




A Bayesian Based Deep Unrolling Algorithm for Single-Photon Lidar Systems

Jakeoung Koo , Abderrahim Halimi , *Senior Member, IEEE*, and Stephen McLaughlin , *Fellow, IEEE*

Abstract—Deploying 3D single-photon Lidar imaging in real world applications presents multiple challenges including imaging in high noise environments. Several algorithms have been proposed to address these issues based on statistical or learning-based frameworks. Statistical methods provide rich information about the inferred parameters but are limited by the assumed model correlation structures, while deep learning methods show state-of-the-art performance but limited inference guarantees, preventing their extended use in critical applications. This paper unrolls a statistical Bayesian algorithm into a new deep learning architecture for robust image reconstruction from single-photon Lidar data, i.e. the algorithm's iterative steps are converted into neural network layers. The resulting algorithm benefits from the advantages of both statistical and learning based frameworks, providing best estimates with improved network interpretability. Compared to existing learning-based solutions, the proposed architecture requires a reduced number of trainable parameters, is more robust to noise and mismodelling of the system impulse response function, and provides richer information about the estimates including uncertainty measures. Results on synthetic and real data show competitive results regarding the quality of the inference and computational complexity when compared to state-of-the-art algorithms.

Index Terms—3D reconstruction, Lidar, single-photon imaging, algorithm unrolling, attention, Bayesian inference.

I. INTRODUCTION

SINGLE-PHOTON light detection and ranging (Lidar) is an emerging technique for reconstructing and analyzing 3D scenes and has a wide range of applications [1], [2]. Using time correlated single-photon counting (TCSPC) technology [3], a single-photon Lidar system builds a histogram of photon counts with respect to their time-of-flights (ToF). Unlike Radar using radio signals [4], single-photon Lidar detects reflected signal at optical frequencies, which enables better cross-range resolution imaging than Radar and, as a consequence, better discerning smaller objects. Due to the counting process, single-photon Lidar data is corrupted by discrete noise (often modeled using a Poisson distribution) which prevents direct application of

existing Radar reconstruction methods. Detecting reflected photons relies on a single-photon sensitive detector known as solid-state single-photon avalanche diode (SPAD), while ToFs are obtained by measuring the time difference between the emission of laser pulses and the detection of reflected photons. The acquired histogram contains depth and reflectivity information about the observed objects, and reconstructing such 3D information from single-photon Lidar data has been a subject of very active research [1], [2].

Several approaches have been designed for image reconstruction of single-photon Lidar systems. An early method known as first-photon imaging [5] was developed to recover 3D images from the first detected photon at each pixel. Since then, many reconstruction algorithms have been proposed, which can be categorized into two broad classes: statistical approaches and data-driven approaches. Statistical methods design a prior-based model to reconstruct 3D scenes, while accounting for the sparsity of single-photon Lidar data [6], [7] and spatial correlations between 2D pixels or within the 3D point cloud [8]. The parameters of the resulting models are then estimated using optimization [9]–[14], Markov chain Monte-Carlo (MCMC) [8], [15], [16], marginalization [17], expectation-maximization [18], [19], or Plug-and-Play methods [20], [21]. Such statistical methods provide good interpretability in the sense that we can predict the results, depending on the considered observation model and imposed priors, but they often require user-defined parameters and hand-crafted priors. Data-driven approaches using deep learning have recently become popular for single-photon Lidar systems. Existing deep learning algorithms [22]–[25] train neural networks from simulated data and aim to generalize to unseen data. Once trained, deep learning models usually do not require user-defined parameters, a distinct advantage, compared to statistical methods. Lindell *et al.* [22] first proposed an end-to-end deep learning model which infers depth profiles from Lidar data. Peng *et al.* [23] suggested a non-local network and showed a state-of-the-art performance in low photon and high noise cases. Despite their excellent performance on challenging data these deep learning methods lack interpretability, require long running times for high dimensional Lidar data and can present over-smoothing artifacts around 3D surface boundaries.

This paper takes advantage of statistical and deep learning approaches, by proposing an interpretable and efficient deep learning architecture for high dimensional Lidar data. We design a neural network architecture by unfolding an iterative Bayesian algorithm [26] in the sense that we replace some of its internal operations with neural network blocks. The

Manuscript received October 13, 2021; revised March 15, 2022; accepted April 20, 2022. Date of publication April 26, 2022; date of current version July 11, 2022. This work was supported in part by the U.K. Royal Academy of Engineering under the Research Fellowship Scheme under Grant RF/201718/17128, and in part by EPSRC under Grant EP/T00097X/1, Grant EP/S000631/1, and Grant EP/S026428/1. The guest editor coordinating the review of this manuscript and approving it for publication was Dr. Miguel Rodrigues. (*Corresponding author: Abderrahim Halimi.*)

The authors are with the School of Engineering and Physical Sciences, Heriot-Watt University, EH144AS Edinburgh, U.K. (e-mail: j.koo@hw.ac.uk; a.halimi@hw.ac.uk; s.mclaughlin@hw.ac.uk).

Digital Object Identifier 10.1109/JSTSP.2022.3170228

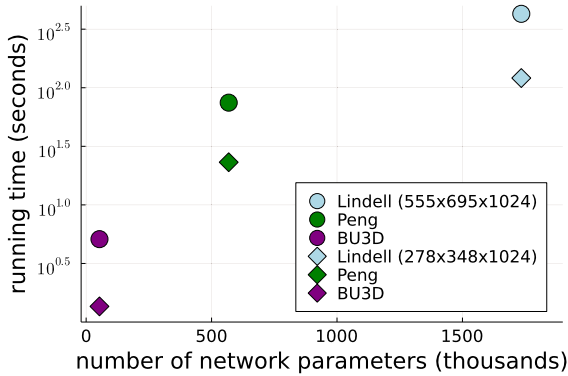


Fig. 1. The proposed model (BU3D) requires fewer parameters and lower running time in high dimensional Lidar data, compared to the state-of-the-art deep learning methods by Lindell [22] and Peng [23]. The running time is tested on two Lidar data cube: one with a size of $555 \times 695 \times 1024$ (circles) and the other with $278 \times 348 \times 1024$ (diamonds).

proposed method is in line with an emerging technique called algorithm unrolling [27], [28]. This technique replaces the steps of a conventional iterative method by neural network blocks, hence exploiting the domain knowledge when designing the network. This framework was first proposed in [27] which unfolded an iterative method known as the iterative shrinkage and thresholding algorithm (ISTA) [29] for sparse coding, and since then, unrolling approach has been successfully applied in many signal processing tasks such as compressive sensing [30], super-resolution [31] and radar imaging [32]. Following the algorithm unrolling approach, our Bayesian-based unrolling model, named BU3D, is made interpretable, so that we can analyze the internal behaviors, and is efficient in terms of the number of network parameters, training time and running time. Fig. 1 compares the running time with respect to the number of network parameters on two different sizes of data. The proposed method is shown to require orders of magnitude fewer parameters and lower computation time, compared to two state-of-the-art deep learning methods of Lindell [22] and Peng [23].

Using multiscale information to process single-photon data is an essential component of several state-of-the-art 3D Lidar reconstruction algorithms. This was exploited in the proposed network which requires an initial estimate of few multiscale depths as input, instead of the large histogram data cube as in [22], [23]. In this way, the knowledge of the system impulse response function (IRF) is exploited to generate the multiscale depths and the high dimensional data is compressed to the essential information. The network layers mimic the iterative steps of the Bayesian algorithm in [26], which alternated between a weighted median step to choose the best depth scale to represent a pixel, and a soft-thresholding step to account for spatial correlations between pixels. Our conversion relies on a popular tool called attention [33]–[37], which computes weights highlighting the features or areas of interest (i.e., areas requiring attention). An attention layer is said to be hard attention [38] if the attention weights are sparse (one-hot encoding), or soft otherwise. In this paper, inspired by the weighted median filter used in [26] and promoting sharp surface’s boundaries, we consider hard

attention to select the best depth scale per pixel, i.e. the one showing the highest attention weight. The proposed network also includes soft attention to improve the 3D object reconstruction by considering local spatial correlations. Results on simulated and real data show the benefit of this model when compared to the state-of-the-art learning-based algorithms [22], [23], as it preserves surface edges and has a lower computational cost (in terms of memory or computational time). Another distinct advantage of the proposed method is in enabling access to uncertainty maps on the predicted depth. The uncertainty maps are obtained by connection to the underlying Bayesian method [26] without additional complexity, while some previous works [39], [40] require multiple passes of inference and averaging steps to predict the uncertainty of the network’s outputs.

In summary, the contributions of this paper are:

- an efficient deep learning model suitable for high-dimensional single-photon Lidar data,
- interpretable neural network blocks, providing uncertainty information on the final depth map,
- a scale selection strategy through a combination of hard and soft attention, showing competitive results when compared to state-of-the-art methods, i.e. fewer artifacts on surface boundaries, and improved robustness to mismodelling effects.

The remainder of the paper is organized as follows. Section II describes the multiscale observation model for single-photon Lidar measurements. Section III reviews the underlying iterative method [26] resulting from a Bayesian hierarchical model. In Section IV, we present the proposed unrolling model with details on the training procedures. In Section V, we analyze the proposed network and evaluate the performance of our method on simulated data as well as real data. Section VI presents the conclusions and future work.

