Graph Signal Restoration Using Nested Deep Algorithm Unrolling

Masatoshi Nagahama, *Student Member, IEEE,* Koki Yamada, *Student Member, IEEE,* Yuichi Tanaka, *Senior Member, IEEE,* Stanley H. Chan, *Senior Member, IEEE,* and Yonina C. Eldar, *Fellow, IEEE*

***Abstract*—Graph signal processing is a ubiquitous task in many applications such as sensor, social, transportation and brain networks, point cloud processing, and graph neural networks. Often, graph signals are corrupted in the sensing process, thus requiring restoration. In this paper, we propose two graph signal restoration methods based on deep algorithm unrolling (DAU). First, we present a graph signal denoiser by unrolling iterations of the alternating direction method of multiplier (ADMM). We then suggest a general restoration method for linear degradation by unrolling iterations of Plug-and-Play ADMM (PnP-ADMM). In the second approach, the unrolled ADMM-based denoiser is incorporated as a submodule, leading to a nested DAU structure. The parameters in the proposed denoising/restoration methods are trainable in an end-to-end manner. Our approach is interpretable and keeps the number of parameters small since we only tune graph-independent regularization parameters. We overcome two main challenges in existing graph signal restoration methods: 1) limited performance of convex optimization algorithms due to fixed parameters which are often determined manually. 2) large number of parameters of graph neural networks that result in difficulty of training. Several experiments for graph signal denoising and interpolation are performed on synthetic and real-world data. The proposed methods show performance improvements over several existing techniques in terms of root mean squared error in both tasks.**

***Index Terms*—Graph signal processing, signal restoration, deep algorithm unrolling, Plug-and-Play ADMM**

I. Introduction

Signal restoration is a ubiquitous task in many applications. Depending on the types of signals, the interconnectivity among samples can often be exploited, for example, signals residing on sensor networks, social networks, transportation networks, and brain networks, power grids, 3D meshes, and point clouds, all have various connectivities which can often be represented as graphs.

Preliminary results of this work was presented in [1].

M. Nagahama and Y. Tanaka are with the Department of Electrical Engineering and Computer Science, Tokyo University of Agriculture and Technology, Koganei, Tokyo 184–8588, Japan. Y. Tanaka is also with PRESTO, Japan Science and Technology Agency, Kawaguchi, Saitama 332–0012, Japan (email: nagahama@msp-lab.org; ytnk@cc.tuat.ac.jp).

K. Yamada was with the Department of Electrical Engineering and Computer Science, Tokyo University of Agriculture and Technology, Koganei, Tokyo 184-8588, Japan. He is now with the Department of Electrical Engineering, Tokyo University of Science, Katsushika, Tokyo 125–8585, Japan (email: k-yamada@rs.tus.ac.jp).

S. H. Chan is with School of Electrical and Computer Engineering, Purdue University, West West Lafayette, IN 47907, USA (email: stanchan@purdue.edu).

Y. C. Eldar is with Faculty of Mathematics and Computer Science, The Weizmann Institute of Science, Rehovot 7610001, Israel (email: yonina.eldar@weizmann.ac.il).

Y. Tanaka was partially funded by JST PRESTO under Grant JPMJPR1935 and JSPS KAKENHI under Grant 20H02145.

A *graph signal* is defined as a signal whose domain is the nodes of the graph. The relations between the samples, i.e., nodes, are given by the edges. In contrast to standard signals on a regular grid such as audio and image signals, graph signal processing (GSP) explicitly exploits the underlying structure of the signal [2]–[4]. GSP has been used in a wide range of applications for irregularly-structured data such as compression [5], sampling and restoration [6]–[10], and analysis of graph signals [11], [12].

Graph signal restoration is an important task aiming to address the problems of noise and missing values. For example in sensor networks, some sensors may not work properly resulting in missing values, and samples on the nodes are often noisy [13]. Many approaches for graph signal restoration have been proposed based on regularized optimization [14], graph filters and filter banks [15]–[17], and deep learning on graphs [18]. These existing works can be classified into two main approaches: 1) model-based restoration and 2) neural network-based restoration.

Model-based restoration: Model-based approaches often rely on convex optimization whose objective function contains a data fidelity term and a regularization term [14]. Signal priors are often required in such tasks because the problem is ill-posed. For example, a smoothness prior like graph total variation (GTV) has demonstrated effectiveness in graph signal denoising, whereas graph spectral filters have been shown to satisfy certain quadratic optimization solutions [19]–[22]. A limitation of model-based restoration methods is that they are often iterative as illustrated in Fig. 1 (left). Performance and speed of the algorithm depend on the hyper-parameters θ (e.g., step size and regularization strength) whose values are determined manually and are fixed throughout the iterations.

Neural network-based restoration: Graph convolutional networks (GCNs) are considered as a counterpart of the convolutional neural networks for image processing [23]. GCNs can automatically learn network parameters to minimize a loss function. However, GCNs have two drawbacks: 1) lack of interpretability and 2) the requirement of a large dataset for training. Furthermore, as reported in [23]–[25], deeper networks cannot always achieve good performance in the graph settings, in contrast to the remarkable success of convolutional networks for signals on a regular grid [26]. Therefore, many GCNs are limited to a small number of layers [27], [28].

As a hybrid approach of the model- and neural network-based restoration methods, we utilize *deep algorithm unrolling* (DAU) by integrating learnable parameters into the iterative algorithm [25], [29]–[32]. As illustrated in Fig. 1 (right), DAU unrolls the iterations of the iterative algorithm and

This article has been accepted for publication in IEEE Transactions on Signal Processing. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TSP.2022.3180546

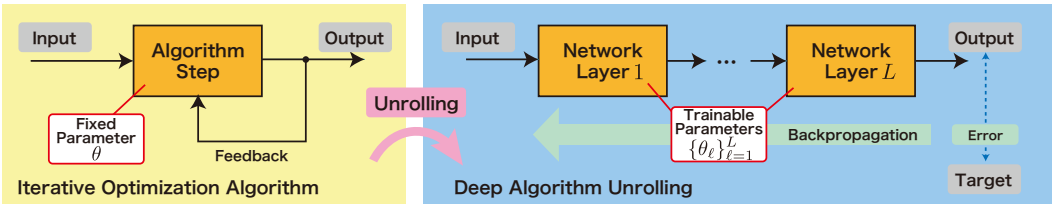


Fig. 1: Conventional iterative optimization algorithm (left) and deep algorithm unrolling (right).

deploys the trainable parameters at each unrolled iteration [32], [33]. Instead of manually choosing the parameters as in the conventional iterative approach, parameters in each unrolled iteration are determined from the training data so as to minimize a loss function. The practical advantages of DAU against the classical iterative solver are faster convergence and performance improvement since the parameters are learned to fit the target signals. Advantages compared with fully parameterized neural networks are the interpretability and a small number of parameters. Hence, the networks can be trained with a small number of training data.

An extension of DAU for graph signal denoising was recently developed in [25], which proposed unrolled GCNs based on two optimization problems of sparse coding and trend filtering. Although the formulation itself allows the network to be arbitrarily deep, the number of layers is set to be very small (typically one middle layer) in its practical implementation. This is because deeper networks do not result in better performance in this case. Additionally, GCNs often assume a fixed graph both in the training and testing phases. However, the underlying graphs are often slightly perturbed in practice. Hence, restoration algorithms should be robust to (small) perturbations of graphs. A detailed comparison between [25] and our approach is further discussed in Section III-D.

In this work, we first propose a simple yet efficient graph signal *denoising* method that utilizes DAU of the alternating direction method of multiplier (ADMM) to solve a minimization problem with two regularizers based on graph total variation and elastic net. In contrast to [25], we only train the graph-independent regularization parameters in the model-based iterative algorithms. The resulting denoising algorithm contains a significantly smaller number of parameters than neural network-based methods while showing better denoising results.

Next, we propose a nested version of DAU based on unrolling the iterations of Plug-and-Play ADMM (PnP-ADMM) [34]–[38]. This version is designed for general graph signal restoration problems with linear degradation. In this approach, the ADMM-based denoiser is plugged into the unrolled PnP-ADMM algorithm leading to a nested DAU structure. All of the parameters in the algorithm are trained in an end-to-end fashion [30], [32], [33].

In contrast to GCN-based methods, parameters to be tuned in the proposed techniques are graph-independent leading to the following advantages:

1) Interpretability: All internal modules are designed based on (convex) optimization algorithms.
2) Ease to train: Our techniques do not require large training data due to the small number of parameters.
3) Transferability: Since our methods only tune graph-independent parameters, we can immediately use the same parameter set for graphs with different sizes.

We also avoid large matrix inversion by using popular acceleration techniques in GSP: 1) precomputing graph Fourier bases and 2) polynomial approximation. Through comprehensive experiments on denoising and interpolation for synthetic and real-world data, our proposed methods are shown to achieve better performance than existing restoration methods including graph low-pass filters, model-based iterative optimization, and [25], in terms of root mean squared error (RMSE).

The remainder of this paper is organized as follows. Signal restoration algorithms using ADMM and PnP-ADMM are introduced in Section II along with notation used throughout the paper. The proposed two restoration methods are introduced in Section III. Experimental results comparing denoising and interpolation performances with existing methods are shown in Sections IV and V. Section VI concludes this paper.

II. Signal Restoration with ADMM

In this section, we first present notations and the problem formulation. Then, we review ADMM and PnP-ADMM which are the fundamental building blocks of our algorithms.

A. Notation

Throughout the paper, vectors and matrices are written in bold style and sets are written as calligraphic letters. An undirected graph $\mathcal{G} = (\mathcal{V}, \mathcal{E}, \mathbf{W})$ consists of a collection of undirected vertices $\mathcal{V} = \{v_i\}_{i=1}^N$ and edges $\mathcal{E} = \{(e_{i,j}, w_{i,j})\}$. The number of vertices and edges is $|\mathcal{V}| = N$ and $|\mathcal{E}|$, respectively; $w_{i,j} \in \mathbb{R}_{\geq 0}$ denotes the edge weight between v_i and v_j. We define a weighted adjacency matrix of $\mathcal{G}$ as an $N \times N$ matrix with $[\mathbf{W}]_{ij} = w_{i,j}$; $[\mathbf{W}]_{ij} = 0$ represents unconnected vertices. In this paper, we consider a graph that does not have self-loops, i.e., $[\mathbf{W}]_{ii} = 0$ for all i. The degree matrix of $\mathcal{G}$ is defined as a diagonal matrix $[\mathbf{D}]_{ii} = \sum_j w_{i,j}$. The combinatorial graph Laplacian matrix of $\mathcal{G}$ is given by $\mathbf{L} = \mathbf{D} - \mathbf{W}$. Since $\mathbf{L}$ is a real symmetric matrix, $\mathbf{L}$ always has an eigendecomposition. Let the eigendecomposition of the graph Laplacian matrix be $\mathbf{L} = \mathbf{U}\mathbf{\Lambda}\mathbf{U}^\top$, where $\mathbf{U}$ is an eigenvector matrix and $\mathbf{\Lambda} = \mathrm{diag}(\lambda_1, \ldots, \lambda_N)$. The weighted graph incidence matrix is denoted as $\mathbf{M} \in \mathbb{R}^{|\mathcal{E}| \times N}$. We index

integers to the set of edges as $\mathcal{E} = \{e_s\}_{s=1}^{|\mathcal{E}|}$. Then, the sth row and tth column of $\mathbf{M}$ corresponding to e_s and v_t is

$$[\mathbf{M}]_{s,t} = \begin{cases} \sqrt{w_{i,j}} & e_s = (v_i, v_j) \text{ and } t = i, \\ -\sqrt{w_{i,j}} & e_s = (v_i, v_j) \text{ and } t = j, \\ 0 & \text{otherwise.} \end{cases}$$

A graph signal $x : \mathcal{V} \to \mathbb{R}$ is a function that assigns a value to each vertex. It can be written as a vector $\mathbf{x} \in \mathbb{R}^N$ in which the ith element $x[i]$ represents the signal value at the ith vertex.

B. General Restoration Problem

Consider an observed graph signal $\mathbf{y} \in \mathbb{R}^N$ which is related to an input graph signal $\mathbf{x} \in \mathbb{R}^N$ as

$$\mathbf{y} = \mathbf{H}\mathbf{x} + \mathbf{n}, \tag{1}$$

where $\mathbf{H} \in \mathbb{R}^{N \times N}$ is a degradation matrix and $\mathbf{n} \sim \mathcal{N}(0, \sigma^2 \mathbf{I})$ is an i.i.d. additive white Gaussian noise (AWGN).

Throughout the paper, we assume that $\mathbf{x}$ is a graph signal, i.e., its domain is given by $\mathcal{G}$. This graph structure will be exploited to provide a prior for the recovery problem. The degradation model (1) generally appears in restoration problems such as denoising, interpolation, deblurring, and super-resolution, to name a few. The main objective of many restoration problems is estimating an unknown $\mathbf{x}$ from a given degraded signal $\mathbf{y}$. We assume that $\mathbf{H}$ is known a priori. In this paper, we perform two representative experiments with the following $\mathbf{H}$: 1) $\mathbf{H} = \mathbf{I}$ (denoising), and 2) a binary $\mathbf{H}$ matrix (interpolation).

