Published in final edited form as:
IEEE Signal Process Mag. 2020 November ; 37(6): 160–173. doi:10.1109/MSP.2020.3013555.

Multiway Graph Signal Processing on Tensors:

Integrative analysis of irregular geometries

Jay S. Stanley III,
Dept. of Mathematics, Yale University, New Haven.

Eric C. Chi,
Dept. of Statistics, NC State University, Raleigh, NC.

Gal Mishne
Halıcıoğlu Data Science Institute, UC San Diego, La Jolla, CA.

Abstract

Graph signal processing (GSP) is an important methodology for studying data residing on irregular structures. As acquired data is increasingly taking the form of multi-way tensors, new signal processing tools are needed to maximally utilize the multi-way structure within the data. In this paper, we review modern signal processing frameworks generalizing GSP to multi-way data, starting from graph signals coupled to familiar regular axes such as time in sensor networks, and then extending to general graphs across all tensor modes. This widely applicable paradigm motivates reformulating and improving upon classical problems and approaches to creatively address the challenges in tensor-based data. We synthesize common themes arising from current efforts to combine GSP with tensor analysis and highlight future directions in extending GSP to the multi-way paradigm.

1. Introduction

Over the past decade, graph signal processing (GSP) [1] has laid the foundation for generalizing classical Fourier theory as defined on a regular grid, such as time, to handle signals on irregular structures, such as networks. GSP, however, is currently limited to single-way analysis: graph signals are processed independently of one another, thus ignoring the geometry between multiple graph signals. In the coming decade, generalizing GSP to handle multi-way data, represented by multidimensional arrays or *tensors*, with graphs underlying each axis of the data will be essential for modern signal processing. This survey discusses the burgeoning family of *multi-way graph signal processing* (MWGSP) methods for analyzing data tensors as a dependent collection of axes.

To introduce the concept of *way*, consider a network of N sensors each measuring a signal sampled at T time points. On the one hand, classic signal processing treats these signals as a collection of N independent 1D time-series ignoring the relation structure of the graph. On the other hand, the standard GSP perspective treats the data as a collection of T independent

jay.stanley@yale.edu.

1D graph signals that describe the state of all sensors for a given time point $t_j \in T$. Both are single-way perspectives that ignore the underlying geometry of the other way (also referred to as *mode*). The recent time-vertex (T-V) framework [2, 3] unifies these perspectives to form a dual-way framework that processes graph signals that are time-varying[1], thus bridging the gap between classical signal processing and GSP. While one of the axes of a T-V signal is a regular grid, time, in general a regular geometry may not underlie any of the ways of the data, e.g. genes and cells in sequencing data or users and items in recommendation systems [4–6]. Thus, the T-V framework is a subset of a more general MWGSP framework that considers the coupling of multiple geometries, whether predefined temporal or spatial axes, or irregular graph-based axes. MWGSP is by definition more versatile and is our main focus.

Classical signal processing and GSP typically process one or two-dimensional signals [1–3, 7] and do not address datasets of higher dimensions. However, such datasets, given as multi-way tensors, are becoming increasingly common in many domains. Mathematically, tensors generalize matrices to higher dimensions [8], and in this work the term tensors includes matrices (as they are 2-tensors). Examples of tensors includes video, hyperspectral imaging, MRI scans, multi-subject fMRI data, chemometrics, epigenetics, trial-based neural data, and higher-order *sparse* tensor data such as databases of crime incident reports, taxi rides or ad click data [9–14]. While tensors are the primary structure for representing D-dimensional signals, research on tensors and signal processing on tensors has primarily focused on factorization methods [8, 15], devoting less attention to leveraging the underlying geometry on the tensor modes. Recent MWGSP approaches incorporate graph smoothness in multiway tensor analysis, both for robust tensor factorization [12–14] and direct data analysis of tensors [9, 10].

In this overview of multi-way data analysis, we present a broad viewpoint to simultaneously consider general graphs underlying all modes of a tensor. Thus, we interpret multi-way analyses in light of graph-based signal processing to consider tensors as *multi-way graph signals* defined on multi-way graphs. GSP is a powerful framework in the multi-way setting, leading to intuitive and uniform interpretations of operations on irregular geometry.

Thus, MWGSP is a non-trivial departure from classical signal processing, producing an opportunity to exploit joint structures and correlations across modes to more accurately model and process signals in real-world applications of current societal importance: climate, spread of epidemics and traffic, as well as complex systems in biology.

Both the GSP and tensor analysis communities have been developing methods for multiway data analysis and have taken different but complementary strategies to solving common problems. We lay the mathematical and theoretical foundations drawing on work from both communities to develop a framework for higher-order signal processing of tensor data, and explore the challenges and algorithms that result when one imposes relational structure along all axes of data tensors. At the heart of this framework is the graph Laplacian, which provides a basis for harmonic analysis of data in MWGSP and an important regularizer in

[1]Note that the graph itself is static while the signals are time-varying.

modeling and recovery of multi-way graph signals. We illustrate the breadth of MWGSP by reinterpreting classic techniques, such as the 2-D discrete Fourier transform, as a special case of MWGSP and introduce a general Multi-Way Graph Fourier Transform (MWGFT). Further, we review novel multi-way regularizations that are not immediately obvious by viewing the data purely as a tensor. Thus, we synthesize into a coherent family a spectrum of recent and novel MWGSP methods across varied applications in inpainting, denoising, data completion, factor analysis, dictionary learning, and graph learning [10, 11, 4, 16–21].

The organization of this paper is as follows. Sec. II reviews standard GSP, which we refer to as single-way GSP. Sec. III introduces tensors and multilinear operators and constructs multi-way graphs, transforms, and filters. Sec. IV briefly highlights two recent multiway frameworks: the time-vertex framework, a natural development of MWGSP that couples a known time axis to a graph topology, and the Generalized Graph Signal Processing framework which extends MWGSP by coupling non-discrete and arbitrary geometries into a single signal processing framework. Sec. V moves to multi-way signal modeling and recovery, where graph-based multi-way methods are used in a broad range of tasks. Sec. VI concludes with open questions for future work.

II. Single-way GSP

GSP generalizes classical signal processing from *regular* Euclidean geometries such as time and space, to irregular, and non-Euclidean geometries represented discretely by a graph. In this section, we review basic concepts.[2]

a) Graphs:

This tutorial considers undirected, connected, and weighted graphs $\mathscr{G} = \{\mathscr{V}, \mathscr{E}, \mathbf{W}\}$ consisting of a finite vertex set $\mathscr{V}$, an edge set $\mathscr{E}$, and a weighted adjacency matrix $\mathbf{W}$. If two vertices v_i, v_j are connected, then $(v_i, v_j) \in \mathscr{E}$, and $\mathbf{W}_{i,j} = \mathbf{W}_{j,i} > 0$; otherwise $\mathbf{W}_{i,j} = \mathbf{W}_{j,i} = 0$. We employ a superscript parenthetical index to reference graphs and their accompanying characteristics from a set of graphs $\mathscr{G}^{(i)}$, i.e., $G = \left\{\mathscr{G}^{(i)} = \left(\mathscr{V}^{(i)}, \mathscr{E}^{(i)}, \mathbf{W}^{(i)}\right)\right\}_{i=1}^{D}$. Contextually we will refer to the cardinality of the vertex set of a graph $\mathscr{G}^{(i)}$ as $\left|\mathscr{V}^{(i)}\right| = n_i$. When parenthetical indexing is not used, we refer to a general graph $\mathscr{G}$ on n nodes. For details on how to construct a graph see the box “Graph construction.”

b) Graph Signals:

A signal $f : \mathscr{V} \to \mathbb{R}^n$ on the vertices of a graph on n nodes may be represented as a vector $f \in \mathbb{R}^n$, where $\mathrm{f}_i = f(i)$ is the signal value at vertex $v_i \in \mathscr{V}$.

The graph Fourier transform decomposes a graph signal in terms of the eigenvectors of a graph shift operator. Many choices have been proposed for graph shifts, including the adjacency matrix $\mathbf{W}$ and various forms of the graph Laplacian $\mathscr{L}$, a second order difference operator over the edge set of the graph. In this paper we use the popular combinatorial graph

[2]A complete survey of graph signal processing is provided in [1].

Laplacian defined as $\mathscr{L} := \mathbf{D} - \mathbf{W}$, where the degree matrix D is diagonal with elements $\mathbf{D}_{ii} = \sum_j \mathbf{W}_{ij}$.

This matrix is real and symmetric. Its eigendecomposition is $\mathscr{L} = \Psi \Lambda_{\mathscr{G}} \Psi^*$, where the columns of Ψ are a complete set of orthonormal eigenvectors $\{\psi_\ell\}_{\ell=0}^{n-1}$, Ψ^* is the conjugate transpose of Ψ, and the diagonal of $\Lambda_{\mathscr{G}}$ are the real eigenvalues $\{\lambda_\ell\}_{\ell=0}^{n-1}$.

c) Graph Fourier Analysis:

The Graph Fourier Transform (GFT) and its inverse are

$$\hat{f}(\lambda_\ell) := \langle \mathrm{f}, \psi_\ell \rangle = \sum_{k=1}^{N} f(k)\psi_\ell^*(k) \quad \text{and} \quad f(k) = \sum_{\ell=0}^{N-1} \hat{f}(\lambda_\ell)\psi_\ell(k), \tag{1}$$

or in matrix form $\mathrm{GFT}\{\mathrm{f}\} = \Psi^* \mathrm{f}$. The GFT generalizes the classical Fourier transform since the former is the spectral expansion of a vector in the discrete graph Laplacian eigensystem while the latter is the spectral expansion of a function in the eigensystem of the continuous Laplacian operator. Indeed, the GFT is synonymous with the discrete Fourier transform (DFT) when the graph Laplacian is built on a cyclic path or ring graph. It is typical to reinforce the classical Fourier analogy by referring to the eigenvectors of $\mathscr{L}$ as graph harmonics and the eigenvalues as graph frequencies and indexing the harmonics in ascending order of the eigenvalues such that the lowest indexed harmonics are the smoothest elements of the graph eigenbasis.