II. MULTISCALE OBSERVATION MODEL

This section presents the considered Poisson-based observation model for single-photon Lidar system, which is required to derive the underlying Bayesian algorithm in Section III. Akin to [26], we include multiscale information in the observation model. Single-photon Lidar systems provide range information about the scene by measuring the time difference between emission of light pulses and detection of photons. Collecting such time delays, the Lidar system builds a histogram of counts denoted by $y_{n,t} \in \{0, 1, 2, \dots\}$ where n represents the pixel index and t the time bin index. The observed photon counts are commonly assumed to follow the Poisson distribution with mean value $s_{n,t}$ as follows $y_{n,t} \sim \mathcal{P}(s_{n,t})$ [9], [12]. Assuming one target per each pixel n , the observation model for $s_{n,t}$ can be written as

$$s_{n,t} = r_n g(t - d_n) + b_n, \quad (1)$$

where r_n is the reflectivity of the target, d_n the depth information of the target, b_n the background photons due to ambient light and detector dark counts and g is the system IRF. We approximate the system IRF g by the Gaussian function $\mathcal{N}(t; \mu, \sigma^2)$ with the mean μ and the standard deviation σ and consider

that $\sum_{t=1}^T g(t - d_n) = 1$ for all n , for all possible depths of the scene [10], [26]. By assuming independent observations between $y_{n,t}, \forall n, t$, the joint likelihood for $\mathbf{Y} = \{y_{n,t}\}$ can be written as

$$p(\mathbf{Y} | \mathbf{d}, \mathbf{r}, \mathbf{b}) = \prod_{n=1}^N \prod_{t=1}^T \frac{s_{n,t}^{y_{n,t}}}{y_{n,t}!} \exp^{-s_{n,t}}, \quad (2)$$

where \mathbf{d} , \mathbf{r} , \mathbf{b} represent the column vectors of size N gathering depth, reflectivity and background parameters, respectively. Without background photons, the maximum likelihood estimate of the reflectivity can be computed as $r_n^{\text{ML}} = \bar{s}_n = \sum_{t=1}^T y_{n,t}$ and the depth as the following (known as the log-matched filter):

$$d_n^{\text{ML}} = \arg \max_d \sum_t y_{n,t} \log g(t - d). \quad (3)$$

In this case, the likelihood can be written to be proportional to the following (See Appendix of [26] for the details)

$$p(\mathbf{y}_n | r_n, d_n) \propto \mathcal{G}(r_n; 1 + \bar{s}_n, 1) Q(\mathbf{y}_n) \times \mathcal{N}(d_n; d_n^{\text{ML}}, \bar{\sigma}^2), \quad (4)$$

where $\mathcal{G}(x; \cdot, \cdot)$ is the gamma distribution with shape and scale parameters, Q is a function of \mathbf{y}_n and $\bar{\sigma}^2 := \sigma^2 / \bar{s}_n$. To be able to deal with high noise in Lidar data, it is common to incorporate multiscale information, as demonstrated in statistical [12], [26] as well as deep learning methods [22], [23], [25]. We employ a similar multiscale approach, using the fact that low-pass filtered histograms (resulting in summing neighbouring pixels) still follow a Poisson distribution. We generate L downsampled histograms $\mathbf{y}_n^{(\ell)}$ with $\ell \in \{2, \dots, L\}$, by spatially downsampling the original histogram data $\mathbf{y}_n^{(1)} := \mathbf{y}_n$ with uniform filters. This multiscale data can be efficiently computed using convolution with different uniform kernel sizes. Assuming the same observation model in (4), the likelihood for each downsampled histogram $\mathbf{y}_n^{(\ell)}$ can be written as

$$p(\mathbf{y}_n^{(\ell)} | r_n^{(\ell)}, d_n^{(\ell)}) \propto \mathcal{G}(r_n^{(\ell)}; 1 + \bar{s}_n^{(\ell)}, 1) Q(\mathbf{y}_n^{(\ell)}) \times \mathcal{N}(d_n^{(\ell)}; d_n^{\text{ML}(\ell)}, \bar{\sigma}^{2(\ell)}), \quad (5)$$

where $\bar{s}_n^{(\ell)} = \sum_{t=1}^T y_{n,t}^{(\ell)}$ and $\bar{\sigma}^{2(\ell)} = \sigma^2 / \bar{s}_n^{(\ell)}$. For example, we can consider $L = 4$ scales with different kernel sizes such as 1×1 , 3×3 , 7×7 and 13×13 . The detailed procedure to generate multiscale depths is explained in Sec. IV-C.

III. UNDERLYING BAYESIAN ALGORITHM

In this section, we review an underlying Bayesian algorithm proposed by Halimi *et al.* [26] which inspired the design of our deep learning method in Section IV. This method [26] follows a Bayesian approach, by considering prior distributions on the unknown depth as well as their uncertainty information. The prior distributions will be combined with the observation model in (5) to derive the posterior distribution, which contains rich information regarding the parameters of interest. To exploit this distribution, the method in [26] approximated the parameter's maximum-a-posteriori (MAP) estimator using a coordinate

descent method. Although this method can estimate both depth and reflectivity, for the purpose of this paper, we only consider estimating depth profiles.

A. Prior and Posterior Distribution

The observation model for multiscale depths $d^{(\ell)}$ is derived in (5). From this multiscale information, the goal now is to estimate the true depth denoted by a latent variable x . A prior is imposed on this latent variable, requiring spatial smoothness within a homogeneous surface while preserving the discontinuity around the boundaries of the surfaces. To satisfy this requirement and estimate a robust depth map, Halimi *et al.* [26] introduced some pre-defined weights called *guidance weights* between local pixels for each scale. A high value of $w_{n',n}^{(\ell)}$ encourages the latent variable x_n to be similar to $d_{n'}^{(\ell)}$. Using the guidance weights, the latent variable x is assigned the conditional Laplace distribution

$$x_n | d_{\nu_n}^{(1,\dots,L)}, w_{\nu_n,n}^{(1,\dots,L)}, \epsilon_n \sim \prod_{n' \in \nu_n} \left[\prod_{\ell=1}^L \mathcal{L} \left(x_n; d_{n'}^{(\ell)}, \frac{\epsilon_n}{w_{n',n}^{(\ell)}} \right) \right] \quad (6)$$

where $\mathcal{L}(\cdot; \mu, \psi)$ is the Laplace distribution with the mean μ and the scale parameter ψ , ν_n represents the local neighbourhood around the n th pixel and ϵ_n is the variance of the depth x_n . To ensure the positivity of the variance ϵ , it is assigned a conjugate inverse gamma distribution as

$$\epsilon \sim \prod_n \mathcal{IG}(\epsilon_n; \alpha_d, \beta_d) \quad (7)$$

where α_d and β_d are user set positive hyperparameters. Combining the prior distributions in (6) and (7) and the likelihood in (5), the posterior distribution reduces to

$$p(x, \epsilon, \mathbf{D} | \mathbf{Y}, \mathbf{W}) \propto p(\mathbf{Y} | \mathbf{D}) p(x, \mathbf{D} | \epsilon, \mathbf{W}) p(\epsilon) \quad (8)$$

where \mathbf{W} represents the guidance weights and \mathbf{D} represents the multiscale depths $d^{(1,\dots,L)}$.

B. Iterative Algorithm

To approximate the parameter's MAP estimates, a coordinate descent method is employed to minimize the negative log-posterior of (8). The algorithm proposed in [26] updates one variable at a time while fixing other variables and is summarized in Algorithm 1. The updates of unknown variables can be divided into three parts. Firstly, the latent variable x_n is updated using a weighted median filtering as follows

$$x_n \leftarrow \underset{x}{\operatorname{argmin}} \mathcal{C}(x) = \sum_{\ell, n' \in \nu_n} w_{n',n}^{(\ell)} |x - d_{n'}^{(\ell)}|. \quad (9)$$

This operation will be replaced by attention mechanisms in the proposed deep learning model in the squeeze block in Section IV-A. Secondly, the multiscale depths $d^{(1,\dots,L)}$ are updated by minimizing the negative log-conditional distributions of \mathbf{D}

Algorithm 1: Iterative Bayesian Algorithm [26].

- 1: Input: Lidar data \mathbf{Y} , the number of scales L
 - 2: Construct downsampled histograms $\mathbf{Y}^{(1,\dots,L)}$
 - 3: Compute initial multiscale depths $\mathbf{d}^{\text{ML}(1,\dots,L)}$
 - 4: Compute the guidance weights \mathbf{W}
 - 5: **while** not converge **do**
 - 6: Update the variable \mathbf{x} by (9)
 - 7: Update the multiscale depths $\mathbf{d}^{(1,\dots,L)}$ by (10)
 - 8: Update the uncertainty information by (12)
 - 9: **break** if the convergence criteria are satisfied
 - 10: **end while**
 - 11: Output: \mathbf{x}, ϵ
-

Algorithm 2: Proposed Unrolling Method (BU3D).

- 1: Input: Lidar data \mathbf{Y}
 - 2: // number of scales L , number of stages K
 - 3: Construct downsampled histograms // Sec. IV-C
 - 4: Compute initial multiscale depths \mathbf{d}^{ML} // Sec. IV-C
 - 5: **for** $k = 1, 2, \dots, K - 1$ // k th stage
 - 6: Update \mathbf{x} by the squeeze block // Sec. IV-A 1
 - 7: Update $\mathbf{d}^{(1,\dots,L)}$ by the expansion block // Sec. IV-A 2
 - 8: **end for**
 - 9: Compute \mathbf{x} by the squeeze block // Sec. IV-A 1
 - 10: Compute uncertainty information ϵ by (20)
 - 11: Output: \mathbf{x}, ϵ
-

in (8) as follows:

$$d_n^{(\ell)} \leftarrow \underset{d}{\operatorname{argmin}} \frac{[d - d_n^{\text{ML}(\ell)}]^2}{2\bar{\sigma}^2(\ell)} + \sum_{n' \in \nu_n} \frac{w_{n,n'}^{(\ell)} |d - x_{n'}|}{\epsilon_{n'}}. \quad (10)$$

This operator is known as a generalized soft-thresholding operator and can be solved analytically [41]. It will be replaced by the expansion block in the proposed network in Section IV-A. Lastly, given the estimations of \mathbf{x} and \mathbf{d} , the depth uncertainty information can be evaluated by considering the depth variance. The conditional distribution of ϵ is given by

$$\epsilon_n \mid \mathbf{x}, \mathbf{D}, \mathbf{W} \sim \mathcal{IG} [L\bar{N} + \alpha_d, \mathcal{C}(x_n) + \beta_d], \quad (11)$$

where $\bar{N} = |\nu_n|$ is the number of neighbors considered. The mode of this distribution represents the MAP estimator of ϵ_n and is given by

$$\hat{\epsilon}_n \leftarrow (\mathcal{C}(x_n) + \beta_d) / (L\bar{N} + \alpha_d + 1). \quad (12)$$

This formula will subsequently provide a basis to estimate the uncertainty of the depth map estimated by the neural network, explained in Sec. V-A. As mentioned in the previous subsection, the guidance weights \mathbf{W} connect the latent variable \mathbf{x} to multiscale depths. These weights play an important role in the performance of the algorithm. Halimi *et al.* [26] determines the weights \mathbf{W} based on the deviation of $\mathbf{d}^{\text{ML}(\ell)}$, $\forall \ell$, from a given reference depth map, while this paper proposes to learn them from the data, as described in the following section.

IV. PROPOSED UNROLLING METHOD

Motivated by Algorithm 1, we propose a Bayesian-based Unrolling model for 3D Lidar imaging (BU3D). As mentioned in Section I, the main idea of algorithm unrolling is to unfold an underlying iterative method and mimic its operations with neural network blocks. Here, we replace the operations of Algorithm 1 by neural network layers. The major components of the proposed network use attention modules, which allow learning the weights \mathbf{W} , i.e., the correlations between local pixels at the multiscale depths. The proposed method for inference (after training) is summarized in Algorithm 2.

