Robust Time-Varying Graph Signal Recovery Over Dynamic Topology

Eisuke Yamagata, Student Member, IEEE, Shunsuke Ono, Member, IEEE

Abstract—We propose a time-varying graph signal recovery method that estimates the true graph signal from an observation corrupted by missing values, outliers of unknown positions, and some random noise. Furthermore, we assume the underlying graph to be time-varying like the signals, which we integrate into our formulation for better performance. Conventional studies on time-varying graph signal recovery have been focusing on online estimation and graph learning under the assumption that entire dynamic graphs are unavailable. However, there are many practical situations where the underlying graphs can be observed or easily generated by simple algorithms like the $k$-nearest neighbor, especially when targeting physical sensing data, where the graphs can be defined to represent spatial correlations.

To address such cases, in this paper, we tackle a dynamic graph Laplacian-based recovery problem on given dynamic graphs. To solve this, we formulate the recovery problem as a constrained convex optimization problem to estimate both the time-varying graph signal and the sparsely modeled outliers simultaneously. We integrate the graph dynamics into the formulation by exploiting the given dynamic graph. In such a manner, we succeed in separating the different types of corruption and achieving state-of-the-art recovery performance in both synthetic and real-world problem settings. We also conduct an extensive study to compare the contribution of vertex and temporal domain regularization on the recovery performance.

Index Terms—Graph signal processing, graph signal recovery, time-varying graph signal, dynamic topology, constrained optimization.

I. INTRODUCTION

The concept of graph signal, defined by signal values observed on the vertex set $V$ of a graph $G$, has been intensely researched as an effective approach to represent irregularly structured data. Conventional signal processing is based on spatially or temporally regular structures (e.g., images and sounds), and thus, the relations between signal values are also regular, which provides no further information for us to leverage. On the other hand, graph signal representations and graph signal processing [1]–[3] explicitly represent relations between signal values with vertices and weighted edges, which we can exploit as priors in the vertex domain. Irregularly structured data such as traffic and sensor network data, geographical data, mesh data, and biomedical data all benefit from such representation.

Following the history of conventional signal processing, recovering the true graph signals from the often corrupted observations is a necessity in processing the data for further use. Many algorithms have been proposed to address the problem, mostly methods based on either the vertex [4]–[8] or the graph spectral domain [9]–[11]. These methods are proposed to exploit the smoothness of the signals on the vertex domain. A graph signal is smooth when signal values of two vertices connected by edges with large weights are similar. Such traits can often be observed on various real-world graph signals, for example, temperature data observed on a network of sensors where the edges represent the physical distances of the sensors. Sensors that are close-by would be connected by edges with large weights, and would also most likely observe similar temperatures, which gives us a smooth graph signal.

Although leveraging the graph signal smoothness can successfully recover the true signals, these methods ignore the temporal domain. In real-life scenarios, many of the above-mentioned data can easily be sampled continuously to form time-coherent data, and therefore, priors based on the temporal domain should be effectively utilized to improve estimation results.

A. Related Work

Several time-varying graph signal recovery methods have been discussed to leverage signal smoothness prior in the temporal domain together with that of the vertex domain.

The earlier studies involve filtering and Fourier transform of product graphs [16]. The more recent studies include spectral-based approaches like Joint time-vertex Fourier transform (JFT) [12], [17], [18] and vertex-domain-based approaches that leverages smoothness on the vertex and temporal domains [13]. JFT jointly applies Fourier transform to both the temporal direction and vertex direction to fully leverage priors of the two domains. In [13], the formulation is based on leveraging the smoothness of the temporal difference signal on the underlying graph, which effectively utilizes both domains for the estimation.

Generally speaking, time-varying graph signal recovery methods are often discussed under the assumption that the underlying graph is static, especially in cases of physical sensing. This is because, in practical applications, the graph (the weights of the edges) is pre-defined heuristically usually by a Gaussian kernel of spatial coordinates. Whether the graph represents a geographical network or a cranial nerve system or any other physical network, as long as the physical coordinates of the vertices are static, the graph is also static. Therefore, very little work has been done in the area of time-varying graph signal recovery on dynamic graph topology.

However, in the context of graph learning [19], [20], where graphs are learned from the graph signals, it is much more
natural to expect the graphs that house time-varying graph signals to be dynamic, just like the signals [21]. Under the assumption that dynamic graphs are better representations of the underlying structures that time-varying graph signals are observed on, some recovery methods have been proposed to exploit the dynamics of the graphs [14], [15].

The authors in [14] proposed a method to estimate time-varying graph signals while also estimating dynamic graphs, which are unknown, based on an online strategy to keep track of the temporal variation of both the graph and the graph signal. In [15], a space-time kernel is proposed leverage both domains on a graph extended in the vertex domain to represent the temporal domain.

### B. Contributions and Paper Organization

In this paper, we consider situations where the underlying dynamic graph topology can be observed or easily generated, in which case the complicated process of graph learning or online estimation can and should be avoided. There are many practical applications where graph signals can be observed on graphs that are explicitly dynamic that we can readily exploit, for example, sensing data observed on a network of sensor-loaded drones, cars, smartphones, or any other mobile devices. Considering how humans are sociable beings, graphs that represent social network services, where the vertices and the edges represent the users and their relationships, respectively, are going to be dynamic. Especially in physical sensing, graphs can be easily generated to represent spatial correlations by using simple algorithms like the $k$-nearest neighbor and defining the weights of the edges from the Gaussian kernel of the spatial coordinates of the vertices. This suggests that we should discuss recovery methods built upon pre-defined dynamic graph topologies.

To address this problem, we propose a method more focused on recovering signals under the assumption that dynamic graphs are available. Furthermore, we estimate sparsely modeled outliers together with the graph signals, making the method much more robust to various types of noise and corruptions. We also restore missing values, which can represent sensor malfunctions or maintenance in real-world settings. We impose data fidelity and outlier sparsity as hard constraints rather than part of cost functions to facilitate parameter tuning by decoupling the parameters, as has been addressed, e.g., in [22]–[25].

Table I is a simple representation of where our method lies in relation to the conventional methods. The conventional works force the following trade-offs to their formulations in order to consider online estimation, which is unnecessary in the above-mentioned problem setting: the inability to capture the global temporal correlations in [14], and the limitation to quadratic costs to adopt a Kalman filtering strategy in [15]. We overcome these trade-offs to fully utilize the temporal correlations on the entire time-slots, and select regularization terms that can better capture the characteristics of the target data.

The main contributions of this paper are as follows.

- We tackle an unaddressed problem setting, that is, recovering time-varying graph signals from noisy incomplete observations, possibly with outliers, on given dynamic graphs.
- We establish an optimization-based recovery method that leverages priors of both vertex and temporal domains and integrate the dynamics of graph topology into the formulation.
- We design a framework for experiments on time-varying graph signal recovery over dynamic graphs based on synthetic data.
- We conduct extensive experiments not only for validating the proposed method but for a comprehensive study on the performance of various regularizations terms and data.

In the following sections, we first cover the preliminaries of graph signal processing in Section II and then move on to the proposed method in Section III. In Section IV, we illustrate the experimental results both quantitively and visually and discuss the obtained results.