Despite these analogies, it is non-trivial to directly extend classical tools to signals on graphs. For example, there is no straightforward analogue of convolution in the time domain to convolution in the vertex domain. Instead, filtering signals in the GFT domain is defined analogously to filtering in the frequency domain, with a filtering function $\hat{h}(\cdot)$ applied to the eigenvalues λ_ℓ that take the place of the frequencies:

$$\tilde{f}(k) = \sum_{\ell=0}^{N-1} \hat{h}(\lambda_\ell)\hat{f}(\lambda_\ell)\psi_\ell(k), \tag{2}$$

where $\tilde{f}$ is the result of filtering f with the graph spectral filter $h(\mathscr{L})$. This spectral analogy is a common approach for generalizing classical notions that lack clear vertex interpretations.

III. Extending GSP to multi-way spaces

Classical D-dimensional Fourier analysis provides a template for constructing unified geometries from various data sources. The D-dimensional Fourier transform applies a 1-dimensional Fourier transform to each axis of the data sequentially. For example, a 2D-DFT applied to an $n_1 \times n_2$ real image X is

$$\text{2D-DFT}\{\mathrm{X}\} = \mathrm{DFT}_c(\mathrm{DFT}_r(\mathrm{X})) = \mathrm{DFT}_r(\mathrm{DFT}_c(\mathrm{X})) = \mathrm{U}_{n_1}\mathrm{X}\mathrm{U}_{n_2}, \tag{3}$$

where DFT_r (DFT_c) applies the DFT to the rows (columns) of X and U_n denotes a normalized n-point DFT matrix: $\mathrm{U}_n(t, k) = \frac{1}{\sqrt{n}}\exp\{-2\pi \mathrm{j} t(k-1)/n\}$ for t, $k = 1, \ldots, n$. This 2D transform decomposes the input into a set of plane waves.

The 2D graph Fourier transform (2D-GFT) is algebraically analogous to the 2D-DFT. For two graphs $\mathscr{G}^{(1)}$ and $\mathscr{G}^{(2)}$ on n_1 and n_2 vertices, the 2D-DFT of $X \in \mathbb{R}^{n_1 \times n_2}$ is

$$\text{2D-GFT}(\mathrm{X}) = \mathrm{GFT}_{n_1}\big(\mathrm{GFT}_{n_2}(\mathrm{X})\big) = \mathrm{GFT}_{n_2}\big(\mathrm{GFT}_{n_1}(\mathrm{X})\big), \tag{4}$$

and was presented in [7] as a method for efficiently processing big-data. Note that when $\mathscr{G}^{(1)} = \mathscr{P}^{n_1}$ and $\mathscr{G}^{(2)} = \mathscr{P}^{n_2}$, i.e., they are *cyclic path graphs* on n_1 and n_2 vertices, this transform is equivalent to a 2D-DFT [7].

In this section, we present the MWGSP framework for general D-dimensional signal processing on coupled and irregular domains, which enables holistic data analysis by considering relational structures on potentially *all* modes of a mutli-way signal. MWGSP encompasses standard GSP while extending fundamental GSP tools such as graph filters to D-dimensions. Furthermore, because graphs can be used to model discrete structures from classical signal processing, MWGSP forms an intuitive superset of discrete signal processing on domains such as images or video.

A. Tensors

Tensors are both a data structure representing D-dimensional signals, as well as a mathematical tool for analyzing multilinear spaces. We use both perspectives to formulate MWGSP. In this paper, we adopt the tensor terminology and notation used by [8].

1) Tensors as a D-dimensional array: The number of ways or modes of a tensor is its *order*. Vectors are tensors of order one and denoted by boldface lowercase letters, e.g., a. Matrices are tensors of order two and denoted by boldface capital letters, e.g., A. Tensors of higher-order, namely order three and greater, we denote by boldface Euler script letters, e.g., $\mathscr{A}$. If $\mathscr{A}$ is a D-way data array of size $n_1 \times \cdots \times n_D$, we say $\mathscr{A}$ is a tensor of order D.

There are multiple operations to reshape tensors, used for convenient calculations. *Vectorization* maps the elements of a matrix into a vector in column-major order. That is, for $\mathrm{X} \in \mathbb{R}^{n_1 \times n_2}$,

$$\mathrm{vec}(\mathrm{X}) = \left[\mathrm{X}_{1,1}, \ldots, \mathrm{X}_{n_1,1}, \mathrm{X}_{12}, \ldots, \mathrm{X}_{n_1,2}, \ldots, \mathrm{X}_{1,n_2}, \ldots, \mathrm{X}_{n_1,n_2}\right]^{\mathrm{T}}.$$

A tensor mode-d vectorization operator, $\mathrm{vec}_d(X)$ is similarly defined by stacking the elements of X in mode-d major order. Let $\mathrm{ten}(\mathrm{x}, \ell, \{n_1, \ldots, n_D\}) = X$ be the ℓth *tensorization* of x, which is the inverse of the ℓmajor vectorization of X. Denote by $n_{\backslash \ell} = \prod_{i=1}^{\ell-1} n_i \prod_{j=\ell+1}^{D} n_j$ the product of all factor sizes except for the ℓth factor. Then, let $\mathrm{mat}(X, \ell) = \mathrm{X}^{(\ell)} \in \mathbb{R}^{n_\ell \times n_{\backslash \ell}}$ be the mode-ℓmatricization of X formed by setting the ℓth mode of X to the rows of $\mathrm{X}^{(\ell)}$, vectorizing the remaining modes to form the columns of $\mathrm{X}^{(\ell)}$ as in Fig. 1.

2) Tensor products: Up to this point we have avoided explicitly constructing D-dimensional transforms. In the 2D case, applying a two-dimensional transform is calculated via linear operators as in (3); generalizing to higher-order tensors requires multilinear operators. Therefore, we introduce the *tensor product* and its discrete form, the *Kronecker product*. These products are powerful tools for succinctly describing D-dimensional transforms.

The great utility of the tensor product is that it simultaneously transforms spaces alongside their linear operators. This is the so-called *universal property* of the tensor product. In brief, it states that the tensor product, denoted by ⊗, of two vector spaces V and W is the unique result of a bilinear map $\varphi: V \times W \rightarrow V \otimes W$. The power in φ is that it uniquely factors any bilinear map on $V \times W$ into a linear map on $V \otimes W$. The universal property implies that the tensor product is symmetric: $V \otimes W$ is a canonical isomorphism of $W \otimes V$. Though the tensor product is defined in terms of two vector spaces, it can be applied repeatedly to combine many domains, so we generically refer to it as a product of many spaces.

In this paper, we are concerned with the tensor product on Hilbert spaces $\mathscr{H}^{(k)}, k = 1, \ldots, D$. These metric spaces include both continuous and discrete Euclidean domains from classical signal processing, as well as the non-Euclidean vertex domain. Since tensor products on Hilbert spaces produce Hilbert spaces, we can combine time, space, vertex, or other signal processing domains via the tensor product and remain in a Hilbert space. Under some constraints, an orthonormal basis for the product of D Hilbert spaces is admitted directly by the tensor product of the factor spaces. These properties of the tensor product are the mathematical foundations for the remainder of this tutorial, in which we construct a multi-way signal processing framework based on unifying multiple input spaces and their Fourier operators into a single linear representation.

Kronecker products: The Kronecker product produces the matrix of a tensor product with respect to a standard basis and generalizes the outer product of vectors $\mathrm{x}\mathrm{y}^*$ for $\mathrm{x} \in \mathbb{C}^m$ and $\mathrm{y} \in \mathbb{C}^n$. For analogy, it is common to use the same notation to denote the Kronecker and tensor product.

The Kronecker product is associative. Consequently the matrix M that is the Kronecker product of a sequence of D matrices $\mathrm{M}^{(k)} \in \mathbb{C}^{n_k \times n_k}$ for $k = 1, \ldots, D$ is

$$\mathrm{M} = \bigotimes_{k=1}^{D} \mathrm{M}^{(k)} = \mathrm{M}^{(1)} \otimes \left(\bigotimes_{k=2}^{D} \mathrm{M}^{(k)} \right) = \left(\bigotimes_{k=1}^{D-1} \mathrm{M}^{(k)} \right) \otimes \mathrm{M}^{(D)} = \mathrm{M}^{(1)} \otimes \cdots \otimes \mathrm{M}^{(D)}. \tag{5}$$

It is important to note that the Kronecker product is in general non-commutative. For brevity, we will apply a decremental Kronecker product using the notation

$$\downarrow \bigotimes_{k=1}^{D} \mathrm{M}^{(k)} = \bigotimes_{k=0}^{D-1} \mathrm{M}^{(D-k)} = \mathrm{M}^{(D)} \otimes \cdots \otimes \mathrm{M}^{(1)}.$$

The Kronecker product has many convenient algebraic properties for computing multidimensional transforms.

Vectorization enables one to express bilinear matrix multiplication as a linear transformation

$$\mathrm{vec}(\mathrm{CXB}) = \left(\mathrm{B}^{\top} \otimes \mathrm{C}\right)\mathrm{vec}(\mathrm{X}), \tag{6}$$

assuming that the dimensions of C, X, B are compatible such that CXB is a valid operation. This identity is a discrete realization of the universal property of tensors, and shows that the Kronecker product corresponds to a bilinear operator. We will use this identity to 1) construct multi-dimensional discrete Fourier bases, and 2) decompose multi-way algorithms for computational efficiency.

B. Multi-way transforms and filters

We now apply (6) to explicitly construct a 2D-GFT. If $\Psi^{(1)}$ and $\Psi^{(2)}$ are Fourier bases for graph signals on any two graphs $\mathcal{G}_1$ and $\mathcal{G}_2$, a 2D-GFT basis is $\Psi^{(2)} \otimes \Psi^{(1)}$. This is a single orthonormal basis of dimension $\left|\mathcal{V}^{(1)}\right|\left|\mathcal{V}^{(2)}\right| \times \left|\mathcal{V}^{(1)}\right|\left|\mathcal{V}^{(2)}\right|$, which can be used to describe a 2D graph signal $\mathrm{X} \in \mathbb{R}^{n_1 \times n_2}$ in the geometry of a single multi-way graph by the GFT

$$\hat{\mathrm{x}} = \left(\Psi^{(2)} \otimes \Psi^{(1)}\right)^{*}\mathrm{vec}(\mathrm{X}).$$

Unlike the DFT, where it is clear that increasing dimension yields grids, cubes, and hypercubes, interpreting the geometry of $\Psi^{(2)} \otimes \Psi^{(1)}$ is less intuitive. For this, we must turn to a *graph product*.