A. Network

Fig. 2 gives an overview of the proposed neural network. The network inputs an initial estimation of multiscale depths $\mathbf{d}^{\text{ML}(1,2,\dots,L)}$ (See Section IV-C); and outputs the estimated depth \mathbf{x} . The network consists of K stages where each stage has the same structure (except for the last stage) with different network parameters and is designed to resemble one iteration of Algorithm 1. Each stage begins with the feature extraction step having three consecutive convolution layers to obtain \mathbf{d}_{feat} . After that, each stage has two main blocks: *squeeze* and *expansion*. In this subsection, for the simplicity of notation, we use the variable symbols for the first stage and omit the dependency on the stage k unless explicitly mentioned.

1) *The Squeeze Block (Unrolling Line 6 in Algo. 1)*: This is a key element in the network as it estimates a single depth by using the multiscale depths and their features, as shown in Fig. 3. This block is inspired by the weighted median filtering (9) in line 6 of Algorithm 1. It considers hard attention [38] to select the scale with the highest attention weight for each pixel, and takes the single depth value on that scale. Specifically, the squeeze block first computes attention weights in a module named *PACConv (cube)*, a variant of the so-called pixel attention [37]. Pixel attention considers attention weights at a pixel level, so that the attention weights have the same size as the input. The module *PACConv (cube)* computes internal weights which are multiplied by the depth features, yielding the attention weights $\mathbf{w}^{(1,2,\dots,L)}$. The latter weights indicate the importance of each scale, and only one scale is chosen by the argmax operation, yielding the squeezed depth. Formally, the squeezed depth for the n th pixel is computed by

$$x_n = d_n^{(\ell')}, \quad \ell' = \underset{\ell \in \{1,\dots,L\}}{\operatorname{argmax}} w_n^{(\ell)}, \quad (13)$$

where $w_n^{(\ell)}$ denotes the attention weight for the ℓ th scale. Since the argmax operation is not differentiable, we replace it with the alternative differentiable Gumbel-SoftMax [42], [43].

2) *The Expansion Block (Unrolling Line 7 in Algo. 1)*: This block refines multiscale depths to obtain $\bar{\mathbf{d}}$ as the weighted average between multiscale depths \mathbf{d} and the squeezed depth \mathbf{x} . This expansion block corresponds to the soft-thresholding step (10) in line 7 of Algorithm 1, as it updates the multiscale depths based on

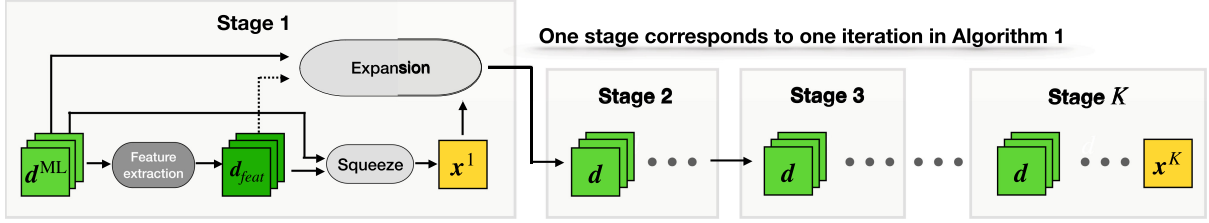


Fig. 2. Overview of the proposed network. The network consists of K stages and each stage inputs a tuple of multiscale depths (\mathbf{d}^{ML}), estimates a squeezed depth (\mathbf{x}) and refines the multiscale depths. The final stage's output of the network is a squeezed depth (\mathbf{x}^K). For the illustration, the network is shown for the case of three multiscales $L = 3$.

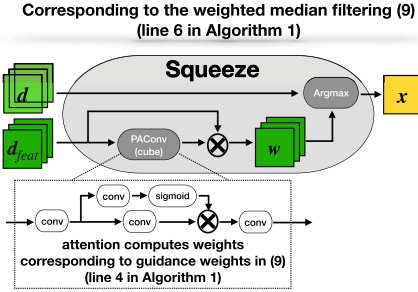


Fig. 3. The squeeze block estimates a squeezed depth (\mathbf{x}) from a tuple of multiscale depths (\mathbf{d}) and their features (\mathbf{d}_{feat}). The symbol \otimes denotes elementwise multiplication.

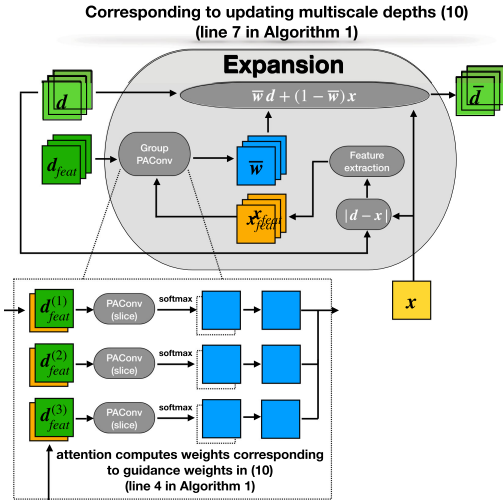


Fig. 4. The expansion block refines the previous multiscale depths (\mathbf{d}), by comparing their features (\mathbf{d}_{feat}) with the feature of the absolute difference between multiscale depths and the squeezed depth ($|\mathbf{d} - \mathbf{x}|$). This block outputs the refined multiscale depths ($\bar{\mathbf{d}}$).

the obtained squeezed depth. However, we design the expansion block to exploit the attention framework, rather than mimicking exactly the soft-thresholding operator (10). As shown in Fig. 4, the expansion block has three inputs: the multiscale depths, their features \mathbf{d}_{feat} and the squeezed depth \mathbf{x} . As indicated by the weighted average, the weights will help combine values from either the outlier-free squeezed depth \mathbf{x} or the multiscale depths $\mathbf{d}^{(\ell)}$, where \mathbf{x} values will be promoted if the two depths are significantly different. This highlights the importance of the absolute difference between the multiscale depths and the

squeezed depth for each scale (i.e., $\forall \ell, |\mathbf{d}^{(\ell)} - \mathbf{x}|$). The latter difference is fed together with the multiscale depth features \mathbf{d}_{feat} into a module named *Group PACConv* to compute the weights. This module consists of L independent sub-modules, where the l th sub-module inputs one scale depth feature $\mathbf{d}_{feat}^{(\ell)}$ and the difference feature $|\mathbf{d}^{(\ell)} - \mathbf{x}|$. From such input, an attention module named *PACConv (slice)* estimates internal weights for each scale. A softmax operator is then applied to these internal weights after multiplying them by a coefficient ρ (note that ρ is introduced to enforce weights sparsity and is fixed to $\rho = 2$ throughout the paper). The softmax operator outputs two normalized channels per scale, we only consider the first channel related to the multiscale depth parts and denoted by \bar{w} (See the red rectangles in the bottom part of Fig. 4). Then, the expanded depth $\bar{d}_n^{(\ell)}$ for the l th scale and the n th pixel is obtained by the convex combination of $d_n^{(\ell)}$ and x_n as follows:

$$\bar{d}_n^{(\ell)} = \bar{w}_n^{(\ell)} d_n^{(\ell)} + (1 - \bar{w}_n^{(\ell)}) x_n, \quad 0 \leq \bar{w}_n^{(\ell)} \leq 1. \quad (14)$$

We have described one stage of the network corresponding to one iteration in Algorithm 1. All stages have the same structure, except the last which only has the squeeze block to produce the final output of the network.

3) *Network Learnable Parameters*: Throughout the network, all the convolution layers use the 3×3 kernel with the LeakyReLU activation [44] without bias and have the same number of output channels as the input. For example, if an input of the convolution layer is of size $12 \times \text{Height} \times \text{Weight}$, the output size will be the same and the number of learnable parameters on this layer is $3 \times 3 \times 12 \times 12 = 1296$ parameters, following the structure of standard convolutional layers [45]. The module *Group PACConv* consists of 12 independent sub-modules where each has 144 learnable parameters from 4 convolutions whose input and output channels are 2 (i.e., $3 \times 3 \times 2 \times 2 = 144$ parameters). Therefore, each stage contains 14688 learnable parameters, except for the last one which has 9072 parameters. Table I summarizes the network operations together with the corresponding number of learnable parameters, when considering $K = 4$ stages.

4) *Property of the Network*: Interestingly, the final depth value of the proposed network is bounded pixelwise by the initial estimates of the multiscale depths. To state formally, consider the values of the multiscale depths $d_n^{(\ell)}, \forall \ell$ and the squeezed depth x_n at the first stage. Since the squeeze block chooses an

TABLE I

SUMMARY OF THE ARCHITECTURE WITH THE OUTPUT SHAPE AND THE NUMBER OF LEARNABLE PARAMETERS. STAGE 2 AND 3 HAVE THE SAME STRUCTURE AS STAGE 1. CONV DENOTES THE CONVOLUTION LAYER, AND H AND W REPRESENT THE HEIGHT AND WIDTH OF AN ARRAY, RESPECTIVELY

Stage	Block	Layers	Output size	Parameters
1	Feature extract.	(3 Conv layers)	$H \times W \times 12$	3,888
		Squeeze	PACConv (4 Conv layers)	$H \times W \times 12$
	Expansion	Argmax	$H \times W \times 1$	
		Feature extract. (3 Conv)	$H \times W \times 12$	3,888
		Group PACConv	$H \times W \times 12$	1,728
	Convex combination	$H \times W \times 12$		
2			$H \times W \times 12$	14,688
3			$H \times W \times 12$	14,688
4	Feature extract.	(3 Conv layers)	$H \times W \times 12$	3,888
		Squeeze	PACConv (4 Conv layers)	$H \times W \times 12$
		Argmax	$H \times W \times 1$	
Total				53,136

element among L elements of $d_n^{(\ell)}$ with $\ell \in \{1, \dots, L\}$, it holds that

$$\min \{d_n^{(1)}, \dots, d_n^{(\ell)}\} \leq x_n \leq \max \{d_n^{(1)}, \dots, d_n^{(\ell)}\}. \quad (15)$$

Meanwhile, the expanded depths denoted by $\bar{d}_n^{(\ell)}$ are a convex combination of $d_n^{(\ell)}$ and x_n with normalized weights in Eq. (14), so we have

$$\min \{d_n^{(\ell)}, x_n\} \leq \bar{d}_n^{(\ell)} \leq \max \{d_n^{(\ell)}, x_n\}. \quad (16)$$

Combining (15) and (16), the expanded depths are bounded pixelwise by the initial multiscale depths

$$\min \{d_n^{(1)}, \dots, d_n^{(\ell)}\} \leq \bar{d}_n^{(\ell)} \leq \max \{d_n^{(1)}, \dots, d_n^{(\ell)}\}. \quad (17)$$

Since each stage has the same structure, this relation holds for the next stages and the final squeezed depth value has the same bound as in (15). This property has pros and cons. We can predict the behaviour of the network, so that it will not produce some extreme depth values. On the other hand, the proposed network requires the range of initial multiscale depths to cover the underlying true depth for each pixel.