C. Plug-and-Play ADMM

1) ADMM: Many inverse problems are posed as the following unconstrained minimization problem:

$$\min_{\mathbf{x} \in \mathbb{R}^N} \frac{1}{2}\|\mathbf{y} - \mathbf{H}\mathbf{x}\|_2^2 + \lambda g(\mathbf{A}\mathbf{x}), \tag{2}$$

where g is some regularization function, $\lambda \in \mathbb{R}_{\geq 0}$ is the regularization parameter, and $\mathbf{A} \in \mathbb{R}^{M \times N}$ is an arbitrary matrix. A widely used algorithm to solve (2) is the alternating direction of multipliers (ADMM) which has been used to solve generic unconstrained optimization problems with non-differentiable convex functions (see [39] for details). Through variable splitting, the general problem (2) is rewritten as the following constrained minimization problem:

$$\begin{aligned}(\tilde{\mathbf{x}}, \tilde{\mathbf{s}}) = \underset{\mathbf{x} \in \mathbb{R}^N, \mathbf{s} \in \mathbb{R}^M}{\operatorname{argmin}} \ & \frac{1}{2}\|\mathbf{y} - \mathbf{H}\mathbf{x}\|_2^2 + \lambda g(\mathbf{s}), \\ & \text{subject to } \mathbf{s} = \mathbf{A}\mathbf{x}. \end{aligned} \tag{3}$$

Applying ADMM to (3) leads to the following sequence of subproblems:

$$\mathbf{x}^{(p+1)} = \left(\mathbf{H}^\top \mathbf{H} + \rho \mathbf{A}^\top \mathbf{A}\right)^{-1} \left(\mathbf{H}^\top \mathbf{y} + \rho \mathbf{A}^\top \bar{\mathbf{x}}^{(p)}\right), \tag{4a}$$

$$\mathbf{s}^{(p+1)} = \underset{\mathbf{s} \in \mathbb{R}^M}{\operatorname{argmin}} \lambda g(\mathbf{s}) + \frac{\rho}{2}\|\mathbf{s} - \bar{\mathbf{s}}^{(p)}\|_2^2, \tag{4b}$$

$$\mathbf{t}^{(p+1)} = \mathbf{t}^{(p)} + \mathbf{A}\mathbf{x}^{(p+1)} - \mathbf{s}^{(p+1)}, \tag{4c}$$

where $\mathbf{t}^{(p)} \in \mathbb{R}^M$ is the Lagrangian multiplier, $\mathbf{s}^{(p)} \in \mathbb{R}^M$ is an auxiliary variable, g is the regularization function in (2), $\bar{\mathbf{x}}^{(p)} \stackrel{\text{def}}{=} \mathbf{s}^{(p)} - \mathbf{t}^{(p)}$ and $\bar{\mathbf{s}}^{(p)} \stackrel{\text{def}}{=} \mathbf{A}\mathbf{x}^{(p+1)} + \mathbf{t}^{(p)}$.

2) Plug-and-Play ADMM: PnP-ADMM is a variation of the classical ADMM [34] for the problem of (3) with $\mathbf{A} = \mathbf{I}$. Oftentimes, (4a) and (4b) are called the *inverse step* and *denoising step* (i.e., denoiser), respectively [35]. A notable feature is that any off-the-shelf denoiser, including deep neural networks, can be used instead of naively solving (4b) without explicitly specifying regularization terms g before implementation. Such examples are found in [40]–[42].

Empirically, PnP-ADMM has demonstrated improved performance over the standard ADMM with explicit regularization in some image restoration tasks [36], [43], [44]. Graph signal restoration with PnP-ADMM is also studied in [21] showing improved restoration performance over the existing model-based techniques.

In this paper, we follow an approach of PnP-ADMM proposed in [37]. Suppose that two initial variables $\mathbf{s}^{(0)}, \mathbf{t}^{(0)} \in \mathbb{R}^N$ are set. The algorithm of PnP-ADMM corresponding to (4a)–(4c) (again, assuming $\mathbf{A} = \mathbf{I}$) is represented as

$$\mathbf{x}^{(p+1)} = \left(\mathbf{H}^\top \mathbf{H} + \rho \mathbf{I}\right)^{-1} \left(\mathbf{H}^\top \mathbf{y} + \rho \left(\mathbf{s}^{(p)} - \mathbf{t}^{(p)}\right)\right), \tag{5a}$$

$$\mathbf{s}^{(p+1)} = \mathcal{D}_g\left(\mathbf{x}^{(p+1)} + \mathbf{t}^{(p)}\right), \tag{5b}$$

$$\mathbf{t}^{(p+1)} = \mathbf{t}^{(p)} + \mathbf{x}^{(p+1)} - \mathbf{s}^{(p+1)}, \tag{5c}$$

where $\mathcal{D}_g$ is an off-the-shelf denoiser. Note that we still need to determine the parameter ρ and the off-the-shelf graph signal denoiser $\mathcal{D}_g$ (and its internal parameters) prior to running the algorithm.

The key idea of our proposed method is to unroll the ADMM and PnP-ADMM for graph signal processing.

III. Graph Signal Restoration Algorithms

In this section, we propose the following two graph signal restoration methods, both based on DAU.

1) GraphDAU: Graph signal denoiser by unrolling ADMM to address the problem $\mathbf{H} = \mathbf{I}$. We consider a mixture of ℓ_1 and ℓ_2 regularization terms like the elastic net [45]. GraphDAU works as a better independent denoiser than the model-based and deep-learning-based approaches.
2) NestDAU: General graph signal restoration algorithm by unrolling PnP-ADMM to handle a generic $\mathbf{H}$. We plug the GraphDAU into each layer of an unrolled PnP-ADMM as a denoiser.

Our methods are illustrated in Fig. 2.

A. GraphDAU

GraphDAU considers the case where $\mathbf{H} = \mathbf{I}$ due to signal denoising and $\mathbf{A} = \mathbf{M}$ in (2). It combines the regularization terms of graph total variation (GTV) and graph Laplacian regularization [46], leading to

$$\mathcal{D}_g(\mathbf{y}) = \underset{\mathbf{x} \in \mathbb{R}^N}{\operatorname{argmin}} \frac{1}{2}\|\mathbf{x} - \mathbf{y}\|_2^2 + \lambda_1 \|\mathbf{M}\mathbf{x}\|_1 + \frac{\lambda_2}{2}\|\mathbf{M}\mathbf{x}\|_2^2, \tag{6}$$

This article has been accepted for publication in IEEE Transactions on Signal Processing. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TSP.2022.3180546

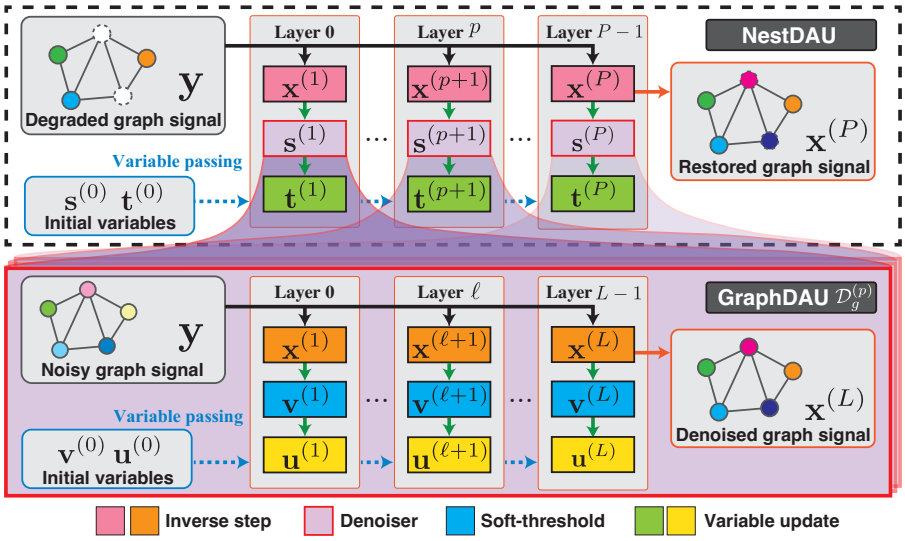


Fig. 2: Overview of GraphDAU and NestDAU for graph signal restoration. The top represents NestDAU: General restoration based on PnP-ADMM. The bottom represents GraphDAU: Denoiser based on ADMM. GraphDAU can be used as the off-the-shelf denoiser in NestDAU.

where $\|\mathbf{M}\mathbf{x}\|_1$ and $\|\mathbf{M}\mathbf{x}\|_2^2 = \mathbf{x}^\top\mathbf{L}\mathbf{x}$ (since $\mathbf{L} = \mathbf{M}^\top\mathbf{M}$) are the regularization terms for first-order and second-order differences, respectively, and λ_1 and λ_2 are nonnegative regularization parameters. The second and third terms in (6) can be written explicitly as

$$\|\mathbf{M}\mathbf{x}\|_1 = \sum_{j\in\mathcal{N}_i} \sqrt{w_{ij}}\ |x_i - x_j|, \tag{7}$$

$$\|\mathbf{M}\mathbf{x}\|_2^2 = \sum_{j\in\mathcal{N}_i} w_{ij}\ (x_i - x_j)^2, \tag{8}$$

where $\mathcal{N}_i$ is a set of vertices connecting with v_i. The norms (7) and (8) are effective regularization functions for piecewise constant and smooth graph signals [20].

In this paper, GraphDAU is only applied for denoising and not used for the general restoration problems in (2). This is because we use various acceleration techniques introduced in Section III-A3 under the assumption $\mathbf{H} = \mathbf{I}$.

We utilize ADMM as a baseline iterative solver of (6). The variable splitting is applied to (6) with $\mathbf{v} = \mathbf{M}\mathbf{x}$, leading to the following constrained minimization problem:

$$\mathcal{D}_g(\mathbf{y}) = \underset{\mathbf{x}\in\mathbb{R}^N,\mathbf{v}\in\mathbb{R}^{|\mathcal{E}|}}{\operatorname{argmin}} \frac{1}{2}\|\mathbf{x}-\mathbf{y}\|_2^2 + \lambda_1\|\mathbf{v}\|_1 + \frac{\lambda_2}{2}\|\mathbf{v}\|_2^2, \quad \text{subject to } \mathbf{v} = \mathbf{M}\mathbf{x}. \tag{9}$$

The solution of (9) can be found by solving a sequence of the following subproblems [47]:

$$\mathbf{x}^{(\ell+1)} = \left(\mathbf{I} + \frac{1}{\gamma}\mathbf{M}^\top\mathbf{M}\right)^{-1}\left(\mathbf{y} + \frac{1}{\gamma}\mathbf{M}^\top(\mathbf{v}^{(\ell)} - \mathbf{u}^{(\ell)})\right), \tag{10a}$$

$$\mathbf{v}^{(\ell+1)} = \frac{1}{1+\lambda_2\gamma} S_{\lambda_1\gamma}(\mathbf{M}\mathbf{x}^{(\ell+1)} + \mathbf{u}^{(\ell)}), \tag{10b}$$

$$\mathbf{u}^{(\ell+1)} = \mathbf{u}^{(\ell)} + \mathbf{M}\mathbf{x}^{(\ell+1)} - \mathbf{v}^{(\ell+1)}, \tag{10c}$$

where γ is the step size of the algorithm and $S_{\lambda_1\gamma}$ is the soft-thresholding operator

$$[S_{\lambda_1\gamma}(\mathbf{x})]_i = \operatorname{sgn}(x_i)\max\{|x_i| - \lambda_1\gamma, 0\}, \tag{11}$$

where $\operatorname{sgn}(\cdot)$ denotes the signum function.

Next, we unroll the iteration of (10a)–(10c) to design a trainable $\mathcal{D}_g$. In other words, instead of using fixed parameters in (10a)–(10b), we deploy trainable parameters in each iteration. The terms including $\mathbf{M}$ and $\mathbf{M}^\top\mathbf{M}$ in (10a)–(10c) are graph filters, i.e., *graph convolution*, and are fixed: We only tune three parameters, γ, λ_1, and λ_2, in each unrolled iteration. This is because we aim to construct an interpretable and easy-to-train graph signal restoration algorithm. The training configurations are described later in Section IV-A2[1]. In the following sections, we propose two forms of GraphDAU and introduce its acceleration techniques.

1) GraphDAU-TV: In this method, we only consider the ℓ_1 term of (9) by setting $\lambda_2 = 0$. Then, we choose γ and $\gamma\lambda_1$ to be learnable, i.e., $\gamma \to \{\gamma_\ell\}_{\ell=0}^{L-1}$ and $\gamma\lambda_1 \to \{\beta_\ell\}_{\ell=0}^{L-1}$. This regularization is based on the assumption that the signal is piecewise constant.

2) GraphDAU-EN: This GraphDAU is based on a combination of the ℓ_1 and ℓ_2 regularizations in (9) like the elastic net (EN), defined by the weighted incidence matrix $\mathbf{M}$. We introduce a set of trainable parameters $1/(1+\lambda_2\gamma) \to \{\alpha_\ell\}_{\ell=0}^{L-1}$ in addition to $\{\gamma_\ell\}_{\ell=0}^{L-1}$ and $\{\beta_\ell\}_{\ell=0}^{L-1}$. This method automatically controls piecewise and smoothness terms at each layer.

3) Algorithm Acceleration: The graph filter $(\mathbf{I} + \frac{1}{\gamma}\mathbf{M}^\top\mathbf{M})^{-1}$ in (10a) requires matrix inversion and its computational complexity is typically $\mathcal{O}(N^3)$ (for a dense matrix). If each layer requires calculating the inversion, the

[1]The detailed gradient computations for training the parameters are given in the Appendix.

complexity becomes $\mathcal{O}(N^3L)$. We consider accelerating GraphDAU by the following two popular techniques: 1) eigendecomposition of $\mathbf{L}$, and 2) Chebyshev polynomial approximation of a graph filter.

