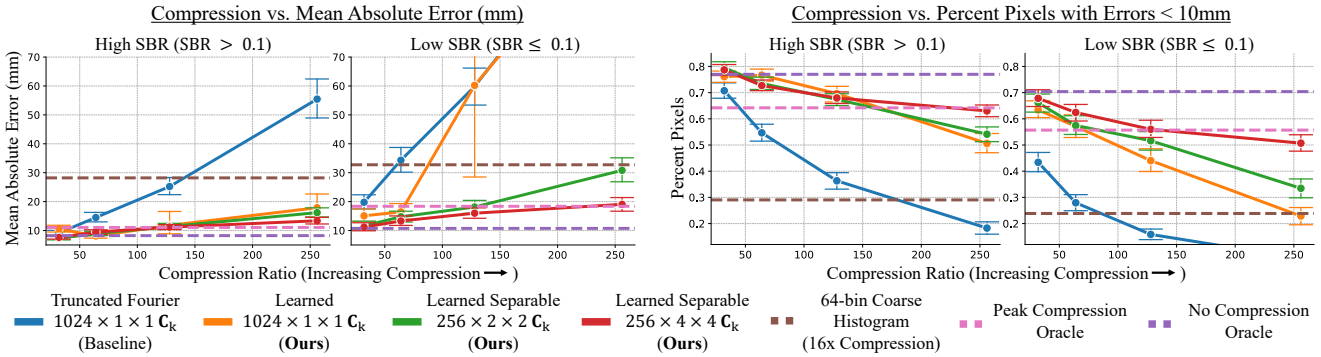

Figure 4. **Compression vs. Test Set Metrics.** The two left-most plots show the mean absolute error computed over the test set as we increase compression. Similarly, the two right-most plots show the mean percent of pixels whose absolute depth errors were < 10mm. The simulated test set images were divided into low (SBR ≤ 0.1) and high (SBR > 0.1) SBR groups to be able to disentangle the impact of SBR on the performance of each model. The dashed lines show the peak and no compression baselines whose compression levels do not vary. Each line corresponds to a fixed coding tensor design for which we vary K to control the compression level. Moreover, each point for a given compression level corresponds to a single set of coding tensors jointly optimized with the depth estimation 3D CNN.

dimensions of $\mathbf{C}_k$ (i.e., $M_t \times M_r \times M_c$), the size of the compressive histogram (i.e., K), and if $\mathbf{C}_k$ is separable.

6.1. Baselines and Performance Metrics

We compare against the following baselines:

- **Temporal Truncated Fourier C [36, 13]:** A compressive histogram that uses coding tensors with dimensions $1024 \times 1 \times 1$ and whose weights are set using the first $K/2$ frequencies of the Fourier matrix. In the supplement, we compare against additional Fourier-based $\mathbf{C}$ [13].
- **Temporal Coarse Histogram C:** Here $\mathbf{C}$ is a box downsampling operator along the temporal dimensions which produces a coarse histogram with K bins.
- **No Compression Oracle:** In this baseline, we assume the ideal scenario where the histogram tensor is transferred off-sensor and processed with the depth estimation 3D CNN. Similar to [32], we train this model with an initial learning rate of 10^{-4} and total variation regularization.
- **Peak Compression Oracle:** This baseline implements an ideal SPAD camera with sufficient in-sensor memory to store the histogram tensor and sufficient computation power to compute per-pixel depths through an "argmax" along the time axis. To process the noisy 2D depth images with the 3D CNN, we generate a 3D grid where all elements are zero except for one element per spatial location whose index is proportional to the depth. This model is trained like the no-compression oracle.

Similar to our proposed approach, all compressive histogram baselines described here, implemented their $\mathbf{C}$ as a compressive histogram layer, with fixed weights, whose outputs are processed by the depth estimation 3D CNN.

Evaluation Metrics: The 3D imaging performance of each model is summarized using two metrics: (1) the mean absolute depth error (MAE), (2) and the percent of pixels with absolute depth errors that are lower than 10mm. To understand the performance under these metrics we divide the test set into different SBR ranges and report the metrics for each range individually. We also visualize the overall dataset performance as scatter plots (e.g., Fig. 7) where each point shows the MAE for a given test scene and their color hue represents the mean SBR of the scene. Outliers with an MAE larger than 50mm are not visible in the plot, however, they are included in the calculation of the statistics. Finally, when comparing different compressive histogram strategies the compression ratio is fixed. The compression ratio (CR) is the ratio of the block size and the length of the compressive histogram (i.e., CR $= (M_t \cdot M_r \cdot M_c \cdot)/K$).