In this paper, we generalized the limited problem setting in our previous work [26] and formulated a new optimization problem for the signal recovery. Unlike in [26], we further conducted extensive experiments on synthetic data for comprehensive validation and detailed discussion. We also confirmed the validity on real-world data.

### II. PRELIMINARIES

#### A. Notation

Throughout the paper, scalars, vectors and matrices are denoted by normal (e.g. $\lambda$, $\epsilon$), lower-case bold (e.g. $x$, $u$), and uppercase bold (e.g. $\mathbf{L}$, $\mathbf{X}$) letters, respectively. The element on the $i$th row, $j$th column a matrix $\mathbf{X}$ is denoted by $X_{i,j}$.
The vectorization of a matrix $X \in \mathbb{R}^{n \times m}$ is denoted by $\text{vec}(X) = [x_1^T, x_2^T, \ldots, x_m^T]^T$, where $x_i$ is the $i$th column of matrix $X$.

### B. Graph Laplacian

Generally speaking, in graph signal processing, a weighted graph $G(V, E, W)$ is represented by a graph Laplacian $L \in \mathbb{R}^{n \times n}$, where $V$, $E$, and $W \in \mathbb{R}^{n \times n}$ denote the set of $n$ vertices, the set of edges, and the weight adjacency matrix, respectively. The combinatorial graph Laplacian of $G$ is defined by

$$L := D - W. \quad (1)$$

For the weight adjacency matrix $W$, $W_{i,j} > 0$ when vertices $i$ and $j$ are connected, and $W_{i,j} = 0$, otherwise. The degree matrix $D \in \mathbb{R}^{n \times n}$ is a diagonal matrix whose $i$th element is the degree of vertex $i$. A degree of a vertex is the sum of the weights of all the edges connected to the vertex. Note that all graph Laplacians of undirected graphs are real symmetric positive semidefinite matrices.

### C. Smooth Graph Signal Recovery

A graph signal $x \in \mathbb{R}^n$ is smooth on the vertex domain when signal values of two vertices connected by edges with large weights are similar. The smoothness of graph signals can quantitatively be measured by the graph Laplacian quadratic form:

$$x^T L x = \sum_{i,j \in E} W_{i,j} (x(i) - x(j))^2, \quad (2)$$

where the $i$th element of vector $x$ is denoted by $x(i)$. The smaller the value of the quadratic form, the smoother the graph signal $x$ is on the graph Laplacian $L$. To expand this to an $m$ series of graph signals $X \in \mathbb{R}^{n \times m}$, we define the signal smoothness by

$$\sum_{t=1}^{m} x_t^T L x_t = \text{Tr}(X^T LX), \quad (3)$$

where $\text{Tr}$ denotes the trace of a matrix.

In the literature of optimization-based signal recovery, the graph Laplacian quadratic form is minimized to leverage signal smoothness. Typically, the recovery of graph signal $X$ from an observation $\hat{X} = X + N$ ($N$ is some additive noise) can be expressed as

$$\min_{Y \in \mathbb{R}^{n \times m}} \text{Tr}(Y^T LY) + \alpha ||X - Y||_F^2, \quad (4)$$

where $|| \cdot ||_F$ denotes the Frobenius-norm of a matrix. The second regularization term is the quadratic data fidelity term and $\alpha$ is the balancing weight.

### D. Proximal Tools

The proximity operator of index $\gamma > 0$ of a proper lower semicontinuous convex function $f \in \Gamma_0(\mathbb{R}^n)$ is defined as

$$\text{prox}_{\gamma f} : \mathbb{R}^n \rightarrow \mathbb{R}^n : \gamma \rightarrow \arg \min_{y} f(y) + \frac{1}{2\gamma} ||y - x||^2, \quad (5)$$

The set of all proper lower semicontinuous convex functions on $\mathbb{R}^n$ is denoted by $\Gamma_0(\mathbb{R}^n)$.

The indicator function of a nonempty closed convex set $C$, denoted by $i_C$, is defined as

$$i_C(x) := \begin{cases} 0, & \text{if } x \in C, \\ \infty, & \text{otherwise}. \end{cases} \quad (6)$$

Since the function returns $\infty$ when the input vector is outside of $C$, it acts as the hard constraint represented by $C$ in minimization. The proximity operator of $i_C$ is the metric projection onto $C$, defined by

$$\text{prox}_{i_C}(x) = P_C(x) := \arg \min_{y \in C} ||y - x||^2. \quad (7)$$

### E. Primal-Dual Splitting Method

A primal-dual splitting method (PDS) [28], [29] can solve optimization problems in the form of

$$\min_{u} f_1(u) + f_2(u) + f_3(Au), \quad (8)$$

where $f_1$ is a differentiable convex function with the $\beta$-Lipschitzian gradient $\nabla f_1$ for some $\beta > 0$, proximity operators of $f_2$ and $f_3$ (which are proper lower semicontinuous convex functions) are efficiently computable (proximable), and $A$ is a matrix. The problem is solved by the algorithm:

$$u^{(i+1)} = \text{prox}_{\gamma_1 f_2} [u^{(i)} - \gamma_1 (\nabla f_1(u^{(i)}) + A^T v^{(i)})], \quad (9a)$$

$$v^{(i+1)} = \text{prox}_{\gamma_2 f_3} [v^{(i)} + \gamma_2 A(2u^{(i+1)} - u^{(i)})], \quad (9b)$$

where $f_3^*$ is the Fenchel-Rockafellar conjugate function of $f_3$ and the stepsizes $\gamma_1, \gamma_2 > 0$ satisfy $\frac{1}{\gamma_1} - \gamma_2 \lambda_1(A^T A) \geq \frac{\beta}{2}$ ($\lambda_1(\cdot)$ is the maximum eigenvalue of $\cdot$). The proximity operator of $f^*$ can be stated as the following:

$$\text{prox}_{\gamma f}(x) = x - \gamma \text{prox}_{\gamma^{-1} f}(\gamma^{-1} x). \quad (10)$$

The sequence $(u^{(k)})_{k \in \mathbb{N}}$ converges to a solution of (8) under some conditions on $f_2$, $f_3$, and $A$. PDS has played a central role in various signal recovery methods, e.g., [31]–[35]. A comprehensive review on PDS can be found in [36].

### III. Proposed Method

#### A. Problem Formulation

Consider the following graph signal observation model:

$$X = \Phi(\bar{X} + S + N), \quad (11)$$

where $X = [x_1, x_2, \ldots, x_p] \in \mathbb{R}^{n \times p}$ is the observation of an $n$ vertices $\times p$ time-slots time-varying graph signal, and $\bar{X} \in \mathbb{R}^{n \times p}$ is the true signal. For the noises, $S \in \mathbb{R}^{n \times p}$ and $N \in \mathbb{R}^{n \times p}$ are outliers and some random additive noise, respectively. Missing values are represented by a masking operator $\Phi$. Each graph signal $x_k$ is observed on a corresponding graph Laplacian $L_k$ at time $t = k$.