Precomputing Eigendecomposition: In this approach, we precompute the eigendecomposition (EVD) of $\mathbf{L}$. The inverse matrix in (10a) can be decomposed as

$$\left(\mathbf{I}+\frac{1}{\gamma_\ell}\mathbf{L}\right)^{-1}=\mathbf{U}\left(\mathbf{I}+\frac{1}{\gamma_\ell}\mathbf{\Lambda}\right)^{-1}\mathbf{U}^\top. \tag{12}$$

Since $\mathbf{I}+(1/\gamma_\ell)\mathbf{\Lambda}$ is a diagonal matrix, the inversion has $\mathcal{O}(N)$ complexity. If $\mathcal{G}$ does not change frequently throughout the iterations (which often is the case), the eigenvalues $\mathbf{\Lambda}$ and eigenvectors $\mathbf{U}$ are fixed. Therefore, the eigendecomposition of the graph Laplacian is performed only once. This GraphDAU with acceleration is represented with the suffix -E and is summarized in Algorithm 1.

Chebyshev Polynomial Approximation: This technique approximates (12) with a polynomial, for example, using the Chebyshev polynomial approximation (CPA) (see [48], [49] for details).

First, we rewrite the inverse step at the ℓth layer corresponding to (12) as

$$\mathbf{x}^{(\ell+1)}=\mathcal{H}^{(\ell)}(\mathbf{L})\widetilde{\mathbf{y}}^{(\ell)}, \tag{13}$$

where $\widetilde{\mathbf{y}}^{(\ell)}=\mathbf{y}+\frac{1}{\gamma_\ell}\mathbf{M}^\top\left(\mathbf{v}^{(\ell)}-\mathbf{u}^{(\ell)}\right)$ and $\mathcal{H}^{(\ell)}(\mathbf{L}):=\mathbf{U}\mathcal{H}^{(\ell)}(\mathbf{\Lambda})\mathbf{U}^\top$ is the filter function. This filter kernel has the following graph frequency response:

$$\mathcal{H}^{(\ell)}(\mathbf{\Lambda})=\mathrm{diag}\left(h^{(\ell)}(\lambda_1),\ldots,h^{(\ell)}(\lambda_N)\right), \tag{14}$$

where $h^{(\ell)}(x)=\gamma_\ell/(\gamma_\ell+x)$ is the filter kernel which acts as a graph low-pass filter. By performing K-truncated Chebyshev approximations to $h^{(\ell)}(x)$, the approximated version $\widetilde{\mathcal{H}}^{(\ell)}(\mathbf{L})$ is represented as:

$$\mathbf{x}^{(\ell+1)}=\widetilde{\mathcal{H}}^{(\ell)}(\mathbf{L})\widetilde{\mathbf{y}}^{(\ell)}. \tag{15}$$

GraphDAU with Chebyshev polynomial approximation is specified by a suffix -C in Algorithm 1.

B. NestDAU: Unrolled PnP-ADMM with GraphDAU as the Denoiser

Next, we develop a restoration algorithm for general $\mathbf{H}$ in (2). The baseline algorithm we consider is PnP-ADMM introduced in (4a)–(4c) because it is able to adapt to general $\mathbf{H}$. In addition, any denoiser can be used in its internal algorithm to boost performance.

Suppose that the iteration number P is given. We then unroll (5a)–(5c) of the PnP-ADMM iterations to construct P layer networks. That is, we set ρ in (5a) to be learnable, i.e., $\rho\rightarrow\{\rho_p\}_{p=0}^{P-1}$ in which p indicates the layer number. The restoration steps are equivalent to those in PnP-ADMM with P iterations, but each iteration is conducted with different regularization parameters.

The important part of the restoration algorithm is the design of the off-the-shelf denoiser $\mathcal{D}_g$ in (5b) since (5a) and (5c) are independent of the underlying graph. In this paper, we aim to keep the algorithm fully interpretable and the number of parameters small for efficient training, and thereby, we utilize GraphDAU in each layer as $\mathcal{D}_g^{(p)}$. As a result, the restoration algorithm has a nested unrolled structure as shown in Fig. 2. Based on this structure, we refer to the proposed method as *NestDAU*. Note that all the parameters in NestDAU, including those in GraphDAU, can be trained in an end-to-end fashion from a training set. The training details are presented in Section IV-A2.

Algorithm 2 shows the details of NestDAU. Note that we perform two representative signal restoration experiments (i.e., denoising and interpolation) in this paper, but NestDAU can be applicable to other cases as well, e.g., deblurring [50] and point cloud super-resolution [51].

Algorithm 1 GraphDAU for graph signal denoising $\mathcal{D}_g^{(p)}$

NOTE: Background colors correspond to those in Fig. 2.

Input: noisy graph signal $\mathbf{y}$; graph incidence matrix $\mathbf{M}$; initial variables $\mathbf{v}^{(0)},\mathbf{u}^{(0)}$; network layers L; polynomial order K (for GraphDAU-TV-C and GraphDAU-EN-C).

Output: denoised graph signal $\mathbf{x}^{(L)}$.

compute the graph Laplacian $\mathbf{L}=\mathbf{M}^\top\mathbf{M}$

If GraphDAU-TV-E or GraphDAU-EN-E **then**

  compute the eigendecomposition $\mathbf{L}=\mathbf{U}\mathbf{\Lambda}\mathbf{U}^\top$

for $\ell=0,\cdots,L-1$ **do**

$$\widetilde{\mathbf{y}}^{(\ell)}\leftarrow\mathbf{y}+\tfrac{1}{\gamma_\ell}\mathbf{M}^\top\left(\mathbf{v}^{(\ell)}-\mathbf{u}^{(\ell)}\right)$$

$$\mathbf{x}^{(\ell+1)}\leftarrow\begin{cases}\mathbf{U}\left(\mathbf{I}+\frac{1}{\gamma_\ell}\mathbf{\Lambda}\right)^{-1}\mathbf{U}^\top\widetilde{\mathbf{y}}^{(\ell)}\\ \quad\text{for GraphDAU-TV-E and}\\ \quad\text{GraphDAU-EN-E}\\ \widetilde{\mathcal{H}}^{(\ell)}(\mathbf{L})\widetilde{\mathbf{y}}^{(\ell)}\\ \quad\text{for GraphDAU-TV-C and}\\ \quad\text{GraphDAU-EN-C}\end{cases}$$

$$\mathbf{v}^{(\ell+1)}\leftarrow\begin{cases}S_{\beta_\ell}\left(\mathbf{M}\mathbf{x}^{(\ell+1)}+\mathbf{u}^{(\ell)}\right)\\ \quad\text{for GraphDAU-TV}\\ \alpha_\ell S_{\beta_\ell}\left(\mathbf{M}\mathbf{x}^{(\ell+1)}+\mathbf{u}^{(\ell)}\right)\\ \quad\text{for GraphDAU-EN}\end{cases}$$

$$\mathbf{u}^{(\ell+1)}\leftarrow\mathbf{u}^{(\ell)}+\mathbf{M}\mathbf{x}^{(\ell+1)}-\mathbf{v}^{(\ell+1)}$$

end for

return $\mathbf{x}^{(L)}$

C. Summary of Computation Issues

In Table I, we compare the proposed methods in terms of the regularization function, the acceleration technique, the number of parameters, and the computational complexity. NestDAUs are classified based on its GraphDAU specifications and have the same suffix as the corresponding GraphDAU.

The number of parameters linearly increases in proportion to the number of layers L but is independent of N. The

This article has been accepted for publication in IEEE Transactions on Signal Processing. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TSP.2022.3180546

TABLE I: Comparison among the proposed methods.

Methods	Task	Regularization	Acceleration	# of params	Computational complexity
GraphDAU-TV-E GraphDAU-TV-C	Denoising	Graph total variation	Precomputing EVD CPA	$2L$	$\mathcal{O}(N^3+N^2L)$ $\mathcal{O}(K\|\mathcal{E}\|L)$
GraphDAU-EN-E GraphDAU-EN-C		Elastic net	Precomputing EVD CPA	$3L$	$\mathcal{O}(N^3+N^2L)$ $\mathcal{O}(K\|\mathcal{E}\|L)$
NestDAU-TV-E NestDAU-TV-C	General restoration	Graph total variation	Precomputing EVD CPA	$(2L+1)P$	$\mathcal{O}(N^3+N^2LP)$ $\mathcal{O}(K\|\mathcal{E}\|LP)$
NestDAU-EN-E NestDAU-EN-C		Elastic net	Precomputing EVD CPA	$(3L+1)P$	$\mathcal{O}(N^3+N^2LP)$ $\mathcal{O}(K\|\mathcal{E}\|LP)$

Algorithm 2 NestDAU for graph signal restoration
NOTE: Background colors correspond to those in Fig. 2.

Input: degraded graph signal $\mathbf{y}$; initial variables $\mathbf{s}^{(0)}, \mathbf{t}^{(0)}$; network layers P; trainable denoiser $\mathcal{D}_g^{(p)}$.
Output: restored graph signal $\mathbf{x}^{(P)}$.

1: **for** $p = 0, \cdots, P-1$ **do**
2: $\mathbf{x}^{(p+1)} \leftarrow \left(\mathbf{H}^\top\mathbf{H} + \rho_p\mathbf{I}\right)^{-1}\left(\mathbf{H}^\top\mathbf{y} + \rho_p\left(\mathbf{s}^{(p)} - \mathbf{t}^{(p)}\right)\right)$
3: $\mathbf{s}^{(p+1)} \leftarrow \mathcal{D}_g^{(p)}\left(\mathbf{x}^{(p+1)} + \mathbf{t}^{(p)}\right)$
4: $\mathbf{t}^{(p+1)} \leftarrow \mathbf{t}^{(p)} + \mathbf{x}^{(p+1)} - \mathbf{s}^{(p+1)}$
5: **end for**
6: **return** $\mathbf{x}^{(P)}$

complexity mainly depends on the use of EVD. The methods with EVD have complexities depending on the number of nodes N, while those with CPA only rely on the number of edges $|\mathcal{E}|$ and the polynomial order K; $K|\mathcal{E}|$ is generally much smaller than N^2 when N becomes large.

As mentioned, the proposed methods require training data (i.e., a set of ground-truth and degraded data) to tune parameters. They come from the hyperparameter(s) of the original (PnP-)ADMM algorithms. Note that, even for a regular ADMM, we need to determine the optimal hyperparameter(s) for practical applications: This often needs training data.

In general, many trainable parameters in deep learning require a large dataset to avoid overfitting. This implies that GNNs require many training data. In contrast, NestDAU and GraphDAU have significantly fewer parameters than representative deep learning methods. This leads to that the proposed method can train with the small number of training data, which is beneficial for practical applications. This is experimentally verified in Sections IV and V.

D. Comparison to [25]

Two approaches for graph signal denoising based on DAU, called graph unrolling sparse coding (GUSC) and graph unrolling trend filtering (GUTF), were proposed in [25]. Since they have the same objective as that for GraphDAU, we compare the details of DAU-based graph signal restoration methods in Table II.

First, GUSC/GUTF only consider the problem of graph signal denoising. This is the same objective as that of GraphDAU, while NestDAU focuses on a generic restoration problem in (1). This is possible by employing the PnP-ADMM as a prototype of the iterative algorithm. Second, the regularization of GUSC/GUTF only contains the ℓ_1 term, while GraphDAU also includes an ℓ_2 term $\|\mathbf{M}\mathbf{x}\|_2^2 = \mathbf{x}^\top\mathbf{L}\mathbf{x}$, which is beneficial for globally smooth signals. GraphDAU-EN can automatically control the regularization weights between the ℓ_1 and ℓ_2 terms, leading to flexibility in capturing signal characteristics. Third, GUSC and GUTF train parameters in an unsupervised setting while our proposed methods train the network in a supervised way. In the following experiments, we train GUSC/GUTF in a supervised setting for a fair comparison. Extending GraphDAU and NestDAU to the unsupervised setting is left for future work.

In (10a), we keep the structure of the original graph filter $h(\mathbf{L}) = (\mathbf{I} + \frac{1}{\gamma}\mathbf{L})^{-1}$ of the ADMM algorithm and only train a graph-independent parameter γ. As such, GraphDAU performs stably with many layers (typically $L = 10$ in the experiments). In contrast, GUSC/GUTF use GCNs for its internal algorithm. Therefore, they result in few middle layers (as reported in [25], they have only one middle layer in the experiment). They reduce many learnable parameters compared to usual GCNs thanks to their edge-weight-sharing convolution, however, they still contain many parameters. A detailed comparison of the number of parameters is presented along with the restoration performance in Section IV.

IV. Experimental Results: Denoising

In the following two sections, we compare graph signal restoration performances of NestDAU and GraphDAU with existing methods using synthesized and real-world data. In both sections, parameters of the proposed and neural network-based methods are trained by setting the mean squared error (MSE) $\frac{1}{N}\|\widehat{\mathbf{x}} - \mathbf{x}^*\|_2^2$ as a loss function, where $\widehat{\mathbf{x}} \in \mathbb{R}^N$ is the restored signal and $\mathbf{x}^* \in \mathbb{R}^N$ is the ground-truth signal available during the training phase.

In this section, we consider denoising corresponding to $\mathbf{H} = \mathbf{I}$ in (1).

We conduct three experiments:

1) Denoising on fixed graphs;
2) Denoising on graphs with perturbation;
3) Transferring tuned parameters to different N.