6.2. 3D Imaging Performance

Compression vs. Performance: Fig. 4 shows the performance of compressive histograms as a function of the compression ratio. The learned coding tensors consistently outperform the temporal Fourier-based $\mathbf{C}$. At low SBR and CR > 100, it becomes essential for the learned coding tensors to utilize spatio-temporal information (i.e., green and red lines). Moreover, the proposed models can outperform the peak compression oracle for CR ≤ 64. Overall, learned spatio-temporal coding tensors provide robust performance that degrades gracefully as compression increases.

Importance of Learned Coding Tensors: Fig. 5 compares the depth reconstructions of compressive histograms with coding tensors that were optimized (ours) against coding tensors that were fixed and not optimized throughout training. The extreme quantization in coarse histograms causes large systematic depth errors. Random unoptimized coding tensors consistently produce lower-quality depth reconstructions. A well-designed coding tensor based on Fourier codes can produce reasonable depth reconstructions at 64x compression, however, at 128x compression, scene details become blurred. The proposed learned coding tensors are

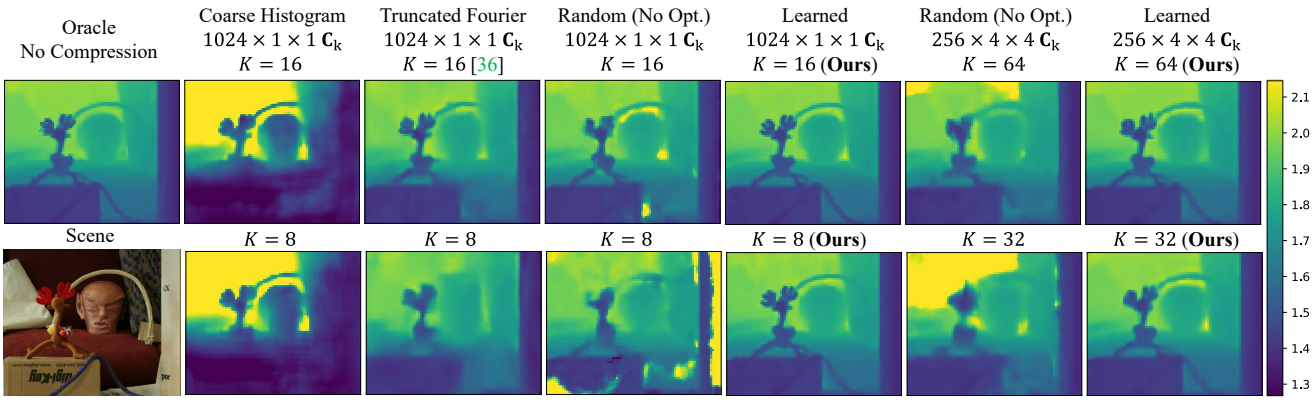

Figure 5. **Importance of Learned Coding Tensors.** Depth reconstructions at 64x (top) and 128x (bottom) compression for compressive histogram models whose coding tensors were hand-designed (coarse histogram and Fourier-based), learned (proposed), and not learned (randomly initialized). The simulated scene had an SBR=0.1, where the mean signal and background photon levels were [50, 500].

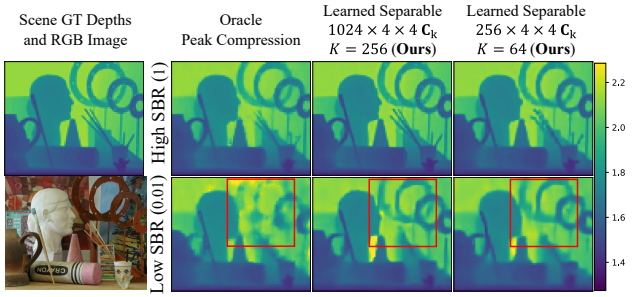

Figure 6. **What if we could compute depths in-pixel?** Depth reconstructions at high and low SBR with mean signal and background photon levels of [10, 10] and [10, 1000], respectively. The compressive histograms have a compression of 64.