1 This algorithm is a generalization of the primal-dual hybrid gradient method [30]
We propose the following optimization problem to estimate the true graph signal in the above model:

$$\min_{Y \in \mathbb{R}^{n \times p}, S \in \mathbb{R}^{n \times p}} \sum_{k=1}^{p} (y_k^T L_k y_k) + \lambda R(D(Y))$$

subject to $$\|X - \Phi(Y + S)\|_F \leq \varepsilon, \|S\|_1 \leq \eta,$$

where $$Y = [y_1, y_2, \ldots, y_p] \in \mathbb{R}^{n \times p}$$ is the estimated time-varying graph signal, $$S \in \mathbb{R}^{n \times p}$$ is the matrix of estimated outliers, $$L_k \in \mathbb{R}^{n \times n}$$ is the graph Laplacian at time $$k$$, and $$\lambda > 0$$ is the temporal regularization parameter. The $$\ell_1$$-norm of a matrix is denoted by $$\|\cdot\|_1$$, and $$\varepsilon$$ and $$\eta$$ are the radius of the Frobenius and $$\ell_1$$-norm ball, respectively.

The first regularization term enforces the smoothness of the graph signal $$y_k$$ over $$L_k$$ by penalizing the graph Laplacian quadratic form. Unlike many conventional works [13, 37] that measure the smoothness of the time-varying graph signal on a single graph (refer Eq. 4), the proposed regularization measures the signal smoothness per time-slot. By doing so, we can integrate the dynamics of the vertex domain to the formulation.

The second term penalizes the temporal difference signal:

$$D(Y) = [y_2 - y_1, y_3 - y_2, \ldots, y_p - y_{p-1}],$$

where $$D$$ is a temporal difference operator, to leverage signal smoothness in the temporal domain. We consider $$R$$ to be either $$\|\cdot\|_1$$ or $$\|\cdot\|_2^2$$ depending on the nature of the signal of interest.

We impose data fidelity and noise sparsity as hard constraints in (12) to facilitate parameter tuning, as has been addressed, e.g., in [22]–[25].

The main philosophy behind our formulation is that the underlying graph is often available in many real-world graph signal recovery situations, in which case our method would perform advantageously compared to the conventional methods [14, 15] that consider graph learning and online estimations. Especially in physical sensing situations, (e.g. using a dynamic network of sensor loaded drones for physical sensing) the underlying dynamic graph topology can easily be generated per time-slot using the $$k$$-nearest neighbors algorithm and weights defined by the Gaussian kernel of spatial coordinates.

In cases where no dynamic graphs are available in any way (when $$L_1 = L_2 = \ldots = L_k$$), our formulation would be the equivalent of the following formulation:

$$\min_{Y \in \mathbb{R}^{n \times p}, S \in \mathbb{R}^{n \times p}} \mathbb{Tr}(Y^T LY) + \lambda R(D(Y))$$

subject to $$\|X - \Phi(Y + S)\|_F \leq \varepsilon, \|S\|_1 \leq \eta.$$ Note that this is still a novel formulation for time-varying graph signal recovery on static graphs in terms of temporal domain regularization and sparse noise estimation.

**B. Algorithm**

We use the primal-dual splitting method (PDS) [28] to solve (12). By vectorizing $$X$$ and using indicator function, $$t_{B_1^{(x)}}$$ and $$t_{B_1^{(s)}}$$ where:

$$B^{(x)}_1 = \{z \in \mathbb{R}^{np} \mid ||z - x||_2 \leq \varepsilon\},$$

$$B^{(s)}_1 = \{z \in \mathbb{R}^{np} \mid ||z||_1 \leq \eta\},$$

we can reformulate (12) as

$$\min_{\text{vec}(Y) \in \mathbb{R}^{np}, \text{vec}(S) \in \mathbb{R}^{np}} \sum_{k=1}^{p} ||\text{vec}(y_k^T L_k y_k) + \lambda R(D \text{vec}(Y))$$

$$+ t_{B_1^{(x)}}(\text{vec}(x), s) (\Phi(\text{vec}(Y) + \text{vec}(S))) + t_{B_1^{(s)}}(\text{vec}(S)).$$

The matrix $$D \in \mathbb{R}^{n(p-1) \times np}$$ and diagonal matrix $$\Phi \in \mathbb{R}^{np \times np}$$ are the temporal linear difference operator and a masking operator for vectorized variables, respectively.

By defining $$u := [\text{vec}(Y)^T \text{vec}(S)]^T$$, and $$v := [v_1, v_2]$$ ($$v_1 \in \mathbb{R}^{p-1}, v_2 \in \mathbb{R}^{np}$$), and $$f_1, f_2, f_3, A$$ as

$$f_1(u) := \sum_{k=1}^{p} ||y_k^T L_k y_k||,$$

$$f_2(u) := t_{B_1^{(x)}}(\text{vec}(S)),$$

$$f_3(v) := \lambda R(v_1) + t_{B_1^{(x)}}(v_2),$$

$$A := \begin{bmatrix} D & O \\ \Phi & \Phi \end{bmatrix},$$

since $$\sum_{k=1}^{p} ||y_k^T L_k y_k||$$ is differentiable and the gradient is Lipschitz continuous and $$t_{B_1^{(x)}}(\text{vec}(S))$$ and $$\lambda R(v_1) + t_{B_1^{(x)}}(v_2)$$ are proper lower semicontinuous convex functions, the problem in (17) is reduced to (8), which can be solved by PDS (refer to Algorithm 1). Note that although the algorithm is written in the vectorized form, the actual implementation is written in the matrix form.

The $$\nabla f_1$$ is given by

$$\nabla f_1(u) = 2[D_1 y_1^T L_2 y_2^T \ldots L_p y_p^T]^T.$$ (19)

The proximity operators of $$f_2$$ and $$f_3$$ can be computed by the proximity operators of $$t_{B_1^{(x)}}(\text{vec}(S))$$ and $$t_{B_1^{(x)}}(\text{vec}(S)).$$

The proximity operator of $$f_2$$ is given by

$$\text{prox}_{t_{B_1^{(x)}}(\text{vec}(S))}(z) = P_{t_{B_1^{(x)}}(\text{vec}(S))} = z - \lambda \text{prox}_{t_{B_1^{(x)}}}(z).$$

And when $$R$$ is $$\|\cdot\|_1$$, the proximity operator is equivalent to the soft-thresholding operation:

$$\text{prox}_{t_{B_1^{(x)}}(\text{vec}(S))}(z) = [S \text{Tr}(z, \gamma)]_j = \text{sgn}(z_j) \max\{0, |z_j| - \gamma\},$$

and when $$R$$ is $$\|\cdot\|_2^2$$, the proximity operator of is

$$\text{prox}_{t_{B_1^{(x)}}(\text{vec}(S))}(z) = \arg \min \|y\|_2^2 + \frac{1}{2\gamma} \|y - z\|_2^2$$

$$= \frac{z}{2\gamma + 1}.$$ (24)

The proximity operator of $$t_{B_1^{(x)}}(\text{vec}(S))$$ is given by

$$\text{prox}_{t_{B_1^{(x)}}(\text{vec}(S))}(z) = P_{t_{B_1^{(x)}}(\text{vec}(S))}$$

$$= \begin{cases} z, & \text{if } z \in B_1^{(x)} \\ x + \frac{1}{||z - x||_2^2} (z - x), & \text{otherwise}. \end{cases}$$ (26)
Algorithm 1 Algorithm for solving (12)

Input: Input signal $X$, graph Laplacian $L_k (k = 1, 2, ..., p)$
Output: Output signal $Y^{(i)}$