B. Loss

Motivated by the Laplace prior in (6), we define the training loss for depths as the ℓ_1 -norm distance between the predicted depth and ground-truth depth. We additionally impose a constraint that the intermediate squeezed depths should be similar to the ground-truth depth \mathbf{x}^* during training. The motivation for this constraint is twofold. It can prevent the neural network from losing key information in the initial stages and it can help avoid the vanishing gradient problem, by providing more paths in computational graphs for backpropagation. With the additional constraint, the training loss function \mathcal{L} is defined as

$$\mathcal{L}(\theta) = \sum_{k=1}^K \|\mathbf{x}^k(\theta) - \mathbf{x}^*\|_1, \quad (18)$$

where θ denotes the neural network parameters, \mathbf{x}^k represents the intermediate squeezed depth in the k stage and K is the total number of stages.

 TABLE II
 PROCEDURE TO ESTIMATE INITIAL MULTISCALE DEPTHS

1. Filter Lidar data	2. Downsample with the filter size	3. Estimate initial depths
$\mathbf{Y} * g$ (cross correlation)	1×1	$\mathbf{d}^{\text{ML}(1)}$
\mathbf{Y} : Lidar data	3×3	$\mathbf{d}^{\text{ML}(2)}$
g : System IRF	7×7	$\mathbf{d}^{\text{ML}(3)}$
	13×13	$\mathbf{d}^{\text{ML}(4)}$
$\mathbf{Y} * g * \omega_1$	1×1	$\mathbf{d}^{\text{ML}(5)}$
ω_1 : Uniform filter kernel (size: $7 \times 7 \times 7$)	3×3	$\mathbf{d}^{\text{ML}(6)}$
	7×7	$\mathbf{d}^{\text{ML}(7)}$
	13×13	$\mathbf{d}^{\text{ML}(8)}$
$\mathbf{Y} * g * \omega_2$	1×1	$\mathbf{d}^{\text{ML}(9)}$
ω_2 : Uniform filter kernel (size: $13 \times 13 \times 13$)	3×3	$\mathbf{d}^{\text{ML}(10)}$
	7×7	$\mathbf{d}^{\text{ML}(11)}$
	13×13	$\mathbf{d}^{\text{ML}(12)}$

C. Estimation of Initial Multiscale Depths

As a reminder, the input of the proposed network is a tuple of multiscale depths, rather than the large volume histogram data. From the histogram data, we aim at extracting initial multiscale depths without losing important information, while providing several depth values to cover the true one. For this goal, we consider several 3D low-pass filtered Lidar histograms as summarized in Table II. We first apply the cross correlation to the original Lidar data with the system IRF. To this cross correlated data, we apply the 3D convolution with the uniform filters with the sizes of $7 \times 7 \times 7$ and $13 \times 13 \times 13$, generating two additional histograms. Each of the three histograms is then spatially downsampled with 4 different kernel sizes. This results in 12 filtered histograms in total, where we locate the main peak's position in each filtered histogram and for each pixel, to obtain initial multiscale depths $\mathbf{d}^{\text{ML}(\ell)}, \forall \ell$. Note that we can exploit the separability of uniform filters for efficient computation. Note also that the actual IRF could be used and that we do not impose any constraints on the IRF shape. It is worth mentioning that previous deep learning models [22], [23] do not account for a known system IRF in their architectures, but might learn it implicitly during training.

D. Training Procedures

To train the neural network, we generate synthetic data by simulating SPAD measurements with $T = 1024$ time bins, using the Poisson observation model in (1). We choose 9 scenes from the Middlebury stereo dataset [46] (with image sizes 555×650) and 21 scenes from the Sintel stereo dataset [47] (with image sizes 436×1024) for the training dataset and 2 scenes from [47] for the validation set. To make our network robust to different noise levels, we consider different scenarios based on the average number of Photons-Per-Pixel (PPP); and the average Signal-to-Background Ratio (SBR), defined as

$$\text{PPP} = \frac{1}{N} \sum_{n=1}^N (r_n + b_n T), \quad \text{SBR} = \frac{\sum_{n=1}^N r_n}{\sum_{n=1}^N b_n T}.$$

We found it helpful to include the clean data to prevent overfitting where the network favors some specific scales and so we

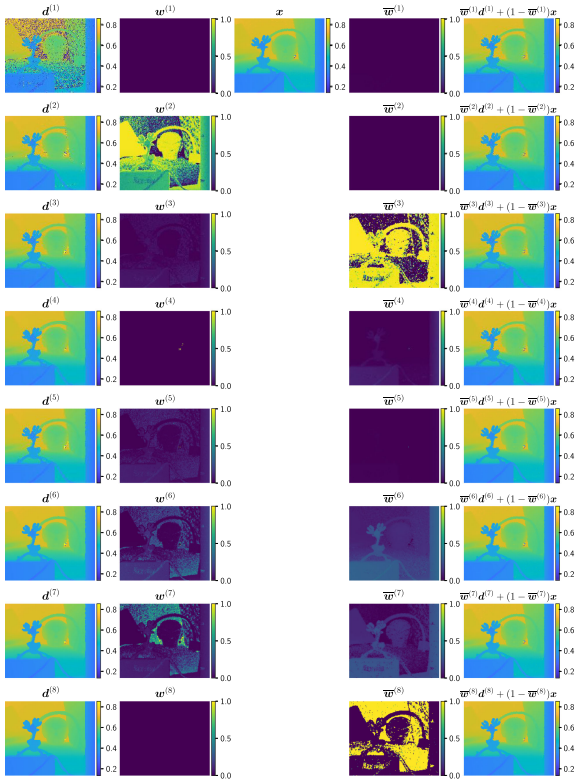


Fig. 5. Visualization of the internal outputs in the first stage. For the purpose of presentation, only eight scales ($\ell = 1, \dots, 8$) are visualized. [1st column] shows the initial multiscale depths (\mathbf{d}) and [2nd column] their corresponding attention weights (\mathbf{w}) in the squeeze block. [3rd column] shows the squeezed depth (\mathbf{x}), [4th column] the attention weights in the expansion block and [5th column] the expanded multiscale depths computed by Eq. (14).

consider 4 cases: (PPP=1, SBR=1), (PPP=1, SBR=64), (PPP=64, SBR=1), (PPP=64, SBR=64). To save the GPU memory during training, we extract patches of size 256×256 with stride 48, rather than processing the original images. We implement our model in PyTorch and use ADAM [48] as an optimizer with the default hyperparameter ($\beta_1=0.9, \beta_2=0.999$) and the batchsize 16. We train the model for 200 epochs with the initial learning rate 0.0001 which is reduced by half at epoch 100. The training was performed on a Linux server with a NVIDIA RTX 3090 GPU, which takes about 9 hours.

V. EXPERIMENTAL RESULTS

In this section, we perform the experiments to analyze our model and show the relative advantages over other reconstruction methods on synthetic datasets as well as real datasets.

A. Analysis of the Network

Test dataset: For the test data, we simulated Lidar data with $T = 1024$ time bins, from two scenes of *Art* (555×695) and *Reindeer* (555×671) in the Middlebury stereo dataset [46] which did not belong to our training sets. The reference depth and reflectivity maps of these two scenes are visualized in the first column of Figs. 7 and 9, respectively. In particular, the Reindeer scene contains extremely low-photon regions which

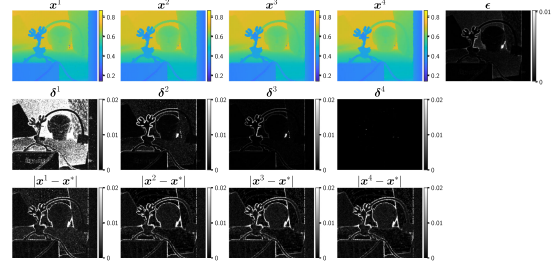


Fig. 6. Intermediate squeezed depths along the four stages (top), the difference between the multiscale depths and the corresponding squeezed depth in each stage (middle) and the errors between the squeezed depths and the ground-truth (bottom). The last column shows our estimated uncertainty map.

TABLE III
EFFECT OF THE NUMBER OF STAGES K AND THE NUMBER OF SCALES L . THE ERROR DAE IS PRESENTED WITH THE MEAN VALUES AND THE STANDARD DEVIATION, EVALUATED ON 98 DIFFERENT LIDAR DATA WITH DIFFERENT PPP AND SBR ON THE ART AND REINDEER SCENE

K	L	DAE	Parameters	Run time (sec)
2	12	0.0055 ± 0.0129	23,760	5.1 ± 0.1
3		0.0046 ± 0.0119	38,448	
4		0.0040 ± 0.0082	53,136	
5		0.0043 ± 0.0112	67,824	
4	4	0.0074 ± 0.0208	5,841	
	8	0.0053 ± 0.0151	24,768	4.0
	12	0.0040 ± 0.0082	53,136	5.1

will challenge the reconstruction algorithms. Note that these test data are larger than the data reported in previous deep learning works [22], [23].

Interpretability: Thanks to our unrolling strategy, we can interpret our neural network via the connection to the underlying Bayesian method. We first inspect whether the first stage can successfully discard outliers. Fig. 5 visualizes the outputs of the internal blocks in the first stage. Each pixel of the squeezed depth \mathbf{x} (3rd column) is obtained using (13), where the multiscale depths and weights are represented in the 1st and 2nd columns, respectively. As shown in the first row, the first scale depth $\mathbf{d}^{(1)}$ shows many outliers, which leads to zero values in $\mathbf{w}^{(1)}$. On the other hand, the second scale depth $\mathbf{d}^{(2)}$ contains important features with less noise and its attention weight $\mathbf{w}^{(2)}$ contains many high values. We still observe noise in $\mathbf{d}^{(2)}$ especially around the low-photon regions. In such areas, $\mathbf{d}^{(4)}$ and $\mathbf{d}^{(7)}$ show more smoothed depth values and $\mathbf{w}^{(4)}, \mathbf{w}^{(7)}$ receive higher attention weights. In this way, the proposed network can successfully remove noise and discard many outliers in the first stage. The 4th column shows the attention weights for the expansion block where $\bar{\mathbf{w}}^{(1)}$ has only zero values, which indicates that the first scale depth $\mathbf{d}^{(1)}$ will be discarded in the next stage.