Product graphs: MWGSP relies on a graph underlying each mode of the given tensor data. The question is: What joint geometry arises from these graphs, and what multilinear operators exist on this joint graph structure? Our approach is to construct a multiway graph $\mathcal{G} = \{\mathcal{V}, \mathcal{E}, \mathbf{W}\}$ over the entirety of a data X as the *product graph* of a set of *factor* graphs $G = \left\{\mathcal{G}^{(1)}, \ldots, \mathcal{G}^{(D)}\right\}$.

For example, if $X \in \mathbb{R}^{n_1 \times n_2 \times n_3}$ contains the results of a n_3 sample longitudinal survey of n_2 genes on a cohort of n_1 patients, then the intramodal relationships of X are modeled by separate graphs $\mathcal{G}^{(1)}$, in which each patient is a vertex, $\mathcal{G}^{(2)}$, in which each gene is a vertex, and $\mathcal{G}^{(3)}$, which represents time as a path graph on n_3 vertices. We will use this example throughout this section, though our derivation generalizes to tensors of arbitrary order.

While one could treat matrix-valued slices of X as signals on each individual graph, we use the graph product to model X as a single graph signal on $\mathcal{G}$. We begin by constructing $\mathcal{V}$, the vertices of $\mathcal{G}$, which for all graph products is performed by assigning a single vertex to every element in the Cartesian product of the factor vertex sets i.e., $\mathcal{V} = \mathcal{V}^{(1)} \times \cdots \times \mathcal{V}^{(D)}$. Thus, the cardinality of the vertex set of $\mathcal{G}$ is $n = \prod_{k=1}^{D} n_k$. For example, our longitudinal survey will be modeled by the product graph $\mathcal{G}$ on $n = n_1 n_2 n_3$ vertices. As a Cartesian product, the elements $v \in \mathcal{V}$ can be expressed as the tuple $v = (\textit{patient}, \textit{gene}, \textit{time})$. The experimental observation tensor can be modeled as a graph signal[3] $\mathrm{x} = \mathrm{vec}(X)$ in $\mathbb{R}^n$.

Our next step is to learn the topology of $\mathcal{G}$ by mapping the edge sets (weights) of the factor graphs into a single set of product edges (weights) $\mathcal{E}$. There are a variety of graph products, each of which differs from each other only in the construction of this map. We focus on the *Cartesian graph product* as it is the most widely employed in multi-way algorithms. However, other products such as the tensor and strong graph products each induce novel edge topologies that warrant further exploration for MWGSP [7].

Cartesian graph products: We denote the Cartesian product of D graphs as

$$\mathcal{G} = \mathop{\square}_{\ell=1}^{D} \mathcal{G}^{(\ell)} = \mathcal{G}^{(1)} \square \cdots \square \mathcal{G}^{(D)}. \tag{7}$$

The Cartesian graph product is intuitively an XOR product since for any two vertices

$$\left\{ v = \left(v^{(1)}, \ldots, v^{(D)}\right), u = \left(u^{(1)}, \ldots, u^{(D)}\right) \right\} \subset \mathcal{V}, \tag{8}$$

the edge (v, u) exists if and only if there exists a single i such that $\left(v^{(i)}, u^{(i)}\right) \in \mathcal{E}^{(i)}$ and $v^{(\ell)} = u^{(\ell)}$ for all $\ell \neq i$. In other words, the vertices of $\mathcal{G}$ are connected if and only if exclusively one pair of factor vertices are adjacent and the remaining factor vertices are the same. Figure 2a illustrates the generation of an $n_1 \times n_2$ 2D grid graph via the product of two path graphs on n_1 and n_2 vertices.

The Cartesian graph product can induce topological properties such as regularity onto a graph. Since the path graph basis is well-characterized as a discrete Fourier basis, it is a convenient tool for including Euclidean domains in multi-way analysis. For example, we can model time series and longitudinal graph signals as a single vector using a path graph product. In the case of our gene expression data X, the product of the gene and patient mode graphs with a path on n_3 vertices, i.e., $\mathcal{G}^{(1)} \square \mathcal{G}^{(2)} \square \mathcal{P}^{n_3}$, models the data by treating the temporal mode as a sequence. One can intuit this operation as copying $\mathcal{G}^{(1)} \square \mathcal{G}^{(2)} n_3$ times and connecting edges between each copy.

Product graph matrices: The Kronecker product links graph shift operators on Cartesian product graphs to the corresponding operators on the factors. The *Kronecker sum* of D matrices $\mathrm{A}^{(k)} \in \mathbb{C}^{n_k \times n_k}$ for $k = 1, \ldots, D$ is

[3]We can do this because the vectorization $\mathrm{vec}(X)$ is isomorphic to X, which can be shown using (6).

$$\mathrm{A} = \bigoplus_{k=1}^{D} \mathrm{A}^{(k)} = \sum_{k=1}^{D} \mathrm{I}_{n_{>k}} \otimes \mathrm{A}^{(k)} \otimes \mathrm{I}_{n_{<k}}, \text{ where } n_{>k} = \prod_{\ell=k+1}^{D} n_{\ell}, \text{ and } n_{<k} = \prod_{\ell=1}^{k-1} n_{\ell}.$$

The joint adjacency matrix A and graph Laplacian $\mathscr{L}$ are constructed by the Kronecker sum of their corresponding factor graph matrices. The eigensystem of a Kronecker sum is generated by the pairwise sum of the eigenvalues of its factors and the tensor product of the factor eigenbases [22, Thm. 4.4.5]. Thus, the Fourier basis Ψ for the product graph $\mathscr{G}$ is immediate from the factors. For $k = 1, \ldots, D$ let $\left(\lambda_{\ell_k}, \psi_{\ell_k}\right)$ be the ℓ_kth eigenpair of $\mathscr{L}^{(k)}$ for $0 \le \ell_k \le n_k - 1$ Then let $I_{\ell} = (\ell_1, \ldots, \ell_D) \in [n_1] \times \ldots \times [n_D]$ be a multi-index to the ℓth eigenpair of $\mathscr{L}$. The product graph Fourier basis is then

$$\left(\lambda_{I_{\ell}}, \psi_{I_{\ell}}\right) = \left(\sum_{k=1}^{D} \lambda_{\ell_k}^{(k)}, \downarrow \bigotimes_{k=1}^{D} \psi_{\ell_k}^{(k)}\right). \tag{9}$$

Thus, the MWGFT of a multiway graph signal X is

$$\hat{\mathrm{x}} = \Psi^{*}\mathrm{vec}(X) = \left(\downarrow \bigotimes_{k=1}^{D} \Psi^{(k)}\right)^{*} \mathrm{vec}(X). \tag{10}$$

This formulation includes applying a single-way transform along one mode of the tensor, for example, $\mathrm{DFT}_{n_1}\{X\} = \left(\downarrow \otimes_{k=2}^{D} \mathrm{I}_{n_k} \otimes \mathrm{U}_{n_1}\right)^{T} \mathrm{vec}(X)$ applies the DFT along the first mode of the tensor.

Efficient MWGSP by graph factorization: On the surface, the computational cost of a MWGFT (and MWGSP in general) seems high as multi-way product graphs are often much larger than their individual factors; the cardinality of the product vertex set is the product of the number of vertices in each factor. However, the product graph structure actually yields efficient algorithms. With small adjustments to fundamental operations like matrix multiplication, in the best case one can effectively reduce the computational burden of an order D tensor with $n = \prod_{\ell=1}^{D} n_{\ell}$ total elements to a sequence of problems on $n^{(1/D)}$ elements. The computational strategy is to apply Equation (6) and its order-D generalization to avoid storing and computing large product graph operators.

We introduce the order-D form of Equation (6) via an algorithm. Given a sequence of operators $\mathrm{M}^{(\ell)} \in \mathbb{R}^{n_{\ell}}$, $\ell = 1, \ldots, D$, an efficient algorithm for computing $\mathrm{y} = \otimes_{\ell=1}^{D} \mathrm{M}^{(\ell)} \mathrm{vec}(X)$ proceeds by applying each $\mathrm{M}^{(\ell)}$ to the corresponding mode-wise matricization of X. Algorithm 1 presents pseudocode for computing this product.

Algorithm 1 D-tensor multilinear transformations

```
1: Initialize $\mathscr{Y} = X$
2: for $\ell = 1, \dots, D$ do
3:     Matricize: $\mathrm{Y}^{(\ell)} = \mathrm{mat}(\mathscr{Y}, \ell)$
4:     Factor update: $\mathrm{Y}^{(\ell)} = \mathrm{M}^{(\ell)^\top} \mathrm{Y}^{(\ell)}$
5:     Reform tensor: $\mathscr{Y} = \mathrm{ten}\left(\mathrm{Y}^{(\ell)}, \ell, \{n_1, \dots, n_D\}\right)$
6: end for
7: vectorization: $\mathrm{y} = \mathrm{vec}(\mathscr{Y})$.
```

As a sequential product of an $n_\ell \times n_\ell$ matrix with an $n \times n \mid \ell$ matrix, this method can dramatically improve the cost of algorithms that depends on matrix multiplication. Further, the number of operations only depends on computations over smaller factor matrices, enabling one to perform computations on the product graph without computing and storing expensive operators.

For example, consider the computational cost of applying an MWGFT for a product graph $\mathscr{G}$ on $n = \prod_{\ell=1}^{D} n_\ell$ nodes. In the worst case, Algorithm 1 is as fast as directly computing (10). However, in the best-case scenario $n_\ell = \sqrt[D]{n}$ for all $\ell = 1, \dots, D$, and computing D graph Fourier bases of $\sqrt[D]{n} \times \sqrt[D]{n}$ requires $\mathcal{O}\left(n^{3/D}\right)$ operations. To compute a MWGFT using the factor bases, we use $\Psi^{(\ell)}$ as the sequence of operators in Alg. 1, which costs $\mathcal{O}\left(Dn^{1/D+1}\right)$ operations. This improves upon the standard GFT, which costs $\mathcal{O}\left(n^3\right)$ operations to obtain an eigenbasis and $\mathcal{O}\left(n^2\right)$ operations to apply. For example, when $D = 3$ and $n_1 = n_2 = n_3 = \sqrt[3]{n}$, we obtain an asymptotically linear factorization of a graph Fourier basis for $\mathscr{G}$, and the corresponding MWGFT can be applied in $\mathcal{O}\left(3n^{1/3+1}\right)$ operations.