Initialization: $Y^{(0)} = X \in \mathbb{R}^{n \times p}, S^{(0)} = 0$

1: while A stopping criterion is not satisfied do
2: \quad $\text{vec}(Y^{(i+1)}) = \text{vec}(Y^{(i)}) - \gamma_1 (\nabla f_1 (\text{vec}(Y^{(i)}) + PD^T v_1^{(i)} + \Phi^T v_2^{(i)}))$
3: \quad $\text{vec}(S^{(i+1)}) = P_{B_1}(\text{vec}(S^{(i)}) - \gamma_2 \Phi^T v_2^{(i)}))$
4: \quad $v_1^{(i)} \leftarrow v_1^{(i)} + \gamma_2 D (2 \text{vec}(Y^{(i+1)}) - \text{vec}(Y^{(i)}))$
5: \quad $v_2^{(i)} \leftarrow v_2^{(i)} + \gamma_2 \Phi ((2 \text{vec}(Y^{(i+1)}) - \text{vec}(Y^{(i)})) + (2 \text{vec}(S^{(i+1)}) - \text{vec}(S^{(i)})))$
6: \quad $v_1^{(i+1)} = v_1^{(i)} - \gamma_2 \text{prox}_{\frac{1}{2}\sigma} R \left( \frac{1}{\gamma_2} v_1^{(i)} \right)$
7: \quad $v_2^{(i+1)} = v_2^{(i)} + \gamma_2 \Phi (\text{vec}(X_{i,p})) (\frac{1}{\gamma_2} v_2^{(i)})$
8: \quad $i \leftarrow i + 1$
9: end while
10: return $Y^{(i)}$

IV. EXPERIMENTAL RESULTS

A. Dataset

1) Synthetic Dataset: To test the methods (Table II) on a dynamic graph Laplacian setting, we constructed a synthetic dataset that simulates a network of sensor-loaded drones remotely sensing a smooth distribution. In a 2D plane, 64 or 128 vertices, which represent drones, are generated randomly from a uniform distribution. They observe signal values calculated by inputting their coordinates to the underlying smooth distribution $(0, 1)$, in this case, a combination of several multivariate normal distributions (Fig. 1 (a)). The vertices move across the 2D-plane at random degrees and a pre-set velocity $v$ for 100 or 200 time-slots to generate a time-varying graph signal $X \in \mathbb{R}^{64 \times 100}$ or $X \in \mathbb{R}^{128 \times 200}$ and a dynamic graph Laplacian $L \in \mathbb{R}^{64 \times 64 \times 100}$ or $L \in \mathbb{R}^{128 \times 128 \times 100}$. The graphs are constructed by the $k$-nearest neighbors algorithm ($k = 4$). The weights are decided by the Gaussian kernel of spatial coordinates of the corresponding two vertices. Then, $X$ is corrupted by adding $S$ and $N_\sigma$ and being masked by $\Phi$ to generate the observation $\tilde{X}$. Outlier $S$ is a matrix of impulsive noise that appears at each vertex at a probability of $P_s$, where each value of the noise is uniformly distributed in the interval $[-1, 1]$. Noise $N_\sigma$ is an additive white Gaussian noise of variance $\sigma^2$, and $\Phi$ masks the signals at a probability of $P_p$. The graph construction is implemented by using GSPBox [39]. We also constructed a piece-wise flat version of the dataset (Fig. 1 (b)) by rounding the signal values of Fig. 1 (a) to the nearest fifth of the signal range.

2) Real-world Dataset: We use the time-varying temperature data in the island of Hokkaido3, an island in the north of Japan. The dataset is comprised of 172 vertices and 365 time-slots of time-varying graph signals. We use either 100 or 200 time-slots of these time-slots for our experiments, and add noise to the signals in the same manner as we did in the synthetic dataset. Because the temperature data is observed on a static sensor network, an explicit dynamic graph is not available. Therefore, we generate a graph Laplacian from the true graph signal per time-slots beforehand. We generate the edges by the $k$-nearest neighbors algorithm ($k = 4$) and define the weights by the Gaussian kernel of signal values of the corresponding two vertices.

Note that the dynamic graph Laplacian that we are going to be using is not something that we will have access in practical situations. The main objective of the experiments using the real-world dataset is to confirm the theoretical advantages of integrating graph dynamics to the recovery method. The recovery performance of using an ideal dynamic graph Laplacian would give us some insight into the benefits of adopting dynamic graph learning methods [21] to static graphs before applying our recovery method.

B. Experimental Settings

In the experiments, we evaluate our method on synthetic and real-world datasets. Regarding the implementation for Algorithm 1, the stopping criterion is set to $\|Y^{(i)} - Y^{(i-1)}\| \leq 1.0 \times 10^{-4}$, $\|S^{(i)} - S^{(i-1)}\| \leq 1.0 \times 10^{-4}$. The regularization parameter $\lambda$ is tuned per method and setting to exhibit the best performance. The performance of the methods are measured in PSNR[dB]:

$$\text{PSNR} (Y, \tilde{X}) = 20 \log_{10} \frac{\text{MAX}_X}{\text{MAX}_Y}\|X - \tilde{X}\|_F^2, (\text{MAX}_X = 1).$$

We compare the proposed methods (Methods (G) ~ (J) in Table II) to other methods (Methods (A) ~ (F)). Note that the Methods (G) ~ (J) are the different implementations of the proposed formulation (12): (H) and (J) are the $\ell_1$ and Frobenius-norm versions, respectively, and (G) and (I) are their static counterparts. For the temporal domain, we use either $\|D (Y)\|_1$ or $\|D (Y)\|_F^2$, and for the vertex domain, we use $\text{Tr} (Y^TLY)$ or $\sum_{k=1}^P \sum_{y \in L_k} y_k y_k$. For $\text{Tr} (Y^TLY)$, where it uses only one graph Laplacian, we input $L_1$, the first graph constructed at $t = 1$.

Method (E) is a method proposed in [13], which enforces the smoothness of the temporal difference signal $D (Y)$ over the graph by penalizing $\text{Tr} (\langle (D (Y))^T L (D (Y)) \rangle)$. It leverages priors of both vertex and temporal domains but is based on a static graph model. Thus, to evaluate the effectiveness of the dynamic graph model, we also modify Method (E) by altering the regularization term to $\sum_{k=1}^{p-1} (y_{k+1} - y_k)^T L_k (y_{k+1} - y_k)$ (Method (F)). In Method (F), the smoothness term is considered separately for each time-slot, thus can leverage the dynamics of the time-varying graph Laplacian.

Note that although the original implementation of [13] enforces signal fidelity by an additive regularization, Method (E) and Method (F) are enforced by a fidelity constraint like the rest of the methods (A) ~ (J). All of the methods share the same constraints $\|X - \Phi (Y + S)\|_F \leq \varepsilon, \|S\|_1 \leq \eta$. This is to avoid parameter tuning in cases where the noise intensity is known a priori or can be estimated, and in our experiments, $\varepsilon$ and $\eta$ are set to:

$$\varepsilon = 0.9\sigma \sqrt{np (1 - P_s) (1 - P_p)}, \eta = \frac{P_p}{2} np.$$ (28)

All of the methods are implemented by PDS.