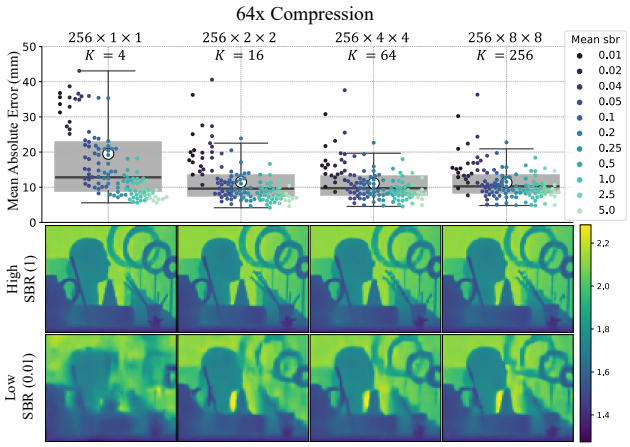

Figure 7. **When do spatio-temporal coding tensors help?** Each scatter plot point corresponds to the MAE of a test scene. The images directly below each model correspond to the depth reconstructions for two test examples at high and low SBR levels whose mean signal and background photons per pixel are [10, 10] and [10, 1000], respectively. For a fixed compression level, the spatial block size of each model is increased from left to right and K is adjusted to maintain the same compression level. The coding tensors for all models in this plot are learned and separable.

able to generate high-quality reconstructions comparable to the no-compression oracle. Overall, optimizing the coding tensors can provide non-trivial performance gains.

What if we could compute depths in-pixel? Fig. 6 compares the depth reconstruction quality of two learned coding tensors at 64x compression with the peak compression oracle described in Sec. 6.1. At high SBR, all methods recover the fine and coarse scene details. At low SBR the peak compression oracle fails to reconstruct high-level scene structures such as the rings in the red box, while the learned coding tensors better preserve these coarse and fine details.

6.3. Exploring the Coding Tensor Design Space

When does spatio-temporal coding help? Fig. 7 shows the effect of increasing the spatial dimension of $\mathbf{C}$, at 64x compression. At high SBR, all methods have similar MAE, but, coding tensors with smaller spatial dimensions better preserve fine details (e.g., sticks). On the other hand, at low SBR, coding tensors with larger spatial dimensions preserve high-level details such as the pot handle. This difference is also observed in the scatter plot where the mean and median of the $256 \times 1 \times 1$ $\mathbf{C}$ do not match which indicates multiple low SBR scenes with high MAE.

How does reducing the size of $\mathbf{C}$ affect performance? Fig. 8 shows the effect of reducing the size of $\mathbf{C}$ at 128x compression. The coding tensors size is reduced by training models with separable coding tensors that operate on smaller histogram blocks. The performance difference between full and separable coding tensors ($1024 \times 4 \times 4$) is negligible. As we further reduce the number of parameters in $\mathbf{C}$, the overall performance degrades. Coding tensors with fewer parameters that operate on smaller histogram blocks tend to produce blurrier reconstructions. This can be observed in the red box where the coding tensors with less than 10,000 parameters blurs the spikes. Nonetheless, as discussed in Sec. 4.1, a parameter-efficient $\mathbf{C}$ is desirable due to the limited in-sensor memory. Ultimately, a practical

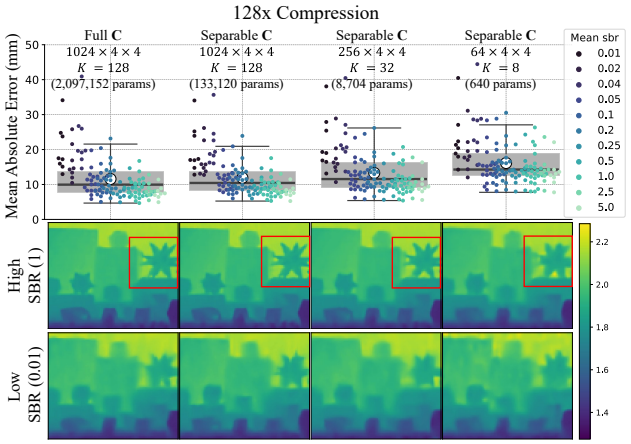

Figure 8. **How does Reducing Size of C Affect Performance?.** Each scatter plot point corresponds to the MAE of a test scene. The images directly below each model correspond to the depth reconstructions for two test scenes at high and low SBR levels whose mean signal and background photons per pixel are [10, 10] and [10, 1000], respectively. The size of $\mathbf{C}$ is reduced from left to right by making it separable or reducing the M_t.

compressive SPAD-based 3D camera design requires determining the trade-off between parameter efficiency and 3D imaging quality, which may depend on the application.