In the following subsections, we describe the details of the denoising experiment. We also show an in-depth analysis of the proposed methods in terms of the number of layers (i.e., L or P) and the polynomial order K.

TABLE II: Comparison between related works and ours.

	GUSC/GUTF [25]	GraphDAU	NestDAU
Optimization algorithm for unrolling	Half-quadratic splitting [52]	ADMM [39]	PnP-ADMM [34]
Problem setting	unsupervised & supervised	supervised	
Considered restoration problem	denoising	denoising	general restoration

TABLE III: Training configuration.

Batch size	1
Epochs	≤ 3
Weight decay	1.0×10^4
Optimizer	Adam [58]
Learning rate	0.02
Scheduler	StepLR

A. Methods and Training Configurations

1) Alternative Methods: We compare the denoising performance with several existing methods using smoothing filters and optimization approaches:

- Graph spectral diffusion with heat kernel (HD) [53];
- Spectral graph bilateral filter (SGBF) [22], [54];
- ADMM-based smoothing with a fixed parameter; ((10a)–(10c)) with 10 iterations;
- PnP-ADMM-based smoothing with fixed parameters with 8 iterations [21]: Its formulation is given in Section III-B and off-the-shelf denoisers are HD or SGBF.

Filtering operations of the algorithms are partly implemented by `pygsp` [55]. For a fair comparison, their fixed parameters are tuned by performing a grid search on the validation data to minimize RMSE.

We also include the following deep learning-based methods for comparison:

- Multi-layer perceptron (MLP);
- Graph convolutional network (GCN) and that with residual connections (GCN-R) [23];
- Graph attention networks (GAT) [56];
- Graph unrolling-based trend filtering (GUTF) [25];
- Graph unrolling-based sparse coding (GUSC) [25].

These existing methods are set to 64 dimensions as a hidden layer of neural nets as in the setting in [25]. These methods and ours are implemented with `Pytorch` [57]. MLP, GCN, GCN-R, and GAT are trained for 30 epochs that lead to convergence of the loss function. GUTF and GUSC are trained with the same hyper-parameters as [25], but they are trained in the supervised setting in this paper.

2) Training Configuration: On the basis of preliminary experiments, hyper-parameters used for training of the proposed methods are summarized in Table III. Training scheduler StepLR in `Pytorch` is used to gradually decay the learning rate by multiplying 0.6 each epoch. Since our proposed methods have a small number of parameters, training usually converges in no more than three epochs. A detailed performance analysis is discussed in Section IV-F.

B. Datasets and Setup

Here, we describe the details of the experiments and datasets. The dataset specifications are summarized in Table IV.

1) Denoising on Fixed Graphs: The first experiment is graph signal denoising for the following fixed graphs:

- Synthetic signals on a community graph having three clusters ($N = 250$);
- Synthetic signals on a random sensor graph ($N = 150$);
- Temperature data in the United States ($N = 614$).

We assume that the graph is consistent in all of the training, validation, and testing phases.

Characteristics of Graphs and Graph Signals: The community graph is generated by `pygsp` [55] and is shown in Fig. 3a. We synthetically create piecewise constant graph signals based on the cluster labels of the community graph. Note that the cluster labels are different while the graph itself is fixed. Each cluster in the graph is assigned an integer value between 1 to 6 randomly as its cluster label. Then, AWGN ($\sigma = \{0.5, 1.0\}$) is added to the ground-truth signals.

The random sensor graph is also obtained by `pygsp` [55] and is shown in Fig. 4a. On the random sensor graph, piecewise-smooth signals are synthesized in the following manner. First, vertices on a graph are partitioned into eight non-overlapping subgraphs $\{\mathcal{G}_k\}_{k=1}^8$. Then, smooth signals on $\mathcal{G}_k$ are synthesized based on the first three eigenvectors of the graph Laplacian of $\mathcal{G}_k$. Let $\mathbf{L}_k$ and $\mathbf{U}_k$ be the graph Laplacian of $\mathcal{G}_k$ and its eigenvector matrix, respectively. Then, a smooth signal on $\mathcal{G}_k$ is given by

$$\mathbf{x}_k = \mathbf{U}_{k,3}\mathbf{d}, \tag{16}$$

where $\mathbf{U}_{k,3}$ is the first three eigenvectors in $\mathbf{U}_k$ and $\mathbf{d} \in \mathbb{R}^3$ are expansion coefficients whose element is randomly selected from $[0, 5]$. Finally, a piecewise-smooth signal on $\mathcal{G}$ is obtained by combining eight $\mathbf{x}_k$'s as follows:

$$\mathbf{x} = \sum_k \mathbf{1}_{\mathcal{C}_k}\mathbf{x}_k \tag{17}$$

where $\mathbf{1}_{\mathcal{C}_k} \in \{0, 1\}^{N \times |\mathcal{C}_k|}$ is the indicator matrix in which $[\mathbf{1}_{\mathcal{C}_k}]_{i,j} = 1$ when the node i in $\mathcal{G}$ corresponds to the node j in $\mathcal{G}_k$ and 0 otherwise. AWGN ($\sigma = \{0.5, 1.0\}$) is added to the ground-truth signals.

In order to demonstrate the effectiveness of our method for real-world data, we use daily average temperature data in the United States in 2017, provided by QCLCD[2] [59]. The data contain local temperatures recorded at weather stations, yet they include missing observations. To obtain the completed data (as the ground truth) for a year, we conduct the following preprocessing: 1) 614 stations (out of 7501 ones) having relatively few missing values are selected. 2) Missing values in these stations are filled using the average temperatures observed at the same station in the previous and subsequent days. For experiment, we split the dataset into three parts: 304

[2]https://www.ncdc.noaa.gov/orders/qclcd/

This article has been accepted for publication in IEEE Transactions on Signal Processing. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TSP.2022.3180546

TABLE IV: Summary of data used in Section IV

Experiment	Graph	Signal characteristic	Type	Data splitting (train/valid/test)
Denoising on fixed graphs	Community graph	Piecewise constant	Synthetic data	500/50/50
	Sensor graph	Piecewise smooth	Synthetic data	500/50/50
	U.S. geometry	Daily temperature (time-series)	Real data	304/30/31
Denoising on perturbed graphs	Sensor graph	Piecewise constant		500/50/50
		Piecewise smooth	Synthetic data	500/50/50
		Globally smooth		500/50/50
	3D Point clouds	RGB color attributes	Real data	129/43/44
Parameter transfer	3D Point clouds	RGB color attributes for different N	Real data	N/A

TABLE V: Denoising results on fixed graphs (average RMSEs for test data)

					Community graph (Piecewise constant)		Random sensor graph (Piecewise smooth)		U.S. temperature (Globally smooth)			
Methods	# params	L	K	P	$\sigma = 0.5$	1.0	0.5	1.0	3.0	5.0	7.0	9.0
Noisy	-	-	-	-	0.495	1.002	0.499	0.996	2.986	5.032	7.029	9.012
HD	-	-	-	-	0.230	0.325	0.405	0.598	1.712	2.149	2.475	2.751
SGBF	-	-	-	-	0.195	0.256	0.394	0.588	1.731	2.167	2.470	2.762
ADMM (GTV)	-	10	-	-	0.114	0.233	0.378	0.603	1.784	2.249	2.483	2.805
PnP-HD	-	-	-	8	0.218	0.324	0.400	0.592	1.706	2.168	2.467	2.745
PnP-SGBF	-	-	-	8	0.195	0.294	0.399	0.586	1.730	2.164	2.457	2.749
MLP	4,353	-	-	-	0.436	0.795	0.454	0.739	2.892	4.588	6.634	7.816
GCN	4,353	-	-	-	0.258	0.283	0.649	0.705	2.208	2.380	2.653	2.948
GCN-R	4,353	-	-	-	0.272	0.305	0.718	0.759	2.285	2.411	2.642	2.931
GAT	2,050	-	-	-	0.236	0.247	0.560	0.752	2.213	2.776	3.042	3.698
GUTF	19,397	-	-	-	0.100	0.158	0.419	0.523	2.127	2.293	2.551	2.891
GUSC	11,205	-	-	-	0.139	0.206	0.383	**0.512**	2.069	2.248	2.587	2.935
GraphDAU-TV-E	20	10	-	-	0.060	0.117	0.364	0.583	1.688	2.154	2.436	2.775
GraphDAU-TV-C	20	10	10	-	0.073	0.142	0.364	0.583	1.693	2.170	2.536	2.743
GraphDAU-EN-E	30	10	-	-	0.081	0.138	0.340	0.547	**1.652**	2.123	2.411	2.723
GraphDAU-EN-C	30	10	10	-	0.095	0.165	0.339	0.554	1.685	2.151	2.479	2.745
NestDAU-TV-E	168	10	-	8	**0.054**	0.106	**0.324**	0.559	1.661	2.153	2.429	2.736
NestDAU-TV-C	168	10	10	8	0.056	**0.103**	0.374	0.631	1.665	2.124	2.458	2.730
NestDAU-EN-E	248	10	-	8	0.072	0.105	**0.324**	0.528	1.656	**2.087**	**2.409**	2.713
NestDAU-EN-C	248	10	10	8	0.061	0.110	0.330	0.530	1.654	2.100	2.434	**2.674**

training (January to October), 30 validation (November), and 31 testing (December) data. In this experiment, we study four noise strengths of AWGN, i.e., $\sigma = \{3.0, 5.0, 7.0, 9.0\}$. The weighted graph is constructed by an 8-nearest neighbor (NN) graph based on the stations' geographical coordinates.

2) Denoising on Graphs with Perturbation: The second experiment is conducted for signals on graphs with perturbation to verify the robustness of the proposed method to small perturbations of the underlying graph. Indeed, the tuned parameters for one graph are not expected to work properly for a completely different graph because the topologies and graph Fourier basis on different graphs are different. However, signals on similar graphs, in terms of their edge weights, could have similar characteristics and therefore, it is expected that the learned parameters for one graph could work satisfactorily for the signal on another graph if these two graphs are similar enough.

Note that many graph neural network-based methods assume the graph is fixed, while our approach based on DAU only needs to tune graph-independent parameters. Thus, we can use different graphs in each epoch for training, validation, and testing. In this experiment, we only showcase the performance with a comparison to the model-based methods because the model-based approaches are applicable even if the graphs are different.

We used the following graph signals:

- Synthetic signals on random sensor graphs ($N = 150$) having piecewise-constant, piecewise-smooth, and globally-smooth characteristics;
- RGB color attributes on 3D point clouds ($N = 1,000$).

Characteristics of Graphs and Graph Signals: For the experiment on random sensor graphs, each graph is synthetically generated by using a different seed of `graphs.Sensor` from `pygsp` [55]. This results in that all graphs have different topologies and edge weights, but their characteristics are similar.

We then synthetically generate the following graph signals:

a) Piecewise constant signals: We first partition each graph into five clusters with non-overlapping nodes and randomly assign an integer for each cluster between 1 to 6. The cluster labels are used as a graph signal.
b) Piecewise smooth signal: Similar to the piecewise constant case, we first partition each graph into five clusters with non-overlapping nodes. The signal is generated by (16) and (17).
c) Globally smooth signal: The signal is obtained with a linear combination of the first five graph Fourier basis with random expansion coefficients $\mathbf{d} \in \mathbb{R}^5$ like (16).

In this experiment, we study two noise strengths of AWGN, i.e., $\sigma = \{0.5, 1.0\}$.

As real data on graphs with perturbation, we use the color attributes of 3D point clouds from JPEG Pleno Database[3] [60],

[3]http://plenodb.jpeg.org/pc/microsoft/

where the human motions are captured as point clouds. We randomly sample $1,000$ points from the original data. Then, weighted graphs are constructed using a 4-NN method whose weights are determined based on the Euclidean distance. Graphs in this dataset are, therefore, not fixed because the Euclidean distances between points are different due to random sampling. AWGN with $\sigma = \{20, 30, 40\}$ is added to each sample to yield a noisy signal. Note that the implementation of the proposed methods are conducted channel-wise so that the parameters are adjusted to each channel.

3) Parameter Transfer for Different N: The number of nodes of a graph directly influences computation complexities for all (training, validation, and testing) phases. To apply the proposed methods to a signal with large N, naive training results in large computational burden. Motivated by this, in the third experiment, we consider transferring the learned parameters to graph signals having different N. That is, parameters trained with signals on a small graph $\mathcal{G}'$ with N' ($\ll N$) nodes are reused for evaluation with signals on $\mathcal{G}$ with N nodes. This approach can be easily realized with the proposed method since its parameters are independent of N.

We first train the GraphDAU-TV-C (i.e., Chebyshev polynomial version of GraphDAU based on the GTV regularization) on the 3D point cloud datasets with $N' = 1,000$ points. After that, the pre-trained parameters are applied to the datasets with larger $N = \{2,000,\ 5,000,\ 10,000\}$.

C. Denoising Results: Fixed Graph

The experimental results on the fixed graphs are summarized with the number of parameters in Table V. Visualizations of the denoising results are also shown in Figs. 3, 4, and 5.

In most cases, the proposed methods show RMSE improvements compared to all of the alternative methods. It is observed that the proposed approach successfully restored graph signals having various characteristics. Note that, in spite of the performance improvements, our methods have a significantly smaller number of parameters than the neural network-based approaches.

Although GraphDAU-TV and -EN outperform existing methods, NestDAUs provide even better performance by incorporating GraphDAUs as submodules of NestDAU. These results imply that the nested structure is effective for graph signal restoration. NestDAU using EVD often outperforms that using CPA in most datasets and conditions.