---

3The Japan Meteorological Agency provided the daily temperature data from their website at https://www.jma.go.jp/jma/index.html.
Fig. 1. (a): The smooth distribution constructed by a combination of several multivariate normal distributions. (b): A piece-wise flat version of (a).

Fig. 2. The temperature data of Hokkaido in 2015. Note the smoothness in the vertex domain and the seasonal changes in the temporal domain.

Fig. 3. The time signal of a random vertex in the 64 × 100 dataset: (a)∼(d) are of different velocity \( v \). The noise intensity is \( \sigma = 0.05, P_s = 0.05, P_p = 0.05 \). The red line indicates the signal recovered by Method (J). The signal gets less smooth as the velocity gets higher.

C. Results and Discussion

1) Dynamic vs Static Graph Laplacian: Looking at all experimental settings on all datasets, the formulations integrated with a time-varying graph Laplacian (Methods (D), (F), (H), (J)) performed better than their counterparts that only consider a static graph (Methods (C), (E), (G), (I)). Especially looking at Methods (C) and (D) in the synthetic data experiments, a pure comparison between static and dynamic graph-based regularization, the dynamic regularization is almost always better than the static counterpart by \( 3 \sim 6 \) [dB]. In fact, comparing Methods (G) and (I) to Methods (A) and (B), the static-based vertex domain regularization even seems to be hindering the overall recovery performance. Using \( L_1 \), an inaccurate representation of the graph at time \( t \), for every timeslot should clearly be discouraged.

Regarding the experiments on real-world data, the performance gap between dynamic and static graph-based methods indicates the untapped potential of combining dynamic graph learning with signal recovery.

2) Vertex vs Temporal Domain Priors: Comparing Methods (A) and (B) to Methods (C) and (D), a comparison between vertex and temporal domain regularizations, leveraging the temporal prior is much more effective than leveraging the vertex-domain prior for the synthetic dataset. This can also be observed from the relatively small performance gains between Methods (A), (B), and Methods (H), (J). In fact, in Table III, there is one setting where Method (A) performed better than Method (J). We speculate that this is because the low velocity and noise intensity make the graph signal extremely smooth in the temporal domain, whereas the vertex domain is relatively less smooth even in the noiseless case (Fig. 1 (a)). Therefore, the temporal domain prior is adequate to recover the signal and the vertex domain is unnecessary.
Fig. 4. The time signal of a random vertex in the 128 × 200 dataset: (a)−(d) are of different velocity \( v \). The noise intensity is \( \sigma = 0.05, P_s = 0.05, P_p = 0.05 \). The red line indicates the signal recovered by Method (J). The signal gets less smooth as the velocity gets higher.

Fig. 5. The time signal of a random vertex in the Hokkaido temperature dataset: (a) and (b) are of different noise intensity. The red line indicates the signal recovered by Method (J). Note how the signal is much less smooth in the temporal domain compared to the synthetic dataset.

| Method | Regularization | Vertex domain | Temporal domain |
|--------|----------------|---------------|-----------------|
| (A)    | -              | \( \| D(Y) \|_1 \) |                 |
| (B)    | -              | \( \| D(Y) \|_2 \) |                 |
| (C) [37] | \( \text{Tr}(Y^\top L Y) \) | - |                 |
| (D)    | \( \sum_{k=1}^p y_k^\top L_k y_k \) | - |                 |
| (E) [13] | \( \text{Tr}((D(Y))^\top L(D(Y))) \) | - |                 |
| (F)    | \( \sum_{k=1}^{p-1} (y_{k+1} - y_k)^\top L_k (y_{k+1} - y_k) \) | - |                 |
| (G)    | \( \text{Tr}(Y^\top L Y) \) | \( \| D(Y) \|_1 \) |                 |
| (H) Ours | \( \sum_{k=1}^p y_k^\top L_k y_k \) | \( \| D(Y) \|_1 \) |                 |
| (I)    | \( \text{Tr}(Y^\top L Y) \) | \( \| D(Y) \|_1 \) |                 |
| (J) Ours | \( \sum_{k=1}^p y_k^\top L_k y_k \) | \( \| D(Y) \|_1 \) |                 |

However, the vertex domain regularization performs much better than the temporal domain regularization in the real-world dataset. Comparing Fig. 3 and 4 to 5, the real-world data is clearly not as smooth in the temporal domain as in the synthetic data. On the other hand, comparing Fig. 6 and 7 to 8, the real-world data is much smoother in the vertex domain compared to the synthetic data. The reversal in the performance of the two regularizations is due to the difference in the nature of the datasets.

Considering how Method (J) performed better in all but one setting, we claim our method to be robust to the varying nature of datasets.

3) Sensor Velocity: From Tables III and IV, we can see how all the methods that leverage the temporal domain drop their performance as the velocity \( v \) of the sensors gets larger. This is because as the velocity gets higher, the signals get less smooth in the temporal domain (refer Fig. 3 and 4), and thus Methods (A) and (B) drop their performance, whereas Methods (C) and (D) remain unchanged. Likewise, the performance differences between Methods (H), (J) and (A), (B) get larger as the velocity gets higher.

In practical settings, the difference in sensor velocity is often equal to the difference in sampling frequency. A sensor that samples at a constant frequency moving in various velocities is the same as a sensor that moves at a constant velocity and samples at various frequencies (given that the underlying distribution is not time-variant). Our method can tolerate higher velocities compared to temporal domain-only Methods.
TABLE III
THE AVERAGE RECOVERY PERFORMANCE ON THE 64 × 100 SYNTHETIC DATASET MEASURED IN PSNR [dB]