6.4. Evaluation on Real-world Data

Fig. 9 shows the depth reconstructions at 256x compression of multiple compressive histograms and the no compression baseline on the real-world experimental data from [21]. In low SBR scenes, such as the outdoor capture of the ball falling down stairs (first row), Truncated Fourier blurs the staircase edges, while the learned spatio-temporal $\mathbf{C}$ preserved these details. Compared to the no compression oracle, compressive histogram models produce less smooth depth images with some small artifacts. This suggests that the compressive histogram models could benefit from a spatial regularizer such as the one used when training the no compression oracle. Additional results and details on this evaluation are available in the supplement.

6.5. On-chip Implementation and Power Analysis

To validate the feasibility of compressive histograms, we implement them on the UltraPhase chip which has a 3×6 processor core array [4]. At this time, UltraPhase has not been 3D stacked on a SPAD sensor, thus we only evaluate its compute and data readout power consumption. Fig. 10 shows the power dissipated by UltraPhase when processing photon timestamps with different methods and transferring data off-sensor. Although compressive histograms dissipate more power on computation (blue), this is less than 0.3% of the overall power consumption, which is dominated by data readout. Due to limited memory in UltraPhase (4096 bits per core), our method learns $\mathbf{C}^{\text{spatial}}$ and uses Fourier codes

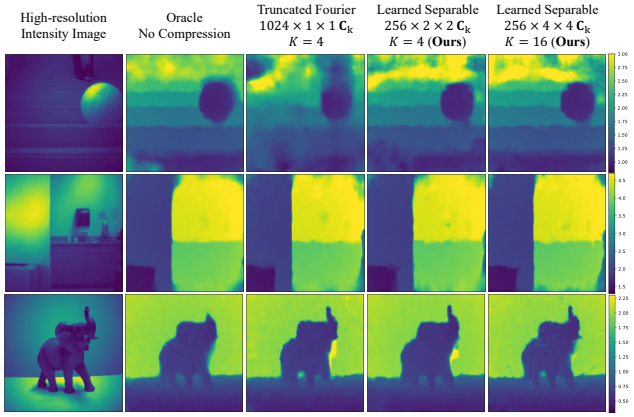

Figure 9. **Depth Reconstructions of Real-world SPAD Data at 256x Compression.** Depth reconstructions of different scenes captured with a SPAD-based 3D camera prototype [21].

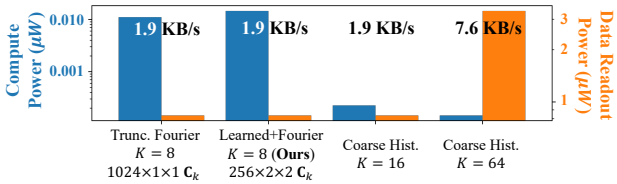

Figure 10. **Compute vs. Data Readout Power.** Average power dissipated by 2x2 SPADs processing 500 timestamps per depth frame and operating at 30 depth frames per second. Compute power is obtained from the implementation on the UltraPhase SPAD processors [4]. Readout power is estimated from the output data rate assuming the readout scheme of the SwissSPAD2 [42]. Coarse histograms use 8-bit precision, while compressive histograms use 16-bit precision. Note the difference in blue and orange scales.

for $\mathbf{C}^{\text{temporal}}$ due to their memory efficient implementation. We also quantize $\mathbf{C}$ to 8 bits and find no degradation in performance. Refer to the supplement for additional details.

7. Discussion

SPAD-based 3D cameras encounter a data bottleneck between the SPAD array and the compute module when transferring the photon timestamps. A histogram tensor can help summarize timestamps at low resolutions, but as megapixel SPAD arrays become available, histogram tensors also lead to a data bottleneck. To overcome this limitation we proposed compressive histograms as a compact representation that can be built on-the-fly, as each photon is detected. As a consequence, compressive histograms can reduce the in-sensor memory and data rates because neither the photon data stream nor a histogram tensor needs to be stored or transferred. Our results show that high-quality depth information can be recovered from a learnt compressive histogram representation that is up to 2 orders of magnitude smaller than a histogram tensor representation.