Edge density: The graph edge density impacts the scalability of signal processing algorithms for multi-way data. Matrix equations can be efficiently solved by iteratively computing sparse matrix-vector products. The computational complexity of such algorithms, which include fundamental techniques such as Krylov subspace methods and polynomial approximation, typically depend linearly on the number of edges in the graphs, e.g., [2, 25, 5]. This dependency suggests using the sparsest possible graph that still captures the main similarity structure along each mode. Indeed, a common strategy is to construct sparse Laplacian matrices [25] or edge-incidence matrices [5], using k-nearest-neighbor graphs which produce edge sets whose cardinality is linear in the number of nodes. Yet, given sparse factors graph, there is no guarantee that the product will be sparse. Thus, major efficiency gains for multi-way algorithms can be made by replacing iterative matrix-vector multiplications (both sparse and dense) with a sequence of factor graph sparse matrix-vector multiplications using Algorithm 1.

Three immediate applications for such a factorization are multi-way filter approximations [see, e.g. 2], compressive spectral clustering [26], and fast graph Fourier transforms [27]. We detail the former, while briefly describing future directions for the latter. For filtering, one could spectrally define and exactly compute a multi-way product graph filter (see box on Multi-way Filters) using the MWGSP techniques described in the previous section. Yet, Chebyshev approximations [2] are an efficient, robust, and accurate technique for approximate filtering. These approaches approximate spectrally defined filters by applying a recurrently defined weighted matrix-vector multiplication. Efficient multi-way Chebyshev approximation leverages the Kronecker sum definition for product graph Laplacians $\mathscr{L}$. That is, by noting that $\mathscr{L}\mathrm{x} = \sum_{k=1}^{D}\left(\mathrm{I}_{n>k} \otimes \mathscr{L}^{(k)} \otimes \mathrm{I}_{n<k}\right)\mathrm{x}$ is equivalent to computing

$$\left(\mathrm{I}_{n>1} \otimes \mathscr{L}^{(1)}\right)\mathrm{x} + \left(\mathrm{I}_{n>2} \otimes \mathscr{L}^{(2)} \otimes \mathrm{I}_{n_1}\right)\mathrm{x} + \ldots + \left(\mathscr{L}^{(D)}\mathrm{I}_{n<D}\right)\mathrm{x},$$

it is clear that Chebyshev approximations of functions on $\mathscr{L}$ (such as spectral graph wavelets) can be written as a sum of sparse matrix vector multiplications; the total operations are now dominated by the densest factor graph.

The efficiency of this approach cannot be understated, as it facilitates many algorithms, including the compressive spectral algorithm [26]. Indeed, it is increasingly common to estimate geometric and spectral qualities of the graph Laplacian by applying ideal filter approximations for eigencounting and coherence estimation. Finally, factor graph sparsity and Algorithm 1 could be combined with recently proposed approaches for approximate orthogonal decompositions [27] to construct a *fast product graph Fourier transform*. This algorithm would admit striking similarities to the classical fast Fourier transform.

IV. MWGSP FRAMEWORKS

Here we highlight two recent multi-way frameworks: time-vertex framework [2] and Generalized GSP [28].

A. Time-vertex framework

The *joint time-vertex* (T-V) framework [2, 3, 23] arose to address the limitations of GSP in analyzing dynamic data on graphs. This required generalizing harmonic analysis to a coupled time-graph setting by connecting a regular axis (time) to an arbitrary graph. The central application of these techniques are to analyze graph signals that are time-varying, for example, a time-series that reside on a sensor graph. Each time point of this series is itself a graph signal, while each vertex on the graph maps to a time-series of T samples. This enables learning covariate structures from T-V signals, which are bivariate functions on the vertex and time domain. Such sequences of graph signals are commonly collected longitudinally through sensor networks, video, health data, and social networks.

The Joint time-vertex Fourier Transform (JFT) [2] for a T-V signal $\mathrm{X} \in \mathbb{R}^{|V| \times T}$ is defined as

$$\mathrm{JFT}(\mathrm{X}) = \Psi^{*}\mathrm{X}\mathrm{U}_{T} \quad \text{or} \quad \mathrm{JFT}\{\mathrm{vec}(\mathrm{X})\} = \left(\overline{\mathrm{U}}_{T} \otimes \Psi\right)^{*}\mathrm{vec}(\mathrm{X}),$$

such that the multi-way Fourier transform of a T-V signal is a tensor product of the DFT basis with a GFT basis (see Fig. 3). Consequently, the JFT admits a fast transform in which one first performs an FFT along the time mode of the data before taking the GFT of the result, thus requiring only one Laplacian diagonalizaiton.

Including the DFT basis in this framework immediately admits novel joint time-vertex structures that are based on classical tools, such as variational norms that combine classical variation with graph variation [2] introduce. For efficient filter analysis, they also propose an FFT and Chebyshev based algorithm for computing fast T-V filters, which applies to both separable and non-separable filters; see an example of T-V filtering in Fig. 2. Finally, overcomplete dictionary representations are constructed as a tensor-like composition of graph spectral dictionaries with classical short-time Fourier transform (STFT) and wavelet frames. These joint dictionaries can be constructed to form frames, enabling the analysis and manipulation of data in terms of time-frequency-vertex-frequency localized atoms. T-V spectral filtering was also introduced in [3], as well as a T-V Kalman filter, with both batch and online function estimators. Further works have integrated ideas from classical signal processing such as stationarity to graph and T-V signals [29, 30, 23]. Thus, recent developments in the T-V framework can serve as a road-map for the future development of general MWGSP methods.

B. Generalized Graph Signal Processing

Another recent development is that of the Generalized Graph Signal Processing [28] framework which extends the notions of MWGSP to arbitrary, non-graphical geometries. Generalized GSP facilitates multivariate signal processing of interesting signals in which at least one domain lacks a discrete geometry. This framework recognizes that the key intuition of graph signal processing is the utility of irregular, non-Euclidean geometries for analyzing signals. However, where GSP techniques axiomatize a finite relational structure encoded by a graph shift operator, Generalized GSP extends classical Fourier analogies to arbitrary Hilbert spaces (i.e., complete inner product spaces) $\mathscr{H} \in H$ equipped with a compact, self-adjoint operator A. This broad class of geometries contains GSP, as the standard space of square summable graph signals, i.e., $L^2(V) = \{f : V \mapsto \mathbb{C}, \|f\|_2 < \infty\}$ is itself a Hilbert space.

The geometries and corresponding signals that can be induced by Generalized GSP offer an intriguing juxtaposition of continuous and discrete topologies. As an example, consider the tensor product of a graph $\mathscr{G}$ with the space of square integrable functions on an interval, e.g. $\mathscr{G} \otimes \mathscr{L}^2([-1, 1])$. Graph signals in this space map each vertex to a L^2 function. Conversely, L^2 functions can be mapped to specific vertices. To generate a Fourier basis for the product space, one simply takes the tensor product of the factor space eigenbases. This is a promising future direction for MWGSP, as it implies that one can, for instance, combine graph Fourier bases with generalized Fourier bases for innovative signal representations.

[11] proposed an early example of Generalized GSP, though under a different name. This work modeled videos and collections of related matrices as matrix-valued graph signals using *matrix convolutional networks*. The authors aimed to solve the challenging missing data problem of node undersampling: some matrix slices from the networks are completely unobserved. When matrices have a low-rank graph Fourier transform, the network's graph structure enables recovery of missing slices. In light of the development of Generalized GSP,

it is clear that [11] proposed an algorithm for denoising of multi-way signals on $\mathcal{G} \otimes \mathbb{R}^{n_1 \times n_2}$.

V. Signal processing on multi-way graphs

In the previous section, we focused on signal processing through the lens of harmonic analysis, using the graph Laplacian to analyze data in the spectral domain. In this section, we focus on signal modeling and recovery in the multi-way setting through the lens of optimization, where the graph Laplacian serves the role of imposing signal smoothness. Including graph structures along the modes of multi-way matrices and higher-order tensors has led to more robust and efficient approaches for denoising, matrix completion and inpainting, collaborative filtering, recommendation systems, biclustering, factorization, and dictionary learning [4, 16, 18, 11, 10, 21]. We begin with dual-graph modeling in the matrix setting and then extend to the higher-order tensor setting. In the tensor setting we review both using multi-way graph regularization in tensor factorization methods and in complementary fashion using tensor factorization in signal modeling and recovering to make graph regularization computationally tractable.

A. Signal processing on dual graphs

The quadratic form of the graph Laplacian of a graph $\mathcal{G}$ quantifies the *smoothness* of a signal f with respect to the graph, where the smoother a signal is the smaller the value:

$$\mathrm{f}^\top \mathscr{L} \mathrm{f} = \sum_{(i,j) \in \mathscr{E}} \mathbf{W}_{i,j}(\mathrm{f}_i - \mathrm{f}_j)^2. \quad (11)$$

Consequently, the typical model in th multi-way signal recovery setting is to add dual row-column graph regularizers of the form $\gamma_r \mathrm{Tr}(\mathrm{X}^\top \mathscr{L}_r \mathrm{X}) + \gamma_c \mathrm{Tr}(\mathrm{X} \mathscr{L}_c \mathrm{X}^\top)$ to classical problem formulations; such regularization incentivizes the recovered signal to be smooth with respect to the underlying data graphs (11). The matrices $\mathscr{L}_r$ and $\mathscr{L}_c$ denote the graph Laplacians on the rows and columns of X respectively, and the nonnegative tuning parameters γ_r and γ_c trade off data fit with smoothness with respect to the row and column geometries encoded in $\mathscr{L}_r$ and $\mathscr{L}_c$ respectively.