D. Denoising Results: Graphs with Perturbation

Table VI summarizes the results of the second experiment, denoising on graphs with small perturbation. Overall, our algorithms outperform the alternatives as in the case for the fixed graph. The proposed techniques show RMSE improvements for all of the signal types under consideration. This implies that our methods effectively reflect the signal prior as the tuned parameters through training, leading to robustness against a slight change of graphs.

E. Transferring Tuned Parameters for Different N

The results of the third experiment, transferring the tuned parameters to different N, are summarized in Table VII. This shows that even if the number of nodes increases, the proposed method works well as long as the signal and graph properties are similar. Fig. 6 shows the visualization of noisy and denoised results.

F. Performance Study: The Number of Layers

Along with the denoising results, the effect of the number of layers is studied here. We use the dataset of the fixed community graph whose details are described in Section IV-B.

1) GraphDAU: Fig. 7a shows the performance analysis in terms of the number of layers L of GraphDAU for $L \in \{1, \ldots, 30\}$. The average RMSE in the test data is reported. As can be seen in the figure, the RMSE of GraphDAU rapidly decreases for $L \leq 10$, whereas there is a slight improvement for $L > 10$. We observed that GraphDAU-TV-E steadily decreases RMSEs while they are slightly oscillated for GraphDAU-EN-E. Fig. 7b shows the influence of the polynomial order $K \in \{2, \ldots 30\}$ of GraphDAU-TV-C and -EN-C with $L = 10$. Both methods almost monotonically decrease RMSEs as K becomes larger.

2) NestDAU: Fig. 7c shows the performance in terms of the number of layers P of NestDAU. The submodule GraphDAU contains $L = 10$ for using EVD and $L = 10$ and $K = 10$ with that using CPA. The number of layers is selected to $P \in \{1, \ldots, 10\}$. For NestDAU, all configurations are stable in terms of the layer size P. Even if the in-loop denoisers are changed, the performances are almost equivalent.

G. RMSE Analysis during Training

Fig. 8 shows the average RMSEs of the validation data during training. The data used are signals on community graphs ($\sigma = 0.5$) described in Section IV-B. As shown in the figure, the RMSEs rapidly decrease with less than 250 iterations (that is, the number of training data). Furthermore, NestDAUs converge faster than their GraphDAU counterparts.

V. Experimental Results: Interpolation

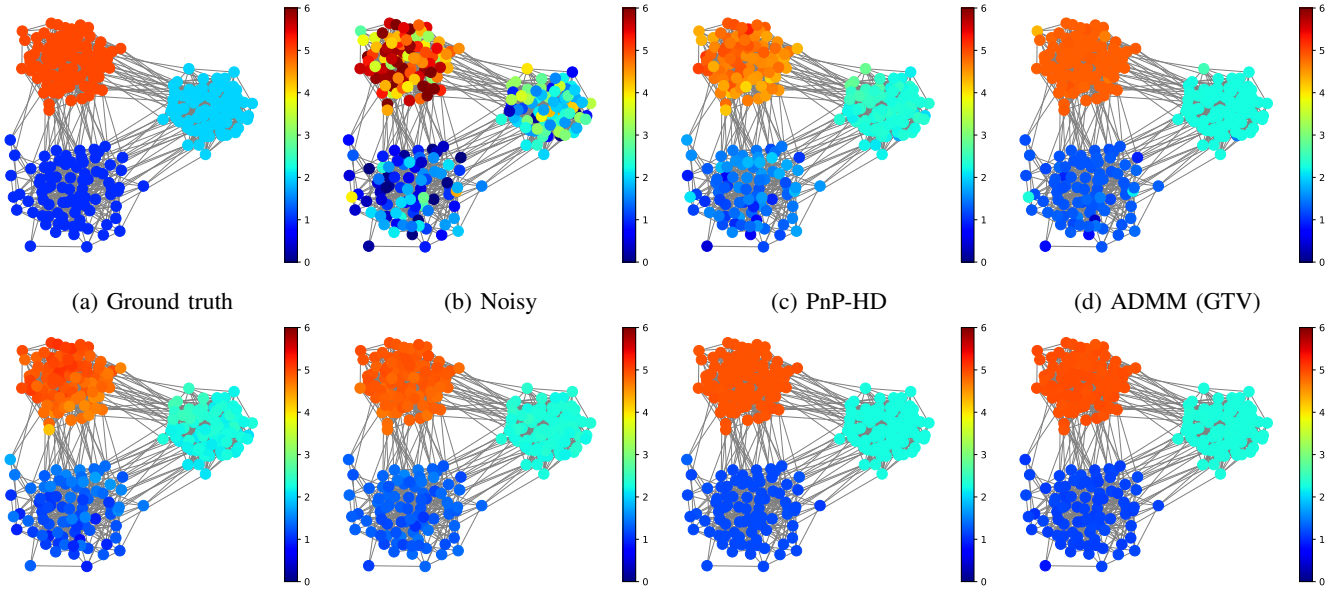
In this section, graph signal interpolation is performed and compared with the alternative methods. We assume the nodes for missing signal values are known and they are set to zero. This leads to a diagonal binary matrix $\mathbf{H} = \text{diag}\{0, 1\}^N$ in (1) with various missing rates.

A. Alternative Methods

For interpolation, the following techniques are selected for comparison:

- Bandlimited graph signal recovery based on graph sampling theory [8]: Bandwidth is set to $N/10$;
- PnP-ADMM-based interpolation with fixed parameters with 8 iterations [21]: Its formulation is given in Section III-B and off-the-shelf denoisers are HD or SGBF;

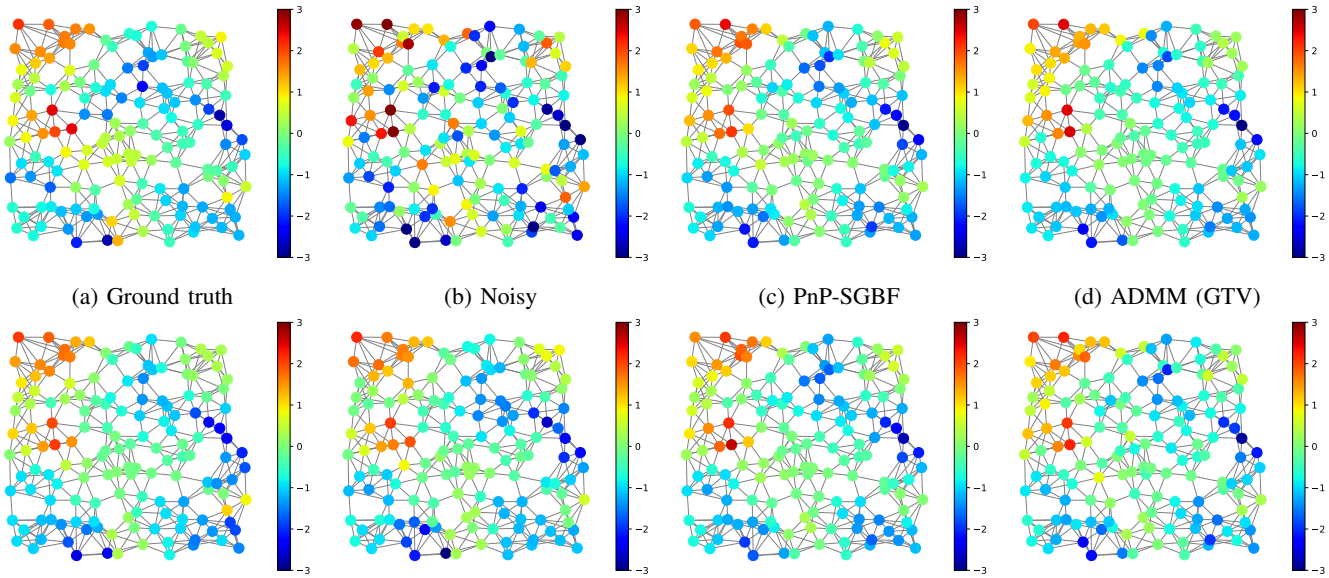
This article has been accepted for publication in IEEE Transactions on Signal Processing. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TSP.2022.3180546



(a) Ground truth (b) Noisy (c) PnP-HD (d) ADMM (GTV)

(e) GUSC (f) GUTF (g) GraphDAU-TV-E (h) NestDAU-TV-C

Fig. 3: Visualization: Denoising results of signals on community graph with $\sigma = 1.0$.



(a) Ground truth (b) Noisy (c) PnP-SGBF (d) ADMM (GTV)

(e) GUSC (f) GUTF (g) GraphDAU-EN-E (h) NestDAU-TV-C

Fig. 4: Denoising signals on random sensor graph ($\sigma = 1.0$).

- GUTF [25];
- GUSC [25].

Although GUSC and GUTF are originally developed for a denoising task, we also include these methods to compare with neural network-based approaches. The setup is the same as the previous section.

B. Datasets and Setup

We used the following graph signals for interpolation:

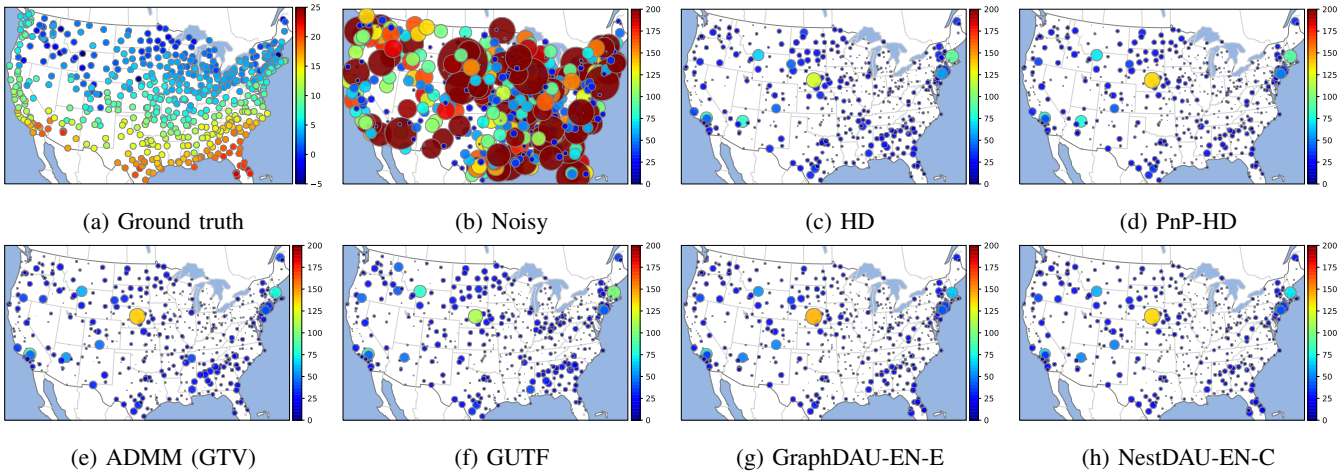
- Synthetic signals on a community graph having three clusters ($N = 250$);
- Temperature data of the United States ($N = 614$).

They are the same signals as those used in the denoising experiment in the previous section.

Characteristics of Graphs and Graph Signals: Synthetic graph signals on the community graph are generated in the same setup as that of the denoising experiment. We then consider two interpolation conditions: 1) noiseless and 2) noisy (AWGN with $\sigma = 0.5$). Three types of missing rate are considered: 30%, 50%, and 70%.

The U.S. temperature data are also used in this experiment as a real-world example. In this case, AWGN ($\sigma = 9.0$) are added onto the observed daily temperature data with the same

This article has been accepted for publication in IEEE Transactions on Signal Processing. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TSP.2022.3180546



(a) Ground truth (b) Noisy (c) HD (d) PnP-HD

(e) ADMM (GTV) (f) GUTF (g) GraphDAU-EN-E (h) NestDAU-EN-C

Fig. 5: Denoising results of U.S. temperature data ($\sigma = 9.0$). (a) is the original signal. The noisy and denoised results show the differences from the original signal: For visualization, the area of the nodes is proportional to the magnitude of the error, i.e., a large node has a large error.

TABLE VI: Denoising results on graphs with perturbation (average RMSEs for test data)

				Random sensor graph						3D Point Clouds		
				Piecewise constant		Piecewise smooth		Globally smooth		RGB colors		
Methods	L	K	P	$\sigma = 0.5$	1.0	0.5	1.0	0.5	1.0	20	30	40
Noisy	-	-	-	0.500	1.000	0.500	1.000	0.500	1.000	18.18	26.00	33.19
HD	-	-	-	0.412	0.627	0.417	0.646	0.357	0.537	11.84	14.74	17.42
SGBF	-	-	-	0.405	0.634	0.411	0.691	0.336	0.459	14.76	16.12	17.94
ADMM (GTV)	10	-	-	0.271	0.505	0.462	0.652	0.361	0.543	10.96	14.08	16.84
PnP-HD	-	-	8	0.415	0.645	0.413	0.645	0.343	0.538	11.89	14.97	17.78
PnP-SGBF	-	-	8	0.403	0.630	0.412	0.635	0.308	0.462	15.92	20.49	24.33
GraphDAU-TV-E	10	-	-	0.240	0.446	0.385	0.629	0.306	0.467	10.92	13.97	16.69
GraphDAU-TV-C	10	10	-	0.221	0.447	0.385	0.623	0.308	0.473	10.88	13.70	15.90
GraphDAU-EN-E	10	-	-	**0.192**	0.401	0.394	0.629	0.312	0.451	**10.74**	13.83	16.64
GraphDAU-EN-C	10	10	-	0.230	0.436	0.398	0.638	0.293	**0.436**	10.88	**13.67**	15.90
NestDAU-TV-E	10	-	8	0.206	0.407	0.368	0.613	**0.290**	0.439	10.83	13.75	16.06
NestDAU-TV-C	10	10	8	0.207	0.402	0.371	**0.606**	**0.290**	0.439	10.84	13.68	**15.87**
NestDAU-EN-E	10	-	8	0.206	0.410	0.371	0.612	**0.290**	0.441	10.89	13.92	16.45
NestDAU-EN-C	10	10	8	0.211	**0.398**	**0.366**	0.609	0.295	**0.436**	10.81	13.65	15.89

TABLE VII: Parameter transfer of 3D point clouds (average RMSEs for test data)

	N'	N		
	1,000	2,000	5,000	10,000
Noisy ($\sigma = 30$)	26.00	25.96	25.94	25.95
Denoised	13.70	12.69	11.63	11.03

setting as the denoising experiment. Then, missing rates are set to 30%, 50%, and 70% to validate the interpolation method. Note that the missing nodes are randomly chosen, i.e., $\mathbf{H}$ are set to be different across all data.