Practical compressive histogram operating points: The

learned separable 256x4x4 and 256x2x2 $\mathbf{C}$, achieved a good balance between parameter efficiency vs. reconstruction quality, and require on the order of 10-100kbits of memory. In an UltraPhase-like chip this would require distributing $\mathbf{C}$ among at least 8x8 cores (4kbits per core). If the temporal dimension is fixed to Fourier-based $\mathbf{C}^{\text{temporal}}$ such as the ones from [13, 36], the number of parameters can be reduced by 10-40x (see Suppl. Sec. 3.3) which enables storing $\mathbf{C}$ on a per-pixel basis and made the UltraPhase evaluation possible. Overall, for a 1MP SPAD sensor with 1000 bins per-pixel, compressive histograms that can provide more than 50x compression would result in practical data rates of ~ 0.6GB/s which USB 3.2 can support.

Trade-off space: Compressive histograms establish a trade-off between reconstruction fidelity, data bandwidth, and in-sensor compute and memory resources. For a fixed bandwidth, a coarse histogram requires arguably the lowest in-sensor resources but leads to poor reconstructions. Fourier-based histograms can improve reconstruction fidelity, at the expense of increased in-sensor computation and memory. The proposed generalization of compressive histograms allows further increasing reconstruction accuracy, at the expense of additional in-sensor memory over Fourier-based methods. Ultimately, the correct trade-off for a given scenario will be determined by power, application, and hardware constraints.

Although compressive histograms require more in-sensor computation than coarse histograms, our method presents a practical solution from a power consumption and a real-time application perspective. Power consumption is primarily determined by data rates. Fig. 10 and Suppl. Fig. 4 shows that despite compressive histograms dissipating 100x more processing power than coarse histograms, their overall power consumption is still lower even when data rates are only reduced by 2x. The processing time for building compressive histograms is ∼0.5ms (Suppl. Fig. 5). From a real-time (*i.e.*, 30 FPS) or a high-speed application standpoint, this additional processing time is nearly negligible since it is overlapped with acquisition/exposure time.

From a hardware perspective, the importance of reducing the memory overhead of $\mathbf{C}$ depends on resolution. In low-resolution SPAD cameras, memory-efficient $\mathbf{C}$ are essential because they are stored among a single or a few pixels. The proposed method allowed integrating memory-efficient Fourier codes with learned spatial codes, which were implemented on the 2x2 pixels of UltraPhase (Suppl. Sec. 2). As resolution increases and $\mathbf{C}$ is shared among more pixels, the memory overhead reduces, and less parameter-efficient $\mathbf{C}$ with increased reconstruction fidelity can be considered.

Bias in Learned $\mathbf{C}$: In the supplement, we show that coding tensors with $M_t = N_t$ develop a depth range bias. These coding tensors learn to zero out photons coming from distances that are less common in the dataset since they are usually background/noise photons. Interestingly, learned coding tensors with $M_t < N_t$ avoid this bias and generalize to depths that are less common in the training set.

Generalization: There are multiple generalization axes including signal and ambient light levels, sensor and laser parameters (*e.g.*, resolution, pulse waveform), and scene response complexity (*e.g.*, depth range, indirect reflections). Our evaluation has probed some of them. Specifically, our training set only included a subset of the signal and ambient light levels that the test set contained. Furthermore, the model was trained and tested on histogram tensors with different resolutions. Finally, the dataset bias investigation showed that models with $N_t \leq 256$ generalized well to less prevalent depths during training (Suppl. Sec. 1).

We further discuss the model's ability to generalize in other scenarios. For instance, all learned models (including baselines) are unlikely to generalize well to wider laser pulses, because if the model is trained with a narrow pulse width, it learns to extract signal information from high frequencies which will only contain noise in the wider pulse. Similarly, these learned models may generalize to narrower pulse widths. Furthermore, we anticipate a learned $\mathbf{C}$ with the properties described in [13] to be robust to diffuse indirect reflections but may require data augmentation to generalize to other light transport scenarios.

Why not compute depths in-sensor? SPAD-based 3D cameras with large in-pixel memory could store per-pixel histograms and reduce data rates by computing depths in-pixel (i.e., peak compression oracle). Our results show that compressive histogram can provide similar reconstruction quality and outperform this method at low SBR without requiring the storage of the full histogram tensor in-sensor.