Table I presents formulations of these different algorithms; multiple extensions and other methods exist in the literature. For the time-vertex framework [2], the graph on the columns is a temporal graph modeled explicitly with a ring graph Laplacian $\mathscr{L}_T$. The mapping $\mathscr{P}_\Omega$ is a projection operator on the set of observed entries Θ in missing data scenarios. Methods may differ in their fidelity term minimizing the Frobenius norm for denoising or 1-norm to impart robustness to outliers [25], and several methods assume a low-rank structure, either with a nuclear norm penalty [31] or with an explicit low-rank factorization of the data matrix Y as DX, sometimes with additional constraints on the factor matrices (non-negativity [33], sparsity [16]. A few methods aim to solve a matrix completion problem (see Fig. 4). Finally, while most instances of graph regularization rely on the quadratic penalty term $\mathrm{Tr}(\mathrm{X}^\top \mathscr{L}_r \mathrm{X}) = \sum_{(i,j) \in \mathscr{E}_r} w_{i,j} \|\mathrm{X}_{i\cdot} - \mathrm{X}_{j\cdot}\|_2^2$, the biclustering formulation in [5, 32] employs a penalty that is either linear in the l_2-norm or concave and continuously differentiable relying on the mapping $\Omega(\|X_{i\cdot} - X_{j\cdot}\|_2)$. The motivation there is that convex penalties, either when Ω is linear or quadratic, do not introduce enough smoothing for small differences and too much smoothing for large differences, resulting in poorer clustering results.

Typically an alternating optimization algorithm is used to solve the various problems in Table I. The T-V regularization problem is the only one with a closed form solution given by a joint non-separable low-pass filter (generalizing Tikhonov regularization to the T-V case). The graph Dual regularized Non-negative Matrix Factorization (DNMF) [33] relies on an alternating optimization scheme for the non-negative factor matrices. Other solutions are computed with proximal methods such as Alternating-Direction Method of Multipliers (ADMM) to handle multiple regularization terms via variable splitting. Dual-graph regularized approaches have been shown to consistently out-perform their non-regularized or single-graph regularized counterparts across a wide range of applications and domains. In Fig. 4(a) we compare several approaches for matrix completion [31, 34, 25] with single way or multi-way graph regularization on the ORL dataset with 10% or 50% entries missing at random. The ORL [35] dataset consists of 300 images of faces (30 people with 10 images per person), which are flattened into 2576 features. We used a row graph that connects similar images together and a column graph that ignores the natural 2D grid geometry and instead considers a wider geometry in the image plane. To set γ_r, γ_c, we ran each method for a range of values and selected the result with best performance. For comparison to single-way graph regularization, we also set $\gamma_c = 0$ in MCG [31] and FRPCAG [34] to ignore the graph on the feature (column) space. In general, γ_r and γ_c induce row and column smoothness at different levels and their choice should be driven by the trade-off in the smoothness of the data along the two modes and the aspect ratio of the matrix, or informed by cross-validation.

We report the relative reconstruction error on the missing values, averaged over 10 realizations. The multi-way graph regularized approaches out-performed their corresponding single-way versions ($\gamma_c = 0$) in all cases. Both FRPCAG and MCG always out-performed RPCAG, a single-way graph regularised method.

B. Tensor processing on graphs

A challenge of many well-studied problems in signal processing and machine learning is that algorithm complexity typically grows exponentially when one considers tensors with three or more modes. Early multi-way data analysis approaches flattened data tensors to matrices and then applied classical two-way analysis techniques. Flattening, however, obscures higher-order patterns and interactions between the different modes of the data. Thus, multilinear tensor decompositions have been the main workhorse in tensor signal processing and data analysis, generalizing the notion of matrix factorizations to higher-order tensors, and have become common in applications such as hyperspectral and biomedical imaging.

While there is no single generalization of a spectral decomposition for tensors, the two most common tensor decompositions are the CANDECOMP/PARAFAC (CP) decomposition (see

Fig. 1) and the Tucker decomposition [8]. Just as the singular value decomposition can be used to construct a lower-dimensional approximation to a data matrix, finding a coupled pair of lower dimensional subspaces for the rows and columns, these two decompositions can be used to construct lower dimensional approximations to a D-way tensor $\mathcal{X} \in \mathbb{R}^{n_1 \times n_2 \times \cdots \times n_D}$. Under mild conditions, the CP decomposition, which approximates $\mathcal{X}$ by a sum of rank-one tensors, is unique up to scaling and permutations of the columns of its factor matrices [8], but the CP factor matrices typically cannot be guaranteed to have orthogonal columns. The Tucker decomposition permits orthonormal factor matrices but in general fail to have unique representations [8]. Much of the multi-way literature has focused on improving and developing new tensor factorizations. Graph-based regularizations along modes of the tensor are proving versatile for developing robust tensor and low-rank decompositions [12–14], as well as new approaches to problems in higher order data processing such as tensor completion, data imputation, recommendation system, feature selection, anomaly detection, and co-clustering [17, 11, 18–20, 10]—a generalization of biclustering to tensors. Generalization of these problems to tensors incurs a higher computational cost than the equivalent matrix problems. Thus multi-way graph-regularized formulations typically combine a low-rank tensor factorization with graph-based regularization along the rows of the factor matrices; for example [20, 18] rely on a CP decomposition while [13] relies on a Tucker decomposition. In [38], a Tucker decomposition is used within MWGSP, to construct wavelets on multislice graphs in a two-stage approach.

An example of combining tensor decompositions with graph regularization is the following “low-rank + sparse” model for anomaly detection in internet traffic data [20]:

$$\begin{aligned} \min_{\mathcal{X}, \varepsilon \left\{\mathrm{A}^{(i)}\right\}_i} \|(\mathcal{Y} - \mathcal{E}) - \mathcal{X}\|_{\mathrm{F}}^2 + \sum_{i=1}^{d} \gamma_i \mathrm{Tr}\left(\left(\mathrm{A}^{(i)}\right)^{\top} \mathcal{L}_i \mathrm{A}^{(i)}\right) \quad \text{s.t.} \quad \mathcal{X} \\ = \sum_{i=1}^{R} \mathrm{a}_i^{(1)} \circ \mathrm{a}_i^{(2)} \circ \mathrm{a}_i^{(3)}, \quad \|\mathcal{E}\|_0 \le \epsilon, \end{aligned} \tag{12}$$

where $\mathcal{Y}$ is a data tensor and $\mathcal{E}$ is the tensor of sparse outliers. The equality constraint on $\mathcal{X}$ requires that $\mathcal{X}$ has a rank-R CP decomposition where $\mathrm{a}_i^{(d)}$ is the ith column of the dth factor matrix $\mathrm{A}^{(d)} \in \mathbb{R}^{n_d \times R}$ and $\circ$ denotes an outer product. Note that the graph regularization terms in (12) are applied to the factor matrices $\mathrm{A}^{(i)} \in \mathbb{R}^{n_i \times R}$, reducing the computational complexity of the estimation algorithm. Decomposing a data tensor into the sum of a low-rank and sparse components is also used in [12, 19, 13].

In [14], computational complexity is further reduced by pre-calculating $\mathrm{P}_R^{(i)}$ mode-specific graph Laplacian eigenvectors of rank R from the matricization of the tensor along each mode and using these in solving tensor-robust PCA. The solution relies on projecting the tensor onto a tensor product of the graph basis $\left\{\mathrm{P}_R^{(i)}\right\}$, resulting in a formulation to similar to the Tucker decomposition.

Co-clustering assumes the observed tensor is the sum of a “checkerbox” tensor (under suitable permutations along the modes) and additive noise. For example, Chi et al. [10] propose estimating a “checkerbox” tensor with the minimizer to a convex criterion. In the case of 3-way tensor, the criterion is

$$\frac{1}{2}\|Y - X\|_{\mathrm{F}}^2 + \gamma\left[\sum_{(i,j)\in\mathscr{E}^{(1)}} w_{ij}^{(1)}\|X_{i::} - X_{j::}\|_{\mathrm{F}} + \sum_{(i,j)\in\mathscr{E}^{(2)}} w_{ij}^{(2)}\|X_{:i:} - X_{:j:}\|_{\mathrm{F}} + \sum_{(i,j)\in\mathscr{E}^{(3)}} w_{ij}^{(3)}\|X_{::i} - X_{::j}\|_{\mathrm{F}}\right],$$

where $\mathscr{E}^{(d)}$ is a set of edges for the mode-d graph, γ is a nonnegative tuning parameter, and $w_{ij}^{(d)}$ is a weight encoding the similarity between the ith and jth mode-d slices. Minimizing the criterion in (13) can be interpreted as denoising all modes of the tensor simultaneously via vector-valued graph total-variation.

C. Manifold learning on multi-way data

Tensor factorization can fail to recover meaningful latent variables when nonlinear relationships exist among slices along each of modes. Manifold learning overcomes such limitations by estimating *nonlinear* mappings from high-dimensional data to low-dimensional representations (embeddings). While GSP uses the eigenvectors of the graph Laplacian as a basis in which to linearly expand graph signals (1), manifold learning uses the eigenvectors ψ_ℓ themselves as a nonlinear d-dimensional map Ψ for the datapoints $\{\mathrm{x}_i\}_i$ as $\Psi: \mathrm{x}_i \to (\psi_1(i), \ldots, \psi_d(i))$.

A naïve strategy to apply manifold learning to the multi-way data is to take the D different matricizations of a D-way tensor and construct a graph Laplacian using a generic metric on each of the D modes independently, thereby ignoring the higher-order coupled structure in the tensor. Recent work [6, 9, 32], however, incorporate higher-order tensor structure in manifold learning by thoughtfully designing the similarity measures used to construct the mode k graph weights $\mathbf{W}^{(k)}$. The co-manifold learning framework can be viewed as blending GSP and manifold learning together and has most recently extended to tensors and the missing data setting [9, 32].

From a MWGSP perspective, the key contribution of this line of work is a new metric that is defined between tensor slices as the difference between a graph-based multiscale decomposition of each slice along its remaining modes; for example the distance between two horizontal slices in a 3-way tensor is

$$\begin{aligned} d\left(\mathscr{X}_{i..}, \mathscr{X}_{j..}\right) &= \left\|\left(\mathrm{M}^{(3)} \otimes \mathrm{M}^{(2)}\right)\mathrm{vec}\left(\mathscr{X}_{i::} - \mathscr{X}_{j::}\right)\right\|_1 \\ &= \left\|\mathrm{vec}\left(\mathrm{M}^{(2)}\left(\mathscr{X}_{i::} - \mathscr{X}_{j::}\right)\left(\mathrm{M}^{(3)}\right)^{\top}\right)\right\|_1, \end{aligned} \quad (13)$$

where $M^{(k)}$ is a multiscale transform in the kth mode. This metric was shown to be a tree-based Earth-mover's distance in the 2D setting [39]. The resulting similarity depends on a multi-way multiscale difference between slices, and has been successfully used in practice to construct weighted graphs in multiway data. The multiscale decompositions are constructed either from data-adaptive tree transforms [6] or through a series of multi-way graph-based co-clustering solutions [32].