C. Interpolation Results

The RMSE results obtained by the proposed and existing methods are summarized in Table VIII. The visualizations of the interpolation results are also shown in Figs. 9 and 10.

As can be seen, the proposed approaches show better RMSE than the alternatives. For the community graph, NestDAU-TV shows better results than NestDAU-EN. This is because NestDAU-TV reflects the prior of the graph signals, i.e., piecewise constant. For the U.S. temperature data, NestDAU-EN is better than NestDAU-TV because the temperature data tend to be very smooth on the graph. In particular, NestDAU-EN-C outperforms the others in all missing rates. This implies that the proposed NestDAU presents its effectiveness beyond denoising.

VI. Concluding Remarks

In this paper, we proposed graph signal denoising and restoration methods based on ADMM and Plug-and-Play ADMM with deep algorithm unrolling, respectively. The ADMM-based unrolled denoiser automatically controls its regularization strengths by tuning its parameters from training data. The PnP-ADMM-based unrolled restoration is applicable to any linear degradation matrix and contains the proposed ADMM-based denoiser in its sub-module, leading to a nested DAU structure. The unrolled restoration methods provide fully interpretable structures and have a small number of parameters with respect to fully parameterized neural networks. The techniques only tune layer-wise trainable parameters in the iterative algorithm and do not include fully-connected neural networks. This implies that we only need a small set of training data: It is beneficial especially for graph signals because their underlying structures often change. In extensive

This article has been accepted for publication in IEEE Transactions on Signal Processing. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TSP.2022.3180546

TABLE VIII: Interpolation results. (average RMSEs for test data)

			Community graph						U.S. temperature		
			noiseless			$\sigma = 0.5$			$\sigma = 9.0$		
	P	params/miss(%)	30	50	70	30	50	70	30	50	70
Noisy + missing	-	-	2.034	2.633	3.092	2.099	2.691	3.155	9.143	9.169	9.170
Graph sampling theory	-	-	0.214	0.279	0.571	0.341	0.465	1.411	3.961	4.851	10.06
PnP-HD	8	-	0.131	0.179	0.386	0.263	0.316	0.462	2.938	3.155	3.558
PnP-SGBF	8	-	0.136	0.174	0.268	0.218	0.236	0.331	2.982	3.293	3.512
GUSC	-	11,270	0.384	0.543	0.706	0.414	0.551	0.722	3.853	4.921	6.657
GUTF	-	19,397	0.309	0.470	0.641	0.338	0.472	0.636	3.478	4.395	6.162
NestDAU-TV-E	8	168	0.013	0.025	**0.059**	0.077	0.094	**0.140**	2.940	3.177	3.491
NestDAU-TV-C	8	168	**0.012**	**0.022**	0.082	**0.072**	**0.093**	0.148	2.968	3.179	3.501
NestDAU-EN-E	8	248	0.077	0.140	0.160	0.107	0.185	0.311	2.907	3.116	3.466
NestDAU-EN-C	8	248	0.084	0.120	0.175	0.123	0.128	0.321	**2.903**	**3.105**	**3.461**

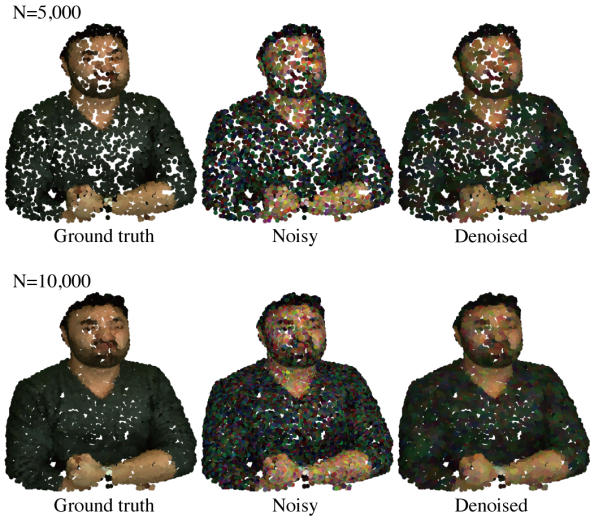


Fig. 6: **Parameter transfer:** The parameters of GraphDAU-TV-C with $L = 10$ and $K = 10$ are trained with $N = 1,000$. Then, the model with trained parameters are applied to the larger number of points ($N = 5,000$ (top row) and $N = 10,000$ (bottom row)).

experiments, the proposed methods experimentally outperform various alternative techniques for graph signal restoration. Furthermore, we can reuse the learned parameters for graphs with different sizes.

APPENDIX

Here, we present some non-trivial gradient computations of the trainable parameters with respect to the learnable parameters in GraphDAU.

First, let $\mathcal{L}$ be the loss function. Its partial derivatives with respect to parameters are given as follows using the chain rule:

$$\frac{\partial \mathcal{L}}{\partial \gamma_\ell} = \frac{\partial \mathcal{L}}{\partial \mathbf{x}^{(\ell+1)}} \cdot \frac{\partial \mathbf{x}^{(\ell+1)}}{\partial \gamma_\ell}, \tag{18}$$

$$\frac{\partial \mathcal{L}}{\partial \beta_\ell} = \frac{\partial \mathcal{L}}{\partial \mathbf{v}^{(\ell+1)}} \cdot \frac{\partial \mathbf{v}^{(\ell+1)}}{\partial \beta_\ell}, \tag{19}$$

$$\frac{\partial \mathcal{L}}{\partial \alpha_\ell} = \frac{\partial \mathcal{L}}{\partial \mathbf{v}^{(\ell+1)}} \cdot \frac{\partial \mathbf{v}^{(\ell+1)}}{\partial \alpha_\ell}. \tag{20}$$

From (10a), $\mathbf{x}^{(\ell+1)}$ is given as

$$\begin{aligned}\mathbf{x}^{(\ell+1)} =& \mathbf{U}\mathrm{diag}\left(\frac{\gamma_\ell}{\gamma_\ell+\lambda_1},\dots,\frac{\gamma_\ell}{\gamma_\ell+\lambda_N}\right)\mathbf{U}^\top \\ &\times\left(\mathbf{y}+\frac{1}{\gamma_\ell}\mathbf{M}^\top(\mathbf{v}^{(\ell)}-\mathbf{u}^{(\ell)})\right) \\ =& \mathbf{U}\left[\mathrm{diag}\left(\frac{\gamma_\ell}{\gamma_\ell+\lambda_1},\dots,\frac{\gamma_\ell}{\gamma_\ell+\lambda_N}\right)\mathbf{U}^\top\mathbf{y}\right. \\ &\left.+\ \mathrm{diag}\left(\frac{1}{\gamma_\ell+\lambda_1},\dots,\frac{1}{\gamma_\ell+\lambda_N}\right)\widetilde{\mathbf{x}}^{(\ell)}\right],\end{aligned}$$

where $\widetilde{\mathbf{x}}^{(\ell)} = \mathbf{U}^\top\mathbf{M}^\top(\mathbf{v}^{(\ell)}-\mathbf{u}^{(\ell)})$. Then, $\frac{\partial \mathbf{x}^{(\ell+1)}}{\partial \gamma_\ell}$ in (18) is calculated as follows:

$$\begin{aligned}\frac{\partial \mathbf{x}^{(\ell+1)}}{\partial \gamma_\ell} =& \mathbf{U}\left[\frac{\partial}{\partial \gamma_\ell}\mathrm{diag}\left(\frac{\gamma_\ell}{\gamma_\ell+\lambda_1},\dots,\frac{\gamma_\ell}{\gamma_\ell+\lambda_N}\right)\mathbf{U}^\top\mathbf{y}\right. \\ &\left.+\ \frac{\partial}{\partial \gamma_\ell}\mathrm{diag}\left(\frac{1}{\gamma_\ell+\lambda_1},\dots,\frac{1}{\gamma_\ell+\lambda_N}\right)\widetilde{\mathbf{x}}^{(\ell)}\right] \\ =& \mathbf{U}\left[\mathrm{diag}\left(\frac{\lambda_1}{(\gamma_\ell+\lambda_1)^2},\dots,\frac{\lambda_N}{(\gamma_\ell+\lambda_N)^2}\right)\mathbf{U}^\top\mathbf{y}\right. \\ &\left.-\ \mathrm{diag}\left(\frac{1}{(\gamma_\ell+\lambda_1)^2},\dots,\frac{1}{(\gamma_\ell+\lambda_N)^2}\right)\widetilde{\mathbf{x}}^{(\ell)}\right].\end{aligned} \tag{21}$$

We also derive the partial derivatives with respect to β_ℓ for both GraphDAU-TV and EN. Let $\widetilde{\mathbf{v}}^{(\ell)} = \mathbf{M}\mathbf{x}^{(\ell+1)} + \mathbf{u}^{(\ell)}$ and $\sigma(\cdot)$ be the ReLU activation function. The soft-thresholding operator can be represented with two ReLU functions as

$$S_{\beta_\ell}(\widetilde{\mathbf{v}}_i^{(\ell)}) = \sigma(\widetilde{\mathbf{v}}_i^{(\ell)} - \beta_\ell) - \sigma(-\widetilde{\mathbf{v}}_i^{(\ell)} - \beta_\ell).$$

Therefore, the auxiliary variable $\mathbf{v}^{(\ell+1)}$ is explicitly given by $\mathbf{v}^{(\ell+1)} = S_{\beta_\ell}(\widetilde{\mathbf{v}}^{(\ell)})$, and its gradient for β_ℓ is shown as follows:

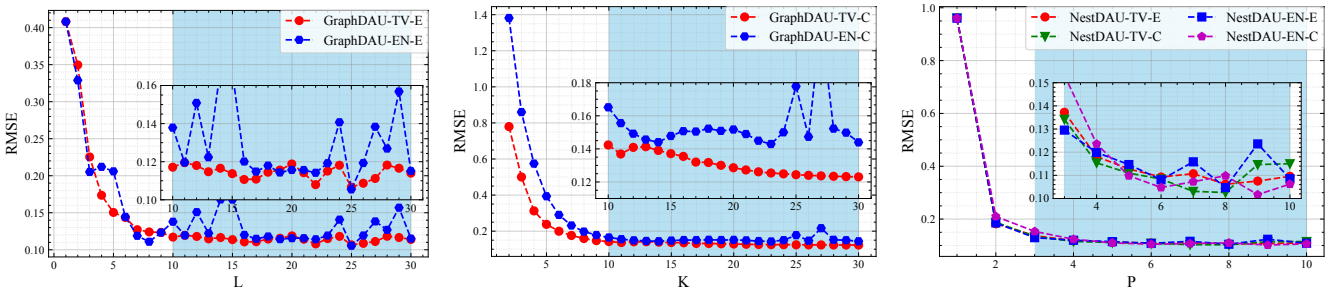
$$\frac{\partial \mathbf{v}_i^{(\ell+1)}}{\partial \beta_\ell} = -\sigma'(\widetilde{\mathbf{v}}_i^{(\ell)} - \beta_\ell) + \sigma'(-\widetilde{\mathbf{v}}_i^{(\ell)} - \beta_\ell), \tag{22}$$

where $\sigma'(\cdot)$ is the derivative of $\sigma(\cdot)$. GraphDAU-EN also contains the parameter α_ℓ. By taking the partial derivative of $\mathbf{v}^{(\ell+1)}$ with respect to α_ℓ is given as follows:

$$\mathbf{v}_i^{(\ell+1)} = \alpha_\ell\left(\sigma(\widetilde{\mathbf{v}}_i^{(\ell)} - \beta_\ell) - \sigma(-\widetilde{\mathbf{v}}_i^{(\ell)} - \beta_\ell)\right),$$

$$\frac{\partial \mathbf{v}_i^{(\ell+1)}}{\partial \alpha_\ell} = \sigma(\widetilde{\mathbf{v}}_i^{(\ell)} - \beta_\ell) - \sigma(-\widetilde{\mathbf{v}}_i^{(\ell)} - \beta_\ell). \tag{23}$$

This article has been accepted for publication in IEEE Transactions on Signal Processing. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TSP.2022.3180546



(a) GraphDAU: L. (b) GraphDAU: K. (c) NestDAU: P.

Fig. 7: Denoiser analysis with the results on community graph ($\sigma = 1.0$).