| Method | 64 vertices × 100 time-slots | Piece-wise Flat |
|--------|-----------------------------|-----------------|
|        | Smooth                      |                 |
|        | $v = 0.25$                  | $v = 0.5$       | $v = 0.75$ | $v = 1.0$ | $v = 0.25$ | $v = 0.5$ | $v = 0.75$ | $v = 1.0$ |
| (A)    | 26.94                       | 25.03           | 23.83      | 22.82     | 26.88      | 24.69     | 23.44      | 22.28     |
| (B)    | **28.67**                   | 26.06           | 24.53      | 23.30     | 26.53      | 24.83     | 23.73      | 22.66     |
| (C) [37]| 18.16                      | 17.97           | 18.04      | 17.96     | 17.84      | 17.64     | 17.72      | 17.61     |
| (D)    | 22.48                       | 22.33           | 22.53      | 22.72     | 21.80      | 21.57     | 21.74      | 21.99     |
| (E) [13]| 25.19                      | 23.11           | 22.00      | 21.08     | 24.08      | 23.35     | 21.40      | 20.56     |
| (F)    | 25.16                       | 23.29           | 23.34      | 21.61     | 24.11      | 22.38     | 21.60      | 20.95     |
| (G)    | 27.04                       | 24.86           | 23.65      | 22.65     | 26.44      | 24.36     | 23.23      | 22.21     |
| (H) Ours | 27.38                      | 25.89           | 25.04      | 24.47     | **27.07**  | **25.31** | **24.46**  | 23.76     |
| (I)    | 27.43                       | 25.22           | 23.99      | 22.97     | 25.76      | 24.29     | 23.31      | 22.39     |
| (J) Ours | **28.64**                  | **26.55**       | **25.30**  | **24.58** | 26.52      | 25.15     | 24.33      | **23.77** |
| Observ. | 15.81                      | 15.89           | 15.91      | 15.94     | 15.76      | 15.84     | 15.87      | 15.90     |
|        |                             |                 | $\sigma = 0.1$, $P_\ell = 0.1$, $P_p = 0.1$ | $\sigma = 0.1$, $P_\ell = 0.1$, $P_p = 0.1$ |
| (A)    | 22.16                       | 20.36           | 19.51      | 18.72     | 21.70      | 19.93     | 19.07      | 18.27     |
| (B)    | 23.23                       | 21.13           | 20.04      | 19.16     | 22.39      | 20.50     | 19.49      | 18.68     |
| (C) [37]| 15.53                      | 15.42           | 15.40      | 15.41     | 15.15      | 15.01     | 15.00      | 15.02     |
| (D)    | 18.82                       | 18.86           | 18.90      | 19.02     | 18.30      | 18.27     | 18.34      | 18.48     |
| (E) [13]| 20.43                      | 18.85           | 18.03      | 17.42     | 19.89      | 18.34     | 17.54      | 16.96     |
| (F)    | 20.39                       | 19.18           | 18.42      | 17.90     | 19.92      | 18.62     | 17.90      | 17.39     |
| (G)    | 22.43                       | 20.36           | 19.50      | 18.79     | 21.64      | 19.85     | 19.03      | 18.35     |
| (H) Ours | 22.93                      | 21.71           | 21.09      | 20.74     | 22.36      | 21.07     | 20.46      | 20.15     |
| (I)    | 22.52                       | 20.64           | 19.75      | 19.04     | 21.71      | 20.05     | 19.23      | 18.57     |
| (J) Ours | **23.76**                  | **22.12**       | **21.31**  | **20.81** | **22.75**  | **21.31** | **20.60**  | **20.20** |
| Observ. | 12.50                      | 12.58           | 12.57      | 12.59     | 12.45      | 12.53     | 12.53      | 12.55     |

(A) and (B), in other words, can tolerate lower sampling frequencies. This means that leveraging both domains can realize a more cost-efficient recovery method by lowering the sampling frequency.

4) Number of Vertices: By comparing Methods (C) and (D) in the experiments on the 64 × 100 dataset and 128 × 200 dataset, we can see that the vertex domain regularization performs better with a larger number of vertices. This is because, in our synthetic setting, as the number of vertices gets larger, the sensor network becomes spatially denser. We generate the graph Laplacians using the $k$-nearest neighbors algorithm ($k = 4$) and Gaussian kernels of spatial coordinates, under the assumption that vertices that are spatially close to each other have similar signal values. Although graph Laplacians generated in such a manner are known to be good smooth representations of graphs in many situations, the accuracy of the representation can vary. In our case, a denser graph means that the four vertices that a vertex is connected to are spatially closer to each other, and thus more likely to have similar signal values. Therefore, the setting with more vertices meets the conditions of the assumption that the data is generated better.

5) Number of Time-slots: By comparing Methods (A) and (B) in the experiments on the synthetic and real-world datasets, we can see that the temporal domain regularization performs better with more time-slots. Adding this to the above-mentioned discussion on the number of vertices, our method benefits from datasets larger in both vertex and temporal domains.

6) Smooth and Piece-wise Flat Signals: For the methods that leverage the temporal prior, we expected methods that use the Frobenius-norm regularization to perform better on the smooth dataset, and the methods that use the $\ell_1$-norm regularization to perform better on the piece-wise flat dataset, considering how the $\ell_1$-norm regularization tends to work better on staircase parts of the data. However, for some cases with higher noise intensities and higher velocities, the Frobenius-norm regularization performed better even for the piece-wise flat dataset. Looking at how methods performed as expected in lower noise intensity and lower velocity settings, we speculate that higher noise intensity and higher velocity compromised the sparsity of the temporal difference signal, and therefore the smoothing ability of the Frobenius-norm regularization outweighed the pros of the $\ell_1$-norm regularization.

That aside, to enhance the performance of the proposed method on piece-wise flat data, we can consider adopting a $\ell_1$-norm based regularization for the vertex domain [4] as well, which works better for piece-wise flat data as it prevents over-smoothing the staircase parts of the data in the vertex domain.

7) Visual Analysis: Referring to the visual results of the experiments (Fig. 6, 7, 8), we can see how our methods can recover the graph signals fairly well. We should note that in noise intense settings (Fig. 7), the small area in the top middle of the graph with high signal values is being overly
TABLE IV
The average recovery performance on the 128 × 200 synthetic dataset measured in PSNR [dB]

| Method | 128 vertices ×200 time-slots | Piece-wise Flat |
|--------|-----------------------------|-----------------|
|        | Smooth                     |  Piece-wise Flat |
|        | \(v = 0.25\) | \(v = 0.5\) | \(v = 0.75\) | \(v = 1.0\) | \(v = 0.25\) | \(v = 0.5\) | \(v = 0.75\) | \(v = 1.0\) |
| \(\sigma = 0.05, \sigma_p = 0.05, \sigma_p = 0.05\) |
| (A)    | 27.46  | 25.40  | 24.07  | 22.96  | 27.54  | 24.96  | 23.49  | 22.43  |
| (B)    | 29.37  | 26.51  | 24.90  | 23.68  | 26.84  | 25.06  | 23.88  | 22.91  |
| (C)  [37] | 18.67  | 18.65  | 18.61  | 18.56  | 18.28  | 18.26  | 18.22  | 18.17  |
| (D)    | 24.78  | 24.73  | 24.75  | 24.63  | 23.66  | 23.62  | 23.61  | 23.52  |
| (E)  [13] | 27.20  | 24.79  | 23.39  | 22.28  | 25.58  | 23.71  | 22.53  | 21.57  |
| (F)    | 27.20  | 24.94  | 23.65  | 22.66  | 25.50  | 23.75  | 22.70  | 21.88  |
| (G)    | 27.55  | 25.22  | 24.04  | 22.94  | 27.05  | 24.75  | 23.39  | 22.37  |
| (H) Ours | 28.43  | 26.89  | 26.13  | 25.45  | \textbf{27.88} | \textbf{26.06} | \textbf{25.17} | \textbf{24.44} |
| (I)    | 27.89  | 25.62  | 24.32  | 23.28  | 26.05  | 24.52  | 23.46  | 22.60  |
| (J) Ours | \textbf{29.67} | \textbf{27.32} | \textbf{26.22} | \textbf{25.54} | 26.98  | 25.05  | 24.93  | 24.41  |
| Observ. | 15.82  | 15.80  | 15.81  | 15.79  | 15.78  | 15.74  | 15.75  | 15.74  |