VI. Future Outlook

As multi-way signal processing frameworks continue to mature, several challenges remain ahead. While novel techniques are continually introduced into single-way graph signal processing, one approach to developing multi-way techniques is to identify, extend, and adapt techniques which are particularly useful for multi-way signals. For instance, multi-way analysis on directed graphs will greatly broaden the versatility of MWGSP. From a computational perspective, it is clear that the efficiency gains offered by the march of single-way GSP march towards fast transforms [27] are compounded in the multi-way setting.

From a theoretical perspective, open questions include 1) What additional advantages can be gained by treating classical domains as lying on graphs? 2) How do we learn mode-specific or coupled graphs from data, in general and in dynamical settings? 3) Are such tensor datasets typically low-rank or high-rank? 4) How do we process data whose generative model is nonlinear across the different modes?

From a practical perspective, ongoing growth in computational power and parallel computing have enabled large-scale analyses. The MWGSP framework can leverage these recent advances in computational building blocks. Nonetheless, there are existing computational challenges, such as applications requiring online real-time processing. Thus, future directions include developing online and distributed versions of multi-way graph signal processing, especially in the presence of large-scale data, where streaming solutions are necessary (the data does not fit in memory). In addition, there is need for new optimization techniques to efficiently solve problems that combine tensors with graph-based penalties. Deep learning is also emerging as a framework to learn rather than design wavelet-type filterbanks in signal processing and these approaches can be extended to the graph and multi-way settings to learn joint multiscale decompositions. Finally, as the GSP community continues to address real-world data domains such as climate, traffic, and biomedical research, inter-disciplinary collaboration is essential to define relevant problems and demonstrate significant utility of these approaches within a domain.

Acknowledgments

This research was supported by the National Institutes of Health grants R01RGM131642, P50CA121974, R01HG008383, R01GM135928, R01EB026936, and the National Science Foundation grant DMS-1752692.

References

[1]. Ortega A, Frossard P, Kovačević J, Moura JM, and Vandergheynst P, "Graph signal processing: Overview, challenges, and applications," Proceedings of the IEEE, vol. 106, no. 5, pp. 808–828, 2018.

[2]. Grassi F, Loukas A, Perraudin N, and Ricaud B, “A time-vertex signal processing framework: Scalable processing and meaningful representations for time-series on graphs,” IEEE Trans. Signal Process., vol. 66, no. 3, pp. 817–829, 2017.

[3]. Romero D, Ioannidis VN, and Giannakis GB, “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.

[4]. Rao N, Yu H-F, Ravikumar PK, and Dhillon IS, “Collaborative filtering with graph information: Consistency and scalable methods,” in Adv. Neural Inf. Process. Syst, 2015, pp. 2107–2115.

[5]. Chi EC, Allen GI, and Baraniuk RG, “Convex Biclustering,” Biometrics, vol. 73, no. 1, pp. 10–19, 2017. [PubMed: 27163413]

[6]. Mishne G, Talmon R, Cohen I, Coifman RR, and Kluger Y, “Data-driven tree transforms and metrics,” IEEE Trans. Signal Inf. Process. Netw, vol. 4, pp. 451–466, 2018. [PubMed: 30116772]

[7]. Sandryhaila A and Moura JMF, “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.

[8]. Kolda TG and Bader BW, “Tensor decompositions and applications,” SIAM Review, vol. 51, no. 3, pp. 455–500, 2009.

[9]. Mishne G, Talmon R, Meir R, Schiller J, Lavzin M, Dubin U, and Coifman RR, “Hierarchical coupled-geometry analysis for neuronal structure and activity pattern discovery,” IEEE J. Sel. Topics Signal Process., vol. 10, no. 7, pp. 1238–1253, 2016.

[10]. Chi EC, Gaines BR, Sun WW, Zhou H, and Yang J, “Provable convex co-clustering of tensors,” arXiv:1803.06518 [stat.ME], 2018.

[11]. Sun Q, Yan M, Donoho D, and Boyd S, “Convolutional imputation of matrix networks,” in ICML, 2018, pp. 4818–4827.

[12]. Nie Y, Chen L, Zhu H, Du S, Yue T, and Cao X, “Graph-regularized tensor robust principal component analysis for hyperspectral image denoising,” Applied optics, vol. 56, no. 22, pp. 6094–6102, 2017. [PubMed: 29047801]

[13]. Zhang K, Wang M, Yang S, and Jiao L, “Spatial–spectral-graph-regularized low-rank tensor decomposition for multispectral and hyperspectral image fusion,” IEEE J. Sel. Topics Appl. Earth Observ. Remote Sens, vol. 11, no. 4, pp. 1030–1040, 2018.

[14]. Shahid N, Grassi F, and Vandergheynst P, “Tensor robust PCA on graphs,” in IEEE Internat. Conf. Acoust. Speech and Signal Process. (ICASSP), 2019, pp. 5406–5410.

[15]. Cichocki A, Mandic D, De Lathauwer L, Zhou G, Zhao Q, Caiafa C, and Phan HA, “Tensor decompositions for signal processing applications: From two-way to multiway component analysis,” IEEE Signal Process. Mag, vol. 32, no. 2, pp. 145–163, 3 2015.

[16]. Yankelevsky Y and Elad M, “Dual graph regularized dictionary learning,” IEEE Trans. Signal Inf. Process. Netw, vol. 2, no. 4, pp. 611–624, 2016.

[17]. Li C, Zhao Q, Li J, Cichocki A, and Guo L, “Multi-tensor completion with common structures,” in AAAI, 2015.

[18]. Ioannidis VN, Zamzam AS, Giannakis GB, and Sidiropoulos ND, “Coupled graph and tensor factorization for recommender systems and community detection,” IEEE Trans. Knowl. Data Eng, pp. 1–1, 2019.

[19]. Su Y, Bai X, Li W, Jing P, Zhang J, and Liu J, “Graph regularized low-rank tensor representation for feature selection,” J Vis Commun Image R, vol. 56, pp. 234–244, 2018.

[20]. Xie K, Li X, Wang X, Xie G, Wen J, and Zhang D, “Graph based tensor recovery for accurate internet anomaly detection,” in IEEE INFOCOM, 2018, pp. 1502–1510.

[21]. Rabin N and Fishelov D, “Two directional Laplacian pyramids with application to data imputation,” Adv Comput Math, pp. 1–24, 2019.

[22]. Horn RA and Johnson CR, Topics in matrix analysis. Cambridge university press, 1994.

[23]. Loukas A and Perraudin N, “Stationary time-vertex signal processing,” EURASIP J. Adv. Signal Process., vol. 2019, no. 1, p. 36, 2019. [PubMed: 31983922]

[24]. Baumgardner MF, Biehl LL, and Landgrebe DA, "220 Band AVIRIS Hyperspectral Image Data Set: June 12, 1992 Indian Pine Test Site 3," 9 2015 [Online]. Available: https://purr.purdue.edu/publications/1947/1

[25]. Shahid N, Perraudin N, Kalofolias V, Puy G, and Vandergheynst P, "Fast robust PCA on graphs," IEEE J. Sel. Topics Signal Process., vol. 10, no. 4, pp. 740–756, 2016.

[26]. Tremblay N, Puy G, Gribonval R, and Vandergheynst P, "Compressive spectral clustering," in ICML, 2016, pp. 1002–1011.

[27]. Frerix T and Bruna J, "Approximating orthogonal matrices with effective Givens factorization," in ICML, 2019, pp. 1993–2001.

[28]. Ji F and Tay WP, "A Hilbert space theory of generalized graph signal processing," IEEE Trans. Signal Process., vol. 67, no. 24, pp. 6188–6203, 2019.

[29]. Girault B, "Stationary graph signals using an isometric graph translation," in EUSIPCO, 2015, pp. 1516–1520.

[30]. Marques AG, Segarra S, Leus G, and Ribeiro A, "Stationary graph processes and spectral estimation," IEEE Trans. Signal Process., vol. 65, no. 22, pp. 5911–5926, 2017.

[31]. Kalofolias V, Bresson X, Bronstein M, and Vandergheynst P, "Matrix completion on graphs," arXiv preprint arXiv:1408.1717, 2014.

[32]. Mishne G, Chi EC, and Coifman RR, "Co-manifold learning with missing data," in ICML, 2019.

[33]. Shang F, Jiao L, and Wang F, "Graph dual regularization non-negative matrix factorization for co-clustering," Pattern Recognition, vol. 45, no. 6, pp. 2237–2250, 2012.

[34]. Shahid N, Kalofolias V, Bresson X, Bronstein M, and Vandergheynst P, "Robust principal component analysis on graphs," in Proceedings of the IEEE International Conference on Computer Vision, 2015, pp. 2812–2820.

[35]. Samaria FS and Harter AC, "Parameterisation of a stochastic model for human face identification," in Proc. IEEE workshop on applications of computer vision, 1994, pp. 138–142.

[36]. Benzi K, Kalofolias V, Bresson X, and Vandergheynst P, "Song recommendation with non-negative matrix factorization and graph total variation," in ICASSP-2016, 3 2016, pp. 2439–2443.

[37]. Mateos G, Segarra S, Marques AG, and Ribeiro A, "Connecting the dots: Identifying network structure via graph signal processing," IEEE Signal Process. Mag, vol. 36, no. 3, pp. 16–43, 2019.

[38]. Leonardi N and Van De Ville D, "Tight wavelet frames on multislice graphs," IEEE Trans. Sig. Process., vol. 61, no. 13, pp. 3357–3367, 7 2013.

[39]. Coifman RR and Leeb WE, "Earth mover's distance and equivalent metrics for spaces with hierarchical partition trees," Yale University, Tech. Rep, 2013, technical report YALEU/DCS/TR1482.