We now investigate how the squeezed depths improve along the stages. To quantify the change within each stage, we define the difference δ^k in the k th stage between the multiscale depths and the corresponding squeezed depth, for each pixel n , as follows:

$$\delta_n^k = \frac{1}{L+2} \sum_{\ell=1}^L |x_n^k - d_n^{k,(\ell)}|, \quad (19)$$

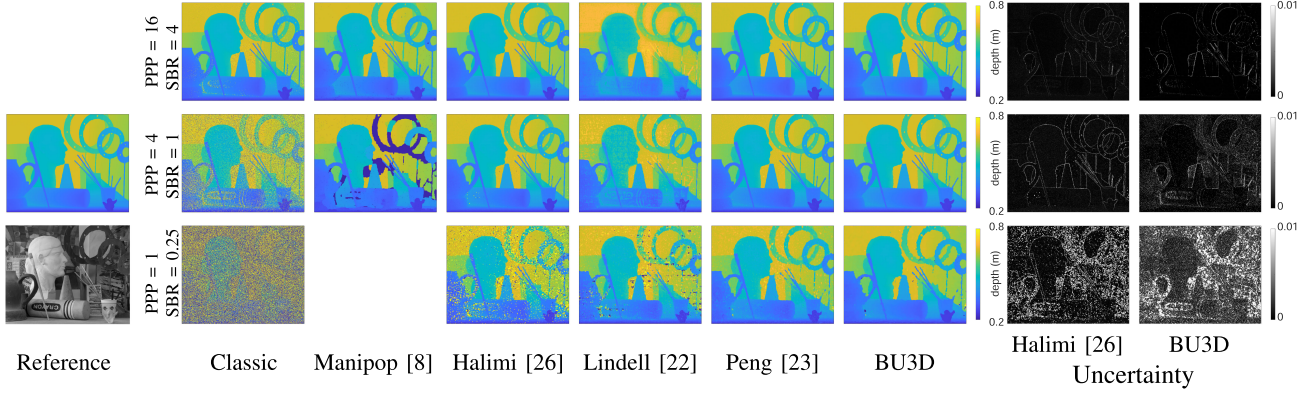


Fig. 7. Reconstructed depth maps with different PPP and SBR levels on the Art scene. The first column shows the reference depth map (middle) and the reflectivity image (bottom). The last two columns show the estimated uncertainty maps by Halimi and the proposed method (BU3D).

where $d_n^{k,(\ell)}$ is the multiscale depth value in the k th stage. A small value of δ would indicate a small improvement so that we may not need a further stage. We also define the uncertainty ϵ of our final depth map \mathbf{x}^K via the connection to the mode of the depth variance (11) in the underlying method with $\bar{N} = 1$, as follows:

$$\epsilon_n = \frac{1}{K-1} \sum_{k=1}^{K-1} \frac{C_n^k + \beta_d}{L+2+\alpha_d}, \quad C_n^k = \sum_{\ell=1}^L \bar{w}_n^{k,(\ell)} |d_n^{k,(\ell)} - x_n^K|, \quad (20)$$

where \bar{w}^k is the softmax-normalized version of $1 - \bar{w}^k$, ensuring $\sum_{\ell=1}^L \bar{w}_n^{k,(\ell)} = 1$. The weights \bar{w} play a similar role to the guidance weights in (10), and we set the hyperparameters α_d, β_d to small values to obtain a non-informative prior. Fig. 6 shows the intermediate squeezed depths \mathbf{x}^k for each stage k and δ^k which decreases along the stages. The last row shows the errors between the squeezed depths and the ground-truth depth map \mathbf{x}^* . In the first stage, the errors appear on the background due to outliers, but such errors decrease along the stages. The last column shows the estimated uncertainty map ϵ . It indicates high uncertainty around object edges and areas with low reflectivity.

Evaluation metrics: To analyze our model quantitatively, we use three evaluation metrics. We employ a standard metric, Depth Absolute Error (DAE) defined as $\text{DAE}(\mathbf{x}, \mathbf{x}^*) = \frac{1}{N} \|\mathbf{x} - \mathbf{x}^*\|_1$, where N is the number of pixels, which is useful for measuring the overall disparity quality. To better evaluate surface boundaries, we use an additional metric called Soft Edge Error (SEE) [49], which measures the local error only at the edges. Formally, it is defined as

$$\text{SEE}(\mathbf{x}, \mathbf{x}^*) = \gamma \sum_{n \in \text{Edge}(\mathbf{x}^*)} \min_{j \in \nu_n} |x_j - x_j^*|,$$

where ν_n is a 3×3 local window around the n th pixel, $\gamma := 10/|\text{Edge}(\mathbf{x}^*)|$ is a scale factor, and $\text{Edge}(\mathbf{x}^*)$ represents a set of edge locations in the ground-truth depth map \mathbf{x}^* obtained using the Canny edge detector [50]. We also report the root mean square error: $\text{RMSE}(\mathbf{x}, \mathbf{x}^*) = \sqrt{\|\mathbf{x} - \mathbf{x}^*\|_2^2 / N}$, as previously used in [22], [23].

Ablation study: We study the effect of the number of stages K and scales L . As shown in Table III, we first fix $L = 12$ and vary

the number of stages K from 2 to 5. We evaluate the performance on 98 different Lidar data with different levels of PPP and SBR both ranging from 0.25 to 1024. We note a decreasing error for increasing number of stages, but when $K = 5$, the error is shown to increase possibly due to overfitting to our training data. For example, we observed that the case $K = 5$ gives a worse performance than that of $K = 4$ when SBR is less than 1 which did not belong to our training set. The number of stages affects the running time by a small margin, because most of the computational cost comes from generating the initial multiscale depths. Next, we test the effect of the total number of scales explained in Section IV-C. The number of scales L affects the error, the number of parameters and the running time. To balance the trade-off between the performance and the network size, we choose $K = 4$ and $L = 12$ throughout the rest of the experiments.

B. Results on Simulated Data

In this experiment, we use the same simulated dataset and the evaluation metrics described in the previous subsection.

Comparison methods: We compare our model to existing reconstruction methods without additional sensor fusion. We consider a state-of-the-art statistical method called Manipop [8] and the underlying iterative Bayesian method in Algorithm 1 by Halimi *et al.* [26]. In Algorithm 1, we consider the same filter size consistent to the case of $L = 4$ in Table II. We also compare to two state-of-the-art deep learning models: Lindell *et al.* [22] and Peng *et al.* [23]. We use the publicly available pre-trained model for [22] and as no pre-trained model is available for [23], we train this model using the authors' publicly available codes. We also report the result of the classical algorithm obtained by applying a matched filter to the Lidar data by the system IRF.

Qualitative comparison: Fig. 7 shows the reconstructed depth maps on the Art scene. In the case of high PPP and SBR, all the methods reconstruct well except for Lindell *et al.* [22] which loses some details. In the challenging data case, we first notice that Manipop is conservative and shows many zero pixels to indicate the absence of a target in them. Other algorithms detect more targets, with the proposed algorithm showing the best robustness to outliers. These results are confirmed in Fig. 8

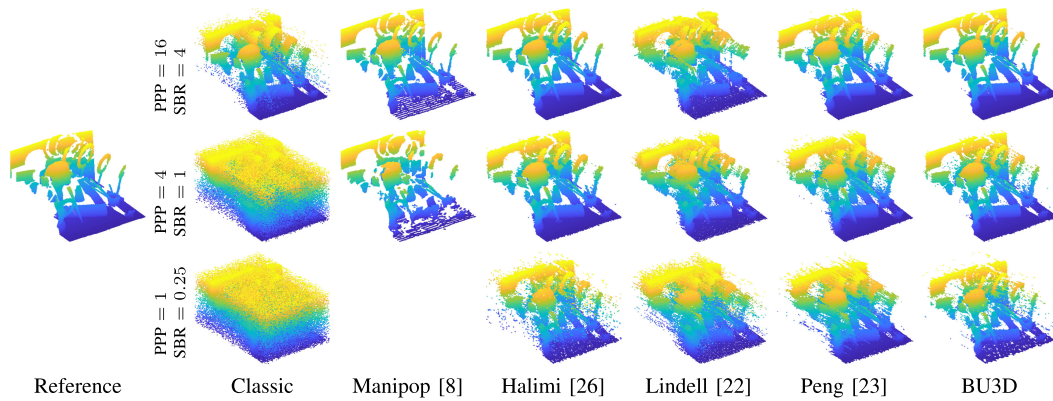


Fig. 8. Point cloud representation of reconstruction results on the Art scene. The first column shows the reference point cloud.

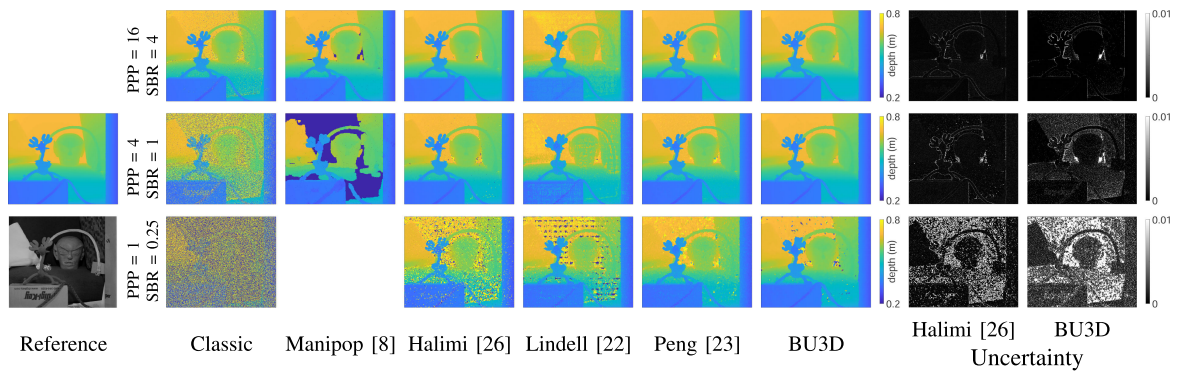


Fig. 9. Reconstructed depth maps with different PPP and SBR levels on the Art scene. The first column shows the reference depth map (middle) and the reflectivity image (bottom). The last two columns show the estimated uncertainty maps by Halimi and the proposed method (BU3D).