| Method | 172 vertices ×100 time-slots | 172 vertices ×200 time-slots |
|--------|-----------------------------|-------------------------------|
|        | \(\sigma = 0.1\), \(\sigma_p = 0.1\), \(\sigma_p = 0.1\) |
| (A)    | 22.26  | 20.62  | 19.59  | 18.83  | 21.80  | 20.10  | 19.12  | 18.33  |
| (B)    | 23.35  | 21.37  | 20.16  | 19.31  | 22.51  | 20.68  | 19.56  | 18.77  |
| (C)  [37] | 16.09  | 16.06  | 16.00  | 15.93  | 15.65  | 15.63  | 15.57  | 15.52  |
| (D)    | 20.61  | 20.61  | 20.58  | 20.55  | 19.93  | 19.93  | 19.89  | 19.86  |
| (E)  [13] | 21.68  | 19.99  | 18.95  | 18.21  | 21.04  | 19.39  | 18.40  | 17.70  |
| (F)    | 21.77  | 20.18  | 19.18  | 18.55  | 21.09  | 19.57  | 18.61  | 18.02  |
| (G)    | 22.50  | 20.63  | 19.62  | 18.92  | 21.76  | 20.06  | 19.11  | 18.42  |
| (H) Ours | 23.54  | 22.48  | 21.91  | 21.46  | 22.86  | 21.73  | 21.15  | 20.70  |
| (I)    | 22.56  | 20.84  | 19.83  | 19.13  | 21.76  | 20.20  | 19.27  | 18.61  |
| (J) Ours | \textbf{24.22} | \textbf{22.70} | \textbf{21.94} | \textbf{21.47} | \textbf{23.15} | \textbf{21.79} | \textbf{21.13} | \textbf{20.70} |
| Observ. | 12.61  | 12.58  | 12.59  | 12.58  | 12.56  | 12.53  | 12.54  | 12.53  |

TABLE V
The average recovery performance on the Hokkaido temperature dataset measured in PSNR [dB]

smoothed. Putting aside the difficulty of the setting due to the noise intensity, the nature of the underlying distribution (Fig.1) is also making the problem harder to solve. The part of the distribution with high values appears as spike signals in both vertex and temporal domains. Comparing Fig. 6 and 7, denser graphs are easier to recover, as more vertices are part of the high signal value area, facilitating the signal smoothness in the vertex domain. For the same reason, lower signal velocity facilitates signal smoothness in the temporal domain, enhancing the performance of the methods that leverage the temporal domain prior.

V. CONCLUSION

In this paper, we proposed a novel time-varying graph signal recovery problem based on an explicitly dynamic graph model, a model that assumes the given underlying graph to be time-variant like the signals. We proposed a method to solve the problem that effectively leverages the priors of both the vertex and temporal domains, while also taking advantage of the dynamics of the graph. By estimating sparsely modeled
outliers jointly with the graph signals, we realize a recovery method robust to various types of noises. Through experiments on synthetic and real-world data, we compared our method to other methods based on regularization on the vertex and temporal domains and analyzed how the priors of the two domains contribute to the recovery performance. We believe that the proposed problem setting is realistic, especially in situations like remote sensing using mobile sensors where we can easily generate dynamic graph Laplacians that represent spatial correlations of the sensors. Dynamic sensor networks are very feasible and practical sensing methods, and we place our research as the first step in tackling the problem.

REFERENCES

[1] A. Sandryhaila and J. M. F. Moura, “Discrete signal processing on graphs,” IEEE Transactions on Signal Processing, vol. 61, no. 7, pp. 1644–1656, 2013.
[2] 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 Processing Magazine, vol. 30, no. 3, pp. 83–98, 2013.
[3] S. K. Narang and A. Ortega, “Compact support biorthogonal wavelet filterbanks for arbitrary undirected graphs,” IEEE Transactions on Signal Processing, vol. 61, no. 19, pp. 4673–4685, 2013.
[4] C. Couprie, L. Grady, L. Najman, J.-C. Pesquet, and H. Talbot, “Dual constrained TV-based regularization on graphs,” SIAM Journal on Imaging Sciences, vol. 6, no. 3, pp. 1246–1273, 2013.
[5] S. Ono, I. Yamada, and I. Kumazawa, “Total generalized variation for graph signals,” in 2015 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), 2015, pp. 5456–5460.
[6] P. Berger, G. Hannak, and G. Matz, “Graph signal recovery via primal-dual algorithms for total variation minimization,” IEEE Journal of Selected Topics in Signal Processing, vol. 11, no. 6, pp. 842–855, 2017.
[7] C. Dinesh, G. Cheung, and I. V. Bajić, “3d point cloud color denoising using convex graph-signal smoothness priors,” in 2019 IEEE 21st International Workshop on Multimedia Signal Processing (MMSP), 2019, pp. 1–6.
[8] S. Chen and Y. C. Eldar, “Graph signal denoising via unrolling networks,” in ICASSP 2021 - 2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), 2021, pp. 5290–5294.
[9] S. Chen, A. Sandryhaila, J. M. F. Moura, and J. Kovacevic, “Signal denoising on graphs via graph filtering,” in 2014 IEEE Global Conference on Signal and Information Processing (GlobalSIP), 2014, pp. 872–876.
[10] M. Onuki, S. Ono, M. Yamagishi, and Y. Tanaka, “Graph signal denoising via trilateral filter on graph spectral domain,” IEEE Transactions on Signal and Information Processing over Networks, vol. 2, no. 2, pp.

![Fig. 6. Graph signal recovery results of the synthetic 64 × 100 dataset (σ, P_s, P_p = 0.1, v = 0.5). Note that the color range for the observations are clipped to [0,1] for a better illustrative comparison. The actual observations are noisier than the figures show. Method (J) best recovers the graph signal. The area with high signal values is difficult to recover, since it compromises the signal smoothness in both vertex and temporal domains.](image-url)
Fig. 7. Graph signal recovery results of the synthetic $128 \times 200$ dataset $(\sigma, P_s, P_p = 0.1, \nu = 0.25)$. Note that the color range for the observations are clipped to [0,1] for a better illustrative comparison. The actual observations are noisier than the figures show. Method (J) best recovers the graph signal. It exhibits better recovery results compared to the $64 \times 100$ dataset since the graph is physically denser, which makes the graph generated by the $k$-nearest neighbor algorithm a better representation of the network.

137–148, 2016.

[11] M. Nagahama, K. Yamada, Y. Tanaka, S. H. Chan, and Y. C. Eldar, “Graph signal denoising using nested-structured deep algorithm unrolling,” in ICASSP 2021 - 2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), 2021, pp. 5280–5284.

[12] F. Grassi, A. Loukas, N. Perraudin, and B. Ricaud, “A time-vertex signal processing framework: Scalable processing and meaningful representations for time-series on graphs,” IEEE Transactions on Signal Processing, vol. 66, no. 3, pp. 817–829, 2018.

[13] K. Qiu, X. Mao, X. Shen, X. Wang, T. Li, and Y. Gu, “Time-varying graph signal reconstruction,” IEEE Journal of Selected Topics in Signal Processing, vol. 11, no. 6, pp. 870–883, 2017.

[14] S. Sardellitti, S. Barbarossa, and P. Di Lorenzo, “Online learning of time-varying signals and graphs,” in ICASSP 2021 - 2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), 2021, pp. 5230–5234.

[15] D. Romero, N. I. Vassilis, and G. B. Giannakis, “Kernel-based reconstruction of space-time functions on dynamic graphs,” IEEE Journal of Selected Topics in Signal Processing, vol. 11, no. 6, pp. 856–869, 2017.

[16] A. Sandryhaila and J. Moura, “Big data analysis with signal processing on graphs: Representation and processing of massive data sets with irregular structure,” IEEE Signal Processing Magazine, vol. 31, no. 5, pp. 80–90, 2014.