Order-3 tensors

For simplicity, we briefly review some tensor terminology for the 3-way tensor $\chi \in \mathbb{C}^{n_1 \times n_2 \times n_3}$. The size of each mode is denoted by n_k, with n_1 being the number of columns, n_2 the number of rows, and n_3 being the number of *tubes* [8]. Video and time-series recording of matrix valued signals are a common application for tensors of this form (Fig. 1). In videos, the first and second modes of the tensor encode pixel values for each frame, while the third mode indexes the frames in time.

We can *slice* a video tensor to produce different views of the data as presented in Fig. 1.

Multi-way signal compression

A key motivation for MWGSP is the capability of encoding multi-way data in a compact way. Transforms with good energy compactness summarize the data well and can be used to construct efficient regularizers for regression problems. Figure 3 demonstrates energy compression in four datasets. The Dancer mesh [2] shown in Fig. 2b couples n_1 = 1502 points to temporal evolution across n_2 = 570 timesteps. The Molene weather dataset [23] (n_1 = 32 weather stations measuring temperatures over n_2 = 24 hours across n_3 = 30 days) couples a spatially determined graph to two temporal scales (hours, days). The time lapse video [2] couples a 2D spatial grid (492 $\times$ 853 pixels) to a temporal axis (602 timesteps), while the hyperspectral dataset [24] couples a 2D spatial grid (145 $\times$ 145 pixels) to 200 spectral bandwidths (treated as a graph). All graphs were constructed using k-nearest neighbors with weighted edges set using a Gaussian kernel on the matricized modes of the tensor. To measure energy compactness, we compute the relevant among the GFT, DFT (temporal axis), JFT, 2D-DFT (spatial grid), 3D-DFT (spatail grid+temporal axis) and MWGFT (all tensor modes) transforms for each dataset. We replace the spectrum coefficients with magnitudes smaller than the p-th percentile with zeros and perform the corresponding inverse transform on the resulting coefficients. The normalized compression error is computed from the signal reconstructed after thresholding the values of the transforms below the p-th percentile, denoted X_p, and given by $\|\mathrm{vec}(X_P - X)\|_2 / \|\mathrm{vec}(X)\|_2$. MWGFT achieves the best compactness in all datasets, providing insight that there are advantages to treating classical domains (time and space) as lying on graphs themselves.

Multi-way Filters

It is natural to define spectral filters for multi-way graph signals on the product $\mathcal{G}$ as a function over the product graph eigenvalues $h: \Lambda \mapsto \mathbb{R}$ as if they are traditional spectrally defined GSP filters. Since these functions operate on the product eigenvalues, they directly consider the edge topology induced by a particular choice of product. Yet, it is feasible to develop filters for multi-way graph signals on $\mathcal{G}$ that are defined by multivariate functions $h: \Lambda^{(1)} \times \cdots \times \Lambda^{D} \mapsto \mathbb{R}$. These multivariate filters are split into two classes: separable and nonseparable.

Separable filters have multivariate response functions that can be written as the product of separate univariate functions. In the $D = 2$ case, a separable filter for the product graph could be written as $\left(\mathrm{H}^{(2)} \otimes \mathrm{H}^{(1)}\right)\mathrm{x}$ in which $\mathrm{H}^{(1)} = \Psi^{(1)} h^{(1)}\left(\Lambda^{(1)}\right)\Psi^{(1)^{*}}$ and $\mathrm{H}^{(2)} = \Psi^{(2)} h^{(2)}\left(\Lambda^{(2)}\right)\Psi^{(2)^{*}}$. Since this Kronecker product is permutation equivalent, we can treat its operation as an order-independent unimodal filtering of x (6). If $\mathrm{H}^{(1)}$ and $\mathrm{H}^{(2)}$ are both filters defined in a Laplacian eigenbasis of their respective factor graph, then the tensor product $\left(\mathrm{H}^{(2)} \otimes \mathrm{H}^{(1)}\right)$ is also diagonalized by the product eigenbasis. Thus, this filter is merely a reweighting of the product graph eigenbasis. In Figure 2e, we demonstrate the application of a product of mode-wise heat filters to a graph signal on a grid (Fig. 2d top) and to a time-vertex signal which is a dynamic mesh (Fig. 2d bottom). While there is a choice of τ_1 and τ_2 such that certain regions of this filter can be computed from a heat kernel on the Cartesian product graph spectrum, such an approach abandons the flexibility of bilinear filtering. By separability, each mode can be analyzed independently of the other by setting the appropriate τ_k to 0. This enables analyzing a joint signal along each mode independently, for example by filtering out high frequency structure along one domain while preserving the frequency content of the other mode. A D-way separable filter applied to $\mathrm{x} = \mathrm{vec}(X)$ is given by

$$\tilde{\mathrm{x}} = \Psi h\left(\mathop{\times}_{k=1}^{D} \Lambda^{(k)}\right)\Psi^{*}\mathrm{x},$$

where $h\left(\times_{k=1}^{D} \Lambda^{(k)}\right)$ is a diagonal matrix whose elements are given by $\prod_{k=1}^{D} h^{(k)}\left(\lambda_{\ell_k}\right)$, i.e., the product of separate spectral functions $h^{(k)}$ for each factor graph $\mathcal{G}^{(k)}$.

Nonseparable filters cannot be designed from separate univariate filters on each mode. This class of filters encompasses a broad class of functions that include many filters defined in terms of the product graph eigenvalues, as well as multivariate functions (Fig. 2f). Indeed, [2] find that one cannot in general describe PDEs that describe diffusion, wave, or disease propagation with separable filters, as the relation between frequencies is not independent.

Graph Construction

A question that arises in graph-based methods is how to construct the graphs themselves. In some applications, e.g., social or citation networks, the graph is known a-priori. In transportation or communication networks, vertices represent physical locations (traffic intersections) or sensors (routers in a wifi network), and edges encodes connected locations. In other settings there is no a-priori graph, and the topology must be learned from the data. We describe common strategies and challenges.

Data-driven graph:

One of the most popular ways to construct a graph is from the data itself, for example, using a k-nearest neighbor graph with Gaussian kernel weights. For example, in our simulations the row graph weights are $\mathbf{W}_{i,j}^{(1)} = \exp\left\{-\|\mathrm{X}_{i\cdot} - \mathrm{X}_{j\cdot}\|_2^2/\sigma\right\}$ if rows i and j are k-nearest neighbors and zero otherwise, σ is bandwidth parameter and we set $k = 7$. One difficulty that arises is that in the presence of noise, outliers and missing entries, constructing a graph from the data yields a corrupted graph. Fig. 4(b) compares a 'noisy' graph constructed from the missing data to an 'oracle' graph constructed from the original complete data. The noisy graph along the images ($\mathbf{A}^{(1)}$) connects images of different people together while the noisy feature graph ($\mathbf{A}^{(2)}$) loses local pixel geometry. Results in Fig. 4(a) demonstrates that for higher percent of missing values the noisy graph degrades the performance compared to the oracle graph.

Graphs from side information:

Supplementary information can be leveraged to define similarity structure among rows or columns for the purpose of graph construction. In some cases, there may be a natural geometry that easily translates into similarity graphs for rows and columns. For example, in [23] the authors constructed the graph among the weather stations using their physical coordinates. In other cases, other supplemental data sets may be leveraged to provide similarity structure among rows or columns. As an example in music recommendation systems, in [36] the authors used a publicly available playlist categorization as well as summary statistics extracted from the audio signal to construct a graph for estimating a latent association matrix between playlists and songs.

Graph learning:

In [16, 19] the graphs on the feature space are learned alongside the signal, by minimizing over $\mathscr{L}$ in addition to the signal recovery in the optimization problem. For a detailed review see [37].

Dynamically varying graphs:

Graphs may not be static, presenting a current challenge in GSP, which is especially acute in time-vertex frameworks which admit time as one of components in the analysis. Challenges include determining how to identify when a graph needs to be updated, i.e., when the underlying topology has changed. The challenge of accounting for dynamically varying graphs also poses computational questions, namely what are computationally efficient ways to update graphs within the processing framework that will minimally spawn artifacts at transitions?

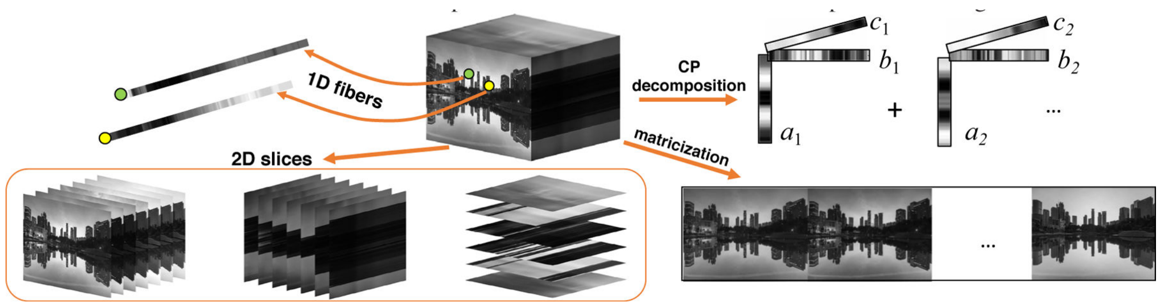

Fig. 1.

Tensor terminology. A Time-lapse video is an order-3 tensor. Tensor slices (left to right): A *frontal slice* is the matrix $X_{::k}$, formed by selecting the *k*-th frame of the video. The *lateral slice*, $X_{:j:}$, is a matrix (viewable as an image) that shows the time evolution of the *j*-th column of pixels in the input. The *horizontal slice* $X_{i::}$ similarly contains the time evolution of one row of pixels. 2D indexing of 3rd order tensors yields a *1D fiber*. For example, the tubular fiber $X_{ij:}$ is an n_3 dimensional time-series of the *i, j*th pixel across all frames; the two tubular fibers correspond to the highlighted pixels in the tensor. Mode-1 *matricization* concatenates all frontal slices side by side. *CP decomposition* is a sum of rank-1 tensors.