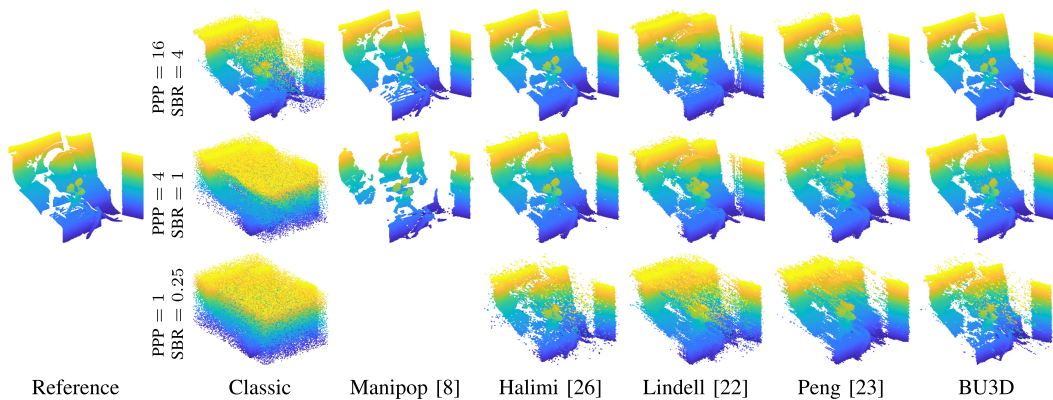


Fig. 10. Point cloud representation of reconstruction results on the Art scene. The first column shows the reference point cloud.

showing the point cloud representation of the reconstruction results. When $PPP=4$ and $SBR=1$, the previous deep learning methods suffer from so-called flying pixel artifacts (also called bleeding effects) [51] around the surface boundaries, while less artifacts are observed in Halimi *et al.* [26] and in our method. When $PPP=1$ and $SBR=0.25$, Halimi *et al.* [26] yields many outliers, compared to the proposed method. These two methods estimate similar uncertainty maps, however, the proposed method indicates higher uncertainty in noisy regions (e.g., see region behind the cone for $PPP=4$, $SBR=1$). Consistent results are observed for the Reindeer scene in Fig. 9 and Fig. 10.

Quantitative comparison: In Table IV, we quantitatively evaluate our model in different levels of PPP and SBR. Peng *et al.* [23] overall outperforms other methods in terms of RMSE, but it yields high errors in DAE and SEE. One reason is due to oversmoothing artifacts around the boundaries of surfaces. On the other hand, our method yields the lowest errors when $PPP=1$ in terms of DAE and SEE. When PPP is 16, both Halimi [26] and the proposed method show an overall good performance in terms of DAE and SEE. Manipop shows good performance for clean data at $PPP=16$ and $SBR=4$, but its errors are higher in challenging cases, because Manipop sets as zero

TABLE IV
 QUANTITATIVE COMPARISON ON THE ART AND REINDEER SCENE WITH DIFFERENT LEVELS OF PPP AND SBR

	Art, SBR = 0.25			Art, SBR = 4			Reindeer, SBR = 0.25			Reindeer, SBR = 4		
	DAE	SEE	RMSE	DAE	SEE	RMSE	DAE	SEE	RMSE	DAE	SEE	RMSE
PPP = 1												
Manipop [8]	-	-	-	0.1883	1.3787	0.3161	-	-	-	0.2579	1.2064	0.3919
Halimi [26]	0.2188	0.3269	0.5739	0.0206	0.0170	0.1422	0.1581	0.1550	0.4891	0.0110	0.0076	0.0874
Lindell [22]	0.0489	0.0597	0.2040	0.0115	0.0172	0.0323	0.0916	0.0874	0.3112	0.0111	0.0308	0.0313
Peng [23]	0.0170	0.0424	0.0722	0.0060	0.0130	0.0151	0.0132	0.0425	0.0371	0.0066	0.0115	0.0166
BU3D	0.0145	0.0336	0.0827	0.0037	0.0121	0.0209	0.0101	0.0207	0.0493	0.0042	0.0061	0.0219
PPP = 4												
Manipop	-	-	-	0.0145	0.0503	0.0528	-	-	-	0.0241	0.1287	0.1103
Halimi	0.0452	0.0211	0.2444	0.0046	0.0030	0.0508	0.0229	0.0174	0.1636	0.0028	0.0019	0.0159
Lindell	0.0239	0.0725	0.0428	0.0787	0.5421	0.0881	0.0273	0.1003	0.0419	0.0735	0.4967	0.0822
Peng	0.0073	0.0161	0.0275	0.0043	0.0056	0.0112	0.0068	0.0132	0.0193	0.0041	0.0046	0.0108
BU3D	0.0052	0.0141	0.0326	0.0026	0.0057	0.0150	0.0050	0.0084	0.0253	0.0027	0.0019	0.0153
PPP = 16												
Manipop	0.0930	0.0643	0.2171	0.0038	0.0039	0.0261	0.1753	0.0631	0.3271	0.0063	0.0060	0.0545
Halimi	0.0097	0.0065	0.1009	0.0024	0.0010	0.0334	0.0040	0.0023	0.0390	0.0013	0.0008	0.0045
Lindell	0.0296	0.0669	0.0511	0.0280	0.0518	0.0455	0.0253	0.1005	0.0430	0.0212	0.0528	0.0312
Peng	0.0044	0.0067	0.0144	0.0031	0.0032	0.0114	0.0043	0.0051	0.0120	0.0027	0.0026	0.0065
BU3D	0.0029	0.0082	0.0226	0.0019	0.0014	0.0126	0.0028	0.0027	0.0174	0.0019	0.0008	0.0129

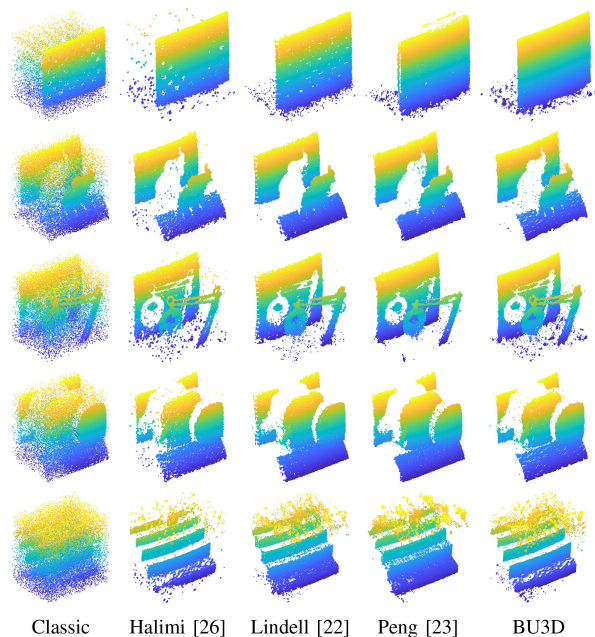
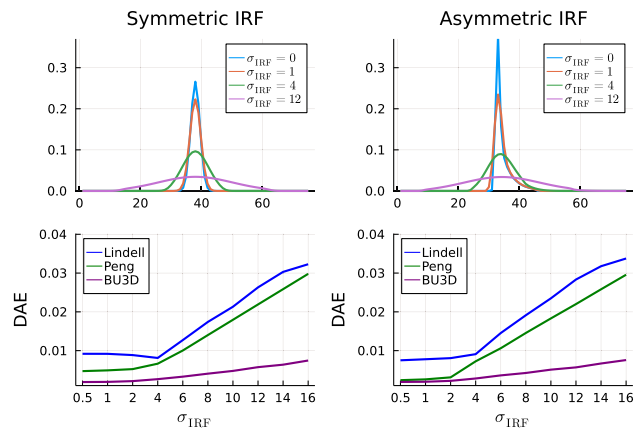


Fig. 11. Errors in terms of different levels of SBR and PPP on the Art scene by three methods: Halimi, Peng and the proposed method (BU3D) (top-to-bottom). Three evaluation metrics of DAE, SEE and RMSE are used (left-to-right) and the error values are presented in log scale.

non-target regions. Although the used metrics are not fair to Manipop, we report the errors for reference and we empty the numbers when they are not meaningful.

As shown in Fig. 11, we further conduct an extensive experiment in a wide range of PPP and SBR levels when comparing with Halimi *et al.* [26] and Peng *et al.* [23]. The underlying Bayesian method [26] gives an excellent performance on the clean data, but its performance rapidly degrades in the low-photon and high noise cases. In such cases, Peng [23] and our method show comparable results, but the errors in Peng’s method begin to increase when PPP is higher than 16 and SBR is higher


 Fig. 12. Generalization test on different system IRFs. When generating test data, different system IRFs are considered by applying Gaussian smoothing with varying standard deviations σ_{IRF} on the two baseline IRFs: a Gaussian IRF (top-left) and a realistic asymmetric IRF (top-right). The bottom row shows the DAE by Lindell, Peng and the proposed method (BU3D) with varying σ_{IRF} from the Gaussian IRF (bottom-left) and the asymmetric IRF (bottom-right).

than 512. Compared to Peng’s result, the proposed method offers a more consistent performance even when PPP and SBR are high.

Generalizability on different system IRFs: Unlike previous work [22], [23], the proposed method incorporates the system IRF, as explained in Section IV-C. Here, we test how robust our method is to changes affecting the system IRF. We consider two types of baseline IRFs with 15 non-zero time bins: a symmetric IRF given by the Gaussian function and a realistic asymmetric IRF. On these baseline IRFs, we apply a Gaussian smoothing with different standard deviations σ_{IRF} and use the resulting IRF to generate test data on the Art scene with PPP=4 and SBR=4. The first row of Fig. 12 shows the shapes of different IRFs where a large value of σ_{IRF} increases the IRF’s width. The second row shows the errors with respect to σ_{IRF} when considering the compared networks (without