[17] N. Perraudin, A. Loukas, F. Grassi, and P. Vanderghynst, “Towards stationary time-vertex signal processing,” in 2017 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), 2017, pp. 3914–3918.

[18] A. Loukas and D. Foucard, “Frequency analysis of time-varying graph signals,” in 2016 IEEE Global Conference on Signal and Information Processing (GlobalSIP), 2016, pp. 346–350.

[19] H. E. Egilmez, E. Pavez, and A. Ortega, “Graph learning from data under laplacian and structural constraints,” IEEE Journal of Selected Topics in Signal Processing, vol. 11, no. 6, pp. 825–841, 2017.

[20] Z. Kang, H. Pan, S. C. H. Hoi, and Z. Xu, “Robust graph learning from noisy data,” IEEE Transactions on Cybernetics, vol. 50, no. 5, pp. 1833–1843, 2020.

[21] K. Yamada, Y. Tanaka, and A. Ortega, “Time-varying graph learning with constraints on graph temporal variation,” arXiv preprint arXiv:2001.03346, 2020.

[22] M. Afonso, J. Bioucas-Dias, and M. Figueiredo, “An augmented Lagrangian approach to the constrained optimization formulation of imaging inverse problems,” IEEE Transactions on Image Processing, vol. 20, no. 3, pp. 681–695, 2011.

[23] G. Chierchia, N. Pustelnik, J.-C. Pesquet, and B. Pesquet-Popescu, “Epigraphical projection and proximal tools for solving constrained convex optimization problems,” Signal, Image and Video Processing, vol. 9, no. 8, pp. 1737–1749, 2015.

[24] S. Ono and I. Yamada, “Signal recovery with certain involved convex data-fidelity constraints,” IEEE Transactions on Signal Processing, vol. 63, no. 22, pp. 6149–6163, 2015.

[25] S. Ono, “$L_0$ gradient projection,” IEEE Transactions on Image Processing, vol. 26, no. 4, p. 1554–1564, Apr. 2017. [Online]. Available: https://doi.org/10.1109/TIP.2017.2651392
Real data

$\text{n} = 172$

PSNR: 11.76

Obsevation

PSNR: 22.85

Method (A)

PSNR: 23.10

Method (B)

PSNR: 27.45

Method (C) [37]

PSNR: 27.84

Method (D)

PSNR: 26.34

Method (E) [13]

PSNR: 26.27

Method (F)

PSNR: 26.86

Method (G)

PSNR: 27.23

Method (H) Ours

PSNR: 27.51

Method (I)

PSNR: 27.89

Method (J) Ours

Fig. 8. Graph signal recovery results of the Hokkaido temperature data on May 5, 2015 ($\sigma, P_s, P_p = 0.1$). Note that the color range for the observations are clipped to [0,1] for a better illustrative comparison. The actual observations are noisier than the figures show. Note how the graph signal is extremely smooth in the vertex domain, causing the Methods (C) and (D) to perform better than the Methods (A) and (B).

[26] E. Yamagata and S. Ono, “Recovery of time series of graph signals over dynamic topology,” in APSIPA Annual Summit and Conference (APSIPA ASC), Nov. 2021. (invited paper).

[27] J.-J. Moreau, “Dual convex functions and proximal points in a hilbert space,” CRAS, Paris, vol. 255, pp. 2897–2899, 1962.

[28] L. Condat, “A primal–dual splitting method for convex optimization involving lipschitzian, proximable and linear composite terms,” Journal of Optimization Theory and Applications, vol. 158, 08 2013.

[29] B. C. Vu, “A splitting algorithm for dual monotone inclusions involving cocoercive operators,” Advances in Computational Mathematics, vol. 38, no. 3, pp. 667–681, 2013.

[30] A. Chambolle and T. Pock, “A first-order primal-dual algorithm for convex problems with applications to imaging,” Journal of mathematical imaging and vision, vol. 40, no. 1, pp. 120–145, 2011.

[31] L. Condat, “A generic proximal algorithm for convex optimization—application to total variation minimization,” IEEE Signal Processing Letters, vol. 21, no. 8, pp. 985–989, 2014.

[32] S. Ono and I. Yamada, “Hierarchical convex optimization with primal-dual splitting,” IEEE Transactions on Signal Processing, vol. 63, no. 2, pp. 373–388, 2014.

[33] S. Ono, “Primal-dual plug-and-play image restoration,” IEEE Signal Processing Letters, vol. 24, no. 8, pp. 1108–1112, 2017.

[34] J. Boulanger, N. Pustelnik, L. Condat, L. Sengmanivong, and T. Piolot, “Nonsmooth convex optimization for structured illumination microscopy image reconstruction,” Inverse Problems, vol. 34, no. 9, p. 095004, 2018.

[35] S. Kyochi, S. Ono, and I. W. Selesnick, “Epigraphical relaxation for minimizing layered mixed norms,” IEEE Transactions on Signal Processing, vol. 69, pp. 2923–2938, 2021.

[36] N. Komodakis and J.-C. Pesquet, “Playing with duality: An overview of recent primal-dual approaches for solving large-scale optimization problems,” IEEE Signal Processing Magazine, vol. 32, no. 6, pp. 31–54, 2015.

[37] X. Dong, D. Thanou, P. Frossard, and P. Vandergheynst, “Learning laplacian matrix in smooth graph signal representations,” IEEE Transactions on Signal Processing, vol. 64, no. 23, pp. 6160–6173, 2016.

[38] L. Condat, “Fast projection onto the simplex and the $\ell_1$ ball,” Mathematical Programming, vol. 158, p. 575–585, 2016.

[39] N. Perraudin, J. Paratte, D. Shuman, L. Martin, V. Kalofolias, P. Vandergheynst, and D. K. Hammond, “GSPBOX: A toolbox for signal processing on graphs,” ArXiv e-prints, 2014.
Eisuke Yamagata (S’21) received a B.E. degree in Computer Science in 2020 from the Tokyo Institute of Technology. He is currently pursuing an M.E. degree at the Department of Computer Science in the Tokyo Institute of Technology. His current research interests are in graph signal processing and time series analysis.

Shunsuke Ono (S’11–M’15) received a B.E. degree in Computer Science in 2010 and M.E. and Ph.D. degrees in communications and computer engineering in 2012 and 2014 from the Tokyo Institute of Technology, respectively. From April 2012 to September 2014, he was a Research Fellow (DC1) of the Japan Society for the Promotion of Science (JSPS). He is currently an Associate Professor in the Department of Computer Science, School of Computing, Tokyo Institute of Technology. From October 2016 to March 2020, he was a Researcher of Precursory Research for Embryonic Science and Technology (PRESTO), Japan Science and Technology Corporation (JST), Tokyo, Japan. His research interests include signal processing, computational imaging, hyperspectral imaging and fusion, mathematical optimization, and data science. Dr. Ono received the Young Researchers’ Award and the Excellent Paper Award from the IEICE in 2013 and 2014, respectively, the Outstanding Student Journal Paper Award and the Young Author Best Paper Award from the IEEE SPS Japan Chapter in 2014 and 2020, respectively, and the Funai Research Award from the Funai Foundation in 2017. He has been an Associate Editor of IEEE TRANSACTIONS ON SIGNAL AND INFORMATION PROCESSING OVER NETWORKS since 2019.