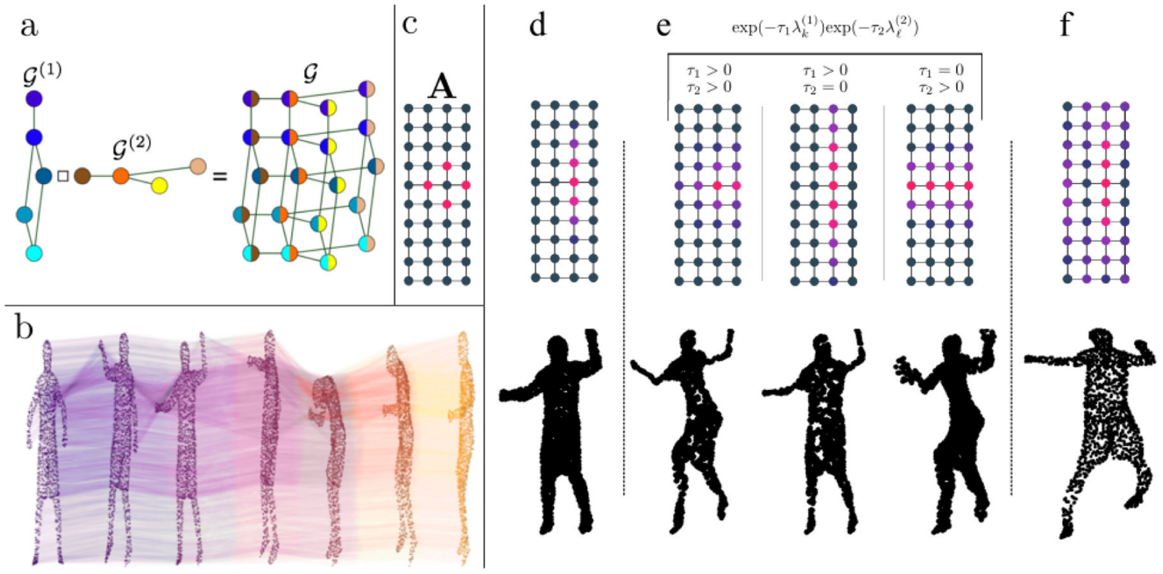
a
$\mathcal{G}^{(1)}$
$\mathcal{G}^{(2)}$
$\mathcal{G}$
b
c
A
d
e
$\exp(-\tau_1\lambda_k^{(1)})\exp(-\tau_2\lambda_\ell^{(2)})$
$\tau_1 > 0$ $\tau_2 > 0$
$\tau_1 > 0$ $\tau_2 = 0$
$\tau_1 = 0$ $\tau_2 > 0$
f

Fig. 2.

Multi-way graphs, signals and spectral filters. (a) The Cartesian graph product generates a copy of $\mathscr{G}^{(1)}$ at each vertex of $\mathscr{G}^{(2)}$. (b) A multiway graph formed from a dynamic mesh [2]. This time-vertex graph (purple = t_0, yellow = t_7) connects each point in the mesh to its counterpart in adjacent frames. The temporal evolution of the 3D coordinates is a graph signal on this graph. (c) A single column of the joint adjacency matrix of a 2D grid shifts signals to their neighbors. (d-f) Multiway filtering. (Top) A multiway signal on a 2D grid. This signal can be decomposed into an impulse and a smooth signal; thus it is bandlimited along one way of the grid. (Bottom) A frame of the Dancer mesh. (e) A separable diffusion filter is the product of domain-specific heat kernels. Separable filters can filter along both axes in unison (left), or each axis independently (middle / right). (f) Non-separable filter. For the dynamic mesh, filtering along only one mode reveals either skeleton structure ($\tau_2 = 0$) or averaged (blurred) dynamics of the figure ($\tau_1 = 0$), but joint separable or non-separable filtering reveals joint dependencies.

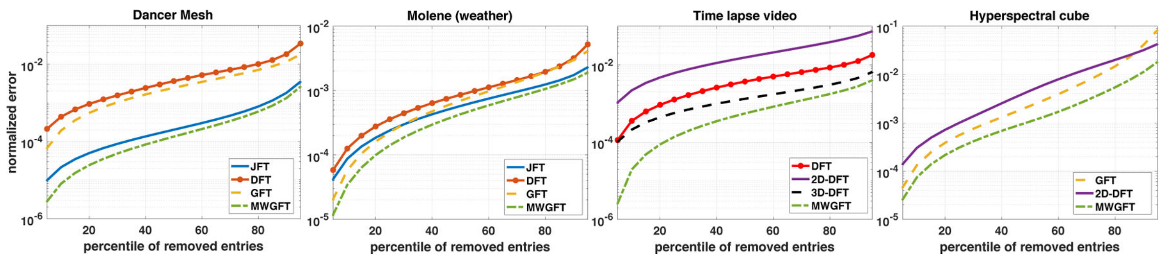

Fig. 3.
Compactness of single-way and multi-way transforms for different datasets: dancer mesh (Fig. 2), Molene weather, time-lapse video (Fig. 1) and AVIRIS Indiana Pines hyperspectral image.

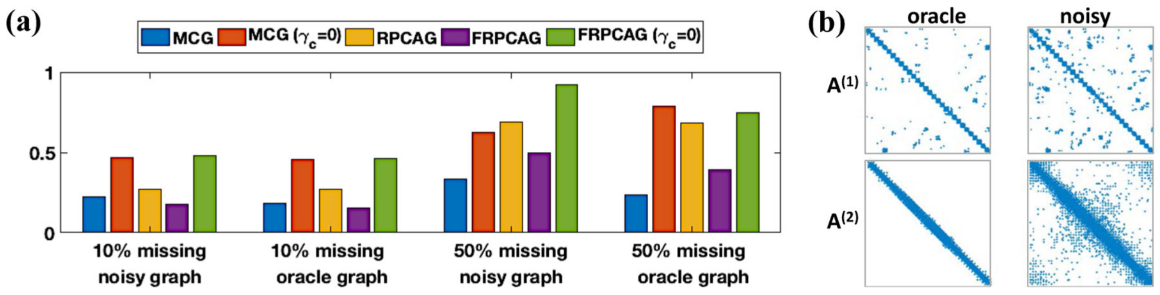

Fig. 4.

Matrix completion on the ORL dataset. (a) Relative error for 10% and 50% missing values using noisy and oracle graphs. (b) Adjacency matrix of row $\mathbf{A}_r$ and column $\mathbf{A}_c$ graphs for complete data ('oracle') and 50% missing data ('noisy').

TABLE I

Multi-way Graph regularization formulations

	Fidelity term	**Graph regularizers**	**additional constraints**
MCG [31]	$\|\mathcal{P}_\Theta(\mathrm{Y}-\mathrm{X})\|_\mathrm{F}^2$	$\gamma_r \mathrm{Tr}\left(\mathrm{X}^\top \mathcal{L}_r \mathrm{X}\right) + \gamma_c \mathrm{Tr}\left(\mathrm{X}\mathcal{L}_c \mathrm{X}^\top\right)$	$\gamma_n \|\mathrm{X}\|_*$
CFGI [4]	$\|\mathcal{P}_\Theta(\mathrm{Y}-\mathrm{DX})\|_\mathrm{F}^2$	$\gamma\left(\mathrm{Tr}\left(\mathrm{D}^\top \mathcal{L}_r \mathrm{D}\right) + \mathrm{Tr}\left(\mathrm{X}\mathcal{L}_c \mathrm{X}^\top\right)\right)$	$\alpha\|\mathrm{D}\|_\mathrm{F}^2 + \beta\|\mathrm{X}\|_\mathrm{F}^2$
DGRDL [16]	$\|\mathrm{Y}-\mathrm{DX}\|_\mathrm{F}^2$	$\gamma_r \mathrm{Tr}\left(\mathrm{D}^\top \mathcal{L}_r \mathrm{D}\right) + \gamma_c \mathrm{Tr}\left(\mathrm{X}\mathcal{L}_c \mathrm{X}^\top\right)$	$\|\mathrm{x}_i\|_0$
T-V Reg [2]	$\|\mathrm{Y}-\mathrm{X}\|_\mathrm{F}^2$	$\gamma_r \mathrm{Tr}\left(\mathrm{X}^\top \mathcal{L}_G \mathrm{X}\right) + \gamma_c \mathrm{Tr}\left(\mathrm{X}\mathcal{L}_T \mathrm{X}^\top\right)$	
T-V Inpaint [2]	$\|\mathcal{P}_\Theta(\mathrm{Y}-\mathrm{X})\|_\mathrm{F}^2$	$\gamma_r \mathrm{Tr}\left(\mathrm{X}^\top \mathcal{L}_G \mathrm{X}\right) + \gamma_c \mathrm{Tr}\left(\mathrm{X}\mathcal{L}_T \mathrm{X}^\top\right)$	
Cvx Biclust [5]	$\|\mathrm{Y}-\mathrm{X}\|_\mathrm{F}^2$	$\gamma_r \sum_{(i,j)\in\mathcal{E}_r} w_{i,j}\|\mathrm{X}_{i\cdot} - \mathrm{X}_{j\cdot}\|_2 +$ $\gamma_c \sum_{(i,j)\in\mathcal{E}_c} \widetilde{w}_{i,j}\|\mathrm{X}_{\cdot i} - \mathrm{X}_{\cdot j}\|_2$	
Comani-missing [32]	$\|\mathcal{P}_\Theta(\mathrm{Y}-\mathrm{X})\|_\mathrm{F}^2$	$\gamma_r \sum_{(i,j)\in\mathcal{E}_r} \Omega\left(\|\mathrm{X}_{i\cdot} - \mathrm{X}_{j\cdot}\|_2\right) +$ $\gamma_c \sum_{(i,j)\in\mathcal{E}_c} \Omega\left(\|\mathrm{X}_{\cdot i} - \mathrm{X}_{\cdot j}\|_2\right)\|_2$	
FRPCAG [25]	$\|\mathrm{Y}-\mathrm{X}\|_1$	$\gamma_r \mathrm{Tr}\left(\mathrm{X}^\top \mathcal{L}_r \mathrm{X}\right) + \gamma_c \mathrm{Tr}\left(\mathrm{X}\mathcal{L}_c \mathrm{X}^\top\right)$	
DNMF [33]	$\|\mathrm{Y}-\mathrm{DX}\|_\mathrm{F}^2$	$\gamma_r \mathrm{Tr}\left(\mathrm{D}^\top \mathcal{L}_r \mathrm{D}\right) + \gamma_c \mathrm{Tr}\left(\mathrm{X}\mathcal{L}_c \mathrm{X}^\top\right)$	$\mathrm{D} \geq 0,\ \mathrm{X} \geq 0$