Additional Coding Tensor Designs: Although we find promising empirical results for the coding tensor representations described in this paper, the optimal set of coding tensors will depend on the exact hardware specifications (e.g, in-pixel memory, system bandwidth) and scene-dependent parameters (e.g., SBR, geometry, albedo). Additional lightweight $\mathbf{C}$ designs could rely on other factorization techniques and weight quantization.

Task-specific Compressive Histograms: Histogram tensors are used in other *active* single-photon imaging modalities such as fluorescence lifetime microscopy [34], non-line-of-sight [10, 23, 29], and diffuse optical tomography [22, 45, 24]. Our framework could be used to find compressive representations optimized for these applications.

Acknowledgments: This work was supported by the Department of Energy and National Nuclear Security Administration (DE-NA0003921), Air Force (FA9550-21-1-0341), Swiss National Science Foundation (200021_166289), and NSF Awards 1846884 (A.V.), 1943149 (M.G.), 2107060 (M.G.), 2138471 (A.I.). U.S. DOE full legal disclaimer: https://www.osti.gov/stip/about/disclaimer.

References

[1] How multi-beam flash lidar works. https://ouster.com/blog/how-multi-beam-flash-lidar-works/. Accessed: 2022-03-07. 1

[2] The iphone 12 - lidar at your fingertips. https://www.forbes.com/sites/sabbirrangwala/2020/11/12/the-iphone-12lidar-at-your-fingertips/?sh=580b8bab3e28. Accessed: 2022-03-07. 1

[3] Supreeth Achar, Joseph R Bartels, William L Whittaker, Kiriakos N Kutulakos, and Srinivasa G Narasimhan. Epipolar time-of-flight imaging. *ACM Transactions on Graphics (TOG)*, 36(4):37, 2017. 3

[4] Andrei Ardelean. Computational imaging spad cameras. page 164, 2023. 2, 8

[5] Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E Hinton. Layer normalization. *arXiv preprint arXiv:1607.06450*, 2016. 5

[6] Laurie Bose, Jianing Chen, Stephen J Carey, Piotr Dudek, and Walterio Mayol-Cuevas. A camera that cnns: Towards embedded neural networks on pixel processor arrays. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 1335–1344, 2019. 2, 4

[7] Laurie Bose, Piotr Dudek, Jianing Chen, Stephen J Carey, and Walterio W Mayol-Cuevas. Fully embedding fast convolutional networks on pixel processor arrays. In *European Conference on Computer Vision*, pages 488–503. Springer, 2020. 2

[8] Francesco Mattioli Della Rocca, Hanning Mai, Sam W Hutchings, Tarek Al Abbas, Kasper Buckbee, Andreas Tsiamis, Peter Lomax, Istvan Gyongy, Neale AW Dutton, and Robert K Henderson. A 128× 128 spad motion-triggered time-of-flight image sensor with in-pixel histogram and column-parallel vision processor. *IEEE Journal of Solid-State Circuits*, 55(7):1762–1775, 2020. 2

[9] Piotr Dudek, Thomas Richardson, Laurie Bose, Stephen Carey, Jianing Chen, Colin Greatwood, Yanan Liu, and Walterio Mayol-Cuevas. Sensor-level computer vision with pixel processor arrays for agile robots. *Science Robotics*, 7(67):eabl7755, 2022. 2

[10] Daniele Faccio, Andreas Velten, and Gordon Wetzstein. Non-line-of-sight imaging. *Nature Reviews Physics*, 2(6):318–327, 2020. 9

[11] Anant Gupta, Atul Ingle, and Mohit Gupta. Asynchronous single-photon 3d imaging. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 7909–7918, 2019. 3

[12] Anant Gupta, Atul Ingle, Andreas Velten, and Mohit Gupta. Photon-flooded single-photon 3d cameras. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 6770–6779, 2019. 3

[13] Felipe Gutierrez-Barragan, Atul Ingle, Trevor Seets, Mohit Gupta, and Andreas Velten. Compressive single-photon 3d cameras. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 17854–17864, 2022. 2, 3, 4, 5, 6, 9

[14] Felipe Gutierrez-Barragan, Syed Azer Reza, Andreas Velten, and Mohit Gupta. Practical coding function design for time-of-flight imaging. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1566–1574, 2019. 5