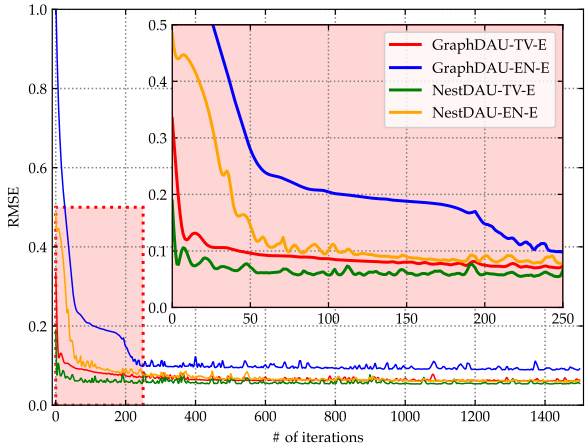


Fig. 8: Average RMSEs of the validation data for graph signal denoising. The upper right box is the zoomed-in part of the red area at bottom left.

REFERENCES

[1] M. Nagahama, K. Yamada, Y. Tanaka, S. H. Chan, and Y. C. Eldar, "Graph signal denoising using nested-structured deep algorithm unrolling," in *Proc. IEEE Int. Conf. Acoust. Speech. Signal Process.*, 2021, pp. 5280–5284.

[2] A. Sandryhaila and J. M. F. Moura, "Discrete signal processing on graphs," *IEEE Trans. Signal Process.*, vol. 61, no. 7, pp. 1644–1656, Apr. 2013.

[3] D. I. Shuman, S. K. Narang, P. Frossard, A. Ortega, and P. Vandergheynst, "The emerging field of signal processing on graphs: Extending high-dimensional data analysis to networks and other irregular domains," *IEEE Signal Process. Mag.*, vol. 30, no. 3, pp. 83–98, May 2013.

[4] A. Ortega, P. Frossard, J. Kovačević, J. M. F. Moura, and P. Vandergheynst, "Graph signal processing: Overview, challenges, and applications," *in Proc. the IEEE*, vol. 106, no. 5, pp. 808–828, May 2018.

[5] W. Hu, G. Cheung, A. Ortega, and O. C. Au, "Multiresolution graph fourier transform for compression of piecewise smooth images," *IEEE Trans. on Image Process.*, vol. 24, no. 1, pp. 419–433, Jan. 2015.

[6] S. Chen, R. Varma, A. Sandryhaila, and J. Kovačević, "Discrete signal processing on graphs: Sampling theory," *IEEE Trans. Signal Process.*, vol. 63, no. 24, pp. 6510–6523, Dec. 2015.

[7] A. Anis, A. Gadde, and A. Ortega, "Efficient sampling set selection for bandlimited graph signals using graph spectral proxies," *IEEE Trans. Signal Process.*, vol. 64, no. 14, pp. 3775–3789, Jul. 2016.

[8] Y. Tanaka, Y. C. Eldar, A. Ortega, and G. Cheung, "Sampling signals on graphs: From theory to applications," *IEEE Signal Process. Mag.*, vol. 37, no. 6, pp. 14–30, Nov. 2020.

[9] J. Hara, Y. Tanaka, and Y. C. Eldar, "Generalized graph spectral sampling with stochastic priors," in *Proc. IEEE Int. Conf. Acoust. Speech. Signal Process.*, 2020, pp. 5680–5684.

[10] Y. Tanaka and Y. C. Eldar, "Generalized sampling on graphs with subspace and smoothness priors," *IEEE Trans. Signal Process.*, vol. 68, pp. 2272–2286, 2020.

[11] A. Sakiyama, K. Watanabe, and Y. Tanaka, "Spectral graph wavelets and filter banks with low approximation error," *IEEE Trans. Signal Inf. Process. Netw.*, vol. 2, no. 3, pp. 230–245, Sep. 2016.

[12] D. I. Shuman, "Localized spectral graph filter frames: A unifying framework, survey of design considerations, and numerical comparison," *IEEE Signal Process. Mag.*, vol. 37, no. 6, pp. 43–63, Nov. 2020.

[13] S. K. Narang, A. Gadde, and A. Ortega, "Signal processing techniques for interpolation in graph structured data," *in Proc. IEEE Int. Conf. Acoust. Speech Signal Process.*, pp. 5445–5449, May 2013.

[14] S. Chen, A. Sandryhaila, J. M. F. Moura, and J. Kovačević, "Signal recovery on graphs: Variation minimization," *IEEE Trans. Signal Process.*, vol. 63, no. 17, pp. 4609–4624, Sep. 2015.

[15] S. Chen, A. Sandryhaila, J. M. F. Moura, and J. Kovacevic, "Signal denoising on graphs via graph filtering," in *Proc. IEEE Global Conf. on Signal and Inf. Process.*, Dec. 2014, pp. 872–876.

[16] M. Onuki, S. Ono, M. Yamagishi, and Y. Tanaka, "Graph signal denoising via trilateral filter on graph spectral domain," *IEEE Trans. on Signal and Inf. Process. over Networks*, vol. 2, no. 2, pp. 137–148, Jun. 2016.

[17] Y. Tanaka and A. Sakiyama, "M-channel oversampled graph filter banks," *IEEE Trans. Signal Process.*, vol. 62, no. 14, pp. 3578–3590, Jul. 2014.

[18] T. H. Do, D. Minh Nguyen, and N. Deligiannis, "Graph auto-encoder for graph signal denoising," *in Proc. IEEE Int. Conf. Acoust. Speech. Signal Process.*, pp. 3322–3326, May 2020.

[19] T. Chan, S. Osher, and J. Shen, "The digital TV filter and nonlinear denoising," *IEEE Trans. on Image Process.*, vol. 10, no. 2, pp. 231–241, Feb. 2001.

[20] S. Ono, I. Yamada, and I. Kumazawa, "Total generalized variation for graph signals," *in Proc. IEEE Int. Conf. Acoust. Speech. Signal Process.*, pp. 5456–5460, Apr. 2015.

[21] Y. Yazaki, Y. Tanaka, and S. H. Chan, "Interpolation and denoising of graph signals using Plug-and-Play ADMM," *in Proc. IEEE Int. Conf. Acoust. Speech. Signal Process.*, pp. 5431–5435, May 2019.

[22] A. Gadde, S. K. Narang, and A. Ortega, "Bilateral filter: Graph spectral interpretation and extensions," *in Proc. IEEE Int. Conf. Image Process.*, Mar. 2013.

[23] T. N. Kipf and M. Welling, "Semi-supervised classification with graph convolutional networks," in *Proc. Int. Conf. Learn. Representations*, 2017, p. 14.

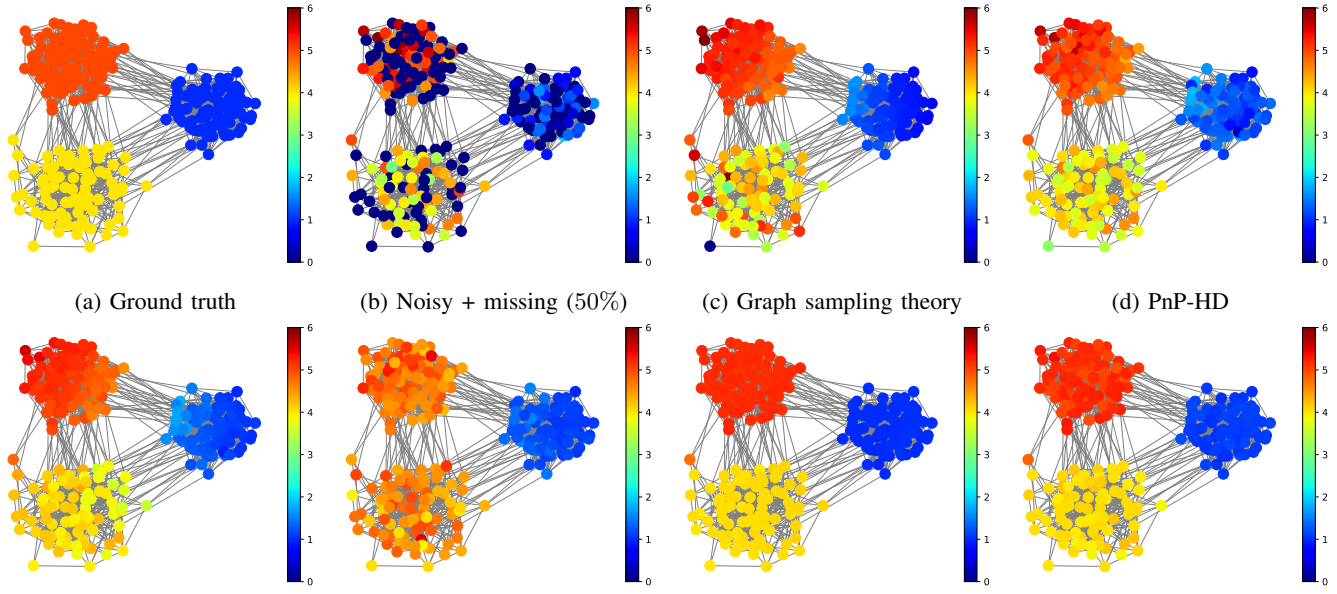
[24] M. Defferrard, X. Bresson, and P. Vandergheynst, "Convolutional neural networks on graphs with fast localized spectral filtering," *in Proc. Int. Conf. Neural Inf. Process. Syst.*, pp. 3844–3852, 2016.

[25] S. Chen, Y. C. Eldar, and L. Zhao, "Graph unrolling networks: Interpretable neural networks for graph signal denoising," *IEEE Trans. Signal Process.*, Jun. 2020.

[26] D. Ulyanov, A. Vedaldi, and V. Lempitsky, "Deep image prior," *Int J Comput Vis*, vol. 128, no. 7, pp. 1867–1888, Jul. 2020.

[27] M. Zhang, Z. Cui, M. Neumann, and Y. Chen, "An end-to-end deep learning architecture for graph classification," in *Proc. AAAI*, 2018, p. 8.

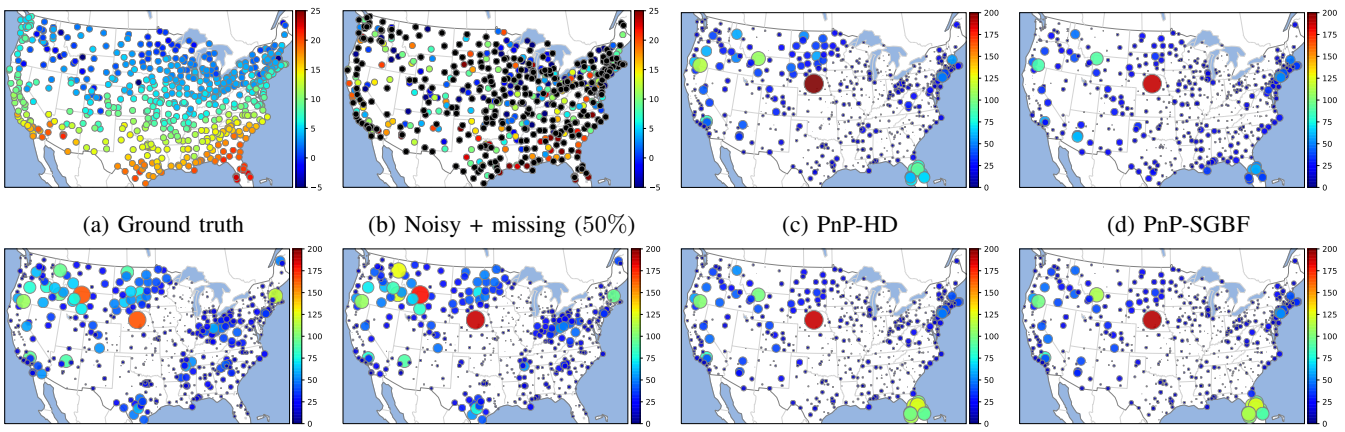
This article has been accepted for publication in IEEE Transactions on Signal Processing. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TSP.2022.3180546



(a) Ground truth (b) Noisy + missing (50%) (c) Graph sampling theory (d) PnP-HD

(e) PnP-SGBF (f) GUTF (g) NestDAU-TV-E (h) NestDAU-EN-E

Fig. 9: Interpolation results of a community graph (AWGN ($\sigma = 0.5$) with 50% missing). The proposed method ((g) and (h)) captured the property of the ground truth signal.



(a) Ground truth (b) Noisy + missing (50%) (c) PnP-HD (d) PnP-SGBF

(e) GUSC (f) GUTF (g) NestDAU-TV-E (h) NestDAU-EN-C

Fig. 10: Interpolation results of U.S temperature data. (a) and (b) are shown in the original scale. For clear visualization, the node size of the interpolation results are set to be proportional to the magnitude of the error.

[28] Y. Wang, Y. Sun, Z. Liu, S. E. Sarma, M. M. Bronstein, and J. M. Solomon, "Dynamic graph CNN for learning on point clouds," *ACM Transactions on Graphics*, 2019.

[29] X. Zhang, Y. Lu, J. Liu, and B. Dong, "Dynamically unfolding recurrent restorer: A moving endpoint control method for image restoration," *Proc. Int. Conf. Learn. Representations*, Sep. 2018.

[30] Y. Li, M. Tofighi, J. Geng, V. Monga, and Y. C. Eldar, "Efficient and interpretable deep blind image deblurring via algorithm unrolling," *IEEE Trans. Comput. Imaging*, vol. 6, pp. 666–681, 2020.

[31] C. Bertocchi, E. Chouzenoux, M.-C. Corbineau, J.-C. Pesquet, and M. Prato, "Deep unfolding of a proximal interior point method for image restoration," *Inverse Problems*, vol. 36, no. 3, p. 034005, Feb. 2020.