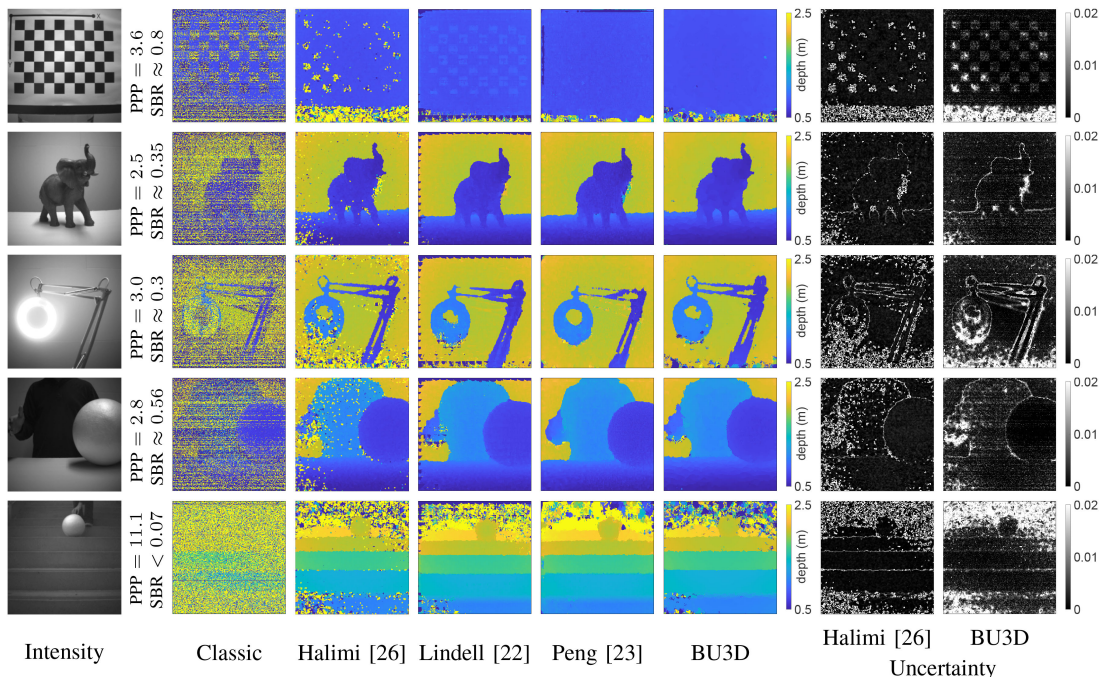


Fig. 13. Reconstructed depth maps on the real dataset. The first column shows a reference intensity image and the last two columns show the uncertainty maps estimated by Halimi and the proposed method (BU3D).

TABLE V

RUN TIME ON THE ART SCENE WITH THE RESOLUTION OF $555 \times 695 \times 1024$. THE RUN TIME OF THE PROPOSED METHOD (BU3D) IS PRESENTED INTO TWO PARTS: THE ESTIMATION OF INITIAL MULTISCALE DEPTHS AND THE INFERENCE BY THE NETWORK

Method	Device	Run time (sec)	Train time	Parameters
Classic		13.7		
Manipop [8]	CPU	2011.3		
Halimi [26]		157.7		
Lindell [22]	GPU	427.6	24 hours	1,728,996
Peng [23]		74.6	35 hours	568,298
BU3D	CPU	$317 + 4.7 = 322.1$	9 hours	53,136
	GPU	$5.07 + 0.07 = 5.1$		

retraining with the modified IRFs). The performance of our method is shown to be less affected by the different IRFs in both symmetric and asymmetric cases. This result highlights the robustness of our method to the mismodelling of the system IRF.

Efficiency of the network: Table V compares the number of parameters of the compared deep learning methods and their running times on the Art scene. The proposed method shows the fastest running time when using a GPU device. The previous deep learning models [22], [23] could not take as input the full Lidar data due to the GPU memory limit, so they process small size patches and stitch the resulting depths together to obtain the final estimate. This is why they have a large running time with high dimensional data. Meanwhile, in the proposed method, most of the computational costs come from the estimation of initial multiscale depths, which takes 317 seconds on a CPU device and 5.07 seconds on a GPU device. The parameters of our method are an order of magnitude less than those of [22], [23],

and hence the proposed method requires shorter training time, which highlights the efficiency of our network.

It is worth mentioning that during training, Lindell and Peng use 13,800 patches of SPAD measurements with the size $32 \times 32 \times 1024$ simulated from NYU v2 dataset [52], while the proposed method uses 7,860 patches of SPAD measurements with the size $256 \times 256 \times 1024$ simulated from [46], [47] where each patch is compressed into multiscale depths of size $256 \times 256 \times 12$ to serve as an input to our network.

C. Results on Real Data

We evaluate the proposed method on a real dataset provided in [22] which captures real scenes under challenging scenarios. The Lidar data cubes have the resolution of $256 \times 256 \times 1536$ and the first column of Fig. 13 shows the reference intensity images of 4 indoor scenes (1st to 4th row) and 1 outdoor scene (the last row). In the figure, we report the PPP and SBR levels which are approximately estimated. Due to the high noise on the real data, Manipop does not yield meaningful surfaces, so we omit its results. As shown in the checkerboard scene (1st row), Peng [23] and our method yield flat depth maps on the checkerboard, while other methods observe inaccurate depth maps affected by the textures of the checkerboard. Compared to Peng's result, the proposed method gives a more flat depth map within the checkerboard and has less artifacts around the top and left borders. In the elephant scene (2nd row), compared to other methods, the proposed method reconstructs better the boundary of the elephant. In the lamp scene (3rd row), Halimi [26] reconstructs the structures well, but it suffers from outliers, while deep learning methods overall obtain less noisy results. Both Lindell [22] and Peng [23] lose some details on

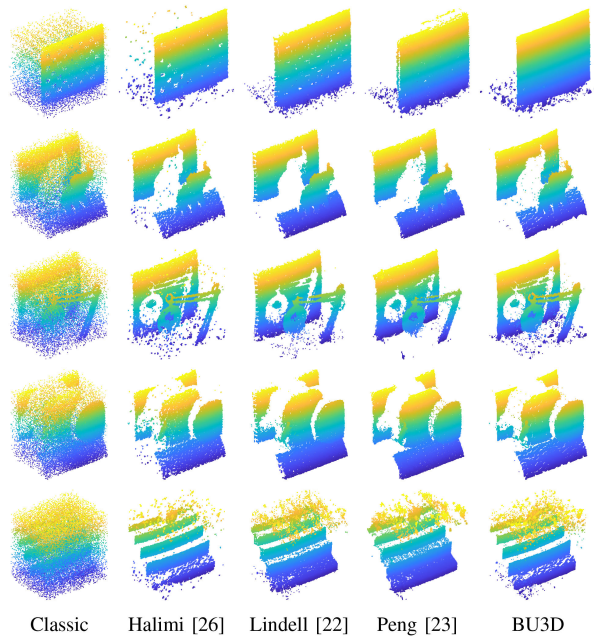


Fig. 14. Point cloud representation of reconstruction results on the real dataset.

TABLE VI

 COMPARISON OF RUNNING TIME (IN SECONDS) AVERAGED ON FIVE REAL SCENES WITH THE RESOLUTION OF $256 \times 256 \times 1536$

Classic	Halimi [26]	Lindell [22]	Peng [23]	BU3D
4.1 (CPU)	24.3 (CPU)	222.1 (GPU)	44.5 (GPU)	1.2 (GPU)

the top of the lamp whereas the proposed method obtains better reconstruction. In the 4th row, Peng’s method fails to capture the hand in the middle left region, while our method still captures it. Fig. 14 shows the reconstruction results represented by point clouds. The last row shows the reconstruction on the stair scene which has a very low SBR level due to strong sunlights. The previous deep learning methods often result in bleeding artifacts between the steps on the stair, while Halimi [26] and our method show fewer such artifacts. Finally, Table VI compares the running time, where the fastest results are obtained by the proposed method confirming its efficiency.

VI. CONCLUSION AND DISCUSSION

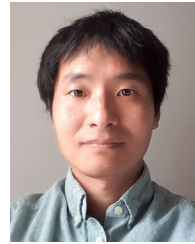
In this paper, we have proposed a new deep learning model to reconstruct depth profiles from single-photon Lidar data, taking advantages of statistical models and data-driven approaches. We design our neural network by unrolling a previous iterative Bayesian method [26], exploiting the domain knowledge on a single-photon Lidar system. This unrolling strategy improves the interpretability and efficiency of the proposed network in terms of the network size, and the training and testing times. The resulting network is also more robust to mismodeling effects due to differences between training and testing data, than classical architectures. The numerical experiments show that the proposed model can reconstruct high quality depth maps in challenging scenarios with less artifacts around the surface boundaries. The

proposed model can be extended by accounting for the multi-scale reflectivity map as an additional input or exploiting guiding information from other systems (e.g. for sensor fusion). These interesting extensions will be considered in future work.