[15] Istvan Gyongy, Sam W Hutchings, Abderrahim Halimi, Max Tyler, Susan Chan, Feng Zhu, Stephen McLaughlin, Robert K Henderson, and Jonathan Leach. High-speed 3d sensing via hybrid-mode imaging and guided upsampling. *Optica*, 7(10):1253–1260, 2020. 1, 2, 3

[16] Felix Heide, Steven Diamond, David B Lindell, and Gordon Wetzstein. Sub-picosecond photon-efficient 3d imaging using single-photon sensors. *Scientific reports*, 8(1):1–8, 2018. 3

[17] Andrew G Howard, Menglong Zhu, Bo Chen, Dmitry Kalenichenko, Weijun Wang, Tobias Weyand, Marco Andreetto, and Hartwig Adam. Mobilenets: Efficient convolutional neural networks for mobile vision applications. *arXiv preprint arXiv:1704.04861*, 2017. 4

[18] Sam W Hutchings, Nick Johnston, Istvan Gyongy, Tarek Al Abbas, Neale AW Dutton, Max Tyler, Susan Chan, Jonathan Leach, and Robert K Henderson. A reconfigurable 3-d-stacked spad imager with in-pixel histogramming for flash lidar or high-speed time-of-flight imaging. *IEEE Journal of Solid-State Circuits*, 54(11):2947–2956, 2019. 2

[19] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014. 5

[20] Oichi Kumagai, Junichi Ohmachi, Masao Matsumura, Shinichiro Yagi, Kenichi Tayu, Keitaro Amagawa, Tomohiro Matsukawa, Osamu Ozawa, Daisuke Hirono, Yasuhiro Shinozuka, et al. A 189x600 back-illuminated stacked spad direct time-of-flight depth sensor for automotive lidar systems. In *2021 IEEE International Solid-State Circuits Conference (ISSCC)*, volume 64, pages 110–112. IEEE, 2021. 2

[21] David B Lindell, Matthew O'Toole, and Gordon Wetzstein. Single-photon 3d imaging with deep sensor fusion. *ACM Trans. Graph.*, 37(4):113–1, 2018. 4, 5, 8

[22] David B Lindell and Gordon Wetzstein. Three-dimensional imaging through scattering media based on confocal diffuse tomography. *Nature communications*, 11(1):1–8, 2020. 9

[23] Xiaochun Liu, Sebastian Bauer, and Andreas Velten. Phasor field diffraction based reconstruction for fast non-line-of-sight imaging systems. *Nature communications*, 11(1):1–13, 2020. 9

[24] Ashley Lyons, Francesco Tonolini, Alessandro Boccolini, Audrey Repetti, Robert Henderson, Yves Wiaux, and Daniele Faccio. Computational time-of-flight diffuse optical tomography. *Nature Photonics*, 13(8):575–579, 2019. 9

[25] Julien NP Martel, Lorenz K Mueller, Stephen J Carey, Piotr Dudek, and Gordon Wetzstein. Neural sensors: Learning pixel exposures for hdr imaging and video compressive sensing with programmable sensors. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 42(7):1642–1653, 2020. 2

[26] Parsa Mirdehghan, Wenzheng Chen, and Kiriakos N Kutulakos. Optimal structured light à la carte. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 6248–6257, 2018. 5

[27] Kazuhiro Morimoto, Andrei Ardelean, Ming-Lo Wu, Arin Can Ulku, Ivan Michel Antolovic, Claudio Bruschini, and Edoardo Charbon. Megapixel time-gated spad image sensor for 2d and 3d imaging applications. *Optica*, 7(4):346–354, 2020. 1

[28] K Morimoto, J Iwata, M Shinohara, H Sekine, A Abdelghafar, H Tsuchiya, Y Kuroda, K Tojima, W Endo, Y Maehashi, et al. 3.2 megapixel 3d-stacked charge focusing spad for low-light imaging and depth sensing. In *2021 IEEE International Electron Devices Meeting (IEDM)*, pages 20–2. IEEE, 2021. 1

[29] Ji Hyun Nam, Eric Brandt, Sebastian Bauer, Xiaochun Liu, Marco Renna, Alberto Tosi, Eftychios Sifakis, and Andreas Velten. Low-latency time-of-flight non-line-of-sight imaging at 5 frames per second. *Nature communications*, 12(1):1–10, 2021. 9