[32] 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, 2021.

[33] K. Gregor and Y. LeCun, "Learning fast approximations of sparse coding," in *Proc. Int. Conf. Mach. Learn.*, 2010, pp. 399–406.

[34] S. V. Venkatakrishnan, C. A. Bouman, and B. Wohlberg, "Plug-and-play priors for model based reconstruction," in *Proc. IEEE Global Conf. on Signal and Inf. Process.* IEEE, Dec. 2013, pp. 945–948.

[35] R. Ahmad, C. A. Bouman, G. T. Buzzard, S. Chan, S. Liu, E. T. Reehorst, and P. Schniter, "Plug-and-play methods for magnetic resonance imaging: Using denoisers for image recovery," *IEEE Signal Process. Mag.*, vol. 37, no. 1, pp. 105–116, Jan. 2020.

[36] S. Sreehari, S. V. Venkatakrishnan, B. Wohlberg, L. F. Drummy, J. P. Simmons, and C. A. Bouman, "Plug-and-play priors for bright field electron tomography and sparse interpolation," *IEEE Trans. Comput. Imaging*, pp. 1–1, 2016.

[37] S. H. Chan, X. Wang, and O. A. Elgendy, "Plug-and-play ADMM for image restoration: Fixed-point convergence and applications," *IEEE Trans. Comput. Imaging*, vol. 3, no. 1, pp. 84–98, Mar. 2017.

[38] S. H. Chan, "Performance analysis of plug-and-play ADMM: A graph signal processing perspective," *IEEE Trans. Comput. Imaging*, vol. 5, no. 2, pp. 274–286, Jun. 2019.

[39] S. Boyd, "Distributed optimization and statistical learning via the alternating direction method of multipliers," *FNT in Machine Learning*, vol. 3, no. 1, pp. 1–122, 2010.

[40] K. Zhang, W. Zuo, S. Gu, and L. Zhang, "Learning deep CNN denoiser

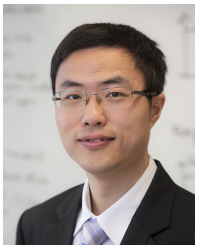
prior for image restoration," in *IEEE/CVF Conf. on Comput. Vision and Pattern Recognition*. IEEE, 2017, pp. 2808–2817.
[41] K. Zhang, W. Zuo, and L. Zhang, "Deep Plug-and-Play super-resolution for arbitrary blur kernels," in *IEEE/CVF Conf. on Computer Vision and Pattern Recognition*. IEEE, 2019, pp. 1671–1681.
[42] K. Wei, A. Aviles-Rivero, J. Liang, Y. Fu, C.-B. Schönlieb, and H. Huang, "Tuning-free Plug-and-Play proximal algorithm for inverse imaging problems," in *Proc. Int. Conf. Mach. Learn. ICML*, 2020.
[43] U. S. Kamilov, H. Mansour, and B. Wohlberg, "A Plug-and-Play priors approach for solving nonlinear imaging inverse problems," *IEEE Signal Process. Lett.*, vol. 24, no. 12, pp. 1872–1876, 2017.
[44] S. K. Shastri, R. Ahmad, and P. Schniter, "Autotuning Plug-and-Play algorithms for MRI," *arXiv:2012.00887 [cs, math]*, 2020.
[45] H. Zou and T. Hastie, "Regularization and variable selection via the elastic net," *J Royal Statistical Soc B*, vol. 67, no. 2, pp. 301–320, Apr. 2005.
[46] J. Pang and G. Cheung, "Graph laplacian regularization for image denoising: Analysis in the continuous domain," *IEEE Trans. Image Process.*, vol. 26, no. 4, pp. 1770–1785, Apr. 2017.
[47] N. Parikh and S. Boyd, "Proximal algorithms," *OPT*, vol. 1, no. 3, pp. 127–239, Jan. 2014.
[48] S. G. Mallat, *A Wavelet Tour of Signal Processing: The Sparse Way*, 3rd ed. Elsevier/Academic Press, 2009.
[49] M. Onuki, S. Ono, K. Shirai, and Y. Tanaka, "Fast singular value shrinkage with chebyshev polynomial approximation based on signal sparsity," *IEEE Trans. Signal Process.*, vol. 65, no. 22, pp. 6083–6096, Nov. 2017.
[50] K. Yamamoto, M. Onuki, and Y. Tanaka, "Deblurring of point cloud attributes in graph spectral domain," in *Proc. IEEE Int. Conf. Image Process.*, 2016, pp. 1559–1563.
[51] C. Dinesh, G. Cheung, and I. V. Bajić, "Super-resolution of 3D color point clouds via fast graph total variation," in *Proc. IEEE Int. Conf. Acoust. Speech. Signal Process.*, 2020, pp. 1983–1987.
[52] Y. Wang, J. Yang, W. Yin, and Y. Zhang, "A new alternating minimization algorithm for total variation image reconstruction," *SIAM J. Imaging Sci.*, vol. 1, no. 3, pp. 248–272, Jan. 2008.
[53] F. Zhang and E. R. Hancock, "Graph spectral image smoothing using the heat kernel," *Pattern Recognition*, vol. 41, no. 11, pp. 3328–3342, Nov. 2008.
[54] Y. Wang, A. Ortega, D. Tian, and A. Vetro, "A graph-based joint bilateral approach for depth enhancement," *in Proc. IEEE Int. Conf. Acoust. Speech. Signal Process.*, pp. 885–889, May 2014.
[55] M. Defferrard, L. Martin, R. Pena, and N. Perraudin, "Pygsp: Graph signal processing in python," Zenodo, Oct. 2017.
[56] P. Veličković, G. Cucurull, A. Casanova, A. Romero, P. Liò, and Y. Bengio, "Graph attention networks," in *Proc. Int. Conf. Learn. Representations*, 2018.
[57] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, A. Desmaison, A. Kopf, E. Yang, Z. DeVito, M. Raison, A. Tejani, S. Chilamkurthy, B. Steiner, L. Fang, J. Bai, and S. Chintala, "Pytorch: An imperative style, high-performance deep learning library," *in Proc. Int. Conf. Neural Inf. Process. Syst.*, pp. 8026–8037, 2019.
[58] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," in *Proc. Int. Conf. Learn. Representations*, 2015, pp. 1–15.
[59] National Oceanic and Atmospheric Administration, "Quality controlled local climatological data (QCLCD)," https://www.ncdc.noaa.gov/orders/qclcd/.
[60] C. Loop, Q. Cai, S. Orts-Escolano, and P. A. Chou, "Microsoft voxelized upper bodies - a voxelized point cloud dataset."

Masatoshi Nagahama (S'20) received the B.E. degree in Computer and Information Sciences in 2020 and the M.E. degree in Engineering in 2022 from Tokyo University of Agriculture and Technology, Tokyo, Japan. His current research interests are graph signal processing and deep algorithm unrolling.

Koki Yamada (S'19) received the B.S. degree in physics and informatics in 2015 and the M.S. degree in science and engineering in 2017 from Yamaguchi University. He also received the Ph.D. degree in engineering from Tokyo University of Agriculture and Technology in 2022. He has been an Assistant Professor in the Department of Electrical Engineering, Tokyo University of Science, Tokyo, Japan, since 2022. He was a Research Fellow (DC2) of the Japan Society for the Promotion of Science (JSPS) from 2020 to 2022. His current research interests are graph signal processing and high-dimensional data analysis.

Yuichi Tanaka (S'06–M'07–SM'18) received the B.E., M.E. and Ph.D. degrees in electrical engineering from Keio University, Yokohama, Japan, in 2003, 2005, and 2007, respectively. He was a Postdoctoral Scholar at Keio University, Yokohama, Japan, from 2007 to 2008, and supported by the Japan Society for the Promotion of Science (JSPS). From 2006 to 2008, he was also a visiting scholar at the University of California, San Diego. From 2008 to 2012, he was an Assistant Professor in the Department of Information Science, Utsunomiya University, Tochigi, Japan. Since 2012, he has been an Associate Professor in the Department of Electrical Engineering and Computer Science, Tokyo University of Agriculture and Technology, Tokyo, Japan. Currently he has a cross appointment as a PRESTO Researcher, Japan Science and Technology Agency. His current research interests are in the field of high-dimensional signal processing and machine learning which includes: graph signal processing, geometric deep learning, sensor networks, image/video processing in extreme situations, biomedical signal processing, and remote sensing.

Dr. Tanaka served as an associate editor for the IEEE TRANSACTIONS ON SIGNAL PROCESSING from 2016 to 2020 and the *IEICE Transactions on Fundamentals* from 2013 to 2017. Currently he is an elected member of the APSIPA SIPTM (Signal and Information Processing Theory and Methods) and IVM (Image, Video and Multimedia) Technical Committees. He was a recipient of the Yasujiro Niwa Outstanding Paper Award in 2010, the TELECOM System Technology Award in 2011, and Ando Incentive Prize for the Study of Electronics in 2015. He also received IEEE Signal Processing Society Japan Best Paper Award in 2016 and Best Paper Awards in APSIPA ASC 2014 and 2015.



Stanley H. Chan (S'06–M'12–SM'17) received the B.Eng. degree (with first-class honor) in Electrical Engineering from the University of Hong Kong in 2007, the M.A. degree in Mathematics from the University of California at San Diego in 2009, and the Ph.D. degree in Electrical Engineering from the University of California at San Diego in 2011. From 2012 to 2014, he was a postdoctoral research fellow at Harvard John A. Paulson School of Engineering and Applied Sciences. He joined Purdue University, West Lafayette, IN in 2014, where he is currently an Elmore Associate Professor of Electrical and Computer Engineering.

Dr. Chan is a recipient of the Best Paper Award of IEEE International Conference on Image Processing 2016, IEEE Signal Processing Cup 2016 Second Prize, Purdue College of Engineering Exceptional Early Career Teaching Award 2019, Purdue College of Engineering Outstanding Graduate Mentor Award 2016, and Eta Kappa Nu (Beta Chapter) Outstanding Teaching Award 2015. He is also a recipient of the Croucher Foundation Fellowship for Postdoctoral Research 2012-2013 and the Croucher Foundation Scholarship for PhD Studies 2008-2010. His research interests include single-photon imaging, imaging through atmospheric turbulence, and computational photography.

He is an Associate Editor of the IEEE Transactions on Computational Imaging since 2018 where he is recognized as an outstanding editorial board member in 2021. He also served as an Associate Editor of the OSA Optics Express in 2016-2018, and an Elected Member of the IEEE Signal Processing Society Technical Committee in Computational Imaging in 2015-2020.

Yonina C. Eldar (S'98–M'02–SM'07–F'12) received the B.Sc. degree in Physics in 1995 and the B.Sc. degree in Electrical Engineering in 1996 both from Tel-Aviv University (TAU), Tel-Aviv, Israel, and the Ph.D. degree in Electrical Engineering and Computer Science in 2002 from the Massachusetts Institute of Technology (MIT), Cambridge. She is currently a Professor in the Department of Mathematics and Computer Science, Weizmann Institute of Science, Rehovot, Israel. She was previously a Professor in the Department of Electrical Engineering at the Technion. She is also a Visiting Professor at MIT, a Visiting Scientist at the Broad Institute, and an Adjunct Professor at Duke University and was a Visiting Professor at Stanford. She is a member of the Israel Academy of Sciences and Humanities (elected 2017), an IEEE Fellow and a EURASIP Fellow. Her research interests are in the broad areas of statistical signal processing, sampling theory and compressed sensing, learning and optimization methods, and their applications to biology, medical imaging and optics.

Dr. Eldar has received many awards for excellence in research and teaching, including the IEEE Signal Processing Society Technical Achievement Award (2013), the IEEE/AESS Fred Nathanson Memorial Radar Award (2014), and the IEEE Kiyo Tomiyasu Award (2016). She was a Horev Fellow of the Leaders in Science and Technology program at the Technion and an Alon Fellow. She received the Michael Bruno Memorial Award from the Rothschild Foundation, the Weizmann Prize for Exact Sciences, the Wolf Foundation Krill Prize for Excellence in Scientific Research, the Henry Taub Prize for Excellence in Research (twice), the Hershel Rich Innovation Award (three times), the Award for Women with Distinguished Contributions, the Andre and Bella Meyer Lectureship, the Career Development Chair at the Technion, the Muriel & David Jacknow Award for Excellence in Teaching, and the Technion's Award for Excellence in Teaching (two times). She received several best paper awards and best demo awards together with her research students and colleagues including the SIAM outstanding Paper Prize, the UFFC Outstanding Paper Award, the Signal Processing Society Best Paper Award and the IET Circuits, Devices and Systems Premium Award, was selected as one of the 50 most influential women in Israel and in Asia, and is a highly cited researcher.

She was a member of the Young Israel Academy of Science and Humanities and the Israel Committee for Higher Education. She is the Editor in Chief of Foundations and Trends in Signal Processing, a member of the IEEE Sensor Array and Multichannel Technical Committee and serves on several other IEEE committees. In the past, she was a Signal Processing Society Distinguished Lecturer, member of the IEEE Signal Processing Theory and Methods and Bio Imaging Signal Processing technical committees, and served as an associate editor for the IEEE Transactions On Signal Processing, the EURASIP Journal of Signal Processing, the SIAM Journal on Matrix Analysis and Applications, and the SIAM Journal on Imaging Sciences. She was Co-Chair and Technical Co-Chair of several international conferences and workshops. She is author of the book "Sampling Theory: Beyond Bandlimited Systems" and co-author of five other books published by Cambridge University Press.