REFERENCES

- [1] A. M. Wallace, A. Halimi, and G. S. Buller, “Full waveform LiDAR for adverse weather conditions,” *IEEE Trans. Veh. Technol.*, vol. 69, no. 7, pp. 7064–7077, Jul. 2020.
- [2] J. Rapp, J. Tachella, Y. Altmann, S. McLaughlin, and V. K. Goyal, “Advances in single-photon LiDAR for autonomous vehicles: Working principles, challenges, and recent advances,” *IEEE Signal Process. Mag.*, vol. 37, no. 4, pp. 62–71, Jul. 2020.
- [3] G. Buller and A. Wallace, “Ranging and three-dimensional imaging using time-correlated single-photon counting and point-by-point acquisition,” *IEEE J. Sel. Top. Quantum Electron.*, vol. 13, no. 4, pp. 1006–1015, Jul./Aug. 2007.
- [4] L. Zhao, L. Wang, L. Yang, A. M. Zoubir, and G. Bi, “The race to improve radar imagery: An overview of recent progress in statistical sparsity-based techniques,” *IEEE Signal Process. Mag.*, vol. 33, no. 6, pp. 85–102, Nov. 2016.
- [5] A. Kirmani *et al.*, “First-photon imaging,” *Science*, vol. 343, no. 6166, pp. 58–61, 2014.
- [6] A. Halimi, R. Tobin, A. McCarthy, J. Bioucas-Dias, S. McLaughlin, and G. Buller, “Restoration of multidimensional sparse single-photon 3D-LiDAR images,” *IEEE Trans. Comput. Imag.*, vol. 6, pp. 138–152, 2020.
- [7] S. Chen *et al.*, “Learning non-local spatial correlations to restore sparse 3D single-photon data,” *IEEE Trans. Image Process.*, vol. 29, pp. 3119–3131, 2020.
- [8] J. Tachella *et al.*, “Bayesian 3D reconstruction of complex scenes from single-photon LiDAR data,” *SIAM J. Imag. Sci.*, vol. 12, no. 1, pp. 521–550, 2019.
- [9] D. Shin, A. Kirmani, V. K. Goyal, and J. H. Shapiro, “Photon-efficient computational 3-D and reflectivity imaging with single-photon detectors,” *IEEE Trans. Comput. Imag.*, vol. 1, no. 2, pp. 112–125, Jun. 2015.
- [10] A. Halimi *et al.*, “Restoration of intensity and depth images constructed using sparse single-photon data,” in *Proc. Eur. Signal Process. Conf.*, 2016, pp. 86–90.
- [11] A. M. Pawlikowska, A. Halimi, R. A. Lamb, and G. S. Buller, “Single-photon three-dimensional imaging at up to 10 kilometers range,” *Opt. Exp.*, vol. 25, no. 10, pp. 11919–11931, May 2017.
- [12] J. Rapp and V. K. Goyal, “A few photons among many: Unmixing signal and noise for photon-efficient active imaging,” *IEEE Trans. Comput. Imag.*, vol. 3, no. 3, pp. 445–459, Sep. 2017.
- [13] A. Halimi, R. Tobin, A. McCarthy, J. Bioucas-Dias, S. McLaughlin, and G. S. Buller, “Robust restoration of sparse multidimensional single-photon LiDAR images,” *IEEE Trans. Comput. Imag.*, vol. 6, pp. 138–152, 2020, doi: [10.1109/TCL.2019.2929918](https://doi.org/10.1109/TCL.2019.2929918).
- [14] R. Tobin, A. Halimi, A. McCarthy, P. Soan, and G. Buller, “Robust real-time 3D imaging of moving scenes through atmospheric obscursants using single-photon LiDAR,” *Sci. Rep.*, vol. 11, no. 1, pp. 1–13, 2021.
- [15] S. Hernandez-Marin, A. M. Wallace, and G. J. Gibson, “Multilayered 3D LiDAR image construction using spatial models in a bayesian framework,” *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 30, no. 6, pp. 1028–1040, Jun. 2008.
- [16] A. Halimi, A. Maccarone, A. McCarthy, S. McLaughlin, and G. S. Buller, “Object depth profile and reflectivity restoration from sparse single-photon data acquired in underwater environments,” *IEEE Trans. Comput. Imag.*, vol. 3, no. 3, pp. 472–484, Sep. 2017.
- [17] M. A. A. Belmekki, R. Tobin, G. S. Buller, S. McLaughlin, and A. Halimi, “Fast task-based adaptive sampling for 3D single-photon multispectral LiDAR data,” *IEEE Trans. Comput. Imag.*, vol. 8, pp. 174–187, Feb. 2022.
- [18] Y. Altmann and S. McLaughlin, “Range estimation from single-photon LiDAR data using a stochastic em approach,” in *Proc. Eur. Signal Process. Conf.*, 2018, pp. 1112–1116.
- [19] Q. Legros, S. Meignen, S. McLaughlin, and Y. Altmann, “Expectation-maximization based approach to 3D reconstruction from single-waveform multispectral LiDAR data,” *IEEE Trans. Comput. Imag.*, vol. 6, pp. 1033–1043, 2020, doi: [10.1109/TCL.2020.2997305](https://doi.org/10.1109/TCL.2020.2997305).
- [20] S. V. Venkatakrishnan, C. A. Bouman, and B. Wohlberg, “Plug-and-play priors for model based reconstruction,” in *Proc. IEEE Glob. Conf. Signal Inf. Process.*, 2013, pp. 945–948.

- [21] J. Tachella *et al.*, "Real-time 3D reconstruction from single-photon LiDAR data using plug-and-play point cloud denoisers," *Nat. Commun.*, vol. 10, no. 1, pp. 1–6, 2019.
- [22] D. B. Lindell, M. O'Toole, and G. Wetzstein, "Single-photon 3D imaging with deep sensor fusion," *ACM Trans. Graph.*, vol. 37, no. 4, pp. 1–12, 2018.
- [23] J. Peng, Z. Xiong, X. Huang, Z.-P. Li, D. Liu, and F. Xu, "Photon-efficient 3D imaging with a non-local neural network," in *Proc. Eur. Conf. Comput. Vis.*, 2020, pp. 225–241.
- [24] Z. Sun, D. B. Lindell, O. Solgaard, and G. Wetzstein, "SPADNet: Deep RGB-SPAD sensor fusion assisted by monocular depth estimation," *Opt. Exp.*, vol. 28, no. 10, 2020, Art. no. 14948.
- [25] A. Ruget, S. McLaughlin, R. K. Henderson, I. Gyongy, A. Halimi, and J. Leach, "Robust super-resolution depth imaging via a multi-feature fusion deep network," *Opt. Exp.*, vol. 29, no. 8, 2021, Art. no. 11917.
- [26] A. Halimi, A. Maccarone, R. Lamb, G. S. Buller, and S. McLaughlin, "Robust and guided bayesian reconstruction of single-photon 3D LiDAR data: Application to multispectral and underwater imaging," *IEEE Trans. Comput. Imag.*, vol. 7, pp. 961–974, Sep. 2021.
- [27] K. Gregor and Y. LeCun, "Learning fast approximations of sparse coding," in *Proc. Int. Conf. Mach. Learn.*, 2010, pp. 399–406.
- [28] V. Monga, Y. Li, and Y. C. Eldar, "Algorithm unrolling: Interpretable, efficient deep learning for signal and image processing," *IEEE Signal Process. Mag.*, vol. 38, no. 2, pp. 18–44, Mar. 2021.
- [29] A. Beck and M. Teboulle, "A fast iterative shrinkage-thresholding algorithm for linear inverse problems," *SIAM J. Imag. Sci.*, vol. 2, no. 1, pp. 183–202, 2009.
- [30] Y. Yang, J. Sun, H. Li, and Z. Xu, "Deep admm-net for compressive sensing MRI," in *Proc. Adv. Neural Inf. Process. Syst.*, 2016.
- [31] K. Zhang, L. V. Gool, and R. Timofte, "Deep unfolding network for image super-resolution," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2020, pp. 3217–3226.
- [32] W. Pu, C. Zhou, Y. C. Eldar, and M. R. Rodrigues, "Robust learned shrinkage-thresholding (REST): Robust unrolling for sparse recover," 2021, *arXiv:2110.10391*.
- [33] A. Vaswani *et al.*, "Attention is all you need," in *Proc. Neural Inf. Process. Syst.*, 2017.
- [34] J. Hu, L. Shen, S. Albanie, G. Sun, and E. Wu, "Squeeze-and-excitation networks," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 42, no. 8, pp. 2011–2023, Aug. 2020.
- [35] X. Wang, R. Girshick, A. Gupta, and K. He, "Non-local neural networks," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2018, pp. 7794–7803.
- [36] S. Woo, J. Park, J.-Y. Lee, and I. S. Kweon, "CBAM: Convolutional block attention module," in *Proc. Eur. Conf. Comput. Vis.*, 2018, pp. 3–19.
- [37] H. Zhao, X. Kong, J. He, Y. Qiao, and C. Dong, "Efficient image super-resolution using pixel attention," in *Proc. Eur. Conf. Comput. Vis. Workshop*, 2020, pp. 56–72.
- [38] K. Xu *et al.*, "Attend and tell: Neural image caption generation with visual attention," in *Proc. Int. Conf. Mach. Learn.*, 2015, pp. 2048–2057.
- [39] Y. Gal and Z. Ghahramani, "Dropout as a bayesian approximation: Representing model uncertainty in deep learning," in *Proc. Int. Conf. Mach. Learn.*, 2016, pp. 1050–1059.
- [40] B. Lakshminarayanan, A. Pritzel, and C. Blundell, "Simple and scalable predictive uncertainty estimation using deep ensembles," in *Proc. Neural Inf. Process. Syst.*, 2016.
- [41] N. Parikh and S. Boyd, "Proximal algorithms," *Found. Trends Optim.*, vol. 1, no. 3, pp. 127–239, 2014.
- [42] C. J. Maddison, A. Mnih, and Y. W. Teh, "The concrete distribution: A continuous relaxation of discrete random variables," in *Proc. Int. Conf. Learn. Representations*, 2017.
- [43] E. Jang, S. Gu, and B. Poole, "Categorical reparameterization with gumbel-softmax," in *Proc. Int. Conf. Learn. Representations*, 2017.
- [44] B. Xu, N. Wang, T. Chen, and M. Li, "Empirical evaluation of rectified activations in convolutional network," 2015, *arXiv:150500853*.
- [45] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," in *Proc. Int. Conf. Learn. Representations*, 2015.
- [46] H. Hirschmuller and D. Scharstein, "Evaluation of cost functions for stereo matching," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2007, pp. 1–8.
- [47] D. J. Butler, J. Wulff, G. B. Stanley, and M. J. Black, "A naturalistic open source movie for optical flow evaluation," in *Proc. Eur. Conf. Comput. Vis.*, 2012, pp. 611–625.
- [48] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," in *Proc. Int. Conf. Learn. Representations*, 2015.
- [49] C. Chen, X. Chen, and H. Cheng, "On the over-smoothing problem of CNN based disparity estimation," in *Proc. Int. Conf. Comput. Vis.*, 2019, pp. 8997–9005.
- [50] J. Canny, "A computational approach to edge detection," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 8, no. 6, pp. 679–698, Nov. 1986.
- [51] F. Tosi, Y. Liao, C. Schmitt, and A. Geiger, "SMD-Nets: Stereo mixture density networks," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2021, pp. 8942–8952.
- [52] N. Silberman, D. Hoiem, P. Kohli, and R. Fergus, "Indoor segmentation and support inference from RGBD images," in *Proc. Eur. Conf. Comput. Vis.*, 2012, pp. 746–760.



graph neural networks.

Jakeoung Koo received the B.Eng. and M.Eng. degrees from Chung-Ang University, Seoul, South Korea, in 2015 and 2017, respectively, and the Ph.D. degree from the Technical University of Denmark, Denmark, in 2021. He was a Postdoctoral Research Associate with the School of Engineering and Physical Sciences, Heriot-Watt University, Edinburgh, U.K. He is currently a Postdoctoral Research Fellow with the School of Computing, National University of Singapore, Singapore. His research interests include computational imaging, geometry processing, and



image processing, with applications to single-photon depth imaging, remote sensing and medical imaging. He was the recipient of a five years RAEng Research Fellowship, an elected Member of the EURASIP Technical Area Committees (TMTSP since 2019 and SPMuS since 2022) and an Associate Editor for *Digital Signal Processing-Elsevier* since 2022.



Stephen McLaughlin (Fellow, IEEE) received the B.Sc. degree from the University of Glasgow, Glasgow, U.K., in 1981 and the Ph.D. degree from the University of Edinburgh, Edinburgh, U.K., in 1990. From 1981 to 1986, he was a Development Engineer with Industry. In 1986, he joined the Department of Electronics and Electrical Engineering with the University of Edinburgh and ultimately held a Chair in Electronic Communication Systems. In October 2011, he joined Heriot-Watt University, Edinburgh, U.K., as a Professor of signal processing. Prof. McLaughlin is a Fellow of the Royal Academy of Engineering, the Royal Society of Edinburgh, the Institute of Engineering and Technology, a EURASIP.