[30] Cindy M Nguyen, Julien NP Martel, and Gordon Wetzstein. Learning spatially varying pixel exposures for motion deblurring. *arXiv preprint arXiv:2204.07267*, 2022. 2

[31] Sara Pellegrini, Gerald S Buller, Jason M Smith, Andrew M Wallace, and Sergio Cova. Laser-based distance measurement using picosecond resolution time-correlated single-photon counting. *Measurement Science and Technology*, 11(6):712, 2000. 1

[32] Jiayong Peng, Zhiwei Xiong, Xin Huang, Zheng-Ping Li, Dong Liu, and Feihu Xu. Photon-efficient 3d imaging with a non-local neural network. In *European Conference on Computer Vision*, pages 225–241. Springer, 2020. 2, 4, 5, 6

[33] Valentin Poisson, William Guicquero, Gilles Sicard, et al. Luminance-depth reconstruction from compressed time-of-flight histograms. *IEEE Transactions on Computational Imaging*, 8:148–161, 2022. 2

[34] Kayvan Samimi, Danielle E Desa, Wei Lin, Kurt Weiss, Joe Li, Jan Huisken, Veronika Miskolci, Anna Huttenlocher, Jenu V Chacko, Andreas Velten, et al. Light sheet autofluorescence lifetime imaging with a single photon avalanche diode array. *bioRxiv*, pages 2023–02, 2023. 9

[35] Daniel Scharstein and Chris Pal. Learning conditional random fields for stereo. In *2007 IEEE Conference on Computer Vision and Pattern Recognition*, pages 1–8. IEEE, 2007. 5

[36] Michael P. Sheehan, Julián Tachella, and Mike E. Davies. A sketching framework for reduced data transfer in photon counting lidar. *IEEE Transactions on Computational Imaging*, 7:989–1004, 2021. 2, 3, 4, 6, 9

[37] Michael P Sheehan, Julián Tachella, and Mike E Davies. Surface detection for sketched single photon lidar. *arXiv preprint arXiv:2105.06920*, 2021. 2

[38] Nathan Silberman, Derek Hoiem, Pushmeet Kohli, and Rob Fergus. Indoor segmentation and support inference from rgbd images. In *European conference on computer vision*, pages 746–760. Springer, 2012. 5

[39] Haley M So, Julien NP Martel, Piotr Dudek, and Gordon Wetzstein. Mantissacam: Learning snapshot high-dynamic-range imaging with perceptually-based in-pixel irradiance encoding. *arXiv preprint arXiv:2112.05221*, 2021. 2

[40] Zhanghao Sun, David B Lindell, Olav Solgaard, and Gordon Wetzstein. Spadnet: deep rgb-spad sensor fusion assisted by monocular depth estimation. *Optics express*, 28(10):14948–14962, 2020. 5

[41] Julián Tachella, Michael P Sheehan, and Mike E Davies. Sketched rt3d: How to reconstruct billions of photons per second. In *ICASSP 2022-2022 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 1566–1570. IEEE, 2022. 2

[42] Arin Can Ulku, Claudio Bruschini, Ivan Michel Antolović, Yung Kuo, Rinat Ankri, Shimon Weiss, Xavier Michalet, and Edoardo Charbon. A 512× 512 spad image sensor with integrated gating for widefield flim. *IEEE Journal of Selected Topics in Quantum Electronics*, 25(1):1–12, 2018. 8

[43] Edwin Vargas, Julien NP Martel, Gordon Wetzstein, and Henry Arguello. Time-multiplexed coded aperture imaging: Learned coded aperture and pixel exposures for compressive imaging systems. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 2692–2702, 2021. 2

[44] Chao Zhang, Ning Zhang, Zhijie Ma, Letian Wang, Yu Qin, Jieyang Jia, and Kai Zang. A 240× 160 3d-stacked spad dtof image sensor with rolling shutter and in-pixel histogram for mobile devices. *IEEE Open Journal of the Solid-State Circuits Society*, 2:3–11, 2021. 4

[45] Yongyi Zhao, Ankit Raghuram, Hyun Kim, Andreas Hielscher, Jacob T Robinson, and Ashok Narayanan Veeraraghavan. High resolution, deep imaging using confocal time-of-flight diffuse optical tomography. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2021. 9